Haze: Privacy-Preserving Real-Time Traffic Statistics

Joshua Brown Olga Ohrimenko Roberto Tamassia
{jwsbrown,olya,rt}@cs.brown.edu
Brown University
Providence, RI 02912

ABSTRACT

We consider traffic-update mobile applications that let users learn traffic conditions based on reports from other users. These applications are becoming increasingly popular (e.g., Waze reported 30 million users in 2013) since they aggregate real-time road traffic updates from actual users traveling on the roads. However, the providers of these mobile services have access to such sensitive information as timestamped locations and movements of its users. In this paper, we describe *Haze*, a protocol for traffic-update applications that supports the creation of traffic statistics from user reports while protecting the privacy of the users. Haze relies on a small subset of users to jointly aggregate encrypted speed and alert data and report the result to the service provider. We use jury-voting protocols based on threshold cryptosystem and differential privacy techniques to hide user data from anyone participating in the protocol while allowing only aggregate information to be extracted and sent to the service provider. We show that Haze is effective in practice by developing a prototype implementation and performing experiments on a real-world dataset of car trajectories.

1. INTRODUCTION

The interest in having access to our location based data has been confirmed one more time in June of this year when media articles reported that Google outbid Facebook to acquire navigation startup Waze [30] for $1 billion.[1] Waze is a mobile application that provides real-time traffic data to its users. This data comes from the users themselves, who contribute fresh data by uploading their GPS coordinates. Crowdsourcing navigation with 30 million users allows real time updates and, hence, is very useful when picking a route to avoid delays (see Figure 1).

User location data, however, contains very sensitive information. Analyzing GPS travel data can reveal the location of an individual's house and work, since there is a limited number of possible routes that one can take [15]. Moreover, since these data points are timestamped, one can also learn home departure and arrival times, as well as trip duration and purpose [22, 2]. In the case of Waze, user location data can be joined with his Google account and potentially reveal even more information. Moreover, court cases of unreported tracking confirm that mobile users are not comfortable with sharing their location data [18].

The challenge in preserving user privacy in services such as Waze is due to the nature of their functionality. In order to give quality service to its users, location data has to be collected for aggregation and statistical processing. However, we observe that precise user data is not required to provide traffic information such as current speed on the roads or presence of congestions. Consider the traffic information displayed by Waze and Google Maps in Figures 1 and 2, respectively. It consists of color-coded road segments representing traffic level, which is enough to plan routes effectively. To report such statistics, the service provider only needs to know if enough drivers are traveling at full speed, or if a significant number of users has reported slight or severe delays. Also, note that the maps display no data for the less-traveled roads. Conveniently for this type of application, areas which do not receive enough observations to allow reliable statistics are typically not a source of travel delays and can be safely ignored.

Motivated by the goal of providing privacy protection in crowdsourced real-time traffic update services, we propose a privacy-preserving version of such services, called Haze. Given that the data used by the navigation mobile app is already contributed by the drivers, we show that the aggregation functionality also can be outsourced and distributed among the users. Moreover, we develop a method that, allows meaningful traffic statistics to be extracted even when operating on user encrypted data.

Haze is a history-independent protocol that is invoked by the service provider whenever she wishes to update the traffic map. Figure 3 gives a high-level overview of the Haze framework. The task of traffic map update is outsourced to the users and is split in several phases: some users upload their data (users on the left in Figure 3), while others aggregate this data (users, aka authorities, on the right), and the final result is reported back to users by the service provider. We reduce data upload to a voting protocol where

[1]"Google Confirms Waze Maps App Purchase" in Wall Street Journal, Jun 11 2013.

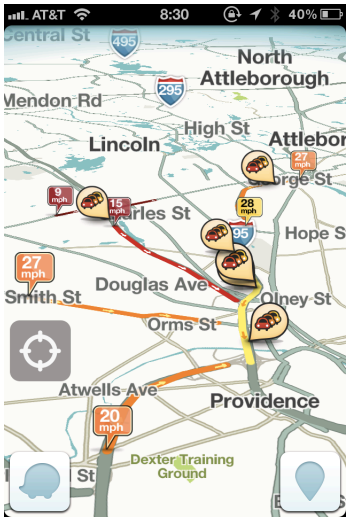


Figure 1: Snapshot of Waze traffic update. The icon with multiple cars shows traffic jam and the color of the roads shows the traffic speed.

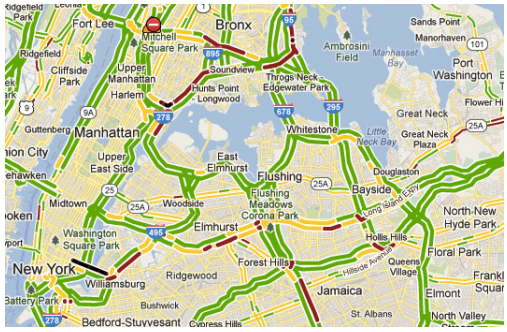


Figure 2: Snapshot of Google Maps traffic update. As in Waze the color shows the traffic speed: high (green), medium (yellow) and low (red).

instead of sending the exact data value, the user casts a vote for an observation that fits his data, e.g., instead of sending 45mph as his current speed he votes for the range 40–50mph. User vote is protected by encryption performed by the user before the upload. Hence, neither the service provider nor authorities have access to plain data of other users. Authorities tally the encrypted votes for all observations and report those that are deemed significant, i.e., enough users have reported the same observation.

Haze also takes into account privacy leaks that data encryption cannot solve, as well some of the problems it introduces. First, a potential privacy leak due to multiple runs of Haze arises when a user is not present at all invocations, e.g., because the user reached his destination. The Haze protocol guarantees differential privacy [13], i.e., it protects the privacy of each individual user by adding noise to the reported statistics so that a user's observation cannot significantly change the traffic status. Secondly, submitting encrypted data may encourage malicious users to report invalid data, e.g., by submitting speed reports for multiple roads. Haze prevents this behavior by enforcing users to prove the validity of their submitted observations. The dataflow of Haze is

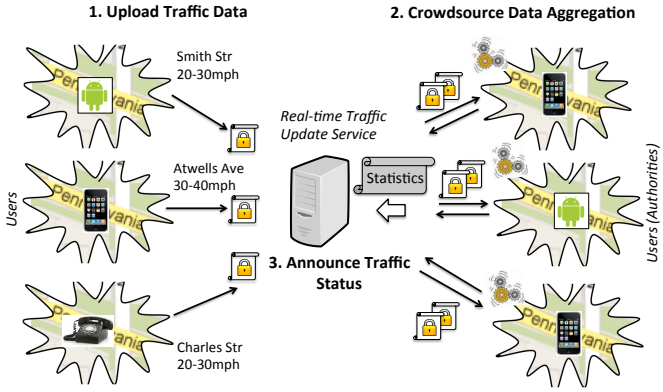


Figure 3: Haze overview: 1. Traffic data is uploaded encrypted by the users. 2. Other users (authorities) process encrypted data to extract aggregate statistics. 3. The traffic statistics are published by the service provider.

