Towards a Practical Differentially Private Collaborative Phone Blacklisting System

Daniele Ucci
Department of Computer, Control, and Management Engineering "Antonio Ruberti", "La Sapienza" University of Rome
ucci@diag.uniroma1.it

Roberto Perdisci
University of Georgia
Georgia Institute of Technology
perdisci@{uga,gatech}.edu

Jaewoo Lee
University of Georgia
jaewoo.lee@uga.edu

Mustaque Ahamad
Georgia Institute of Technology
mustaq@cc.gatech.edu

ABSTRACT

Spam phone calls have been rapidly growing from nuisance to an increasingly effective scam delivery tool. To counter this increasingly successful attack vector, a number of commercial smartphone apps that promise to block spam phone calls have appeared on app stores, and are now used by hundreds of thousands or even millions of users. However, following a business model similar to some online social network services, these apps often collect call records or other potentially sensitive information from users' phones with little or no formal privacy guarantees.

In this paper, we study whether it is possible to build a practical collaborative phone blacklisting system that makes use of local differential privacy (LDP) mechanisms to provide clear privacy guarantees. We analyze the challenges and trade-offs related to using LDP, evaluate our LDP-based system on real-world user-reported call records collected by the FTC, and show that it is possible to learn a phone blacklist using a reasonable overall privacy budget and at the same time preserve users' privacy while maintaining utility for the learned blacklist.

CCS CONCEPTS

• **Security and privacy → Privacy-preserving protocols**.

KEYWORDS

Phone Spam, Collaborative Blacklisting, Local Differential Privacy

ACM Reference Format:
Daniele Ucci, Roberto Perdisci, Jaewoo Lee, and Mustaque Ahamad. 2020. Towards a Practical Differentially Private Collaborative Phone Blacklisting System. In *Annual Computer Security Applications Conference (ACSAC 2020), December 7–11, 2020, Austin, USA.* ACM, New York, NY, USA, 16 pages. https://doi.org/10.1145/3427228.3427239

ACSAC 2020, December 7–11, 2020, Austin, USA

ACM ISBN 978-1-4503-8858-0/20/12…$15.00
https://doi.org/10.1145/3427228.3427239

1 INTRODUCTION

Spam phone calls have been rapidly growing from nuisance to supporting well-coordinating fraudulent campaigns [4, 15, 18]. To counter this increasingly successful attack vector, federal agencies such as the US Federal Trade Commission (FTC) have been working with telephone carriers to design systems for blocking robocalls (i.e., automated calls) [12, 13]. At the same time, a number of smartphone apps that promise to block spam phone calls have appeared on app stores [25, 26, 31], and smartphone vendors, such as Google [16], are embedding some spam blocking functionalities into their default phone apps.

Currently, most spam blocking apps rely on caller ID blacklisting, whereby calls from phone numbers that are known to have been involved in spamming or numerous unwanted calls are blocked (either automatically, or upon explicit user consent). Recently, Pandit et al. [23] have studied how to learn such blacklists from a variety of data sources, including user complaints, phone honeypot call detail records (CDRs) and call recordings. Existing commercial apps, such as Youmail [31] and TouchPal [21], mostly base their blocking approach on user complaints. Other popular apps, such as TrueCaller [26], also use information collected from users' contact lists to distinguish between possible legitimate and unknown/unwanted calls[1]. However, in the recent past TrueCaller has experienced significant backlash due to privacy concerns related to the sharing of users' contact lists with a centralized service. Google recently implemented a built-in feature in Android phones to protect against possible spam calls. Nonetheless, Android phones may send information about received calls to Google without strong privacy guarantees [16].

While learning a blacklist from CDRs collected by phone honeypots [23] is a promising approach that poses little or no privacy risks, it suffers from some drawbacks. First, operating a phone honeypot is expensive, as thousands of phone numbers have to be acquired from telephone carriers. Furthermore, in [23] it has been reported that the spam calls targeting the honeypot were skewed towards business-oriented campaigns, likely because the honeypot numbers were mostly re-purposed business numbers (perhaps because re-purposing a user's number may pose some privacy risks, since others might still try to reach a specific person at that number). Conversely, leveraging user complaints also has some drawbacks. For instance, for a user to be able to complain or label a number (as in

[1] These behavior are inferred merely from publicly available information; further details on the inner-workings of commercial apps are difficult to obtain and their technical approach cannot be fully evaluated for comparison.

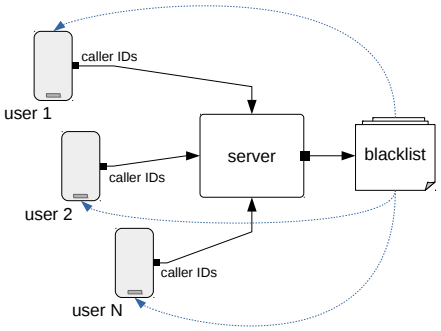

Figure 1: System overview. Caller IDs are collected with local differential privacy. After learning, blacklist updates are propagated back to users.

TouchPal [21]), the user has to answer to and identify the purpose of the call. However, only a fraction of users typically answers and listens to calls from unknown numbers (i.e., numbers not registered in the contact list). Furthermore, user-provided call labels are quite noisy and a relatively high number of complaints about the same number need to be observed, before being able to accurately label the source of the calls [21]. This may delay the insertion of a spam number into the blacklist, thus leaving open a time window for the spammers to succeed in their campaigns.

One possible solution would be to use an approach similar to the CDR-based blacklisting proposed in [23], while using real phone numbers as "live honeypots." In other words, if a smartphone app could leverage the call logs of real phone users without requiring the users to explicitly label the phone calls, this would provide a solution to the drawbacks mentioned above. Unfortunately, this may obviously pose serious privacy risks to users. For instance, knowing that a user received a phone call from a specific phone number related to a cancer treatment clinic may reveal that the user (or a close family member) is a cancer patient.

Research Question: *Can these privacy concerns be mitigated, and the users' call logs be collaboratively contributed to enable learning an accurate phone blacklist with strong privacy guarantees?*

To answer the above research question, in this paper we study whether it is possible to design a *practical* phone blacklisting system that leverages differential privacy [7] mechanisms to collaboratively learn effective anti-spam phone blacklists while providing strong privacy guarantees. Specifically, we leverage a state-of-the art *local differential privacy* (LDP) mechanisms for *generic heavy-hitter detection* that has been shown to work only in theory [3], and focus on adapting it to enable the implementation of a concrete privacy-preserving collaborative blacklist learning system that could be deployed on real smartphone devices. To the best of our knowledge, we are the first to study the application of local differential privacy to building blacklist-based defenses, and specifically towards defenses against telephony spam.

Figure 1 shows an overview of our system. Participating users install an app that can implement the following high-level functionalities (more details about the client app are provided in Section 5): when the user receives a call, the app will first check if the caller ID (i.e., the calling number) is in the users' contacts list; if yes, the caller ID is considered as *trusted* and ignored, otherwise the caller ID is considered to be *unknown* and buffered for reporting. Unknown caller IDs are then checked against a blacklist; if a match is found, the user can be alerted that the incoming phone call originates from a phone number known to have been involved in spamming activities, so that the user can decide whether to reject the call. At the end of a predefined time window (e.g., once daily), the app will report unknown caller IDs from which the user received a phone call (including both unanswered and accepted calls). Consequently, the server will receive daily reports from each user, which consist of the list of unknown caller IDs observed by the client apps running on each device. As we will explain in Section 5, all the caller IDs are delivered by the client apps to the server via a novel LDP mechanism. This is done to provide privacy guarantees and minimize the risk of the server learning any sensitive information about single users' phone calls (e.g., whether the user may be a cancer patient, given that she has received calls from a cancer treatment clinic). At the same time, while the users' privacy is protected, the server is able to identify *heavy hitter* caller IDs that are highly likely associated with new spamming activities. Hence, our system preserves user privacy by making it difficult for the server to learn the list of caller IDs that are contacting the users, while keeping its capability of building a blacklist of possible spammers.

While LDP mechanisms provide strong privacy guarantees, they are often studied in theoretical terms and their applicability to practical, real-world security problems is often left as a secondary consideration. On the contrary, in this paper we focus primarily on adapting a state-of-the-art LDP mechanism for heavy hitter detection [3] to make it *practical*, so that it can be used in the smartphone app described above to report the list of caller IDs to the server. Furthermore, we evaluate the ability of the server to accurately reconstruct the (noisy) reported caller IDs under different privacy budgets, and evaluate the utility of the learned blacklist. To this end, we implement both the client-side (i.e., smartphone side) and server-side (i.e., blacklist learning side) LDP protocol, leaving other app implementation details (e.g., user preferences and controls) to future work.

In summary, we make the following contributions:

- We explore how to build a privacy-preserving collaborative phone blacklisting system using local differential privacy (LDP). Specifically, we expose what are the challenges related to building a *practical* LDP-based system that is able to learn a phone blacklist from caller ID data provided by a pool of contributing users, and propose a number of approaches to overcome these challenges. To the best of our knowledge, our system is the first application of LDP protocols to building a defense against phone spam.
- We implement our blacklisting system using a new LDP protocol for heavy hitter detection. Our protocol is built upon a state-of-the-art protocol previously proposed in [3]. We first show that [3] is not practical, in that it cannot be applied *as is* to collaborative phone blacklisting. We then introduce novel LDP protocol modifications, such as data bucketization and variance-reduction mechanisms, to enable heavy hitter detection by building a LDP-based phone blacklisting approach that could be deployed on real smartphones.
- We evaluate our LDP-based system on real-world user reported call records collected by the FTC. Specifically, we analyze multiple different trade-offs, including the trade-off

between the privacy budget assigned to the different components of our LDP protocol and the overall blacklist learning accuracy. Our results indicate that it is possible to learn a phone blacklist using a reasonable overall privacy budget, and to preserve users' privacy while maintaining utility for the learned blacklist.

2 PROBLEM DEFINITION AND APPROACH

In this section, we outline our threat model and briefly describe our approach towards collaboratively building phone blacklists in a privacy-preserving way.

Threat Model In designing our phone blacklisting system (see Figure 1), we make the following assumptions:

- We consider the caller ID related to phone calls received by users as privacy sensitive (e.g., see the cancer clinic example given in Section 1). However, we do not consider the caller ID area code prefix (e.g., the first three digit of a US phone number) as sensitive. The reason is that each area code includes millions of possible phone numbers (e.g., 10^7 numbers in the US). Therefore, even if the attacker learns that a given user received a phone call from a given area code prefix, she would be faced with very high uncertainty regarding what specific number actually called the user.
- We assume the privacy-preserving data collection app running on each user's device is trusted. Namely, we assume the app correctly implements our proposed LDP protocol (detailed in Algorithm 2), and that it does not directly collect and report any other user data to the server other than the *unknown* phone numbers from which calls were received.
- We also assume that the server correctly executes the server-side of our LDP protocol, to learn a useful phone blacklist that can be propagated back to the users to help them block future spam calls. At the same time, we assume that the server may at some point be compromised (or subpoenaed), allowing an adversary to access future users' reports. Unlike traditional curator-based differential privacy mechanisms, our use of LDP mechanisms guarantees that, in the event of a breach of the server, the privacy of users' phone call records can still be preserved (see Section 3, for details).

It is worth noting that the server may be able to observe the IP address of each reporting device. Furthermore, in a practical deployment, the server may realistically implement an authentication mechanism that requires users to register to the blacklisting service (e.g., by providing an email address, password, etc.), to be allowed to (privately) report call records and receive blacklist updates. In this case, the identity of the users may be known to the server, and a server breach may expose such identities. However, in this paper *we focus exclusively on protecting the privacy of users' phone call records*, rather than anonymity. Protecting the IP address and identity of users may be achieved via other security mechanisms that are outside the scope of this work.

Approach Overview According to recent work on phone blacklisting [22, 23], it is clear that most spammers will tend to call a large number of users, in an attempt to identify a subset of them who may fall for a scam. Therefore, given a large and distributed user population, it is reasonable to consider *heavy hitters* as candidate spammers. In other words, a caller ID that is reported as *unknown* by a significant fraction of participating smartphones satisfies the volume and diversity features used in previous work [22, 23], and can be considered for blacklisting.

Following the high-level approach proposed in previous work, we therefore cast the problem of learning a phone blacklist as a *heavy hitter detection* problem. The main research question we investigate in this paper is the following: using the system depicted in Figure 1, is it possible to accurately detect heavy hitter caller IDs while providing local differential privacy guarantees?

To investigate the above research question, we start from a state-of-the-art LDP protocol for heavy hitter detection proposed by Bassily and Smith [3], which throughout the rest of the paper we will refer to as SH (short for *succinct histogram*). Unfortunately, we have found that the SH protocol is not suitable *as is* for providing a solution to our application scenario (explained in details in Section 4). Among the main issues we found is the fact that SH tends to work well only *in expectation*. As we aim to build a practical blacklisting system, we would like our system to perform well for realistic, limited population sizes (e.g., thousands of users). Furthermore, the protocol used in [3] for calculating the frequency of occurrence for a heavy hitter (i.e., the number of calls made by a likely spammer, in our case) is complex and difficult to implement efficiently (to the best of our knowledge, no implementation of the full [3] protocol is publicly available).

