Lightweight and Privacy-preserving Location Proofs for Intelligent Transportation Systems

Yifan Zhang¹, Chiu C. Tan², Fengyuan Xu³, Hao Han⁴, and Qun Li¹

¹College of William and Mary
²Temple University
³NEC Laboratories America
⁴Intelligent Automation, Inc

April 25, 2014
Lightweight and Privacy-preserving Location Proofs for Intelligent Transportation Systems

Yifan Zhang*, Chiu C. Tan†, Fengyuan Xu‡, Hao Han§ and Qun Li*

*College of William and Mary, †Temple University, ‡NEC Laboratories America, §Intelligent Automation, Inc

Abstract—Transportation systems relying on vehicles to collect data for services such as road condition monitoring are vulnerable to malicious vehicles injecting large amounts of fake data. A particularly serious type of attack is that attackers report fake information about numerous places without actually presenting there. In this paper, we present VProof, a vehicle location proof scheme that enables a vehicle to prove its location claims match its historical locations. With VProof, vehicles construct their location proofs by simply extracting relevant contents from the packets received from roadside units. Our scheme is lightweight, since there is no communication required for a prover to obtain a location proof. Our scheme also well preserves users’ privacy, as we do not put any information that can be related to a user’s ID in a location proof. We have implemented a prototype VProof system and evaluated it with extensive real-world experiments. Our evaluation results show that VProof is able to reliably prove vehicle’s locations without leaking any user privacy.

I. INTRODUCTION

Ubiquitous computing technologies, such as sensing and wireless communication, have enabled Intelligent Transportation Systems (ITS), with which people can achieve safer and more efficient everyday transportation. In today’s Intelligent Transportation Systems (ITS), a popular category of applications is that vehicles report information about the transportation system elements (e.g., drivers and road conditions) to the ITS system for services like real time traffic control and roads maintenance [1], [2]. Successes of recent research projects on vehicle-based data sensing and collection [3]–[5] have bolstered such ITS data collection applications. However, before accepting data about a location reported by a vehicle, ITS operators need to verify if the vehicle visited the location at the time indicated in the reported data. Failing to do so will allow malicious users to launch an attack to the ITS system by reporting fake information about places where he did not actually visit. The damages of the attack are particularly serious, since the attacker can report fake information about numerous places by just clicking mouse at home.

To verify whether a vehicle’s location claims match its actual historical locations, ITS operators need a location proof scheme featuring the following properties. First, the location proof should be lightweight. This property is extremely important in vehicular environments, since location proof issuers may need to issue location proofs to tens or hundreds of vehicles on a busy road at the same time. Second, the location proof needs to well preserve users’ location privacy. Concerns about users’ (we will use “user” and “vehicle” alternately) location privacy have become major considerations when deploying location-related services in ITS systems [6]. Car owners can simply opt out of providing any data if their privacy is threatened. Third, the location proof scheme needs to be able to generate fine-grained location proofs, because the locations reported in the user collected information (e.g., there is a pothole somewhere on the road) have fine granularity.

To detect malicious users who report bogus data, the conventional solution is to assign each vehicle with some cryptographic keys. Each vehicle will sign each piece of data with its secret key before uploading to the ITS system. The idea is that by having a means to track back users, the amount of bogus data will be reduced, since malicious users do not want to be caught. Similar schemes have also been proposed to protect the privacy of honest users to encourage participation [7]–[9]. Nonetheless, these solutions all require deploying a large-scale PKI (i.e., Public-key Infrastructure) scheme to associate specific keys with individual vehicles. The use of large-scale PKI scheme usually requires complicated management of digital certificates, as well as complex coordination between the administrative authority and participating users. While the PKI-based approach may be possible in theory, for instance, a PKI administered by a local DMV, it is less clear if this will be done in practice, especially in a large country. We can point to the difficulties in getting the different states in the U.S. to standardize on a common driver’s license as evidence of the impracticality of a widely deployed PKI solution. Thus, a solution that does not rely on such large scale infrastructure to provide privacy protections is needed.

The idea of location proofs has been considered by other types of applications before. The general approach is to let certain authorized entities with fixed geolocations perform as location proof issuers. The location proof issuers issue location proofs, which are unique and unforgeable, to nearby location provers, who need to prove their historical locations to a location verifier later. A location prover is believed to be in the vicinity of a proof issuer at a certain time if the prover possesses valid location proofs [10]–[12]. We cannot apply the same approach towards vehicular environments because the location proof granularity achieved by the existing location proof solutions is coarse: they can only prove that at a certain time a user was within the communication radius of a proof issuer (e.g., the communication range of a wireless AP), but not at a finer granularity. This allows a malicious user to statically collect the location proofs issued by a proof issuer and report fake information about places where he never visited but are within the proof issuer’s communication radius. Meanwhile, the existing solutions require a proof issuer to
perform multiple rounds of interactions with a prover to issue a location proof specifically to the prover, which is not scalable to vehicular environments, where lots of vehicles may be requesting location proofs at the same time.

We propose VProof, a vehicular location proof solution designed without using PKI schemes. In our solution, RSUs (roadside units), which are necessary components of ITS systems [1], [2], continuously broadcast packets that are specifically for the location proof functionalities. We name these packets as VPackets. We observe that the Received Signal Strength (RSS) of the series of VPackets received by a vehicle when it passes an RSU exhibit similar patterns over time if the transmission power of the VPackets is fixed (we will elaborate on this observation later in Section IV). In VProof, if a user claims his data were collected at a certain time, we require him to show that he has seen the correct RSS pattern of the VPackets sent by a nearby RSU. With our scheme, the correct RSS pattern can only be obtained if the user indeed drove past the claimed place. Thus, we achieve fine-grained location proofs and prevent malicious users from statically collecting location proofs. Our scheme has the following unique properties compared to the related work. First, our scheme is lightweight. In VProof, RSUs (proof issuers) issue location proofs by simply broadcasting packets, and vehicles (provers) construct location proofs by extracting relevant contents from the RSU packets they received. This makes our scheme scalable in busy vehicular environments where lots of vehicles may be requesting location proofs at the same time. Second, the VProof scheme is privacy-preserving, because it puts no information that can be related to a user’s identification. Third, the VProof scheme can provide fine-grained and unforgeable location proofs.

Although using RSS patterns to design a location proof solution for vehicular environment seems promising at first glance, it brings the following three major challenges.

First, the open propagation of RF signals allows everyone to see the VPackets’ RSS patterns, which makes it difficult to maintain the unforgeability of location proofs. We address this challenge by randomly changing the transmission power of the VPackets. Thus, the RSS of VPackets exhibit no pattern from the vehicles’ perspectives. When the location proofs are submitted to ITS operators for verification, our solution allows only the operators to restore the inherent RSS patterns, which are VPackets RSS patterns if those VPackets were sent with a fixed power.

