Ad Hoc Networks 80 (2018) 16–30

Contents lists available at ScienceDirect

Ad Hoc Networks

journal homepage: www.elsevier.com/locate/adhoc

K Privacy: Towards improving privacy strength while preserving utility

Josh Joy*, Dylan Gray, Ciaran McGoldrick, Mario Gerla

University of California - Los Angeles, Los Angeles CA 90095, USA

ARTICLE INFO

Article history:
Received 19 February 2018
Accepted 21 May 2018
Available online 31 May 2018

Keywords:
Internet of Vehicles
Location privacy
Privacy-preserving data collection
Differential privacy

ABSTRACT

Future autonomous vehicles will generate, collect, aggregate and consume significant volumes of data as key gateway devices in emerging Internet of Things scenarios. While vehicles are widely accepted as one of the most challenging mobility contexts in which to achieve effective data communications, less attention has been paid to the privacy of data emerging from these vehicles. The quality and usability of such privatized data will lie at the heart of future safe and efficient transportation solutions.

In this paper, we present the K Privacy mechanism. K Privacy is to our knowledge the first such mechanism that enables data creators to submit multiple contradictory responses to a query, whilst preserving utility measured as the absolute error from the actual original data. The functionalities are achieved in both a scalable and secure fashion. For instance, individual location data can be obfuscated while preserving utility, thereby enabling the scheme to transparently integrate with existing systems (e.g. Waze). A new cryptographic primitive Function Secret Sharing is used to achieve non-attributable writes and we show an order of magnitude improvement from the default implementation.

1. Introduction

Researchers are becoming increasingly interested in studying smart city activities and interactions, such as pedestrians, drivers and traffic, city resources (e.g., energy) and city environment (e.g., pollution, noise). These studies are commonly based on Open Shared Data made available by several Smart City testbeds around the country. To this end, Open Data Science enables researchers to collect the data, analyze and process it with Data Mining and Machine Learning techniques and create accurate models that allow them to credibly validate smart city design methodologies.

There is a growing demand for researchers and manufacturers to deploy their technologies in real vehicles, roads and cities. Rather than requiring each stakeholder working in the area to create new solutions for securing their experimental or vehicular infrastructure, we propose a highly scalable, common "privacy" infrastructure that enables the non-attributable dissemination of data, whilst simultaneously conserving and preserving the critical information content properties required for value added service provision by aggregators and upstream analysts.

As smart city experiments are frequently performed on massive scale with public participants, it is prudent to surmise that some may seek to exploit the data for illicit purposes. Publicly releasing data with exact answers to queries (without sanitization) has resulted in numerous privacy violations and attacks, e.g., relating to unintentional medical data disclosure of high profile governors [1], shutdowns of seemingly innocuous open data machine learning competitions [2], location tracking attacks and DoS attacks [3], and unintentional sharing of mobility patterns of high profile US citizens with foreign governments [4]. k-anonymity introduced by Sweeney in 1998 [1] was among the first privacy techniques to address publicly releasing data in a privacy-preserving manner. Roughly speaking, k-anonymity seeks to blend a single data owner's personal attribute with at least k other data owners such that the single data owner is indistinguishable from $k-1$ other data owners. For example, if a particular data owner's record reporting a particular disease is publicly released with 1000 other data owners records with the same disease, the data owner is indistinguishable from 999 other data owners.

However, there are known impossibility results for attempts to preserve privacy while releasing exact answers. Dinur and Nissim showed in 2003 that it is impossible to reveal exact aggregation information while simultaneously preserving privacy (against a polynomial adversary) [5]. Thus, perturbation *must* be injected in order to guarantee privacy (privacy defined as an adversary is unable to determine the value of a targeted individual with a probability greater than 50%) [5]. Alternatively, noiseless privacy has been proposed which does not add additional privacy noise. However, noiseless privacy requires strong adversary assumptions such as the adversary has limited or no auxiliary information and is restrictive regarding multiple queries and composability [6]. Thus, in order to have Open Shared Data, on Smart City or larger scale,

* Corresponding author.
E-mail addresses: jjoy@cs.ucla.edu (J. Joy), dylangray9@cs.ucla.edu (D. Gray), ciaran@cs.ucla.edu (C. McGoldrick), gerla@cs.ucla.edu (M. Gerla).

https://doi.org/10.1016/j.adhoc.2018.05.009
1570-8705/

there will be some notion of absolute error or distance from the ground truth due to the required perturbation.

Differential privacy is one such privacy definition which enables realisation of this concept, and it is currently viewed as the gold standard. Roughly speaking, differential privacy says that the ability of an adversary to inflict harm should be essentially independent of whether any individual opts in to or out of, the dataset [7]. Thus, a data owner may safely utilise differential privacy techniques when sharing their personal data, as it enables them control insight into their personal information.

The Laplace mechanism satisfies differential privacy by adding privacy noise independent of the database size [8] by drawing privacy noise from the Laplace distribution. The Laplace mechanism is calibrated to the max difference between any two rows in the database. That is, the noise is sufficient to protect the max leakage that any particular data owner induces. For example, first a service aggregates all the data owners truthful responses. Then, the aggregation service draws from the Laplace distribution by calibrating the variance according to the desired privacy strength. Drawing from other distributions such as Gaussian also satisfies differential privacy, though the Laplace mechanism provides differential privacy and Gaussian provides (ϵ, δ)-differential privacy [9].

Now, consider if the Laplace mechanism is used and we desire to improve the privacy strength. The privacy strength of a given mechanism is determined by the epsilon ϵ value, which corresponds to the privacy loss measured as the ratio of the max difference between any two differing outputs. Naturally, it follows that increasing the value of ϵ adds privacy noise mitigating any utility benefits as the privacy noise increases.

Another observation is that the use of the Laplace mechanism requires each individual to truthfully respond, relying on the output perturbation to provide privacy. This requires extra caution in the sensitive queries posed. For example, suppose we query everyone at the Brooklyn Bridge to understand how many people are currently at the Brooklyn Bridge. Regardless of the cryptographic technique or privacy mechanism used, the act of responding signals to an adversary that the data owner was indeed at the Brooklyn Bridge.

In this paper, we present the K Privacy mechanism. K Privacy achieves *scalable* privacy by increasing the size k of the participating data owners to provide privacy protection. The queried population is increased beyond those at the Brooklyn Bridge, say to the entire New York City metropolitan area. Thus, the act of responding no longer signals that the data owner is at the Brooklyn Bridge. The data owner plausibly is now anywhere in New York City. At the same time, K Privacy preserves the absolute error. The absolute error does not increase or expand due to the distortion of the underlying truthful population distribution (e.g., the query distorts and decreases the percentage from 100% of data owners at the Brooklyn Bridge to less than 1% of the New York City population is currently at the Brooklyn Bridge). Additionally, in our K Privacy mechanism, data owners perform cryptographic private writes which dissociate the identifier from the data value. The aggregation operators also perform multi-party computation (MPC) to protect against malicious data owners that try to corrupt the aggregate answer.

We evaluate the scalability and accuracy of our privacy-preserving approach utilizing a vehicular crowdsourcing scenario comprising of approximately one million records. In this dataset, each vehicle reports its location utilizing the California Transportation Dataset from magnetic pavement sensors (see Section 8.1). To demonstrate the efficiency and scalability of our approach, we crowdsource and privately write 128,000 vehicle (data owner) locations in under a minute with a key size of less than 15 KB, a square root reduction compared to the trivial solution whose key size would be linear in the number of data owners (i.e., the key size is the number of data owners times the message bit size). This key size reduction allows us to increase the database size to simultaneously accommodate hundreds of thousands of data owners.

This work demonstrates, for the first time, that personal data can be crowdsourced at scale with constant error(preserving the information content of interest to upstream aggregators and analysts), strong privacy guarantees, protected with scalable cryptographic private writes, and accurately disclosed. We believe this represents an exciting new contribution to open data science, driven by the need for privacy-preserving crowdsourcing in mobile cloud contexts (e.g., traffic management).

2. Related work

Privacy definitions. Differential privacy [8–11] has been proposed as a privacy definition such that anything that can be learned if a particular data owner is added to the database could have also been learned before the data owner was added. A data owner is thus "safe" to participate as statistical inferences amongst the aggregate are learned yet specific information regarding the individual is not learned.

However, careful consideration needs to be done when applying differentially private mechanisms in practice. For example, there is a drawback to the Laplace mechanism in graph datasets such as social networks [12,13] or vehicle commuting patterns. Even if a particular data owner does *not* participate, their friends that do participate leak information that can be used to deprivatize the targeted data owner (e.g., shadow profiles). For example, it is possible to learn political beliefs or sexual orientation even if a particular individual does not participate and maintain an active profile in an online social network. An adversary simply needs to analyze the similarity metrics amongst the social circles that a data owner participates in to understand politics beliefs or sexual orientation [14–18].

Furthermore, even if the graph structures of the social network are eventually anonymized and released, an adversary needs to simply participate and influence the graph structure (e.g., joining a social network) to learn and influence the actual social graph before it's privatized and released. Thus, to provide stronger protection a mechanism must also perturb the underlying *structure* of the data itself yet preserve accuracy as the underlying distribution structure becomes distorted.

Sampling can be applied to weaken the associations within a graph structure. This is achieved whereby responses are randomly discarded in order to reduce the the graph dependencies leaked by a targeted individuals connections. The severed connections reduces the social circle size and makes it challenging for the adversary to make similarity inferences from reduced social circles alone. Thus, it has been shown that the strength of privacy mechanisms are increased by applying sampling and reducing the privacy leakage [13,19–21].

Another popular technique which satisfies differential privacy is the randomized response mechanism, originally proposed in the 1960s [22,23]. Randomized response has been shown to be optimal in the local differentially private model [24] and is used by many companies today (e.g., Apple, Google [25]) due to its simplicity while satisfying the differential privacy guarantee.

However, these protocols, such as Rappor [25], require an inordinate amount of samples, yet still do not preserve utility. For example, even if 1 billion reports are collected, statistics from close to 1 million reports may not show up in the analysis. Thus, these type of local differentially private protocols are best suited for tracking heavy-hitters (e.g., counting the most commonly occurring elements in peaky power-law distributions) [26]. We elaborate further in Section 4.3.

Some protocols which leverage randomized response style mechanisms have made assumptions that the majority of the underlying truthful population truthfully responds "Yes" (e.g., the percentage is greater than 2/3 or 3/4) in order to preserve accuracy. However, it's not clear what privacy guarantees will be provided as any adversary is able to successfully guess with greater than 50% probability the value of any data owner in such a population. For example, suppose our query is how many home owners reside within 15 blocks from the beach, yet we ask only those home owners within 20 blocks from the beach.

