Pool Inference Attacks on Local Differential Privacy: Quantifying the Privacy Guarantees of Apple's Count Mean Sketch in Practice

Andrea Gadotti, *Imperial College London;* Florimond Houssiau, *Alan Turing Institute;* Meenatchi Sundaram Muthu Selva Annamalai and Yves-Alexandre de Montjoye, *Imperial College London*

https://www.usenix.org/conference/usenixsecurity22/presentation/gadotti

This paper is included in the Proceedings of the 31st USENIX Security Symposium.

August 10–12, 2022 • Boston, MA, USA

978-1-939133-31-1

Open access to the Proceedings of the 31st USENIX Security Symposium is sponsored by USENIX.

Pool Inference Attacks on Local Differential Privacy: Quantifying the Privacy Guarantees of Apple's Count Mean Sketch in Practice

Andrea Gadotti
Imperial College London

Florimond Houssiau
Alan Turing Institute

Meenatchi Sundaram Muthu Selva Annamalai
Imperial College London

Yves-Alexandre de Montjoye*
Imperial College London

Abstract

Behavioral data generated by users' devices, ranging from emoji use to pages visited, are collected at scale to improve apps and services. These data, however, contain fine-grained records and can reveal sensitive information about individual users. Local differential privacy has been used by companies as a solution to collect data from users while preserving privacy. We here first introduce pool inference attacks, where an adversary has access to a user's obfuscated data, defines pools of objects, and exploits the user's polarized behavior in multiple data collections to infer the user's preferred pool. Second, we instantiate this attack against Count Mean Sketch, a local differential privacy mechanism proposed by Apple and deployed in iOS and Mac OS devices, using a Bayesian model. Using Apple's parameters for the privacy loss ε, we then consider two specific attacks: one in the emojis setting — where an adversary aims at inferring a user's preferred skin tone for emojis — and one against visited websites — where an adversary wants to learn the political orientation of a user from the news websites they visit. In both cases, we show the attack to be much more effective than a random guess when the adversary collects enough data. We find that users with high polarization and relevant interest are significantly more vulnerable, and we show that our attack is well-calibrated, allowing the adversary to target such vulnerable users. We finally validate our results for the emojis setting using user data from Twitter. Taken together, our results show that pool inference attacks are a concern for data protected by local differential privacy mechanisms with a large ε, emphasizing the need for additional technical safeguards and the need for more research on how to apply local differential privacy for multiple collections.

1 Introduction

User's behavioral data, ranging from words typed to processes running on the phone, are collected by operating systems, apps, and services. This data allows companies to better understand user behavior, detect issues, and ultimately improve services. For instance, iOS and Mac OS devices keep track of websites that the user visits using the Safari browser, together with the user's preferences on videos that play automatically when the page is loaded [4]. Aggregated over millions of users, this data allows Apple to learn on which websites the users generally want videos to play automatically and to set default auto-play policies in Safari [4].

Local differential privacy, a variation of differential privacy, is among the main solutions for such data collection. Mechanisms satisfying local differential privacy avoid users having to trust anyone, including the data curator. The mechanism takes as input the original data recorded on the user device (original objects) and shares with the curator a randomized version (obfuscated objects) which should not reveal (almost) anything about the original information [13, 21, 39]. A large literature exists on mechanisms satisfying local differential

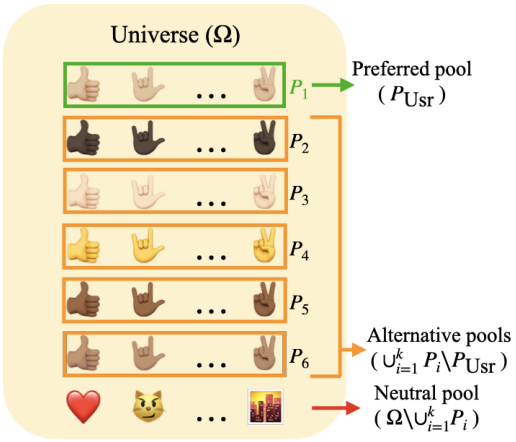

Figure 1: Example of pools defined on a universe Ω consisting of emojis, when the adversary is interested in determining the skin tone that is most often selected by the user. In this case, Usr's preferred pool is the one containing medium-light skin tone emojis.

*Email: deMontjoye@imperial.ac.uk; Corresponding author.

privacy [39] and some mechanisms have been deployed at scale by Google [15], Microsoft [11], and Apple [4].

One of these mechanisms is Count Mean Sketch (CMS). CMS is used on iOS and Mac OS devices to report both emojis used and websites visited to Apple. Featured in Apple's keynote, local differential privacy allows the company to "help discover the usage patterns of a large number of users without compromising individual privacy" [17]. This implementation, and in particular Apple's choice of the parameter ε, has come under criticism from privacy researchers. It is generally believed that ε — which controls the *privacy loss* incurred by the user — should typically not exceed $\ln(3)$ (≈ 1.10) [14]. Soon after the technology was deployed, it was found that Apple's implementation uses $\varepsilon = 4$ when collecting emoji usage data and $\varepsilon = 8$ when collecting web domain data [33].

Apple's choice to only consider the privacy loss per submission — once a day for both the web domain and emoji data — instead of a total privacy loss ε_{tot} (after which objects would no longer be collected from the user [2]) has similarly raised concerns on theoretical ground. While Apple states that they remove user identifiers and IP addresses after the obfuscated objects are received by their server [4], this is a measure that relies on trust and hence conflicts with local differential privacy's purpose of protecting against an untrusted curator[1]. It is indeed well-known that the mathematical guarantees offered by local differential privacy degrade as multiple objects are collected from the same user, something that can be quantified with an upper bound using the Composition Theorem [13] ($\varepsilon_{\text{tot}} \leq \varepsilon_1 + \ldots + \varepsilon_n$). Regardless of how revealing the user's original data may be, a low ε_{tot} would guarantee that the obfuscated data will never leak much information. However, ε_{tot} is a worst-case theoretical measure: it is unclear the extent to which collecting multiple objects and using a large ε for each object open the door to attacks in practice.

Pool inference attack. In this paper we propose the first — to the best of our knowledge — quantification of the practical privacy guarantees provided by a deployed local differential privacy mechanism. We design a novel attack against CMS — which we call *pool inference attack* — that works as follows: first, the adversary receives a sequence of obfuscated objects from a user; second, the adversary defines pools of interest for the attack (i.e. disjoint groups of objects); third, the adversary runs the attack to determine the user's preferred pool — i.e. the pool whose objects are most likely to be selected by the user — along with a confidence score for the inference. In our first use case, the adversary defines the pools to be groups of emojis divided by skin tone (see Figure 1), the goal of the attack being then to infer which is the emoji skin tone used most frequently by the user.

[1] If one assumes that Apple removes any identifier — so that objects from the same user are not linked together and cannot be linked back to individual users —, then local differential privacy would be mostly unnecessary in the first place. Collecting the original non-obfuscated objects and removing any identifier would already preserve privacy in most settings.

Contributions. We make the following contributions: *(i)* We propose pool inference attacks, a new class of attacks aiming at quantifying the sensitive information leaked by local differential privacy mechanisms in practice. We formalize the attack model as a game which can be applied to any mechanism that obfuscates objects independently. *(ii)* We propose a general Bayesian model for pool inference attacks that can be adapted to most local differential privacy mechanisms. The attack uses a hierarchical probability model that simultaneously encodes properties of the user's behavior, the obfuscation of the mechanism, and auxiliary information that may be available to the adversary. *(iii)* We instantiate the attack against synthetic users in two practical settings where the adversary's goal is to infer user preferences (1) for emoji skin tone or (2) political news website. We study the impact that properties of user behavior — such as polarization — have on the attack's effectiveness, and show that our attack can estimate the probability that its output is correct. We also show that, in some cases, CMS provides little protection compared to a scenario where the user simply submits the true object without any local differential privacy. *(iv)* We simulate the attack in the emojis setting using data from Twitter, and find it to be very effective on users who frequently select emojis supporting skin tones. *(v)* We discuss potential solutions and mitigation strategies that may prevent our attack or make it less effective.

2 Background

We now define local differential privacy and the CMS algorithm, introducing the notation that will be used in the paper.

Local differential privacy [22]. A local differential privacy mechanism is a randomized algorithm that takes as input an *original object* from a set Ω and returns an *obfuscated object* from a set $\mathcal{Y}$. For example, Ω could be the set of all emojis and $\mathcal{Y}$ could be the set of binary vectors of a fixed length. We call Ω the *universe of (original) objects* and $\mathcal{Y}$ the *space of obfuscated objects*. Intuitively, the algorithm enforces local differential privacy if the probability that an input produces a certain output is roughly equal for all inputs. Formally:

Let $\mathcal{A}\colon \Omega \to \mathcal{Y}$ be a randomized mechanism. $\mathcal{A}$ satisfies ε-local differential privacy if $e^{-\varepsilon}\Pr[\mathcal{A}(x') = y] \leq \Pr[\mathcal{A}(x) = y] \leq e^{\varepsilon}\Pr[\mathcal{A}(x') = y]$ for any inputs $x, x' \in \Omega$ and output $y \in \mathcal{Y}$.

We abbreviate the obfuscated object $\mathcal{A}(x)$ with $\widetilde{x}$.

Count Mean Sketch [4]. CMS takes as input objects in the universe Ω that the user has selected (e.g. emojis inserted while typing a message) and returns a binary vector of length m (together with an index), where m is typically much smaller than $|\Omega|$. It uses a family $\mathcal{H} = \{h_1, \ldots, h_{|\mathcal{H}|}\}$ of hash functions that map each object in Ω to an integer in $\{1, \ldots, m\}$. Given an original object $x \in \Omega$, CMS samples uniformly at random a hash function $h_j \in \mathcal{H}$ and produces the one-hot vector $v_x^{h_j}$ of size m which is 1 at position $h_j(x)$ and 0 in all

other entries. The vector $v_x^{h_j}$ can be seen as a compressed version of x. Each bit of $v_x^{h_j}$ is then randomly flipped with probability $1/(1+e^{\varepsilon/2})$ or left unchanged with the remaining probability $e^{\varepsilon/2}/(1+e^{\varepsilon/2})$, obtaining the obfuscated vector $\tilde{v}_x^{h_j}$. The output of CMS consists of the obfuscated vector and the index of the hash function used to compute it:

$$\widetilde{x} = \text{CMS}(x;\, \varepsilon, m, \mathcal{H}) = (\tilde{v}_x^{h_j}, j)$$

CMS satisfies ε-local differential privacy for any $\varepsilon > 0$ [4]. The parameters ε, m and $\mathcal{H}$ used by CMS on users' devices are typically set by the data curator. In particular, smaller ε yield lower accuracy, but give better privacy guarantees. Moreover, the hash functions satisfy some technical properties that ensure their behavior is tractable with probabilistic methods — see Appendix A.1 for this and other details on CMS.

We note that the use of hash functions is not necessary to achieve local differential privacy, but they make CMS more space-efficient and offer additional privacy protection due to hash collisions[2]. In fact, even if no bits are flipped, there are often many original objects producing the same one-hot vector, with the exact number depending on m and on the hash function. Collisions make it impossible to infer the original object from the obfuscated object. However, if the user is likely to select most objects from a specific set (pool), after multiple observations this fact can be inferred despite hash collisions. This is the intuition behind our attack.

3 Pool inference attacks against local differential privacy

We define a new general attack model against local differential privacy mechanisms, that we call *pool inference attack model*. We then propose an attack for this attack model, which we call the Bayesian Pool Inference Attack (BPIA).

3.1 Formalizing the pool inference attack model

We consider an attack where objects are semantically grouped in *pools* (e.g. skin tone of emojis, political orientation of news websites), and the adversary tries to infer which pool a target user samples from most frequently (their *preferred pool*). Formally, we define the pool inference attack model as a game between an adversary Adv and a target user Usr who obfuscates their data with a mechanism $\mathcal{A}$. We model the user behavior as a probability distribution Φ_{Usr} over the universe Ω, reflecting the target user's preferences for the objects in Ω — i.e. the probability that Usr selects a certain object.

[2]We note that the additional protection coming from collisions is not captured by the privacy loss ε, and hence requires practical attacks like ours to be quantified.

[3]The estimated popularity is always assumed to be known to Adv but may be uninformative when Adv uses no auxiliary information, i.e. if $\mathcal{I} = \emptyset$.