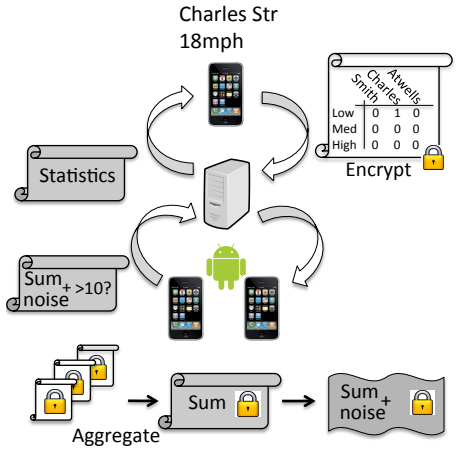


Figure 4: Data flow in Haze.

shown in Figure 4.

In summary, we make the following contributions:

- We describe a protocol for real-time privacy preserving traffic data statistics. It can report most of the information that a mobile app for traffic and road status can, i.e., traffic flow, presence of congestions, accidents, hazards, road work and gas stations.
- In contrast to previous work in this area [26, 23, 27, 28], we model gathering of traffic data as a voting protocol which allows us to hide the exact number of users that have made a certain observation.
- We design a customized differentially private mechanism which protects an individual user vote from changing the reported statistics by perturbing it. Previous work on protecting location based data either considers a curator that has access to all the data in the clear and adds noise to it [1, 16], or individual users add noise to their data before encrypting and reporting it [27, 28]. The former approach is not applicable

in the distributed scenario, while the latter may accumulate too much noise since clients do not coordinate this phase. Our protocol enjoys the benefits of both approaches: the noise addition is centralized while the data comes encrypted and has distributed origin.

- Our protocol does not rely on a trusted third party and pre-generated encryption keys while still offering user privacy and protection against implausible user data.
- We build a prototype of Haze and test its performance on real-world dataset.

The rest of this paper is organized as follows. We describe the model behind Haze in Section 2. The cryptographic constructions used by Haze are overviewed in Section 3. In Section 4, we present the Haze protocol. In Section 5, we analyze the security properties of Haze. Work related to Haze is discussed in Section 6. An experimental evaluation of Haze is given in Section 7.

2. MODEL

2.1 Setting

Participants We refer to the provider of the traffic information service as *server* and drivers that use this service, e.g., by downloading an app, as a set of *users*. The subset of users that participate in aggregation phase is referred to as *authorities*. Data aggregation is more expensive than data upload, hence, the role of an authority is assigned at random every time the protocol is invoked.

Data Since the protocol is run on modern mobile services, the application can easily determine the GPS coordinates of a user. We assume that user speed can be automatically determined by two timestamped GPS coordinates. Other data such as traffic jam, accident, hazards, road closure, gas station, or presence of a speed camera are manually entered by a user. We also assume that roads and their partition into segments by the server are publicly available and the user can determine them from his GPS coordinates.

Communication The users, including the authorities, interact with each other by sending messages to the server. We assume the server will not drop messages since this would result in non-accurate statistics and deteriorate her service.

2.2 Aggregated Statistics

Our method can report observations about road status such as current speed, traffic jam, construction, presence of speed cameras etc. As noted earlier, in order to report such data, the service provider does not need to know how many users have reported the presence of traffic jam, as long as she knows that enough users have observed it. We model such functionality as a voting procedure: users vote either 0 or 1 if they observed some event or not, the votes are then tallied, and the event is reported only if there were enough observations. For example, for speed reports we create several non-overlapping speed ranges that users can vote for. The set of all speed ranges is defined as $\mathcal{C}$ and the number of ranges in $\mathcal{C}$ depends on the granularity of information the service provider wants to report for the road segment i, e.g., $\mathcal{C} = \{(0,30),(30,60),(60,90)$ vs. $\mathcal{C} = \{(0,10),(10,20),\ldots,(80,90)\}$. A user then votes for the speed range that corresponds to his travel speed. To report the speed on segment i, one can pick the range that at least T_i users have voted for. Since several ranges in $\mathcal{C}$ may have more than T_i users, T_i may need to be adjusted by the service provider. For traffic jam reports $\mathcal{C}$ is simply a singular set {"traffic jam"}. A potential concern for the case when $|\mathcal{C}| > 1$ is that a malicious user may vote more than once. In Section 4.3 we show how to prevent users from such behavior.

2.3 Assumptions

We assume that users that upload their data are legitimate users and have a certified signing public key that allows them to sign their messages to the server. This allows anyone downloading the message to verify that it came from a valid user and has not been tampered with. We can relax this assumption when deploying the protocol with applications that support social network between its users. Since in this model the users have already established trust with their friends and family. For example, Waze allows users to sign in with their Facebook account.

2.4 Attack Model and Security Properties

The server is malicious and is interested in learning as much as possible about user data and, hence, cannot be trusted. The server may also collude with users and at most half of the authorities. The users are not trusted with their data reports and may try to swing traffic status in their favor.

We guarantee the following properties in the honest but curious model: every participant behaves according to his protocol but is curious to learn more about other users' data. In the malicious case, the protocol can succeed and be secure as long as at least half of the authorities are honest.

Server and Authority Obliviousness [28] The service provider and the authorities should not learn anything about user data beyond the aggregation information.

User Differential Privacy The user's data is protected by the differentially private mechanism that adds noise to the statistics before it is released. In particular, the output of the protocol in cases when the user uploads his data and not are equally likely. This property is useful when the protocol is run multiple times and the user does not participate in all the runs.

User Accountability Since the users upload their data anonymously, they may maliciously add misleading information such as being on several roads at the same time, or uploading inconsistent data (e.g., several speed measurements). Our protocol adds a verification step that allows to verify if the user's data is valid without learning what it actually is.

Fault Tolerance Haze can report statistics as long as at least half of the users acting as authorities remain active during an invocation. Every user that is not an authority can become inactive as soon as he uploads the data. Moreover, we do not require all legitimate users to participate in the protocol every time it is invoked. The user may upload information whenever he wishes to.

3. CRYPTOGRAPHIC PRIMITIVES

Haze relies on several cryptographic primitives that we briefly overview in this section.