Zero-knowledge privacy [12] is a cryptographically influenced privacy definition that is strictly stronger than differential privacy. Zero-knowledge private mechanisms add a sampling step to distort the underlying structure. For example, crowd-blending privacy [13] is weaker than differential privacy; however, with a pre-sampling step, satisfies both differential privacy and zero-knowledge privacy. However, these mechanisms are suited for the trusted centralized system model. A single database breach would violate all previous privacy guarantees, as each data owner's exact and unprotected answer is recorded in the database. Additionally, zero-knowledge private mechanisms rely on aggressive sampling to achieve strong privacy, which significantly degrades accuracy estimations.

Distributional privacy [27] is a privacy mechanism which says that the released aggregate information only reveals the underlying ground truth distribution and nothing more. Each data owner is protected by the randomness of the other randomly selected data owners rather than by adding explicit privacy noise to the output. The indistinguishability from the underlying distribution protects individual data owners and is strictly stronger than differential privacy. However, it is computationally inefficient though can work over a large class of queries known as Vapnik-Chervonenkis (VC) dimension.

Sampling. Sampling whereby a centralized aggregator randomly discards responses has been previously formulated as a mechanism to amplify privacy [13,19–21,28]. The intuition is that when sampling approximates the original aggregate information, an attacker is unable to distinguish when sampling is performed and which data owners are sampled. These privacy mechanisms range from sampling without a sanitization mechanism, sampling to amplify a differentially private mechanism, sampling that tolerates a bias, and even sampling a weaker privacy notion such as k-anonymity to amplify the privacy guarantees.

However, sampling alone has several issues. First, data owners are not protected by plausible deniability as data owners respond truthfully and never a contradictory "No" response. Second, the estimation of the underlying truthful "Yes" responses quickly degrades as we increase the population that truthfully responds "No", due to the sampling error.

Multi-party computation. Multi-party computation (MPC) is a secure computation model whereby parties jointly compute a function such that each party only learns the aggregate output and nothing more. However, MPC mechanisms that release the *exact* answer has no strong privacy guarantees against active privacy attacks, particularly when the data is publicly published.

A participant that does not perturb their responses and provides their *exact* answer is easily attacked by an adversary that knows the values of $n-1$ participants. For example, an adversary first runs a counting query that includes all n data owners and then runs a second counting query over $n-1$ data owners (the targeted data owner is the excluded row). Subtracting the two results *reveals* the value of the targeted data owner.

In contrast, the differential privacy model assumes a strong adversary that knows the $n-1$ data owner values. In this paper we combine MPC and differential privacy by introducing a sampling-based privacy mechanism that maintains constant error and show a performance optimization for a new cryptographic primitive named Function Secret Sharing [29].

Homomorphic cryptography. Homomorphic cryptography enables computation over encrypted data without ever requiring access to the plaintext data. However, applying homomorphic cryptography alone to compute exact aggregate statistics does not provide privacy protection and is vulnerable to the same $n-1$ adversarial attacks as multi-party computation. Typically, such cryptographic approaches are combined with differentially private mechanisms in order to protect data owners.

Private data upload. Wang et al. [30] employed and extended the function secret sharing primitive to enable efficient private information retrieval operations that protect the data owner's queries from being learned by the database servers. They proposed an optimization by using the Matyas-Meyer-Oseas one-way compression function as an alternative to the heavy AES operations for the *two party* case. Wang et al. achieves a 2.5x speedup by utilizing one-way compression functions. However, our K Privacy *also* demonstrates a *one order of magnitude* improvement over the default implementation of the function secret sharing protocol for the *multi-party* database aggregator scenario.

3. Threat model

The attack: an adversary can utilize the database size (number of participants) to deduce if a particular individual is included. However, the *exact* population (database) size or *exact* number of participating data owners is not published or released. This mitigates auxiliary attacks whereby the adversary uses the exact counts to reconstruct the database.

The attack: an adversary can individually inspect the responses of each data owner to ascertain their truthful response. However, we select sampling probabilities less than 50% so that an adversary does not gain an inference advantage of greater than 50%. We also require a distributed set of aggregators whereby at least one honest aggregator does not collude with the others (e.g., a privacy watchdog like the EFF).

The attack: only a single honest data owner (or very few data owners less than k) participate allowing an adversary to easily deprivatize the honest data owner. However, the aggregation parties proceed in epochs where they only combine their results if at least k data owners (a threshold that can be configured) participate. The honest aggregation party may refuse to share their results with the other aggregation parties thereby halting the protocol, if the number of participating data owners is below the desired threshold.

Differential privacy guarantee. The protocol we propose is a 2 round protocol. However, it is 2 rounds in the sense that 2 different values are uploaded by a single data owner. However, these values are not able to be linked to each other due to the cryptographic private write in Section 6.

The attack: a network adversary attempts to drop the upload of the 2nd round, in order to isolate the value of a single data owner in round 1 to deprivatize the data owner and learn their truthful value. However, round 1 and round 2 are sent as a tuple $\langle round_1, round_2 \rangle$, so if any round is dropped by the network adversary both rounds are dropped.

Pollution attacks. In this work we consider three different pollution attacks : (1) a malicious data owner who attempts to corrupt the private write by attempting to write to multiple rows of the database instead of a single row (2) a malicious data owner who answers a query with a single, large value in order to inflate the aggregate sum (3) a malicious data owner who repeatedly answers a query within a single epoch. More details can be found in Section 7.

In order for the aggregators to safely accept a particular data owner's contribution, each share must be verified to be only write to a single row, i.e. the shares should resolve to unit vectors. The aggregators perform multi-party computation for each data owner in order to verify the FSS shares. We utilize recent constructions in FSS verification that do not rely on any public-key primitives [31].

The observation is the following: the dot product of any unit vector with itself is one, while the dot product of a non-unit vector is the square of the magnitude. The data owners submit blinded shares to each aggregator and then compute over the blinded shares so that the actual unit vector (i.e. data owner actual response) is not revealed. The aggregators perform an MPC to verify that the dot product of the blinded shares with itself evaluates to one, ensuring that the shares are properly formed. As long as there is at least one honest aggregation party, no aggregator learns which database row is written to. Invalid FSS shares can be quickly XORed out of the intermediate results once they are detected. Further details of the scheme can be found in the Section 7.

4. Warm up construction

Let us consider first a warm up scenario whereby each data owner privatized their truthful response utilizing the randomized response privacy mechanism and then privately uploads with an information theoretic guarantee.

First, we explain how we discretize real-numbered values to integers. We then formally describe the randomized response mechanism. Then, we describe the information-theoretic private upload mechanism. Finally, we integrate both techniques and show the limitations, in particular, constant error is not maintained as the non-truthful population increases. In addition, the information-theoretic private upload requires a keysize on the order of the database size making it prohibitively expensive for hundreds of thousands of data owners. We motivate the need for a more sophisticated sampling based privacy mechanism (the K Privacy mechanism) and more sophisticated techniques for cryptographic compression.

4.1. Goal

Our primary goal is to enable large scale private querying of the population utilizing sampling mechanisms while maintaining constant error. For example, suppose we query the number of vehicles on a busy stretch of the highway. We could query only those at a particular stretch of the highway. However, we would know the stretch of the highway location of any participating data owner. Thus, the privacy protection is quite limited regardless of any perturbation performed for this particular query.

The privacy protection would be improved if we query everyone in the city. The additional data owners provide plausible deniability by increasing the potential pool of candidates that sometimes respond "Yes" indicating they are at the particular stretch of highway. Now, if we know that someone participated in the query all we can immediately deduce is they are "somewhere" in the city.

More generally, let Yes_{pop} refer to the truthful "Yes" fraction of the population and No_{pop} refer to the truthful "No" fraction of the population. We seek to increase the No_{pop} by expanding the query to include more participating data owners. This results in lowering the percentage of the Yes_{pop}. Using the previous example, querying only those at the particular stretch of the highway would result in the Yes_{pop} being 100%. Expanding the query to the city would reduce this percentage to say 5% or even 1% or lower.

Query. The query is posted on the web. Data owners periodically check the web and download the query. The query is persistent and is set to expire days or weeks in the future.

4.2. Discretization

We illustrate the scheme using location coordinate data, although the discretization scheme can be employed for all real valued data. Suppose a data owner currently on London Bridge participates in the protocol. First, the location is discretized to a location identifier (ID) as seen in Fig. 1. For example, using a 16 bit identifier provides 65,536 possible locations, which covers a 64 x 64 mile square with 0.25 mile sections for a total of 4,096 square miles. For comparison Paris is 41 square miles, London is 607 square miles, New York City is 305 square miles [32]. In Fig. 1, London Bridge corresponds to location ID 8.

4.3. Sampling error

We now examine how the Randomized Response [22,23] mechanism grows in error as the No_{pop} increases. We first formally describe the Randomized Response mechanism and then describe how the sampling error increases with the population. We will later show in Section 5 how to preserve the utility.

(Randomized response) We use two independent and biased coins. Let π_1 and π_2 refer to the heads probabilities of the first and second biased coin toss respectively. The coin toss parameters are published publicly while the number of data owners is private and needs to be estimated.

$$Privatized\ Value_{Yes} = \begin{cases} 1 & \text{with probability} \\ & \pi_1 + (1-\pi_1) \times \pi_2 \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

That is, the Yes_{pop} subpopulation responds "Yes" with probability $\pi_1 + (1-\pi_1) \times \pi_2$. Otherwise they respond "No".

$$Privatized\ Value_{No} = \begin{cases} 1 & \text{with probability} \\ & (1-\pi_1) \times \pi_2 \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

That is, the No_{pop} subpopulation responds "Yes" with probability $(1-\pi_1) \times \pi_2$. Otherwise they respond "No".

(Expected value) We now formulate the expected value in order to carry out the estimation of the underlying population. The expected value of those that respond '1' (i.e., privatized "Yes") is the sum of the binomial distribution of each subpopulation.

$$E[1] = \pi_1 \times Yes_{pop} + (1-\pi_1) \times \pi_2 \times (Yes_{pop} + No_{pop}) \quad (3)$$

(Estimator) We solve for Yes_{pop} by the following. Let the aggregated privatized count E[1] defined in Eq. (3) be denoted as *Private Sum*.

$$Yes_{pop} = \frac{Private\ Sum - (1-\pi_1) \times \pi_2 \times (Yes_{pop} + No_{pop})}{\pi_1} \quad (4)$$

That is, we first subtract from Private Sum of the "privacy noise". We then divide by the first flip π_1 which is the sampling parameter which determines how frequently a data owner truthfully responds "Yes" from the Yes_{pop} subpopulation.

(Sampling error) Suppose published parameters of the coin tosses are configured independently with $\pi_1 = 0.85$, $\pi_2 = 0.3$ and 100 data owners. We estimate the underlying "Yes" truthful population using Eq. (4) by aggregating the privatized responses from all data owners, subtracting the expected value of $(1-0.85) \times 0.3 \times 100$ and dividing by 0.8[1].

