Efficient Private Statistics with Succinct Sketches

Luca Melis, George Danezis, and Emiliano De Cristofaro
Department of Computer Science, University College London
{luca.melis.14, g.danezis, e.decristofaro}@ucl.ac.uk

***Abstract*—Large-scale collection of contextual information is often essential in order to gather statistics, train machine learning models, and extract knowledge from data. The ability to do so in a *privacy-preserving* way – i.e., without collecting fine-grained user data – enables a number of additional computational scenarios that would be hard, or outright impossible, to realize without strong privacy guarantees. In this paper, we present the design and implementation of practical techniques for privately gathering statistics from large data streams. We build on efficient cryptographic protocols for private aggregation and on data structures for succinct data representation, namely, Count-Min Sketch and Count Sketch. These allow us to reduce the communication and computation complexity incurred by each data source (e.g., end-users) from linear to logarithmic in the size of their input, while introducing a parametrized upper-bounded error that does not compromise the quality of the statistics. We then show how to use our techniques, efficiently, to instantiate real-world privacy-friendly systems, supporting recommendations for media streaming services, prediction of user locations, and computation of median statistics for Tor hidden services.**

I. Introduction

The increasing amount of contextual information collected by multitudes of always-on, always-connected devices makes it increasingly possible to extract value and knowledge from statistical data. For instance, Google analyzes GPS locations reported by mobile devices to calculate the speed along a road and generate live traffic maps (Google Traffic), and search data to estimate and predict flu activity (Google Flu Trends). Alas, the large-scale collection of user data raises serious privacy, confidentiality, and liability concerns. This motivates the need for efficient and scalable techniques allowing providers to *privately* gather statistics, and to use such statistics to train models and facilitate predictions. Our work is actually inspired by a few real-world problems:

P1 Online streaming services routinely collect statistics about videos watched by their users, and provide them with personalized suggestions, typically, using recommender systems. In particular, we will focus on recommendations for BBC's iPlayer [1], an online platform offering free streaming of TV programs.

P2 Urban planning committees, as well as mass transport operators, are keen on gathering statistics about movements and commuting paths, aiming to improve transportation services and predict future trends, e.g., to respond to anomalies and disruptions on short notice [57, 60].

P3 The Tor network [28] needs to collect traffic statistics such as the number of, and traffic generated by, hidden services, in order to fine tune design decisions and convince their funders of the value of the network [32].

In general, we are interested in scenarios where providers need to train models based on aggregate statistics gathered from many data sources, and our goal is to do so without disclosing fine-grained information about single sources. In theory, we could turn to existing cryptographic protocols for privacy-friendly aggregation: using homomorphic encryption or secret sharing untrusted aggregators can collect encrypted readings but only decrypt the sum [9, 13, 15, 33, 47, 61]. However, these tools require each data source to perform a number of cryptographic operations, and transmit a number of ciphertexts, *linear* in the size of their input, which makes them impractical when sources contribute large streams. For instance, in scenario P1, we need to collect distributions of "co-views" (i.e., pairs of videos watched by the same user) in order to perform recommendations based on K-Nearest Neighbor (KNN) algorithms [25]: even when only hundreds of programs are available, each user would have to encrypt and transmit a matrix of hundreds of thousands of values.

Also, differential privacy could be used to let aggregators add noise to datasets so that other parties may perform statistical queries while the probability of identifying single records is minimized [23]. However, differential privacy alone would not protect the privacy of single data sources w.r.t. the aggregators themselves. Although recent work such as RAPPOR [34] supports, via input perturbation, differentially-private statistics collection with an untrusted aggregator, it actually requires millions of users in order to obtain reasonably accurate answers.

Our insight is to combine privacy-preserving aggregation with data structures supporting succinct data representation, namely, *Count-Min Sketch* [22] and *Count Sketch* [16] (introduced in Section II-B). Private aggregation is performed over the sketches, rather than the raw inputs. Despite an upper-bounded error in the aggregate is introduced, this allows us to reduce communication and computational complexity (for the cryptographic operations) from *linear* to *logarithmic* in the size of the inputs. We then use the resulting private statistics tools to instantiate protocols and build systems addressing applications P1–P3 discussed above, where the error does not affect the overall quality of the computation.

More precisely, in Section III, we present a privacy-preserving recommender system allowing online streaming services like BBC's iPlayer to support recommendations without

NDSS '16, 21-24 February 2016, San Diego, CA, USA

http://dx.doi.org/10.14722/ndss.2016.23175

tracking their users. Users' browsers encrypt and transmit a succinct representation of the co-view matrix (i.e., pairs of videos they have watched) so that the BBC can only decrypt the aggregate matrix (i.e., how many users have watched a given pair). This is broadcast back to the users and used to derive recommendations. Next, in Section IV, we introduce an Android application enabling users to report to a service provider their locations over time, in a privacy-preserving way, i.e., so that only aggregate statistics are disclosed. We then show that these can be used to train a model geared to predict future movements. Finally, in Section V, we build a system for privately computing statistics of Tor hidden services, aiming to address the conflict between the importance to collect (and publish) such statistics and the risk of harming the privacy of individual Tor users. This addresses an open problem raised by the Tor Project [39]. We show how to estimate median statistics by collecting an encrypted frequency distribution of the statistics across all Hidden Services Directories (HSDir).

We also discuss real-world deployment and present full-blown implementations of our techniques, in JavaScript, Android, and/or Python. Our design makes it extremely easy for anyone to integrate our techniques – as simple as installing a package from a public repository. User-side deployment is transparent too, as client-side code can run in the browser (in JavaScript), thus requiring no additional software to be installed or technical understanding of the cryptographic layer.

Our techniques are not limited to one particular model: on the contrary, we can support different trust, robustness, and deployment models. Although our three applications all gather statistics via private sketch aggregation, they do differ in a few key characteristics. The privacy-friendly recommendation and location prediction systems (cf. Section III–IV) build atop a privacy-preserving aggregation scheme where private keys sum up to zero [47, 61], and use the aggregator itself as a bulletin board to distribute users' public keys. We implement them in JavaScript to support seamless web application deployment and portability to multiple browsers as well as Android. On the other hand, our first-of-its-kind protocol for median statistics of Tor hidden services (cf. Section V) uses additively homomorphic threshold decryption, relying on a set of non-colluding authorities. It is developed in Python so that it can be integrated on Tor Hidden Service Directories. We also show how to integrate differential privacy guarantees by adding noise to leaked *intermediate* values during the median estimation process which does not involve non-linear operations.

Paper organization. The rest of the paper is organized as follows. Next section introduces relevant background information, then, Section III and Section IV present, respectively, a privacy-preserving recommender system for online broadcasters and an Android-based private location prediction service. Section V introduces a system for privately computing the median statistics of Tor hidden services. After reviewing related work in Section VI, the paper concludes with Section VII.

II. Preliminaries

A. Cryptographic Background

Computational Diffie Hellman Assumption. Let $\mathbb{G}$ be a cyclic group of order q ($|q| = \tau$, for security parameter τ), with generator g. We say that the Computational Diffie Hellman (CDH) problem is hard if, for any probabilistic polynomial-time algorithm $\mathcal{A}$ and random x, y drawn from $\mathbb{Z}_q$:

$$\Pr\left[\mathcal{A}(\mathbb{G}, q, g, g^x, g^y) = g^{xy}\right]$$

is negligible in the security parameter τ.

Decisional Diffie Hellman Assumption. Let $\mathbb{G}$ be a cyclic group of order q ($|q| = \tau$), with generator g. We say that the Decisional Diffie Hellman (DDH) problem is hard if, for any probabilistic polynomial-time algorithm $\mathcal{A}'$ and random x, y, z drawn from $\mathbb{Z}_q$:

$$\left| \Pr\left[\mathcal{A}'(\mathbb{G}, q, g, g^x, g^y, g^z) = 1\right] - \Pr\left[\mathcal{A}'(\mathbb{G}, q, g, g^x, g^y, g^{xy}) = 1\right] \right|$$

is negligible in the security parameter τ.

Pairwise Independent Hash Functions. Let H be a family of *random-looking* hash functions mapping values from a domain [D] to a range [R]. H is *pairwise independent* iff $\forall x \neq y \in [D]$ and $\forall a_1, a_2 \in [R]$: $\Pr_{h \in H}\left[h(x) = a_1 \wedge h(y) = a_2\right] = \frac{1}{R^2}$.

B. Count-Min Sketch and Count Sketch

Count-Min Sketch [22] is a data structure that can be used to provide a succinct sublinear-space representation of multi-sets. An interesting property is that they enable aggregation of the multi-sets represented by two or more sketches using a linear operation on the sketches themselves. Prior uses of Count-Min Sketch include summarizing large amounts of frequency data for sensing, networking, natural language processing, and database applications [2].

***Definition* 1** (*Count-Min Sketch).* A Count-Min Sketch with parameters (ϵ, δ) is a two-dimensional array (table) X, with width w and depth d. Given parameters (ϵ, δ), set $d = \lceil \ln T/\delta \rceil$ and $w = \lceil e/\epsilon \rceil$, where T is the number of items to be counted. Each entry of the table is initialized to zero. Then, d hash functions $h_j : \{0,1\}^* \rightarrow \{0,1\}^w$, are chosen uniformly at random from a pairwise-independent family $\mathcal{H}$.

Update Procedure. To update item i by a quantity c_i, c_i is added to one element in each row, where the element in row j is determined by the hash function h_j. The update is denoted as (i, c_i). More precisely, to update the count for item i to $c_i \in \mathbb{N}$, for each row j of X, set:

$$X[j, h_j(i)] \leftarrow X[j, h_j(i)] + c_i$$

Estimation Procedure. To estimate the count $\hat{c_i}$ for item i, we take the minimum of the estimates of c_i from every row of X:

$$\hat{c_i} \leftarrow \min_j X[j, h_j(i)]$$

Error Upper Bound. Given estimate $\hat{c_i}$, it holds:

1) $c_i \leq \hat{c_i}$
2) $\hat{c_i} \leq c_i + \epsilon \sum_{j=1}^{T} |c_j|$ with probability $1 - \delta$.

(where c_i is the true counter).

