You Unlocked the Mt.Everest Badge on Foursquare! Countering Location Fraud in GeoSocial Networks

Bogdan Carbunar
School of Computing and Information Sciences
Florida International University, Miami, FL
Email: carbunar@cs.fiu.edu

Rahul Potharaju
Department of Computer Science
Purdue University, West Lafayette, IN
Email:rpothara@purdue.edu

***Abstract*—GeoSocial Networks (GSNs) are online social networks centered on the location information of their users. Users "check-in" their location and use it to acquire location-based special status (e.g., badges, mayorships) and receive venue dependent rewards. The strategy of rewarding user participation however makes cheating a profitable behavior. In this paper we introduce XACT , a suite of venue-oriented secure location verification mechanisms that enable venues and GSN providers to certify the locations claimed by users. We prove that XACT is correct, secure and easy to use. We validate the need for secure location verification mechanisms by collecting and analyzing data from the most popular GSNs today: 780,000 Foursquare users and 143,000 Gowalla users. Through a proof-of-concept implementation on a Revision C4 BeagleBoard embedded system we show that XACT is easy to deploy and economically viable. We analytically and empirically prove that XACT detects location cheating attacks.**

I. INTRODUCTION

Online social networks are tools that allow users to connect and maintain contact with friends and family. Geosocial Networks (GSNs) extend online social networks with, and center their functionality on the *location* of their users. Location is shared by subscribers with their friends, then used by GSN providers to enable targeted advertising and by venue owners to promote their businesses through spatio-temporal incentives. Many GSN providers have emerged in the past few years, including the popular Foursquare [1], Gowalla [2], Yelp [3] and Facebook Places [4].

Most GSNs provide similar functionality: Users *check-in* at *venues* where they are present, effectively reporting their location to the geosocial network provider. As a reward, users receive *badges* and *mayorships* (or *virtual items* in Gowalla) as well as financial rewards. Franchises like Ann Taylor, GAP, Lufthansa, Starbucks and Pizza Hut have modified their business model to offer substantial discounts to users performing frequent check-ins. The use of incentives however introduces reasons for cheating, motivating users to commit *location fraud*: falsely claim to be at a location, to receive undeserved rewards or social status. Even with GPS verification mechanisms in place, committing location fraud has been largely simplified by the recent emergence of specialized applications for the most popular mobile eco-systems (LocationSpoofer [5] for iPhone and GPSCheat [6] for Android) [1]. Such behavior places undue burden on participating venues, as proved by the recent surge in the numbers of fake check-ins and "*instant*" mayors [8].

Data we have collected from more than 780,000 Foursquare users and the entire Gowalla user set (143,000 users) confirms the impact of this problem: GSN users are actively checking-in and collecting badges, and many venues record tens of daily check-ins. Thus, contention and hence cheating incentives do exist, making it necessary to carefully balance incentives with more effective verifications of user location claims.

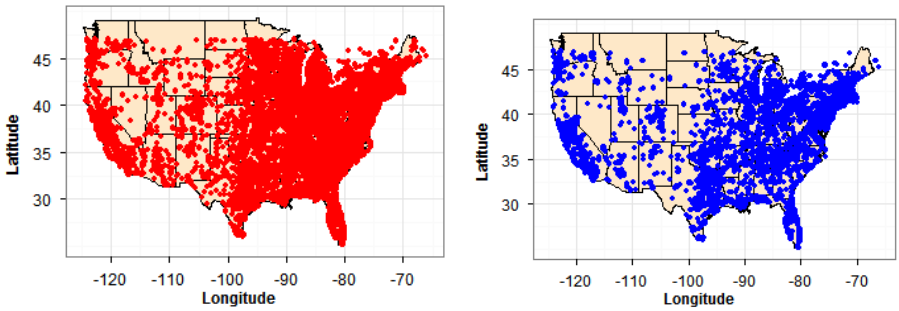
To address this problem, we exploit the insight that venues have the most to gain from properly rewarding users – their main goal is to retain customers and attract new users. We introduce then XACT , a suite of venue-oriented, secure location verification mechanisms, that require participating venues to deploy minimalist equipment. To promote its adoptability, we design XACT to be not only secure and correct, but also user friendly, economical and easy to deploy. XACT consists of mechanisms that (i) broadcast unpredictable Wi-Fi SSIDs, (ii) display QR codes encoding venue certified information, and (iii) implement challenge/response protocols.

Besides securing the reward systems of participating venues, XACT can also be applied to detecting fake reviews in review-centered geosocial networks like Yelp [3] and TripAdvisor [9]. Since users need to have been present to review a venue, location verification may be the first step in identifying suspicious reviews. Furthermore, XACT can also be used to enable users to validate their location-centric tweets.

We propose a proof-of-concept implementation on a Revision C4 BeagleBoard [10] embedded system, to shows that the cost imposed on venues is small and a one time effort (no monthly fees). We prove that XACT requires at least one attacker to be present at the venue and show that it detects wormhole attacks by imposing noticeable overheads on attackers - up to 12 times higher than on honest users.

The paper is organized as follows. In Section II we present the system model, organized around the data collected from Foursquare and Gowalla, we describe the attacker model as well as the requirements of the solution and the used tools. In Section III we introduce XACT , and prove its correctness and security. In Section IV we describe our proof-of-concept prototype and analyze XACT 's wormhole attack prevention ability. In Section V we discuss related work, extend it

[1]In fact, He et al. [7] proved the feasibility of fake check-ins in Foursquare



(a) (b)

Fig. 1. Properties of our diastase (a) **Geographical distribution of Foursquare users**: Foursquare is most popular in the eastern half of the United States with New York being the most popular city, (b) **Geographical distribution of Gowalla users**: Exhibits similar properties as Foursquare though not as densely covered.

and apply it in the context of geosocial networks. Finally, Section VI concludes.