However, a drawback to the randomized response mechanism is that the estimation error quickly increases with the population size due to the underlying truthful distribution distortion. For example, say we are interested in how many vehicles are at a popular

[1] For instance with a 60% truthful population, the answer to the first toss is $0.6 \times 0.85 = 0.51$ and the answer to the second toss is $(1-0.85) \times 0.3 = 0.045$

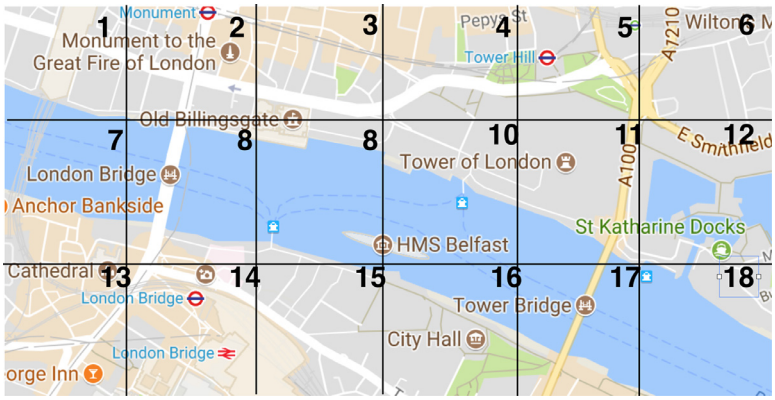

Fig. 1. *Locationdiscretization.* Each location (latitude,longitude) is discretized to a location identifier which corresponds to a 0.25 square mile block. London Bridge corresponds to location ID 8.

stretch of the highway. Say we configure $\pi_1 = 0.85$ and $\pi_2 = 0.3$. We query 10,000 vehicles asking for their current location and only 100 vehicles are at the particular area we are interested in (i.e., 1% of the population truthfully responds "Yes"). The standard deviation due to the privacy noise will be 21[2] which is slightly tolerable. However, a query over one million vehicles (now only 0.01% of the population truthfully responds "Yes") will incur a standard deviation of 212. The estimate of the ground truth (100) will incur a *large* absolute error when the aggregated privatized responses are two or even three standard deviations (i.e., 95% or 99% of the time) away from the expected value, as the mechanism subtracts *only* the expected value of the noise.

We desire better calibration over the privacy mechanism and a mechanism which maintains constant error as the No_{pop} population scales up. We introduce the K Privacy mechanism in the next section. Though first, we examine how to ensure that data owners are able to privately upload their responses.

4.4. Private upload

Now consider that we would like to write into a database without any of the database operators learning which row we wrote into. Knowledge of the row and thus data uploaded by the owner would allow the operator to track the owner over subsequent epochs even if the data is privatized.

We assume a distributed database setting such that, so long as one database operator remains honest and does not collude with other database operators, it is not possible to learn which database row was written into (as long as there are at least two data owners participating). The data owner should continuously re-select a new database row at random every epoch.

(Two party) Let us first consider two operators and two databases. The protocol proceeds as follows as shown in Fig. 2.

Assume the database is represented by n rows. Each data owner uploads a message of size m bits, in a randomly chosen row. Without loss of generality, for our example we describe now we assume m is one bit. Thus the database is a bitstring. Extending to a message size of more than one bit would only require a larger finite field (instead of finite field size 2 we could choose a prime number larger than the desired message size in bits).

Each data owner begins with a bitstring of length n (the size of the database). The data owner uniformly at random selects an index of the bitstring and sets its message value (assume it is to 1). Every other index is set to 0.

Next, the data owner creates a key by generating a random bitstring of length n.

The data owner then XORs the randomly generated bitstring key with the bitstring containing the message (that has only one index set) to produce the encrypted bitstring.

The data owner then transmits the encrypted bitstring to one database operator and the key bitstring to the other database operator. The data owner may randomly decide which database operator to send the encrypted and key bitstrings.

The same process repeats for each data owner. That is, a second data owner repeats the process of uniformly at random selecting an index of the bitstring to set to 1, generating a key bitstring, and encrypting the bitstring.

As the database operators receive each bitstring (either encrypted or key bitstring), each database operator cumulatively XORs the bitstrings.

Finally, at the end of an agreed epoch, the database operators share the cumulatively XORed bitstrings with each other. By doing so, they are able to reconstruct a database consisting of each data owners message at their specified indexes. The privacy guarantee is that the database operators are unable to determine which data owner wrote to which index, as long as there are at least two participating data owners and there is at least one database operator that does not collude with any other. (There is also an assumption that the data owners do not write to the same index, though collisions can be probabilistically avoided by sizing the database large enough to minimize the likelihood of collisions).

(Multi-party) Now, to generalize the 2 database operators to Z database operators, the protocol proceeds as follows.

The data owner has a bitstring of length n. The data owner uniformly at random selects the index of the bitstring to write to and sets the value to 1. Every other index is set to 0.

Next, the data owner generates $Z-1$ random bitstrings of length n. These bitstrings serve as the "virtual" single key.

The data owner cumulatively XORs the $Z-1$ bitstring keys with the bitstring containing the message (where only index is set) to produce the encrypted bitstring.

The data owner transmits each separate bitstring key to a different database operator and the encrypted bitstring to the other database operator. The data owner may randomly decide which database operator to send the encrypted and key bitstrings.

The multi-party protocol then proceeds the same as the previous two database operator case. Each database operator cumulatively XORs the received bitstrings and then shares the cumulative XOR results with each other to reconstruct the database. The privacy guarantee holds that each database operator is unable to determine which data owner wrote to which index, as long as there

[2] $(\sqrt{(1-0.85)\times 0.3\times 10,000})$

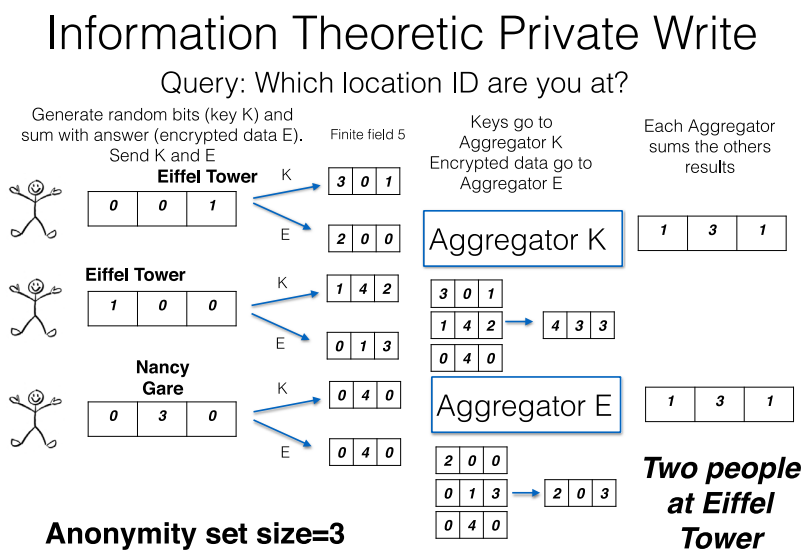

Fig. 2. Each data owner uniformly at random selects a slot to write their location ID. The aggregators are unable to determine which data owner wrote to a particular slot, as long as there is one honest aggregator who does not collude. The aggregate count of each location ID is computed as the final step.

are at least two participating data owners and there is at least one database operator that does not collude with any other.

(Key Size) The issue is that the key size is the length of the database n. Suppose the number of data owners is on the order of millions and the database row size is several hundred bits. The bitstring size will be of the order of several hundred MBs, which is prohibitively expensive for mobile devices and edge networks continually uploading every few minutes.

We could compress each of the key bitstrings by using a pseudorandom generator (PRG) for the Z-1 keys. However, we somehow must compress the bitstring containing the message (that has only one index set). Unfortunately, by definition of a PRG, it is computationally difficult to generate a PRG seed that expands to the desired bitstring. We must utilize a more sophisticated approached to enable cryptographic compression of the bitstring. We utilize a new primitive named Function Secret Sharing (FSS) [29] and show a performance optimization by selective choice of the parameters in Section 6.

5. K Privacy mechanism

We now describe our K Privacy mechanism that achieves constant error even where the population which does not truthfully respond "Yes" (No_{pop}) increases. We illustrate the scheme using location coordinate data, although the scheme can be employed for all real valued data.

Illustration. To illustrate and demonstrate the K Privacy mechanism, we employ the following example. Suppose we are interested in the distribution of vehicles across London. We query every vehicle in London asking for their location coordinates. Each data owner responds to a binary version of the binary query such as "Are you at the London Bridge?". The mechanism has two rounds and proceeds as follows.

Suppose a particular data owner is at London Bridge. First, the location is discretized to a location identifier (ID) as described in Section 4.2. In this case the location ID is 8 as shown in Fig. 1.

Next, the data owner tosses a multi-sided die. One side samples whether the data owner should respond truthfully for their location ID. The remaining sides selects a location ID for the data owner to respond.

Suppose in the first round the data owner is sampled and selected. The data owner should respond "Yes" (they are at London Bridge).

In the second round the sampled data owner should abstain from responding at all.

A privatized sum is computed by aggregating the "Yes" counts in each round.

Finally, estimation occurs by subtracting the privatized sum in round one from round two and dividing by the sampling parameter.

The following three privacy observations are made. First, a majority of the population provides privacy noise by randomly responding either "Yes" or "No" regardless of their truthful response. Second, plausible deniability is provided as each data owner probabilistically responds opposite of their truthful response. Finally, *every* data owner acts as a potential candidate for the truthful population. Our assumption is that every data owner is active in both rounds and only the aggregate counts are released.

5.1. Binary value

We now formally introduce the binary value K Privacy mechanism whereby a data owner responds either "No" or "Yes", either 0 or 1 respectively.

(Round one) In the first round each data owner tosses a three sided die with probabilities π_s, π_{Yes}, and π_{No}. Let π_s be the probability that a data owner truthfully responds. Otherwise, regardless of their truthful response let π_{Yes} be the probability that a data owner randomly responds "Yes" and π_{No} be the probability that a data owner randomly responds "No".

$$Round\ One_{Yes} = \begin{cases} \mathbf{1} & \mathbf{with\ probability}\ \pi_s \\ 1 & \text{with probability}\ \pi_{Yes} \\ 0 & \text{with probability}\ \pi_{No} \end{cases} \tag{5}$$

$$Round\ One_{No} = \begin{cases} 1 & \text{with probability}\ \pi_{Yes} \\ \mathbf{0} & \mathbf{with\ probability}\ \pi_s \\ 0 & \text{with probability}\ \pi_{No} \end{cases} \tag{6}$$

At this point, privacy noise has been added and thus the underlying truthful distribution is becoming distorted as the number of non-truthful data owners participate. The distortion makes it difficult to estimate the the underlying truthful distribution as

we have one equation and two variables (number of truthful and non-truthful data owners).

Thus, we execute a second round while fixing the two variables enabling us to solve for the truthful population estimate.