Count Sketch [16] is a data structure which provides an estimate for an item's frequency in a stream. Count Sketch has the same update procedure as Count-Min Sketch, but differs

in the estimation. Specifically, given the table X built on the stream, the row estimate of c_i (which is the counter of item i) for row j is computed based on two buckets: $X[i, h_j(i)]$ and $X[i, h'_j(i)]$, where $h'_j(i)$ is defined as:

$$h'_j(i) := \begin{cases} h_j(i) - 1 & \text{if } h_j(i) \bmod 2 = 0 \\ h_j(i) + 1 & \text{if } h_j(i) \bmod 2 = 1 \end{cases}$$

The estimate of c_i for row j is then

$$\big(X[j, h_j(i)] - X[j, h'_j(i)]\big)$$

To estimate the count $\hat{c_i}$ for item i, we take the median of the estimates of c_i from every row of X:

$$\hat{c_i} \leftarrow \underset{j}{median} \left(X[j, h_j(i)] - X[j, h'_j(i)]\right)$$

Both Count-Min and Count Sketch are linear: the element-wise sum of the sketches representing two multi-sets yields the sketch of their union.

C. Differential Privacy

Differentially private mechanisms allow a party publishing a dataset to make sure that only a bounded amount of information is leaked. Output perturbation mechanisms modify a statistic on a dataset D, prior to its release, using a randomized algorithm A, so that the output of A does not reveal too much information about any particular row in D.

***Definition* 2** (*ϵ-Differential privacy* [30])**.** A randomized algorithm A satisfies ϵ-differential privacy, if for any two neighbor datasets D_1 and D_2 that differ only in one row, and for any possible output R of A, it holds:

$$\Pr\left[A(D_1) = R\right] \leq e^{\epsilon} \cdot \Pr\left[A(D_2) = R\right]$$

Note that ϵ here is used differently than in the Count-Min Sketch's definition. Although this somewhat overloads the notation for ϵ, it is actually clear from the context if it relates to the data structure or to the differential privacy setting.

Laplace Mechanism. In Section V, we use the differentially private Laplace mechanism [31], which perturbs the output of a function F. Given F, the Laplace mechanism transforms F into a differentially private algorithm, by adding independent and identically distributed (i.i.d.) noise (denoted as η) into each output value of F. The noise η is sampled from a Laplace distribution $Lap(\lambda)$ with the following probability density function: $Pr[\eta = x] = \frac{1}{2\lambda} e^{|x|\lambda}$. Dwork [30] proves that the Laplace mechanism ensures ϵ-differential privacy if $\lambda \geq \frac{S(F)}{\epsilon}$, with $S(F)$ denoting the sensitivity of F, defined as:

$$S(F) = \max_{D_1, D_2} ||F(D_1) - F(D_2)||_1$$

where $||\cdot||_1$ denotes the L1 norm, and D_1 and D_2 are any two neighbor datasets. Intuitively, $S(F)$ measures the maximum possible change in F's output when we modify one arbitrary row in F's input.

D. ItemKNN-based Recommender Systems

Recommender systems are used to predict the utility of a certain item for a particular user, based on their previous ratings as well as those of other "similar" users [58]. Consider a set of N users and a list of M items: for each user, a rating can be associated to each item, based, e.g., on the user's explicit opinion about the item (e.g., 1 to 5 stars) or by implicitly deriving it from purchase records or browser history.

Machine learning can be used to predict the expected rating of an unrated item for a given user. The *K-Nearest Neighbor (KNN)* classification algorithm finds the top-K nearest neighbors for a given item, so that ratings associated with these are combined to predict unknown ratings. In this paper, we use a variant called *ItemKNN* [59]. The algorithm is trained using an item-to-item similarity matrix (correlation matrix), where each element expresses the similarity between a pair of items, and the Cosine Similarity is computed between vectors of items (e.g., user ratings for each item).

If ratings are binary values (e.g., viewed/not viewed), as in one of our applications (see Section III), the Cosine Similarity between items a and b is:

$$\{Sim\}_{ab} = \frac{C_{ab}}{\sqrt{C_a \cdot C_b}} \tag{1}$$

where C_{ab}, C_a, and C_b denote, respectively, the number of people who rated both a and b, a, and b. Given the similarity matrix, we can identify the nearest neighbors for each item as the items with the highest correlation values. The final model then consists of the identity of the nearest neighbors and their correlation values (or *weights*) which are used in the prediction process, i.e., the items that should be recommended.

Note that, with ItemKNN, given the item-to-item matrix, each user could independently compare their ratings with the nearest neighbors of each item in the model. Upon finding a match, the weight is added to the prediction score for that item. The items are then ranked by their prediction scores and the top K are taken as recommendations.

E. Exponential Weighted Moving Average (EWMA)

Exponential Weighted Moving Average (EWMA) models [62] can predict future values based on past values weighted with exponentially decreasing weights toward older values. Given a signal over time $r(t)$, we indicate with $\tilde{r}(t+1)$ the predicted value of $r(t+1)$ given the past observations, $r(t')$, at time $t' \leq t$. Predicted signal $\tilde{r}(t+1)$ is estimated as:

$$\tilde{r}(t+1) = \sum_{t'=1}^{t} \alpha(1-\alpha)^{t-t'} r(t')$$

where $\alpha \in (0, 1)$ is the smoothing coefficient, and $t' = 1, \ldots, t$ indicates the training window, i.e., 1 corresponds to the oldest observation while t is the most recent one.

In the rest of this work, we present efficient techniques to estimate, in a private and distributed way, the training datasets required for ItemKNN-based Recommender System, Exponential Weighted Moving Average (EWMA) modeling, as well as median and other frequency statistics. The mechanisms combine traditional linear aggregation with sketches, for efficiency, and, when needed, differential privacy to limit information leakage.

III. Private Recommender Systems For Streaming Services

Media streaming services are becoming increasingly popular as numerous dedicated providers (e.g., Netflix, Amazon, Hulu) as well as "traditional" broadcasting services (e.g., BBC, CNN, Al-Jazeera) offer digital access to TV shows, movies, documentaries, and news. One of the providers' goals is often continuous user engagement, thus, new content should periodically be suggested to users based on their interests. These recommendations are usually provided by means of *recommender systems* [3, 41] like ItemKNN (cf. Section II-D), which typically require the full availability of users' ratings, whereas, we focus on a model where a provider like the BBC provides recommendations to its users, e.g., on iPlayer, *without tracking* their preferences and viewings. Note that iPlayer does not actually require users to register or have an account, which further motivates the need to protect users' privacy.

A. Overview

We present a novel privacy-friendly recommender system where the ItemKNN algorithm is trained using only aggregate statistics. Aiming to build a global matrix of co-views (i.e., pairs of programs watched by the same user) in a privacy-preserving way, we rely on (i) private data aggregation based on secret sharing (inspired by the "low overhead protocol" in [47]), and (ii) the Count-Min Sketch data structure to reduce the computation/communication overhead, trading off an upper-bounded error with increased efficiency.

Recommendations are derived, based on ItemKNN, as follows: users' interests are modeled as a (symmetric) item-to-item matrix $I = \{0,1\}^{M \times M}$, where I_{ab} is set to 1 if the user has watched both programs a and b and to 0 otherwise. I_{aa} is set to 1 if the user has watched the program a. The Cosine Similarity $\{Sim\}_{ab}$ between programs a and b can be computed from item-to-item matrices using Equation 1. The Cosine Similarity is then used by each user to derive personalized recommendations as described in Section II-D.

System Model. Our system involves a `tally` (e.g., the BBC) and a set of `users`, and no other trusted/semi-trusted authority:

1) `Users`, possibly organized in groups, compute their (secret) blinding factors, based on the public keys of the other `users`, in such a way that they all sum up to zero. They encrypt their local Count-Min Sketch entries (representing their co-view matrix) using these blinding factors, and send the resulting ciphertexts to the `tally`.
2) The `tally` receives the encrypted Count-Min Sketch from each `user`, aggregates the encrypted counts, and decrypts the aggregates. These are broadcast back to the `users`, who use them to recover an estimate of the global similarity matrix and derive personalized ItemKNN-based recommendations.

Notation. In the rest of this section, we denote with N the number of `users`, with M the total number of items, and with $L = d \cdot w$ the number of items in a Count-Min Sketch table. Also, let $\mathbb{G}$ be a cyclic group of prime order q for which the Computational Diffie-Hellman problem (CDH) is hard and g be the generator of the same group. $\mathrm{H} : \{0,1\}^* \rightarrow \mathbb{Z}_q$ denotes a cryptographic hash function mapping strings of arbitrary length to integers in $\mathbb{Z}_q$. Finally, "||" denotes the concatenation operator and $a \in_r A$ means that a is sampled at random from A. We assume the system runs on input public parameters $\mathbb{G}, g, q$, where g generates a group of order q in $\mathbb{G}$.

B. Protocol

We now present the details of our proposed protocol. Its cryptographic layer is also summarized in Figure 1.

Setup. Each `user` $\mathcal{U}_i$ ($i \in [1, N]$) generates a private key $x_i \in_r \mathbb{G}$, and computes and publishes public key $y_i = g^{x_i} \bmod q$. Public keys of all `users` are distributed to each other, using a public bulletin board or the `tally` itself.

As discussed later in this section, `users` might be organized in *groups* in order to facilitate aggregation. To ease presentation, we discuss the protocol steps for a single group of `users`, as combining aggregates from different groups is trivial and can be done, in the clear, by the `tally`.

Count-Min Sketch construction. We assume each `user` $\mathcal{U}_i$ holds an input vector of data points $I = \{I_c \in \mathbb{N},\ c = 1, \ldots, T\}$, which represents $\mathcal{U}_i$'s co-view matrix (i.e., $T = M \cdot M/2$). First, $\mathcal{U}_i$ initializes a Count-Min Sketch table X_i with all zero entries. In the following, we represent $\mathcal{U}_i$'s Count-Min Sketch table $X_i \in \mathbb{N}^{d \times w}$ as a vector of length $L = d \cdot w$. Then, $\mathcal{U}_i$ encodes I in the Count-Min Sketch using the update procedure described in Section II-B, where the following pairwise-independent hash function is employed:

$$h(x) = ((ax + b) \bmod p) \bmod w$$

for $a \neq 0, b$ random integers modulo a random prime p. At the end of this step, $\mathcal{U}_i$ has built a Count-Min Sketch table $X_i = \{X_{i_\ell}\}_{\ell=1}^{L}$ (with $L = d \cdot w$ as per Definition 1).