El Gamal Cryptosystem All user data is encrypted before it gets sent to the authorities via the server. We use El Gamal cryptosystem [14] with public key PK, secret key SK and parameters p and q where q is a large prime (2048 bits in our implementation) and $p = 2qk + 1$. The message space is a subgroup $\mathbb{Z}_p^*$ of order q. Encryption of message m from this subgroup is denoted by $\mathsf{encrypt}_{\mathsf{PK}}(\xi, m)$ where ξ is picked randomly. Note that every message is encrypted together with a random value ξ, hence, encrypting the same message twice will very likely yield a new ciphertext. For ease of notation, we write $\mathsf{encrypt}_{\mathsf{PK}}(m)$, where the random nonce ξ is implied. As we will see later, no participant can make a call to function $\mathsf{decrypt}_{\mathsf{SK}}(m)$. An important property of El Gamal cryptosystem that we will take advantage of is additive homomorphism: $\mathsf{encrypt}_{\mathsf{PK}}(m_1 + m_2) = \mathsf{encrypt}_{\mathsf{PK}}(m_1) + \mathsf{encrypt}_{\mathsf{PK}}(m_2)$. (Here, operator + is overloaded for simplicity since in reality ciphertexts are tuples of two elements, which are multiplied correspondingly).

Threshold Cryptosystem Haze protects user data from anyone participating in the protocol, hence, there is no participant who can individually decrypt user data. To this end, we use threshold El Gamal cryptosystem [25] where secret key SK is split into shares between n participants, such that no single participant is able to determine SK nor decrypt a message encrypted under PK. Moreover, a ciphertext can be decrypted as long as at least $n/2$ participants remain honest.

Distributed Key Generation is required to generate keys for threshold cryptosystem. We use a distributed key generation protocol of [17], which succeeds if at least $n/2$ participants execute the protocol correctly. The high-level idea of this protocol is as follows: every participant k generates n secret shares $\mathsf{SK}_{k,k'}$ and sends one to every other participant k'. The secret share of participant k is the sum of all shares received from other participants, i.e., $\sum_{k'} \mathsf{SK}_{k',k}$. The public key is the sum of masked own shares of every participant k, i.e., sum of masked $\mathsf{SK}_{k,k}$. The actual protocol [17] is much more involved and takes into account participants that generate inconsistent shares. Haze will require only a subset of users $\mathcal{A}$, aka authorities, to participate in this protocol, and this subset can be different every time the protocol is invoked.

Zero Knowledge Proof An important feature of Haze is that all road observations and their aggregation is performed on encrypted data. However, malicious users may take advantage of this, e.g., by reporting several speed measurements or reporting traffic jam at multiple locations. To prevent submission of non-plausible data, users are required to return a proof for every ciphertext they submit. Since this proof should not reveal anything about the actual value to the verifier, we use zero knowledge (ZK) proof with Fiat-Shamir heuristic to make it non-interactive [8]. In Haze, we are interested in verifying if the ciphertext submitted by the user is an encryption of a message m that belongs to a set of plausible values (e.g., $\{0, 1\}$) without revealing which one.

Equality Test To report the aggregate statistics we mentioned in Section 2.2, a set of users $\mathcal{A}$ will have to determine whether the observations of the drivers are significant to be reported or not. The significance of the results is determined by the number of drivers that agree on an observation on the same road. Hence, users in the set $\mathcal{A}$ will be required to perform equality testing on encrypted values. To determine whether e is an encryption of m, we will use the method described in [20] which reduces the problem to determining if a ciphertext is equal to 1. The method does not require users to decrypt e but requires to generate a new set of keys via Distributed Key Generation protocol.

MixNet One of the goals of Haze is to release useful statistics about traffic status while revealing as little information as possible. For example, a report of traffic jam on a particular road segment does not require the service provider to know how many drivers have reported it, as long as she is convinced that a reasonable number of drivers have. Hence, we cannot run equality tests on the ciphertext e and known threshold values. Instead, we require the set of users, aka authorities, who run the equality test to create a permutation of encrypted threshold values and run equality test against this set (see below). We employ publicly verifiable MixNet protocol to create such a permutation. A MixNet protocol allows the authorities to encrypt and permute a set of values, $\mathcal{S}$, in a sequence and output the final permutation of encrypted values of $\mathcal{S}$, such that no one, including the authorities, can trace the values of $\mathcal{S}$ to their location in the output permutation. The protocol proceeds as follows. The first authority trivially encrypts $\mathcal{S}$ as $\mathcal{P}^0$, i.e., this encryption is not hiding and everyone can verify that $\mathcal{P}^0$ is an encryption of $\mathcal{S}$. This authority then randomly reencrypts and permutes $\mathcal{P}^0$ to get a permutation $\mathcal{P}^1$ and sends it to the next authority. In general, the k-th authority participating in the MixNet receives $\mathcal{P}^{k-1}$, randomly reencrypts and permutes it, and passes the resulting permutation, $\mathcal{P}^k$, to the next authority. The permutation produced by the last authority is the output of the MixNet. The process does not require authorities to decrypt the permutation that they receive. To make sure that every authority performs the reencryption and permutation faithfully, we use a method by [24], that allows efficient verification of the mix without revealing the underlying permutation.

As we will see later, Haze also uses MixNets to hide the noise added to the reported statistics.

Inequality Test We perform operation $\mathsf{InequalityTest}$ to learn if an encrypted tally e of those who voted for some observation is above a threshold T or not. We use the method of [20] to determine the result of a comparison operation without decrypting e. Instead of computing $\mathsf{decrypt}_{\mathsf{SK}}(e) \geq T$, the authorities compute the result of

$$\bigvee_{p=0}^{T-1} \left(\mathsf{decrypt}_{\mathsf{SK}}(e) \stackrel{?}{=} p \right). \tag{1}$$

The above equation can be easily computed by invoking the equality test T times. However, we cannot run this procedure sequentially for values $\{0, \ldots, T-1\}$, since this reveals exact count when the number of users is below T. Instead, authorities engage in the MixNet protocol on the set $\{0, \ldots, T-1\}$. The equality tests are then done against a permutation of encrypted values of 0 to $T-1$. If expression 1 returns 0, then $\mathsf{decrypt}_{\mathsf{SK}}(e)$ is above the threshold,

and InequalityTest returns true.

Differential Privacy [11, 13] is used to preserve privacy of an individual in statistical databases. This notion is captured by the following constraint: Let X and X' be two databases which differ only in one record, then a mechanism g is (ϵ, δ)-differentially private if $\forall S \subseteq \mathsf{Output}(g)$

$$\Pr[g(X) \in S] \leq \exp(\epsilon) \Pr[g(X') \in S] + \delta$$

In other words, an output of g will be as likely in the presence of an individual record, as in its absence.

