Collaborative Bluetooth-based Location Authentication on Smart Phones

Manoop Talasila, Reza Curtmola, Cristian Borcea

Computer Science Department, New Jersey Institute of Technology, Newark, NJ, USA

Abstract

Third-party location-based services are independent of wireless carriers and receive the user location from mobile devices GPS. A major problem in this context is how to prevent service abuse by malicious users who submit false locations by tampering with their phones. This paper presents LINK (Location authentication through Immediate Neighbors Knowledge), a location authentication protocol working independent of wireless carriers, in which nearby users help authenticate each other's location claims using Bluetooth communication. Simulation results demonstrate that LINK thwarts individual user attacks and a number of colluding users attacks. Experimental results over Android phones show that LINK works well at walking speeds and phone battery is not impacted significantly even for relatively high usage.

Keywords: Secure Location Authentication, Trust, Smart Phones

1. Introduction

Recently, location-based services (LBS) have started to be decoupled from the wireless network carriers, as illustrated by third-party services such as Foursquare [1], Google's Mobile Ads [2], Loopt [3], Brightkite [4], and Google's Latitude [5]. As such, LBSs must rely on mobile devices to provide their location, determined using GPS or other localization mechanisms. A major problem in this case is how to prevent service abuse by malicious users who tamper with the localization system on the mobile devices. For example, how can a coupon LBS verify that a user receiving a coupon is indeed in a 2-mile radius of the store? How can a social networking LBS that shows nearby users prevent false location claims used to infer the location of further-away users? How can a news agency authenticate the claimed location of a geo-tagged photo uploaded by citizens located at an event of public interest?

Although a significant number of publications tackled the location authentication problem, all of them assumed support from the wireless network infrastructure [6, 7] or from a deployed localization infrastructure using distance-bounding techniques [8, 9]. Typically, these solutions

Email addresses: `mt57@njit.edu` (Manoop Talasila), `crix@njit.edu` (Reza Curtmola), `borcea@njit.edu` (Cristian Borcea)

are based on signal measurements between the mobile devices and fixed, trusted beacons or base stations (e.g., cell towers, WiFi access points) with known locations [10]. The problem tackled in this paper is different as we aim for a solution that works without any support from the network/localization infrastructure. Such a solution is important because wireless carriers may refuse to authenticate user location for third-party LBSs due to legal and commercial reasons: they may not be allowed by laws to share any type of user location data, and they may not want to help their competition in the LBS area. Furthermore, using carrier support and existing privacy preserving methods [11] may not work as desired to solve the problem addressed in this paper. The first reason is commercial as mentioned above. The second reason is the low accuracy of the authenticated location. If the privacy preserving methods involve k-anonymity or similar methods, the accuracy of the claimed location may not be very good. For example, the carrier may provide coarse-grained location authentication in order to satisfy k-anonymity constraints (e.g., authenticate that the user is in a certain city or a large region of that city). Such authentication may not be useful for many LBSs, which may require fine-grained location authentication.

This paper proposes LINK (Location authentication through Immediate Neighbors Knowledge), a secure location authentication protocol in which users help authenticate each other's location claims. LINK associates trust scores with users, and nearby mobile devices belonging to users with high trust scores play similar roles with the trusted beacons/base stations in existing location authentication solutions. The main idea is to leverage the neighborhood knowledge available through short-range wireless technologies, such as Bluetooth which is available on most cell phones, to verify if a user is in the vicinity of other users with high trust scores.

LINK employs a Location Certification Authority (LCA) that interacts with LBSs and mobile users over the Internet. Before submitting a location authentication request to LCA, the claimers must broadcast a message to their neighbors using short-range wireless ad hoc communication. In response to this message, the neighbors send certification messages to LCA over the Internet. The LCA decides the claim's authenticity based on spatio-temporal correlation between users and the trust score associated with each user. The protocol leverages the centralized nature of the LCA to compute the trust scores based on past interactions and historical score trends. LINK works best in dense networks that provide enough neighbors. Nevertheless, the protocol was designed to be resilient to situations when users are alone.

Extensive simulation results and security analysis show LINK quickly thwarts attacks from individual malicious claimers or malicious verifiers. Over time, it also detects a number of more complex attacks involving groups of users colluding out-of-band.

LINK is implemented on Android-based Motorola Droid 2 phones. Mobile applications can use a simple location authentication API provided by the LINK package. To participate as verifiers in the system, users have to start a LINK background process. The LCA implementation helps the mobiles to avoid Bluetooth inquiry clashes when simultaneous claims are performed by multiple claimers. For testing purposes, we implemented a coupon LBS. Both this LBS and the LCA are implemented in Java.

In the experimental evaluation, we quantified the response latency and battery consumption associated with LINK. The results from a test-bed with six phones demonstrate that the response latency is low enough (typically 10-12s) for a static claimer to successfully complete the protocol before its verifiers, moving at walking speed, would go out of the Bluetooth transmission range. In terms of power consumption, a fully charged phone is capable of running thousands of claims and tens of thousands of certifications. Thus, users are not expected to turn off LINK due to power concerns. Micro-benchmark results show Bluetooth Discovery and Bluetooth connection establishment are the tasks which consume the most CPU cycles and battery power.

The rest of the paper is organized as follows. Section 2 presents motivating real-world scenarios. Section 3 defines the assumptions and the adversarial model. Section 4 describes the LINK protocol, and Section 5 analyzes its security. Section 6 presents the simulation results. The implementation and experimental evaluation are presented in Sections 7 and 8. The related work is discussed in Section 9, and the paper concludes in Section 10.

2. Motivating Scenarios

Decoupling LBSs from wireless carriers and allowing users to control location sharing (i.e., location is determined on the mobile device and not by the network infrastructure) improve the overall user experience in terms of flexibility and privacy. However, this setting requires a secure location authentication mechanism that is independent of the network providers. In the following, we motivate the need for such a mechanism by presenting several scenarios involving malicious behavior.

Location-based coupons [12, 13]. Stores would like to offer discount coupons to people who spend longer time shopping in the store [14]. Similarly, restaurants could offer priority seating for frequent customers. Malicious users may claim to be in the store or restaurant in order to receive these benefits. Thus, location authentication is necessary to prove that the customer is in the store or restaurant (and potentially to prove that the customer has been there for a longer time).

Location-based Social Networking [3, 1]. Consider the scenario where users can see on their mobile devices the location of nearby friends (e.g., Loopt). In this scenario, users can falsify their location in order to infer the location of their friends, ultimately resulting in a tracking application. Other social applications (e.g., Foursquare) give out prizes or discounts to people who check frequently at some places. In this case, malicious users can claim false locations to benefit from these prizes. Thus, location authentication is required for such social services/applications.

Location-based advertisements [15, 16, 2]. Consider a scenario where a person is employed to advertise a product near some public areas in NY city using electronic fliers broadcasted over short-range wireless networking (which can be cheaper than using cellular services). In this case, the person submits her location to an advertising LBS in order to download the electronic fliers for that area. Since the person is paid based on how many areas of interest she visits, her location has to be verified.

Traffic jam alerts[17, 18]. Suppose that a traffic jam LBS accepts alerts from people driving on congested roads and then distributes the alerts to other drivers. This LBS has to ensure at least the alert location validity because malicious users may try to proactively divert the traffic on roads ahead in order to empty these roads for themselves.

Citizen real-time journalism[19, 20]. In recent years, many sites (including well known newspapers and news agencies) have resorted to photos uploaded by citizens located in the middle of various events of public interest. In this way, real-time information from anywhere across the globe can be shared with the public as soon as the event happens. Similar to the previous example, malicious users may try to claim that an event is happening at a certain location while being somewhere else. Therefore, a first step toward validating the geo-tagged photos is to authenticate the claimed location.

3. Preliminaries

This section defines the interacting entities in our environment, the assumptions we make about the system, and the adversarial model.

Interacting entities. The entities in the system are:

- *Claimer*: The mobile user who claims a certain location and subsequently has to prove the claim's authenticity.
- *Verifier*: A mobile user in the vicinity of the claimer (as defined by the transmission range of the wireless interface, which is Bluetooth in our implementation). This user receives a request from the claimer to certify the claimer's location and does so by sending a message to the LCA.
- *Location Certification Authority (LCA)*: A service provided in the Internet that can be contacted by location-based services to authenticate claimers' location. All mobile users who need to authenticate their location are registered with the LCA.
- *Location-based Service (LBS)*: The service that receives the location information from mobile users and provides responses as a function of this location.

System and Adversarial Model. We assume that each mobile device has means to determine its location. This location is considered to be approximate, within typical GPS or other localization systems limits. We assume the LCA is trusted and the communication between mobile users and the LCA occurs over secure channels, e.g., the communication is secured using SSL/TLS. We also assume that each user has a pair of public/private keys and a digital certificate from a PKI. Similarly, we assume the LCA can retrieve and verify the certificate of any user. All communication happens over the Internet, except the short-range communication between claimers and verifiers.

We choose Bluetooth for short-range communication in LINK because of its pervasiveness in cell phones and its short transmission range (10m) which provides good accuracy for location verification. However, LINK can leverage WiFi during its initial deployment in order to increase the network density. This solution trades off location accuracy for number of verifiers.

LCA can be a bottleneck and single point of failure in the system. Currently, we do not address this issue, but standard distributed systems techniques can be used to improve the LCA's scalability and fault-tolerance. For example, an individual LCA server/cluster can be assigned to handle a specific geographic region, thus reducing the communication overhead significantly (i.e., communication between LCA servers is only required to access user's data when she travels away from the home region). LINK also needs significant memory and storage space to store historic data about each pair of users who interact in the system. To alleviate this issue, a distributed implementation of the LCA could use just the recent history (e.g., past month) to compute trust score trends, use efficient data intensive parallel computing frameworks such as *Hadoop* [21] to pre-compute these trends offline, and employ distributed caching systems such as *memcached* [22] to achieve lower latency for authentication decisions.