To address the above limitations of the SH protocol, we introduce three LDP protocol modifications:

(1) We propose a novel response randomizer that has the effect of reducing the variance in the noisy inputs received by the server-side of the SH protocol, thus increasing server-side heavy hitter reconstruction accuracy even in the case of a limited user population (Section 4).
(2) Second, we replace the frequency oracle part of the SH protocol proposed in [3] with a much simpler protocol recently proposed in [28], whose implementation is publicly available (Section 3.4).
(3) To increase the relative frequency of heavy hitter caller IDs and boost the likelihood that the server will be able to correctly reconstruct them and add them to the blacklist, we introduce a bucketization mechanism. In essence, before a user (more precisely, the app running on the user's phone) reports one or more caller IDs to the server, the caller IDs are first grouped according to their three-digit area code. Then, the client-side portion of the SH protocol is run independently per each single group (i.e., per each area code). The intuition here is that some spammers tend to use phone numbers from specific area codes. For instance, IRS phone scams are often performed using caller IDs with a 202 prefix (Washington DC area code), as this may trick more users into believing it is truly the IRS that is calling. By grouping caller IDs based on area code, spam numbers also tend to group, increasing their relative frequency compared to all other caller IDs with the same prefix. This effect is discussed in details in Section C.

Section 4 presents the details of our LDP protocol.

Caller ID Spoofing Caller ID spoofing is the main limiting factor for the effectiveness of phone blacklists in general, as also acknowledged in previous work [21, 23]. Previous research on phone blacklisting [21, 23] regards the prevention of caller ID spoofing as an orthogonal research direction, leaving it to future work. This choice can be justified by noting that the FCC has mandated that all US phone companies must implement caller ID authentication by June 30, 2021 [11]. In response, telephone carriers have started activating an authentication protocol known as SHAKEN/STIR [15]. In our work, we make similar considerations as in previous work, and focus our attention on the feasibility of building phone blacklists using user-provided data with strong privacy guarantees. We therefore consider dealing with caller ID spoofing to be outside the scope of this paper.

3 BACKGROUND

3.1 Notation

Suppose there are n users, and that each user j holds an item v_j drawn from a domain $\mathcal{V}$ of size d (in our case, $\mathcal{V}$ is the set of valid phone numbers). For each item $v \in \mathcal{V}$, its frequency $f(v)$ is defined as the fraction of users who hold v, i.e., $f(v) = |\{j \in [n] : v_j = v\}|/n$, where $[n]$ denotes the set $\{1,2,\ldots,n\}$. For notational simplicity, we omit the subscript j when it's clear from the context.

A frequency oracle (FO) is a function that can (privately) estimate the frequency of any item $v \in \mathcal{V}$ among the user population.

For a vector $\mathbf{x} = (x_1,\ldots,x_m)$, we will use the array index notation $\mathbf{x}[i]$ to denote the i^{th} entry, i.e., $\mathbf{x}[i] = x_i$. Similarly, $\mathbf{X}[i,j]$ denotes the entry at location (i,j) for a matrix $\mathbf{X}$.

3.2 Local Differential Privacy

Differential privacy can be applied to two different settings: centralized and local. In the centralized setting, it is assumed that there exists a *trusted data curator* who collects personal data $\mathbf{v} = (v_1,\ldots,v_n)$ from users, analyzes it, and releases the results after applying a differentially private transformation. On the other hand, in the local setting there is *no single trusted third party*. To protect privacy, each user independently perturbs her record v_j into $\tilde{v}_j = \mathcal{A}(v_j)$ using a randomized algorithm $\mathcal{A}$, and only shares the perturbed version with an aggregator (the centralized server responsible for blacklist learning, in our application). The local differential privacy (LDP) model provides stronger privacy protection than the centralized model, because it protects privacy even when the aggregator (i.e., the blacklist learning server, in our case) is compromised and controlled by an adversary. The level of privacy protection depends on a privacy budget parameter ε, as formally defined in [6]; the smaller ε, the greater the privacy guarantees.

3.3 The Succinct Histogram Protocol

Bassily and Smith [3] proposed an ε-LDP protocol, called Succinct Histogram (SH), for detecting heavy hitters over a large domain $\mathcal{V}$. In their work, the authors assume that each user has a single item to share with the server.

Unfortunately, in [3] the client- and server-side of the protocol are presented as "interleaved" in a single algorithm, and to the best of our knowledge a practical implementation of the client-server protocol was not provided. To make the LDP protocol in [3] practical and applicable to our collaborative blacklist learning system, we provide a new but equivalent representation of the protocol proposed in [3] that focuses on the interactions between clients (i.e., the system contributors) and server. Due to space limitations, we report our new client-server formulation in Appendix A (see Algorithms 6 and 7).

Algorithm 1: $\mathcal{R}_{\text{bas}}(\mathbf{x},\varepsilon)$: ε-Basic Randomizer

Input: m-bit string $\mathbf{x}$, privacy budget ε
Sample $r \leftarrow [m]$ uniformly at random.
if $\mathbf{x} \neq \mathbf{0}$ **then**
  $z_r = \begin{cases} c \cdot m \cdot x_r & \text{w.p. } \frac{e^\varepsilon}{e^\varepsilon+1} \\ -c \cdot m \cdot x_r & \text{w.p. } \frac{1}{e^\varepsilon+1} \end{cases}$, where $c = \frac{e^\varepsilon+1}{e^\varepsilon-1}$.
else
  Choose z_r uniformly from $\{c\sqrt{m}, -c\sqrt{m}\}$
return $\mathbf{z} = (0,\ldots,0,z_r,0,\ldots,0)$

The SH protocol works as follows. First, each user $j \in [n]$ encodes her item $v_j \in \mathcal{V}$ into a bit string of length m using a binary error-correcting code (Enc,Dec)[2]. For notational simplicity, we let $\text{Enc}(\cdot) = \mathbf{c}(\cdot)$. Let $\mathbf{x}_j \in \{-1/\sqrt{m}, 1/\sqrt{m}\}^m$ be the encoded binary string. The encoded item $\mathbf{x}_j$ and its decoding are respectively given by

$$\mathbf{x}_j = \text{Enc}(v_j) = \mathbf{c}(v_j) \text{ and } \text{Dec}(\mathbf{x}_j) = v_j.$$

For privacy, each user j perturbs $\mathbf{x}_j$ into a noisy report $\mathbf{z}_j = \mathcal{R}_{\text{bas}}(\mathbf{x}_j,\varepsilon)$ using a randomizer $\mathcal{R}_{\text{bas}}$ and sends it to the server. The pseudo-code of randomizer $\mathcal{R}_{\text{bas}}$ is described in Algorithm 1.

To simplify the heavy hitter detection problem, Bassily and Smith applied the idea of isolating heavy hitters into different channels using a pairwise independent hash function $H : \mathcal{V} \to [K]$, whereby an item v is mapped to channel $H(v)$.

This has the effect that, with high probability, no two unique heavy hitter items are mapped to the same channel (when K is sufficiently large). For each channel , users with $H(v_j) = v^*$ encode v_j into $\mathbf{x}_j = \text{Enc}(v_j)$ and send the perturbed version of $\mathbf{x}_j$; whereas $\mathbf{x}_j = \mathbf{0}$ for users with $H(v_j) \neq v^*$ and $\mathcal{R}_{\text{bas}}(\mathbf{0})$ is reported to the server.

Given a set of noisy reports $\{\mathbf{z}_1,\ldots,\mathbf{z}_n\}$ collected from n users, the server aggregates them to $\bar{\mathbf{z}}$ (line 6 in Algorithm 7, in Appendix), rounds it to the nearest valid encoding $\mathbf{y}$ (line 7-8 in Algorithm 7, in Appendix), and finally reconstructs the heavy hitter item by decoding it into $\hat{v} = \text{Dec}(\mathbf{y})$. To estimate the frequency of $\hat{v}$, the server collects another set of noisy reports $\{\mathbf{w}_1,\ldots,\mathbf{w}_n\}$ and estimates the frequency as follows:

$$\hat{f}(\hat{v}) = \langle \frac{1}{n}\sum_{j=1}^{n} \mathbf{w}_j, \mathbf{c}(\hat{v}) \rangle = \frac{1}{n}\sum_{j=1}^{n} \mathbf{w}_j[r_j] \cdot \mathbf{q}[r_j],$$

where $\mathbf{q} = \mathbf{c}(\hat{v})$.

To filter out possible false positives, similarly to the previous phase the server collects noisy reports $\mathbf{w}_j$ from users and aggregates them in a single bitstring $\overline{\mathbf{w}}$. For each reconstructed value $\hat{v}$ in Γ, its frequency $f(\hat{v})$ is estimated using a frequency oracle (FO) function. If the computed estimate $\hat{f}(\hat{v})$ is less than a threshold η, then $\hat{v}$ is removed from Γ. After this filtering phase, the server can then return the set of detected heavy hitters.

[2]Specifically, the protocol requires a $[2^m, k, d]_2$ binary error-correcting code, where 2^m, k, and d represent the codeword length, encoded message length (in bits), and minimum distance, respectively, in which the relative distance $d/2^m$ has to be included in the interval $(0, 1/2)$.

The threshold η plays a crucial role in the heavy hitter detection:

$$\eta = \frac{2T+1}{\varepsilon}\sqrt{\frac{\log(d)\log(1/\beta)}{n}} \quad (1)$$

where β [3] is a parameter related to the confidence the server has on the heavy hitters it has detected. The same parameter β also influences the number of protocol rounds, T [3].

We now analyze the properties of the basic randomizer. It is easy to see that for every encoded item $\mathbf{x} \in \{\frac{1}{\sqrt{m}}, -\frac{1}{\sqrt{m}}\}^m \cup \{\mathbf{0}\}$ its noisy report $\mathbf{w} = (w_1,\ldots,w_m)$ is an unbiased estimator of $\mathbf{x}$. For users with $\mathbf{x} \neq \mathbf{0}$ and an integer $r \in [m]$,

$$\mathbb{E}[w_r] = cm\left(\frac{e^\varepsilon}{e^\varepsilon+1} - \frac{1}{e^\varepsilon+1}\right)x_r = m \cdot x_r \text{ and}$$
$$\mathbb{E}[\mathbf{w}] = \frac{1}{m}(\mathbb{E}[w_1],\ldots,\mathbb{E}[w_m])^\intercal = \mathbf{x}.$$

For users with $\mathbf{x} = \mathbf{0}$, $\mathbb{E}[w_r] = 0$ for $\forall r \in [m]$, and hence $\mathbb{E}[\mathbf{w}] = \mathbf{0} = \mathbf{x}$. It is also easy to see that for $v \in \mathcal{V}$ the estimated frequency $\hat{f}(v)$ has the following properties (see Appendix B for details):

$$\mathbb{E}[\hat{f}(v)] = f(v)$$

and

$$\mathrm{Var}(\hat{f}(v)) = \frac{1}{n}\left\{\left(\frac{e^\varepsilon+1}{e^\varepsilon-1}\right)^2 - f(v)\right\}. \quad (2)$$

Limitations: The SH protocol described in [3] was presented in a purely formal way, without addressing limitations that exist in practical systems. For instance, the original SH protocol was formulated in an "asymptotic" setting, in which a large number of reporting clients is assumed. While the protocol works well *in expectation*, it presents a number of practical drawbacks, which we discuss in Section 4.

3.4 Frequency Oracle Protocol

Wang *et al.* [28] recently proposed the Optimal Local Hashing (OLH) protocol for estimating the frequency of items belonging to a given domain. It satisfies ε-LDP and is simpler and logically equivalent to the frequency oracle proposed in [3]. Instead of transmitting a single bit, that is the result of mapping an item i to a binary value $\{c\sqrt{m}, -c\sqrt{m}\}$, the n users who participate in the system simply hash their items into a value in $[g]$, where $g \geq 2$. The pseudo-code of the OLH randomizer is reported in Algorithm 5 (in Appendix A). For further details, we refer the reader to [28] and to its publicly available implementation [27]. It is worth noting that OLH is limited to frequency estimation, and that it is not suitable by itself for heavy hitter detection in large domains, as for the case in which the domain includes all possible valid phone numbers.

4 SYSTEM DETAILS

As mentioned earlier, we envision a collaborative blacklisting system consisting of n distinct smartphones and a centralized server C, as shown in Figure 1. C is responsible for receiving data from the participating phones and for computing a blacklist of phone numbers (i.e., caller IDs) likely associated with phone spamming activities. Once computed, the blacklist can be propagated back to the participating phones to enable flagging future unwanted calls as *likely spam*.

