Dissertation

Security of Machine Learning Systems

—

Thorsten Eisenhofer

A dissertation submitted to the *Faculty of Computer Science* at *Ruhr University Bochum* for the degree of

Doctor of Natural Sciences

Reviewer: Prof. Dr. Thorsten Holz
CISPA Helmholtz Center for Information Security

Prof. Dr. Nicolas Papernot
University of Toronto and Vector Institute

Prof. Dr. Konrad Rieck
Technische Universität Berlin

Defense: June 19, 2023

Abstract

Machine learning (ML) models are not programmed explicitly but are learnt from a set of data points. This data-centric paradigm allows for diverse applications, and ML models are now widely deployed in practice as internal components of *ML systems*. This inclusion of machine learning, however, introduces a new attack surface to these systems since ML models are vulnerable to a myriad of possible attacks. While prior work made remarkable progress in understanding such attacks from the perspective of the model, the deployment in practical systems introduces additional constraints, and commonly studied threat models do not sufficiently express the knowledge, capabilities, and goals of practical adversaries.

In this work, we therefore investigate the security of machine learning with a systems security approach. By viewing the ML model as part of a system, we study the increased attack surface of practical systems and how such systems can be secured. We start by investigating how a vulnerability within an ML component can be leveraged for an attack against the enclosing system. As a practical example, we consider assignment systems which are increasingly used to assist the academic reviewing process. Subsequently, we turn to the security of an ML system from a defender's point of view. By including the context of a specific deployment, we propose a principled approach to leverage domain-specific priors to improve the robustness of internal ML models. We practically evaluate our approach on speech recognition models employed as subcomponents in voice assistants. Finally, we consider the lifecycle of an ML system. Computer systems are not static and are continuously maintained and updated. For an ML system, this also affects internal ML models. As a practical example, we focus on the problem of verifiable machine unlearning that requires capturing consistency across model updates and evolving datasets.

Acknowledgements

First and foremost, I want to thank my advisor Thorsten Holz for all the support in the past years. Thank you for providing such a great work environment, leaving us the freedom to pursue our own research ideas, and always support us with our goals.

I would also like to thank Nicolas Papernot for hosting me in the cleverhans lab and all the support along the way. Special thanks also to Varun Chandrasekaran for taking such good care of me during my internship and afterwards.

Many thanks to Konrad Rieck for all the support with our project and for letting me join your group at the TU Berlin as a postdoctoral researcher. I am looking forward to many interesting discussions in the future!

To my colleagues and friends I found along the way. To Lea Schönherr: for great collaborations on our academic and sometimes not so academic projects. To Lars Speckemeier: for many hours of discussion and all our late-night sessions. To Nils Albartus, Nils Bars, Lukas Bernhard, Merlin Chlosta, Joel Frank, Tobias Scharnowski, Nico Schiller, and Moritz Schlögel: research is full of ups and downs and it is great to have you to always rely on! To Avital Shafran and Yannis Cattan: thanks for making our summer in Toronto unforgettable.

To my family and especially my sister Heike Eisenhofer for always having my back and to Doreen Riepel for supporting me throughout the doctorate and beyond.

Finally, to all those who have been a part of my getting there: Hojjat Aghakhani, Tim Blazytko, Jonathan Evertz, Maximilian Golla, Philipp Görz, Dorothea Kolossa, Christopher Kruegel, Jana Kolot, Jonas Möller, Erwin Quiring, Roei Schuster, Giovanni Vigna, and many more. Thank you all!

Contents

I Introduction **1**

1 Introduction **3**

2 Background **9**

2.1 Machine Learning 9

2.2 Adversarial Machine Learning 13

3 Security of ML Systems **17**

3.1 Feature-Problem-Space Attacks 20

3.2 Domain-Specific Priors 25

3.3 Security Beyond the Model 30

4 Conclusions **35**

References **37**

II Publications **49**

List of Publications **51**

A Subverting Automatic Paper-Reviewer Assignment **53**

B Taming Audio Adversarial Examples **103**

C Verifiable and Provably Secure Machine Unlearning **147**

Part I

—

Introduction

Introduction

Artificial intelligence is on the rise and affects many aspects of our lives. Its disruptive potential is compared with the invention of electricity, the internet, or the smartphone and will fundamentally transform the way we work, communicate, receive healthcare, travel, and learn [63, 33, 30]. Enabled by exponential advances in *Machine Learning* (ML) and computing capabilities, underlying algorithms allow for a paradigm shift in computer science: instead of being programmed explicitly, ML algorithms *learn* from data itself. These algorithms can automatically find structure in their *training data*, which are condensed to ML *models* that can be used to reason about the underlying problem domain. This versatility of machine learning allows a broad applicability, and ML models are now regularly deployed in practice, including in safety and security-critical systems such as antivirus scanners [54, 98, 34], firewalls [76], autonomous cars [71], CSAM detectors [8], mobile networks [45], and user authentication [102, 39].

The inclusion of machine learning, however, introduces a new attack surface to these systems. An ML model is commonly learnt under the assumption that its training data follow the same statistical distribution that is later observed when the model is used. While this is an effective assumption to guide the training process and generally allows ML models to gain robustness against natural variations of inputs, it implicitly neglects the presence of an adversary. In particular, an adversary can arbitrarily deviate from this distribution and fool a model with carefully computed inputs; thus, undermining the model's integrity [99, 38, 49].

This observation spawned a new field of science about the security and trustworthiness of machine learning, which resulted in a myriad of possible attacks. Besides integrity attacks, prior work discovered the vulnerability of ML models to attacks against their confidentiality [105, 52, 67], availability [85, 95, 96], and also new attack vectors emerged regarding the privacy of user data [74, 4, 26] or fairness of model predictions [29, 116, 46]. But in this field, ML models were predominantly studied in isolation, which is an oversimplification compared to practical applications that deploy models as part of an ML *system* [9, 90]. Such a deployment might introduce additional constraints for an attack and can limit the knowledge that an adversary can gather about the internals of a system. The input and output of an ML component might not be easily accessible from the outside of the system or are part of complex pre- and post-processing pipelines. In particular, pre-processing steps are usually not bijective in structured problem domains, and an attack against an ML model often does not transfer directly to an attack against the system using that model [82, 84]. For this, an adversary must also find an input that manipulates the ML model *and* a suitable input in the underlying problem domain. Also, real-world adversaries are not bounded by the magnitude or the type of their modifications, whereas the literature often makes such assumptions. Finally, and most importantly, the goal of an adversary is typically not to attack an ML component itself but leverage a vulnerability in such a component to an attack against the enclosing system. As a consequence, commonly studied threat models do not sufficiently express the knowledge, capabilities, and goals of practical adversaries.

In this work, we investigate the security of machine learning from a systems security perspective. By considering the ML model as part of a system, we can explore the increased attack surface of practical systems but also analyze how such systems can be secured.

From a defender's point of view, it can also be an advantage to consider an ML model in a larger context. It is challenging to secure generic ML models against all possible attack vectors — demonstrated by the vast body of literature on attacks [e. g., 11, 99, 38, 77, 78, 12, 87] and (broken) defenses [e. g., 19, 20, 17, 108, 79]. By considering the context of the deployment, we can focus our efforts on *relevant* classes of attacks and appropriate defenses. For example, when there is no data flow from the input of the system to an ML component, an attack against the integrity of a model might not be possible and does not need to be considered. Likewise, if there is no data flow from

an ML model outside the system, we might not have to worry about privacy leakage attacks. Furthermore, knowledge about the problem domain can be used to make a model more robust for a specific application. This is akin to a white-list approach in computer security. Rather than defending against all possible inputs, we restrict the inputs to classes of known inputs.

We start our investigation and study an attack against a practical ML system to understand how an adversary can leverage a vulnerability of an internal ML component to an attack against the system itself. As a practical example, we consider systems for automatic paper-reviewer assignments [23, 81]. Such systems are now regularly employed in various scientific fields to support the academic reviewing process and help the program chair of an academic conference or the editor of a journal to find a suitable match of submissions to a group of reviewers. Internally, these systems rely on machine learning for *topic modeling* [13, 28, 48] to extract and represent both reviewers as well as submissions with representative high-level topic vectors, which are used to measure their similarity and to facilitate a good assignment. However, by using machine learning, these systems become vulnerable to manipulations. To demonstrate this, we consider an adversary that adapts its submission to mislead the topic modeling component and selects its reviewers. This is challenging since the topic modeling component is used only internally and is, therefore, not directly exposed by the system. Hence, the adversary cannot directly modify inputs in the input space or *feature space* of this component but only make modifications in the *problem space* of the system (i. e., the system expects PDF files as input). Furthermore, even if the adversary has complete control over the output of the topic modeling component, changing the system's output (i. e., the assignment) is not straightforward because of concurring reviewers and submissions. To address both of these issues, we propose a novel optimization strategy that operates simultaneously in the feature space of the topic modeling component and the problem space of the system. To guide the optimization, we use the system's output as an oracle for the attack. This allows us to integrate constraints among spaces and effectively craft adversarial submissions.

Second, we consider the security of an ML system from the perspective of a defender. ML systems are applied in a specific context, and we can use domain-specific priors to improve their robustness against attacks. As a practical example, we consider voice assistants, which use ML to transcribe spoken content from raw audio signals into text. The internal speech recognition component is vulnerable to hidden commands injected

into inconspicuous audio signals such as news broadcasts, music, or birds twittering. In particular, an adversary can slightly perturb an audio signal to fool the system into transcribing arbitrary target transcriptions or mounting a denial of service attack which cancels out any transcriptions. The perturbations themselves are unobtrusive and barely noticeable for human listeners [21, 89, 87]. Under a realistic threat model, such an attack is always possible as practical adversaries must be assumed to have full control over the input to the system. Hence, in the worst case, the adversary needs to make more significant changes to the input. This again highlights the mismatch between adversaries usually studied in the literature (with bounded perturbations) and real-world attacks. Therefore, rather than preventing the attack in any circumstance, we focus on making the attack *noticeable*. In this case—while still viable—the attack loses most of its malicious potential. The vulnerability in the recognizer stems from a mismatch between the expectations of a human listener and the inner workings of the system [49], and our goal is to align these better. First, we use a model of the human auditory system [50] to identify and filter parts in the input to the system that are *inaudible* to humans. Intuitively, these parts should not contribute any information to the recognizer. They do, however, provide space for an attacker to hide adversarial noise [118, 89]. Second, speech recognition systems typically expect an audio signal with a frequency range of $0 - 8000\,\text{Hz}$. As the task of the system is the transcription of spoken content, we can further reduce the attack surface by restricting the audio signal to frequencies that carry human voice, which are approximately between $300 - 5000\,\text{Hz}$ [72]. By combining both of these insights, we systematically remove hideouts in the signal for an adversary and force an attack into *audible* ranges. Our evaluation shows that this approach is effective, and we find that inputs computed by a strong adversary are of poor quality and are easily distinguishable from benign audio inputs.

Finally, we take a step back and consider the lifecycle of an ML system. Computer systems often run for extended periods of time and are continuously updated and maintained. This also affects internal ML models, which are updated, e. g., with new training data points. Recall that the goal during training is to generalize structure from the training set about the problem domain. For an ML model to perform well on real-world inputs during deployment, this training data must resembles inputs later observed by the model as closely as possible. For this reason, in many cases, ML models are trained on data points directly shared by users of a system (e. g., the speech recognition component of a voice assistant might be trained on recordings from

users of the system [88]). The collection and usage of such data is strictly mandated by regulations like the GDPR [1], CCPA [2], or PIPEDA [3]. Among others, these regulations govern *the right to be forgotten* and, in particular, entitle individuals to self-determine the possession of their private data and also compel a deletion. However, fulfilling such a deletion request can be problematic when the data is used for training an internal ML component. ML models are prone to memorization [31, 15] and can leak information about individual training data points [18, 62]. Consequently, it does not suffice to delete a data point from the training set, but a model trained on this set also needs to be updated. This can be done using *machine unlearning*, which removes data points from the training set of an ML model *after* training [14, 40, 110]. However, a dishonest service provider might not unlearn a data point faithfully to avoid the computational costs associated with updating a model [14, 40], or they might not be willing to pay the cost of degradation in model utility [35, 92]. Therefore, our goal is to make this unlearning verifiable from a user's perspective. For this, we cannot solely consider the ML model in isolation but must capture consistency across model updates and evolving datasets (i. e., we need to consider the *history* of the system using the model). This is to prevent a data point from being re-added at a later iteration of a model. Moreover, the contribution of a data point (towards model parameters) can be approximated from other entries in a dataset, and model parameters can be identical when trained with or without a data point [104, 95]. Hence, any approach that verifies that the influence of an unlearnt data point is absent from the processed parameters is insufficient. To account for these challenges, we formulate unlearning as a security problem with a formal framework and propose an algorithmic approach for verification. In this framework, we instantiate a generic protocol for verifiable unlearning using methods from verifiable computation (such as SNARKs [42, 93]) and hash chains. We prove the security of this protocol based on cryptographic assumptions and practically implement and evaluate the main building blocks for different unlearning mechanisms from the literature.

Thesis contributions. In summary, we make the following key contributions:

- *Feature-problem-space attacks.* We study the increased attack surface of ML systems inherited by internal ML components. We consider a practical scenario and construct an attack against a system for automatic paper-reviewer assignments.

This attack is based on a novel optimization strategy that operates simultaneously in the ML component's feature space and the system's problem space.

- *Domain-specific priors.* We discuss a principled approach to increase the robustness of ML components within a system by integrating application-specific priors. As a practical example, we consider the task of automatic speech recognition. We utilize domain-specific knowledge to better align speech recognition systems with human expectations. This augmentation forces an adversary into audible ranges.

- *Security beyond the model.* Verifiable machine unlearning cannot be solved by naive one-shot approaches. Moreover, it requires proving that the unlearning algorithm was executed; otherwise, the user will not know that their point was indeed deleted. To capture this, we propose an algorithmic approach and a formal security definition. Under this definition, we construct a generic instantiation and prove its security.

Structure of thesis. This thesis is divided into two parts. This first part serves as an introduction to the security of ML systems. The upcoming Chapter 2 introduces the necessary background on machine learning and its properties in adversarial environments. In Chapter 3, we start by introducing ML systems and discuss salient concepts of how the inclusion of ML components affects the security of the system. Our discussion then revolves around three case studies — each highlighting a significant concept of systems-level attacks and defenses. Finally, we conclude our introduction in Chapter 4 and discuss potential directions for future work. The publications underpinning the first part are provided subsequently in the second part in appendices A to C.

Background

This thesis investigates the security of systems that internally rely on machine learning components. As a basis for this, we require background on machine learning and its properties in adversarial environments, which we want to introduce in the following. We start with a short overview of machine learning, focusing on supervised models that are widely used in practice. Based on this, we continue to discuss attack vectors enabled by the underlying learning paradigm. For this discussion, we will focus on adversarial examples as an example of how we can study the worst-case performance of a model. This will lead us to the notion of adversarial robustness and limitations of current countermeasures. Finally, we briefly introduce different threat models used in the literature.

2.1 Machine Learning

A machine learning system is not programmed explicitly but *learnt* from a set of data points or *dataset*. Many different learning paradigms and problem classes exist, but fundamental principles may be best illustrated with supervised machine learning to solve regression and classification tasks. In supervised machine learning, the goal is to learn a mapping between a space of inputs and a space of outputs based on labeled training examples. Other notable learning paradigms include unsupervised learning, when the training data is not labeled, and reinforcement learning where the goal is to learn an agent's policy interacting with its environment.

Supervised machine learning. Supervised machine learning is the process of learning a parameterized function m_Θ (often called a *model*) that can predict an output (from the space of outputs $\mathcal{Y}$) given an input (from the space of inputs $\mathcal{X}$):

$$m_\Theta : \mathcal{X} \to \mathcal{Y}.$$

For a regression problem we want to predict a real-valued scalar $y \in \mathbb{R}$ given an input vector $\mathbf{x} \in \mathbb{R}^n$. This can be done with a linear regression model where m_Θ is of the form

$$\hat{y} = w_1 \cdot \mathbf{x} + w_0,$$

with parameters $\Theta = (w_1, w_0) \in \mathbb{R}^n \times \mathbb{R}$. In a classification task, the output or *prediction* would take a discrete value $\hat{y} \in [1, \ldots, c]$ (i. e., one of c classes) or might describe a probability distribution over all classes. For example, in binary classification, we can use a logistic regression model

$$\hat{y} = \sigma(w_1 \cdot \mathbf{x} + w_0),$$

that can predict the probability of $\mathbf{x}$ belonging to one of two classes. The difference to linear regression is the sigmoid function σ defined as

$$\sigma(t) = \frac{1}{1 + e^{-t}},$$

which projects the output of the model between $[0, 1]$. Parameters of the model need to be carefully chosen such that the output forms the required probability distribution.

Deep neural networks. Both linear and logistic regression are fundamentally limited as they can only express linear relationships between the input and the output. Therefore, more complex models, such as *Deep Neural Networks* (DNNs), are often considered. DNNs are networks of neurons arranged in layers inspired by biological neural networks. The first layer is the input layer, followed by several *hidden* layers, and finally, the output layer. In the case of a fully-connected network with K layer as depicted in Figure 2.1, the k-th layer has s_k neurons which are parametrized by weights $w_{i,j}^{(k)}$ for $k > 1$ and describe the pairwise connection strength between the i-th neuron in layer $k-1$ and the j-th neuron in layer k. The model parameters follow as the set of all weights, i. e., $\Theta = \{w_{i,j}^{(k)}\}$ with i, j, k as defined before. The output of the model is computed from left to right. The input layer contains one neuron for each input value, and

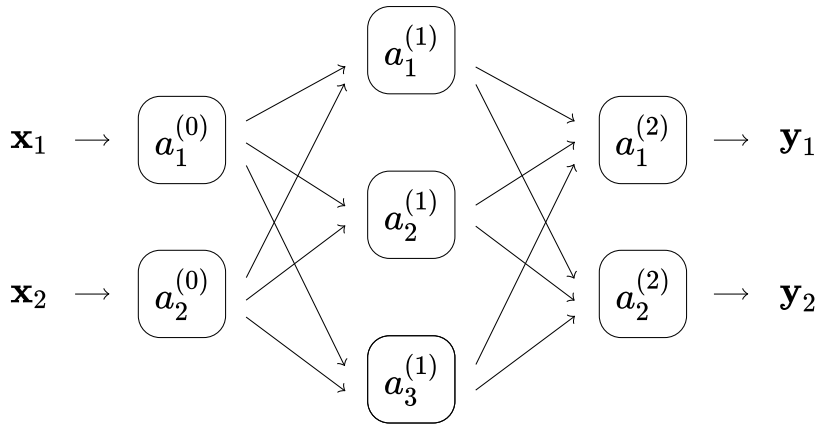


Figure 2.1: **Deep neural network.** Neural network with one hidden layer and two input and output neurons each. This is an example of a full-connected neural network as each neuron in a layer depends on all neurons of the previous layer.

in this first layer, each neuron outputs its input value, i. e., $a_j^{(0)} = \mathbf{x}_j \; \forall j \in \{1, \cdots, s_1\}$. In each following layer k, the j-th neuron computes

$$\hat{a}_j^{(k)} = w_{0,j}^{(k)} + \sum_{i=1}^{s_{k-1}} a_i^{(k-1)} \cdot w_{i,j}^{(k)},$$

which is fed into a non-linear *activation function* such as the sigmoid function σ. This step is crucial to introduce non-linearities in the learnt mapping. The output or *activation value* of a neuron follows as $a_j^{(k)} = \sigma(\hat{a}_j^{(k)})$. Finally, the outputs of all neurons in the final layer are outputted, i. e., $\mathbf{y}_j = a_j^{(K)} \; \forall j \in \{1, \cdots, s_K\}$.

Fully-connected networks are built on the intuition that each layer builds on the computation of previous layers, which allows them to represent complex relationships. They serves as a foundation for many specialized architectures such as *Convolutional Neural Networks* (CNNs) [60], *Recurrent Neural Networks* [86], or *Transformer Models* [109]. In general, there is no universally optimal architecture [111], and the choice depends on the nature and complexity of the mapping that needs to be learnt.

Training an ML model. Once a model m_Θ is selected, we need to find suitable parameters Θ. The *training* aims to learn these parameters from a finite dataset $D \coloneqq \{d = (\mathbf{x}, y) \sim \mathcal{D}\}$ with $\mathcal{D}$ being the true data distribution. To assess a particular set of parameters, we use a *loss function* $l \colon \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_+$, mapping the models predicted and the true output to a positive number that represents the quality of the model's

prediction. A common choice for regression problems is to use the mean squared error loss, while the cross-entropy loss is often used for classification problems.

Regardless of the specific loss function, the goal is to find a model m_Θ that minimizes the following *risk*:

$$\mathsf{R}(m_\Theta) \coloneqq \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}}[l(m_\Theta(\mathbf{x}, y))].$$

In other words, we are looking for parameters Θ where the expectation over the loss is minimal; thus, the model makes the least mistakes. Unfortunately, the true data distribution $\mathcal{D}$ is usually unknown. Therefore, in practice, we approximate this quantity with a finite dataset D which leads to the *empirical risk*:

$$\hat{\mathsf{R}}(m_\Theta, D) \coloneqq \frac{1}{|D|} \sum_{(\mathbf{x},y)\in D} l(m_\Theta(\mathbf{x}, y)).$$

Equipped with this definition, we can now proceed to train model parameters Θ. This can be done with *Stochastic Gradient Descent* (SGD). The idea behind SGD is simple: We start with randomly initialized parameters Θ and iteratively improve upon them. In each step, we consider a data sample $(\mathbf{x}, y) \in D$. For this data point, we first compute the model's prediction $\hat{y} = m_\Theta(\mathbf{x})$ and loss value $l(\hat{y}, y)$. We then compute the derivative from this loss value with regard to the model's parameters Θ, which can be summarized by the gradient vector $\nabla_\Theta l(m_\Theta(\mathbf{x}, y)$ (i. e., each entry of the gradient corresponds to the derivative of one entry in Θ). To reduce the loss of the model on data point $\mathbf{x}$, we subtract the gradient from the current model parameters:

$$\Theta \coloneqq \Theta - \eta \cdot \nabla_\Theta l(m_\Theta(\mathbf{x}, y)).$$

To control the *learning rate* in each step, a scalar value η is used. Multiple passes (called *epochs*) are often repeated through the dataset until the risk is sufficiently low. Note that for this assessment, we cannot use the same dataset as model parameters are prone to *overfitting* to a specific data set. Instead, we use a *test set* sampled from the same underlying data distribution $\mathcal{D}$, which is used to estimate the true risk. The full process of training can be described as

$$\Theta \coloneqq \Theta_{initial} + \sum_{e\in[E]} \sum_{d\in D} \Lambda_{e,d},$$

with E being the number of epochs and $\Lambda_{e,d}$ the update on the model's parameter from data point d in epoch e. In practice, this approach is often extended to batches of data points to reduce each update's variance, and regularization terms are added [37].

2.2 Adversarial Machine Learning

The paradigm of learning from data introduces a new attack surface as learnt models might contain blind spots (e. g., when there is not enough support in the data for certain edge cases), or an adversary might exploit the very nature of training to embed backdoors in the learnt model. In general, we can divide attacks into the three classical security goals: confidentiality [105, 52, 67], integrity [99, 38, 49], and availability [85, 95, 96] attacks but also new attack vectors emerge, such as attacks against a model's privacy [74, 4, 26] or fairness of model predictions [29, 116, 46].

Adversarial examples. Most prominent are integrity attacks during inference, which we want to discuss in the following for the setting of the classification task. Here, the adversary computes an input that fools a model into outputting a false prediction. This modified input is referred to as an *adversarial example*. The very existence of such inputs is not surprising, as every function can be manipulated when controlling the input. What is special about adversarial examples, on the other hand, is their effectiveness: minor modifications of an input often already suffice to have full control over a model's output which highlights the brittleness of current ML models in *worst-case* scenarios.

Formally, our goal is to find a vector δ that perturbs an input $\mathbf{x}$ such that $m_\Theta(\mathbf{x}) \neq m_\Theta(\mathbf{x} + \delta)$. This can be done with the same approach as used during training, but instead of minimizing the loss, we *maximize* it for a specific input point. To control the modification δ, we define a perturbation set Δ of allowed modifications. A common choice for Δ is, for example, to consider an L_p-ball around the input, e. g., $\Delta \coloneqq \{\delta : ||\delta||_\infty \leq \epsilon\}$ for small ϵ with $||t||_\infty = \max_i |t_i|$. The attack objective can then be described by:

$$\underset{\delta \in \Delta}{\text{maximize}}\ l(m_\Theta(\mathbf{x} + \delta), y).$$

We can further extend this strategy to *targeted* attacks where the goal is not only a misclassification but also to choose the target label y_{target}:

$$\underset{\delta \in \Delta}{\text{maximize}}\ l(m_\Theta(\mathbf{x} + \delta), y) - l(m_\Theta(\mathbf{x} + \delta), y_{\text{target}}).$$

This optimization simultaneously tries to maximize the distance to the correct label while minimizing the distance to the target label.

Adversarial robustness. Fundamentally, these attacks result from the way a model is trained. In particular, the goal during training is to minimize the (empirical) risk as introduced above and a model only gains robustness against naturally occurring perturbations (e. g., different lighting conditions for an image classifier). Adversarial examples, on the other hand, are *worst-case* perturbations specifically computed to move input samples across learnt decision boundaries.

As a consequence, adversarial examples can be considered as out-of-distribution samples from the underlying data distribution. Hence, to improve resilience against such outliers, a possible approach is to detect malicious inputs prior to feeding them into the model. Indeed, Grosse et al. [41] showed a significant statistical difference between natural and adversarial inputs. However, once an adversary is aware of a particular detection mechanism, it proved very difficult in practice to prevent the adversary from factoring this into the computation and bypassing the filter [19].

As an alternative approach to detection, it is possible to make the model itself more robust and modify the training goal to take adversarial perturbations into account explicitly. This can be formalized with the *adversarial risk* [66]:

$$\mathsf{R}_{\text{adv}}(m_\Theta) = \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}}[\max_{\delta\in\Delta} l(m_\Theta(\mathbf{x}+\delta, y))],$$

and analogously the empirical adversarial risk

$$\hat{\mathsf{R}}_{\text{adv}}(m_\Theta, D) = \frac{1}{|D|}\sum_{(\mathbf{x},y)\in D} \max_{\delta\in\Delta} l(m_\Theta(\mathbf{x}+\delta, y)).$$

In this definition, we obtain a model that gains robustness against all possible perturbations from a perturbation set Δ. This optimization criterium is, for example, implemented as *adversarial training* [38, 66, 113] and was shown to successfully improve the robustness of the learned model. Interestingly, robust models are often less accurate than their normally trained counterparts as these need to generalize better on so-called *robust features*. On the other hand, a model can use any available signal in the data during standard training [49, 106].

Threat model. To systematically study the security of ML models in adversarial environments, we assume a *threat model* that summarizes the *capabilities*, *knowledge*, and *goals* of an adversary:

- *Goal.* The goal describes the objective of an attack. As discussed earlier, for ML models, an attacker might undermine a model's confidentiality, integrity, availability, privacy, or fairness. The goal also reflects whether an attack is untargeted or targeted.

- *Knowledge.* Adversaries can be distinguished by the amount of knowledge they have on the target model. For example, whether an adversary has access to a model's parameters Θ or its training data D. Depending on this, adversaries are typically classified as *white-box* with full access, *black-box* with no access, or *grey-box* for everything between.

- *Capabilities.* With the capabilities of an adversary, we can describe its abilities to interact with the target. For example, which parts an adversary can manipulate (e. g., inputs to the trained model or the training data) or the magnitudes of modifications (i. e., the perturbation set we assume).

Explicitly defining the threat model allows to make specific claims about the security of a model with respect to the assumed adversary. For example, with adversarial training, a model gains robustness to adversaries restricted by their capabilities defined with the perturbation set. Importantly, this generally does not allow claims for adversaries not covered by the threat model. Therefore, it is critical that the threat model matches real-world adversaries as closely as possible. Unfortunately, prior work often assumes threat models that are too restrictive or do not align well with the goals of real-world adversaries. In the remaining work, we will investigate this observation in detail and consider the security of machine learning from a practical perspective.

Security of ML Systems

We proceed with our discussion about the security of ML systems. These systems internally rely on machine learning components as part of their computation pipeline. Figure 2 depicts a high-level overview of such a system. In general, ML systems are the common way to deploy ML models in practice since underlying models can usually not be used in isolation [9, 90]. For example, an ML model used for detecting pedestrians might be embedded in a larger system of an autonomous vehicle [71]. This system might record a camera feed and uses the ML component only to classify the camera data. An antivirus program might use an ML component as part of a triaging pipeline [98, 54, 34]. Here the system's output might only be the final result: whether the input is benign or malicious. Smartphones use machine learning for face or fingerprint recognition to authenticate their users and many more [102, 39, 8].

By using machine learning, these systems inherit the attack surface of internal ML models, which makes them vulnerable to attacks against these. But, extending an attack from an ML model to an attack against the system is more complex. The output of a system might not be the output of the ML component; likewise, an input can be pre-processed by a larger pipeline. Hence, an adversary might only have limited feedback from the output of the ML component or only limited control over the input. Moreover, the functionality of the system itself could be unknown. Systems can be composed of several components with complex information flows between them. Most critically, however, the goal of an adversary is usually not to attack an ML model but leverage such an attack against the enclosing system. Consequently, compared to

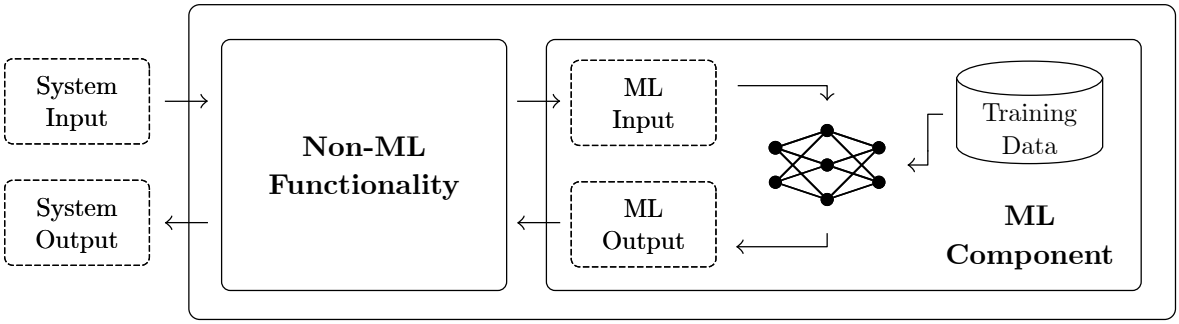


Figure 3.1: **ML System.** An ML system uses machine learning internally in a subcomponent. This component is not programmed explicitly but *learnt* from a set of training data points. As the system also uses non-ML functionality, inputs to the ML this component might differ to the input to the overall system and the system output might differ from the output of the ML component.

attacking an ML model in isolation, adversaries against ML systems differ in their *capabilities*, *knowledge*, and *goals*.

In the following, we want to categorize and discuss such differences in more detail and when the system perspective might even be an advantage to fend off an attack. Subsequently, we consider three different case studies to highlight salient concepts further.

Additional constraints. ML models expect inputs from their input space. In an ML system, we therefore need to convert the inputs from the system prior to feeding it to an ML component. This conversion is facilitated with a *feature extractor* that projects inputs from a systems *problem space* into a vector of a models *feature space* (i. e., its input space). Feature extractors can be very simple (e. g., taking an image's grey scale pixel values) but can become very complex (e. g., parsing a malware executable file into high-level features).

Most prior works on the robustness of ML models focus on attacking models in their feature space. The conversion from the problem space into the feature space, however, needs to be considered for an attack against the system. Unfortunately, the mapping between spaces is usually not bijective as multiple inputs might map to the same feature vector, and not every feature vector can be realized in the problem space [82, 84]. In Section 3.1, we analyze in detail how an adversary can integrate such constraints into an attack that simultaneously considers both the feature space and problem space. This

case study considers a real ML system used for paper-reviewer assignments during the academic reviewing process. Interestingly, the mapping can also be used as an attack vector. For example, in image-scaling attacks, an adversary can manipulate an image to change its appearance after being downscaled during the pre-processing of an image classifier [83], or unintelligible commands can be hidden in an audio signal and uncovered by the signal processing of a speech recognition system [5].

From the side of a defender, it can be beneficial to consider the entire computation pipeline and not only the feature space to improve the robustness of the ML component. By considering the problem space, it is possible to *add* constraints on the feature extraction. This can be in the form of domain-specific knowledge, e. g., we add restrictions to the input space of the ML model to *reduce* the attack surface of a model. We investigate this in Section 3.2 and look at the robustness of automatic speech recognition. We consider a model of the human auditory system to restrict the feature extraction from audio signals to ranges that are audible for human listeners. We demonstrate that this forces an adversary into audible ranges, significantly reducing the attack surface.

Information flow. An implicit assumption of feature-space attacks is that an adversary has complete control over the input and output of the targeted model. The problem-feature-space conversion is one example where this might not hold when considering a model in a larger context. Furthermore, from a system's point of view, it might be possible that there is no connection between a system's and the model's input (e. g., when an ML component is solely used internally [90]). This allows a defender to rule out complete classes of attacks by tracking the information flow between components. This is especially important as defending the ML model against every possible attack is generally infeasible, and countermeasures are expensive to deploy [97, 25, 107].

Lifecycle of a system. Prior work often implicitly assumes static models trained once on a fixed training set. However, this assumption does not hold in practice where the underlying data distribution might shift, and model's need to be continuously updated. Consider, for instance, an ML system for malware detection where the internal model needs to keep up with novel attack vectors. On the one hand, this can add uncertainty to an attack [94] since a particular model variant might be unknown to the adversary. On the other hand, this can also further increase the attack surface. In particular, an ML model can "forget" about old data points, known as *catastrophic*

forgetting [55], and thus become less confident and more vulnerable to inputs from this region.

Furthermore, a service provider might collect data during the runtime of a system. Adding this data to the training set of an ML model can address distribution shifts and generally improve the alignment between the training data and data observed during runtime. Section 3.3 considers the scenario where a service provider might be legally compelled to remove such data samples. Specifically, we focus on the case that data points were used for training an ML model, and a service provider is required to update the ML model to remove any influence of this data. To capture the consistency of model updates and evolving datasets, we must also consider a *system's history*.

3.1 Feature-Problem-Space Attacks

Equipped with this overview, we now want to focus our attention on three concrete case studies for attacking and defending ML systems. Recall that an adversary against an ML system needs to consider the whole processing pipeline, and an attack against an internal ML model is just one piece of a successful attack. In the first case study, we therefore want to understand how we can practically leverage a vulnerability within an ML subcomponent for an attack against the enclosing system. As an example, we will consider assignment systems as they are now increasingly used during the academic reviewing process. They are an excellent example since they internally rely on machine learning as a subcomponent and work on PDF files requiring non-trivial pre-processing.

Paper-reviewer assignments. The task of the assignment system is to match a set of reviewers $\mathcal{R}$ with a set of submissions $\mathcal{S}$ in a way to optimize the similarity between reviewer and submissions while achieving a balanced assignment (i. e., each submission should be assigned to $\mathsf{L}_{\mathbf{x}}$ reviewers and each reviewer should review at most L_r submissions). These systems use machine learning to distill the expertise of reviewer and the contents of submissions to facilitate a good match. For this, each reviewer $r \in \mathcal{R}$ is represented by a set of selected publications A_r. With these archives and the set of submissions, the assignment is roughly divided into three steps [81, 23]:

- **Text pre-processing.** Assignment systems typically work on PDF documents as their inputs. The first step is to convert such a PDF document z from the problem space $\mathcal{Z}$ of a system into a vector representation $\mathbf{x}$ from the feature

space $\mathcal{F}$ amenable for further computations. Therefore, a text extractor (such as `pdftotext`) is used to extract text from the PDF file z. The extracted text is then (1) tokenized into individual words, (2) numbers or stop words that do not carry much meaning (e. g., *the*, *a*, ...) are removed, and (3) only word stems are considered (e. g., *attacker*, *attack*, *attacking*, ... are mapped to the same stem *attack*). This tokenization and normalizations steps are summarized by a preprocessor function φ. After that, the normalized words are converted into a vector representation using a feature extractor ϕ by counting the number of occurrences of words from a vocabulary $\mathcal{V}$ (i. e., a *bag-of-words*). The result of this pre-processing is a vector $\mathbf{x} \in \mathbb{N}^{|\mathcal{V}|}$.

- **Topic modeling.** The key to the assignment is the automatic extraction of high-level topics from these word vectors. Formally, a word vector $\mathbf{x}$ is mapped to a low-dimensional topic vector $\theta_{\mathbf{x}}$ from a topic space $\mathcal{T}$:

$$\Gamma \colon \mathbb{N}^{|\mathcal{V}|} \longrightarrow \mathcal{T}, \quad \mathbf{x} \mapsto \theta_{\mathbf{x}}.$$

  A common instantiation for Γ are unsupervised topic modeling techniques such as the *Latent Dirichlet Allocation* (LDA) [48, 13]. LDA assumes a generative probabilistic process that creates a set of documents representing each document by a random mixture over a set of latent topics. Training a LDA model on a corpus of documents allows one to discover this topic space automatically. Moreover, we can use the model to infer the implicit topic vector from a new document. Intuitively, similar documents are "close" in this topic space and, thus, allow us to measure their similarity using this representation.

- **Paper-reviewer assignment.** Using the extracted topics, a similarity score $b_{r,\mathbf{x}}$ between a reviewer $r \in \mathcal{R}$ and a submission $\mathbf{x} \in \mathcal{S}$ is computed by taking the dot-product of their topic vectors

$$b_{r,\mathbf{x}} \coloneqq \Gamma(A_r) \cdot \Gamma(\mathbf{x})^\top.$$

This score is high when topic vectors share many topics and low otherwise. Based on the pairwise similarity, we can then compute a ranking of reviewers for each submission and formulate the final assignment A as

$$\begin{aligned} \underset{\mathsf{A}}{\text{maximize}} \quad & \sum_{r} \sum_{\mathbf{x}} b_{r,\mathbf{x}} \cdot \mathsf{A}_{r,\mathbf{x}} \\ \text{subject to} \quad & \mathsf{A}_{r,\mathbf{x}} \in \{0,1\} \; \forall r, \mathbf{x} \\ & \sum_{r} \mathsf{A}_{r,\mathbf{x}} \leq \mathsf{L}_{\mathbf{x}} \; \forall \mathbf{x} \text{ and } \sum_{\mathbf{x}} \mathsf{A}_{r,\mathbf{x}} \leq \mathsf{L}_{r} \; \forall r, \end{aligned}$$

with load constraints L_x and $\mathsf{L}_\mathbf{r}$ as defined above [101].

Adversarial papers. We consider an adversary that, given a submission z, aims to find a submission z' which gets assigned a targeted set of reviewers in A. Therefore, we assume that the adversary modifies z to manipulate the topic model and move the submissions similarity closer to selected and farer from rejected reviewer, which subverts assignment A. To highlight the difference to a classical attack against the ML model itself, let us further assume that the adversary has full control over the topic vector (i. e., they can attack the ML model in its feature space). In this case, the adversary still needs to overcome two important challenges: First, (1) they need to find a topic vector that results in the targeted assignment (i. e., manipulate the output of the system), and second, (2) find an input in the problem space of the system (i. e., a PDF file) that is mapped to the corresponding feature vector inputted to the topic model. Both steps are non-trivial as they depend on the relationship between different reviewers on the final assignment and the mapping from the input to the system and extracted word counts. To address this, we consider a hybrid optimization strategy that uses the system's output as guidance for the attack and alternates between the feature space $\mathcal{F}$ of the topic model and the problem space $\mathcal{Z}$ of the system as depicted in Figure 3.2. In the following, we describe the attack individually in feature space respectively problem space and then introduce the combined optimization problem.

Feature-space attack. Let $\mathbf{x}$ be the extracted word counts from submission z. In the feature space, we want to manipulate reviewers $R_\mathbf{x}$ assigned to z and, more specifically, select and reject arbitrary reviewers from $R_\mathbf{x}$. Formally, we define two sets, R_{sel} and

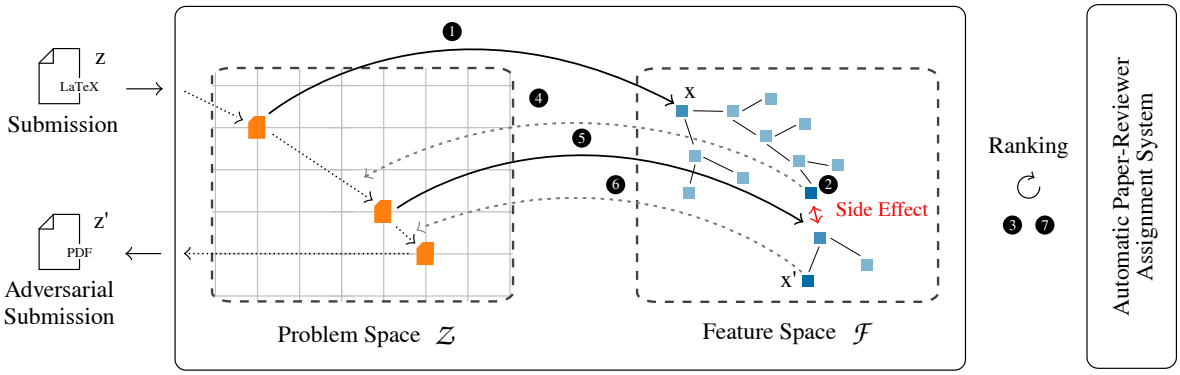


Figure 3.2: **Feature-problem-space attack.** Given an initial submission z, the goal is to find a submission z' that gets assigned the target reviewers. In the first step ❶, the submission is converted into a word vector $\mathbf{x}$ in feature space. The feature-space attack modifies this vector to change assigned reviewers (step ❷ and ❸). Using the problem-space attack, the modifications are projected back into the problem space (step ❹). Due to side effects and limitations of transformations, the submission shifts in problem space. Therfore, the attack is repeated from this new starting position until it is successful (❺–❼).

R_{rej}. Our goal is to find a modification vector $\delta \in \mathcal{F}$ such that the modified submission vector $\mathbf{x}' = \mathbf{x} + \delta$ fulfills

$$\begin{aligned} r \in R_{sel} &\Rightarrow r \in R_{x'}, \text{ and} \\ r \in R_{rej} &\Rightarrow r \notin R_{x'}, \forall r \in \mathcal{R}. \end{aligned} \tag{3.1}$$

To restrict the amount of modifications, we further require $||\delta||_1 \leq L_1^{\max}$ and $||\delta||_\infty \leq L_\infty^{\max}$. The former allows to restrict the absolute number of modifications to the submission, and the latter prevents individual words from being added or removed too often, which might raise suspicion. Based on Equation 3.1, an adversary needs to make changes in the latent topic space defined by the underlying topic model. This is difficult as topic vectors are computed as an expensive and, more importantly, probabilistic simulation procedure. Thus, typical gradient-style attacks are not applicable. Furthermore, Zhou et al. [119] showed that manipulating—when formulated as a combinatorial problem—is *NP-hard*.

To address this, a common approach would be to replace the inference procedure with an efficient approximation such as a neural network [47, 117]. However, such approximations unavoidably introduce noise in the inferred topic vectors [119]. Critical, in our setting, submissions need to be maneuvered very carefully in the topic space. Con-

sider, for example, the case with two reviewers, r_1 and r_2, that share most topics (i.e., $\Gamma(A_{r_1}) \approx \Gamma(A_{r_2})$). Increasing the similarity between r_1 and the submission inevitably also increases the similarity to r_2. Factoring this in requires very precise estimates of the topic vectors.

For this reason, we use a different approach and rely on a stochastic beam search guided directly by the output of the topic model. This allows us to work on the *exact* topic vectors and efficiently navigate the submission through the topic space. In each step of the search, we create a set of candidate submissions by adding and removing words from the feature vector. These candidates get rated using the ranking outputted by the assignment system, and the search is continued with the top candidates. This allows us to integrate the target assignment into the optimization objective, and the result are modifications δ such that $\mathbf{x}' = \mathbf{x} + \delta$ fulfills Equation 3.1.

Problem-space attack. In the second part of the attack, modifications δ need to be realized in the problem space. Therefore, we modify the PDF file to add and remove words from the extracted feature vector. For this, we define problem-space transformations

$$\omega \colon \mathcal{Z} \to \mathcal{Z},\ z \mapsto z',$$

that allow introducing modifications to the extracted word counts $\mathbf{x}$. We consider three different classes of transformations: format-level transformations that hide modifications directly in the PDF format, encoding-level transformations that exploit the text encoding (such as the substitution of characters), and text-level transformers that only work on the visible text (e. g., we include paragraphs of text sampled from a language model or replace words with a synonym).

In general, format-level transformations are the most capable but—once detected—are typically not deniable. In contrast, text-level transformations only perform changes to the visible text and can thus be plausibly deniable. However, as these transformations are visible, we need to introduce an additional attack budget that trades off conspicuousness with the ability to introduce modifications. For example, it might be suspicious if too many spelling mistakes are added. To understand if resulting submissions remain unobtrusive for a given budget, we perform a user study (cf. Appendix A). In this study, we observe that human reviewers generally struggle to differentiate between naturally occurring noise in the submission (e. g., a spelling mistake or redundant references) and the modifications introduced by our transformations. The total detection precision of 33% further underlines this with a recall of only 8%.

Using the transformations, we can define a chain $\Omega = \omega_k \circ \cdots \circ \omega_2 \circ \omega_1$ that successively transform the targeted submission into an adversarial paper. We require that this transformations preserve the semantics and plausibility such that the submission remains meaningful and modifications remain inconspicuous. We summarize these constraints as Υ and write $\Omega(z) \models \Upsilon$ if a chain of transformations Ω fulfills these constraints.

Problem-feature-space attack. Combining both constraints from the feature space and the problem space, we arrive at the following optimization problem:

$$\begin{aligned} & r \in R_{\text{sel}} \Rightarrow r \in R_{\mathbf{x}'}, \text{ and} \\ & r \in R_{\text{rej}} \Rightarrow r \notin R_{\mathbf{x}'}, \forall r \in \mathcal{R} \\ \text{subject to} \quad & ||\delta||_1 \leq L_1^{\max} \text{ and } ||\delta||_\infty \leq L_\infty^{\max}, \\ & \Omega(z) \models \Upsilon \end{aligned} \tag{3.2}$$

with $\mathbf{x} = \phi \circ \rho(z)$, $\mathbf{x}' = \phi \circ \rho(\Omega(z))$, and $\delta = (\mathbf{x}' - \mathbf{x})$. We design a novel optimization strategy alternating between the problem space in the feature space attack. Figure 3.2, depicts an overview of this strategy. The adversary starts the attack by converting the original submission into the feature space of the topic model (step ❶). Subsequently, the feature-space attack computes a modification vector (step ❷ and ❸). Using the problem-space attack, these modifications are mapped back into the problem-space (step ❹). Due to side effects and limitations of transformation, the submission unavoidably shifts in the discrete problem space. Therefore, we continue the search from this new position until the attack is successful (step ❺–❼). The full attack is described in Appendix A. Our analysis shows that it is effective to subvert the assignment of an automatic system.

3.2 Domain-Specific Priors

In the previous case study, we discussed a principled approach to integrate problem-space constraints into an attack against ML systems. Now, we want to switch sides and investigate how to use information about the problem space to make a system more robust. As a practical example for this investigation, we focus on voice assistants. They enjoy increasing popularity and are integrated as personal assistants into our smartphones, cars, or as stand-alone devices at home. Voice assistants continuously listen in their environment for spoken commands to perform different tasks such as controlling

music, sending personal messages, or as the control center of a smart home. Internally, they use ML to transcribe spoken content into text, interpreted as commands.

Our particular interest is in the robustness of this speech recognition component, which we want to improve by embedding domain-specific knowledge about speech. Speech recognition is an ideal example because the properties and limitations of the human auditory system are very well studied. In practice, there are different architectures of speech recognition pipelines. We focus on hybrid architectures deployed, for example, in Amazon Alexa [89] or Sonos Voice Assistant [7]. In this case, the speech recognition component consists of three parts:

- **Feature extraction.** The input to the speech recognition component is a raw audio wave. This signal is pre-processed into a high-level feature representation in the first step. Therefore, the input waveform is divided into overlapping frames, and each frame is transformed with a *Discrete Fourier Transform* (DFT). Features are then given, e. g., as log-scaled magnitudes of the DFT-transformed signal.

- **Acoustic model DNN.** In the next step, a deep neural network predicts the probabilities for distinct speech sounds (i.e., *phones*) in each frame. The phonetic description, together with their context, are described by a language model. Thus, the DNN outputs the most likely state in this language model rather than directly predicting phones.

- **Language model.** Finally, given all outputs from the neural network, the language model is used to decode the most probable transcription. For this purpose, dynamic programming algorithms (e. g., Viterbi decoding [32]) are used to find the most likely path in the language model.

Hidden commands. In a voice assistant, the speech recognition component is directly exposed to an adversary as the component works on the raw signal. This makes voice assistants vulnerable to audio adversarial examples computed against this component. An adversary can inject hidden commands into an audio signal, which are inaudible for human listeners but let the speech recognition model hallucinate arbitrary transcriptions. For example, an adversary can use unsuspicious signals such as music or birds twittering and add slight noise to this signal. The modified signal sounds

benign for humans, but the system transcribes an adversary-chosen target text with high confidence [21, 89, 87].

