Haze: Privacy-Preserving Real-Time Traffic Statistics

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

ABSTRACT

We consider mobile applications that let users learn traffic conditions based on reports from other users. 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 introduce the model and general approach of *Haze*, a system for traffic-update applications that supports the creation of traffic statistics from user reports while protecting the privacy of the users. We also present preliminary experiments that indicate potential for a practical deployment of Haze.

Categories and Subject Descriptors

D.4.6 [**Security and Protection (K.6.5)**]: Cryptographic controls

General Terms

Security

Keywords

Privacy, traffic statistics, private aggregation

1. INTRODUCTION

Several popular mobile applications provide real-time traffic data to its users. This data comes from the users themselves, who contribute fresh data by uploading their GPS coordinates. For example, the Waze app leverages 30 million users to offer real-time traffic updates (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 [8]. Moreover, since these data points are timestamped, one can also learn home departure and arrival times, as well as trip duration and purpose [2, 11].

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 in Figure 1. 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.

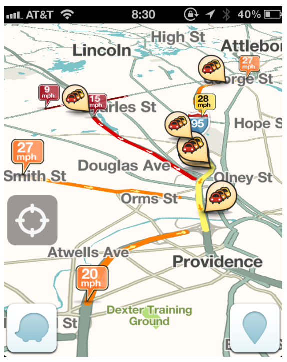

Figure 1: Snapshot of Waze traffic update. The icon with multiple cars indicates a traffic jam and the color of the roads shows the traffic speed.

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 also the aggregation functionality 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 data encrypted by users.

Haze is a history-independent protocol that is invoked by the service provider whenever she wishes to update the traffic map. Figure 2 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 2), while others aggregate this data (users, aka authorities, on the right), and the final result is reported back to users by the service provider.

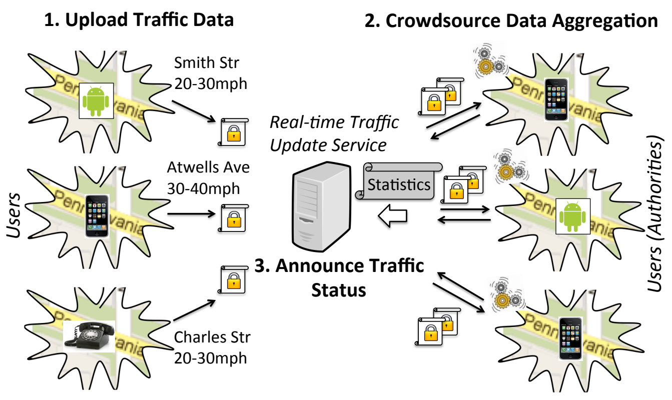

Figure 2: Haze overview: 1. Traffic data is uploaded encrypted by the users. 2. Other users (authorities) process encrypted data to extract aggregate statistics. 3. Traffic statistics are published by the service provider.

We reduce data upload to a voting protocol where 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. The user's 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 as 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 [7], 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.

In this paper, we introduce a general approach for real-time privacy-preserving traffic statistics. Unlike previous work in this area [12, 13, 14, 15], we model gathering of traffic data as a voting protocol that allows us to hide the exact number of users that have made a certain observation. Our approach also supports differential privacy, which prevents an individual user vote from changing the reported statistics. Previous work on protecting location-based data either employs a centralized model, where a trusted curator adds noise to the data [1, 9], or a distributed framework that only supports reports for a single aggregated result [14, 15]. We also overview preliminary experiments conducted on a simulation of Haze, using a real-world dataset. An extended version of this paper presents the Haze protocol, its cryptographic primitives, and a formal security analysis [3].

2. MODEL

We refer to the provider of the traffic information service as the *server* and to the drivers that use this service, e.g., by downloading an app, as the *users*. The subset of users that participate in the aggregation phase is referred to as the *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.

Since the protocol is run on modern mobile devices, 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.

Our method can report aggregates of the above observations. 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 a 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. 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.

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.

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 in their favor. In view of this attack model, Haze guarantees the following properties. *Server and Authority Obliviousness* [15]: the service provider and the authorities should not learn anything about user data beyond the aggregated information. *User Differential Privacy:* the user's data is protected by a differentially private mechanism that adds noise to the statistics before it is released. *User Accountability:* Haze verifies that data uploaded by the users is valid without learning individual information. *Fault Tolerance:* Haze can report statistics as long as at least half of the users serving as authorities remain active during an invocation. Moreover, we do not require all legitimate users to participate in the protocol every time it is invoked.

3. HAZE

The Haze protocol is invoked every time the service provider wishes to report statistics about a certain event, e.g., average speed or presence of road 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 (the *authorities*).

During the *Setup* phase, the authorities establish encryption keys to allow users to encrypt their observations. The server also specifies the set of plausible candidate observations for this event. Next, the users participate in the *Upload* phase, where they send encrypted data to the server. The transmitted data record consists of the votes by the user for the candidate observations. Note that authorities also can participate in this phase. Finally, during the *Aggregation* 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.

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, depending on the event candidate observations could be $\mathcal{C} = \{\text{“traffic jam”}\}$ or $\mathcal{C} = \{(0, 30), (30, 60), (60, 90)\}$ to denote speed ranges. 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 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. The authorities generate and announce the public key PK and store their individual share of the secret key.