II. Architecture and Model

The geosocial network (GSN) consists of a provider, S, hosting the system and serving a number of subscribers. To use the provider's services, a client application needs to be downloaded and installed. Subscribers can then register and receive initial service credentials, including a unique user id; let Id_A denote the id of user A. In the following we use the terms *user* and *subscriber* to refer to users of the service and the term *client* to denote the software provided by the service and installed by users on their devices.

A. Dissecting GeoSocial Networks

In the following, we model the online geosocial network provider S after Foursquare [1] and Gowalla [2], the most popular in existence to date. Foursquare provides a touch of "gamification" to location based services: The users report their location, through *check-ins* at venues of interest, share it with friends and are awarded "points" and "badges" (e.g., "Adventurer", "Explorer", or "Superstar"). A user earns a badge when it accumulates a certain number of check-ins, at the same or different venues. Badges are called "pins" in Gowalla. The user with the most check-in days at a venue for a consecutive chain of 60 days becomes the "Mayor" of the venue. Foursquare has partnered with a long list of venues (bars, cafes, restaurants, etc) to reward "check-in" users with freebies and specials. This strategy has made Foursquare very popular, with a constantly growing user base, which we currently estimate at over 14 million users, increasing at a rate of almost 40 users/min.

Venues and Check-ins: The provider supports a given set of locations, defined in terms of discrete points-of-interests (POIs) or sites: restaurants, dentist offices, etc. During a check-in, the user's application (client) captures the GPS location and displays a list of close-by venues – the user can choose one. In the following, we use the term *check-in venue* to refer to a venue where a check-in is claimed to be performed. We call a *fake check-in* to be a check-in performed when the user is not physically located at the check-in venue.

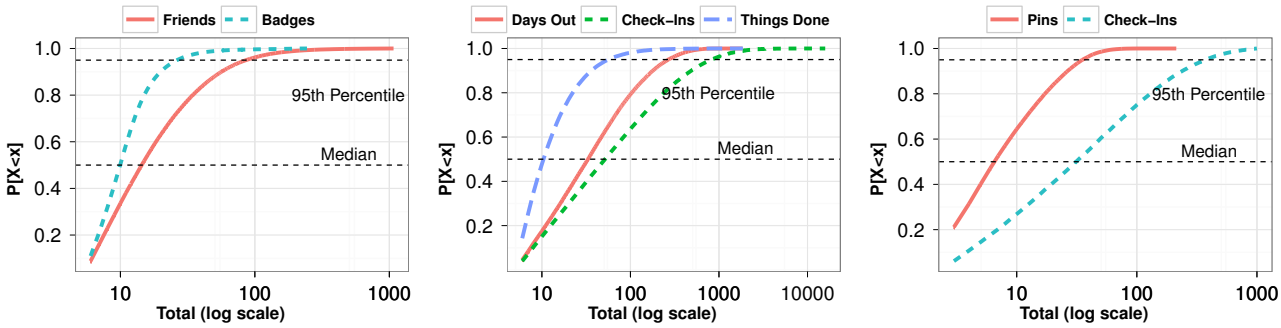
Location Verifications: An excellent example of security by obscurity, location verification mechanisms are kept secret by GSN providers. However, once attackers discover the nature and parameters of these verifications, they can easily circumvent them. Based on our experience with Foursquare, we conjecture that the following are among their verification mechanisms:

- **GPS Verification**: During a check-in, the Foursquare app uses the device's GPS to only display close-by venues. This method can be circumvented with third-party software like GPSCheat [6] or by hijacking the GPS module of the smartphone [11] using rootkits.
- **Auto-Excluding Venues**: To prevent multiple check-ins, venues around the user's previous check-in venue are filtered out during immediately subsequent check-ins.
- **Epoch Based Check-ins**: To prevent a user from checking-in at the same venue multiple times within a short interval, the GSN provider divides time into epochs (e.g., one day for Foursquare) and limits the number of check-ins per client per epoch per any site. However, if the value of the epoch is leaked, the attacker can follow a greedy strategy to check-in as many times as the epoch permits.
- **Obeying Laws of Physics**: GSNs verify that the distance and time between per-user consecutive check-in venues are consistent with the laws of physics: the distance can be physically traversed within the recorded time interval.

Datasets. To confirm the relevance of location verification mechanisms, we needed to understand if geosocial network subscribers are active in terms of numbers of check-ins performed, badges obtained, users befriended and things done at locations. We have collected publicly available data from Foursquare and Gowalla[2] using their public APIs. We have collected profiles of 781,239 Foursquare users (out of 5 million queried) and the entire Gowalla set – 143,476 users. For every Foursquare/Gowalla user, we have gathered the user profile, the total number of friends, check-ins and "days out" (days the user was actively performing check-ins).

The data collected from Foursquare and Gowalla lacks

[2]Note that as of September 24, 2011, gathering this kind of data is no longer possible as Gowalla changed its user interface. "Check-in" related information is no longer shown on Gowalla website



(a) (b) (c)

Fig. 2. Properties of our datasets of 781,239 users of Foursquare and 143,476 users of Gowalla. (a) **Foursquare Friends and Badges Distribution**: people are actively participating and making friends, (b) **Foursquare Days-Out, Check-Ins and Things-Done Distribution**: The 95th percentile of Days-Out lying around 200 agrees with the recent increase in Foursquare marketing campaigns. (c) **Gowalla Pins and Check-Ins Distribution**: Indicates that the user base of Gowalla is equally active.

per-user location information for the venues where check-ins were performed. Therefore, we have also used the TAMU[3] dataset [12] in our analysis. The TAMU dataset was collected from Twitter [13] and includes information from Foursquare users who have tweeted about their check-ins; it contains per-user information on check-in venues. The dataset contains 224,804 users and a total of 22,387,930 check-ins/tweets. Each tweet entry contains a user id, the time of occurrence as well as the venue id along with its latitude and longitude.