Encryption. To participate in the privacy-preserving sketch aggregation, each `user` $\mathcal{U}_i$ first needs to generate blinding factors. At round s, for each $\ell = 1, \ldots, L$, `user` $\mathcal{U}_i$ computes:

$$k_{i_\ell} = \sum_{\substack{j=1 \\ j \neq i}}^{N} \mathrm{H}(y_j^{x_i} || \ell || s) \cdot (-1)^{i>j} \bmod q$$

where

$$(-1)^{i>j} := \begin{cases} -1 & \text{if } i > j \\ 1 & \text{otherwise} \end{cases}$$

Note that the sum of all k_{i_ℓ}'s equals to zero:

$$\sum_{i=1}^{N} k_{i_\ell} = \sum_{i=1}^{N} \sum_{\substack{j=1 \\ j \neq i}}^{N} \mathrm{H}(y_j^{x_i} || \ell || s) \cdot (-1)^{i>j} = 0$$

Then, for each entry X_{i_ℓ}, $\mathcal{U}_i$ encrypts X_{i_ℓ} as $b_{i_\ell} = X_{i_\ell} + k_{i_\ell} \bmod 2^{32}$, as only 32 bits of b_{i_ℓ} are enough for our application, and sends the resulting ciphertext to the `tally`.

Aggregation. The `tally` receives the ciphertexts from the N `users` and (obliviously) aggregates the sketches. Specifically, for $\ell = 1, \ldots, L$, it computes:

$$C_\ell = \sum_{i=1}^{N} b_{i_\ell} = \sum_{i=1}^{N} k_{i_\ell} + \sum_{i=1}^{N} X_{i_\ell} = \sum_{i=1}^{N} X_{i_\ell} \bmod 2^{32}$$

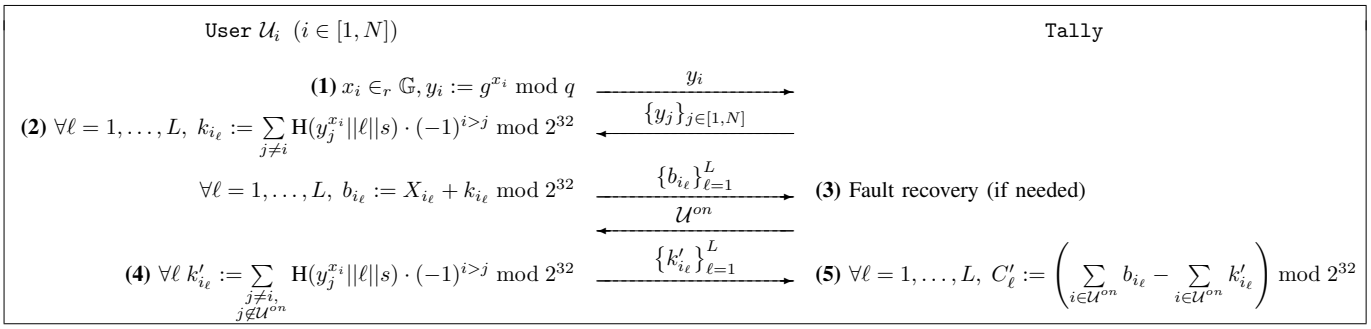


Figure 1: Cryptographic layer of our private recommender system for online streaming services. At setup **(1)**, `users` compute their secret share and send their public key to the `tally`, who broadcasts them to the other `users`. During the encryption phase **(2)**, each `user` computes the blinding factors, encrypts their Count-Min Sketch and sends it to the `tally`. In case not all `users` have sent the data, the `tally` broadcasts $\mathcal{U}^{on}$, the subset of `users` that did **(3)**. These compute new blinding factors and send them to the `tally` **(4)**. Aggregate sketches are then recovered by the `tally` **(5)**.

where C_ℓ denotes the ℓ-th item in the aggregate Count-Min Sketch table. $\{C_\ell\}_{\ell=1}^{L}$, are broadcast back to the `users` (but can obviously be used locally at the `tally` too), who use them to recover an estimate of the global matrix and derive personalized recommendations via the ItemKNN algorithm.

Fault Tolerance. If, during the aggregation phase, only a subset of `users` report their values b_{i_ℓ} to the `tally`, the sum of the k_{i_ℓ}'s is no longer equal to zero and the aggregate items C_ℓ cannot be decrypted. However, it is possible to recover as follows: Let $\mathcal{U}^{on}$ denote the list of `users` who have submitted the data in the aggregation phase. The `tally` sends $\mathcal{U}^{on}$ to each $\mathcal{U}_i \in \mathcal{U}^{on}$. Then, $\mathcal{U}_i$ computes, for each $\ell = 1, \ldots, L$,

$$k'_{i_\ell} = \sum_{\substack{j=1 \\ j \neq i, j \notin \mathcal{U}^{on}}}^{N} \mathrm{H}(y_j^{x_i} || \ell || s) \cdot (-1)^{i>j} \bmod q$$

and sends these values back to the `tally`.

Assuming all `users` in $\mathcal{U}^{on}$ submit the values k'_{i_ℓ}, the `tally` can recover the entries in the aggregate sketches (for `users` in $\mathcal{U}^{on}$) by computing:

$$C'_\ell = \left(\sum_{i \in \mathcal{U}^{on}} b_{i_\ell} - \sum_{i \in \mathcal{U}^{on}} k'_{i_\ell} \right) \bmod 2^{32}$$

Groups. Although the protocol can cope with faults, we should nonetheless minimize the probability of missed contributions. Moreover, as discussed in Section III-D, the protocol's complexity also depends on the number of `users` and, in the case of iPlayer, there can be peaks of hundreds of thousands of users per hour[1]. Consequently, we need to organize `users` into reasonably sized groups. As mentioned earlier, combining aggregates from different groups is straightforward and can be done, in the clear, by the `tally`.

We argue that a good choice is between 100 and 1,000 `users` per group, as also supported by our empirical evaluation presented later. There could be a few different ways to form groups: for instance, the `tally` could group `users` in physical proximity and/or select `users` that are watching/listening a video with at least a couple of minutes left to watch. Also note that `users` not involved in the protocol (or having limited "history") can get recommendations too as the `tally` can still provide them with the global co-view matrix, which, even though it does not include their own contribution, can be used by the ItemKNN algorithm to derive recommendations.

Security Analysis. The security of our scheme, in the honest-but-curious model, is straightforwardly guaranteed by that of the "low overhead" private aggregation scheme by Kursawe et al. [47], which is secure under the CDH assumption. We modify it to cope with users faults and to aggregate Count-Min Sketch entries, rather than the actual data, and this does not affect the privacy properties of the scheme. In case of passive collusions between `users`, the confidentiality of the data provided by the non-colluding `users` is still preserved. Finally, note that malicious active `users` could report fake values in order to invalidate the final aggregation values, however, protocol's integrity could be preserved using verifiable tools such as zero-knowledge proofs and commitments, an extension we leave as part of future work, along with considering a malicious `tally`.

C. Prototype Implementation

We have implemented the `tally`'s functionalities as a web application running on the server-side JavaScript environment *Node.js* (or *Node* for short).[2] We also use *Express.js*[3] to organize our application into a Model View Controller (MVC) web architecture and *Socket.io*[4] to set up bidirectional web-socket connections. Integrating our solution is as simple as installing a *Node* module through the Node Package Manager (NPM) and importing it from any web application, thus requiring no familiarity with the inner workings of the cryptographic and aggregation layers.

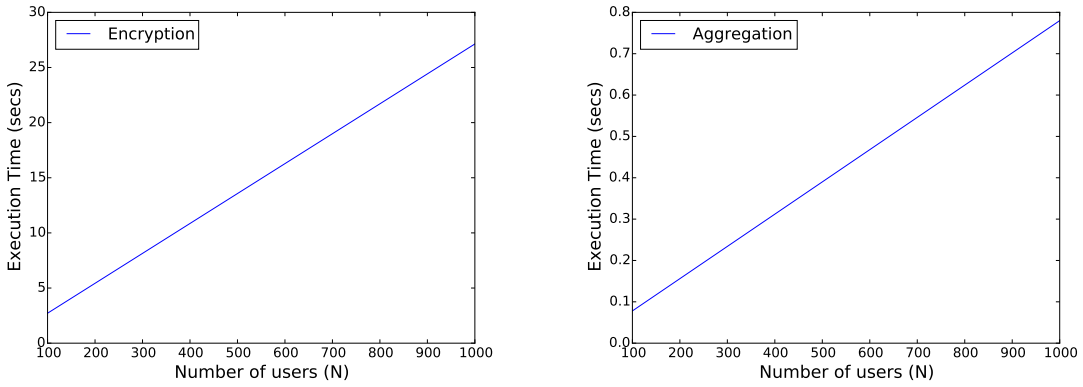
The module for `user`'s functionalities is modeled as the client-side of the web application and can be run as simple JavaScript code embedded on a HTML page. Therefore, it requires no deployment or installation of any additional software by the `users`, but runs directly in the browser, transparently, when `users` visit `tally`'s website. Our JavaScript implementation is also compatible with smartphone browsers (e.g., the Android version of Chrome), nevertheless, we have also

[1] http://downloads.bbc.co.uk/mediacentre/iplayer/iplayer-performance-may15.pdf

[2] https://nodejs.org/

[3] http://expressjs.com/

[4] http://socket.io/



(a) Client **(b)** Server

Figure 2: Execution time for increasing number of `users` (with 700 programs).

implemented a stand-alone Android application using Apache Cordova.[5] The source code of both our browser and Android app is available upon request, so that developers can simply import and extend our code for their own applications.

Cryptographic Operations. The cryptographic layer of the protocol is also written in JavaScript, using the Ed25519 curve [8] implementation available from *Elliptic.js*,[6] which supports 256-bit points and provides security comparable to a 128-bit security parameter. SHA-256 is used for (cryptographic) hashing operations.

D. Performance Evaluation

We now analyze the performance of our system, both analytically (reporting asymptotic complexities) and empirically.

Asymptotic Complexities. The setup phase carried out by the `user` requires $O(N)$ random group points (where N is the number of total `users`) and $O(N)$ messages need to be sent for all the `users` to distribute the public keys. To generate the blinding factors, each `user` then needs to perform $O(N)$ exponentiations in $\mathbb{G}$ and $O(L \cdot N)$ hashing operations. Count-Min Sketch encryption (at `user`'s side) requires $O(L)$ integer additions in $\mathbb{Z}_q$, one for each of the $L = O(\log(M^2))$ Count-Min Sketch entries, while communication complexity amounts to $O(L)$ 32-bits integers for each `user`. To complete the aggregation, the `tally` computes $O(L \cdot N)$ linear operations.

The use of the Count-Min Sketch significantly speeds up the efficiency of the system. In fact, without them, each `user` would need to perform $O(N(M^2))$ hashing operations and send $O(M^2)$ 32-bit integers, while the `tally` would need to compute $O(N(M^2))$ operations.

