Deceptive Deletions for Protecting Withdrawn Posts on Social Media Platforms

Mohsen Minaei*¶†, S Chandra Mouli*‡, Mainack Mondal§, Bruno Ribeiro‡, Aniket Kate‡
† Visa Research, Email: mominaei@visa.com
§ IIT Kharagpur, Email: mainack@cse.iitkgp.ac.in
‡ Purdue University, Email: {chandr, ribeirob, aniket}@purdue.edu

***Abstract*—Over-sharing poorly-worded thoughts and personal information is prevalent on online social platforms. In many of these cases, users regret posting such content. To retrospectively rectify these errors in users' sharing decisions, most platforms offer (deletion) mechanisms to withdraw the content, and social media users often utilize them. Ironically and perhaps unfortunately, these deletions make users more susceptible to privacy violations by malicious actors who specifically hunt post deletions at large scale. The reason for such hunting is simple: deleting a post acts as a powerful signal that the post might be damaging to its owner. Today, multiple archival services are already scanning social media for these deleted posts. Moreover, as we demonstrate in this work, powerful machine learning models can detect damaging deletions at scale.**

Towards restraining such a global adversary against users' right to be forgotten, we introduce *Deceptive Deletion*, a decoy mechanism that minimizes the adversarial advantage. Our mechanism injects decoy deletions, hence creating a two-player minmax game between an *adversary* that seeks to classify damaging content among the deleted posts and a *challenger* that employs decoy deletions to masquerade real damaging deletions. We formalize the Deceptive Game between the two players, determine conditions under which either the adversary or the challenger provably wins the game, and discuss the scenarios in-between these two extremes. We apply the *Deceptive Deletion* mechanism to a real-world task on Twitter: hiding damaging tweet deletions. We show that a powerful global adversary can be beaten by a powerful challenger, raising the bar significantly and giving a glimmer of hope in the ability to be *really* forgotten on social platforms.

I. Introduction

Every day, millions of users share billions of (often personal) posts on online social media platforms like Facebook and Twitter. This information is routinely archived and analyzed by multiple third parties ranging from individuals to state-level actors [22], [29], [48], [49], [71], [74], [75], [84]. Although the majority of these social media posts are benign, users also routinely post regrettable content on social media [24], [80], [94] that they later wish to retract. Subsequently, most social platforms provide user-initiated deletion mechanisms that allow users to rectify their sharing decisions and delete past posts. Not surprisingly, users take advantage of these deletion mechanisms enthusiastically—Mondal et al. [65] showed that nearly one-third of six-year-old Twitter-posts were deleted. In another work, Tinati et al. [83] showed that this number is much higher in Instagram, where almost half of the pictures posted within a six month period had been removed.

Ironically, current user-initiated deletion mechanisms may have an unintended effect: third-party archival services can identify deleted posts and infer that deleted posts might contain *damaging* content from the post creator's point of view (i.e., having an adverse effect on the personal/professional life of the content creator). In other words, deletion might inadvertently make it easier to identify damaging content. Indeed, today it is possible to detect deletions at scale: Twitter, for one, advertises user deletions in their streaming API[1] via deletion notifications [7], [8] so that third-party developers can remove these posts from their database. Similarly, Pushshift [16], [25] is an archival system for all the contents on Reddit and Removeddit [18] uses this archive to publicize all the deleted posts and comments on Reddit. A malicious data-collector can simply leverage these notifications to flag deleted posts as possibly *damaging* and further use them against the users [5], [6], [91]. Importantly, the hand-picked politicians and celebrities are *not* the only parties at the receiving end of these attacks. We find that the malicious data-collector can develop learning models to automate the process and perform an non-targeted (or global) attack at a large-scale; e.g., Fallait Pas Supprimer [13] (i.e., "Should Not Delete" in English) is a Twitter account that collects and publishes the deleted tweets of not only the French politicians and celebrities but also noncelebrity French users with less than a thousand followers.

Asking the users not to post regrettable content on social platforms in the first place may seem like a good first step. However, users cannot accurately predict what content would be damaging to them in the future (e.g., after a breakup or before applying to a job). Zhou et al. [94] and Wang et al. [87] propose multiple types of classifiers (Naive Bayes, SVM, Decision Trees, and Neural Networks) to detect regrettable posts using users' history and to proactively advise users even before the publication of posts. However, this proactive approach cannot prevent users from publishing future-regrettable posts. It is inevitable to focus on *reactive* mechanisms to assist users with protecting their post deletions.

Recently Minaei et al. [62] proposed an intermittent withdrawal mechanism to tackle this challenge of hiding user-initiated deletions. They offer a deniability guarantee for user-initiated deletions in the form of an availability-privacy trade-off and ensure that when a post is deleted, the adversary

* Both authors contributed equally and are considered co-first authors.
¶ This work was done while this author was at Purdue University.

Network and Distributed Systems Security (NDSS) Symposium 2021
21-25 February 2021, Virtual
ISBN 1-891562-66-5
https://dx.doi.org/10.14722/ndss.2021.23139
www.ndss-symposium.org

[1] Twitter provides a random sample of the publicly posted Twitter data in real time to the third parties via streaming API.

cannot be immediately certain if it was actually deleted or temporarily made unavailable by the platform. Their trade-off could be useful for future social and archival platforms; however, in current commercial social media platforms like Twitter, sacrificing even a small fraction of availability for *all the posts* is undesirable.

To this end, our research question is straightforward, yet highly relevant—*can we enhance the privacy of the deleted and possibly damaging posts at scale without excessively affecting the functionality of the platform?*

Contributions. We make the following contributions.

First, we demonstrate the impact of deletion detection attacks by performing a proof-of-concept attack on real-world social media posts to identify damaging content. Specifically, we use a crowdsourced labeled corpus of deleted posts from Twitter to train an adversary (a classifier). We demonstrate that our adversary is capable of detecting damaging posts with high probability (an increase of 27 percentage points in its F-score). Thus, it is feasible for the adversary to use automated methods for detecting damaging posts on a large scale. In fact, we expect systems such as Fallait Pas Supprimer [13] to employ analogous learning techniques soon to improve their detection.

Second, to overcome the problem of detecting damaging deletions, we introduce a novel deletion mechanism, Deceptive Deletions, that raises the bar for the adversary in identifying damaging content. Given a set of damaging posts (i.e., posts that adversary can leverage to blackmail the user) that users want to delete, the Deceptive Deletion system (also known as a challenger) carefully *selects* k additional posts for each damaging post and deletes them along with the damaging posts. The system-selected posts, henceforth called the *decoy posts*, are taken from a pool of posts (i.e., non-damaging non-deleted) provided by volunteers. The deletions of the decoy posts will confuse the adversary in distinguishing damaging posts from the (non-damaging) decoy posts. Intuitively, Deceptive Deletion is more effective if the selected decoy posts are similar to the damaging posts. These two opposite goals create a minmax game between the adversary and the challenger that we further analyze.

Third, we introduce the Deceptive Learning Game, which formally describes the minmax game between the adversary and the challenger. We start by considering a static adversary that tunes the parameters of its system (e.g., classifier for determining the damaging posts) up until a certain point in time. However, powerful adversaries are adaptive and continually tune their models as they obtain more deletions including the decoy deletions made by the challenger. Therefore, in the second phase, we consider an adaptive adversary and describe the optimization problem of the adaptive adversary and challenger as a minmax game.[2]

We identify conditions under which either only the adaptive adversary or only the challenger provably wins the minmax game and discuss the scenarios in-between these two extremes. To the best of our knowledge, this is the first attempt to develop a computational model for quantitative assessment of the damaging deletions in the presence of both static and adaptive adversaries.

[2]See [51] for another example of a minmax game in adversarial learning.

Finally, we empirically demonstrate that with access to a set of non-damaging volunteered posts, we can leverage Deceptive Deletions to hide damaging deletions against both static and adaptive adversary effectively. We use real-world Twitter data to demonstrate the effectiveness of the challenger. Specifically, we show that even when we consider only two decoy posts per damaging deletion, the adversarial performance (F-score) drops to 42% from 75% in the absence of any privacy-preserving deletion mechanism.

II. Background and Related Work

A. Exisiting Content Deletion Mechanisms to Provide Privacy

Today, most archival and social media websites (e.g., Twitter, Facebook) enable users to delete their content. Recent studies [20], [60], [65] show that a significant number of users deleted content—35% of Twitter posts are deleted within six years of posting them. This user-initiated deletion is also related to the "Right to be Forgotten" [88], [91]. However, this user-initiated content deletion suffered from the *Streisand effect* – attempting to hide some information has the unintended consequence of gaining more attention [91]. Consequently, there is a need to provide *deletion privacy* to users.

In addition to user-initiated deletions, there exist some premeditated withdrawal mechanisms where all historical content is eventually deleted automatically to provide deletion privacy. These mechanisms can be broadly classified into two categories. First, in *age-based withdrawal*, platforms like Snapchat [1] and Dust [4] and systems like Vanish [42], [43] and EphPub [28] automatically withdraw a piece of content after a preset time. Second, to make premeditated withdrawal more usable, Mondal et al. [65] proposed *inactivity-based withdrawal*, where posts will be withdrawn only if they become inactive, i.e., there is no interaction with the post for a specified time period (e.g., no more views by other users).

However, even the premeditated withdrawals are not free from problems of their own. First, all the posts will eventually get deleted, removing all archival history from the platform. Second, if posts are deleted before the preset time or in-spite of high interaction, the adversary can be certain that the deletion was user-intended, violating deletion privacy.

Minaei et al. [61], [62] presented a new intermittent withdrawal mechanism for all non-deleted posts, which provides a trade-off between availability and deletion privacy. In a nutshell, their system ensures that if an adversary found that a post is not available, then the adversary cannot be certain if the post is user-deleted or simply taken down by the platform temporarily. Although this mechanism is useful for large internet archives, in platforms such as Facebook and Twitter, where content availability is crucial to the users and platform, a privacy-availability trade-off might not be feasible. Furthermore, the intermittent withdrawal mechanism does not consider the adversary's background knowledge about other deleted posts. Our work aims to bridge this gap and provide a novel learning-based mechanism which considers an adaptive adversary who aims to uncover tweet deletion.

Tianti et al. [83] offer intuitions for predicting posts deletions on Instagram with the goal of managing the storage of posts on the servers: Once a post is archived, it becomes

computationally expensive to erase it; thus, predicting deletions can help in reducing the overheads of being compliant with the "right to be forgotten" regulations. These predictions in the non-adversarial setting, however, does not apply to our minmax game between the adversary and the challenger.

Garg et al. [41] formalize the right to be forgotten using platforms as a cryptographic game. While interesting, their definitions and suggested tools such as history-independent data structures are not applicable to our setting where the adversary has continuous access to the collected data.

B. Obfuscation using Noise Injection

Our mechanism is not without precedent, and it is inspired by earlier work of obfuscation by noise injection. There has been a line of work in the area of (non-cryptographic) private information retrieval [38], [47], [66], [69] that obfuscates the users' interest using dummy queries as noise to avoid user profiling. Howe et al. proposed TrackMeNot [9], [47], which issues randomized search queries to prevent the search engines in building any practical profile of the users based on their actual queries. Similar works [38], [66], generate $k-1$ other queries (dummy ones) for each user query and submit all k queries at the same time. We note that all of the systems mentioned so far consider hiding each query separately. However, a determined adversary may be able to find a user's interests by observing a sequence of such obfuscated queries. Multiple works have investigated such weaknesses [23], [69], [70].

Some relatively new techniques further try to overcome these shortcomings by *smartly* generating the $k-1$ queries. For example, Petit et al. proposed PEAS [72], where they provide a combination of unlinkability and indistinguishability. However, apart from introducing an overhead for encrypting the user queries, their method also requires two proxy servers that are non-colluding, hence weakening the adversarial model. K-subscription [67] is yet another work that proposes an obfuscation based approach that enables the user to follow privacy-sensitive channels in Twitter by requiring the users to follow $k-1$ other channels to hide the user interests from the microblogging service. However, the K-subscription has a negative social impact for the user as the user's social connections will see the user following these dummy channels. These shortcomings, both social and technical, motivated our particular design decision for Deceptive Deletions.

C. Adversarial Machine Learning

Traditional adversarial learning settings [32] involve two players: a classifier and an attacker. The classifier seeks to label the inputs whereas the attacker tries to *modify* the inputs such that the classifier will misclassify them. Adversarial machine learning has also been used as a defense with the roles reversed where the defender attacks the adversary's classifier. For example, in [50], the adversary tries to extract users private attributes from their public data while the defender modifies the public data of the users in order to fool the adversarys classifier. Our setting is different in that we are *not allowed to modify* the examples. Instead, the challenger wishes to attack the adversary's classifier by injecting *hard-to-classify* examples into the adversary's train/test datasets (i.e., the deletion set). A key constraint for the challenger is that it has to *select the examples from a preexisting set* of volunteered posts. This is because the challenger can only delete existing posts, and cannot generate fake posts.

As we detail in the subsequent sections, the adaptive adversary trains on these injected examples as well. With a faint relation to our work, data poisoning attacks [58], [81] focus primarily on injecting poisoned samples into a classifier's training data with the sole purpose of deteriorating the classifier. In contrast, our primary goal is to inject examples only into the adversary's test dataset, especially because data poisoning attacks typically require the freedom to arbitrarily construct data samples, which is not possible in our setting.

III. System Model and Overview

A. System

We consider a data-sharing *platform* (e.g., Twitter or Facebook) as the public bulletin board where individuals can upload and view content. *Users* are the post owners that are able to publish/delete their posts, and view posts from other users. In this work, we consider discrete time intervals in which the users upload and delete posts (Figure 1 ❶). A time interval could be as small as a minute or even a week, depending on the platform. We define two types of posts.