(Round two) In the second round only the data owner which was selected and sampled with probability π_s does not participate (effectively writes 0). The remaining data owners stay with the responses from round one.

$$Round\ Two = \begin{cases} \varnothing & \textbf{with probability } \pi_s \\ 1 & \text{with probability } \pi_{Yes} \\ 0 & \text{with probability } \pi_{No} \end{cases} \tag{7}$$

Now combining the second round with the first round we obtain accurate estimations as we see below.

(Expected values) We now formulate the expected values as follows. The subscript refers to the round number. That is, 1_1 refers to output 1 and round 1. The first round of expected values are:

$$\begin{aligned} E[1_1] &= \pi_{Yes} \times TOTAL_{pop} + \pi_s \times Yes_{pop} \\ E[0_1] &= \pi_{No} \times TOTAL_{pop} + \pi_s \times No_{pop} \end{aligned} \tag{8}$$

That is, both the Yes_{pop} and No_{pop} contribute both "Yes" and "No" responses while a small subpopulation responds truthfully.

The second round of expected values are:

$$\begin{aligned} E[1_2] &= \pi_{Yes} \times TOTAL_{pop} \\ E[0_2] &= \pi_{No} \times TOTAL_{pop} \end{aligned} \tag{9}$$

That is, the small subpopulation from round 1 samples out and does not participate (effectively writes 0). The remaining data owners randomly respond "Yes" or "No" while remaining at their round one responses.

(Estimator) We solve for the Yes_{pop} population by subtracting round one by round two as follows. Let $Private\ Sum_{\text{“}Yes\text{”},1}$ refer to the aggregated privatized counts for output space "Yes" and round 1.

$$Yes_{pop} = \frac{Private\ Sum_{\text{“}Yes\text{”},1} - Private\ Sum_{\text{“}Yes\text{”},2}}{\pi_s} \tag{10}$$

That is, we subtract the privatized sum of output space "Yes" round 1 from the output space "Yes" of round 2. The result is the sampled "Yes" aggregate. We then obtain the estimation by dividing by the sampling parameter.

The sampling error affects only the Yes_{pop} as seen in Eq. (8). Thus we are able to scale the No_{pop} yet retain constant error. Plausible deniability is provided as each data owner may respond to either output space based on the coin toss parameters.

5.2. Multiple (simultaneous) values

We now examine how to privatize the multiple choice scenario whereby there are multiple values and the data owner should select a single value. We extend the binary value mechanism defined in the previous section. Multiple values are applicable to most real-world scenarios (as opposed to the binary value mechanism). The location coordinate grid scenario, explained in Section 4.2 and illustrated in Fig. 1, explains a scenario where there are multiple locations (i.e., location IDs) and the data owner is currently at a single location ID. Recall that the data owner's truthful response is discretized to an integer value greater than 0.

However, we desire more than simply randomizing multiple choices. The K Privacy mechanism has each data owner respond with *multiple, simultaneous, and contradictory* responses while maintaining constant error. For example, if there are 9 locations, each data owner probabilistically responds they may be in 9 locations simultaneously.

Suppose a particular data owner is at the London Bridge. The first round proceeds as follows. For the location they are currently at (e.g., London Bridge) the data owner flips a biased coin and if heads responds truthfully. When queried about other locations the data owner randomly also responds they are at the location.

Say the particular data owner from Fig. 1 was sampled and selected. Their response should be with location ID 8. In addition, say they were selected for location IDs 1,2,4,5. Thus, the round one response would be 1,2,4,5,8.

In the second round, the data owner should *not* respond they are at the London Bridge. The remaining responses stay. Thus, the round two response would be 1,2,4,5.

A privatized sum is computed by aggregating the location ID counts in each round.

The estimated count for each location ID value is then calculated by subtracting the privatized sum of the second round from the first round and then dividing by the sampling parameter.

There are several privacy observations. Similar to the binary value scenario in Section 5.1, both privacy noise and plausible deniability is provided. However, now a data owner responds with multiple *contradictory* responses claiming to be in multiple locations at once. At the same time in the second round, *all* selected and sampled data owners across every location will silently not participate. Every data owner now blends with every other data owner.

We now formally describe the multiple (simultaneous) values K Privacy mechanism.

(Round one) Let V represent all outputs for which the data owner does not truthfully respond "Yes". Let V' be the special output for which the data owner truthfully responds "Yes".

In the first round the data owner should truthfully respond according to the sampling parameter only for their truthful output value. For all other values the data owner randomly responds regardless of their truthful value.

$$Round\ One = \begin{cases} V' & \text{with probability } \pi_s \\ V, V' & \text{with probability } \pi_V \\ 0 & \text{with probability } 1 - \pi_V - \pi_s \end{cases} \tag{11}$$

That is, a data owner responds with multiple contradictory responses.

(Round two) In the second round all data owners stay with their round one response and respond randomly regardless of their underlying truthful response. All the data owners that were sampled and selected to respond truthfully do not participate in the second round.

$$Round\ Two = \begin{cases} \varnothing & \textbf{with probability } \pi_s \\ V, V' & \text{with probability } \pi_V \\ 0 & \text{with probability } 1 - \pi_V - \pi_s \end{cases} \tag{12}$$

That is, all truthful responses equally fall out from the equation.

(Expected values) The first round of expected values are as follows.

$$E[V_1] = \pi_V \times TOTAL + \pi_s \times Yes_{pop} \tag{13}$$

That is, for each value both populations randomly contribute, with a small percentage contributing by responding truthfully.

The second round of expected values are all the same regardless of the underlying truthful response:

$$E[V_2] = \pi_V \times TOTAL \tag{14}$$

That is, everyone randomly contributes. The sampled and selected percentage that truthfully responded in round one do not participate (effectively write 0) and respond now in round two.

(Estimator) To solve for the *YES* population we subtract the second round from the first round and iterate for each output value as follows:

$$YES = \frac{Private\ Sum_{\text{“}Yes\text{”},1} - Private\ Sum_{\text{“}Yes\text{”},2}}{\pi_s} \tag{15}$$

The sampled and selected population, by not participating in round two (effectively write 0), allows us to baseline the privacy noise and perform estimation for the sampled truthful population.

5.3. Differential privacy guarantee

K Privacy satisfies differential privacy as we show in this section. We first examine the binary value mechanism and then the multiple value mechanism. The definition of differential privacy can be found in the Appendix Appendix A.

(Binary Value) The differential privacy leakage is measured as the maximum ratio of the binary output space given the underlying truthful answer is "Yes" and "No" respectively.

In round one, the output space "Yes" is slightly more likely as the truthful response is sampled in addition to being responded randomly. In round two, there is no privacy leakage as both output space "Yes" and "No" are both equally likely and indistinguishable given the truthful answer is either "Yes" or "No" respectively.

Thus, we are interested in the privacy leakage of output 1 round 1 (1_1) as follows:

$$\epsilon_{DP} = \max\left(\ln\left(\frac{\Pr[1_1|\text{“}Yes\text{”}]}{\Pr[1_1|\text{“}No\text{”}]}\right), \ln\left(\frac{\Pr[1_1|\text{“}No\text{”}]}{\Pr[1_1|\text{“}Yes\text{”}]}\right)\right) \tag{16}$$

$$\frac{\Pr[1_1|\text{“}Yes\text{”}]}{\Pr[1_1|\text{“}No\text{”}]} = \frac{\pi_{V'} + \pi_s}{\pi_{V'}} \tag{17}$$

$$\frac{\Pr[1_1|\text{“}No\text{”}]}{\Pr[1_1|\text{“}Yes\text{”}]} = \frac{\pi_{V'}}{\pi_{V'} + \pi_s} \tag{18}$$

$$\epsilon_{DP} = \max\left(\ln\left(\frac{\pi_{V'} + \pi_s}{\pi_{V'}}\right), \ln\left(\frac{\pi_{V'}}{\pi_{V'} + \pi_s}\right)\right) \tag{19}$$

(Multiple (simultaneous) values) The differential privacy leakage is measured as the maximum ratio of the multiple output space given the underlying truthful answer is any combination of two values of the output space of size V.

In round one, for any two outputs whereby a data owner would not truthfully respond with those values, the outputs are indistinguishable and there is no privacy leakage as the response is chosen randomly. The only privacy leakage occurs when the data owner truthfully responds for their output space.

For *output V', round 1* ($\mathbf{V_1'}$), the privacy leakage is as follows:

$$\epsilon_{DP} = \max\left(\ln\left(\frac{\Pr[V_1'|V']}{\Pr[V_1'|\neg V']}\right), \ln\left(\frac{\Pr[V_1'|\neg V']}{\Pr[V_1'|V']}\right)\right) \tag{20}$$

$$\frac{\Pr[V_1'|V']}{\Pr[V_1'|\neg V']} = \frac{\pi_{V'} + \pi_s}{\pi_{V'}} \tag{21}$$

$$\frac{\Pr[V_1'|\neg V']}{\Pr[V_1'|V']} = \frac{\pi_{V'}}{\pi_{V'} + \pi_s} \tag{22}$$

$$\epsilon_{DP} = \max\left(\ln\left(\frac{\pi_{V'} + \pi_s}{\pi_{V'}}\right), \ln\left(\frac{\pi_{V'}}{\pi_{V'} + \pi_s}\right)\right) \tag{23}$$

In round two, again there is no privacy leakage for those values the data owner would not truthfully respond as the response is chosen randomly. The only privacy leakage occurs for those sampled and selected data owners that do not participate (effectively write 0).

For *output V', round 2* ($\mathbf{V_2'}$), the privacy leakage is as follows:

$$\epsilon_{DP} = \max\left(\ln\left(\frac{\Pr[V_2'|V']}{\Pr[V_2'|\neg V']}\right), \ln\left(\frac{\Pr[V_2'|\neg V']}{\Pr[V_2'|V']}\right)\right) \tag{24}$$

$$\frac{\Pr[V_2'|V']}{\Pr[V_2'|\neg V']} = \frac{\pi_{V'} - \pi_s}{\pi_{V'}} \tag{25}$$

$$\frac{\Pr[V_2'|\neg V']}{\Pr[V_2'|V']} = \frac{\pi_{V'}}{\pi_{V'} - \pi_s} \tag{26}$$

$$\epsilon_{DP} = \max\left(\ln\left(\frac{\pi_{V'} - \pi_s}{\pi_{V'}}\right), \ln\left(\frac{\pi_{V'}}{\pi_{V'} - \pi_s}\right)\right) \tag{27}$$

We then take the maximum leakage of round 1 (Eq. (23)) and round 2 (Eq. (27)).

6. Private data upload

We now describe the Function Secret Sharing (FSS) [29] primitive and how it enables a nearly square root reduction of the key size. We then introduce our parameter optimization to achieve more than an *order of magnitude improvement* over the default implementation.