Figures 1(a) and 1(b) show the geographical distribution of the user "home cities" in the US subset of the Foursquare and Gowalla datasets. It proves that not only the coasts but also the entire eastern half of the US is actively using Foursquare. These results are in agreement with results from Cheng *et al.* [12] and Cha *et al.* [14]. We observe that as expected, the Gowalla distribution is similar, but not as dense as that of Foursquare.

Furthermore, Figure 2(a) shows the cumulative distribution function (CDF) in Foursquare for the number of friends per active user (user with at least 4 check-ins) and the number of badges per user: 45% of users (between the median and the 95-th percentile) have between 20 and almost 100 friends and between 10 and 40 badges. Figure 2(b) shows the CDF for the check-ins, days-out and things-done of the same user set. 45% of the population is very active - having performed between 80 and 950 check-ins and done between 10 and 80 "things" in 50 to 200 days of activity. Figure 2(c) shows a similar situation in Gowalla: half the users have more than 8 pins (check-ins) and more than 50 friends.

For the same user sets, Figure 3(a) shows the scatter plot of check-ins versus days out where each point represents a single Foursquare user. Figure II-A shows a similar graph for Gowalla users. Note that a user can perform multiple check-ins in a single day. The graph shows that the number of check-ins increases rapidly with the number of days-out indicating that most users visit multiple places on any given day.

Figure 3(b) shows the scatter plot of check-ins vs. users in Foursquare, for the city of Babylon in Long Island, NY, containing 14,958 active users out of a total of 58,093 users. The plot indicates the presence of hotspot locations - the few points in the tail end of the graph indicate that these hotspots were visited by 800-1500 users and were reported in up to 5000 check-ins. The locations in the center are representative of an average location.

Conclusions: GSNs are picking up in popularity: users are willing to report their location information and are actively collecting location-based rewards. Moreover, many venues record very high number of check-ins, in possibly a contest for acquiring mayorships and badges. Competition and financial incentives motivate dishonest behavior, making fraud-proof location verification mechanisms a necessity.

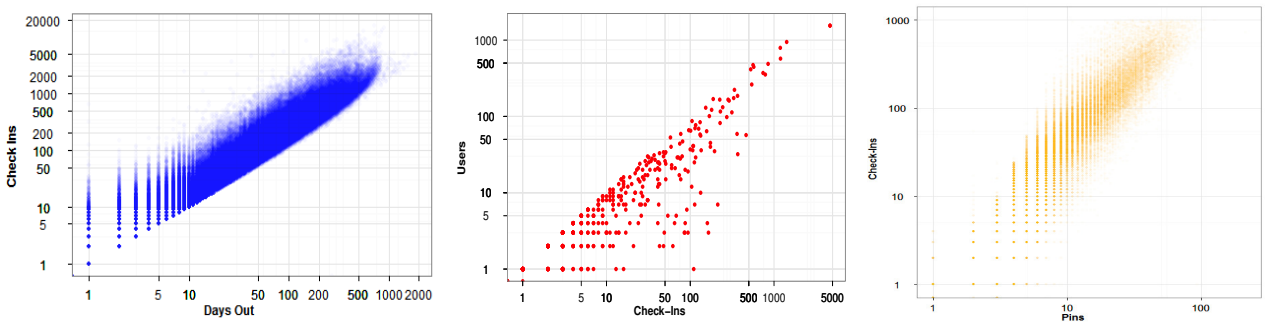
B. Attacker Model

We assume the clients are malicious. Malicious clients can be installed by outsiders that are able to corrupt existing devices or may be installed by insiders, or system subscribers. We assume malicious users/clients can collude to improve the success probability of their attacks. However, we assume colluding adversaries do not share secret keying information, considered to be private. Malicious clients can launch *location based* attacks: Clients cheat on their location by claiming to be in places where they are not [6]), claiming to have location based information they do not have or incorrectly claiming to be in the vicinity of other users. Of particular interest are *wormhole* attacks. In a wormhole attack, an attacker present at a venue captures protocol packets, communicates them over alternative channels to remote collaborators – enabling all attackers to prove presence at the venue.

C. Solution Requirements

A location verification solution consists of a set of protocols $Geo = \{Setup, RegisterVenue, Subscribe, CheckIn\}$. $Setup$ is executed initially by the service provider, to generate system parameters. $Subscribe$ is executed when a user registers with the GSN provider and receives unique identifying information. $RegisterVenue$ is invoked by a user when registering a new venue with the GSN provider. $RegisterVenue$ returns -1 if the operation fails and 0 otherwise. $CheckIn$ is

[3]We thank the authors for sharing their data.



(a) (b)

Fig. 3. (a) Scatterplot of Days out vs. Check-ins of 781,239 users, (b) Scatterplot for Checkin vs. Users in Babylon, NY. (c) Scatterplot of Pins vs. Check-Ins of 143,476 Gowalla users.

executed when a client reports the venue where it is currently located. $CheckIn$ returns -1 if the check-in fails and 0 if it succeeds. A solution implementing Geo needs to satisfy the following requirements:

Correctness. A check-in at a venue V succeeds with high probability if the client is located at V.

CheckIn Security. Clients are unable to perform check-in fraud. Formally, an adversary not present at a venue V, has only negligible probability in successfully completing an execution of $CheckIn$ at V.

User Friendly. The solution is easy to use.

Ease of Deployment. Minimize the investment cost for the GSN provider, participating venues and users.

D. Tools

We briefly describe several tools on which we build our solution.