- **User-deleted posts** A user could delete a post for two primary reasons [20], [62], [65]:
  - **Damaging posts:** the post contained *damaging* content to the user's personal or professional life, or
  - **Non-damaging posts:** the post was out-dated, contained spelling mistakes, etc.

  An adversary's goal is to find the damaging posts among all the deleted ones that could be used to blackmail the corresponding owners of the post.
- **Volunteered posts** We consider a subset of non-deleted posts that users *willingly* offer to be deleted to protect the privacy of other users whenever needed. These volunteered posts are non-damaging and cannot be used by the adversary to blackmail the user of the post. We discuss the challenges of obtaining volunteered posts in Section VI.
  A challenger's goal is to select a subset of volunteered posts (i.e., non-damaging) and delete them such that the aforementioned adversary is unable to distinguish between the damaging and the non-damaging post deletions. We denote the posts selected by the challenger as ***decoy posts****.*

Notation. We use a subscript t to denote the time interval and superscripts $\delta, +, \mathrm{v}, *$ to denote the post type. In particular, $\mathbb{D}_t$ is all the uploaded and deleted posts in time interval t. Then we denote all the deleted posts (user- and challenger-deleted) in that interval as $\mathbb{D}_t^{\delta}$, the **damaging posts** as $\mathbb{D}_t^{+}$, and **volunteered posts** by $\mathbb{D}_t^{\mathrm{v}}$. The **decoy posts** that a challenger selects for deletion to fool the adversary is denoted by $\mathbb{G}_t^{*}$. Note that $\mathbb{G}_t^{*} \subseteq \mathbb{D}_t^{\mathrm{v}} \subseteq \mathbb{D}_t \backslash \mathbb{D}_t^{\delta}$.

B. Adversary's Actions and Assumptions

Task. At a given time interval, the task of the adversary is to correctly label all the deleted posts as being damaging to the post-owner or not. We *do not* focus on local attackers (or

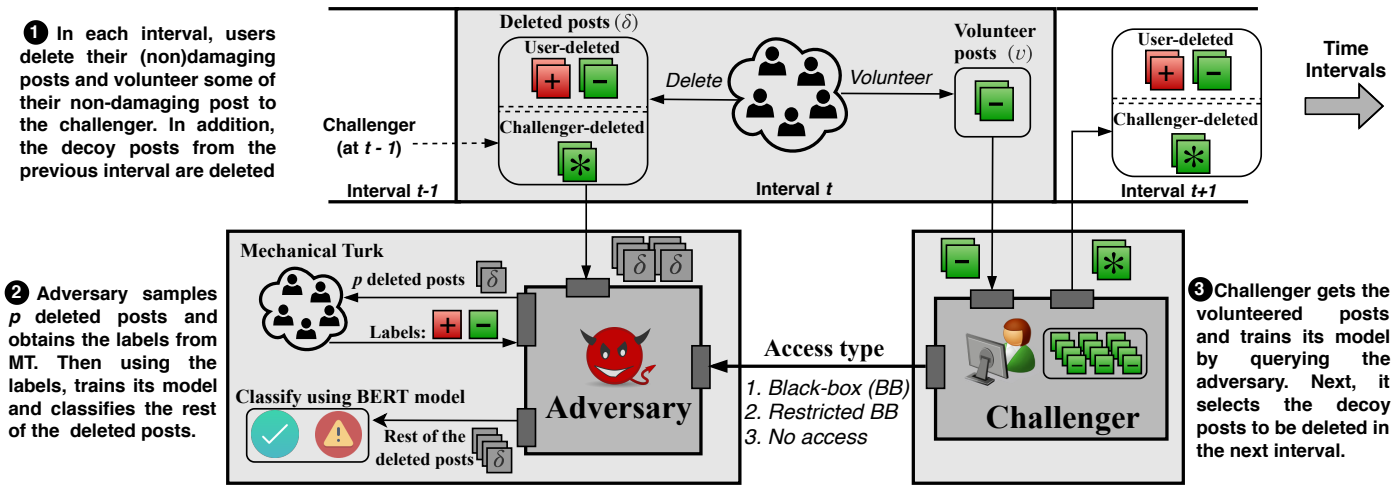


Fig. 1: Overview of Deceptive Deletions. In each interval, the deletions are shown by gray squares with 'δ'. The deleted posts could be of three types: users' damaging deletions shown by red squares with '$+$', users' non-damaging deletions shown by green squares with '$-$' and challengers' decoys posts shown by green squares with '$*$'. Further, we denote the volunteer posts offered to the challenger during each interval by green squares with '$-$' to indicate that they are non-damaging.

stalkers) targeting individuals or small groups of users.[3] Our global adversary instead seeks damaging deletions on a large scale, rummaging through all the deleted posts to find as many damaging ones as possible. Fallait Pas Supprimer [13] (from Section I) is a real-world example of the global adversary.

Data access. At any given time interval, we assume that the adversary is able to obtain all the deleted posts by comparing different archived snapshots of the platform. Although this strong data assumption benefits the adversary tremendously, we show in Section V-D that Deceptive Deletions can protect the users' damaging deletions. Further, we discuss a few techniques that the platforms can use to restrict and limit the adversary's access to the users' profile in Section VI-C.

Labels. Our global, non-stalker adversary is not able to obtain the *true label* (damaging or non-damaging) of the post from the user. Instead, the adversary uses a crowdsourcing service like Mechanical Turk (MTurk) [21] to obtain a proxy for these true labels. Although the labels obtained from the Mechanical Turkers (MTurkers) reflect societal values and not the user's intention, following previous work [87], we assume they closely match the *true labels* in our experiments. This is reasonable as the adversary can expend a significant amount of effort and money to obtain these *true labels*, at least for a small set of posts, that will ultimately be used to train a machine learning model.

Budget. Since there is a cost associated with acquiring label for each deleted post from the MTurkers, the aim of the adversary is to *learn to detect the damaging deletions* under a budget constraint. We consider two types of budget constraints:

- **limited budget** where the adversary can only obtain the labels for a fixed number of posts B_{static}, and
- **fixed recurring budget** where the adversary obtains the labels for a fixed number of posts B_{adapt} *in each interval*.

The adversary with a limited budget is called the **static adversary** since it does not train after exhausting its budget. On the other hand, the adversary with a fixed recurring budget keeps adapting to the new deletions in each time interval, and hence is dubbed the **adaptive adversary**.

Player actions. At every time interval t, the adversary obtains a set of posts $\mathbb{A}_t^\delta$ for training by sampling part of the deleted posts, say p, from $\mathbb{D}_t^\delta$, an operation denoted by $\mathbb{A}_t^\delta \overset{p}{\sim} \mathbb{D}_t^\delta$. The adversary uses MTurk to label the sampled dataset $\mathbb{A}_t^\delta$. After training, the task of the adversary is to classify the rest of the deleted posts of that time interval. Additionally, as the adversary gets better over time, it also *relabels* all the posts deleted from the past intervals. The test set for the adversary is all the deleted posts from current and previous time intervals that were not used for training; i.e., $\bigcup_{t' \le t}(\mathbb{D}_{t'}^\delta \setminus \mathbb{A}_{t'}^\delta)$. Figure 1 ❷ shows the adversary's actions.

Note that although an adaptive adversary can sample $p = B_{\text{adapt}}$ deleted posts at *every* time interval and use MTurkers to label them, a static adversary can only obtain the labels until it runs out of the limited budget (after $\tau = B_{\text{static}}/p$ time intervals). After this period, a static adversary does not train itself with new deleted posts.

Performance metrics. The adversary wishes to increase *precision* and *recall* for the classification of deleted posts into damaging and non-damaging sets. At every time interval t, we report adversary's F-score[4] over the test set described above: deleted posts of all the past intervals, i.e., $\bigcup_{t' \le t}(\mathbb{D}_{t'}^\delta \setminus \mathbb{A}_{t'}^\delta)$.

[3] Such stalkers can easily label their posts manually, and protecting against such an attack is extremely hard if not impossible. For example, consider that a stalker continuously takes snapshots of its targeted user profile with the goal of identifying the user's deletions. With its background/auxiliary information about the user (i.e., knowing what contents are considered sensitive to the target), it can effectively identify the damaging deletions. We claim that, in this full-information model, protection against such a local adversary is impossible.

[4] F-score = $2 \cdot precision \cdot recall/(precision + recall)$

C. Challenger's Actions and Assumptions

Task. In the presence of an adversary as described above, the task of a challenger is to obtain volunteered posts (i.e. non-damaging and non-deleted posts) from users, select a subset of these posts and delete them in order to fool the adversary into misclassifying these *challenger-deleted* posts as damaging. The challenger is honest, does not collude with the adversary, and works with the users (data owners) to protect their damaging deletions. Other than the platforms themselves, third party services such as "tweetDelete" [10] can take the role of the challenger as well. In Section VI-D, we discuss the flaws in a possible alternate approach where the challenger is allowed to *generate* tweets rather than *select* from pool of volunteered posts.

Data access. The challenger can be implemented by the platform or a third-party deletion service [3], [10], [11], that has access to the posts of the users. Additionally, we assume that there are users over the platform who volunteer a subset of their non-damaging posts to be deleted anytime (or within a time frame) by the challenger, possibly, in return for privacy benefits for their (and other users') damaging deletions.

Labels. The challenger is implemented as part of the platform (or a third-party service permitted by the user). Thus, unlike the adversary that obtains proxy labels from crowdsourcing platforms, it has access to the *true labels*— damaging or non-damaging, from the owner of the post. This is easily implemented: before deleting a post, the user can specify whether the post is damaging (and needs protection). This access to the *true labels* is an advantage that challenger has over the adversary and hence can train more accurate models.

Access to the adversary. The challenger not only knows the presence of a global adversary trying to classify the deleted posts into damaging and non-damaging posts but also can observe its behaviour.[5]. As a result, we consider three types of accesses to the adversary:

- **no access** where the challenger has no information about the adversary.
- **monitored black-box access with a recurring query budget of** B_g where the challenger can obtain the adversary's classification probability for a limited number of posts B_g every time interval, but the access is *monitored*, i.e., the adversary can take note of every post queried and treat them separately.
- **black-box access** where the challenger can obtain the adversary's classification probabilities for any post.

Here, *no access* is the weakest assumption that defines the lower-bounds for our challenger's success. Nevertheless, we expect the challenger to have some access to the adversary's classification. An *unrestricted* black-box access serves as an upper bound for the challenger assuming that it can train a precise surrogate model of the adversary's classifier using its own training data. While employing such a surrogate model is common practice in the literature [54], [68], it can be hard to obtain in real world without knowing the adversary's exact architecture and training data. Our monitored black-box assumption with a recurring query budget (henceforth, interchangeably called the restricted black-box access) balances practicality of the access versus the feasibility of defending against an adversary with that access. In Section IV, we introduce three challengers (oracle, D^2 and random) corresponding to the three types of accesses.

[5]Fallait Pas Supprimer [13] posts all its output on Twitter itself.

Player actions. At every time interval t, the challenger receives new volunteer posts from the users and adds them to a set that stores the volunteered posts collected up until this point. Next, based on the type of access, it obtains the adversary's classification probabilities for some number of volunteer posts (the number is dependent on the access which we detail in Section IV). Finally, it selects *decoy posts*, a subset of the volunteered posts collected up until this point and deletes these posts in interval $t{+}1$ (hence the adversary sees these *challenger-deleted* posts in interval $t{+}1$ as part of the deleted set $\mathbb{D}^{\delta}_{t+1}$). Figure 1 ❸ shows the challenger's actions.

Performance metrics. The challenger, in direct contrast to the adversary, wishes to *decrease* adversary's precision and recall for the classification of deleted posts. Adversary's precision will decrease if it classifies the injected decoy posts as damaging (increased false-positives). On the other hand, adversary's recall will decrease if it learns to be conservative in order to ignore the decoy posts (increased false-negatives).

IV. The Deceptive Learning Game

The deceptive learning game is a two-player zero-sum non-cooperative game over time intervals $t = 1, 2, ...$ (units) between an adversary who wishes to *find* users' damaging deletions, and a challenger who wishes to *hide* the said damaging deletions. The challenger achieves this by deleting volunteers' non-damaging posts as decoys. While the adversary's goal is to maximize its precision/recall scores on the classification task, the challenger's goal is to minimize them.

We denote each post by (x, y), where $x \in \mathbb{X}$ represents the features of the post (i.e., text, comments, etc.) and $y \in \{0, 1\}$ denotes its true label such that $y = 1$ if the post is damaging and $y = 0$ if it is non-damaging. In the following subsections, we describe the actions of each player in the time interval t.

A. Adversary

We denote the adversary's classifier at the beginning of interval t by $a(\ \cdot\ ; \theta_{t-1}) : \mathbb{X} \to [0, 1]$ parameterized by θ_{t-1} such that $a(x; \theta_{t-1}) \coloneqq P(\hat{y} = 1 \mid x; \theta_{t-1})$ is the predicted probability of the post x being damaging. The adversary collects all the deletions that happen in this interval (i.e., $\mathbb{D}^{\delta}_t$) and samples p posts, denoted by $\mathbb{A}^{\delta}_t$. The adversary then uses MTurk to obtain a proxy for the true labels of these p posts.