Any claimer or verifier may be malicious. When acting individually, malicious claimers may lie about their location. Malicious verifiers may refuse to cooperate when asked to certify the location of a claimer and may also lie about their own location in order to slander a legitimate claimer. Additionally, malicious users may perform stronger attacks by colluding with each other

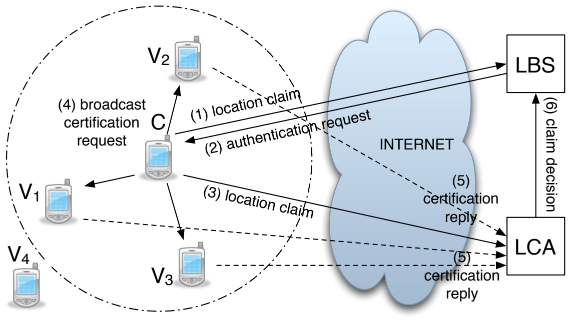


Figure 1: Basic Protocol Operation (where C = claimer, V$_i$ = verifiers, LBS = Location-Based Service, LCA = Location Certification Authority).

in order to verify each others false claims. Colluding users may also attempt two classic attacks: mafia fraud and terrorist fraud [23].

We do not consider selfish attacks, in which users seek to reap the benefits of participating in the system without having to expend their own resources (e.g., battery). These attacks are solved by leveraging the centralized nature of LCA, which enforces a tit-for-tat mechanism, similar to those found in P2P protocols such as BitTorrent [24], to incentivize nodes to participate in verifications. Only users registered with the LCA can participate in the system as claimers and verifiers. The tit-for-tat mechanism requires the verifiers to submit verifications in order to be allowed to submit claims. New users are allowed to submit a few claims before being requested to perform verifications.

Finally, we rely on the fact that a user cannot easily obtain multiple user IDs because the user ID is derived from a user certificate and obtaining digital certificates is not cheap; this deters Sybil attacks [25]. Further, techniques such as [26, 27], complimentary to our protocol, can be used to address these attacks.

4. Protocol Design

This section presents the basic LINK operation, describes the strategies used by LCA to decide whether to accept or reject a claim, and then details how trust scores and verification history are used to detect strong attacks from malicious users who change their behavior over time or collude with each other.

4.1. Basic Protocol Operation

All mobile users who want to use LINK must register with the LCA. During registration, the LCA generates a userID based on the user's digital certificate. Users of the system do not have to register with the LBS because they submit their LCA-generated userID to the LBS together with their requests. By not requiring the users to register with each LBS, we simplify the protocol.

At registration time, the LCA assigns an initial trust score for the user (which can be set to a default value or assigned based on other criteria). Trust scores are maintained and used by the LCA to decide the validity of location claims. A user's trust score is additively increased when her claim is successfully authenticated and multiplicatively decreased otherwise in order to discourage malicious behavior. This policy of updating the scores is demonstrated to work well for the studied attacks, as shown in section 6. The values of all trust score increments, decrements, and thresholds are presented in the same section. A similar trust score updating policy has been shown to be effective in P2P networks as well [28].

LCA also maintains a verification count of each user to determine whether the user is participating in verifications or simply using the system for her own claims. A user needs to perform at least VC_{th} verifications in order to be allowed to submit one claim (VC_{th} is the verification count threshold). Each time a verifier submits a verification, her verification count is incremented by 1, and each time a claimer submits a request to LCA, her verification count is decreased by VC_{th}. If a claimer's verification count reaches below VC_{th}, the claimer is informed to participate in other claimers verifications and her claims are not processed until her verification count reaches above VC_{th}.

Figure 1 illustrates the basic LINK operation. The pseudo-code describing the actions of claimers, verifiers, LCA, and LBS is presented in Algorithm 1. LINK messages are signed. When we say that a protocol entity sends a signed message, we mean that the entity computes a digital signature over the entire message and appends this signature at the end of the message. In step 1, a user (the claimer) wants to use the LBS and submits her location (Claimer Pseudo-code, line 5). The LBS then asks the claimer to authenticate her location (step 2) (LBS Pseudo-code, line 3). In response, the claimer will send a signed message to LCA (step 3) which consists of *(userID, serviceID, location, seq-no, serviceID, verifiers' IDs)* (Claimer Pseudo-code, line 12). The sequence number (*seq-no*) is used to protect against replay attacks (to be discussed in Section 5). The *serviceID* is an identifier of the LBS. The verifiers' IDs consists of the list of verifiers discovered by the claimer's Bluetooth scan; in this way, LCA will ignore the certification replies received from any other verifiers (the purpose of this step is to defend against mafia fraud attacks as detailed in Section 5). Furthermore, the LCA timestamps and stores each newly received claim.

The claimer then starts the verification process by broadcasting to its neighbors a location certification request over the short-range wireless interface (step 4). This message is signed and consists of *(userID, serviceID, location, seq-no)*, with the same sequence number as the claim in step 3. The neighbors who receive the message, acting as verifiers for the claimer, will send a signed certification reply message to LCA (step 5) (Verifier Pseudo-code, line 8). This message consists of *(userID, location, certification-request)*, where the userID and location are those of the verifier and certification-request is the certification-request broadcasted by the claimer. The certification-request is included to allow the LCA to match the claim and its certification messages. Additionally, it proves that indeed the certification-reply is in response to the claimer's request.

The LCA waits for the certification reply messages for a short period of time and then starts the decision process (described next in section 4.2). Finally, the LCA informs the LBS about its decision (step 6) (LCA Pseudo-code, line 9), causing the LBS to provide or deny service to the claimer (LBS Pseudo-code, line 8).

4.2. LCA Decision Process

In the following, we present the pseudo-code (Algorithm 2) and description of the LCA decision process. For the sake of clarity, this description skips most of the details regarding the use of historical data when making decisions, which are presented in Section 4.3.

Algorithm 1 LINK Pseudo-Code

Claimer Pseudo-code:

```
 1: // KC$_{pr}$ = private key of claimer
 2: // KC$_{pu}$ = public key of claimer
 3: // seq-no = provided by LCA before each claim
 4: // LBS-request = [userID, location]
 5: send(LBS, LBS-request)
 6: receive(LBS, response)
 7: if $response == Authenticate$ then
 8:     certification-request = [userID, serviceID, location, seq-no];
 9:     certification-request += sign(KC$_{pr}$, certification-request);
10:     verifiers = BluetoothDiscoveryScan();
11:     signed-verifiers = verifiers + sign(KC$_{pr}$, verifiers);
12:     send(LCA, certification-request, signed-verifiers);
13:     broadcast(verifiers, certification-request);
14:     receive(LBS, response);
```

Verifier Pseudo-code:

```
1: // KV$_{pr}$ = private key of verifier
2: // KV$_{pu}$ = public key of verifier
3: // Verifier enables Bluetooth Radio to discoverable mode
4: claimer = BluetoothDiscovered();
5: receive(claimer, certification-request);
6: certification-reply = [userID, location, certification-request];
7: certification-reply += sign(KV$_{pr}$, certification-reply);
8: send(LCA, certification-reply);
9: // Potential request from LCA to authenticate its location
```

LCA Pseudo-code:

```
1: receive(claimer, certification-request, verifiers)
2: if $verify(KC_{pu}, certification-request)$ and $verify(KC_{pu}, verifiers)$ then
3:     for $v$ = 1 to verifiers.size() do
4:         receive(v, certification-reply[v]);
5:         if $verify(KV_{pu}, certification{-}reply[v])$ and $verify(KC_{pu}, getCertificationRequest(certification{-}reply[v]))$ then
6:             storeDataForDecision(certification-request, certification-reply)
7: decision = decisionProcess(claimer, getClaimLocation(certification-request));
8: LBS = getServiceID(certification-request);
9: send(LBS, decision);
```

LBS Pseudo-code:

```
1: receive(claimer, LBS-request);
2: if $locationAuthentication == Required$ then
3:     send(claimer, Authenticate);
4:     receive(LCA, decision);
5:     if $decision == Reject$ then
6:         send(claimer, ServiceDenied)
7:         return
8: send(claimer, response);
```

4.2.1. Claimer lies

The LCA first checks the user spatio-temporal correlation by comparing the currently claimed location with the location of the user's previously recorded claim (lines 1-3 in the algorithm). If it is not physically possible to move between these locations in the time period between the two claims, the new claim is rejected.

If the claimer's location satisfies the spatio-temporal correlation, the LCA selects only the "good" verifiers who responded to the certification request and who are in the list of verifiers reported by the claimer (lines 5-12). These verifiers must include in their certification reply the correct certification request signed by the claimer (not shown in the code) and must satisfy the spatio-temporal correlation themselves. Additionally, they must have trust scores above a certain threshold. We only use "good" verifiers because verifiers with low scores may be malicious and may try to slander the claimer. Nevertheless, the low score verifiers respond to certification requests in order to be allowed to submit their own certification claims (i.e., tit-for-tat mechanism) and, thus, potentially improve their trust scores.

4.2.2. Colluding users help claimer

After selecting the "good" verifiers, the LCA checks if they are colluding with the claimer to provide false verifications (lines 13-15), and it rejects the claim if that is the case. This *collusion check* is described in detail in Section 4.3.

4.2.3. Contradictory verifications

If the LCA does not detect collusion between the claimer and verifiers, it accepts or rejects the claim based on the difference between the sums of the trust scores of the two sets of verifiers (lines 16-23), those who agree with the location submitted by the claimer (Y_{sum}) and those who do not (N_{sum}). Of course, the decision is easy as long as all the verifiers agree with each other. The difficulty comes when the verifiers do not agree with each other. This could be due to two causes: malicious individual verifiers or verifiers colluding with the claimer who have escaped detection.