Computation Overhead. We have also simulated the execution of our private recommender system and measured execution times (averaged over 100 iterations) for all operations. Simulations have been performed on a machine running Ubuntu Trusty (Ubuntu 14.04.2 LTS), equipped with a 2.4 GHz CPU i5-520M and 4GB RAM.

[5] https://cordova.apache.org/

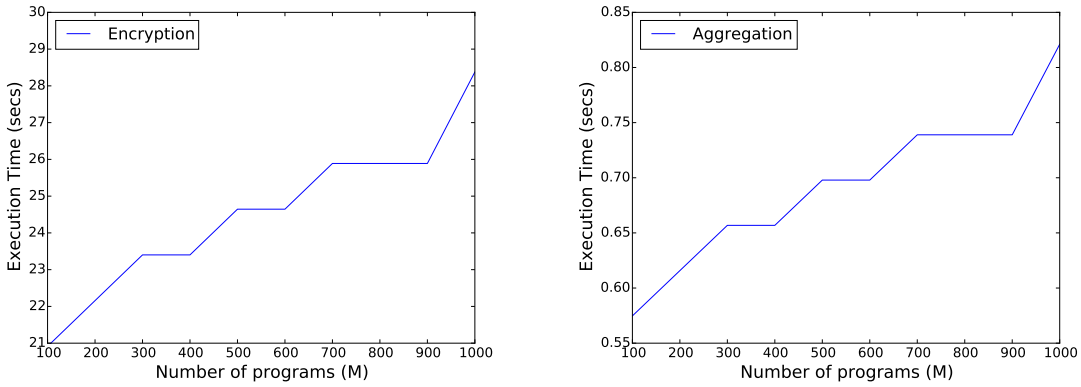
[6] https://github.com/indutny/elliptic

In Figure 2, we plot running times of protocol's client- and server-side for an increasing number of `users`, fixing the number of programs to 700 (the average number of programs available on iPlayer) and the sketch parameters to $\epsilon = \delta = 0.01$ (see Definition 1). Using this setting, the number of rows d and columns w of the Count-Min Sketch amounts to $d = 18$, $w = 272$ leading to a Count-Min Sketch of size $L = d \cdot w = 18 \cdot 272 = 4{,}896$. Running times grow linearly in the number of `users`. As illustrated in Figure 2(a), the encryption, performed by each `user` (see step (2) in Figure 1), takes 2.7 seconds with 100 `users` and 27 seconds with 1,000 `users`, while Figure 2(b) reveals that `tally` completes the aggregation (step (5) in Figure 1) in 78ms (resp., 780ms) with 100 (resp., 1,000) `users`.

We then measure the execution time for an increasing number of programs and a fixed number of `users`, i.e., 1,000. Figure 3(a) illustrates running times' logarithmic growth for encryption, ranging from 21 seconds with 100 programs to 28 seconds with 1,000 programs. Figure 3(b) illustrates `tally`'s execution times for the aggregation, which approximately range from 600ms to 800ms. Note that the "stair" effect of the plots in Figure 3 is due to the fact that the Count-Min Sketch size can be the same with close numbers of programs.

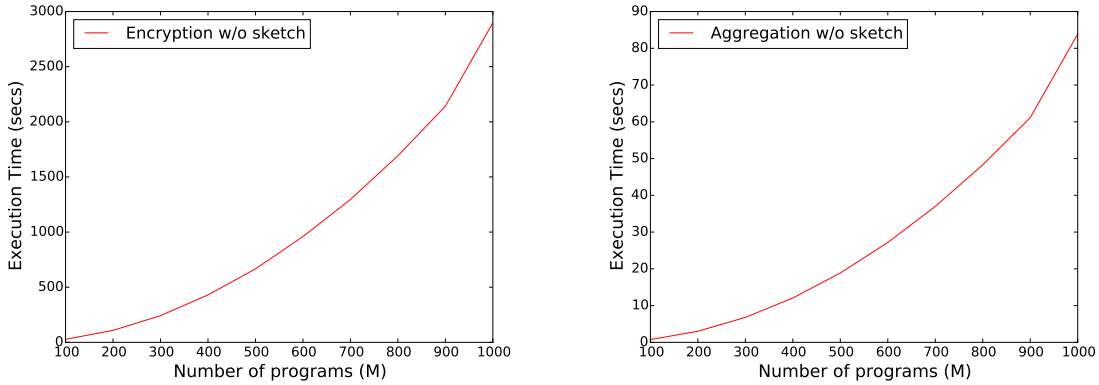
Without the compression factor of the Count-Min Sketch, the running times for both `user` and `tally` would grow linearly in the size of the co-view matrix (i.e., $M \cdot M/2$), yielding remarkably slower executions. As illustrated in Figure 4(a), with 1,000 `users` and 1,000 programs, running time for each `user` amounts to almost 50 minutes instead of 28 seconds using the sketch, whereas, the aggregation at the `tally` completes in almost one and a half minute (versus less than one second using Count-Min Sketch). Finally, execution time of the *ItemKNN* operations carried out at `user`'s side, with 700 programs, amounts to 850ms for each `user`.

Communication Overhead. In Table I, we report the amount of bytes exchanged between all parties for different number of `users` and Count-Min Sketch sizes, fixing the number of programs to 700. Note that, without the compressing factor of the sketch, with 700 programs, each `user` would have to send 960KB instead of 20KB.



(a) Client **(b)** Server

Figure 3: Execution time for increasing number of programs (with 1,000 `users`).



(a) Client **(b)** Server

Figure 4: Execution time for increasing number of programs (with 1,000 `users`) *without* Count-Min Sketch.

#Users	Bytes (Tally to User)	Sketch Size	Bytes (User to Tally)
100	3,200	4,896	19,584
200	6,400	2,448	9,792
300	9,600	1,638	6,552
400	12,800	1,224	4,896
500	16,000	972	3,888
600	19,200	810	3,240
700	22,400	702	2,808
800	25,600	612	2,448
900	28,800	540	2,160
1000	32,000	486	1,944

TABLE I: Bytes exchanged by `user` and `tally` for different #`users` and size of the Count-Min Sketch, considering 700 programs.

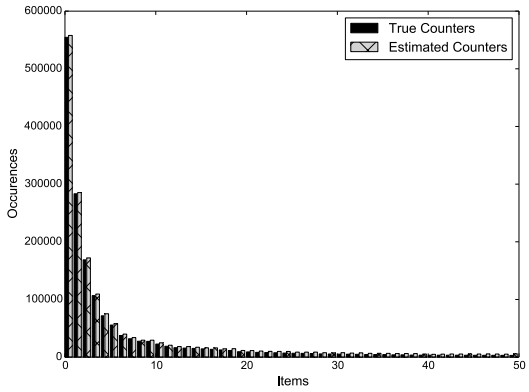
Accuracy Estimation. Finally, we evaluate the accuracy loss due to the use of Count-Min Sketch, specifically, over the most 50 frequent items, using a synthetic dataset sampled from a zipfian distribution simulating a million `users`. We set the Count-Min Sketch parameters to be $\epsilon = 0.01$ and $\delta = 0.01$ as we have measured an acceptable accuracy loss level introduced by the Count-Min Sketch (see below). Once again, we fix the number of programs to $M = 700$, leading to a Count-Min Sketch of size $L = 4{,}896$. Figure 5(a) shows that the Count-Min Sketch estimation over the most 50 frequent items is almost indistinguishable from the true population.

We also plot, in Figure 5(b), the average error, defined as $|\hat{c}_i - c_i| / \sum_j |c_j|$, over the most 50 frequent items with an increasing number of `users`, while fixing $M = 700$, $\delta = 0.01$ (yielding a total number of items to update on the Count-Min Sketch of $T = M \cdot M/2 = 245{,}000$) and three choices of the ϵ parameter, i.e., $0.01, 0.05$, and 0.1. The average error decreases with more `users` and smaller values of ϵ. Standard deviation values are infinitesimal, thus, we do not include them in the plot as they would not be visible.

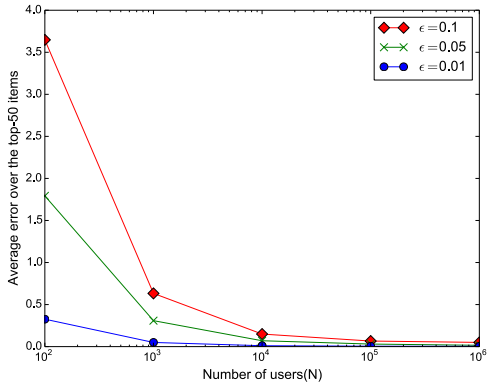
IV. Private Aggregate Location Prediction

The rapid proliferation of smartphones, with 2 billion estimated users by the year 2016 [26], makes it increasingly easy (and appealing) to track users' locations and movements using sensors like GPS and WiFi. This contextual information can be extremely useful to train machine learning algorithms and predict future events, paving the way for anticipatory mobile computing [57]. Location and movement models can be used, e.g., for traffic mitigation, road monitoring, and hazard detection [44], as well as to guide decision frameworks to respond to anomalies and disruptions on short notice.

Pervasive location sensing, however, raises important privacy concerns as single individuals' movements can easily be



(a) True vs estimated counters



(b) Average error for different values of ϵ

Figure 5: Visualizing the accuracy of the Count-Min Sketch for the most 50 frequent items (with 700 programs and sketch size 4,896).

tracked and sensitive information could be exposed. If home and work locations can be deduced from anonymized location traces, single individuals can be uniquely re-identified [38]. Moreover, location patterns have been shown to leak personal information, e.g., taxi drivers' religion and individuals' visits to gentleman's clubs.[7]

In this section, we instantiate a smartphone application enabling `users` to report, to a service provider (`tally`), their locations over time. `Users`' privacy is protected as only aggregate (over many `users`) location statistics are disclosed. We then show how these statistics can be used to train a model and predict future movements, and support private computation and prediction of "heat maps" relying on the aggregate counts of people in a given area over a period of time.

System Model. We operate in the same model as our privacy-friendly recommender system (cf. Section III-B), involving a `tally` that privately aggregates location statistics contributed from a set of `users`, and re-use the same cryptographic layer. Once again, we support efficient computation of private statistics using (i) Count-Min Sketch's succinct data representation and (ii) privacy-preserving aggregation with `users`' blinding factors summing up to zero.

