IJREAT International Journal of Research in Engineering & Advanced Technology, Volume 1, Issue 1, March, 2013
ISSN: 2320 - 8791
www.ijreat.org

Vehicle-Based Traffic Monitoring Via Virtual Trip Lines

Archana.V[1],Mr.Navin K [2]

[1]Department of Information Technology, SRM University, Chennai, Tamil Nadu, India

[2]Assistant professor, Department of Information Technology, SRM University, Chennai, Tamil Nadu, India

Abstract
Traffic monitoring using probe vehicles with GPS receivers promises significant improvements in cost, coverage, and accuracy over dedicated infrastructure systems. Current approaches, however, raise privacy concerns because they require participants to reveal their positions to an external traffic monitoring server. To address this challenge, I describe a system based on virtual trip lines and an associated cloaking technique, followed by another system design in which we relax the privacy requirements to maximize the accuracy of real-time traffic estimation. I introduce virtual trip lines which are geographic markers that indicate where vehicles should provide speed updates. These markers are placed to avoid specific privacy sensitive locations. They also allow aggregating and cloaking several location updates based on trip line identifiers, without knowing the actual geographic locations of these trip lines. Thus, they facilitate the design of a distributed architecture, in which no single entity has a complete knowledge of probe identities and fine-grained location information. I have implemented the system with GPS smart phone clients and conducted a controlled experiment with 100 phone-equipped drivers circling a highway segment, which was later extended into a year-long public deployment.

Index Terms-Algorithms, design, experimentation, security, privacy, GPS, traffic, data integrity.

1. Introduction

Personal navigation services in vehicles enable theeffective delivery and presentation of high-resolution only be achieved when sufficient users participate. While it remains possible to leverage existing telematics platforms or navigation systems, these platforms are not openly programmable and thus hard to retrofit for this purpose. The collected data for the system should be as accurate as possible. The accuracy of data could be degraded by GPS positioning inaccuracy and bogus data injected by malicious users.

To address these challenges, I propose a novel trafficMonitoring system design based on the concept of virtual trip lines (VTLs) and experimentally evaluates its feasibility. Virtual trip lines are geographic markers stored in the mobile phone client, which trigger a position and speed update when a probe vehicle trajectory intersects a trip line. Through privacy-aware placement of these trip lines, clientsneed not rely on a trustworthy server. The system is designed for GPS-enabled cell phones to enable rapid software deployment to a large and increasing number ofProgrammable smart phones. The basic aim of the project falls under Traffic Monitoring System based on location and speed updates. Location Updates based on Time was the main feature which was concentrated in the existing system. I propose a new technique by which monitoring the user updates based on Location Updates. The location updates are done based on 2 processes:Location Updates based on Trip Line Id, Speed Updates.

2. Proposed System

To overcome these existing challenges, proposed in traffic monitoring system builds on the novel concept of virtual trip lines. A virtual trip line is a line segment in geographic space that, when crossed, triggers a client's location update to the traffic monitoring server.

2.1 Virtual Trip Lines:

The basic aim of the project falls under Traffic Monitoring System based on location and speed updates. Location Updates based on Time was the main feature which was concentrated in the existing system. The proposed a new technique by which monitor the user updates based on Location Updates.Whenever the user starts to update the location based on GPS facility in the mobile, then through this 3 stage level of authentication processes. First, the user requests an authentication id to the VTL Generator via a Proxy Server. The VTL Generator provides the user a key for the user as an initial authentication process. The key is then forwarded by the user in return to the VTL Generator which in turn forwards a One Time Password for the user's mobile phone to validate the user. The VTL Generator stores the encrypted key (one time password) and when the user updates the information, the server uses the key to decrypt the content. All the updates from the user are processed via a Proxy Server such as all information must not be stored under a single party. An authentication level validation will be done at the Proxy End also.

Additionally, I have added Google web app engine to my web applications so that we need not to wait for routers to get connected. Through this our web applications will run on Googleinfrastructure. App Engine applications are easy to build, easy to maintain, and easy to scale as your traffic and data storage needs grow. With App Engine, there are no servers to maintain: just upload the application, and it's ready to serve the users.