Algorithm 1 $\text{Gen}^{p_i}(1^\lambda, x, y)$: Generate Shares

1: Let $G : \{0,1\}^\lambda \longrightarrow \{0,1\}^{m\mu}$ be a PRG
2: Let $\mu \leftarrow \lceil 2^{n/2} \times 2^{p-1/2} \rceil$. Let $\nu \leftarrow \lceil 2^n/\mu \rceil$
3: Use the higher and lower bits of the input x as a pair $x = (\gamma\prime, \delta\prime)$, $\gamma\prime \in \lceil \nu \rceil$ $\delta\prime \in \lceil \mu \rceil$
4: Choose ν arrays $A_1, ..., A_\nu$ s.t. $A_\gamma \in_R O_p$ and $A_{\gamma\prime} \in_R E_p$ for all $\gamma\prime \neq \gamma$
5: Choose 2^{p-1} random strings $cw_1, ..., cw_{2^{p-1}} \in 0, 1^{m\mu}$ s.t. $\bigoplus_{j=1}^{2^{p-1}} (cw_j \oplus G(s_{\gamma,j})) = e_\delta \cdot b$
6: Set $\sigma_{i,\gamma\prime} \leftarrow (s_{\gamma\prime,1} \cdot A_{\gamma\prime}[i,1]) \parallel ... \parallel (s_{\gamma\prime,2^{p-1}} \cdot A_{\gamma\prime}[i, 2^{p-1}])$ for all $1 \leq i \leq p$, $1 \leq \gamma\prime \leq \nu$.
7: Set $\sigma_i = \sigma_{i,1} \parallel ... \parallel ...\sigma_{i,\nu}$ for $1 \leq i \leq$ p
8: Let $k_i = (\sigma_i \parallel cw_1 \parallel ... \parallel cw_{2^{p-1}})$ for $1 \leq i \leq$ p
9: Return $(k_i, ..., k_p)$

6.1. Function secret sharing background

Recall our earlier construction in Section 4.4 whereby data owners privately upload data into a distributed database without any of the Z database operators learning into which row a particular data owner wrote (assuming there is at least one honest database operator that does not collude and there are at least two honest data owners). Each data owner specifies their upload data by uniformly at random selecting a row from the database to write their message. This selection of a single row a and writing a message m can be viewed as a point function F_a whereby $F_a(x) = m$ iff $x = a$ and 0 otherwise.

The key idea of FSS to obtain the key size reduction is the use of a pseudorandom generator (PRG) to compress the key size. However, as we previously saw simply using a PRG alone is not enough as it's not computationally feasible to find Z random seeds that cumulatively XORed together produce a desired output. We can find $Z-1$ random seeds, cumulatively XOR them together, then XOR with the desired output to find the correction word bitstring needed to XOR with the seeds to produce a desired output. It is this observation that multi-party FSS exploits to achieve the nearly square root key size reduction.

FSS addresses the issue of the correction word being the length of the database (thus the key size the length of the database) by the use of a special matrix. Indexing into the matrix is done by using the lower and higher bits of the input to lookup into the matrix. Each matrix row contains a set of PRG seeds. The expansion of a particular subset of the PRG seeds are then combined by XOR with the correction word. The resultant bitstring is then the length of the correction word and contains both the desired output as well as other random noise. Using the higher order bits of a as

a lookup into the resultant bitstring will locate the desired output within the resultant bitstring. All other inputs will produce random noise.

Thus, the cryptographic compression is achieved by the using of the PRG combined with correction words with length roughly the square root of the size of the database. Further details can be found in the paper by Boyle et al. [29].

However, it turns out that by adjusting the length of the correction words and number of PRG seeds greatly impacts the performance of the FSS protocols. We now explain our FSS parameter optimization.

6.2. Function secret sharing optimization

FSS relies on symmetric cryptography. Thus, we utilize AES in counter mode for the pseudorandom generator.

There are two symmetric cryptography overheads that FSS incurs. The first is that the default FSS evaluation algorithm repeatedly evaluates the same seeds multiple times. The only difference between each evaluation is that different positions of the seed expansion evaluation is used based on the input's lower bits. The second overhead is balancing the number of seeds with seed expansions.

Our algorithm optimization is described in Algorithm 6.2 and

Algorithm 2 EvaluateShare$^{p_i}(k_i)$: Evaluate Share

1: Let $\mu \leftarrow \lceil 2^{n/2} \times 2^{p-1/2} \rceil$. Let $\nu \leftarrow \lceil 2^n/\mu \rceil$
2: Use the higher and lower bits of the input x as a pair $x = (\gamma\prime, \delta\prime)$, $\gamma\prime \in \lceil \nu \rceil$ $\delta\prime \in \lceil \mu \rceil$
3: **for** $j = 1, ..., \nu$ **do**
4: $\quad y_j \leftarrow \text{Eval}(j, k_i)$
5: $\quad$ Let $result_j \leftarrow (y_j[1] \parallel ... \parallel y_j[\mu])$
6: **end for**
7: Return $(result_1, ..., result_\nu)$

Algorithm 6.2. The share generation algorithm 6.2, is the same as

Algorithm 3 Eval$^{p_i}(\nu\prime, k_i)$

1: Let $G : \{0,1\}^\lambda \longrightarrow \{0,1\}^{m\mu}$ be a PRG
2: Parse k_i as $k_i = (\sigma_i, cw_i, ..., cw_{2^{p-1}})$
3: Parse σ_i as $\sigma_i = s_{1,1} \parallel ... \parallel s_{1,2^{p-1}} \parallel ... \parallel s_{\nu,2^{p-1}}$
4: Let $y_i \leftarrow \bigoplus_{1 \le j \le 2^{p-1}} (cw_j \oplus G(s_{\gamma\prime,j}))$ where $s_{\gamma\prime,j} \neq 0$
5: Return y_i

described in [29]. The difference is in the share evaluation. The default implementation performs 2^n PRG seed initializations. However, the full PRG evaluation is the same for each value of γ'. Thus, we need to perform ν PRG seed initializations instead of repeating the same PRG evaluation. We do *one* evaluate per δ, which means that we can reuse the same evaluated output and just take different partitions for varying γ. The default FSS version evaluates δ times the same seeds in order to extract the differing γ sections.

Our second optimization is balancing the number of total seeds generated and evaluated with the prg expansion length of each seed. Increasing the length of the seed reduces the number of total seeds required. It's faster to expand a single side as opposed to multiple expansions of differing seeds. However, the cost is an increase in the length of the key size. Thus, we chose our parameters by performing microbenchmarks to understand the tradeoffs in.

7. Pollution protection

In order to guard the private writes against a single data owner writing to multiple rows, we utilize a novel and efficient FSS share verification technique (which does not require any public-key primitives) that is performed by the aggregation parties [31]. The FSS share verification ensures that each data owner writes a unit vector (i.e., a single row). Details of the MPC technique can be found in the Appendix B. Our evaluation results can be found in Section 8.3.

Next, to prevent a single answer, such as a large number, from distorting the aggregate sum, we utilize a bit vector response which limits the data owner to only replying "No" ('0') or "Yes" ('1').

Finally, data owners are authenticated to prevent Sybils and multiple responses within a single epoch. The authentication does not allow the aggregators to learn which row a data owner is writing to. Each data owner performs a cryptographic private write that is protected as long as there is at least one honest aggregator who does not collude (as we previously shown in Section 6).

Defining the error threshold for the number of malicious data owners who falsify their responses (i.e., intentionally answering "No" instead of "Yes") is not considered in this work. However, efficient techniques exist which ensure commitment to the randomized response protocol [33].

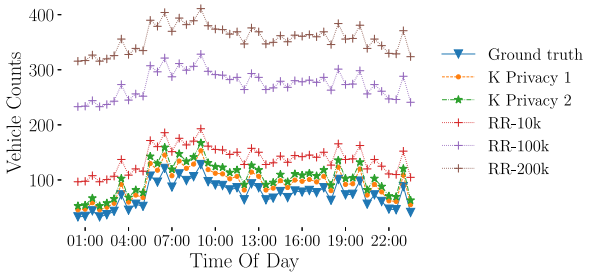

Fig. 3. *(Vehicle counts)* K Privacy Each vehicle reports it's current location.

8. Evaluation

We evaluate the accuracy of the K Privacy mechanism. Next, we describe the performance gains of the FSS parameter optimization. Finally, we evaluate the efficiency of the pollution protection technique. The K Privacy mechanism constrains the coin toss probabilities to below 50%, to ensure the adversary is not given an advantage to statistically guess the data owner's truthful response better than 50% of the time.

8.1. Accuracy

(PeMS Data) We evaluate the K Privacy mechanism over a real dataset rather than with arbitrary distributions. We utilize the California Transportation Dataset from magnetic pavement sensors[34] collected in LA\Ventura California freeways [35]. There are a total of 3865 stations and 999,359 vehicles total. We assign virtual identities to each vehicle. Each vehicle announces the station it is currently at. We select a single popular highway station. Every vehicle at the station reports "Yes" while every other vehicle in the population truthfully reports "No". We evaluate over a 24 h time period. K Privacy 1 has a sampling parameter of 45% and K Privacy 2 has a sampling parameter of 25%. The randomized response mechanism has $\pi_1 = 0.8$ and $\pi_2 = 0.2$.

Fig. 3 compares the K Privacy mechanism with the Randomized Response mechanism. K Privacy is able to maintain constant error even at 1 million vehicles, while the Randomized Response quickly incurs error. Upper bounds are shown with a 95% confidence interval.

We next examine the vehicle speed distribution across the freeways at evening rush hour. Fig. 6 is with the population at the

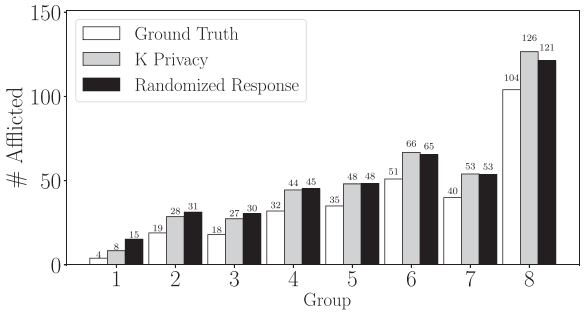

Fig. 4. *(Heart chest pain)* Number of individuals out of 303 with specific types of heart related chest pain.

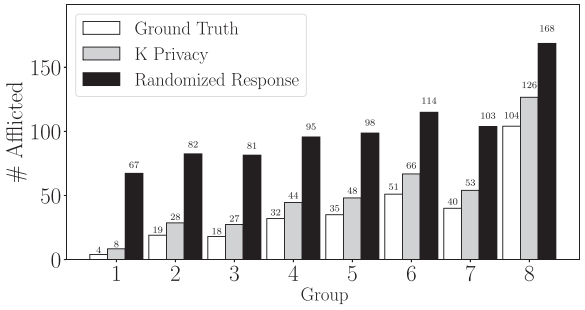

Fig. 5. *(Heart chest pain)* Number of individuals out of 10,000 with specific types of heart related chest pain.