Each participating smartphone runs an application that collects information about phone calls received from *unknown* phone numbers, where *unknown* here means that the caller's phone number was not registered into the smartphone's contact list. More precisely, let p_i be a participating smartphone and c_j be a caller ID. If p_i receives a call from c_j and c_j is not in p_i's contact list, then c_j is labeled as *unknown* and reported to C by p_i. Notice that, in this scenario, only the caller ID c_j is reported, and no information about the content of the call is shared with C.

To preserve the privacy of phone calls received by participating users (i.e., the owners of the phones that contribute data to C), the caller ID data collection app running on each smartphone implements a local differentially private (LDP) algorithm, whose details are described below in this section.

4.1 Overview of LDP Protocol

Following the intuitions and motivations for our approach provided in Section 2, we cast the problem of learning a phone blacklist as a *heavy hitter detection* problem. To this end, we build our solution upon the SH protocol for heavy hitter detection proposed in [3] and summarized in Section 3. Unfortunately, the original SH protocol is not directly suitable for our application, because it was formulated in a theoretical "asymptotic" setting in which a large number of reporting clients is assumed [3].

First, we implemented a practical client-server formulation of the SH protocol proposed in [3]. After performing pilot experiments, we found that applying the protocol *as is* in a setting with a limited number of clients (e.g., in the tens of thousand) and in which we aim to correctly reconstruct heavy hitters with a relatively low volume (e.g., only a few hundreds of hits) was not possible without setting an extremely high privacy budget, thus completely jeopardizing users' privacy.

To make SH usable in practice and adapt it to our phone blacklisting problem, we therefore designed and implemented a number of modifications that address the following two fundamental problems:

- *Sparsity of user reports*: In the SH protocol, the larger the items domain $\mathcal{V}$, the more frequent an item must be to be correctly reconstructed by the server C with high probability. Namely, an item must be reported by a larger and larger population, as the cardinality of $\mathcal{V}$ increases, thus potentially impeding the reconstruction of spam phone numbers involved in campaigns that reach only a portion of the contributing users.
- *High variance in the sum of item reports hinders noise cancellation*: The sum $\overline{\mathbf{z}}$ in Algorithm 7 (line 6) is affected by the high variance of the distribution of the sum of each bit. Ideally, the noisy random bits sent by users who do not hold value v (i.e.,that transmit a randomized version of $\mathbf{x} = \mathbf{0}$ in Algorithm 6, lines 7-8) should cancel out during summation. While this holds *in expectation*, in practice (with a finite number of participants) it is highly unlikely to have the very same number of clients who transmit $\frac{1}{\sqrt{m}}$ as clients who transmit $-\frac{1}{\sqrt{m}}$, potentially causing the reconstruction of the wrong bit value at the server side.

We address the first issue by introducing a data bucketization mechanism. Specifically, we take advantage of characteristics of

Algorithm 2: Modified SH-Client($v, \mathcal{H}, T, K, \varepsilon_{\text{HH}}, \varepsilon_{\text{OLH}}$)

Input: the value to be sent v, a fixed list of hash functions $\mathcal{H}$, # of repetitions T, # of channels K, privacy parameterblacks ε_{HH} and ε_{OLH}

/* sending noisy reports for heavy hitter detection */

$\rho \leftarrow prefix(v)$

$\sigma \leftarrow v \setminus \rho$

mygrayfor *mygrayt* $= 1$ **to** T **mygraydo**

  mygray$H \leftarrow \mathcal{H}[t]$

  mygrayforeach *mygraychannel* $k \in [K]$ **mygraydo**

    if $H(\sigma) = k$ **then**

      $\mathbf{x} = \text{Enc}(\sigma)$

    mygray**else**

      $\mathbf{x} = \mathbf{0}$

    $\mathbf{z}^{(t,k,\rho)} \leftarrow \mathcal{R}_{\text{ext}}\left(\mathbf{x}, \frac{\varepsilon_{\text{HH}}}{2T}\right)$

    Send $\mathbf{z}^{(t,k,\rho)}$ to the server on channel k

/* sending noisy report for heavy hitter frequency estimation */

$w^{(\rho)} \leftarrow \mathcal{R}_{\text{OLH}}(v, \varepsilon_{\text{OLH}})$

Send $w^{(\rho)}$ to the server

Algorithm 3: Modified SH-Server(T, K, P, OLH)

Input: # of repetition T, # of channels K, set of prefixes P, a frequency oracle OLH, threshold η

Output: list of heavy hitters Γ

/* detecting heavy hitters */

mygray$\Gamma \leftarrow \emptyset$

mygrayfor *mygrayt* $= 1$ **to** T **mygraydo**

  foreach *prefix* $\rho \in [\text{P}]$ **do**

    mygrayforeach *mygraychannel* $k \in [K]$ **mygraydo**

      blackforeach *user* $j \in [n_\rho]$ **blackdo**

        $\mathbf{z}_j \leftarrow \mathbf{z}^{(t,k,\rho)}$ value received from user j on channel k, having prefix ρ;

      $\overline{\mathbf{z}} = \frac{1}{n_\rho} \sum_{j=1}^{n_\rho} \mathbf{z}_j$

      mygrayfor $i = 1$ *mygray***to** m **mygraydo**

        mygrayy$[i] \leftarrow \begin{cases} \frac{1}{\sqrt{m}} & \text{if } \overline{\mathbf{z}}[i] \geq 0 \\ -\frac{1}{\sqrt{m}} & \text{otherwise.} \end{cases}$

      $\hat{\sigma} \leftarrow \text{Dec}(\mathbf{y})$

      $\hat{v} \leftarrow$ append σ to ρ

      mygray**if** $\hat{v} \notin \Gamma$ **then** add $\hat{v}$ to Γ

/* filtering out false positives */

foreach *prefix* $\rho \in [\text{P}]$ **do**

  foreach *user* $j \in [n_\rho]$ **do**

    $\overline{\mathbf{w}}[j] \leftarrow w^{(\rho)}$ value received from user j having prefix ρ

  mygrayforeach *mygray*$\hat{v} \in \Gamma$ **mygraydo**

    $\hat{f}(\hat{v}) \leftarrow$ estimate the frequency of $\hat{v}$ using OLH($\overline{\mathbf{w}}$)

    mygray**if** $\hat{f}(\hat{v}) < \eta$ **then** remove $\hat{v}$ from Γ

mygray**return** $\{(v, \hat{f}(v)) : v \in \Gamma\}$

the phone blacklisting problem to (1) reduce the dimensionality of the items domain, and (2) partition the problem domain to increase the relative frequency of heavy hitters. To achieve (1), we divide phone numbers into area code prefix (e.g., the first three digits in a US telephone number) and phone number suffix (e.g., the remaining seven digits, for a US phone number).

In telephony, area codes are typically not considered to be sensitive. For instance, the FTC dataset protects the privacy of complaining users by publishing only their area code [14] (see also Sections 2 and 5.1). As outlined in Section 2, we aim to protect the privacy of the phone number suffix. Hence, given the large number of phone numbers that share the same prefix, clients can transmit the area code *as is*, and apply the (modified) SH protocol only to the phone number suffix, thus reducing the number of bits needed to represent each phone number. To achieve (2), we assign a separate communication channel between clients and server to each area code, and run an instance of the (modified) SH protocol independently for each area code. This has the effect of clustering phone numbers based on their prefix. Because in some cases phone spam campaigns are conducted using specific area codes (e.g., a Washington DC area code for IRS spam campaigns, or an 800 prefix for tech support scams, etc.), this *bucketization* of phone numbers has the effect of amplifying the relative frequency of spam-related caller IDs in some of the area code buckets (or clusters), thus making it easier to detect heavy hitters. Figure 2 shows a example of how in practice bucketization helps to amplify the relative frequency of heavy hitters.

In summary, we (i) group phone numbers by area code, (ii) split area code and phone suffix, (iii) select the client-server communication channel based on the area code, and (iv) let each client transmit the area code in clear (i.e., no LDP) and run the SH protocol over phone number suffixes within transmitted area codes.

To address the high variance in the sum of item reports, we introduce a new *extended randomizer* to replace the original randomizer proposed in [3] and reported in Algorithm 1. The main idea is to use a three-value randomizer. For instance, when $\mathbf{x} = \mathbf{0}$ must be sent, instead of choosing a random bit value between $\{\frac{1}{\sqrt{m}}, -\frac{1}{\sqrt{m}}\}$, the client app will choose between three values: $\{\frac{1}{\sqrt{m}}, 0, -\frac{1}{\sqrt{m}}\}$, with different probabilities. Our externded randomizer is defined in Algorithm 4. In Appendix D we formally show how this extended randomizer helps reducing the variance, thus increasing the accuracy with which privately-reported phone numbers are reconstructed on the server side.

Notice also that while in the following we present our LDP protocol under the assumption that each user has a single item to share with the server (e.g., one unknown phone number report per day), in real scenarios some users may either have multiple items to share or nothing to share at all (e.g., no phone calls received in a given day). In this case, the protocol can be easily extended as proposed in [24], by sampling a single telephone number from the set of unknown calls that the app has collected. Conversely, if a user has nothing to

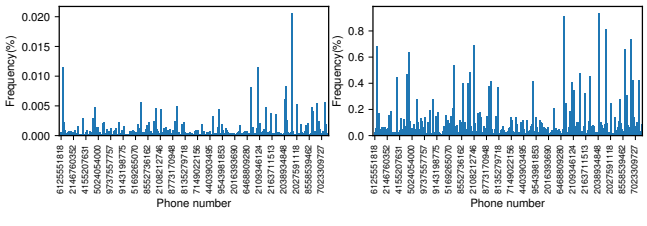

Figure 2: Phone number frequency before (left) and after (right) bucketization, computed on one day of complaints from the FTC dataset.

Algorithm 4: $\mathcal{R}_{\text{ext}}(\mathbf{x}, \varepsilon)$: ε-extended Randomizer

Input: m-bit string $\mathbf{x}$, privacy budget ε

Sample $r \leftarrow [m]$ uniformly at random.

if $\mathbf{x} \neq \mathbf{0}$ **then**

  $z_r = \begin{cases} c \cdot m \cdot x_r & \text{w.p. } p \\ -c \cdot m \cdot x_r & \text{w.p. } q \\ 0 & \text{w.p. } 1-p-q \end{cases}$, where $c > 0$.

else

  $z_r = \begin{cases} c\sqrt{m} & \text{w.p. } \theta \\ -c\sqrt{m} & \text{w.p. } \theta \\ 0 & \text{w.p. } 1-2\theta \end{cases}$

return $\mathbf{z} = (0, \ldots, 0, z_r, 0, \ldots, 0)$

share, the app can generate a dummy (but legitimate) phone number to be sent to the server.

It is also important to notice that organizing phone number reports in buckets allows the server to count the number of users that will participate to a protocol run, per each bucket. In Appendix C, we discuss how the server can use this number to estimate the probability that at least one heavy hitter in a specific bucket will be successfully reconstructed. If such probability is low, the server can avoid executing the protocol for those buckets and inform the clients of this decision, thus preventing those clients from wasting their privacy budget. In practice, once the server receives the area codes from each client, it could send a message back to the clients letting them know if they should send (using the LDP protocol) the remaining portion of the caller IDs they observed (i.e., the remaining seven digits) or not.

The new LDP protocol resulting from our improvements over the original SH protocol is shown in Algorithms 2 and 3, where new pseudo-code is highlighted in black, and code that remains the same as in the original SH protocol is shaded in gray. Unlike the original version, we explicitly allocate two different privacy budgets, $\varepsilon_{\mathsf{HH}}$ and $\varepsilon_{\mathsf{OLH}}$, assigned respectively to heavy hitter detection and frequency estimation. It is worth mentioning that $\varepsilon_{\mathsf{HH}}$ is the total privacy budget spent by each user to send noisy reports to the server during the T protocol rounds (lines $3-11$). In this new formulation $\varepsilon = (\varepsilon_{\mathsf{HH}} + \varepsilon_{\mathsf{OLH}})$ and the protocol is $(\varepsilon_{\mathsf{HH}} + \varepsilon_{\mathsf{OLH}})$-differentially private, as proved in Appendix D.

5 LDP PROTOCOL EVALUATION

In this section, we present an evaluation of our LDP protocol. It is important to notice that we focus primarily on estimating the accuracy of our system with respect to *reconstructing and detecting heavy hitters*. We will discuss the utility of the phone blacklist that may be learned from the detected heavy hitters separately, in Section 6.

5.1 Dataset

Ideally, to evaluate our protocol we would need to collect a dataset of phone call records from thousands of users. Even though we assume only calls from *unknown* numbers (i.e., numbers not stored in the users' respective contact lists) would be of interest for detecting potential spam phone numbers, collecting such a dataset is very difficult, due exactly to the same privacy issue we aim to solve in this paper. As a proxy for that dataset, we make use of real-world phone data extracted from user complaints to the FTC. In essence, the FTC allows users in the US to report unwanted phone calls. Reported complaints that are made available to the public typically include the time of the complaint, the full caller ID, the user's phone number area code, and a label indicating the type of phone spam activity. Notice that the FTC aim to protect the privacy of the users who report a complaint; that's why they only publish users' phone number prefixes. On the other hand, the caller IDs are reported in clear because the complaining users explicitly label them as *unwanted caller*, essentially consenting to their public release. On the other hand, our system does not require users to explicitly label unwanted phone calls; all *unknown* caller IDs are reported, and privacy is preserved via LDP mechanisms (please refer to Section 1, where we motivate the advantages of this approach).