Alignment from human and system. Such attacks with audio adversarial examples are difficult to fend off. In practical scenarios, we must assume that an adversary has full control over the input. Consequently, an adversary can always succeed; in the worst case, they have to change the complete input. However, at a certain level of modifications, an attack becomes noticeable, diminishing its malicious potential. Therefore, we want to look at audio adversarial examples from a different perspective: *When we accept that adversarial examples exist, what else can we do?* The main problem with current attacks is that they can be carried out inconspicuously, pointing at the mismatch between the inner workings of the system and the human auditory system. To bridge this gap, our goal is to align the system with the auditory system better. In other words, we want to make the attack audible.

Psychoacoustics. We base our construction on psychophysics, and, in particular, we are interested in the subfield of psychoacoustics, which allows us to describe the limitations of the human auditory system [120]. This allows to identify and remove ranges of the audio signal which are inaudible for humans and thus should not carrier relevant information for the transcription. Furthermore, the task of the speech recognition system is to transcribe spoken content. Thus, we can restrict the bandwidth of the input to those frequencies that carry human voice. More formally, we augment the speech recognition pipeline by adding two filtering steps:

Psychoacoustic filtering. For our construction, we use the psychoacoustic hearing thresholds from the MPEG-1 psychoacoustic model [50]. These thresholds define how dependencies between certain frequencies mask other audio signal parts. Intuitively, these parts of the signal should not contribute any information to the recognizer. They do, however, provide space for an attacker to hide adversarial noise. Based on this, we derive a mask to remove inaudible parts of the audio. We compare the absolute values of the complex valued *Short-Time Fourier Transform* (STFT) representation of the audio signal $\mathbf{S}$ with the hearing thresholds $\mathbf{H}$ and define a mask via

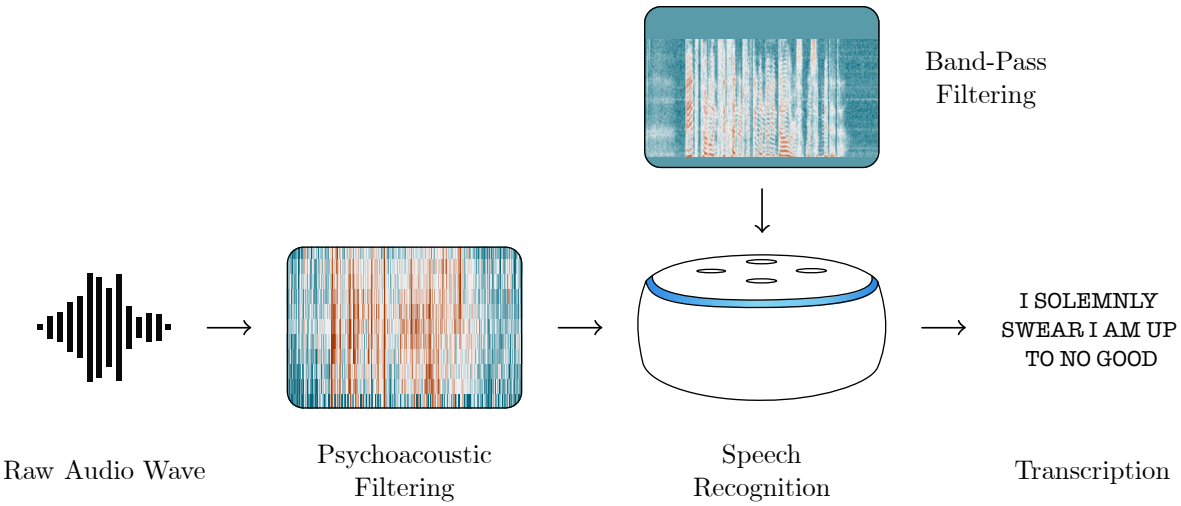


Figure 3.3: **Augmented system.** To augment the speech recognition component, we use psychoacoustic filtering to remove range that are inaudible for human listeners. Moreover, we use a band-pass filter to restrict the audio signal to those frequencies used by human voice.

$$\mathbf{M}(i,j) = \begin{cases} 0 & \text{if } \mathbf{S}(i,j) \leq \mathbf{H}(i,j) + \Phi \\ 1 & \text{else} \end{cases}, \tag{3.3}$$

with $i = 0, \ldots, I-1$ and $j = 0, \ldots, J-1$. We use the parameter Φ to control the effect of the hearing thresholds. For $\Phi = 0$, we use the hearing thresholds exactly. For values $\Phi > 0$, more aggressive filtering is applied, and for smaller values, we retain more from the original signal. We then multiply all values of the signal $\mathbf{S}$ with the mask $\mathbf{M}$

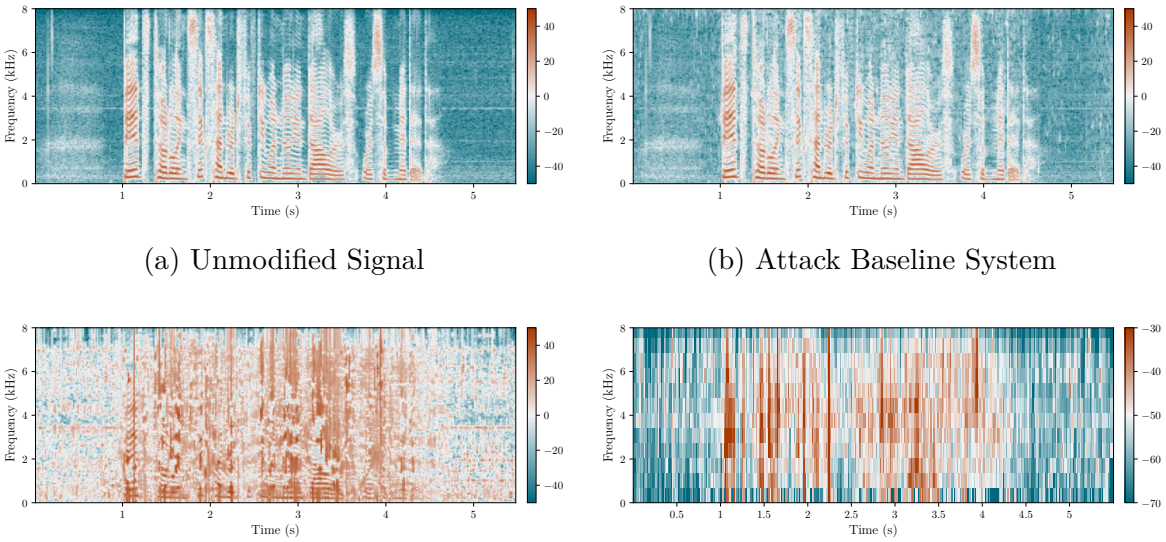
$$\mathbf{T} = \mathbf{S} \odot \mathbf{M}, \tag{3.4}$$

to obtain the filtered signal $\mathbf{T}$.

Band-pass filter. Secondly, the human voice frequency range is limited to a band of approx. $300 - 5000\,\text{Hz}$, we also add a band-pass filter to reduce the attack surface further. This can be described as

$$\mathbf{T}(i,j) = 0 \quad \forall\, \mathsf{f}_{\max} < j < \mathsf{f}_{\min}, \tag{3.5}$$

where $\mathsf{f}_{\max}$ and $\mathsf{f}_{\min}$ describe the lower and the upper cut-off frequencies of the band-pass.



(a) Unmodified Signal

(b) Attack Baseline System

(c) Attack Augmented System ($\Phi = 12$)

(d) Hearing Thresholds

Figure 3.4: **Spectrograms of adversarial examples.** Figure 3.4a shows the unmodified signal, Figure 3.4b depicts the baseline with an adversarial example computed against the unmodified speech recognition component, Figure 3.4c an adversarial example computed with the adaptive attack against the augmented component, and Figure 3.4d shows the computed hearing thresholds for the adversarial example.

Results. Based on these principles, we augment a system as depicted in Figure 3.3. In Appendix B, we describe this construction in detail. The required amount of modifications that an adversary needs to introduce depends on various parameters. For example, if only a few words need to be changed to achieve the targeted transcription, the attack is generally easier. Also, the choice of the target utterance itself is essential. If the corresponding signal does not include spoken content, it is much easier to force the recognizer into an output [89] since the attack does not need to "remove" the actual content from the transcription first.

In our experiments, we therefore consider three different types of audio signals (i. e., birds, music, and speech samples). Moreover, we chose the target transcription such that it is both efficient and effective to introduce in order to decouple its influence on the attack; that is, we are interested in "easy" attacks to understand better understand the effect of the augmentations rather than the effect of more complex target sentences.

We observe that (1) restricting the input to those frequencies that carry human voice can help the system to improve transcriptions (indicating that indeed unnecessary

information is included) and (2) by removing parts of the input that are inaudible for human we can successfully force an adversary into audible ranges which makes the attack clearly perceivable. Figure 3.4 shows the power spectra between the unmodified signal and an attack against the baseline and augmented systems. We observe that adversarial examples computed against the augmented system are of poor quality and are easily distinguishable from benign audio and adversarial examples constructed for the baseline system.

3.3 Security Beyond the Model

For the last case study, we want to take a step back and look at the lifecycle of an ML system. In many cases, learnt components are continuously updated, for example, to improve their performance or to address a distribution shift between the training data and currently observed data [44]. Recall that ML models are trained with the assumption that statistical properties of data points used during training closely resemble the properties of data points the models sees during deployment. To achieve this, in many cases, service providers collect and use data shared from users of the system. For example, a service provider might store the interactions from a user with a system and use this data later as part of an ML model's training data (e. g., recordings from a voice assistant [88]).

Machine unlearning. In practice, the collection and usage of this data is strictly mandated [1, 2, 3]. In particular, *the right to be forgotten* entitles individuals to self-determine the possession of their private data and also compel a deletion. However, fulfilling such a deletion request can be problematic when the data is used for training an ML model as these can leak information about their training data points [18, 62]. Consequently, deleting a data point from the training set does not suffice, but any model trained on this set needs to be updated as well.

This can be done with *machine unlearning*, which allows to remove training data points from an ML model *after* training. Recall that a trained model can be described as a sum over individual model updates (cf. Section 2.1). To remove a data point d^* from model parameters Θ, we are looking for a model with parameters Θ' defined as follows:

$$\Theta' := \Theta_{initial} + \sum_{e \in [E]} \sum_{d \in D \setminus \{d^*\}} \Lambda_{e,d}.$$

The canonical approach to achieve this is *re-training* the model without the data point d^*. Albeit computationally expensive, it has the advantage that the contribution of a data point is completely removed from the parameters. In general, unlearning can broadly be classified into two categories: exact unlearning (such as re-training), where there are guarantees that the datas contribution is entirely removed [14, 16, 114, 75], and approximate unlearning, where the guarantees tolerate some error [43, 24, 103, 10, 91, 36, 40, 110] which are often more efficient.

Verifiable machine unlearning. Regardless of the unlearning technique, verifying that a data point is unlearnt is difficult from the users perspective. A naïve solution might be to provide users access to the model's parameters and ask them to locally run influence techniques [56] to understand if their data point contributed to the model. However, even under this strong assumption (i. e., granting access users to the parameters), such an approach does not suffice. Recent works demonstrate that a models parameters can be identical when trained with or without a data point [95, 104].

The data attribution problem is far more fundamental; correlated entries in a dataset may result in similar contributions to the models final parameters. Imagine that two users, A and B, have the same (or even similar) data: if A requests unlearning but not B, influence techniques indicate that data from A still affects the model, which would be possible even if the server had unlearnt their data because the data of user B is still in the models training set.

Therefore, to prove unlearning, we pursue an algorithmic approach. Rather than trying to verify unlearning by examining changes in the model, we require the server to present a proof that the unlearning was correctly executed. This proof consists of two parts:

- *Proof of training.* Unlearning can only be proven with respect to a model m and a dataset D. Hence, first, we need to establish that model m was obtained from dataset D. Therefore, the server produces a *proof of training*.

- *Proof of unlearning.* Whenever a data point d^* is deleted from the training data D of model m, we require a *proof of unlearning* that proves that an unlearning

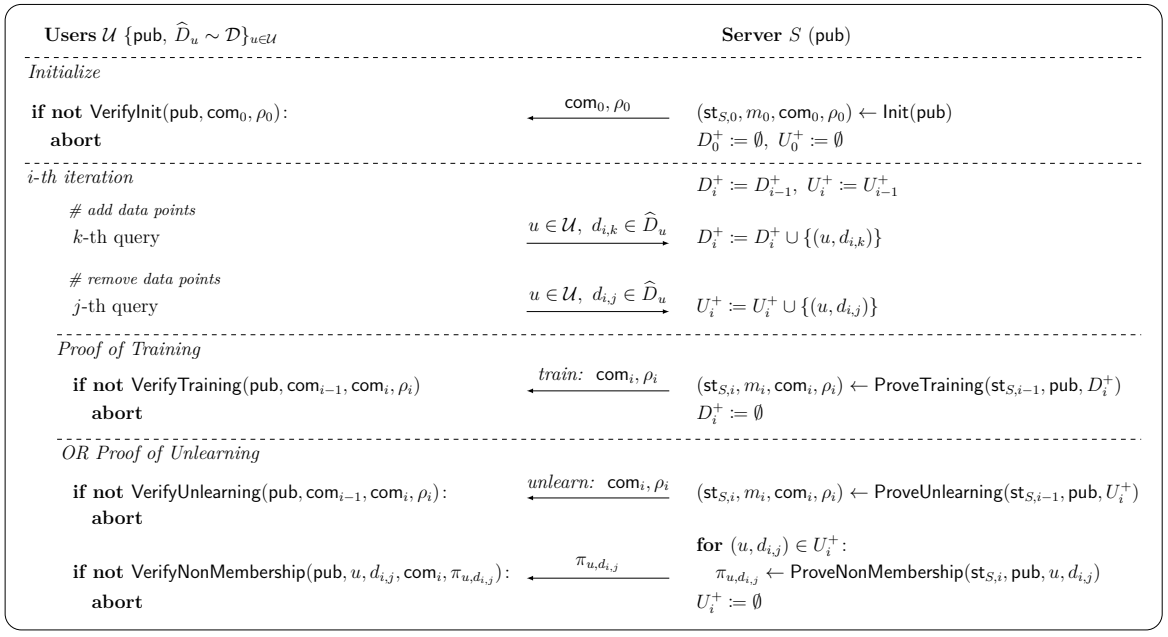


Figure 3.5: **Unlearning Framework.** In this framework, protocols for verifiable machine unlearning can be instantiated. These are executed iteratively between a set of users $\mathcal{U}$ and a server S.

algorithm was executed that removes the contribution of data point d^* from the model's parameters. This proof establishes that the updated model m' is now *conceptually* trained on a dataset D' that does not include d^*. To verify that d^* is not part of this training data, we further require an additional proof of this non-membership, i. e., $d^* \notin D'$.

Furthermore, we need to capture the consistency of the training data across model updates and evolving datasets. This requires an iteration-based protocol and cannot be solved by naive one-shot verifications.

Unlearning framework. In Appendix C, we present a formal framework to describe such iteration-based protocols for unlearning. Figure 3.5 shows an overview of this framework. Our focus is on ML systems trained by a server S using data shared by users $\mathcal{U}$. We assume that each user $u \in \mathcal{U}$ holds a dataset $\widehat{D}_u$ sampled from the distribution of data points $\mathcal{D}$. To describe training and unlearning algorithms, we define a generic interface of *admissible functions* based on triplets (f_I, f_T, f_U) for initialization, training,

and unlearning (respectively). These functions are agreed upon by the users and server and are part of public parameters pub.

The framework considers protocols where the execution proceeds in iterations after an initialization phase. At the beginning of each iteration i, users can request the addition or removal of data points. These are stored in sets D_i^+ and U_i^+. After this, the server either performs a *proof of training* or a *proof of unlearning*. For training, the server updates model m_i with data points in D_i^+ using training function f_T. The server then commits to the updated model and dataset with com_i. Furthermore, the server computes proof ρ_i to prove the correct execution of f_T. Both the commitment and proof are sent to and verified by the users. For unlearning, the procedure first follows analogously with U_i^+ and unlearning function f_U. In addition, for each data point $d_{i,j} \in U_i^+$, the server subsequently creates a *proof of non-membership* $\pi_{u,d_{i,j}}$ that establish that data point $d_{i,j}$ from user u is not part of the current training data set. This proof can be verified from user u against the current commitment com_i.

Instantiation. To instantiate a protocol in this framework, we need a mechanism to prove the execution of functions f_I, f_T, and f_U. For this, we base our construction on *Verifiable Computation* (VC) and, more specifically, SNARK [42, 93] proof systems as a generic approach to proving the correct execution of algorithms. We define the execution of f_I, f_T, and f_U in terms of statements in the proof system and use this to construct proofs that the users can verify.

Second, for the dataset commitment, we internally split the dataset into training data D and unlearnt data U. These sets are updated whenever data points are added or removed from the dataset. Based on these, we maintain two hash lists from which we use the head as the current commitment.

Appendix C contains a complete description of this construction. Here, we also prove the correctness of this instantiation within the framework and its security based on cryptographic assumptions. Moreover, we implement the main building blocks for three different unlearning mechanisms from the machine unlearning literature and compare their performance on different ML models and benchmark datasets.

Conclusions

This thesis investigates the security of machine learning models from a systems security perspective. In this final chapter, we summarize our salient findings and discuss potential directions for future work.

Concluding remarks. The inclusion of an ML model can make a system more vulnerable. Yet, its deployment in a practical system introduces additional constraints for an adversary, and an attack against the ML model is only a single step of a successful attack against a system. We believe that the robustness of ML models cannot be assessed in isolation and must be viewed within the context of the enclosing system. From a research perspective, we are still in the early stages. More work is required to understand the attack surface introduced by including ML models in a system and how we can effectively protect our systems. This is becoming an urgent endeavor as the integration of ML components exploded in the past and will likely continue to grow.

Future research directions. In the following, we want to discuss the research directions necessary to pursue in the future.

Threat model. There is a mismatch between the study of model robustness and the security of systems that uses these components. Currently assumed threat models are insufficient and often do not transfer to real-world deployments of ML models [9]. In this thesis, we discussed essential aspects and implications of this mismatch, but this can only be considered a starting point. One of the most important directions for

future work is to depart from commonly studied threat models and consider real-world adversaries against real-world systems. Future research must align assumptions of an adversaries goals, knowledge, and capabilities to real threat vectors and extend the study of monolithic and static ML models to practical systems.

Classical systems security. ML components are often used for tasks that are hard to solve programmatically but amenable to be learnt from large sets of data (e. g., classifying images [58], voice recognition [115], or recognizing faces [100]). Inspired by these successes, recent efforts increasingly investigate how learnt components can replace heuristics in "classical systems". For example, in a database system, ML can optimize the internal performance [57, 68, 70, 69, 80] or be used for automatic optimization of configurations [53, 64, 6]. Another example are memory management systems that can use ML for memory allocation [65] or to optimize garbage collection [51, 22]. Similar to our discussion, these replacements with learnt components unavoidably open up a new attack surface [90] but, more critically, might even allow new forms of attacks on the interplay with classic attack strategies. For example, an adversary could construct an attack against a learnt allocator and use this to escalate privileges in the underlying system by enabling a memory safety vulnerability. Future work needs to understand such attack vectors and their associated risks.

Countermeasures beyond the model. Since the advent of attacks against ML models, the research community has put significant effort into understanding what enables these attacks and constructing appropriate countermeasures. This resulted in promising approaches for improving a model's robustness with empirical (e. g., adversarial training [38, 66, 113]) and formal guarantees (e. g., certified robustness [27, 112, 61]). Ultimately, however, all current approaches are still limited to toy problems and do not scale to the capabilities of realistic adversaries. In this work, we looked at model robustness from a system's perspective and how domain-specific knowledge can be used to design an application-specific countermeasure. Integrating information about a particular deployment of an ML model and considering robustness as a systems security problem is a promising direction for further research. For example, by using established techniques from the systems security literature, such as information flow control [59, 73], we can track information through a specific deployment of a system and decide if a model is accessible (and thus vulnerable) from an attacker.

References

[1] General Data Protection Regulation (GDPR). Official Legal Text, 2016.

[2] California Consumer Privacy Act (CCPA). Official Legal Text, 2018.

[3] Personal Information Protection and Electronic Documents Act (PIPEDA). Official Legal Text, 2019.

[4] Martín Abadi, Andy Chu, Ian J. Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep Learning with Differential Privacy. In *ACM Conference on Computer and Communications Security (CCS)*, 2016.

[5] Hadi Abdullah, Washington Garcia, Christian Peeters, Patrick Traynor, Kevin R. B. Butler, and Joseph Wilson. Practical Hidden Voice Attacks against Speech and Speaker Recognition Systems. In *Symposium on Network and Distributed System Security (NDSS)*, 2019.

[6] Dana Van Aken, Andrew Pavlo, Geoffrey J. Gordon, and Bohan Zhang. Automatic Database Management System Tuning Through Large-scale Machine Learning. In *Conference on Management of Data (SIGMOD)*, 2017.

[7] David Leroy Alice Coucke, Joseph Dureau and Sébastien Maury. On-device Voice Control on Sonos Speakers. Blog post on Sonos Tech Blog, 2022.

[8] Apple. CSAM Detection. Technical Report.

[9] Giovanni Apruzzese, Hyrum S Anderson, Savino Dambra, David Freeman, Fabio Pierazzi, and Kevin Alejandro Roundy. Position: "Real Attackers Don't Compute Gradients": Bridging the Gap Between Adversarial ML Research and Practice. In *IEEE Conference on Secure and Trustworthy Machine Learning (SaTML)*, 2023.

[10] Thomas Baumhauer, Pascal Schöttle, and Matthias Zeppelzauer. Machine Unlearning: Linear Filtration for Logit-based Classifiers. *Machine Learning*, 2022.

[11] 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 *Machine Learning and Knowledge Discovery in Databases (ECML)*, 2013.

[12] Battista Biggio and Fabio Roli. Wild patterns: Ten years after the rise of adversarial machine learning. *Pattern Recognition*, 2018.

[13] David Blei, Andrew Ng, and Michael Jordan. Latent Dirichlet Allocation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2002.

[14] Lucas Bourtoule, Varun Chandrasekaran, Christopher A. Choquette-Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. Machine Unlearning. In *IEEE Symposium on Security and Privacy (S&P)*, 2021.

[15] Gavin Brown, Mark Bun, Vitaly Feldman, Adam D. Smith, and Kunal Talwar. When is Memorization of Irrelevant Training Data Necessary for High-Accuracy Learning? In *ACM SIGACT Symposium on Theory of Computing (STOC)*, 2021.

[16] Yinzhi Cao and Junfeng Yang. Towards Making Systems Forget with Machine Unlearning. In *IEEE Symposium on Security and Privacy (S&P)*, 2015.

[17] Nicholas Carlini, Anish Athalye, Nicolas Papernot, Wieland Brendel, Jonas Rauber, Dimitris Tsipras, Ian Goodfellow, and Aleksander Madry. On Evaluating Adversarial Robustness. *Computing Research Repository (CoRR)*, abs/1902.06705, 2019.

[18] Nicholas Carlini, Chang Liu, Úlfar Erlingsson, Jernej Kos, and Dawn Song. The Secret Sharer: Evaluating and Testing Unintended Memorization in Neural Networks. In *USENIX Security Symposium*, 2019.

[19] Nicholas Carlini and David Wagner. Adversarial Examples are Not Easily Detected: Bypassing Ten Detection Methods. In *ACM Workshop on Artificial Intelligence and Security (AISec)*, 2017.

[20] Nicholas Carlini and David Wagner. Towards Evaluating the Robustness of Neural Networks. In *IEEE Symposium on Security and Privacy (S&P)*, 2017.

[21] Nicholas Carlini and David Wagner. Audio Adversarial Examples: Targeted Attacks on Speech-to-Text. In *IEEE Deep Learning and Security Workshop (DLS)*, 2018.

[22] Lujing Cen, Ryan Marcus, Hongzi Mao, Justin Gottschlich, Mohammad Alizadeh, and Tim Kraska. Learned Garbage Collection. In *Workshop on Machine Learning and Programming Languages (MAPL)*, 2020.

[23] Laurent Charlin and Richard Zemel. The Toronto Paper Matching System: An Automated Paper-Reviewer Assignment System. In *International Conference on Machine Learning (ICML)*, 2013.

[24] Min Chen, Zhikun Zhang, Tianhao Wang, Michael Backes, Mathias Humbert, and Yang Zhang. Graph Unlearning. In *ACM Conference on Computer and Communications Security (CCS)*, 2021.

[25] Robert S. Chen, Brendan Lucier, Yaron Singer, and Vasilis Syrgkanis. Robust Optimization for Non-Convex Objectives. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[26] Christopher A. Choquette-Choo, Florian Tramèr, Nicholas Carlini, and Nicolas Papernot. Label-Only Membership Inference Attacks. In *International Conference on Machine Learning (ICML)*, 2021.

[27] Jeremy M. Cohen, Elan Rosenfeld, and J. Zico Kolter. Certified Adversarial Robustness via Randomized Smoothing. In *International Conference on Machine Learning (ICML)*, 2019.

[28] William M. Darling. A Theoretical and Practical Implementation Tutorial on Topic Modeling and Gibbs Sampling. In *Annual Meeting of the Assoc. for Computational Linguistics: Human Language Technologies (HLT)*, 2011.

[29] Cynthia Dwork, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Richard S. Zemel. Fairness Through Awareness. In *Innovations in Theoretical Computer Science (ITCS)*, 2012.

[30] Tyna Eloundou, Sam Manning, Pamela Mishkin, and Daniel Rock. GPTs are GPTs: An Early Look at the Labor Market Impact Potential of Large Language Models. *Computing Research Repository (CoRR)*, 2023.

[31] Vitaly Feldman. Does Learning Require Memorization? A Short Tale About a Long Tail. In *ACM SIGACT Symposium on Theory of Computing (STOC)*, 2020.

[32] G David Forney. The Viterbi Algorithm. *Proceedings of the IEEE*, 1973.

[33] Bill Gates. The Age of AI has begun. Blog post on personal blog.

[34] GData. Wir Ändern die Spielregeln. Blog post on company blog.

[35] Amirata Ghorbani and James Zou. Data Shapley: Equitable Valuation of Data for Machine Learning. In *International Conference on Machine Learning (ICML)*, 2019.

[36] Aditya Golatkar, Alessandro Achille, and Stefano Soatto. Eternal Sunshine of the Spotless Net: Selective Forgetting in Deep Networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[37] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT press, 2016.

[38] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and Harnessing Adversarial Examples. In *International Conference on Learning Representations (ICLR)*, 2015.

[39] Google. Our Approach to Facial Recognition. Post on company blog.

[40] Laura Graves, Vineel Nagisetty, and Vijay Ganesh. Amnesiac Machine Learning. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2021.

[41] Kathrin Grosse, Praveen Manoharan, Nicolas Papernot, Michael Backes, and Patrick D. McDaniel. On the (Statistical) Detection of Adversarial Examples. *Computing Research Repository (CoRR)*, 2017.

[42] Jens Groth. On the Size of Pairing-Based Non-interactive Arguments. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT)*, 2016.

[43] Chuan Guo, Tom Goldstein, Awni Hannun, and Laurens Van Der Maaten. Certified Data Removal from Machine Learning Models. In *International Conference on Machine Learning (ICML)*, 2020.

[44] Raia Hadsell, Dushyant Rao, Andrei A Rusu, and Razvan Pascanu. Embracing Change: Continual Learning in Deep Neural Networks. *Trends in Cognitive Sciences*, 2020.

[45] Abdelfatteh Haidine, Fatima Zahra Salmam, Abdelhak Aqqal, and Aziz Dahbi. Artificial Intelligence and Machine Learning in 5G and Beyond: A Survey and Perspectives. *Moving Broadband Mobile Communications Forward: Intelligent Technologies for 5G and Beyond*, 2021.

[46] Moritz Hardt, Eric Price, and Nati Srebro. Equality of Opportunity in Supervised Learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2016.

[47] Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the Knowledge in a Neural Network. *Computing Research Repository (CoRR)*, 2015.

[48] Matthew D. Hoffman, David M. Blei, and Francis R. Bach. Online Learning for Latent Dirichlet Allocation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2010.

[49] Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Logan Engstrom, Brandon Tran, and Aleksander Madry. Adversarial Examples Are Not Bugs, They Are Features. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[50] ISO Central Secretary. Information Technology – Coding of Moving Pictures and Associated Audio for Digital Storage Media at Up to 1.5 Mbits/s – Part3: Audio. Standard 11172-3, International Organization for Standardization, 1993.

[51] Nicholas Jacek and J. Eliot B. Moss. Learning When to Garbage Collect with Random Forests. In *International Symposium on Memory Management (ISMM)*, 2019.

[52] Hengrui Jia, Christopher A. Choquette-Choo, Varun Chandrasekaran, and Nicolas Papernot. Entangled Watermarks as a Defense against Model Extraction. In *USENIX Security Symposium*, 2021.

[53] Konstantinos Kanellis, Ramnatthan Alagappan, and Shivaram Venkataraman. Too Many Knobs to Tune? Towards Faster Database Tuning by Pre-selecting Important Knobs. In *Workshop on Hot Topics in Storage and File Systems (HotStorage)*, 2020.

[54] Kaspersky. Machine Learning in Cybersecurity. Blog post on company blog.

[55] James Kirkpatrick, Razvan Pascanu, Neil C. Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming Catastrophic Forgetting in Neural Networks. *Computing Research Repository (CoRR)*, 2016.

[56] Pang Wei Koh and Percy Liang. Understanding Black-box Predictions via Influence Functions. In *International Conference on Machine Learning (ICML)*.

[57] Sanjay Krishnan, Zongheng Yang, Ken Goldberg, Joseph M. Hellerstein, and Ion Stoica. Learning to Optimize Join Queries With Deep Reinforcement Learning. *Computing Research Repository (CoRR)*, 2018.

[58] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2012.

[59] Maxwell N. Krohn, Alexander Yip, Micah Z. Brodsky, Natan Cliffer, M. Frans Kaashoek, Eddie Kohler, and Robert Tappan Morris. Information Flow Control for Standard OS Abstractions. In *Symposium on Operating Systems Principles (SOSP)*, 2007.

[60] Yann LeCun. Generalization and network design strategies. *Connectionism in Perspective*, 1989.

[61] Mathias Lécuyer, Vaggelis Atlidakis, Roxana Geambasu, Daniel Hsu, and Suman Jana. Certified Robustness to Adversarial Examples with Differential Privacy. In *IEEE Symposium on Security and Privacy (S&P)*, 2019.

[62] Klas Leino and Matt Fredrikson. Stolen Memories: Leveraging Model Memorization for Calibrated White-Box Membership Inference. In *USENIX Security Symposium*, 2020.

[63] Shana Lynch. Andrew Ng: Why AI Is the New Electricity. Article on Stanford Graduate School of Business.

[64] Lin Ma, Dana Van Aken, Ahmed Hefny, Gustavo Mezerhane, Andrew Pavlo, and Geoffrey J. Gordon. Query-based Workload Forecasting for Self-Driving Database Management Systems. In *Conference on Management of Data (SIGMOD)*, 2018.

[65] Martin Maas, David G. Andersen, Michael Isard, Mohammad Mahdi Javanmard, Kathryn S. McKinley, and Colin Raffel. Learning-based Memory Allocation for C++ Server Workloads. In *Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2020.

[66] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards Deep Learning Models Resistant to Adversarial Attacks. In *International Conference on Learning Representations (ICLR)*, 2018.

[67] Pratyush Maini, Mohammad Yaghini, and Nicolas Papernot. Dataset Inference: Ownership Resolution in Machine Learning. In *International Conference on Learning Representations (ICLR)*, 2021.

[68] Ryan Marcus, Parimarjan Negi, Hongzi Mao, Nesime Tatbul, Mohammad Alizadeh, and Tim Kraska. Bao: Making Learned Query Optimization Practical. *SIGMOD Record*, 2022.

[69] Ryan Marcus and Olga Papaemmanouil. Deep Reinforcement Learning for Join Order Enumeration. In *Workshop on Exploiting Artificial Intelligence Techniques for Data Management*, 2018.

[70] Ryan C. Marcus, Parimarjan Negi, Hongzi Mao, Chi Zhang, Mohammad Alizadeh, Tim Kraska, Olga Papaemmanouil, and Nesime Tatbul. Neo: A Learned Query Optimizer. *Very Large Data Base Endowment (VLDB Endowment)*, 2019.

[71] Bernard Marr. The Amazing Ways Tesla Is Using Artificial Intelligence And Big Data. Article on Forbes.com.

[72] Brian B. Monson, Eric J. Hunter, Andrew J. Lotto, and Brad H. Story. The Perceptual Significance of High-frequency Energy in the Human Voice. *Frontiers in Psychology*, 2014.

[73] Andrew C. Myers. JFlow: Practical Mostly-Static Information Flow Control. In Andrew W. Appel and Alex Aiken, editors, *Symposium on Principles of Programming Languages (POPL)*, 1999.

[74] Arvind Narayanan and Vitaly Shmatikov. Robust De-anonymization of Large Sparse Datasets. In *IEEE Symposium on Security and Privacy (S&P)*, 2008.

[75] Seth Neel, Aaron Roth, and Saeed Sharifi-Malvajerdi. Descent-to-Delete: Gradient-Based Methods for Machine Unlearning. In *Algorithmic Learning Theory (ALT)*, 2021.

[76] Palo Alto Networks. What is an ML-Powered NGFW? Blog post on company blog.

[77] Nicolas Papernot, Patrick D. McDaniel, Somesh Jha, Matt Fredrikson, Z. Berkay Celik, and Ananthram Swami. The Limitations of Deep Learning in Adversarial Settings. In *IEEE European Symposium on Security and Privacy (EuroS&P)*, 2016.

[78] Nicolas Papernot, Patrick D. McDaniel, Arunesh Sinha, and Michael P. Wellman. Towards the Science of Security and Privacy in Machine Learning. *Computing Research Repository (CoRR)*, 2016.

[79] Nicolas Papernot, Patrick D. McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a Defense to Adversarial Perturbations Against Deep Neural Networks. In *IEEE Symposium on Security and Privacy (S&P)*, 2016.

[80] Yongjoo Park, Shucheng Zhong, and Barzan Mozafari. QuickSel: Quick Selectivity Learning with Mixture Models. In *Conference on Management of Data (SIGMOD)*, 2020.

[81] Bryan Parno. Autobid. Public GitHub Repository.

[82] Fabio Pierazzi, Feargus Pendlebury, Jacopo Cortellazzi, and Lorenzo Cavallaro. Intriguing Properties of Adversarial ML Attacks in the Problem Space. In *IEEE Symposium on Security and Privacy (S&P)*, 2020.

[83] Erwin Quiring, David Klein, Daniel Arp, Martin Johns, and Konrad Rieck. Adversarial Preprocessing: Understanding and Preventing Image-Scaling Attacks in Machine Learning. In *USENIX Security Symposium*, 2020.

[84] Erwin Quiring, Alwin Maier, and Konrad Rieck. Misleading Authorship Attribution of Source Code using Adversarial Learning. In *USENIX Security Symposium*, 2019.

[85] Adnan Siraj Rakin, Zhezhi He, and Deliang Fan. Bit-Flip Attack: Crushing Neural Network With Progressive Bit Search. In *International Conference on Computer Vision (ICCV)*, 2019.

[86] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning Representations by Back-propagating Errors. *nature*, 1986.

[87] Lea Schönherr, Thorsten Eisenhofer, Steffen Zeiler, Thorsten Holz, and Dorothea Kolossa. Imperio: Robust Over-the-Air Adversarial Examples for Automatic Speech Recognition Systems. In *Annual Computer Security Applications Conference (ACSAC)*, 2020.

[88] Lea Schönherr, Maximilian Golla, Thorsten Eisenhofer, Jan Wiele, Dorothea Kolossa, and Thorsten Holz. Exploring Accidental Triggers of Smart Speakers. *Computer Speech & Language (CSL)*, 2022.

[89] Lea Schönherr, Katharina Kohls, Steffen Zeiler, Thorsten Holz, and Dorothea Kolossa. Adversarial Attacks Against Automatic Speech Recognition Systems via Psychoacoustic Hiding. In *Symposium on Network and Distributed System Security (NDSS)*, 2019.

[90] Roei Schuster, Jin Peng Zhou, Thorsten Eisenhofer, Paul Grubbs, and Nicolas Papernot. Learned Systems Security. *Computing Research Repository (CoRR)*, 2022.

[91] Ayush Sekhari, Jayadev Acharya, Gautam Kamath, and Ananda Theertha Suresh. Remember What You Want to Forget: Algorithms for Machine Unlearning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[92] Ozan Sener and Silvio Savarese. Active Learning for Convolutional Neural Networks: A Core-Set Approach. In *International Conference on Learning Representations (ICLR)*, 2017.

[93] Srinath Setty. Spartan: Efficient and General-Purpose zkSNARKs Without Trusted Setup. In *Annual International Cryptology Conference (CRYPTO)*, 2020.

[94] Shawn Shan, Wenxin Ding, Emily Wenger, Haitao Zheng, and Ben Y. Zhao. Post-breach Recovery: Protection against White-box Adversarial Examples for Leaked DNN Models. In *ACM Conference on Computer and Communications Security (CCS)*, 2022.

[95] Ilia Shumailov, Zakhar Shumaylov, Dmitry Kazhdan, Yiren Zhao, Nicolas Papernot, Murat A. Erdogdu, and Ross J. Anderson. Manipulating SGD with Data Ordering Attacks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[96] Ilia Shumailov, Yiren Zhao, Daniel Bates, Nicolas Papernot, Robert D. Mullins, and Ross Anderson. Sponge examples: Energy-latency attacks on neural networks. In *IEEE European Symposium on Security and Privacy (EuroS&P)*, 2021.

[97] Shuang Song, Kamalika Chaudhuri, and Anand D. Sarwate. Stochastic Gradient Descent with Differentially Private Updates. In *Global Conference on Signal and Information Processing (GlobalSIP)*, 2013.

[98] Sophos. Intercept X: Powered by Deep Learning. Blog post on company blog.

[99] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing Properties of Neural Networks. In *International Conference on Learning Representations (ICLR)*, 2014.

[100] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. DeepFace: Closing the Gap to Human-Level Performance in Face Verification. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[101] Camillo J Taylor. On the Optimal Assignment of Conference Papers to Reviewers. Technical report, 2008.

[102] Apple Computer Vision Machine Learning Team. An On-device Deep Neural Network for Face Detection. Post on company blog.

[103] Anvith Thudi, Gabriel Deza, Varun Chandrasekaran, and Nicolas Papernot. Unrolling SGD: Understanding Factors Influencing Machine Unlearning. In *IEEE European Symposium on Security and Privacy (EuroS&P)*, 2022.

[104] Anvith Thudi, Hengrui Jia, Ilia Shumailov, and Nicolas Papernot. On the Necessity of Auditable Algorithmic Definitions for Machine Unlearning. In *USENIX Security Symposium*, 2022.

[105] Florian Tramèr, Fan Zhang, Ari Juels, Michael K. Reiter, and Thomas Ristenpart. Stealing Machine Learning Models via Prediction APIs. In *USENIX Security Symposium*, 2016.

[106] Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. Robustness May Be at Odds with Accuracy. In *International Conference on Learning Representations (ICLR)*, 2019.

[107] Sakshi Udeshi, Shanshan Peng, Gerald Woo, Lionell Loh, Louth Rawshan, and Sudipta Chattopadhyay. Model Agnostic Defence Against Backdoor Attacks in Machine Learning. *IEEE Transactions on Reliability*, 2022.

[108] Jonathan Uesato, Brendan O'Donoghue, Pushmeet Kohli, and Aäron van den Oord. Adversarial risk and the dangers of evaluating against weak attacks. In *International Conference on Machine Learning (ICML)*, 2018.

[109] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is All you Need. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[110] Alexander Warnecke, Lukas Pirch, Christian Wressnegger, and Konrad Rieck. Machine Unlearning of Features and Labels. In *Symposium on Network and Distributed System Security (NDSS)*, 2023.

[111] David H. Wolpert and William G. Macready. No Free Lunch Theorems for Optimization. *IEEE Transactions on Evolutionary Computation*, 1997.

[112] Eric Wong and J. Zico Kolter. Provable Defenses against Adversarial Examples via the Convex Outer Adversarial Polytope. In *International Conference on Machine Learning (ICML)*, 2018.

[113] Eric Wong, Leslie Rice, and J. Zico Kolter. Fast is Better Than Free: Revisiting Adversarial Training. In *International Conference on Learning Representations (ICLR)*, 2020.

[114] Yinjun Wu, Edgar Dobriban, and Susan B. Davidson. DeltaGrad: Rapid Retraining of Machine Learning Models. In *International Conference on Machine Learning (ICML)*, 2020.

[115] Wayne Xiong, Jasha Droppo, Xuedong Huang, Frank Seide, Mike Seltzer, Andreas Stolcke, Dong Yu, and Geoffrey Zweig. Achieving Human Parity in Conversational Speech Recognition. *Computing Research Repository (CoRR)*, 2016.

[116] Richard S. Zemel, Yu Wu, Kevin Swersky, Toniann Pitassi, and Cynthia Dwork. Learning Fair Representations. In *International Conference on Machine Learning (ICML)*, 2013.

[117] Dongxu Zhang, Tianyi Luo, and Dong Wang. Learning from LDA Using Deep Neural Networks. In *Conference on Natural Language Processing and Chinese Computing, (NLPCC)*, 2016.

[118] Guoming Zhang, Chen Yan, Xiaoyu Ji, Tianchen Zhang, Taimin Zhang, and Wenyuan Xu. DolphinAttack: Inaudible Voice Commands. In *ACM Conference on Computer and Communications Security (CCS)*, 2017.

[119] Qi Zhou, Haipeng Chen, Yitao Zheng, and Zhen Wang. EvaLDA: Efficient Evasion Attacks Towards Latent Dirichlet Allocation. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2021.

[120] Eberhard Zwicker and Hugo Fastl. *Psychoacoustics: Facts and Models*. 2013.

Part II

—

Publications

List of Publications

Below is the list of papers in this thesis which are also included in Appendices A to C. A list of additional contributions is given on the next page.

Publications in this Thesis

[1] Thorsten Eisenhofer, Erwin Quiring, Jonas Möller, Doreen Riepel, Thorsten Holz, and Konrad Rieck. No more Reviewer #2: Subverting Automatic Paper-Reviewer Assignment using Adversarial Learning. In *USENIX Security Symposium*, 2023.

[2] Thorsten Eisenhofer, Lea Schönherr, Joel Frank, Lars Speckemeier, Dorothea Kolossa, and Thorsten Holz. Dompteur: Taming Audio Adversarial Examples. In *USENIX Security Symposium*, 2021.

[3] Thorsten Eisenhofer, Doreen Riepel, Varun Chandrasekaran, Esha Ghosh, Olga Ohrimenko, and Nicolas Papernot. Verifiable and Provably Secure Machine Unlearning. *Computing Research Repository (CoRR)*, 2022.

Other Contributions

[1] Joel Frank, Franziska Herbert, Jonas Ricker, Lea Schönherr, Thorsten Eisenhofer, Asja Fischer, Markus Dürmuth, and Thorsten Holz. "Most Seemed Real": A Representative Study on Human Detection of Deepfakes Across Media and Countries. *Manuscript in Submission*, 2023.

[2] David Pape, Sina Däubener, Thorsten Eisenhofer, Antonio Cina, and Lea Schönherr. Stealing with Uncertainty Quantification Models. *Manuscript in Submission*, 2023.

[3] Hojjat Aghakhani, Thorsten Eisenhofer, Lea Schönherr, Dorothea Kolossa, Thorsten Holz, Christopher Kruegel, and Giovanni Vigna. VenoMave: Targeted Poisoning Against Speech Recognition. In *Secure and Trustworthy Machine Learning (SaTML)*, 2023.

[4] Nico Schiller, Merlin Chlosta, Moritz Schloegel, Nils Bars, Thorsten Eisenhofer, Tobias Scharnowski, Felix Domke, Lea Schönherr, and Thorsten Holz. Drone Security and the Mysterious Case of DJI's DroneID. In *Network and Distributed System Security Symposium (NDSS)*, 2023.

[5] Roei Schuster, Jin Peng Zhou, Thorsten Eisenhofer, Paul Grubbs, and Nicolas Papernot. Learned Systems Security. *Computing Research Repository (CoRR)*, 2023.

[6] Michel Abdalla, Thorsten Eisenhofer, Eike Kiltz, Sabrina Kunzweiler, and Doreen Riepel. Password-Authenticated Key Exchange from Group Actions. In *Annual International Cryptology Conference (CRYPTO)*, 2022.

[7] Lea Schönherr, Maximilian Golla, Thorsten Eisenhofer, Jan Wiele, Dorothea Kolossa, and Thorsten Holz. Exploring Accidental Triggers of Smart Speakers. *Computer Speech & Language (CSL)*, 2022.

[8] Joel Frank, Thorsten Eisenhofer, Lea Schönherr, Asja Fischer, Dorothea Kolossa, and Thorsten Holz. Leveraging Frequency Analysis for Deep Fake Image Recognition. In *International Conference on Machine Learning (ICML)*, 2020.

[9] Lea Schönherr, Thorsten Eisenhofer, Steffen Zeiler, Thorsten Holz, and Dorothea Kolossa. Imperio: Robust Over-the-Air Adversarial Examples for Automatic Speech Recognition Systems. In *Annual Computer Security Applications Conference (ACSAC)*, 2020.

No more Reviewer #2: Subverting Automatic Paper-Reviewer Assignment using Adversarial Learning

Publication Data

Thorsten Eisenhofer, Erwin Quiring, Jonas Möller, Doreen Riepel, Thorsten Holz, and Konrad Rieck. No more Reviewer #2: Subverting Automatic Paper-Reviewer Assignment using Adversarial Learning. In *USENIX Security Symposium*, 2023.

No more Reviewer #2: Subverting Automatic Paper-Reviewer Assignment using Adversarial Learning

Thorsten Eisenhofer[1,*], Erwin Quiring[1,2,*], Jonas Möller[3], Doreen Riepel[1], Thorsten Holz[4], Konrad Rieck[3]

[1] Ruhr University Bochum
[2] International Computer Science Institute (ICSI) Berkeley
[3] Technische Universität Berlin
[4] CISPA Helmholtz Center for Information Security

Abstract

The number of papers submitted to academic conferences is steadily rising in many scientific disciplines. To handle this growth, systems for automatic *paper-reviewer assignments* are increasingly used during the reviewing process. These systems use statistical topic models to characterize the content of submissions and automate the assignment to reviewers. In this paper, we show that this automation can be manipulated using adversarial learning. We propose an attack that adapts a given paper so that it misleads the assignment and selects its own reviewers. Our attack is based on a novel optimization strategy that alternates between the feature space and problem space to realize unobtrusive changes to the paper. To evaluate the feasibility of our attack, we simulate the paper-reviewer assignment of an actual security conference (IEEE S&P) with 165 reviewers on the program committee. Our results show that we can successfully select and remove reviewers without access to the assignment system. Moreover, we demonstrate that the manipulated papers remain plausible and are often indistinguishable from benign submissions.

[*] Shared first authorship

1 Introduction

Peer review is a major pillar of academic research and the scientific publication process. Despite its well-known weaknesses, it is still an essential instrument for ensuring high-quality standards through the independent evaluation of scientific findings [1, 2, 3]. For this evaluation, a *submission* is assigned to a group of reviewers, taking into account their expertise, preferences, and potential biases. For conferences, this assignment is traditionally carried out by a program chair, while for journals, the task is performed by an editor. This mechanism has proven effective in the past, but is becoming increasingly difficult to realize as research communities grow. For example, the number of papers submitted to top-tier security conferences is increasing exponentially, reaching over 3,000 submissions in 2020. Likewise, the number of reviewers continuously grows for all major security conferences [4].

To handle this growth, conference management tools have become indispensable in peer review. They allow reviewers to bid for submissions and support the program chair to find a good assignment based on a best-effort matching. Unfortunately, even these tools reach their limit when the number of submissions continues to grow and manual bidding becomes intractable, as for example, in the area of machine learning. Major conferences in this area regularly have over 10,000 submissions that need to be distributed among more than 7,000 reviewers [5]. For this reason, conference management tools are increasingly extended with automatic systems for *paper-reviewer assignment* [6, 7]. These systems use topic models from machine learning to assess reviewer expertise, filter submissions, and automate the assignment process.

In this work, we show that this automation can be exploited to manipulate the assignment of reviewers. In contrast to prior work that focused on bid manipulations and reviewer collusion [8, 9], our attack rests on adversarial learning. In particular, we propose an attack that adapts a given paper so that it misleads the underlying topic model. This enables us to reject and select specific reviewers from the program committee. To reach this goal, we introduce a novel optimization strategy that alternates between the feature space and problem space when adapting a paper. This optimization allows us to preserve the semantics and plausibility of the document, while carefully changing the assignment of reviewers.

Our attack consists of two alternating steps: First, we aim at misleading the topic model employed in current assignment systems [7, 6]. This model defines a latent

topic space that is difficult to attack because neither gradients nor an explicit decision boundary exist. To address this problem, we develop a search algorithm for exploring the latent space and manipulating decisions in it. As a counterpart, we introduce a framework for modifying papers in the problem space. This framework provides several transformations for adapting the paper's content, ranging from invisible comments to synonym replacement and generated text. These transformations enable us to preserve the paper's semantics, while gradually changing the assignment of reviewers.