The adversary uses this labeled training data in the following optimization problem to update its parameters,

$$\theta_t = \arg\min_{\theta} \mathcal{L}_{\text{NLL}}(\theta; \mathbb{A}^{\delta}_t)\,, \tag{1}$$

where $\mathcal{L}_{\text{NLL}}$ is the standard negative log-likelihood loss for the classification task, given by,

$$\mathcal{L}_{\text{NLL}}(\theta; \mathbb{A}^{\delta}_t) = \sum_{(x,y)\in\mathbb{A}^{\delta}_t} -y \log\left(a(x;\theta)\right) - (1-y)\log\left(1 - a(x;\theta)\right)\,.$$

After training, the adversary uses the trained model $a(\ \cdot\ ; \theta_t)$ to predict the labels of the rest of the deleted posts of time

Algorithm 1: Adversary

input : $\mathbb{D}^{\delta}$; /* Deleted posts in this interval */

Sample p posts $\mathbb{A}^{\delta} \overset{p}{\sim} \mathbb{D}^{\delta}$;

Query MTurk and obtain labels for $\mathbb{A}^{\delta}$;

Obtain optimal parameters θ^* by solving Equation (1) ;

return $a(\ \cdot\ ; \theta^*)$

interval t, i.e., $\mathbb{D}^{\delta}_t \setminus \mathbb{A}^{\delta}_t$ along with all the deleted posts that it had already predicted in the past. This way the adversary hopes to capture damaging posts that were missed earlier. Hence, we report the adversary's performance on all the past deletions (not including the training data): $\bigcup_{t' \leq t}(\mathbb{D}^{\delta}_{t'} \setminus \mathbb{A}^{\delta}_{t'})$.

Static vs Adaptive Adversary. Since the static adversary has a limited budget, first it chooses the number of time intervals for training, say τ, and accordingly samples $p = B_{\text{static}}/\tau$ posts for querying MTurk to obtain labels. The adaptive adversary has a fixed recurring budget of B_{adapt} and hence, can sample $p = B_{\text{adapt}}$ posts every interval. This allows the adaptive adversary to train itself with new training data (of size B_{adapt}) every interval indefinitely. Algorithm 1 depicts adversary's actions within a time interval (subscript t removed for clarity).

B. Challenger

In the presence of such an adversary, the challenger's goal is to collect volunteered posts (non-damaging) from users and selectively delete these posts in order to confuse the adversary.

As described before, $\mathbb{D}^{\text{v}}_t$ is the set of posts volunteered by users in the time interval t. Let $\mathbb{G}^*_{\leq t}$ be the set of decoy posts deleted by the challenger in the current and past intervals. At the end of interval t, the challenger collects all the volunteered posts from the current and past intervals (except the posts that it has already used as decoys). The *available* set of volunteered posts is denoted by $\mathbb{D}^{\text{v}}_{\leq t} \equiv (\bigcup_{t' \leq t} \mathbb{D}^{\text{v}}_{t'}) \setminus (\bigcup_{t' \leq t} \mathbb{G}^*_{t'})$. Note that $(x, y) \in \mathbb{D}^{\text{v}}_{\leq t} \implies y = 0$, i.e., the volunteered posts are non-damaging by definition. For ease of notation, let $N^{\text{v}} := |\mathbb{D}^{\text{v}}_{\leq t}|$ be the number of volunteered posts collected till interval t.

Then, the goal of the challenger is to construct the decoy set $\mathbb{G}^*_{t+1} \subseteq \mathbb{D}^{\text{v}}_{\leq t}$ and delete these posts during the next time interval $t{+}1$ in order to fool the adversary into misclassifying these challenger-deleted non-damaging posts as user-deleted damaging posts. Formally, we want to choose K decoy posts (denoted by a K-hot vector $\boldsymbol{w}$) that maximizes the negative-log likelihood loss for the adversary's classifier, given by the following optimization problem,

$$\boldsymbol{w}^* = \arg\max_{\boldsymbol{w}} V(\boldsymbol{w}; \mathbb{D}^{\text{v}}_{\leq t})$$
$$\text{s.t.} \quad ||\boldsymbol{w}||_1 = K, \qquad \boldsymbol{w} \in \{0,1\}^{N^{\text{v}}}, \tag{2}$$

where

$$V(\boldsymbol{w}; \mathbb{D}^{\text{v}}_{\leq t}) = \sum_{i=1}^{N^{\text{v}}} -w_i \cdot \log(1 - a(x_i; \theta_t)), \tag{3}$$

and x_i is the i-th volunteered post in $\mathbb{D}^{\text{v}}_{\leq t}$. The cost function $V(\boldsymbol{w}; \mathbb{D}^{\text{v}}_{\leq t})$ in Equation (3) is simply the negative log-likelihood of the adversary over the set $\mathbb{D}^{\text{v}}_{\leq t}$ weighted by a K-hot vector $\boldsymbol{w}$. Equation (3) uses the fact that the set only contains non-damaging posts (i.e., $y_i = 0$).

Algorithm 2: Challenger

input : $\mathbb{D}^{\text{v}}$, K, accessType

$\mathbb{G}^* \leftarrow \emptyset$;

if *accessType* = *none* **then**

  /* Random challenger */

  $\mathbb{G}^* \overset{K}{\sim} \mathbb{D}^{\text{v}}$;

else if *accessType* = *black-box* **then**

  /* Oracle challenger */

  $\mathbb{G}^* \leftarrow \{x_i : x_i \in \mathbb{D}^{\text{v}} \wedge a(x_i; \theta)$ is in the top $K\}$;

else if *accessType* = *monitored black-box (budget B_g)* **then**

  /* D^2 challenger */

  Sample B_g posts for training $\mathbb{D}^{\text{v,train}} \overset{B_g}{\sim} \mathbb{D}^{\text{v}}$;

  $\mathbb{D}^{\text{v,test}} \leftarrow \mathbb{D}^{\text{v}} \setminus \mathbb{D}^{\text{v,train}}$;

  Query $a(x_i; \theta)$ for all $(x_i, y_i = 0) \in \mathbb{D}^{\text{v,train}}$;

  Obtain optimal parameters ϕ^* by solving Equation (4) ;

  $\mathbb{G}^* \leftarrow \{x_i : x_i \in \mathbb{D}^{\text{v,test}} \wedge g(x_i; \phi^*)$ is in the top $K\}$;

return $\mathbb{G}^*$;

Consequently, $\boldsymbol{w}^*$ optimized in such a fashion selects K posts from the set $\mathbb{D}^{\text{v}}_{\leq t}$ that *maximizes* the adversary's negative log-likelihood loss. The set of K selected posts can be trivially constructed as $\mathbb{G}^*_{t+1} = \{x_i : i \in \{1, \ldots, N^{\text{v}}\} \wedge w_i = 1\}$. The challenger deletes $\mathbb{G}^*_{t+1}$ over the next time interval $t{+}1$ (hence the adversary sees these posts as part of the deleted set $\mathbb{D}^{\delta}_{t+1}$). Note that the challenger uses the adversary's classifier $a(\ \cdot\ ;\ \theta_t)$ to create decoy posts for $t{+}1$. However, as per Section IV-A, in interval $t{+}1$ the adversary first trains over a sample of the deleted posts (including the decoy posts) and updates its classifier to $a(\ \cdot\ ; \theta_{t+1})$ before classifying the rest of the deleted posts of $t{+}1$. Hence, the challenger is always at a disadvantage (one step behind).

Next, we describe three challengers corresponding to the access types discussed in Section III-C: *no access*, *black-box access* and *monitored black-box access with a query budget*.

Random challenger (no access). We begin with the case where the challenger has *no access* to the adversary's classifier and there is no side-information available to the challenger. With no access to the adversary's classification probabilities $a(\ \cdot\ ; \theta_t)$, the optimization problem in Equation (2) cannot be solved. We introduce the naive *random challenger* that simply samples K posts randomly from the available volunteered posts $\mathbb{D}^{\text{v}}_{\leq t}$ and deletes them, i.e., $\mathbb{G}^*_{t+1} \overset{K}{\sim} \mathbb{D}^{\text{v}}_{\leq t}$. This is the only viable approach if the challenger has no information about the adversary's classifier.

Oracle challenger (black-box access). Next we consider the challenger that has a black-box access to the adversary's classifier with no query budget, i.e., at any time interval t, the challenger can query the adversary with a post x and expect the adversary's predicted probability $a(x; \theta_t)$ in response without the adversary's knowledge. Armed with the black-box access, oracle challenger can simply maximize Equation (2) by choosing the top K posts with highest values for $a(x_i; \theta_t)$.

D^2 Challenger (monitored black-box access with query budget B_g). The oracle challenger assumes an *unmonitored* black-box access to the adversary *with an infinite query budget* which can be hard to obtain in practice. In what follows, we relax the access and assume a *monitored* black-box access *with a recurring query budget of B_g*. In other words, queries to the adversary, while being limited per interval, are also monitored and possibly flagged by the adversary. The adversary can simply take note of these queries as performed by a potential challenger, hence negating any privacy benefits from injecting decoy posts. Whenever the adversary sees a *deleted post* identical to one that it was previously *queried* about, it can ignore the post as it is likely non-damaging.

Here we design a challenger, henceforth dubbed D^2, that *trains to select decoy posts* from any given volunteered set. In other words, the D^2 challenger makes use of the monitored black-box access to the adversary only during training. Hence it can be used to find the decoy posts without querying the adversary; for example in a held-out volunteered set (separate from the training set). Additionally, the D^2 challenger queries the adversary for only B_g posts every time interval.

We denote the challenger's model at the beginning of interval t by $g(\ \cdot\ ; \phi_{t-1}) : \mathbb{X} \to \mathbb{R}$ parameterized by ϕ_{t-1}. For a given volunteer post x, $g(x; \phi_{t-1})$ gives an unnormalized score for how likely the post will be mislabeled as damaging; higher the score, higher the misclassification probability.

First, the D^2 challenger samples B_g posts for training from the available volunteered set $\mathbb{D}^{\text{v}}_{\leq t}$ collected till interval t. We denote the train and test sets of the D^2 challenger as $\mathbb{D}^{\text{v,train}}_{\leq t}$ and $\mathbb{D}^{\text{v,test}}_{\leq t}$ of sizes B_g and $N^{\text{v}} - B_g$ respectively. Then, the goal of the D^2 is to find optimal parameters ϕ_t by solving a continuous relaxation of Equation (2) presented below,

$$\phi_t = \arg\max_{\phi} \tilde{V}(\phi; \mathbb{D}^{\text{v,train}}_{\leq t}) \tag{4}$$

where

$$\tilde{V}(\phi; \mathbb{D}^{\text{v,train}}_{\leq t}) = \sum_{i=1}^{B_g} -\alpha(x_i; \phi, \mathbb{D}^{\text{v,train}}_{\leq t})\ \log(1 - a(x_i; \theta_t))\ ,$$

and

$$\alpha(x_i; \phi, \mathbb{D}^{\text{v,train}}_{\leq t}) = \frac{\exp\left(g(x_i; \phi)\right)}{\sum_{j=1}^{B_g} \exp\left(g(x_j; \phi)\right)}\ ,$$

is a softmax over the challenger outputs for all the examples in $\mathbb{D}^{\text{v,train}}_{\leq t}$. The softmax function makes sure that $0 \leq \alpha(\ \cdot\ ; \phi, \mathbb{D}^{\text{v,train}}_{\leq t}) \leq 1$ and $\sum_{j=1}^{B_g} \alpha(x_j; \phi, \mathbb{D}^{\text{v,train}}_{\leq t}) = 1$. The continuous relaxation in Equation (4) allows the D^2 challenger to train a neural network model parameterized by ϕ via backpropagation.

We now show that optimizing the relaxed objective in Equation (4) results in the best objective value for Equation (2).

Proposition 1. *For any given volunteered set $\mathbb{D}^{v}$ with N non-deleted posts,*

$$\max_{\phi} \tilde{V}(\phi; \mathbb{D}^{v}) = \max_{w_1, \ldots, w_N} V(w_1, \ldots, w_N; \mathbb{D}^{v})$$

We present proof of the proposition in Appendix B.

Algorithm 3: Deceptive Game

input : accessType, K

1 $\mathbb{G}^*_1 \leftarrow \emptyset$;
2 $\mathbb{D}^{\text{v}}_{\leq 0} \leftarrow \emptyset$;
3 **for** $t \leftarrow 1$ **to** n **do**
4 $\quad \mathbb{D}^{\delta}_t,\ \mathbb{D}^{\text{v}}_t \leftarrow$ Users(t) ; `/* deleted and volunteered posts of the users at interval t */`
5 $\quad \mathbb{D}^{\delta}_t \leftarrow \mathbb{D}^{\delta}_t \cup \mathbb{G}^*_t$; `/* user- and challenger-deleted posts at interval t */`
6 $\quad$ **if** *Adversary's budget has not exhausted* **then**
7 $\quad\quad a(\ \cdot\ , \theta_t) \leftarrow$ Adversary($\mathbb{D}^{\delta}_t$) ;
8 $\quad \mathbb{D}^{\text{v}}_{\leq t} \leftarrow (\mathbb{D}^{\text{v}}_{\leq t-1} \setminus \mathbb{G}^*_t) \cup \mathbb{D}^{\text{v}}_t$; `/* available volunteered set */`
9 $\quad \mathbb{G}^*_{t+1} \leftarrow$ Challenger($\mathbb{D}^{\text{v}}_{\leq t}, K,$ accessType)
10 **end**

Finally, the D^2 challenger with optimal parameters ϕ_t computes $g(x; \phi_t)$ for all $(x, y = 0) \in \mathbb{D}^{\text{v,test}}_{\leq t}$, and constructs G^*_{t+1} by choosing the examples with top K values for $g(\ \cdot\ ; \phi_t)$. Algorithm 2 shows the actions of the challenger within a time interval (subscript t removed for clarity).

C. Deceptive Learning Game

Algorithm 3 presents the game between the adversary and the challenger. In each time interval, users independently delete and volunteer posts (line 4). The platform/deletion-service additionally deletes the challenger-selected decoy posts (line 5). The adversary obtains all the deleted posts and queries the MTurk with a small subset of the posts for labels (if the adversary has not exhausted the budget). With this labeled set of deleted posts, the adversary trains its classifier (lines 6-7). The challenger collects new volunteered posts (line 8) and builds decoy posts to be injected in the next interval (line 9). This results in a real-life game between the adversary and the challenger, where each adapts to the other.