In this paper, we treat each complaint as if a user participating in our collaborative blacklist learning system had received a phone call from the complained-about number at the time recorded in the complaint. We were able to obtain a large set of user complaints collected by the FTC between Feb. 17th 2016 and Mar. 17th, 2016, for a total of 29 days, which consists of 471,460 complaints. For the sake of this evaluation, we consider only valid 10-digit caller IDs, which constitute about 95% of the entire dataset. The distribution of complaints is characterized by two properties: (a) the volume of complaints follows a weekly pattern, with fewer complaints submitted on weekends; and (2) the distribution of complaints per caller ID has a long tail, whereby most phone numbers receive only one complaint but there also exist many phone numbers that receive hundreds of complaints (see Figures 9 and 10 in appendix for details).

As the FTC dataset does not contain an identifier for the reporting users, without loss of generality we assume that, within a day, each complaint is reported by a different user. Based on this, we determine the pool of users that would participate in our collaborative blacklisting as follows: we compute the maximum number of reports seen in one day, throughout the entire month of FTC data, which is equal to 23,188; we then set the number of participants to that number. In days in which fewer than 23,188 users sent complaints to the FTC, we assume that the remaining users did not have any calls to report (i.e., they did not receive any calls from *unknown* numbers on those days). However, to preserve differential privacy guarantees, *all* users must send a report every time the protocol runs (e.g., daily, in our evaluation). Therefore, if a user has no calls to report, the app on her smartphone will generate a random (but valid) 10-digit number, and report that number to the server using the LDP protocol, exactly as if she received a call from that number.

5.2 LDP Protocol Configuration

In all our experiments we use area code bucketization, with the same parameter settings for all buckets. We also compare results obtained using the basic randomizer proposed in [3] to results obtained using our extended randomizer (Algorithm 4), using the same parameter settings for each, to allow for an "apples to apples" comparison.

We set two different privacy budgets to run the heavy hitter detection and the frequency estimation protocol. Respectively, we experiment with budget $\varepsilon_{\mathsf{HH}} \in \{12, 8.8, 7, 5.6, 4.4\}$ for heavy hitter detection, and $\varepsilon_{\mathsf{OLH}} = 3$ for frequency estimation. We chose the values $\varepsilon_{\mathsf{HH}}$ for both the basic and extended randomizer so that the randomizer sends the correct phone number with probability approximately equal to 0.95, 0.90, 0.85, 0.80 and 0.75. Moreover, we do not allocate a $\varepsilon_{\mathsf{HH}}$ value lower than 4.4 given that, as later discussed in Section 7, the utility of the learned blacklist is already 0 when $\varepsilon_{\mathsf{HH}} = 4.4$ (see Figure 7).

It is important to also notice that guaranteeing differential privacy under continual observation [8] in the more difficult LDP setting is still an open problem in differential privacy research. A complete solution to such a challenging open problem is therefore left to future work. Nonetheless, one possible mitigation may be to design the data collection app so that a user does not report the same number more than once (see Section 7 for further discussion).

To implement the binary encoding of phone numbers (see Algorithm 2 line 7), we use a Reed Muller error-correcting code, $RM(3,5)$; this is a $[32,26,4]_2$ code with relative distance equal to $1/8$ and error correcting capability equal to 1 bit [17]. Notice that $k = 26$ bits is sufficient to encode 7-digit phone numbers, which requires a minimum of

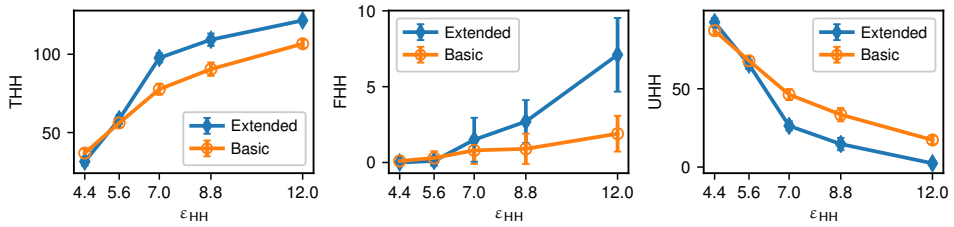

Figure 3: True, false and undetected heavy hitters (respectively, THHs, FHHs and UHHs). Parameters: $T=2$, $\varepsilon_{OLH}=3$, and $\tau=143$.

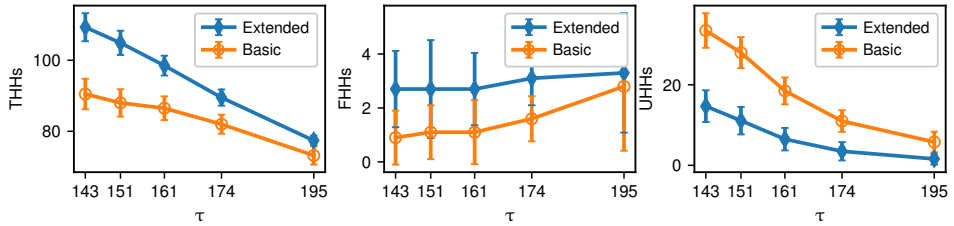

Figure 4: True, false and undetected heavy hitters (respectively, THHs, FHHs and UHHs). Parameters: $T=2$, $\varepsilon_{HH}=8.8$, $\varepsilon_{OLH}=3$.

24 bits. At server-side, we run the heavy hitter detection phase only for those buckets containing more than a minimum number, τ, of complaints, as motivated in Sections 4.1 and C. We experiment with 5 different values of τ in the set {143,151,161,174,195}. These values correspond, respectively, to a 75%, 80%, 85%, 90%, 95% probability of correctly reconstructing, at least, a phone number per bucket (see also Equation 4).

5.3 Measuring Heavy Hitter Detections

We now define how we measure accuracy for the LDP protocol. Notice that in this section we consider accuracy strictly for the *heavy hitter detection* task accomplished by the LDP protocol. This is related to but different from the utility of the blacklist that can be learned over the phone numbers reconstructed by the server-side LDP protocol (see Section 6 for results on blacklist utility).

Let v be a phone number, and $c(v)$ be the number of users who reported a call from v. We say that $c(v)$ is the *ground truth frequency* of v. Moreover, let $\hat{f}(v)$ be the number of reports about v estimated by the server after running the LDP protocol, and η be the detection threshold for heavy hitter detection defined in Equation 1 (see also Algorithms 7 and 3). Also, as discussed in Section 5.2, let τ be the minimum number of complaints necessary for the server to decide whether to run the heavy hitter detection phase for a bucket.

In theory, we could simply use η as heavy hitter frequency threshold, to measure true and false detections. However, given Equation 3 and substituting practical values of ε and β, η tends to be much smaller than τ. For instance, considering $\varepsilon=15$, $\beta=0.751$, $T=2$, and $d=10^7$, and assuming $n=1000$ users reporting caller IDs to a given area code bucket, we obtain $\eta\cong 0.023$. Thus, the heavy hitter detection threshold (in terms of number of reports per caller ID) would be $\eta\cdot n=23$. In other words, a phone number would be considered a heavy hitter if it is reported more than 22 times. Yet, as we discussed in Section C, the minimum number of reports needed for the server to correctly reconstruct a phone number with high probability is much higher than 23 (e.g., at least 84 reports are needed to have a 50% chance of correct reconstruction). We therefore use τ as the heavy hitter detection threshold, rather than relying on η. Specifically, we define the following quantities.

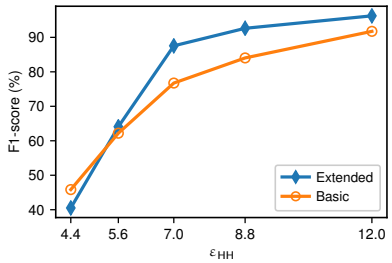

Figure 5: F1-score with parameters $T=2$, $\varepsilon_{OLH}=3$, and $\tau=143$.

- **True Heavy Hitters** (THHs). We have a *true heavy hitter* detection for v if both $c(v)>black\tau$ and $\hat{f}(v)>black\tau$.
- **False Heavy Hitters** (FHHs). We have a false heavy hitter detection for v if $c(v)\leq black\tau$ whereas $\hat{f}(v)>black\tau$.
- **Undetected Heavy Hitters** (UHHs). We have an undetected heavy hitter if $c(v)>black\tau$ whereas $\hat{f}(v)\leq black\tau$.

It is worth noting that FHHs are typically due to a phone number v whose true frequency $c(v)$ is just below τ, and for which the noise introduced by the LDP protocol causes the server to (by chance) estimate its frequency above the heavy hitter detection threshold. On the other hand, UHHs represent heavy hitters that the protocol fails to detect, due to the random noise added by the clients. Given the above definitions, and their analogy with true and false positives in detection systems, we measure the F1-score of the heavy hitter detection protocol as:

- **Recall**: $R=THHs/(THHs+UHHs)$
- **Precision**: $P=THHs/(THHs+FHHs)$
- **F1-score**: $F_1=2*(P*R)/(P+R)$

5.4 LDP Heavy Hitter Detection Accuracy

Figure 3 shows the number of THH detected for different privacy budgets ε_{HH}, when $T=2$, and $\varepsilon_{OLH}=3$ (error bars represent one standard deviation). The figure compares the accuracy that can be

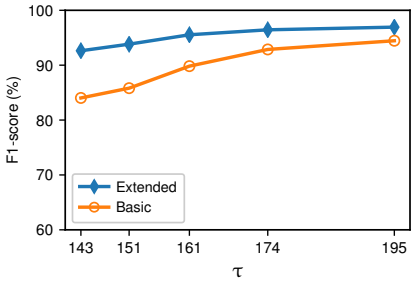

Figure 6: F1-score with parameters: $T = 2$, $\varepsilon_{HH} = 8.8$, $\varepsilon_{OLH} = 3$, **and** $\tau = 143$.

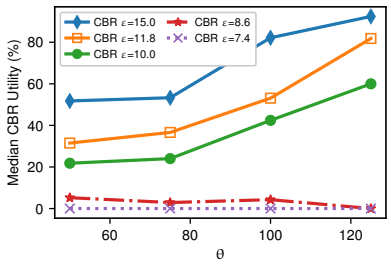

Figure 7: CBR: percentage of calls blocked compared to the baseline.

obtained by using the basic randomizer (as in [3]) and our extended randomizer (Algorithm 4). The maximum privacy budget spent daily by each client running the LDP protocol can be computed by summing privacy budgets ε_{HH} and ε_{OLH} (e.g., $\varepsilon = 15$, when $\varepsilon_{HH} = 12$ and $\varepsilon_{OLH} = 3$). It is worth noting that an user may have no phone number to report: in that case, the privacy budget spent by the client would be 0. Each experimental evaluation with a given ε_{HH} and ε_{OLH} was repeated 10 times, and the results averaged.

Figure 3 also reports the number of FHHs and UHHs obtained for different values of the privacy budget. As can be seen, the LDP protocol detects less than 8 FHHs, on average. As mentioned in Section 5.3, such false heavy hitters are phone numbers whose frequency is just below τ and whose LDP-estimated frequency happens to slightly exceed the heavy hitter detection threshold due to the randomization of user contributions. In addition, the higher the ε_{HH} allocated for detecting heavy hitters, the higher the probability of correctly reconstructing phone numbers whose frequency is just below τ and, hence, generating FHHs. Figure 4 shows how THHs, FHHs, and UHHs vary with τ, using the same parameters of the previous experimental evaluations, but fixing ε_{HH} to 8.8. As can be observed, increasing τ decreases the total number of detectable heavy hitters (i.e., the sum of THHs and UHHs), as expected, since fewer and fewer reported caller IDs will have a true frequency $c(v) > \tau$.

It is also important to notice that, as shown in Figure 5, overall our LDP protocol with the extended randomizer performs better than using the basic randomizer proposed in [3], when $\varepsilon_{HH} \in \{12, 8.8, 7\}$, which yield an F1-score above 85%. The scores have been computed by using THHs, FHHs, and UHHs depicted in Figure 3, averaged across 10 runs. For lower values of ε_{HH}, the F1-score decreases significantly, and the extended randomizer tends to perform slightly worse than the basic randomizer. This is because the probability of injecting noise in the reports (at the clients side) increases considerably. This aspect, in combination with the fact that, by definition, the extended randomizer has a lower probability of sending the correct report to the server, compared the basic randomizer, determines a slight reduction in performance for low values of ε_{HH}.