To empirically evaluate the practical feasibility of our attack, we simulate the paper-reviewer assignment of the 43rd IEEE Symposium on Security and Privacy (IEEE S&P) with the original program committee of 165 reviewers in both a *black-box* and a *white-box* threat scenario. As the basis for our attacks, we consider 32 original submissions that are publicly available with LaTeX source code.

Our white-box adversary achieves an alarming performance: we can successfully remove *any* of the initially assigned reviewers from a submission, and even scale the attack to completely choose *all* reviewers in the automated assignment process. In the black-box scenario, we can craft adversarial papers that transfer to an unknown target system by only using public knowledge about a conference. We achieve a success rate of up to 90% to select a reviewer and 81% to reject one. Furthermore, we demonstrate that the attack remains robust against variations in the training data.

Our work points to a serious problem in the current peer review process: With the application of machine learning, the process inherits vulnerabilities and becomes susceptible to new forms of manipulation. We discuss potential defenses: (1) For the feature space, robust topic modeling may limit the attacker's capabilities and (2) for the problem space, we recommend using optical character recognition (OCR) techniques to retrieve the displayed text. Nevertheless, these safeguards cannot completely fend off our manipulations and reviewers should be made aware of this threat.

Contributions. We make the following key contributions:

- *Attack against topic models.* We introduce a novel attack against topic models suitable for manipulating the ranking of reviewers. The attack does not depend on the availability of gradients and explores the latent topic space through an efficient beam search.

- *Problem-space transformations.* Our attack ensures that both the semantics and plausibility of the generated adversarial papers are preserved. This goal is achieved

by a variety of transformations that carefully manipulate the document format and text of a submission.

- *Adversarial papers.* We present a method for constructing adversarial papers in a black-box and white-box scenario, unveiling a serious problem in automatic reviewer assignment. The attack rests on a novel hybrid approach to construct adversarial examples in discrete domains

Examples of the created adversarial papers are provided at https://github.com/rub-syssec/adversarial-papers. We also make our code and artifacts publicly available here.

2 Technical Background

Let us start by reviewing the necessary background for the design of our attack, covering the process of paper-review assignment and the underlying topic modeling.

Systems for paper-reviewer assignment. To cope with the abundance of submissions, several concepts have been proposed to assign reviewers to submissions [e.g., 10, 11, 12, 13]. In practice, the most widely used concept is *The Toronto Paper Matching System* (TPMS) by Charlin and Zemel [7]. Because of its high-quality assignments and direct integration with Microsoft's conference management tool CMT [14], TPMS is used by numerous conferences in different fields, including ACM CCS in 2017–2019 and NeurIPS/ICML. TPMS can be considered the de facto standard for automatic matching of papers to reviewers. Unfortunately, the implementation of TPMS is not publicly available and thus we focus in this work on *Autobid* [6], an open-source realization of the TPMS concept. Autobid closely follows the process described by Charlin and Zemel [7]. The system has been designed to work alongside HotCRP [15] and was used to support reviewer assignment at the IEEE Symposium on Security and Privacy (S&P) in 2017 and 2018.

Technically, TPMS and Autobid implement a processing pipeline similar to most matching concepts: (a) the text from the submission document is extracted and cleansed using natural language processing, (b) the preprocessed text is then mapped to the latent space of a topic model, and finally (c) an assignment is determined by deriving a ranking of reviewers. In the following, we review these steps in detail.

(a) Text preprocessing. When working with natural languages, multiple steps are required to bring text into a form suitable for machine learning (see Figure 1). As paper

submissions can be provided in different formats, the pipeline starts by extracting text from the underlying document, typically the PDF format. This original document resides in the problem space of our attack and is denoted as z in the following. Autobid employs the tool `pdftotext` for this task, which is used in our evaluation in Section 4. The extracted text is then normalized using a preprocessor function ρ. Typically, it is tokenized, converted to lowercase, and stemmed [16]. Subsequently, stop words are removed so that each submission is now represented as a sequence of filtered stems. Autobid employs the NLTK package [17] to perform this task.

Finally, a feature extractor Φ maps the input $\rho(z)$ to a bag-of-words vector $\mathbf{x} \in \mathbb{N}^{|\mathcal{V}|}$ with $\mathcal{V}$ being the vocabulary formed over all words (stems). That is, a submission is represented by a high-dimensional vector whose individual dimensions reflect the count of words. Although this representation is frequently applied in supervised learning, the high dimensionality is problematic for unsupervised learning and complicates determining topics in the submission.

(b) Topic modeling. The key to matching reviewers to papers is the automatic identification of topics in the text. This unsupervised learning task is denoted as *topic modeling*. While there exist several algorithms for this modeling, many assignment systems, including TPMS and Autobid, use *Latent Dirichlet Allocation* (LDA). LDA is a Bayesian probabilistic method for topic modeling that allows representing a document as a low-dimension mixture of latent topics. Formally, we define this representation as a function

$$\Gamma \colon \mathbb{N}^{|\mathcal{V}|} \longrightarrow \mathcal{T}, \quad \mathbf{x} \mapsto \theta_{\mathbf{x}}$$

mapping a bag-of-words vector $\mathbf{x}$ to a low-dimensional vector space $\mathcal{T}$, whose dimensions reflect different topics.

Generally, LDA is modeled as a generative probabilistic process [18, 19, 20]. It assumes a corpus D of documents and models each document as a random mixture over

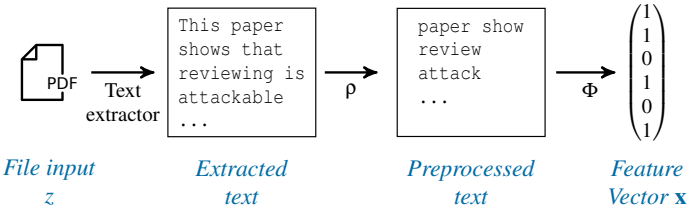


Figure 1: Text preprocessing in paper-reviewer assignment.

a set of latent topics T. A topic t is characterized by a multinomial distribution over the vocabulary $\mathcal{V}$, and drawn from a Dirichlet distribution $\phi_t \sim \mathit{Dirichlet}(\beta)$ with the prior β. The Dirichlet prior is usually sparse (i.e., $\beta < 1$) to model that words are not uniformly assigned to topics. Given these topics, for each document $\mathbf{x} \in D$, a distribution of topics $\theta_\mathbf{x} \sim \mathit{Dirichlet}(\alpha)$ is drawn. Again, the prior α is sparse to account for that documents are usually only associated with a small number of topics. Finally, for each word $w_i \in \mathbf{x}$, a topic $t_i \sim \mathit{Multinom}(\theta_\mathbf{x})$ is selected and the observed word $w_i \sim \mathit{Multinom}(\phi_{t_i})$ is drawn. This process can be summarized by the joint probability

$$P(\mathbf{w}, \mathbf{t}, \theta, \phi | \alpha, \beta) = P(\phi|\beta)P(\theta|\alpha)P(\mathbf{t}|\theta)P(\mathbf{w}|\phi_\mathbf{t}) \tag{1}$$

with $\mathbf{w} = (w_1, \ldots, w_{|\mathbf{x}|})$ and $\mathbf{t} = (t_1, \ldots, t_{|\mathbf{x}|})$.

To create a topic model in practice, we need to reverse this process and learn the posterior distribution of the latent variables $\mathbf{t}$, θ, and ϕ given the *observed* documents D. Specifically, we need to solve

$$P(\theta, \phi, \mathbf{t} | \mathbf{w}, \alpha, \beta) = \frac{P(\theta, \phi, \mathbf{t}, \mathbf{w} | \alpha, \beta)}{P(\mathbf{w}|\alpha, \beta)}. \tag{2}$$

Solving this equation is intractable as the term $P(\mathbf{w}|\alpha, \beta)$ cannot be computed exactly [18]. To address this, different approximated techniques, such as variational inference [18, 19] or Gibbs Sampling [20], are typically used for implementations of LDA. Autobid builds on variational inference based on the implementation of GenSim [21].

For the feature vector $\mathbf{x}$ of a new submission, the same technique—conditioned on the corpus D—is used to compute the corresponding topic mixture $\theta_\mathbf{x}$. Attacking this process is challenging, as no gradients or other guides for moving in the direction of particular topics are directly available. Hence, we develop a new search algorithm for subverting the topic assignment of LDA in Section 3.1.

(c) Paper-reviewer assignment. Finally, the topic model is used to estimate the reviewer expertise and automate the matching of submissions to reviewers. More specifically, let $\mathcal{R}$ be the set of all potential reviewers and $\mathcal{S}$ a set of submissions $\mathbf{x} \in \mathbb{N}^{|\mathcal{V}|}$. For each reviewer r, we collect an archive $A_r \in \mathbb{N}^{|\mathcal{V}|}$ representative of the reviewer's expertise and interests. Since researchers are naturally described best by their works, this could, for example, be a selected set of previously published papers. The corresponding archives are modeled as a union over all papers.

For each pair of reviewer r and submission $\mathbf{x}$, a *bidding score* $b_{r,\mathbf{x}}$ is calculated. This score reflects the similarity between the reviewer's archive A_r and a submission $\mathbf{x}$: the

more similar, the higher the score. Given the topic extractor $\Gamma(\cdot)$, a reviewer r and a submission $\mathbf{x}$, Autobid defines the bidding score as the following dot-product

$$b_{r,\mathbf{x}} \coloneqq \Gamma(A_r) \cdot \Gamma(\mathbf{x})^\top. \tag{3}$$

Subsequently, these bidding scores are used for the final assignment $\mathsf{A} \in \{0,1\}^{|\mathcal{R}|\times|\mathcal{S}|}$ with the goal to maximize the similarity between reviewers and submissions. In this phase, additional constraints are included: the assignment is subjected to (1) the targeted number of reviewers $\mathsf{L}_\mathbf{x}$ assigned to a submission and (2) the maximum number of submissions L_r assigned to a reviewer. More formally, we can describe the assignment as the following bipartite matching problem:

$$\begin{aligned}
\underset{\mathsf{A}}{\text{maximize}} \quad & \sum_r \sum_\mathbf{x} b_{r,\mathbf{x}} \cdot \mathsf{A}_{r,\mathbf{x}} \\
\text{subject to} \quad & \mathsf{A}_{r,\mathbf{x}} \in \{0,1\}\ \forall r, \mathbf{x} \\
& \sum_r \mathsf{A}_{r,\mathbf{x}} \leq \mathsf{L}_\mathbf{x}\ \forall \mathbf{x} \\
& \sum_\mathbf{x} \mathsf{A}_{r,\mathbf{x}} \leq \mathsf{L}_r\ \forall r
\end{aligned}$$

This optimization problem can then be reformulated and efficiently solved with *Linear Programming (LP)* [22].

3 Adversarial Papers

We proceed to introduce our approach for subverting the paper-reviewer assignments. To this end, we first define a threat model for our attack, and then examine challenges and required steps to control the matching.

Threat model. We consider a scenario where the adversary only modifies her submission—the *adversarial paper*—to manipulate the assigned reviewers. We assume two representative classes of adversaries with varying degrees of knowledge. First, we focus on *white-box adversaries* with complete access to the assignment system, including the trained model and reviewer archives. This represents a very strong class of adversaries and allows us to generally study the strength as well as limitations of our attack against assignment systems. Second, we study the more realistic scenario with a *black-box adversary*. The adversary is assumed to have only a general knowledge about the assignment system (i. e., AutoBid is an open-source project [6]). No access to the training data

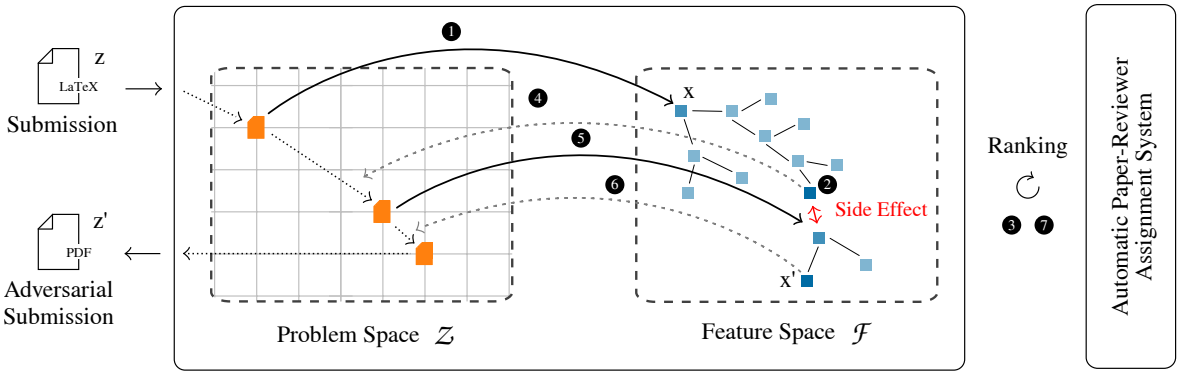


Figure 2: **Feature-problem-space attack.** For a submission z, we construct an adversarial submission z' that leads to a targeted assignment of reviewers. Our attack alternately switches between $\mathcal{Z}$ and $\mathcal{F}$. In step ❶, we extract word counts $\mathbf{x}$ from submission z, and use a search algorithm to change $\mathbf{x}$ in $\mathcal{F}$ to obtain the desired ranking (step ❷). To guide this search, we query the paper-reviewer assignment system for scores (step ❸). Next, we realize the modifications in the problem space $\mathcal{Z}$ and manipulate z (step ❹). Projecting the resulting submission back to $\mathcal{F}$, the submission vector will be shifted due to side effects and transformation limitations (step ❺). This shift is considered by continuing the search process from this new position and repeating this process iteratively (step ❻) until we obtain a valid adversarial submission z' (step ❼).

and learned model is given. In this setting, we envision adversaries that exploit public information about a conference, such as knowledge about the program committee.

Challenges. The adversary has to operate both in the problem space $\mathcal{Z}$ and the feature space $\mathcal{F}$. The former consists of the input objects (e. g., the LaTeX source files of the paper); the latter contains the feature vectors that are used as inputs for the learning system. In contrast to domains like image recognition, the mapping from the problem space to the feature space is not bijective, i. e., there is no one-to-one correspondence between $\mathcal{Z}$ and $\mathcal{F}$. This poses a challenge for the adversary because a promising feature vector may not be mapped to a valid submission. A further obstacle is that some modifications in the problem space cannot be applied without side effects: If an adversary, for instance, adds a sentence to include a particular word, she inevitably adds other words that change the feature vector.

To deal with these challenges, we introduce a *hybrid* optimization strategy that alternates between the feature-space and problem-space representations of the attack submission. This optimization enables us to preserve the semantics and plausibility of

the document, while at the same time gradually changing the assignment of reviewers. A general overview of our attack is outlined in Figure 2. Second, we transfer problem-space restrictions to the feature space. In this way, we resolve restrictions in a generic manner without adjusting our problem-space transformations.

Attack goals. Given a submission z, our goal is to find an adversarial paper z' that leads to the adversary's targeted review assignment. In the feature space, we thus want to manipulate the set of assigned reviewers $R_{\mathbf{x}}$. That is, we want to *select* and *reject* arbitrary reviewers to be included in respectively excluded from $R_{\mathbf{x}}$. Formally, we define two sets R_{sel} and R_{rej} and our goal is to find a vector $\delta \in \mathcal{F}$ such that the modified word counts $\mathbf{x}' \coloneqq \mathbf{x} + \delta$ fulfill

$$\begin{aligned} &r \in R_{\text{sel}} \Rightarrow r \in R_{\mathbf{x}'}, \text{ and} \\ &r \in R_{\text{rej}} \Rightarrow r \notin R_{\mathbf{x}'}, \forall r \in \mathcal{R}. \end{aligned} \tag{4}$$

We require every reviewer $r \in R_{\text{sel}}$ is included in $R_{\mathbf{x}'}$ and likewise every reviewer $r \in R_{\text{rej}}$ excluded from $R_{\mathbf{x}'}$. In addition, we take care that the targeted solution is feasible with $|R_{\text{sel}}| \leq \mathsf{L}_{\mathbf{x}}$ and $|R_{\text{rej}}| \leq |\mathcal{R}| - \mathsf{L}_{\mathbf{x}}$.

Furthermore, we restrict the modifications to $||\delta||_1 \leq L_1^{\max}$ and $||\delta||_\infty \leq L_\infty^{\max}$. The L_1 constraint limits the amount of modifications to the submissions and makes the attack less suspicious. Similarly, the L_∞ constraint restricts the maximum change to a single feature, so that a word is not included too frequently. Finally, with respect to the concrete assignment process, we assume an automatic matching that always selects the reviewers with the highest assignment scores. We note that this assumption can be relaxed, as shown by Jecmen et al. [8], and combined with colluding reviewers. We further discuss the impact of concurring submissions in Section 7.

For manipulations in the problem space, we design various transformations for adapting the submission z. We denote a single transformation by $\omega : \mathcal{Z} \longrightarrow \mathcal{Z},\ z \mapsto z'$, where multiple transformations can be chained together as $\Omega = \omega_k \circ \cdots \circ \omega_2 \circ \omega_1$. To avoid transformations from creating artifacts and visible clues, we introduce the following problem-space constraints: First, we need to *preserve the semantics* of the text, so that the paper is still a meaningful submission. Second, we add a *plausibility* constraint, that is, the modifications should be as inconspicuous as possible. We summarize the constraints as Υ and write $\Omega(z) \models \Upsilon$ if a transformation sequence Ω on a submission fulfills these constraints.

Optimization problem. We arrive at the following optimization problem for generating adversarial examples, integrating constraints from the problem space and the feature space:

$$\begin{aligned} & r \in R_{\text{sel}} \Rightarrow r \in R_{\mathbf{x}'}, \text{ and} \\ & r \in R_{\text{rej}} \Rightarrow r \notin R_{\mathbf{x}'}, \forall r \in \mathcal{R} \\ \text{subject to} \quad & ||\delta||_1 \leq L_1^{\max} \text{ and } ||\delta||_\infty \leq L_\infty^{\max} \\ & \Omega(z) \models \Upsilon \end{aligned} \tag{5}$$

with $\mathbf{x} = \Phi \circ \rho(z)$, $\mathbf{x}' = \Phi \circ \rho(\Omega(z))$, and $\delta = (\mathbf{x}' - \mathbf{x})$. We proceed to realize this optimization strategy by first introducing our attack in the feature space and then in the problem space, before merging both components.

3.1 Feature Space

In an automatic paper-reviewer assignment system, the set of reviewers $R_{\mathbf{x}}$ for a submission is determined by the computed assignment scores $b_{r,\mathbf{x}}$ between reviewers r (characterized by their archives A_r) and the submission vector $\mathbf{x}$:

$$b_{r,\mathbf{x}} \coloneqq \Gamma(A_r) \cdot \Gamma(\mathbf{x})^\top \tag{6}$$

To change the assignment score and thus affect the matching, we can only influence the extracted high-level features $\Gamma(\mathbf{x})$ since A_r is fixed for a given set of $\mathcal{R}$. However, even when we have full control over $\Gamma(\mathbf{x})$, changing the relative ordering—the *ranking*—between reviewers is not straightforward. For instance, suppose we have two reviewers r_1 and r_2 that share most topics (i.e., $\Gamma(A_{r_1}) \approx \Gamma(A_{r_2})$), adjusting $\Gamma(\mathbf{x})$ in this case will have a similar effect on both. In particular, if we naïvely try to increase the assignment score from r_1, we simultaneously also increase the score of r_2 and vice versa. Even if reviewers are not working in the same area, their topic distributions often partially overlap, as their research builds on similar principles and concepts. Hence, to modify the ranking we need to carefully maneuver the submission in the feature space. This is significantly more challenging compared to attacking a classification, as we need to both attack the model's prediction while simultaneously considering effects on concurring reviewers.

Our attack is further complicated by the fact that altering the topic distribution itself is a challenging task, since we need to make changes in the latent topic space. For LDA,

this distribution $\Gamma(\mathbf{x}) = \theta_{\mathbf{x}}$ is computed using a probabilistic inference procedure. Thus, typical gradient-style attacks are not applicable. Indeed, Zhou et al. [23] even show that the manipulation of this inference is *NP-hard*. Moreover, LDA typically assigns only a small weight to individual words, so an attacker is required to manipulate a comparatively large set of words for subverting the topic assignment.

To address both of these challenges, we use a stochastic beam search. For a given submission vector $\mathbf{x}$, we start with an empty modification vector δ which is extended in each iteration until we find a successful submission vector $\mathbf{x}' \coloneqq \mathbf{x} + \delta$ or a maximum number of iteration I is reached. During this search, we consider B candidate vectors in parallel and select successors after each iteration with a probability increasing as a function of the candidates' loss.

Loss. For our search, we define the following loss function to evaluate the quality of a submission $\mathbf{x}$ in terms of the objective from Equation 4 that incorporates the selection and rejection of reviewers:

$$\ell \coloneqq \ell_{\text{sel}} + \ell_{\text{rej}} \tag{7}$$

For selected reviewers, the loss ℓ_{sel} is reduced when the assignment scores $b_{\hat{r},\mathbf{x}}$ increase or when the ranks of the reviewers improve (i. e., when reviewers ascend in the ranking):

$$\ell_{\text{sel}} \coloneqq \sum_{\hat{r} \in R_{\text{sel}}} \text{rank}_{\mathbf{x}}^{\hat{r}} \cdot (1 - b_{\hat{r},\mathbf{x}}) \tag{8}$$

where $\text{rank}_{\mathbf{x}}^{\hat{r}}$ is the rank of reviewer $\hat{r}$ for submission $\mathbf{x}$. Similarly, for rejected reviewers the loss ℓ_{rej} is reduced when the assignment scores $b_{\check{r},\mathbf{x}}$ decrease:

$$\ell_{\text{rej}} \coloneqq \sum_{\check{r} \in R_{\text{rej}}} \max(\text{rank}_{\mathbf{x}}^{\text{rej}} - \text{rank}_{\mathbf{x}}^{\check{r}}, 0) \cdot (b_{\check{r},\mathbf{x}} - b_{r_{\text{rej}},\mathbf{x}}) \tag{9}$$

where $\text{rank}_{\mathbf{x}}^{\text{rej}}$ is the target rank for a rejected reviewer (i. e., rejected reviewer are pushed down towards this rank) and $b_{r_{\text{rej}},\mathbf{x}}$ is the corresponding assignment score. This loss is designed to focus on reviewers that are far off, but simultaneously allows reviewers to provide "room" for following reviewers, for example, when we want to move a group of reviewers upwards/downwards in the ranking.

We consider a submission vector $\mathbf{x}$ successful when the objective from Equation 4 is fulfilled. At this point, we are naturally just at the boundary of the underlying decision function. To make the submission vector more *resilient*, we could continue to decrease the loss. However, since we already successfully ordered the reviewer (i. e., the ranking),

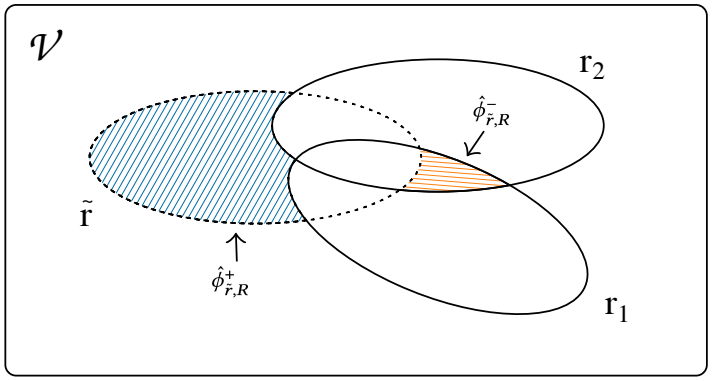


Figure 3: **Reviewer words.** Based on the topic model, we associate reviewer with a set of predictive words. Given target reviewer $\tilde{r}$ and a set of concurring reviewers $R = \{r_1, r_2\}$, we construct distributions $\hat{\phi}^{+}_{\tilde{r},R}$ and $\hat{\phi}^{-}_{\tilde{r},R}$. Sampling from these distributions yield words that are only predictive for reviewer $\tilde{r}$ respectively reviewers in R.

we are more interested in maximizing the *margin* of selected and rejected reviewers to the border of $R_{\mathbf{x}}$. We denote this margin as Φ and set $\ell := -\Phi$ whenever $\mathbf{x}$ satisfies Equation 4. Decreasing the loss is then equivalent to maximizing Φ.

Candidate generation. A key operation of the beam search is the generation of new candidate vectors. We create a successor from a given submission by adding (respectively removing) k words to adjust topic distribution $\theta_{\mathbf{x}} = \Gamma(\mathbf{x})$ and ultimately the ranking of submission $\mathbf{x}$. To select words, we represent (broadly speaking) each reviewer by a set of *predictive* words and sample words that lie in the disjunction between a target and its concurring reviewers. An example of this is shown in Figure 3.

To construct these sets, we first represent each reviewer r by a *reviewer to words* distribution $\hat{\phi}_r$ over vocabulary $\mathcal{V}$. Intuitively, this distribution assigns each word the probability how predictive it is for r. Formally, we define the probability mass function for $\hat{\phi}_r$ as follows:

$$Q_r \colon \mathcal{V} \to \mathbb{R}, \quad w \mapsto \frac{\frac{1}{|T|} \sum_{t \in T} P(w \mid t)\ P(t \mid r)}{\sum_{w \in \mathcal{V}} \frac{1}{|T|} \sum_{t \in T} P(w \mid t)\ P(t \mid r)}$$

Remember that each topic t defines a distribution over $\mathcal{V}$ and each reviewer can be represented by $\Gamma(A_r)$. Q_r assigns each word the average probability over all topics T scaled by the relevance of topic t for reviewer r. Randomly sampling from $\hat{\phi}_r$ thus yield words with a probability given as a function of their *predictiveness* for r. In practice, $\mathcal{V}$ is typically large and most words are assigned with an insignificant probability. To

improve performance, we therefore restrict $\hat{\phi}_r$ to the ν words with highest probability. We rescale the mass function to sum up to 1 so that $\hat{\phi}$ forms a valid distribution.

To select r, we could now simply add predictive words sampled from this distribution. However, as described earlier, naively doing this will likely have unwanted side effects because of concurring reviewers. To account for this, we further refine this distribution and simultaneously consider multiple reviewers. Let $\tilde{r}$ be a targeted reviewer and R a set of concurring reviewers. We want to restrict $\hat{\phi}_{\tilde{r}}$ to only include words that are predictive for $\tilde{r}$ but not for any reviewer in R. Specifically, we define $\hat{\phi}^{+}_{\tilde{r},R}$ with

$$Q^{+}_{\tilde{r},R} \colon \mathcal{V} \to \mathbb{R}, \quad w \mapsto \begin{cases} Q_{\tilde{r}}(w) & \text{if } Q_{\tilde{r}}(w) \neq 0 \wedge \\ & \quad \forall r \in R : Q_r(w) = 0 \\ 0 & \text{otherwise} \end{cases}$$

Subsequently, to form a valid probability mass function, we rescale $Q^{+}_{\tilde{r},R}$ to sum up to 1. Note for $R = \emptyset$ it follows $\hat{\phi}^{+}_{\tilde{r},R} = \hat{\phi}_{\tilde{r}}$. Sampling from $\hat{\phi}^{+}_{\tilde{r},R}$ only yields words that are predictive for $\tilde{r}$ but not R. Often we are also interested in the opposite case, i.e., words that are predictive for all reviewer in R but not for $\tilde{r}$ (e.g., when we want to remove words to promote $\tilde{r}$ in the ranking). Analogous, we define $\hat{\phi}^{-}_{\tilde{r},R}$ and write

$$Q^{-}_{\tilde{r},R} \colon \mathcal{V} \to \mathbb{R}, \quad w \mapsto \begin{cases} \frac{1}{|R|} \sum_{r \in R} Q_r(w) & \text{if } Q_{\tilde{r}}(w) = 0 \wedge \\ & \quad \forall r \in R : Q_r(w) \neq 0 \\ 0 & \text{otherwise} \end{cases}$$

Again, we rescale $Q^{-}_{\tilde{r},R}$ to sum up to 1. For $R = \emptyset$, the distribution $\hat{\phi}^{-}_{\tilde{r},R}$ is not well defined, as its mass function always evaluates to 0 and we thus set $\hat{\phi}^{-}_{\tilde{r},R} := \hat{\phi}_{\tilde{r}}$. Figure 3 graphically depict this construction. For reviewer selection, we consider sets of concurring reviewer R that are close to r in the ranking. Specifically, we randomly sample M subsets from

$$R \subseteq Pow(\{r \mid \forall r \neq \tilde{r} \in \mathcal{R} : 0 \leq \text{rank}_{\tilde{r}} - \text{rank}_r - \upsilon \leq \omega\})$$

for a given reviewer window ω with offset υ. In other words, we exploit locality and focus on reviewer that are either before or close behind r in the ranking. For each subset, we create two candidates by (1) adding k words from $\hat{\phi}^{+}_{\tilde{r},R}$ respectively (2) remove k words from $\hat{\phi}^{-}_{\tilde{r},R}$. Reviewer rejection follows analogous with the distributions interchanged and sets sampled from

$$R \subseteq Pow(\{r \mid \forall r \neq \tilde{r} \in \mathcal{R} : -\omega \leq \text{rank}_{\tilde{r}} - \text{rank}_r + \upsilon \leq 0\})$$

Table 1: **Problem-space transformations.** Overview of transformations to realize modifications in the problem space. They are grouped by deniability (text, encoding, format) and the capability to add or delete words. For a detailed description, see Appendix J.

		Modification	
Type	Transformation	Add.	Del.
Text-level	Reference addition	●	○
	Synonym	●	●
	Spelling mistake	○	●
	Language model	●	○
Encoding-level	Homoglyph	○	●
Format-level	Hidden box	●	●

Finally, for multiple target reviewer in R_{sel} and R_{rej}, we consider the union of candidates from individual reviewer.

3.2 Problem Space

The result of the feature space attack is a modification vector $\delta \in \mathcal{F}$ containing the words that have to be modified in the problem space. These words must be incorporated into an actual template PDF file $z' \in \mathcal{Z}$ such that both the semantics and plausibility constraints are satisfied. Fortunately, the assignment system obtains a document as input and not the raw text. This provides an adversary with more capabilities and flexibility. She can carefully manipulate the text of her submission as well as exploit weak spots in the text representation or document format.

Consequently, we divide the modifications into *text-level*, *encoding-level*, and *format-level* transformations—sorted according to their deniability. Text-level modifications operate on the actual text, so that only targeted modifications are possible. However, the modifications are deniable if the submission raises suspicion during reviewing. Encoding-level and format-level transformations manipulate the text representation and format, respectively, and enable large modifications, but are not deniable once de-

tected. Table 1 lists the transformations implemented in our approach. For a detailed overview, we refer the reader to Appendix J.

Text-level transformations. We begin with transformations that are based solely on changes to the visible text and applicable to any text format. As such, they cannot be readily recognized without a semantic analysis of the text.

(a) Reference addition. As the first transformation, we consider additions to the submission's bibliography. The transformation adds real references that contain the words to be added. As references, we use publications from security conferences and security-related technical reports. Our evaluation demonstrates that this transformation is very effective, while creating plausible and semantics-preserving changes to a paper. However, it introduces side effects, as not only selected words are added, but also parts of the conference names, authors, and titles. This motivates the hybrid search strategy that we outline in Section 3.3.

(b) Synonym. We develop a transformation that replaces a word with a *synonym*. To enhance the quality of the proposed synonyms, instead of using a general model for the English language [e. g., 24, 25, 26], we use a security-domain specific neural embedding that we compute on a collection of 11,770 security papers. Section 7 presents the dataset. This domain-specific model increases the quality of the synonyms, so that this transformation is also difficult to spot.

(c) Spelling mistake. As a third type of text-level manipulations, we implement a spelling-mistake transformation, which is common for misleading text classifiers [27, 28]. Here, we improve on prior work by trying to find typographical errors from a list of common misspellings [29] instead of introducing arbitrary mistakes. For example, the suffix *ance* is often confused with *ence*, so that "appearance" becomes the unobtrusive misspelling "appearence". If we do not find such errors, we apply a common procedure from the adversarial learning literature: We either swap two adjacent letters or delete a letter in the word [24, 27, 28].

(d) Language model. Finally, we apply the large-scale unsupervised language model OPT [30] to create text containing the words to be added. To generate security-related text, we finetune the model using the corpus of 11,770 security papers. While the created sentences are not necessarily plausible, this transformation allows us to technically evaluate the possibility that an adversary creates new text to insert words. Given the increasing capabilities of language models, we expect the chances of creating plausible

text to rise in the long run. Moreover, we assume that in practice attackers would manually polish the generated text to reduce their detection probability.

Encoding-level transformations. As the second class of transformations, we consider manipulations of the text encoding. These manipulations may include the substitution of characters, the application of unicode operations, or changes to the font face and color. For our implementation, we focus on *homoglyph* transformation, inspired by previous works that replaces characters with visually similar counterparts [24, 31]. By replacing a character with a homoglyph, we can remove selected words from the bag-of-words vector used for the topic model. Similarly, there are several other strategies for tampering with text encoding [32]. Since these manipulations also change only the visual appearance of the text, we consider homoglyphs as a representative example of the class of encoding-level transformations.

Format-level transformations. As the third class of transformations, we focus on changes specific to the underlying document format, such as accessibility features, scripting support, and parsing ambiguity [33]. As an example of this class of transformations, we consider *hidden boxes* in the PDF format. Our transformation relies on accessibility support with the latex package `accsupp` to define an invisible alternative text in a hidden box associated with a word. The text extractor processes the alternate text, while PDF readers display only the original word. This discrepancy allows an attacker to add words as alternate text. Likewise, she can put an empty alternative text over a word that should be removed.

Improved transformations. In addition, we exploit the preprocessing implemented by assignment systems. First, we benefit from stemming, so that the transformations only need to add or delete *stems* instead of words. This increases the possibilities to find suitable text manipulations. For example, an attacker can modify the words *attacker* or *attackable* to remove the feature *attack*, since both are reduced to the same stem. Second, we exploit the filtering of stop words. The hidden box transformation requires sacrificing a word for defining an alternative text. As stop words are not part of the feature vector, no side effects occur if they are changed.

3.3 Feature-Problem-Space Attack

We are now equipped with (i) a strategy to find modifications $\delta \in \mathcal{F}$ and (ii) transformations $\omega \in \mathcal{Z}$ to realize δ in a paper submission. The ultimately missing piece is an optimization strategy that brings these two components together. In general, this optimization is responsible for guiding the transformations towards the targeted assignment. In the following, we first present the basic principle of our applied strategy and then introduce two practical extensions.

Hybrid optimization strategy. Due to the challenges around the problem space and the feature space, we use a strategy that *switches alternately* between $\mathcal{Z}$ and $\mathcal{F}$. Figure 2 on page 62 schematically illustrates our alternating approach. For an initial submission z, the adversary extracts the features (step ❶) and performs a feature-space attack (step ❷ and ❸). As Φ is not invertible, the adversary then has to find suitable transformations in the problem space (step ❹) that realize the requested modifications. This leads to a new feature vector in $\mathcal{F}$ (step ❺). However, this vector is shifted due to side effects and limitations of the transformations. Consequently, the adversary continues her search from this new position and repeats the process iteratively until the target is reached or the maximum number of iterations have passed.

We note that side effects are not always negative as assumed by prior work [34]. In our evaluation, for example, we found that the additional words introduced by the reference transformation can further push a reviewer's rank towards the target, since the additional words may also relate to other selected reviewers, for example, due to co-authors or paper titles. However, the impact of side effects is difficult to predict in advance, so that an optimization strategy should be capable of dealing with positive as well as negative side effects.

Constraint mapping $\mathcal{Z} \to \mathcal{F}$. Our first extension to this hybrid strategy addresses the complexity of problem-space modifications. In practice, not every requested modification from $\mathcal{F}$ can be realized in $\mathcal{Z}$ with the implemented transformations due to PDF and LaTeX restrictions. For example, in LaTeX, homoglpyhs are not usable in the listing environment, while the hidden box is not applicable in captions or section titles. In general, such restrictions are difficult to predict given the large number of possible LaTeX packages. Instead of solving such shortcomings in the problem space by tediously adjusting the transformations to each special case, we resort to a more generic approach and transfer problem-space constraints back to the feature space. The

transformers in $\mathcal{Z}$ first collect words that cannot be handled, which are then blocked from being sampled during candidate generation in $\mathcal{F}$.

Surrogate models. We introduce a second extension for the black-box scenario. In this scenario, the adversary has no access to the victim model. Still, she can leverage public information about the program committee, collect papers from its members, and assemble a dataset similar to the original training data. This allows her to train a *surrogate model* that enables preparing an adversarial paper without access to the assignment system. This strategy has been successfully used for attacks against neural networks [35]. However, in our case, this strategy is hindered by a problem: LDA models suffer from high variance [36, 37]. Even if the adversary had access to the original data, she would still get different models with varying predictions. This makes it unlikely that an adversarial paper computed for one model transfers to another.

As a remedy, we propose to use an ensemble of surrogate models to better approximate the space of possible LDA models. We run the attack simultaneously for multiple models until being successful against *all* surrogates. To this end, we extend the feature-space attack to multiple target models: (i) we create candidates for each surrogate model independently and consider the union over all surrogates and (ii) we compute the loss as the sum of individual losses over all surrogates. Intuitively, this increases the robustness of an adversarial paper and, consequently, improves the success rate that the attack transfers to the unknown victim model.

4 Evaluation

In the following, we evaluate the efficacy of the proposed approach to prepare adversarial papers. To this end, we simulate the automatic paper-reviewer assignment process of a real conference with the full program committee (PC). We consider two different scenarios: First, we demonstrate how a white-box attacker with full-knowledge about the target system can select and reject reviewers for a submission. Second, we consider a black-box adversary with only limited knowledge and no access to the trained assignment system. We show that such an adversary can generate effective surrogate models by exploiting public knowledge about the conference. Finally, we verify that the manipulated papers are plausible and preserve the semantics of the text.

Setup. We use Autobid [6] as an open-source realization of the TPMS concept [7]. We simulate an automatic assignment for the *43rd IEEE Symposium on Security and*

Privacy with the full PC and set the paper load $\mathsf{L}_{\mathbf{x}} = 5$ (i. e., assign each submission to five reviewers). In contrast to the real conference, we assume a fully automated assignment without load balancing and conflicts (see Section 7). As we do not have access to the original submissions, we use the accepted papers as substitutes. In total, we find 32 papers on the arXiv e-Print archive with LaTeX source, which we use for our evaluation.

The PC consists of 165 persons. For each PC member, we construct an archive A_r of papers representative for the person's expertise by crawling their *Google Scholar* profile. We select 20 paper for each reviewer and compile the corpus as the union of these archives. To simulate a black-box scenario, we additionally generate *surrogate corpuses* that overlap with the original data between 0% and 100%. Appendix A describes this process in detail. In all cases, we train Autobid with the default configuration on a given corpus.

For each attack, we perform a grid search on its parameters to realize a reasonable trade-off between efficacy and efficiency. We start by relaxing any constraints on δ ($L_1^{\max} = \infty$ and $L_\infty^{\max} = \infty$) and run the attack with at most $S = 8$ transitions between the feature space and problem space (see Appendix B for details). All experiments are performed on a server with 256 GB RAM and two Intel Xeon Gold 5320 CPUs.

Performance measures. We use three measures to evaluate the attack's performance. First, we consider an adversarial paper z' to be *successful* if the constraints from Equation 5 are fulfilled. Second, to quantify modifications to the submission, we use two standard measures: L_1 and L_∞ norm. Given the modified word counts $\mathbf{x}' \coloneqq \mathbf{x} + \delta$, these are computed as

$$||\delta||_1 = \sum_i |\delta_i| \text{ and } ||\delta||_\infty = \max_i |\delta_i| \,. \tag{10}$$

L_1 is the absolute number of modified words and provides a general overview on the total amount of modifications. Intuitively, we are interested in minimizing L_1 to make an attack less suspicious. Similarly, L_∞ is the maximum change in a single dimension (i.e., a single word) and ensures that a single word is not included too frequently. Third, we assess the *semantics* and *plausibility* of the manipulated papers in a user study with security researchers.

Table 2: **Feature-space search.** We compare our attack with two baselines: hill climbing and morphing. For this comparison, we consider three attack objectives: (1) selecting, (2) rejecting, and (3) substituting of reviewers.

	Selection		Rejection		Substitution	
L_1 norm						
Our attack	704	×1.00	1032	×1.00	2059	×1.00
Hill climbing	1652	×2.35	2255	×2.18	5526	×2.68
Morphing	3059	×4.35	-	× -	-	× -
L_∞ norm						
Our attack	17	×1.00	43	×1.00	62	×1.00
Hill climbing	38	×2.22	44	×1.02	98	×1.58
Morphing	45	×2.63	-	× -	-	× -

4.1 White-box Scenario

In our first scenario, we focus on a white-box scenario and consider three attack objectives: (1) selection, (2) rejection, and (3) substitution of a reviewer. For these objectives, we focus on reviewers that are already "close" to a submission in the assignment system. For example, a paper on binary analysis would raise suspicion if it would get assigned to a reviewer with a cryptography background.

We use the initial assignment scores of the submission as a proxy to simulate this setting. We determine potential reviewers by computing the ranking for the unmodified submission and consider the 10 reviewers with highest scores. To study objective (1), we sample reviewers from the ranks 6–10 and attempt to get them assigned to the submission. Analogously, for objective (2), we select reviewers from the ranks 1–5 and aim at eliminating their assignment. Finally, for objective (3), we first select a reviewer for removal and then a counterpart for selection. We repeat this procedure 100 times with random combinations of papers and reviewers for each objective. Moreover, to account for the high variance of LDA, we train the topic model 8 times and average results in the following.

Feature-space search. We start our evaluation by examining the feature-space search of our attack in detail. For this experiment, we consider format-level transformations

that can realize arbitrary changes. Other transformations are evaluated later when we investigate the problem-space side of our attack.

The results of this experiment are presented in Table 2 and further detailed in Appendix C. We observe that our approach is very effective: 99.7 % of the attacks finish successfully with a median run-time of 7 minutes. The number of performed changes shows high variance, ranging between 9 and 22,621 adapted words. Despite this broad range, however, the average manipulation involves only between 704 and 1,032 words for objectives (1) and (2), respectively. For reference, an unmodified submission contains 7,649 words on average, so that the necessary changes for preparing an adversarial paper amount to 9% and 13% of the words.

Among the three objectives, we see a trend that selecting a reviewer is more efficient than rejecting one. Rejected reviewers have—per construction—a high assignment score, and hence share many topics with nearby reviewers. In contrast, for selected reviewers it is easier to determine topics with less side effects. The third scenario, where we both reject and select a reviewer, is naturally the hardest case. Generally, we observe that topic models based on LDA are comparatively robust against adversarial noise, in relation to neural networks which can be deceived into a misclassification by changing only a few words [e.g., 27, 24].

Baseline experiments. To put these numbers into perspective, we examine two baselines. First, we implement a hill climbing approach that directly manipulates the topic vector of a submission (cf. Equation 6) by sampling words from the topic-word distributions associated with a reviewer. For the second baseline, we consider an approach that morphs a target submission with papers that already contains the correct topic-word distribution. To find these papers, we compute all assignments of the training corpus and identify submissions to which the target reviewer is assigned. We then repeatedly select words from these submissions and expand our adversarial paper until we reach the target. In rare cases, we could not find papers in which the reviewer is highly rated. We exclude such cases from our experiments.

Considering all three objectives, the hill climbing approach shows a lower success rate: Only 92.2 % of the papers are successfully manipulated. The failed submissions either reach the maximum number of 1,000 iterations or get stuck in a local minimum. In successful cases, the attacker needs to introduce more than twice as many changes compared to our attack and the median L_1 norm increases from 704–2,059 to 1,652–

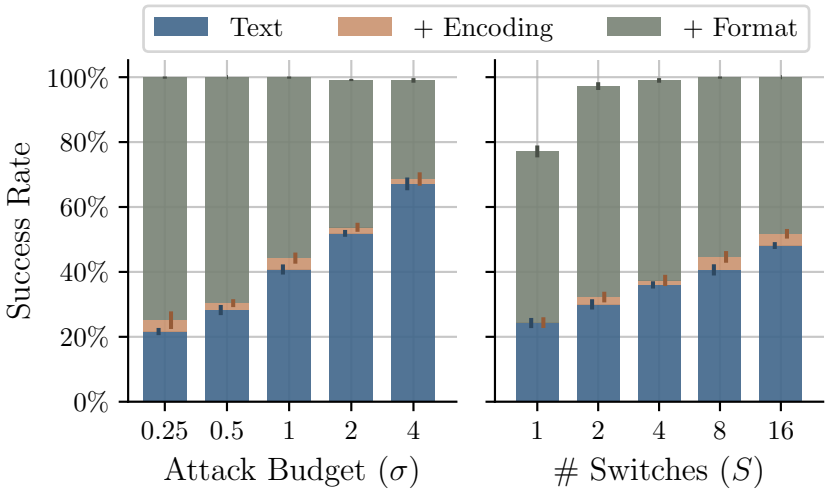


Figure 4: **Feature-problem-space attack.** We simulate the attack with differently scaled attack budgets σ (left) and $S = 8$ switches. We repeat the experiment (right) with the base budget $\sigma = 1$ and vary S. For both cases, we randomly select 100 targets from all three objectives that require $\leq 1,000$ changes in $\mathcal{F}$. We report the mean success rate over 8 repetitions.

5,526 words. For the morphing baseline, the attack is successful in only 91.1 % of the cases and again needs to introduce significantly more words. We find that the median L_1 norm increases by a factor of 4.35 with a maximum of 29,291 modified words for a single attack.

Generalization of attack. To investigate the generalization of our attack, we repeat this experiment for a second real conference. In particular, we simulate the assignment of the *29th USENIX Security Symposium* with 120 reviewers. We consider 24 original submissions and construct targets as before. Results of this experiment are presented in Appendix D. We observe a similar performance across all three objectives, indicating the general applicability of our attack.

Scaling of target reviewers. Next, we scale the attack to larger sets of target reviewers and consider different combinations for selecting, rejecting, and substituting reviewers. We allow an attacker to select up to five target reviewers, which is equivalent to replacing *all* of the initially assigned reviewers. Furthermore, we allow the rejection of up to two reviewers. We focus again on close reviewers and randomly select 100 sets of targets per combination.

The results are summarized in Appendix E. The attack remains effective and we can successfully craft adversarial papers in most of the cases. We observe a clear trend that with increasing numbers of target reviewers, we need to perform more changes to the submission. For example, to select all five reviewers, in the median we need to modify 5,968 words. This is expected: we have to move the submission in the topic space from the initially-assigned reviewers to the targeted ones. By adding more reviewers, we include more constraints which results in a significant amount of modifications.

All transformations. So far, we have focused on format-level transformations to realize manipulations. These transformations exploit intrinsics of the submission format, which effectively allows us to make arbitrary changes to a PDF file. An attacker likely has access to similar transformations in any practical setting. In fact, robust parsing of PDF files has been shown to be a hard problem [e.g., 38]. However, we believe it is important for an attacker to minimize any traces and consider different classes of transformations as introduced in Section 3.2.

(a) Attack budget For this experiment, we introduce an attack budget to describe the maximum amount of allowed modifications for a given transformation. This budget trades off the ability of a transformation to introduce changes with their conspicuousness, since too many (visible) modifications will likely lead to a rejected submission. In particular, we assume a maximum of 25 real and 5 adaptive added BibTeX entries, at most 25 replacements of words with synonyms, no more than 20 spelling mistakes, and up to 10 requested words on average through a text from a language model. In Section 4.4, we validate these parameters and assess if the resulting adversarial papers are unobtrusive to human observers.

As a result of the attack budget, we cannot realize arbitrary modifications, since their total amount is restricted. To study this in more detail, we consider the success rate as a function of the attack budget scaled with a factor σ between 0.25 and 4. During the attack, we split the budget equally across 8 feature-problem-space transitions. We require that targets are feasible with this budget and randomly select 100 targets from the three attack objectives that require $\leq 1,000$ changes in $\mathcal{F}$. Finally, we consider three different configurations: (1) text-level transformations, (2) text-level and encoding-level transformations, and (3) text-level, encoding-level, and format-level transformations combined. We do not restrict the budget for format-level transformations as these transformations are generally not visible.

The results are shown on the left side of Figure 4. For text-level transformations and text-level & encoding-level transformations, we see an increase in the success rate when the attack budget grows. For the base budget ($\sigma = 1$), 40.75% of the adversarial papers can be prepared with text-level transformations only. That is, no changes in the format and encoding are necessary for manipulating the reviewer assignment. This can be further improved by increasing the budget, for instance, 67.13% of the papers become adversarial by scaling it to 4. For smaller budgets, however, we observe that there is often not enough capacity to realize the required modifications. Still, using format-level transformations improves the success rate to 100% in almost all cases. In rare case, we observe that the attack gets stuck in a local minima. Interestingly, this is more likely with larger budgets. In these cases, the attack makes bigger steps per iteration which introduces more side effects. From the perspective of an attacker, this can be resolved by either increasing the number of switches or reducing the budget.