D. Analysis: Who Wins the Game?

In what follows, we analyze the scenarios where either the adversary or the challenger wins the deceptive learning game. We show that the volunteered set, $\mathbb{D}^{\text{v}}$, plays a significant role in deciding the winner of the game. First, we need the definition of support of a distribution.

Definition 1 (Support)**.** *Let $\Omega = \{x : \forall x, p(x) > 0\}$ be the support of distribution $p(x)$, i.e., the set of all possible features x with non-zero probability.*

Let $p^+(x)$ be the distribution of the features of damaging posts, with the corresponding support denoted by Ω^+. Then, a post x is in Ω^+ if there is a non-zero probability that it is a damaging post. Similarly, Ω^{v} is the support of the distribution of volunteered posts p^{v}. Next, we analyze the two extreme scenarios of non-overlapping supports (i.e., $\Omega^{\text{v}} \cap \Omega^+ = \emptyset$) and fully-overlapping supports (i.e., $\Omega^{\text{v}} = \Omega^+$). These extreme scenarios correspond to the following simple questions respectively: (a) "*what if all the posts volunteered by users have completely different features than the damaging posts?*" and

(b) "*what if the volunteered posts have very similar or same features as those of damaging posts?*".

1) Non-overlapping Support: Adversary Wins:

Proposition 2 (*Non-overlapping* support)**.** *Assume $\Omega^v \cap \Omega^+ = \emptyset$, i.e., the supports of volunteered and damaging posts do not overlap. Then, there is always a powerful-enough adversary to defeat the challenger.*

An Illustrative Example: Consider the example provided in Figure 2a. The two classes (denoted by red circles and green crosses respectively) have non-overlapping support. We show the decision boundary of the adaptive adversary in this setting dataset after 50 intervals of the deceptive learning game. We see that the adversary can perfectly label the points even in the presence of the oracle challenger.

Real-world scenario: The non-overlapping case could happen in an online social platform if its users are very conservative in volunteering posts to the challenger. Consider for example, none of the volunteered posts contained any sensitive keyword, whereas all the damaging posts had at least one sensitive keyword, a clear case of non-overlapping supports. In such a scenario, the adversary will win the game as detailed above.

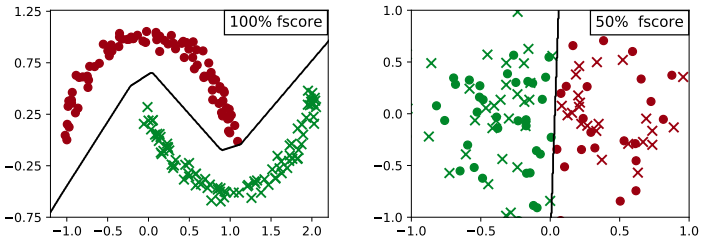
2) Fully-overlapping Support: Challenger Wins:

Proposition 3 (Fully overlapping support)**.** *Assume $\Omega^v = \Omega^+$, i.e., the supports of volunteered and damaging posts fully overlap. Then, given enough volunteered posts in $\mathbb{D}^v$, the challenger always defeats the adversary (in both static and adaptive scenarios). More precisely, if the challenger selects k decoys per damaging post in $\mathbb{D}^\delta$, then the adversary's probability of identifying a damaging post in $\mathbb{D}^\delta$ is in average at most $1/(k+1)$.*

An Illustrative Example: Consider the example provided in Figure 2b where the two classes (red circles and green crosses respectively) have fully overlapping supports (as they are drawn from a Gaussian distribution with different means). We show the decision boundary of the adaptive adversary in this setting after 50 intervals of the deceptive learning game. We see that for any decision boundary, there exist points in Ω^v that a challenger can choose such that the adversary mislabels them as damaging.

Real-world scenario: The fully-overlapping case could happen in an online social platform if the definition of what constitutes as damaging varies across the platform's users. For example, user A could consider a post with a single sensitive word (e.g., a swear word) as damaging, whereas another user B from a different background might consider the same post as completely innocuous and volunteer the post. In such a scenario, the challenger will use volunteered posts from user B to protect the damaging posts of user A. Hence, the challenger will win the game against even the most powerful adversary with infinite data.

Propositions 2 and 3 are important to understand the two extreme cases —where either the challenger clearly wins or the adversary clearly wins— as important insights, even though these clear-cut cases are unlikely to happen in practice.



(a) Non-overlapping supports **(b)** Fully overlapping supports

Fig. 2: Two examples illustrating the two possible scenarios relating to the supports of volunteered posts and damaging posts: non-overlapping (left) and fully overlapping (right). The black line denotes the decision boundary of the adaptive adversary after 50 intervals of the deceptive learning game.

Most real-world applications will likely fall between these two extremes, where the supports only partially overlap. In such scenarios, the adversary wins outside the overlap (i.e., can classify everything correctly outside the overlap), and the challenger wins inside the overlap. In other words, extremely sensitive and damaging posts cannot be protected as they will have no overlap with any of the volunteered posts. However, as we show in the next section, with a reasonable volunteered set, the challenger can make it hard for the adversary to detect damaging deletions.

V. System Evaluation on Twitter Deletions

In this section we evaluate the efficiency of an adversary when Deceptive Deletions is applied to the real-world problem of concealing damaging deletions in Twitter. In this evaluation we first create and prepare sets of (non)damaging tweets. Then we use these sets to train the challenger and adversary classifiers and analyze their performance.

A. Data Collection

In this work, we select Twitter as our experimental social media platform. We note that it was certainly plausible to perform the exact experiment on other social platforms. However we chose Twitter due to its popularity and feasibility of data collection. Specifically, in order to evaluate the challenger we needed a real-world dataset which includes (i) both deleted and non-deleted tweets (i.e., Twitter posts) and (ii) deleted tweets that contain both damaging and non-damaging tweets. To that end, we use two data sources to create such a dataset.

1) Deceptive Deletion dataset: We collected 1% of daily random tweet samples from the Twitter API from Oct 2015 - May 2018. Eliminating non-English tweets, we accumulated over one billion tweets. In the next step, we construct the damaging and volunteered sets.

To construct the damaging set, we first needed to identify the deleted tweets[6]. We sampled 300,000 tweets from the aforementioned collected data, and leveraging the Twitter API, we identified the tweets that were deleted at the time of our experiment (Jan 30th, 2020). In total, we identified 92,326

[6]we only considered user-initiated deletion (not platform-initiated ones).

deleted tweets. The next step was to obtain ground truth labels for the deleted tweets—i.e., detect and assign "true" labels to damaging tweets and "false" labels to rest. We used the crowdsourcing service Amazon Mechanical Turk (MTurk) [21] to obtain a proxy for these true labels. However, there were two challenges– First, it was impractical to ask our annotators to label 92,326 tweets. Second, since the dataset was highly imbalanced, a simple random sample of tweets for labeling would have resulted in a majority of non-damaging tweets.

Thus we followed prior work [87], [94] and filtered the deleted tweets using a simple sensitive keyword-based approach [94] (i.e., identify posts with sensitive keywords) to have a higher chance of collecting possibly damaging tweets. The complete list of keywords (over 1500 words) can be found in http://bit.ly/1LQD22F. This approach resulted in 33,000 potentially damaging tweets, and we randomly sampled 3,500 tweets to be labeled by annotators on MTurk. The mean number of sensitive keywords in each tweet within our data set was 2.55. We have also considered the experiment of skipping the filtering step explained above. We refer interested readers to Appendix A for detailed information after reading this section (as only the differences with the filtering approach are highlighted there).

Note that, in addition to the cursing and sexual keywords, our sensitive keyword-based approach considered keywords related to the topics of religion, race, job, relationship, health, violence, etc. Intuitively, if a post does not contain any such sensitive keywords then the likelihood of the post being damaging is very low. We confirmed this intuition by asking MTurk annotators to label 150 tweets which did not contain any sensitive keyword as (non)damaging. More than 97% of these 150 tweets were labeled as non-damaging by annotators. We surmised that in practice, the adversary will also leverage a similar filtering approach to reduce its overhead and increase its chances of finding damaging posts. Note that, in this experiment we have only considered the text of the tweets. However, the adversary can use additional user information, but labeling the posts (for training) based on the entire sets of posts of the users is infeasible for a large-scale attack.

In total, out of our sampled 3,500 deleted tweets, we obtained labels for 3,177 tweets (excluding annotations from Turkers who failed our quality control checks as described later). Among the labeled tweets, 1,272 were identified as damaging, and $1,905$ were identified as non-damaging.

Data labeling using MTurk. We acknowledge that ideally, the tweet labels should have been assigned by the posters themselves. However, since we collected random tweets at large-scale using the Twitter API, we could not track down and pursue original posters to label their deleted tweets. Furthermore, following up with specific users for labeling their deleted posts is likely to cross the ethical boundary of this academic work (see Section V-B). To that end, we note that there is a crowdsourcing based alternative which is already leveraged by earlier work to assign sensitivity labels [27], [31], [87]. Specifically, these studies determined the sensitivity of social media posts by simply aggregating crowdsourced sensitivity labels provided by multiple MTurk workers (Turkers). Thus, we took a similar approach as mentioned next.

On MTurk, tasks (e.g., completing surveys) are called Human Intelligence Tasks or HITs. Turkers can participate in a survey by accepting the corresponding HIT only if they meet all the criteria associated with that HIT (set by the person(s) who created the HIT). We leverage this feature to ensure the reliability of our results. Specifically we asked that the Turkers taking our survey should: (i) have at least 50 approved HITs. (ii) have an assignment approval rate higher than 90%, and (iii) have their location set to United States. This last criterion ensured consistency of our Turkers' linguistic background. In our experiment each HIT consisted of annotating 20 tweets with true (damaging) or false (non-damaging) labels. We allowed the Turkers to skip some tweets in case they feel uncomfortable for any reason. We compensated 0.5 USD for each HIT and on average it took the Turkers 193 seconds to complete each HIT.

To control the quality of annotation by Turkers, we included two hand-crafted control tweets with known labels in each HIT. These control tweets were randomly selected from two very small sets of clearly non-damaging or damaging tweets and were inserted at random locations within the selection of 20 tweets. For example a damaging control tweet was: *"I think I have enough knowledge to make a suicide bomb now! Might need it New Year's Eve"* and non-damaging control tweet was: *"Prayers with all the people in the hurricane irma"*. If for a HIT, the responses to these control tweets did not match the expected label, we conservatively discarded all twenty annotations in that HIT.

We countered possible bias resulting from the order of presentation of tweets via randomizing the order of tweets in every HIT. Even if two Turkers annotated the same set of tweets, the order of those tweets was different. Furthermore, to ease the subjectivity of the labels from each participant, for each tweet we collected the annotations of multiple Turkers and took the majority vote. In our experiment, we created the HITs such that each tweet was annotated by 3 distinct Turkers. After receiving the responses, for each tweet we assigned the final label (indicating damaging or non-damaging) based on the majority vote.

We emphasize that in the real world, the burden of labeling the posts via crowdsourcing is on the adversary(see Section III-B, *Labels* subsection). The challenger, on the other hand, can be implemented as a service within the platform and can obtain the true labels directly from the post-owners. Therefore, existence of any mislabeled data will negatively impact only the adversary (see Section III-C, *Labels* subsection).

2) #Donttweet dataset: Recently Wang et al. [87] proposed "#Donttweetthis". "#Donttweetthis" is a quantitative model that identifies potentially sensitive content and notifies users so that they can rethink before posting those content on social platforms. Wang et al. created the training data for their model by (i) identifying possibly sensitive tweets by checking for the existence of sensitive keywords within the text and then (ii) using crowd-sourcing (i.e., using MTurk) to annotate the sensitivity of each tweet by three annotators.

The data collection approach used by "#Donttweetthis" (section 3 of [87]) is very similar to ours. Therefore, to enrich our dataset and be able to evaluate the challenger over more intervals, we acquired their labeled tweets. Using the Twitter API, we queried the tweets using their corresponding IDs and

identified the deleted ones (at the time of writing, Jan 30th, 2020). In total, we obtained 851 deleted tweets, where 418 were labeled as sensitive (damaging), and the remaining 433 were labeled as non-sensitive (non-damaging). The mean of sensitive keywords in each tweet within this set was 1.7.

Summary of collected data. In summary, combining the two datasets explained above, we obtained labels for $4,028$ deleted tweets establishing the user deleted set. Among the deleted tweets $1,690$ were labeled as damaging constructing our damaging set ($\mathbb{D}^+$). As we will demonstrate in the results section, in our evaluation the four thousand labeled tweets (larger than that of prior works [87], [94]) allows for 10 intervals for the game between the adversary and challenger.

Furthermore, for our experiment, we consider $k = 1, 2, 5$ (i.e., number of decoy posts for each damaging post). To accommodate these values of k and construct a volunteer pool that the challenger can make meaningful selections from, we sampled 100,000 non-deleted tweets uniformly at random from the 1% daily tweet samples posted between Jan 1st, 2018 May 31st, 2018 to build the volunteered set. The non-deleted tweets are assumed to be non-damaging. We consider this assumption to be reasonable as if a tweet contains some damaging content then its owner would not keep that post on its profile. In practice, we can forgo this assumption as the volunteer users themselves offer the volunteer posts. The average number of sensitive keywords in each tweet in this set was 0.41.

B. Ethical Considerations

Recall that in order to create our evaluation dataset we needed to show some deleted tweets to Turkers for the annotation task. Thus, we were significantly concerned about the ethics of our annotation task. Consequently, we discussed at length with the Institutional Review Board (IRB) of the lead author's institute and deployed the annotation task only after we obtained the necessary IRB approval. Next we will detail, how, in our final annotation task protocol we took quite involved precautionary steps for protecting the privacy of the users who deleted their tweets.