Notation of Algorithm 2:

c: claimer
$V = \{v_0, v_1, \ldots v_n\}$: Set of verifiers for claimer c
N_{set}: Set of verifiers who do not agree with c's location claim
v_i: The i-th verifier in V
T_{v_i}: Trust score of verifier v_i
T_c: Trust score of claimer
W_{v_i}: Weighted trust score of verifier v_i
L_c: Location claimed by claimer
L_{v_i}: Location claimed by verifier v_i
IND_{tr}: Individual threshold to eliminate the low-scored verifiers
AVG_{tr}: Average threshold to ensure enough difference in averages
VRF_{cnt}=0: Variable to hold recursive call count
INC=0.1: Additive increment
DEC=0.5: Multiplicative decrement
secVer[]: Array to hold the response of second level verifications

Algorithm 2 Decision Process Pseudo-Code

```
decisionProcess(c, $L_c$):
:run to validate the location $L_c$ claimed by c
if $SpatioTempCorrelation(c) == FALSE$ then
    $T_c = T_c$*DEC
    return Reject
VRF$_{cnt}$++
if $hasNeighbors(c) == TRUE$ then
    for $i$ = 0 to $n$ do
        if $SpatioTempCorrelation(v_i) == FALSE$ then
            $T_{v_i} = T_{v_i}$*DEC
            remove $v_i$ from set V
        $W_{v_i}$ = getUpdatedWeightScore($v_i$,c)
        if $W_{v_i} \leq IND_{tr}$ then
            remove $v_i$ from set V
    if $V.notEmpty()$ then
        if $checkCollusion(V, c) == TRUE$ then
            return Reject
        (absAVGDiff,$Y_{sum}$,$N_{sum}$) = getTrustScore(V,c)
        if $absAVGDiff \geq AVG_{tr}$ then
            if $Y_{sum} \geq N_{sum}$ then
                $T_c = T_c$+INC
                return Accept
            else
                $T_c = T_c$*DEC
                return Reject
        else
            if $trend(c) == POOR$ then
                $T_c = T_c$*DEC
                return Reject
            else if $verifyScoreTrends(N_{set}) == POOR$ then
                if $T_c \leq IND_{tr}$ then
                    return Ignore
                else
                    $T_c = T_c$-INC
                    return Accept
            else if $VRF_{cnt} == 2$ then
                return Ignore {//2nd level claim is ignored}
            else
                for $i$ = 0 to $N_{set}$.size() do
                    secVer[i] = decisionProcess($v_i$,$L_{v_i}$)
                if $Majority(secVer) == Ignore$ or $Reject$ then
                    $T_c = T_c$+INC
                    return Accept
                else
                    $T_c = T_c$*DEC
                    return Reject
if $VRF_{cnt} == 2$ then
    return Ignore {//2nd level claim is ignored}
else if $trend(c) == POOR$ then
    $T_c = T_c$*DEC
    return Reject
else if $T_c \leq IND_{tr}$ then
    return Ignore
else
    $T_c = T_c$-INC
    return Accept
```

If the difference between the trust score sums of two sets of verifiers is above a certain threshold, the LCA decides according to the "winning" set. If it is low, the LCA does not make a decision yet. It continues by checking the trust score trend of the claimer (lines 24-27): if this trend is poor, with a pattern of frequent score increases and decreases, the claimer is deemed malicious and the request rejected. Otherwise, the LCA checks the score trends of the verifiers who disagree with the claimer (lines 28). If these verifiers are deemed malicious, the claim is accepted. Otherwise, the claim is ignored, which forces the claimer to try another authentication later.

Note that even if the claim is accepted in this phase, the trust score of the claimer is preventively decremented by a small value (lines 32-33). In this way, a claimer who submits several claims which are barely accepted will receive a low trust score over time; this trust score will prevent future "accepts" in this phase (lines 29-30) until her trust scores improves.

If the trend scores of both the claimer and the verifiers are good, the verifiers are challenged to authenticate their location (lines 34-44). This second level verification is done through a recursive call to the same *decisionProcess()* function. This function is invoked for all verifiers who do not agree with the claimer (lines 37-38). If the majority of these verifiers cannot authenticate their location (i.e., Ignore or Reject answers), the claim is accepted (lines 39- 41). Otherwise, the claim is rejected. The VRF_{cnt} variable is used to keep track of the recursive call count. Since we perform just one additional verification level, the function returns when its value is 2.

4.2.4. Claimer with no verifiers

The LCA deals with the case when no "good" verifiers are found to certify the claim in lines 45-54 of the algorithm (this includes no verifiers at all). If the claimer's trust score trend is good and her trust score is higher than a certain threshold, the claim is accepted. In this situation, the claimer's trust score is decreased by a small value $INC = 0.1$ as shown at line 53 in Algorithm 2 to protect against malicious claimers who do not broadcast a certification request to their neighbors when they make a claim. Over time, a user must submit claims that are verified by other users; otherwise, all her claims will be rejected.

4.3. Use of Historical Data in LCA Decision

The LCA maintains for each user the following historical data: (1) all values of the user's trust score collected over time, and (2) a list of all users who provided verifications for this user together with a verification count for each of them. These data are used to detect and prevent attacks from malicious users who change their behavior over time or who collude with each other.

Trust score trend verification. The goal of this verification is to analyze the historical trust values for a user and find malicious patterns. This happens typically when there are no good verifiers around a claimer or when the verifiers contradict each other with no clear majority saying to accept or reject the claim.

For example, a malicious user can submit a number of truthful claims to improve her trust score and then submit a malicious claim without broadcasting a certification request to her neighbors. Practically, the user claims to have no neighbors. This type of attack is impossible to detect without verifying the historical trust scores. To prevent such an attack, the LCA counts how many times has a user's trust score been decreased over time. If this number is larger than a certain percentage of the total number of claims issued by that user (10% in our implementation), the trend is considered malicious. More complex functions or learning methods could be used, but this simple function works well for many types of attacks, as demonstrated by our experiments.

Colluding users verification. Groups of users may use out-of-band communication to coordinate attacks. For example, they can send location certification messages to LCA on behalf of each other with agreed-upon locations. To mitigate such attacks, the LCA maintains an NxN matrix M that tracks users certifying each other's claims (N is the total number of users in the system). M[i][c] counts how many times user i has acted as verifier for user c. The basic idea is that colluding users will certify frequently each other's claims compared with the rest of the users in the system. However, identifying colluding users based solely on this criterion will not work because a spouse or a colleague at the office can certify very frequently the location of certain users. Furthermore, a set of colluding malicious users can use various permutations of subsets of malicious verifiers to reduce the chances of being detected.

Therefore, we propose two enhancements. First, the LCA algorithm uses weighted trust scores for verifiers with at least two verifications for a claimer. The weighted trust score of a verifier v is $W_v = T_v/\log_2(M[i][c])$, where T_v is the actual trust score of v. The more a user certifies another user's claims, the less its certifying information will contribute in the LCA decision. We choose a log function to induce a slower decrease of the trust score as the count increases. Nevertheless, a small group of colluding users can quickly end up with all their weighted scores falling below the threshold for "good" users, thus stopping the attack.

This enhancement is used until enough verification data is collected. Then, it is used in conjunction with the second enhancement, which discriminates between colluding malicious users and legitimate users who just happen to verify often for a claimer. LINK rejects a claim if the following conditions are satisfied for the claimer:

1. The number of claims verified by each potentially colluding user is greater than a significant fraction of the total number of claims issued by the claimer, and
2. The number of potentially colluding users who satisfy the first condition is greater than a significant fraction of the total number of verifiers for the claimer.

The pseudo-code for this function is presented in Algorithm 3. The function uses a dynamic threshold, W_{max}, which is a percentage of the total number of claims issued by the claimer over time (in our implementation, this percentage, β, is set to 30%). Since W_{max} is dynamically set, the algorithm can adapt its behavior over time. The function then computes the percentage of the verifiers who already reached W_{max} (lines 3-6). If a significant number of verifiers reached this threshold, the algorithm considers them malicious and punishes them together with the claimer. Simulation results presented in Section 6 demonstrate the advantage of punishing the verifiers as well vs. a method that would punish only the claimer.

A higher percentage of users verifying often for the same claimer is a strong indication of malicious behavior (the parameter α, set to 10% in our implementation, is used for this purpose). The underlying assumption is that a legitimate user going about her business is verified by many users over time, and only a few of them would verify often (e.g., family, lab mates).

Lines 7-13 show how the decision is made. If the number of potentially colluding verifiers is greater than α, the claimer and those verifiers are punished. Note that we do not punish a verifier who did not participate in verifications for this claimer since last time she was punished (line 10). In this way, the verifiers can redeem themselves, but at the same time, their contribution is still remembered in M. Finally, as shown in lines 14-18, if the percentage of potentially colluding users is less than α, the counts for those users are reset to allow them to have a greater contribution in future verifications for the claimer (this is correlated with the weighted trust score described previously).