4. HAZE PROTOCOL

First, we outline the Haze protocol in the idealized case when a trusted third party (TTP) is available to assist in the execution of the protocol by playing the role of an intermediary between the users and the service provider.

Every time a user wishes to report a traffic observation (e.g., the current user's speed on a given road segment), the user simply sends it to the TTP. The TTP aggregates the received observations into traffic reports (e.g., a report could state that the average speed on a given road segment is between 40 and 50 mph) and adds noise to the reported value to protect the privacy of the users when they are leaving or joining the protocol. Once the number of observations generating a report is above a certain threshold, the TTP transmits the report to the service provide. Note that in this scenario, the server does not see the association between the traffic reports forwarded by the TTP and the users who generated them. Also, the traffic reports received by the TTP have values that incorporate noise.

Since a protocol that employs a TTP is rather unrealistic, Haze does not assume the existence of a TTP. Instead, Haze simulates a TTP by means of a distributed protocol that splits the operations of the TTP among the server and a subset of the users, called authorities. We will show that we can guarantee the security properties stated in Section 2.4 as long as at least half of the authorities execute the protocol correctly.

4.1 Outline

The Haze protocol is invoked every time the service provider wishes to report statistics about a certain event, e.g., average speed on a set of roads or presence of construction work. The protocol consists of three phases: Setup, Data Upload, and Aggregation.

The service provider, aka the server, is involved in every phase of the protocol while users are engaged in two of the three phases depending on their role. In particular, a user can either contribute to the protocol by providing his traffic observation or by assisting in the privacy-preserving aggregation of the data provided by other users. We refer to the users who perform aggregations as *authorities*.

During the *Setup* phase, the authorities establish encryption keys to allow users to encrypt and hide their observations. The server also specifies the set of plausible candidate observations for this event, e.g., $\mathcal{C} = \{$"traffic jam"$\}$ or $\mathcal{C} = \{(0, 30), (30, 60), (60, 90)\}$. Next, the users participate in the *Upload* phase, by sending encrypted data to the server. The transmitted data record consists of the votes by the user for the candidate observations, where the vote is either 0 or 1 depending on whether the user has actually made this observation or not. For example, for speed ranges above the user would vote $\{1, 0, 0\}$ if his speed is 18mph. Note that authorities also can participate in this phase. Finally, during the *Result Computation* phase, the authorities operate on encrypted user data to report data of the event in question. In this phase, the authorities tally user votes, add noise to preserve the privacy of individual votes, and notify the server of observations that were reported by a significant number of users.

Haze is history independent and does not rely on a fixed set of participating users (and, hence, authorities). Thus, the role of a user is established during the setup phase of a single invocation of Haze and can change the next time the protocol is run. More importantly, the encryption keys that are created during the setup phase are used only once.

In the rest of this section, we describe the three phases of the Haze protocol.

The notation used is summarized in Table 1.

4.2 Setup

The server sets up the protocol by specifying a type of traffic for which she wishes to collect statistics. She then picks a set $\mathcal{C}$ of possible categories of values for this event. For example, to collect statistics on vehicle speed, the server can pick the following set of speed ranges: $\mathcal{C} = \{(0, 10), (10, 30), (30, 50), (50, \infty)\}$. Let C denote the size of set $\mathcal{C}$. A user reports an observation by casting a *ballot* V, which is a C-tuple of binary votes on the observation categories for the event. In the above example, let c denote the speed range $(30, 50)$. A user going at speed 40mph will set $V[c] = 1$ and $V[d] = 0$ for any other speed range $d \in \mathcal{C} - c$.

The server also picks N road segments for which she is interested in traffic statistics. For each road segment i, she sets a minimum threshold T_i on the number of users that report an observation for the statistics to be deemed significant.

The protocol relies on a set of users, $\mathcal{A}$, called *authorities*, to perform aggregation on user data in a privacy-preserving manner. The authorities are picked at random to reduce the

Table 1: Notation for the Haze protocol, referring to the speed statistics example.

N	number of road segments
M	number of users
$r(j)$, $s(j)$	road segment and value observed by user j
$\mathcal{C}$, $C = \lvert\mathcal{C}\rvert$	set of observation categories and its size
T_i	minimum number of users for receiving speed statistics on the ith road segment
$V_j[i][c]$	vote by user j on observation category c for road segment i
PK, SK	public key and secret key for El Gamal cryptosystem
$\mathcal{A}$, $A = \lvert\mathcal{A}\rvert$	set of authorities and its size
$\mathcal{N}$	array of noises for the differentially-private mechanism

chance of selecting users that may collude with the server. A user may flip a coin and decide if he is an authority or not. However, he may lie about the outcome of his coin. To make this process publicly verifiable, we propose that users extract randomness from a publicly verifiable random source, such as temperature at an airport or stock-index price.

Once the set of users who serve as authorities is identified, the authorities run the distributed key generation protocol from Section 3. By the end of this protocol, the public key, denoted PK, is announced and every authority, k, has a share of the encryption secret key, denoted SK_k. By using the threshold encryption scheme, $A/2$ authorities are enough to decrypt the data, where $A = |\mathcal{A}|$. This approach allows Haze to be fault tolerant in case $A/2 - 1$ authorities lose connectivity, and, at the same time, allows to defend against a coalition of at most $A/2-1$ malicious authorities who may want to try to decrypt data of individual users.

4.3 Data Upload

The data upload phase is summarized in Algorithm 1.

Voting We recall that a user casts a ballot, V, to indicate his observation. The C components of the ballot correspond to the categories for the observation. Thus, the user should vote 1 for only one category and 0 for the rest. Let $r(j)$ be the road segment that user j is traveling on and let $s(j)$ be his observation (e.g., $s(j) = 40$). Instead of sending value $s(j)$, the user casts vote 1 for the component of the ballot associated with the category c that includes value $s(j)$, that is, the user sets $V[c] = 1$ for category c such that $s(j) \in c$ (e.g., c denotes the speed range $(30, 50)$). Since all the communication between the users is done via the server, user j encrypts the votes in the ballot by using the public key PK, which had been generated by the authorities in the previous step (setup). We recall that we use a randomized encryption method. Thus, any two votes are computationally indistinguishable.

Suppose a user is in road segment i. To hide his location, the user casts a fictitious ballots for all other road segments in the system besides its real ballot for road segment i. Note that fictitious ballots should have all components set to 0. Thus, each user submits N ballots with C votes each.

