The Man Who *Was* There: Validating Check-ins in Location-Based Services

Iasonas Polakis*
FORTH-ICS, Greece
polakis@ics.forth.gr

Stamatis Volanis
FORTH-ICS, Greece
sebolani@ics.forth.gr

Elias Athanasopoulos
Columbia University, USA
elathan@cs.columbia.edu

Evangelos P. Markatos
FORTH-ICS, Greece
markatos@ics.forth.gr

ABSTRACT

The growing popularity of location-based services (LBS) has led to the emergence of an economy where users announce their location to their peers, indirectly advertising certain businesses. Venues attract customers through offers and discounts for users of such services. Unfortunately, this economy can become a target of attackers with the intent of disrupting the system for fun and, possibly, profit. This threat has raised the attention of LBS, which have invested efforts in preventing fake check-ins. In this paper, we create a platform for testing the feasibility of *fake-location attacks*, and present our case study of two popular services, namely Foursquare and Facebook Places. We discover their detection mechanisms and demonstrate that both services are still vulnerable. We implement an adaptive attack algorithm that takes our findings into account and uses information from the LBS at run-time, to maximize its impact.This strategy can effectively sustain mayorship in all Foursquare venues and, thus, deter legitimate users from participating. Furthermore, our experimental results validate that detection-based mechanisms are not effective against fake check-ins, and new directions should be taken for designing countermeasures. Hence, we implement a system that employs near field communication (NFC) hardware and a check-in protocol that is based on delegation and asymmetric cryptography, to eliminate fake-location attacks.

1. INTRODUCTION

Several location-based services have emerged during the last couple of years, Foursquare and Facebook Places being the most famous examples. The core operation of these social utilities is based on a large number of users that are willing to share their true geographic location. Users of these systems announce their location to the rest of the community or their on-line contacts, and can win awards depending on how often they share their location. For example, American Express offers discounts as an incentive for their customers to connect their account with the Foursquare application [2].

Foursquare is currently the most successful LBS. A very important aspect of its business model is the rewarding system for users that frequently check into specific venues. The user with the most check-ins for a venue in the last sixty days is crowned the venue's mayor. Venues attract customers by providing special offers for their mayors. This entails an incentive for users and, therefore, it is crucial to prevent fake check-ins that will have a negative impact on the system and deter users from participating [26]. As articles describing simple methods to post fake check-ins were published (e.g., [9]), Foursquare implemented a cheating-detection mechanism for prohibiting cheating users from becoming mayors. The deployment of such a feature [10, 5] was mandatory for reassuring users that cheating was deterred, and preventing a major decrease of the user base. Another important aspect of Foursquare is its recommender system [29], built upon the suggestions and tips left by users after checking into a venue. According to their CEO [7], "*check-ins drive the data, which drive the recommendation engine*". Since these services base their operation on the honest disclosure of location, it is vital for clients to transmit their position accurately so as to prevent the loss of the user base and the degradation of the recommender system.

Various methods have been used to post fake locations to mobile social networks. The most trivial is to hijack the GPS driver and provide applications with arbitrary coordinates. There are research efforts for the development of Trusted Sensors [33], including geolocation sensors. If smartphones are equipped with a trusted computing base, tampering with the data returned from the GPS antenna, or modifying the system to receive the coordinates from an application, can be harder although, arguably, still possible [35]. In this paper, we do not tamper with GPS readings, but conduct a systematic study of how the *application layer* of LBS can be leveraged to transmit fake information. Thus, our methodology is not affected by the presence of trusted sensors.

We create a testing platform that leverages public APIs available to application developers, and follow a black-box approach where we perform arbitrary check-ins in various places of the world, without changing our actual physical location. We then systematically analyze all server-side heuris-

*Iasonas Polakis, Stamatis Volanis and Evangelos Markatos are also with the University of Crete.

ACSAC '13 Dec. 9-13, 2013, New Orleans, Louisiana USA'

http://dx.doi.org/10.1145/2523649.2523653.

tics that aim to detect misbehaving clients. To ensure the completeness of our study, we also conduct experiments where we masquerade our actions to appear as if originating from the official applications, to reveal potential differences of the detection heuristics for public API calls. This is achieved using authentication tokens extracted from the official mobile applications through low-level reverse engineering.

We reveal a series of thresholds, which, if taken into consideration, allow a user to check into Foursquare, while traveling around the globe with a speed of over 900 mph. We also discover that users can check into a venue from as far as 200 meters away, and the maximum number of check-ins a user is allowed to commit is enforced using a 24 hour sliding window. Our technique revealed a bug in Facebook Places which allows anyone to perform check-ins all over the world with *unlimited* speed. No fix has yet been released.

Based on our findings we create an attack algorithm that takes into account heuristic thresholds and can maintain continuous mayorship in a set of venues across the globe. By employing 10,000 accounts, sold in the underground market for \$150-\$450, our adaptive algorithm can acquire the mayorship of all venues, and severely impact Foursquare's business model. Our experiments demonstrate that anomaly detection heuristics cannot secure a LBS against fake-location attacks. Detecting malicious clients and distinguishing fake check-ins from legitimate ones is not trivial. Even if heuristics become stricter, the attacker can simply follow a stealthier approach, as multiple accounts are used to carry out the attack. Stricter heuristics will also result in the system becoming too restrictive for legitimate users as well, which can have a negative impact on user participation. We argue that new directions need to be followed for securing LBS.

We present our proof-of-concept implementation of *Validated Check-in*, an NFC server solution, along with a security analysis of how it holds up against a series of attacks, as well as an evaluation of its performance. With a total cost of about \$75 at retail price, we consider our system to be ideal for deployment by venues that offer awards to LBS customers. The contributions of this paper are the following:

- We create a platform for evaluating the efficiency of server-side components employed by LBS for identifying clients providing fake locations. Our findings indicate that anomaly detection based heuristics are not sufficient for capturing clients that misbehave.
- To stress our experimental findings, we develop an adaptive attack algorithm that maximizes the impact of fake-location attacks, while remaining undetected. We show how an attacker can deploy a system-wide attack, that will have a significant impact on Foursquare's business model, with less than \$1,000.
- We implement Validated Check-in, a system designed to be deployed at LBS venues and ensure user presence during the check-in process. Based on commodity NFC hardware, it can protect against fake-location attacks. We design a check-in protocol for eliminating a range of attacks, and evaluate our system's performance.

2. LOCATION-BASED SERVICES

With smartphones, users can use networking services on the go. This introduces the aspect of location, which has led to the blooming of LBS, that allow users to inform their contacts of their current location.