Algorithm 3 Collusion Check Pseudo-Code

Notation:

```
M: NxN matrix for keeping count of certifications for pairs (claimer, verifier)
W_max: Threshold for count $w$
k: Number of verifiers with $w \geq W_{max}$ for claimer c
V: Set of active verifiers for claimer c
T_c: Trust score of claimer
T_{v_i}: Trust score of i-th verifier in V
m: Number of verifiers with $w > 0$ for claimer c
NumCl_c: Total number of claims made by claimer c
W_rst: reset value
```

checkCollusion(V, c):

```
1: $W_{max} = \beta * NumCl_c$
2: for $i = 0$ to $M.size$ do
3:     if $M[i][c] \geq W_{max}$ then
4:         k++;
5:     if $M[i][c] > 0$ then
6:         m++;
7: if $k/m \geq \alpha$ then
8:     $T_c = T_c * DEC$
9:     for $i = 0$ to $M.size$ do
10:        if $(M[i][c] \geq W_{max})$ and $((i \in V)$ or $(i.punished == FALSE))$ then
11:            $T_i = T_i * DEC$
12:            $i.punished = TRUE$
13:    return TRUE
14: else
15:    for $i = 0$ to $V.size$ do
16:        if $M[V[i]][c] \geq W_{max}$ then
17:            $M[V[i]][c] = W_{rst}$
18:    return FALSE
```

5. Security Analysis

LINK's goal is to prevent malicious users from claiming an incorrect location and to accept truthful location claims from legitimate users. The decision made by the LCA to accept or reject a claim relies on the trust scores of the users involved in this claim (i.e., claimer and verifiers). Thus, from a security perspective, the protocol's goal is to ensure that *over time* the trust score of malicious users will decrease, whereas the score of legitimate users will increase. LINK uses an additive increase and multiplicative decrease scheme to manage trust scores in order to discourage malicious behavior.

There are certain limits to the amount of adversarial presence that LINK can tolerate. For example, LINK cannot deal with an arbitrarily large number of malicious colluding verifiers supporting a malicious claimer because it becomes very difficult to identify the set of colluding users. Similarly, LINK cannot protect against users who accumulate high scores and very rarely issue false claims while pretending to have no neighbors (i.e., the user does not broadcast a certification request). An example of such situation is a "hit and run" attack, when the user does not return to the system after issuing a false claim. This type of behavior cannot be prevented even in other real-world systems that rely on user reputation, such as Ebay. Thus, we do not focus on preventing such attacks. Instead, we focus on preventing users that *systematically* exhibit malicious behavior. Up to a certain amount of adversarial presence, our simulation results in

Section 6 show that the protocol is able to decrease over time the scores of users that exhibit malicious behavior consistently and to increase the scores of legitimate users.

All certification requests and replies are digitally signed, thus the attacker cannot forge them, nor can she deny messages signed under her private key. Attackers may attempt simple attacks such as causing the LCA to use the wrong certification replies to verify a location claim. LINK prevents this attack by requiring verifiers to embed the certification request in the certification reply sent to the LCA. This also prevents attackers from arbitrarily creating certification replies that do not correspond to any certification request, as they will be discarded by the LCA.

Another class of attacks claims a location too far from the previously claimed location. In LINK, the LCA prevents these attacks by detecting it is not feasible to travel such a large distance in the amount of time between the claims.

The LCA's decision making process is facilitated when there is a clear difference between the trust scores of legitimate and malicious users. This corresponds to a stage in which the user scores have stabilized (i.e., malicious scores have low scores and legitimate users have high scores). However, there may be cases when this score difference is not significant and it becomes challenging to differentiate between a legitimate verifier vouching against a malicious claimer and a malicious verifier slandering a legitimate claimer. In this case, the LCA's decision relies on several heuristic rules. The true nature of a user (malicious or legitimate) may be reflected in the user's score trend and the LCA can decide based on the score trends of the claimer and verifiers. The LCA may also potentially require the verifiers to prove their location. This additional verification can reveal malicious verifiers which are certifying a position claim (even though they are not in the vicinity of the claimed position), because the verifiers will not be able to prove their claimed location.

Replay Attack. Attackers may try to slander other honest nodes by intercepting their certification requests and then replaying them at a later time in a different location. However, the LCA is able to detect that it has already processed a certification request (extracted from a certification reply) because each such request contains a sequence number and the LCA maintains a record of the latest sequence number for each user. Thus, such duplicate requests will be ignored.

Individual Malicious Claimer or Verifier Attacks. We now consider individual malicious claimers that claim a false location. If the claimer follows the protocol and broadcasts the certification request, the LCA will reject the claim because the claimer's neighbors provide the correct location and prevail over the claimer. However, the claimer may choose not to broadcast the certification request and only contact the LCA. If the attacker has a good trust score, she will get away with a few false claims. The impact of this attack is limited because the attacker trust score is decreased by a small decrement for each such claim, and she will soon end up with a low trust score; consequently, all future claims without verifiers will be rejected. Accepting a few false claims is a trade-off we adopt in LINK in order to accept location claims from legitimate users that occasionally may have no neighbors.

An individual malicious verifier may slander a legitimate user who claims a correct location. However, in general, the legitimate user has a higher trust score than the malicious user. Moreover, the other (if any) neighbors of the legitimate user will support the claim. The LCA will thus accept the claim.

Colluding Attack. A group of colluding attackers may try to verify each other's false locations using out-of-band channels to coordinate with each other. For example, one attacker claims a false position and the other attackers in the group support the claim. LINK deals with this attack by recording the history of verifiers for each claimer and gradually decreasing the contribution of verifiers that repeatedly certify for the same claimer (see Section 4.3). Even if this attack may be successful initially, repeated certifications from the same group of colluding verifiers will eventually be ignored (as shown by our simulations in Section 6).

Mafia Fraud. In this attack, colluding users try to slander honest claimers without being detected, which may lead to denial-of-service. For example, a malicious node M_1 overhears the legitimate claimer's certification request and relays it to a remote collaborator M_2; M_2 then re-broadcasts this certification request pretending to be the legitimate claimer. This results in conflicting certification replies from honest neighbors of the legitimate claimer and honest neighbors of M_2 from a different location. This attack is prevented in LINK because the LCA uses the list of verifiers reported by the legitimate claimer from its Bluetooth scan. Therefore, LCA ignores the certification replies of the extra verifiers who are not listed by the legitimate claimer. These extra verifiers are not punished by LCA, as they are being exploited by the colluding malicious users. Furthermore, it is difficult for colluding users to follow certain users in order to succeed in such an attack.

Limitations and Future Work. The thresholds in the protocol are set based on our expectations of normal user behavior. However, they can be modified or even adapted dynamically in the future.

LINK was designed under the assumption that users are not alone very often when sending the location authentication requests. As such, it can lead to significant false positive rates for this type of scenario. Thus, LINK is best applicable to environments in which user density is relatively high.

Terrorist fraud is another type of attack in which one attacker relays the certification request to a colluding attacker at a different location, in order to falsely claim the presence at that different location. For example, a malicious node M_1 located at location L_1 relays its certification request for location L_2 to collaborator M_2 located at L_2. M_2 then broadcast M_1's request to nearby verifiers. Verifiers certify this location request, and as a result the LCA falsely believes that M1 is located at L_2. This attack is less useful in practice and is hard to mount, as it requires one of the malicious users to be located at the desired location. From a philosophical perspective, attacker M_1 *has a physical presence* at the falsely claimed location through its collaborator M_2, so this attack is arguably not a severe violation of location authentication. The attack could be prevented by using distance bounding protocols [29, 30, 31]. However, applying such protocols is non-trivial. Two major issues are: (a) These protocols consider the verifiers as trusted where as LINK verifiers are not trusted and (b) special hardware (i.e., radio) not available on current phones may be needed to determine distances with high accuracy.

In addition, a group of colluding attackers may attempt a more sophisticated version of the terrorist fraud attack, in which the malicious collaborators share their cryptographic credentials to sign the false location claims for each other. In practice, this scenario is unlikely because a malicious user will be reluctant to share her credentials with another malicious user [31].

We implicitly assume that all mobile devices have the same nominal wireless transmission range. One can imagine ways to break this assumption, such as using non-standard wireless interfaces that can listen or transmit at higher distances such as the BlueSniper rifle from DEFCON '04. In this way, a claimer may be able to convince verifiers that she is indeed nearby, while being significantly farther away. Such attacks can also be prevented similar to the above solution for terrorist fraud using distance bounding protocols [29, 30, 31].

Location privacy could be an issue for verifiers. Potential solutions may include rate limitations (e.g., number of verifications per hour or day), place limitations (e.g., do not participate in verifications in certain places), or even turning LINK off when not needed for claims. However, the tit-for-tat mechanism requires the verifiers to submit verifications in order to be allowed to submit claims. To protect verifier privacy against other mobile users in proximity, the verification messages could be encrypted as well.

6. Simulations

This section presents the evaluation of LINK using the ns-2 simulator. The two main goals of the evaluation are: (1) Measuring the false negative rate (i.e., percentage of accepted malicious claims) and false positive rate (i.e., percentage of denied truthful claims) under various scenarios, and (2) Verifying whether LINK's performance improves over time as expected.

6.1. Simulation Setup

The simulation setup parameters are presented in Table 1. The average number of neighbors per user considering these parameters is slightly higher than 5. Since we are interested to measure LINK's security performance, not its network overhead, we made the following simplifying changes in the simulations. Bluetooth is emulated by WiFi with a transmission range of 10m. This results in faster transmissions as it does not account for Bluetooth discovery and connection establishment. However, the impact on security of this simplification is limited due to the low walking speeds considered in these experiments. Section 8 will present experimental results on smart phone that quantify the effect of Bluetooth discovery and Piconet formation. The second simplification is that the communication between the LCA and the users does not have any delay; the same applies for the out-of band communication between colluding users. Finally, a few packets can be lost due to wireless contention because we did not employ reliable communication in our simulation. However, given the low claim rate, their impact is minimal.