(b) Problem-feature-space transitions. To better understand the influence of the alternating search on the success rate of our attack, we conduct an additional experiment. In particular, we simulate our attack for different numbers of transitions $S \in \{1, 2, 4, 8, 16\}$ between the problem space and the feature space. We consider the same targets as before and set the attack budget to $\sigma = 1$.

The results of this experiment are depicted on the right side of Figure 4. Increasing the number of transitions has a significant effect on the success rate. For all configurations, we see a steady improvement when the number of problem-feature-space transitions increases. Notably, even the format-level transformations *require* multiple transitions in some cases. The success rate increases from 77.13%—with no transitions—to 100% when increasing S. By alternating between $\mathcal{F}$ and $\mathcal{Z}$ we share constraints between problem and feature space to find modifications that can be realized in the problem space. This further underlines that it is beneficial and in fact necessary to considreser both spaces together.

4.2 Black-box Scenario

In practice, an attacker typically does not have unrestricted access to the target system. In the following, we therefore assume a black-box scenario and consider an adversary with only limited knowledge. In particular, this adversary cannot access the assignment system and its training data. Instead, we demonstrate that she could

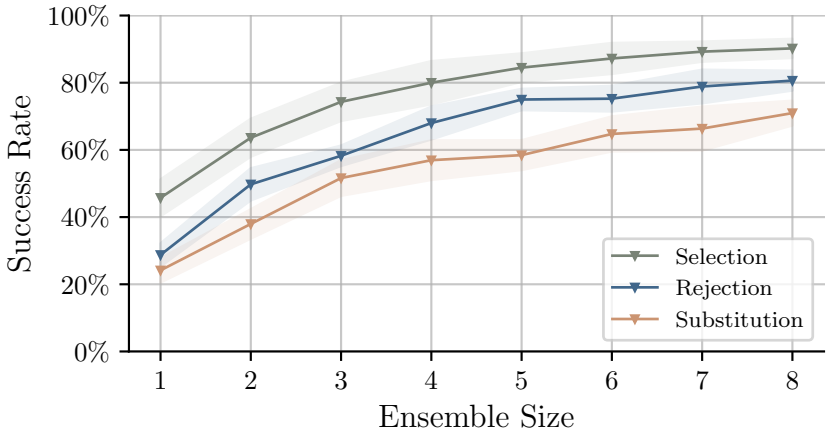


Figure 5: **Surrogate ensemble sizes.** We simulate the attack with varying numbers of surrogate models. For each ensemble size, we report the mean success rate over 8 target systems with 100 targets each for all three attack objective.

leverage her knowledge about the program committee and construct a surrogate dataset to train her own models for preparing adversarial papers.

The assignment systems AutoBid and TPMS do not specify how the corpus for training a topic model is constructed. They only require that the selected publications are representative of the reviewers. Hence, even if we do not know the exact composition of the training data, we can still collect a surrogate corpus of representative data with public information, such as recent papers of the PC members, and transfer our attack between models. In practice, the success of this transfer depends on two factors: (a) the stability of the surrogate models and (b) the overlap of publications between the original training data and the surrogate corpus.

Stability of surrogate models. The training of LDA introduces high variance [36, 37], so that adversarial papers naïvely computed against one model will likely not transfer to another. To account for this instability, we approximate the model space and consider an *ensemble* of surrogate models. That is, we run our attack simultaneously against multiple surrogate models trained on the same data. We focus on format-level transformations and repeat the attacks for all three objectives. We vary the number of models in the ensemble from 1 to 8 and consider an overlap of 70% between the underlying surrogate corpus and the original training data.

Figure 5 show the results of this experiment. Across all objectives, we see an improvement of the success rate when increasing the number of surrogate models. This is

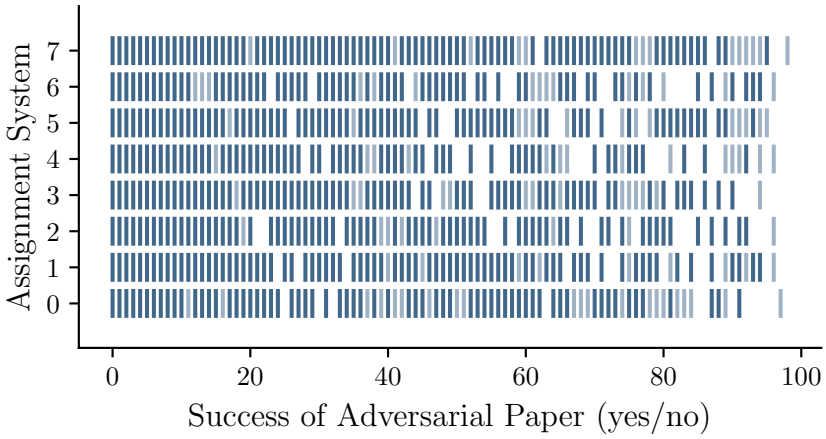


Figure 6: **Transferability.** We visualize the transferability of 100 adversarial paper among 8 target assignment systems. Attacks were performed with an ensemble size of 8 and we focus on the selection objective. Adversarial papers where the unmodified submission is already successful are displayed in light blue.

intuitive: the adversarial papers are optimized against all models and thus more likely to transfer to other models. This robustness, however, comes at a cost and the number of modifications increases as well. The median L_1 norm increases from 1,990 to 7,556 when considering 8 instead of a single surrogate model (see Appendix F).

As a result, an adversary in the black-box scenario must find a trade-off: If she needs a successful attack with high probability, she must sacrifice detectability and modify a large number of words. If, on the other end, she only wants to increase her chances for a specific assignment, she can operate without an ensemble and adapt only a few words.

To further study the transferability of our attack, we sample 100 target reviewer from the median ranking computed over 8 assignment systems and simulate the attack with an ensemble of 8 surrogates. Figure 6 visualizes how the resulting adversarial papers transfer among the target systems. 96% of the papers are successful against four or more target systems and 34 % are successful considering all 8 systems.

Overlap of surrogate corpus. To understand the role of the surrogate corpus, we finally repeat the previous experiment with varying levels of overlap. Surprisingly, the attack remains robust against variations in training data. The success rate only fluctuates slightly: 78.0% (100% overlap), 80.0% (70% overlap), 79.6% (30% overlap), and 82.8% (0% overlap). To explain this, we compute the cross-entropy of the reviewer-

to-words distributions $\hat{\phi}_r$ for models trained on training data with different overlaps. We observe that the cross-entropy between models trained on the same dataset (i.e., 100% overlap) is in the same range compared to models trained on different data (cf. Appendix G for details). As LDA models trained on the same corpus already vary significantly, our attack seeks a robust solution that transfers well if the surrogate models have less overlap with the original training data.

4.3 Plausibility and Semantics

Finally, we empirically verify if the adversarial modifications are (a) plausible and (b) preserve the semantics of the text.

Study design. As dataset, we use the combined set of original and adversarial papers from our evaluation. In total, we select seven original papers and their adversarial counterparts, ensuring varying topics and transformations. The attack budget is $\sigma = 1.00$. Due to a limited number of participants, we focus on visible transformations (i.e. encoding-level and text-level) that a reviewer could detect. Each participant selects ("bids on") one paper. This selection cannot be changed afterwards and participants are secretly assigned either to the adversarial or to the unmodified version. Each participant will only check one paper to avoid potential bias and fatigue effects.

We design the review process along two phases. Our methodology here is inspired by the work from Bajwa et al. [39] and Sullivan et al. [40]. In the first phase, we request participants to write a mini-review (as a proxy task) for a given paper. In the second phase, we ask if they think the paper has been manipulated. Importantly, the answers of phase 1 cannot be changed. This two-phase separation allows us to observe two factors: First, we can analyze how suspicious adversarial papers are to an unaware reader. Second, once the reader is informed, we can learn about which transformations are noticeable and make our attack detectable. In each phase, we provide a template with questions on a numerical scale from 1–5, together with free text fields for justifying the rating. Participants are debriefed finally. We obtained approval from our institution's Institutional Review Board (IRB) and our study protocol was deemed to comply with all regulations.

Results. We recruited 21 security researchers (15× PhD students, 4× postdocs, 1× faculty, 1× other). All participants are familiar with the academic review process but have different review experience (7× have not written a review before, 4× between 1-2

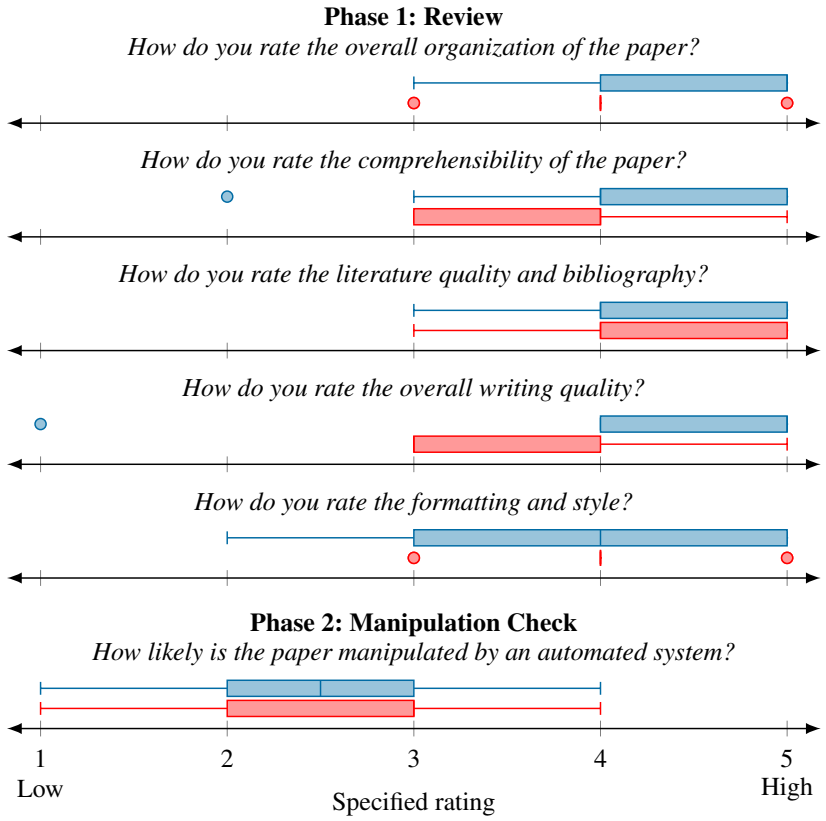


Figure 7: **Ratings of benign and adversarial papers.** For each question, the upper boxplot shows the ratings from the benign papers, the lower boxplot from the adversarial papers.

reviews, 6× between 3-10, and 4× at least 10 reviews). The participants reviewed a total of 12 adversarial and 9 original submissions.

Figure 7 summarizes the results. Benign and adversarial submissions are rated similar across all review questions. No participant was certain that a paper was manipulated (i. e., gave it a ranking of 5) and only a single of the 12 manipulated submissions was flagged as suspicious with a rating of 4. This was justified with missing references and redundancy in the text—neither of which were introduced by our attack. Interestingly, this reviewer did notice the spelling mistake and language model transformer (when asked about the writing quality), but did not attribute this as a sign for manipulation. This is opposed to two false positive ratings of benign papers, which results in a overall detection precision of 33% with a recall of only 8%. This highlights the difficulty to detect any introduced modifications.

Finally, we check that the semantics of the papers are not changed. With the limited attack budget, only small, bounded changes are made to a paper. This is further supported by the organization and comprehensibility ratings in Figure 7, which are similar between manipulated and benign submissions.

5 Discussion

Our work reveals a notable vulnerability in systems for automatic paper-reviewer assignment. In the following, we discuss further aspects of our findings, including limitations, defenses, and the implications and benefits of an attack.

Committee size. We simulate an automatic assignment for two large security conferences with committees composed of 120 and 165 reviewers, respectively. Considering the current trend, it is likely that these conferences will continue to grow larger. In the following, we want to understand how an increased set of concurring reviewers impacts the attack. Therefore, we consider committees with 100–500 reviewers sampled from a pool of 528 reviewers (taken from USENIX and S&P committees between 2018–2022) with a total of 11,770 papers in the underlying corpus. Appendix H shows the number of required changes as a function of the committee size. Across the three objectives—selection, rejection, and substitution—we observe only a minor impact on the attack. The attack remains successful with a success rate between 98.00% – 98.92% and the number of required modifications remains largely unaffected. For the smallest committee considered (with 100 reviewers), we observe a slightly larger uncertainty. Intuitively, in this case the assignment is more dependent on the particular choice of the committee which gets averaged out for larger committees.

Load balancing and conflicts. Our attack focuses on the manipulation of assignment scores and assumes a direct matching from PC members to submissions. For a complete end-to-end attack, an attacker would also need to take load balancing of submissions and reviewer conflicts into account. For example, the target submission might compete with another paper on the same topic and get a different assignment despite a successful manipulation of the topic model.

Unfortunately, these obstacles are hard to model, as conflicts and the other submissions are typically not known to the adversary. Instead, we can generally improve the resilience of our attack. By increasing the margin Φ of the target reviewer to others, we can make a matching assignment more likely. Interestingly, in this case, conflicts

can be even seen as a simplification: if a target reviewer is the top candidate among all reviewers $\mathcal{R}$, she is also the top candidate for only a subset of reviewers (i.e., all unconflicted reviewers).

To further understand the role of this margin, we simulate the selection of a reviewer for different values of $\Phi \in \{0, 0.1, 0.2\}$ and varying numbers of concurring submissions between 200 and 1,000 (sampled from a hold-out corpus). We model the full assignment to maximize similarity subjected to load constraints as introduced in Section A. We assume $\mathsf{L}_{\mathbf{x}} = 5$ reviews per paper and that each reviewer is assigned $\mathsf{L}_r = 10$ submissions. Appendix I shows the attack's success rate as a function of the number of concurring submissions. The attack remains effective but we observe a slight downward trend of its success rate. This is expected: with increasing number of submissions, there exist more similar paper that compete for a given reviewer. An attacker can account for this by (1) increasing the margin and, as the attack is undetectable (in general), an attacker could (2) further increase her chances by repeating the attack (e.g., resubmitting a rejected paper).

Paper corpus. We select *accepted* papers from IEEE S&P 2022 as basis for our evaluation. This selection leads to a potential bias, as rejected submissions are not considered. However, we do not expect any impact on our results. Papers follow a common structure, so that our transformations in LaTeX are applicable in general. The feature-space algorithm works on bag-of-words vectors, which is just another representation for any paper. In Appendix D, we test our attack with papers from the 29th USENIX Security Symposium and find no significant difference in our results.

Countermeasures and defenses. Our results show that systems based on topic models such as LDA have relatively strong robustness towards adversarial noise. This stands in stark contrast to neural networks, where changing only a few words can already lead to a misclassification [e.g., 27, 24]. However, our work also demonstrates that LDA-based systems are still vulnerable to adversarial examples and there is a need for appropriate defenses.

Unfortunately, text-level manipulations are challenging to fend off, as they can only be spotted on the semantic level. In our user study, participants often struggled to differentiate adversarial modifications from benign issues and an adversary can always *manually* rewrite an adversarial paper to further reduce the detection probability. Moreover, even completely machine generated text—such as done with our OPT-based

transformer—is often indistinguishable from natural text [41, 42]. The underlying models are evolving rapidly and current state-of-the-art models such as InstructGPT [43] and Galactica [44] are now actively used for academic writing.

For encoding-level and format-level changes, however, defenses are feasible: The root cause of these manipulations is the disparity between human perception and parser-based text extraction. Thus, an effective defense needs to mimic a human reader as close as possible similar to defenses recently proposed for adversarial examples in other domains [e.g. 45]. To evaluate this defense, we replace the parser-based text extraction (`pdftotext`) with an optical character recognition (OCR) (`tesseract`). We observe that for the modified system the encoding-level and format-level attacks now completely fail, while the performance of the text-level attacks remains unaffected. At the same time, however, we observe a large increase in runtime. Compared to the parser-based extraction, OCR is orders of magnitude slower and needs an average time of 56 s for a single submission compared to 0.14 s with conventional text extraction.

Other countermeasures can be more tailored to the individual transformations: Flag usage of unusual font encodings to prevent homoglyph attacks, remove comment boxes and non-typeset pages in a preprocessing step, or automatically verify the bibliography entries using online bibliography databases.

Benefits and implications. Manipulating a submission comes with a considerable risk if the attack is detected. This can range from a desk reject over a submission ban at a specific venue to permanent damage of the authors' scientific reputation [46]. Nevertheless, recent incidents show that academic misconduct happens. Dishonest authors, for example, leveraged synthetic texts to increase the paper output [47]. Moreover, *collusion rings* exist where authors and reviewers collaborate to accept each other's papers [48]. Automated assignment techniques can raise the bar for dishonest collaborations considerably [49], yet our work shows that these techniques need to be implemented with care. Apart from collusion rings, dishonest authors can also work alone: They can try to promote an unfamiliar reviewer who might overlook paper issues and thus more likely submit a positive review.

We believe that dishonest authors more likely risk *deniable* manipulations such as a few spelling mistakes or additional references. Our evaluation shows this is sometimes already enough, for example, to promote an unfamiliar reviewer. As the line between adversarial and benign issues in a paper is often not clear, such an attack can be hard to

discover. All in all, the automatic assignment of papers enables not only manipulations that undermine the entire reviewing process, but also small-scale attacks in which assignments are tweaked by a few deniable changes.

6 Related Work

Our attack touches different areas of security research. In the following, we examine related concepts and methods.

Adversarial learning. A large body of work has focused on methods for creating adversarial examples that mislead learning-based systems [50]. However, most of this work considers attacks in the image domain and assumes a one-to-one mapping between pixels and features. This assumption does not hold in discrete domains, leading to the notion of *problem-space attacks* [34, 51]. Our work follows this research strand and introduces a new hybrid attack strategy for operating in both the feature space and problem space. Furthermore, we examine weak spots in *text preprocessing*, which extend the attack surface for adversarial papers. These findings complement prior work advocating that the security of preprocessing in machine learning needs be considered in general [52].

Table 3 summarizes prior work on misleading text classifiers. While we build on some insights developed in these works, text classification and paper assignment differ in substantial aspects: First, the majority of prior work focuses on untargeted attacks that aim at removing individual features. In our case, however, we have to consider a targeted attack where an adversary needs to specifically change the assignment of reviewers. Second, prior attacks often directly exploit the gradient of neural networks or compute a gradient by using word importance scores. Such gradient-style attacks are not applicable to probabilistic topic models.

In view of these differences, our work is more related to the attack from Zhou et al. [23] which studies the manipulation of LDA. The authors show that an evasion is *NP-hard* and present an attack to promote and demote individual LDA topics. For our manipulation, however, we need to adjust not only individual topics but the complete topic distribution as well as consider side effects with concurring reviewers.

Attacks on assignment systems. Finally, another strain of research has explored the robustness of paper-reviewer assignment systems. Most of these works are based on *content-masking attacks* [33, 57], which use format-level transformation to exploit

Table 3: Overview of related attacks against text classifiers.

	Perturbation				Constr.		Attack		
Paper	Char	Word	Sentence	Format	Semantics	Plausibility	Untargeted	Targeted	Classifier
This work	●	●	●	●	✓	✓	✓	✓	Assign.
Alzantot et al. [53]	●	●			✓		✓		NN
Ebrahimi et al. [54]	●	●			✓		✓		NN
Eger et al. [31]	●				✓		✓		NN
Gao et al. [27]	●				✓		✓		NN
Iyyer et al. [55]			●		✓		✓		NN
Jin et al. [25]		●			✓		✓		NN
Li et al. [24]	●	●			✓		✓		NN,LR
Liu et al. [28]	●	●			✓		✓		NN
Papernot et al. [56]		●					✓		NN
Ren et al.[26]		●			✓		✓		NN

the discrepancy between displayed and extracted text. More specifically, Markwood et al. [33] and Tran and Jaiswal [57], similar to our work, target the paper-reviewer assignment task. Their attack is evaluated against Latent Semantic Indexing [58]—that is not used in real-world systems like TPMS. Although Tran and Jaiswal [57] recognize the shortcomings of format-level transformations, they do not explore text-level transformations or the interplay between the problem space and feature space of topic models.

Complementary to our work, a further line of research focuses on the collusion of reviewers. These works have analyzed semi-automatic paper matching systems under the assumption that malicious researchers can manipulate the paper assignment by carefully adjusting their paper biddings. Jecmen et al. [8] propose a probabilistic matching to decrease the probability of a malicious reviewer to be assigned to a target submission, while Wu et al. [9] tries to limit the disproportional influence of malicious biddings.

7 Conclusion

In this paper, we demonstrate that current systems for automatic paper-reviewer assignments are vulnerable and can be misled by *adversarial papers*. On a broader

level, we develop a novel framework for constructing adversarial examples in discrete domains through joint optimization in the problem space and feature space. Based on this framework, we can craft objects that satisfy real-world constraints and evade machine-learning models at the same time.

In summary, our work demonstrates a significant attack surface of current matching systems and motivates further security analysis prior to their deployment. As a result, we have informed the developers of TPMS and Autobid about our findings, as part of a responsible disclosure process.

Acknowledgments

We thank our shepherd and reviewers for their valuable comments and suggestions. We also thank Ajeeth Kularajan, Andreas Müller, Jonathan Evertz, and Sina Wette for their assistance as well as Charlotte Schwedes and Annabelle Walle for their support with the user study. This work was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy – EXC 2092 CASA – 390781972), the German Federal Ministry of Education and Research under the grant BIFOLD23B, and the European Research Council (ERC) under the consolidator grants MALFOY (101043410) and RS^3 (101045669). Moreover, this work was supported by a fellowship within the IFI program of the German Academic Exchange Service (DAAD) funded by the Federal Ministry of Education and Research (BMBF).

References

[1] Ananta Soneji, Faris Bugra Kokulu, Carlos Rubio-Medrano, Tiffany Bao, Ruoyu Wang, Yan Shoshitaishvili, and Adam Doupé. "Flawed, but like democracy we dont have a better system": The Experts Insights on the Peer Review Process of Evaluating Security Papers. In *IEEE Symposium on Security and Privacy (S&P)*, 2022.

[2] Neil Lawrence and Corinna Cortes. The NIPS Experiment. Post on personal blog, 2014.

[3] Hannah Bast. How Objective is Peer Review: The ESA Experiment. Blog post on CACM, 2018.

[4] Davide Balzarotti. System Security Circus. Post on personal blog, 2020.

[5] Hsuan-Tien Lin, Maria Florina Balcan, Raia Hadsell, and MarcAurelio Ranzato. What we learned from NeurIPS 2020 reviewing process. Blog post on Medium, 2020.

[6] Bryan Parno. Autobid. Public GitHub Repository.

[7] Laurent Charlin and Richard Zemel. The Toronto Paper Matching System: An Automated Paper-Reviewer Assignment System. In *International Conference on Machine Learning (ICML)*, 2013.

[8] Steven Jecmen, Hanrui Zhang, Ryan Liu, Nihar B. Shah, Vincent Conitzer, and Fei Fang. Mitigating Manipulation in Peer Review via Randomized Reviewer Assignments. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[9] Ruihan Wu, Chuan Guo, Felix Wu, Rahul Kidambi, Laurens van der Maaten, and Kilian Q. Weinberger. Making Paper Reviewing Robust to Bid Manipulation Attacks. In *International Conference on Machine Learning (ICML)*, 2021.

[10] Xiang Liu, Torsten Suel, and Nasir Memon. A Robust Model for Paper Reviewer Assignment. In *ACM Conference on Recommender Systems (RecSys)*, 2014.

[11] Xinlian Li and Toyohide Watanabe. Automatic Paper-to-reviewer Assignment, based on the Matching Degree of the Reviewers. In *International Conference in Knowledge Based and Intelligent Information and Engineering Systems (KES)*,

2013.

[12] Ivan Stelmakh, Nihar B. Shah, and Aarti Singh. PeerReview4All: Fair and Accurate Reviewer Assignment in Peer Review. In *Conference on Algorithmic Learning Theory (ALT)*, 2019.

[13] Cheng Long, Raymond Chi-Wing Wong, Yu Peng, and Liangliang Ye. On Good and Fair Paper-Reviewer Assignment. In *IEEE International Conference on Data Mining (ICDM)*, 2013.

[14] Surajit Chaudhuri et al. Conference Management Toolkit (CMT).

[15] Eddie Kohler et al. HotCRP Conference Review Software.

[16] Julie Beth Lovins. Development of a Stemming Algorithm. *Mechanical Translation and Computational Linguistics*, 1968.

[17] Steven Bird, Ewan Klein, and Edward Loper. *Natural Language Processing with Python*. O'Reilly, 2009.

[18] David Blei, Andrew Ng, and Michael Jordan. Latent Dirichlet Allocation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2002.

[19] Matthew D. Hoffman, David M. Blei, and Francis R. Bach. Online Learning for Latent Dirichlet Allocation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2010.

[20] William M. Darling. A Theoretical and Practical Implementation Tutorial on Topic Modeling and Gibbs Sampling. In *Annual Meeting of the Assoc. for Computational Linguistics: Human Language Technologies (HLT)*, 2011.

[21] Radim Řehůřek and Petr Sojka. Software Framework for Topic Modelling with Large Corpora. In *LREC Workshop on New Challenges for NLP Frameworks*, 2010.

[22] Camillo J Taylor. On the Optimal Assignment of Conference Papers to Reviewers. Technical report, 2008.

[23] Qi Zhou, Haipeng Chen, Yitao Zheng, and Zhen Wang. EvaLDA: Efficient Evasion Attacks Towards Latent Dirichlet Allocation. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2021.

[24] Jinfeng Li, Shouling Ji, Tianyu Du, Bo Li, and Ting Wang. TextBugger: Generating Adversarial Text Against Real-world Applications. In *Symposium on Network and Distributed System Security (NDSS)*, 2019.

[25] Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. Is BERT Really Robust? A Strong Baseline for Natural Language Attack on Text Classification and Entailment. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2020.

[26] Shuhuai Ren, Yihe Deng, Kun He, and Wanxiang Che. Generating Natural Language Adversarial Examples through Probability Weighted Word Saliency. In *Annual Meeting of the Assoc. for Computational Linguistics (ACL)*, 2019.

[27] Ji Gao, Jack Lanchantin, Mary Lou Soffa, and Yanjun Qi. Black-Box Generation of Adversarial Text Sequences to Evade Deep Learning Classifiers. In *IEEE Security and Privacy Workshops (SPW)*, 2018.

[28] Hui Liu, Yongzheng Zhang, Yipeng Wang, Zheng Lin, and Yige Chen. Joint Character-Level Word Embedding and Adversarial Stability Training to Defend Adversarial Text. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2020.

[29] The Most Common English Misspellings. Blogpost on Lexico, 2021.

[30] Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona T. Diab, Xian Li, Xi Victoria Lin, and other. OPT: Open Pre-trained Transformer Language Models. *Computing Research Repository (CoRR)*, 2022.

[31] Steffen Eger, Gözde Gül Sahin, Andreas Rücklé, Ji-Ung Lee, Claudia Schulz, Mohsen Mesgar, Krishnkant Swarnkar, Edwin Simpson, and Iryna Gurevych. Text Processing Like Humans Do: Visually Attacking and Shielding NLP Systems. In *Conference of the North American Chapter of the Assoc. for Computational Linguistics: Human Language Technologies, (NAACL-HLT)*, 2019.

[32] Nicholas Boucher, Ilia Shumailov, Ross Anderson, and Nicolas Papernot. Bad Characters: Imperceptible NLP Attacks. In *IEEE Symposium on Security and Privacy (S&P)*, 2022.

[33] Ian Markwood, Dakun Shen, Yao Liu, and Zhuo Lu. PDF Mirage: Content Masking Attack Against Information-Based Online Services. In *USENIX Security Symposium*, 2017.

[34] Fabio Pierazzi, Feargus Pendlebury, Jacopo Cortellazzi, and Lorenzo Cavallaro. Intriguing Properties of Adversarial ML Attacks in the Problem Space. In *IEEE Symposium on Security and Privacy (S&P)*, 2020.

[35] Nicolas Papernot, Patrick D. McDaniel, and Ian J. Goodfellow. Transferability in Machine Learning: From Phenomena to Black-Box Attacks using Adversarial Samples. *Computing Research Repository (CoRR)*, 2016.

[36] Amritanshu Agrawal, Wei Fu, and Tim Menzies. What is Wrong with Topic Modeling? And how to Fix it Using Search-Based Software Engineering. *Information and Software Technology*, 2018.

[37] Mika V. Mäntylä, Maëlick Claes, and Umar Farooq. Measuring LDA Topic Stability from Clusters of Replicated Runs. In *International Symposium on Empirical Software Engineering and Measurement (ESEM)*, 2018.

[38] Curtis Carmony, Xunchao Hu, Heng Yin, Abhishek Vasisht Bhaskar, and Mu Zhang. Extract Me If You Can: Abusing PDF Parsers in Malware Detectors. In *Symposium on Network and Distributed System Security (NDSS)*, 2016.

[39] Nida ul Habib Bajwa, Markus Langer, Cornelius J König, and Hannah Honecker. What Might get Published in Management and Applied Psychology? Experimentally Manipulating Implicit Expectations of Reviewers Regarding Hedges. *Scientometrics*, 2019.

[40] Sherry E Sullivan, Yehuda Baruch, and Hazlon Schepmyer. The Why, What, and How of Reviewer Education: A Human Capital Approach. *Journal of Management Education*, 2010.

[41] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and other. Language Models are Few-Shot Learners. *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[42] Jaron Mink, Licheng Luo, Natã M. Barbosa, Olivia Figueira, Yang Wang, and Gang Wang. DeepPhish: Understanding User Trust Towards Artificially Generated Profiles in Online Social Networks. In *USENIX Security Symposium*, 2022.

[43] Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, and other.

Training Language Models to Follow Instructions with Human Feedback. *Computing Research Repository (CoRR)*, 2022.

[44] Ross Taylor, Marcin Kardas, Guillem Cucurull, Thomas Scialom, Anthony Hartshorn, Elvis Saravia, Andrew Poulton, Viktor Kerkez, and Robert Stojnic. Galactica: A Large Language Model for Science. *Computing Research Repository (CoRR)*, 2022.

[45] Thorsten Eisenhofer, Lea Schönherr, Joel Frank, Lars Speckemeier, Dorothea Kolossa, and Thorsten Holz. Dompteur: Taming Audio Adversarial Examples. In *USENIX Security Symposium*, 2021.

[46] Nihar B. Shah. Challenges, Experiments, and Computational Solutions in Peer Review. *Communications of the ACM*, 2022.

[47] Guillaume Cabanac, Cyril Labbé, and Alexander Magazinov. Tortured Phrases: A Dubious Writing Style Emerging in Science. Evidence of Critical Issues Affecting Established Journals. *Computing Research Repository (CoRR)*, 2021.

[48] Michael L. Littman. Collusion Rings Threaten the Integrity of Computer Science Research. *Communications of the ACM*, 2021.

[49] Kevin Leyton-Brown, Mausam, Yatin Nandwani, Hedayat Zarkoob, Chris Cameron, Neil Newman, and Dinesh Raghu. Matching Papers and Reviewers at Large Conferences. *Computing Research Repository (CoRR)*, 2022.

[50] Battista Biggio and Fabio Roli. Wild patterns: Ten Years After the Rise of Adversarial Machine Learning. *Pattern Recognition*, 2018.

[51] Erwin Quiring, Alwin Maier, and Konrad Rieck. Misleading Authorship Attribution of Source Code using Adversarial Learning. In *USENIX Security Symposium*, 2019.

[52] Erwin Quiring, David Klein, Daniel Arp, Martin Johns, and Konrad Rieck. Adversarial Preprocessing: Understanding and Preventing Image-Scaling Attacks in Machine Learning. In *USENIX Security Symposium*, 2020.

[53] Moustafa Alzantot, Yash Sharma, Ahmed Elgohary, Bo-Jhang Ho, Mani B. Srivastava, and Kai-Wei Chang. Generating Natural Language Adversarial Examples. In *Conference on Empirical Methods in Natural Language Processing*, 2018.

[54] Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. HotFlip: White-Box Adversarial Examples for Text Classification. In *Annual Meeting of the Assoc. for Computational Linguistics (ACL)*, 2018.

[55] Mohit Iyyer, John Wieting, Kevin Gimpel, and Luke Zettlemoyer. Adversarial Example Generation with Syntactically Controlled Paraphrase Networks. In *Conference of the North American Chapter of the Assoc. for Computational Linguistics: Human Language Technologies, (NAACL-HLT)*, 2018.

[56] Nicolas Papernot, Patrick D. McDaniel, Ananthram Swami, and Richard E. Harang. Crafting Adversarial Input Sequences for Recurrent Neural Networks. In *IEEE Military Communications Conference (MILCOM)*, 2016.

[57] Dat Tran and Chetan Jaiswal. PDFPhantom: Exploiting PDF Attacks Against Academic Conferences' Paper Submission Process with Counterattack. In *IEEE Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)*, 2019.

[58] Scott C. Deerwester, Susan T. Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman. Indexing by Latent Semantic Analysis. *Journal of the American Society for Information Science*, 1990.

[59] Tomás Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. Distributed Representations of Words and Phrases and their Compositionality. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2013.

A Training Corpus

In the following, we describe how the corpus for the simulated paper-reviewer assignment process is generated. The PC of the *43rd IEEE Symposium on Security and Privacy* conference consists of 165 persons. For each PC member, we construct an archive of papers representative for the person's expertise and interests by crawling their *Google Scholar* profile. In rare cases, this profile is not available and we use the profile from *DBLP computer science bibliography* instead. We sort all papers first by year and then by number of citations to obtain an approximation of the recent research interests. From this list, we remove all papers with no citation and for which we cannot obtain a PDF file (e.g., paywalled files we cannot access). Furthermore, we remove papers that are already used as a target submission. From the remaining list, we select the first 40 papers (if available).

To construct reviewer archives A_r, we randomly sample 20 paper for each reviewer and compile the corpus as the union of these archives. The remaining 20 papers are used to simulate the black-box scenario. Here, we consider different levels of overlaps between 0% (i.e., no overlap between the training data of the surrogates and target system) and 100% (i.e., complete overlap).

B Hyperparameter Selection

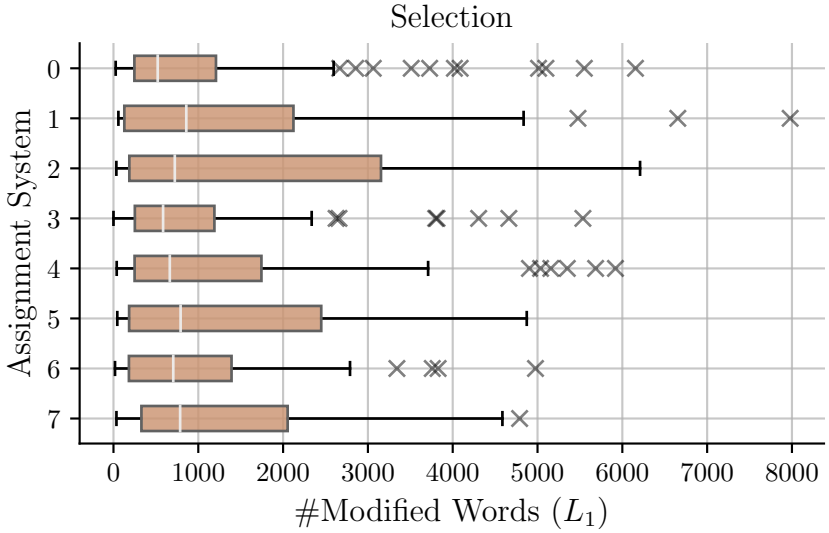
In the following, we describe how we determine the hyperparameters of our attack in the two scenarios.

White-box scenario. We perform a grid search over 100 randomly sampled targets from all three objectives and optimize parameters as a trade-off between attack efficacy and efficiency. To not overfit to a specific model, we train 8 AutoBid systems on different random seeds and randomly select one system per target. Note that an attacker with full-knowledge could also choose parameters that perform best for a specific target. We set the beam width $B = 8 \in \{2^0, \ldots, 2^3\}$, step size $k = 128 \in \{2^5, \ldots, 2^8\}$, number of successors $M = 512 \in \{2^7, \ldots, 2^9\}$, and reviewer window to $\omega = 6 \in \{2^1, \ldots, 2^3\}$ with offset $\upsilon = 3 \in \{0, \ldots, 3\}$. The target rank for rejected reviewer is set to $rank_{\mathbf{x}}^{rej} = 10$ and we consider $\nu = 5000$ words per reviewer. We run the attack for at most $I = 1000$ iterations with at most $S = 8$ transitions between spaces and a target margin of $\Phi = -0.02$.

Black-box scenario. We repeat the grid search and train 8 systems on a surrogate corpus at 70% overlap. We randomly sample 100 targets from all three objectives and assign each a random surrogate system. We set the beam width $B = 4 \in \{2^0, \ldots, 2^3\}$, step size $k = 256 \in \{2^5, \ldots, 2^8\}$, number of successors $M = 128 \in \{2^7, \ldots, 2^9\}$, and reviewer window to $\omega = 2 \in \{2^1, \ldots, 2^3\}$ with offset $\upsilon = 1 \in \{0, \ldots, 3\}$. Finally, to increase the robustness of our attack, we set the margin as $\Phi = -0.16$. All other parameters are the same as before.

C Feature-Space Search

We report the L_1 norms of individual attacks exemplary for the selection objective. We consider 8 different assignment system and sample 100 random targets per system (i. e., 800 attacks in total).



D Generalization of Attack

We empirically evaluate our attack on two conferences with differently sized committees: (a) the *29th USENIX Security Symposium* with 120 reviewers and (b) the *43rd IEEE Symposium on Security and Privacy* conference with a larger committee consisting of 165 reviewers. We simulate the attack for all three objectives and report the aggregated success rate, the median running time, and the median L_1 and L_∞ norm.

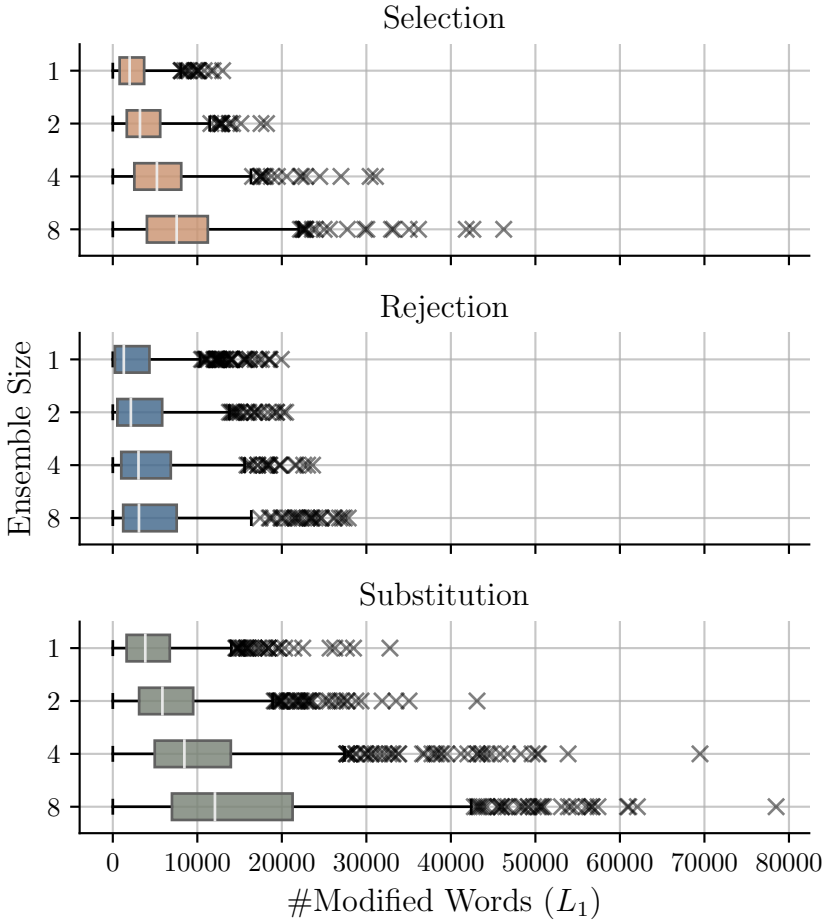
	Success Rate	Running Time	L_1	L_∞
USENIX '20	99.62 %	7m 38s	1033	30
IEEE S&P '22	99.67 %	7m 12s	1115	35

E Scaling of Target Reviewer

We run the attack for different combinations of the number of selected and rejected target reviewers. For each combination, we report the median L_1 norm as well as the success rate over 100 targets.

F Surrogate Ensembles

We report the L_1 norms for the black-box scenario with varying sizes of surrogate ensembles. We report the L_1 over 100 targets for all three objectives.



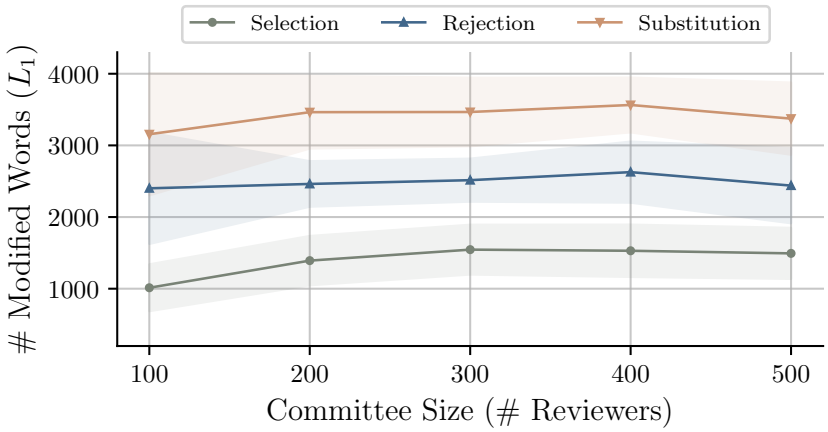
G Overlap

We compare the cross-entropy of reviewer-to-words distributions across models trained on a corpus with different overlaps. We randomly select 10 reviewers and report the mean cross-entropy and standard deviation between 8 models each (i.e., 64 pairs per overlap and reviewer).

#	Overlap			
	0%	30%	70%	100%
1	13.19 ± 0.46	13.13 ± 0.47	13.12 ± 0.37	13.20 ± 0.44
2	12.56 ± 0.29	12.55 ± 0.37	12.64 ± 0.34	12.50 ± 0.29
3	13.58 ± 0.63	13.56 ± 0.56	13.47 ± 0.62	13.52 ± 0.63
4	12.43 ± 0.50	12.29 ± 0.48	12.35 ± 0.54	12.32 ± 0.50
5	13.41 ± 0.51	13.41 ± 0.61	13.50 ± 0.56	13.31 ± 0.66
6	12.84 ± 0.23	12.81 ± 0.21	12.93 ± 0.25	12.90 ± 0.23
7	14.20 ± 0.42	14.28 ± 0.44	14.39 ± 0.48	14.08 ± 0.41
8	13.57 ± 0.46	13.59 ± 0.46	13.55 ± 0.40	13.66 ± 0.42
9	13.44 ± 0.72	13.33 ± 0.68	13.54 ± 0.67	13.44 ± 0.76
10	15.24 ± 0.59	15.08 ± 0.59	15.31 ± 0.66	14.88 ± 0.61

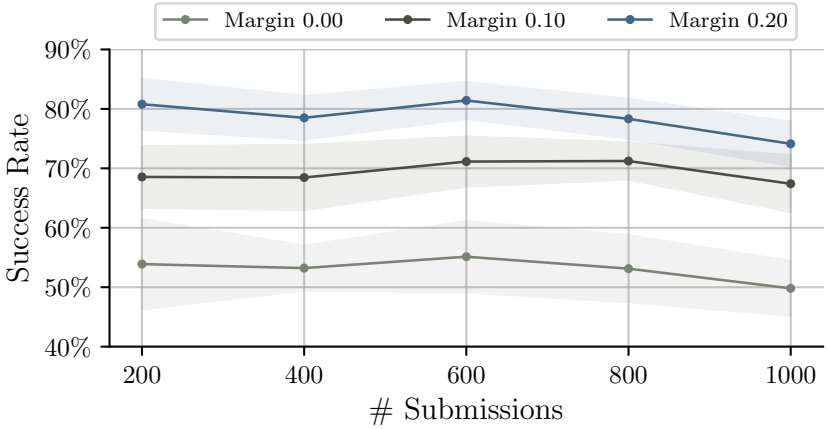
H Committee Size

We simulate the attack with varying sizes of the program committee. For each size, we report the mean number of required modifications over 8 target systems each sampled with a random committee. For each objective, we compute 280 adversarial papers per target system.



I Load balancing

We simulate the attack with varying numbers of concurring submissions between 200 and 1,000. We report the mean success rate over 8 target systems each sampled with a random committee. For each objective, we compute 280 adversarial papers per target system.



J Problem-Space Transformations

Detailed description of problem-space transformations. For the transformations' categorization, see Table 1.

Transformation	Description
Reference addition	Given a database of bibtex records of real papers, this transformation adds papers to the bibliography. It has two options: • *Add unmodified papers.* This option treats the insertion as optimization problem. It tries to find k bibtex records such that the number of added words is maximized. This allows us to maximize the impact of the added papers in the bibliography. • *Add adapted papers.* This option adds r words into a randomly selected bibtex record, which is then added to the bibliography. This transformation allows us to add very specific words which are difficult to add in the normal text in a meaningful way. In the experiments, r is set to 3, i. e., each added bibliography entry has only 3 additional words to avoid suspicion.
Synonym	This transformation replaces words by synonyms using a security domain-specific word embedding [59]. To this end, the word embeddings are computed on a collection of 11,770 security papers (Section 7 presents the dataset). Two options are implemented: • *Add.* Allows adding a word. For each word in the text, it obtains its synonyms. If one of the synonyms is in the list of words that should be added, the synonym is used as replacement for the text word. • *Delete.* Allows removing a word by replacing it with one of its synonyms. The transformation iterates over possible synonyms and only uses a synonym if it has the same part-of-speech (POS) tag as the original word. From the remaining set of synonyms, the transformation randomly chooses a candidate.
Spelling mistake	Inserts a spelling mistake into a word that should be deleted. • *Most common misspelling.* This option tries to find a misspelling from a list of 78 rules for most common misspellings, such as *appearence* instead of *appearance* (rule: ends with -ance), or *basicly* instead of *basically* (rule: ends with -ally). • *Swap or delete.* Swap two adjacent letters or delete a letter in the word. Chooses between both ways randomly. The transformation first tries to find a common misspelling, and if not possible, it applies the swap-or-delete strategy.
Language model	Uses a language model, here OPT [30], to create sentences with the requested words. To create more security-related sentences, we use the corpus from Section 7 consisting of 11,770 security papers to finetune the OPT-350m model. Equipped with this model, the transformer appends new text at the end of the related work or discussion section. To this end, we extract some text before the insertion position and ask the model to complete the text while choosing suitable words from the set of requested words.
Homoglyph	Replaces a single character in a word by a visually identical or similar homoglyph. For instance, we can replace the Latin letter *A* by its Cyrillic equivalent.
Hidden box	Uses the accessibility support with the latex package *accsupp* that allows defining an alternative text over an input text. Only the input text is visible, while the feature extractor processes the alternative text. This allows adding an arbitrary number of words as alternative text. As the input text is not processed, we can also delete words or text in this way. Two options are implemented: • *Add.* Allows adding an arbitrary number of words in the alternative text. This step requires defining the alternative text at least over a visible word that is, however, not extracted as feature afterwards anymore. To reduce side effects, the transformation first checks if the attack requests a word to be reduced. If so, it lays the alternative text over this word. Otherwise, a stop word is chosen that would be ignored in the preprocessing stage anyway. The step thus reduces possible side effects. • *Delete.* Adds an empty alternative text over the input word that needs to be removed, so that the word is not extracted anymore.

Dompteur: Taming Audio Adversarial Examples

Publication Data

Thorsten Eisenhofer, Lea Schönherr, Joel Frank, Lars Speckemeier, Dorothea Kolossa, and Thorsten Holz. Dompteur: Taming Audio Adversarial Examples. In *USENIX Security Symposium*, 2021.

Dompteur: Taming Audio Adversarial Examples

Thorsten Eisenhofer[1], Lea Schönherr[1], Joel Frank[1], Lars Speckemeier[2], Dorothea Kolossa[1], Thorsten Holz[1]

[1] Ruhr University Bochum
[2] University College London

Abstract

Adversarial examples seem to be inevitable. These specifically crafted inputs allow attackers to arbitrarily manipulate machine learning systems. Even worse, they often seem harmless to human observers. In our digital society, this poses a significant threat. For example, *Automatic Speech Recognition* (ASR) systems, which serve as hands-free interfaces to many kinds of systems, can be attacked with inputs incomprehensible for human listeners. The research community has unsuccessfully tried several approaches to tackle this problem.

In this paper we propose a different perspective: We accept the presence of adversarial examples against ASR systems, but we require them to be *perceivable* by human listeners. By applying the principles of *psychoacoustics*, we can remove semantically irrelevant information from the ASR input and train a model that resembles human perception more closely. We implement our idea in a tool named DOMPTEUR* and demonstrate that our augmented system, in contrast to an unmodified baseline, successfully focuses on perceptible ranges of the input signal. This change forces adversarial examples into the audible range, while using minimal computational overhead and preserving benign performance. To evaluate our approach, we construct an *adaptive attacker* that actively tries to avoid our augmentations and demonstrate that adversarial examples from this attacker remain clearly perceivable. Finally, we substantiate our claims by performing a hearing test with crowd-sourced human listeners.