QR Codes: A Quick Response code (QRC) is a two dimensional (matrix barcode) consisting of black modules arranged in a square pattern on a white background. QRCs encode information – they can store up to 2,953 bytes and are designed for fast readability.

Cryptographic Tools: We use one-more-forgery resistant signature schemes. Briefly, one-more-forgery resistance means that provided with k signatures of its choice, an adversary has only negligible advantage in convincing a challenger to accept a signature on a message that is not among the k original signatures. We rely on cryptographically secure pseudo-random number generators (CSPRNG), where an adversary not knowing the seed has only negligible advantage in distinguishing the generator's output sequence from a random sequence. We use HMACs (Hash-based Message Authentication Codes) [15] for calculating a message authentication code (MAC) involving a cryptographic hash function in combination with a secret key. As with any MAC, it may be used to simultaneously to verify both the data integrity and the authenticity of a message.

III. XACT Design

Being the most affected by fake check-ins, venue owners are the most motivated to prevent check-in fraud. In this section we introduce XACT , a venue-oriented secure location verification solution that instantiates Geo. XACT requires participating venue owners to install minimal additional equipment; we later show the investment is not only inexpensive, but also present on most recent smartphones. An important assumption that we *do not* make, is that the equipment installed has Internet connectivity and is able to communicate directly with the GSN provider. Besides making our approach completely portable (e.g., can be installed anywhere inside the venue), XACT imposes solely a one-time cost for the venue owner (no monthly fees).

A. Overview

XACT consists of three location verification mechanisms: a (i) **WiFi-enabled Embedded System** (WES), capable of switching its network card in ad hoc mode, a (ii) **Feedback-enabled Embedded System** (FES) equipped with an LCD screen and a proximity sensor or a (iii) **Network Embedded System** (NES) capable of communicating with nearby devices. Most smartphones have *all* these three capabilities and can be used as an implementation platform. In the following we use XACTR_V to refer to the device that a venue V deploys within its site to "*exact*" this mechanism. We also consider time epochs to be further divided into frames. The length of time frames is a system parameter and should be on the order of seconds.

In WES, each venue V shares an SSID string with the GSN provider S. The SSID changes frequently (once per time frame), in a manner predictable only by the venue and S. During a check-in, the client needs to scan the SSIDs in its vicinity and send the list to S. If the SSID list contains the venue's SSID for the current time frame, the check-in succeeds.

In FES, the venue displays a QR code on its LCD. The QR code encodes the venue owner's signature on a set of values intended to prevent replay attacks. During a check-in, the client takes a snapshot of the QR code displayed on the venue's LCD, extracts the encoded value and reports it to S. If the signature verifies, the check-in succeeds. Through an embedded proximity sensor, XACTR_V changes the QR code each time a user reads it.

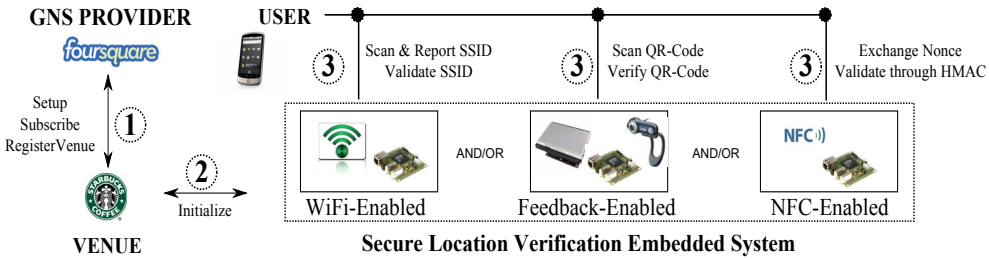


Fig. 4. System design of Secure Location Verifications: XACT can be built using one or more of the shown components offering a tradeoff between deployment cost and security.

Algorithm 1 Pseudocode of FES and NES CheckIn: XACTR_V angle.

```
1.Object implementation SPOTR_V;
2.  int T;                #timestamp in ms
3.  int ΔT;               #expiration interval
4.  int pk_V, pr_V;       #signature key pair
5.  Operation CheckIn()
6.      T := getCurrentTime();
7.      if (algoType = FES) then
8.          σ := sign(pr_V, T, ΔT);
9.          string qrCode := T + ΔT + σ;
10.         displayCode(qrCode);
11.     if (algoType = NES) then
12.         Connection con := waitForConn();
13.         con.send(T, ΔT, R);
14.         int T_1 := getCurrentTime();
15.         string msg := con.recv();
16.         int T_2 := getCurrentTime();
17.         if (T_2 − T_1 < threshold) then
18.             σ := sign(pr_V, msg);
19.             con.send(σ);
20.     fi fi
21. end
```

The main goal of NES is to prevent wormhole attacks – through local interaction between the client and XACTR_V . During a check-in, the client establishes a local connection with XACTR_V , and XACTR_V initiates a challenge-response protocol. XACTR_V certifies only the timeliness of the client's reply. The GSN provider S verifies not only the certificate issued by XACTR_V but also the correctness of the client's response. To achieve this, each client shares a unique symmetric key K with S.

B. Protocol Details

XACT instantiates *Geo*. We now formally define each protocol of XACT .

Setup$(S())$*:* The provider S generates a public and private key pair, pub_S and $priv_S$. During the installation process, each client receives S's public key, pub_S.

Subscribe$(C(pData), S(pub_S, priv_S))$*:* To register a new user account, the client C and provider S run the following two steps:

- C generates a key pair, $(pub_C, priv_C)$ and a symmetric key K. C sends personal data $pData$ including name, e-mail address and cellphone IMEI along with public key pub_C and the key K to S.
- S generates a unique client id, Id_C and a public key certificate for C, $cert_C$. $cert_C$ includes information from $pData$, as well as pub_C and a (initially empty) list of venues owned by C. S stores $cert_C$ and K indexed under Id_C, then sends $cert_C$ and Id_C to C.