We recognize that, in the context of our evaluation, the primary risk to the deleted-tweet-owners was the possibility of linking deleted tweets with deleted-tweet-owner profiles during annotation. This intuition is supported by prior research [56], [65] who suggested applying selective anonymization for research on deleted content. Thus, we anonymized all deleted tweets by replacing personally identifiable information or PII (e.g., usernames, mentions, user ids, and links) with placeholder text. For example, we replaced user accounts (i.e., words starting with @) and url-links with "UserAccount" and "Link" respectively. Moreover, one of the authors manually went over each of these redacted posts to ensure anonymization of PII before showing them to Turkers.

C. Experiment Setup

Partitioning the data for different time intervals. Recall from Section III that we discretize time into intervals. In our experiments, we choose $T = 10$ intervals in total (a choice made based on the number of collected tweets). Consequently, we partition our dataset into 10 intervals. Ideally, the partitions should be based on the creation and deletion timestamps of the tweets. Unfortunately however, the Twitter API does not provide deletion timestamps. Hence, we randomly shuffle the tweets and divide them into 10 equally sized partitions.

BERT model. In line with our approach to model the most-powerful adversary as best as we possibly can, we use a state-of-the-art natural language processing model: the BERT (Bidirectional Encoder Representations from Transformers) language model [36], both for the adversary and for the challenger. Specifically, we use $\text{BERT}_{\text{BASE}}$ model that consists of 12 transformer blocks, a hidden layer size of 768 and 12 self-attention heads (110M parameters in total). BERT has been shown to perform exceedingly well in a number of downstream NLP tasks [36]. We use HuggingFace's [89] implementation of the BERT model that was already *pre-trained* on masked language modeling and next sentence prediction tasks.

BERT uses WordPiece embeddings [90] to convert each word in the input tweet to an embedding vector. The concatenated embedding vector is passed to the BERT neural network model. In our experiments, we only give the text of the tweet as input to both the adversary and the challenger to make it amenable to the pre-trained BERT models. Other tweet features such as deletion timestamps, number of likes, etc. could be used by both the adversary and the challenger to improve their performance. Note however that Propositions 2 and 3 still apply as long as the adversary and the challenger have the same information.

We *fine-tune* the BERT model on our datasets as prescribed by Devlin et al [36]. In each interval, the adversary's classifier is fine-tuned for the classification of tweets into damaging and non-damaging using the negative log-likelihood loss in Equation (1). We use a batch size of 32 and sample equal number of damaging and non-damaging tweets in each batch. This procedure results in better trained models as it avoids the scenario where a randomly sampled batch is too imbalanced (for example, no damaging tweet sampled in the batch). A separate BERT model is fine-tuned for the challenger using the loss function in Equation (4). Note that no balancing is required here since all the input tweets to the challenger model are non-damaging. We note that explaining the exact strategy employed by BERT models to classify text is an active research topic and complementary to our efforts. However, we highlight that our challenger does not use any information about either the adversary's exact model or its parameters.

Budget constraints: We allow a limited budget of $B_{\text{static}} = 200$ deleted tweets for the static adversary and set $\tau = 1$, i.e., the static adversary only trains during the first out of the ten intervals. Similarly for the adaptive adversary, we allow a fixed recurring budget of $B_{\text{adapt}} = 200$ deleted tweets every interval. There are no budget restraints for random and oracle challengers (having no access and black-box access respectively). However, we restrict the D^2 challenger to have the same (recurring) query budget as the adaptive adversary's recurring budget to keep the game fair, i.e., $B_g = B_{\text{adapt}} = 200$.

We simulate the game described in Algorithm 3 with an adversary and a challenger, both implemented as BERT language models, with 10 different random seeds. We repeat the experiments for $k = 1, 2, 5$ where k denotes the number of decoy posts added per damaging deletion.

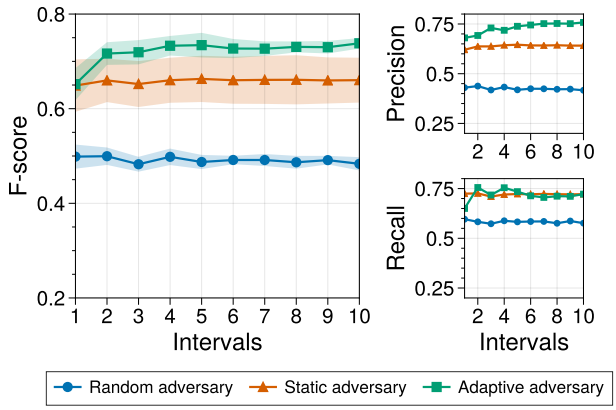


Fig. 3: F-score of different adversaries (random, static, adaptive) when no privacy preserving deletion mechanism is in place. Shaded areas represent 95% confidence intervals.

D. Results

Figures 3 and 4 show the F-scores (with 95% confidence intervals), precision and recall for different adversaries over 10 time intervals. We make the following key observations.

Detection of damaging deletions in social media platforms is a serious concern. We start by considering the case where no privacy-preserving deletion mechanism is in place (i.e., no challenger to inject decoy deletions). In such a scenario, we compare the efficiency of different types of adversaries ten intervals shown in Figure 3.

The random adversary labels the posts based on the prior distribution of the deleted tweets (around 42% damaging and 58% non-damaging every interval). As expected, the adversary achieves a 42% precision and 58% recall resulting in an F-score of about 48% in each interval.

As shown in Figure 3, in the first interval, the static adversary achieves a 17 percentage points (i.e., a 35%) increase in its F-score compared to the random adversary, and remains almost constant over the rest of the intervals. On the other hand, the adaptive adversary receives new training data every interval and trains its classifier continually, and hence is able to increase its F-score even further by about 10 percentage points (56% increase compared to the random adversary) at the end of the 10th interval.

This shows that even normal users of social media platforms, not only celebrities and politicians, are vulnerable to the detection of their damaging deletions. Furthermore, the adversaries can automate this attack on a large-scale with an insignificant amount of overhead (access to a small dataset of posts with the corresponding labels), highlighting the necessity for a much-needed privacy-preserving mechanism for the users' damaging deletions in today's social platforms.

Injecting decoy deletions decreases the adversarial performance. As explained in Sections III and IV, we consider three challengers corresponding to the three types of accesses to the adversary's model – *no access*, *black-box access*, and *restricted black-box access*. In the following, we compare the performance of the adversaries in the presence of the respective challengers against the absence of any challenger case above.

No access: The top row of Figure 4 shows the performance of the three adversaries (random, static, and adaptive) in the presence of the random challenger. We observe that although the F-score of both the static and the adaptive adversary decreases for all values of k, the reduction is not significant (only 7 percentage points for k=1 compared to the no-challenger case) In fact, both the adversaries still perform much better than the random adversary. This shows that protection of damaging deletions in the no-access scenario is possible but severely limited.

Black-box access: The middle row of Figure 4 shows the performance of the adversaries in the presence of an oracle challenger. Not surprisingly, this approach is very effective at lowering the (static and adaptive) adversaries' F-scores (close to random for k=1, 2; i.e., 20 and 35 percentage point reduction in the case of k=1 for the static and adaptive adversary respectively compared to the no-challenger case).

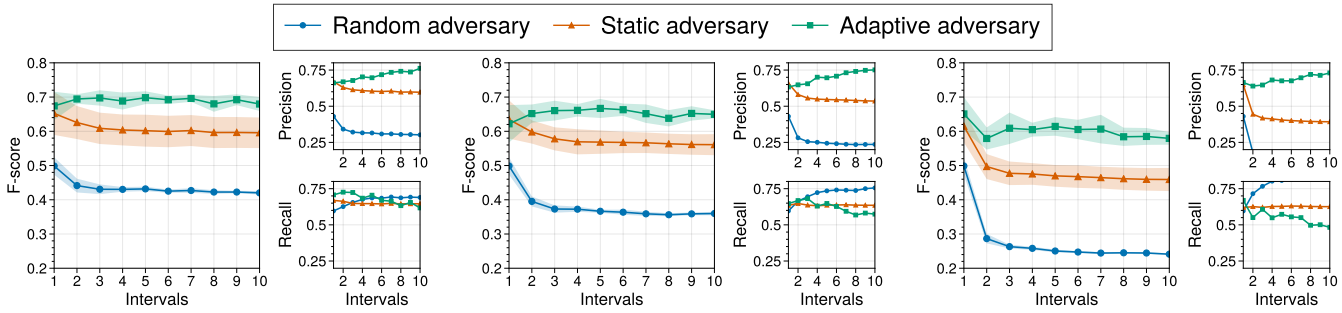
We also observe a major difference between the static and the adaptive adversaries in the presence of a competitive challenger. The static adversary retains the same recall performance (as in the no-challenger case) but loses drastically in precision, i.e., it classifies a large number of decoy posts as damaging. On the other hand, the adaptive adversary tries to *adapt* to the presence of decoy posts and becomes highly conservative – retains the same precision performance (as in the no-challenger case) but suffers heavily in the recall performance, i.e., it classifies a large number of damaging posts as non-damaging.

Restricted black-box access: The bottom row of Figure 4 shows the performance of the adversaries in the presence of the D^2 challenger. The performance of the D^2 challenger is comparable to the oracle challenger. The adversaries' F-scores in the presence of the D^2 challenger is close to 45% for the case of k=1 (20 and 30 percentage point reduction for the static and adaptive adversaries respectively compared to the no-challenger case). We also observe a precision-recall trade-off separating the static and the adaptive adversary (i.e., the static adversary loses in precision, whereas the adaptive adversary loses in recall) similar to the one described in the presence of an oracle challenger .

Overall, we conclude that the D^2 challenger is able to successfully raise the bar for the adversaries in identifying damaging deletions *without* requiring an unmonitored black-box access with infinite query budget.

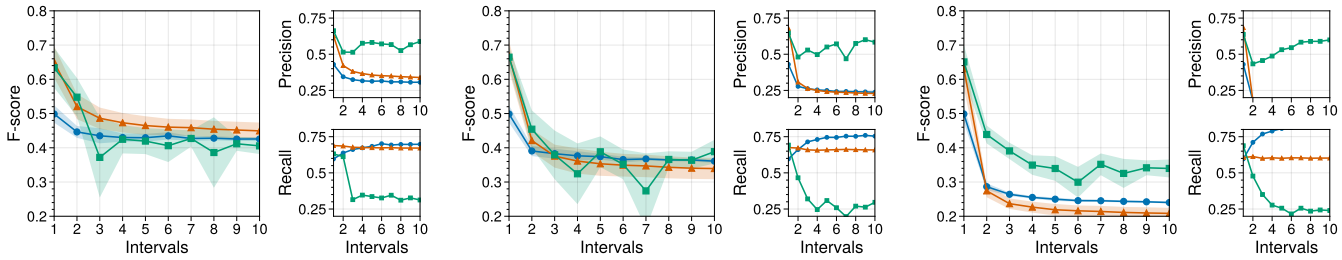
The increase of decoy posts (k) results in lower adversarial performance with diminishing returns. While examining each row of Figure 4 individually, we see that the performance of the adversaries always decreases as k, the number of decoy deletions per damaging deletion, increases. However, we also observe that $k = 1$ is enough to reduce the F-scores of the adversaries to 45% (close to the random adversary). Since the goal of most social platforms is to retain as many posts as possible, it would *not* be in the platform's best interests to use much larger values of k or to delete the entire volunteered set.

Observation of damaging and decoy posts. In Table I in the Appendix, we show damaging tweets (as labeled by the AMT workers) and decoy tweets (chosen by the D^2 challenger from a set of non-deleted tweets). We observe that even though the decoy tweets typically seem to have sensitive words, they do not possess content damaging to the owner.



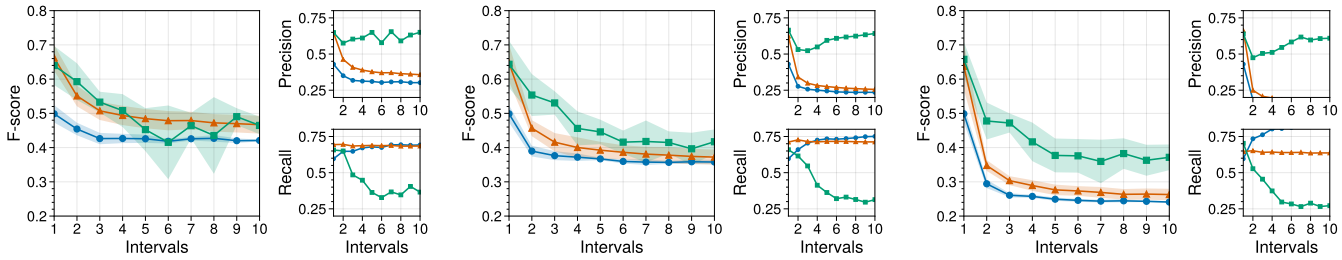
(a) Random challenger ($k = 1$) **(b)** Random challenger ($k = 2$) **(c)** Random challenger ($k = 5$)

(No access.) Adversaries (random, static and adaptive) in the presence of **random challenger** with $k = 1, 2, 5$.



(d) Oracle challenger ($k = 1$) **(e)** Oracle challenger ($k = 2$) **(f)** Oracle challenger ($k = 5$)

(Black-box access.) Adversaries (random, static and adaptive) in the presence of **oracle challenger** with $k = 1, 2, 5$.



(g) D^2 challenger ($k = 1$) **(h)** D^2 challenger ($k = 2$) **(i)** D^2 challenger ($k = 5$)

(Restricted black-box access.) Adversaries (random, static and adaptive) in the presence of **D^2 challenger** with $k = 1, 2, 5$.

Fig. 4: F-score (with 95% confidence intervals), precision and recall for the three adversaries (random, static and adaptive) in the presence of different challengers corresponding to different accesses with $k = 1, 2, 5$. Key observation: D^2 challenger fools the adversaries almost as well as the oracle challenger but with a *restricted* black-box access.