Second, it is widely known that RSS measurements are sensitive to many factors, such as hardware differences, moving obstacles and even different weather conditions. Thus, it is challenging to design a reliable location proof solution based on RSS measurements. To address this challenge, we design an RSS pattern comparison algorithm that can effectively deal with RSS measurement dynamics.

Third, in a busy vehicular wireless environment, where there are a large number of ongoing wireless communications, it is unlikely that a vehicle can receive or process all the RSU VPackets, based on which the vehicle constructs the location proofs. This requires our location proof scheme to be resilient to packet losses. Our RSS pattern comparison algorithm is also designed to fulfill this requirement.

Our contributions can be summarized as follows.

- We propose VProof, a lightweight and privacy-preserving location proof solution that verifies if a vehicle’s location claims match its historical locations. To the best of our knowledge, VProof is the first location proof solution designed for vehicular environments, and is also the first that does not rely on PKI to achieve the functionalities of location proofs.

- We design an efficient algorithm that can reliably determine if two series of packet RSS are similar given potential packet losses and inaccurate RSS measurements.

- We implement a prototype of the VProof system, and evaluate the prototype system with extensive experiments performed on real road conditions.

The rest of the paper is organized as follows. In Section II, we introduce the related work of VProof. In Section III, we introduce the background of the proposed scheme. In Section IV, we motivate our paper by first describing the limitations of applying the existing location proof schemes in vehicular environments, and then presenting two key observations that led us to the VProof scheme. An overview and the details of the solution are provided in Section V and Section VI respectively. In Section VIII, we discuss several important issues related to the proposed scheme. We present our prototype system implementation and real-world experiment results in Section IX, and conclude the paper in Section X.

II. RELATED WORK

A. Existing location proof solutions

Location proofs have been suggested as a way for users to prove their past locations in location based services [10]–[12]. A typical proof construction requires a user to perform several rounds of interaction with the proof issuer to derive a location proof, which is later used by the proof verifier to verify the user’s location. Later work by [12] improves on this process by preventing the proof issuer from learning the user’s location. We will discuss the limitations of applying the existing location proof solutions in vehicular environments in details later in Section IV.

B. Location privacy in vehicular networks

To prevent users from submitting fake information in vehicular networks, the existing solutions typically use anonymous authentication. Work by Xi et al [7] proposes a symmetric random key-set scheme, where each vehicles possesses a set of symmetric keys randomly chosen from a key pool, to authenticate vehicles into vehicular networks. Scheme proposed by Calandriello et al. [8] addresses the problem of anonymous message authentication using asymmetric keys and group signature in vehicular networks. ECPP [9] achieves anonymous message authentication under the help of its on-the-fly short-time anonymous keys between vehicles and RSUs.
The reasoning behind the existing approaches is that if car owners are aware that bogus data can be traced back to them, they will not intentionally upload incorrect data. However, car owners can simply opt out of providing any data if their privacy is threatened. Our work provides a technique to enable a user to prove his historical locations are in accordance with the data he reports. This allows ITS operators to prevent the attacks where users report information about places they did not visit. Our approach achieves strong user privacy protection, since we do not place any information regarding the user’s identity nor link any cryptographic keys with the user, and thus there is no way users reporting data to the ITS systems can be traced. We argue that our approach is more suitable in situations where user privacy outweighs other concerns.

Concerns about users’ location privacy have become major considerations when deploying location-related services in ITS systems [6]. Existing solutions use group navigation and dynamic pseudonyms [13], mix-zones and vehicular mix-networks [14] or group communication [15] to defend users’ location privacy. VProof also well protects users’ location privacy since we do not place any information regarding the user’s identity nor link any cryptographic keys with the user.

C. Changing packet transmission power and measuring packet received signal strength

Changing wireless packet transmission power and measuring the received signal strength (RSS) have recently been used in localizations [16], [17] and rogue vehicular AP detection [18], [19]. All these existing works rely on accurate RSS measurements to determine the distance between two communicating wireless nodes. In our work, we do not have dependence on accurate RSS measurements for the following two reasons. First, changing packet transmission power and measuring RSS in our solution are not to determine the distance between two wireless nodes. Instead, we use them to hide the inherent RSS patterns from users. Thus, our work does not need as accurate RSS readings as the existing works do. Second, our RSS pattern similarity comparison algorithm is specifically designed to cope with inaccurate RSS measurements and packet losses.

III. BACKGROUND

A. ITS data collection applications

In ITS data collection applications, vehicles sense and collect data about surrounding elements by their add-on vehicle sensors. The sensed data, along with the corresponding metadata that tell when and where the sensed data were generated, are uploaded to backend servers located at the infrastructure side, either immediately after they are generated or sometime later when appropriate [20]–[22]. The ITS systems operate vehicular networks that enable wireless communication between vehicles and the infrastructure via roadside units (RSUs). We assume that RSUs are controlled by the ITS operator, and all vehicles are equipped with GPS receivers.

B. Threat model

We consider the threat that malicious users target at disrupting ITS systems by reporting fake information about numerous places where they did not actually visit. If there is no scheme to allow ITS operators to verify whether the reporting users have actually visited the places indicated in the reported data, a malicious user can easily generate and report bogus data about lots of places without actually visiting those places. The amount of the bogus data could overwhelm that of the honest data. Existing works for filtering abnormal data in vehicular networks [23], [24] do not work in this case, because they hold an assumption that the amount of abnormal data should not be more than that of normal data. Meanwhile, as we discussed previously, we prefer not to use PKI to solve the problem. Thus, we ask the question: without using PKI systems, can we provide a scheme to let ITS operators verify if a user’s historical locations are in accordance with the data he submits so that they can prevent the threat we just described?

We make the following assumptions about malicious users. First, malicious users have the same equipments as honest users, and have certain knowledge of the information about the RSUs in the ITS system, such as the ESSIDs and GPS locations of the RSUs. But they do not know any secret keys shared between the RSUs and the ITS system. Second, malicious users cannot control any infrastructure units or replicate them in exactly the same way. For example, malicious users cannot replicate a certain legitimate RSU by placing the same hardware on the same roadside pole. This prevents malicious users from obtaining similar profiling data as the authority does.

IV. MOTIVATION

A. Limitations of the current location proof solutions

Our work is motivated by the following three major limitations of the existing location proof solutions [10]–[12] to be applied in vehicular environments.