Symbol	Description	Known to Adv
Adv	Adversary (runs the attack)	
Usr	User (target of the attack)	
Ω	Universe of (original) objects	Yes
$\mathcal{Y}$	Space of obfuscated objects	Yes
$\mathcal{A}$	Mechanism	Yes
CMS	Count Mean Sketch mechanism	Yes
ε	Privacy loss parameter	Yes
$\mathcal{H}$	Family of hash functions	Yes
m	Length of obfuscated vector	Yes
n	Number of observations	Yes
$x_1, \ldots, x_n$	Original objects	No
Φ_{Usr}	Usr's behavior	No
P_{Usr}	Usr's preferred pool	No
$\{P_i : P_i \neq P_{\text{Usr}}\}$	Usr's alternative pools	No
γ_{Usr}	Usr's relevant interest	No
δ_{Usr}	Usr's polarization	No
p_Ω	True object popularity	No
$\widetilde{x}_1, \ldots, \widetilde{x}_n$	Obfuscated objects (or observations)	Yes
$P_1, \ldots, P_k$	Adv's pools of interest	Yes
$\Omega \setminus \cup_{i=1}^k P_i$	Neutral pool	Yes
$\mathcal{I}$	Adv's auxiliary information	Yes
score(P_i)	Adv's score for pool P_i	Yes
$\widehat{P}_{\text{Usr}}$	Adv's estimated preferred pool	Yes
conf$(\widehat{P}_{\text{Usr}})$	Adv's confidence value	Yes
$\widehat{\Phi}$	Adv's user representation	Yes
$\hat{p}_\Omega$	Adv's estimated object popularity	Yes[3]

Table 1: Notation and definitions. We indicate which elements are known to the adversary according to the pool inference attack model.

Pool Inference Game.

- *Step 1.* Usr samples n original objects $x_1, \ldots, x_n$ independently according to Φ_{Usr}. Then, Usr runs $\mathcal{A}(x_t)$ independently on each x_t, producing the obfuscated objects $\widetilde{x}_1, \ldots, \widetilde{x}_n$.
- *Step 2.* Adv selects k *pools of interest* $P_1, \ldots, P_k \subseteq \Omega$, which are pairwise disjoint subsets of Ω that can have arbitrary and different sizes.
- *Step 3.* Usr sends $\widetilde{x}_1, \ldots, \widetilde{x}_n$ to Adv.
- *Step 4.* Adv runs an attack that returns one pool $\widehat{P}_{\text{Usr}} \in \{P_1, \ldots, P_k\}$, which we call Adv*'s estimated preferred pool*.

Adv wins the game if $\widehat{P}_{\text{Usr}} = P_{\text{Usr}}$, where

$$P_{\text{Usr}} \stackrel{\text{def}}{=} \arg\max_{P_1,\ldots,P_k} \Phi_{\text{Usr}}(P_i)$$

is the user's (true) *preferred pool among* $P_1, \ldots, P_k$.

Without loss of generality, we always assume that the preferred pool P_{Usr} is unique, i.e. $\Phi_{\text{Usr}}(P_i) < \Phi_{\text{Usr}}(P_{\text{Usr}})$ for all $P_i \neq P_{\text{Usr}}$. We refer to all the pools in $\{P_i : P_i \neq P_{\text{Usr}}\}$ as *alternative pools*. We also define the *neutral pool* as the set $\Omega \setminus \cup_{i=1}^k P_i$ of all objects not in any pool, and call its elements *neutral objects*. Figure 1 provides an illustration of these definitions where the pools are defined in a universe of emojis grouped by skin tone.

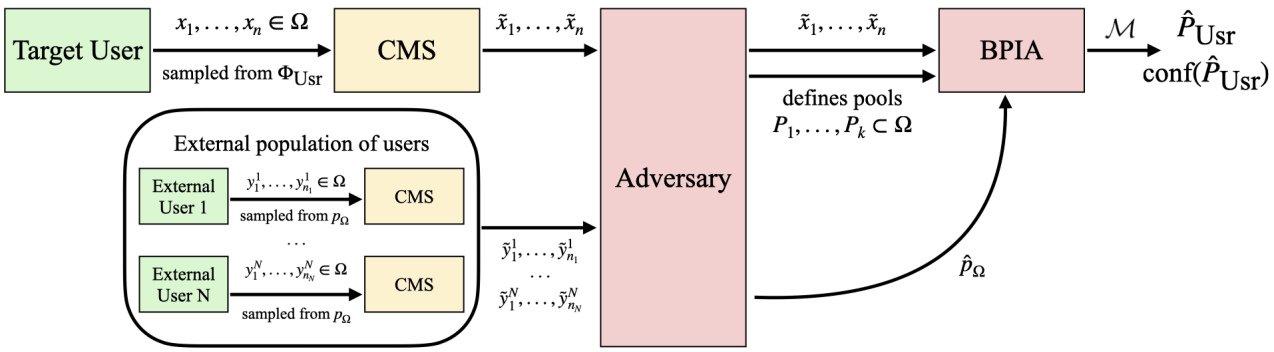

Figure 2: Diagram summarizing the Bayesian Pool Inference Attack (BPIA).

We note that, in practice, Adv could repeat the attack in Step 4 using the same obfuscated objects received in Step 3, but using different pools. For example, the attack could be first run using pools for skin tone, and then again using pools grouped by gender. While this may be a likely case in a real-world setting where the adversary may try to infer as much sensitive information as possible, in this paper we limit our analysis to the case when the attack is run only once for each user (i.e. for each instance of the game).

Adversary's knowledge. No information is shared from Usr to Adv or vice versa, except in Step 3, where Usr sends the obfuscated objects to Adv. The pools defined by the adversary do not depend on the objects sampled by the user (and vice versa). The only information that Adv knows about Usr are the obfuscated objects $\widetilde{x}_1,\ldots,\widetilde{x}_n$, which we call *observations*. We also admit the possibility that Adv has access to some auxiliary information $\mathcal{I}$, which represents general knowledge about the population (not about Usr specifically) that can be used in the attack in Step 4. Finally, we assume that Adv knows the universe Ω, the privacy loss ε, and any other internal parameter used when applying $\mathcal{A}$ — a standard assumption for attacks, where the specifications of the system are assumed to be public. Table 1 summarizes the notation and what is known to the adversary.

Behavioral parameters. Usr's behavior determines how vulnerable they are to a pool inference attack: Usr might mostly use objects in the neutral pool, or their preference for their preferred pool might not be strong. For example, Usr might use skin-toned emojis only very rarely; moreover, regardless of the relevant interest, it might be that Usr selects the medium-light skin tone more frequently, but actually selects other skin tones often as well. To capture these properties of Usr's behavior, we define two *behavioral parameters*: the *relevant interest* γ_{Usr} (how often Usr samples from pools of interest) and the *polarization* δ_{Usr} (among pools of interest, how often Usr samples from their preferred pool). Formally:

$$\gamma_{\text{Usr}} \stackrel{\text{def}}{=} \Phi_{\text{Usr}}\left(\bigcup_{i=1}^{k} P_i\right) \quad \text{and} \quad \delta_{\text{Usr}} \stackrel{\text{def}}{=} \frac{1}{\gamma_{\text{Usr}}}\Phi_{\text{Usr}}(P_{\text{Usr}})$$

which satisfy $0 < \gamma_{\text{Usr}} \leq 1$ and $\frac{1}{k} < \delta_{\text{Usr}} \leq 1$ (since P_{Usr} is maximal with respect to Φ_{Usr}). While these parameters are unknown to Adv, they are useful to describe each game instance and characterize the user's behavior. In section 4, we show that these parameters capture how vulnerable the target user is *with respect to the specific set of pools chosen by Adv*.

3.2 BPIA: A Bayesian pool inference attack

We propose an attack using Bayesian inference for the pool inference attack model, that we call BPIA (*Bayesian Pool Inference Attack*). We first summarize the intuition behind the attack. Given Usr's obfuscated objects $\widetilde{x}_1,\ldots,\widetilde{x}_n$, BPIA uses Bayesian inference to compute, for each pool P_i, the a posteriori probability that P_i is Usr's preferred pool:

$$\Pr[P_{\text{Usr}} = P_i \mid \widetilde{x}_1,\ldots,\widetilde{x}_n] \tag{1}$$

To compute this probability, BPIA must take into account (1) the uncertainty on Usr's preferred pool and behavioral parameters, (2) the randomness of Usr's behavior, and (3) the randomness of the mechanism $\mathcal{A}$. To do this, BPIA uses a hierarchical model that combines the three types of uncertainty. In particular, for (2), BPIA would ideally use the user behavior Φ_{Usr}, but this is unknown to Adv. Instead, BPIA uses a function $\overline{\Phi}$ — that we call *user representation* — parameterized by three parameters γ, δ and ι, which models a simple user behavior. We now give the details of the hierarchical model, the user representation and BPIA's output.

Hierarchical model. We propose a general hierarchical model $\mathcal{M} = (\mathcal{A}, \overline{\Phi}, \mathcal{I})$, where $\mathcal{A}$ is the obfuscation mechanism, $\overline{\Phi}$ is a *user representation* of the (unknown) user behavior, and $\mathcal{I}$ is some additional auxiliary information that contains general facts about the population (see below).

The user representation is a distribution $\overline{\Phi}(x \mid \iota, \gamma, \delta, \mathcal{I})$, parameterized by $\iota \in \{1,\ldots,k\}$ (the preferred pool), $\gamma \in (0,1]$, $\delta \in (1/k, 1]$ (behavioral parameters), and the auxiliary information $\mathcal{I}$. The function $\overline{\Phi}(x \mid \iota, \gamma, \delta, \mathcal{I})$ gives the (assumed) probability of choosing an original object x if the user has P_ι as their preferred pool, behavioral parameters γ and δ, and subject to additional auxiliary information $\mathcal{I}$.

Intuitively, $\mathcal{M}$ models a user who is first assigned the preferred pool P_ι, the relevant interest γ and the polarization δ uniformly at random; then, the user samples n original objects independently according to $\overline{\Phi}(\cdot \mid \iota, \gamma, \delta, \mathcal{I})$; and finally obfuscates them using $\mathcal{A}$. Formally, $\mathcal{M}$ is given by three hyperparameters ι, γ, δ, the random variable $(X_1, \ldots, X_n)$ representing the sampling of the original objects, and the random variable $(\widetilde{X}_1, \ldots, \widetilde{X}_n)$ denoting its randomly obfuscated version, with the auxiliary information $\mathcal{I}$ being treated as a fixed parameter:

$$\begin{aligned} \iota &\sim \text{Uniform}(\{1, \ldots, k\}) \\ \gamma &\sim \text{Uniform}((0, 1]) \\ \delta &\sim \text{Uniform}((1/k, 1]) \\ X_t \mid \iota, \gamma, \delta &\sim \overline{\Phi}(\cdot \mid \iota, \gamma, \delta, \mathcal{I}) \quad \forall t \in \{1, \ldots, n\} \\ \widetilde{X}_1, \ldots, \widetilde{X}_n \mid X_1, \ldots, X_n &\sim \mathcal{A}(X_1), \ldots, \mathcal{A}(X_n) \end{aligned}$$

Using this model, the adversary is able to compute the probabilities in eq. 1: $\Pr_{\mathcal{M}}[P_{\text{Usr}} = P_i \mid \widetilde{x}_1, \ldots, \widetilde{x}_n]$. While in this model the hyperparameters ι, γ, and δ are uniformly distributed — reflecting an adversary who has no informative prior on P_{Usr}, γ_{Usr}, and δ_{Usr} — this could likely be improved in practical settings where the adversary may have access to additional sources of information (see Appendix A.6).

User representation. Our user representation $\overline{\Phi}(x \mid \iota, \gamma, \delta, \hat{p}_\Omega)$ models the user assuming the following behavior: the user first chooses a pool (the neutral pool with probability $1-\gamma$, their preferred pool P_ι with probability $\gamma\delta$, or any of the alternative pools with equal probability $\frac{1}{k-1}\gamma(1-\delta)$), then samples an object from the selected pool according to some *estimated object popularity* $\hat{p}_\Omega$. This object popularity is a distribution over Ω that — intuitively — captures the differences in likelihood for objects *within the same pool*. For example, $\hat{p}_\Omega$ can capture the fact that, among emojis with the same skin tone, the thumb-up emoji is much more popular across users than most of the others. We assume that the adversary has access to this estimated object popularity as additional auxiliary information: $\mathcal{I} = \hat{p}_\Omega$. In section 6 we discuss how an adversary could acquire the object popularity from external sources or even estimate it from obfuscated objects collected from other users. Furthermore, when the adversary does not have any auxiliary information, Adv can use an *uninformative* object popularity $\hat{p}_\Omega$, such as the uniform distribution on Ω.