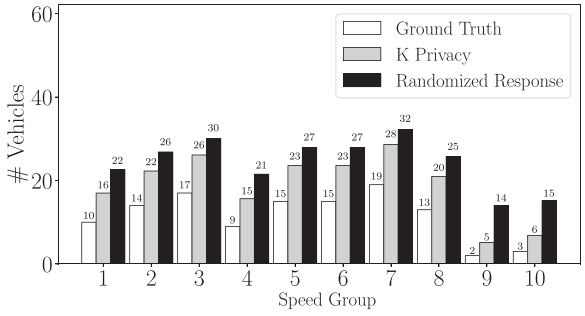

Fig. 6. *(Vehicle speed distribution)* Lane 1 speed distribution.

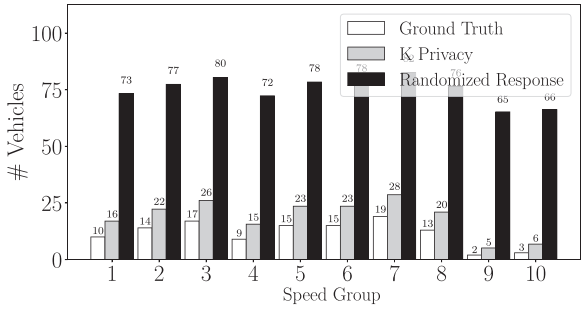

Fig. 7. *(Vehicle speed distribution)* Lane 1 speed distribution over 10,000 vehicles (only 354 are currently amongst the queried 3 lanes).

specific stretch of the freeway. Fig. 7 expands the query population to 10,000 vehicles (9646) are not at the particular freeway stretch being monitored. The figures show the speed distribution whereby there are 10 groups for the following speeds "$1-10''$ is group 1, "$11-20''$ is group 2, etc. Upper bounds are shown with a 95% confidence interval.

(Heart data) We next evaluate over medical data. We utilize the UCI open data repository [36] for heart related data. Figs. 4 and 5

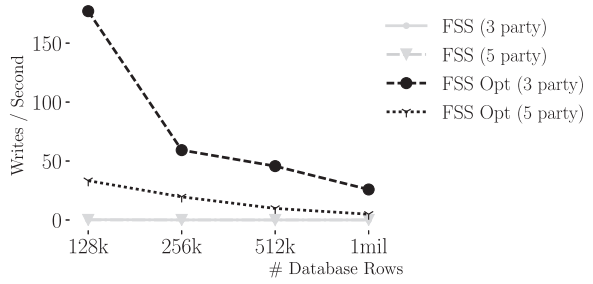

Fig. 8. *(FSS Microbenchmark)* Number of share evaluations (client share uploads) per second. Bigger is better.

show the number of afflicted data owners with a particular type of chest pain. The four types of chest pain are typical angina, atypical angina, non-anginal pain, and asymptomatic. Each group corresponds to a particular chest pain and gender for a total of eight groups. Fig. 5 scales the population to 10,000 whereby 303 are the original dataset and the remaining 9697 data owners provide chaff. The K Privacy mechanism maintains constant error and the randomized response quickly incurs error. Upper bounds are shown with a 95% confidence interval.

8.2. Privacy

Fig. 11evaluates the privacy leakage comparing K Privacy and the Randomized Response mechanism. K Privacy uses the equation defined in 5.3 to measure the privacy leakage. The Randomized Response mechanism privacy leakage is defined in the Appendix A.1.

The coin toss parameters used in Fig. 11 has Randomized Response $flip1 = 0.8$ and $flip2 = 0.2$. K Privacy has a sampling parameter of 0.45. We could increase the value of the randomized response *flip*2 though the absolute error would grow even larger than show in Fig. 3.

8.3. Scalability

FSS optimization We evaluate our implementation of non-attributable writes on Amazon EC2 with c4.2xlarge instances to understand the impact of our optimization of the Function Secret Sharing primitive.

We first perform a microbenchmark to evaluate the improvement of our optimization of evaluating shares to be performed by the aggregators. Fig. 8 shows our microbenchmark for the 3 party and 5 party case. Our microbenchmark shows several orders of magnitude improvement over the default implementation.

We examine the key size of each share as we increase the size of the message that is privately uploaded. Fig. 12 shows a modest increase in the key size as the message size increases. We then evaluate the generation of shares. We evaluate the trade-off of key size by expanding a single seed versus expanding multiple seeds with a smaller length. Fig. 9 shows the effect of the FSS optimization on the time to generate shares. We achieve close to half a reduction in share generation time.

We next evaluate our evaluation optimization on Amazon EC2. Fig. 10 shows the effect of applying the FSS optimization for the evaluation of the shares as described in Section 6.2. We are able to achieve an order of magnitude improvement over the default implementation.

Share verification. We now discuss the evaluation of our implementation of the FSS share verification [31]. The three algorithms for creating the blinding structure are "square", "product", and "inverse" (see Appendix B for more details) (Fig. 12).

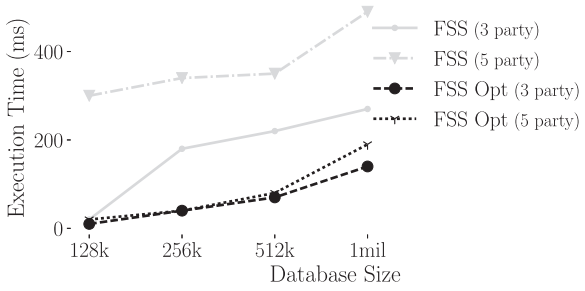

Fig. 9. *FSS Share Generation.* FSS versus FSS optimized client share upload. Smaller is better.

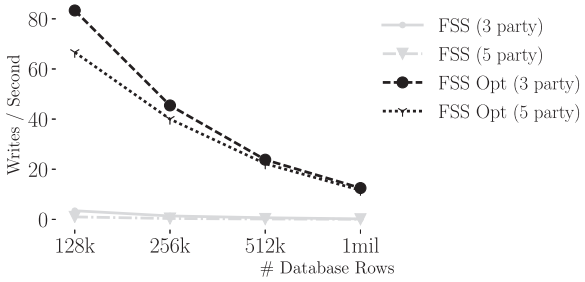

Fig. 10. *FSS Scaling Optimization.* FSS versus FSS optimized (client share uploads) per second. Bigger is better.

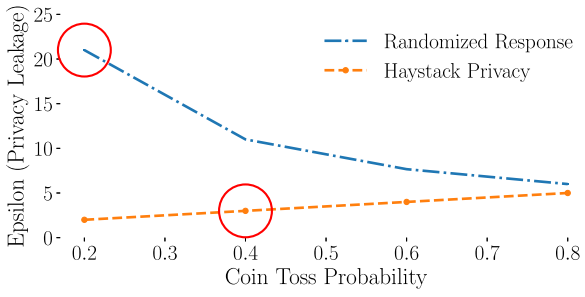

Fig. 11. *(Privacy Leakage)* K Privacy compared with randomized response privacy leakage as the coin toss probability increases. Higher epsilon means more information is leaked.

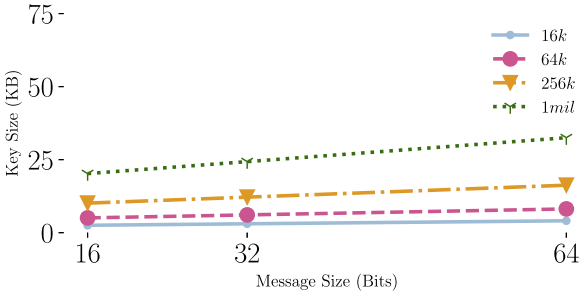

Fig. 12. FSS keysize.

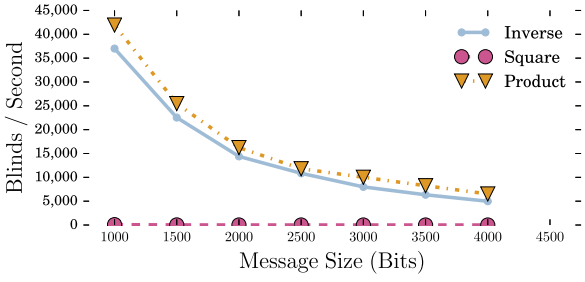

Fig. 13. MPC verification benchmark.

Fig. 13 shows the scalability of the blinding operations. "Product" is slightly faster than "square" as "product" must only do$(p-1)$ multiplications, while "square" does $(p-1)$ exponent operations. "Inverse" is the slowest as it performs $(p-1)$ multiplications and then a finite field inverse, where p is the number of parties. The MPC verification performed by the aggregation servers is on the order of a couple hundred milliseconds and is extremely efficient.

9. Conclusion

In this paper, we present the K Privacy mechanism and demonstrate how to (i) improve the privacy strength while preserving utility, (ii) achieve scalable non-attributable writes, and (iii) provide protection against pollution attacks whereby a single data owner may attempt to corrupt the entire database. To demonstrate its real-world applicability and practicality, the K Privacy mechanism was implemented on Amazon's AWS cloud and shown to scalably achieve these properties. We believe this represents an important and timely advance towards open and shared Internet of Vehicles data.

Appendix A. Differential privacy

Differential privacy has become the *gold standard* privacy mechanism which ensures that the output of a sanitization mechanism does not violate the privacy of any individual inputs.

Definition 1 [8,10]**.** (ϵ-Differential Privacy). A privacy mechanism $San()$ provides ϵ-differential privacy if, for all datasets D_1 and D_2 differing on at most one record (i.e., the Hamming distance $H()$ is $H(D_1, D_2) \leq 1$), and for all outputs $O \subseteq Range(San())$:

$$\sup_{D_1,D_2} \frac{\Pr[San(D_1) \in O]}{\Pr[San(D_2) \in O]} \leq exp(\epsilon) \tag{A.1}$$

That is, the probability that a privacy mechanism *San* produces a given output is almost independent of the presence or absence of any individual record in the dataset. The closer the distributions are (i.e., smaller ϵ), the stronger the privacy guarantees become and vice versa. That is, a larger ϵ means that the two dataset distribution are far apart and leaks more information. A single record will induce distinguishable output fluctuations. We desire smaller ϵ values to induce ϵ *indistinguishability*.

A1. Randomized response privacy guarantee

A1.1. Privacy guarantee of randomized response

The randomized response mechanism achieves ϵ-differential privacy, where:

$$\epsilon = \max\left(\ln\left(\frac{\Pr[\text{Resp='Yes' }|\text{'Yes'}]}{\Pr[\text{Resp='Yes' }|\text{'No'}]}\right), \ln\left(\frac{\Pr[\text{Resp='Yes' }|\text{'No'}]}{\Pr[\text{Resp='Yes' }|\text{'Yes'}]}\right)\right)$$

More specifically, the randomized response mechanism [23] achieves ϵ-differential privacy, where:

$$\epsilon = \ln\left(\frac{\pi_1 + (1-\pi_1) \times \pi_2}{(1-\pi_1) \times \pi_2}\right) \tag{A.2}$$

That is, if a data owner has the sensitive attribute A, then the randomized answer will be "Yes" with the probability of '$\pi_1 + (1-\pi_1) \times \pi_2$'. Else, if a data owner does not have the sensitive at-

tribute, then the randomized answer will become "Yes" with the probability of '$(1-\pi_1)\times\pi_2$'.