First, the existing solutions can only be used to prove at certain time a user was within the communication range of the proof issuer (an RSU in our case) but not at a finer granularity. This critical drawback allows a malicious user to sit tight at a certain location within the RF range of an RSU, and legitimately report bogus data about other locations within the same range. The resulting damages are even greater in systems adopting long range wireless communication techniques. In our solution, if a user claims he was at a certain place when he collected some data, we require him to show that he has seen the correct RSS pattern of the packets sent by a nearby RSU that he must drive by the claimed place to obtain.

Second, the construction of a location proof in the existing solutions requires several rounds of interactions between the proof issuer and the user, which makes them impractical.

---

1As discussed later in Section VIII, we do not consider the threat that a malicious user physically presents at a place and report fake information about it, as we deem this threat has much less impacts than the one we are considering.
constant across the RSU’s RF range over time. Figure 2 shows

plots that are transmitted by the same RSU using two different

packets with two different transmission powers and different moving obstacles (i.e., cars and trucks) on the

shapes, which are due to factors like different temperatures

there are slight differences in RSS amplitudes and pattern

in the afternoon of another day. We can see that, although

on the road, and series B were collected during peak hours

collected in the morning of one day when there were less cars

of packets collected in the experiment, where series A were

different times. Figure 1 shows the RSS patterns of two series

car past the roadside wireless node and collected its packets at

power, at the roadside of a downtown environment. We drove a

experiments, we deployed an wireless node, which broadcast

passes an RSU, which is continuously broadcasting packets

fixed transmission power

B. The observation on RSS patterns of RSU packets with a

fixed transmission power

Through real-world experiments, we observe that the RSS

of a series of RSU packets received by a vehicle when it

passes an RSU, which is continuously broadcasting packets

with a fixed power, exhibit similar patterns over time. In the

experiments, we deployed an wireless node, which broadcast

packets at a rate of 100 packets/s with the full transmission

power, at the roadside of a downtown environment. We drove a
car past the roadside wireless node and collected its packets at
different times. Figure 1 shows the RSS patterns of two series

of packets collected in the experiment, where series A were

collected in the morning of one day when there were less cars

on the road, and series B were collected during peak hours in
the afternoon of another day. We can see that, although

there are slight differences in RSS amplitudes and pattern

shapes, which are due to factors like different temperatures and
different moving obstacles (i.e., cars and trucks) on the
road, the two series of packets do exhibit similar RSS patterns.

C. The observation on relatively constant RSS difference of
packets with two different transmission powers

We also observe that the RSS difference between two pack-
ets that are transmitted by the same RSU using two different

powers and are received at the same location is roughly a
constant across the RSU’s RF range over time. Figure 2 shows

the RSS of two groups of RSU packet series. Packet series A
and B, which were transmitted using the full power, are the
same as in Figure 1. Packet series A’ and B’, which were
transmitted using half of the full power, were obtained at
the same times as A and B respectively. We align the two
RSU packet series obtained in the same experiment based on
the distance between each packet’s reception location and a
fixed starting point. We can observe that the RSS difference
under two different transmission powers at the same location
is roughly the same (around 8 dbm). We will quantify how
stable this RSS difference is later in Section IX.

V. Solution Overview

With the above two observations, we design a location
proof scheme using RSS of RSU packets, which are pub-
iclty observable. Generally speaking, we let RSUs continu-
ously broadcast packets that are specifically for the location
proof functionalities (named as “VPackets”). Each VPacket
is broadcast using a randomly chosen transmission power.
Since the transmission power is randomly selected, the RSS
of the VPackets received by vehicles exhibit no pattern. Each
VPacket incorporates some encrypted information including
the transmission power of the packet. Vehicles collect the
VPackets, construct location proofs based on information in the
VPackets and their own GPS readings, and submit the
location proofs to the ITS system for verification. Using the
information in the location proofs, specifically the transmission
power of each VPacket, the ITS operators can restore the
inherent VPacket RSS patterns, which are the RSS patterns
if the VPackets were transmitted using the full power. The
location proofs are deemed as valid only if they can be used to
correctly restore the inherent RSS patterns of RSUs. Since
the transmission power of each VPacket is only known by the
ITS operators, we enforce the unforgeability of the location
proofs VProof constructs.

The general operation flow of VProof is shown in Figure 3. The
pre-application operations (i.e., the operations performed
before the data collection applications are deployed) include:

• Step PRE-1: the ITS operator constructs a database that
stares different RSUs’ VPacket RSS traces.

• Step PRE-2: for each RSU, the operator assigns it a
unique secret and configures it with a pre-calculated
VPacket broadcast rate. The RSU secret is used by

in vehicular environments as the contact durations between
vehicles and RSUs may be very short. By contrast, with
VProof, no interaction between the proof issuer and the user
is required.

Third, the existing solutions rely on PKI systems that we
are trying to avoid, since the deployment of a large scale PKI
scheme for ITS is unlikely to be realized in the near future.

Through real-world measurements, we make the following
two key observations that led us to design our location proof
solution by utilizing RSU packets RSS patterns.

B. The observation on RSS patterns of RSU packets with a
fixed transmission power

With the above two observations, we design a location
proof scheme using RSS of RSU packets, which are pub-
iclty observable. Generally speaking, we let RSUs continu-
ously broadcast packets that are specifically for the location
proof functionalities (named as “VPackets”). Each VPacket
is broadcast using a randomly chosen transmission power.
Since the transmission power is randomly selected, the RSS
of the VPackets received by vehicles exhibit no pattern. Each
VPacket incorporates some encrypted information including
the transmission power of the packet. Vehicles collect the
VPackets, construct location proofs based on information in the
VPackets and their own GPS readings, and submit the
location proofs to the ITS system for verification. Using the
information in the location proofs, specifically the transmission
power of each VPacket, the ITS operators can restore the
inherent VPacket RSS patterns, which are the RSS patterns
if the VPackets were transmitted using the full power. The
location proofs are deemed as valid only if they can be used to
correctly restore the inherent RSS patterns of RSUs. Since
the transmission power of each VPacket is only known by the
ITS operators, we enforce the unforgeability of the location
proofs VProof constructs.

The general operation flow of VProof is shown in Figure 3. The
pre-application operations (i.e., the operations performed
before the data collection applications are deployed) include:

• Step PRE-1: the ITS operator constructs a database that
stares different RSUs’ VPacket RSS traces.

• Step PRE-2: for each RSU, the operator assigns it a
unique secret and configures it with a pre-calculated
VPacket broadcast rate. The RSU secret is used by
both the RSU and the system operator to generate/verify location proofs. The VPacket broadcast rate theoretically ensures a vehicle can receive a desired number of VPackets during each coherence time period.