Proving Valid Votes Since votes are encrypted and no participant learns the votes of other participants, a malicious user may upload invalid data. For example he could cast vote 2 (instead of 0 or 1), or vote for all road segments in his neighborhood to create the appearance of traffic and discourage other users from driving there. To detect malicious behavior, we require every user to also submit a non-interactive zero knowledge proof (Section 3) of the integrity of his votes. In particular, each user has to give a proof that in his ballot, at most one vote is 1 and the remaining votes are 0. Recall that an NIZK proof allows one to verify a property of a data set without learning the data set itself.

Overall, for every road i, a user j submits ballot $V_j[i]$ of C encrypted votes. Also, user j submits an NIZK proof Π_j of the validity of the ballots. Namely, Π_j proves that across all ballots submitted by user j, every vote is either 0 or 1, and there is at most one vote equal to 1.

Algorithm 1 Protocol executed by user j in the data upload phase (Section 4.3)

input: $r(j)$: current road segment of user j; $s(j)$: value observed by user j on road segment $r(j)$;.
output: for each road segment i, ballot $V_j[i]$ cast by user j on road segment i, consisting of C encrypted votes $V_j[i][c]$ on every observation category c; and Π_j: an NIZK proof of the integrity of the N ballots cast by user j

for all road segments $i \in \{1, \ldots, N\}$ **do**
  for all speed categories $c \in \mathcal{C}$ **do**
    { Vote 1 for the real observation and 0 for the rest}
    if $r(j) = i$ AND $s(j) \in c$ **then**
      $V_j[i][c] \leftarrow \mathsf{encrypt}_{\mathsf{PK}}(1)$
    else
      $V_j[i][c] \leftarrow \mathsf{encrypt}_{\mathsf{PK}}(0)$
    end if
  end for
  $\Pi_j[i] \leftarrow \mathsf{NIZK\_PROOF}(V[i])$
  Send ballot $V_j[i]$ to server
end for
Send proof Π_j of the validity of the ballots to server

4.4 Aggregation

The aggregation phase begins after the server receives and forwards to the authorities the encrypted ballots (V_j) and proofs of integrity (Π_j). We assume that the server will forward all data received since she is interested in providing informative service to her users and stay competitive with similar applications. The aggregation phase is outlined in Algorithm 2.

Data Verification and Aggregation Every authority verifies the integrity of the users' votes by running a non-interactive zero-knowledge verification protocol on the encrypted ballot V_j of every user j, using proof Π_j. Once all the votes are verified, every authority tallies the votes for every road segment and category into an array E. Note that E contains encrypted sum of votes since the underlying crypto system allows addition of encrypted values.

Differential Privacy We wish to protect every user from an adversary who is trying to trace the user between several runs of Haze. Since a user leaving the protocol may lead to changes in the traffic results reported for the road he was traveling on, and in turn reveal to the adversary the information we are trying to hide. We wish to protect such attacks by adding a noise to the statistics before releasing them to the service provider. In particular, we wish to make the output of the protocol differentially private: should be bounded. When adding noise we need to decide on the distribution of noise, how to pick values from this distribution, and at the same time hide which value was picked.

We define a special noise distribution as follows. Recall that given a vector of 0/1 votes $P = [p_1, \ldots, p_M]$ we wish to compute a function $f(P, T) = 0$, if $\sum P[i] < T$, and $f(P, T) = 1$, otherwise. We design a customized mechanism g by picking q uniformly from a set of size $1/\delta$: $\mathcal{N} = \{q \mid -\lfloor 1/2\delta \rfloor < q \leq \lfloor 1/2\delta \rfloor\}$, and setting $g(P, T) = 0$, if $\sum P[i] + q < T$, and $g(P, T) = 1$, otherwise. In Section 5 we show that g is differentially private.

Algorithm 2 Protocol executed by authority k in the aggregation phase (Section 4.4).

Step 1. Verify and aggregate encrypted ballots
input: encrypted ballots and their proofs for all the users, i.e., pairs (V_j, Π_j) for $j = 1 \dots M$
output: encrypted tally of the valid votes for every road segment and category, represented by a C-dimensional array E of size N
{Check the votes}
for all users j **do**
  if $\mathsf{verifyProof}(V_i, \Pi_i) = \mathsf{false}$ **then**
    remove j from the set of users
  end if
end for
{Aggregate votes for each road segment and category}
for all road segments i and categories c **do**
  $E[i][c] \leftarrow \sum_j V_j[i][c]$
end for

Step 2. Generate noise
input: Set of noises $\mathcal{N}$ { for differential privacy }
output: for all road segments i, and observation categories, $c \in \mathcal{C}$, permutation $\mathcal{P}^A_{i,c}$ of $\mathcal{N}$
for all road segments i and categories c **do**
  Participate in MixNet protocol for $\mathcal{N}$ to generate $\mathcal{P}^A_{i,c}$
end for

Step 3. Add noise
{Add noise from the last permutation sequence to the encrypted tallied values}
for all road segments i and categories c **do**
  $E'[i][c] \leftarrow E[i][c] + \mathcal{P}^A_{i,c}[1]$
end for

Step 4. Report the statistics
for all road segments $i \in \{1, \dots, N\}$ **do**
  for all observation categories $c \in \mathcal{C}$ **do**
    if $\mathsf{InequalityTest}(E'[i][c], T_i) = \mathsf{true}$ **then**
      Report (i, c) {c was picked by at least T_i users}
    end if
  end for
end for

To simulate the behavior of g, the authorities create a permutation of the encrypted $\mathcal{N}$ by invoking the MixNet protocol from Section 3. The first encrypted noise of the last permutation is then added to the final result. As long as at least half of the authorities execute MixNet correctly, no authority can track noise values in $\mathcal{N}$ to values in the final permutation $\mathcal{P}^A$. Each released statistics should be treated independently when adding noise from $\mathcal{N}$. Hence, the authorities mix a total $N \times C$ noise arrays of size $|\mathcal{N}|$.

Statistics Extraction Let E' be an array of tallied encrypted values with added noise for every road segment and observation. To extract information from E', the authorities need to determine if the event has been observed by a significant number of users. In particular, for every road segment i and observation category c, the authorities need to check if at least T_i users voted for c on the ith segment, i.e., if $\mathsf{decrypt}_{\mathsf{SK}}(E'[i][c]) \geq T_i$. Recall that we do not allow anyone participating in the protocol, including authorities, to learn the exact number of users who reported statistics for individual road segments. Hence, the authorities are not allowed to decrypt E'. Even if an authority is malicious, he cannot decrypt E' unless other $A/2-2$ authorities are malicious as well. Moreover, as we mentioned earlier comparison over encrypted data is not trivial and expensive.