Formally, the representation $\overline{\Phi}$ is defined as follows:

$$\overline{\Phi}(x \mid \iota, \gamma, \delta, \hat{p}_\Omega) = \begin{cases} \gamma\delta \frac{\hat{p}_\Omega(x)}{\hat{p}_\Omega(P_\iota)} & \text{if } x \in P_\iota \\ \frac{1}{k-1}\gamma(1-\delta)\frac{\hat{p}_\Omega(x)}{\hat{p}_\Omega(P_i)} & \text{if } x \in P_i, i \neq \iota \\ (1-\gamma)\frac{\hat{p}_\Omega(x)}{\hat{p}_\Omega(\Omega \setminus \cup_{i=1}^k P_i)} & \text{if } x \in \Omega \setminus \cup_{i=1}^k P_i \end{cases} \tag{2}$$

We note that in the equation, $\hat{p}_\Omega(x)$ is always normalized by the total popularity of the pool that x belongs to. In other words, $\hat{p}_\Omega(x)$ is used exclusively to differentiate the probability of different objects within the same pool — it has no effect on the overall probability that $\overline{\Phi}$ assigns to a pool (and hence to the pool's score, see next paragraph).

We emphasize that the user representation $\overline{\Phi}(x \mid \iota, \gamma, \delta, \hat{p}_\Omega)$ is a simple *model* for the user's behavior: Adv does not know whether the representation correctly describes the actual user behavior Φ_{Usr}, and does not know the exact value of P_{Usr}, γ_{Usr}, and δ_{Usr}. In particular, our user representation does not account for (1) individual preferences within pools differing from $\hat{p}_\Omega$, and (2) preferences between non-preferred pools (since our model assumes that the user selects among alternative pools uniformly at random). In the extended version of this paper we present some results that quantify how the correctness of the user representation affects the effectiveness of the attack.

Maximum a posteriori estimate. The attack attempts to find the user's preferred pool from their obfuscated objects by computing the posterior probability of each pool. For each pool P_i, the adversary computes a *score* proportional to the conditional probability that $P_{\text{Usr}} = P_i$ under the model $\mathcal{M}$:

$$\text{score}(P_i) \propto \Pr_{\mathcal{M}}[P_{\text{Usr}} = P_i \mid \widetilde{x}_1, \ldots, \widetilde{x}_n] \tag{3}$$

The adversary then selects the *maximum a posteriori estimate* for the user's preferred pool, as the pool with maximal score:

$$\widehat{P}_{\text{Usr}} = \arg\max_{P_1, \ldots, P_k} \text{score}(P_i)$$

If several pools have maximal score, the estimate is selected uniformly at random from these. The attack also computes a confidence value $\text{conf}(\widehat{P}_{\text{Usr}})$ quantifying the probability (under the model $\mathcal{M}$) that the estimate is correct:

$$\text{conf}(\widehat{P}_{\text{Usr}}) \stackrel{\text{def}}{=} \Pr_{\mathcal{M}}[\widehat{P}_{\text{Usr}} = P_{\text{Usr}} \mid \widetilde{x}_1, \ldots, \widetilde{x}_n] = \frac{\text{score}(\widehat{P}_{\text{Usr}})}{\sum_{i=1}^k \text{score}(P_i)}$$

For an arbitrary confidence threshold τ defined by the adversary, the attack outputs $\widehat{P}_{\text{Usr}}$ if $\text{conf}(\widehat{P}_{\text{Usr}}) \geq \tau$ and *null* otherwise. The threshold τ hence allows the adversary to set the minimum level of confidence that they require to trust the attack's estimate $\widehat{P}_{\text{Usr}}$. The attack is successful if the estimate is correct, i.e. $\widehat{P}_{\text{Usr}} = P_{\text{Usr}}$.

Score computation. Under the model $\mathcal{M}$, the scores defined in eq. 3 are computed as the probability that $P_{\text{Usr}} = P_i$ after observing $\widetilde{x}_1, \ldots, \widetilde{x}_n$, obtained by integrating the conditional distribution over γ and δ and applying Bayes's law:

$$\text{score}(P_i) \propto \int_0^1 \int_{\frac{1}{k}}^1 \prod_{t=1}^n \sum_{z \in \Omega} \Pr_{\mathcal{A}}[\widetilde{x}_t \mid z]\ \overline{\Phi}(z \mid i, \gamma, \delta, \hat{p}_\Omega)\ d\delta\, d\gamma \tag{4}$$

The term $\Pr_{\mathcal{A}}[\widetilde{x}_t \mid z]$ is the probability that the output of $\mathcal{A}(z)$ is the observation $\widetilde{x}_t$. We give a formal proof of correctness of the score in the extended paper. We next show how to compute this for CMS.

Attacking CMS. To execute BPIA against the mechanism $\mathcal{A} = \text{CMS}$, we need to determine the value of $\Pr_{\text{CMS}}[\widetilde{x} \mid z]$ for any $\widetilde{x}$ and any z. First of all, we note that

$$\Pr_{\text{CMS}}[\widetilde{x} \mid z] = \Pr_{\text{CMS}}[(\tilde{v}_x^{h_j}, j) \mid z] = \Pr[\tilde{v}_x^{h_j} \mid j, z] \Pr[j \mid z]$$

Since j is selected uniformly at random, we have that $\Pr[j \mid z] = \Pr[j]$ is constant for any z and can be moved outside of the integral in eq. 4. Hence, this is a multiplicative value that is constant across pools and can be ignored.

$\Pr[\tilde{v}_x^{h_j} \mid j, z]$ is the probability of obtaining the obfuscated vector $\tilde{v}_x^{h_j}$ when the original object is z and the selected hash function is h_j. Since Adv knows all CMS parameters — including the hash functions in $\mathcal{H}$ — they can compute the one-hot vector $v_z^{h_j}$. The probability is then derived by observing how many bits need to be flipped in order to obtain $\tilde{v}_x^{h_j}$ from $v_z^{h_j}$, i.e. their Hamming distance. Let $\xi = 1/(1+e^{\varepsilon/2})$ be the probability of flipping one bit and let $\|\cdot\|_1$ denote the L_1 norm. We obtain:

$$\Pr[\tilde{v}_x^{h_j} \mid j, z] = \xi^{\left\|v_z^{h_j} - \tilde{v}_x^{h_j}\right\|_1} (1-\xi)^{m - \left\|v_z^{h_j} - \tilde{v}_x^{h_j}\right\|_1} \tag{5}$$

We note that eq. 5, when used to compute the score in eq. 4, automatically captures the uncertainty coming from the random flipping of bits and from hash collisions as well. For example, if two objects in different pools share the same hash value, this would make it impossible to distinguish which of them (if any) was Usr's original object. The attack takes this fact into account when computing the scores for those pools.

4 Experiments on synthetic users

In this section we empirically validate our BPIA attack against CMS for synthetic users. For each user, we define the behavior Φ_{Usr} and we then use it to sample the original objects. This allows us to evaluate the attack for different user profiles (relevant interest and polarization) and compare the results across different settings.

Experiment design

We simulate BPIA in various experiment scenarios. Each *experiment scenario* is defined by the following parameters:

(i) the universe Ω;

(ii) the CMS parameters ε, m, $\mathcal{H}$ (see section 2);

(iii) the pools of interest $P_1, \ldots, P_k \subseteq \Omega$ picked by Adv for the attack;

(iv) the *true object popularity* p_Ω, a distribution on Ω (not known to Adv);

(v) the *estimated object popularity* $\hat{p}_\Omega$ (known to Adv);

(vi) the number of observations n that Adv has access to.

Using these parameters, we run 150,000 independent instances of the pool inference game, with one (independent) synthetic user per instance. For each user Usr, we sample the user's relevant interest γ_{Usr} and polarization δ_{Usr} uniformly at random from $(0,1]$ and $(1/k,1]$, respectively. As will become clear from the results, these behavioral parameters strongly impact the success rate of BPIA. Sampling the parameters uniformly allows us to study the effectiveness of the attack on users with different levels of vulnerability.

User behavior. We select Usr's preferred pool P_{Usr} uniformly at random from $\{P_1, \ldots, P_k\}$. For each instance of the game, we use the randomly sampled γ_{Usr}, δ_{Usr}, and P_{Usr} to define Usr's behavior Φ_{Usr}, as follows:

$$\Phi_{\text{Usr}}(x) \stackrel{\text{def}}{=} \begin{cases} \gamma_{\text{Usr}}\delta_{\text{Usr}} \frac{p_\Omega(x)}{p_\Omega(P_{\text{Usr}})} & \text{if } x \in P_{\text{Usr}} \\ \frac{1}{k-1}\gamma_{\text{Usr}}(1-\delta_{\text{Usr}}) \frac{p_\Omega(x)}{p_\Omega(P_i)} & \text{if } x \in P_i \neq P_{\text{Usr}} \\ (1-\gamma_{\text{Usr}}) \frac{p_\Omega(x)}{p_\Omega(\Omega \setminus \cup_{i=1}^k P_i)} & \text{if } x \in \Omega \setminus \cup_{i=1}^k P_i \end{cases} \tag{6}$$

This means that to sample each original object, the user first selects P_{Usr} with probability $\gamma_{\text{Usr}}\delta_{\text{Usr}}$, any other pool of interest with probability $\frac{1}{k-1}\gamma_{\text{Usr}}(1-\delta_{\text{Usr}})$ and the neutral pool with probability $1-\gamma_{\text{Usr}}$. Once one pool has been selected, the original object is sampled within that pool according to the object popularity p_Ω.

For an instance of the game, we sample n objects from Φ_{Usr} and obfuscate them with $\text{CMS}(\cdot; \varepsilon, m, \mathcal{H})$. Note here that Φ_{Usr} corresponds to Adv's user representation $\overline{\Phi}$ in eq. 2 but using p_Ω (as Adv does not know the true popularity p_Ω). The robustness results we report in the extended paper quantify the effectiveness of the attack when Usr uses a noisy version of p_Ω instead of the exact one.

Non-private scenario. To understand the protection provided by CMS against BPIA, we also report results for an idealized scenario where the mechanism $\mathcal{A}$ simply reveals the original object x (i.e. $\mathcal{A}$ is the identity function), and hence Adv has access to the original objects $x_1, \ldots, x_n$. We refer to this as the *non-private* scenario. In the non-private scenario, BPIA works in the same way as for CMS but the score in eq. 3 is computed by setting $\Pr_{\mathcal{A}}[\widetilde{x}_t \mid z] = 1$ if $x_t = z$ and $\Pr_{\mathcal{A}}[\widetilde{x}_t \mid z] = 0$ if $x_t \neq z$.

Baseline. For each scenario, we report as baseline the attack that always makes a guess (i.e. has fixed confidence score $\text{conf} = 1$) and returns one of the pools $P_1, \ldots, P_k$ uniformly at random. Since we select the user's preferred pool uniformly at random in the experiments, the baseline attack is correct with probability $1/k$.

Types of adversary. We simulate two types of adversaries: Adv_{weak} and Adv_{strong}. Adv_{strong} has access to auxiliary information on objects' popularity $\hat{p}_\Omega$ that approximates p_Ω, while Adv_{weak} uses a uniform $\hat{p}_\Omega$.

We consider Adv_{strong} to represent a realistic scenario for a typical deployment of local differential privacy (see Discussion). Indeed $\hat{p}_\Omega$ can be estimated from auxiliary information

derived from an external dataset $\widetilde{\mathcal{D}}_{ext}$ of CMS-obfuscated objects collected from other users. We here simulate Adv_{strong}'s estimation of $\hat{p}_\Omega$ by independently sampling $N = 10^6$ original objects from p_Ω obtaining $\mathcal{D}_{ext} = \{y_1, \ldots, y_N\}$. We then obfuscate them with CMS, obtaining the external dataset $\widetilde{\mathcal{D}}_{\text{ext}} = \{\widetilde{y}_1, \ldots, \widetilde{y}_N\}$ which would typically be available to Adv_{strong}. Using Apple's algorithm [4] the adversary approximates the frequencies of objects of the original dataset $\mathcal{D}_{ext}$, then projects these frequencies to the probability simplex using alternating projection [6] in order to obtain the estimated object popularity $\hat{p}_\Omega$ (which approximates p_Ω well when the number of users N is sufficiently large).