The during-application operations (i.e., the operations performed with the data collection applications) include:

- Step DUR-1 (proof issuing): RSUs broadcast each VPacket at the configured rate using a randomly selected transmission power.
- Step DUR-2 (proof construction): upon receiving the VPackets, vehicles construct location proofs by extracting relevant contents from the VPackets.
- Step DUR-3 (proof submission): when vehicles upload newly collected data to the backend server, they also upload all the location proofs constructed since the last data submission.
- Step DUR-4 (proof verification): the ITS operator verifies the location proofs according to the following sub-steps.
  
  First, the operator verifies if the location proofs are constructed using authentic VPackets. Then he constructs a user RSS series based on the location proofs, and preprocess the user RSS series to smooth out the unpredictable vehicle moving patterns when vehicles received the VPackets, such as stops due to red lights and slow driving due to traffic jams. Before feeding the user RSS series into the RSS similarity comparison algorithm, the operator restores the inherent RSS pattern (i.e., the RSS pattern if all the VPackets were transmitted using the full power). Finally, the operator determines if the location proofs are valid based on the pattern similarities between the restored user RSS series and the DB RSS traces.

We describe the details of each of the above steps in the following section.

VI. SOLUTION

A. PRE-1: VPacket RSS trace database construction

The RSS trace DB contains VPacket RSS traces of each RSU. In our design, there are \( N \) RSS traces associated with each possible vehicle trajectory around each RSU \( U_i \). Each RSS trace contains an RSS series of VPackets collected by driving a car past \( U_i \) on the trajectory. Figure 4 shows an example of the possible trajectories around a “T” shape crossing. During the profiling process, \( U_i \) broadcasts VPackets using the full transmission power. In the DB, assuming there are \( L \) transmission powers, \( U_i \) is also associated with average RSS differences between each non-full powers \( P_j (j = 1, \cdots, L - 1) \) and the full power \( P_f \): \( RSS_{U_i, P_j \rightarrow P_f} \). We will describe the experiment that obtains this average RSS difference in Section IX. The profiling process only needs to be done once.

B. PRE-2: RSU secrets and VPacket rate configuration

The system operator generates an RSU-specific secret \( s_{U_i} \) for each RSU \( U_i \), and configures it to \( U_i \). The secret \( s_{U_i} \) is for encrypting/decrypting the VPacket’s transmission power, and also for generating/verifying the VPacket authentication message (described later). For a vehicular wireless node moving in an outdoor environment, the RSS of its received packets vary a lot. To deal with RSS instability, we want users to receive \( n \) VPackets within a period of coherence time, which is the time duration over which the RSS is considered to be not varying. Then we can take the average of these \( n \) VPackets’ RSS as a data point in the RSS series. So the VPacket rate for an RSU \( U_i \) is calculated as \( \frac{n}{T_{coherence}} \), where \( T_{coherence} \) is the average coherence time of the road section around \( U_i \).

C. DUR-1: VPackets broadcast (by RSU)

VPacket transmission power selection. An RSU \( U_i \) uses Algorithm 1 to select the transmission power for each VPacket when it is generated. If the timer \( Timer_{pwr} \) is expired at the time when the algorithm is executed, the algorithm selects the transmission power randomly from the \( L \) power levels (line 2), with \( P_f \) being the full power and \( P_1, \cdots, P_{L-1} \) being the other \( L - 1 \) none-full power levels. Otherwise, the algorithm returns the power level given by the last random selection (line 3).

---

![Fig. 3. Operation flow of VProof.](image-url)

![Fig. 4. The six possible trajectories of a “T” shape crossing.](image-url)

---

### TABLE I

<table>
<thead>
<tr>
<th>Notation</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>( U_i, U_j )</td>
<td>RSU IDs</td>
</tr>
<tr>
<td>( N )</td>
<td>number of traces assoc. with each trajectory in the DB</td>
</tr>
<tr>
<td>( L )</td>
<td>number of VPacket transmission powers</td>
</tr>
<tr>
<td>( P_j )</td>
<td>non-full transmission powers ( (j = 1, \cdots, L - 1) )</td>
</tr>
<tr>
<td>( P_f )</td>
<td>the full transmission power</td>
</tr>
<tr>
<td>( RSS_{U_i, P_j \rightarrow P_f} )</td>
<td>average RSS difference between VPackets transmitted under a non-full power ( P_j ) and the full power ( P_f )</td>
</tr>
<tr>
<td>( s_{U_i} )</td>
<td>secret for ( U_i )</td>
</tr>
</tbody>
</table>
To ensure V Packets are broadcast using the same power level during at least a period of coherence time \( T_{coherence} \), the duration of \( T_{pwr} \) is set randomly between \( T_1 \) and \( T_2 \) (line 3), both of which are greater than \( T_{coherence} \). After the transmission power \( p \) is determined, \( U_i \) encrypts \( p \) using a symmetric-key algorithm \( SE \), with the combination of \( s_U \) and time \( t \) (i.e., the time when the V Packet is generated) as the cryptographic key:

\[
C_p \leftarrow SE_{s_U,t}(p). \tag{1}
\]

**VPacket authentication message generation.** In order to verify if the location proofs are constructed based on authentic V Packets, an RSU \( U_i \) generates a V Packet authentication message (VAM) for each V Packet as

\[
VAM \leftarrow H(U_i, s_U, t, C_p), \tag{2}
\]

where \( H \) is a cryptographic hash function (e.g., MD5 and SHA-1) that hases \( U_i \) (the ID of the RSU), \( s_U \) (the secret of the RSU), \( t \) (the time when the V Packet is generated) and \( C_p \) (the ciphertext of the transmission power \( p \)) into a single piece of message.

Finally, the RSU \( U_i \) puts the VAM, \( C_p \) and \( t \) into the V Packet, and broadcasts the V Packet using power \( p \).

**D. DUR-2.3: Location proof construction and submission (by vehicles)**

Upon receiving a V Packet, a vehicle constructs a location proof (LP) as

\[
LP = \langle U_i, \ VAM, \ t, \ RSS, \ C_p, \ LOC \rangle, \tag{3}
\]

where (1) \( U_i \) is the ID of the V Packet’s originating RSU, (2) VAM is the V Packet authentication message, (3) \( t \) is the time when the V Packet is generated, (4) RSS is the received signal strength of the V Packet, (5) \( C_p \) is the ciphertext of the V Packet transmission power \( p \), and (6) LOC is the vehicle’s GPS location when the packet is received. The first five items are extracted from the V Packet, and the last one is obtained from the vehicle’s onboard GPS device.

When the user uploads newly collected data to the ITS system, he also uploads all the location proofs constructed since the last data submission to the ITS system.