Overview. We assume a 2-D space territory $\mathcal{R}$ is partitioned into a grid of $|S| = p \times p$ cells ($S = \{S[1,1], S[1,2], \ldots, S[p,p]\}$), and t finite intervals (time slots) $[t_{j-1}, t_j]$, where $j \in \mathbb{N}^+$. Let $S_i^{(t_j)}$ be the grid containing, for each cell, the number of times the `user` $\mathcal{U}_i$ has logged her position (using a GPS measurement) within that particular cell over $t \in [t_{j-1}, t_j]$. `User` $\mathcal{U}_i$, for each time slot $[t_{j-1}, t_j]$, builds the grid $S_i^{(t_j)}$ with locations logged over time, maps the grid into a Count-Min Sketch, and sends the encrypted sketch to the `tally`. This aggregates and decrypts them, reconstructing the grid containing the (estimated) aggregate locations.

The location statistics can be used to display 'heat maps" (e.g., a graphical representation of congestion), or to perform time-series based prediction over a sequence of heat maps. Using an Exponential Weighted Moving Average (EWMA) model (see Section II-E), we can predict the future popularity of a cell, by relying on the past (approximated) observations for that cell. Other machine learning techniques, e.g., Multivariate Support Vector Machines or Logistic Regression, could also be used for the prediction, but we consider it to be beyond the scope of this paper to investigate new predictors.

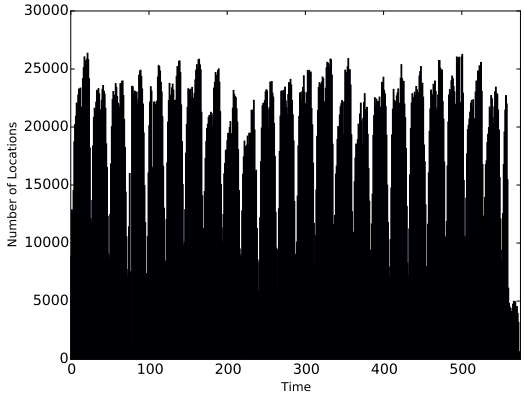


Figure 6: Number of taxi locations over time.

The San Francisco Cabs Dataset. To evaluate the feasibility of our intuition, we use a publicly available dataset containing mobility traces of San Francisco taxi cabs.[8] The dataset contains 11 million GPS coordinates, generated by 536 taxis over almost a month in May 2008. We group the taxi locations in time slots of one hour, leading to a total of 575 epochs. Figure 6 shows the presence of weekly and daily patterns in the number of taxi locations over time (i.e. hourly time slots) and peaks of roughly 25,000 total hourly contributions.

Succinct Data Representation. We investigate whether succinct data representation could be applied to the problem of collecting location statistics, and measure the accuracy loss introduced by the Count-Min Sketch's compact representation. In Figure 7, we plot the average error defined as $|\hat{c}_i - c_i| / \sum_j |c_j|$ and the relative standard deviation over the most 100 popular cells for each time slot, while fixing $\epsilon = \delta = 0.01$ and the total number of cells to $|S| = 100 \times 100$ (yielding a Count-Min Sketch of size $L = 3,808$). Observe that the average error

[7] See http://on.mash.to/1ByncHD and https://goo.gl/Ta5JYG.

[8] http://cabspotting.org/

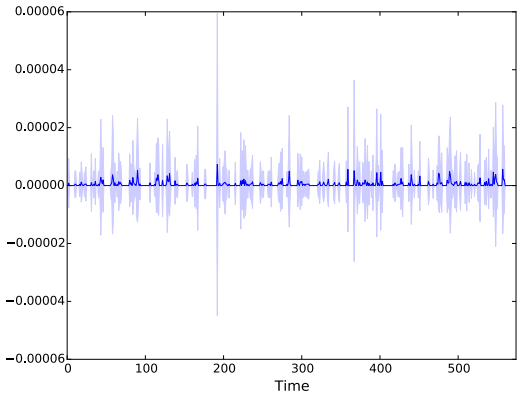


Figure 7: Average error introduced by the Count-Min Sketch on the aggregate statistics for the top-100 locations.

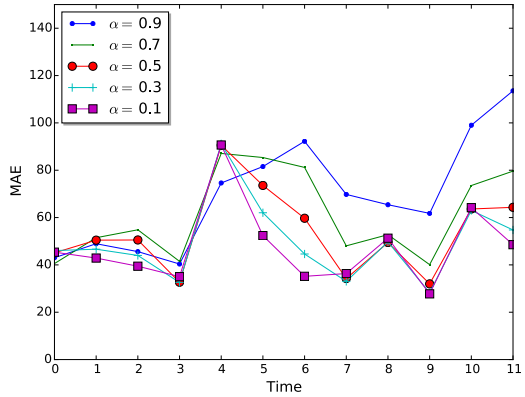


Figure 8: Mean absolute error in the prediction for different values of prediction algorithm's parameter α.

is infinitesimal for every time slots.

Heat Map Prediction. Next, we focus on predicting future heat maps using the EWMA algorithm introduced in Section II-E. We start by evaluating the accuracy of EWMA-based prediction relying on the aggregates collected *without using the Count-Min Sketch*. We perform the prediction over a subset of 12 consecutive epochs having the maximum number of reported locations, giving the past 24 hours observations as input to the EWMA algorithm. Figure 8 plots the Mean Absolute Error (MAE) in the prediction compared to the ground truth over the most 100 popular cells, considering different values of α, i.e., EWMA's smoothing coefficient (cf. Section II-E). The plot shows that, in almost all slots, lower values of α lead to more accurate results.

We then perform the prediction over the approximate heat maps, i.e., *using the sketches*. We focus on the same time slot, and fix $\alpha = 0.1$. Figure 9 shows the error introduced by the Count-Min Sketch in the prediction, for each time slot considered, with respect to the prediction based on the *"real"* heat maps. We observe that this error, while fluctuating, is appreciably low for every prediction, thus confirming the feasibility of our techniques for the problem of privately predicting future heat maps.

Once again, we have implemented our techniques in JavaScript, with the server-side running as a *Node* module, and

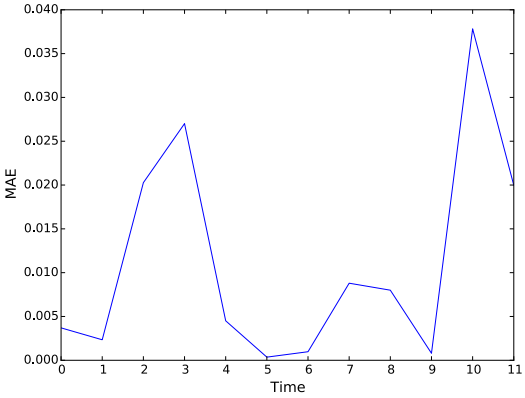


Figure 9: Mean absolute error introduced by the Count-Min Sketch on the prediction accuracy.

client-side running as an open-source Android application built using Apache Cordova. Source code is available upon request. Note that, due to space limitations, a performance evaluation of our implementations is not presented in this version as it would anyway mirror the one presented in Section III.

V. Gathering Statistics on Tor Hidden Services

The privacy-preserving collection of statistics using efficient data structures, seeking a trade-off between accuracy and efficiency, has also interesting applications in non-user facing settings such as collecting network statistics from servers or routers. In this section, we present a novel mechanism geared to privately gather statistics in the context of the Tor anonymity network [28]. The Tor project has recently received funding to improve monitoring of load and usage of Tor hidden services.[9] This motivates them to extract aggregate statistics about the number of hidden service descriptors from multiple Hidden Service Directory authorities. In order to ensure robustness, the Tor project has determined that the median – rather than the mean – of these volumes should be calculated, which is beyond privacy-friendly statistics approaches like Privex [32].

In this section, we first describe the protocol for estimating median statistics using Count Sketch, then, we present the design and deployment of its prototype implementation, along with its performance evaluation.

A. Private Median Estimation using Count Sketch

We rely on the Count Sketch [16] data structure, which closely resembles Count-Min Sketch, used in Sections III–IV. Recall from Section II-B that building a Count Sketch follows the same process as a Count-Min Sketch, thus leading to a $d \cdot w$ table of positive integer values, whereas, the estimation of an item's frequency is slightly different: for each row, d_i, a hash function is applied to the item leading to a column w_j. An unbiased estimator of the frequency of the item is the value at this position minus the value at an adjacent position – and the median of those estimators is the final estimated frequency. What is key to the success of our techniques is that the estimate of the frequency of specific values, as well as sets of values, is a simple linear sum of Count Sketch entries;

[9] https://www.torproject.org/docs/hidden-services.html.en

computing it does not require non-linear (e.g., *min*) operations as for the Count-Min Sketch.

For this application, we build on privacy-preserving data aggregation based on threshold public-key encryption, specifically, an Additively Homomorphic Elliptic-Curve variant of El Gamal (AH-ECC) [7], summarized below. This allows us to seamlessly tolerate missing contributions – following an approached first proposed by Jawurek et al. [45].

AH-ECC consists of the following three algorithms (using a multiplicative notation):

1) *KeyGen*(1^τ): Given a security parameter τ, choose an elliptic curve E and (g_1, g_2) public generators on E, generating a group of order q. Choose a random private key $x \in \mathbb{Z}_q$, define the public key as $pk = {g_1}^x$, and output public parameters (E, g_1, g_2, pk) and private key x.
2) *Encrypt*(m, pk): The message m is encrypted by computing two elliptic curve points as $(A, B) := ({g_1}^r, {pk}^r {g_2}^m)$, where $r \in \mathbb{Z}_q$ is selected at random. The ciphertext is thus the tuple of points (A, B).
3) *Decrypt*(A, B, x): Decryption is performed by computing the element $BA^{-x} = {g_2}^m$. We can achieve constant time decryption by pre-computing a table of discrete logarithms which is then used to recover m from ${g_2}^m$ (this solution is practical for small values of m).

AH-ECC is additively homomorphic since an element-wise multiplication of ciphertexts yields an encryption of their sum.

Setup. Our system relies on a set of authorities that can jointly decrypt a ciphertext from the AH-ECC additively homomorphic public-key cryptosystem. During setup, each authority generates their public and private key and a group public key is computed by multiplying all the authorities' public keys. Note that we operate in a distributed system setting (i.e., the Tor network), therefore, similar to PrivEx [32], one can easily instantiate decryption authorities.

Protocol. Using Count Sketch, we can collect a number of private readings from Hidden Service Directories (HSDir), and compute an approximation of the median. Each HSDir builds a Count Sketch, inserts its private values into it, encrypts it, and sends it to the authorities. These aggregate all sketches by homomorphically adding them element-wise, yielding an encrypted sketch summarizing the set of all HSDir values.