The estimated object popularity $\hat{p}_\Omega$ is the only difference between Adv_{weak} and Adv_{strong}. Both adversaries use the same hierarchical model $\mathcal{M}$ with the same hyperparameters (in particular, they both always integrate over uniformly distributed γ and δ when computing the pools' scores).

Importantly, we note that the effectiveness of the attack in the non-private scenario is the same for Adv_{strong} and Adv_{weak}. In the non-private scenario, there is no uncertainty regarding the original input — as the output and the input objects coincide — and hence knowing the object popularity does not bring any advantage.[4]

Metrics. For a given threshold τ, we call *null users* all the users for which the attack does not make a guess ($\text{conf}(\widehat{P}_{\text{Usr}}) < \tau$). We then use the following three metrics to measure the effectiveness of our attack in a given scenario:

1. The *null rate* is the fraction of null users (out of all the 150,000 users) for a given value of τ;
2. The *precision* is the success rate of the attack for all non-null users, i.e. the fraction of non-null users such that $\widehat{P}_{\text{Usr}} = P_{\text{Usr}}$. That is, the fraction of users for which the attack's guess is correct, out of the users for which a guess is made (which depends on the threshold τ);
3. The *area under the precision-null rate curve* (AUC-PN) is the area under the curve obtained by plotting the precision vs the null rate for all possible threshold values between 0 and 1. Since the threshold τ can be adapted by the adversary to adjust the tradeoff between precision and null rate, the AUC-PN captures the overall effectiveness of the attack (in the specific scenario).

Settings. In this paper, we focus on two specific use cases of CMS implemented by Apple in iOS and Mac OS [4]:

> **Setting 1: Emojis.** In this use case, the device keeps track of which emojis — the original objects — are selected by the user when typing. These are obfuscated by CMS and submitted to Apple. The universe of objects contains 2600 emojis, i.e. $|\Omega| = 2600$.
>
> **Setting 2: Web domains.** For this setting, the original objects are the web domains that the user visits using the built-in browser (together with preferences regarding videos autoplay). The implementation of CMS keeps track of 250,000 web domains, i.e. $|\Omega| = 250000$.

Apple's implementation sets $m = 1024$ and $|\mathcal{H}| = 65536$, with $\varepsilon = 8$ for web domains and $\varepsilon = 4$ for emojis.

[4] This fact can be proved formally by noticing that in the non-private scenario the sum in eq. 4 reduces to one single term, so that the object popularity for each observation can be moved outside of the integral and is constant across each pool's score.

Setting 1: Emojis

We consider an adversary Adv that runs our BPIA attack with the goal of inferring Usr's preferred emoji skin tone (see Figure 1). To this end, Adv defines six pools of size 228, corresponding to the six skin tones supported for 228 emojis in the Unicode Emoji v11.0 standard [35].

We define the true object popularity p_Ω as a mixture of Zipfian distributions — reflecting the fact that a few emojis are much more popular than others [15] (we discuss this choice in more detail in the extended paper). Formally, we consider the partition of Ω given by the pools $P_1, \ldots, P_k$ and the neutral pool $Q = \Omega \setminus \cup_{i=1}^{k} P_i$. For each $P_i = \{x^i_1, \ldots, x^i_{|P_i|}\}$, we take the Zipfian probability mass function given by:

$$f_{P_i}(x^i_j) = \frac{1/j^{1.2}}{\sum_{c=1}^{|P_i|} 1/c^{1.2}} \tag{7}$$

and similarly for the neutral pool. Finally we define:

$$p_\Omega(x) \overset{\text{def}}{\propto} \begin{cases} f_{P_i}(x) & \text{if } x \in P_i, \quad i = 1, \ldots, k \\ f_Q(x) & \text{if } x \in Q \end{cases} \tag{8}$$

Results for Adv_{weak} and Adv_{strong}. We simulate the attack with $n = 7, 30, 90, 180$ observations. Since Apple collects one obfuscated object per day [2], these correspond to about 1 week, 1 month, 3 months, and 6 months, respectively. While 6 months may seem a long time, most users are likely to keep their iOS and Mac OS devices —and submit obfuscated objects — for much longer than that [31].

Table 2 shows that our attack performs well for Adv_{strong}, already reaching an AUC-PN of 0.8 after $n = 90$ observations. Figure 3 shows the attack's full precision-null rate curves. Here again, we see that Adv_{strong} performs much better than the baseline, reaching a precision of 0.29 after only 7 observations, and 0.64 after 180 observations when making a guess for all users (null rate = 0).

Restricting the attack to only users for which the attack is more confident (higher thresholds) allows the adversary to considerably increase the precision while making predictions on a significant number of users. For instance, for $n = 90$, the attack reaches a precision of 1 for a null rate of 0.95. This means that the attack makes no mistake when executed on the top 5% users (i.e. the users whose confidence score is in the top 5%). Even with a week of observations ($n = 7$), the attack

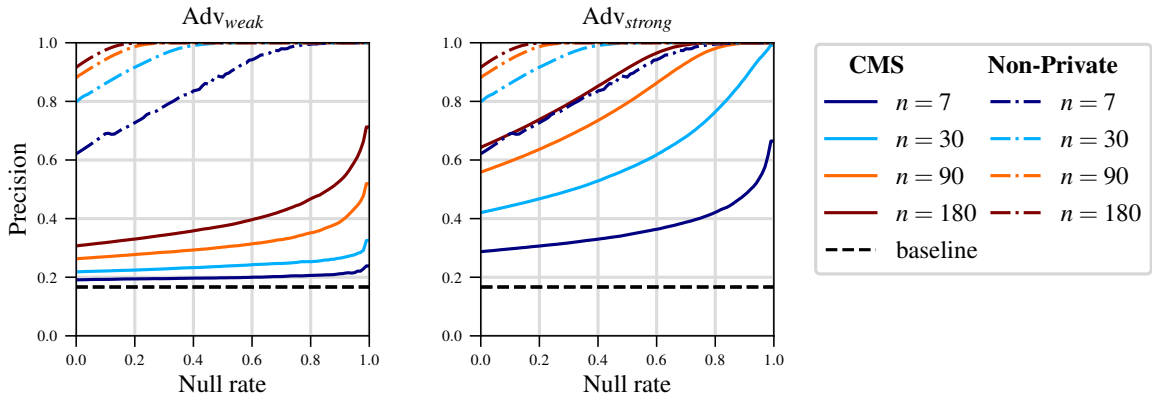

Figure 3: Precision-null rate curves in the emojis setting for Adv$_{weak}$ and Adv$_{strong}$. The results for the non-private scenario are the same for Adv$_{weak}$ and Adv$_{strong}$.

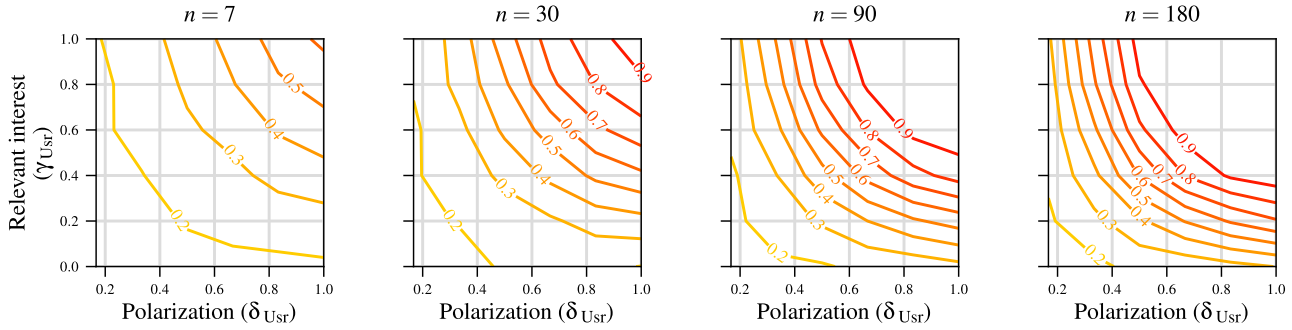

Figure 4: Precision depending on γ_{Usr} and δ_{Usr} for Adv$_{strong}$ in the emojis setting when the attack always makes a guess (null rate $= 0$). The attack is more efficient when the user's relevant interest and polarization are higher. The figure is generated by computing, for each value of γ_{Usr} and δ_{Usr}, the precision of the attack on users with (approximately) those relevant interest and polarization values. We note that δ_{Usr} is always greater than $1/(6-1) = 0.2$ by definition (see section 3).

	$n = 7$	$n = 30$	$n = 90$	$n = 180$
Adv$_{weak}$	0.20	0.24	0.32	0.40
Adv$_{strong}$	0.37	0.61	0.80	0.88
Non-private	0.86	0.96	0.99	0.99

Table 2: AUC-PN values in the emojis setting.

reaches 48% precision (2.9 times better than the baseline) when focusing on the top 10% of users.

The results for Adv$_{weak}$, while significantly better than the baseline, are not as good. When making a guess for all the users, Adv$_{weak}$ only reaches a precision of 0.19 for $n = 7$ (as opposed to 0.29 for Adv$_{strong}$). Even after $n = 180$ observations, the precision only increases to 0.31 for null rate $= 0$, and to 0.53 when focusing on the top 10% users. These results emphasize the importance of the adversary using auxiliary information during the attack.

The reason Adv$_{strong}$ achieves much better results compared to Adv$_{weak}$ can be intuitively explained as follows: BPIA uses the object popularity to reduce the indistinguishability of the obfuscated objects. In principle, each obfuscated object may be the output of CMS run on any original object. However, if the attack knows that some of these objects are less likely to be picked (compared to others *in the same pool*), the posterior probability that one of them was the actual input can be reduced accordingly. The score defined in eq. 4 captures this fact to compute each pool's posterior probability. We provide additional results on this point in Appendix A.3.

Results in the non-private scenario. In order to contextualize our results, we also measure the accuracy of BPIA in the non-private scenario, when the adversary has access to the user's original objects $x_1, \ldots, x_n$ (i.e. without hashing nor obfuscation). This gives an upper bound to the attack: even when the adversary observes the original objects, they can still make mistakes when estimating the user's preferred pool. This is due to the stochastic nature of the user behavior Φ_{Usr}. For example, a user might use emojis with a certain skin tone most of the times but, for most users, there is a non-zero and possibly significant probability that the user selects emojis with a different skin tone (alternative pool) or even an

emoji with no skin tone (neutral pool). Hence, even in the non-private scenario the attack might not be 100% effective.

Figure 3 shows the attack to be highly effective in the non-private scenario, although not perfect. While the difference in effectiveness between Adv_{weak} and non-private remains large for any number of observations, the protection offered by CMS decreases as n increases.

Impact of the behavioral parameters. Figure 4 shows how the precision of the attack increases with both behavioral parameters δ_{Usr} and γ_{Usr} for Adv_{strong} when the null rate is 0, i.e. when the attack makes a guess for all users. Users with larger polarization and relevant interest tend to be, on average, much more vulnerable than other users. For instance for $n = 90$ and for Adv_{strong}, the precision of the attack on a user with $\gamma_{\text{Usr}} = 0.2$ and $\delta_{\text{Usr}} = 0.17$ is lower than 20%, while it already increases to more than 80% for a user with $\gamma_{\text{Usr}} = 0.6$ and $\delta_{\text{Usr}} = 0.67$. Overall, Adv_{strong} performs well over a large range of values of γ_{Usr} and δ_{Usr} for $n \geq 90$.

Setting 2: Web domains

We here consider the case of an adversary attempting to infer the target user's potential political orientation from news sites that they visit. In this hypothetical setting, the adversary assumes that users are more likely to visit news websites whose political orientation is aligned with their own political views [19, 32]. The adversary hence defines the pools as sets of news websites grouped by political orientation. We here use AllSides's Media Bias rating for 60 major English-language news websites [1]. The Chart divides media into five groups: left, lean left, center, lean right, right, which contain respectively 14, 13, 13, 10, and 10 unique news websites[5] (see Figure 5). In this experiment we randomly assign a popularity to all websites in the universe. For each object $x \in \Omega$, $p_\Omega(x)$ is sampled uniformly at random from $[0,1]$ (and then rescaled to ensure that p_Ω has total mass adding up to 1). To reduce the computational time required to run the attack on 150,000 users, we run the experiments with a universe of size $|\Omega| = 2000$ (instead of the original 250,000). We show in Appendix A.4 that this has no impact on the estimated effectiveness of the attack.