**E. DUR-4: Location proof verification (by ITS operators)**

The ITS operator divides the location proofs received from the same upload connection into batches such that each batch of location proofs share the same RSU ID. Then the operator verifies the location proofs batch by batch. A valid batch of location proofs with RSU ID \( U_i \) indicate the proof submitter has actually driven past \( U_i \) at the time indicated in the location proofs.

**VPacket authentication message verification.** Given a batch of \( m \) location proofs, the ITS operator verifies the VAM contained in each location proof as follows. He computes the message content by using formula (2) with the parameters \( U_i, t, C_p \) and \( s_U \), where \( U_i, t \) and \( C_p \) are extracted from the location proof, and \( s_U \) is kept by the operator. If the computed content is different from the VAM contained in the location proof, the VAM is deemed as invalid. An invalid VAM indicates that a least one parameter of \( U_i, t \) and \( C_p \) provided in the location proof has been tampered with. If there exists one location proof containing invalid VAM, the whole batch of location proofs are invalid. Note that a batch of location proofs with valid VAMs does not necessarily means the batch of location proofs are valid, because a malicious user can statically collect V Packets, obtain valid VAMs and present them in the location proofs.

**User RSS series construction.** Given a batch of \( m \) location proofs with valid VAMs, the operator constructs a user RSS series \( S_{user} \) according to Algorithm 2. In the algorithm, \( S_{raw} \) is the sequence of RSS in the \( m \) location proofs, \( T_{raw} \) is the corresponding sequence of V Packet generation times, and \( P_{raw} \) is the corresponding sequence of deciphered transmission power levels. The algorithm outputs the user RSS series \( S_{user} \), in which each data point is computed as the average RSS (line 7) of the V Packets that were transmitted under the same power and within a period of coherence time (line 6). The algorithm also outputs \( P_{S_{user}} \), a sequence of deciphered transmission power levels. Each data point of \( P_{S_{user}} \) corresponds to an RSS data point in \( S_{user} \) (line 8).

**User RSS series preprocessing.** The preprocessing of the user RSS series \( S_{user} \) has two goals. The first is to make \( S_{user} \) location-even. In real road situations, users may stop on the road for a while (due to red lights), or drive with a speed that is far less than the speed limit (due to congested traffic). In these cases, \( S_{user} \) will contain much more data points measured around some locations than from other locations. Our algorithm tunes \( S_{user} \) by removing the redundant points based on the parameters of \( LOC \) and \( t \) presented in the corresponding location proofs. The second goal of the preprocessing is to identify the user’s trajectory based on which the operator can select the corresponding RSS traces from the RSS DB. The trajectory identification is based on the GPS locations contained in the proofs.

**User RSS series pattern restoration.** The user RSS series \( S_{user} = v_1 \cdots v_n \) has no pattern since the V Packets were broadcast using random transmission powers. Therefore, before comparing the patterns of user-submitted RSS series and
that this information is associated with \( U \) where power). This is accomplished by adding pattern if all the VPackets were transmitted using the full needs to restore the profiled RSS series stored in the trace DB, the ITS operator with the user trajectory from the RSS trace DB, and derives between the user RSS series \( S \) constructing \( S \) noise floors. To address this issue, our RSS series comparison the same received packet, because they may have different series have similar patterns for the following three reasons. 1 valid if there are enough amount of matches.

the user RSS series \( S \) challenges, we designed a series have similar patterns for the following three reasons. 1 valid if there are enough amount of matches. 1

User RSS series validation. To validate the user RSS series \( S_{\text{user}} \), the ITS operator fetches the \( N \) RSS traces associated with the user trajectory from the RSS trace DB, and derives \( N \) DB RSS series \( S_{\text{DB},i} \) \( i \in \{1, N\} \) in the same way as constructing \( S_{\text{user}} \). The ITS operators compares the similarity between the user RSS series \( S_{\text{user}} \) and each of the \( N \) profiled RSS traces respectively. The user RSS series is deemed as valid if there are enough amount of matches.

As we mentioned earlier, it is difficult to decide if two RSS series have similar patterns for the following three reasons. First, different hardwares may have different readings on the same received packet, because they may have different noise floors. To address this issue, our RSS series comparison algorithm is designed to compare patterns of the quantized series, which are not impacted by amplitudes of the RSS readings. Second, RSS measurements are sensitive to many factors especially in an outdoor moving environment. Third, due to busy vehicular wireless environment, vehicles may not received all the VPackets based on which the location proofs are constructed. To address the second and the third challenges, we designed a dynamic time warping (DTW) [25], [26] based algorithm to compare two RSS series.

Our RSS series similarity comparison algorithm compares the user RSS series \( S_{\text{user}} \) with each of the \( N \) DB RSS series \( S_{\text{DB},i} \) according to the following steps.

The first step is to quantize both the RSS series \( S_{\text{user}} \) and

\[
S_{\text{DB},i}\text{ using a }K\text{-number alphabet. The goal of the RSS series quantization is to, as pointed out previously, remove the factors that can cause different amplitudes on RSS readings (e.g., hardware differences). The quantized value of each data point in a RSS series reflects the position of data point’s value within the value range of the RSS series. The quantization algorithm is given in algorithm 3. We use a simple example to illustrate how the quantization process works. Suppose there is an RSS series whose RSS values are in the range of \([1, 30]\), and we want to quantize them using a 3-number alphabet \(\{0, 1, 2\}\) (i.e., \(K = 3\)). Then our algorithm converts the data points in the RSS series with values from in the ranges of \([1, 10]\), \([11, 20]\) and \([21, 30]\) to numbers 0, 1 and 2 respectively.

The second step is to obtain the warped versions of the quantized RSS series. The goal of this step is to cope with inaccurate RSS measurements and potential missing data points due to losses of VPackets. In this step, both quantized \( S_{\text{user}} \) and quantized \( S_{\text{DB},i} \) are converted to their corresponding warped versions using a dynamic time warping (DTW) based algorithm. Here we use a detailed example to show how our DTW algorithm works and its benefits. Suppose \( M ="1222221100" \) and \( N ="1022110000" \) are two RSS sequences quantized with a 3-number alphabet \(\{0,1,2\}\). We can see that the two RSS sequences have the similar pattern (they both change follow the \(1 \to 2 \to 1 \to 0 \) pattern). However, the quantized values are not exactly aligned. For example, the second quantized value in the sequence \(N\) is 0 instead of 1 or 2, which can happen due to inaccurate RSS measurements. Meanwhile, we can see that the number of some quantized values in a sequence is less than that in another sequence (for instance, \(M\) has less 0 and \(N\) has less 2). This can happen when there are VPacket losses. We use the following formula to calculate the distance between \(M\) and \(N\) as \(\text{distance}(M, N) = \sum_{i=1}^{l} |m_i - n_i|\), where \(m_i\) and \(n_i\) are the values of the \(i\)-th bit in \(M\) and \(N\) respectively, and \(l\) is the length of both sequences. The distance between the original \(M\) and \(N\) is 6 (Figure 5 (a)). Our algorithm calculates a warping path between \(M\) and \(N\) by using dynamic time warping, which is basically a form of dynamic programming, and converts \(M\) and \(N\) to their warped versions \(M'\) and \(N'\).
based on the warping path (Figure 5 (b)). The distance of
the two warped series is 1 (Figure 5 (c)). Now we can see
that the DTW algorithm can well identify the similar pattern
of two misaligned sequences. Please note that although in the
example the two sequences have the same number of bits, it is
not necessary that the two original sequences should have the
same length. However, the two warped series will be equal-
length, because both of them are constructed based on the
same warping path.