* The French word for *tamer*

1 Introduction

The advent of deep learning has changed our digital society. Starting from simple recommendation techniques [1] or image recognition applications [2], machine-learning systems have evolved to solve and play games on par with humans [3, 4, 5, 6], to predict protein structures [7], identify faces [8], or recognize speech at the level of human listeners [9]. These systems are now virtually ubiquitous and are being granted access to critical and sensitive parts of our daily lives. They serve as our personal assistants [10], unlock our smart homes' doors [11], or drive our autonomous cars [12].

Given these circumstances, the discovery of *adversarial examples* [13] has had a shattering impact. These specifically crafted inputs can completely mislead machine learning-based systems. Mainly studied for image recognition [13], in this work, we study how adversarial examples can affect *Automatic Speech Recognition* (ASR) systems. Preliminary research has already transferred adversarial attacks to the audio domain [14, 15, 16, 17, 18, 19]. The most advanced attacks start from a harmless input signal and change the model's prediction towards a target transcription while simultaneously *hiding* their malicious intent in the inaudible audio spectrum.

To address such attacks, the research community has developed various defense mechanisms [20, 21, 22, 23, 24, 25]. All of the proposed defenses—in the ever-lasting cat-and-mouse game between attackers and defenders—have subsequently been broken [26]. Recently, Shamir et al. [27] even demonstrated that, given certain constraints, we can expect to always find adversarial examples for our models.

Considering these circumstances, we ask the following research question: *When we accept that adversarial examples exist, what else can we do?* We propose a paradigm shift: Instead of preventing *all* adversarial examples, we accept the presence of *some*, but we want them to be audibly changed.

To achieve this shift, we take inspiration from the machine learning community, which sheds a different light on adversarial examples: Illyas et al. [28] interpret the presence of adversarial examples as a disconnection between human expectations and the reality of a mathematical function trained to minimize an objective. We tend to think that machine learning models must learn meaningful features, e. g., a cat has paws. However, this is a human's perspective on what makes a cat a cat. Machine learning systems instead use *any* available feature they can incorporate in their decision

process. Consequently, Illyas et al. demonstrate that image classifiers utilize so-called *brittle features*, which are highly predictive, yet not recognizable by humans.

Recognizing this mismatch between human expectations and the inner workings of machine learning systems, we propose a novel design principle for ASR system inspired by the human auditory system. Our approach is based on two key insights: (i) the human voice frequency is limited to the band ranges of approximately $300 - 5000\,Hz$ [29], while ASR systems are typically trained on 16kHz signals, which range from $0 - 8000\,Hz$, and (ii) audio signal can carry information, inaudible to humans [15]. Given these insights, we modify the ASR system by restricting its access to frequencies and applying psychoacoustic modeling to remove *inaudible* ranges. The effects are twofold: The ASR system can learn a better approximation of the human perception during training (i.e., discarding unnecessary information), while simultaneously, adversaries are forced to place any adversarial perturbation into audible ranges.

We implement these principles in a prototype we call Dompteur. In a series of experiments, we demonstrate that our prototype more closely models the human auditory system. More specifically, we successfully show that our ASR system, in contrast to an unmodified baseline, focuses on perceptible ranges of the audio signal. Following Carlini et al. [30], we depart from the lab settings predominantly studied in prior work: We assume a white-box attacker with real-world capabilities, i.e., we grant them full knowledge of the system and they can introduce an unbounded amount of perturbations. Even under these conditions, we are able to force the attacker to produce adversarial examples with an average of 24.33 dB of added perturbations while remaining accurate for benign inputs. Additionally, we conduct a large scale user study with 355 participants. The study confirms that the adversarial examples constructed for Dompteur are easily distinguishable from benign audio samples and adversarial examples constructed for the baseline system.

In summary, we make the following key contributions:

- *Constructing an augmented ASR.* We utilize our key insights to bring ASR systems in better alignment with human expectations and demonstrate that traditional ASR systems indeed utilize non-audible signals that are not recognizable by humans.

- *Evaluation against adaptive attacker.* We construct a realistic scenario where the attacker can adapt to the augmented system. We show that we successfully force

the attacker into the audible range, causing an average of 24.33 dB added noise to the adversarial examples. We could not find adversarial examples when applying very aggressive filtering; however, this causes a drop in the benign performance.

- *User study.* To study the auditory quality of adversarial examples, we perform a user study with an extensive crowd-sourced listening test. Our results demonstrate that the adversarial examples against our system are significantly more perceptible by humans.

To support further research in this area, we open-source our prototype implementation, our pre-trained models, and audio samples online at github.com/rub-syssec/dompteur.

2 Technical Background

In the following, we discuss the background necessary to understand our augmentation of the ASR system. For this purpose, we briefly introduce the fundamental concepts of ASRs and give an overview of adversarial examples. Since our approach fundamentally relies on psychoacoustic modeling, we also explain masking effects in human perception.

Speech recognition. ASR constitutes the computational core of today's voice interfaces. Given an audio signal, the task of an ASR system is to transcribe any spoken content automatically. For this purpose, traditionally, purely statistical models were used. They now have been replaced by modern systems based on deep learning methods [31, 32, 33], often in the form of hybrid neural/statistical models [34].

In this paper, we consider the open-source toolkit Kaldi [35] as an example of such a modern hybrid system. Its high performance on many benchmark tasks has led to its broad use throughout the research community as well as in commercial products like e. g., Amazon's Alexa [36, 37, 38].

Kaldi, and similar DNN/HMM hybrid systems can generally be described as three-stage systems:

1. *Feature extraction.* For the feature extraction, a frame-wise *discrete Fourier transform* (DFT) is performed on the raw audio data to retrieve a frequency representation of the input signal. The input features of the *Deep Neural Networks* (DNN) are often given by the log-scaled magnitudes of the DFT-transformed signal.

2. *Acoustic model DNN.* The DNN acts as the *acoustic model* of the ASR system. It calculates the probabilities for each of the distinct speech sounds (called *phones*) of its trained language being present in each time frame from its DFT input features. Alternatively, it may compute probabilities, not of phones, but of so-called *clustered tri-phones* or, more generally, of data-driven units termed *senones*.

3. *Decoding.* The output matrix of the DNN is used together with an *hidden Markov model* (HMM)-based language model to find the most likely sequence of words, i. e., the most probable transcription. For this purpose, a dynamic programming algorithm, e.g., Viterbi decoding, is used to search the best path through the underlying HMM. The language model describes the probabilities of word sequences, and the acoustic model output gives the probability of being in each HMM state at each time.

Psychoacoustic modeling. Recent attacks against ASR systems exploit intrinsics of the human auditory system to make adversarial examples less conspicuous [17, 39, 40, 41]. Specifically, these attacks utilize limitations of human perception to hide modifications of the input audio signal within inaudible ranges. We use the same effects for our approach to *remove* inaudible components from the input:

- *Absolute hearing threshold.* Human listeners can only perceive sounds in a limited frequency range, which diminishes with age. Moreover, for each frequency, the sound pressure is important to determine whether the signal component is in the audible range for humans. Measuring the *hearing thresholds*, i. e., the necessary sound pressures for each frequency to be audible in otherwise quiet environments, one can determine the so-called *absolute hearing threshold* as depicted in Figure 1a. Generally speaking, everything above the *absolute hearing thresholds* is perceptible in principle by humans, which is not the case for the area under the curve. As can be seen, much more energy is required for a signal to be perceived at the lower and higher frequencies. Note that the described thresholds only hold for cases where no other sound is present.

- *Frequency masking.* The presence of another sound—a so-called *masking tone*—can change the described *hearing thresholds* to cover a larger area. This *masking effect* of the masking tone depends on its sound pressure and frequency. Figure 1b

shows an example of a 1 kHz masking tone, with its induced changes of the *hearing thresholds* indicated by the dashed line.

- *Temporal masking.* Like frequency masking, temporal masking is also caused by other sounds, but these sounds have the same frequency as the masked tone and are close to it in the time domain, as shown in Figure 1c. Its root cause lies in the fact that the auditory system needs a certain amount of time, in the range of a few hundreds of milliseconds, to recover after processing a higher-energy sound event to be able to perceive a new, less energetic sound. Interestingly, this effect does not only occur at the end of a sound but also, although much less distinct, at the beginning of a sound. This seeming causal contradiction can be explained by the processing of the sound in the human auditory system.

Adversarial examples. Since the seminal papers by Szegedy et al. [13] and Biggio et al. [42], a field of research has formed around adversarial examples. The basic idea is simple: An attacker starts with a valid input to a machine learning system. Then, they add small perturbations to that input with the ultimate goal of changing the resulting prediction (or in our case, the transcription of the ASR).

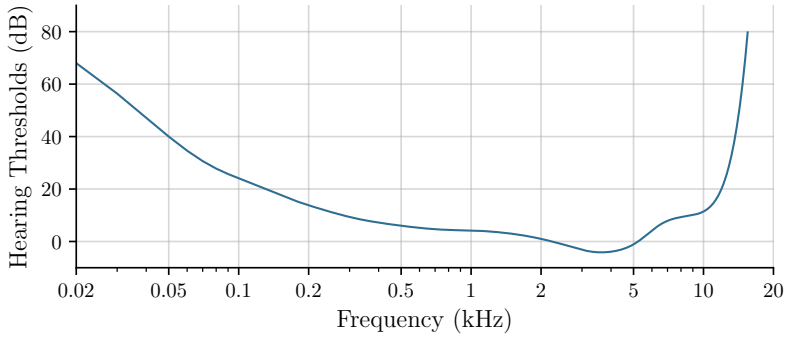
More formally, given a machine learning model f and an input-prediction pair $\langle\, x,\ y \,\rangle$, where $f(x) = y$, we want to find a small perturbation δ s.t.:

$$x' = x + \delta \quad \wedge \quad f(x') \neq f(x).$$

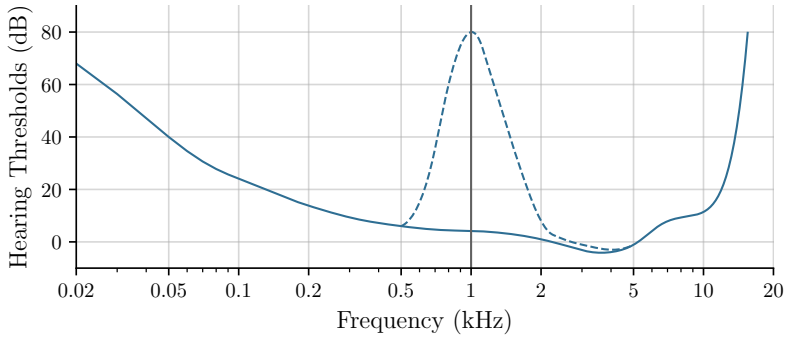
In this paper, we consider a stronger type of attack, a targeted one. This has two reasons: the first is that an untargeted attack in the audio domain is fairly easy to achieve. The second is that a targeted attack provides a far more appealing (and thus, far more threatening) real-life use case for adversarial examples. More formally, the attacker wants to perturb an input phrase x (i.e., an audio signal) with a transcription y (e.g., "Play the Beatles") in such a way that the ASR transcribes an attacker-chosen transcription y' (e.g., "Unlock the front door"). This can be achieved by computing an adversarial example x' based on a small adversarial perturbation δ s.t.:

$$x' = x + \delta \quad \wedge \quad ASR(x') = y' \quad \wedge \quad y \neq y'. \tag{1}$$

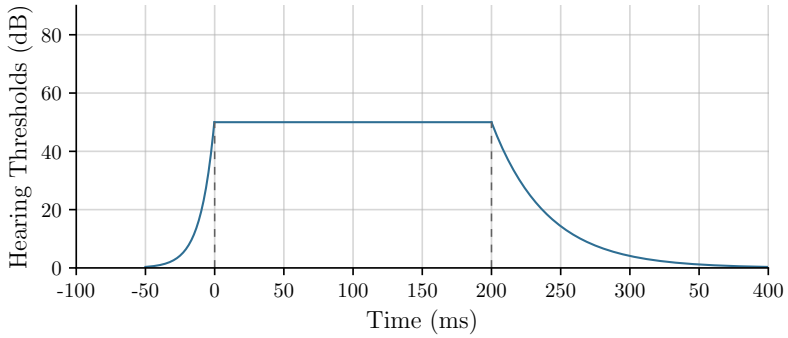
There exist a multitude of techniques for creating such adversarial examples. We use the method introduced by Schönherr et al. [17] for our evaluation in Section 4. The



(a) Absolute Hearing Thresholds



(b) Frequency Masking



(c) Temporal Masking

Figure 1: **Psychoacoustic allows to describe limitations of the human auditory system.** Figure 1a shows the average human hearing threshold in quiet. Figure 1b shows an example of masking, illustrating how a loud tone at 1kHz shifts the hearing thresholds of nearby frequencies and Figure 1c shows how the recovery time of the auditory system after processing a loud signal leads to temporal masking.

method can be divided into three parts: In a first step, attackers choose a fixed output matrix of the DNN to maximize the probability of obtaining their desired transcription y'. As introduced before, this matrix is used in the ASR system's decoding step to obtain the final transcription. They then utilize gradient descent to perturb a starting input x (i. e., an audio signal feed into the DNN), to obtain a new input x', which produces the desired matrix. This approach is generally chosen in white-box attacks [16, 18]. Note that we omit the feature extraction part of the ASR; however, Schönherr et al. have shown that this part can be integrated into the gradient step itself [17]. A third (optional) step is to utilize psychoacoustic hearing thresholds to restrict the added perturbations to inaudible frequency ranges. More technical details can be found in the original publication [17].

3 Modeling the Human Auditory System

We now motivate and explain our design to better align the ASR system with human perception. Our approach is based on the fact that the human auditory system only uses a subset of the information contained in an audio signal to form an understanding of its content. In contrast, ASR systems are not limited to specific input ranges and utilize every available signal – even those *inaudible* for the human auditory system. Consequently, an attacker can easily hide changes within those ranges. Intuitively, the smaller the overlap between these two worlds, the harder it becomes for an attacker to add malicious perturbations that are inaudible to a human listener. This is akin to reducing the attack surface in traditional systems security.

To tackle these issues, we leverage the following two design principles in our approach:

(i) *Removing inaudible parts:* As discussed in Section B, audio signals typically carry information imperceptible to human listeners. Thus, before passing the input to the network, we utilize psychoacoustic modeling to remove these parts.

(ii) *Restricting frequency access:* The human voice frequency range is limited to a band of approximately $300 - 5000\,Hz$ [29]. Thus, we implement a band-pass filter between the feature extraction and model stage (cf. Section B) to restrict the acoustic model to the appropriate frequencies.

3.1 Implementation

In the following, we present an overview of the implementation of our proposed augmentations. We extend the state-of-the-art ASR toolkit Kaldi with our augmentations to build a prototype implementation called Dompteur. Note that our proposed methods are universal and can be applied to *any* ASR system.

Psychoacoustic filtering. Based on the psychoacoustic model of MPEG-1 [43], we use psychoacoustic hearing thresholds to remove parts of the audio that are not perceivable to humans. These thresholds define how dependencies between certain frequencies can mask, i.e., make inaudible, other parts of an audio signal. Intuitively, these parts of the signal should not contribute any information to the recognizer. They do, however, provide space for an attacker to hide adversarial noise.

We compare the absolute values of the complex valued *short-time Fourier transform* (STFT) representation of the audio signal $\mathbf{S}$ with the hearing thresholds $\mathbf{H}$ and define a mask via

$$\mathbf{M}(n,k) = \begin{cases} 0 & \text{if } \mathbf{S}(n,k) \leq \mathbf{H}(n,k) + \Phi \\ 1 & \text{else} \end{cases}, \tag{2}$$

with $n = 0, \ldots, N-1$ and $k = 0, \ldots, K-1$. We use the parameter Φ to control the effect of the hearing thresholds. For $\Phi = 0$, we use the original hearing threshold, for higher values we use a more aggressive filtering, and for smaller values we retain more from the original signal. We explore this in detail in Section 4. We then multiply all values of the signal $\mathbf{S}$ with the mask $\mathbf{M}$

$$\mathbf{T} = \mathbf{S} \odot \mathbf{M}, \tag{3}$$

to obtain the filtered signal $\mathbf{T}$.

Band-pass filter. High and low frequencies are not part of human speech and do not contribute significant information. Yet, they can again provide space for an attacker to hide adversarial noise. For this reason, we remove low and high frequencies of the audio signal in the frequency domain. We apply a band-pass filter after the feature extraction of the system by discarding those frequencies that are smaller or larger than certain

thresholds (the so-called cut-off frequencies). Formally, the filtering can be described via

$$\mathbf{T}(n,k) = 0 \quad \forall\, f_{\max} < k < f_{\min}, \tag{4}$$

where $f_{\max}$ and $f_{\min}$ describe the lower and the upper cut-off frequencies of the band-pass.

3.2 Attacker Model

While some of our augmentations improve the ASR system's overall performance, we are specifically interested in its performance against adversarial perturbations. To achieve any meaningful results, we believe the attacker needs to have *complete* control over the input. Following guidelines recently established by Carlini et al. [30], we embark from theoretical attack vectors towards the definition of a realistic threat model, capturing real-world capabilities of attackers.

The key underlying insight is that the amount of perturbations caused by a real-world attack cannot be limited. This is easy to see: in the worst case, the attacker can always force the target output by replacing the input with the corresponding audio command. Note that this, in turn, implies that we cannot completely prevent adversarial attacks *without* also restricting benign inputs.

We can also not rely on obfuscation. Previous works have successfully shown so-called parameter-stealing attacks, which build an approximation of a black-box system [44, 45, 46, 47, 48]. Since an attacker has full control over this approximated model, they can utilize powerful white-box attacks against it, which transfer to the black-box model.

In summary, we use the following attacker model:

- *Attacker knowledge:* Following Kerckhoffs' principle [49], we consider a *white-box* scenario, where the attacker has complete knowledge of the system, including all model parameters, training data, etc.

- *Attacker goals:* To maximize practical impact, we assume a targeted attack, i. e., the attacker attempts to perturb a given input x to fool a speech recognition system into outputting a false, *attacker-controlled* target transcription y' based on Equation (1).

- *Attacker capabilities:* The attacker is granted complete control over the input, and we explicitly do not restrict them in any way on how δ should be crafted. Note, however, that δ is commonly minimized during computation according to some distance metric. For example, by measuring the *perceived* noise, an attacker might try to minimize the conspicuousness of their attack [17].

We choose this attacker model with the following in mind: We aim to limit the attacker, not in the amount of applied perturbations, but rather confine the nature of perturbations themselves. In particular, we want adversarial perturbations to be clearly perceptible by humans and, thus, strongly perturb the initial input such that the added noise becomes audible for a human listener. In this case, an attack—although still viable—significantly loses its malicious impact in practice.

4 Evaluation

With the help of the following experiments, we empirically verify and assess our proposed approach according to the following three main aspects:

(i) *Benign performance.* The augmentation of the system should impair the performance on benign input as little as possible. We analyze different parameter combinations for the psychoacoustics and our band-pass filter to show that our augmented model retains its practical use.

(ii) *Adaptive attacker.* To analyze the efficacy of the augmented system, we construct and assess its robustness against adversarial examples generated by a strong attacker with white-box access to the system. This attacker is aware of our augmentations and *actively* factors them into the optimization.

(iii) *Listening test.* Finally, we verify the success of our method by a crowd-sourced user study. We conduct a listening test, investigating the quality (i.e., the inconspicuousness) of the adversarial examples computed from the adaptive attacker against the augmented ASR system.

All experiments were performed on a server running Ubuntu 18.04, with 128 GB RAM, an Intel Xeon Gold 6130 CPU, and four Nvidia GeForce RTX 2080 Ti. For our experiments, we use Kaldi in version 5.3 and train the system with the default settings from the *Wall Street Journal* (WSJ) training recipe.

4.1 Metrics

To assess the quality of adversarial examples both in terms of efficacy and inconspicuousness, we use two standard measures.

Word Error Rate (WER). The *Word Error Rate* (WER) is computed based on the Levenshtein distance [50], which describes the *edit distance* between the reference transcription and the ASR output (i.e., the minimum number of edits required to transform the output text of the ASR system into the correct text).

We compute the Levenshtein distance $\mathcal{L}$ as the sum over all substituted words S, inserted words I, and deleted words D:

$$\text{WER} = 100 \cdot \frac{\mathcal{L}}{N} = 100 \cdot \frac{S + D + I}{N},$$

where N is the total number of words of the reference text. The smaller the WER, the fewer errors were made by the ASR system.

To evaluate the efficacy of adversarial examples, we measure the WER between the adversarial target transcription and the output of the ASR system. Thus, a *successful adversarial example* has a WER of 0 %, i. e., fully matching the desired target description y'. Note that the WER can also reach values above 100 %, e. g., when many words are inserted. This can especially happen with unsuccessful adversarial examples, where mostly the original text is transcribed, which leads to many insertions.

Segmental Signal-to-Noise Ratio (SNRseg). The WER can only measure the success of an adversarial example in fooling an ASR system. For a real attack, we are also interested in the (in-) conspicuousness of adversarial examples, i. e., the level of the added perturbations. For this purpose, we quantify the changes that an attacker applies to the audio signal. Specifically, we use the *Signal-to-Noise Ratio* (SNR) to measure the added perturbations. More precisely, we compute the *Segmental Signal-to-Noise Ratio* (SNRseg) [51, 52], a more accurate measure of distortion than the SNR, when signals are aligned [52].

Given the original audio signal $x(t)$ and the adversarial perturbations $\sigma(t)$ defined over the sample index t, the SNRseg can be computed via

$$\text{SNRseg(dB)} = \frac{10}{K} \sum_{k=0}^{K-1} \log_{10} \frac{\sum_{t=Tk}^{Tk+T-1} x^2(t)}{\sum_{t=Tk}^{Tk+T-1} \sigma^2(t)},$$

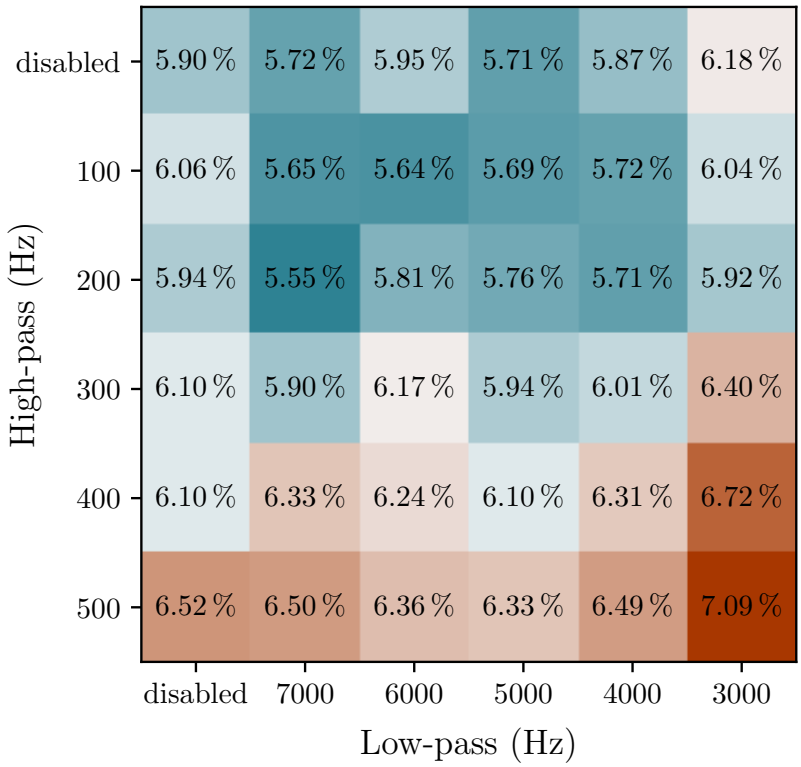


Figure 2: **Word Error Rate (WER) for different band-pass filters.** For each filter, we train three models and report the best accuracy in terms of WER (the lower, the better).

with T being the number of samples in a segment and K the total number of segments. For our experiments, we set the segment length to 16 ms, which corresponds to $T = 256$ samples for a 16 kHz sampling rate.

The *higher* the SNRseg, the *less* noise has been added to the audio signal. Hence, an adversarial example is considered less conspicuous for higher SNRseg values. Note that we use the SNRseg ratio only as an approximation for the perceived noise. We perform a listening test with humans for a realistic assessment and show that the results of the listening test correlate with the reported SNRseg (cf. Section 4.4).

4.2 Benign Performance

Our goal is to preserve accuracy on benign inputs (i. e., non-malicious, unaltered speech) while simultaneously impeding an attacker as much as possible. To retain that accuracy as much as possible, the parameters of the band-pass, and the psychoacoustic filter need to be carefully adjusted. Note that adversarial robustness is generally correlated with

a loss in accuracy for image classification models [53]. Accordingly, we assume that higher adversarial robustness likely incurs a trade-off on benign input performance.

All models in this section are trained with the default settings for the *Wall Street Journal* (WSJ) training recipe of the Kaldi toolkit [35]. The corresponding WSJ-based speech corpus [54] contains approximately 81 hours of training data and consists of uttered sentences from the Wall Street Journal.

We train three models for each configuration and report the WER on the test set for the model with the best performance. For the test set, we use the `eval92` subset consisting of 333 utterances with a combined length of approximately 42 minutes.

Band-pass filtering. The band-pass filter limits the signal's frequency range by removing frequencies below and above certain thresholds. Our goal is to remove parts of the audio that are not used by the human voice. We treat these values as classical hyperparameters and select the best performing combination by grid searching over different cut-off frequencies; for each combination, we train a model from scratch, using the training procedure outlined above. The results are depicted in Figure 2. If we narrow the filtered band (i. e., remove more information), the WER gradually increases and, therefore, worsens the recognizer's accuracy. However, for many cases, even when removing a significant fraction of the signal, the augmented system either achieves comparable results or even surpasses the baseline (WER 5.90 %). In the best case, we reach an improvement by 0.35 % percentage points to a WER of 5.55 % (200 Hz-7000 Hz). This serves as evidence that the unmodified input contains signals that are not needed for transcription. In Section 4.3.3, we further confirm this insight by analyzing models with narrower bands. We hypothesize that incorporating a band-pass filter might generally improve the performance of ASR systems but note that further research on this is needed.

For the remaining experiments, if not indicated otherwise, we use the 200-7000 Hz band-pass.

Psychoacoustic filtering. The band-pass filter allows us to remove high- and low-frequency parts of the signal; however, the attacker can still hide within this band in inaudible ranges. Therefore, we use psychoacoustic filtering as described in Section B to remove these parts in the signal. We evaluate different settings for Φ from Equation (2) – by increasing Φ, we artificially increase the hearing thresholds, resulting in more

Table 1: **Recognition rate of the ASR system on benign input.** We report the performance of an unmodified Kaldi system as well as two variants hardened by our approach. For our model, the scaling factor ϕ is set to 0 and the band-pass filter configured with 200-7000Hz. Note, when feeding standard input to Dompteur, we disable its psychoacoustic filtering capabilities.

	Kaldi	Dompteur	
		w/o band-pass	w/ band-pass
Standard Input	WER 5.90 %	WER 6.20 %	WER 6.33 %
Processed Input	WER 8.74 %	WER 6.50 %	WER 6.10 %

aggressive filtering. We plot the results in Figure 3 for both psychoacoustic filtering and a baseline WER, with and without band-pass, respectively. The WER increases with increasing Φ, i. e., the performance drops if more of the signal is removed, independent of the band-pass filter.

When we use no band-pass filter, the WER increases from 5.90 % (baseline) to 6.50 % for $\Phi = 0\,\text{dB}$, which is equivalent to removing everything below the human hearing thresholds. When we use more aggressive filtering—which results in better adversarial robustness (cf. Section 4.3)—the WER increases up to 8.05 % for $\Phi = 14\,\text{dB}$. Note that the benefits of the band-pass filter remain even in the presence of psychoacoustic filtering and results in improving the WER to 6.10 % ($\Phi = 0\,\text{dB}$) and 7.83 % ($\Phi = 14\,\text{dB}$). We take this as another sign that a band-pass filter might generally be applicable to ASR systems.

Cross-model benign accuracy. Finally, we want to evaluate if Dompteur indeed only uses relevant information. To test this hypothesis, we compare three different models. One completely unaugmented model (i. e., an unmodified version of Kaldi), one model trained with psychoacoustics filtering, and one model trained with both psychoacoustics filtering and a band-pass filter. We feed these models two types of inputs: (i) *standard inputs*, i. e., inputs directly lifted from the WSJ training set, and (ii) *processed inputs*, these inputs are processed by our psychoacoustic filtering. If our intuitive understanding is correct and Dompteur does indeed learn a better model of the human auditory system, it should retain a low WER even when presented with non-

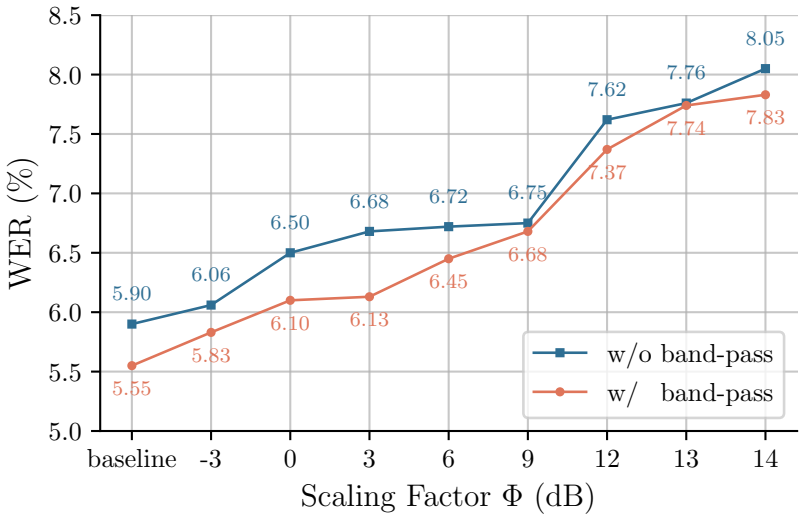


Figure 3: **Recognition rate for psychoacoustic filtering.** For each ϕ we train a model both with and without band-pass filter (200-7000Hz) and report the best accuracy from three repetitions. A negative scaling factor partially retains inaudible ranges. Note that the benefits of the band-pass filter are retrained, even when we incorporate psychoacoustic filtering.

filtered input. Thus, the model has learned to *ignore* unnecessary parts of the input. The results are shown in Table 1 and match our hypothesis: Dompteur's performance only drops slightly (6.10 % → 6.33 %) when presented with unfiltered input or does even improve if the band-pass is disabled (6.50 % → 6.20 %). Kaldi, on the other hand, heavily relies on this information when transcribing audio, increasing its WER by 2.84 percentage point (5.90 % → 8.74 %). Thus, the results further substantiate our intuition that we filter only irrelevant information with our approach.

4.3 Adaptive Attacker

We now want to evaluate how robust Dompteur is against adversarial examples. We construct a strong attacker with complete knowledge about the system and, in particular, our modifications. Ultimately, this allows us to create successfully adversarial examples. However, as inaudible ranges are removed, the attacker is now forced into human-perceptible ranges, and, consequently, the attack loses much of its malicious impact. We provide further support for this claim in Section 4.4 by performing a user study to measure the perceived quality of adversarial examples computed with this attack.

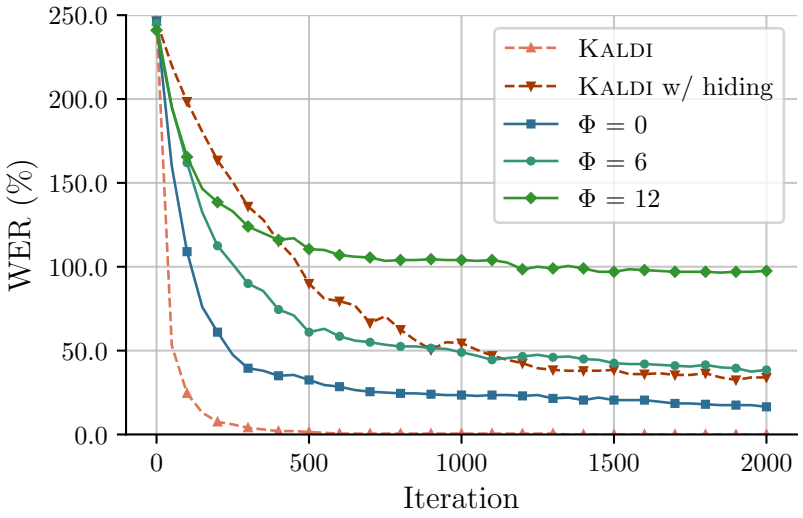


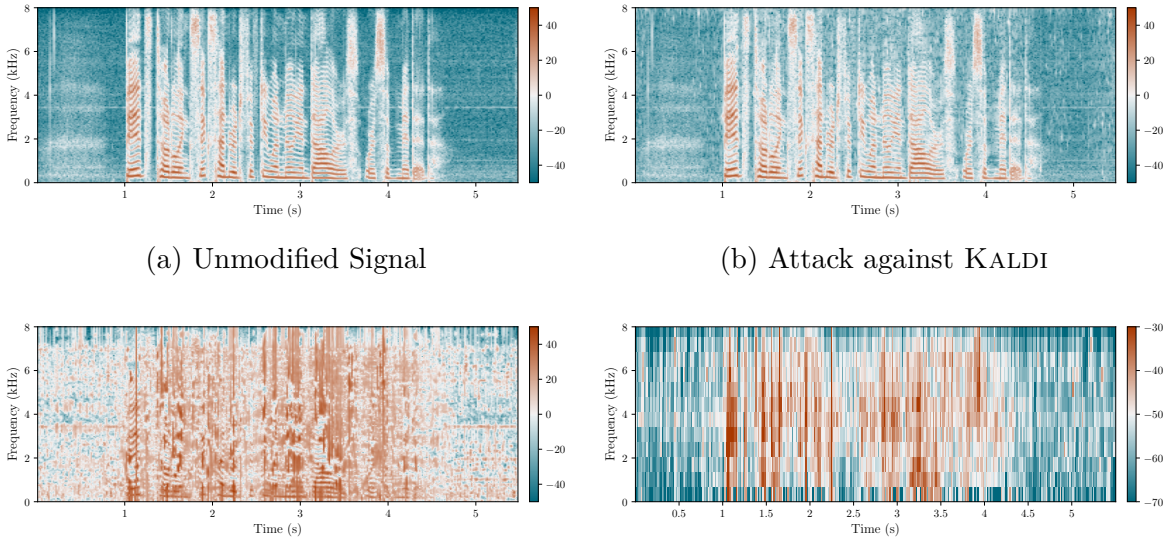
Figure 4: **Progress of attack for computing adversarial examples.** We run the attack against multiple instances of Dompteur with different values of Φ and a 200Hz-7000Hz band-pass filter. The baseline refers to the attack from Schönherr et al. [17] against an unaltered instance of Kaldi. For each attack report the Word Error Rate (WER) for the first 2000 iterations.

Attack. We base our evaluation on the attack by Schönherr et al. [17], which presented a strong attack that works with Kaldi. Recent results show that it is crucial to design adaptive attacks as simple as possible while simultaneously resolving any obstacles for the optimization [55]. To design such an attacker against Dompteur, we need to adjust the attack to consider the augmentations in the optimization. Therefore, we extend the baseline attack against Kaldi to include both the band-pass and psychoacoustic filter into the computation. This allows the attacker to compute gradients for the entire model in a white-box fashion.

More specifically, we extend the gradient descent step s.t. (i) the band-pass filter and (ii) the psychoacoustic filter component back-propagates the gradient respectively.

(i) *Band-pass filter.* For the band-pass filter we remove those frequencies that are smaller and larger than the cut-off frequencies of the band-pass filter. This is also applied to the gradients of the back propagated gradient to ignore changes that will fall into ranges of the removed signal

$$\nabla_{\mathbf{T}(n,k)} = 0 \quad \forall f_{\max} < k < f_{\min}. \tag{5}$$



(a) Unmodified Signal

(b) Attack against Kaldi

(c) Attack against Dompteur ($\Phi = 12$)

(d) Hearing Thresholds

Figure 5: **Spectrograms of adversarial examples.** Figure 5a shows the unmodified signal, Figure 5b depicts the baseline with an adversarial example computed against Kaldi with psychoacoustic hiding, Figure 5c an adversarial example computed with the adaptive attack against Dompteur, and Figure 5d shows the computed hearing thresholds for the adversarial example.

(ii) *Psychoacoustic filter.* The same principle is used for the psychoacoustic filtering, where we use the mask $\mathbf{M}$ to zero out components of the signal that the network will not process

$$\nabla_{\mathbf{S}} = \nabla_{\mathbf{T}} \odot \mathbf{M}. \tag{6}$$

Experimental setup. We evaluate the attack against different versions of Dompteur. Each model uses a $200 - 7000\,Hz$ band-pass filter, and we vary the degrees of the psychoacoustic filtering ($\Phi \in \{0, 3, 6, 9, 12, 13, 14\}$). We compare the results against two baselines to evaluate the inconspicuousness of the created adversarial examples. First, we run the attack of Schönherr et al. without psychoacoustic hiding against an unaltered version Kaldi. Second, we re-enable psychoacoustic hiding and run the original attack against Kaldi, to generate state-of-the-art inaudible adversarial examples. As a sanity check, we also ran the original attack (i. e., with psychoacoustic hiding) against Dompteur. As expected, this attack did not create any adversarial examples since we filter the explicit ranges the attacker wants to utilize.

As a target for all configurations, we select 50 utterances with an approximate length of 5s from the WSJ speech corpus test set `eval92`. The exact subset can be found in Appendix A. We use the same target sentence *send secret financial report* for all samples.

These parameters are chosen such that an attacker needs to introduce ~4.8 phones per second into the target audio, which Schönherr et al. suggests as both effective and efficiently possible [17]. Furthermore, we picked the utterances and target sentence to be *easy* for an attacker in order to decouple the influence on our analysis. Specifically, for these targets the baseline has a very high success rate and low SNRseg (cf. Table 2). Note that the attack is capable of introducing arbitrary target sentences (up to a certain length). In Section 4.3.2, we further analyze the influence of the phone rate, and in particular, the influence of the target utterance and sentence on the SNRseg. We compute adversarial examples for different learning rates and a maximum of 2000 iterations. This number is sufficient for the attack to converge, as shown in Figure 4, where the WER is plotted as a function of the number of iterations.

Results. The main results are summarized in Table 2. We report the average SNRseg over all adversarial examples, the best ($\text{SNRseg}_{\text{max}}$), and the number of successful adversarial examples created.

We evaluate the attack using different learning rates (0.05, 0.10, 0.5, and 1). In our experiments, we observed that while small learning rates generally produce less noisy adversarial examples, they simultaneously get more stuck in local optima. Thus, to simulate an attacker that would run an extensive search and uses the best result we also report the intersection of successful adversarial examples over all learning rates. If success rate is the primary goal, we recommend a higher learning rate.

By increasing Φ, we can successfully force the attacker into audible ranges while also decreasing the attack's success rate. When using very aggressive filtering ($\Phi = 14$), we can prevent the creation of adversarial examples completely, albeit with a hit on the benign WER ($5.55\,\% \rightarrow 7.83\,\%$). Note, however, that we only examined 50 samples

of the test corpus, and other samples might still produce valid adversarial examples. We see that adversarial examples for the augmented systems are more distorted for all configurations compared to the baselines. When using $\Phi \geq 12$, we force a negative SNRseg for all learning rates. For these adversarial examples, the noise (i. e., adversarial perturbations) energy exceeds the energy of the signal. With respect to the baselines,

Table 2: **Number of successful Adversarial Examples (AEs) and Segmental Signal-to-Noise (SNRseg) ratio for the experiments with the adaptive attacker.** We report the numbers for all computed adversarial examples against the augmented models as well as our two baselines (with and without psychoacoustic hiding). As the success rate and SNRseg depend on the learning rate, we combine these in the last row. For this, we select the best (i.e., least noisy) AE for each utterance among the four learning rates. For the SNRseg, we only consider successful AEs. The higher the SNSseg, the *less* noise (i. e., adversarial perturbation) is present in the audio signal. Negative values indicate that the energy of the noise exceeds the energy in the original signal.

		Kaldi		Dompteur						
Learning Rate	Metric	baseline w/o hiding	baseline w/ hiding	$\Phi = 0$	$\Phi = 3$	$\Phi = 6$	$\Phi = 9$	$\Phi = 12$	$\Phi = 13$	$\Phi = 14$
0.05	*AEs*	50/50	17/50	31/50	28/50	10/50	4/50	0/50	0/50	0/50
	SNR	5.80/ 14.44	13.48/ 18.50	6.03/10.63	3.61/ 8.31	1.21/5.53	1.50/ 3.23	—	—	—
0.01	*AEs*	50/50	28/50	38/50	34/50	22/50	10/50	0/50	0/50	0/50
	SNR	2.15/ 10.59	9.36/ 15.81	3.74/ 9.53	0.47/ 6.41	-0.68/3.60	-1.31/ 1.10	—	—	—
0.5	*AEs*	49/50	23/50	48/50	44/50	42/50	20/50	1/50	1/50	0/50
	SNR	-8.54/ -0.02	1.08/ 8.63	-3.78/ 3.24	-6.51/ 0.11	-7.74/-1.47	-8.69/-3.35	-13.56/-13.56	-15.69/-15.69	—
1	*AEs*	50/50	16/50	49/50	50/50	43/50	23/50	1/50	1/50	0/50
	SNR	-13.68/ -5.03	-1.81/ 4.50	-7.44/-0.29	-10.50/-3.00	-10.99/-4.34	-11.98/-6.37	-17.69/-17.69	-11.73/-11.73	—
Best AEs	***AEs***	**50/50**	**37/50**	**50/50**	**50/50**	**46/50**	**27/50**	**2/50**	**2/50**	**0/50**
	SNR	**5.80/ 14.44**	**8.71/ 18.50**	**3.36/10.63**	**0.85/ 8.31**	**-4.71/5.53**	**-7.14/ 3.23**	**-15.62/-13.56**	**-13.71/-11.73**	—

AEs: Successful adversarial examples; *SNR*: SNRseg/$\text{SNRseg}_{\text{max}}$ in dB

the noise energy increases on average by 21.42 dB (without psychoacoustic hiding) and 24.33 dB (with hiding enabled). This means there is, on average, ten times more energy in the adversarial perturbations than in the original audio signal. A graphical illustration can be found in Figure 5, where we plot the power spectra of different adversarial examples compared to the original signal.

4.3.1 Non-Speech Audio Content

The task of an ASR system is to transcribe audio files with spoken content. An attacker, however, might pick other content, i.e., music or ambient noise, to obfuscate his hidden commands. Thus, we additionally evaluated adversarial examples based on audio files containing music and bird sounds. The results are presented in Table 3.

We can repeat our observations from the previous experiment. When we utilize a more aggressive filter, we observe that the perturbation energy of adversarial examples

Table 3: **Number of successful Adversarial Examples (AEs) and mean Segmental Signal-to-Noise (SNRseg) ratio for non-speech audio content.** For each AE, we selected the least noisiest example, from running the attack with learning rates ($\{0.05, 0.1, 0.5, 1.\}$). For the SNRseg we only consider successful AEs and report the difference to the baseline (Kaldi). We highlight the highest loss in bold.

	Birds			Music		
	AEs	SNRseg (dB)	Loss	AEs	SNRseg (dB)	Loss
Kaldi						
w/o hiding	50/50	11.83		45/50	23.26	
w/ hiding	5/50	17.76	(+5.93)	3/50	28.06	(+4.80)
Dompteur						
$\Phi = 0$	50/50	9.58	(-2.25)	50/50	26.35	(+3.09)
$\Phi = 6$	31/50	-2.15	(-13.98)	45/50	16.03	(-7.23)
$\Phi = 12$	5/50	-12.25	(**-24.08**)	3/50	1.94	(**-21.32**)

increases with respect to the baselines by up to 24.08 dB (birds) and 21.32 dB (music). Equally, the attack's general success decreases to 5/50 (birds) and 3/50 (music) successful adversarial examples.

Note that the SNRseg for music samples are in general higher than that of speech and bird files as these samples have a more dynamic range of signal energy. Hence, potentially added adversarial perturbations have a smaller impact on the calculation of the SNRseg. The absolute amount of added perturbations, however, is similar to that of other content. Thus, when listening to the created adversarial examples[†] the samples are similarly distorted. This is further confirmed in Section 4.4 with our listening test.

4.3.2 Target Phone Rate

The success of the attack depends on the ratio between the length of the audio file and the length of the target text, which we refer to as the *target phone rate*. This rate describes how many phones an attacker can hide within one second of audio content.

In our experiments, we used the default ratios recommended by Schönherr et al. However, a better rate might exist for our setting. Therefore, to evaluate the effect of

[†] rub-syssec.github.io/dompteur

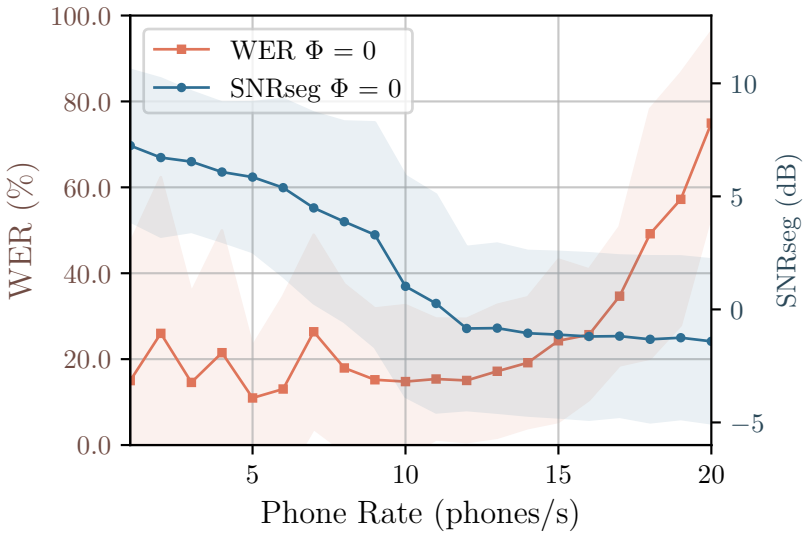


Figure 6: **Word Error Rate (WER) and Segmental Signal-to-Noise (SNRseg) ratio for different phone rates.** We report the mean and std. deviation for adversarial examples computed for targets with varying length.

the target phone rate, we sample target texts of varying lengths from the WSJ corpus and compute adversarial examples for different target phone rates. We pick phone rates ranging from 1 to 20 and run 20 attacks for each of them for at most 1000 iterations, resulting in 400 attacks.

The results in Figure 6 show that, in general, with increasing phone rates, the SNRseg decreases and stagnates for target phone rate beyond 12. This is expected as the attacker tries to hide more phones and, consequently, needs to change the signal more drastically. Thus, we conclude that the default settings are adequate for our setting.

4.3.3 Band-Pass Cut-off Frequencies

So far, we only considered a relatively wide band-pass filter (200-7000 Hz). We also want to investigate other cut-off frequencies. Thus, we disable the psychoacoustic filtering and compute adversarial examples for different models examined in Section 4.2. We run the attack for each band-pass model with 20 speech samples for at most 1000 iterations.

The results are reported in Table 4. We observe that the energy amount of adversarial perturbation remains relatively constant for different filters, which is expected since the

Table 4: **Attack for different cut-off frequencies of the band-pass filter.** We report the number of successful adversarial examples (AEs) and the mean Segmental Signal-to-Noise (SNRseg) ratio. For the SNRseg we only consider successful AEs.

Band-pass	300Hz-7000Hz	300Hz-5000Hz	300Hz-3000Hz	500Hz-7000Hz	500Hz-5000Hz	500Hz-3000Hz
AEs	18/20	18/20	11/20	20/20	17/20	12/20
SNRseg	7.82	7.55	7.27	8.45	7.90	7.39
WER	5.90 %	5.94 %	6.40 %	6.50 %	6.33 %	7.09 %

attacker has complete knowledge of the system. As we narrow the frequency band, the attacker adopts and puts more perturbation within these bands.

Apart from the SNRseg, we also observe a decrease in the attack success, especially for small high cut-off frequencies, with only 11/20 (300-3000 Hz) and 12/20 (500-3000 Hz) successful adversarial examples.

4.4 Listening Tests

Our goal is to make an adversarial attack noticeable by forcing modification to an audio signal into perceptible ranges. We have used the SNRseg as a proxy of the perceived audio quality of generated adversarial examples. However, this value can only give a rough approximation, and we are in general more interested in the judgment of human listeners. Specifically, we are interested to quantify if and to what extent malicious perturbations are audible to human listeners.

Therefore, we have conducted a *Multiple Stimuli with Hidden Reference and Anchor* (MUSHRA) test [56], a commonly used test to assess the quality of audio stimuli. This test allows us to get a ranking of the perceived quality of adversarial examples in comparison to an unmodified reference signal. Based on this measure, we can derive whether a participant 1) could detect any difference between an adversarial example and a clean signal (i.e., whether perturbations are audible) and, 2) obtain a subjective estimate on the amount of perceived perturbations (i.e., poorly rated samples are perceived more noisy).

Table 5: **Regression results for perceived sound quality predicted by different audio stimuli.** The dependent variable is the quality score assigned to each audio stimulus. We trained three different models, one for each data set (speech/music/bird). Each model consists of two steps, with the first step entering the audio stimulus as a predictor and the second step entering type of device as a covariate. All models include the control variables gender, age, and language. All regressions use ordinary least squares. Cluster adjusted standard errors are indicated in parentheses. The R^2 values indicate the percentage of the variance of the perceived sound quality explained by the respective audio stimuli.