Foursquare has over 30 million users and 1 million registered businesses, with users conducting millions of check-ins per day. The concept of achievements for users based on their check-in behavior was integral to its success. Achievements belong to three different categories: points, badges, and mayorships. Users earn points for every activity such as adding a new venue, while badges require combinations of activities. The mayorship is awarded to the user with the most check-ins for that venue in the last 60 days, and only one check-in per venue is allowed each day. As a result Foursquare is perceived as a game and, thus, cheating users discourage honest users from further participation.

Facebook Places follows a similar approach, where users can check into places and share that information with their friends. It does not present an award system to create a "gaming" experience. Nonetheless, it also provides venue owners with the ability to create offers for users that check into their place. Recently, Facebook merged this service into its system and discontinued it as a separate service. For the remainder of the paper, we will refer to this component as Places. Furthermore, venues are referred to as pages, however we will retain Foursquare's naming convention.

Check-in economy. The opportunity for venues to use LBS for advertising and attracting customers has led to the creation of a new business model that relies on users' activities combined with their geographical location. When users check into venues and post that information on their profiles, they are actually advertising the venues. As a result, an increasing number of venues are attracting Foursquare users by offering awards, ranging from discount prices to free products. This is similar to the *Like economy* [15], associated with users *liking* particular resources in Facebook, evolving with check-ins stemming from Foursquare's activity. Ensuring the produced economy is stable, requires that check-ins reflect clients announcing their true geographical location. However, this stability seems very fragile. A logical consequence of venues using Foursquare's achievement system for offering awards is the appearance of users cheating the system for fun or profit. This is done through *fake-location attacks*, where users check into venues without being there.

Fake-location attacks have a major impact on the credibility of LBS. They pose a great threat as competitive users will leave the system if fairness is not ensured and, thus, break all economics associated with these services. To make matters worse, as smartphones become widely used and the popularity of such services greatly increases, these attacks are bound to transit from sporadic incidents to organized fraud. We argue that LBS share certain properties that render them vulnerable. Thus, it is important to explore such attacks in detail and design effective countermeasures. We identify the following fundamental properties:

1. User location is sent from the client (user's device).
2. LBS has no definitive way of verifying the location.
3. Heuristics are used to detect behavior that exceeds acceptable limits, using the following information: (a) venue location, (b) user location, (c) timestamp of check-in, (d) history of previous user check-ins.
4. With such limited information, heuristics can only be applied on the following: (a) user's distance from venue, (b) user's speed between successive check-ins, (c) distance traveled in certain time windows, (d) number of check-ins in certain time windows.

As long as (2) stands true, LBS will remain vulnerable to fake-location attacks. The limited nature of information available (3), dictates the types of heuristics that can be deployed (4). We have designed our system to be fully configurable in regards to such heuristics. Thus, it can be used to identify the heuristic thresholds of any LBS that follows (1, 2) and demonstrate the extent of potential attacks. Our testing infrastructure can also assist LBS providers in detecting implementation bugs (as we demonstrate with Places).

User location. We expect user location to play a pivotal role in future services with functionality that will deviate from a simple check-in approach. Foursquare is also expanding by utilizing user data to build a reliable recommender system.By implementing an effective mechanism for validating the location reported by users, we can create a stable foundation for other novel services to be built upon.

3. METHODOLOGY

Our initial goal is to create an infrastructure that is able to perform arbitrary check-ins in LBS. Even though our current implementation supports Foursquare and Places, our testing approach is applicable to any LBS.

Use of Public APIs. Both Foursquare and Places provide public APIs that allow the development of custom applications. They include a set of HTTP requests, which cover the complete functionality of the service. The Places API is provided as part of the Facebook Graph API. For developing applications one only needs to register and obtain API credentials. Application code is not reviewed and, thus, anybody can create applications that post fake check-ins.

Mimic Official Applications. For one experiment we want to masquerade all calls made by our custom application to seem as if originating from the official one. Our goal is to explore whether the official app includes further information (i.e., custom headers) that makes it receive different "treatment" from the service. To do this, we must format all API calls like the official ones, and use the corresponding authorization token for each user[1]. Revealing the original protocol is challenging, since communication is sent over an encrypted channel using HTTPS. We extract the required information by modifying the application to provide us with the actual communication. If the requests are sent in cleartext, decompiling the application is not necessary. However, sending requests over SSL/TLS is considered safe practice and we expect that most LBS will do so.

Black-box testing. To reveal the detection mechanisms deployed by LBS, we follow a black-box testing approach and use test profiles that post arbitrary check-ins. We design our system to allow the configuration of several parameters of user behavior. By modifying the behavior, we are able to trigger the heuristics and identify their thresholds.

Ethical considerations. To minimize the impact of our experiments, and analyze the detection mechanisms without affecting other users, we took two precautionary measures. First, when exploring the heuristic thresholds, we modified our accounts so as not to acquire mayorships in venues which already had mayors. Specifically, mayorships are not awarded to accounts without a profile photo. Second, when experimenting with our adaptive attack algorithm, we targeted small venues with no mayors and used multiple accounts to serve as other customers. We were able to explore the heuristics in depth and demonstrate our automated attack, without having a negative impact on legitimate users.

[1]Each `(user, application)` pair has a unique access token.

4. SYSTEM IMPLEMENTATION

Our system has been implemented in `Python` as a collection of components, and can run on any computer.

Venue Crawler. Foursquare and Places have API functions that search for venues based on certain parameters. Given a set of coordinates, and a category description (e.g., bar), both services return a list of relevant venues nearby. The *Venue Crawler* takes as input a set of coordinates and searches for different categories of venues. For every venue we collect the name, venue ID, and location coordinates. We submit the venue's coordinates as our user's coordinates (unless we want our user to appear as being at a distance).

User Authentication. This part of our central component is responsible for authenticating the user to the LBS. It takes as input the user's access token used for authenticating with the service. We can select to authenticate with the access token that was created for use by our custom application, or the one extracted from the official application. Based on which one we select, we can appear to be sending the check-ins from the custom application or the official one.

Check-in Manager. This implements the core functionality of our system as it simulates a user checking into venues. It takes as input a list of venues that will be used for the arbitrary check-ins, and a set of values that configure the user's behavior. Several aspects of user behavior can be configured to explore the heuristics deployed by a LBS.

5. MEASUREMENTS - FOURSQUARE

Foursquare has implemented a system, which they refer to as "cheater code", for detecting users that post fake check-ins. While the mechanism has not been disclosed, according to Foursquare [8] detection is based on information from: (i) the user's phone, (ii) the official application, and (iii) an advanced detection algorithm. A check-in is accepted even if it triggers one of the heuristics, however, it is not taken into consideration for mayorships. In Table 1 we provide a short description of the heuristics we discovered, along with the threshold values after which check-ins are flagged.