In the final step, the algorithm calculates the similarity score
between the two RSS series, based on which a conclusion is
made. The distance of two misaligned sequences. Please note
that although in the example the two sequences have the same
number of bits, it is not necessary that the two original sequences
should have the same length. However, the two warped series
will be equal-length, because both of them are constructed based
on the same warping path.

In the final step, the algorithm calculates the similarity score
between the two RSS series, based on which a conclusion is
drawn. Suppose the length of the two equal-length quantized
warped RSS series \( S'_{user} \) and \( S'_{DB,i} \) is \( l \), the similarity score
between \( S'_{user} \) and \( S'_{DB,i} \) is defined as

\[
\text{sim-score} = 1 - \frac{\text{hamming_dist}(S'_{user}, S'_{DB,i})}{l},
\]

where the function \( \text{hamming_dist}(\cdot) \) calculates the hamming
distance, which is the number of different bits, between the
two input sequences. If the score is higher than a threshold
\( s_{\text{thresh}} \), the user RSS series \( S'_{user} \) is considered to be similar
to the \( i \)-th DB RSS series \( S'_{DB,i} \). We will empirically identify
the suitable value of \( s_{\text{thresh}} \) later in Section IX. Within the
comparisons between \( S'_{user} \) and the \( N \) DB RSS series, if there
are no less than \( n_{\text{thresh}} \cdot N \) matches, the batch of location
proofs are identified as valid.

VII. Threat Prevention Analysis

VProof preserves privacy for users participating the data
collection applications, since it does not require any information
that can be related to a user’s ID. VProof can efficiently prevent
the threat that a malicious user reports data about
location \( L \)’s conditions at a certain time \( T \) without presenting
at \( L \) at time \( T \). Let’s consider the following four cases of the threat.

Case I. The malicious user did not drive past the location
\( L \) at time \( T \). In this case, the malicious user cannot obtain
the valid VPacket authentication message (VAM), which will
make their location proofs rejected in the first step in the proof
verification process.

Case II. The malicious user drove past the location \( L \) at
time \( T' \), and saved the received V Packets, based on which he
constructed location proofs for another time \( T \). In this case,
the location proofs will be rejected, because the VAMs are not
valid (recall that the construction of VAM depends on several
parameters including time).

Case III. The malicious user drove past the location \( L' \)
at time \( T' \), and saved the received V Packets, based on which he
constructed location proofs for another location \( L \), where
\( L \) and \( L' \) are covered by signals of two different RSUs
respectively. In this case, the location proofs will be rejected,
also because the VAMs are not valid (they are constructed
during different RSU IDs and RSU secrets).

Case IV. The malicious user statically collected V Packets at
location \( L' \), where \( L \) and \( L' \) are covered by signals of the same
RSU, around time \( T \). This is the case the existing location
proof solutions cannot prevent. By contrast, VProof can easily
reject those location proofs, because the restored pattern of the
RSS series constructed from the statically collected location
proofs cannot match that of the RSS traces stored in the RSS
DB. Meanwhile, the malicious user cannot tune the parameter
RSS contained in the location proofs so that the restored RSS
pattern matches RSS DB’s record, since the malicious user has
no knowledge about the VPacket transmission power, which
is randomly selected and only known by the operator. We will
empirically evaluate the case that a malicious user submits
location proofs constructed with valid VAMs but guessed RSS
in the next section.

VIII. DISCUSSION

A. Privacy vs. Accountability

VProof provides a new option of location proofs with strong
user privacy protection. We realize that there is a tradeoff
between privacy and accountability. For instance, in VProof,
ITS operators cannot trace malicious users if there exist some,
since there is no way users submitting data can be traced.
Although this approach sacrifices the ability to trace down
malicious users reporting fake data, it greatly protects users’
privacy, and thus encourages participation of data collection.
We argue that our solution is more suitable for the situations
where privacy concerns outweigh any other concerns, as car
owners can simply opt out of providing any data if their
privacy is threatened.

B. Reporting fake information with valid location proofs

By verifying if a user’s historical locations match the data
he submits, we prevent the threat that a malicious user reports
fake information about numerous places he did not actually
visit. However, our solution does not deal with the threat that
malicious users report fake information with valid location
proofs (i.e., the malicious users drive their cars around a
particular area, collect V Packets, construct valid location
proofs and report fake information about that area). We consider
this problem to have less impacts than the threat we are considering
for the following three reasons. First, this problem is equiva-
 lent to the case where honest users unintentionally submit data
sensed by defective data collection devices. Second, compared
to the threat of interest where the malicious users can influence
as many places as he wants, reporting fake information with
valid location proofs can only impact a small amount of places
where the malicious users have actually visited. Third, the malicious users have much greater difficulties to report fake information with valid location proofs than to report fake information about places without actually visiting there, since instead of sitting somewhere statically, they need to constantly drive around the area of interest to launch the attack.

C. Attacks by offline profiling inherent RSS patterns

Another possible way to compromise our scheme is that malicious users first obtain RSS traces similar to those stored in the RSS trace DB by offline profiling, then they construct valid location proofs by applying RSU transmission power change histories to the profiled RSS traces. To launch the suggested attack, malicious users first need to obtain similar RSS patterns as the system operators do, which is hard to achieve, because we assume malicious users cannot replicate infrastructure units (such as RSUs) (Section III-B). Moreover, to launch the suggested attack, a malicious user also has to stay within the radio range of an RSU, and learns its transmission power change history. Therefore, the damage caused by the attack only covers the radio range of that particular RSU. Consequently, our scheme can still prevent malicious users from reporting fake information about a large number of places that they never visited.

D. Wormhole attacks

An wormhole attack is launched by two colluding attackers. In this attack, an attacker drives in the communication range of an RSU, collects authentic VPackets from the RSU and delivers the VPackets to another remote attacker. Then the remote attacker reports fake information with valid location proofs constructed from the authentic VPackets. In this case, the wormhole attack is equivalent to our previous discussion point (i.e., reporting fake information with valid location proofs), as we can treat the two colluded attackers as one attacker.