Appendix B. Share verification

We now describe the MPC protocol [31] run amongst the aggregator parties to verify all data owner shares. The protocol does not violate data owner privacy and is extremely efficient as it does not utilize any publick-key primitives and relies solely on finite field operations.

We first describe the MPC protocol in detail and then provide an example.

MPCprotocol Let p represent the number of parties participating in the protocol.

Let n represent the unit vector length (e.g., length of the bit string or number of database slots).

Let m represent the number of bits of the message M. Let $M \in \mathbb{F}_Z$ where Z is a relatively large prime number.

Given $\mathbb{F}_Z$ a finite field of characteristic Z where Z is a relatively large prime, let $\mathbf{R}$ be a blinding (randomization) matrix where the the values in the first row are chosen uniformly at random over $0, ..., Z-1$.

This is a particular randomization matrix such that elements of each row is raised to the power of the first row, where the power is equivalent to the row number. There will be a total of p rows, one for each party. That is,

$$\mathbf{R} = \begin{bmatrix} r_1 & r_2 & ... & r_n \\ r_1^2 & r_2^2 & ... & r_n^2 \\ ... & ... & ... & ... \\ r_1^p & r_2^p & ... & r_n^p \end{bmatrix} \tag{B.1}$$

We wish to secretly share a unit vector and verify that the shares correctly sum to the unit vector.

For example,

$$\hat{\mathbf{u}} = \begin{bmatrix} 0 \\ M \\ ... \\ 0 \end{bmatrix} \tag{B.2}$$

The value can be m bits taking on a value from the finite field of character of characteristic p where p is a relatively large prime.

To share $\hat{\mathbf{u}}$, the user can randomly generate a total p vectors $\mathbf{V_i}$

$$\mathbf{V_i} = \begin{bmatrix} v_{i,1} \\ v_{i,2} \\ ... \\ v_{i,n} \end{bmatrix} \tag{B.3}$$

such that

$$\sum_{i=1}^{p} \mathbf{V_i} = \hat{\mathbf{u}} \tag{B.4}$$

We then blind these values such that

$$\sum_{i=1}^{p} \mathbf{R}\cdot\mathbf{V_i} = \mathbf{R}\cdot\hat{\mathbf{u}} \tag{B.5}$$

Let's describe an example where $n=2$ and $p=3$.

We know that sum of the vectors should equal the unit vector.

$$\begin{bmatrix} v_{1,1} \\ v_{1,2} \end{bmatrix} + \begin{bmatrix} v_{2,1} \\ v_{2,2} \end{bmatrix} + \begin{bmatrix} v_{3,1} \\ v_{3,2} \end{bmatrix} = \hat{\mathbf{u}} \tag{B.6}$$

We now apply the randomization (blinding) matrix.

$$\begin{bmatrix} r_1 & r_2 \\ r_1^2 & r_2^2 \\ r_1^3 & r_2^3 \end{bmatrix} \cdot \begin{bmatrix} v_{1,1} \\ v_{1,2} \end{bmatrix} + \begin{bmatrix} r_1 & r_2 \\ r_1^2 & r_2^2 \\ r_1^3 & r_2^3 \end{bmatrix} \cdot \begin{bmatrix} v_{2,1} \\ v_{2,2} \end{bmatrix} + \begin{bmatrix} r_1 & r_2 \\ r_1^2 & r_2^2 \\ r_1^3 & r_2^3 \end{bmatrix} \cdot \begin{bmatrix} v_{3,1} \\ v_{3,2} \end{bmatrix} = \mathbf{R}\cdot\hat{\mathbf{u}} \tag{B.7}$$

$$\begin{bmatrix} r_1\cdot v_{1,1} + r_2\cdot v_{1,2} \\ r_1^2\cdot v_{1,1} + r_2^2\cdot v_{1,2} \\ r_1^3\cdot v_{1,1} + r_2^3\cdot v_{1,2} \end{bmatrix} + \begin{bmatrix} r_1\cdot v_{2,1} + r_2\cdot v_{2,2} \\ r_1^2\cdot v_{2,1} + r_2^2\cdot v_{2,2} \\ r_1^3\cdot v_{2,1} + r_2^3\cdot v_{2,2} \end{bmatrix} + \begin{bmatrix} r_1\cdot v_{3,1} + r_2\cdot v_{3,2} \\ r_1^2\cdot v_{3,1} + r_2^2\cdot v_{3,2} \\ r_1^3\cdot v_{3,1} + r_2^3\cdot v_{3,2} \end{bmatrix} = \mathbf{R}\cdot\hat{\mathbf{u}} \tag{B.8}$$

$$\begin{bmatrix} r_1\ (v_{1,1}+v_{2,1}+v_{3,1}) + r_2\ (v_{1,2}+v_{2,2}+v_{3,2}) \\ r_1^2\ (v_{1,1}+v_{2,1}+v_{3,1}) + r_2^2\ (v_{1,2}+v_{2,2}+v_{3,2}) \\ r_1^3\ (v_{1,1}+v_{2,1}+v_{3,1}) + r_2^3\ (v_{1,2}+v_{2,2}+v_{3,2}) \end{bmatrix} = \mathbf{R}\cdot\hat{\mathbf{u}} \tag{B.9}$$

Since the summation of the elements of a unit vector should sum to zero, we can denote the value as follows

$$\begin{bmatrix} a+b \\ a^2+b^2 \\ a^3+b^3 \end{bmatrix} = \mathbf{R}\cdot\hat{\mathbf{u}} \tag{B.10}$$

From Eq. (B.6) that the sum of the vectors is the unit vector. Thus, we then know that if the shares are properly formed that a and b should represent either all zeros or the blinded message. Thus, $(a+b)^2-(a^2+b^2)=0$ and $(a+b)^3-(a^3+b^3)=0$.

If a and b are both zero then the terms fall out.

In the case of only a or b being the blinded message the terms fall out.

If both a and b are non-zero then the difference will be a non-zero value. These shares are invalid and should be discarded.

B1. Alternate algorithms

There are two alternate algorithms for the "square" algorithm described above, which were also presented in [31]. The same process is used, but the structure of the blinding matrix is different, as well as the final check of $\mathbf{R}\cdot\hat{\mathbf{u}}$. The first algorithm is the "product" algorithm where

$$\mathbf{R} = \begin{bmatrix} r_{1,1} & r_{2,1} & ... & r_{n,1} \\ r_{1,2} & r_{2,2} & ... & r_{n,2} \\ ... & ... & ... & ... \\ r_{1,p} & r_{2,p} & ... & r_{n,p} \end{bmatrix} \tag{B.11}$$

such that

$$\forall i \prod_{j=1}^{p-1} r_{i,j} = r_{i,p} \tag{B.12}$$

Then, we can apply the blinding matrix to our vectors $\mathbf{V_i}$, to achieve the final result:

$$\begin{bmatrix} a_1+b_1 \\ a_2+b_2 \\ a_3+b_3 \end{bmatrix} = \mathbf{R}\cdot\hat{\mathbf{u}} \tag{B.13}$$

where

$$\prod_{i=1}^{p-1}(a_i + b_i) = a_p + b_p \tag{B.14}$$

An Alternative scheme is the "inverse" algorithm, which has a blinding matrix of

$$\mathbf{R} = \begin{bmatrix} r_{1,1} & r_{2,1} & \dots & r_{n,1} \\ r_{1,2} & r_{2,2} & \dots & r_{n,2} \\ \dots & \dots & \dots & \dots \\ r_{1,p} & r_{2,p} & \dots & r_{n,p} \end{bmatrix} \tag{B.15}$$

such that

$$\forall i \prod_{j=1}^{p} r_{i,j} = 1 \tag{B.16}$$

Then,

$$\begin{bmatrix} a_1 + b_1 \\ a_2 + b_2 \\ a_3 + b_3 \end{bmatrix} = \mathbf{R} \cdot \hat{\mathbf{u}} \tag{B.17}$$

where

$$\prod_{i=1}^{p}(a_i + b_i) = 1 \tag{B.18}$$

B2. Share verification analysis

Here we analyze the protocol to ensure that data owners' responses are correctly formed unit vectors where all indexes are zero except for only one index.

Correctness The protocol outputs whether the final answer is a unit vector (i.e., all the indexes are zero except for one location). If the vector is all zeroes then the sum will be zero. If the answer is a unit vector then the blinded message terms fall out leaving zero. If the vector is not a unit vector, the sum will be non-zero and we can discard this share.

Privacy All parties only view their own input and the final output. The blinding mechanism effectively masks the data owners true value.

Fairness All parties which participate will all view the same final answer as the shares sum to the same value.

References

[1] L. Sweeney, k-anonymity: a model for protecting privacy, international journal of uncertainty, Fuzziness Knowl. Based Syst. 10 (5) (2002) 557–570.

[2] N. Hunt, Netflix Prize Update, http://blog.netflix.com/2010/03/this-is-neil-hunt-chief-product-officer.html. URL https://web.archive.org/web/20100315105936/http://blog.netflix.com/2010/03/this-is-neil-hunt-chief-product-officer.html.

[3] G. Wang, B. Wang, T. Wang, A. Nika, H. Zheng, B.Y. Zhao, Defending against sybil devices in crowdsourced mapping services, in: R.K. Balan, A. Misra, S. Agarwal, C. Mascolo (Eds.), Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services, MobiSys 2016, Singapore,June 26-30, ACM, 2016, pp. 179–191, doi:10.1145/2906388.2906420.

[4] S. Rodriguez, Someone in China Might Know Where You Rode in Ubers Last Year, 2017, https://www.inc.com/salvador-rodriguez/uber-baidu-security.html,https://www.inc.com/salvador-rodriguez/uber-baidu-security.html.

[5] I. Dinur, K. Nissim, Revealing information while preserving privacy, in: F. Neven, C. Beeri, T. Milo (Eds.), Proceedings of the Twenty-Second ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 9-12, 2003, ACM, San Diego, CA, USA, 2003, pp. 202–210, doi:10.1145/773153.773173.

[6] R. Bhaskar, A. Bhowmick, V. Goyal, S. Laxman, A. Thakurta, Noiseless database privacy, in: D.H. Lee, X. Wang (Eds.), Proceedings of the Advances in Cryptology - ASIACRYPT 2011 - 17th International Conference on the Theory and Application of Cryptology and Information Security, Seoul, South Korea, December 4-8, 2011. Proceedings, Vol. 7073 of Lecture Notes in Computer Science, Springer, 2011, pp. 215–232, doi:10.1007/978-3-642-25385-012.