Here again, we simulate two adversaries: Adv_{weak} who uses an uninformative (uniform) object popularity $\hat{p}_\Omega$, and Adv_{strong} who uses $N = 10^6$ obfuscated objects from an external population to derive the estimated popularity $\hat{p}_\Omega$.

Results for Adv_{weak} and Adv_{strong}. Table 3 reports the AUC-PN of the attack (computed on all users, for any relevant interest and polarization), and shows that both adversaries are very effective. Adv_{weak} and Adv_{strong} reach high AUC-PN with few observations. For example, they both obtain AUC-PN ≥ 0.95 with $n = 90$ observations.

Interestingly, the effectiveness of Adv_{strong} in this scenario is very similar to the one of Adv_{weak} — a stark difference from the emojis setting (Table 3 and Figure 6). This can be explained by the comparatively much smaller pools in this use case compared to the emojis setting (average pool size of 12, compared to 228 in the emoji setting). Indeed as pools get smaller, both the risk of hash collisions between two objects of different pools and the uncertainty introduced by the randomized obfuscation increase (see Appendix A.3).

The small difference in both AUC-PN and precision, between both adversaries and the non-private scenario further confirms that, when pools are small, CMS provides little additional protection.

Impact of the behavioral parameters. Figure 7 shows the precision for $\tau = 0$ as a function of the relevant interest γ_{Usr} — the fraction of the time a user visits one of the 60 news websites — and the polarization δ_{Usr} for Adv_{strong}. We omit the results for Adv_{weak} as they are almost identical. Similarly to the emojis setting (Figure 4), we find that a user's behavioral parameters strongly affect how vulnerable they are. For instance, for Adv_{strong} and $n = 90$, the attack will be correct 91% of the time on a user who visits news websites 20% of

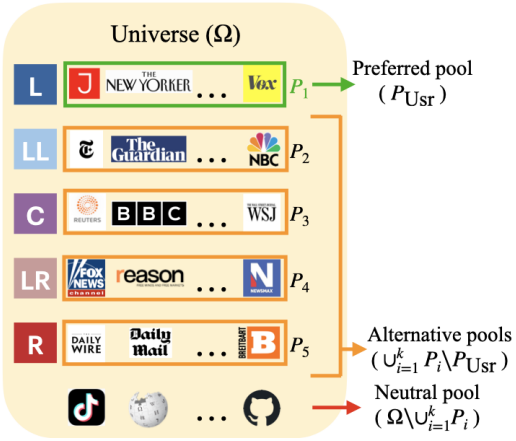

Figure 5: Pools for the web domains setting. Each pool groups together websites for 60 major news outlets according to their political orientation from the 2021 AllSides Media Bias Chart (left, lean left, center, lean right, right). In this case, Usr visits most frequently news websites in the left pool.

	$n = 7$	$n = 30$	$n = 90$	$n = 180$
Adv_{weak}	0.72	0.89	0.95	0.97
Adv_{strong}	0.74	0.90	0.96	0.98
Non-private	0.87	0.96	0.99	0.99

Table 3: AUC-PN values in the web domains setting.

[5] In a few cases, the chart by AllSides has two entries for the same website — e.g., for The Wall Street Journal, the *news only* section is rated center and the *opinion* section is rated lean right. As these share the same web domain, for simplicity we include just the *news only* entries in the pools of interest.

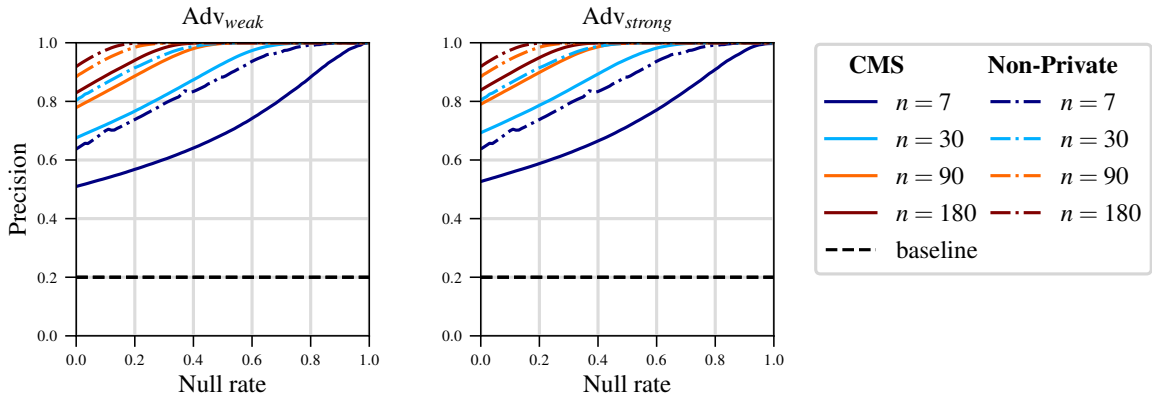

Figure 6: Precision-null rate curves in the web domains setting for Adv$_{weak}$ and Adv$_{strong}$. The results for the non-private scenario are the same for Adv$_{weak}$ and Adv$_{strong}$.

the time ($\gamma_{\text{Usr}} = 0.2$) and is strongly polarized ($\delta_{\text{Usr}} = 0.83$) but would only reach 40% if instead they read diverse sources ($\delta_{\text{Usr}} = 0.33$) or 25% if instead they only visit news websites less than 1% of the time ($\gamma_{\text{Usr}} \leq 0.01$).

Reliability of the confidence score. We have shown that while our attack gives good results overall, it is particularly effective for certain users, in particular users with a high degree of polarization and relevant interest.

Figure 8 shows that our attack's confidence score is well calibrated: for both adversaries, use cases, and number of observations. This makes the attack a concern in practice as it allows an adversary to estimate the probability of the attack to be successful against a specific target user Usr by looking only at Usr's obfuscated objects.

5 Experiments on Twitter data

We now simulate the attack in the emojis setting using data collected from Twitter. Our experiments serve two purposes: first, they validate the hierarchical model $\mathcal{M}$ (including the user representation $\overline{\Phi}$) in practice; second, they prove that the attack *can* be very effective on real-world users (see the discussion in section 6).

Dataset. We use the dataset of tweets collected by Robertson et al. [29], which contains about 18M tweets from 42K Twitter users, and derive a dataset $\mathcal{D}$ containing only the emojis sent by each users. We then apply a random 80-20 split to $\mathcal{D}$ and obtain: $\mathcal{D}_{att}$, containing the users who used at least one emoji supporting skin tones, on which we simulate the attack; and $\mathcal{D}_{ext}$, containing the external population that Adv$_{strong}$ uses to compute the emojis estimated popularity $\hat{p}_\Omega$. We simulate the BPIA attack instantiating the Pool Inference Game on each user in $\mathcal{D}_{att}$, treating each emoji as an original object to which we apply CMS. The full details on how we produce the datasets and run the attack are given in Appendix A.2.

Results for Adv$_{strong}$. We instantiate the game only for Adv$_{strong}$, since our experiments on synthetic data already showed that, in most cases, Adv$_{weak}$ is not very effective in the emojis setting. Figure 9 (left) shows that the attack is overall very effective on the users in $\mathcal{D}_{att}$. For any number of observations, the precision is > 0.5 (2.5 times better than the baseline) when the attack is run on all the users in $\mathcal{D}_{att}$. After only $n = 7$ observations, the precision on the top 20% of the users (the 20% of the users with the highest confidence score) is above 0.61, going up to 0.825 after 180 observations. The attack however struggles to reach perfect precision: with $n = 180$, to achieve a precision of 0.95 the adversary needs to restrict the attack to the top 10% of the users. This is mostly because, contrary to the synthetic data, $\mathcal{D}_{att}$ contains very few users who have both high γ_{Usr} and δ_{Usr} (see Appendix A.2).

Figure 10 shows the precision of BPIA depending on γ_{Usr} and δ_{Usr} when making a guess on every user. These results are mostly consistent with those computed using the synthetic data (Figure 4). As expected, the attack is not very effective on users with low γ_{Usr} and δ_{Usr}, but works remarkably well on high-polarization users who have medium to high relevant interest. For example, after 90 observations, the attack achieves 0.82 to 0.9 precision on the users with polarization over 0.8 and relevant interest at least 0.4. Overall, these results validate the applicability of BPIA's model $\mathcal{M}$.

Reliability of the confidence score. Figure 9 (right) confirms that the confidence score computed by BPIA can be used to accurately estimate the probability that the estimated preferred pool is correct. This validates the fact that BPIA can be used to distinguish and target the most vulnerable users.

6 Discussion

In this paper we propose pool inference, a new attack model that quantifies some practical privacy risks that may affect implementations of local differential privacy mechanisms. We formalize the attack model as a game and propose a Bayesian pool inference attack (BPIA) that applies to any local differ-

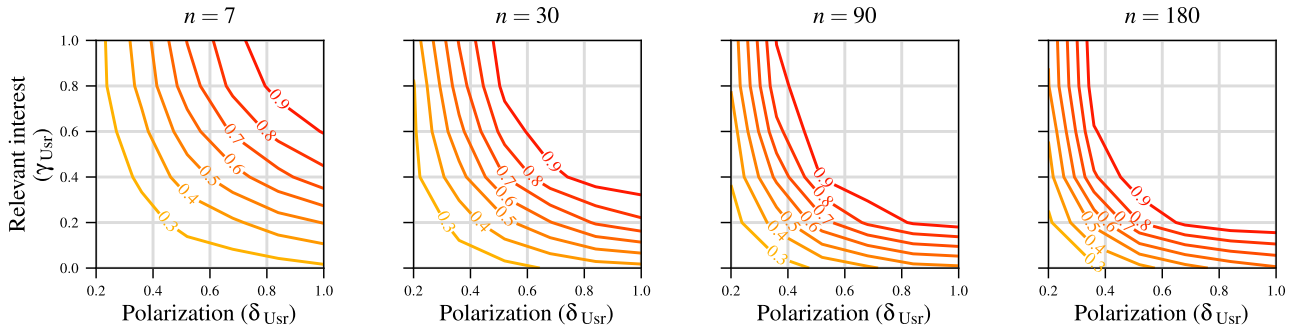

Figure 7: Precision depending on γ_{Usr} and δ_{Usr} for Adv$_{strong}$ in the web domains setting when the attack always makes a guess.

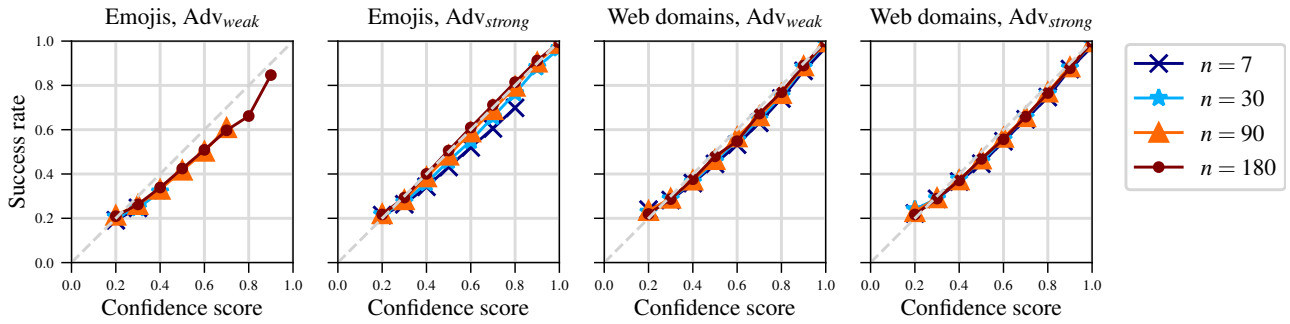

Figure 8: Success rate as a function of the confidence score for both Adv$_{weak}$ and Adv$_{strong}$ and in both the emojis and web domains settings. The confidence score computed by the attack accurately estimates the probability that the attack is correct.

ential privacy mechanism that processes each object independently. We simulate BPIA against Apple's CMS mechanism for emojis and web domains and study its effectiveness in different scenarios. We show that the attack can successfully allow an adversary to infer sensitive properties of a user's behavior. We further show that BPIA works best on users who are more polarized — and may hence require the strongest privacy protections. To the best of our knowledge, this is the first attack designed against a real-world implementation of local differential privacy. Taken together, our results show that the BPIA attack is a practical threat for Apple's devices, where CMS is implemented with large ε parameters and without limiting the cumulative privacy loss after multiple observations.