RegisterVenue$(O(Id_O, cert_O, V, Loc_V), S(priv_S))$*:* To register a venue V, the owner O sends a description of the venue and its location to the provider S. S retrieves the certificate $cert_O$ stored for O and updates it to include the venue V in the list of venues owned by O. S associates a record Rec_V with each new venue V. Rec_V is initialized with the name and location of the venue and the client id of the owner, $Rec_V = [V, Loc_V, Id_O]$. S sends the updated $cert_O$ to O. The owner boots up XACTR_V and instructs it to generate a signing key pair (pk_V, pr_V). XACTR_V also generates a random value R_V, stores it along with the key pair and generates a QR code encoding the public key pk_V and random R_V. The owner uses her mobile device to take a snapshot of the QR code, decodes and retrieves pk_V and R_V, and sends them to S. S stores pk_V and R_V in Rec_V, $Rec_V = [V, Loc_V, Id_O, pk_V, R_V]$. Then, depending on the XACT component, XACTR_V performs the following steps:

- **WES:** Use R_V as a random seed to initialize a cryptographically secure pseudo random number generator (CSPRNG) G. Turn on the Wi-Fi card in ad hoc mode, and set the ssid to be $SSID_{V,f} = G_{R_V}(f) \ mod \ s$, where f is the current time frame and s is the desired bit length for the ssid string. During each new time frame, the ssid of XACTR_V changes accordingly.
- **FES:** Sample the current time T, append an expiration interval ΔT and sign the resulting string, $\sigma = S_V(T, \Delta T)$. $S_V(M)$ denotes the signature of message M with the private key pr_V. Encode T, ΔT and σ in a QR code and display the code (Algorithm 1, lines 7-10).

CheckIn$(C(Id_C, V, f, pub_S), S(priv_S, f))$*:* To check-in at a venue V during time frame f, the client C sends Id_C and V to the provider S. S retrieves the record $Rec_V = [V, Loc_V, Id_O, pk_V, R_V]$ associated with the venue V, where Id_O is the venue owner's identifier, pk_V is XACTR_V 's public key and R_V is a random value generated (during $RegisterVenue$ by XACTR_V . S sends to C a fresh random nonce R_S. Depending on the type(s) of location verification the user agrees on, one or more of the following actions are performed:

• **WES:** With the Wi-Fi of its smartphone on, C scans and reports to S the ssid's of all ad hoc networks in its vicinity. S uses the R_V from Rec_V to compute XACTR$_V$'s ssid during frame f, $SSID_{V,f} = G_{R_V}(f)\ mod\ s$. S validates the check-in if $SSID_{V,f}$ is included in the list received from the client. Otherwise, it returns -1.

• **FES:** The user approaches XACTR$_V$ and records (using its smartphone's camera) the QR code displayed on its screen. The client C recovers the value encoded in the QR code, $T, \Delta T, S_V(T, \Delta T)$, and reports it to S. S verifies first the expiration condition: current time is within $[T, T + \Delta T]$. S retrieves the public key of venue V, pk_V from Rec_V and verifies the signature in the last field of the message received from C. If both verifications pass, S validates the check-in. When XACTR$_V$ determines the QR code has been read (see Section IV), it samples the current time, generates a new signature and displays a new QR code.

• **NES:** The client C of a user present at V sets up a connection with XACTR$_V$ over local communication media (Wi-Fi, Bluetooth, NFC). XACTR$_V$ generates a random nonce R_V, records the time T_1 and sends R_V, T_1 to C over the established connection. C computes an HMAC of R_V, T_1 using its key K (see $Subscribe$), $Hmac_K(R_V, T_1)$. C sends this value, along with the nonce R_S previously received from S) back to XACTR$_V$. Let T_2 be the time when XACTR$_V$ receives this message. XACTR$_V$ verifies that $T_2 - T_1$ is within an expected range for local communications. If the verification fails, XACTR$_V$ aborts. Otherwise, XACTR$_V$ signs the message $auth_V = (R_S, R_V, T_1, T_2, HMAC_K(R_V, T_1))$ with its owner's key, producing $\sigma_V = S_O(auth_V)$, and sends $auth_V$ and σ_V to C along with the identity of the venue V. C forwards these values to S. S verifies that the first field of $auth_V$ is the random value sent initially to C (to prevent replay attacks). It then retrieves Rec_V and uses pk_V to verify σ_V against $auth_V$. Using the key K stored for C (see $Subscribe$), S verifies that the fifth field of $auth$ is a correct HMAC computed by C on the second and third fields of $auth$. If either verification fails, S returns -1. Otherwise, it validates the check-in.

C. Analysis

We now prove several properties of XACT .

Theorem 1: XACT is correct.

Proof: It is straightforward to see that a client present at a venue V will be able to obtain with high probability SSID, QR code and HMACs issued by a venue – the GSN provider S will then validate C's check-in at V. ■

Theorem 2: XACT is CheckIn secure.