[7] C. Dwork, A Firm Foundation for Private Data Analysis, 2011, https://cacm.acm.org/magazines/2011/1/103226-a-firm-foundation-for-private-data-analysis/fulltext.

[8] C. Dwork, F. McSherry, K. Nissim, A. Smith, Calibrating noise to sensitivity in private data analysis, TCC, 2006,

[9] C. Dwork, A. Roth, The algorithmic foundations of differential privacy, Found. Trends Theor. Comput. Sci. 9 (3-4) (2014) 211–407, doi:10.1561/0400000042.

[10] C. Dwork, Differential privacy, in: M. Bugliesi, B. Preneel, V. Sassone, I. Wegener (Eds.), Proceedings of the 33rd International Colloquium on Automata, Languages and Programming, ICALP 2006, Venice, Italy, July 10-14, Part II, Vol. 4052 of Lecture Notes in Computer Science, Springer, 2006, pp. 1–12, doi:10.1007/117870061.

[11] C. Dwork, K. Kenthapadi, F. McSherry, I. Mironov, M. Naor, Our data, ourselves: privacy via distributed noise generation, in: Proceedings of the EUROCRYPT, 2006,

[12] J. Gehrke, E. Lui, R. Pass, Towards privacy for social networks: A zero-knowledge based definition of privacy, in: Y. Ishai (Ed.), Proceedings of the 8th Theory of Cryptography Conference, TCC 2011, Providence, RI, USA, March 28-30, 2011. Vol. 6597 of Lecture Notes in Computer Science, Springer, 2011, pp. 432–449, doi:10.1007/978-3-642-19571-626.

[13] J. Gehrke, M. Hay, E. Lui, R. Pass, Crowd-blending privacy, in: R. Safavi-Naini, R. Canetti (Eds.), Proceedings of the 32nd Annual Cryptology Conference, Advances in Cryptology - CRYPTO 2012 -, Santa Barbara, CA, USA, August 19-23, Vol. 7417 of Lecture Notes in Computer Science, Springer, 2012, pp. 479–496, doi:10.1007/978-3-642-32009-528.

[14] E. Sarigöl, D. Garcia, F. Schweitzer, Online privacy as a collective phenomenon, in: A. Sala, A. Goel, K.P. Gummadi (Eds.), Proceedings of the Second ACM Conference on Online Social Networks, COSN 2014, Dublin, Ireland, October 1-2, 2014, ACM, 2014, pp. 95–106, doi:10.1145/2660460.2660470.

[15] E.A. Horvat, M. Hanselmann, F.A. Hamprecht, K.A. Zweig, One plus one makes three (for social networks), PLOS One 7 (4) (2012) 1–8, doi:10.1371/journal.pone.0034740.

[16] M. Moore, Gay Men' Can be Identified by their Facebook Friends', 2009, http://www.telegraph.co.uk/technology/facebook/6213590/Gay-men-can-be-identified-by-their-Facebook-friends.html,http://www.telegraph.co.uk/technology/facebook/6213590/Gay-men-can-be-identified-by-their-Facebook-friends.html.

[17] M. Kosinski, D. Stillwell, T. Graepel, Private traits and attributes are predictable from digital records of human behavior, Proc. Natl. Acad. Sci. 110 (15) (2013) 5802–5805. https://doi.org/10.1073/pnas.1218772110.

[18] C. Jernigan, B. Mistree, Gaydar: Facebook friendships expose sexual orientation, First Mon. 14(10). doi: doi:10.5210/fm.v14i10.2611.

[19] K. Nissim, S. Raskhodnikova, A.D. Smith, Smooth sensitivity and sampling in private data analysis, in: D.S. Johnson, U. Feige (Eds.), Proceedings of the 39th Annual ACM Symposium on Theory of Computing, San Diego, California, USA, June 11-13, 2007, ACM, 2007, pp. 75–84. https://doi.org/10.1145/1250790.1250803.

[20] S.P. Kasiviswanathan, H.K. Lee, K. Nissim, S. Raskhodnikova, A.D. Smith, What can we learn privately? in: Proceedings of the 49th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2008, October 25-28, 2008, Philadelphia, PA, USA, IEEE Computer Society, 2008, pp. 531–540, doi:10.1109/FOCS.2008.27.

[21] N. Li, W.H. Qardaji, D. Su, On sampling, anonymization, and differential privacy or, k-anonymization meets differential privacy, in: H.Y. Youm, Y. Won (Eds.), Proceedings of the 7th ACM Symposium on Information, Compuer and Communications Security, ASIACCS '12, Seoul, Korea, May 2-4, ACM, 2012, pp. 32–33, doi:10.1145/2414456.2414474.

[22] S.L. Warner, Randomized response: a survey technique for eliminating evasive answer bias, J. Am. Stat. Assoc. 60 (309) (1965) 63–69.

[23] J.A. Fox, P.E. Tracy, Randomized Response: A Method for Sensitive Surveys, Beverly Hills California Sage Publications, 1986.

[24] J.C. Duchi, M.J. Wainwright, M.I. Jordan, Local privacy and minimax bounds: sharp rates for probability estimation, in: C.J.C. Burges, L. Bottou, Z. Ghahramani, K.Q. Weinberger (Eds.), Proceedings of the 27th Annual Conference on Neural Information Processing Systems and Proceedings of a meeting held on Advances in Neural Information Processing Systems 26: 2013. December 5-8, 2013, Lake Tahoe, Nevada, United States., 2013, pp. 1529–1537. http://papers.nips.cc/paper/5013-local-privacy-and-minimax-bounds-sharp-rates-for-probability-estimation

[25] U. Erlingsson, V. Pihur, A. Korolova, RAPPOR: randomized aggregatable privacy-preserving ordinal response, in: G. Ahn, M. Yung, N. Li (Eds.), Proceedings of the ACM SIGSAC Conference on Computer and Communications Security, Scottsdale, AZ, USA, November 3-7, 2014, ACM, 2014, pp. 1054–1067, doi:10.1145/2660267.2660348.

[26] A. Bittau, U. Erlingsson, P. Maniatis, I. Mironov, A. Raghunathan, D. Lie, M. Rudominer, U. Kode, J. Tinnés, B. Seefeld, Prochlo: strong privacy for analytics in the crowd, in: Proceedings of the 26th Symposium on Operating Systems Principles, Shanghai, China, October 28-31, 2017, ACM, 2017, pp. 441–459, doi:10.1145/3132747.3132769.

[27] A. Blum, K. Ligett, A. Roth, A learning theory approach to noninteractive database privacy, J. ACM 60 (2) (2013) 12:1–12:25, doi:10.1145/2450142.2450148.

[28] K. Chaudhuri, N. Mishra, When random sampling preserves privacy, in: C. Dwork (Ed.), Proceedings of the 26th Annual International Cryptology Conference, Advances in Cryptology - CRYPTO 2006, , Santa Barbara, California, USA, August 20-24, Vol. 4117 of Lecture Notes in Computer Science, Springer, 2006, pp. 198–213, doi:10.1007/1181817512.

[29] E. Boyle, N. Gilboa, Y. Ishai, Function secret sharing, in: E. Oswald, M. Fischlin (Eds.), Proceedings of the 34th Annual International Conference on the

Theory and Applications of Cryptographic Techniques, Advances in Cryptology - EUROCRYPT 2015 - Sofia, Bulgaria, April 26-30, Part II, Vol. 9057 of Lecture Notes in Computer Science, Springer, 2015, pp. 337–367, doi:10.1007/978-3-662-46803-612.

[30] F. Wang, C. Yun, S. Goldwasser, V. Vaikuntanathan, M. Zaharia, Splinter: Practical private queries on public data, in: A. Akella, J. Howell (Eds.), Proceedings of the 14th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2017, Boston, MA, USA, March 27-29, 2017, USENIX Association, 2017, pp. 299–313. https://www.usenix.org/conference/nsdi17/technical-sessions/presentation/wang-frank

[31] E. Boyle, N. Gilboa, Y. Ishai, Function secret sharing:improvements and extensions, in: E.R. Weippl, S. Katzenbeisser, C. Kruegel, A.C. Myers, S. Halevi (Eds.), Proceedings of the ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016, ACM, 2016, pp. 1292–1303, doi:10.1145/2976749.2978429.

[32] Wikipedia, List of United States Cities by Area, 2017. https://en.wikipedia.org/wiki/List_of_United_States_cities_by_area,https://en.wikipedia.org/wiki/List_of_United_States_cities_by_area.

[33] H. Kikuchi, J. Akiyama, G. Nakamura, H. Gobioff, Stochastic voting protocol to protect voters privacy, in: Proceedings of the IEEE Workshop on Internet Applications, WIAPP '99, IEEE Computer Society, Washington, DC, USA, 1999, pp. 103–105. http://dl.acm.org/citation.cfm?id=519623.837364

[34] California, Department of Transportation, 2017. http://pems.dot.ca.gov/,http://pems.dot.ca.gov/.

[35] Googles, Waze Announces Government Data Exchange Program with 10 Initial Partners, 2017. http://www.dot.ca.gov/cwwp/InformationPageForward.do,http://www.dot.ca.gov/cwwp/InformationPageForward.do.

[36] M. Lichman, UCI Machine Learning Repository, 2013. http://archive.ics.uci.edu/ml.

Joshua Joy is a Ph.D. Candidate in Computer Science at UCLA under the guidance of Professor Mario Gerla. His research interests include the Internet of Vehicles and scalable privacy within vehicular clouds. He recently led the deployment of CrowdZen at UCLA. CrowdZen privately computes the real-time activity levels across the UCLA campus and is used daily by thousands of students.

Dylan is a software engineer specializing in security. After receiving his masters from UCLA, he now works at Microsoft in the security field.

Ciarán Mc Goldrickis an Associate Professor at Trinity College Dublin's School of Computer Science and Statistics. His research spans challenged communications and networking, energy, security and scholarship and he is widely published in these fields. He focuses on questions such as: How will future devices and systems work?, as opposed to how do today's devices operate. Ciarán holds a Ph.D. in Electronic Engineering, with specialisms in Control and Signal and Image Processing. He has been a tenured academic in Trinity College Dublin for almost two decades, and has also held visiting appointments in UCLA. His research is, and has been, supported by National and International Awards, and he is an an active advocate for professionalism and excellence in learning and scholarship.

Dr. Mario Gerla is Professor of Computer Science at UCLA. He holds the Jon Postel Chair in Networking. He received his Ph.D. degree from UCLA. At UCLA, as a graduate student, in the early 70, he was part of the team that developed the ARPANET models and protocols that formed the basis of the modern INTERNET. After a period in industry, he joined the UCLA Faculty in 1976. Dr. Gerla is IEEE Fellow, serves on the IEEE TON Scientific Advisory Board and was recognized with the ACM Sigmobile Outstanding Contribution Award in 2015 and the IEEE INFOCOM Achievement Award in 2018.