3. System Architecture:

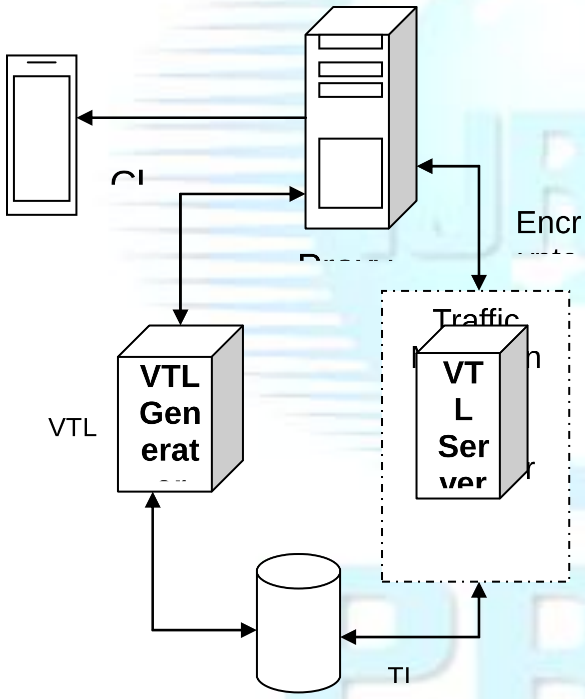

Fig1: Virtual trip line: Privacy-preserving traffic monitoring system architecture.

3.1 Achieving Authenticated but Anonymous Data Collection:

In order to achieve the anonymization of measurementuploads from clients while authenticating the sender of the measurements, I split the actions of authentication and data processing into two different entities, which it call the ID proxy server and the traffic monitoring server. By separately encrypting the identification information and the sensing measurements (i.e., trip line ID, speed, and direction) with different keys, it prevent each entity from observing both the identification and the sensing measurements. Fig. 1 shows the resulting system architecture. It includes four key entities: probe vehicles with the cell phone handsets, an ID proxy server, a traffic monitoring service provider, and a VTL generator. Each probe vehicle carries a GPS-enabled mobile handset that executes the client application. This application is responsible for the following functions: downloading and caching trip lines from the VTL server, detecting trip line traversal, and sending measurements to the service provider. To determine trip linetraversals, probe vehicles check if the line between the current GPS position and the previous GPS position intersects with any of the trip lines in its cache. Upon traversal, handsets create a VTL measurement including trip line ID, speed readings, time stamps, and the direction of traversal, and encrypt it with the VTL server's public key.
Handsets then transmit this measurement to the ID proxyserver over an encrypted and authenticated communication link setup for each handset separately. The handset and the ID proxy server share an authentication key in advance. The ID proxy server's responsibility is to first authenticate each client to prevent unauthorized measurements and then forward anonymized measurements to the VTL server. Since the VTL measurement is encrypted with the VTL server's key, the ID proxy server cannot access the VTL measurement content. It has knowledge of which phone transmitted a VTL measurement, but no knowledge of the phone's position. The ID proxy server strips off the identifying information and forwards the anonymous VTL measurement to the VTL server over another secure communication link.

The VTL server aggregates measurements from a largenumber of probe vehicles and uses them for estimating traffic conditions. The VTL generator determines the position of trip lines, stores them in a database, and distributes trip lines to probe vehicles when any download request from probe vehicles is received. Similar to the ID proxy server, each handset and the VTL generator share an authentication key in advance. The VTL generator first authenticates each download requester to prevent unauthorized requests and can encrypt trip lines with a key agreed upon between the requester and the VTL generator.1 Both the download request message and the response messageare integrity protected by a message authentication code.

A virtual trip line is a line segment in geographic space that, when crossed, triggers a client's location update to the trafficmonitoring server. More specifically, it is defined by

[vtlid, x1, y1, x2, y2, d],

Where vtlid is the virtual trip line ID, x1, y1, x2, and y2 arethe (x,y) coordinates of two line endpoints, and d is a default direction vector (e.g., N-S or E-W). The default direction vector encodes the valid direction in which the virtual trip line can be crossed. This directionally specific attribute can be used to reject location updates from vehicles crossing VTLs in the opposite direction, which can occur due to GPS errors and dense road networks. Also, in case that a single VTL covers both directions on highways if it is long enough (to cover both northbound and southbound, or westbound and eastbound), the clientsdetect the direction from two successive coordinates andsimply code the direction into 0 or 1 based on default direction vector. When a vehicle traverses the trip line, its measurementupdate includes the time, trip line ID, speed, and the direction of crossing. The trip lines are pregenerated, downloaded, and stored in clients. To check any crossings, I set the sampling period of a single-chip GPS/A-GPS module in each smartphone and retrieve the position readings. Since our setup did not provide speed information, can calculate the mean speed using two successive location readings (in our implementation, every 3 seconds).