VI. Discussion

A. Adversarial Deception Tactics

The adversary can use different techniques to sabotage the challenger. Here, we mention some prominent systems attacks and their effects on the challenger.

Denial of Service attack. One of such attacks could be a simple Denial of Service (DoS), where the attacker submits requests for many damaging deletions to consume all the volunteer posts. First, we remind that the volunteered posts are a renewable resource, not a finite resource, as the users create, volunteer and delete posts in each time interval. Regardless, a DOS attack is possible wherein the adversary can use up all volunteered posts collected up until this point.

A standard way to avoid such attacks is to limit the number of damaging deletions that can be protected for each user in one time interval (we assume that the adversary can have many *adversarial users* to help with the DoS attack but is not allowed to use bots [26], [30], [37], [40], [82], [86]). As is clear from Section IV-D, the challenger's defense is dependent on the distribution and number of volunteered posts. If there are more *adversarial users* than volunteers, then the adversary can win the game.

We implemented the DoS attack as follows: in every interval, the adversary deletes as much as the standard deletions. We observed that the F-score did not change in this situation.

Volunteer Identification attack. In a volunteer identification attack, the adversary deletes a bunch of posts and uses the process of doing so to identify individuals who volunteer posts to the challenger for deletion. First, we note that in each time interval there is a large number of posts being deleted (> 100 million tweets daily [62]). Thus the posts deleted by the adversary (to try to identify volunteers) and the corresponding decoy deletions are mixed with other (damaging/non-damaging/decoy) deletions. In such a case, identifying the volunteers is equivalent to separating the decoy deletions from the damaging deletions; reducing to the original task. Additionally, the challenger does not delete the decoy posts at the same time as the original damaging deletion but does so in batches spread out within the time interval.

Further, the volunteers can also have damaging deletions of their own. Even if an adversary is able to identify volunteers, the adversary still needs to figure out which of the volunteer's deletions are decoys. If the adversary ignores all posts from volunteers, then a simple protection for the users is to become a volunteer, which helps our cause.

Adversary disguising as volunteer. In this attack, the adversary can take the role of a volunteer (or hire many volunteers) to offer posts to the challenger. Subsequently, the challenger may select the adversary's posts as decoys in the later intervals; however, these posts do not provide deletion privacy as the adversary will be able to discard these decoy posts easily. This effect can be mitigated with the help of more genuine volunteers and increasing the number of decoys per damaging deletion. This points to a more fundamental problem with any crowdsourcing approach: if the number of adversarial volunteers is more than the number of genuine volunteers, the approach fails.

Differentiating between different damaging categories. In this work, all the damaging posts are treated the same. However, in practice, the damaging posts fall into different categories, and some may be more harmful to the users than others. As a result, the adversary can focus on those categories more carefully. In such a case, the challengers outputs and loss function need to be modified—the challenger needs to output a weight per damaging category for each decoy post (indicating the likelihood of fooling the adversary as a damaging post of that category). The challenger would also have to balance the different categories of decoy posts to keep the same distribution of categories as in the real damaging posts.

B. Obtaining volunteered posts from users

Volunteer posts are a significant component of our system. We identify that there are already deletion services which enable users to delete their content in bulk (e.g., "twitWipe" [12] and "tweetDelete" [10] for Twitter, "Social Book Post Manager" [2] for Facebook, "Cleaner for IG" [14] for Instagram, "Nuke Reddit History" [15], and multiple bots on RequestABot subreddit for Reddit). Our system can benefit from these bulk deletions to construct the volunteered posts pool. In such a scenario, whenever a user bulk-deletes it will mark its damaging posts and the remaining posts will be considered as "volunteered" with a guarantee that they will be deleted within a fixed time period.

We contacted the deletion services mentioned above and shared our proposal, Deceptive Deletions, for the privacy of users' damaging deletions. The responses that we received have been positive. They attest that, with Deceptive Deletions, an attacker that observes the deletion of users in large numbers will have a harder time figuring out which of the deleted posts contain sensitive material.

Nevertheless, other strategies could be more effective, for instance, one based on costs and rewards. Under such a strategy, each user seeking privacy for his/her damaging deletions is required to pay a cost for the service, whereas the users that volunteer their non-damaging posts to be deleted by the challenger (at any future point in time) are rewarded[7]. The costs and rewards can be monetary or can be in terms of the number of posts themselves (i.e., a user has to volunteer a certain number of her non-damaging posts to protect her damaging deletion). Nevertheless, in an ideal world, the volunteered set could also be obtained from altruistic users who offer their non-damaging posts for the protection of other users' deletions.

Finally, we emphasize that (as observed in Section V-D) even when there is one decoy post for each damaging post ($k = 1$), the task of the adversary becomes significantly harder. Further, as we state in Appendix A, the percentage of damaging deletions versus the non-damaging ones is significantly lower (i.e., 18% to 82%). Therefore, we can reckon that obtaining the pool of volunteer posts is realizable.

C. Rate Limiting The Adversary's Data Access

In this work, we consider a very powerful adversary in terms of data access—it is capable of taking snapshots of the entire platform at different times to identify deleted posts (see Section III-B). However, in practice, platforms can use rate-limiting techniques to restrict access of the adversary to the users' profile. Client-side strategies [63], [64], deferred responding [17], and the common limitations on source IP address, user, and API key [17], [19] are some of the well-known practices. A more sophisticated approach is to use computational puzzles, where the adversary can only access the data after successfully computing a puzzle given by the data platform. Sample domains include data breach mitigation [57], [85], DDOS [34], [52], spam-prevention [39], and practical cryptocurrencies [59]. These types of data limiting restrictions are interesting future work and will only improve our results. In such a case, the adversary will not be able to observe all the users' profiles constantly, or it will have blackout periods of the users' profiles (not observing the deletions).

D. Deceptive Learning Game vs GANs

Recall that in our setting, the task of the challenger is to select posts from a pre-defined volunteered set $\mathbb{D}^{\mathrm{v}}$. An alternative approach is to use generative models [35], [44], [55], [73], [92] to generate fake texts —see Zhang et al. [93] for a recent survey and Radford et al. [73] for the state-of-the-art— enabling the challenger to *generate* decoy posts instead of selecting them from a pre-defined set. However, we note that such generative models might not be favorable or even effective in practical systems.

[7] Other distributed systems use similar concept such as BitTorrent [53], [79].

Let us consider the case of generating decoy posts on Twitter. Twitter posts are attached with a persistent non-anonymous user identities [31]. Since, uploading fake posts from real user accounts raises serious ethical concerns, one should create multiple bot accounts that will upload machine-generated fake posts to be used as decoy posts (by deleting them later). However, unfortunately, detection of bot accounts is a well studied problem [26], [30], [37], [40], [82], [86]. Moreover, when an adversary detects a bot, any decoy post from that bot account will be similarly unmasked. Therefore, in non-anonymous platforms like Twitter, selecting the decoy posts from the posts of actual users is arguably a more practical approach.

VII. Conclusion and Future Work

In this paper, we show the necessity for deletion privacy by presenting an attack where an adversary is able to increase its performance (F-score) in identifying damaging posts by 56% compared to random guessing. Such an attack enables the system like Fallait Pas Supprimer to perform large-scale automated damaging deletion detection, and leaves users with "damned if I do, damned if I don't" dilemma.

To overcome the attack, we introduce Deceptive Deletions (which we also denote as challenger), a new deletion mechanism that selects a set of non-damaging posts (decoy posts) to be deleted along with the damaging ones to confuse the adversary in identifying the damaging posts. These conflicting goals create a minmax game between the adversary and the challenger where we formally describe the Deceptive Learning Game between the two parties. We further describe conditions for two extreme scenarios: one where the adversary always wins, and another where the challenger always wins. We also show practical effectiveness of challenger over a real task on Twitter, where the bar is significantly raised against a strong adaptive adversary in automatically detecting damaging posts. Specifically, we show that even when we consider only two decoy posts for each damaging deletion the adversarial performance (F-score) drops to 65%, 42% and 38% where the challenger has no-access, restricted black-box access and black-box access respectively. This performance indicates a significant improvement over the performance of the same adversary (75% F-score) when no privacy preserving deletion mechanism is in effect. As a result, we significantly *raise the bar* for the adversary going after damaging deletions over the social platform.

Our work paves a new research path for the privacy-preserving deletions which aim to protect against a practical, resourceful adversary. In addition, our deceptive learning game can be adapted for current/future works in the domain of Private Information Retrieval [38], [47], [66], [69] that have a similar setting for injecting decoy queries to protect the users' privacy. Further, the challenger introduced in this work is considered to be honest and to not misuse the damaging deletions against the users. Considering distributed or federated protocols with multiple challengers as well as private multi-party computation [33], [76]–[78] can be a promising future work to mitigate the complete trust of the challenger.

Acknowledgment

We would like to thank Z. Berkay Celik for insightful suggestions on an early draft. We would also like to thank Bo Lou and his team for providing part of the dataset used in our experiment section. This work was funded in part by the National Science Foundation (NSF) Awards CAREER IIS1943364, CCF-1918483, CNS-1719196, and by the ARO, under the U.S. Army Research Laboratory contract number W911NF-09-2-0053, the Purdue Integrative Data Science Initiative, the Purdue Research Foundation, and the Wabash Heartland Innovation Network. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the sponsors.

References

[1] "Snapchat," https://www.snapchat.com/.

[2] "Social book post manager," https://chrome.google.com/webstore/detail/social-book-post-manager/ljfidlkcmdmmibngdfikhffffdmphjae.

[3] "Twittereraser deletion service," https://www.tweeteraser.com/statistics.

[4] "Dust," https://www.usedust.com/, 2016.

[5] "SNL's first latina cast member is caught out deleting thousands of tweets, some of which were 'racist and offensive'," http://www.dailymail.co.uk/news/article-3805356/SNL-s-Latina-cast-member-caught-deleting-thousands-tweets-racist-offensive.html, 2016.

[6] "24 tweets Ed Sheeran will probably delete soon," https://www.buzzfeed.com/mjs538/we-r-who-we-r-is-a-good-song-tho, 2017.

[7] "The streaming apis," https://dev.twitter.com/streaming/overview, 2017.

[8] "Streaming message types," https://developer.twitter.com/en/docs/tweets/filter-realtime/guides/streaming-message-types, 2017.

[9] "Trackmenot," https://cs.nyu.edu/trackmenot/, 2017.

[10] "Tweetdelete," https://www.tweetdelete.net/, 2017.

[11] "Tweetdeleter: Delete many tweets with one click!" https://www.tweetdeleter.com, 2017.

[12] "Twitwipe," http://twitwipe.com/, 2017.

[13] "Collection of deleted tweets & annoying content," https://twitter.com/fallaitpassuppr?lang=en, 2019.

[14] "Cleaner for instagram," https://play.google.com/store/apps/details?id=ro.novasoft.cleanerig, 2020.

[15] "Nuke reddit history," https://chrome.google.com/webstore/detail/nuke-reddit-history/aclagjkmidmkcdhkhlicmgkgmpgccaod, 2020.

[16] "Pushshift," https://pushshift.io/, 2020.

[17] "Rate-limiting strategies and techniques," https://cloud.google.com/solutions/rate-limiting-strategies-techniques, 2020.

[18] "Removeddit," http://removeddit.com/, 2020.

[19] "What is rate limiting? — rate limiting and bots," https://www.cloudflare.com/learning/bots/what-is-rate-limiting/, 2020.

[20] H. Almuhimedi, S. Wilson, B. Liu, N. Sadeh, and A. Acquisti, "Tweets are forever: A large-scale quantitative analysis of deleted tweets," in *CSCW'13*.

[21] Amazon, "Mechanical Turk," https://www.mturk.com/mturk/welcome.

[22] P. André, M. Bernstein, and K. Luther, "Who gives a tweet?: Evaluating microblog content value," in *CSCW '12*, 2012.

[23] E. Balsa, C. Troncoso, and C. Diaz, "Ob-pws: Obfuscation-based private web search," in *Security and Privacy (SP)*, 2012.

[24] L. Bauer, L. F. Cranor, S. Komanduri, M. L. Mazurek, M. K. Reiter, M. Sleeper, and B. Ur, "The post anachronism: The temporal dimension of facebook privacy," in *ACM WPES '13*.

[25] J. Baumgartner, S. Zannettou, B. Keegan, M. Squire, and J. Blackburn, "The pushshift reddit dataset," *arXiv preprint arXiv:2001.08435*, 2020.

[26] A. Bessi and E. Ferrara, "Social bots distort the 2016 us presidential election online discussion," 2016.

[27] J. A. Biega, K. P. Gummadi, I. Mele, D. Milchevski, C. Tryfonopoulos, and G. Weikum, "R-Susceptibility: An IR-Centric Approach to Assessing Privacy Risks for Users in Online Communities," in *Proceedings of the 39th International ACM SIGIR Conference*, 2016.

[28] C. Castelluccia, E. De Cristofaro, A. Francillon, and M.-A. Kaafar, "Ephpub: Toward robust ephemeral publishing," in *ICNP'11*.

[29] M. D. Choudhury, S. Counts, E. Horvitz, and M. Gamon, "Predicting depression via social media," in *Proceedings of AAAI Conference on Weblogs and Social Media (ICWSM'2013)*, ser. ICWSM, 2013.

[30] Z. Chu, S. Gianvecchio, H. Wang, and S. Jajodia, "Detecting automation of twitter accounts: Are you a human, bot, or cyborg?" *IEEE Transactions on Dependable and Secure Computing*, vol. 9, no. 6, 2012.

[31] D. Correa, L. A. Silva, M. Mondal, F. Benevenuto, and K. P. Gummadi, "The Many Shades of Anonymity: Characterizing Anonymous Social Media Content," in *Proceedings of The 9th International AAAI Conference on Weblogs and Social Media*, Oxford, UK, 2015.