To simulate users mobility, we used the Time-variant Community Mobility Model (TVCM model) [32] which has the realistic mobility characteristics observed from wireless LAN user traces. Specifically, TVCM selects frequently visited communities (areas that a node visits frequently) and different time periods in which the node periodically re-appears at the same location. We use the following values of TVCM model in our simulations: 5 communities, 3 periods, and

Table 1: Simulation setup for the LINK protocol

Parameter	**Value**
`Simulation area`	100m x 120m
`Number of nodes`	200
`% of malicious users`	1, 2, 5, 10, 15
`Colluding user group size`	4, 6, 8, 10, 12
`Bluetooth transmission range`	10m
`Simulation time`	210min
`Node speed`	2m/sec
`Claim generation rate (uniform)`	1/min, 1/2min, 1/4min, 1/8min
`Trust score range`	0.0 to 1.0
`Initial user trust score`	0.5
``` ``Good'' user trust score threshold ```	0.3
`Low trust score difference threshold`	0.2
`Trust score increment`	0.1
`Trust score decrement - common case`	0.5
`Trust score decrement - no neighbors`	0.1

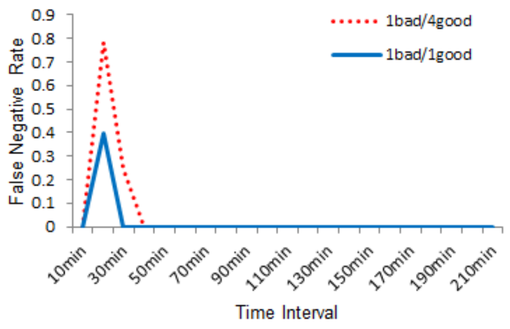


Figure 2: False negative rate over time for individual malicious claimers with mixed behavior. The claim generation rate is 1 per minute, 15% of the users are malicious, and average speed is 1m/s.

randomly placed communities represented as squares having the edge length 20m. The TVCM features help in providing a close approximation of real-life mobility patterns compared to the often-used random waypoint mobility model (RWP). Nevertheless, to collect additional results, we ran simulations using both TVCM and RWP. For most experiments, we have seen similar results between the TVCM model and the RWP model. Therefore, we omit the RWP results. There is one case, however, in which the results for TVCM are worse than the results for RWP: it is the "always malicious individual verifiers", and this difference will be pointed out when we discuss this case.

6.2. Simulation Results

Always malicious individual claimers. In this set of experiments, a certain number of non-colluding malicious users send only malicious claims; however, they verify correctly for other claims.

If malicious claimers broadcast certification requests, the false negative rate is always 0. These claimers are punished and, because of low trust scores, they will not participate in future verifications. For higher numbers of malicious claimers, the observed false positive rate is very low (under 0.1%), but not 0. The reason is that a small number of good users remain without neighbors for several claims and, consequently, their trust score is decreased; similarly, their trust score trend may seem malicious. Thus, their truthful claims are rejected if they have no neighbors. The users can overcome this rare issue if they are made aware that the protocol works best when they have neighbors.

If malicious claimers do not broadcast certification requests, a few of their claims are accepted initially because it appears that they have no neighbors. If a claimer continues to send this type of claim, her trust score falls below the "good" user threshold and all her future claims without verifiers are rejected. Thus, the false negative rate will become almost 0 over time. The false positive rate remains very low in this case.

Sometimes malicious individual claimers. In this set of experiments, a malicious user attempts to "game" the system by sending not only malicious claims but also truthful claims to improve her trust score. We have evaluated two scenarios: (1) Malicious users sending one truthful claim, followed by one false claim throughout the simulation, (2) Malicious users sending one false claim for every four truthful claims. For the first 10 minutes of the simulation, they

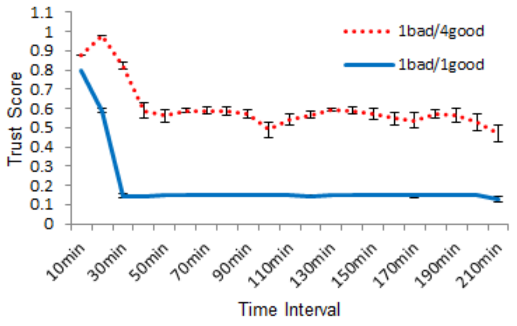


Figure 3: Trust score of malicious users with mixed behavior over time. The claim generation rate is 1 per minute, 15% of the users are malicious, and average speed is 1m/s. Error bars for 95% confidence intervals are plotted.

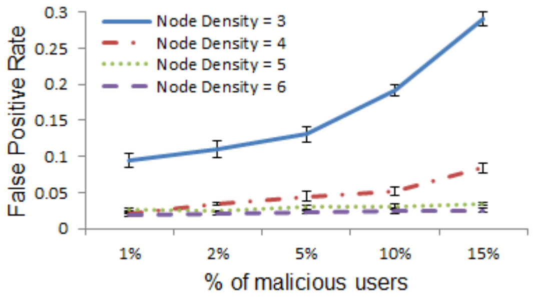


Figure 4: False positive rate as a function of the percentage of malicious verifiers for different node densities. The claim generation rate is 1 per minute and average speed is 1m/s. Error bars for 95% confidence intervals are plotted.

send only truthful claims to increase their trust score. Furthermore, these users do not broadcast certification requests to avoid being proved wrong by others.

Figure 2 shows that LINK quickly detects these malicious users. Initially, the false claims are accepted because the users claim to have no neighbors and have good trust scores. After a few such claims are accepted, LINK detects the attacks based on the analysis of the trust score trends and punishes the attackers.

Figure 3 illustrates how the average trust score of the malicious users varies over time. For the first type of malicious users, the multiplicative decrease followed by an additive increase cannot bring the score above the "good" user threshold; hence, their claims are rejected even without the trust score trend analysis. However, for the second type of malicious users, the average trust score is typically greater than the "good" user threshold. Nevertheless, they are detected based on the trust score trend analysis. In these simulations, the trust score range is between 0 and 1, i.e. additive increase of a trust score is done untill it reaches 1, then it is not incremented anymore. It stays at 1, until there is a claim rejection or colluding verifier punishment.

Always malicious individual verifiers. The goal of this set of experiments is to evaluate LINK's performance when individual malicious verifiers try to slander good claimers. In these experiments, there are only good claimers, but a certain percentage of users will always provide malicious verifications.

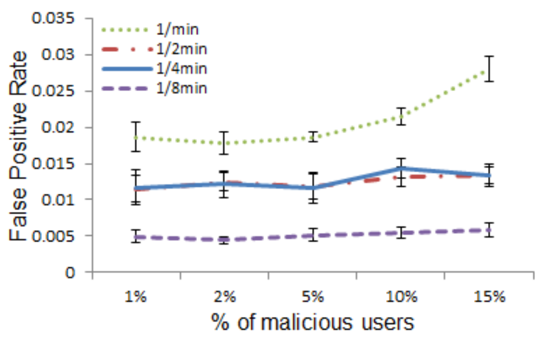


Figure 5: False positive rate as a function of the percentage of malicious verifiers for different claim generation rates. The average speed is 1m/s. Error bars for 95% confidence intervals are plotted.

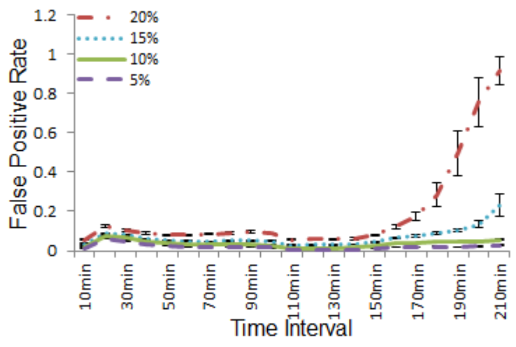


Figure 6: False positive rate over time for different percentages of malicious verifiers. The claim generation rate is 1 per minute and the average speed is 1m/s. Error bars for 95% confidence intervals are plotted.

Figure 4 shows that LINK is best suited for city environments where user density of at least 5 or higher can be easily found. We observe that for user density less than 4 LINK cannot afford more than 10% malicious verifiers, and for user density of 3 or less LINK will see high false positive rates due to no verifiers or due to the malicious verifiers around the claimer.

From Figure 5, we observe that LINK performs well even for a relatively high number of malicious verifiers, with a false positive rate of at most 2%. The 2% rate happens when a claimer has just one or two neighbors and those neighbors are malicious. However, a claimer can easily address this attack by re-sending a claim from a more populated area to increase the number of verifiers.

Of course, as the number of malicious verifiers increases, LINK can be defeated. Figure 6 shows that once the percentage of malicious users goes above 20%, the false positive rate increases dramatically. This is because the trust score of the slandered users decreases below the threshold and they cannot participate in verifications, which compounds the effect of slandering. This is the only result for which we observed significant differences between the TVCM model and the RWP model, with RWP leading to better results. This difference is due to the fact that nodes in same community in TVCM move together more frequently, which compounds the effect of slandering by malicious verifiers.

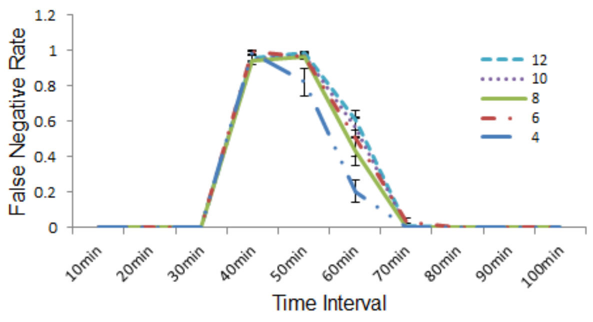


Figure 7: False negative rate over time for colluding users. Each curve is for a different colluding group size. Only 50% of the colluding users participate in each verification, thus maximizing their chances to remain undetected. The claim generation rate is 1 per minute and the average speed is 1m/s. Error bars for 95% confidence intervals are plotted.

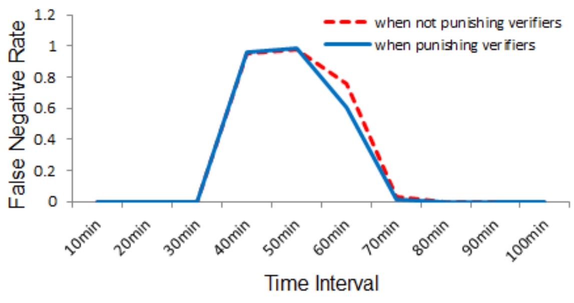


Figure 8: False negative rate over time when punishing and not punishing colluding verifiers. The size of the colluding group is 12, and 50% of these users participate in each verification. The claim generation rate is 1 per minute and the average speed is 1m/s.

Colluding malicious claimers. This set of experiments evaluates the strongest attack we considered against LINK. Groups of malicious users collude, using out-of-band communication, to verify for each other. Furthermore, colluding users can form arbitrary verification subgroups; in this way, their collusion is more difficult to detect. To achieve high trust score for the colluding users, we consider that they submit truthful claims for the first 30 minutes of the simulation. Then, they submit only malicious claims. As these are colluding users, there are no honest verifiers involved in these simulations.

Figure 7 shows that LINK's dynamic mechanism for collusion detection works well for these group sizes (up to 6% of the total nodes collude with each other). After a short period of high false negative rates, the rates decrease sharply and subsequently no false claims are accepted.

In LINK, all colluding users are punished when they are found to be malicious (i.e., the claimer and the verifiers). This decision could result in a few "good" verifiers being punished once in a while (e.g., family members). Figures 8 and 9 shows the false negative and positive rates, respectively, when punishing and not punishing the verifiers (i.e., the claimers are always punished). We observe that LINK takes a little more time to catch the colluding users while not punishing verifiers; at the same time, a small increase in the false positive rate is observed while punishing the verifiers. Since this increase in the false positive rate is not significant, we prefer to punish the verifiers in order to detect malicious claims sooner.