E. RSS database preparation

Building an RSS DB, which provides ground truth RSS patterns, for a large scale VProof deployment requires a substantial amount of preparation work. We emphasize that the RSS profiling process is a one-time effort. Furthermore, a promising way to reduce the preparation effort is to crowdsource the RSS profiling task to authorized users (i.e., delegate the profiling task to trusted everyday users). We leave utilizing crowdsourcing to expedite the RSS DB preparation as our future work.

F. RSU deployment and coverage

Our solution uses RSU to broadcast VPackets, based on which the location proofs are constructed. Thus, to protect roads from being harmed by malicious users reporting fake information about places they did not actually visit, we need the roads to be covered by RSU signals. In ITS systems, the ability to make vehicles communicate with roadside infrastructure is critical to achieve safety benefits [1], [2]. Deploying RSUs is the necessary step to enable the vehicle to infrastructure communication. Therefore, we believe RSU coverage will not be a problem in future ITS systems.

IX. Evaluation

A. Implementation and experimental setup

The prototype system. Our prototype VProof system consists of an wireless access point (Wiligear WBD-500 integrated radio board) mounted at roadside that serves as an RSU, a vehicular node equipped with a wireless receiver (Lenovo T61 + wireless card + external omni-directional antenna) that acts as ITS system users and a backend server (Dell T3500) that performs the operations done by the operator. The wireless AP runs a program that controls VPacket transmission power and broadcast rate according to the VProof scheme. The vehicular wireless node runs a program that constructs location proofs when VPackets arrive. To study whether hardware differences at user side have impacts on our scheme, we have used two different wireless cards on the vehicular wireless node, Ubiquiti SWX-SRC and Wistron CB9-GP, both of which have an external antenna socket. The backend server processes the location proofs constructed by the vehicular node offline. Note that in real ITS data collection applications, the sensed data and meta-data are not required to be uploaded to the backend server in real time [20]–[22]. Users can choose to upload the data anytime they feel appropriate, for example, when the uploads will not contend with other important tasks. Therefore, our choice of letting the backend server process the location proofs offline conforms the reality.

Per-packet transmission power control. The program we ran on the wireless AP is able to change the transmission power for each packet. This per-packet power control is achieved by specifying the desired transmission power in the packet’s radiotap structure. With the per-packet transmission power control, it is possible to change VPacket’s transmission power randomly while not affecting the normal tasks done by the RSU (e.g., beacon broadcasting).

Experimental setup. We have conducted extensive experiments on real road situations to evaluate our solution. The experiments were performed at three different locations in downtown environment with busy road traffic. We built the RSS DB by profiling the RSU RSS pattern (i.e., we drove car past the RSU, which was broadcasting VPackets using the full transmission power, and collected the VPackets) at the three locations in one day. We then performed experiments to collect user RSS traces (i.e., we drove car past the RSU, which was broadcasting VPackets using randomly selected transmission powers, and collected the VPackets) at each location in three other days. Three RSU transmission power levels were used: full power, $\frac{3}{4}$ power and half power. Note that all the three tested cases involved relatively straight routes. We expect our scheme works similarly for cases containing turnings and other irregular routes. This is because VPacket RSS pattern of a particular route is relatively stable as long as the surrounding structures remain unchanged. Furthermore, the driving speed in our experiment was around 20 mph to 30 mph. We expect
our solution also works in high speed scenario. This is because VPacket rate by RSUs is adapted dynamically based on the average road speed: recall that VPacket rate is calculated based on the coherence time of the environment, which is closely related to speed limit of the road.

B. Experimental results

Coherence time measurement. We measured the coherence time of the wireless channel at each of the three experiment locations as Camp and Knightly [27] did. Specifically, we let the RSU broadcast small packets (100 bytes/packet) at a very high packet rate (500 packets/s). Then we measured the RSS differences between different size of VPacket windows based on which we determined the coherence time of the channel. We found that the coherence times at our experiment locations were around 100 ms. Thus in our experiments, we set the VPacket rate to 100 frames/s, as we wanted users to receive 10 VPackets per period of coherence time. Meanwhile, according to Algorithm 2, when constructing the RSS series, we computed an RSS data point as the average RSS of VPackets broadcast using random transmission powers at the same location between two VPackets that are transmitted under the same power and within 100ms.

Quantifying the stableness of RSS difference of VPackets with two different powers. To restore the inherent RSS pattern from the RSS of VPackets broadcast using random transmission powers, our solution relies on the fact that the difference of RSS at the same location between two VPackets that are transmitted by the same RSU using two different powers is roughly a constant across the entire communication range of the RSU. We conducted an experiment to quantify how stable this RSS difference is. In the experiment, we let the roadside wireless AP change between two transmission powers periodically while broadcasting the VPackets. In the first half of a coherence time period, the VPackets were transmitted using the full power, and in the second half, the VPacket transmission power was set to half of the full power. The VPacket rate was 100 frames/s. We drove a car past the RSU and collected the VPackets. We processed the received VPackets by first averaging the RSS of consecutive VPackets with the same transmission power. Then we calculated the RSS difference between two sets of VPackets that were transmitted with two transmission powers within a period of coherence time. Finally, we found that the average RSS difference is 8.55 dBm with a standard deviation of 2.81 dBm. As we will show later, this RSS difference is already stable enough that allows us to keep the fidelity of RSS patterns during the pattern restoration process.

RSS pattern comparisons. We have collected over 50 user traces at the three experiment locations, which are denoted as road section A, B and C respectively. When comparing these honest user traces to their corresponding DB traces, we used an alphabet of four numbers (i.e., $K = 4$) at the quantization step. Empirically, we found that the similarity scores between all the user traces and the corresponding DB traces are larger than 0.8. Thus in our scheme, we set the threshold similarity score $s_{\text{thresh}}$ as 0.8.

We show the RSS pattern comparisons for traces collected at the road section A. Figure 6 (a1), (b1) and (c1) show the patterns of three original RSS series constructed from three user traces, $S_1$, $S_2$ and $S_3$, that were collected at the road section A. Figure 6 (a2), (b2) and (c2) show the patterns of the corresponding restored RSS series. Figure 6 (d) shows the pattern of the RSS series constructed from the DB's trace. We have only show the center part of the patterns for better illustration. The randomly selected transmission power of VPackets allows the RSS of VPackets received within a short range of distance exhibit no fixed pattern. Therefore, we can see that the three original RSS series are all different (the similarity scores of any pair of the three original RSS series are less than 0.58). However, once we restore the original RSS
series to their corresponding full power RSS series, the three user traces do exhibit similar RSS patterns with the DB trace. Figure 6(e) shows the similarity scores when comparing the DB trace against the original RSS series and against the three restored RSS series. It shows that the similarity scores between each original RSS series and the DB’s RSS series are smaller than 0.55, and the similarity scores between each restored RSS series and the DB’s RSS series are all larger than 0.81.