5.1 Service Responses

All check-ins posted by our system through the API, receive a response message. If a check-in is considered legitimate, Foursquare returns a message verifying the check-in, while ones that are considered cheating receive an error message. By configuring our users to perform specific actions with precise timing, we can model various types of behavior. Based on the response messages, we know when a specific heuristic was triggered, and based on the user actions we discover the conditions under which it happened.

GPS distance: the user's location exceeds the maximum acceptable distance from the venue. The distance is calculated based on the user's coordinates sent by the application, and the venue's coordinates in the Foursquare database.

High speed: the user's speed exceeds the maximum threshold. Speed is calculated based on the time elapsed between the current check-in and the previous *legitimate* check-in, and the distance of the respective venues.

Rapid fire: the user exceeds the maximum number of acceptable check-ins for a certain time window.

Heuristic	Description	Threshold	Range	IT | IN
Maximum check-ins	Number of check-ins in specific time window.	5 check-ins 8 check-ins 49 check-ins 90 check-ins	time $\leq$ 1 minute time $\leq$ 15 minutes time $\leq$ 24 hours time $\leq$ 72 hours	-
User speed	Elapsed time and distance traveled between check-ins.	4 km/min 25 km/min	dst $<$ 100 km dst $\geq$ 100 km	0.2% | 49% 3.0% | 37%
GPS distance	Distance between coordinates of user and venue.	200 meters	-	1.1% | 5.5%

Table 1: Detection heuristics and the respective thresholds after which check-ins are flagged as cheating.

5.2 Device-based heuristics

The official application queries the device for the GPS coordinates, which are correlated with those of the venue. However, in large venues, a user may be present but not at the exact coordinates the system has registered. Additionally, cell phone GPS readings cannot always identify the location with high accuracy, and include an "accuracy" parameter as an indication of a margin of error. Applications can query the device if such information is needed. The API check-in function has an optional field for this information. To compensate for low accuracy readings, heuristics allow check-ins from a certain distance. While a reasonable threshold will facilitate legitimate users, high-distance tolerance enables users to cheat without spoofing the GPS data. To detect the threshold we conduct *Experiment A*:

- The system takes as input a predefined list of venues and their coordinates. After each check-in, it waits for a specific amount of time, before the next check-in. The amount of time is large enough, to avoid triggering heuristics that enforce constraints on user speed. It calculates a set of coordinates for the user that are X meters away from the venue, and increases X by Y meters after every check-in. We repeat for multiple values of X, Y.

Results were consistent across all experiments conducted over a period of 6 months. The results from a representative experiment can be seen in Figure 1. Check-ins are accepted from up to 200m away. Once the distance between a user's reported position and that of the venue exceed that threshold, the check-in is flagged as cheating and receives the "GPS distance" error. A high threshold makes it trivial for users to check into venues without being near them. We discuss the accepted check-ins for over 200m in Section 5.4.

5.3 User-behavior heuristics

A series of variables, based solely on the user's behavior, are evaluated before a check-in is deemed legitimate.

Maximum check-ins. Foursquare sets a limit on the number of check-ins in a given time window. To estimate the threshold of this heuristic we setup *Experiment B*:

- Our system takes as input a predefined list of venues and their coordinates. It places the user at the venue's exact coordinates, and after each check-in, waits for X seconds, before the next check-in.
- If a check-in receives the "rapid fire" error message we follow one of three different strategies. *Constant*: follow the same pattern, and sleep for X seconds after each check-in. *24h Mute*: sleep for 24 hours before attempting another check-in. *Exponential Back-off*: whenever an error message is received, X is doubled. In all 3 cases, whenever the user successfully checks-in again, X is reset to its original value. We repeat this procedure with different values of X.

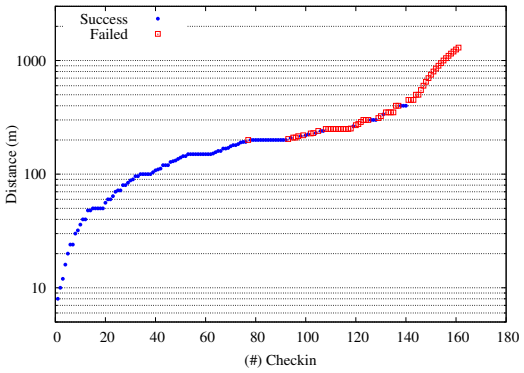

Figure 1: The distance between the GPS coordinates of the user and the venue.

Table 1 summarizes our results. Different thresholds apply for the number of check-ins users are permitted to make in certain time windows. Specifically, constraints are set for prohibiting bursts of check-ins by allowing a small number of check-ins to be posted within a time window of 15 minutes. Foursquare also sets a limit on the number of check-ins a user can commit in a 24 hour period. Figure 2 shows the results for the three approaches (marked with FS), with the bold sections indicating successful check-ins. We can see that all users receive errors after 49 check-ins. Foursquare does not reset the number of check-ins for a day at a specific moment in time, but checks them within a 24-hour sliding window. The user that never stops bombarding the service with check-ins can escape the ban period only for very short time windows (the short lines in the Constant FS bar), because even check-ins that receive error messages count as part of the 49 allowed. The user that pauses for 24h after the first error, escapes the ban period faster than the other two. Foursquare also examines the check-ins committed in a 72-hour window, and allows only 90 check-ins. Even if a user commits 49 check-ins in the first 24 hours, he cannot exceed the threshold of 90 in a given 72 hour window. The 3 timing strategies used are not the most efficient in regards to attacking an LBS, but aim at revealing the detection mech-

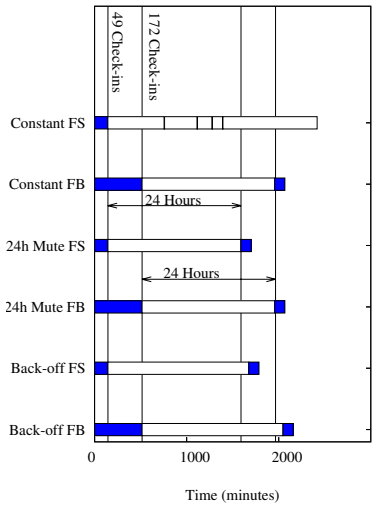

Figure 2: The maximum check-ins allowed per user. We depict three strategies. They all simulate a user that checks in with a constant rate, until it is prohibited by the service, upon which the strategy changes.