Finally, Figure 6 reports the F1-score as τ changes. Higher values of τ allow us to obtain higher scores, because the number of UHHs decreases (see Figure 4). The basic and extended randomizers follow similar trends, though the extended randomizer performs better than the basic one, independently from the choice of τ.

6 BLACKLIST UTILITY

In Section 5 we evaluated the ability of our LDP protocol to accurately detect heavy hitter caller IDs. We now look at how a blacklist learned over heavy hitters detected using our protocol would fare compared to when no privacy is preserved, whereby caller IDs are collected from users' phones and sent directly to the server (no noise added). To compare these scenarios, we leverage the *call blocking rate* (CBR) metric proposed in [23].

In a way similar to [23], we define a blacklist $\mathbb{B}$ as a set of caller IDs that have been reported by users more than θ times. Specifically, as in [23], we use a sliding window mechanism, whereby a blacklist $\mathbb{B}$ is updated daily by cumulatively adding *daily heavy hitter* caller IDs observed over the past time window (one week, in our experiments). Blacklisted caller IDs older than the sliding window are forgotten, and removed from $\mathbb{B}$. As an example, making use again of the FTC dataset (see Section 5.1), the blacklist that each user deploys on February 24th contains all the heavy hitters detected each day during the week going from Feb. 17th to Feb. 23rd. The CBR is then computed by measuring how many calls are flagged by $\mathbb{B}$ on the day of deployment.

To enable a comparison between the private and non-private versions of blacklist learning, we set the same fixed heavy hitter detection threshold θ for both. In other words, in the case when no privacy is offered, caller IDs that are reported by more than θ users in a day are considered as potential spammers. Similarly, when our LDP protocol is used to learn the blacklist, we fix the heavy hitter detection threshold $\tau = \theta$. The other protocol parameters in this experiment are set to $T = 2$ and $\varepsilon_{OLH} = 3$, while varying ε_{HH}.

As a baseline, we compute (over the FTC dataset) the median call blocking rate CBR* that can be achieved throughout a month of FTC reports, without applying any privacy-preserving mechanism and for different values of θ (we compute the median because it is less sensitive to outliers, compared to the average). Then, we compare CBR* to the CBR obtained by the blacklist learned using our LDP protocol, by computing the median of the fraction of calls that our blacklist would block, compared to CBR*. The results are reported in Figure 7. As can be seen, as the overall privacy budget ε increases, the CBR approaches the baseline CBR*, which is indicated by the 100% mark. It can be noticed that the difference with the baseline increases as θ reduces. This is because it is more unlikely that the server will correctly reconstruct caller IDs that have a lower number of reports (see Section C). Therefore, as θ decreases, heavy hitters with low frequency (close to θ) can still be detected in the scenario without privacy, but become more difficult to detect for our LDP protocol.

In practice, whenever a user receives a call from an *unknown* caller ID that is in the blacklist, the app will inform the user that the number is suspicious, and potentially involved in spamming. The user may ultimately decide to pick up the call, but use more caution when interacting with the other party.

7 DISCUSSION

In Section 5.4, we have reported several results related to the accuracy of the proposed LDP protocol using different privacy budgets and confidence parameters. Depending on how much budget the server provides to system users, the SH protocol parameters can be tuned to control privacy/utility trade-offs. As mentioned in Section 5.2, Apple uses up to $\varepsilon = 16$ [1] as privacy budget for gathering statistics. For instance, the Safari browser allows for two user contributions per day, with $\varepsilon = 8$ each. On the other hand, in this paper we experimented with a maximum privacy budget of $\varepsilon = 15$ with one user contribution per day. While the privacy budget may seem somewhat high compared to *non-local* differential privacy applications, it is worth noting that this is due to the inherent complexity of LDP. For instance, it has been shown that ε-LDP distribution estimators require k/ε^2 times larger datasets than a comparable non-private algorithm [6, 19], where k is the size of the input alphabet (i.e., k is the number of possible phone number combinations, in our case). As k can be very large, higher values of ε allow us to achieve an acceptable utility even with relatively small values of the sample set size n (i.e., the number of noisy reports received by the server). Furthermore, we also showed that even for lower values of epsilon (e.g., $\varepsilon = 11.8$), blacklist utility can still be reasonable (e.g., around 80% of the CBR* obtained in the scenario with no privacy, as shown in Section 6). A privacy budget that can provide more privacy while keeping a good performance trade-off is $\varepsilon = 10$ (with $T = 2$, $\varepsilon_{\mathsf{HH}} = 7$, $\varepsilon_{\mathsf{OLH}} = 3$, and $\tau = 143$): our experimental evaluation shows an F1-score higher than 75% with the detection of more than 97 potential spam phone numbers per day, on average.

A limitation of our system, which is common to practical deployments of LDP such as in the case of Apple and other vendors, is that guaranteeing differential privacy under continual observation [8] in an LDP setting is still an open research problem in differential privacy. As a possible mitigation, the data collection app running on the user's phone can keep history of the reported numbers and avoid reporting the same calling number more than once within a given time window (e.g., one month). This would make it much more difficult for the server to identify a phone number that may have called a specific user with high frequency (e.g, once a day), since it will be reported only once by that user. At the same time, if the same number is reported only once but by many users, it can still be detected as heavy hitter and added to the blacklist.

It is also possible that a legitimate phone number may be reported by many users, such as in the case of school alert numbers or other types of emergency phone numbers that may contact a large number of users at once, since these numbers may not be recorded in every user's contact list. Such phone numbers may potentially be detected as heavy hitters, and thus considered by the server for blacklisting. However, the server could check the validity of a number, before propagating it to the blacklist. For instance, the server could use automated *reverse phone number lookup* services (e.g., whitepages.com) to filter out possible false positives related to emergency numbers.

Our work is based on the heavy hitter LDP protocol proposed in [3], which, to the best of our knowledge, was one of very few state-of-the-art LDP protocols for heavy hitter detection at the time when we started the research presented in this paper. Alongside [3], RAPPOR [9, 10] is another protocol that could be adapted to fit our problem. However, it has been shown that RAPPOR performs less well than a more recent protocol named TreeHist [2], and that in turn TreeHist itself has a higher worst-case error, compared to the original SH protocol proposed in [3]. Similarly, it has been shown in [28] that for frequency estimation the OLH protocol (which we summarized in Section 3.4 and used in our system) performs better than RAPPOR.

Recently, a few new LDP protocols for heavy hitter detection have been also proposed [2, 24, 29]. However, [3] remains a state-of-the-art protocol that has inspired more recent works. Furthermore, in this paper we focus on studying how to make LDP heavy hitter detection practical to address an important and previously unsolved security problem: *privacy-preserving collaborative phone blacklisting*. We believe that the application-specific trade-offs between privacy and utility we presented in this paper would still be relevant even if [3] was replaced by a different LDP heavy hitter detection protocol.

In Section 5, we performed experiments with a fixed value of parameter $T = 2$. In the original formulation of the SH protocol [3], T is directly related to the parameter β we mentioned in Section 3. While it would be possible in theory to use higher parameter values, increasing T (by varying β) would result in a higher number of protocol rounds, and would thus consume a much larger privacy budget ε for each user. Conversely, increasing T while keeping ε fixed would cause a significant degradation of heavy hitter detection accuracy, and in turn of the blacklist utility. Therefore, for the sake of brevity, we did not report experimental results obtained with larger values of T.

8 RELATED WORK

Besides RAPPOR [9, 10], which we briefly discussed in Section 7, there exist other works related to LDP heavy hitter detection; we briefly discuss them below. However, it should be noted that our work is different from the ones discussed here. Our main contributions are in adapting a state-of-the-art protocol proposed in [3] to make it practical, and in using the adapted protocol to build a collaborative phone blacklisting system with provable privacy guarantees.

In [24], the SH protocol proposed in [3] is extended to handle set-valued data, where each user holds a set of items $\mathbf{v} = \{v_1, \ldots, v_t\} \subseteq \mathcal{V}$. One difficulty in the set-valued data setting is that the length of the itemset each user has is different. To address this challenge, Qin et al [24] proposed a protocol, called LDPMiner, for finding heavy hitters from set-valued data. The main idea of LDPMiner is to pad each user's itemset with dummy items to ensure that it has the fixed length ℓ. Each user randomly samples one item from $\mathbf{v}$ and reports the item using the SH protocol. The estimated frequency of items in LDPMiner is multiplied by ℓ to account for the random sampling procedure.

Bassily et al. [2] and Wang et al. [29] independently proposed a similar protocol that iteratively identifies heavy hitters using a prefix tree. In their protocol, users are randomly split into g disjoint groups. At iteration i, the server receives noisy reports from the users in the i^{th} group, . Each user in the i^{th} group reports the randomized version of the first l_i bits of the encoded item (i.e., a prefix of length l_i), where $l_1 < l_2 \cdots < l_g$. After aggregating the user reports from the i^{th} group, the server identifies frequent prefixes C_i of length l_i and builds the candidate heavy hitter items of length l_{i+1} by concatenating C_i with strings in $\{0,1\}^{l_{i+1}-l_i}$.

Recently, Wang et al. [30] provided a thorough analysis on the "pad-and-sampling-based frequency oracle (PSFO)" and proposed an

LDP solution to the frequent itemset mining problem. Their protocol adaptively chooses between two algorithms based on the size of the domain $|\mathcal{V}|$.

9 CONCLUSION

We proposed a novel collaborative detection system that learns a list of spam-related phone numbers from call records contributed by participating users. Our system makes use of local differential privacy to provide clear privacy guarantees. We evaluated the system on real-world user-reported call records collected by the FTC, and showed that it is possible to learn a phone blacklist in a privacy preserving way using a reasonable overall privacy budget, while at the same time maintaining the utility of the learned blacklist.

ACKNOWLEDGMENTS

We would like to thank the anonymous reviewers for their constructive comments and suggestions on how to improve this paper. This material is based in part upon work supported by the National Science Foundation (NSF) under grants No. 1514035, 1514052, and 1943046. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF.