We use the Inequality Test (Section 3) for each value $E'[i][c]$ and threshold T_i. Observations for which the test returns true (i.e., the number of users who voted for this observation exceeds the threshold) are reported to the server. Note that the users who are not authorities do not participate during this phase, and can go offline as soon as they upload their data. Also, users are not required to participate in every data upload and, hence, the protocol is tolerant against any number of users going offline.

4.5 Extensions

Data upload requires each user to encrypt and send $N \times C$ messages when El Gamal encryption is used. One can improve the complexity to $N + C$ if the BGN [3] scheme is used; this scheme is also additively homomorphic but allows one multiplication operation on the ciphertexts.

Since users may be crossing multiple road segments when uploading the data, our protocol can be easily extended to multiple-segment voting. In this case, the ballot-validity proof should include a proof that the user did not vote for more than the allowed number of segments.

5. SECURITY

In this section, we show that Haze provides several formal privacy guarantees. We begin by proving that the noise addition step yields differential privacy.

THEOREM 1. *Let P be a vector of 0/1 votes $P = [p_1, \dots, p_M]$ and q be a value picked uniformly at random from set $\mathcal{N} = \{-\lfloor 1/2\delta \rfloor + 1, \dots, \lfloor 1/2\delta \rfloor\}$. Define mechanism g as $g(P,T) = 0$, if $\sum P[i] + q < T$, and $g(P,T) = 1$, otherwise. We have that g is $(0, \delta)$ differentially private.*

PROOF. We show that for each $x \in \mathsf{Output}(g)$, we have

$$\Pr[g(P,T) = x] \leq \exp(\epsilon) \Pr[g(P',T) = x] + \delta : \quad (2)$$

where P and P' differ in one location, i.e., either $\sum P[i] = \sum P'[i] + 1$ or $\sum P[i] = \sum P'[i] - 1$.

We consider the case when $x = 1$ (the case for $x = 0$ is symmetrical) and expand Inequality 2:

$$\Pr[g(P,T) = 1] - \Pr[g(P',T) = 1] \quad = \quad (3)$$

$$\Pr[\sum P[i] + q \geq T] - \Pr[\sum P'[i] + q \geq T] \quad (4)$$

Note that whenever $\sum P[i] = \sum P'[i] - 1$, $\Pr[\sum P[i] + q \geq T] \leq \Pr[\sum P'[i] + q \geq T]$ and the expression in Equation 4 is always less than δ. Hence, we consider the case when $\sum P[i] = \sum P'[i] + 1$ in Equation 4:

$$\Pr[\sum P[i] + q \geq T] - \Pr[\sum P[i] - 1 + q \geq T] \quad =$$

$$\frac{|\mathcal{N}| - T + \sum P[i]}{|\mathcal{N}|} - \frac{|\mathcal{N}| - T + \sum P[i] - 1}{|\mathcal{N}|} \quad = \quad \delta$$

□

Haze protects individual user privacy even when the service provider colludes with authorities and users, i.e., the protocol is server and authority oblivious (see Section 2.4).

THEOREM 2. *Haze is server oblivious and authority oblivious if less than half of of authorities collude with the server.*

PROOF. *(Sketch)* The privacy of user votes for an individual segment and observation pair follows from the security of a single observation (ballot) in the jury voting protocol in [20]. We can extend the analysis to multiple segments and observations since the only inference that the adversary can make from seeing previous observation results, is the number of users who can potentially vote for the rest of the ballots. This again reduces to the proof for a single observation where the number of users participating in voting is known, i.e., as it is in [20]. The proof depends on the correct execution of underlying cryptographic primitives used in the protocol, i.e., Distributed Key Generation and MixNet, which operate correctly as long as at least half of the authorities are not malicious. If the adversary colludes with a subset of users, i.e., she knows their votes, then she can learn if the remaining honest users voted above or below the threshold, but cannot infer the individual votes. □

6. RELATED WORK

In this section, we discuss work related to Haze. We begin by discussing systems with goals similar to those of Haze. Table 2 gives a comparison of Haze with these systems.

Private Aggregation of Distributed Data Preserving the privacy of distributed spatiotemporal data during aggregation by an untrusted party has been studied in [4, 26, 23, 27, 28, 5]. PrivStats [26] provides a scheme where mobile users report encrypted position data to location-based applications that aggregate and report statistics on this data. PrivStats preserves the privacy of the user's location and allows the application to verify that users have provided valid data. The method uses a smoothing module that adds fictitious records and noise to the data. Also, the smoothing module has a secret key to the encryption scheme and is responsible for decrypting server-aggregated statistics. This module is installed on cars and is fully trusted. Hence, PrivStats's trust model is very different from ours.

The system by Carbunar *et al.* [5] collects aggregated location based statistics via a voting protocol. It relies on a trusted anonymizer between users and parties interested in aggregated results (called venues). Instead, Haze does not rely on a trusted third party. Also, unlike Haze, this method does not provide differential privacy guarantees. The model is also different from ours as it involves 3 parties: users, social network providers and venues. The user receives his encryption key from the provider and sends encrypted data to the venue, proving vote validity to him. Haze does not rely on receiving keys from a service provider; instead, the authorities run a distributed key-generation protocol. Haze also differs in the type of statistics it provides: it reports whether a vote count is higher or lower than some threshold, but never the exact number of users. Instead, in [5], the number of users is revealed, since the adversary can decrypt the tally only if exactly k users have casted their votes.

Table 2: Comparing Haze with previous work.

	Carbunar *et al.* [5]	Chan *et al.* [6]	PASTE [27]	PrivStats [26]	Sepia [4]	Haze
Differential Privacy		✓	✓			✓
Fault Tolerance		✓		✓	✓	✓
Multiple Statistics	✓			✓	✓	✓
Distributed Key Generation					✓	✓
No Trusted Third Party		✓	✓		✓	✓

The protocols described in [23, 27, 28] preserve distributed data aggregation by requiring each user or node to add differentially-private noise to their data before reporting it. Monreale *et al.* [23] describe a differentially-private mechanism that allows every node to report a trajectory of its moves, while preserving privacy of the individual moves. However, perturbing the reported trajectory does not hide the link between the user and his data. Even though the server does not know precisely when the nodes moved from one location to another, she learns the trend of the node behavior. On the other hand, in the framework of Rastogi and Nath [27] and that of Shi *et al.* [28], the server never sees user data in the clear. In their schemes, every user adds noise to his data, encrypts it using additively homomorphic encryption with his share of secret key, and sends it to the server, who then aggregates received data. To be able to decrypt the aggregated result, in [27], the server sends the aggregated value back to every user for decryption and then joins the partial decryptions to get the final result. The method of [28] also uses an encryption scheme that allows an aggregated value to be decrypted only if every user has contributed his data. Chan *et al.* [6] show how to make these methods fault tolerant by requiring every user to submit their data logarithmic number of times. Both methods are described in the scenario where all users are observing the data that can be aggregated into a single result, e.g., sum over data points of the same type. It becomes tricky to adapt these methods to traffic statistics that we consider here. In our scenario different users may provide observations for different roads, and, hence, every road has its own aggregated result. A naive extension to [27] and [28] would allow the server to link the users and the roads they are on. It is also worth noting that in Haze, we apply differentially private mechanism to aggregated result, while in [27, 28], users independently apply noise before submitting their data. Since this is a distributed mechanism, the noise variance may reduce the utility of the aggregated result.