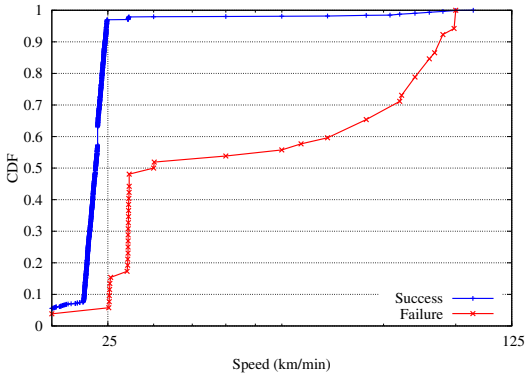

Figure 3: Speed heuristic for distances over 100km.

anism deployed by the system. In Section 7 we describe our adaptive attack that uses a more efficient timing strategy.

User speed. This heuristic measures the geographic distance traveled and the time elapsed between two consecutive check-ins. The "high speed" error is returned when the acceptable thresholds are exceeded. This is the main heuristic for detecting fake-location check-ins, as it is impossible to post legitimate check-ins that exceed the thresholds set by Foursquare. We conduct *experiment C*:

- Our system takes as input a predefined list of venues and a user velocity V. After each check-in, it calculates the exact distance X to the next venue and how many seconds T it must wait before checking into the next venue, so the user will appear to be traveling at a speed V. We repeat this for different values of X and V.

We model our users to appear as traveling at any speed we want. Depending on the value of X, Foursquare allows different speeds. For distances up to 100 kilometers users are allowed to travel at 4 kilometers per minute (approximately 150 miles per hour). For any distance longer than 100 km, users can travel at any speed below 25 kpm (approximately 932 mph) which is much faster than commercial airplanes (that average 600 mph). Figure 3 shows the results from a representative set of experiments for distances longer than 100 km. When conducting experiments with user speeds over the threshold, several interspersed check-ins are accepted. This is because Foursquare uses the last accepted (i.e., not flagged as cheating) check-in as the user's last location when calculating the elapsed time and traveled distance. Thus, it perceives that it took a longer time than it actually did to travel the distance. This results in calculating a speed that is below the threshold, and accepting the check-in. When the speed is slightly over the thresholds, the number of accepted and flagged check-ins are almost equal.

Traveling distance constraints. We explore if any constraints apply for the distance users can travel. We conduct *Experiment D*, which is the same as *Experiment C* except that we use a list of venues, located in different countries. We model our user to travel right below the speed threshold, at 24 kilometers per minute. At that speed, all check-ins posted by our system were accepted, and our user covered a distance of 36,120 kilometers (which is 90% of the circumference of Earth) in 25 hours. As our user traveled steadily for over a day, we conclude that there are no heuristics for imposing constraints on the distance a user can cover.

History heuristics. We compare the thresholds for three accounts with varying behavior: an account with no check-in history, one with many legitimate check-ins and a few that exceeded the maximum number allowed, and one that greatly exceeded all thresholds. We repeat our previous experiments with all accounts running simultaneously with the exact same variables. Results showed that thresholds are the same regardless of the user's cheating history.

Cheating penalties. We also found that Foursquare does not impose any penalties on users that have triggered the heuristics, and no "ban" periods are enforced.

5.4 Heuristic inconsistencies

During our experiments, we detected inconsistent behavior of the heuristics. In the first case heuristics are triggered while we remain beneath the thresholds (inconsistent triggering), and in the second case their mechanisms are not triggered by behavior that exceeds the thresholds (inconsistent non-triggering). While they may not be errors of the detection mechanism in all cases, they do present an inconsistent behavior in regards to the thresholds calculated based on the extensive number of experiments.Nonetheless, we refrain from the standard terms of false positives and false negatives used for evaluating detection mechanisms. The last column of Table 1 shows the percentage of these cases. In the user speed experiments, the ratio of inconsistently accepted check-ins is high due to the way Foursquare calculates user speed, as explained in *experiment C*. Here, we omit these and present some other incidents as examples.

Inconsistent triggering (IT). In experiments with a speed beneath the threshold (e.g., 0.26 kpm), some check-ins received the "high speed" error. In several cases we received the "GPS distance" error even though the user had the exact coordinates Foursquare returns for the venue. If we immediately repeated the check-in, it was deemed legitimate.

Inconsistent non-triggering (IN). In several cases our system was able to check in our users from as far as 900m away. While a velocity of (right below) 25 kpm was the max-

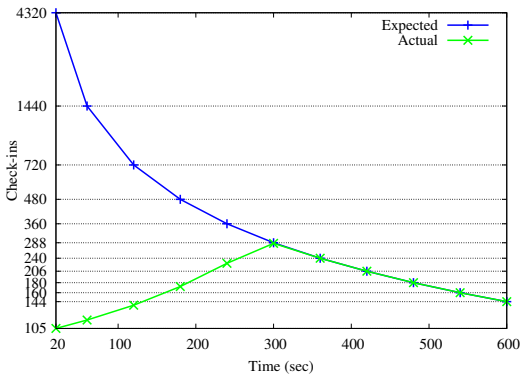

Figure 4: Expected vs. achieved check-ins for different time intervals T between check-ins for 1 day.

imum speed our user could exhibit without any check-ins being flagged, in the experiments with higher speeds some check-ins were considered legitimate. We had a check-in accepted with a speed of 107 kpm, without an intermediate flagged check-in to alter the speed calculation by Foursquare.

6. MEASUREMENTS - PLACES

Service Responses. Places also returns error messages: *significant distance from previous check-in* when the user travels too fast, and *too many places in a short amount of time* when the user exceeds a number of allowed check-ins.

Device-based heuristics. We replicate *Experiment A*, as outlined in Section 5, to reveal the maximum distance from which users can check into venues. For a series of venues, we gradually increase the user's distance from the venue's location. For each check-in attempt, we send the venue's ID and the user's location coordinates. We did not receive an error message for any of the distances we tried. As distances increased even more, we were able to check our users into places with coordinates located on different continents. Our experiments show that the user's coordinates are never compared to those of the venue when a user checks in, and any coordinates are accepted. We setup *Experiment E*, to demonstrate how this can be exploited by an attacker:

- Our system takes as input a list of venues from around the world, and a set of user coordinates. For every venue, the user checks-in with the same location coordinates, regardless of the venue's location. The user waits for 1 minute between check-ins.