Once the authorities have computed the aggregate sketch, an interactive divide-and-conquer algorithm is applied to estimate the median given the range of its possible values is known. At each iteration, the number of sample values in the range is known, starting with the full range and all values received. The range is then halved and the sum of all elements falling in the first half of the range is jointly decrypted. If the median falls within first half of the range it is retained for the next iteration, otherwise the second half of the range is considered at the next iteration. The process stops once the range is a single element. Following the master theorem [21], we know that this process converges in $O(\log n)$ steps, for n elements in the domain of the values/median. Due to frequency estimations for the ranges using Count Sketches that provide noisy estimates, we expect this median to be close, but possibly not exactly the same as the true sample median, depending on the Count Sketch parameters δ and ϵ.

Output Privacy. Note that this process is not "perfectly" private in a traditional secure computation setting, as the volume of reported values falling within the intermediate ranges considered is leaked. This may be dealt with in two ways: (1) the leakage may be considered acceptable and the algorithm run as described, or (2) the technique can be enhanced to provide differential privacy by adding noise to each intermediate value.

Differentially Private Estimates. The sensitivity [31] of the estimates in any range of values using the Count Sketch is at most d, since each HSDir contribution increases by at most 1 in at most d values into the $d \cdot w$ Count Sketch table. Therefore, we can achieve ϵ-differential privacy if we add, to each decrypted value, noise from a Laplace distribution with mean zero and variance $\xi \cdot d/\epsilon$, where ξ is the number of decrypted intermediate results and ϵ the differential privacy parameter. However, doing so may result in the divide-and-conquer algorithm mis-estimating the range in which the median lies, and results in further mistakes in the final median estimate. (As discussed in Section II-C, although we use ϵ to denote a parameter for both Count Sketch and differential privacy, it is clear from the context which one it relates to.)

B. Implementation and Evaluation

We implement and evaluate the proposed scheme aiming to: (i) estimate the trade-off between size of the sketch and the accuracy of the median computation, (ii) evaluate the cost of cryptographic computation and communication overheads, and (iii) assess the trade-off between the accuracy of the median and the quality of protection that may be achieved through the differentially private mechanism.

For our evaluation, we instantiate AH-ECC using the NIST-P224 curve as provided by the OpenSSL library and its optimizations by Käsper [46]. Our implementation of the cryptographic core of the private median scheme amounts to 300 lines of Python code using the *petlib* OpenSSL wrapper[10], and another 350 lines of Python include unit tests and measurement code. All experiments have been performed on a Xubuntu Trusty (Ubuntu 14.04.2 LTS) Linux VM, running on a 64 bit Windows 7 host (CPU i7-4700MQ, 2.4Ghz, 16GB RAM). Our Python implementation is easily pluggable as part of the Tor infrastructure and does not require changes within the Tor (C-based) core functionalities.

We first illustrate the performance and accuracy of estimating the median using this technique with both sketch parameters ϵ and δ equal to either 0.25 or 0.05 against the London Atlas Dataset[11] in Table II (see Appendix). The error rate is computed as the absolute value of difference between the estimated and true median divided by the true median.

Further results are presented on an experimental setup that uses as a reference problem the median estimation in a set of 1,200 sample values, drawn from a mixture distribution: 1,000 values from a Normal distribution with mean 300 and variance

[10] https://github.com/gdanezis/petlib

[11] http://data.london.gov.uk/dataset/ward-profiles-and-atlas

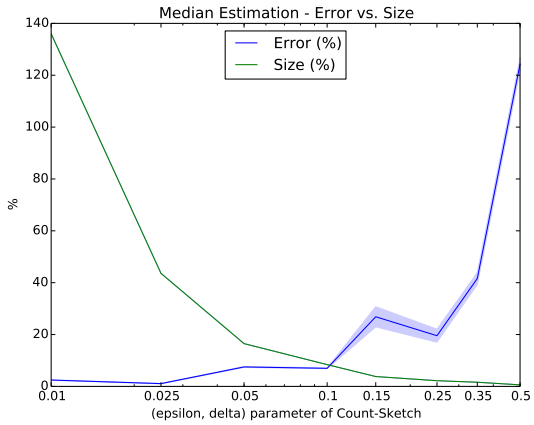


Figure 10: Count Sketch size versus estimation quality.

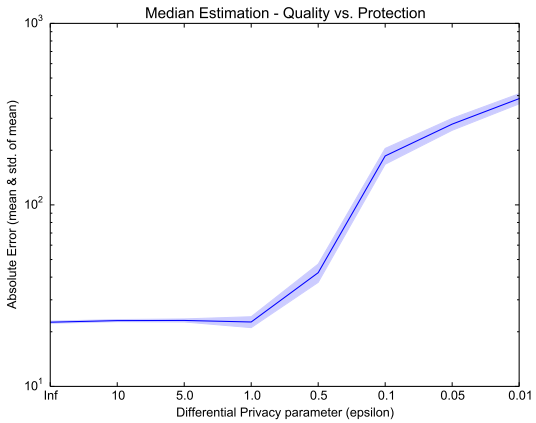


Figure 11: Quality versus differential privacy protection.

25, and 200 values drawn from a Normal distribution with mean 500 and variance 200. This reference problem closely matches the settings of the Tor project both in terms of the range of vales (assumed to be within $[0, 1000]$) and the number of samples [32].

Quality vs. Size. Figure 10 illustrates the trade-off between the quality of the estimation of the median algorithm and the size overhead of the Count Sketch. The size overhead (green slim line) is computed as the number of encrypted elements in the sketch as compared with the number of elements in the range of the median (1,000 for our reference problem). The estimation accuracy (blue broader line) is represented as the fraction of the absolute deviation of the estimate from the real value over the real sample median (light blue region represents the standard deviation of the mean over 40 experiments for each datapoint). Thus both qualities can be represented as percentages.

The trade off between the size of the sketch and the accuracy of the estimate is evident: as the sketch size reaches a smaller fraction of the total possible number of values, the error becomes larger than the range of the median. Thus, Count Sketch with parameters $\epsilon, \delta < 0.025$ are unnecessary, since they do not lead to a reduction of the information that needs to be transmitted from each client to the authorities; conversely, for $0.15 < \epsilon, \delta$ the estimate of the median deviates by more than 20% of its true value making it highly unreliable.

For all subsequent experiments, we consider a Count Sketch with values $\epsilon = \delta = 0.05$, leading to $d = 3$ and $w = 55$. As outlined in Figure 10, this represents a good trade-off between the size of the Count Sketch (16.5% of transmitting all values) and the error.

True Size and Performance. When implemented using NIST-P224 curves, the reference Count Sketch may be serialized in 10,898 bytes. Each Count Sketch takes 0.001 sec to encrypt at each HSDir, and it takes 1.456 seconds to aggregate 1,200 sketches at each authority (0.001 sec per sketch). As expected, from the range of the reference problem, 10 decryption iterations are sufficient to converge to the median (therefore $\xi = 10$). The number of homomorphic additions for each decryption round is linear in the range of the median and their total computational cost is the same order of magnitude as a full Count Sketch encryption. It is clear from these figures that the computational overhead of the proposed technique is eminently practical, and the bandwidth overhead acceptable.

Quality vs. Differential Privacy Protection. Figure 11 illustrates the trade-off between the quality of the median estimation and the quality of differential privacy protection. The x-axis represents the ϵ parameter of the differentially private system, and the y-axis the absolute error between the estimate and the true sample median. Differential privacy with parameter $\epsilon = 0.5$ can be provided without significantly affecting the quality of the median estimate. However, for $\epsilon < 0.5$ the volume of the error grows exponentially (note the log scale of the x-axis). While the exact value of a meaningful ϵ parameter is often debated in the literature, we conclude that the mechanism only provides a limited degree of protection, and no ability to readily tune up protection: utility degrades very rapidly as the security parameter ϵ decreases.

VI. Related Work

This section reviews prior work on privacy-preserving techniques applied to data aggregation, recommender systems, machine learning, participatory sensing, as well as efficient data structures for succinct representation.

A. Privacy-Preserving Aggregation

Kursawe et al. [47] introduce a few cryptographic constructions to aggregate energy consumptions in the context of smart metering, relying on Diffie-Hellman, bilinear maps, and a *"low overhead"* protocol where meters' encryption keys sum up to zero. Our schemes for the private recommender system (Section III) and location prediction (Section IV) rely on a protocol inspired by [47]'s "low overhead" protocol, but perform private aggregation using succinct data representation rather than the raw inputs. Using Count-Min Sketch [22], we reduce computation and communication overhead incurred by each user from linear to logarithmic in the size of the input. We also show how to recover from node failures, i.e., in our schemes, the aggregator can still retrieve the statistics (and train models) even when a subset of users go offline or fail to report data.

Castelluccia et al. [13] propose a new homomorphic encryption to allow intermediate wireless sensor nodes to aggregate encrypted data gathered from other nodes. Shi et al. [61] combine private aggregation with differential privacy supporting the aggregation of encrypted perturbed readings reported by the meters. Individual amounts of random noise cancel each other out during aggregation, except for a specific amount that guarantees computational differential privacy. Their protocol is also so that encryption keys sum up to zero but, unlike ours, requires solving a discrete logarithm and the presence of a trusted dealer. Jawurek et al. [45] propose a privacy-friendly aggregation scheme with robustness against missing user inputs, by including additional authorities that facilitate the protocol but do not learn any secrets or inputs. However, at least one of the authorities has to be honest, i.e., if all collude, the protocol does not provide any privacy guarantee. Chan et al. [15] also provides fault tolerance by extending [61]'s protocol, however, with a poly-logarithmic penalty. Additional, more loosely related, private aggregation schemes include [9, 13, 33].

A combination of homomorphic encryption and differential privacy has been explored by Chen et al. [19], allowing third parties to gather web analytics. Users encrypt their data using the data aggregator public key and send them to a proxy, who adds noise to the ciphertexts and forwards the results to the data aggregator. The latter computes the aggregates after decrypting each individual contribution. However, this scheme introduces a large overhead both in terms of communication (one KB per single bit of user data) and computation (one public key operation per single bit). In the same line of work, Akkus et al. [4] propose a system providing differential privacy guarantees. Their scheme scales better than [19] as it requires users to encrypt fewer bits per query, but still relies on expensive public-key crypto operations. In [18], the authors propose a scheme based on a similar trust model as [19] but with an enhanced scalability by using simple exclusive-or (XOR) operations rather than public key operations. However, their proposal still relies on honest-but-curious servers that do not collude with each other.

Erlingsson et al. [34] introduce RAPPOR, which enables the collection of browser statistics on values and strings provided by a large number of clients (e.g. homepage settings, running processes, etc.), including categories, frequencies, and histograms. RAPPOR supports privacy-preserving data-collection mechanism by relying on randomized responses via input perturbation, aiming to guarantee local differential privacy for individual reports. This, however, requires millions of users in order to obtain approximate answers to queries.