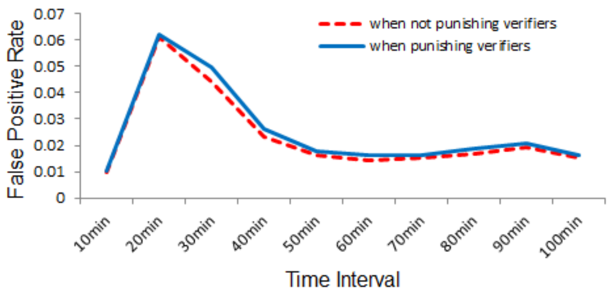
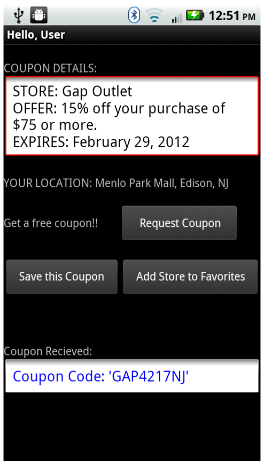


Figure 9: False positive rate over time when punishing and not punishing colluding verifiers. All parameters are the same as in Figure 8.


12:51 PM
Hello, User

COUPON DETAILS:

STORE: Gap Outlet
OFFER: 15% off your purchase of $75 or more.
EXPIRES: February 29, 2012

YOUR LOCATION: Menlo Park Mall, Edison, NJ

Get a free coupon!!

Coupon Recieved:

Coupon Code: 'GAP4217NJ'

Figure 10: Coupon application on Android phone

7. Implementation

The LINK prototype has been implemented and tested on Motorola Droid 2 smart phones installed with Android OS 2.2. These phones have 512 MB RAM, 1 GHz processor, Bluetooth 2.1, WiFi 802.11 b/g/n, 8 GB on board storage, and 8 GB microSD storage. Since we did not have data plans on our phones, all experiments were performed by connecting to the Internet over WiFi.

The implementation consists of two main components: (1) an Android client side package that provides applications with a simple API to perform location authentication and allows users to start a background LINK verification process on the phones; (2) the LCA server implemented in Java. The communication between applications and LBSs can be done through any standard or custom API/protocol. To test LINK, we implemented, in Java, a Coupon LBS and its associated application that runs on smart phones (illustrated in Figure 10).

7.1. Client API

We present the client API in the context of the Coupon LBS and its corresponding application. This service distributes location-based electronic discount coupons to people passing by a shopping mall. To prevent users located farther away to receive these coupons, the service has to authenticate their location.

The corresponding application is implemented as an Android Application Project. The user is provided with a simple 'Request Coupon' button as shown in Figure 10. The application submits the current location of the user to LBS and waits for an answer. Upon receiving the location authentication request from the LBS, the application invokes the *submit_claim* LINK API. An optimization done in the LINK package implementation was to limit the Bluetooth discovery to 5.12s instead of the standard 10.24s. This improves significantly the response time and saves energy on the phones (as shown in Section 8) at the expense of rarely missing a neighboring phone. In [33], the authors show a single inquirer can locate 99% of all scanning devices within transmission range in 5.12s.

The Bluetooth discovery's call back function returns the list of discovered Bluetooth devices. This list may contain devices that do not belong to phones running LINK. Trying to establish Bluetooth connections with these devices will lead to useless consumption of resources and extra-delays. Therefore, when *submit_claim* function contacts the LCA, it submits not only the location to be authenticated, but also the list of discovered Bluetooth devices. LCA answers with the assigned transaction ID for this claim and also provides the list of registered LINK-enabled devices. LINK on the phone will now establish a Bluetooth connection with each of these devices, sequentially, to send the claim certification request for the given transaction ID. Finally, the *submit_claim* function waits for the LCA answer, which is then returned to the application. If the response is positive, the application invokes another function to receive the coupon from the LBS; the LBS is informed of the authentication decision by the LCA directly.

For the verifier's side, which is not invoked from applications, the user has to start a LINK server that makes the Bluetooth listener active (i.e., puts Bluetooth in discoverable mode). This mode allows any claimer to find the phone in order to request the certification reply for their location claim. When a verifier's Bluetooth listener receives the certification request from a claimer, it invokes the *submit_verification* API. This function reads the claimer's message, generates the verification message, and sends it to the LCA. All the messages are signed using 'SHA1withRSA' algorithm from the 'java.security.Signature' package (1024 bits key size).

7.2. LCA Server

LCA is a multi-threaded server that maintains the claim transaction's hashmap, list of all user's details (ID, Bluetooth device address, RSA Public key, Trust score etc.), and all users weight matrices used in the decision process. One of the important implementation decisions is how long should a thread that received a claim wait for verifications to arrive. [1] This is necessary because some verifier phones may be turned off during the verification process, go out of the Bluetooth transmission range before the connection with the claimer is made, or even act maliciously and refuse to answer. This last example could lead to a denial of service attack on the LCA. Thus, the LCA cannot wait (potentially forever) until all expected verification messages arrive. It needs a timeout after which it makes the decision based on the verification received up to that moment.

We considered a waiting function linear in the number of verifiers. The linear increase is due to the sequential establishment of Bluetooth connections between the claimer and verifiers (i.e., they cannot be done in parallel). Since such a connection takes about 1.2s, we defined the

[1] the LCA knows the number of verifiers from the submitted claim message.

Table 2: Latency table for individual LINK tasks

Task	**Total time taken (s)**
`WiFi communication RTT`	0.350
`Bluetooth discovery`	5.000
`Bluetooth connection`	1.200
`Signing message`	0.020
`Verifying message`	0.006

waiting time *w = number of verifiers * 2s*, where 2s is an upper bound for the connection latency. However, this fixed waiting time could lead to long delays in situations when there are many verifiers and one or two do not answer at all. Therefore, we decided to adapt (i.e., reduce) this waiting time as function of the number of received verifications. Upon each received verification, *w = w * 4/5*. The value of the reduction factor can be further tuned, but so far it worked well in our experiments.

Once all the verification replies are received or the timeout expires, the LCA processes the claim through the decision process algorithm. Finally, the LCA informs the claimer and the LBS about its decision.

8. Experimental Evaluation

The main goals of these experiments are to understand the performance of LINK in terms of end-to-end response latency and power consumption when varying the number of claimers and verifiers. Additionally, we ran micro-benchmarks to understand the cost of individual tasks in LINK.

We used six Motorola Droid 2 smart phones for running the experiments. The LCA and LBS server programs are deployed on Windows-based DELL Latitude D820 laptops, having Intel Core 2 CPU at 2.16GHz and 3.25GB RAM. Before starting the Coupon application, the smart phones' WiFi interfaces are switched on. Additionally, the background LINK processes which listen for incoming certification requests are started on each phone.

8.1. Measurements Methodology

For latency, the roundtrip time of the entire LINK protocol (LINK RTT) is measured in seconds at the claimer mobile's coupon application program. For battery consumption, PowerTutor [34] available in the Android market is used to collect power readings every second. The log files generated by PowerTutor are parsed to extract the CPU and WiFi power usage for our application's process ID. Separate tests are performed to benchmark the Bluetooth tasks as PowerTutor does not provide the Bluetooth radio power usage in its logs. All values are measured by taking the average for 50 claims for each test case.

8.2. Micro-Benchmark Results

In these experiments, we used just two phones, one claimer and one verifier. Table 2 show the latency breakdown for each individual task in LINK. Bluetooth Discovery and Bluetooth connection are the tasks which took the major part of the response time. Note that we limited Bluetooth discovery to 5.12s, as explained in Section 7, to reduce the latency. From these results,

Table 3: Energy consumption for individual LINK tasks

Task	**Energy Consumed (Joules)**
`WiFi communication RTT`	0.100
`Bluetooth discovery`	5.428
`Bluetooth connection (Claimer side)`	0.320
`Bluetooth connection (Verifier side)`	0.017
`Signing message`	0.010
`Verifying message`	0.004

we estimate that LINK latency will be around 7s for one verifier; the latency increases linearly with the number of verifiers because the Bluetooth connections are established sequentially.

Table 3 shows the energy consumption breakdown for each task in LINK. The results show Bluetooth discovery consumes the most energy, while the Bluetooth connection's energy consumption will add up with the increase in the number of verifiers per claim.

8.3. End-to-End Performance Results

8.3.1. One claimer and different number of verifiers

We performed this set of experiments to see whether the estimation of latency and energy consumption based on the micro-benchmark results is correct. In these experiments we measured the LINK RTT and the total energy consumption at the claimer, while varying the number of verifiers. The results in Figure11 demonstrate that in terms of latency the estimations are highly accurate. In terms of power, we observe a linear increase in the difference between the estimations and the measured values. The difference becomes significant (>25%) for 5 verifiers. This difference could be explained by the extra-energy spent by Bluetooth to create and maintain Piconets with higher number of slaves.