	Speech		Music		Bird	
	Step 1	Step 2	Step 1	Step 2	Step 1	Step 2
Audio	-.905**	-.905**	-.871**	-.871**	-.830**	-.830**
stimulus	(.131)	(.131)	(.166)	(.166)	(.171)	(.171)
Device		.030**		.008		.045**
		(.473)		(.597)		(.615)
Controls	Included		Included		Included	
Obs.	4259		4259		4259	
R^2	.820	.821	.760	.761	.690	.692

P-value $< 0.05 =$ *, P-value $< 0.01 =$ **

Study design. In a MUSHRA test, the participants are presented with a set of differently processed audio files, the *audio stimuli*. They are asked to rate the quality of these stimuli on a scale from 0 (bad) to 100 (excellent). To judge whether the participants are able to distinguish between different audio conditions, a MUSHRA test includes two additional stimuli: (i) an unaltered version of the original signal (the so-called *reference*) and (ii) a worst-case version of the signal, which is created by artificial degrading the original stimulus (the so-called *anchor*). In an ideal setting, the reference should be rated best, the anchor worst.

We want to rank the perceived quality of adversarial examples computed against Dompteur and Kaldi. For Dompteur, we select three different versions: each model uses a $200 - 7000\,Hz$ band-pass filter, and we vary the degree of the psychoacoustic filtering ($\Phi \in \{0, 6, 12\}$). For Kaldi, we calculate adversarial examples against the

unaltered system with psychoacoustic hiding enabled (cf. Section B) to compare against state-of-the-art adversarial examples.

As the reference, we use the original utterance, on which the adversarial examples are based. To be a valid comparison, we require the anchor to sound *similar*, yet noisier than the adversarial examples. Otherwise, it could be trivially identified and would not serve as a valid comparison.

Thus, we construct the anchor as follows: For a given set, we scale and sum the noise of each of the three adversarial examples and add this sum to the original stimulus, such that 1) each noise signal contributes the same amount of energy and 2) the SNRseg of the anchor is at least 6dB lower than the SNRseg of any of the adversarial examples in the set.

We have prepared a MUSHRA test with six test sets based on three different audio types: two speech sample sets, two music sample sets, and two sample sets with bird sounds.

These sets were selected among the sets of successful adversarial examples against all four models. For each set, we picked the samples whose adversarial examples produced the highest SNR (i. e., the "cleanest") for the strongest version of Dompteur ($\Phi = 12$). The target text remained the same for all adversarial examples, and in all cases, the attacks were successful within 2000 iterations.

Results. To test our assumptions in the field, we have conducted a large-scale experimental study. The G*Power 3 analysis [57] identified that a sample size of 324 was needed to detect a high effect size of $\eta^2 = .50$ with sufficient power ($1-\beta > .80$) for the main effect of univariate analyses of variance (UNIANOVA) among six experimental conditions and a significance level of $\alpha = .05$.

We used Amazon MTurk to recruit 355 participants ($\mu_{\text{age}} = 41.61$ years, $\sigma_{\text{age}} = 10.96$; 56.60% female). Participants were only allowed to use a computer and no mobile device. However, they were free to use headphones or speakers as long as they indicated what type of listening device was used. To filter individuals who did not meet the technical requirements needed, or did not understand or follow the instructions, we used a control question to exclude all participants who failed to distinguish the anchor from the reference correctly.

In the main part of the experiment, participants were presented with six different audio sets (2 of each: speech/bird/music), each of which contained six audio stim-

uli varying in sound quality. After listening to each sound, they were asked to rank the individual stimulus by its perceived sound quality. After completing of the tasks, participants answered demographic questions, were debriefed (MTurk default), and compensated with 3.00 USD. The participant required on average approximately 20 minutes to finish the test.

In a first step, we first use an UNIANOVA to examine whether there is a significant difference between the six audio stimuli and the perceived sound quality. Our analysis reveals a significant main effect of the audio stimulus on the perceived sound quality, $F(5, 12780) = 8335.610$, $p < .001$, $\eta^2 = .765$. With an alpha level of $> 1\%$ for our p-value and an effect size of $\eta^2 > .5$, our result shows a high experimental significance [58]. Thus, we can conclude that Dompteur indeed forces adversarial perturbations into the perceptible acoustic range of human listeners.

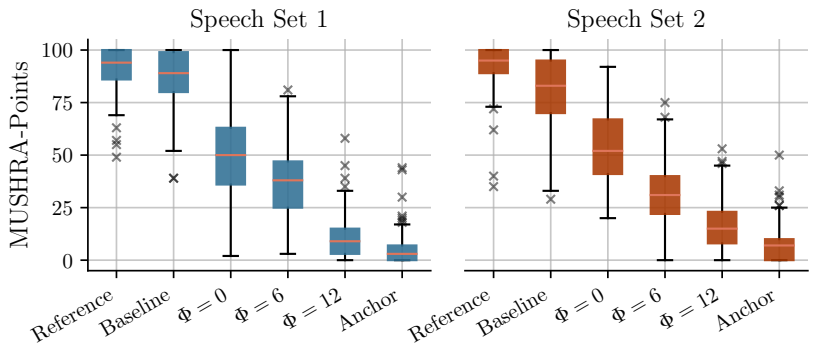
To examine whether the effect remains stable across different audio samples and listening devices, we further conducted multiple regression analyses. We entered the audio stimuli as our main predictors (first step) and the type of device (second step) as covariates for each analysis. Our results remain stable across all audio types. The highest predictive power was found in the *speech* sets, where 82.1% of the variance is explained by our regression model, followed by *music* (76.1%) and *bird* sets (69.2%) (see Table 5 for details). Moreover, we found a small yet significant positive coefficient for the type of device used across all audio types. This finding suggests that headphone users generally indicate higher quality rankings, potentially due to better sound perceptions. The results with listening device *speaker* are presented in Figure 7. Importantly, all results remain stable across the control variables of age, gender, and first language.

In conclusion, the results strongly support our hypothesis that Dompteur forces the attacker into the audible range, making the attack clearly noticeable for human listeners.

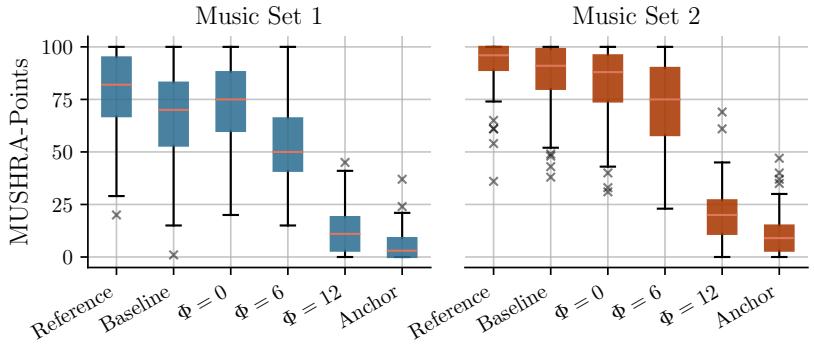
5 Related Work

In this section, we summarize research related to our work, surveying recent attacks and countermeasures.

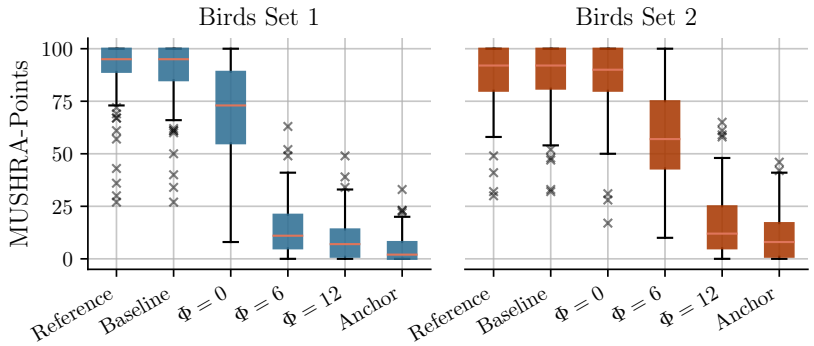
Audio adversarial examples. Carlini and Wagner [59] introduced targeted audio adversarial examples for ASR systems. For the attack, they assume a white-box attacker



(a) Speech



(b) Music



(c) Birds

Figure 7: **Ratings of participants with listening device *speaker*.** In the user study, we tested six audio samples, divided into two samples each of spoken content, music and bird twittering.

and use an optimization-based method to construct general adversarial examples for arbitrary target phrases against the ASR system DeepSpeech [32].

Similarly, Schönherr et al. [17] and Yuan et al. [16] have proposed an attack against the Kaldi [35] toolkit. Both assume a white-box attacker and also use optimization-

based methods to find adversarial examples. Furthermore, the attack from Schönherr et al. [17] can optionally compute adversarial examples that are especially unobtrusive for human listeners.

Alzantot et al. [60] proposed a black-box attack, which does not require knowledge about the model. For this, the authors have used a genetic algorithm to create their adversarial examples for a keyword spotting system. Khare et al. [61] proposed a black-box attack based on evolutionary optimization, and also Taori et al. [62] presented a similar approach in their paper.

Recently, Chen et al. [63] and Schönherr et al. [18] published works where they can calculate over-the-air attacks, where adversarial examples are optimized such that these remain viable if played via a loudspeaker by considering room characteristics.

Aghakhani et al. [64] presented another line of attack, namely a poisoning attack against ASR systems. In contrast to adversarial examples, these are attacks against the training set of a machine learning system, with the target to manipulate the training data s.t a model that is trained with the poisoned data set misclassifies specific inputs.

Abdullah et al. [19] provides a detailed overview of existing attacks in their systemization of knowledge on attacks against speech systems.

Countermeasures. There is a long line of research about countermeasures against adversarial examples in general and especially in the image domain (e. g., [23, 24, 25]), but most of the proposed defenses were shown to be broken once an attacker is aware of the employed mechanism. In fact, due to the difficulty to create robust adversarial example defenses, Carlini et al. proposed guidelines for the evaluation of adversarial robustness. They list all important properties of a successful countermeasure against adversarial examples [30]. Compared to the image domain, defenses against audio adversarial examples remained relatively unnoticed so far. For the audio domain, only a few works have investigated possible countermeasures. Moreover, these tend to focus on specific attacks and not adaptive attackers.

Ma et al. [65] describe how the correlation of audio and video streams can be used to detect adversarial examples for an audiovisual speech recognition task. However, all of these simple approaches—while reasonable in principle—are specifically trained for a defined set of attacks, and hence an attacker can easily leverage that knowledge as demonstrated repeatedly in the image domain [25].

Zeng et al. [66] proposed an approach inspired by multiversion programming. Therefore, the authors combine the output of multiple ASR systems and calculate a *similarity score* between the transcriptions. If these differ too much, the input is assumed to be an adversarial example. The security of this approach relies on the property that current audio adversarial examples do not transfer between systems — an assumption that has been already shown to be wrong in the image domain [45].

Yang et al. [67], also utilize specific properties of the audio domain and uses the temporal dependency of the input signal. For this, they compare the transcription of the whole utterance with a segment-wise transcription of the utterance. In the case of a benign example, both transcriptions should be the same, which will not be the case for an adversarial example. This proved effective against static attacks, and the authors also construct and discussed various adaptive attacks but these were later shown to be insufficient [55].

Besides approaches that aim to harden models against adversarial examples, there is a line of research that focuses on detecting adversarial examples: Liu and Ditzler [68] utilizing quantization error of the activations of the neural network, which appear to be different for adversarial and benign audio examples. Däubener et al. [69] trained neural networks capable of uncertainty quantification to train a classifier on different uncertainty measures to detect adversarial examples as outliers. Even if they trained their classifier on benign examples only, it will most likely not work for any kind of attack, especially those aware of the detection mechanism.

In contrast, our approach does not rely on detection by augmenting the entire system to become more resilient against adversarial examples. The basic principle of this has been discussed as a defense mechanism in the image domain with JPEG compression [70, 71] as well as in the audio domain by Carlini and Wagner [59], Rajaratnam et al. [72], Andronic et al. [73], and Olivier et al. [74]. These approaches, however, were only used as a pre-processing step to remove semantically irrelevant parts from the input and thereby destroy adversarial perturbations added by (static) attackers. In contrast, we aim to train an ASR system that uses the same information set as the human auditory systems. Consequently, adversarial examples computed against this system are also restricted to this set, and an attack cannot be hidden in inaudible ranges. Similar to the referenced approaches, we rely on psychoacoustics and baseband filtering. However, we do not solely employ this as a pre-processing step but train a new system with our augmentation data (i.e., removing imperceptible information from the

training set). This allows us to not simply destroy adversarial perturbations but rather confine the available attack surface.

6 Discussion

We have shown how we can augment an ASR system by utilizing psychoacoustics in conjunction with a band-pass filter to effectively remove semantically irrelevant information from audio signals. This allows us to train a hardened system that is more aligned with human perception.

Model hardening. Our results from Section 4.2 suggest that the hardened models primarily utilize information available within audible ranges. Specifically, we observe that models trained on the unmodified data set appear to use *any* available signals and utilize information *both* from audible and non-audible ranges. This is reflected in the accuracy drop when presented with psychoacoustically filtered input (where only audible ranges are available). In contrast, the augmented model performs comparably well on both types of input. Hence, the model focuses on the perceivable audible ranges and *ignores* the rest.

Robustness of the system. We demonstrated how we can create a more realistic attacker, which actively factors in the augmentations during the calculation of adversarial examples. In this case, however, the attack is forced into the audible range. This makes the attack significant more perceptible — resulting in an average SNRseg drop of up to 24.33 dB for speech samples. These results also transfer to other types of audio content (i.e., music and birds tweeting) and are further confirmed by the listening test conducted in Section 4.4. In summary, the results of these experiments show that an attack is clearly perceivable. Further, we find that the adversarial examples, calculated with the adaptive attack, are easily distinguishable from benign audio files by humans.

Implementation choices. In general, our augmentations can be implemented in the form of low-cost pre-processing steps with no noteworthy performance overhead. Only the model needs to be retrained from scratch. However, the cost of this could—in theory—be partially alleviated by transfer learning. We leave this question as an interesting direction for future research.

Robustness-performance tradeoff. The results of the adaptive attack (cf. Table 2) show that a larger margin Φ leads to stronger robustness. Specifically, for $\Phi = 14$, the attacker was unable to find *any* successful adversarial example in our experiments. However, this incurs an expected robustness-performance trade-off as previous research indicates that adversarial robustness is generally correlated with a loss in accuracy [53].

In the case of our strong white-box attacker, we recommend a margin $\Phi \geq 12$, which result in a degraded system performance by at least 1.82 percentage points in terms of the benign WER. In this case, though, we already granted the attacker many concessions: full access to the model with all parameters, ideal playback (i.e., adversarial examples are fed directly into the recognizer and are not played over-the-air), and an easy target. We chose to study our attacker in this setting as this poses the strongest class of attacks and allows us to gain meaningful insights.

In contrast to white-box attacks, black-box attack don't have direct access to the gradient and for example rely on surrogate models [75] or generative algorithms [76] to construct adversarial examples. Therefore, adversarial examples from these attacks are typically more conspicuous and can even introduce semantic changes such that humans can perceive the hidden transcription if they are made aware of it [75]. Considering our augmentations, we expect that current black-box attacks are able to construct valid adversarial examples against Dompteur. However, we expect these to be significantly more noisy (in comparison to the adaptive attacker) as Dompteur forces modifications to the signal into audible ranges regardless of the underlying attack strategy. Especially in a realistic over-the-air setting, we suspect much higher distortions since the attacker is much more constrained. In such a setting, a smaller Φ might also already suffice. We leave this as an interesting research direction for future work.

Improvement of the attack. The adaptive attack presented in Section 4.3 can successfully compute adversarial examples, except for very aggressive filtering. While Figure 4 clearly shows that the attack has converged, we were still unable to find working adversarial examples. However, other target/input utterance combinations may still exist, for which the attack works and novel attack strategies should be studied.

Forcing semantics into adversarial examples. We have shown how we can force adversarial audio attacks into the audible range. This makes them clearly perceivable. Ultimately, the goal is to push adversarial examples towards the perceptual bound-

ary between original and adversarial message. Intuitively, adversarial examples should require such extensive modification that a *human listener* will perceive the target transcription, i. e., that the adversarial perturbation carries *semantic* meaning. We view our work as a first successful step into that direction and leave the exploration of this strategy as an interesting question for future work.

7 Conclusion

In this work, we proposed a broadly applicable design principle for ASR systems that enables them to resemble the human auditory system more closely. To demonstrate the principle, we implemented a prototype of our approach in a tool called Dompteur. More specifically, we augment Kaldi using psychoacoustic filtering in conjunction with a band-pass filter. In several experiments, we demonstrate that our method renders our system more robust against adversarial examples, while retaining a high accuracy on benign audio input.

We have argued that an attacker can find adversarial examples for any kind of countermeasure, particularly if we assume the attack to have full white-box access to the system. Specifically, we have calculated adversarial examples for Dompteur via an adaptive attack, which leverages the full knowledge of the proposed countermeasures. Although this attack is successful in computing adversarial examples, we show that the attack becomes much less effective. More importantly, we find that adversarial examples are of poor quality, as demonstrated by the SNRseg and our listening test.

In summary, we have taken the first steps towards bridging the gap between human expectations and the reality of ASR systems—hence taming adversarial attacks to a certain extent by robbing them of their stealth abilities.

Acknowledgments

We would like to thank our shepherd Xiaoyu Ji and the anonymous reviewers for their valuable comments and suggestions. We also thank our colleagues Nils Bars, Merlin Chlosta, Sina Däubener, Asja Fischer, Jan Freiwald, Moritz Schlögel, Steffen Zeiler for their feedback and fruitful discussions. This work was supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy – EXC-2092 CaSa – 390781972.

References

[1] Michael J Pazzani and Daniel Billsus. Content-Based Recommendation Systems. In *The Adaptive Web*. Springer, 2007.

[2] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2012.

[3] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, et al. Human-level Control through Deep Reinforcement Learning. *nature*, 2015.

[4] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the Game of Go with Deep Neural Networks and Tree Search. *nature*, 2016.

[5] Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Dębiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, et al. Dota 2 with Large Scale Deep Reinforcement Learning. *Computing Research Repository (CoRR)*, abs/1912.06680, 2019.

[6] Oriol Vinyals, Igor Babuschkin, Wojciech M. Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H. Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster Level in StarCraft II using Multi-Agent Reinforcement Learning. *nature*, 2019.

[7] Andrew W. Senior, Richard Evans, John Jumper, James Kirkpatrick, Laurent Sifre, Tim Green, Chongli Qin, Augustin Žídek, Alexander WR Nelson, Alex Bridgland, et al. Improved Protein Structure Prediction using Potentials from Deep Learning. *nature*, 2020.

[8] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. DeepFace: Closing the Gap to Human-Level Performance in Face Verification. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[9] Wayne Xiong, Jasha Droppo, Xuedong Huang, Frank Seide, Mike Seltzer, Andreas Stolcke, Dong Yu, and Geoffrey Zweig. Achieving Human Parity in Conversational Speech Recognition. *Computing Research Repository (CoRR)*, abs/1610.05256, 2016.

[10] Ashwin Ram, Rohit Prasad, Chandra Khatri, Anu Venkatesh, Raefer Gabriel, Qing Liu, Jeff Nunn, Behnam Hedayatnia, Ming Cheng, Ashish Nagar, et al. Conversational AI: The Science Behind the Alexa Prize. In *Alexa Prize*, 2017.

[11] Lauren Goode. Amazon's Alexa will now lock your door for you (if you have a 'smart' lock). https://www.theverge.com/circuitbreaker/2016/7/28/12305678/amazon-alexa-works-with-august-smart-lock-door-WiFi-bridge. Accessed: 2021-06-02.

[12] Stephen Shankland. Meet Tesla's self-driving car computer and its two AI brains. https://www.cnet.com/news/ meet-tesla-self-driving-car-computer-and-its-two-ai-brains/. Accessed: 2021-06-02.

[13] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing Properties of Neural Networks. In *International Conference on Learning Representations (ICLR)*, 2014.

[14] Liwei Song and Prateek Mittal. POSTER: Inaudible Voice Commands. In *ACM Conference on Computer and Communications Security (CCS)*, 2017.

[15] Guoming Zhang, Chen Yan, Xiaoyu Ji, Tianchen Zhang, Taimin Zhang, and Wenyuan Xu. DolphinAttack: Inaudible Voice Commands. In *ACM Conference on Computer and Communications Security (CCS)*, 2017.

[16] Xuejing Yuan, Yuxuan Chen, Yue Zhao, Yunhui Long, Xiaokang Liu, Kai Chen, Shengzhi Zhang, Heqing Huang, Xiaofeng Wang, and Carl A. Gunter. CommanderSong: A Systematic Approach for Practical Adversarial Voice Recognition. In *USENIX Security Symposium*, 2018.

[17] Lea Schönherr, Katharina Kohls, Steffen Zeiler, Thorsten Holz, and Dorothea Kolossa. Adversarial Attacks Against Automatic Speech Recognition Systems via Psychoacoustic Hiding. In *Symposium on Network and Distributed System Security (NDSS)*, 2019.

[18] Lea Schönherr, Thorsten Eisenhofer, Steffen Zeiler, Thorsten Holz, and Dorothea Kolossa. Imperio: Robust Over-the-Air Adversarial Examples for Automatic Speech Recognition Systems. In *Annual Computer Security Applications Conference (ACSAC)*, 2020.

[19] Hadi Abdullah, Kevin Warren, Vincent Bindschaedler, Nicolas Papernot, and Patrick Traynor. SoK: The Faults in our ASRs: An Overview of Attacks against Automatic Speech Recognition and Speaker Identification Systems. In *IEEE Symposium on Security and Privacy (S&P)*, 2020.

[20] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z. Berkay Celik, and Ananthram Swami. The Limitations of Deep Learning in Adversarial Settings. In *IEEE European Symposium on Security and Privacy (EuroS&P)*, 2015.

[21] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. DeepFool: A Simple and Accurate Method to Fool Deep Neural Networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[22] Nicholas Carlini and David Wagner. Towards Evaluating the Robustness of Neural Networks. In *IEEE Symposium on Security and Privacy (S&P)*, 2017.

[23] Jan Hendrik Metzen, Tim Genewein, Volker Fischer, and Bastian Bischoff. On Detecting Adversarial Perturbations. In *International Conference on Learning Representations (ICLR)*, 2017.

[24] Reuben Feinman, Ryan R. Curtin, Saurabh Shintre, and Andrew B. Gardner. Detecting Adversarial Samples from Artifacts. *Computing Research Repository (CoRR)*, abs/1703.00410, 2017.

[25] Nicholas Carlini and David Wagner. Adversarial Examples are Not Easily Detected: Bypassing Ten Detection Methods. In *ACM Workshop on Artificial Intelligence and Security (AISec)*, 2017.

[26] Justin Gilmer, Ryan P. Adams, Ian Goodfellow, David Andersen, and George E. Dahl. Motivating the Rules of the Game for Adversarial Example Research. *Computing Research Repository (CoRR)*, abs/1807.06732, 2018.

[27] Adi Shamir, Itay Safran, Eyal Ronen, and Orr Dunkelman. A Simple Explanation for the Existence of Adversarial Examples with Small Hamming. *Computing*

Research Repository (CoRR), abs/1901.10861, 2019.

[28] Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Logan Engstrom, Brandon Tran, and Aleksander Madry. Adversarial Examples Are Not Bugs, They Are Features. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[29] Brian B. Monson, Eric J. Hunter, Andrew J. Lotto, and Brad H. Story. The Perceptual Significance of High-frequency Energy in the Human Voice. *Frontiers in Psychology*, 2014.

[30] Nicholas Carlini, Anish Athalye, Nicolas Papernot, Wieland Brendel, Jonas Rauber, Dimitris Tsipras, Ian Goodfellow, and Aleksander Madry. On Evaluating Adversarial Robustness. *Computing Research Repository (CoRR)*, abs/1902.06705, 2019.

[31] Herve A. Bourlard and Nelson Morgan. *Connectionist Speech Recognition: A Hybrid Approach*. Kluwer Press, 1994.

[32] Awni Hannun, Carl Case, Jared Casper, Bryan Catanzaro, Greg Diamos, Erich Elsen, Ryan Prenger, Sanjeev Satheesh, Shubho Sengupta, Adam Coates, and Andrew Y. Ng. Deep Speech: Scaling Up End-to-End Speech Recognition. *Computing Research Repository (CoRR)*, abs/1412.5567, 2014.

[33] Alex Graves and Navdeep Jaitly. Towards End-to-End Speech Recognition with Recurrent Neural Networks. In *International Conference on Machine Learning (ICML)*, 2014.

[34] Jian Kang, Wei-Qiang Zhang, Wei-Wei Liu, Jia Liu, and Michael T. Johnson. Advanced Recurrent Network-Based Hybrid Acoustic Models for Low Resource Speech Recognition. *EURASIP Journal on Audio, Speech, and Music Processing*, 2018.

[35] Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, Jan Silovsky, Georg Stemmer, and Karel Vesely. The Kaldi Speech Recognition Toolkit. In *IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, 2011.

[36] Jun Du, Yan-Hui Tu, Lei Sun, Feng Ma, Hai-Kun Wang, Jia Pan, Cong Liu, Jing-Dong Chen, and Chin-Hui Lee. The USTC-iFlytek System for CHiME-4

Challenge. In *ISCA Workshop on Speech Processing in Everyday Environments (CHiME)*, 2016.

[37] Naoyuki Kanda, Rintaro Ikeshita, Shota Horiguchi, Yusuke Fujita, Kenji Nagamatsu, Xiaofei Wang, Vimal Manohar, Nelson Enrique Yalta Soplin, Matthew Maciejewski, Szu-Jui Chen, et al. The Hitachi/JHU CHiME-5 System: Advances in Speech Recognition for Everyday Home Environments Using Multiple Microphone Arrays. In *ISCA Workshop on Speech Processing in Everyday Environments (CHiME)*, 2018.

[38] Ivan Medennikov, Ivan Sorokin, Aleksei Romanenko, Dmitry Popov, Yuri Khokhlov, Tatiana Prisyach, Nikolay Malkovskii, Vladimir Bataev, Sergei Astapov, Maxim Korenevsky, and Alexander Zatvornitskiy. The STC System for the CHiME 2018 Challenge. In *ISCA Workshop on Speech Processing in Everyday Environments (CHiME)*, 2018.

[39] Yao Qin, Nicholas Carlini, Ian Goodfellow, Garrison Cottrell, and Colin Raffel. Imperceptible, Robust, and Targeted Adversarial Examples for Automatic Speech Recognition. In *International Conference on Machine Learning (ICML)*, 2019.

[40] Hadi Abdullah, Washington Garcia, Christian Peeters, Patrick Traynor, Kevin R. B. Butler, and Joseph Wilson. Practical Hidden Voice Attacks against Speech and Speaker Recognition Systems. In *Symposium on Network and Distributed System Security (NDSS)*, 2019.

[41] Joseph Szurley and J. Zico Kolter. Perceptual Based Adversarial Audio Attacks. *Computing Research Repository (CoRR)*, abs/1906.06355, 2019.

[42] Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim rndi, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. Evasion Attacks against Machine Learning at Test Time. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases (ECML PKDD)*, 2013.

[43] ISO Central Secretary. Information Technology – Coding of Moving Pictures and Associated Audio for Digital Storage Media at Up to 1.5 Mbits/s – Part3: Audio. Standard 11172-3, International Organization for Standardization, 1993.

[44] Andrew Ilyas, Logan Engstrom, Anish Athalye, and Jessy Lin. Black-box Adversarial Attacks with Limited Queries and Information. In *International Conference*

on Machine Learning (ICML), 2018.

[45] Nicolas Papernot, Patrick D. McDaniel, and Ian J. Goodfellow. Transferability in Machine Learning: From Phenomena to Black-Box Attacks using Adversarial Samples. *Computing Research Repository (CoRR)*, abs/1605.07277, 2016.

[46] Florian Tramèr, Fan Zhang, Ari Juels, Michael K. Reiter, and Thomas Ristenpart. Stealing Machine Learning Models via Prediction APIs. In *USENIX Security Symposium*, 2016.

[47] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z. Berkay Celik, and Ananthram Swami. Practical Black-Box Attacks Against Machine Learning. In *ACM Symposium on Information, Computer and Communications Security (ASIACCS)*, 2017.

[48] Binghui Wang and Neil Zhenqiang Gong. Stealing Hyperparameters in Machine Learning. In *IEEE Symposium on Security and Privacy (S&P)*, 2018.

[49] Auguste Kerckhoffs. La Cryptographic Militaire. *Journal des Sciences Militaires*, 1883.

[50] Gonzalo Navarro. A Guided Tour to Approximate String Matching. *ACM Computing Surveys (CSUR)*, 2001.

[51] Stephen Voranl and Connie Sholl. Perception-Based Objective Estimators of Speech. In *IEEE Workshop on Speech Coding for Telecommunications: Speech Coding for Interoperable Global Colmmunications (SCFT)*, 1995.

[52] Wonho Yang. *Enhanced Modified Bark Spectral Distortion (EMBSD): An Objective Speech Quality Measure Based on Audible Distortion and Cognition Model.* PhD thesis, Temple University, 1999.

[53] Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. Robustness May be at Odds with Accuracy. In *International Conference on Learning Representations (ICLR)*, 2019.

[54] Douglas B. Paul and Janet M. Baker. The Design for the Wall Street Journal-Based CSR Corpus. In *Workshop on Speech and Natural Language*, 1992.

[55] Florian Tramer, Nicholas Carlini, Wieland Brendel, and Aleksander Madry. On Adaptive Attacks to Adversarial Example Defenses. In *Advances in Neural Infor-*

mation Processing Systems (NeurIPS), 2020.

[56] Nadja Schinkel-Bielefeld, Netaya Lotze, and Frederik Nagel. Audio Quality Evaluation by Experienced and Inexperienced Listeners. In *International Congress on Acoustics (ICA)*, 2013.

[57] Franz Faul, Edgar Erdfelder, Axel Buchner, and Albert-Georg Lang. Statistical Power Analyses using G* Power 3.1: Tests for Correlation and Regression Analyses. *Behavior Research Methods*, 2009.

[58] John T.E. Richardson. Eta Squared and Partial Eta Squared as Measures of Effect Size in Educational Research. *Educational Research Review*, 2011.

[59] Nicholas Carlini and David Wagner. Audio Adversarial Examples: Targeted Attacks on Speech-to-Text. In *IEEE Deep Learning and Security Workshop (DLS)*, 2018.

[60] Moustafa Alzantot, Bharathan Balaji, and Mani Srivastava. Did you hear that? Adversarial Examples Against Automatic Speech Recognition. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[61] Senthil Mani Shreya Khare, Rahul Aralikatte. Adversarial Black-Box Attacks on Automatic Speech Recognition Systems using Multi-Objective Evolutionary Optimization. In *Conference of the International Speech Communication Association (INTERSPEECH)*, 2019.

[62] Rohan Taori, Amog Kamsetty, Brenton Chu, and Nikita Vemuri. Targeted Adversarial Examples for Black Box Audio Systems. In *IEEE Deep Learning and Security Workshop (DLS)*, 2019.

[63] Tao Chen, Longfei Shangguan, Zhenjiang Li, and Kyle Jamieson. Metamorph: Injecting Inaudible Commands Into Over-the-Air Voice Controlled Systems. In *Symposium on Network and Distributed System Security (NDSS)*, 2020.

[64] Hojjat Aghakhani, Thorsten Eisenhofer, Lea Schönherr, Dorothea Kolossa, Thorsten Holz, Christopher Kruegel, and Giovanni Vigna. VENOMAVE: Clean-Label Poisoning Against Speech Recognition. *Computing Research Repository (CoRR)*, abs/2010.10682, 2020.

[65] Pingchuan Ma, Stavros Petridis, and Maja Pantic. Detecting Adversarial Attacks On Audio-Visual Speech Recognition. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021.

[66] Qiang Zeng, Jianhai Su, Chenglong Fu, Golam Kayas, and Lannan Luo. A Multiversion Programming Inspired Approach to Detecting Audio Adversarial Examples. In *Conference on Dependable Systems and Networks (DSN)*, 2019.

[67] Zhuolin Yang, Bo Li, Pin-Yu Chen, and Dawn Song. Characterizing Audio Adversarial Examples Using Temporal Dependency. In *International Conference on Learning Representations (ICLR)*, 2019.

[68] Heng Liu and Gregory Ditzler. Detecting Adversarial Audio via Activation Quantization Error. In *International Joint Conference on Neural Networks (IJCNN)*, 2020.

[69] Sina Däubener, Lea Schönherr, Asja Fischer, and Dorothea Kolossa. Detecting Adversarial Examples for Speech Recognition via Uncertainty Quantification. In *Conference of the International Speech Communication Association (INTERSPEECH)*, 2020.

[70] Gintare Karolina Dziugaite, Zoubin Ghahramani, and Daniel M. Roy. A Study of the Effect of JPG Compression on Adversarial Images. *Computing Research Repository (CoRR)*, abs/1608.00853, 2016.

[71] Nilaksh Das, Madhuri Shanbhogue, Shang-Tse Chen, Fred Hohman, Siwei Li, Li Chen, Michael E. Kounavis, and Duen Horng Chau. Shield: Fast, Practical Defense and vaccination for Deep Learning using JPEG Compression. In *International Conference on Knowledge Discovery and Data Mining (KDD)*, 2018.

[72] Krishan Rajaratnam, Kunal Shah, and Jugal Kalita. Isolated and Ensemble Audio Preprocessing Methods for Detecting Adversarial Examples against Automatic Speech Recognition. In *Conference on Computational Linguistics and Speech Processing (ROCLING)*, 2018.

[73] Iustina Andronic, Ludwig Kürzinger, Edgar Ricardo Chavez Rosas, Gerhard Rigoll, and Bernhard U Seeber. MP3 Compression to Diminish Adversarial Noise in End-to-End Speech Recognition. In *International Conference on Speech and Computer*, 2020.

[74] Raphael Olivier, Bhiksha Raj, and Muhammad Shah. High-Frequency Adversarial Defense for Speech and Audio. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021.

[75] Yuxuan Chen, Xuejing Yuan, Jiangshan Zhang, Yue Zhao, Shengzhi Zhang, Kai Chen, and XiaoFeng Wang. Devil's Whisper: A General Approach for Physical Adversarial Attacks against Commercial Black-box Speech Recognition Devices. In *USENIX Security Symposium*, 2020.

[76] Tianyu Du, Shouling Ji, Jinfeng Li, Qinchen Gu, Ting Wang, and Raheem Beyah. SirenAttack: Generating Adversarial Audio for End-to-End Acoustic Systems. In *ACM Symposium on Information, Computer and Communications Security (ASIACCS)*, 2020.

A Targets

Target utterances for the experiments with the adaptive attacker. For the experiments we select 50 utterances as target with an approximate length of 5s from the WSJ speech corpus test set `eval92`.

Utterance	Length	Utterance	Length	Utterance	Length
440c0407	5.47s	440c040i	5.07s	440c040j	4.08s
441c0409	4.91s	441c040c	5.57s	441c040l	5.26s
441c040m	5.50s	441c040s	4.58s	441c040y	4.08s
442c0402	5.14s	442c040c	5.69s	442c040d	4.80s
442c040h	4.63s	442c040k	5.37s	442c040w	5.21s
443c0402	5.05s	443c040b	4.69s	443c040c	4.73s
443c040d	5.54s	443c040j	4.61s	443c040l	4.23s
443c040p	4.10s	443c040v	4.82s	443c0417	4.55s
444c0407	4.76s	444c0409	5.20s	444c040i	4.98s
444c040n	5.18s	444c040u	4.37s	444c040w	5.52s
444c040z	4.29s	444c0410	4.16s	445c0409	5.43s
445c040j	4.99s	445c040l	5.64s	445c040w	4.92s
445c040x	4.68s	445c0411	4.34s	446c0402	5.59s
446c040b	4.10s	446c040d	5.18s	446c040e	4.94s
446c040f	4.66s	446c040o	5.33s	446c040p	5.04s
446c040s	5.18s	446c040v	4.07s	447c040g	4.64s
447c040p	5.23s	447c040z	4.68s		

Verifiable and Provably Secure Machine Unlearning

Publication Data

Thorsten Eisenhofer, Doreen Riepel, Varun Chandrasekaran, Esha Ghosh, Olga Ohrimenko, and Nicolas Papernot. Verifiable and Provably Secure Machine Unlearning. *Computing Research Repository (CoRR)*, 2022.

Verifiable and Provably Secure Machine Unlearning

Thorsten Eisenhofer[1,*], Doreen Riepel[1], Varun Chandrasekaran[2,3], Esha Ghosh[2], Olga Ohrimenko[4], Nicolas Papernot[5,6]

[1] Ruhr University Bochum
[2] Microsoft Research
[3] University of Illinois Urbana-Champaign
[4] The University of Melbourne
[5] University of Toronto
[6] Vector Institute

Abstract

Machine unlearning aims to remove points from the training dataset of a machine learning model after training; for example when a user requests their data to be deleted. While many machine unlearning methods have been proposed, none of them enable users to audit the procedure. Furthermore, recent work shows a user is unable to verify if their data was unlearnt from an inspection of the model alone. Rather than reasoning about model parameters, we propose to view verifiable unlearning as a security problem. To this end, we present the first cryptographic definition of verifiable unlearning to formally capture the guarantees of a machine unlearning system. In this framework, the server first computes a proof that the model was trained on a dataset D. Given a user's data point d requested to be deleted, the server updates the model using an unlearning algorithm. It then provides a proof of the correct execution of unlearning and that $d \notin D'$, where D' is the new training dataset. Our framework is generally applicable to different unlearning techniques that we abstract as *admissible functions*. We instantiate the framework, based on cryptographic assumptions, using SNARKs and hash chains. Finally, we implement the protocol for three different unlearning techniques (retraining-based, amnesiac, and optimization-based) to validate its feasibility for linear regression, logistic regression, and neural networks.

[*] Work done while the author was interning at Vector Institute.

1 Introduction

The right to be forgotten entitles individuals to self-determine the possession of their private data and compel its deletion. In practice, this is now mandated by recent regulations like the GDPR [1], CCPA [2], or PIPEDA [3]. Consider the case where a company or service provider collects data from its users. These regulations allow users to request a deletion of their data, and legally compels the company to fulfil the request. However, this can be challenging when the data is used for downstream analyses, *e.g.,* training machine learning (ML) models, where the relationship between model parameters and the data used to obtain them is complex [4]. In particular, ML models are known to memorize information from their training set [5, 6], resulting in a myriad of attacks against the privacy of training data [7, 8].

Thus, techniques have been introduced for *unlearning*: a trained model is updated to remove the influence a training point had on the model's parameters and predictions [9]. Regardless of the particular approach, existing techniques [10, 11, 12, 13, 14, 15, 16] suffer from one critical limitation: they are unable to provide the user with a proof that their data was indeed unlearnt. Put another way, the user is asked to blindly trust that the server executed the unlearning algorithm to remove their data with no ability to verify this. This is problematic because dishonest service providers may falsify unlearning to avoid paying the large computational costs or to maintain model utility [17, 18].

Additionally, verifying that a point is unlearnt is non-trivial *from the user's perspective*. A primary reason is that users (or third-party auditors) cannot determine whether a data point is unlearnt (or not) by comparing the model's predictions or parameters before and after the claimed unlearning. The complex relationship between training data, models' parameters, and their predictions make it difficult to isolate the effects of any training point. In fact, prior work [19, 20] demonstrates that a model's parameters can be identical when trained with or without a data point.

To address these concerns, we propose *a cryptographic approach to verify unlearning*. Rather than trying to verify unlearning by examining changes in the model, we ask the service provider (*i.e.,* the server) to present a cryptographic proof that an *agreed-upon* unlearning process was executed. This leads us to view unlearning as a security problem that we aim to solve with formal guarantees.

In this paper we propose *the first formal security definition* of verifiable machine unlearning. Our framework describes an iteration-based protocol and requires the server to prove that it has honestly updated the model and dataset in each iteration, either due to training with new data or unlearning previously used data. Only then does the user have sufficient guarantees about deletion of their data. Under this definition, we can instantiate protocols using any unlearning technique and any cryptographic primitives that have appropriate security guarantees.

We identified several challenges while developing the framework that we believe are inherent to unlearning.

1. Verifying unlearning cannot be solved by naive one-shot verifiable computation as it requires a user to be able to verify that their data was not re-added at later stages. Hence, the definition has to capture all model updates due to new points added or points being deleted.

2. The relationship between an updated model and the evolving dataset needs to be formally captured for verification. For example, a naive way would be to define this relationship as a re-training function, *i.e.,* the updated model is the result of training on the evolved dataset. This can be viewed as "exact unlearning". However, since other (approximate) unlearning techniques exist, we define this relationship as a set of functions that we call *admissible functions*. This abstraction captures the relationship between models and datasets via initialization, training and unlearning functions.

3. As we observe above, the security definition needs to capture consistency of data during training and unlearning, *and* across model updates and evolving datasets. Though this can be done by passing the whole dataset between the verification steps (training and unlearning) and sending data to the user, we aim to verify consistency in a succinct manner. To this end, we define a strong notion of extractor-based security, capturing that the server must know some underlying dataset in order to compute a valid proof.

Our framework is general and we later demonstrate its applicability to three different unlearning techniques. Notably, none of these have been proved using verifiable computation before. We focus our discussion below on re-training based unlearning, one of

the unlearning techniques, and give an overview of our framework. In this approach, the server retrains a model without using a data point d that needs to be unlearnt.

In our framework, we identify the following guarantees that need to be satisfied: (a) the model was trained from some dataset and (b) the user's data point is not present in this dataset. Thus, the framework has two major components. First, the server computes a *proof of training* whenever data points are added to the model's training data: this establishes that it trained the model on a particular dataset. Second, when a user submits a request to unlearn a specific data point, the server computes a *proof of unlearning*. It proves that the model was updated addressing the request, and additionally provides the user with a *proof* that their data point was indeed unlearnt and not part of the training set. These proofs should also ensure that no data point can be added back to the training set *after* it was unlearnt.

We present a fully instantiated protocol in our framework. This instantiation uses SNARK-based verifiable computation for the proof of model updates induced by training or unlearning and hash chains for the proof of non-membership in a training set since the time unlearning request was made.

Finally, we provide the first implementation of verifiable unlearning based on cryptographic primitives. In particular, we use Spartan [21] for verifiable training and unlearning. Spartan is a transparent SNARK, *i.e.,* it does not require a trusted setup, which is desirable for our purpose. We consider three unlearning techniques: retraining-based unlearning, amnesiac unlearning [14] and optimization-based unlearning [22, 23]. We demonstrate the versatility and scalability of our construction on a variety of binary classification tasks from the PMLB benchmark suite [24], using linear regression, logistic regression, and simple neural networks.

Contributions. We make the following contributions:

- *Formal framework.* We introduce a general framework to construct protocols for verifiable machine unlearning. Our framework is designed to be general enough to capture different unlearning algorithms and secure primitives.

- *Security definition of verifiable machine unlearning.* We then propose a formal security definition of a verifiable machine unlearning scheme. This game-based definition allows one to prove security of their instantiation of the unlearning

protocol. Our framework models verifiable unlearning as a 2-party protocol (executed between the server and users).

- *Instantiation.* We present a fully instantiated protocol in our framework. This construction is based on a generic interface for training and unlearning and thus applicable to any training and unlearning algorithm (as captured by our admissible functions abstraction).

- *Practical implementation.* We implement the protocol's main functionality, study its applicability to three unlearning techniques, different classes of ML models and benchmark datasets. We observe that compared to training verification, verifying unlearning adds a small cost and efficiency of unlearning depends on the unlearning technique.

2 Background

In this section, we discuss the preliminaries needed to understand the contributions of our work.

Notation. Throughout the paper, let λ denote the security parameter. We call a function negligible in λ—denoted by $\mathsf{negl}(\lambda)$—if it is smaller than the inverse of any polynomial for all large enough values of λ. $[m : n]$ denotes the set $\{m, m+1, ..., n\}$ for integers $m < n$. For $m = 1$, we simply write $[n]$. $y \leftarrow \mathsf{M}(x_1, x_2, \ldots)$ denotes that on input $x_1, x_2, \ldots$, the probabilistic algorithm M returns y. An adversary $\mathcal{A}$ is a probabilistic algorithm, and is *efficient* or Probabilistic Polynomial-Time (PPT) if its run-time is bounded by some polynomial in the length of its input. We will use code-based games, where $\Pr[\mathsf{G} \Rightarrow 1]$ denotes the probability that the final output of game G is 1.

2.1 Machine Learning Preliminaries

Supervised machine learning. Supervised machine learning (ML) is the process of learning a parameterized function f_θ (often called a *model*) that is able to predict an output (from the space of outputs $\mathcal{Y}$) given an input (from the space of inputs $\mathcal{X}$), *i.e.,* $f_\theta : \mathcal{X} \to \mathcal{Y}$. Commonly learnt functions include linear regression, logistic regression, and feed-forward neural networks.

The parameters of this function are typically optimized using methods such as stochastic gradient descent (SGD). Let $\theta_{initial}$ be randomly initialized parameters and $D = \{d_1, \ldots, d_n\}$ a set of training data points, where each $d = (d_x, d_y) \in \mathcal{X} \times \mathcal{Y}$. During training we iteratively update parameters as $\theta' := \theta - \eta \nabla_\theta \mathcal{L}(f_\theta(d_x), d_y)$ for points $d \in D$ where $\mathcal{L}$ is a suitably chosen loss function (*e.g.,* cross-entropy loss) and η the *learning rate*.

In SGD, the update is calculated for a randomly chosen input $d \in D$ in each step. In practice, this is often extended to *batches* of data points in order to reduce the variance of each update. Often, multiple passes (called *epochs*) are repeated through the dataset. We can describe the full process of *training* a model m (interchangeably used with θ_m) by

$$\theta_m := \theta_{initial} + \sum_{e \in [E]} \sum_{d \in D} \Delta_{e,d} \,,$$

with E being the number of epochs and $\Delta_{e,d}$ the update on the model's parameter from data point d in epoch e.

Machine unlearning. In machine unlearning the goal is to design algorithms that enable an ML model (specifically, its parameters) to forget the contribution of a (subset of) data point(s). The canonical approach for this is to naively retrain the model from scratch. Hence, removing a data point d^* from a model m with *retraining-based* unlearning can be described as

$$\theta_{m'} := \theta_{initial} + \sum_{e \in [E]} \sum_{d \in D \setminus \{d^*\}} \Delta_{e,d} \,.$$

As the resulting model $\theta_{m'}$ is completely devoid of data point d^* (by construction), this is an example for *exact unlearning* [11, 9, 25, 26], which is desirable but often prohibitively expensive.

More practical unlearning techniques where the contribution of a data point cannot be *completely* removed and the guarantees tolerate some error [12, 27, 28, 15, 16, 13, 14] are commonly referred to as *approximate unlearning*. An example for this is *amnesiac unlearning* [14]. Given a model m, removing a data point d^* with amnesiac unlearning means that we compute

$$\theta_{m'} := \theta_m - \sum_{e \in [E]} \Delta_{e,d^*} \,.$$

In other words, to unlearn data point d^*, we remove the updates to the model's parameters that were *directly* computed on that data point for all training epochs E. Yet, amnesiac unlearning only provides approximate guarantees since updates from unlearnt data points indirectly also influence updates from later points during the iterative nature of the training [28].

Other approaches for approximate unlearning formulate unlearning as an optimization problem [22, 23] (similar to training). In every step, instead of reducing the loss of a data point, we increase it. We refer to this as *optimization-based* unlearning. Formally, we iteratively compute an update Δ_{e,d^*} for the current model that is *subtracted* from its parameters:

$$\theta_{m'} \coloneqq \theta_m - \sum_{e \in [\hat{E}]} \Delta_{e,d^*} \,,$$

where $\hat{E}$ denotes the number of *unlearning epochs* and Δ_{e,d^*} the update from data point d^* in epoch e. For model parameters θ (in epoch e), we define $\Delta_{e,d^*} \coloneqq -\hat{\eta} \nabla_\theta \mathcal{L}(f_\theta(d_x^*), d_y^*)$ with *unlearning rate* $\hat{\eta}$ and loss function $\mathcal{L}$.