The client software registers the task for checking the traversal of trip lines as an event handler for GPS module location updates, which is automatically invoked whenever a new position reading becomes available. As an example of required storage and bandwidth consumption, consider the San Francisco Bay Area, the total road network of which contains about 20,000 road segments, according to the Digital Line Graph 1:24K scale maps of the San Francisco Bay Area Regional Database managed by USGS. Assuming that the system on average places one trip line per segment, this results in 166 KB of storage.

Virtual trip lines control disclosure of location updates by sampling in space rather than sampling in time, since clients generate updates at predefined geographic locations (compared to sending updates at periodic time intervals). The rationale for this approach is that at specific locations, traffic information is more valuable and certain locations are more privacy sensitive than others. Through carefulplacement of trip lines, the system can thus better manage data quality and privacy than through a uniform sampling interval. In addition, the ability to store trip lines on the clients can reduce the dependency on trustworthy infrastructure for coordination.

3.2 Virtual Trip Line Measurements:

Noisy GPS readings can be filtered either on the client side or the server side. Server side processing can allow for a computationally expensive algorithm to filter out noisy GPS readings, for example, using map-matching algorithms. However, it requires clients to send detailed traces to a server, which incurs increased network bandwidth consumption and privacy concerns. Instead, I address filtering on the client, with the specific goals of subsampling GPS readings to reduce the frequency of trip line measurement computations (i.e., checking whether the line between two GPS readings intersects with any trip lines), and removing the need of any client side or server side map matching algorithm, which is a computationally expensive algorithm for resource constrained devices.

I have observed that GPS position error can createfalse VTL crossings and inaccurate VTL velocity measurementsin the following cases:

1) GPS position error. When a vehicle stops near a trip line, error in the GPS position can create successive position measurements with a zigzag pattern over the VTL, which can lead to multiple false trip line crossings. These crossings can be eliminated by requiring a minimum distance between successiveGPS readings.
2) Intermittent GPS. When the time interval betweentwo GPS positions becomes large (e.g., due to lost GPS signal), the inferred trajectory connecting these two location measurements no longer describes the actual movement of a vehicle. To eliminate false trip line crossings by this type of unrealistic trajectory, an upper bound of time gap between successive GPS samples is required.
3) Infeasible speed. In areas prone to high GPS positerror (e.g., urban areas with high-rise buildings), the speed computed from a finite difference approximation of the successive positions (required by the GPS receiver in our implementation) becomes infeasible.

Algorithm1 below describes in detail that implementationof a light-weight client filtering algorithm to treat thecommon situations above. The algorithm proceeds asfollows:First, if the GPS sample l is the first update, it issimply saved to CurrLocationFiltered (lines 4-7). Without a previous update, it cannot compute the speed or headingof the current update or confirm it as valid. Assuming aprevious update exists, the validity of the next update canbe determined based on the computed speed and thetemporal/spatial gap from previously filtered GPS readingcalled PrevLocationFiltered. I consider the current locationinvalid if it is updated long after the previous update(lines 10-14), if it has not traveled a minimum distance(lines 16-18, e.g., stopped at the traffic signal), or if has aspeed glitch (lines 19-30).

Algorithm 1: Tripline Crossing Detection Algorithm

```
1: 0=thresholdToSwitchBadToGood
2: T = subsampling interval
3: for all GPS sample l do
4: if PrevLocationFiltered is null then
5: CurrLocationFiltered = LastGoodRefPoint = l;
6: LastLocationUpdateTimestamp = l:t; goto
TripLineChecking;
```

```
7: end if
8: TimeGap = l:t - LastLocationUpdateTimestamp;
9: LastLocationUpdateTimestamp = l:t;
10: if TimeGap is too large then
11: LastGoodRefPoint = l; LastBadRefPoint = null; n = 0;
12: CurrLocationFiltered = l; PrevLocationFiltered =null;
13: gotoTripLineChecking;
14: end if
15: Calculate speed against LastGoodRefPoint;
16: if a vehicle has not moved far enough then
17: LastBadRefPoint = null; n = 0;CurrLocationFiltered = null;
18: gotoTripLineChecking;
19: else if speed glitch is true then
20: Re-calculate speed against LastBadRefPoint;
21: if speed glitch is false then
22: if þþn is greater than _ then
23: n = 0; LastBadRefPoint = null; LastGoodRefPoint = l;
24: filteredLoc = SmoothingFilter(LastBadRefPoint, l);
25: CurrLocationFiltered = checkReportingInterval(filteredLoc, T);
26: end if
27: gotoTripLineChecking;
28: end if
29: LastBadRefPoint = l; gotoTripLineChecking;
30: end if
31: n = 0; filteredLoc =SmoothingFilter(LastGoodRefPoint, l);
32: LastBadRefPoint = null; LastGoodRefPoint = l;
33: CurrLocationFiltered =checkReportingInterval(filteredLoc, T);
34: // TripLineChecking
35: if both CurrLocationFiltered and PrevLocationFiltered not null then
36:traj=SetTrajectory(PrevLocationFiltered,CurrLocationFiltered);
37: for all tripline j in each tile(i) do
38: if tile(i).status is valid then
39: triplineCrossed = CheckCrossing(tripline j,traj);
40: if triplineCrossed is true then
41: Compute speed and heading with traj fortriplineMeasurement;
42: end if
43: end if
44: end for
45: end if
46: if CurrLocationFiltered is not null then
47: PrevLocationFiltered = CurrLocationFiltered;
48: end if
49: end for
```