The relevance of these results comes in the context of the two main questions we aim to answer: (1) Can LINK work well in the presence of mobility?, and (2) Can LINK run on the phones without quickly exhausting the battery?

If both the claimer and verifiers move at regular walking speed (1.2m/s), then LINK RTT should be less than 8s to have any chance to establish Bluetooth connections before users move out of each other's transmission range (i.e., Bluetooth range is 10m). This bound is the worst case scenario, and LINK could barely provide a response with just one verifier. Of course, LINK would perform better if not all users move or move in the same direction. A simple solution to avoid this worst case scenario is to make the claimer aware that she should not walk while submitting claims. This is a reasonable requirement because the claimer needs to interact with the phone to access the LBS anyway. In such a case, the bound on RTT doubles to 16s, and LINK will be able to establish connections with all verifiers (as shown in Figure11). Note that a Piconet is limited to 7 slaves, which limits the maximum RTT. Finally, let us recall that LINK is robust to situations when verifiers run out of the transmission range before the Bluetooth connection is established.

To understand LINK's feasibility from an energy point of view (i.e., to answer the second question posted above), we performed an analysis to see how many claims and verifications can a smart phone execute before it runs out of battery power. The total capacity of the Motorola Droid 2 battery is: 18.5KJ. Since LINK requires WiFi and Bluetooth to be on, we first measured the effect of these wireless interfaces on the phone lifetime. Table 4 shows the results for different

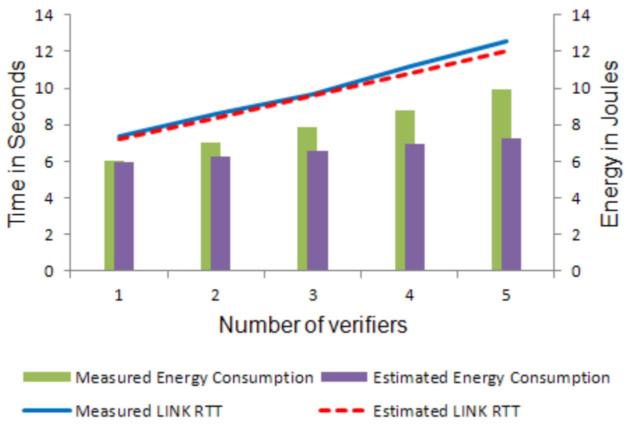


Figure 11: LINK RTT and total energy consumed by claimer per claim function of the number of verifiers

Table 4: Battery life for different WiFi and Bluetooth radio states

	Bluetooth and WiFi off	**Bluetooth off and WiFi on**	**Bluetooth and WiFi on**
Battery life	10Days 16Hrs	3Days 15Hrs	2Days 11Hrs

interface states (without running any applications). We observe that even when both are on all the time, the lifetime is still over 2 days, which is acceptable (most users re-charge their phones at night). The lifetime is even better in reality because Android puts WiFi to sleep when there is no process running that uses WiFi (in our experiments, we forced it to be on all the time).

Next, using this result, we estimate how many claims and verifications can a LINK do with the remaining phone energy:

Number of claims a phone can do until battery is exhausted = ***2,701 Claims***

Number of verifications a phone can do until battery is exhausted = ***20,458 Verifications***

These numbers demonstrate that a fully charged phone is capable of running LINK in real life with minimal impact on the overall battery consumption.

LINK is designed to work for cellular (over 3G) as well, which uses less battery power compared to WiFi [35]. In our experiments, we used WiFi because the phones did not have cellular service plans. In general, LINK will consume even less energy over the cellular network, but its RTT will increase because 3G has typically higher latency than WiFi.

8.3.2. Simultaneous nearby claimers

The goal in this set of experiments is to evaluate how much the RTT and the energy consumption increase when phones act simultaneously as both claimer and verifier. We used three phones which are in the Bluetooth transmission range of each other while sending concurrent claims. For each claim, the other two phones act as verifiers.

Table 5: Average RTT and energy consumption per claim for multi-claimer case vs. single claimer case

	Average RTT (s)	Energy consumed (Joules)
`Multi-Claimer case`	15.31	9.25
`Single-Claimer case`	8.60	7.04

When multiple phones send claims simultaneously, hence perform Bluetooth discovery at the same time, we notice many situations when only one or even no verifier was discovered. In total, only in 35% of the verifiers were discovered for all tests. This behavior is mostly due to the well-known Bluetooth inquiry clash problem[36]. An additional problem is the shortened discovery period (5.12s instead of 10.24s). While simultaneous claims in the same transmission range are expected to be rare in practice, we decided to provide a solution for this problem nevertheless.

Since LCA is aware of all the devices and their claims, it can successfully predict and prevent the Bluetooth inquiry clash by instructing the claimers on when to start their Bluetooth discovery. Practically, a claimer is delayed (for 3s in our experiments) if another one just started a Bluetooth discovery. The delay setting can be determined more dynamically by LCA at runtime depending on factors such as the number of simultaneous claimers and the number of surrounding verifiers in the region. Furthermore, claimers are allowed to perform complete Bluetooth inquiry scans (10.24s).

Table 5 compares the RTT and energy consumption measured when simultaneous claims are performed by three phones with the values measured for the case with one single claimer and two verifiers. With the new settings, the claimers were able to discover all verifiers. However, as expected, this robustness comes at the cost of higher latency (the energy consumption also increases, but not as much as the latency). The good news is that we expect all three claimers to be static according to the guidelines for LINK claimers. Therefore, the increased latency should not impact the successful completion of the protocol.

9. Related Work

Location authentication for mobile users has been studied extensively so far. To the best of our knowledge, all existing solutions employ trusted network/localization infrastructure [8, 9, 37, 38, 39, 40, 41, 42] to detect malicious users claiming false locations. Most of these solutions use distance bounding techniques, in which a beacon acting as verifier challenges the mobile device and measures the elapsed time until the receipt of its response.

None of these solutions, however, can be directly applicable to scenarios that involve interaction between mobile users and third-party services (i.e., services that do not have direct access to the network/localization infrastructure). The main novelty of LINK comes from employing mobile users (more exactly their mobile devices) to certify the location claimed by other users.

The closest infrastructure-based solution to LINK is [43], which uses mobile base stations to authenticate location in sensor networks. This solution authenticates location by leveraging verifier mobility: a mobile base station sends a verification request to a sensor from one position and then waits for the response at another position. We share the idea of using mobile verifiers, but this solution cannot be applied for our problem because it is expensive and does not scale.

In the same paper [43], the authors propose that nodes in a mobile ad hoc network (MANET) provide verifications for each other, which is similar to our idea. However, the authors do not consider cases when verifiers are malicious, when the claimer is alone, or when several nodes collude with each other. LINK provides solutions to all these issues. Furthermore, the proposed solution requires all nodes to have passive ranging capabilities (e.g., ultrasonic interface), while LINK works with the existing interfaces on the phones.

In [44], the authors propose location verification for VANETs in which a node can detect the malicious nodes after exchanging neighbor grouping information with other vehicles. However, the system model and design considered in this work vastly differs from LINKs system model and design: Directional antennas are used to perform relative position verification with help of neighboring nodes, whereas LINK uses Bluetooth communication to verify the users absolute location claim. Thus, LINK is more practical when considering the available technologies on existing mobile phones (i.e., Bluetooth vs. directional antennas). Also, most LBSs require absolute location verification, not relative positioning verification. Furthermore, the very nature of VANETs/MANETs makes it difficult to maintain accurate global information about the users in the system due to network partitioning and bandwidth limitations for large networks. Unlike these solutions, LINK leverages Internet connectivity and the centralized LCA to achieve global knowledge about the users in the system (i.e., registered with the LCA) and, thus, accurate location authentication.

Other cooperative location verification protocols in VANETs are mostly dependent on distance bounding techniques or based on challenging the Time-of-Flight of the signals which involves additional infrastructure support. The differences between LINK and these protocols are similar to those between LINK and the infrastructure-based solutions mentioned above.

Similar to our work, SMILE [45] and Ensemble [46] use information collected by mobile devices (keys from nearby users or received signal strength – RSS – values) to provide mutual co-location verification for mobile users. However, they do not provide location verification. RSS signatures in conjunction with RSS fingerprinting could be used for location verification, but such solutions do not scale due to the very dense fingerprinting required to achieve good accuracy.

In [47], the authors propose a protocol similar to LINK in which neighbor nodes use Bluetooth communication to provide location proofs for claimers. Since this protocol focuses mostly on location privacy, it presents only a discussion of potential solutions against malicious claimers and colluding users. LINK, on the other hand, describes, implements, and evaluates the success of solutions that make it resilient to many types of attacks.