Sepia [4] describes a framework for private monitoring of network traffic for multiple events. Their approach is similar to ours in the sense that the system also consists of input peers, who deliver their data encrypted, and privacy peers (called authorities in our protocol), who perform aggregation of this data. Although the setup is similar, the execution of data collection and aggregation is very different from that of Haze. In order to aggregate statistics across events, privacy peers need to group encrypted data by their event id. To this end, privacy peers engage in distributed equality protocol for every submitted encrypted user data. Once aggregated the peers engage in distributed comparison protocol. Haze achieves similar functionality by using voting and mix-net protocols. Additionally, statistics reported by Haze are differential private.

Next, we overview work on the related areas of differential privacy, private location-based queries, and jury voting.

Differential Privacy Differential privacy was formulated in [13] and a survey of further results is presented in [11]. Differential privacy of statistical databases protects privacy of an individual by adding noise to the released data. Noisy data adds uncertainly on whether the data of an individual is present in the database or not. A common scenario is to assume a presence of a curator who has access to all the data and adds noise before releasing it, or adds noise to query results when operating in an interactive mode.

Differentially-private mechanisms have been proposed to report the trajectory of moving objects [1], raw time-series traffic data [16] and spatial data decomposition [7]. However, these methods assume a trusted party that has access to already collected data and is responsible for perturbing it before publishing. This differs significantly from our work where data is distributed and there is no trusted third party to aggregate and anonymize the data. Differential privacy was also considered in the distributed scenario in [12]. However, in this work all users have to engage in a distributed noise generation protocol before releasing individual data.

Private Location Based Queries Location based privacy has been considered also from the view point of a single user who is making queries based on his location and wishes to receive such information without revealing his location [21, 9, 19]. This work can be split into k-anonymization techniques where user's identity cannot be distinguished from other k-1 users, and cloaking techniques, where a larger region is sent during the query instead of precise user location.

Jury Voting The jury voting protocol by Hevia and Kiwi [20] allows participants to determine whether a vote tally is above a threshold or not. Haze extends the participant model and cryptographic primitives of [8] and [20], to accommodate a more complicated ballot structure and guarantee differential privacy of individual votes.

7. EXPERIMENTAL RESULTS

We have developed a full prototype implementation of Haze based on several primitives implemented in the Civitas [10] voting system, including distributed key generation, El Gamal cryptosytem, distributed exponentiation (used in equality testing), and interface to GNU GMP (a multiple-precision arithmetic library). The Haze prototype includes the efficient verification of a mix-protocol from [24], an extension of the NIZK ballot validity protocol to support voting for more than one segment, our differentially-private algorithm, and the reporting of statistics for multiple roads. The experiments were conducted on a 32 core 2.6GHz Opteron 6282 SE with 64GB RAM running 64bit Debian Wheezy. For the El Gamal cryptosystem, we used a 2048 bit key and message space of 160 bits. The noninteractive ZK proofs use the SHA-256 function. The timing of the experiments was averaged across 10 runs. We used data from TAPAS Cologne Project [29]. This dataset contains actual GPS coordinates and speeds of drivers in the city of Cologne, as measured during a 6am–8am time period.

The reported statistics are perturbed by noise depending on parameter δ. In Table 3, we show how the value of δ influences the accuracy of the results in terms of precision and recall measurements. The results are taken from 1,000 data points, during one minute, where 4,000 users had to pick among 3 speed ranges on one road segment, and the threshold on the number of users was set to $T = 11$. As expected, the accuracy of the results degrades with lower values of δ.

We optimize the equality test in the results aggregation step by requiring all users to participate in mixing $\{0, \ldots, T_i - 1\}$ sets for tuples (i, c) (Algorithm 2). Thus, the task is distributed over all M participants and not just the authority set. In Table 4, we measure the total time to run the protocol for different number of users but fixed number of authorities. The reported time is a lapse of time from the start of setup protocol till the authorities output result for every road. Since user can upload his data in parallel and authorities can run the MixNet, ballot verification independently, we allowed one core per protocol participant (the time for 100 participants is estimated from the total time of the sequential execution). It is interesting to note that our protocol takes advantage as the number of users grows since MixNet jobs can be distributed over a larger set.

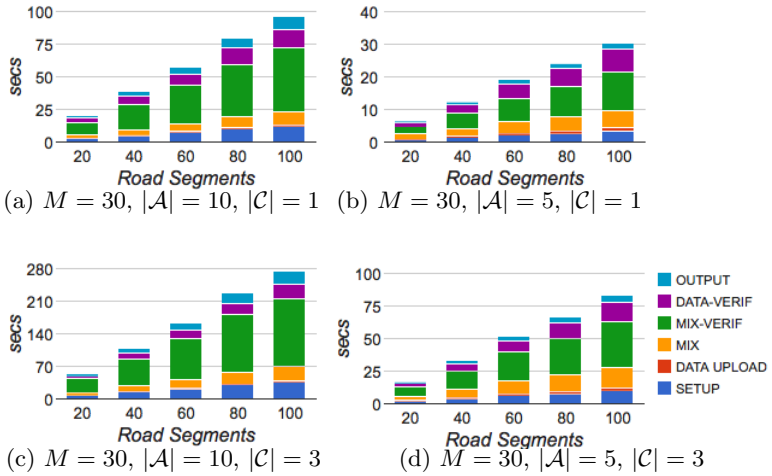
In Figure 5, we show the split of the total time across different steps in the protocol for 30 users and 5 or 10 authorities. As expected, the most expensive part of the protocol is the MixNet verification. We also vary the number of observation categories that users can vote for: a single category in Figures 5(a) and 5(b) (e.g., whether road work has been observed) and 3 categories (e.g., low/mid/high speed ranges) in Figures 5(c) and 5(d). The running time of the protocol increases with the number of observation categories.

Table 3: Precision and recall values for observations reported by Haze differentially-private mechanism during a period of 60 seconds.