With this experiment, we are able to check our user into venues around the globe in just a few minutes. As the user always sends the same coordinates for his location, when the system compares his new position to that of his previous check-in, it detects no change and the speed heuristic is never triggered. Thus, *an attacker can completely bypass the traveling speed constraints and check into venues around the globe with unlimited speed.* After further investigation, we found a bug report [4] submitted to Facebook two weeks prior, for the Graph API. The bug report was acknowledged and received an "assigned" status, but was closed one year after its submission, without any fix being released. This can be attributed to the fact that the report describes the problem in the check-in mechanism without pointing out the

Algorithm 7.1: ATTACK(N)

$L \leftarrow$ LISTOFVENUES(N)
$c \leftarrow 0$
while
do
- $p \leftarrow$ L.DEQUEUE()
- $n \leftarrow$ MAYORCHECKINSATVENUE(p)
- $m \leftarrow$ OURCHECKINSATVENUE(p)
- **if** $m \leq n$
  - **then**
    - $checkin \leftarrow$ CHECKIN(p)
    - CLIST.ENQUEUE($checkin$) (1)
    - $c \leftarrow c + 1$
    - $t \leftarrow$ ADJUSTSLEEP($CList, c$) (2)
    - **if** $c = MAX$
      - **then** $\{c \leftarrow c - 1$
    - SLEEP(t)
- L.ENQUEUE(p)

Figure 5: Pseudo-code of the actual attack.

security implications of this bug, and how it can be exploited to bypass the other detection mechanisms.

User-behavior heuristics. As we can completely bypass the speed heuristic, we repeat *Experiment B* to identify the limit of acceptable check-ins. The results reveal that if this threshold is exceeded, an exact 24 hour ban is applied. Figure 2 compares the check-ins allowed by Foursquare and Places for a time window between check-ins of 180 seconds. We also identify that the threshold is not constant for different intervals between check-ins. We repeat the experiment with different time intervals T for 24 hours each. Subsequently, we calculate the expected check-ins, which are the ideal number of check-ins that should be successful if no heuristic is applied, versus the actual successful ones. As shown in Figure 4, prior to approximately $T < 300$ seconds, the actual check-ins are less than the expected ones, since the heuristic is triggered and a 24 hour ban is applied.

7. ATTACKING LBS

An adversary with the knowledge of the detection mechanisms can create an adaptive attack that maximizes its impact while remaining undetected. In the case of Foursquare a potential attacker would try to acquire the mayorship in top venues around the world, discouraging other users from competing. We focus on Foursquare, because there is clear notion of game incentives. However, it is quite generic and, with minor modifications, can be adapted to other LBS. The following aspects led to the design of our strategy.

Timing between check-ins. Foursquare restricts the number of check-ins allowed, based on a 24 hour sliding window. If a check-in receives a "rapid-fire" error, it is not eligible for points but still counts as one of the check-ins allowed. The attacker must keep account of the timestamps of his check-ins to calculate their number in the last 24 hours. As long as the count of check-ins of the last 24 hours is 30 (since 90 are allowed in 72 hours), no check-ins must be attempted. Once the number reaches 29 he can check in once again. This way, one can commit 1,800 check-ins in a 60 day period.

Minimizing necessary check-ins. Checking into venues after having been crowned the mayor, results in unnecessary

check-ins that should be used for other venues. The adversary can stop checking into a venue once he has acquired the mayorship, and only resume if he temporarily loses it.

Only check-ins that do not trigger one of the heuristics are considered valid and can result in mayorships. We design a strategy to use the limited number of check-ins effectively. Its pseudo-code is presented in Figure 5. The system takes as input a list of N arbitrary venues. A counter c holds the number of check-ins made in the last 24 hours. After selecting the next venue from the list, we retrieve the number of check-ins n the mayor of that venue has. If we are mayors of the venue, we do not check-in and add the venue to the end of our list. If we are not, we check-in and save the relevant information (line 1). We increase the number of check-ins and then run the function to *adjust* the waiting time (line 2): If we haven't reached the MAX allowed check-ins for the last 24 hours, we set t to our normal small sleep interval. If we have reached MAX check-ins, we retrieve the info of the check-in which is located $MAX - 1$ positions from the end. We calculate how much time t our system has to sleep so it "wakes up" 24 hours from that check-in. We decrease our counter, sleep for t, and add the venue to the end of the list.

An adversary with the goal of disrupting the system and deterring legitimate users from participating, will target the most popular venues as this will impact the largest number of users. While obeying the thresholds, in the worst case scenario where each mayorship requires 60 check-ins, the attacker can acquire the mayorship of 30 venues with a single account. We collected the number of check-ins of the mayors of 2,420 of the most popular venues in New York through the API function that returns the most popular venues for a given location. 90% of the venues had a mayor with 36 or less check-ins, and only 2.2% had over 50 check-ins. The average number of check-ins required for mayorship was 17. While it might be higher than the average across all venues, since it reflects activity for popular venues in a metropolitan area, it provides a rough estimation of the average number needed to acquire a mayorship. Thus, an attacker following our attack algorithm can use the 1,800 available check-ins to sustain mayorship in 105 venues, on average, with one account. Based on that, an attacker can maintain constant mayorship in all venues with less than 10,000 accounts.

Verisign released a report about a cybercriminal selling 1.5 million Facebook[2] accounts [1],and the cost of 1,000 accounts without any contacts was $15. For compromised accounts with friends the price ranged from $25 - $45. Assuming such prices are representative, an attacker can acquire the needed number of accounts to sustain mayorship across all Foursquare venues with as little as $150 - $450. Furthermore, Trend Micro released a report [12] about the Russian underground where 2,000 bots can be bought for $200. That number of bots is more than enough for deploying the 10,000 accounts. *Overall, an attacker with the knowledge of the detection heuristics and their respective thresholds, can acquire mayorships across all venues and have a significant impact on Foursquare with less than $1,000.* Similarly, any LBS can be severely damaged with minimal resources.

As the attack is carried out by multiple accounts with legitimate behavior, each targeting a small subset of venues, Foursquare will not be able to distinguish them from other accounts. Even if the heuristics are made more restrictive, the attack variables can easily be modified to remain beneath the new thresholds. Making the heuristics too strict, will have a negative impact as legitimate users will be greatly inconvenienced. Thus, it is evident that detection heuristics are not effective against large-scale fake-location attacks and other types of countermeasures must be implemented.

[2]Foursquare allows to sign-up with a Facebook account.

8. COUNTERMEASURES

Fake-location attacks are possible because clients can communicate an arbitrary geographical position to the service. First, we propose three countermeasures that can hinder attacks by validating the user's location. Next, we discuss the inefficiency of detection mechanisms.Finally, we present our proof-of-concept implementation of Validated Check-in.

8.1 Validating user location

Ensuring user presence. One approach is to enforce verification based on information provided only at a geographical position. By requiring users to also submit information that is only available at a location, the service can validate the user's presence. One can take advantage of the NFC capabilities of smartphones, which enable communication between devices within a very short range (i.e., a few centimeters). By deploying a NFC device at venues, the LBS can validate user check-ins. Interestingly, Foursquare recently introduced unpowered NFC tags to identify the venue and prompt the user to check-in [6]. This minimizes interaction as users need only swap their device over the tag. By building upon this idea, we can hinder fake-location attacks.