As it is based on trust scores, LINK shares a number of similarities with work on reputation systems for P2P and MANETs. For example, CONFIDANT is a protocol [48] that avoids node misbehavior by establishing trust relationships between nodes based on direct and indirect observations reported by other nodes. The CORE protocol [49] takes a similar approach and uses reputation to enforce node cooperation. In contrast with CONFIDANT, CORE requires reputation values received from indirect observations, thus preventing malicious nodes from wrongfully accusing legitimate nodes. Unlike these systems which were designed for MANET, the system in [28] was designed for P2P networks. The management of the reputation values of peer nodes is similar to the way LINK manages the trust scores: when a peer is determined to be malicious, its reputation is cut in half; when a peer provides good service, its reputation is additively increased.

There are two main differences between this type of solution and LINK. First, LINK cannot monitor indirectly additional user actions (such as packet forwarding or file sharing) to assess the trust. Second, LINK employs the centralized LCA to have a global view of the the entire system. As such, it is able to detect malicious trust score trends and collusion attacks.

10. Conclusions

This paper presented LINK, a protocol for location authentication based on certification among mobile users. LINK can be successfully employed to provide location authentication for third-party location-based services without requiring cooperation from the network/localization infrastructure. The simulation results demonstrated that several types of attacks, including strong collusion-based attacks, can be quickly detected while maintaining a very low rate of false positives.

LINK was implemented on Android-based smart phones. Experimental results demonstrated that LINK is feasible in real-life situations. It is fast enough for static claimers to successfully authenticate their location despite the potential mobility of the verifiers. Additionally, its energy consumption does not have a significant impact on battery lifetime on the phones even for relatively high usage.

References

[1] Foursquare website, https://foursquare.com/, 2013.

[2] J. Marshall., Google launches location-based mobile ads, http://www.clickz.com/clickz/news/1725773/google-launches-location-mobile-ads, 2010.

[3] Loopt website, http://www.loopt.com/, 2013.

[4] Brightkite website, http://www.brightkite.com/, 2013.

[5] Google's latitude, http://www.google.com/latitude, 2013.

[6] T. Kindberg, L. Zhang, N. Shankar, Context authentication using constrained channels, in: Proc. of the 4th IEEE Workshop on Mobile Computing Systems and Applications (WMCSA'02), pp. 14–21.

[7] C. Wullems, O. Pozzobon, K. Kubik, Trust your receiver? enhancing location security, GPS World 1 (2004).

[8] S.Brands, D. Chaum, Distance-bounding protocols, in: Proc. of Workshop on the Theory and Application of of Cryptographic Techniques (EUROCRYPT'93), pp. 344–359.

[9] K. Rasmussen, S. Čapkun, Location privacy of distance bounding protocols, in: Proc. of the 15th ACM Conference on Computer and Communications Security (CCS'08), pp. 149–160.

[10] A. Vora, M. Nesterenko, Secure location verification using radio broadcast, IEEE Trans. Dependable Secur. Comput. 3 (2006) 377–385.

[11] T. Jiang, H. J. Wang, Y.-C. Hu, Preserving location privacy in wireless lans, in: Proceedings of the 5th international conference on Mobile systems, applications and services, ACM, pp. 246–257.

[12] L. Jakobson, Coupons on the go, Incentive 179 (2005) 16.

[13] Shop kick mobile app, http://www.shopkick.com/app, 2013.

[14] J. Heskett, T. Jones, G. Loveman, W. Sasser, L. Schlesinger, Putting the service-profit chain to work, Harvard Business Review (2008) 164–174.

[15] H. Haddadi, P. Hui, I. Brown, MobiAd: private and scalable mobile advertising, in: Proceedings of the fifth ACM international workshop on Mobility in the evolving internet architecture (MobiArch'10), ACM, pp. 33–38.

[16] L. Rao., Location-based mobile advertising company jiwire raises $20 million, http://techcrunch.com/2011/05/06/location-based-mobile-advertising-company-jiwire-raises-20-million/, 2011.

[17] J. Herrera, D. Work, R. Herring, X. Ban, Q. Jacobson, A. Bayen, Evaluation of traffic data obtained via gps-enabled mobile phones: The mobile century field experiment, Transportation Research Part C: Emerging Technologies 18 (2010) 568–583.

[18] J. White, C. Thompson, H. Turner, B. Dougherty, D. Schmidt, Wreckwatch: Automatic traffic accident detection and notification with smartphones, Mobile Networks and Applications 16 (2011) 285–303.

[19] K. Toyama, R. Logan, A. Roseway, Geographic location tags on digital images, in: Proceedings of the eleventh ACM international conference on Multimedia (MULTIMEDIA'03), ACM, pp. 156–166.

[20] Photo journalism website, http://www.flickr.com/groups/photojournalism/, 2013.

[21] Hadoop website, http://hadoop.apache.org/, 2013.

[22] Memcached website, http://memcached.org/, 2013.

[23] Y. Desmedt, Major security problems with the unforgeable(feige)-fiat-shamir proofs of identity and how to overcome them, in: SecuriCom (1988), volume 88, pp. 15–17.

[24] B. Cohen, Incentives build robustness in bittorrent, in: Workshop on Economics of Peer-to-Peer systems, volume 6, pp. 68–72.

[25] J. Douceur, The Sybil Attack, in: Proc. of IPTPS '01, pp. 251–260.

[26] J. Newsome, E. Shi, D. Song, A. Perrig, The Sybil attack in sensor networks: analysis & defenses, in: Proc. of IPSN '04, pp. 259–268.

[27] C. Piro, C. Shields, B. N. Levine, Detecting the Sybil attack in mobile ad hoc networks, in: Proc. of SecureComm'06.

[28] X. Chu, X. Chen, K. Zhao, J. Liu, Reputation and trust management in heterogeneous peer-to-peer networks, Springer Telecommunication Systems 44 (2010) 191–203.

[29] D. Singelee, B. Preneel, Location verification using secure distance bounding protocols, in: Proc. of the 2nd IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS'05), pp. 834–840.

[30] S. Capkun, J. Hubaux, Secure positioning in wireless networks, IEEE Journal on Selected Areas in Communications 24 (2006) 221–232.

[31] A. Ranganathan, N. Tippenhauer, B. Škorić, D. Singelée, S. Čapkun, Design and implementation of a terrorist fraud resilient distance bounding system, Computer Security–ESORICS (2012) 415–432.

[32] W.-j. Hsu, T. Spyropoulos, K. Psounis, A. Helmy, Modeling time-variant user mobility in wireless mobile networks, in: INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE, IEEE, pp. 758–766.

[33] B. Peterson, R. Baldwin, J. Kharoufeh, Bluetooth inquiry time characterization and selection, IEEE Transactions on Mobile Computing (2006) 1173–1187.

[34] L. Zhang, B. Tiwana, Z. Qian, Z. Wang, R. Dick, Z. Mao, L. Yang, Accurate online power estimation and automatic battery behavior based power model generation for smartphones, in: Proceedings of the eighth IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis (CODES/ISSS'10), ACM, pp. 105–114.

[35] A. Anand, C. Manikopoulos, Q. Jones, C. Borcea, A quantitative analysis of power consumption for location-aware applications on smart phones, in: IEEE International Symposium on Industrial Electronics (ISIE'07), IEEE, pp. 1986–1991.

[36] N. Ravi, P. Stern, N. Desai, L. Iftode, Accessing ubiquitous services using smart phones, in: Proceedings of the Third IEEE International Conference on Pervasive Computing and Communications (PERCOM'05), IEEE Computer Society, Los Alamitos, CA, USA, 2005, pp. 383–393.

[37] N. Tippenhauer, S. Čapkun, Id-based secure distance bounding and localization, in: Proc. of the 14th European conference on Research in computer security (ESORICS'09), pp. 621–636.

[38] J. T. Chiang, J. Haas, Y.-C. Hu, Secure and precise location verification using distance bounding and simultaneous multilateration, in: Proc. of the 2nd ACM Conference on Wireless Network Security (WiSec'09), pp. 181–192.

[39] N. Sastry, U. Shankar, D. Wagner, Secure verification of location claims, in: Proc. of the 2nd ACM Workshop on Wireless Security (Wise'03), pp. 1–10.

[40] N. Chandran, V. Goyal, R. Moriarty, R. Ostrovsky, Position based cryptography, in: Proc. of the the 29th International Cryptology Conference (CRYPTO'09), pp. 391–407.

[41] V. Shmatikov, M.-H. Wang, Secure verification of location claims with simultaneous distance modification, in: Proc. of the 12th Annual Asian Computing Science Conference (Asian'07), pp. 181–195.

[42] S. Capkun, K. El Defrawy, G. Tsudik, Group distance bounding protocols, in: TRUST'11. Proceedings of the 4th international conference on Trust and trustworthy computing, Springer, 2011, pp. 302–312.

[43] S. Čapkun, K. Rasmussen, M. Čagalj, M. Srivastava, Secure location verification with hidden and mobile base stations, IEEE Transactions on Mobile Computing (2008) 470–483.

[44] Z. Ren, W. Li, Q. Yang, Location verification for vanets routing, in: Wireless and Mobile Computing, Networking and Communications, 2009. WIMOB 2009. IEEE International Conference on, IEEE, pp. 141–146.

[45] J. Manweiler, R. Scudellari, L. Cox, SMILE: Encounter-based trust for mobile social services, in: Proceedings of the 16th ACM conference on Computer and communications security (CCS'09), ACM, pp. 246–255.

[46] A. Kalamandeen, A. Scannell, E. de Lara, A. Sheth, A. LaMarca, Ensemble: cooperative proximity-based authentication, in: Proc. of MobiSys '10, ACM, pp. 331–344.

[47] Z. Zhu, G. Cao, Applaus: A privacy-preserving location proof updating system for location-based services, in: INFOCOM, 2011 Proceedings IEEE, IEEE, 2011, pp. 1889–1897.

[48] S. Buchegger, J.-Y. L. Boudec, Performance analysis of the confidant protocol, in: Proc. of the 3rd ACM International Symposium on Mobile Ad Hoc Networking & Computing (MobiHoc'02), pp. 226–236.

[49] P. Michiardi, R. Molva, Core: A collaborative reputation mechanism to enforce node cooperation in mobile ad hoc networks, in: Proc. of the IFIP TC6/TC11 6th Joint Working Conference on Communications and Multimedia Security, pp. 107–121.