Another point worth mentioning is that of the first two user traces, S1 and S2, and the DB trace were collected using the Ubiquiti wireless card, while the third user trace S3 was collected using the Wistrong wireless card. Although the RSS readings from the Wistrong card were higher, it did not prevent our solution from correctly accepting the honest user trace, since our similarity comparison scheme only depends on the pattern itself but not RSS reading amplitude.

Dealing with VPacket losses. In a busy vehicular wireless environment, vehicles may not receive all the VPackets, based on which the location proofs are constructed. Our scheme addresses this issue by letting RSU broadcast multiple VPackets within a coherence time period and designing a DTW-based algorithm that can effectively deal with missing data points when comparing two RSS series. However, the VPacket losses scenario was not fully manifested in our real-world experiments, since we did not have a busy wireless environment on the road. To evaluate how our solution deals with VPacket losses, we simulated the losses by randomly taking VPackets out from an honest user trace based on certain probability. Then we constructed the RSS series from the user trace with missing VPackets, and compared it with the corresponding DB RSS series. Figure 7(a) shows the performances of the direct distance algorithm (i.e., directly compared two series without performing the DTW process) and the DTW algorithm in dealing with traces with missing VPackets. We can see that the DTW algorithm keeps the similarity score between the honest user trace with missing VPackets and the corresponding DB trace staying above the acceptance threshold as the VPacket lost probability goes up to 70%. In other words, for an honest user trace, as long as 3 of the 10 VPackets sent within the coherence time are received, our DTW algorithm can correctly mark it as valid.

Dealing with guessed RSS series. In this experiment, we considered the case that a malicious user submits location proofs with guessed RSS values in the hope that the restored RSS pattern can match the DB RSS traces. We first constructed an RSS series by randomly generating RSS values that are within the RSS range in our experiments. Then we simulated the malicious user has correctly guessed the value of an data point by replacing a random data point of this guessed series with a corresponding real data point in one of our user RSS series. We increased the proportion of the replacement to see how our scheme responds. Figure 7(b) shows the result. We can see that for a malicious user to get his RSS series accepted, he needs to correctly guess more than 80% of the data points. Since there are usually hundreds or thousands of data points in an RSS series (as in our experiments), we can draw a conclusion that the probability of a malicious user successfully makes a guessed RSS series accepted by our scheme is very low.

Evaluating RSS series quantization. This experiment evaluates how the quantization alphabet size K affects the similarity score of two RSS series. We consider both the case of matched RSS series and the case of mismatched RSS series. For the matched case, we compared S1, S2 and S3, which are three user RSS traces collected at the road section A, with the DB RSS trace of road section A (Figure 8 (a)). For the mismatched case, we compared S1, S2 and S3 with the DB traces of the road section A and road section B respectively (Figure 8 (b) and (c)). We can see that when choosing 4 and 0.8 as the values for parameters K and $s_{thresh}$ our scheme can correctly identify the RSS patterns of traces collected at the same location and also correctly distinguish the RSS patterns of traces collected at different places. According to our experimental results, we can develop an empirical formula to determine the value of $K$: $K = 3 + \left[(0.9 - s_{thresh}) \times 10\right]$.

Evaluating different similarity comparison methods. Our solution applies a DTW algorithm in the RSS series similarity comparison scheme. For a good similarity comparison scheme, the range of similarity scores it generates when comparing two similar series should be as narrow as possible, because a wide range of scores would make it hard to determine the similarity threshold, and it would likely lead to false positives or false negatives.

In this experiment, we compare three methods in terms of their ability to produce narrow score range when comparing two similar series. The first method dynamic time warping (DTW), where the similarity score is calculated based on the warped versions of the two quantized RSS series. The second method is direct distance (DD), where the similarity score is calculated directly based on the two quantized RSS series. The third method we compare is the Discrete Fourier Transform (DFT) method. DFT is a classical method to compare the similarities between time-series [28], [29]. With this method, DFT is first applied to the two time-series to compare. Then the two series are represented by the $2f_c$ coefficients of the first $f_c$ frequencies of their DFT result. Here $f_c$ is called the “cutoff frequency”. Then the similarity is quantified as the distance between the two $2f_c$-dimension vectors. We have tried this method as the RSS series comparison method for our scheme. Suppose after applying DFT, two RSS series $S_1$ and $S_2$ are presented as two $2f_c$-dimension vectors $\alpha_{i,\cos}, \alpha_{i,\sin}, \alpha_{i,\cos}, \alpha_{i,\sin}, \beta_{i,\cos}, \beta_{i,\sin}$, and $\beta_{i,\cos}, \beta_{i,\sin}$, the similarity score between $S_1$ and $S_2$ is computed as:

$$sim\_score = 1 - \frac{\sum_{i=1}^{2f_c} \left(\alpha_{i,\cos} - \beta_{i,\cos}\right)^2 + \left(\alpha_{i,\sin} - \beta_{i,\sin}\right)^2}{\sum_{i=1}^{2f_c} BIGGER(m_{S_{1,i}}, m_{S_{2,i}})},$$

where $m_{S_{k,i}} = \begin{cases} \theta_{i,\cos}^2 + \theta_{i,\sin}^2, & \text{if } k = 1, \\ \theta_{i,\cos}/\theta_{i,\sin}, & \text{if } k = 2; \end{cases}$ and function $BIGGER(\cdot)$ that returns the bigger value of its parameters.
We used the three methods (DTW, DD and DFT) to compare 25 of the (honest) user traces against their corresponding DB traces. The distribution of the computed similarity scores is plotted in Figure 8 (d). In the figure, the top/bottom whiskers show the maximum/minimum values of the similarity scores. The top/bottom of the boxes represent the upper/lower quartiles of the similarity scores, and the bars within the boxes represent the mean values of the scores computed by the three methods. Through this figure we can see that the DTW method outperforms the other two methods in that it produces the narrowest range of similarity scores, which makes it most suitable to serve as our RSS series comparison method.

X. CONCLUSION

We have proposed VProof, a lightweight and privacy-preserving location proof solution that does not rely on PKI systems. We built a VProof prototype system and evaluated it with extensive experiments performed on actual road conditions. The evaluation results show that VProof can effectively verify if users’ location claims match their historic locations without harming their location privacy.

REFERENCES