δ	0.5	0.3	0.1
Precision	97%	98%	86%
Recall	100%	98%	88%

Table 4: Total time to run Haze for $N = 100$ roads, $\mathcal{C} = 3$, $\delta = 1/3$, $|\mathcal{A}| = 10$ and $T = 10$ for all roads.

Number of Users (M)	20	25	30	100
Total Protocol Time (secs)	351	314	274	40



(a) $M = 30$, $|\mathcal{A}| = 10$, $|\mathcal{C}| = 1$ (b) $M = 30$, $|\mathcal{A}| = 5$, $|\mathcal{C}| = 1$

(c) $M = 30$, $|\mathcal{A}| = 10$, $|\mathcal{C}| = 3$ (d) $M = 30$, $|\mathcal{A}| = 5$, $|\mathcal{C}| = 3$

Figure 5: Breakdown of the running time of the Haze protocol by step.

Acknowledgments

This research was supported in part by the National Science Foundation under grants IIS-1212508 and CNS-1228485.

8. REFERENCES

[1] R. Assam, M. Hassani, and T. Seidl. Differential private trajectory protection of moving objects. In *ACM SIGSPATIAL Workshop on GeoStreaming*, IWGS, pages 68–77, 2012.

[2] T. Bhattacharya, L. Kulik, and J. Bailey. Extracting significant places from mobile user GPS trajectories: a bearing change based approach. In *Advances in Geographic Information Systems*, SIGSPATIAL, pages 398–401, 2012.

[3] D. Boneh, E.-J. Goh, and K. Nissim. Evaluating 2-DNF formulas on ciphertexts. In *Theory of Cryptography*, TCC, pages 325–341, 2005.

[4] M. Burkhart, M. Strasser, D. Many, and X. Dimitropoulos. SEPIA: privacy-preserving aggregation of multi-domain network events and statistics. In *USENIX Conf. on Security*, pages 15–15, 2010.

[5] B. Carbunar, M. Rahman, J. Ballesteros, and N. Rishe. Eat the cake and have it too: Privacy preserving location aggregates in geosocial networks. *CoRR*, abs/1304.3513, 2013.

[6] T.-H. Chan, E. Shi, and D. Song. Privacy-preserving stream aggregation with fault tolerance. In *Financial Cryptography and Data Security*, volume 7397 of *LNCS*, pages 200–214, 2012.

[7] G. Cormode, C. Procopiuc, D. Srivastava, E. Shen, and T. Yu. Differentially private spatial decompositions. In *IEEE Int. Conf. on Data Engineering*, ICDE, pages 20–31, 2012.

[8] R. Cramer, R. Gennaro, and B. Schoenmakers. A secure and optimally efficient multi-authority election scheme. In *Theory and Application of Cryptographic Techniques*, EUROCRYPT, pages 103–118, 1997.

[9] M. L. Damiani, E. Bertino, and C. Silvestri. The PROBE framework for the personalized cloaking of private locations. *Trans. Data Privacy*, 3(2):123–148, 2010.

[10] A. M. Davis, D. Chmelev, and M. R. Clarkson. Civitas: Implementation of a threshold cryptosystem. Computing and Information Science Technical Report, Cornell U., 2008. `http://hdl.handle.net/1813/11661`.

[11] C. Dwork. Differential privacy: A survey of results. In *Conf. on Theory and Applications of Models of Computation*, TAMC, pages 1–19, 2008.

[12] C. Dwork, K. Kenthapadi, F. McSherry, I. Mironov, and M. Naor. Our data, ourselves: privacy via distributed noise generation. In *Theory and Applications of Cryptographic Techniques*, EUROCRYPT, pages 486–503, 2006.

[13] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of Cryptography*, TCC, pages 265–284, 2006.

[14] T. El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *Advances in Cryptology*, CRYPTO, pages 10–18, 1985.

[15] V. V. Elango, S. Khoeini, Y. Xu, and R. Guensler. Longitudinal GPS travel data and breach of privacy via enhanced spatial and demographic analysis. In *Transportation Research Board 92nd Annual Meeting*, 2013.

[16] L. Fan, L. Xiong, and V. Sunderam. Differentially private multi-dimensional time series release for traffic monitoring. In *Data and Applications Security and Privacy*, DBSec, 2013. To appear.

[17] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Secure distributed key generation for discrete-log based cryptosystems. *J. Cryptol.*, 20(1):51–83, 2007.

[18] Google Faces $50 Million Lawsuit Over Android Location Tracking. Accessed on June, 2013. `http://www.wired.com/business/2011/05/google-faces-lawsuit`.

[19] M. Gruteser and D. Grunwald. Anonymous usage of location-based services through spatial and temporal cloaking. In *Mobile Systems, Applications and Services*, MobiSys, pages 31–42, 2003.

[20] A. Hevia and M. A. Kiwi. Electronic jury voting protocols. *Theor. Comput. Sci.*, 321(1):73–94, 2004.

[21] P. Kalnis, G. Ghinita, K. Mouratidis, and D. Papadias. Preventing location-based identity inference in anonymous spatial queries. *IEEE Trans. on Knowl. and Data Eng.*, 19(12):1719–1733, 2007.

[22] J. Krumm. Inference attacks on location tracks. In *Int. Conf. on Pervasive Computing*, PERVASIVE, pages 127–143, 2007.

[23] 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*, AGILE, pages 225–245, 2013.

[24] C. A. Neff. A verifiable secret shuffle and its application to e-voting. In *ACM Conf. on Computer and Communications Security*, CCS, pages 116–125, 2001.

[25] T. Pedersen. A threshold cryptosystem without a trusted party. In *Advances in Cryptology*, EUROCRYPT, pages 522–526, 1991.

[26] R. A. Popa, A. J. Blumberg, H. Balakrishnan, and F. H. Li. Privacy and accountability for location-based aggregate statistics. In *ACM Conf. on Computer and Communications Security*, CCS, pages 653–666, 2011.

[27] V. Rastogi and S. Nath. Differentially private aggregation of distributed time-series with transformation and encryption. In *ACM Conf. on Management of Data*, SIGMOD, pages 735–746, 2010.

[28] E. Shi, T.-H. H. Chan, E. G. Rieffel, R. Chow, and D. Song. Privacy-preserving aggregation of time-series data. In *Symp. on Network and Distributed System Security*, NDSS, 2011.

[29] TAPASCologne Project. Accessed on June, 2013. `http://sourceforge.net/apps/mediawiki/sumo/index.php?title=Data/Scenarios/TAPASCologne`.

[30] Waze. Accessed on June, 2013. `http://www.waze.com/`.