Data Upload: A user reports an observation by casting a *ballot* V. Let C denote the size of set $\mathcal{C}$ of possible categories for an event. Then the ballot V is a C-tuple of binary votes on the observation categories for the event. The user votes 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., $s(j) = 40$mph). 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 $(30, 60)$) and $V[d] = 0$ for any other speed range $d \in \mathcal{C} - \{c\}$. A user hides his location by casting a fictitious ballot of 0's for all other road segments in the system besides its real ballot for road segment $r(j)$. Thus, each user submits N ballots with C votes each.

Since all the communication between the users is done via the server, user encrypts the votes in the ballot by using the public key PK, which had been generated by the authorities during the setup phase.

To detect invalid votes cast by users, whether with malicious intent or due to glitches, users submit a proof of the integrity of the votes without revealing their value. 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. Overall, a user j submits a ballot $V_j[i]$ (consisting of C encrypted votes) for every road i, and a proof Π_j of the validity of the ballots.

Aggregation: The aggregation phase begins after the server receives and forwards to the authorities the encrypted ballots ($V_j[i]$) 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.

Every authority verifies the integrity of the users' votes by running a verification protocol on the encrypted ballots $V_j[i]$ 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 sums of votes since we assume the underlying crypto system allows addition of encrypted values.

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 the information we are trying to hide. We protect from such attacks by adding noise to the statistics before releasing them to the service provider. In particular, we make the output of the protocol differentially private. Thus, the change in the protocol output caused by an individual's data is bounded.

4. RELATED WORK

In PrivStats [13], mobile users report encrypted position data to location-based applications via a trusted device that generates noise and performs decryption of aggregated statistics. 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. Unlike Haze, it relies on a trusted third-party for privacy protection, it does not support differential privacy, and assumes that users receive encryption keys from the service provider.

Monreale *et al.* [12] describe a differentially-private mechanism that allows every node to report a perturbed 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, and reveals the behavior trend to the server. On the other hand, in the framework of Rastogi and Nath [14] and that of Shi *et al.* [15], the server never sees user data in the clear. In their schemes, every user adds noise to his data, encrypts it, and sends it to the server, who then aggregates received data. 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 statistics over multiple traffic events that we consider here, where every road has its own set of observations. Also, Haze applies a differentially private mechanism to the aggregated result instead of adding noise in a distributed manner before aggregation, which can potentially reduce the utility of the final 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 cryptographic primitives used for data collection and aggregation are very different from those of Haze. Additionally, the statistics reported by Haze are differentially private.

5. EXPERIMENTAL RESULTS

We have developed a preliminary prototype implementation of Haze based on several primitives implemented in the Civitas [6] voting system. The experiments were conducted on a 32 core 2.6GHz Opteron 6282 SE with 64GB RAM running 64bit Debian Wheezy. We used data from the TAPAS Cologne Project [10]. This dataset contains actual GPS coordinates and speeds of drivers in the city of Cologne, as measured during a 6am–8am time period.

In Table 1, we measure the total time to run the protocol for different numbers of users and 10 authorities. The reported time is the total time it takes to run Haze. 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 of the growing number of users since some steps of the aggregation phase can be distributed over a larger set of users.

In Figure 3, we show the split of the total time across different phases in the protocol for 30 users and 5 authorities. The most expensive part of the protocol is the aggregation phase. We also vary the number of observation categories that users can vote for: a single category in Figure 3 (left) (e.g., whether road work has been observed) and 3 categories (e.g., low/mid/high speed ranges) in Figure 3 (right).

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

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

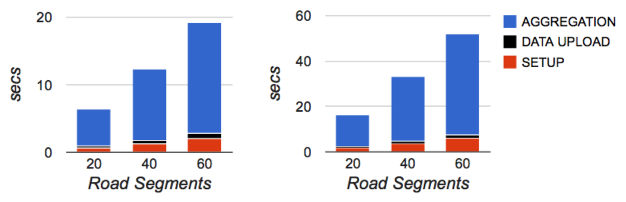

Figure 3: Breakdown of the running time of the Haze protocol into phases for 30 users, $|\mathcal{A}| = 5$, $|\mathcal{C}| = 1$ (left) and $|\mathcal{C}| = 3$ (right).

Acknowledgments

This research was supported in part by the National Science Foundation under grants IIS-1212508 and CNS-1228485. We are grateful to Isabel Cruz for valuable advice on this project. We also thank Naphtali Rishe and Goce Trajcevski for useful discussions.

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 *ACM Conf. on Advances in Geographic Information Systems*, SIGSPATIAL, pages 398–401, 2012.

[3] J. W. S. Brown, O. Ohrimenko, and R. Tamassia. Haze: Privacy-preserving real-time traffic statistics. *CoRR*, abs/1309.3515, 2013.

[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*, 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] 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`.

[7] 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.

[8] 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.

[9] L. Fan, L. Xiong, and V. Sunderam. Differentially private multi-dimensional time series release for traffic monitoring. In *27th Annual IFIP WG 11.3 Conference, DBSec*, pages 33–48, 2013.

[10] TAPASCologne Project. Accessed in June, 2013.

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

[12] 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.

[13] 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.

[14] 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.

[15] 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.