Previous criticism to Apple's implementation. In September 2017, Tang et al. reverse engineered and analyzed Apple's implementation, for which no technical description was yet available. In particular, they found that the choice of the privacy loss ε was not in line with what is deemed mathematically secure [33]. Tang et al. provided a detailed analysis of Apple's system, but did not propose attacks showing how the weakness of the theoretical guarantees could be exploited in practice. Apple disputed the findings by Tang et al., claiming that the system provides far more protection than acknowledged by the researchers [18]. Our experimental evaluation shows that, with Apple's choice of parameters, our BPIA attack could potentially lead to the disclosure of a user's preference for news websites or emoji skin tone. According to their white paper, Apple discards any user identifier when obfuscated objects are ingested by their servers, making it impossible to later link multiple observations from the same users [4]. While this would limit the attack to a single observation, this is an organizational measure that relies on trust, which is what local differential privacy is designed to avoid [13, 21, 39] (see also the discussion on mitigation strategies below).

Representativeness of the experiments. In this paper we validate our attack using both synthetically generated data and Twitter data. The goal of our experiments is to study how the attack performs in several scenarios and to validate the user model $\mathcal{M}$ showing that the attack works on a significant number of real-world users. On the other hand, the aim of our experiments is not to measure the fraction of users in the population who are vulnerable. Firstly, while we use Twitter data as a representation of users' usage of emojis, we do not have access to datasets that record such usage across apps or web browsing data. Secondly, our work focuses only on two attack goals (i.e. sets of pools): determining the preferred emoji skin tone and the political orientation of the most visited news website. As mentioned in section 3, the adversary can run BPIA as many times as they want *on the same data* using any pools they wish. Users who are not vulnerable to the attack with certain pools might be vulnerable with respect to another set of pools. Moreover, the confidence score can be used to reliably estimate which inferences are likely to be

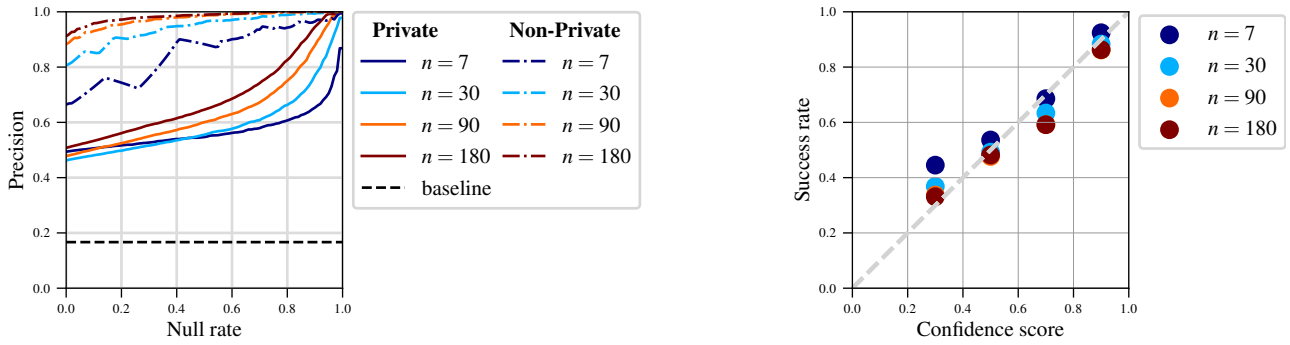

Figure 9: Precision-null rate curves (*left*) and success rate depending on the confidence score (*right*) for Adv_{strong} on Twitter data.

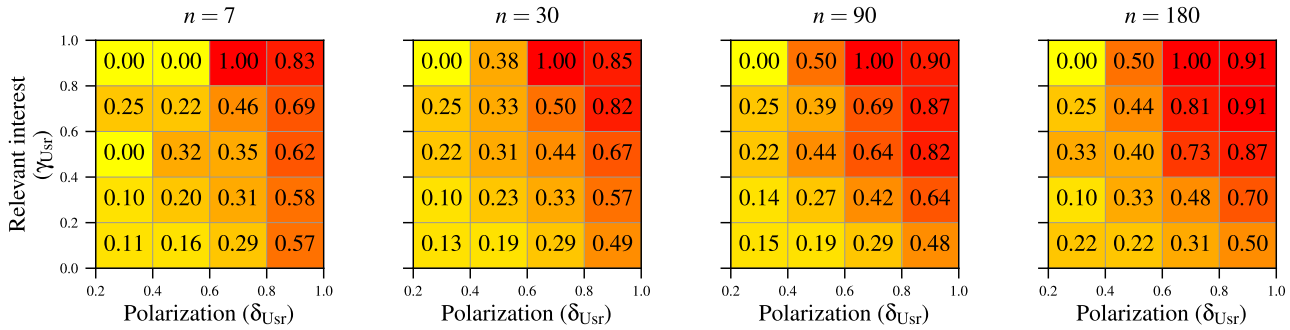

Figure 10: Precision depending on γ_{Usr} and δ_{Usr} for Adv_{strong} on Twitter data when the attack always makes a guess.

correct. Future work may select other privacy-sensitive pools and use our attack to assess different privacy risks.

Using auxiliary population-level knowledge to estimate the object popularity. Our experiments with Adv_{strong} show that an adversary with access to auxiliary information on the overall popularity of objects among the population may be much more effective. The adversary may obtain access to such auxiliary information from a variety of sources, such as social media or studies that report summary statistics on popularity of emojis aggregated over many users. Furthermore, they can be typically estimated by the data curator. In fact, CMS is designed precisely with this scope in mind: estimating the popularity of objects across many users. Hence, if the adversary is the curator themselves, users' privacy is even more at risk.[6] In particular, the method used to estimate the popularity (see section 3) does not require that the adversary knows which objects are collected from which user. Adv_{strong} could be a curator who has never acted maliciously before, always discarding the identifiers that would allow to link objects coming from the same user, but who at some point decides to keep together the observations from the same target user.

[6]We note that assuming that the curator is also the adversary reflects the standard attack model applied to local differential privacy. We believe an external adversary to be less realistic in Apple's case as the obfuscated records are transmitted from the device to Apple through an encrypted connection.

Extending the attack to other mechanisms. While in this paper we focus on the CMS mechanism, our BPIA attack can be used against any local differential privacy mechanisms where $\Pr_{\mathcal{A}}[\widetilde{x}_t \mid z]$ can be computed analytically or estimated empirically. In the extended paper we show how to adapt the attack to run against HCMS, another mechanism proposed and deployed by Apple to identify websites that cause high usage of hardware resources (CPU and memory) [4]. HCMS is similar to CMS, but uses the Hadamard transform to reduce the size of obfuscated objects to a single bit. Despite this, the way to compute $\Pr_{\text{HCMS}}[\widetilde{x}_t \mid z]$ is similar to the one for CMS (see extended paper).

Solutions and mitigation strategies. There are several possible solutions to protect against BPIA, or at least mitigate it. However, to our knowledge, these all come at a cost in terms of utility, or require significant resources to be deployed.

First: Using a smaller ε and limiting the total number of observations per user. We show in Appendix A.5 that using a smaller value of ε reduces the effectiveness of the attack, but it also has a direct impact on utility. Similarly, our results in sections 4 and 5 show that BPIA is less effective when the number of observed obfuscated objects from the target user is lower, but reducing the total number of observations affects utility as well (see Appendix A.5). Moreover, limiting the number of observations might make it impossible to learn how users' preferences evolve over time.

Second: Using a local differential privacy mechanism that addresses the privacy loss over multiple observations. These typically use some form of heuristic memoization — such as Google's RAPPOR [15] — or techniques to reduce the number of observations that are collected [21]. These may offer a better (theoretical) privacy-utility tradeoff when the population-level distribution that needs to be estimated over time does not change frequently. Extending the pool inference attack model and BPIA to these mechanisms could be used to measure this tradeoff in a practical setting and compare it to the tradeoff provided by CMS.

Third: Adopting a different privacy model. In recent work, researchers have proposed techniques that typically go under the name of *shuffled differential privacy* [5, 7, 9, 10, 16]. This is a hybrid privacy model where the obfuscated objects are routed through an intermediary (the *shuffler*) that in turn sends them to the curator. The role of the intermediary is to shuffle the obfuscated objects to anonymize them and make them unlinkable. Shuffled differential privacy has been deployed by Apple and Google in the context of the Exposure Notification System for COVID-19 [3]. While adopting this model for CMS would protect against BPIA, it effectively moves the requirement of users' trust from the curator to the shuffler: if the two collude, the curator would be able to link the objects again [7, 10]. The technical guarantees of the model would be greatly enhanced by using a mix network as the shuffler, but these are extremely hard to deploy in practice [34]. Nevertheless, we believe that the shuffled model is a promising avenue to apply local differential privacy in practice, and we hope this paper will provide evidence of the need for its further development and adoption.

Source code. The code to reproduce the results is available at https://github.com/computationalprivacy/pool-inference.

7 Related work

Our work is part of the line of research studying the guarantees of differential privacy against specific attacks. Previous research has studied the privacy protections of specific differential privacy mechanisms with respect to attacks that simulate real-world adversaries, but this line of work has so far focused on mechanisms for *central* differential privacy — the main variant of differential privacy which assumes a trusted curator and one or more untrusted analysts. Examples include attacks against differential privacy mechanisms to release aggregate location time-series [24–26], synthetic data [30], and machine learning models [20, 23, 28].

To the best of our knowledge, only two papers have empirically investigated the privacy guarantees of a *local* differential privacy mechanism. Pyrgelis et al. [25] propose several attacks on aggregated location data that aim to recover individual users' locations or mobility patterns. They evaluate their attacks against SpotMe [27], a mechanism to obfuscate location data that satisfies local differential privacy [38]. Pyrgelis et al.'s work however considers a different adversarial setting than ours: their attacks apply to location time-series obtained by aggregating the obfuscated objects over multiple users, while in our pool inference attack the adversary has access to the individual obfuscated objects. Our attack could be simply adapted to the SpotMe mechanism[7] in order to infer the user's preferred pool of locations among some pools of interest — an interesting application that we leave to future work.

Chatzikokolakis et al. [8] propose the Bayes security measure, a general metric that quantifies the expected advantage over random guessing of an adversary that observes the output of a mechanism. They then apply their metric to randomized response [37] — a simple local differential privacy mechanism originally conceived to protect privacy in survey responses. They apply randomized response to the US 1990 Census dataset and find that it gives good protection even for values of ε as high as 4.8. However, their evaluation focuses on object indistinguishability — i.e. it considers an adversary that collects an obfuscated object and whose goal is to infer the original object. This is a significantly harder objective compared to pool inference and, in fact, CMS's use of hash functions prevents this even for arbitrarily large values of ε. Our work shows that enforcing object indistinguishability is not enough to protect privacy in a practical setting where the adversary has access to multiple obfuscated objects from the same user.

8 Conclusion

Apple's implementation of local differential privacy in iOS and Mac OS devices has been presented as a "technology to help discover the usage patterns of a large number of users without compromising individual privacy" [17]. Although researchers have criticized Apple's choice of ε and unlimited theoretical privacy loss over multiple observations, to our knowledge no practical attacks have been proposed against the mechanisms deployed by Apple. In this paper, we proposed a Bayesian pool inference attack and we empirically evaluated it on Apple's Count Mean Sketch mechanism as configured on Apple's devices. We showed that, especially on the most vulnerable users, the attack could be used to successfully infer (1) the emoji skin tone that the user selects more frequently and (2) the political orientation of the news websites that the user is more likely to visit. Finally, we discussed how the technical privacy guarantees against our attack could be improved, and indicated where further research is necessary to evaluate the privacy/utility tradeoff of these mitigation strategies.

[7] The SpotMe mechanism is quite similar to CMS, but without hashing. Hence, the probabilities $\Pr_{\mathcal{A}}[\tilde{x} \mid z]$ that are used by the attack (eq. 4) can be computed similarly to the ones for CMS.

References

[1] AllSides. Media Bias Chart™ Version 4. [link], January 2021.

[2] Apple. Differential Privacy Overview. [link].

[3] Apple and Google. Exposure Notification Privacy-preserving Analytics (ENPA). April 2021. [link].

[4] Apple, Differential Privacy Team. Learning with Privacy at Scale. December 2017. [link].