[32] N. Dalvi, P. Domingos, S. Sanghai, D. Verma *et al.*, "Adversarial classification," in *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2004.

[33] J. Darivandpour and M. J. Atallah, "Efficient and secure pattern matching with wildcards using lightweight cryptography," *Computers & Security*, vol. 77, pp. 666–674, 2018.

[34] D. Dean and A. Stubblefield, "Using client puzzles to protect tls." in *USENIX Security Symposium*, vol. 42, 2001.

[35] E. L. Denton, S. Chintala, R. Fergus *et al.*, "Deep generative image models using a laplacian pyramid of adversarial networks," in *Advances in neural information processing systems*, 2015, pp. 1486–1494.

[36] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[37] J. P. Dickerson, V. Kagan, and V. Subrahmanian, "Using sentiment to detect bots on twitter: Are humans more opinionated than bots?" in *Proceedings of the 2014 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*. IEEE Press, 2014.

[38] J. Domingo-Ferrer, A. Solanas, and J. Castellà-Roca, "h (k)-private information retrieval from privacy-uncooperative queryable databases," *Online Information Review*, vol. 33, no. 4, pp. 720–744, 2009.

[39] C. Dwork and M. Naor, "Pricing via processing or combatting junk mail," in *Annual International Cryptology Conference*. Springer, 1992.

[40] E. Ferrara, O. Varol, C. Davis, F. Menczer, and A. Flammini, "The rise of social bots," *Communications of the ACM*, vol. 59, no. 7, 2016.

[41] S. Garg, S. Goldwasser, and P. N. Vasudevan, "Formalizing data deletion in the context of the right to be forgotten," in *EUROCRYPT*, 2020.

[42] R. Geambasu, T. Kohno, A. Krishnamurthy, A. Levy, H. M. Levy, P. Gardner, and V. Moscaritolo, "New directions for self-destructing data," University of Washington, Tech. Rep. UW-CSE-11-08-01, 2011.

[43] R. Geambasu, T. Kohno, A. A. Levy, and H. M. Levy, "Vanish: Increasing data privacy with self-destructing data," in *USENIX*, 2009.

[44] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014.

[45] M. Gutmann and A. Hyvärinen, "Noise-contrastive estimation: A new estimation principle for unnormalized statistical models," in *AISTATS*, 2010, pp. 297–304.

[46] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural networks*, 1989.

[47] D. C. Howe and H. Nissenbaum, "Trackmenot: Resisting surveillance in web search," *Lessons from the Identity trail: Anonymity, privacy, and identity in a networked society*, vol. 23, pp. 417–436, 2009.

[48] B. A. Huberman, D. M. Romero, and F. Wu, "Social networks that matter: Twitter under the microscope," *First Monday*, vol. 14, no. 1, 2009, http://firstmonday.org/article/view/2317/2063.

[49] A. Java, X. Song, T. Finin, and B. Tseng, "Why we twitter: Understanding microblogging usage and communities," in *WebKDD/SNA-KDD*, 2007, http://dl.acm.org/citation.cfm?id=1348556.

[50] J. Jia and N. Z. Gong, "Attriguard: A practical defense against attribute inference attacks via adversarial machine learning," in *USENIX*, 2018.

[51] C. Jin, P. Netrapalli, and M. I. Jordan, "Minmax optimization: Stable limit points of gradient descent ascent are locally optimal," *arXiv preprint arXiv:1902.00618*, 2019.

[52] A. Juels, "Client puzzles: A cryptographic countermeasure against connection depletion attacks," in *Proceedings of NDSS*, 1999.

[53] I. A. Kash, J. K. Lai, H. Zhang, and A. Zohar, "Economics of bittorrent communities," in *Proceedings of the 21st international conference on World Wide Web*, 2012, pp. 221–230.

[54] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," *arXiv preprint arXiv:1607.02533*, 2016.

[55] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. P. Aitken, A. Tejani, J. Totz, Z. Wang *et al.*, "Photo-realistic single image super-resolution using a generative adversarial network." in *CVPR*, vol. 2, no. 3, 2017, p. 4.

[56] J. Maddock, K. Starbird, and R. M. Mason, "Using historical twitter data for research: Ethical challenges of tweet deletions," in *CSCW 2015 Workshop on Ethics for Studying Sociotechnical Systems in a Big Data World. ACM*, 2015.

[57] E. V. Mangipudi, K. Rao, J. Clark, and A. Kate, "Towards automatically penalizing multimedia breaches," in *IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, 2019.

[58] S. Mei and X. Zhu, "Using machine teaching to identify optimal training-set attacks on machine learners." in *AAAI*, 2015.

[59] A. Miller, A. Kosba, J. Katz, and E. Shi, "Nonoutsourceable scratch-off puzzles to discourage bitcoin mining coalitions," in *Proceedings of the 22nd Conference on Computer and Communications Security*, 2015.

[60] M. Minaei, M. Mondal, and A. Kate, ""my friend wanted to talk about it and i didn't": Understanding perceptions of deletion privacy in social platforms," 2020.

[61] M. Minaei, M. Mondal, P. Loiseau, K. Gummadi, and A. Kate, "Forgetting the forgotten with letheia, concealing content deletion from persistent observers," *arXiv preprint arXiv:1710.11271*, 2017.

[62] ——, "Lethe: Conceal content deletion from persistent observers," vol. 2019, no. 1. Sciendo, 2019, pp. 206–226.

[63] J. Mirkovic, G. Prier, and P. Reiher, "Attacking ddos at the source," in *10th IEEE International Conference on Network Protocols, 2002. Proceedings.* IEEE, 2002, pp. 312–321.

[64] ——, "Source-end ddos defense," in *Second IEEE International Symposium on Network Computing and Application*. IEEE, 2003.

[65] M. Mondal, J. Messias, S. Ghosh, K. P. Gummadi, and A. Kate, "Forgetting in social media: Understanding and controlling longitudinal exposure of socially shared data," in *USENIX SOUPS '16*.

[66] M. Murugesan and C. Clifton, "Providing privacy through plausibly deniable search," in *International Conference on Data Mining*, 2009.

[67] P. Papadopoulos, A. Papadogiannakis, M. Polychronakis, A. Zarras, T. Holz, and E. P. Markatos, "K-subscription: Privacy-preserving microblogging browsing through obfuscation," in *Proceedings of the 29th Annual Computer Security Applications Conference*. ACM, 2013.

[68] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against machine learning," in *Proceedings of Asia CCS*, 2017.

[69] S. T. Peddinti and N. Saxena, "On the privacy of web search based on query obfuscation: a case study of trackmenot," in *International Symposium on Privacy Enhancing Technologies Symposium*, 2010.

[70] ——, "On the effectiveness of anonymizing networks for web search privacy," in *Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security*. ACM, 2011, pp. 483–489.

[71] T. Pedersen, "Screening twitter users for depression and ptsd with lexical decision lists," in *Proceedings of the 2nd Workshop on Computational Linguistics and Clinical Psychology: From Linguistic Signal to Clinical Reality*, 2015, http://www.aclweb.org/anthology/W15-1206.

[72] A. Petit, T. Cerqueus, S. B. Mokhtar, L. Brunie, and H. Kosch, "Peas: Private, efficient and accurate web search," in *Trustcom/BigDataSE/ISPA, 2015 IEEE*, vol. 1. IEEE, 2015, pp. 571–580.

[73] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," *preprint*, 2019.

[74] V. Raynauld, "The perfect political storm? the tea party movement, the redefinition of the digital political mediascape, and the birth of online politicking 3.0," Ph.D. dissertation, Carleton University, 2013.

[75] V. Raynauld and J. Greenberg, "Tweet, click, vote: Twitter and the 2010 ottawa municipal election," *Journal of Information Technology & Politics*, vol. 11, no. 4, pp. 412–434, 2014.

[76] M. S. Riazi, "Large-scale privacy-preserving matching and search," Ph.D. dissertation, 2016.

[77] M. S. Riazi, E. M. Songhori, A.-R. Sadeghi, T. Schneider, and F. Koushanfar, "Toward practical secure stable matching," *Proceedings on Privacy Enhancing Technologies*, vol. 2017, no. 1, pp. 62–78, 2017.

[78] M. S. Riazi, "Towards a private new world: Algorithm, protocol, and hardware co-design for large-scale secure computation," Ph.D. dissertation, UC San Diego, 2020.

[79] M. Ripeanu, M. Mowbray, N. Andrade, and A. Lima, "Gifting technologies: A bittorrent case study," *First Monday*, 2006.

[80] M. Sleeper, J. Cranshaw, P. G. Kelley, B. Ur, A. Acquisti, L. F. Cranor, and N. Sadeh, ""i read my twitter the next morning and was astonished": A conversational perspective on twitter regrets," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2013.

[81] J. Steinhardt, P. W. W. Koh, and P. S. Liang, "Certified defenses for data poisoning attacks," in *Advances in neural information processing systems*, 2017, pp. 3517–3529.

[82] G. Stringhini, C. Kruegel, and G. Vigna, "Detecting spammers on social networks," in *Proceedings of the 26th annual computer security applications conference*. ACM, 2010, pp. 1–9.

[83] R. Tinati, A. Madaan, and W. Hall, "Instacan: Examining deleted content on instagram," in *Proceedings of Web Science Conference*, 2017.

[84] S. Tsugawa, Y. Kikuchi, F. Kishino, K. Nakajima, Y. Itoh, and H. Ohsaki, "Recognizing depression from twitter activity," in *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, ser. CHI, 2015.

[85] L. Vargas, G. Hazarika, R. Culpepper, K. R. Butler, T. Shrimpton, D. Szajda, and P. Traynor, "Mitigating risk while complying with data retention laws," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, 2018, pp. 2011–2027.

[86] A. H. Wang, "Detecting spam bots in online social networking sites: a machine learning approach," in *IFIP Annual Conference on Data and Applications Security and Privacy*. Springer, 2010, pp. 335–342.

[87] Q. Wang, H. Xue, F. Li, D. Lee, and B. Luo, "#donttweetthis: Scoring private information in social networks," *Proceedings on Privacy Enhancing Technologies*, vol. 2019, no. 4, pp. 72–92, 2019.

[88] R. H. Weber, "The right to be forgotten more than a pandora's box?" *jipitec*, vol. 2, no. 2, 2011.

[89] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, and J. Brew, "Huggingface's transformers: State-of-the-art natural language processing," *ArXiv*, vol. abs/1910.03771, 2019.

[90] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey *et al.*, "Google's neural machine translation system: Bridging the gap between human and machine translation," *arXiv preprint arXiv:1609.08144*, 2016.

[91] M. Xue, G. Magno, E. Cunha, V. Almeida, and K. W. Ross, "The right to be forgotten in the media: A data-driven study," *PoPETs*, 2016.

[92] G. Ye, Z. Tang, D. Fang, Z. Zhu, Y. Feng, P. Xu, X. Chen, and Z. Wang, "Yet another text captcha solver: A generative adversarial network based approach," in *CCS*. ACM, 2018, pp. 332–348.

[93] W. E. Zhang, Q. Z. Sheng, and A. A. F. Alhazmi, "Generating textual adversarial examples for deep learning models: A survey," *arXiv preprint arXiv:1901.06796*, 2019.

[94] L. Zhou, W. Wang, and K. Chen, "Tweet properly: Analyzing deleted tweets to understand and identify regrettable ones," in *Proceedings of the 25th Conference on World Wide Web (WWW'16)*, April 2016.

APPENDIX

A. System Evaluation Without Keyword Filtering

In Section V-A, we saw that the deleted tweets were filtered using a simple sensitive keyword-based approach [94] (i.e., identify posts with sensitive keywords) to have a higher chance of collecting possibly damaging tweets. Although this approach seems to be a rational choice for the adversary (i.e., it narrows its search for the damaging posts), here we investigate the case of not filtering the posts based on their keywords.

In the first step, we study the ratio of the tweets that contain sensitive keywords to those that do not. We sampled 300,000 random deleted tweets (from the 1% sample tweets of the Twitter API) and observe that 38% of the deleted tweets contained at least one of the sensitive keywords (from [94]), and the remaining 62% did not contain any.

Previously, in Section V-A, we observed the steps of obtaining labels for 3,878 (= 4,028 total labeled tweets - 150 with no sensitive keywords) tweets that contained a sensitive keyword. To follow the 38%-62% ratio explained above, we leveraged the Twitter API and obtained 6,327 deleted tweets that did not contain any sensitive keywords. Next, we labeled all the newly sampled deleted tweets to be non-damaging (instead of labeling the deleted tweets with MTurk). Our rationale is as follow: labeling these tweets in the imbalanced dataset is not reasonable (as well as costly) for our attacker—we ran a small-scale experiment with 150 deleted tweets that did not contain any sensitive keywords and found that less than 5% of them were labeled damaging by the MTurkers (compared to the 43% after filtering these posts). Therefore, for this experiment, we consider all the tweets that do not contain any sensitive keywords to have a non-damaging label.

In summary for this experiment, we had 10,205 deleted tweets which 3,878 of them contained some sensitive keywords and the remaining 6,327 did not. Further, among the 10,205 deleted tweets 1,690 (17%) of them were labeled as damaging and the remaining as non-damaging.

Following the same experimental setup as in Section V-C, we present the results in Figure 5 and Figure 6. We observe that the results follow the same trend as the ones in Section V-D (i.e., the case of filtering tweets based on sensitive keywords). The only difference here is that the performance of the adversary (F-score) has slightly dropped in almost all cases (with and without the challenger). These results show that filtering the posts based on their keywords is an advantages strategy that the adversary will follow to increase its performance.