Proof: (Sketch) We assume for now that no attacker is present at the venue where it attempts to perform a fake check-in. We discuss wormhole attacks next. For WES-based verification, S validates a client check-in if the list of SSIDs reported by C contains $SSID_V = G_{pk_V}(f)\ mod\ s$, generated and advertised by XACTR$_V$. An adversary not present at V that is able to succeed in $CheckIn$ with non-negligible probability can be used to build an adversary that has non-negligible advantage in differentiating the output of the CSPRNG G from a random number. For FES-based verifications, S will validate the check-in if the client presents a valid signature of XACTR$_V$ on non-expired timestamps. An adversary that has a non-negligible advantage in succeeding in a FES $CheckIn$ at a venue where it is not present, can be used as a black box when building an adversary that has a non-negligible advantage in the one-more-forgery signature game (can produce a valid signature for a message not previously signed by the challenger). For the NES-based verification, S validates C's check-in if C can present a signature σ_V from XACTR$_V$, embedding, a random nonce sent initially to C and C's HMAC on a challenge from XACTR$_V$. Then, an adversary with non-negligible advantage in running $CheckIn$ at a venue where it is not present, can be used to build an adversary with a non-negligible advantage in the one-more-forgery signature game or build an adversary that has non-negligible advantage in collision games involving HMACs. ■

Detecting Wormhole Attacks: Let U be an honest user. In a wormhole attack, let P be the user present at venue V and let R be the remote user colluding with P. P helps R in claiming its presence at V. In NES, the wormhole detection takes place primarily at the venue's XACTR$_V$. We focus on a Wi-Fi based communication medium between the user and XACTR$_V$. XACTR$_V$ verifies that the claimed client R is indeed in its vicinity: R needs to compute an HMAC on a random challenge from V and send it back. In the case of an honest client, the total delay is $T_h = T_{hmac} + 2T_{WF}$, where T_{hmac} is the time to compute an HMAC and T_{WF} is the communication latency over Wi-Fi. When R and P launch a wormhole attack, the total delay is $T_{worm} = T_h + 2T_{PR}$, where T_{PR} is the latency of the communication channel between R and P. In Section IV, we show that even if R and P can save time by computing the HMAC on a powerful server, the additional communication latency will expose them during XACTR$_V$ verification. In WES and FES the wormhole attack detection takes place at S. U's overhead in WES is dominated by the cost of scanning SSIDs and the cost of reporting them to S. P and R's overhead includes an additional latency of communication between P and R. In FES, the overhead for U is the QR code scan and decode time plus the communication latency to S, whereas P and R introduce an additional latency cost.

IV. EVALUATION

We prototype XACT using Revision C4 of the BeagleBoard [10] system. BeagleBoard is an OMAP 3530 platform equipped with a minimum set of features to keep the costs to the bare minimum without compromising functionality. The BeagleBoard uses the OMAP 3530 DCCB72 720 MHz version and as we will show later, is sufficient for the limited processing requirements of an XACT . The TPS 65950 is used on the BeagleBoard to provide the required power and can optionally be powered using any battery module pack that is capable of providing a 5V power supply. In our implementation, we use a 4500 mAh Li-ion battery to power

Fig. 5. FES Implementation (top-to-bottom): Detecting moving objects to enable dynamic QR-Code generation, Prototype of QR generating XACT on Beagleboard, Detecting QR-Code using off-the-shelf client software "Bar Code" scanner.

our system through a special, 2-pin barrel jack provided by the BeagleBoard.

We have further used an Android-based Motorola Milestone smartphone featuring an ARM Cortex A8 CPU @ 600 MHz and 256MB RAM and a 16 quadcore server featuring Intel(R) Xeon(R) CPU X7350 @ 2.93GHz and 128GB RAM.

Property	Latency
XACT QR-code generation	7 ms
QR-code detection @20cm	190 ms

TABLE I
QR CODE PERFORMANCE.

Figure IV shows the prototype of FES on the Beagleboard. Since the Beagleboard does not come with built-in sensors, we utilized the ambient light sensor present in an Android-running Motorola Milestone to demonstrate obstacle detection.

The top third of Figure IV shows output from the sensor. The QR-generation application changes the displayed QR-code whenever it detects a person (or object) pass. We use the Milestone to take a picture of the QR-code and decode its content. The ambient light sensor then detects this event and changes the QR code. The middle third of Figure IV shows the BeagleBoard displaying a QR code. The bottom third of the figure shows the view of the user on our Android app, after successfully performing a QR code based CheckIn. Table I shows the average time across 50 runs for generating QR-codes on the BeagleBoard and for decoding them on the Milestone, when the QR code is recorded from a distance of 20cm. As desired, since false events may lead to frequent QR code changes, the cost of generating a QR code is much smaller than the cost of decoding it.

Wormhole Attack Evaluation: Figure 6 shows the times

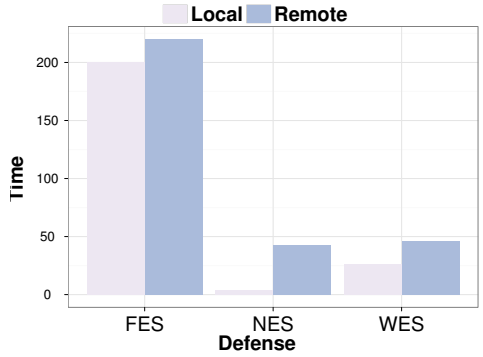


Fig. 6. XACT defenses against wormhole attacks: FES, NES and WES times in the absence and presence of wormhole attacks.

required by FES, NES and WES for honest users and in the presence of wormhole attacks. We have experimented with several systems in Miami, San Francisco and Chicago; the one-way wired communication latency (T_{PR}) was no lower than 19ms even for local systems. For FES, the time on a Motorola Milestone to snap a picture and interpret a QR code (displayed on the BeagleBoard) is 190ms. The first bar in the FES group of Figure 6 displays this value, while the second bar shows the time required when a wormhole attack is deployed. Due to the high overhead of decoding a QR code, the FES approach has a low chance of detecting a wormhole attack: the wormhole attack imposes only a 10% overhead over an honest user.