[5] Borja Balle, James Bell, Adrià Gascón, and Kobbi Nissim. Private Summation in the Multi-Message Shuffle Model. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, CCS '20, pages 657–676, New York, NY, USA, October 2020. Association for Computing Machinery. [link].

[6] Heinz H. Bauschke and Jonathan M. Borwein. On Projection Algorithms for Solving Convex Feasibility Problems. *SIAM Review*, 38(3):367–426, 1996. [link].

[7] Andrea Bittau, Úlfar Erlingsson, Petros Maniatis, Ilya Mironov, Ananth Raghunathan, David Lie, Mitch Rudominer, Ushasree Kode, Julien Tinnes, and Bernhard Seefeld. Prochlo: Strong Privacy for Analytics in the Crowd. In *Proceedings of the Symposium on Operating Systems Principles (SOSP)*, pages 441–459, 2017. [link].

[8] Konstantinos Chatzikokolakis, Giovanni Cherubin, Catuscia Palamidessi, and Carmela Troncoso. The Bayes Security Measure. *arXiv:2011.03396 [cs]*, November 2020. [link].

[9] Albert Cheu. Differential Privacy in the Shuffle Model: A Survey of Separations. *arXiv:2107.11839 [cs]*, July 2021. [link].

[10] Albert Cheu, Adam Smith, Jonathan Ullman, David Zeber, and Maxim Zhilyaev. Distributed Differential Privacy via Shuffling. *arXiv:1808.01394 [cs]*, 11476:375–403, 2019. [link].

[11] Bolin Ding, Janardhan Kulkarni, and Sergey Yekhanin. Collecting telemetry data privately. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pages 3574–3583, Red Hook, NY, USA, December 2017. Curran Associates Inc.

[12] Cynthia Dwork, Nitin Kohli, and Deirdre Mulligan. Differential Privacy in Practice: Expose your Epsilons! *Journal of Privacy and Confidentiality*, 9(2), October 2019. [link].

[13] Cynthia Dwork and Aaron Roth. The Algorithmic Foundations of Differential Privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3-4):211–407, 2013. [link].

[14] Cynthia Dwork and Adam Smith. Differential Privacy for Statistics: What we Know and What we Want to Learn. *Journal of Privacy and Confidentiality*, 1(2), April 2010. [link].

[15] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. RAPPOR: Randomized Aggregatable Privacy-Preserving Ordinal Response. pages 1054–1067. ACM Press, 2014. [link].

[16] Vitaly Feldman, Audra McMillan, and Kunal Talwar. Hiding Among the Clones: A Simple and Nearly Optimal Analysis of Privacy Amplification by Shuffling. *Theory and Practice of Differential Privacy workshop (ICML 2021)*, September 2021. [link].

[17] Andy Greenberg. Apple's 'Differential Privacy' Is About Collecting Your Data—But Not Your Data. *Wired*. [link].

[18] Andy Greenberg. How One of Apple's Key Privacy Safeguards Falls Short. *Wired*. [link].

[19] Shanto Iyengar and Kyu S Hahn. Red Media, Blue Media: Evidence of Ideological Selectivity in Media Use. *Journal of Communication*, 59(1):19–39, 2009. [link].

[20] Bargav Jayaraman and David Evans. Evaluating Differentially Private Machine Learning in Practice. In *28th {USENIX} Security Symposium ({USENIX} Security 19)*, pages 1895–1912, 2019. [link].

[21] Matthew Joseph, Aaron Roth, Jonathan Ullman, and Bo Waggoner. Local Differential Privacy for Evolving Data. *arXiv:1802.07128 [cs]*, February 2018. [link].

[22] S. P. Kasiviswanathan, H. K. Lee, K. Nissim, S. Raskhodnikova, and A. Smith. What Can We Learn Privately? In *2008 49th Annual IEEE Symposium on Foundations of Computer Science*, pages 531–540, October 2008. [link].

[23] Cheolhee Park, Dowon Hong, and Changho Seo. An Attack-Based Evaluation Method for Differentially Private Learning Against Model Inversion Attack. *IEEE Access*, 7:124988–124999, 2019. [link].

[24] Apostolos Pyrgelis, Carmela Troncoso, and Emiliano De Cristofaro. Knock Knock, Who's There? Membership Inference on Aggregate Location Data. In *Proceedings 2018 Network and Distributed System Security Symposium*, San Diego, CA, 2018. Internet Society. [link].

[25] Apostolos Pyrgelis, Carmela Troncoso, and Emiliano De Cristofaro. What Does The Crowd Say About You? Evaluating Aggregation-based Location Privacy. *Proceedings on Privacy Enhancing Technologies*, 2017(4):156–176, October 2017. [link].

[26] Apostolos Pyrgelis, Carmela Troncoso, and Emiliano De Cristofaro. Measuring Membership Privacy on Aggregate Location Time-Series. *Proc. ACM Meas. Anal. Comput. Syst.*, 4(2):36:1–36:28, June 2020. [link].

[27] Daniele Quercia, Ilias Leontiadis, Liam McNamara, Cecilia Mascolo, and Jon Crowcroft. SpotME If You Can: Randomized Responses for Location Obfuscation on Mobile Phones. In *2011 31st International Conference on Distributed Computing Systems*, pages 363–372, June 2011. [link].

[28] Atiqur Rahman, Tanzila Rahman, Robert Laganiere, Noman Mohammed, and Yang Wang. Membership Inference Attack against Differentially Private Deep Learning Model. *Trans. Data Privacy*, 11(1):19, 2018. [link].

[29] Alexander Robertson, Walid Magdy, and Sharon Goldwater. Emoji Skin Tone Modifiers: Analyzing Variation in Usage on Social Media. *Trans. Soc. Comput.*, 3(2):11:1–11:25, April 2020. [link].

[30] Theresa Stadler, Bristena Oprisanu, and Carmela Troncoso. Synthetic Data – Anonymisation Groundhog Day. *arXiv:2011.07018 [cs]*, June 2021. [link].

[31] Statista. Smartphones replacement cycle in the US 2014-2025. [link].

[32] Natalie Jomini Stroud. *Niche News: The Politics of News Choice*. Oxford University Press, 2011. [link].

[33] Jun Tang, Aleksandra Korolova, Xiaolong Bai, Xueqiang Wang, and Xiaofeng Wang. Privacy Loss in Apple's Implementation of Differential Privacy on MacOS 10.12. *arXiv:1709.02753 [cs]*, September 2017. [link].

[34] The DP3T Consortium. DESIRE - A Practical Assessment. May 2020. [link].

[35] Unicode. Emoji List, v11.0. [link].

[36] Salil P. Vadhan. Pseudorandomness. *TCS*, 7(1–3):1–336, December 2012. [link].

[37] Stanley L. Warner. Randomized Response: A Survey Technique for Eliminating Evasive Answer Bias. *Journal of the American Statistical Association*, 60(309):63–69, 1965. [link].

[38] Atsushi Waseda and Ryo Nojima. Analyzing Randomized Response Mechanisms Under Differential Privacy. In Matt Bishop and Anderson C A Nascimento, editors, *Information Security*, Lecture Notes in Computer Science, pages 271–282, Cham, 2016. Springer International Publishing. [link].

[39] Xingxing Xiong, Shubo Liu, Dan Li, Zhaohui Cai, and Xiaoguang Niu. A Comprehensive Survey on Local Differential Privacy. *Security and Communication Networks*, 2020:e8829523, October 2020. [link].

A Appendix

A.1 Details of CMS

The CMS algorithm as proposed by Apple [4] is defined by the procedure CMS. The set $\mathcal{H} = \{h_1, \ldots, h_{|\mathcal{H}|}\}$ is a collection

Procedure CMS(x; $\varepsilon, m, \mathcal{H}$)

Input: original object x; parameters ε, m, $\mathcal{H}$
Output: obfuscated object $\tilde{x}$, index j

sample j uniformly at random from $\{1, \ldots, |\mathcal{H}|\}$
$v \leftarrow \{0\}^m$
$v[h_j(x)] \leftarrow 1$
sample $b \in \{0,1\}^m$, with $\{b[i]\}_{i=1}^n$ iid and $\Pr[b[i] = 1] = 1/(1+e^{\varepsilon/2})$
for $i \leftarrow 0$ **to** m **do**
  if $b[i] = 1$ **then**
    flip $v[i]$
$\tilde{v} \leftarrow v$
return $(\tilde{v}, j)$

of hash functions, where each $h \in \mathcal{H}$ maps every element of Ω to an integer between 0 and $m-1$. The collection $\mathcal{H}$ is sampled uniformly at random from a family of three-wise independent hash functions [36]. For any finite sets A, B, a family $\mathcal{F}$ of functions $A \to B$ is three-wise independent if, for any mutually distinct $a_1, a_2, a_3 \in A$ and for any $b_1, b_2, b_3 \in B$, we have that $\Pr[f(a_1) = b_1, f(a_2) = b_2, f(a_3) = b_3] = 1/|B|^3$, where the probability is computed over the uniformly random selection of the function $f \in \mathcal{F}$. This property is irrelevant for the differential privacy guarantees, but it contributes to the utility achieved by aggregating CMS objects to estimate frequency histograms [4]. Since Apple does not specify the family used in their implementation of CMS, we generate $|\mathcal{H}|$ fully random hash functions by selecting uniformly at random the value of $h(x)$ for any $h \in \mathcal{H}$ and $x \in \Omega$. This method is not space-efficient, as it requires to store the full description of all functions in $\mathcal{H}$, but it has the advantage of removing any possible source of regularity that might artificially improve the accuracy of our attack. In our implementation, we follow Apple's choice of parameters for the number of hash functions and set $|\mathcal{H}| = 65536$ for all the experiments.

A.2 Details of the experiments on Twitter data

Dataset. We use the dataset of tweets collected by Robertson et al. [29], which consists of about 1.8M tweets collected in 2018 from 42K Twitter users. We consider only users who used emojis at least 10 times across all tweets (approx. 26K).

We produce a dataset $\mathcal{D}$ by including, for each user, the first 180 emojis used by the user. This sequence of emojis represents the user's original objects. We then apply a random 80-20 split of $\mathcal{D}$ to obtain $\mathcal{D}_{80}$ and $\mathcal{D}_{20}$. We process $\mathcal{D}_{80}$ to obtain $\mathcal{D}_{att}$ — containing the users on which we will simulate the attack by Adv_{strong} — and $\mathcal{D}_{20}$ to obtain $\mathcal{D}_{ext}$, containing the external population that Adv_{strong} will use to estimate the emojis popularity $\hat{p}_\Omega$:

$\mathcal{D}_{att}$ — To be able to run the attack with up to $n = 180$ observations for each user, if the user has less than 180 objects, we repeat the user's observations in chronological order until we reach 180 objects. For example, if a user originally has only 53 original objects $x_1,\dots,x_{53}$, we produce 127 new objects such that $x_{54} = x_1, x_{55} = x_2, \dots, x_{180} = x_{21}$. Finally, we keep only the users who have at least one emoji supporting skin tones, since they are the users on which the attack can be applied ($\gamma_{\text{Usr}} > 0$). After this, $\mathcal{D}_{att}$ contains ≈18K users with 180 emojis each.

$\mathcal{D}_{ext}$ — Similarly, to obtain $\mathcal{D}_{ext}$ we augment the data in D_{20}. This is necessary because $\mathcal{D}_{20}$ contains only 540K original objects in total, while in the experiments with synthetic data the external dataset contains $N = 10^6$ objects (see section 4). For a fair comparison, we duplicate the original objects in $\mathcal{D}_{20}$ so that the size of $\mathcal{D}_{ext}$ is $2 \times 540\text{K} \approx 10^6$. We note that this does not affect the validity of our experiments: in a realistic setting, the curator (acting as Adv_{strong}) would likely have access to millions of obfuscated objects (see section 6).