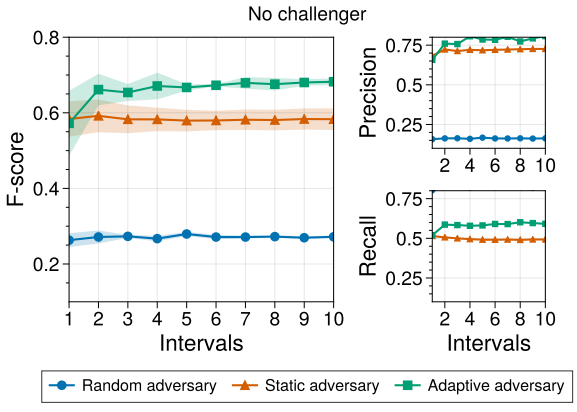
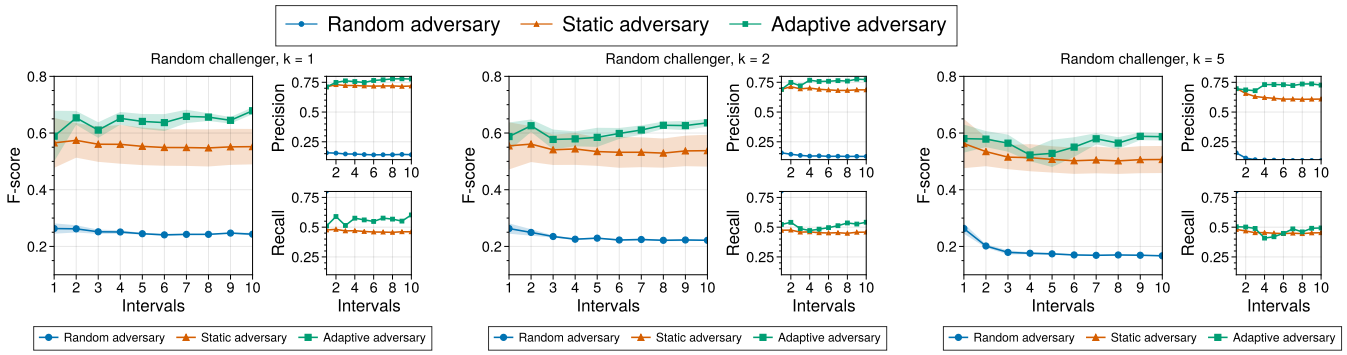
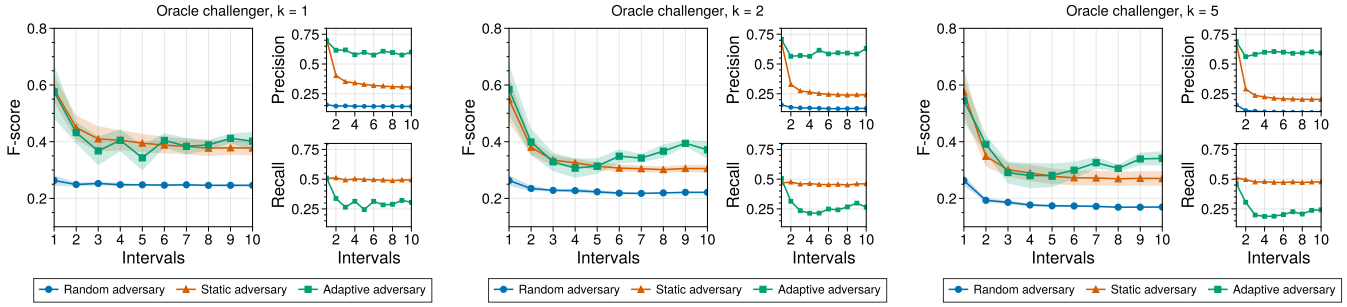


Fig. 5: F-score of different adversaries (random, static, adaptive) when no privacy preserving deletion mechanism is in place. No filtering based on the keywords of tweet were made. Shaded areas represent 95% confidence intervals.



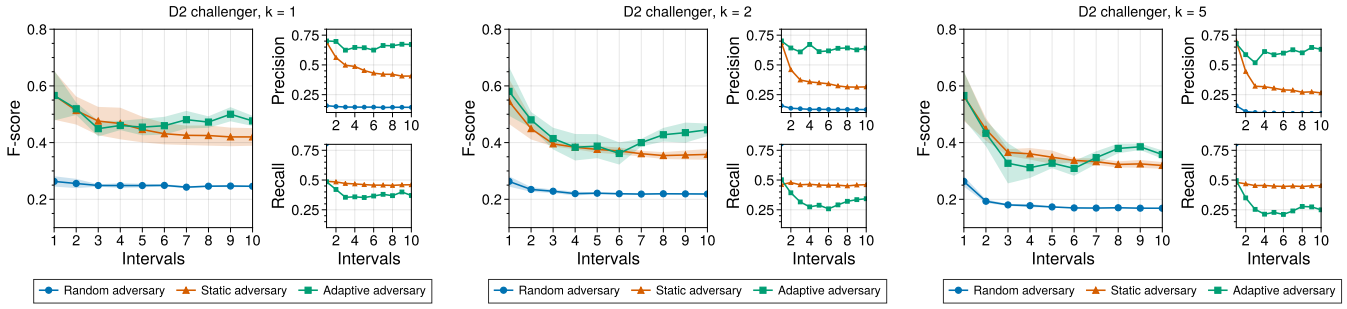
(a) Random challenger ($k = 1$) **(b)** Random challenger ($k = 2$) **(c)** Random challenger ($k = 5$)

(No access.) Adversaries (random, static and adaptive) in the presence of **random challenger** with $k = 1, 2, 5$.



(d) Oracle challenger ($k = 1$) **(e)** Oracle challenger ($k = 2$) **(f)** Oracle challenger ($k = 5$)

(Black-box access.) Adversaries (random, static and adaptive) in the presence of **oracle challenger** with $k = 1, 2, 5$.



(g) D^2 challenger ($k = 1$) **(h)** D^2 challenger ($k = 2$) **(i)** D^2 challenger ($k = 5$)

(Restricted black-box access.) Adversaries (random, static and adaptive) in the presence of **D^2 challenger** with $k = 1, 2, 5$.

Fig. 6: F-score (with 95% confidence intervals), precision and recall for the three adversaries (random, static and adaptive) in the presence of different challengers corresponding to different accesses with $k = 1, 2, 5$. No keyword filtering was applied on this dataset.

B. Proofs

Proposition (Proposition 1.)**.** *For any given volunteered set $\mathbb{D}^v$ with N non-deleted posts,*

$$\max_{\phi} \tilde{V}(\phi; \mathbb{D}^v) = \max_{w_1,\ldots,w_N} V(w_1, \ldots, w_N; \mathbb{D}^v)$$

Proof of Proposition 1: Let $S_1^* = \max_{\phi} \tilde{V}(\phi; \mathbb{D}^v)$ and $S_2^* = \max_{w_1,\ldots,w_N} V(w_1, \ldots, w_N; \mathbb{D}^v)$ be the optimum values for the respective objective functions. First, note that $S_1^* \geq S_2^*$ because the optimal assignment for the discrete objective lies within the solution space of the continuous relaxation. Next, let $L_i = \log(1-a(x_i; \theta_t))$, where x_i is the i-th post in $\mathbb{D}^v$ and let π denote a sorting over them such that $L_{\pi_1} \geq \ldots \geq L_{\pi_N}$. Then, two cases arise – (1) when the top K elements are strictly greater than the rest, $L_{\pi_1} \geq \ldots \geq L_{\pi_K} > L_{\pi_{K+1}} \geq \ldots L_{\pi(N)}$, and (2) when there is atleast one element in the bottom $N-K$ elements that has the same value as one of the top K elements, $L_{\pi_1} \geq \ldots \geq L_{\pi_K} = L_{\pi_{K+1}} \geq \ldots L_{\pi(N)}$. In the former case, the optimal solution is clearly to assign a weight of one to the top K elements and zero to the rest. Any other assignment (even in the continuous solution space) is clearly suboptimal. In the latter case, although there are infinitely many optimal solutions in the continuous domain that distribute the weights differently among the equal elements, the value of the objective function is the same. ■

Proposition (Proposition 2)**.** *Assume $\Omega^v \cap \Omega^+ = \emptyset$, i.e., the supports of volunteered and damaging posts do not overlap. Then, there is always a powerful-enough adversary to defeat the challenger.*

Proof sketch of Proposition 2: Assume the most powerful challenger who can select any post features x from an infinite supply of volunteered posts. However, since $\Omega^{\text{v}} \cap \Omega^+ = \emptyset$, there is no sampling from p^{v} to generate decoy examples that look like they are sampled from p^+. Hence, given enough data, an adversary can find a perfect decision boundary between the damaging posts and the decoy posts. Because neural networks are universal function approximators [46], this powerful adversary always exists and, thus, the challenger can always be defeated in the deceptive learning game. ■

Proposition (Proposition 3)**.** *Assume $\Omega^v = \Omega^+$, i.e., the supports of volunteered and damaging posts fully overlap. Then, given enough volunteered posts in $\mathbb{D}^v$, the challenger always defeats the adversary (in both static and adaptive scenarios). More precisely, if the challenger selects k decoys per damaging post in $\mathbb{D}^\delta$, then the adversary's probability of identifying a damaging post in $\mathbb{D}^\delta$ is in average at most $\frac{1}{k+1}$.*

Proof of Proposition 3: The proof relies on a property of rejection sampling, which states that if the support of two distributions p_1 and p_2 fully overlap, then one can selectively filter samples from p_1 to make the filtered samples have distribution p_2 (a proof of this principle is given in the Appendix). Asymptotically, for each damaging example x in adversary's test data, there are k indistinguishable decoy examples (from the adversary's perspective). This is because, by Bayes theorem

$$p^\delta(y=1|x) = \frac{p^\delta(x|y=1)p^\delta(y=1)}{p^\delta(x|y=1)p^\delta(y=1) + p^\delta(x|y=0)p^\delta(y=0)} \leq \frac{1}{1+k},$$

where the superscript p^δ indicates the distribution of deleted posts $\mathbb{D}^\delta$. The inequality holds by construction, as for all $x \in \mathbb{D}^\delta$ with label one, there are at least $k \geq 1$ samples from $p^{\text{v}}(x)$ with label zero.

Next we show the kind of test distribution shift introduced by the challenger. The challenger-injected distribution is given by the following hypothetical acceptance-rejection sampling algorithm:

1) sample $x \sim p^{\text{v}}(x)$
2) sample $u \sim Uniform(0,1)$ independently of x
3) while $u > p^+(x)/(Mp^{\text{v}}(x))$, reject x and GOTO 1, for some constant M.
4) Accept (output) x as a sample from $p^+(x)$ but with label $y = 0$, as the sample came from $p^{\text{v}}(x)$.
5) While number of samples less than $k|\mathbb{D}^+|$, GOTO 1

Next we prove that the above rejection sampling algorithm produces samples with distribution $p^+(x)$ from examples from decoy examples that have distribution $p^{\text{v}}(x)$. Let X' be a sample from the algorithm described above and $X \sim p^+(x)$, then

$$p(X' = x) = p(X = x|\text{Accept}) = \frac{p(X = x, \text{Accept})}{p(\text{Accept})} = p^+(x)$$

because

$$\begin{aligned}\frac{P(X = x, \text{Accept})}{P(\text{Accept})} &= \frac{P(\text{Accept}|X = x)p(X = x)}{P(\text{Accept})} \\ &= \frac{\frac{p^+(x)}{Mp^{\text{v}}(x)}p^{\text{v}}(x)}{P(\text{Accept})} \\ &= \frac{\frac{p^+(x)}{M}}{P(\text{Accept})} = p^+(x)\end{aligned}$$

as

$$\begin{aligned}P(\text{Accept}) &= \int P(\text{Accept}|X = x)p(X = x)dx \\ &= \int \frac{p(x)}{Mq(x)}q(x)dx \\ &= \frac{1}{M}\int p(x)dx = \frac{1}{M}\end{aligned}$$

The above ideal accept-reject sampling procedure can be reproduced via noise contrastive estimation [45], which is method that can generate data from a known distribution without the need to know $p^+(x)/(Mp^{\text{v}}(x))$ in advance. A variant of the same statistical principle is used today in generative models using Generative Adversarial Networks [44], which uses a minimax game similar to our procedure. Because we train the challenger to mimic the classifier of the adversary, it is easy to construct such rejection sampling method, such that there are in average k decoy examples for every damaging example in the original data. ■

TABLE I: Sample tweet text extracts from the damaging, decoy, and non-damaging datasets.

Tweets' text extract	Tweet Type
"#GrowingUpInTexas Seeing a black person pass by ya front yard and telling your son to pass you the shotgun so you can play shoot em ups"	damaging
"@UserAccount its gods way of punishing you for your sins. fag**t."	damaging
"I don t wanna believe all the women in the auto department at walmart are lesbians Someone prove me wrong Cuz im seeing it"	damaging
"Show up to work on meth once and your nickname is Tweaker for the rest of your life "	damaging
"Listening to this deuchbag behind me at Chipotle diss every girl who comes in hot body but she has no face news check you re fugly"	decoy
"I grab a beer from the fridge put on my Bob Marley record crank that f**ker up and light up a fat one my professor is the sh*t"	decoy
"Kids having kids That sh*t is f**kin crazy to me I d rather be that cool ass uncle that buys the booze aaayyye"	decoy
"I don't understand why people say that watermelon and fried chicken is for black people I love that shit to Dafuq"	decoy
"y'all I just watched "love, simon" for the first time and let me just say that the ugly tears are so f**kin real omfg"	non-damaging
"I want to eat to rid my emotions but I don't want the calories ya feel me"	non-damaging
"Im pretty sure one of my professors has me mistaken for another black woman in my class."	non-damaging
"Anyways, it's 2 am and the Full House theme song is playing in my head on repeat so if you wanna beat me to death do it now please"	non-damaging