REFERENCES

[1] Apple Inc. 2016. Apple Differential Privacy Technical Overview. https://images.apple.com/privacy/docs/Differential_Privacy_Overview.pdf.
[2] Raef Bassily, kobbi nissim, Uri Stemmer, and Abhradeep Guha Thakurta. 2017. Practical Locally Private Heavy Hitters. In *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.). Curran Associates, Inc., 2288–2296. http://papers.nips.cc/paper/6823-practical-locally-private-heavy-hitters.pdf
[3] Raef Bassily and Adam Smith. 2015. Local, private, efficient protocols for succinct histograms. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*. ACM, 127–135.
[4] Tara Siegel Bernard. 2018. Yes, It's Bad. Robocalls, and Their Scams, Are Surging. https://www.nytimes.com/2018/05/06/your-money/robocalls-rise-illegal.html.
[5] Giuseppe Bianchi, Lorenzo Bracciale, and Pierpaolo Loreti. 2012. Better Than Nothing Privacy with Bloom Filters: To What Extent? In *Privacy in Statistical Databases*, Josep Domingo-Ferrer and Ilenia Tinnirello (Eds.). Lecture Notes in Computer Science, Vol. 7556. Springer Berlin Heidelberg, 348–363. https://doi.org/10.1007/978-3-642-33627-0_27
[6] John C. Duchi, Michael I. Jordan, and Martin J. Wainwright. 2013. Local Privacy and Statistical Minimax Rates. In *Proceedings of the 2013 IEEE 54th Annual Symposium on Foundations of Computer Science (FOCS '13)*.
[7] Cynthia Dwork. 2006. Differential privacy. In *in ICALP*. Springer, 1–12.
[8] Cynthia Dwork, Moni Naor, Toniann Pitassi, and Guy N. Rothblum. 2010. Differential Privacy Under Continual Observation. In *Proceedings of the Forty-second ACM Symposium on Theory of Computing* (Cambridge, Massachusetts, USA) *(STOC '10)*. ACM, New York, NY, USA, 715–724. https://doi.org/10.1145/1806689.1806787
[9] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. 2014. RAPPOR: Randomized Aggregatable Privacy-Preserving Ordinal Response. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security* (Scottsdale, Arizona, USA) *(CCS '14)*. ACM, New York, NY, USA, 1054–1067. https://doi.org/10.1145/2660267.2660348
[10] Giulia Fanti, Vasyl Pihur, and Úlfar Erlingsson. 2016. Building a RAPPOR with the Unknown: Privacy-Preserving Learning of Associations and Data Dictionaries. *Proceedings on Privacy Enhancing Technologies (PoPETS)* issue 3, 2016 (2016).
[11] FCC. 2020. FCC MANDATES THAT PHONE COMPANIES IMPLEMENT CALLER ID AUTHENTICATION TO COMBAT SPOOFED ROBOCALLS. https://docs.fcc.gov/public/attachments/DOC-363399A1.pdf.
[12] FTC. 2018. Abusive Robocalls and How We Can Stop Them. https://www.ftc.gov/system/files/documents/public_statements/1366628/p034412_commission_testimony_re_abusive_robocalls_senate_04182018.pdf.
[13] FTC. 2018. FTC and FCC to Host Joint Policy Forum and Consumer Expo to Fight the Scourge of Illegal Robocalls. https://www.ftc.gov/news-events/press-releases/2018/03/ftc-fcc-host-joint-policy-forum-consumer-expo-fight- scourge.
[14] FTC. 2019. Do Not Call (DNC) Reported Calls Data. https://www.ftc.gov/site-information/open-government/data-sets/do-not-call-data.
[15] Brian Fung. 2019. Report: Americans got 26.3 billion robocalls last year, up 46 percent from 2017. https://www.washingtonpost.com/technology/2019/01/29/report-americans-got-billion-robocalls-last-year-up-percent/.
[16] Google. 2019. Use caller ID and spam protection. https://support.google.com/phoneapp/answer/3459196?hl=en.
[17] Venkatesan Guruswami. 2004. *List decoding of error-correcting codes: winning thesis of the 2002 ACM doctoral dissertation competition*. Vol. 3282. Springer Science & Business Media.
[18] IRS. 2018. Phone Scams Pose Serious Threat; Remain on IRS 'Dirty Dozen' List of Tax Scams. https://www.irs.gov/newsroom/phone-scams-pose-serious-threat-remain-on-irs-dirty- dozen-list-of-tax-scams.
[19] Peter Kairouz, Keith Bonawitz, and Daniel Ramage. 2016. Discrete Distribution Estimation under Local Privacy. In *International Conference on Machine Learning*. 2436–2444.
[20] Jack P. C. Kleijnen, Ad A. N. Ridder, and Reuven Y. Rubinstein. 2013. *Variance Reduction Techniques in Monte Carlo Methods*. Springer US.
[21] H. Li, X. Xu, C. Liu, T. Ren, K. Wu, X. Cao, W. Zhang, Y. Yu, and D. Song. 2018. A Machine Learning Approach To Prevent Malicious Calls Over Telephony Networks. In *IEEE Symposium on Security and Privacy (SP)*. 561–577.
[22] Jienan Liu, Babak Rahbarinia, Roberto Perdisci, Haitao Du, and Li Su. 2018. Augmenting Telephone Spam Blacklists by Mining Large CDR Datasets. In *Proceedings of the 2018 on Asia Conference on Computer and Communications Security (ASIACCS '18)*.
[23] Sharbani Pandit, Roberto Perdisci, Mustaque Ahamad, and Payas Gupta. 2018. Towards measuring the effectiveness of telephony blacklists. In *Network and Distributed System Security Symposium (NDSS)*.
[24] Zhan Qin, Yin Yang, Ting Yu, Issa Khalil, Xiaokui Xiao, and Kui Ren. 2016. Heavy hitter estimation over set-valued data with local differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 192–203.
[25] Robocall Blocking. 2019. Caller ID, SMS spam blocking and Dialer. https://play.google.com/store/apps/details?id=com.nomorobo&hl=en_US.
[26] TrueCaller. 2019. Caller ID, SMS spam blocking and Dialer. https://play.google.com/store/apps/details?id=com.truecaller&hl=en_US.
[27] Tianhao Wang. 2018. Sample OLH implementation in Python. https://github.com/vvv214/OLH.
[28] Tianhao Wang, Jeremiah Blocki, Ninghui Li, and Somesh Jha. 2017. Locally Differentially Private Protocols for Frequency Estimation. In *26th USENIX Security Symposium (USENIX Security 17)*.
[29] Tianhao Wang, Ninghui Li, and Somesh Jha. 2017. Locally Differentially Private Heavy Hitter Identification. *arXiv preprint arXiv:1708.06674* (2017).
[30] T. Wang, N. Li, and S. Jha. 2018. Locally Differentially Private Frequent Itemset Mining. In *2018 IEEE Symposium on Security and Privacy (SP)*, Vol. 00. 578–594. https://doi.org/10.1109/SP.2018.00035
[31] YouMail. 2019. Stop Robocalls Forever. https://www.youmail.com.

A CLIENT-SERVER SH ALGORITHMS

Algorithms 6 and 7 show the client-server formulation of the SH protocol discussed in Section 3.3.

In order to run the client protocol, each client first needs to know the number of communication channels K that has to be established with the server for sending private reports. Hence, before starting the SH protocol, the server communicates the correct number of channels K to the clients. Notice that the server is the only one who can compute K, since K depends on the number of users contributing to the system at any given time. For T times, in each channel k and round t in $[T]$, the user sends to the server a randomized report $\mathbf{z}^{(t,k)}$, which represents the (encoded) value of v she holds or a special value $\mathbf{0}$ indicating that the user does not hold a value to be reported.

Algorithm 5: $\mathcal{R}_{\mathrm{OLH}}(v,\varepsilon)$: ε-OLH Randomizer

Input: value v, a hash function H, OLH g parameter, privacy budget ε
$x \leftarrow H(v) \,\%\, g$
Sample $y \leftarrow [g] \setminus \{x\}$ uniformly at random.
$w = \begin{cases} x & \text{w.p. } \frac{e^{\varepsilon}}{e^{\varepsilon}+g-1} \\ y & \text{w.p. } \frac{g-1}{e^{\varepsilon}+g-1} \end{cases}$
return w

Algorithm 6: SH-Client($v, \mathcal{H}, T, K, \varepsilon$)

Input: the m-bit string representation $\mathbf{v}$ of the value v to be sent, a fixed list of hash functions $\mathcal{H}$, # of repetitions T, # of channels K, privacy parameter ε

/* sending noisy reports for heavy hitter detection */

for $t = 1$ **to** T **do**
  $H \leftarrow \mathcal{H}[t]$
  foreach *channel* $k \in [K]$ **do**
    if $H(v) = k$ **then**
      $\mathbf{x} = \text{Enc}(v)$
    else
      $\mathbf{x} = \mathbf{0}$
    $\mathbf{z}^{(t,k)} \leftarrow \mathcal{R}_{\text{bas}}\left(\mathbf{x}, \frac{\varepsilon}{2T+1}\right)$
    Send $\mathbf{z}^{(t,k)}$ to the server on channel k
  /* sending noisy report for heavy hitter frequency estimation */
  $\mathbf{w} \leftarrow \mathcal{R}_{\text{bas}}\left(\mathbf{v}, \frac{\varepsilon}{2T+1}\right)$
  Send $\mathbf{w}$ to the server

The choice of sending the randomized report associated with $\text{Enc}(v)$ (or with $\mathbf{0}$) depends on whether the channel identifier k matches the value returned by the hash function H applied on v. H belongs to a pairwise independent hash function family $\mathcal{H}$, publicly available and accessible to all the clients as part of the client-side protocol configuration.

In each round of the protocol, a different hash function is employed to minimize the probability of collisions among different heavy hitters. Notice that, except for a single channel in which the client sends the private report obtained from $\mathcal{R}_{\text{bas}}$ for a value v, in all the other channels the client sends randomized reports for the special value $\mathbf{0}$ (see Algorithm 6).

On the server side, the server receives in each channel k the private reports $\mathbf{z}^{(t,k)}$ sent by users for each specific run t. In each round and for each channel, the server aggregates the randomized reports to reconstruct the codeword $\mathbf{y}$ whose hash of the original value v corresponds to channel k. Hence, the decoded value $\hat{v}$, if correctly reconstructed, should represent the private information sent by (a non-negligible number of) users in the k-th channel in a specific run of the SH protocol. The set of reconstructed values is stored in the set of potential heavy hitters Γ. Due to noisy reports, some values in Γ may not be heavy hitters.

To filter out possible false positives, similarly to the previous phase the server collects noisy reports $\mathbf{w}_j$ from users and aggregates them in a single bitstring $\overline{\mathbf{w}}$. For each reconstructed value $\hat{v}$ in Γ, its frequency $f(\hat{v})$ is estimated using a frequency oracle (FO) function. If the computed estimate $\hat{f}(\hat{v})$ is less than a threshold η, then $\hat{v}$ is removed from Γ. After this filtering phase, the server can then return the set of detected heavy hitters.

The threshold η plays a crucial role in the heavy hitter detection:

$$\eta = \frac{2T+1}{\varepsilon}\sqrt{\frac{\log(d)\log(1/\beta)}{n}} \tag{3}$$

where β [3] is a parameter related to the confidence the server has on the heavy hitters it has detected. The same parameter β also influences the number of protocol rounds, T [3]. The server-side protocol pseudo-code is represented in Algorithm 7, whereas Algorithm 5 refers to the discussion in Section 3.4.

Algorithm 7: SH-Server(T, K, FO)

Input: # of repetition T, # of channels K, a frequency oracle FO, a threshold η
Output: list of heavy hitters Γ

/* detecting heavy hitters */

$\Gamma \leftarrow \emptyset$
for $t = 1$ **to** T **do**
  foreach *channel* $k \in [K]$ **do**
    foreach *user* $j \in [n]$ **do**
      $\mathbf{z}_j \leftarrow \mathbf{z}^{(t,k)}$ value received from user j on channel k;
    $\overline{\mathbf{z}} = \frac{1}{n}\sum_{j=1}^{n}\mathbf{z}_j$
    for $i = 1$ **to** m **do**
      $\mathbf{y}[i] \leftarrow \begin{cases} \frac{1}{\sqrt{m}} & \text{if } \overline{\mathbf{z}}[i] \geq 0 \\ -\frac{1}{\sqrt{m}} & \text{otherwise.} \end{cases}$
    $\hat{v} \leftarrow \text{Dec}(\mathbf{y})$
    if $\hat{v} \notin \Gamma$ **then** add $\hat{v}$ to Γ
/* filtering out false positives */
foreach *user* $j \in [n]$ **do**
  $\mathbf{w}_j \leftarrow \mathbf{w}$ value received from user j
$\overline{\mathbf{w}} = \frac{1}{n}\sum_{j=1}^{n}\mathbf{w}_j$
foreach $\hat{v} \in \Gamma$ **do**
  $\hat{f}(\hat{v}) \leftarrow$ estimate the frequency of $\hat{v}$ using $\text{FO}(\overline{\mathbf{w}})$
  if $\hat{f}(\hat{v}) < \eta$ **then** remove $\hat{v}$ from Γ
return $\{(v, \hat{f}(v)) : v \in \Gamma\}$

B ANALYSIS OF THE BASIC RANDOMIZER

We first show that the frequency estimate $\hat{f}(v)$ obtained using the basic randomizer is unbiased:

$$\begin{aligned}
\mathbb{E}[\hat{f}(v)] &= \mathbb{E}\Big[\frac{1}{n}\sum_{j=1}^{n}\mathbf{w}_j^\intercal \mathbf{x}_v\Big] \\
&= \frac{1}{n}\left\{\sum_{j:v_j=v}\mathbb{E}[\mathbf{w}_j^\intercal \mathbf{x}_v] + \sum_{j:v_j\neq v}\mathbb{E}[\mathbf{w}_j^\intercal \mathbf{x}_v]\right\} \\
&= \frac{1}{n}\sum_{j:v_j=v}\mathbf{x}_j^\intercal \mathbf{x}_v \\
&= \frac{1}{n}\sum_{j:v_j=v}\|\mathbf{x}_j\|^2 = \frac{\sum_{j:v_j=v}1}{n} = f(v),
\end{aligned}$$

where $\mathbf{x}_v = \mathbf{c}(v)$ denotes the encoding of item v.

We next calculate the variance of the estimate given by the basic randomizer. Let $\mathbf{r} = (r_1,\ldots,r_j,\ldots,r_n)$ be a vector of random bits chosen by user j, where $r_j \in [m]$. By the law of total variance, for an item $v \in \mathcal{V}$, the variance of estimate $\hat{f}(v)$ is

$$\begin{aligned}
\text{Var}(\hat{f}(v)) &= \mathbb{E}[\text{Var}(\hat{f}(v) \mid \mathbf{r})] + \text{Var}(\mathbb{E}[\hat{f}(v) \mid \mathbf{r}]) \\
&= \frac{1}{n}\{(c^2-1)f(v) + (1-f(v))c^2\} \\
&= \frac{c^2 - f(v)}{n},
\end{aligned}$$

where we have

$$\begin{aligned}
&\text{Var}(\hat{f}(v) \mid \mathbf{r}) \\
&= \text{Var}\Big(\frac{1}{n}\sum_{j=1}^{n} w[r_j]\cdot \mathbf{x}_v[r_j] \mid \mathbf{r}\Big) \\
&= \frac{1}{n^2}\text{Var}\Big(\sum_{j=1}^{n} w[r_j] \mid r_j)\mathbf{x}_v[r_j]^2 \\
&= \frac{\mathbf{x}_v[r_j]^2}{n^2}\left\{\sum_{j:v_j=v}\text{Var}(w[r_j] \mid r_j) + \sum_{j:v_j\neq v}\text{Var}(w[r_j] \mid r_j)\right\} \\
&= \frac{\mathbf{x}_v[r_j]^2}{n^2}\Big\{nf(v)(c^2m^2x[r_j]^2 - m^2x[r_j]^2) \\
&\qquad + n(1-f(v))(c^2m - 0^2)\Big\}
\end{aligned}$$

and

$$\mathbb{E}[\hat{f}(v) \mid \mathbf{r}] = \mathbb{E}\left[\frac{1}{n}\sum_{j=1}^{n} w[r_j]\cdot \mathbf{x}_v[r_j] \mid \mathbf{r}\right]$$
$$= \frac{1}{n}\left\{\sum_{j:v_j=v} \mathbb{E}\left[w[r_j]\cdot \mathbf{x}_v[r_j] \mid r_j\right] + \sum_{j:v_j\neq v} 0\right\}$$
$$= \frac{1}{n}\sum_{j:v_j=v} m\cdot x[r_j]^2.$$

C ANALYSIS OF AREA CODE BUCKETIZATION

Let us first analyze how the probability that the server C correctly reconstructs a reported phone number depends on the size of the phone numbers space and the number of reports. In this simplified analysis, we will assume no noise is added to the data transmitted from the clients to the server. In other words, we will follow the fundamental steps of the SH protocol in Algorithm 6, but pretend that the randomizer (line 8) always returns the *true value* of one randomly selected bit.