Additionally, it maintains two reference points, Last GoodRefPoint and LastBadRefPoint. If a series of locations have speed glitches against LastGoodRefPoint, but do not have speed glitches against LastBadRefPoint, I consider LastBadRefPoint and the most recent location in the series as valid (lines 22-26). Next, the location update after the validity check is injected to a smoothing filter (called SmoothingFilter in algorithm 1), which is implemented by an exponentially weighted moving average low-pass filter (lines 24, 31). The smoothing filter produces a smoothed version of speed profile by cutting off abrupt speed changes. The final step is used to reduce the computational overhead created by the frequent checking of virtual trip line crossings on the output of Algorithm 1. Instead of returning a location update at the maximal rate allowed by the GPS receiver, it return a location update only after every T seconds, which is encoded by the function checkReportingInterval (lines 25, 33). A larger T makes computation of trip line crossings more efficient, but if itbecomes too large, valid trip line crossings can be missed and false trip line crossings can be computed. The output returned from the Algorithm 1 is then used by the software routine that computes virtual trip line crossings from consecutive filtered GPS positions (lines 34-48). I check if any line defined by two end positions of each trip line intersects with a trajectory (built by two consecutive filtered GPS positions) in a two dimensional space. If crossed, the algorithm returns a trip line measurement including trip line ID, speed, heading, and time stamp information. All trip lines in downloaded tiles are tested, but limited to valid trip lines. Validity of trip lines can be subject to a combination of trip line's expiration time and user's privacy guidelines.

3.3 Sample outputs through VTL:

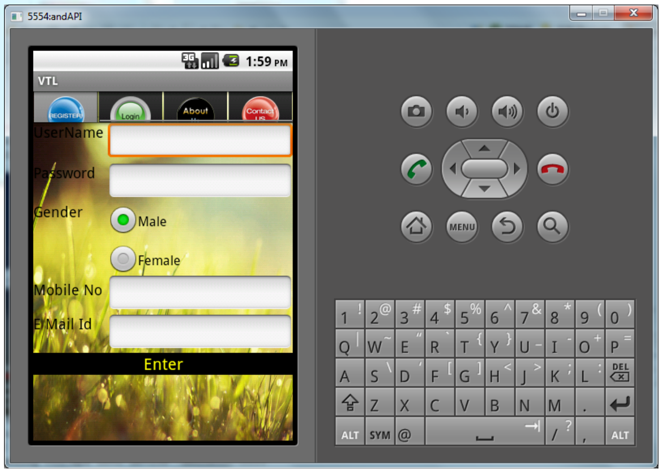

Fig 1: VTL USER REGISTRATION

The above figure is VTL app, through this app vehicle is registered (vehicle acts as a smart phone). Registration is done for authorized purpose.

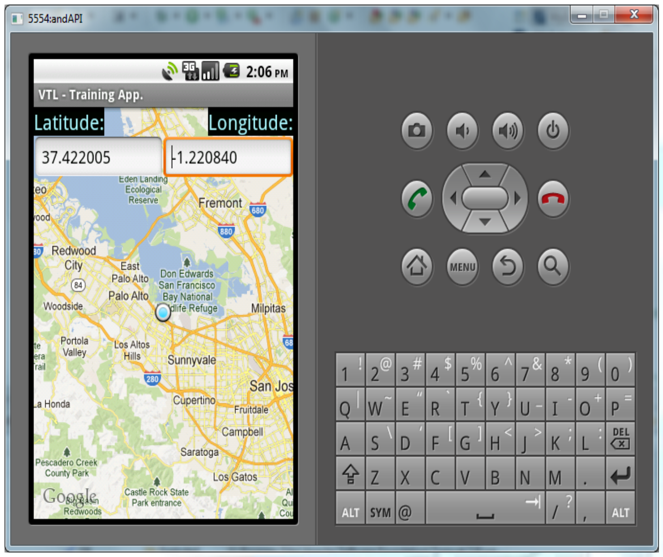

Fig 2: Location updation with latitude and longitude

This sample shows how VTL app finds the location through GPS via longitude and latitude.