Relevant interest and polarization. Using the so obtained dataset $\mathcal{D}_{att}$, we compute each user's preferred pool P_{Usr}, relevant interest γ_{Usr}, and polarization δ_{Usr} on the full sequence of 180 original objects. We recall that these parameters do not depend on the attack and are not known to Adv_{strong}, but they are useful to describe how vulnerable Usr is. Given Usr's sequence of original objects $\underline{x} = x_1,\dots,x_{180}$ and a subset of the universe $S \subseteq \Omega$, with an abuse of language we denote $|\underline{x} \cap S| \stackrel{\text{def}}{=} |\{t : x_t \in S\}|$. Then we compute:

$$P_{\text{Usr}} = \arg\max_{P_i} |\underline{x} \cap P_i|$$

$$\gamma_{\text{Usr}} = \frac{1}{180}|\underline{x} \cap \bigcup_{i=1}^{k} P_i| \quad \text{and} \quad \delta_{\text{Usr}} = \frac{1}{180\gamma_{\text{Usr}}}|\underline{x} \cap P_{\text{Usr}}|$$

Figure 11 shows how the relevant interest γ_{Usr} and the polarization δ_{Usr} are distributed across users in $\mathcal{D}_{att}$. In particular, the results show that relevant interest is overall not very high, with 71.8% of the users having $\gamma_{\text{Usr}} < 0.2$, meaning that they select emojis supporting skin tone about 20% of the times or less. On the other hand, most of them have extremely high polarization: $\delta_{\text{Usr}} > 0.8$ for 75.2% of the users.

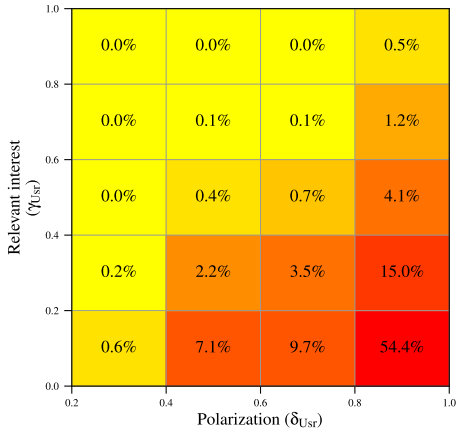

Figure 11: Joint distribution of $(\gamma_{\text{Usr}}, \delta_{\text{Usr}})$ in the dataset $\mathcal{D}_{att}$.

Running the attack. We instantiate the Pool Inference Game on each of the 18K users in $\mathcal{D}_{att}$, independently. Since we are now using real data, we do not need to define Usr's behavior Φ_{Usr} — instead, we select the the first n emojis used by Usr and run CMS (independently) on them to obtain the n obfuscated records. We note that this choice underestimates the success rate of our attack compared to a random sampling of n objects, since it might be that Usr's long-term preferred pool is not the same as in the first n observations on which the attack is run.

A.3 Effect of entropy

For the experiments in section 4, we have assumed that the object popularity follows a Zipfian distribution (with parameter 1.2) within each pool. While the exact shape of the object popularity is not particularly important, for BPIA to be effective it is important that the pools of interest do not contain a large number of objects with non-negligible popularity. However, this requirement applies only when the pools of interest are large (for example, both Adv_{weak} and Adv_{strong} achieve good effectiveness in the news case as the pools are small). More precisely, the effectiveness of the attack is lower when the *entropy* of the popularity within pools is higher. Intuitively, this is because large pools "contain too much noise", but the noise can be ignored if most of the objects contained in them can be (correctly) ignored by BPIA — that is, when most of these objects have low *within-pool* probability of being picked by Usr (and Adv knows that).

We now show some results that illustrate this fact more in detail. We consider the emojis setting and we run the attack using six pools of equal size, again assuming that the popularity within each pool (and in the neutral pool) is distributed according to a Zipfian distribution. We vary both the size of the pools ($|P| = 10, 50, 200, 400$) and the Zipfian distribution parameter ($s = 0, 0.5, 1, 2, 4$), for a total of 20 scenarios. Both these values affect the entropy of the popularity within each

$\|P\|$	$s=0$	$s=0.5$	$s=1$	$s=2$	$s=4$
10	3.32	3.22	2.88	1.78	0.48
50	5.64	5.44	4.61	2.19	0.48
200	7.64	7.36	5.99	2.31	0.48
400	8.64	8.33	6.64	2.33	0.48

Table 4: Entropy of the popularity within each pool of interest when the pool has size $|P|$ and the distribution within the pool follows Zipfian distribution with parameter s.

pool, as illustrated in Table 4. We note that the parameter $s=0$ yields a uniform distribution. For simplicity, we simulate an adversary that knows the true popularity p_Ω and uses it in BPIA, i.e. Adv sets $\hat{p}_\Omega = p_\Omega$. We run BPIA against 1000 users in each scenario, again using eq. 6 to define the user behavior.

Table 5 shows that the AUC-PN of the attack is negatively affected by the entropy of the popularity within pools. For any size of the pools, larger values of s lead to lower entropy (Table 4), which results in better effectiveness of the attack. When the entropy is very low (e.g. for $s=4$), the attack is very effective even when the pools contain 400 objects (AUC-PN $= 0.98$). On the other hand, when $s=0$ — so that the popularity is the uniform distribution —, the entropy is maximal, but the AUC-PN is significantly affected only for larger pools.

$\|P\|$	$s=0$	$s=0.5$	$s=1$	$s=2$	$s=4$
10	0.89	0.90	0.92	0.96	0.98
50	0.69	0.73	0.85	0.97	0.98
200	0.43	0.52	0.80	0.96	0.98
400	0.35	0.42	0.78	0.96	0.98

Table 5: AUC-PN of the attack for $n=180$ observations depending on the size and distribution within the pools.

A.4 Size of the universe

Apple's implementation of CMS for the web domains setting uses a universe Ω containing 250,000 objects. In order to reduce the computational time required to run BPIA on 150,000 users, for the experiments in section 4 we use a smaller universe containing 2,000 objects. We now show that the size of the universe (and, in particular, of the neutral pool) has close to no impact on the effectiveness of the attack.

We run BPIA in the exact same scenario, changing only the value of $|\Omega|$ — in particular, we keep the same pools of size 14, 13, 13, 10, and 10. We use universe sizes $|\Omega| = 1000, 10000, 100000, 250000$. These values are used both in the simulation of the users and in the simulation of the adversary. For each size, we run the simulation on 5,000 users — which leads to a sufficiently accurate estimate of the AUC-PN.

$\|\Omega\|$	$n=7$	$n=30$	$n=90$	$n=180$
1000	0.72	0.90	0.96	0.97
10000	0.71	0.88	0.95	0.97
100000	0.72	0.89	0.95	0.98
250000	0.72	0.89	0.95	0.97

Table 6: Effectiveness of the attack for Adv_{weak} in the web domains setting depending on the size of the universe. The four columns on the right report the AUC-PN.

Table 6 reports the AUC-PN values for Adv_{weak}, which are almost identical for all universe sizes and across all number of observations. We omit the results for Adv_{strong} as they are very similar. Intuitively, the size of the universe is mostly irrelevant to BPIA's effectiveness because the only relevant bits in the obfuscated objects are the ones associated (by the randomly selected hash function) with an original object that belongs to a pool of interest. Hence, objects from the neutral pool are relevant only if they yield a collision with any of these. Since the hash function is randomly selected, the collisions tend to distribute evenly inside the pools across multiple observations.

A.5 Effect of ε

The parameter ε controls the level of the noise in CMS, i.e. the probability that each bit is flipped. The value of ε hence affects the privacy guarantees and, in turn, the effectiveness of BPIA. To quantify this impact, we simulate the attack in the web domains setting, changing only the value of ε. We then show how using smaller ε affects utility.

Effect on the attack. Table 8 reports the results for Adv_{weak}. As expected, the value of ε heavily impacts the AUC-PN. Interestingly, for $\varepsilon = 0.1$ the AUC-PN does not improve over the baseline of 0.2 even after $n=180$ observations — when the total (theoretical) privacy loss is $\varepsilon_{\text{tot}} = 180 \times 0.1 = 18$. This is remarkable because when $\varepsilon_{\text{tot}} = 18$, there are virtually no theoretical privacy guarantees, and yet such ε is sufficient to fully protect against BPIA in the web domains setting. This highlights the importance of quantifying the privacy guarantees of differential privacy mechanisms against realistic attack models and scenarios when the theoretical privacy loss is large [12].

Effect on utility. The results in Table 8 show that BPIA could be made significantly less effective — at least in our setting — by using $\varepsilon \leq 1$ in CMS. We now show that this would however seriously impact the utility of the CMS-obfuscated objects that are collected and aggregated by the curator. To show this, we measure the accuracy of the object popularity that

	$\mathrm{MAE}(\dot{p}_\Omega, p_\Omega)$				$\mathrm{MAPE}_{80}(\dot{p}_\Omega, p_\Omega)$			
ε	$Z = 10^6$	$Z = 10^7$	$Z = 10^8$	$Z = 10^9$	$Z = 10^6$	$Z = 10^7$	$Z = 10^8$	$Z = 10^9$
0.01	0.163794	0.050822	0.015939	0.005230	184449.89%	55912.08%	17980.67%	5835.21%
0.10	0.016129	0.005109	0.001553	0.000493	18304.38%	5665.86%	1849.65%	561.66%
1	0.001572	0.000506	0.000156	0.000052	1659.39%	563.36%	174.64%	57.87%
4	0.000337	0.000111	0.000036	0.000016	376.18%	119.29%	40.34%	18.50%
8	0.000115	0.000038	0.000016	0.000013	126.35%	40.42%	18.55%	14.41%

Table 7: Utility achieved by the curator to estimate the popularity distribution p_Ω when CMS is run using privacy loss ε and Z CMS-obfuscated objects are collected.

ε	$n = 7$	$n = 30$	$n = 90$	$n = 180$
0.01	0.20	0.20	0.20	0.20
0.1	0.20	0.20	0.20	0.20
1	0.23	0.29	0.36	0.40
4	0.40	0.63	0.81	0.88
8	0.72	0.90	0.96	0.97

Table 8: Effectiveness of the attack for Adv_{weak} in the web domains setting depending on the value of ε. The four columns on the right report the AUC-PN.

would be estimated by the data curator, under the different ε values. Since this accuracy also heavily depends on the number of CMS-obfuscated objects that are collected, we show the results for different numbers of CMS-obfuscated objects $Z = 10^6, 10^7, 10^8, 10^9$. This can be interpreted as the total number of objects by all users. For example, if users send 100 objects each on average, then 10^7 users are necessary to collect 10^9 objects.

For a given value of ε and Z, we draw Z original objects from the universe according to the true popularity p_Ω. We then apply CMS with the given ε to obtain Z obfuscated objects, and use Apple's algorithm to derive an estimation of the popularity $\dot{p}_\Omega$. Finally, we measure the accuracy of $\dot{p}_\Omega$ by using two metrics: the mean absolute error $\mathrm{MAE}(\dot{p}_\Omega, p_\Omega)$ and the mean absolute percentage error computed on the top 80% objects[8] $\mathrm{MAPE}_{80}(\dot{p}_\Omega, p_\Omega)$.

The results in Table 7 show that the accuracy of $\dot{p}_\Omega$ are greatly affected by ε, with both the MAE and the MAPE increasing about linearly with ε for any value of Z. In particular, the results show that $\dot{p}_\Omega$ starts reaching an acceptable accuracy ($\mathrm{MAPE}_{80} = 18.55\%$) only when $\varepsilon = 8$ and 10^8 CMS objects are collected. Using $\varepsilon = 1$ for the same number of objects would result in a much larger error ($\mathrm{MAPE}_{80} = 174.64\%$). Even when the curator collects 10^9 CMS-obfuscated objects using $\varepsilon = 1$ still results in much lower utility: the MAPE is 57.87%, about three times as much as for $\varepsilon = 8$ and $Z = 10^8$. While an extensive analysis of the utility of CMS for different use cases is beyond the scope of this paper, these results suggest that mitigating our attack by using a smaller ε parameter in CMS would likely affect utility to an unacceptable level.

[8] Since the MAPE is very sensitive to error on objects with very small probabilities, ignoring the 20% of objects with the lowest probability gives a clearer measure of utility. In practical deployments of local differential privacy, determining the exact popularity of unpopular objects is likely not necessary.

A.6 Improving the attack with other types of auxiliary information.

In the experiments with synthetic users we decide to quantify the effectiveness of the attack by randomizing the user's preferred pool and using uniform priors in the attack for ι, γ and δ. In practice the adversary may use additional types of auxiliary information to estimate distributions that would likely improve the effectiveness of the attack in practice. For example, the adversary might know that the user often uses skin-toned emojis, or that they are likely to use almost always the same skin tone. This information could be incorporated into $\mathcal{M}$ using *behavioral priors* p_γ and p_δ for γ and δ, respectively. In another instance, the adversary might know that the target user lives in a city where the majority of the population is white, and hence might expect the target user to be more likely to use light skin-toned emojis. This information can be used to build a *pool prior* p_ι for ι. We leave the study of these potential improvements for future work.