Let us now consider a domain $\mathcal{V}$, in which each value can be represented using l bits (i.e., $|\mathcal{V}| = 2^l$). Also, let us consider a value $v \in \mathcal{V}$ transmitted by n clients. For the sake of this simplified analysis, on the server side we can view v as a sequence of l different *bins* that are initially empty, and the bits sent by the clients as *balls*. According to the SH protocol for heavy hitter detection, each client transmits only one bit, and therefore the server receives n balls. To correctly reconstruct the value v, at least one ball must fill each bin. As reported in [5], the number of non-empty bins resulting from randomly inserting n balls into l bins has the following probability distribution:

$$U_{l,n}(b) = \frac{\left\{{n \atop b}\right\}\binom{l}{b}b!}{l^n}, \quad \forall b \in \{1,\ldots,l\} \tag{4}$$

where $\left\{{n \atop b}\right\}$ is a Stirling number of the second kind, which expresses the number of ways to partition a set of n elements into b non-empty subsets. The numerator in the equation expresses the number of ways in which n balls fall exactly in b bins out of l available ones. Therefore, for $b = l, U_{l,n} = \frac{\left\{{n \atop l}\right\}l!}{l^n}$ gives us the probability that all bins will be filled.

Intuitively, the larger l, the larger n must be to fill the bins. For instance, in this simplified analysis, the 34-bit representation of a 10-digit phone number p would need to be reported by at least 170 users, for it to have about an 80% probability of being reconstructed at the server side. In reality, the additional noise and the error-correction encoding in the SH protocols further complicate the relationship between l and n. However, it is clear that reducing l also reduces the number of reports above which heavy hitters can be detected with high probability. This motivates our choice of *bucketizing* phone numbers by grouping them based on area codes, and by running a separate instance of the SH protocol per bucket, as only seven digits need to be reported by the SH protocol for each phone number in a bucket. Following the above analysis, 111 reports are sufficient to reconstruct 24-bit values (needed to represent 7-digit numbers) with 80% probability, which equates to about a 34.7% reduction in the number of reports to be received by the server.

As outlined in Section 4.1, Equation 4 can also be used by the server for deciding if the clients that have a report to be sent within a given bucket (i.e., if they need to report a caller ID within a given prefix) should actually send the report (using LDP) or not. Considering 24 bits per phone number, as above, and assuming all clients in the same bucket intend to report the same 7-digit phone number, all buckets receiving less than 84 reports can be easily ignored, because the server will have less than 50% probability of correctly reconstructing a heavy hitter in those buckets. This probability is even lower in practice, since each bucket will likely receive reports about different phone numbers. Instructing clients that intend to send a report to "low density" buckets to stop doing so will prevent running the LDP protocol in vain. Thus, those clients can avoid wasting their privacy budget for those specific LDP protocol runs.

Another benefit of grouping phone numbers by area code is that some spam campaigns tend to use numbers with specific area codes. Figure 8 visually shows this tendency.

Figure 2 shows a more comprehensive view of how the relative frequency of phone numbers in the FTC data is amplified when bucketization is used. Specifically, each vertical line represents the frequency of caller IDs appearing in the FTC complaints dataset. The figure on the left shows the occurrence frequency of phone numbers relative to all complaints received in one day, whereas the figure on the right shows how their relative frequency changes after bucketization (notice the different y-axis scales for the two graphs). The take away from this analysis is that bucketization results in the amplification of the relative frequency of some heavy hitter caller IDs and, hence, in the variance reduction of frequency estimates (see Equation 2), thus increasing the likelihood that heavy hitters will be correctly reconstructed and detected by the server.

D ANALYSIS OF EXTENDED RANDOMIZER

While the frequency estimate $\hat{f}(v)$ of an item $v \in \mathcal{V}$ computed from noisy reports generated using the basic randomizer (in line 10 of Algorithm 6) is unbiased, its variance is often quite large in practice, and this could lead to low accuracy in heavy-hitter detection. Inspired by the antithetic variates technique in Monte Carlo methods [20], we extend the basic randomizer and introduce a new randomizer $\mathcal{R}_{\text{ext}}$ which yields lower variance. The extended randomizer is described in Algorithm 4.

The main difference between the randomizers is in the number of different values each user can report. Notice that $z_r \in \{c\sqrt{m}, 0, -c\sqrt{m}\}$ in the extended randomizer, while $z_r \in \{c\sqrt{m}, -c\sqrt{m}\}$ in the basic randomizer. The idea behind this modification is that the sum of contributions from users who don't have item v to the estimate $\hat{f}(v)$ is non-zero in practice, due to the variance, while in expectation they should cancel out.

The following lemma shows that the extended randomizer provides an unbiased estimate of (encoded) item $\mathbf{x}$.

Lemma 1. *Let $p = \frac{e^\epsilon}{e^\epsilon+2}$, $q = \theta = \frac{1}{e^\epsilon+2}$, and $c = \frac{e^\epsilon+2}{e^\epsilon-1}$. The extended randomizer $\mathcal{R}_{\text{ext}}$ has the following properties:*

(i) For every $\mathbf{x} \in \{-1/\sqrt{m}, 1/\sqrt{m}\} \cup \{\mathbf{0}\}$, $\mathbb{E}[\mathcal{R}_{\text{ext}}(\mathbf{x})] = \mathbf{x}$.

(ii) $\mathcal{R}_{\text{ext}}$ satisfies ϵ-LDP for every $r \in [m]$.

The proof of the above lemma is provided in Appendix D.1.

Given a set of noisy reports $\mathbf{z}_1, \ldots, \mathbf{z}_n$ generated by the extended randomizer, the randomizer yields an unbiased estimate of frequency with smaller variance than the basic randomizer. The following lemma formalizes this discussion, whose proof appears in Appendix D.1.

Lemma 2. *Let $v^* \in \mathcal{V}$ be an item and $\{\mathbf{w}_i\}_{i=1}^n$ be the noisy reports. The frequency estimate $\hat{f}(v^*) = \frac{1}{n}\sum_{j=1}^n \mathbf{w}_j^\intercal \mathbf{c}(v^*)$ has the following properties:*

(i) $\mathbb{E}[\hat{f}(v^*)] = f(v^*)$ *and*

(ii) $\mathrm{Var}(\hat{f}(v^*)) = \frac{1}{n}\left\{f(v^*)\cdot(c^2(p+q)-1)+(1-f(v^*))\cdot 2c^2\theta\right\}$,

where $f(v^*)$ is the true frequency of v^*.

Two important remarks are in order. First, the extended randomizer $\mathcal{R}_{\text{ext}}$ reduces to the basic randomizer $\mathcal{R}_{\text{bas}}$ if we set $c = \frac{e^\varepsilon+1}{e^\varepsilon-1}$, $p = \frac{e^\varepsilon}{e^\varepsilon+1}$, $q = \frac{1}{e^\varepsilon+1}$, and $\theta = \frac{1}{2}$. Second, the above shows that the variance of frequency estimate of an item $v^* \in \mathcal{V}$ can be written as a linear combination of two terms: $c^2(p+q)$ and $2c^2\theta$. While we wish to find optimal parameter values for c,p,q, and θ that minimize the variance, this is not possible because $f(v^*)$ is unknown. Instead, we minimize the maximum of those two terms under ε-LDP constraints:

$$\begin{aligned}
\underset{c,p,q,\theta}{\text{minimize}} \quad & \max\{c^2(p+q), 2c^2\theta\} \\
\text{subject to} \quad & c(p-q)=1 \\
& p-e^\epsilon\theta\le 0,\ -p+e^{-\epsilon}\theta\le 0 \\
& q-e^\epsilon\theta\le 0,\ -q+e^{-\epsilon}\theta\le 0 \\
& p-e^\epsilon q\le 0,\ -p-e^{-\epsilon}q\le 0 \\
& 1-p-q-e^\epsilon(1-2\theta)\le 0 \\
& -1+p+q+e^{-\epsilon}(1-2\theta)\le 0 \\
& 0\le p+q\le 1,\ 0\le\theta\le\frac{1}{2}.
\end{aligned}$$

Solving the above optimization problem gives the following solution:

$$p=\frac{e^\varepsilon}{e^\varepsilon+2},\quad q=\theta=\frac{1}{e^\varepsilon+2},\quad c=\frac{e^\varepsilon+2}{e^\varepsilon-1}. \tag{5}$$

Proposition 1. *The frequency estimate $\hat{f}(v)$ of an item v given by $\mathcal{R}_{\text{ext}}$ has lower variance than that given by $\mathcal{R}_{\text{bas}}$ if*

$$\varepsilon \ge \ln\frac{a+\sqrt{9a^2-20a+12}}{1-a},$$

where $a = f(v)$, i.e., the true frequency of v.

The proof of the above proposition is simple and given in Appendix D.1.

Theorem 1. *Algorithm 3 satisfies black$(\varepsilon_{\text{HH}}+\varepsilon_{\text{OLH}})$-differential privacy.*

The proof of Theorem 1 follows from [3, Theorem 3.4] and is included in the Appendix D.1 for completeness.

D.1 Proofs for Extended Randomizer

Lemma 1. *Let $p=\frac{e^\epsilon}{e^\epsilon+2}$, $q=\theta=\frac{1}{e^\epsilon+2}$, and $c=\frac{e^\epsilon+2}{e^\epsilon-1}$. The extended randomizer $\mathcal{R}_{\text{ext}}$ has the following properties:*

(ii) For every $\mathbf{x}\in\{-1/\sqrt{m},1/\sqrt{m}\}\cup\{\mathbf{0}\}$, $\mathbb{E}[\mathcal{R}_{\text{ext}}(\mathbf{x})]=\mathbf{x}$.

(ii) $\mathcal{R}_{\text{ext}}$ satisfies ϵ-LDP for every $r\in[m]$.

Proof. Consider an item $v^*\in\mathcal{V}$ on a channel $k\in[K]$ and a hash function $H:\mathcal{V}\to[K]$. For users j with $H(v_j)=k$, we have

$$\begin{aligned}
\mathbb{E}[\mathcal{R}_{\text{ext}}(\mathbf{x}_j)] &= \mathbb{E}[\mathbf{z}_j] = \mathbb{E}\big[\mathbb{E}[\mathbf{z}_j \mid r_j]\big] \\
&= \frac{1}{m}(\mathbb{E}[\mathbf{z}_j[1]],\ldots,\mathbb{E}[\mathbf{z}_j[m]])^\intercal \\
&= \frac{1}{m}(cm(p-q)\mathbf{x}_j[1],\ldots,cm(p-q)\mathbf{x}_j[m])^\intercal \\
&= c(p-q+)\mathbf{x}_j.
\end{aligned}$$

Since $c(p-q)=1$, we have $\mathbb{E}[\mathcal{R}_{\text{ext}}(\mathbf{x}_j)]=\mathbf{x}_j$. For those users with $H(v_j)\neq k$, their encoded item $\mathbf{x}_j=\mathrm{Enc}(v_j)=\mathbf{0}$, and we have

$$\mathbb{E}[\mathbf{z}_j]=\frac{1}{m}(c\sqrt{m}\theta-c\sqrt{m}\theta,\ldots,c\sqrt{m}\theta-c\sqrt{m}\theta)=\mathbf{0}=\mathbf{x}_j.$$

This completes the proof of the unbiasedness of $\mathcal{R}_{\text{ext}}$.