For NES, we focus on Wi-Fi connections where the two-way latency ($2T_{WF}$) is under 3ms. On a Motorola Milestone, the time required to compute an HMAC is 0.6ms (average over 10000 operations). The first bar for the NES group shows the sum of these values - the overhead imposed by NES on an honest user. The overhead imposed by a wormhole attack is the roundtrip Wi-Fi latency, plus the HMAC time (0.003ms on the 16 quadcore server, average over 1 million operations), plus the wired two-way communication latency $2T_{PR}$. Note that the HMAC overhead in the wormhole attack is however smaller, assuming a more powerful R. The second bar shows this overhead. FES imposes on wormhole attackers an overhead that is almost 12 times higher than on an honest user. Thus, wormhole attacks can be easily detected by a FES-running XACTR_V .

Finally, for the WES approach, the overhead of an honest user consists of the time of scanning the neighboring SSIDs (6 ms on average over 20 measurements on a Milestone) and the latency of communicating this information to S. The overhead of a wormhole attack includes an additional communication latency. The third set of bars in Figure 6 display these values. The WES overhead imposed on wormhole attackers is 76% higher than that imposed on an honest user. In conclusion, NES over Wi-Fi is the mechanism most capable in detecting wormhole attacks. Furthermore, NES is user friendly, since user feedback is not required during the protocol.

V. Related Work: XACT in Perspective

In this section, we provide a detailed overview of related work, and place it in the context of our work. To achieve this, we look at existing ideas (GPS, peer witnessing and access point based solutions) and extend and tailor them for the setup considered in this paper. We then discuss their ability to satisfy the requirements introduced in Section II-C.

A. Peer Witnessing

Most of the initial work on location verification focuses on fine grained localization. In ad hoc networks capable of both RF and ultrasound communications, Sastry et al. [16] introduced the ECHO protocol. In ECHO, location claims are verified by selecting multiple nodes within the transmission range of the prover. Each verifier sends a packet to the prover over RF which the prover must echo over ultrasound. Howard et al. [17] study the use of indoor Wi-Fi to localize moving robots. Chiang et al. [18] propose a distance bounding approach for solving the location verification problem in an ad hoc network that contains multiple verifiers. The solution relies on a multilateration technique, where the prover must respond simultaneously to challenges issued by multiple verifiers situated in its vicinity. Zhu and Cao [19], through their APPLAUS system, took the next step, by proposing an approach where co-located devices cooperate to build proofs of locations over Bluetooth.

We now extend the peer verification process to the context of geosocial networks: our initial thoughts were that a centralized environment may prove beneficial.

CheckIn$(C(Id, V, T, pub_S), S(priv_S))$: When a user initiates a check-in, S generates a random SSID string and sends it to the client running on the owner's device. The client switches its Wi-Fi interface in ad hoc mode on the specified SSID. The client scans and retrieves the list of SSIDs in its area and reports it to S. S can contact any client at a later time to repeat the scan-and-report procedure outlined above. Based on the client reports, S builds a "scanner" graph, containing a directed edge between nodes corresponding to clients A and B if A reports B.

One problem of this approach stems from Sybil accounts. Fake, Sybil accounts are can easily be created in social networks. For instance, Foursquare provides two user registration methods. In one option, the user needs to provide a name, e-mail address, an optional phone number, location (current city), gender, birthday and a profile picture. In another option, users can register using their Facebook credentials. Note that both options enable an attacker to easily create fake, Sybil accounts. The verification for the first method relies on the user's e-mail address, and fake e-mail addresses are cheap [20]. For the second method, we have shown in [21, 22] how Facebook's defenses can be thwarted to allow the creation of fake accounts.

If multiple Sybils and honest clients check-in at a venue at around the same time, S will build a disconnected scanner graph, containing fake and real components. Reputation systems (see [23, 24] for detailed surveys) can allow clients to

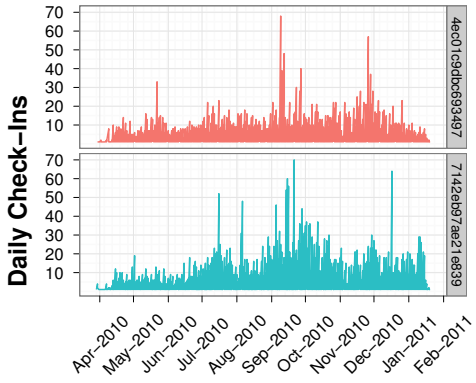


Fig. 7. Per-venue daily check-in evolution for two random venues.

"vote" for or against each other, and enable S to differentiate between fake and real sub-graphs. In the following, however, we present a combination of attacks against such techniques, centered on Sybil accounts, and show that defending against them is a hard problem: Sybil attackers can engineer the proposed solution to their advantage. While such attacks can be thwarted by increasing the difficulty of creating (Sybil) accounts in GSNs, the philosophy of GSN providers seems to be the exact opposite - simplify the joining process.

Synchronized Sybil Attack. The attacker builds an application and runs an instance of it on behalf of each of the Sybil accounts it controls [4]. To bootstrap a good reputation for a Sybil account, the attacker can launch a combination of *mechanical* and *0-Presence* check-ins. In mechanical check-ins, the attacker (or hired labor) performs valid check-ins at several venues. In 0-presence check-ins, the attacker builds first a list of venues and times when the venues are open but not frequented, or even creates its own venues; it then performs fake check-ins at the venues and times selected from this list. Given bootstrapped reputations for controlled Sybil accounts, the attacker coordinates the actions of its Sybils, e.g., each Sybil app checks-in at the same venue around the same time, in an attempt to *outvote* the good clients using the following two approaches:

Ballot Stuffing: The adversary builds a (k, l)-schedule for each Sybil, where k and $0 < l \leq k$ are parameters. The schedule satisfies the following constraints: (i) each time a check-in is performed, at least k Sybils are at the same location, (ii) the distance and time between successive check-ins for a Sybil obey the laws of physics and (iii) any two check-in sets differ in at least l Sybils If a few Sybils already have a good reputation (through mechanical reputation building or 0-presence check-ins) this attack achieves two goals - the reputations of the Sybils are monotonically increasing and their check-ins are validated.