Temporary codes. The service can generate a temporary code for venues that are valid for certain time (e.g., one day), and are only obtainable at the venues. This can be a string, a QR code,or even a NFC tag. While this method has the advantage of not requiring dedicated hardware,it is susceptible to *wormhole attacks* [23] where users share the code with other individuals that will be able to check-in without actually visiting the location. Another drawback is that it can only be used in commercial venues and not public places. Alternatively, a location proof scheme that uses existing access points can be implemented [27, 32]. However, such solutions can be bypassed from users that are within the range of the access point, but not at the actual venue.

Third party verification. Currently, a client's location can be verified by telecommunication providers (they know the cell the device is connected to), and large IT vendors, such as Google, that have constructed extensive maps of wireless access points around the globe. However, the location information is not accurate enough to verify the user's presence within a venue, but only within a larger area.

8.2 Adapt existing detection mechanisms

As we demonstrated, existing heuristic implementations are either too relaxed (Foursquare) or can be bypassed (Places). However, even with more restrictive thresholds, such mechanisms cannot prohibit cheating. Furthermore, they cannot defend against system-wide threats carried out by multiple accounts that are indistinguishable from legitimate ones. On the contrary, our NFC countermeasure can effectively hinder such attacks and provides an affordable solution for sustaining the viability of the emerging business model of LBS.

Penalties for cheating. This is an efficient mechanism for discouraging legitimate users from cheating, but is ineffective against system-wide threats. Due to false positives,

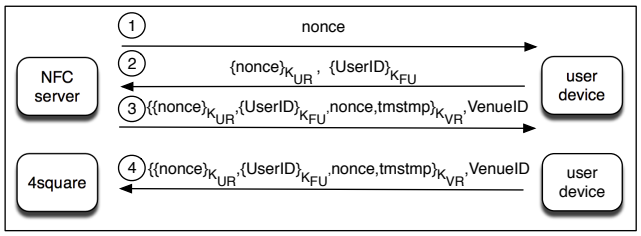

Figure 6: Data exchanged during the check-in.

penalties should be imposed when users repeatedly trigger the heuristics, in which case they should face timeout periods where no check-ins are accepted.If the cheating persists, the user should be permanently banned from the service.

Revocation. As a measure of preventing users from acquiring mayorships through fake-location attacks, Foursquare has introduced a feature that allows venue owners to revoke the mayorship of users which may have cheated. This mechanism does not assist in identifying or preventing fake-location attacks, but in discouraging potential cheaters. This feature might be effective in certain cases, yet there are many conditions where it is not applicable, like in venues with many simultaneous customers (e.g., clubs, shopping malls).

8.3 Implementation of Validated Check-in

We present the details of *Validated Check-in*, our proof-of-concept implementation of the NFC server countermeasure. Affordable electronics frameworks are a rapidly growing market, and we selected two of the most popular devices. First, the Arduino board, an open source electronics prototyping platform, which can be extended through various modules that provide specific functionality. We used Arduino Uno and the Seeedstudio NFC Shield with a total cost of about $50 at retail price. Second, the Model B Raspberry-Pi, an ARM GNU/Linux box, using the Adafruit NFC breakout board with a total cost of $75. To evaluate our testbeds, we developed an Android application which implements the user functionality. The application communicates with the NFC Server, using classes from the `android.nfc` package. Arduino was programmed with the LLCP-SNEP protocol implementation for P2P communication. For the Raspberry-Pi we used the `libnfc` and `openssl` libraries. Our user device was a Samsung Galaxy S3.

Our solution relies on cryptographic primitives for securing communication between the NFC server and user device and prevents different types of attacks. Upon activation of the mobile application, venues and users calculate a set of asymmetric keys. Foursquare must receive a copy of the public keys, and all venues and users save a copy of Foursquare's public key. After setting up the venue account, copying the keys on the NFC server and synchronizing the internal clock through NTP, no Internet connectivity is required.The UserID and VenueID are already used by Foursquare, as parameters in the API calls.

Validated Check-in: the protocol is shown in Figure 6.

1. The NFC server sends a random nonce to the user.
2. The device encrypts the nonce using the user's private key K_{UR}, and the UserID using Foursquare's public key K_{FU}, and sends them both to the NFC server. If they are not received by the server in an acceptable time window, it terminates the process.
3. Using the venue's private key K_{VR}, the NFC server re-encrypts the encrypted UserID and nonce along with the nonce in cleartext and a timestamp. These, as well as the VenueID in cleartext, are sent to the user device.
4. The device sends the data to Foursquare, that uses the VenueID to retrieve the venue's public key K_{VP} and decrypt the ciphertext. If the timestamp is valid, Foursquare uses its private key to decrypt the UserID and verifies it is that of the user that sent the check-in. Then it uses the user's public key to decrypt the nonce value. If it matches the one sent by the venue, the user is checked-in.

Security Analysis of the Validated Check-in protocol.

Fake-location attacks: the attacker creates a bogus check-in request for a venue, while being at a different location. However, during the normal check-in process the information sent to Foursquare contains a timestamp encrypted using the venue's private key. Thus, the user cannot create a valid check-in (even with information from old check-ins).

Wormhole attacks: an attacker located at a venue exchanges information with an accomplice, so he can perform a check-in as well. The NFC server completes the protocol only if the user sends the response within the acceptable time window. However, the nonce cannot be predicted and the challenge can't be relayed in time. The attacker can also use a device that mimics an NFC server and try to check the user into a venue X other than the one he is at. Again, the NFC server at venue X will not receive the challenge in time.

Eavesdropping attacks: an eavesdropper passively monitors the communication between the users and the NFC server to discover the users' identity. However, he is only able to acquire the VenueID which is publicly known, since the UserID is encrypted with Foursquare's public key.

System-wide Sybil Attacks: the attacker aims to disrupt the whole system and drive legitimate users away by acquiring the mayorship of all (or most) venues using multiple accounts [20]. As shown in Section 7, with the existing defense mechanisms, this can be done with minimal resources. With our countermeasure this attack can be deterred, as our NFC server imposes physical constraints on the check-in process. The attackers will have to physically visit each venue (practically impossible) to perform the check-ins.

Targeted Sybil Attacks: the attacker targets a few venues that might offer deals to all customers after a certain number of check-ins. The attacker can utilize several accounts to collect multiple offers. Even with our countermeasure deployed, one can still perform this attack. This can be done with a smartphone that contains the passwords and keys of all the attacker's accounts (or those of accomplices). Nonetheless, our countermeasure will be able to greatly mitigate such an attack, as it imposes physical and time constraints on the check-in process. Due to the NFC technology, the attacker will have to stay next to the server for a unnatural amount of time to check-in a large number of accounts.

Performance analysis of our implementations.