Finally, Elahi et al. [32] present a protocol for privately computing mean statistics on Tor traffic. They introduce two ad-hoc protocols relying, respectively, on secret sharing and distributed decryption. By contrast, our application for gathering private statistics for Tor enables the computation of the median statistics on traffic generated by Tor hidden services – which constituted an open problem [39] – by relying on additively homomorphic encryption and differential privacy.

B. Privacy-preserving Recommender Systems

McSherry and Mironov [51] propose a privacy-preserving recommender system that relies on trusted computing, while Cissée and Albayrak [20] use differential privacy to add privacy guarantees to a few algorithms presented during the Netflix Prize competition. Our private recommender system differs from theirs as we do not rely on trusted computing or differential privacy, but leverage a privacy-friendly aggregation cryptographic protocol and Count-Min Sketch.

Homomorphic encryption based techniques have also been used to perform other machine learning operations on encrypted data, including matrix factorization [56], linear classifiers [11, 40], and decision trees [12]. Building a cloud-based model from multiple user datasets has been also addressed in [49], which explores the feasibility of Fully Homomorphic Encryption (FHE) based techniques. However, at the moment, FHE operations are still prohibitively expensive.

C. Participatory Sensing

Mood et al. [54] propose a privacy-preserving participatory sensing application which allows users to locate nearby friends without disclosing exact locations, via secure function evaluation [65], but do not address the problem of scaling to large streams/number of users. De Cristofaro and Soriente [27] introduce a privacy-enhanced distributed querying infrastructure for participatory and urban sensing systems. Work in [24] and [43] provide either k-anonymity [63] and l-diversity [50] to guarantee anonymity of users through Mix Network techniques [17]. However, their techniques are not provably-secure and they only provide partial confidentiality. Then, [36] suggest data perturbation in a known community for computing statistics and protecting anonymity. Trusted Platform Modules (TPMs) are instead used in [37] and [29] to protect integrity and authenticity of user contents.

In a way, we also address the problem of participatory sensing privacy by proposing a scalable and provable secure technique for collecting user-generated streams of data involving a large number of users.

D. Privacy and Succinct Data Representation

Mir et al. [52] present an efficient scheme guaranteeing differential privacy of data analyses (even when the internal memory of the algorithm may be compromised), using a data structure similar to the Count-Min Sketch to estimate heavy hitters. Work in [14, 42] address the problem of finding heavy hitters' histograms while preserving privacy using a differentially private protocol. Then, [6] addresses the case where individual users randomize their own data and then send differentially private reports to an untrusted server handling reports aggregation. Other proposals combine differential privacy and Count-Min Sketch to obtain aggregate information about vehicle traffic [53] as well as summaries of sparse databases [23].

Ashok et al. [5] present a privacy-preserving protocol for computing the set-union cardinality among several parties using Bloom filters [10]. However, their proposal is insecure, as shown by [64], who also introduces a novel Bloom filter based protocol for set-union and set-intersection cardinality. Lin et al. [48] improve the performance of [55]'s protocol for private proximity testing by reducing the problem to simple equality testing (instead of the more expensive private-preserving threshold set intersection). They use a concise

representation of "location tags", by generating, via shingling, concise sketches—in their context, short strings representing the set of broadcast messages received.

In summary, to the best of our knowledge, our work is the first to show how to combine Count-Min Sketch and privacy-friendly data aggregation to build a private estimated model used for recommendations as well as prediction of future locations. Also, our scheme for Tor hidden services statistics, which combines Count Sketch, additively homomorphic threshold decryption, and differential privacy, is the first to tackle the problem of efficiently computing the median statistics.

VII. Conclusion

This paper presented efficient techniques for privately and efficiently collecting statistics by relying on private data aggregation protocols and succinct data structures. These allowed us to reduce the communication and computation complexity incurred by each data source from linear to logarithmic in the size of the input but only introduced a limited, upper-bounded error in the quality of the statistics.

Our techniques support different trust, robustness, and deployment models and can be applied to a number of interesting real-world problems where aggregate statistics can be used to train models. We presented the design and deployment of a private recommender system for streaming services and a private location prediction service. Our server-side implementation as a JavaScript web application allows developers to easily incorporate it in their projects, while user-side is supported both in the browser (thus requiring users to install no additional software) and in Android. We also designed and implemented (in Python) a scheme for computing the median statistics of Tor hidden services in a privacy-friendly way.

As part of future work, we plan to apply our private recommender system to the BBC news apps for Android, conduct a test deployment of the private location prediction service with a local mass transit operator, and extend our protocols to privately consolidate data shared by different sources [35]. We are also working on releasing a comprehensive framework supporting large-scale privacy-preserving aggregation *as a service*.

Acknowledgements. We would like to thank Chris Newell and Michael Smethurst from the BBC and Aaron Johnson from US Naval Research Labs for motivating our work, respectively, on privacy-preserving recommendation and median statistics in Tor. We are also grateful to Mirco Musulesi, Licia Capra, and Apostolos Pyrgelis for providing feedback and useful comments. Luca Melis and Emiliano De Cristofaro are supported by a Xerox's University Affairs Committee award on "Secure Collaborative Analytics." and "H2020-MSCA-ITN-2015" Project Privacy&Us (ref. 675730). George Danezis is supported in part by EPSRC Grant "EP/M013286/1" and H2020 Grant PANORAMIX (ref. 653497).

References

[1] BBC iPlayer. http://www.bbc.co.uk/iplayer.

[2] Count-Min Sketch and its applications. https://sites.google.com/site/countminsketch/, 2015.

[3] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, 2005.

[4] I. E. Akkus, R. Chen, M. Hardt, P. Francis, and J. Gehrke. Non-tracking Web Analytics. In *ACM CCS*, 2012.

[5] V. G. Ashok and R. Mukkamala. A Scalable and Efficient Privacy Preserving Global Itemset Support Approximation Using Bloom Filters. In *DBSEC*, 2014.

[6] R. Bassily and A. Smith. Local, Private, Efficient Protocols for Succinct Histograms. In *STOC*, 2015.

[7] J. Benaloh. Dense probabilistic encryption. In *SAC*, 1994.

[8] D. J. Bernstein, N. Duif, T. Lange, P. Schwabe, and B.-Y. Yang. High-speed High-Security Signatures. In *CHES*, 2011.

[9] I. Bilogrevic, J. Freudiger, E. De Cristofaro, and E. Uzun. What's the Gist? Privacy-Preserving Aggregation of User Profiles. In *ESORICS*, 2014.

[10] B. H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7), 1970.

[11] J. W. Bos, K. Lauter, and M. Naehrig. Private predictive analysis on encrypted medical data. *Journal of Biomedical Informatics*, 2014.

[12] R. Bost, R. A. Popa, S. Tu, and S. Goldwasser. Machine learning classification over encrypted data. Technical report, Cryptology ePrint Archive Report 2014/331, 2014.

[13] C. Castelluccia, E. Mykletun, and G. Tsudik. Efficient aggregation of encrypted data in wireless sensor networks. In *Mobiquitous*, 2005.

[14] T.-H. H. Chan, M. Li, E. Shi, and W. Xu. Differentially private continual monitoring of heavy hitters from distributed streams. In *PETS*, 2012.

[15] T.-H. H. Chan, E. Shi, and D. Song. Privacy-preserving stream aggregation with fault tolerance. In *FC*, 2012.

[16] M. Charikar, K. Chen, and M. Farach-Colton. Finding frequent items in data streams. In *ICALP*, 2002.

[17] D. L. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of ACM*, 24(2), 1981.

[18] R. Chen, I. E. Akkus, and P. Francis. SplitX: High-performance Private Analytics. In *SIGCOMM*, 2013.

[19] R. Chen, A. Reznichenko, P. Francis, and J. Gehrke. Towards statistical queries over distributed private user data. In *NSDI*, 2012.

[20] R. Cissée and S. Albayrak. An agent-based approach for privacy-preserving recommender systems. In *IFAAMAS*, 2007.

[21] T. H. Cormen, C. E. Leiserson, R. Rivest, and C. Stein. *Introduction to algorithms*. MIT Press Cambridge, 2001.

[22] G. Cormode and S. Muthukrishnan. An Improved Data Stream Summary: The Count-Min Sketch and Its Applications. *Journal of Algorithms*, 2005.

[23] G. Cormode, C. Procopiuc, D. Srivastava, and T. T. Tran. Differentially private summaries for sparse data. In *ICDT*, 2012.

[24] C. Cornelius, A. Kapadia, D. Kotz, D. Peebles, M. Shin, and N. Triandopoulos. AnonySense: Privacy-aware people-centric sensing. In *Mobisys*, 2008.

[25] T. M. Cover and P. E. Hart. Nearest Neighbor Pattern Classification. *IEEE Transactions on Information Theory*, 1967.

[26] S. Curtis. Telegraph – Quarter of the world will be using smartphones in 2016. http://www.telegraph.co.uk/technology/mobile-phones/11287659/Quarter-of-the-world-will-be-using-smartphones-in-2016.html.

[27] E. De Cristofaro and C. Soriente. Extended capabilities for a privacy-enhanced participatory sensing infrastructure. *IEEE TIFS*, 8(12):2021–2033, 2013.

[28] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second-generation Onion Router. Technical report, DTIC Document, 2004.

[29] A. Dua, N. Bulusu, W. Feng, and W. Hu. Towards trustworthy participatory sensing. In *HotSec*, 2009.

[30] C. Dwork. Differential Privacy. In *ICALP*, 2006.

[31] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating Noise to Sensitivity in Private Data Analysis. In *TCC*, 2006.

[32] T. Elahi, G. Danezis, and I. Goldberg. PrivEx: Private Collection of Traffic Statistics for Anonymous Communication Networks. In *ACM CCS*, 2014.

[33] Z. Erkin and G. Tsudik. Private Computation of Spatial and Temporal Power Consumption with Smart Meters. In *ACNS*, 2012.

[34] Ú. Erlingsson, V. Pihur, and A. Korolova. RAPPOR: Randomized Aggregatable Privacy-Preserving Ordinal Response. In *ACM CCS*, 2014.

[35] J. Freudiger, E. De Cristofaro, and A. Brito. Controlled Data Sharing for Collaborative Predictive Blacklisting. In *DIMVA*, 2015.

[36] R. Ganti, N. Pham, Y. Tsai, and T. Abdelzaher. PoolView: stream privacy for grassroots participatory sensing. In *SenSys*, 2008.