2.2 Cryptographic Preliminaries

Collision-resistant hash functions. A function $\mathsf{Hash} : \{0,1\}^n \to \{0,1\}^\kappa$ is collision-resistant if

- It is length-compressing, *i.e.,* $\kappa < n$.
- It is hard to find collisions, *i.e.,* for all PPT adversaries $\mathcal{A}$, and for all security parameters λ,

$$\Pr\left[\begin{matrix}(x_0, x_1) \leftarrow \mathcal{A}(1^\lambda, \mathsf{Hash}) : \\ x_0 \neq x_1 \wedge \mathsf{Hash}(x_0) = \mathsf{Hash}(x_1)\end{matrix}\right] \leq \mathsf{negl}(\lambda) \,.$$

Proof systems. An interactive proof system describes a protocol between a *prover* and a *verifier*, where the prover wants to convince the verifier that some statement ϕ for a given polynomial time decidable relation R is true. The prover holds a witness ω for the statement. We can then express R with a circuit that takes public and private inputs (statement and witness) and returns **true** if the input is in the relation. In this work we are concerned with *non-interactive* proof systems. A Succinct Non-Interactive

Argument of Knowledge (SNARK) allows the prover to non-interactively prove the statement with a short (or succinct) cryptographic proof which can be verified in time sublinear in the size of the statement. We denote a SNARK by Π and define it by the three algorithms $(\Pi.\mathsf{Setup}, \Pi.\mathsf{Prove}, \Pi.\mathsf{Vrfy})$. More formally,

- $\mathsf{pp} \leftarrow \Pi.\mathsf{Setup}(1^\lambda, R)$: The setup algorithm outputs public parameters pp for a polynomial-time decidable relation R.
- $\pi \leftarrow \Pi.\mathsf{Prove}(R, \mathsf{pp}, \phi, \omega)$: The prover algorithm takes as input the pp and $(\phi, \omega) \in R$ and returns an argument π, where ϕ is termed the statement and ω the witness.
- $b \leftarrow \Pi.\mathsf{Vrfy}(R, \mathsf{pp}, \phi, \pi)$: The verification algorithm takes the pp, a statement ϕ and an argument π and returns a bit b, where $b = 1$ indicates success and $b = 0$ indicates failure.

Perfect completeness. Given any true statement, an honest prover should be able to convince an honest verifier. More formally, let $\mathcal{R}$ be a sequence of families of efficiently decidable relations R. For all $R \in \mathcal{R}$ and $(\phi, \omega) \in R$

$$\Pr\left[\Pi.\mathsf{Vrfy}(R, \mathsf{pp}, \phi, \pi) \,\middle|\, \begin{array}{l} \mathsf{pp} \leftarrow \Pi.\mathsf{Setup}(1^\lambda, R); \\ \pi \leftarrow \Pi.\mathsf{Prove}(R, \mathsf{pp}, \phi, \omega) \end{array}\right] = 1\,.$$

Computational soundness. We say that Π is sound if it is not possible to prove a false statement. Let L_R be the language consisting of statements for which there exists corresponding witnesses in R. For a relation $R \sim \mathcal{R}$, we require that for all non-uniform PPT adversaries $\mathcal{A}$

$$\Pr\left[\begin{array}{c} \phi \notin L_R \textbf{ and} \\ \Pi.\mathsf{Vrfy}(R, \mathsf{pp}, \phi, \pi) \end{array} \,\middle|\, \begin{array}{l} \mathsf{pp} \leftarrow \Pi.\mathsf{Setup}(1^\lambda, R); \\ (\phi, \pi) \leftarrow \mathcal{A}(R, \mathsf{pp}) \end{array}\right] \leq \mathsf{negl}(\lambda)\,.$$

We further define the notion of witness extractability or knowledge soundness.

Computational knowledge soundness. Π satisfies computational knowledge soundness if there exists an extractor that can compute a witness whenever the adversary produces a valid argument. More formally, for a relation $R \sim \mathcal{R}$, we require that for all non-uniform PPT adversaries $\mathcal{A}$ there exists a non-uniform PPT extractor $\mathcal{E}$ such that

$$\Pr\left[\begin{array}{c} (\phi, \omega) \notin R \textbf{ and} \\ \Pi.\mathsf{Vrfy}(R, \mathsf{pp}, \phi, \pi) \end{array} \,\middle|\, \begin{array}{l} \mathsf{pp} \leftarrow \Pi.\mathsf{Setup}(1^\lambda, R); \\ ((\phi, \pi); \omega) \leftarrow (\mathcal{A} \| \mathcal{E})(R, \mathsf{pp}) \end{array}\right] \leq \mathsf{negl}(\lambda)\,.$$

We say a SNARK Π is secure if it satisfies perfect completeness, computational soundness and knowledge soundness.

3 Verifiable Machine Unlearning

We consider the following ecosystem: there are many *users*, each of whom has access to a set of data points (or dataset). They share their data with a *server* which uses it to learn an ML model. Users can send requests to either delete or add new data. We focus on ensuring that users can verify that their deletion requests are met.

Threat model. We assume the server is malicious, *i.e.,* it may not execute unlearning. Reasons for this include the server being unwilling to tolerate a degradation in the model's performance after data deletion [17, 18], or pay the computational penalty associated with updating the model [11, 14]. To this end, our focus is to develop a method to verify that the server adheres to users' requests.

Scope and assumptions.

1. *Out-of-band authentication.* We assume that the users have some (out-of-band) mechanism to authenticate to the server. A server will honor unlearning requests from legitimate users only. Since this can be done using standard authentication mechanisms, *e.g.,* digital signatures, we do not model this explicitly. This would prevent a malicious user deleting a point belonging to a different user.

2. *Uniqueness of data points.* We need a mechanism to attribute data points to users: when a user requests their data point to be deleted, it should be clearly identifiable. In practice, multiple users could have the same data point making identification challenging. To resolve this problem, we assume the server prepends a unique identifier to each data point. We refer to such a unique representation as a *data record*.

3. *Sybil data-records.* A malicious server can re-add a data point that was deleted (*e.g.,* by creating a fake user or colluding with an existing user). This would result in a different data record as it would have a different identifier. We note that, in principle, detecting this behaviour is possible but as there are legitimate scenarios where two users may have the same data points, this needs to

be done in an application-specific manner. In scenarios where data points are unique (e.g., in medical settings), this can be trivially detected. For other scenarios, we can define some metric and threshold to determine when data-points d and d are considered to be the same (e.g., when $\text{dist}(d, d) < c$, where c and dist are application-specific). This distance function can then be used to detect misbehaviour by checking if a new point is similar to those previously removed (*e.g.,* using approximate hashing).

Necessity of auditable algorithmic definitions. In the status quo, there is no rigorous way for the user to verify if the model being used is devoid of their data. One naive solution would be to provide the user with the trained parameters (including random seeds) of the model and all the data, and request them to locally re-run the training and compare their model parameters to the server's. Another method could be based on influence techniques [4] to understand if their data contributes to these parameters. Both of these naive solutions suffer from a fundamental problem: it is possible to arrive to same model parameters even if data was deleted. For example, recent work by Shumailov *et al.* [19] and Thudi *et al.* [20] describe how a user's contribution (towards model parameters) can be approximated from other entries in a dataset, rendering such approaches insufficient: the server can claim to have obtained the exact same model parameters from a number of different datasets. Therefore, our approach relies on *proving the execution of an unlearning algorithm*. To prove that a particular record was unlearnt correctly, we further need to trust that the model upon which unlearning is performed was derived from that particular record. Thus, we also require to prove that the training procedure was executed correctly. To capture this formally, we propose a generic interface for *admissible functions* to describe training and unlearning procedures (cf. Appendix 4). These procedures are agreed upon by the users and server beforehand and are part of the public parameters of a protocol.

Desiderata. From the discussion thus far, we unearth two main requirements to achieve our goals.

D1. Given an ML model that was trained on a dataset D and data points are added, we require a *proof of training* to establish that the *updated* model was obtained from the updated dataset by executing a training function all participating parties agreed on.

D2. In a similar fashion, we want to prove the removal of data points. Thus, given an ML model that was trained on a dataset D, machine unlearning mechanisms update this model by (conceptually) *removing* data points from the set D. Then, we require a *proof of unlearning* to establish that the *updated* model was obtained from the updated dataset using an agreed-upon unlearning function. Further, for each removed data point d, we require a proof that d is not part of the (updated) training data D' (*i.e.,* $d \notin D'$), which we denote by a *proof of non-membership*.

To guarantee the absence of deleted data points even after future model updates, it is also necessary to take into account *all* updates to the model, and not only those referring to unlearning (*i.e.,* to prevent a server from re-adding an unlearnt data record).

Additionally, it needs to be ensured that the server uses the most recent model for inference. However, we consider proving and verifying inference an interesting problem which is related to, but also independent of unlearning. We note that there exist several works addressing this problem which we discuss in Section 8.

4 Our Framework

We now present our formal framework for verifiable machine unlearning. This framework defines a generic interface for verifiable machine unlearning capturing desiderata **D1** and **D2** (cf. Section 3) to verify training and unlearning. It allows various instantiations, *e.g.,* using different unlearning techniques or cryptographic primitives. We consider protocols for verifiable machine unlearning that are executed interactively by the two roles introduced in the previous section: a set of users $\mathcal{U}$ and a server S.

Dataset. Let $\mathcal{D}$ be the distribution of data points. Each user $u \in \mathcal{U}$ possesses a set of data points $\widehat{D}_u \sim \mathcal{D}$. At the server side, there is an initially empty dataset $D_0 = \emptyset$ (which is later populated for training an ML model). During the execution of the protocol, users can request to add or delete their data points. Different versions of the server's dataset (as it develops during the execution of the protocol) are denoted by their corresponding index (*i.e.,* $D_0, D_1, \dots$). Recall that we consider data records such that each data point is distinctly identifiable and unique (refer Section 3). Therefore, entries in D_i are tuples of the form $(u, d) \in \mathcal{U} \times \widehat{D}_u$.

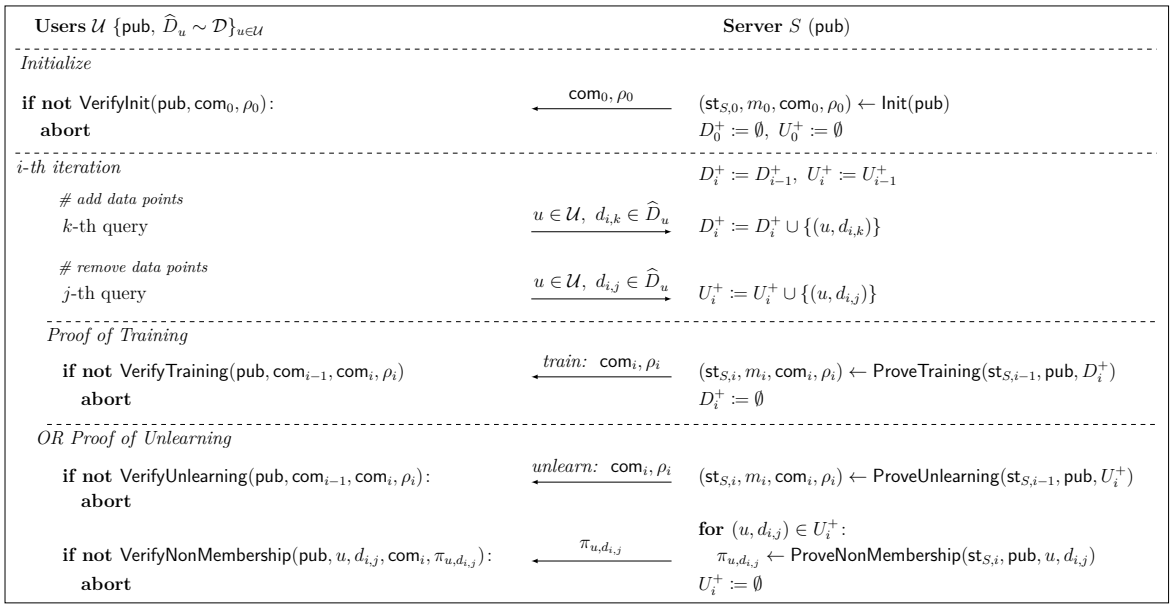


Figure 1: **Unlearning framework.** We describe protocols in this framework based on an admissible functions f. After initialization, execution proceeds in iterations. In the beginning of each iteration i, users $\mathcal{U}$ can issue requests for data to be added or deleted. After this phase, the server S either performs a *proof of training* by adding the requested data records in D_i^+ to the model or a *proof of unlearning* by removing the requested data records in U_i^+. It computes a commitment com_i on the updated model m_i and updated training dataset. Furthermore, the server computes a proof ρ_i that m_i was obtained from this dataset. The users verify this proof and the commitment. In each iteration of unlearning the server additionally creates a *proof of non-membership* for every unlearnt data point conforming to a user that it has complied with a data deletion request. This proof can be verified by the user against com_i.

Admissible functions. We consider triples of functions $f = (f_I, f_T, f_U)$, where f_I is an initialization function, f_T is a training function and f_U is an associated unlearning function. The set of all admissible functions is denoted by $\mathcal{F}$. We let pp_f denote public hyperparameter which are used for initialization. W.l.o.g., we assume the functions to be deterministic. A random seed may be contained in the hyperparameter and stored in the state to derive randomness deterministically. More explicitly:

- $(\mathsf{st}_f, m) := f_I(\mathsf{pp}_f)$: The initialization function f_I takes as input hyperparameter pp_f and outputs the initial state st_f and model m (*e.g.,* its initial weights).

- $(\mathsf{st}_f, m) \coloneqq f_T(\mathsf{st}_f, D^+)$: The training function f_T takes as input the current state st_f and a set D^+ of data points to be added. It outputs the updated state and new model m.
- $(\mathsf{st}_f, m) \coloneqq f_U(\mathsf{st}_f, U^+)$: The unlearning function f_U takes as input the current state st_f and a set U^+ of data points to be deleted. It outputs the updated state and new model m.

These functions allow us to establish an abstraction to track the relation between a model and its underlying dataset; we refer to this as the *conceptual* dataset. If D is the current conceptual dataset, removing data points from U^+ with f_U updates the dataset as $D \coloneqq D \setminus U^+$. We assume that server and users agree on f before executing the protocol (*e.g.,* similar to the TLS handshake protocol).

4.1 Framework Overview

An overview of our framework is depicted in Figure 1. We denote a protocol in this framework by Φ_f, where $f = (f_I, f_T, f_U) \in \mathcal{F}$ is the triplet deployed by the protocol. We then describe the execution of the protocol with two phases (executed in an iterative manner):

P1. Data addition/deletion: Any user can issue an addition/ deletion request to the server at any iteration. The server can batch multiple addition/deletion requests within an iteration. For this, the server stores all requests in intermediate datasets D_i^+ (addition) and U_i^+ (deletion), respectively.

P2. Proof of training (resp. unlearning): At the end of each iteration i, the server updates its dataset by adding (resp. deleting) the data record stored in D_i^+ (resp. U_i^+). For training (resp. unlearning), the server needs to update the model using function f_T (resp. f_U) on all records requested to be added (resp. deleted). It then computes a proof of training (resp. unlearning) to be verified by all users.

For unlearning, the server additionally needs to provide each user who requested a point to be unlearnt with an individual proof that their data record was unlearnt and removed from the server's dataset (*i.e.,* a proof of non-membership). After an update was performed, the dataset D_i^+ (resp. U_i^+) is reset to the empty set.

In the following, we describe the interfaces, including all algorithms, in more detail. The completeness properties of these algorithms are formally presented in Section 4.2.

1. Setup and initialization. A global setup procedure Setup generates public parameters pub, *i.e.,* $\mathsf{pub} \leftarrow \mathsf{Setup}(1^\lambda)$, where λ is the security parameter. We assume that pub additionally includes the admissible functions f and the hyperparameter pp_f. This procedure can be executed either by the server or some external entity, depending on the application. pub is given to all actors. During initialization, the ML model and a state (which captures information needed for subsequent iterations) between the server and users are initialized using the Init and $\mathsf{VerifyInit}$ algorithms. Formally, the algorithms are defined as follows:

Server: $(\mathsf{st}_{S,0}, m_0, \mathsf{com}_0, \rho_0) \leftarrow \mathsf{Init}(\mathsf{pub})$

Init takes as input the public parameters pub and outputs the initial state $\mathsf{st}_{S,0}$, the model m_0, the commitment com_0, and the proof ρ_0. The algorithm runs as follows: the server first suitably initializes model m_0 using initialization algorithm f_I and additional hyperparameter pp_f contained in pub. It stores the resulting state st_f in $\mathsf{st}_{S,0}$. The set of training records is initialized to be empty, *i.e.,* $D_0 \coloneqq \emptyset$. It then commits to m_0 and D_0 with $\mathsf{com}_0 \coloneqq (\mathsf{com}_0^m \| \mathsf{com}_0^D)$. We assume that the commitment to the initial dataset (and all its updates) is computed deterministically from pub using a function Commit, *i.e.,* $\mathsf{com}_0^D \coloneqq \mathsf{Commit}(\mathsf{pub}, D_0)$. Finally, proof ρ_0 attests the initialization of model m_0.

User: $0/1 \leftarrow \mathsf{VerifyInit}(\mathsf{pub}, \mathsf{com}_0, \rho_0)$

$\mathsf{VerifyInit}$ takes as input the public parameters pub, a commitment com_0, and a proof ρ_0. If the verification is successful, it outputs 1. On failure, it outputs 0. All users verify (a) commitment com_0 with $D_0 = \emptyset$, and (b) model initialization m_0 against the proof ρ_0.

2A. Proof of training. In each iteration i in which the server executes a proof of training, it updates the model with any newly added data records using the training function f_T and proves that this was performed correctly. The users verify the resulting proof. Formally, we define the two algorithms as follows:

Server: $(\mathsf{st}_{S,i}, m_i, \mathsf{com}_i, \rho_i) \leftarrow \mathsf{ProveTraining}(\mathsf{st}_{S,i-1}, \mathsf{pub}, D_i^+)$

ProveTraining takes as input the previous state $\mathsf{st}_{S,i-1}$, public parameters pub, the set of new data records D_i^+. It outputs the updated state $\mathsf{st}_{S,i}$, the model m_i, the commitment com_i, and the proof ρ_i. The algorithm runs as follows: the server computes an updated model m_i obtained by executing training function f_T on state st_f and D_i^+. We define the resulting training set of m_i as the union of the previous dataset and all newly added data records, *i.e.,* $D_i \coloneqq D_{i-1} \cup D_i^+$. The server commits to both the model and training data with $\mathsf{com}_i \coloneqq (\mathsf{com}_i^m \| \mathsf{com}_i^D)$. The server then computes the proof ρ_i that (a) model m_i was updated by applying f_T, and (b) training data D_i does not contain any unlearnt record, *i.e.,* $D_i \cap U_i = \emptyset$, where $U_i \coloneqq \bigcup_{k\in[i]} U_k^+$ is the set of all unlearnt data records so far. The proof also attests that (c) the set of unlearnt data records has not changed, *i.e.,* $U_{i-1} = U_i$.

User: $0/1 \leftarrow \mathsf{VerifyTraining}(\mathsf{pub}, \mathsf{com}_{i-1}, \mathsf{com}_i, \rho_i)$

VerifyTraining takes as input the public parameters pub, two commitments com_{i-1}, com_i and a proof ρ_i. It outputs 1 if the verification is successful, and 0 otherwise. All users validate properties (a)-(c) (as described above in ProveTraining) and the update on commitment com_i by verifying the proof ρ_i against the previous commitment com_{i-1} and the new commitment com_i.

2B. Proof of unlearning. In each iteration i in which the server performs a proof of unlearning, it first updates the model using function f_U and thus unlearning all data records requested to be unlearnt. The server proves that this was performed correctly and then computes a proof of non-membership for each individual user who requested to delete their data point, proving that the corresponding records are *absent* in the training data of the updated model. All users verify the correct update. Those users who requested unlearning verify their proof of non-membership. Formally, we define the following four algorithms:

Server: $(\mathsf{st}_{S,i}, m_i, \mathsf{com}_i, \rho_i) \leftarrow \mathsf{ProveUnlearning}(\mathsf{st}_{S,i-1}, \mathsf{pub}, U_i^+)$

ProveUnlearning takes as input the previous state $\mathsf{st}_{S,i-1}$, public parameters pub, and the set U_i^+ which contains data records to be removed. It outputs the updated state $\mathsf{st}_{S,i}$, the updated model m_i, the commitment com_i and the proof ρ_i. Here,

the server unlearns all records collected in U_i^+ and computes the updated m_i by executing function f_U. Thus, conceptually, the new training set of m_i is defined as $D_i \coloneqq D_{i-1} \setminus U_i^+$. Similar to ProveTraining, the server commits to both the model and training data with com_i and computes the proof ρ_i that (a) model m_i was updated by applying f_U, and (b) training data D_i does not contain any unlearnt records, *i.e.,* $D_i \cap U_i = \emptyset^*$, where $U_i \coloneqq U_{i-1} \cup U_i^+$.. The proof also attests that (c) the previous set of unlearnt data records is a subset of the updated set $U_{i-1} \subset U_i$. This ensures that an unlearnt data record is never re-added into the training data.

User: $0/1 \leftarrow \mathsf{VerifyUnlearning}(\mathsf{pub}, \mathsf{com}_{i-1}, \mathsf{com}_i, \rho_i)$

VerifyUnlearning takes as input the public parameters pub, two commitments com_{i-1}, com_i and a proof ρ_i. It outputs 1 if the verification is successful, and 0 otherwise. All users validate properties (a)-(c) (as described above in ProveUnlearning) and the update on commitment com_i by verifying the proof ρ_i against the previous commitment com_{i-1} and the new commitment com_i.

Server: $\pi_{u,d_{i,j}} \leftarrow \mathsf{ProveNonMembership}(\mathsf{st}_{S,i}, \mathsf{pub}, u, d_{i,j})$

ProveNonMembership takes as input the current state $\mathsf{st}_{S,i}$, public parameters pub and a data record $(u, d_{i,j})$. Then, for each record $(u, d_{i,j}) \in U_i^+$, the server computes a proof $\pi_{u,d_{i,j}}$ that this record is not part of the training set of model m_i, *i.e.,* $(u, d_{i,j}) \notin D_i$.

User: $0/1 \leftarrow \mathsf{VerifyNonMembership}(\mathsf{pub}, u, d_{i,j}, \mathsf{com}_i, \pi_{u,d_{i,j}})$

VerifyNonMembership takes as input the public parameters pub, the user identifier u, their unlearnt data point $d_{i,j}$, the commitment of the iteration where $d_{i,j}$ was unlearnt and the proof of unlearning $\pi_{u,d_{i,j}}$. It outputs the result of the verification. The user verifies with both $\pi_{u,d_{i,j}}$ and com_i that $(u, d_{i,j})$ was not part of the training data D_i of model m_i.

Redundant computation. The framework ensures that deleted data records cannot be re-added at a later point. Therefore, it suffices if "an honest majority" verifies the updates. Even if a user does not participate in the protocol after verifying that their

* Note that, depending on the scenario, the equality check between a training and an unlearnt data record can be replaced with a check on whether the records are "close enough" using some distance metric.

data was deleted, verification of updates by an honest majority guarantees correct server behavior.

This can be further optimized: if the users trust a third party (*e.g.,* an auditor), then the $\mathsf{VerifyTraining}$ and the $\mathsf{VerifyUnlearning}$ algorithm can be executed by this entity; the output can be shared with all users (cf. Appendix 7.2 for further discussion).

4.2 Completeness

For completeness, we require that an honest execution of the protocol yields the expected outputs. In particular, if the server is honest, then the users successfully verify the initialization of the model and the proofs for all updates—training and unlearning—performed by the server. Further, a proof of non-membership that was generated for an unlearnt data record is also successfully verified by the corresponding user. In the following, we give a formal definition of computational completeness.

Definition 1 (Completeness)**.** *Let λ be the security parameter. A protocol Φ_f is complete if for all $\mathsf{pub} \leftarrow \mathsf{Setup}(1^\lambda)$, the following properties are satisfied:*

1. *Let $(\mathsf{st}_{S,0}, m_0, \mathsf{com}_0, \rho_0) \leftarrow \mathsf{Init}(\mathsf{pub})$. Then*

$$\Pr[\,\mathsf{VerifyInit}(\mathsf{pub}, \mathsf{com}_0, \rho_0) = 0\,] \leq \mathsf{negl}(\lambda)\ .$$

2. *Let $mode_i \in \{train, unlearn\}$ indicate whether proof of training or proof of unlearning has been performed in iteration i. Let $\mathcal{A}$ be a PPT adversary that outputs a valid sequence of datasets either to be added $\{train\colon\ D_i^+\}$ or to be deleted $\{unlearn\colon\ U_i^+\}$ for all $i \in [\ell]$.*

   For all $i \in [\ell]$, if $mode_i = train$, let $(\mathsf{st}_{S,i}, m_i, \mathsf{com}_i, \rho_i) \leftarrow \mathsf{ProveTraining}(\mathsf{st}_{S,i-1}, \mathsf{pub}, D_i^+)$ and if $mode_i = unlearn$, let $(\mathsf{st}_{S,i}, m_i, \mathsf{com}_i, \rho_i) \leftarrow \mathsf{ProveUnlearning}(\mathsf{st}_{S,i-1}, \mathsf{pub}, U_i^+)$.

   Then for all $mode_i = train$:

$$\Pr[\mathsf{VerifyTraining}(\mathsf{pub}, \mathsf{com}_{i-1}, \mathsf{com}_i, \rho_i) = 0] \leq \mathsf{negl}(\lambda)\ ,$$

   and for all $mode_i = unlearn$:

$$\Pr[\mathsf{VerifyUnlearning}(\mathsf{pub}, \mathsf{com}_{i-1}, \mathsf{com}_i, \rho_i) = 0] \leq \mathsf{negl}(\lambda)\ ,$$

   where validity is defined via the following conditions: $\forall i, j\ s.t.\ i \neq j\colon D_i^+ \cap D_j^+ = \emptyset$ and $\forall i, j\ s.t.\ j < i\colon D_i^+ \cap U_j^+ = \emptyset$.

$\underline{\mathsf{GameUnlearn}_{\mathcal{A},\mathcal{E},\Phi_f,\mathcal{D}}(1^\lambda)}$

$\mathsf{pub} \leftarrow \mathsf{Setup}(1^\lambda)$

$(k, (u,d), \pi_{u,d}, \{mode_i\text{:}\ \ \mathsf{com}_i, \rho_i\}_{i\in[0:\ell]};\ \{D_i\}_{i\in[0:\ell]}) \leftarrow (\mathcal{A}\|\mathcal{E})(\mathsf{pub}, \mathsf{aux})$

 # Pre-processing

$U_k^+ := D_{k-1} \setminus D_k$

Parse com_i as $(\mathsf{com}_i^m \| \mathsf{com}_i^D)$ $\forall i \in [0:\ell]$

 # Evaluate winning condition

if $\mathsf{Commit}(\mathsf{pub}, D_i) = \mathsf{com}_i^D$ $\forall i \in [0:\ell]$ *# Datasets*

  and $\mathsf{VerifyInit}(\mathsf{pub}, \mathsf{com}_0, \rho_0)$ *# Initialization*

  and $\mathsf{VerifyTraining}(\mathsf{pub}, \mathsf{com}_{i-1}, \mathsf{com}_i, \rho_i)$ $\forall i : mode_i = train$ *# Training*

  and $\mathsf{VerifyUnlearning}(\mathsf{pub}, \mathsf{com}_{i-1}, \mathsf{com}_i, \rho_i)$ $\forall i : mode_i = unlearn$ *# Unlearning*

  and $\mathsf{VerifyNonMembership}(\mathsf{pub}, u, d, \mathsf{com}_k, \pi_{u,d})$ *# Non-Membership*

  and $k < \ell$ **and** $(u,d) \in U_k^+$ **and** $(u,d) \in D_\ell$: *# Point unlearnt & re-added*

   return 1

return 0

Figure 2: **Security game.** We define the security of an unlearning protocol Φ_f in terms of game $\mathsf{GameUnlearn}$. The notation $(\mathcal{A}\|\mathcal{E})$ denotes that both algorithms are run on the same input and random coins and assigning their results to variables before resp. after the semicolon. Input aux refers to auxiliary input.

3. *For all* $i \in [\ell]$ *s. t.* $mode_i = unlearn$*: for all* $(u,d) \in U_i^+$*, let* $\pi_{u,d} \leftarrow \mathsf{ProveNonMembership}(\mathsf{st}_{S,i}, \mathsf{pub}, u, d)$*, then*

$$\Pr[\mathsf{VerifyNonMembership}(\mathsf{pub}, u, d, \mathsf{com}_i, \pi_{u,d}) = 0] \leq \mathsf{negl}(\lambda)\ .$$

We require computational completeness here to allow for a wide range of instantiations. For example, an instantiation that works on hash values of data records cannot achieve perfect completeness because of hash collisions. By allowing for computational completeness, however, we only require that it should be hard for a PPT adversary to find such collisions (*i.e.,* with a negligible probability).

4.3 Security Definition

In this section, we present the security definition for unlearning in a game $\mathsf{GameUnlearn}$. The adversary, described by a probabilistic algorithm $\mathcal{A}$, takes the role of the server. Intuitively, the definition captures that a malicious server cannot add (and train on)

a data point that a user requested to delete in a previous iteration. However, we go a step further and let the server choose *which* data records it will add or delete. Thus, the goal of the adversary is to find a data record for which it can prove deletion, but which is re-added in some subsequent iteration.

In order to capture this setting, we propose an *extractability-based* security definition as used in the context of hash functions or SNARKs [29, 30, 31]. That is, we require the existence of an extractor, modelling that the adversary cannot forge a transcript without knowing the underlying datasets. Thus, the adversary in our game has to provide the protocol outputs (*i.e.,* the commitments and proofs), whereas the extractor outputs the corresponding inputs that the adversary used (*i.e.,* the underlying datasets). We give a formal description of game $\mathsf{GameUnlearn}$ in Figure 2, which is divided into the following two stages:

S1. Simulation. The game draws the public parameters pub using Setup and runs the adversary $\mathcal{A}$ on input pub. The extractor $\mathcal{E}$ is run on the same input and random coins. Additionally, we provide benign auxiliary input aux, which captures any extra information that the adversary may have (possibly obtained prior to the start of executing the current protocol). At some point, $\mathcal{A}$ will terminate and output a sequence of tuples $(k, (u, d), \pi_{u,d}, \{mode_i\colon\ \mathsf{com}_i, \rho_i\}_{i\in[0,\ell]})$ for some $\ell \in \mathbb{N}$, where (u, d) is a data record that was proved to be deleted in the k-th iteration, and $mode_i \in \{train, unlearn\}$. At the same time, the extractor outputs a sequence of datasets $(D_0, \ldots, D_\ell)$.

S2. Finalize. After the adversary has terminated, the game uses the extractor's output to compute the set of data points unlearnt in the k-th iteration based on the datasets D_k and D_{k-1}. Recall that the commitment in the framework consists of two parts com_i^m and com_i^D, where we need the second part for verification. The game checks for the following conditions: (a) com_i^D was obtained from D_i, (b) the initial proof ρ_0 verifies for the initial commitment com_0, (c) each proof of training ρ_i verifies for commitments com_{i-1} and com_i, (d) each proof of unlearning ρ_i verifies for commitments com_{i-1} and com_i, (e) the proof of non-membership $\pi_{u,d}$ verifies for (u, d) and com_k, (f) $k < \ell$ and (u, d) was unlearnt in iteration k and re-added in iteration ℓ. If all these properties are satisfied, then the game outputs 1 and $\mathcal{A}$ wins.

We summarize this in the following definition.

Definition 2 (Unlearning)**.** *Let λ be the security parameter and consider game* GameUnlearn *in Figure 2. Protocol Φ_f for data distribution $\mathcal{D}$ is unlearning-secure if for all PPT adversaries $\mathcal{A}$ there exists an extractor $\mathcal{E}$ such that for all benign auxiliary inputs* aux*:*

$$\Pr[\mathsf{GameUnlearn}_{\mathcal{A},\mathcal{E},\Phi_f,\mathcal{D}}(1^\lambda) \Rightarrow 1] \leq \mathsf{negl}(\lambda)\,.$$

5 Instantiation

Our framework defines a general interface to construct protocols for verifiable unlearning. In the following, we present such a protocol based on cryptographic building blocks. We use SNARKs and hash functions where the execution of the admissible functions is proved inside the SNARK and data records are stored in hashed form. By using SNARKs, we can keep the instantiation generic and universally prove its completeness and security for any triplet (f_I, f_T, f_U). A detailed description of the protocol is in Figure 4 in Appendix A.

Data representation. We internally split the data into training data D and unlearnt data U. Therefore, the server stores two ordered sets $\mathcal{H}_D$ and $\mathcal{H}_U$ of hashed training data records and unlearnt data records. From both sets, we additionally compute a hash value in form of a hash chain. This allows for efficient caching of intermediate hashes and, for $\mathcal{H}_U$, enables us to easily prove that entries are only appended to the chain as well as fast membership verification for unlearnt data points. To account for the partition of training and unlearnt data and the admissible function used, we instantiate the commitment com as a tuple of four elements: (a) hash of the state h_{st_f} (defined by admissible function f), (b) hash of the model h_m, (c) hash of the training data h_D, and (d) hash of the unlearnt data h_U. A formal description of how exactly the hash values are computed is given in Appendix A.

Proof system. In order to prove the correct execution of f_I, f_T, and f_U, we use proof systems and more specifically SNARKs. To this end, we define the verification of the initialization, training updates and unlearning updates in terms of a polynomial decidable binary relation R_I, R_T and R_U (respectively) over circuits C_I, C_T and C_U (respectively). These circuits are outlined in Figure 3 and described further below.

$C_I(\textbf{public } h_{\mathsf{st}_{f,0}}, h_{m_0}, h_{D_0}, h_{U_0}, \textbf{ private } \mathsf{st}_{f,0}, m_0)$

```
# Check input for initialization
if h_{st_{f,0}} ≠ HashState(st_{f,0}) or
   h_{m_0} ≠ HashModel(m_0) or
   h_{D_0} ≠ HashData(∅) or
   h_{U_0} ≠ HashData(∅):
    return false
return true
```

$C_T(\textbf{public } h_{\mathsf{st}_{f,i}}, h_{\mathsf{st}_{f,i-1}}, h_{m_i}, h_{D_i}, h_{D_{i-1}}, h_{U_i}, h_{U_{i-1}}, \textbf{ private } \mathsf{st}_{f,i-1}, \mathcal{H}_{U_{i-1}}, D_i^+)$

```
# Check input set of hashed unlearnt data records
if h_{U_{i-1}} ≠ HashData(H_{U_{i-1}}):
    return false
# Update and check set of hashed training data records and unlearnt data records
H_{D_i^+} := {HashDataRecord(u, d)}_{(u,d)∈D_i^+}
if h_{D_i} ≠ AppendHashData(h_{D_{i-1}}, H_{D_i^+}) or h_{U_i} ≠ h_{U_{i-1}} or H_{U_{i-1}} ∩ H_{D_i^+} ≠ ∅:
    return false
# Check input state, perform training and check outputs
if h_{st_{f,i-1}} ≠ HashState(st_{f,i-1}):
   return false
(st_{f,i}, m_i) := f_T(st_{f,i-1}, D_i^+)
if h_{st_{f,i}} ≠ HashState(st_{f,i}) or h_{m_i} ≠ HashModel(m_i):
    return false
return true
```

$C_U(\textbf{public } h_{\mathsf{st}_{f,i}}, h_{\mathsf{st}_{f,i-1}}, h_{m_i}, h_{D_i}, h_{D_{i-1}}, h_{U_i}, h_{U_{i-1}}, \textbf{ private } \mathsf{st}_{f,i-1}, \mathcal{H}_{D_{i-1}}, U_i^+)$

```
# Check input set of hashed training data records
if h_{D_{i-1}} ≠ HashData(H_{D_{i-1}}):
    return false
# Update and check set of hashed unlearnt data records and training data records
H_{U_i^+} := {HashDataRecord(u, d)}_{(u,d)∈U_i^+}
H_{D_i} := H_{D_{i-1}} \ H_{U_i^+}
if h_{U_i} ≠ AppendHashData(h_{U_{i-1}}, H_{U_i^+}) or h_{D_i} ≠ HashData(H_{D_i}):
    return false
# Check input state, perform unlearning and check outputs
h_{st_{f,i-1}} ≠ HashState(st_{f,i-1}):
   return false
(st_{f,i}, m_i) := f_U(st_{f,i-1}, U_i^+)
if h_{st_{f,i}} ≠ HashState(st_{f,i}) or h_{m_i} ≠ HashModel(m_i):
    return false
return true
```

Figure 3: **Circuits C_I, C_T and C_U.** Based on this circuits, we prove correct execution of admissible functions for initialization, proof of training and proof of unlearning. The hash algorithms are further specified in Appendix A.

1. Initialization. During the protocol's initialization, function f_I is run to obtain the initial state $\mathsf{st}_{f,0}$ and initial model m_0. Also, the sets of hashed training data and unlearnt data records are initialized, *i.e.,* $\mathcal{H}_{D_0} = \emptyset$ and $\mathcal{H}_{U_0} = \emptyset$. The commitment consists of hashes to these four values, *i.e.,* $\mathsf{com}_0 = (h_{\mathsf{st}_{f,0}}, h_{m_0}, h_{D_0}, h_{U_0})$. Correct initialization is proved using the SNARK for relation R_I captured by circuit C_I (cf. Figure 3). The proof of training ρ_0 consists of the statement ϕ_0 and resulting SNARK proof π_0, which can be verified by the user using com_0.

2A. Proof of training. The server starts by executing ProveTraining. In the i-th iteration, it first performs the model update by running function f_T on the previous state $\mathsf{st}_{f,i-1}$ and new data records D_i^+, the result being an updated state $\mathsf{st}_{f,i}$ and a new model m_i. Then the server updates the set of hashed training data records $\mathcal{H}_{D_i}$ with D_i^+ and computes the new commitment $\mathsf{com}_i = (h_{\mathsf{st}_{f,i}}, h_{m_i}, h_{D_i}, h_{U_{i-1}})$, where the commitment to the unlearnt data records is the same as in the previous iteration since no data was deleted.

The proof ρ_i is computed using the SNARK for relation R_T captured by circuit C_T (cf. Figure 3). The corresponding statement ϕ_i and proof π_i attest that (a) the model and state were updated correctly with D_i^+, (b) the set of hashed unlearnt data was not changed, and (c) no data record that was previously unlearnt is added. The server sends (ρ_i, com_i) to the users. Subsequently, the users execute VerifyTraining and verify ρ_i using com_i and the previous commitment com_{i-1}.

2B. Proof of unlearning. The proof of unlearning consists of two parts: the model update for deleting data records and the proof of non-membership. The server first runs ProveUnlearning which is similar to ProveTraining. In the i-th iteration, it performs the model update by running function f_U on the previous state $\mathsf{st}_{f,i-1}$ and the set U_i^+ of data records to be deleted. The result is the updated state $\mathsf{st}_{f,i}$ and model m_i. The set $\mathcal{H}_{U_i}$ is computed by appending hashed records of U_i^+ to $\mathcal{H}_{U_{i-1}}$. At the same time, $\mathcal{H}_{D_i}$ is computed from $\mathcal{H}_{D_{i-1}}$ by removing those entries. The commitment com_i consists of the hash values $(h_{\mathsf{st}_{f,i}}, h_{m_i}, h_{D_i}, h_{U_i})$. The whole procedure is proved using circuit C_U (cf. Figure 3) for relation R_U, producing a SNARK proof π_i for the corresponding statement ϕ_i, which can be verified by the user using com_i and com_{i-1}.

The second part of the proof of unlearning is to provide a proof on non-membership to all users that requested their data record $(u, d) \in U_i^+$ to be deleted. We prove this by

proving its membership in $\mathcal{H}_{U_i}$. If $\mathcal{H}_{U_i} \cap \mathcal{H}_{D_i} = \emptyset$, it follows that $(u, d) \notin D_i$ (which we show to hold when proving completeness). Specifically, we use the hash chain for $\mathcal{H}_{U_i}$: for a given data record, we compute a membership proof as a path in this chain; this path can be verified by recomputing the hash chain and comparing the final result with the hash in the commitment (*i.e.,* hash value h_{U_i}).

Thus, in our protocol, the server performs ProveNonMembership by computing the chain path to a data record $(u, d) \in U_i^+$ using the procedure ComputeChainPath (cf. Appendix A). It outputs this as the proof $\pi_{u,d}$ which is then sent to the user. The user uses the hash h_{U_i} from the commitment to verify membership with VerifyChainPath. If the path leads to that hash, the user will accept, and will abort otherwise.

5.1 Completeness and Security

We first show that our instantiation is complete according to Definition 1.

Theorem 1. *Let* Π *be a complete SNARK and* Hash *a collision-resistant hash function. Then the instantiated protocol in Figure 4 satisfies completeness.*

We give a proof sketch; refer to Appendix B for the full proof.

Proof (Sketch). Completeness of the initialization (first property) is easy to observe since the two hashed datasets are initialized as empty and the execution of function f_I is proven with the SNARK for relation R_I. By completeness of the SNARK, the users can successfully verify the proof, additionally using the commitments to state, model and datasets. The second property follows from the completeness of the SNARKs for relations R_T and R_U and collision-resistance of the hash function. However, note that if a hash collision occurs, it is not possible to provide the proof of training. Thus, only computational completeness can be achieved. Given that the proofs of training and unlearning are successful, completeness of the proof of non-membership (third property) follows from the construction and correctness of the hash chain. □

Now we want to prove that our instantiation is a *secure* unlearning protocol according to Definition 2.

Theorem 2. *Let* Hash *be a collision-resistant hash function and* Π *be a secure SNARK. Then the instantiated protocol in Figure 4 satisfies unlearning security.*

We give a proof sketch; refer to Appendix C for the full proof.

Proof (Sketch). Let $\mathcal{A}$ be an adversary in the unlearning security game (cf. Figure 2). By knowledge soundness of the SNARK, there exists an extractor which outputs the witness and thus the datasets D_i corresponding to the outputs of the adversary. We then use the soundness of the SNARK. That is, $\mathcal{A}$ must have computed the proof using a witness, *i.e.,* the state $\mathsf{st}_{f,i}$ and the dataset D_i^+ (in the proof of training) or dataset U_i^+ (in the proof of unlearning), which also determine the model m_i and must correspond to the hash values in the commitment. By collision-resistance of the hash function, the adversary cannot find another state, model or dataset for the same commitment. Thus, applying function f_T (or f_U) to the previous state and datasets results in same state and model as used by $\mathcal{A}$.

Since all proofs as well as the proof of non-membership of data record (u, d) must verify successfully, the hash of (u, d) must be contained in the set $\mathcal{H}_{U_k}$ which was used to create the proof. Here, k is the iteration where (u, d) was unlearnt; the observation holds by assuming soundness of the SNARK and collision-resistance of the hash function. We can further infer that (u, d) must also be part of future sets $\mathcal{H}_{U_i}$, $k < i \leq \ell$ and by collision-resistance (u, d) must also be part of the underlying datasets U_i. Finally, we use the fact that the proof attests that the intersection of $\mathcal{H}_{U_\ell}$ and $\mathcal{H}_{D_\ell}$ is empty. This yields a contradiction and shows that (u, d) cannot be present in the last dataset D_ℓ. □

6 Experimental Evaluation

In this section, we evaluate the performance of our instantiated protocol. First, we implement and compare the protocol's main building blocks for three types of unlearning approaches captured by admissible functions. The techniques from machine unlearning literature we consider are retraining-based unlearning, amnesiac unlearning and optimization-based unlearning (cf. Section 2.1). Second, we study the applicability to different ML models and datasets.

The goal of our experiments is to evaluate the feasibility of verifiable machine unlearning and understand how different unlearning techniques influence the proof creation and verification times. Our salient findings include:

1. The majority of the costs stem from the cost of performing verifiable training, which is at most 5× more expensive than verifiable unlearning. In scenarios where one trusts the training process, this results in immediate savings. Note that we did not aim to optimize the verifiable computation component; this is orthogonal to the problem considered in this paper. We provide suggestions on how to achieve better performance in Section 7.1.

2. Unlearning techniques that rely on simple mechanisms such as adding/subtracting information from model parameters (*e.g.,* amnesiac unlearning) are intuitively cheaper to prove (in comparison to retraining-based approaches). However, hidden costs emerge in having to verify the integrity of the inputs needed for such methods.

All experiments are performed on a server running Ubuntu 22.04 with 256 GB RAM and two Intel Xeon Gold 5320 CPUs. Our code is available at github.com/cleverhans-lab/verifiable-unlearning.

6.1 Cryptographic Primitives

Proof system. Our instantiation is generic and can be implemented with any secure SNARK (cf. Section 2.2) *i.e.,* the SNARK needs to satisfy completeness, soundness, and knowledge soundness. In this work, we use Spartan [21] as it is efficient and, more importantly, transparent, *i.e.,* it does not require a trusted setup. Spartan comes in two variants, as a succinct non-interactive zero-knowledge (NIZK) proof system and as a SNARK. Similar to the work of Angel *et al.* [32], we use the NIZK variant, where verification time is linear in the size of the R1CS instance (see below). By using the SNARK variant, some verification cost can be offset to the server and a one-time pre-processing step for the user.

Depending on the application it might also be sensible to use a different proof system. One alternative would be Groth16 [31], which, for example, requires a trusted setup, but has the advantage of constant verification time and proof size.

Spartan is implemented on the `ristretto255` elliptic curve, a prime-order group abstraction atop `curve25519`. Following prior work on verifiable computation [32, 33, 10], we convert the computation of our circuits into *Rank-1 Constraint Systems* (R1CS) instances; *i.e.,* the statements in R_I, R_T and R_U (cf. Figure 3) are represented as a

constraint system over a *finite field*. More specifically, an R1CS instance is described by a tuple $(\mathbb{F}, A, B, C, io, m)$, where $\mathbb{F}$ is the finite field, $A, B, C \in \mathbb{F}^{m \times m}$ are matrices of size $m \geq |io| + 1$ and io is the public input and output of the instance. R1CS is a generalization of arithmetic circuit satisfiability. We say an R1CS instance is satisfiable if there exists a witness $w \in \mathbb{F}^{m-|io|-1}$ such that $(A \cdot z) \circ (B \cdot z) = (C \cdot z)$ for $z = (io, 1, w)$, where $\cdot$ is the matrix-vector product and $\circ$ the the Hadamard product. Since A, B, C are generally sparse matrices, a parameter n is sometimes specified, denoting the maximum number of non-zero entries in each matrix.

We describe our circuits using the ZoKrates programming language [34] and use the CirC compiler infrastructure [35] to facilitate the conversion to R1CS. The compiler ensures that the computation graph does not have loops, and a "flat" computation is performed. To represent data and other parameters in a finite field, we convert them into *fixed precision* real numbers.

We only require collision-resistance for the hash function. It is beneficial to use an algebraic hash function where most operations can be directly done in the finite field. Bit-wise hash functions such as the SHA family of hash functions are much slower in that regard. We use Poseidon [36] as it is particularly designed for zero-knowledge proof systems. To be used with Spartan, we implement a version of Poseidon for the `ristretto255` curve. Similar to the proof system, our instantiation is generic and can work with any hash function. Other good options are *Pedersen Hash* [37, p.76] or *MIMC* [38].

6.2 Protocol Instantiation

We implement the high-level functions of the instantiated protocol (from Section 5) for retraining-based unlearning, amnesiac unlearning [14], and optimization-based unlearning [22, 23] as introduced in Section 2.1. We consider the subtasks of proof of training, proof of unlearning, and proof of non-membership.

We start our evaluation by comparing and understanding the overheads of each subtask between the techniques. To this end, we consider a linear regression model and train this model for 3 epochs with SGD as a general purpose approach. We use a synthetic dataset D and set the batch size to 1. First, we compute a proof of training with the addition of 100 data points with 10 features each. We set $|D_0| = 0$, $|D_1^+| =$

Table 1: **Run-time of protocol functions.** We compare the running time between the protocols subtasks. We consider retraining-based unlearning, amnesiac unlearning, and optimization-based unlearning. We report the relative difference with retraining in gray.

	Retraining		**Amnesiac**		**Optimization**	
Proof of Training						
R1CS	8,056,887	×1.00	8,130,535	×1.01	7,980,878	×0.99
Π.Prove w/ R_T	4m 32s	×1.00	4m 32s	×1.00	4m 31s	×0.99
Π.Vrfy w/ R_T	1m 36s	×1.00	1m 37s	×1.01	1m 35s	×0.99
Proof of Unlearning						
R1CS	8,102,288	×1.00	616,005	×0.08	919,456	×0.11
Π.Prove w/ R_U	4m 58s	×1.00	2m 18s	×0.46	0m 53s	×0.18
Π.Vrfy w/ R_U	1m 48s	×1.00	0m 49s	×0.45	0m 20s	×0.19
Proof of Non-Membership						
ComputeChainPath	$<$ 1s	×1.00	$<$ 1s	×1.00	$<$ 1s	×1.00
VerifyChainPath	$<$ 1s	×1.00	$<$ 1s	×1.00	$<$ 1s	×1.00

R1CS: #constraints

100, and $|U_0| = 0$ accordingly. Subsequently, we compute the proof of unlearning and simulate the deletion of 10 data points and set $|D_1| = 100$, $|U_1| = 0$, and $|U_2^+| = 10$. For optimization-based unlearning, we unlearn for 3 epochs.

The results from these experiments are presented in Table 1. Across all techniques, compilation time of R1CS instances ranges between 17s (C_U for optimization-based unlearning) and 48m 45s (C_U for retraining-based unlearning).

Proof of training. We observe that the complexity of the training is comparable between unlearning approaches. The underlying R1CS instances have between $7,980,878$—$8,130,535$ constraints and proving time varies insignificantly between 4m 31s—4m 32s. Recall that in amnesiac unlearning, we also need to collect model updates that are later used for unlearning, which introduces negligible overhead compared to the training costs itself.

Proof of unlearning. Runtime of generating and verifying the proof of unlearning shows more variance. Amnesiac unlearning is over 2× faster and optimization-based

Table 2: **Proving time vs. model capacity.** We compare the proving time of proof of training for different classes of models with increasing capacity.