Encryption: an important factor that affects the applicability of our solution is the time needed for the data encryption. Table 2 presents the average times (over 100 runs) for applying RSA encryption to a buffer using keys of various sizes. The Arduino presents the worst performance due to

Keysize	512	1,024	2,048	4,096
Arduino	25,056	224,279	1,587,550	NA
Rasp.Pi	2.745	3.228	5.130	12.150
Galaxy	2.265	2.834	5.042	12.501

Table 2: Average encryption time (ms) for different RSA key sizes (bits).

its limited computational capabilities and RAM size, with similar times to those reported in [13]. Even for small keys, the time required for the user device and NFC server to stay in range is not acceptable in realistic scenarios. For a 512-bit key (which is very weak), the encryption process takes 25 seconds. Thus, the Arduino board is not a suitable solution. On the other hand, the Raspberry-Pi server is very efficient and even with 2,048-bit keys encryption takes merely 5 ms.

Check-in process: Currently, Android requires the user to tap the screen for authorization before data is sent over NFC, and multiple messages can only be sent in batches. As the NFC server has to enforce a strict time window for the challenge-response step, no user interaction should be needed between steps (1) and (2), as that would result in a window large enough for a wormhole attack. However, the data sent in step (2) of our protocol is based on the data received in step (1) and can't be sent in a batch. Thus, the user is required to tap the screen a second time. This is very restrictive for building NFC apps, as it does not allow multiple steps of communication between devices. This has been reported by developers [3], was acknowledged by Google, and is awaiting a fix. To overcome this limitation, we also implemented a version that uses NFC to pair the devices, and sends the protocol data over Bluetooth. This version only requires one tap to initiate the pairing and everything else is done automatically. Nonetheless, we expect this to be fixed soon, enabling our NFC-only approach.

In the Bluetooth version, using 2,048-bit keys and a 32-byte nonce, the entire check-in process lasts 105 ms (average over 100 runs). Based on the encryption times and the time needed to send the data (28.7 ms per message), we set the time window for the challenge-response step to 45 ms. After sending the nonce, the NFC server terminates the process if the response is not received within 45 ms. We plan on conducting a study using a variety of smartphones to calculate the time needed for the encryption, to select a value suitable for real-world deployment. Overall, the performance of Validated Check-in renders it an ideal solution for LBS.

LBS workload: public-key cryptography is considered computationally expensive. Foursquare must validate thousands of check-ins per minute, as users conduct a few million per day. Fortunately, hardware acceleration for cryptographic operations has evolved. Consider that 7 years ago [11] Sun's UltraSPARC T1, equipped with a Modular Arithmetic Unit for RSA, performed 20,425 signature verifications per second with a 2,048-bit key utilizing all 32 cores. Decryption can be further sped up by using GPUs [24] or modern `x86` CPUs, with operations needed by cryptographic algorithms implemented in the hardware, and encryption can be handled at line speeds [25]. Others [14] also argue that cryptographic operations at line speeds are no longer an issue.

Cost: components to build our system cost $75 at retail prices. While this might seem high, a LBS can purchase bulk quantities of the components at much lower prices. Thus, they can provide Validated Check-in to collaborating venues for a very low price, with the ultimate benefit of imposing fairness which will ensure a robust check-in economy.

9. RELATED WORK

Joining LBS. Researchers have tried to understand the motives of users that join systems like Foursquare. Their findings suggest that there is a significant portion of users that participate for the discounts and special offers [26]. This is also supported by [30], where nearly 20% of the test-subjects reported that offers were an important reason for participating. Cramer et al. conduct a user survey [18, 19], and find that both the "gaming" aspect and venue offers are significant incentives for user participation.

Attacking LBS. One of the first to raise awareness about the implications of fake-location attacks against LBS was [22]. Even though part of this paper focuses on the same problem, our work presents several characteristics that differentiates it. While He et al. refer to certain heuristics used by Foursquare, they have not explored them in depth so as to identify their thresholds. We follow a systematic black-box testing approach that accurately identifies the thresholds for each heuristic. Furthermore, they develop a semi-automatic tool that demonstrates that fake check-ins are feasible, but due to the lack of knowledge of thresholds, cannot demonstrate the true extent of potential attacks. Our automated tool systematically explores the detection mechanisms, reveals the extent to which attacks are feasible and highlights their true impact. Finally, our adaptive attack uses information at runtime to avoid redundant check-ins.

Validating User Location. Carbunar et al. [17] present a mechanism that allows users of LBS to communicate with the service in a private manner. However, their solution for validating the user's location, is susceptible to wormhole attacks. In their follow-up paper [16] they propose two extensions for detecting wormhole attacks. While the first approach called NES is somewhat similar to our NFC countermeasure, it presents disadvantages. First, the LBS is required to keep state regarding each check-in attempt by a user, as it has to store the nonce values sent to each user (apart from the keys of all venues and users as we do). It also requires the venue to be able to communicate with the service, while our approach works even without Internet connectivity. The second countermeasure, WES, relies on a WiFi router present at the location that periodically changes its SSID to a value that the user will forward to the LBS. However, there is no description of the process by which the LBS will predict the SSID values of each venue's router at a given time, or the extra state the service will have to store. Finally, neither solution has actually been implemented.

The Echo protocol [34] is for securely verifying location claims, using a time-of-flight approach. This protocol does not require a setup or registration step, which excludes cryptographic operations, and can be deployed for various applications. However, this solution requires devices that can emit both radio and ultrasound frequencies, while we take advantage of the NFC capabilities present in many modern smartphones. Furthermore, for large or non-circular areas, multiple nodes are required. On the other hand, we target a specific deployment scenario, for LBS to ensure user presence at a specific venue during the check-in process.

[28] presents secure protocols for private proximity testing, where two friends can be notified when within a spe-

cific distance of each other, while their location remains secret. Extensive work has been published regarding distance bounding protocols (e.g., [31]), for verifying a user's position. In [21], the RFID distance bounding protocol assumes that the prover does not collude with a third party closer to the verifier and is, thus, vulnerable to wormhole attacks.

10. CONCLUSION

Using a black-box testing approach we revealed the server-side heuristics employed by Foursquare and Places for detecting fake-location attacks and discovered their thresholds. Our technique, without prior knowledge, revealed a bug that allows one to bypass speed constraints set by Places. We also presented an algorithm that leverages all discovered thresholds of Foursquare for maintaining mayorship in venues. Finally, we implemented Validated Check-in, an NFC server that can eliminate system-wide threats, analyzed its security properties and evaluated its performance. We believe that user coordinates will become a prominent feature of Internet usage in general and, quite possibly, of critical services. Thus, our research goals were twofold. First, to demonstrate the extent to which LBS are vulnerable and the true impact of fake-location attacks, so as to draw the attention of the research community. Second, to expose the inefficiency of anomaly detection mechanisms and utilize our findings for designing and implementing an effective countermeasure.