[37] P. Gilbert, L. Cox, J. Jung, and D. Wetherall. Toward trustworthy mobile sensing. In *HotMobile*, 2010.

[38] P. Golle and K. Partridge. On the Anonymity of Home/Work Location Pairs. In *Pervasive computing*, 2009.

[39] D. Goulet, A. Johnson, G. Kadianakis, and K. Loesing. Hidden-Service statistics Reported by Relays. https://research.torproject.org/techreports/hidden-service-stats-2015-04-28.pdf, 2015.

[40] T. Graepel, K. Lauter, and M. Naehrig. ML confidential: Machine Learning on Encrypted Data. In *ICISC*, 2012.

[41] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl. Evaluating Collaborative Filtering Recommender Systems. *ACM Transactions on Information Systems*, 2004.

[42] J. Hsu, S. Khanna, and A. Roth. Distributed Private Heavy Hitters. In *ICALP*, 2012.

[43] K. Huang, S. Kanhere, and W. Hu. Preserving privacy in participatory sensing systems. *Computer Communications*, 33(11), 2010.

[44] B. Hull, V. Bychkovsky, Y. Zhang, K. Chen, M. Goraczko, A. Miu, E. Shih, H. Balakrishnan, and S. Madden. CarTel: A Distributed Mobile Sensor Computing System. In *SenSys*, 2006.

[45] M. Jawurek and F. Kerschbaum. Fault-Tolerant Privacy-Preserving Statistics. In *PETS*, 2012.

[46] E. Käsper. Fast Elliptic Curve Cryptography in OpenSSL. In *FC*, 2012.

[47] K. Kursawe, G. Danezis, and M. Kohlweiss. Privacy-friendly Aggregation for the Smart-grid. In *PETS*, 2011.

[48] Z. Lin, D. F. Kune, and N. Hopper. Efficient Private Proximity Testing with GSM Location Sketches. In *FC*, 2012.

[49] A. López-Alt, E. Tromer, and V. Vaikuntanathan. On-The-Fly Multiparty Computation on the Cloud via Multi-Key Fully Homomorphic Encryption. In *STOC*, 2012.

[50] A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam. l-diversity: Privacy beyond k-anonymity. *ACM TKDD*, 1(1), 2007.

[51] F. McSherry and I. Mironov. Differentially Private Recommender Systems: Building Privacy Into the Net. In *KDD*, 2009.

[52] D. Mir, S. Muthukrishnan, A. Nikolov, and R. N. Wright. Pan-Private Algorithms via Statistics on Sketches. In *PODS*, 2011.

[53] A. Monreale, W. Wang, F. Pratesi, S. Rinzivillo, D. Pedreschi, G. Andrienko, and N. Andrienko. Privacy-Preserving Distributed Movement Data Aggregation. In *Geographic Information Science at the Heart of Europe*, 2013.

[54] B. Mood, D. Gupta, K. Butler, and J. Feigenbaum. Reuse it or lose it: more efficient secure computation through reuse of encrypted values. In *ACM CCS*, 2014.

[55] A. Narayanan, N. Thiagarajan, M. Lakhani, M. Hamburg, and D. Boneh. Location Privacy via Private Proximity Testing. In *NDSS*, 2011.

[56] V. Nikolaenko, S. Ioannidis, U. Weinsberg, M. Joye, N. Taft, and D. Boneh. Privacy-Preserving Matrix Factorization. In *ACM CCS*, 2013.

[57] V. Pejovic and M. Musolesi. Anticipatory Mobile Computing: A Survey of the State of the Art and Research Challenges. *ACM Computing Surveys*, 2015.

[58] P. Resnick and H. R. Varian. Recommender Systems. *Communications of the ACM*, 1997.

[59] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based Collaborative Filtering Recommendation Algorithms. In *WWW*, 2001.

[60] S. Scellato, M. Musolesi, C. Mascolo, V. Latora, and A. T. Campbell. NextPlace: A Spatio-Temporal Prediction Framework for Pervasive Systems. In *Pervasive Computing*, 2011.

[61] E. Shi, T.-H. H. Chan, E. G. Rieffel, R. Chow, and D. Song. Privacy-Preserving Aggregation of Time-Series Data. In *NDSS*, 2011.

[62] F. Soldo, A. Le, and A. Markopoulou. Predictive blacklisting as an implicit recommendation system. In *INFOCOM*, 2010.

[63] L. Sweeney. k-Anonymity: A model for Protecting Privacy. *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, 2002.

[64] J. Tillmanns. Privately computing set-union and set-intersection cardinality via bloom filters. In *ACISP*, 2015.

[65] A. C.-C. Yao. Protocols for secure computations. In *FOCS*, volume 82, 1982.

	Median ($\epsilon, \delta = 0.25$)	Error (%)	Median ($\epsilon, \delta = 0.05$)	Error (%)	Truth
Population - 2015	15143.2	11.3	13215.4	2.8	13600.0
Children aged 0-15 - 2015	2970.8	12.1	2627.6	0.8	2650.0
Working-age (16-64) - 2015	9592.0	2.0	8843.2	5.9	9400.0
Older people aged 65+ - 2015	1284.6	11.4	1345.0	7.2	1450.0
% All Children aged 0-15 - 2015	21.9	10.7	20.1	1.3	19.8
% All Working-age (16-64) - 2015	70.7	5.0	68.8	2.2	67.3
% All Older people aged 65+ - 2015	15.2	37.1	12.0	7.8	11.1
Mean Age - 2013	38.6	8.8	36.9	3.8	35.5
Median Age - 2013	37.7	10.8	35.7	5.1	34.0
Area - Square Kilometres	0.6	68.1	1.6	16.9	1.9
Population density (persons per sq km) - 2013	10231.3	44.8	5792.9	18.0	7067.0
% BAME - 2011	45.6	26.3	35.7	1.0	36.1
% Not Born in UK - 2011	40.1	7.6	40.1	7.6	37.3
% English is First Language of no one in househ...	16.9	41.7	11.8	0.9	11.9
General Fertility Rate - 2013	73.3	14.4	66.8	4.1	64.1
Male life expectancy -2009-13	84.1	5.7	79.6	0.0	79.6
Female life expectancy -2009-13	87.0	3.5	84.9	0.9	84.1
Rate of All Ambulance Incidents per 1,000 popul...	52.5	54.9	98.6	15.3	116.3
Rates of ambulance call outs for alcohol relate...	0.1	78.0	1.0	74.0	0.6
Number Killed or Seriously Injured on the roads...	3.0	1.3	3.5	16.7	3.0
In employment (16-64) - 2011	6532.8	7.0	5843.7	4.2	6103.0
Employment rate (16-64) - 2011	68.5	2.0	70.8	1.3	69.9
Rate of new registrations of migrant workers - ...	42.9	10.7	34.5	11.1	38.8
Number of properties sold - 2013	169.3	1.4	149.8	10.3	167.0
Modelled Household median income estimates 2011/12	31802.6	2.2	29589.3	9.0	32509.0
Number of Household spaces - 2011	5619.1	5.4	5025.9	5.7	5332.0
% detached houses - 2011	2.4	44.7	1.6	62.2	4.3
% semi-detached houses - 2011	29.0	70.6	16.7	1.5	17.0
% terraced houses - 2011	29.4	39.8	21.1	0.6	21.0
% Flat, maisonette or apartment - 2011	53.1	15.1	49.7	7.9	46.1
% Households Owned - 2011	57.3	18.4	53.3	10.2	48.4
% Households Social Rented - 2011	26.0	27.5	19.9	2.4	20.4
% Households Private Rented - 2011	30.9	26.5	26.6	9.1	24.4
% dwellings in council tax bands A or B - 2011	21.2	79.9	10.4	12.2	11.8
% dwellings in council tax bands C, D or E - 2011	63.7	7.5	71.6	3.9	68.9
% dwellings in council tax bands F, G or H - 2011	0.3	96.7	1.4	82.6	8.1
Claimant Rate of Incapacity Benefit - 2014	1.8	80.0	0.9	10.0	1.0
Claimant Rate of Income Support - 2014	4.4	119.6	2.3	16.8	2.0
Claimant Rate of Employment Support Allowance -...	6.9	65.3	4.7	13.0	4.2
Rate of JobSeekers Allowance (JSA) Claimants - ...	5.0	34.6	3.1	16.6	3.7
% dependent children (0-18) in out-of-work hous...	22.2	19.6	19.1	2.6	18.6
% of households with no adults in employment wi...	8.7	67.2	5.3	1.2	5.2
% of lone parents not in employment - 2011	51.9	11.2	47.5	1.6	46.7
(ID2010) - Rank of average score (within London...	366.3	17.4	301.6	3.3	312.0
(ID2010) % of LSOAs in worst 50% nationally - 2010	-6.4	107.7	99.2	19.5	83.0
Average GCSE capped point scores - 2013	369.0	6.0	349.4	0.4	348.0
Unauthorised Absence in All Schools (%) - 2013	1.7	53.5	0.8	26.2	1.1
% with no qualifications - 2011	20.8	19.1	18.8	7.2	17.5
% with Level 4 qualifications and above - 2011	44.4	25.1	39.1	10.1	35.5
A-Level Average Point Score Per Student - 2012/13	715.3	5.7	668.4	1.3	676.9
A-Level Average Point Score Per Entry; 2012/13	215.0	3.1	210.8	1.1	208.5
Crime rate - 2013/14	1163.6	1598.7	47.8	30.3	68.5
Violence against the person rate - 2013/14	1.2	92.5	10.5	35.6	16.3
Robbery rate - 2013/14	1.6	31.8	0.1	94.7	2.3
Theft and Handling rate - 2013/14	-3.5	113.7	11.4	55.6	25.6
Criminal Damage rate - 2013/14	9.1	43.8	5.9	6.6	6.3
Drugs rate - 2013/14	-9.3	321.4	2.8	33.8	4.2
% area that is open space - 2014	30.1	28.3	19.3	17.9	23.5
Cars per household - 2011	1.6	99.4	0.5	35.0	0.8
Average Public Transport Accessibility score - ...	6.8	99.6	4.4	28.9	3.4
% travel by bicycle to work - 2011	12.0	343.9	3.0	12.5	2.7
Turnout at Mayoral election - 2012	38.1	11.5	35.0	2.3	34.2

TABLE II: Median estimation with 22 ciphertexts ($d = 2$, $w = 11$, $\epsilon, \delta = 0.25$) and 165 ciphertexts ($d = 3$, $w = 55$, $\epsilon, \delta = 0.05$) on the London Atlas Dataset.