Next, we prove ϵ-LDP of the extended randomizer. Let v_1 and v_2 be two arbitrary items in $\mathcal{V}$ and $\mathbf{x}_1$ and $\mathbf{x}_2$ be their encodings in $\{-1/\sqrt{m},1/\sqrt{m}\}^m\cup\{\mathbf{0}\}$, respectively. For any $z_r\in\{cmx_r,0,-cmx_r\}$, we have

$$\frac{\Pr[z_r\mid\mathbf{x}_1,r]}{\Pr[z_r\mid\mathbf{x}_2,r]}\le\max\left\{\frac{p}{\theta},\frac{1-2\theta}{1-p-q}\right\}=e^\epsilon.$$

Similarly,

$$\frac{\Pr[z_r\mid\mathbf{x}_1,r]}{\Pr[z_r\mid\mathbf{x}_2,r]}\ge\min\left\{\frac{1-p-q}{\theta},\frac{\theta}{p}\right\}=e^{-\epsilon}.$$

□

Lemma 2. *Let $v^*\in\mathcal{V}$ be an item and $\{\mathbf{w}_i\}_{i=1}^n$ be the noisy reports. The frequency estimate $\hat{f}(v^*)=\frac{1}{n}\sum_{j=1}^n\mathbf{w}_j^\intercal\mathbf{c}(v^*)$ has the following properties:*

(ii) $\mathbb{E}[\hat{f}(v^*)]=f(v^*)$ *and*

(ii) $\mathrm{Var}(\hat{f}(v^*))=\frac{1}{n}\left\{f(v^*)\cdot(c^2(p+q)-1)+(1-f(v^*))\cdot 2c^2\theta\right\}$,

where $f(v^*)$ is the true frequency of v^*.

Proof. Let $\mathbf{x}^*=\mathbf{c}(v^*)$. We first prove the unbiasedness property. Since $\mathbf{w}_j$ is an unbiased estimate of $\mathbf{x}_j$ (i.e., $\mathbb{E}[\mathbf{w}_j]=\mathbf{x}_j$), it is easy to see that $\hat{f}(v^*)$ is also unbiased.

$$\begin{aligned}
\mathbb{E}[\hat{f}(v^*)] &= \mathbb{E}\left[\frac{1}{n}\sum_{j=1}^n\mathbf{w}_j^\intercal\mathbf{c}(v^*)\right] \\
&= \frac{1}{n}\Big\{\sum_{j:v_j=v^*}\mathbb{E}[\mathbf{w}_j^\intercal\mathbf{x}_j]+\sum_{j:v_j\neq v^*}\mathbb{E}[\mathbf{w}_j^\intercal\mathbf{x}^*]\Big\} \\
&= \frac{1}{n}\sum_{j:v_j=v^*}\|\mathbf{x}_j\|^2=\frac{\sum_{j:v_j=v^*}1}{n}=f(v^*).
\end{aligned}$$

To compute the variance $\mathrm{Var}(\hat{f}(v^*))$, we condition on random bits chosen by users. Let $\mathbf{r}=(r_1,\ldots,r_j,\ldots,r_n)$ be a vector, where $r_j\in[m]$ represents the random bit chosen by user j. By the law of total variance,

$$\begin{aligned}
\mathrm{Var}(\hat{f}(v^*)) &= \mathbb{E}\big[\mathrm{Var}(\hat{f}(v^*)\mid\mathbf{r})\big]+\mathrm{Var}(\mathbb{E}[\hat{f}(v^*)\mid\mathbf{r}]) \\
&= \frac{1}{n^2}\mathbb{E}\big[\mathrm{Var}(\sum_{j=1}^n\mathbf{w}[r_j]\cdot\mathbf{x}^*[r_j]\mid r_j)\big] \\
&\quad+\frac{1}{n^2}\mathrm{Var}\left(\mathbb{E}\Big[\sum_{j=1}^n\mathbf{w}[r_j]\cdot\mathbf{x}^*[r_j]\mid r_j\Big]\right) \\
&= \frac{1}{n^2}(\mathbb{E}[A]+\mathrm{Var}(B)).
\end{aligned}\tag{6}$$

The first term is

$$\begin{aligned}
A &= \sum_{j=1}^n\mathrm{Var}(\mathbf{w}[r_j]\mid r_j)\cdot\mathbf{x}^*[r_j]^2 \\
&= \sum_{j:v_j=v^*}\mathrm{Var}(\mathbf{w}[r_j])\cdot\mathbf{x}^*[r_j]^2+\sum_{j:v_j\neq v^*}\mathrm{Var}(\mathbf{w}[r_j])\cdot\mathbf{x}^*[r_j]^2 \\
&= \sum_{j:v_j=v^*}(c^2m^2\mathbf{x}[r_j]^2(p+q)-m^2\mathbf{x}[r_j]^2)\cdot\mathbf{x}^*[r_j]^2 \\
&\quad+\sum_{j:v_j\neq v^*}2c^2m\theta\cdot\mathbf{x}^*[r_j]^2,
\end{aligned}$$

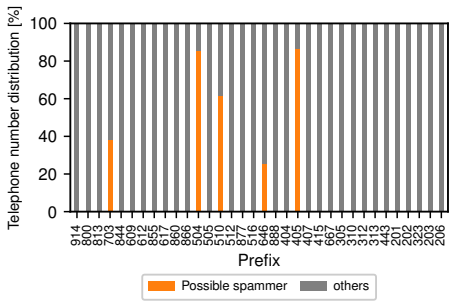

Figure 8: Telephone number distribution of a sample day. Striped bars are related to phone numbers that received more than 100 complaints.

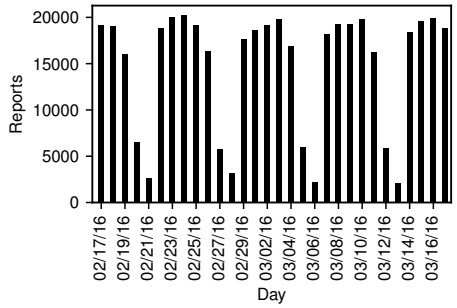

Figure 9: Number of daily complaints received between Feb. 17th and Mar. 17th.

and

$$\begin{aligned}\mathbb{E}[A] &= nf(v^*)\cdot\frac{1}{m}\left(c^2m^2(p+q)\sum_{i=1}^{m}\mathbf{x}_j[i]^4 - m^2\sum_{i=1}^{m}\mathbf{x}_j[i]^4\right)\\ &\quad + n(1-f(v^*)\cdot\frac{1}{m}\cdot 2c^2m\theta\sum_{i=1}^{m}\mathbf{x}^*[i]^2\\ &= nf(v^*)\{c^2(p+q)-1\}+n(1-f(v^*))\cdot 2c^2\theta. \end{aligned} \tag{7}$$

The second term is

$$\begin{aligned}B &= \sum_{j=1}^{n}\mathbb{E}\left[\mathbf{w}[r_j]\cdot\mathbf{c}(v^*)[r_j]\mid r_j\right]\\ &= \sum_{j:v_j=v^*}\mathbb{E}\left[\mathbf{w}[r_j]\right]\cdot\mathbf{x}^*[r_j] + \sum_{j:v_j\neq v^*}\mathbb{E}\left[\mathbf{w}[r_j]\right]\cdot\mathbf{x}^*[r_j]\\ &= \sum_{j:v_j=v^*} cm\mathbf{x}[r_j]^2(p-q) = nf(v^*)\cdot cm\mathbf{x}[r_j]^2(p-q),\end{aligned}$$

and

$$\mathrm{Var}(B) = n^2f(v^*)^2c^2m^2(p-q)^2\mathrm{Var}(\mathbf{x}[r_j]^4) = 0. \tag{8}$$

Plugging (7) and (8) into (6) gives the claimed result. □

Theorem 1. *Algorithm 3 satisfies black$(\varepsilon_{\mathrm{HH}}+\varepsilon_{\mathrm{OLH}})$-differential privacy.*

Proof. Fix a user j and two items $v_j, v_j' \in \mathcal{V}$ held by j. Observe that, in Algorithm 2, for any fixed sequence $\mathcal{H}$ of hash functions each user j makes a report to KT channels, and each report is generated independently. Among K channels, there exists only one channel on which user j sends the noisy report of her true item v_j. On the remaining $K-1$, user j sends the noisy report of a special item $\mathbf{0}$. Thus, changing the user's item from v_j to v_j' changes the distribution of user's report on at most $2T$ channels, and on each channel the ratio of two distributions is bounded by $\exp(\frac{\varepsilon_{\mathrm{HH}}}{2T})$ by the ε-LDP property of the extended randomizer. Since user's reports over separate channels are independent, the corresponding ratio over all the KT channels are bounded by $\exp(\frac{2T\varepsilon_{\mathrm{HH}}}{2T}) = \exp(\varepsilon_{\mathrm{HH}})$. For frequency oracle, user j generates another report using OLH, which satisfies $\varepsilon_{\mathrm{OLH}}$-LDP, and sends it to the server. Again, by independence of user's reports for heavy hitter detection and frequency oracle, the ratio of user's output distribution is bounded by $\exp(\varepsilon_{\mathrm{HH}})\cdot\exp(\varepsilon_{\mathrm{OLH}}) = \exp(\varepsilon_{\mathrm{HH}}+\varepsilon_{\mathrm{OLH}})$. This completes the proof. □

Proposition 1. *The frequency estimate $\hat{f}(v)$ of an item v given by $\mathcal{R}_{\mathrm{ext}}$ has lower variance than that given by $\mathcal{R}_{\mathrm{bas}}$ if*

$$\varepsilon \geq \ln\frac{a+\sqrt{9a^2-20a+12}}{1-a},$$

where $a = f(v)$, i.e., the true frequency of v.

Proof. Using the parameters in (5), we get the variance of frequency estimate $\hat{f}(v)$ given by the extended randomizer:

$$\begin{aligned}\mathrm{Var}(\hat{f}(v)) &= \frac{1}{n}\{f(v)\cdot(c^2(p+q)-1)+(1-f(v))\cdot 2c^2\theta\}\\ &= \frac{1}{n}\left\{f(v)\cdot\left(\frac{3(e^\varepsilon-1)}{(e^\varepsilon-1)^2}\right)+\frac{2(e^\varepsilon+2)}{(e^\varepsilon-1)^2}\right\}.\end{aligned} \tag{9}$$

The variance of $\hat{f}(v)$ for the basic randomizer is

$$\mathrm{Var}(\hat{f}(v)) = \frac{1}{n}\left\{\left(\frac{e^\varepsilon+1}{e^\varepsilon-1}\right)^2 - f(v)\right\}. \tag{10}$$

To find the values of ε such that (9)$\leq$ (10), we set

$$f(v)\left(\frac{3(e^\varepsilon-1)}{(e^\varepsilon-1)^2}\right)+\frac{2(e^\varepsilon+2)}{(e^\varepsilon-1)^2} \leq \left(\frac{e^\varepsilon+1}{e^\varepsilon-1}\right)^2 - f(v).$$

Simplifying and rearranging the terms, the above inequality reduces to

$$(f(v)-1)e^{2\varepsilon}+f(v)e^\varepsilon+(3-2f(v)) \leq 0. \tag{11}$$

Substituting $t = e^\varepsilon$ and $a = f(v)$, we see that the l.h.s. term of the above inequality is a simple quadratic function $g(t) = (a-1)t^2+at+(3-2a)$, where $0 \leq a < 1$. The quadratic function g is concave and has zeros at

$$t = \frac{a\pm\sqrt{a^2-4(a-1)(3-2a)}}{1-a}.$$

Thus, the inequality (11) is satisfied when

$$\varepsilon \geq \ln\frac{a+\sqrt{9a^2-20a+12}}{1-a}.$$

□

E DATASET PROPERTIES

Figure 8 shows the relative frequency of phone numbers that make more than one hundred calls in a day, compared to the total number of calls made by all phone numbers reported within the same area code. These graphs are computed based on phone numbers extracted from unwanted call reports from US residents to the FTC (more details about the FTC data we use are provided in Section 5). Each vertical bar indicates a different area code prefix. The striped portion of the bars indicates the relative fraction of complaints related to numbers that were complained about more than one hundred times in a day. The figure is related to a sample day worth of reports. As can be seen, phone numbers with more than one hundred complaints appear only in a limited number of prefixes. Their relative occurrence frequency is high in their respective area codes, whereas it would be diluted if we considered all 10-digit numbers in just one bucket.

Figure 9 depicts the number of valid reports received each day, showing a weekly pattern in which a much lower number of complaints is received around the weekends. Figure 10 shows the distribution of the number of complaints per caller ID. Specifically, the x-axis lists the number of complaints, and the y-axis show how many phone numbers have received x complaints in a single day, throughout the entire period of observation included in the dataset. It is easy to see that the vast majority of phone numbers received a single daily complaint, but there also exist many phone numbers that received hundreds of complaints in a single day.

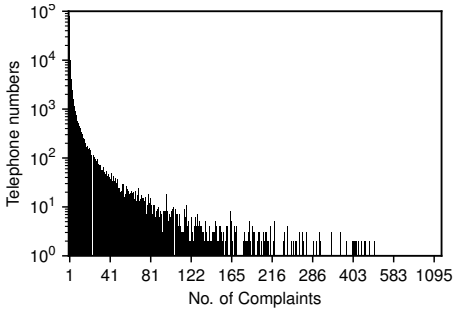

Figure 10: Distribution of daily complaints per caller ID.