11. ACKNOWLEDGMENTS

This research was performed with the financial support of the Prevention of and Fight against Crime Programme European Commission Directorate-General Home Affairs (project GCC). This publication reflects the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein. This work was also supported in part by the FP7-PEOPLE-2010-IOF project XHUNTER, No. 273765, and SysSec, funded by the European Commission under Grant Agreement No. 257007.

12. REFERENCES

[1] 1.5 million facebook accounts offered for sale. http://www.zdnet.com/blog/security/1-5-million-facebook-accounts-offered-for-sale-faq/6304.

[2] American Express discounts in FourSquare. https://sync.americanexpress.com/foursquare/.

[3] Android issues: Enable real nfc p2p communication. http://code.google.com/p/android/issues/detail?id=28014.

[4] Facebook developers - bugs. https://developers.facebook.com/bugs/244713388933143?browse=search_4f12b26febf840e00208758.

[5] Foursquare - follow-up to "mayorships from your couch" post. http://blog.foursquare.com/2010/04/08/505862083/.

[6] Foursquare adds nfc support to its android app. http://techcrunch.com/2012/02/10/foursquare-adds-nfc-support-to-its-android-app/.

[7] Foursquare CEO: 'Not just check-ins and badges.'. http://money.cnn.com/2012/02/29/technology/foursquare_ceo/.

[8] How does foursquare handle cheating? http://support.foursquare.com/entries/188307.

[9] Mayor of the north pole. http://krazydad.com/blog/2010/02/15/mayor-of-the-north-pole/.

[10] On foursquare, cheating, and claiming mayorships from your couch. http://blog.foursquare.com/2010/04/07/503822143/.

[11] Rsa performance of sun fire t2000. http://blogs.sun.com/chichang1/entry/rsa_performance_of_sun_fire.

[12] Russian underground. http://www.trendmicro.com/cloud-content/us/pdfs/security-intelligence/white-papers/wp-russian-underground-101.pdf.

[13] Arkko, J., Keranen, A., and Sethi, M. Practical considerations and implementation experiences in securing smart object networks, 2012. http://tools.ietf.org/html/draft-aks-crypto-sensors-00.

[14] Bittau, A., Hamburg, M., Handley, M., Mazieres, D., and Boneh, D. The case for ubiquitous transport-level encryption. In *Proceedings of the 19th Conference on USENIX Security Symposium* (2010).

[15] C. Gerlitz and A. Helmon. Hit, link, like and share. Organizing the social and the fabric of the web in a like economy. Presented at the DMI mini-conference, volume 24, 2011.

[16] Carbunar, B., and Potharaju, R. You unlocked the mt. everest badge on foursquare! countering location fraud in geosocial networks. In *IEEE 8th International Conference on Mobile Adhoc and Sensor Systems* (2012), MASS, IEEE.

[17] Carbunar, B., Sion, R., Potharaju, R., and Ehsan, M. The shy mayor: Private badges in geosocial networks. In *ACNS* (2012), vol. 7341 of *Lecture Notes in Computer Science*, Springer.

[18] Cramer, H. Gamification and location-sharing : emerging social conflicts. *Proceedings of ACM CHI Workshop on Gamification* (2011).

[19] Cramer, H., Rost, M., and Holmquist, L. E. Performing a check-in: emerging practices, norms and 'conflicts' in location-sharing using foursquare. In *Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services* (2011), MobileHCI, ACM.

[20] Douceur, J. R. The sybil attack. In *the First International Workshop on Peer-to-Peer Systems* (2002), IPTPS '01'.

[21] Hancke, G. P., and Kuhn, M. G. An rfid distance bounding protocol. In *Security and Privacy for Emerging Areas in Communications Networks, 2005. IEEE SecureComm* (2005).

[22] He, W., Liu, X., and Ren, M. Location cheating: A security challenge to location-based social network services. In *Proceedings of the 2011 31st International Conference on Distributed Computing Systems* (2011), ICDCS '11.

[23] Hu, Y.-C., Perrig, A., and Johnson, D. B. Packet leashes: A defense against wormhole attacks in wireless networks. In *INFOCOM* (2003).

[24] Jang, K., Han, S., Han, S., Moon, S., and Park, K. Sslshader: cheap ssl acceleration with commodity processors. In *Proceedings of the 8th USENIX conference on Networked systems design and implementation*, NSDI'11.

[25] Kounavis, M. E., Kang, X., Grewal, K., Eszenyi, M., Gueron, S., and Durham, D. Encrypting the internet. In *Proceedings of the ACM SIGCOMM 2010 conference*, SIGCOMM '10', ACM.

[26] Lindqvist, J., Cranshaw, J., Wiese, J., Hong, J., and Zimmerman, J. I'm the mayor of my house: examining why people use foursquare - a social-driven location sharing application. In *Proceedings of the 2011 annual conference on Human factors in computing systems*, CHI '11, ACM.

[27] Luo, W., and Hengartner, U. Veriplace: a privacy-aware location proof architecture. In *Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems* (2010), GIS '10', ACM.

[28] Narayanan, A., Thiagarajan, N., Lakhani, M., Hamburg, M., and Boneh, D. Location privacy via private proximity testing. In *NDSS* (2011).

[29] Noulas, A., Scellato, S., Mascolo, C., and Pontil, M. An empirical study of geographic user activity patterns in foursquare. In *ICWSM* (2011).

[30] Patil, S., Norcie, G., Kapadia, A., and Lee, A. J. Reasons, rewards, regrets: privacy considerations in location sharing as an interactive practice. In *Proceedings of the Eighth ACM Symposium on Usable Privacy and Security*, SOUPS '12.

[31] Rasmussen, K. B., and Čapkun, S. Realization of RF distance bounding. In *Proceedings of the 19th USENIX conference on Security* (2010).

[32] Saroiu, S., and Wolman, A. Enabling new mobile applications with location proofs. In *Proceedings of the 10th workshop on Mobile Computing Systems and Applications* (2009), HotMobile '09', ACM.

[33] Saroiu, S., and Wolman, A. I am a sensor, and i approve this message. In *Proceedings of the Eleventh Workshop on Mobile Computing Systems & Applications* (2010), HotMobile, ACM.

[34] Sastry, N., Shankar, U., and Wagner, D. Secure verification of location claims. In *Workshop on Wireless Security* (2003).

[35] Tippenhauer, N. O., Pöpper, C., Rasmussen, K. B., and Capkun, S. On the requirements for successful gps spoofing attacks. In *Proceedings of the 18th ACM conference on Computer and communications security* (2011), CCS, ACM.