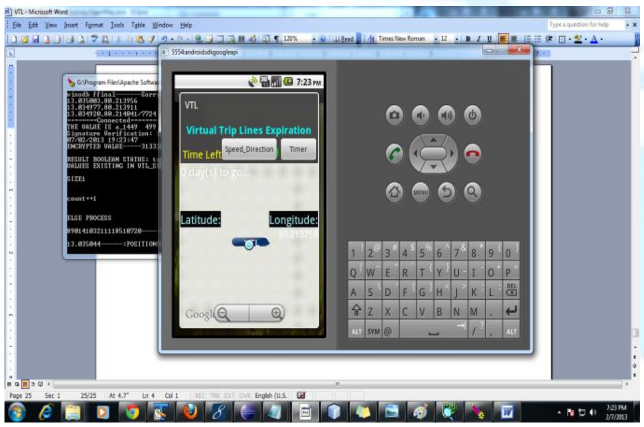

Fig 3: Trip line using VTL

This sample shows how trip line is used, each time it will verifies the client if the trip line is crossed via server. Hence, by this trip line estimates the distance and speed covered by the probe to the client which will be updated by VTL generater in the database. The client can also monitor the route of the probe through server in the VTL training App.

4. Results

This section first evaluates the performance of the trip line crossing detection algorithm presented in Section 3. Then, can analyze the travel time estimation accuracy and privacy preservation of our spatial sampling approaches using virtual trip lines. Spatial sampling approaches to be evaluated here include k-anonymous temporal cloaking and its privacy-relaxed version where itstrip off the requirement of guaranteed privacy via temporal cloaking. The former is the proposed scheme but the latter is still meaningful in that it is a baseline technique to be compared with the proposed scheme and a less complex system with an acceptable privacy protection in the real world.

5. Conclusion

This paper described traffic monitoring system implemented on GPS smartphone platform. The system uses theconcept of virtual trip lines to determine when phones reveal a location update to the traffic monitoring infrastructure. I demonstrated that the introduced scheme, Virtual Trip Lines, successfully addresses known weaknesses of probe vehicle-based traffic monitoring. First, the VTL paradigm achieves strong anonymity through k-anonymous temporal cloaking. Virtual trip lines allow the application of temporal cloaking techniques to ensure k-anonymity properties of the stored data set, without having access to the actual location records of phones. Second, they improve the accuracy of traffic monitoring. The temporal cloaking leads to less than 5 percent reduction in the accuracy of travel time estimates for k values less than 7 compared to periodic sampling techniques and a privacy-relaxed version achieves 5 percent travel time estimation error using only 1-2 percent penetration rate. Third, VTLs enable a light-weight client algorithm for collecting VTL measurements, and can achieve the VTL crossing detection between 50 percent to 98 percent in downtowns while suppressing false alarm less than 11 percent without map matching.

6. References

[1] http://traffic.berkeley.edu, 2011.
[2] http://www.fcc.gov/Bureaus/Wireless, 2011.
[3] http://www.paramics-online.com, 2011.
[4] http://www.privacyrights.org/ar/ChronDataBreaches.htm,2011.
[5] TeleNav, http://www.telenav.net, 2004.
[6] Inrix, http://www.inrix.com, 2006.
[7] Google Maps, http://www.google.com/mobile/maps, 2011.
[8] Waze, http://www.waze.com, 2009.

[9] A. Beresford and F. Stajano, "Mix Zones: User Privacy in Location- Aware Services," Proc. IEEE Ann. Conf. Pervasive Computing and Comm. Workshop, 2004.
[10] C.Claudel and A.A. Bayen, "Lax-Hopf Based Incorporation of Internal Boundary Conditions into Hamilton-Jacobi Equation. Part II: Computational Methods," IEEE Trans. Automatic Control, vol. 55, no. 5, pp. 1158-1174, May 2010.
[11] C. Claudel and A. Bayen,"Lax-hopf Based Incorporation of Internal Boundary Conditions into Hamilton-Jacobi Equation. Part I: Theory," IEEE Trans. Automatic Control, vol. 55, no. 5, pp. 1142-1157, May 2010.
[12] X. Dai, M. Ferman, and R. Roesser, "A Simulation Evaluation of a Real-Time Traffic Information System Using Probe Vehicles," Proc. IEEE Intelligent Transportation Systems (ITS), pp. 475-480, Oct. 2003.