Classifier	R1CS	Π.Prove	Π.Vrfy
Linear Regression	8,056,887	4m 33s	1m 36s
Logistic Regression	9,048,909	5m 8s	1m 45s
Neural Network ($N = 2$)	21,867,010	9m 50s	3m 34s
Neural Network ($N = 4$)	42,030,731	24m 16s	6m 42s

R1CS: #constraints

unlearning over 5× faster than retraining-based unlearning. This is despite the R1CS instance of optimization-based unlearning being almost 50% larger compared to the amnesiac instance (919,456 vs. 616,005 constraints) but it is still more efficient to compute as it is 63% more sparse (*i.e.,* 7,660,455 vs. 12,248,390 entries are non-zero). The main difference is that amnesiac unlearning requires to maintain and verify a state from training (*i.e.,* the model updates) while optimization-based unlearning does not require a state.

Proof system. In general, we observe that verification is 2×-3× faster than proof generation. This is dependent on the choice of the proof system. For example, by using the SNARK variant of Spartan, we can offload some of the verification costs to the server and an additional pre-processing for the user. In this case, proving time increases to 33m 39s—34m 12s for the proof of training across all techniques and verification time reduces to <1s, but the user needs to run a one-time pre-processing step which takes between 8m 6s—8m 12s.

Proof of non-membership. Finally, proof of non-membership is very efficient. The implementation is independent of the unlearning scheme used and both proving and verification requires < 1s.

6.3 Circuit Complexity

The dominant component of the protocols' run-time is the complexity of the circuit used to generate proofs of training and unlearning. This complexity depends mainly on

Table 3: **Scalability to benchmark datasets.** We compute the proof of training for different datasets from the PMLB benchmark suite [24].

Dataset	Size		R1CS	Π.Prove	Π.Vrfy
Creditscore	100	6	3,986,308	2m 22s	0m 47s
Patient	88	8	4,579,718	2m 28s	0m 53s
Cy Young	92	10	5,903,988	3m 16s	1m 9s
Corral	160	6	6,347,236	3m 43s	1m 15s
Lawsuit	264	4	7,190,981	4m 7s	1m 27s
Breast cancer	286	9	16,514,048	9m 25s	3m 18s
Monk3	554	6	21,841,281	13m 36s	4m 32s

Size: #data points × #features
R1CS: #constraints

(a) the unlearning technique, (b) the complexity of the model, and (c) the size of the dataset. In the following, we first consider model complexity and study different classes of models. Next, we look on the complexity of the dataset. In both cases, we focus on retraining-based unlearning as the baseline from Table 1 and, more specifically, on the training circuit C_T.

To understand the effects of the choice of ML model, we follow related work [39], and consider linear regression, logistic regression and neural networks for classification. For the neural networks, we focus on models with one hidden layer and varying numbers of (hidden) neurons $N \in \{2, 4\}$. For activation, we use the sigmoid function and approximate it with a third-order polynomial as done in [40, 41]. Again, we train each model with SGD for 3 epochs on a synthetic dataset consisting of 100 training points with 10 features each.

The results are summarized in Table 2. We observe that the circuit size increases together with the complexity of the model. For instance, the number of R1CS constraints increases by 1.12× to $9,048,909$ constraints when going from linear to logistic regression. This is intuitive: in logistic regression, we additionally need to evaluate the sigmoid activation which induces this overhead. In a similar vein, moving from logistic regression to neural networks increases the circuit further to $21,867,010$ ($N = 2$) and $42,030,731$ ($N = 4$) constraints respectively.

Benchmark datasets. To understand the impact of the dataset and the practical applicability of the protocol, we now turn to benchmark datasets. We choose several datasets from the PMLB benchmark suite [24] (as considered in related work [32] on verifiable computation of numerical optimization problems) and train a linear regression model for all datasets. To make results comparable, we train all models for 3 epochs with a learning rate of 0.1. As commonly done, we split the data into 80:20 train test split. Models achieve a test accuracy between 73 % and 92 %.

Results are presented in Table 3. For all models, we observe a linear dependence between run-time and dataset size. Generating a proof for the smallest dataset with 600 total features (*i.e.,* total points × features) requires 2m 22s and for the largest dataset with 3,324 total features requires 13m 36s.

7 Discussion

In this section, we discuss potential improvements to our work.

7.1 Scalability

Our experiments with the instantiated system show that the run-time of the protocol is dominated by generating and verifying the proof of training and unlearning. We base our construction on *Verified Computation* (VC) and, consequently, inherit its limitations, *e.g.,* in terms of scalability. This can also be observed for other VC-based approaches in the ML setting [42, 43, 40, 41]. Any future advances in VC will lead to run-time improvements for our approach. Nevertheless, we discuss how one can improve performance with the primitives available today.

SNARK-friendly techniques. It is known that certain computations are more amendable to efficient SNARK verification than others. A classic example of this is the development of SNARK-friendly hash functions [36, 38, 37]. Similarly, there exist ML paradigms that are also more amenable to verification. For example, inference using quantized models [44, 45] or lookup tables for expensive computations [44] reduce costs. Furthermore, when there exists a unique ML model (*i.e.,* a global optimum for the underlying optimization problem), proving and verification complexity can be improved even further [32]. In our experiments, we observed that online computation of model updates in optimization-based unlearning is faster than verifying model updates

in amnesiac unlearning as the verification of input values involves expensive calculation of hash values. We envision future work to focus on developing SNARK-friendly unlearning techniques combining above observations.

Offloading computation. Orthogonal to the employed unlearning technique and ML model, one can offload expensive proof generation steps to the user (*e.g.,* the evaluation of a non-linear activation function). We can split the proving and verification processes such that the server creates a proof for certain types of computations and shares partial results with the (honest) user who performs (and thus verifies) expensive computations themselves.

Application-specific relaxations. Finally, depending on the application, it might be possible to avoid the expensive generation of the proof of training. Consider, for instance, an application where data is collected only once and data will *only* be removed at a later point in time (*e.g.,* biomedical user studies or other human-involved data collection processes). In this case, proof of training only needs to be performed once and—if users further trust the initial training phase—it might be sufficient to only prove unlearning.

7.2 Alternative Instantiations

External trust. Our instantiation in Section 5 avoids having a trusted third party and instead relies only on cryptographic protocols to guarantee security. For efficiency purposes and to remove the burden from the user, one can introduce a trusted auditor who verifies on behalf of a user (as we discuss towards the end of Section 4.1). This can be achieved by either having a dedicated trusted third party (*e.g.,* one that does not have a motivation to collude with the server such as another cloud provider), or distributed auditors where trust is established from multiple independent verifications.

Trusted hardware. If TEEs (*e.g.,* Intel SGX [46]) are available, then one can run training and unlearning procedures within it and return a digest signed by a TEE provider to the user. When using a TEE, one needs to consider common concerns such as trusting a hardware vendor, availability of said vendor for signing the digest, limited memory [47], their applicability to ML-related tasks that involve GPU com-

putation [48], and side-channels [49, 50]. Some of these issues were addressed in the independent and concurrent work of Weng *et al.* [51] (cf. Section 8).

Minimizing redundancy. In our instantiation, a user who has requested unlearning is required to verify future updates to ensure that their data point has not been re-added. If we combine VC with an additional proof of secure data erasure, we can give similar guarantees while not requiring the user to verify all updates. However, secure erasure is a non-trivial problem in itself and was considered in *e.g.,* [52]. Formalizing deletion compliance from a server's perspective [53] can also be seen as complementary problem.

7.3 Privacy

Formalizing privacy for unlearning protocols is an interesting direction for future work and requires to establish an additional security definition. Although it is out-of-scope for our work, we want to highlight that our instantiation does not require the users to know the datasets or model. In fact, they only see hash commitments and the SNARK proofs. If the hash function satisfies pseudo-randomness or is modeled as a random oracle, then hash values do not leak any information about the underlying data points as long as the input space is large enough. Additionally, if the SNARK satisfies the zero-knowledge property [54] (which most SNARKs including Spartan do), then the proof also does not leak information about the witness. However, we require users to know whether training or unlearning happened because they need to know which verification procedure to run. Privacy in the context of model inference has been studied more extensively, *e.g.,* Gao *et al.* [55] define security notions for deletion hiding and reconstruction. An overview for different formalizations of inference privacy is also given in [56].

8 Related Work

Our approach for verifiable machine unlearning naturally touches different areas of security and ML research. In the following, we examine related concepts and methods.

Verifying unlearning. Prior work [57, 58] aims at verifying unlearning by embedding backdoors [59] in models (using data whose unlearning is to be verified) and verifying backdoor removal on unlearning. However, such approaches are probabilistic with no theoretical guarantees of when they work, unlike our cryptography-informed approach which produces verifiable proofs.

The work of Guo *et al.* [12] provides end-users with a certificate that the new model is influenced by the specific data in a quantifiably low manner. While this certificate conceptually bounds the influence of a data point from an algorithmic perspective, it provides no guarantee that the entity executing the algorithm (*i.e.,* server) did so correctly. In our work, we aim to capture exactly this and provide cryptographic guarantees of correctness of execution.

Concurrently to our work, Weng *et al.* [51] propose an unlearning framework based on TEE. They model unlearning in two phases: a setup phase, where the user sends data which is used to train an ML model, and a deletion phase, where a new model is trained without the data point that the user requests to delete. Their protocol uses unlearning based on SISA [11] and can be captured by our framework as well. In contrast to our instantiation that is based only on cryptographic primitives, their approach relies on trusted hardware (*i.e.,* the correctness and integrity of the SGX enclave) as well as cryptographic assumptions (*i.e.,* EUF-CMA security of the signature scheme used by the enclave and collision-resistance of the hash function).

Proving model inference. There exist various approaches to proving inference using SNARKs [60, 61, 62, 45, 44] which complements our protocol in that regard. Another approach would be to use trusted execution environments to do so as suggested in [51].

Verifiable computation. We use verifiable computation for proof of training and proof of unlearning. There has been a series of works demonstrating a remarkable progress in making these schemes (and those related to verification of data used for computation) practical [21, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 30]. To verify the computation of training an ML model, Zhao *et al.* [39] also propose verification using a SNARK. However, their primary objective is to design a scheme to ensure that the payments made to servers are correct. In our work, however, we aim to design a scheme to verify the correctness of data deletion when training ML

models. Otti [32] is a compiler that is aimed at designing efficient arithmetic circuits for problems that involve optimization (such as those commonly found in ML). DIZK [10] is a distributed system capable of distributing the compute required for proof creation.

9 Conclusion

The problem of unlearning has gained significant interest in terms of definitions and algorithms for updating model parameters. However, regardless of the definition or the algorithm the server uses to update the model, the user has no way to verify that the server indeed executed the unlearning procedure. In this paper, we define unlearning as a security problem and propose a framework to capture the guarantees verifiable unlearning needs to provide. We propose the first verifiable unlearning procedure based on cryptographic primitives instantiated using SNARKs and hash chains. Our implementation shows the feasibility of our approach on several benchmark datasets and machine learning models. Future work includes determining which unlearning techniques are most suitable for efficient verifiable computation, while at the same time devising methods specifically for verifying machine learning code.

Acknowledgements

Thorsten Eisenhofer and Doreen Riepel were funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germanys Excellence Strategy - EXC 2092 CASA - 390781972. Nicolas Papernot would like to acknowledge his sponsors, who support his research with financial and in-kind contributions, including Apple, CIFAR through the Canada CIFAR AI Chair program, DARPA through the GARD program, Intel, NSERC through the Discovery grant, Meta, and Ontario through the Early Researcher Award program. Resources used in preparing this research were provided, in part, by the Province of Ontario, the Government of Canada through CIFAR, and companies sponsoring the Vector Institute. We would like to thank members of the CleverHans Lab for their feedback. We would also like to thank Sebastian Angel, Jess Woods, and Eleftherios Ioannidis for input related to the SNARK compilers.

References

[1] General Data Protection Regulation (GDPR). Official Legal Text, 2016.

[2] California Consumer Privacy Act (CCPA). Official Legal Text, 2018.

[3] Personal Information Protection and Electronic Documents Act (PIPEDA). Official Legal Text, 2019.

[4] Pang Wei Koh and Percy Liang. Understanding Black-box Predictions via Influence Functions. In *International Conference on Machine Learning (ICML)*.

[5] Vitaly Feldman. Does Learning Require Memorization? A Short Tale About a Long Tail. In *ACM SIGACT Symposium on Theory of Computing (STOC)*, 2020.

[6] Gavin Brown, Mark Bun, Vitaly Feldman, Adam D. Smith, and Kunal Talwar. When is Memorization of Irrelevant Training Data Necessary for High-Accuracy Learning? In *ACM SIGACT Symposium on Theory of Computing (STOC)*, 2021.

[7] Nicholas Carlini, Chang Liu, Úlfar Erlingsson, Jernej Kos, and Dawn Song. The Secret Sharer: Evaluating and Testing Unintended Memorization in Neural Networks. In *USENIX Security Symposium*, 2019.

[8] Klas Leino and Matt Fredrikson. Stolen Memories: Leveraging Model Memorization for Calibrated White-Box Membership Inference. In *USENIX Security Symposium*, 2020.

[9] Yinzhi Cao and Junfeng Yang. Towards Making Systems Forget with Machine Unlearning. In *IEEE Symposium on Security and Privacy (S&P)*, 2015.

[10] Howard Wu, Wenting Zheng, Alessandro Chiesa, Raluca Ada Popa, and Ion Stoica. DIZK: A Distributed Zero Knowledge Proof System. In *USENIX Security Symposium*, 2018.

[11] Lucas Bourtoule, Varun Chandrasekaran, Christopher A. Choquette-Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. Machine Unlearning. In *IEEE Symposium on Security and Privacy (S&P)*, 2021.

[12] Chuan Guo, Tom Goldstein, Awni Hannun, and Laurens Van Der Maaten. Certified Data Removal from Machine Learning Models. In *International Conference on Machine Learning (ICML)*, 2020.

[13] Aditya Golatkar, Alessandro Achille, and Stefano Soatto. Eternal Sunshine of the Spotless Net: Selective Forgetting in Deep Networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[14] Laura Graves, Vineel Nagisetty, and Vijay Ganesh. Amnesiac Machine Learning. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2021.

[15] Thomas Baumhauer, Pascal Schöttle, and Matthias Zeppelzauer. Machine Unlearning: Linear Filtration for Logit-based Classifiers. *Machine Learning*, 2022.

[16] Ayush Sekhari, Jayadev Acharya, Gautam Kamath, and Ananda Theertha Suresh. Remember What You Want to Forget: Algorithms for Machine Unlearning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[17] Ozan Sener and Silvio Savarese. Active Learning for Convolutional Neural Networks: A Core-Set Approach. In *International Conference on Learning Representations (ICLR)*, 2017.

[18] Amirata Ghorbani and James Zou. Data Shapley: Equitable Valuation of Data for Machine Learning. In *International Conference on Machine Learning (ICML)*, 2019.

[19] Ilia Shumailov, Zakhar Shumaylov, Dmitry Kazhdan, Yiren Zhao, Nicolas Papernot, Murat A. Erdogdu, and Ross J. Anderson. Manipulating SGD with Data Ordering Attacks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[20] Anvith Thudi, Hengrui Jia, Ilia Shumailov, and Nicolas Papernot. On the Necessity of Auditable Algorithmic Definitions for Machine Unlearning. In *USENIX Security Symposium*, 2022.

[21] Srinath Setty. Spartan: Efficient and General-Purpose zkSNARKs Without Trusted Setup. In *Annual International Cryptology Conference (CRYPTO)*, 2020.

[22] Joel Jang, Dongkeun Yoon, Sohee Yang, Sungmin Cha, Moontae Lee, Lajanugen Logeswaran, and Minjoon Seo. Knowledge Unlearning for Mitigating Privacy Risks in Language Models. *Computing Research Repository (CoRR)*, 2022.

[23] Alexander Warnecke, Lukas Pirch, Christian Wressnegger, and Konrad Rieck. Machine Unlearning of Features and Labels. In *Symposium on Network and Dis-*

tributed System Security (NDSS), 2023.

[24] Joseph D. Romano, Trang T. Le, William G. La Cava, John T. Gregg, Daniel J. Goldberg, Praneel Chakraborty, Natasha L. Ray, Daniel S. Himmelstein, Weixuan Fu, and Jason H. Moore. PMLB v1.0: An Open-Source Dataset Collection for Benchmarking Machine Learning Methods. *Bioinformatics*, 2022.

[25] Yinjun Wu, Edgar Dobriban, and Susan B. Davidson. DeltaGrad: Rapid Retraining of Machine Learning Models. In *International Conference on Machine Learning (ICML)*, 2020.

[26] Seth Neel, Aaron Roth, and Saeed Sharifi-Malvajerdi. Descent-to-Delete: Gradient-Based Methods for Machine Unlearning. In *Algorithmic Learning Theory (ALT)*, 2021.

[27] Min Chen, Zhikun Zhang, Tianhao Wang, Michael Backes, Mathias Humbert, and Yang Zhang. Graph Unlearning. In *ACM Conference on Computer and Communications Security (CCS)*, 2021.

[28] Anvith Thudi, Gabriel Deza, Varun Chandrasekaran, and Nicolas Papernot. Unrolling SGD: Understanding Factors Influencing Machine Unlearning. In *IEEE European Symposium on Security and Privacy (EuroS&P)*, 2022.

[29] Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. From Extractable Collision Resistance to Succinct Non-Interactive Arguments of Knowledge, and Back Again. In *Innovations in Theoretical Computer Science (ITCS)*, 2012.

[30] Dario Fiore, Cédric Fournet, Esha Ghosh, Markulf Kohlweiss, Olga Ohrimenko, and Bryan Parno. Hash First, Argue Later: Adaptive Verifiable Computations on Outsourced Data. In *ACM Conference on Computer and Communications Security (CCS)*, 2016.

[31] Jens Groth. On the Size of Pairing-Based Non-interactive Arguments. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT)*, 2016.

[32] Sebastian Angel, Andrew J. Blumberg, Eleftherios Ioannidis, and Jess Woods. Efficient Representation of Numerical Optimization Problems for SNARKs. In *USENIX Security Symposium*, 2022.

[33] Alex Ozdemir, Riad S. Wahby, Barry Whitehat, and Dan Boneh. Scaling Verifiable Computation Using Efficient Set Accumulators. In *USENIX Security Symposium*, 2020.

[34] Jacob Eberhardt and Stefan Tai. ZoKrates - Scalable Privacy-Preserving Off-Chain Computations. In *IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, 2018.

[35] Alex Ozdemir, Fraser Brown, and Riad S. Wahby. CirC: Compiler Infrastructure for Proof Systems, Software Verification, and more. In *IEEE Symposium on Security and Privacy (S&P)*, 2022.

[36] Lorenzo Grassi, Dmitry Khovratovich, Christian Rechberger, Arnab Roy, and Markus Schofnegger. Poseidon: A New Hash Function for Zero-Knowledge Proof Systems. In *USENIX Security Symposium*, 2021.

[37] Daira Hopwood, Sean Bowe, Taylor Hornby, and Nathan Wilcox. Zcash. Protocol Specification (Version 2022.3.4), 2022.

[38] Martin R. Albrecht, Lorenzo Grassi, Christian Rechberger, Arnab Roy, and Tyge Tiessen. MiMC: Efficient Encryption and Cryptographic Hashing with Minimal Multiplicative Complexity. In *International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT)*, 2016.

[39] Lingchen Zhao, Qian Wang, Cong Wang, Qi Li, Chao Shen, and Bo Feng. VeriML: Enabling Integrity Assurances and Fair Payments for Machine Learning as a Service. *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, 2021.

[40] Andrey Kim, Yongsoo Song, Miran Kim, Keewoo Lee, and Jung Hee Cheon. Logistic Regression Model Training based on the Approximate Homomorphic Encryption. *Cryptology ePrint Archive*, 2018.

[41] Miran Kim, Yongsoo Song, Shuang Wang, Yuhou Xia, and Xiaoqian Jiang. Secure Logistic Regression Based on Homomorphic Encryption: Design and Evaluation. *JMIR Medical Informatics*, 2018.

[42] Avital Shafran, Gil Segev, Shmuel Peleg, and Yedid Hoshen. Crypto-Oriented Neural Architecture Design. In *IEEE International Conference on Acoustics, Speech*

and Signal Processing, (ICASSP), 2021.

[43] Inbar Helbitz and Shai Avidan. Reducing ReLU Count for Privacy-Preserving CNN Speedup. *Computing Research Repository (CoRR)*, 2021.

[44] Daniel Kang, Tatsunori Hashimoto, Ion Stoica, and Yi Sun. Scaling up Trustless DNN Inference with Zero-Knowledge Proofs. *Computing Research Repository (CoRR)*, 2022.

[45] Boyuan Feng, Lianke Qin, Zhenfei Zhang, Yufei Ding, and Shumo Chu. ZEN: Efficient Zero-Knowledge Proofs for Neural Networks. *Cryptology ePrint Archive*, 2021.

[46] Frank McKeen, Ilya Alexandrovich, Alex Berenzon, Carlos V. Rozas, Hisham Shafi, Vedvyas Shanbhogue, and Uday R. Savagaonkar. Innovative Instructions and Software Model for Isolated Execution. In *Workshop on Hardware and Architectural Support for Security and Privacy (HASP)*, 2013.

[47] Karan Grover, Shruti Tople, Shweta Shinde, Ranjita Bhagwan, and Ramachandran Ramjee. Privado: Practical and Secure DNN Inference with Enclaves. *Computing Research Repository (CoRR)*, 2018.

[48] Stavros Volos, Kapil Vaswani, and Rodrigo Bruno. Graviton: Trusted Execution Environments on GPUs. In *Symposium on Operating Systems Design and Implementation, (OSDI)*, 2018.

[49] Olga Ohrimenko, Felix Schuster, Cedric Fournet, Aastha Mehta, Sebastian Nowozin, Kapil Vaswani, and Manuel Costa. Oblivious Multi-Party Machine Learning on Trusted Processors. In *USENIX Security Symposium*, 2016.

[50] Wenhao Wang, Guoxing Chen, Xiaorui Pan, Yinqian Zhang, XiaoFeng Wang, Vincent Bindschaedler, Haixu Tang, and Carl A Gunter. Leaky Cauldron on the Dark Land: Understanding Memory Side-Channel Hazards in SGX. In *ACM Conference on Computer and Communications Security (CCS)*, 2017.

[51] Jia-Si Weng, Shenglong Yao, Yuefeng Du, Junjie Huang, Jian Weng, and Cong Wang. Proof of Unlearning: Definitions and Instantiation. *Computing Research Repository (CoRR)*, 2022.

[52] Daniele Perito and Gene Tsudik. Secure Code Update for Embedded Devices via Proofs of Secure Erasure. In *European Symposium on Research in Computer Security (ESORICS)*, 2010.

[53] Sanjam Garg, Shafi Goldwasser, and Prashant Nalini Vasudevan. Formalizing Data Deletion in the Context of the Right to Be Forgotten. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT)*, 2020.

[54] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The Knowledge Complexity of Interactive Proof Systems. *SIAM Journal on Computing (SICOMP)*, 1989.

[55] Ji Gao, Sanjam Garg, Mohammad Mahmoody, and Prashant Nalini Vasudevan. Deletion Inference, Reconstruction, and Compliance in Machine (Un)learning. In *Privacy Enhancing Technologies Symposium (PETS)*, 2022.

[56] Ahmed Salem, Giovanni Cherubin, David Evans, Boris Köpf, Andrew Paverd, Anshuman Suri, Shruti Tople, and Santiago Zanella Béguelin. SoK: Let The Privacy Games Begin! A Unified Treatment of Data Inference Privacy in Machine Learning. *Computing Research Repository (CoRR)*, 2022.

[57] Xiangshan Gao, Xingjun Ma, Jingyi Wang, Youcheng Sun, Bo Li, Shouling Ji, Peng Cheng, and Jiming Chen. VeriFi: Towards Verifiable Federated Unlearning. *Computing Research Repository (CoRR)*, 2022.

[58] David Marco Sommer, Liwei Song, Sameer Wagh, and Prateek Mittal. Towards Probabilistic Verification of Machine Unlearning. *Computing Research Repository (CoRR)*, 2020.

[59] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. BadNets: Identifying Vulnerabilities in the Machine Learning Model Supply Chain. *Computing Research Repository (CoRR)*, 2017.

[60] Seunghwan Lee, Hankyung Ko, Jihye Kim, and Hyunok Oh. vCNN: Verifiable Convolutional Neural Network. *Cryptology ePrint Archive*, 2020.

[61] Tianyi Liu, Xiang Xie, and Yupeng Zhang. ZkCNN: Zero Knowledge Proofs for Convolutional Neural Network Predictions and Accuracy. In *ACM Conference on Computer and Communications Security (CCS)*, 2021.

[62] Chenkai Weng, Kang Yang, Xiang Xie, Jonathan Katz, and Xiao Wang. Mystique: Efficient Conversions for Zero-Knowledge Proofs with Applications to Machine Learning. In *USENIX Security Symposium*, 2021.

[63] Srinath TV Setty, Richard McPherson, Andrew J Blumberg, and Michael Walfish. Making Argument Systems for Outsourced Computation Practical (Sometimes). In *Symposium on Network and Distributed System Security (NDSS)*, 2012.

[64] Benjamin Braun, Ariel J. Feldman, Zuocheng Ren, Srinath T. V. Setty, Andrew J. Blumberg, and Michael Walfish. Verifying Computations with State. In *ACM SIGOPS Symposium on Operating Systems Principles (SOSP)*, 2013.

[65] Srinath Setty, Victor Vu, Nikhil Panpalia, Benjamin Braun, Andrew J Blumberg, and Michael Walfish. Taking Proof-Based Verified Computation a Few Steps Closer to Practicality. In *USENIX Security Symposium*, 2012.

[66] Victor Vu, Srinath Setty, Andrew J Blumberg, and Michael Walfish. A Hybrid Architecture for Interactive Verifiable Computation. In *IEEE Symposium on Security and Privacy (S&P)*, 2013.

[67] Srinath Setty, Sebastian Angel, Trinabh Gupta, and Jonathan Lee. Proving the Correct Execution of Concurrent Services in Zero-Knowledge. In *Symposium on Operating Systems Design and Implementation (OSDI)*, 2018.

[68] Jonathan Lee, Kirill Nikitin, and Srinath Setty. Replicated State Machines without Replicated Execution. In *IEEE Symposium on Security and Privacy (S&P)*, 2020.

[69] Eli Ben-Sasson, Alessandro Chiesa, Daniel Genkin, Eran Tromer, and Madars Virza. SNARKs for C: Verifying Program Executions Succinctly and in Zero Knowledge. In *Annual International Cryptology Conference (CRYPTO)*, 2013.

[70] Eli Ben-Sasson, Alessandro Chiesa, Eran Tromer, and Madars Virza. Succinct Non-Interactive Zero Knowledge for a von Neumann Architecture. In *USENIX Security Symposium*, 2014.

[71] Eli Ben-Sasson, Oded Goldreich, Prahladh Harsha, Madhu Sudan, and Salil Vadhan. Robust PCPs of Proximity, Shorter PCPs, and Applications to Coding. *SIAM Journal on Computing*, 2006.

[72] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. Scalable Zero Knowledge with No Trusted Setup. In *Annual International Cryptology Conference (CRYPTO)*, 2019.

[73] Eli Ben-Sasson, Oded Goldreich, Prahladh Harsha, Madhu Sudan, and Salil Vadhan. Short PCPs Verifiable in Polylogarithmic Time. In *IEEE Conference on Computational Complexity (CCC)*, 2005.

[74] Eli Ben-Sasson, Alessandro Chiesa, Daniel Genkin, and Eran Tromer. On the Concrete Efficiency of Probabilistically-Checkable Proofs. In *Symposium on Theory of Computing Conference (STOC)*, 2013.

[75] Rosario Gennaro, Craig Gentry, and Bryan Parno. Non-interactive Verifiable Computing: Outsourcing Computation to Untrusted Workers. In *Annual International Cryptology Conference (CRYPTO)*, 2010.

[76] Bryan Parno, Jon Howell, Craig Gentry, and Mariana Raykova. Pinocchio: Nearly Practical Verifiable Computation. In *IEEE Symposium on Security and Privacy (S&P)*, 2013.

A Our Instantiated Protocol

A schematic overview of our complete instantiated protocol is given in Figure 4. Additional algorithms are described below.

```
HashData(H)
Ψ ≔ Hash(d_∅)
for h_d ∈ H:
    Ψ ≔ Hash(Ψ, h_d)
return Ψ

AppendHashData(h, H)
Ψ ≔ h
for h_d ∈ H:
    Ψ ≔ Hash(Ψ, h_d)
return Ψ

HashDataRecord(u, d = (x, y))
h_d ≔ Hash(u)
for x_j ∈ x:
    h_d ≔ Hash(h_d, Hash(x_j))
h_d ≔ Hash(h_d, Hash(y))
return h_d

HashModel(m = [w_0, ..., w_n])
h_m ≔ Hash(m[0])
for w_i ∈ m[1: ]:
    h_m ≔ Hash(h_m, Hash(w_i))
return h_m

HashState(st_f)
h_{st_f} ≔ Hash(st_f[0])
for s_i ∈ h_{st_f}[1: ]:
    h_{st_f} ≔ Hash(h_{st_f}, s_i)
return h_{st_f}

ComputeChainPath(u, d, H_U)
h_d ≔ HashDataRecord(u, d)
idx_d ≔ H_U.index(h_d)
if idx_d = ⊥:
    return ⊥
# get intermediate hash from chain below d
Ψ ≔ HashData(H_{U_{: idx_d}})
π_{u,d} ≔ [Ψ]
# add path from d
for h_d ∈ H_{U_{idx_d+1:}} :
    π_{u,d}.append(h_d)
return π_{u,d}

VerifyChainPath(u, d, h_U, π_{u,d})
# recompute hash Ψ from path π_{u,d}
Ψ ≔ Hash(π_{u,d}[0], HashDataRecord(u, d))
for node in π_{u,d}[1: ]:
    Ψ ≔ Hash(Ψ, node)
# verify final hash
return ⟦Ψ = h_U⟧
```

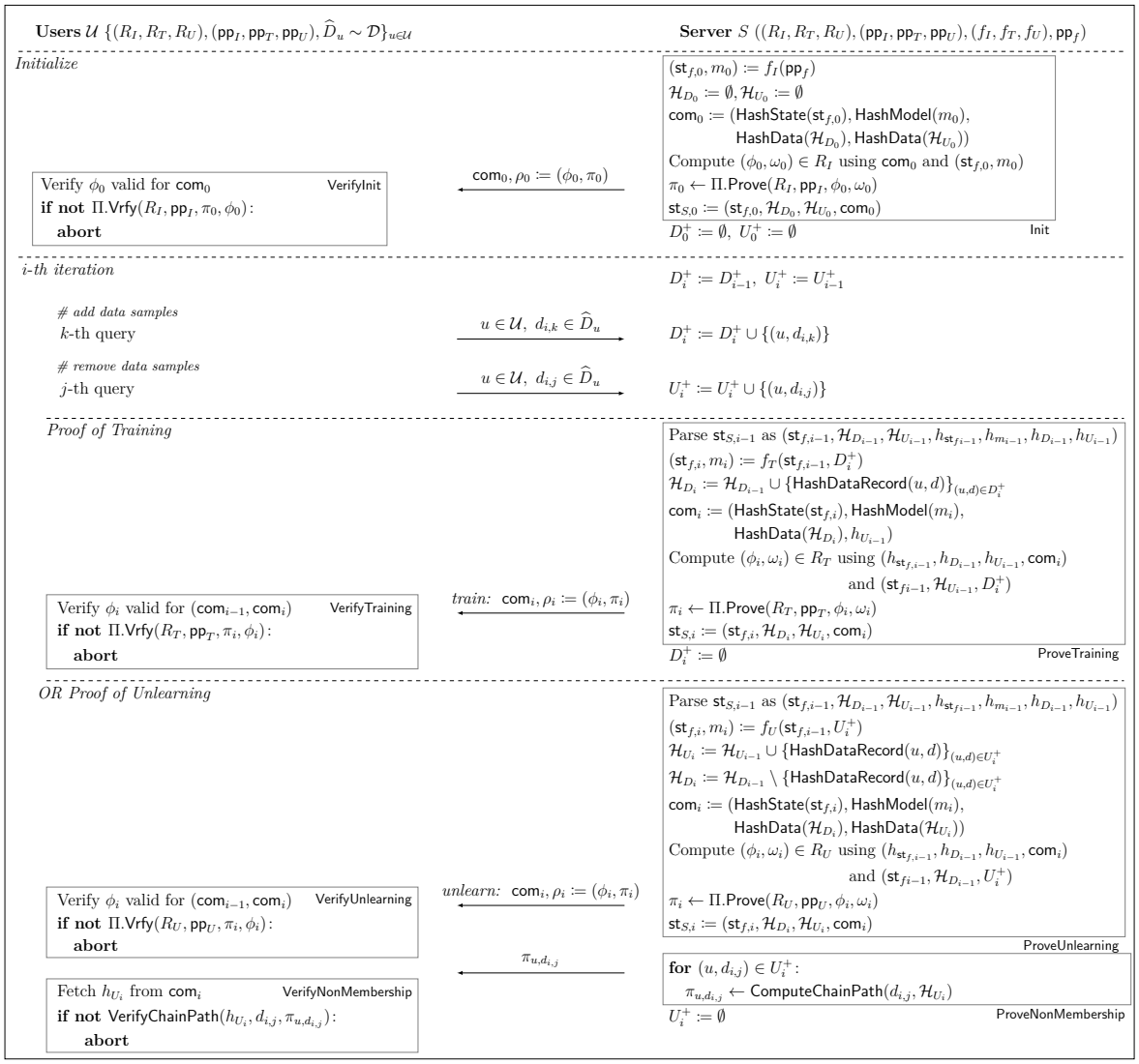


Figure 4: **Instantiated protocol Φ_f.** We instantiate the protocol for the triple of admissible functions $f = (f_I, f_T, f_U)$ with two primitives: a SNARK Π, and a hash function Hash.

B Completeness Proof

Proof (of Theorem 1). We need to prove the three properties in Definition 1 capturing the initialization, the proof of training and the proof of unlearning which includes the proof of non-membership.

Initialization. First, running Init yields the initialized state $\mathsf{st}_{f,0}$ and model m_0 obtained by executing f_I. Using the hash function to commit to those two values and additionally the empty sets $\mathcal{H}_{D_0}$ and $\mathcal{H}_{U_0}$, an instance $(\phi_0, \omega_0) \in R_I$ can be derived and the SNARK proof π_0 can be created using $\Pi.\mathsf{Prove}$. By correctness of the relation and completeness of the SNARK, ϕ_0 will be valid for com_0 and $\Pi.\mathsf{Vrfy}(R_I, \mathsf{pp}_I, \pi_0, \phi_0) = 1$.

Proof of update. For the second property, recall the inputs and outputs of $\mathsf{ProveTraining}$ and $\mathsf{ProveUnlearning}$. The state $\mathsf{st}_{S,i-1}$ contains the state $\mathsf{st}_{f,i-1}$, the set $\mathcal{H}_{D_{i-1}}$ of hashed data records, the set $\mathcal{H}_{U_{i-1}}$ of hashed unlearnt data records and the previous commitment com_{i-1}. In the proof of training, the new state $\mathsf{st}_{f,i}$ and the new model m_i are computed by running function f_T on $\mathsf{st}_{f,i-1}$ and the set D_i^+ of data records to be added. The new sets $\mathcal{H}_{D_i}$ and $\mathcal{H}_{U_i}$ are computed from the previous ones and updated with D_i^+. The commitment is computed by hashing the four components.

Then the training instance $(\phi_i, \omega_i) \in R_T$ can be derived and the SNARK proof π_i computed. The proof attests (cf. Figure 3) that (1) m_i was computed correctly since f_T was executed, (2) the training data does not contain unlearnt record since the hashes of the new data records are not contained in $\mathcal{H}_{U_i}$, and (3) the set of unlearnt data records has not changed since the commitments to the unlearnt data records are the same. By correctness of the relation and perfect completeness of the SNARK, we have $\Pi.\mathsf{Vrfy}(R_T, \mathsf{pp}_T, \pi_i, \phi_i) = 1$.

Note that there exists a special case where the server is unable to create a proof although the datasets are valid. This is the case whenever there exist two distinct data records $(u, d) \in D_i^+$, $(u', d') \in U_i$, where $U_i = \bigcup_{j \in [i]} U_j^+$ is the dataset implicitly contained in $\mathcal{H}_{U_i}$, such that $\mathsf{HashDataRecord}(u, d) = \mathsf{HashDataRecord}(u', d')$. However, we only require computational completeness and assume that the datasets are provided by a PPT adversary. Then this translates to finding a collision for the hash function which happens with negligible probability if the hash function is collision-resistance. Hence, $\mathsf{VerifyTraining}$ will output 1 with probability $1 - \mathsf{negl}(\lambda)$.

Proof of unlearning. Completeness for the proof of unlearning proceeds similar. The new state $\mathsf{st}_{f,i}$ and the new model m_i are computed by running function f_U on $\mathsf{st}_{f,i-1}$ and the set U_i^+ of data records to be deleted. The new sets $\mathcal{H}_{D_i}$ and $\mathcal{H}_{U_i}$ are computed from the previous ones and updated by removing and appending U_i^+, respectively. The commitment is computed by hashing the four components.

The unlearning instance $(\phi_i, \omega_i) \in R_U$ is derived and the SNARK proof π_i that is computed attests (cf. Figure 3) that (1) m_i was computed correctly since f_U was executed, (2) the training data does not contain unlearnt record since we removed the records in U_i^+ from $\mathcal{H}_{D_i}$, and (3) previous set of unlearnt data records is a subset of the updated set since we added the records in U_i^+ to $\mathcal{H}_{U_i}$ to which we commit. By correctness of the relation and perfect completeness of the SNARK, we have $\Pi.\mathsf{Vrfy}(R_U, \mathsf{pp}_U, \pi_i, \phi_i) = 1$ and $\mathsf{VerifyUnlearn}$ will output 1 with probability 1.

Finally, consider the algorithm $\mathsf{ProveNonMembership}$. If a data record (u, d) was unlearnt in iteration i, then its hash is present in $\mathcal{H}_{U_i}$. The proof of non-membership $\pi_{u,d}$ consists of the chain path to (u, d) in the chain of $\mathcal{H}_{U_i}$. Let com_i be the commitment for this iteration, then by correctness of the tree path algorithm, $\mathsf{VerifyNonMembership}$ will output 1 with probability 1. □

C Security Proof

Proof (of Theorem 2). Let $\mathcal{A}$ be an adversary against unlearning security (as defined in Figure 2) of our instantiation. We will first argue that for all $\mathcal{A}$ there exists an extractor $\mathcal{E}$ that outputs the underlying datasets D_i. This follows directly from the knowledge soundness of the SNARK for relations R_I, R_T and R_U. For this, look at the private inputs to the circuits in Figure 3 which translate to the witness. Initialization gives us that $D_0 = \emptyset$. The proof of training inputs D_i^+ and the proof of unlearning inputs U_i^+ such that we can extract $D_i = D_{i-1} \cup D_i^+$ if $mode_i = train$ and $D_i = D_{i-1} \setminus U_i^+$ if $mode_i = unlearn$.

We will now prove the theorem by the sequence of games given in Figure 5 and analyze the probability that these games will output 1.

Game G_0. Let G_0 be the original game $\mathsf{GameUnlearn}$ and $\mathcal{E}$ be the extractor. Recall that the adversary must output a sequence of tuples $(k, (u, d), \pi_{u,d}, \{mode_i\text{:}\ \mathsf{com}_i, \rho_i\}_{i\in[0:\ell]})$ for some $\ell \in \mathbb{N}$, where $\mathsf{com}_i = (h_{\mathsf{st}_{f,i}}, h_{m_i}, h_{D_i}, h_{U_i})$ and $\rho_i = (\phi_i, \pi_i)$ for $i \in [0 : \ell]$. We iterate over the winning conditions and return 0 as soon as one of them is violated

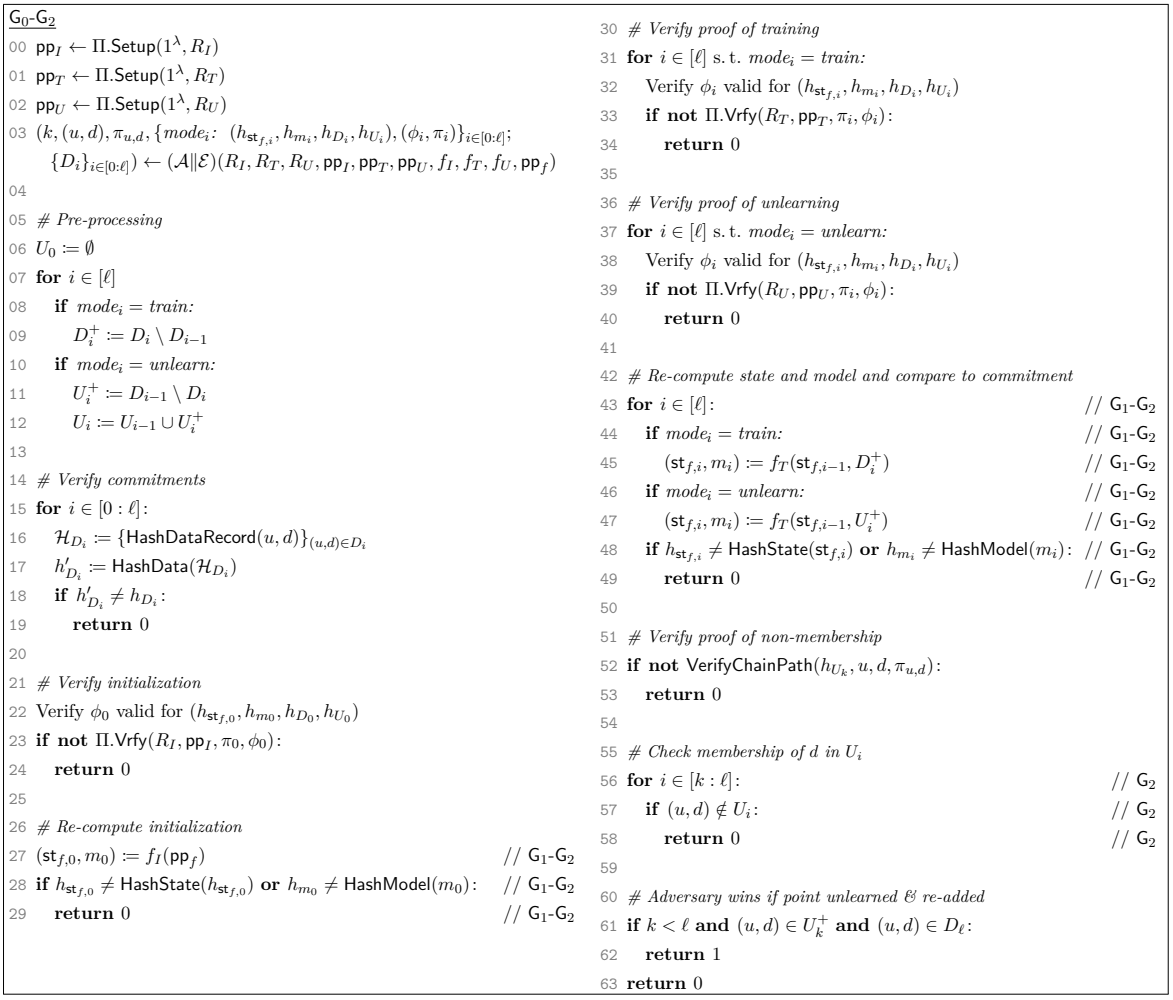


Figure 5: **Games G_0-G_2 for the proof of Theorem 2**. We prove unlearning security for our instantiated protocol Φ_f in Figure 4, where $f = (f_I, f_T, f_U)$ and hyperparameter pp_f are fixed by the participating parties and determine relations R_I, R_T and R_U.

(cf. Figure 5). For book-keeping we also compute all sets of unlearnt data points U_i, as well as the sets D_i^+, U_i^+ from D_i as described for the extractor. Note that this is only a conceptual change at this point and we have

$$\Pr[\mathsf{G}_0 \Rightarrow 1] = \Pr[\mathsf{GameUnlearn}_{\mathcal{A},\mathcal{E},\Phi_f,\mathcal{D}}(1^\lambda) \Rightarrow 1] \ .$$

Game G_1. In G_1, we compute the state $\mathsf{st}_{f,i}$ and the model m_i for each iteration from the corresponding datasets by applying f_I, f_T and f_U. We then check whether the

hashes of state and model correspond to $h_{\mathsf{st}_{f,i}}$ and h_{m_i} in the commitment. If this is not the case, the game outputs 0. We claim

$$|\Pr[\mathsf{G}_1 \Rightarrow 1] - \Pr[\mathsf{G}_0 \Rightarrow 1]| \leq \mathsf{negl}(\lambda) \ .$$

To prove the claim we argue in the following steps:

- First, π_i proves that the adversary knows a state $\mathsf{st}'_{f,i}$ and a dataset $D_i^{+\prime}$ for each proof of training (or a dataset $U_i^{+\prime}$ for each proof of unlearning) such that model m'_i was computed by applying function f_T (or function f_U) to state $\mathsf{st}'_{f,i-1}$ and dataset $D_i^{+\prime}$ (or dataset $U_i^{+\prime}$). It also proves that the commitment aligns with the inputs. Since the functions are deterministic, we thus have $h_{\mathsf{st}_{f,i}} = \mathsf{HashState}(\mathsf{st}'_{f,i})$ and $h_{m_i} = \mathsf{HashData}(m'_i)$ as well as $h_{D_i} = \mathsf{HashData}(\mathcal{H}'_{D_i})$, where $\mathcal{H}'_{D_i}$ is the set of all hashed data records in D'_i.

  By soundness of the SNARK, the adversary can only forge a proof for an invalid statement with negligible probability, so we can assume the proof was generated honestly with a witness. By knowledge soundness, the extractor is able to compute this witness such that $D'_i = D_i$.

- Second, we claim that then $\mathsf{st}'_{f,i} = \mathsf{st}_{f,i}$ and $m'_i = m_i$ are the actual state and model used for the next iteration. This is true unless the adversary finds a collision in the hash function such that $\mathsf{HashState}(\mathsf{st}'_{f,i}) = \mathsf{HashState}(\mathsf{st}_{f,i}) = h_{\mathsf{st}_{f,i}}$ or $\mathsf{HashModel}(m'_i) = \mathsf{HashModel}(m_i) = h_{m_i}$, which we assume to happen only with negligible probability.

Game G_2. In G_2, we check whether the data record (u, d) output by $\mathcal{A}$ is contained in the underlying datasets U_i of the k-th and all subsequent iterations. We will show that

$$|\Pr[\mathsf{G}_2 \Rightarrow 1] - \Pr[\mathsf{G}_1 \Rightarrow 1]| \leq \mathsf{negl}(\lambda) \ .$$

For this we first look again at the SNARK proof π_i and the underlying circuits. If a proof of training is performed, the adversary must prove that $h_{U_i} = h_{U_{i-1}}$. This implies—assuming no hash collision occurs—that $\mathcal{H}_{U_i} = \mathcal{H}_{U_{i-1}}$ and $U_i = U_{i-1}$. If a proof of unlearning is performed, the SNARK proof ensures that $\mathcal{H}_{U_{i-1}} \subset \mathcal{H}_{U_i}$ and thus $U_{i-1} \subset U_i$, again using collision-resistance of the hash function. Thus, if $(u, d) \in U_k$, it must also be true that $(u, d) \in U_{k+1}$, ..., $(u, d) \in U_\ell$. By soundness of the SNARK, the adversary cannot prove a false statement, so the above claims must hold.

We also know that $(u, d) \in U_k$ since the proof of non-membership consists of the path from the hashed data record (u, d) to the hash h_{U_k} contained in the k-th commitment. Since the adversary can only win if the proof verifies successfully, we know that in this case the hash value of (u, d), in the following denoted by $h_{u,d} \coloneqq \mathsf{HashDataRecord}(u, d)$, must be a node in the hash chain constructed from $\mathcal{H}_{U_k}$. Unless the adversary finds another data record (u', d') such that $\mathsf{HashDataRecord}(u', d')$ maps to the same hash value $h_{u,d}$—which happens with negligible probability—the record (u, d) must be contained in U_k.

Finally, we show that $\Pr[\mathsf{G}_2 \Rightarrow 1] \leq \mathsf{negl}(\lambda)$. For this, recall that π_i also attests that no unlearnt data point is contained in the dataset, in particular that the intersection $\mathcal{H}_{D_i} \cap \mathcal{H}_{U_i}$ is empty. Together with the fact that the commitments h_{D_i} and h_{U_i} are constructed from $\mathcal{H}_{D_i}$ and $\mathcal{H}_{U_i}$ (due to soundness of the SNARK), the hashed datasets must have been obtained from the corresponding dataset D_i and U_i (unless the adversary has found a collision in the hash function). Combining with previous results, this implies that $D_i \cap U_i = \emptyset$ for all $i \in [\ell]$. As shown above, we know that $(u, d) \in U_\ell$. The final winning condition requires that $(u, d) \in D_\ell$. This cannot be the case since it would contradict the fact that the intersection of the two sets is empty, which proves the final claim.

Collecting the probabilities yields

$$\Pr[\mathsf{GameUnlearn}_{\mathcal{A},\mathcal{E},\Phi_f,\mathcal{D}}(1^\lambda)] \leq \mathsf{negl}(\lambda) \ ,$$

which concludes the proof of Theorem 2. □