Bad-Mouthing Attacks: If negative reputations are allowed, the attacker acquires a bad reputation for a subset of its Sybil accounts. The attacker then reports the SSIDs of target clients from the accounts of its badly reputable Sybils, negatively

[4]The Sybil apps can run on any attacker controlled machine, even on resources rented from a cloud provider.

affecting the reputation of the targets.

Without being able to limit the creation of Sybil accounts, the fundamental flaw of peer witnessing/reputation based solution stems from the inability of the GSN provider of differentiating honest users from Sybils. Given a set of SSIDs checking-in from the same venue, that form two scanner subgraphs, the GSN provider will be unable to label one as fake; with a careful implementation of bad-mouthing and ballot stuffing attacks, the attacker may build a scanner graph with higher reputation than the real clients.

Figure V-A confirms the ease of performing such attacks; it shows the per-venue check-in distribution over time for random venues from the TAMU dataset from Aug 2010 to Feb 2011. Observe the periodicity in the daily check-in behaviors, ranging from less then 5 (but never 0) to 70 check-ins per day. An adversary controlling only a handful of Sybil accounts is easily able to overpower honest users and bypass peer-witnessing defenses.

B. Access Points

Saroiu and Wolman [25] explored the location proof concept – a piece of data that certifies a receiver to a geographical location. The solution relies on enhanced access points (APs), able to issue such signed proofs. Such APs add their location to their beacons. Upon client request, the APs issue signed location certificates, containing the client's identity, the AP's identity, location and timestamp. Note that multiple certificates can be combined to triangulate clients and obtain more accurate positions. Luo and Hengartner [26] extend this concept with client privacy, achieved with the price of requiring three independent trusted entities.

Access points seem an ideal candidate for solving the secure location verification problem: they are widely accepted and the solution implies relatively small changes to their code base. They do however also have disadvantages: Most APs are owned by regular users who lack the incentives to to install new code. Moreover, AP owners may find it profitable to provide (even proactively) fake location certificates.

C. Comparison Conclusions

In this section we have shown that careful extensions of existing solutions and their application to the geosocial location cheating problem do not satisfy the requirements outlined in Section II-C. We stress that this is an inherent limitation of the environment proposed – the inability of the GSN provider to tell the ground truth without extensive investment in a trustworthy infrastructure. In contrast, XACT requires a small investment only from venues interested in preventing location fraud, cannot be circumvented by a single attacker and are hard to circumvent even by complex wormhole attacks.

VI. CONCLUSIONS

In this paper we study location fraud problems in geosocial networks. We propose Wi-Fi, QR-code and near field based location verifications solutions. Through a proof-of-concept BeagleBoard implementation, we show that they are easy and cheap to deploy as well as user friendly. We prove that our defenses force the presence of at least one attacker at a target venue, while wormhole attacks are detected through the noticeable additional delays over honest user behaviors. In future work we will address the time synchronization requirements between the provider and venue deployed devices.

REFERENCES

[1] Foursquare. https://foursquare.com/.
[2] Gowalla. http://gowalla.com/.
[3] Yelp. http://www.yelp.com.
[4] Facebook Places. http://www.facebook.com/places.
[5] Big Boss. Location spoofer. http://goo.gl/59HMk, 2011.
[6] Gpscheat! http://www.gpscheat.com/.
[7] Wenbo He, Xue Liu, and Mai Ren. Location cheating: A security challenge to location-based social network services. In *Proceedings of IEEE ICDCS*, 2011.
[8] Foursquare Official Blog. On foursquare, cheating, and claiming mayorships from your couch. http://goo.gl/F1Yn5, 2011.
[9] TripAdvisor. http://www.tripadvisor.com/.
[10] G. Coley. Beagleboard system reference manual. *BeagleBoard. org, December*, 2009.
[11] J. Bickford, R. O'Hare, A. Baliga, V. Ganapathy, and L. Iftode. Rootkits on smart phones: attacks, implications and opportunities. In *Proceedings of the Eleventh Workshop on Mobile Computing Systems & Applications*, pages 49–54. ACM, 2010.
[12] Z. Cheng, J. Caverlee, K. Lee, and D. Z. Sui. Exploring millions of footprints in location sharing services. In *Proceeding of the 5th AAAI ICWSM*, 2011.
[13] Twitter. http://www.twitter.com.
[14] M. Cha, H. Haddadi, F. Benevenuto, and K.P. Gummadi. Measuring user influence in twitter: The million follower fallacy. 2010.
[15] H. Krawczyk, R. Canetti, and M. Bellare. Hmac: Keyed-hashing for message authentication. 1997.
[16] Naveen Sastry, Umesh Shankar, and David Wagner. Secure verification of location claims. In *Proceedings of the 2nd ACM workshop on Wireless security*, WiSe '03, 2003.
[17] Andrew Howard, Sajid Siddiqi, and Gaurav S. Sukhatme. An experimental study of localization using wireless ethernet. In *in 4th International Conference on Field and Service Robotics*, 2003.
[18] Jerry T. Chiang, Jason J. Haas, and Yih-Chun Hu. Secure and precise location verification using distance bounding and simultaneous multilateration. In *Proceedings of the second ACM WiSec*, 2009.
[19] Z. Zhu and G. Cao. APPLAUS: A Privacy-Preserving Location Proof Updating System for Location-based Services. In *Proceedings of IEEE INFOCOM*, 2011.
[20] 10 Minute Mail. http://10minutemail.com.
[21] R. Potharaju, B. Carbunar, and C. Nita-Rotaru. ifriendu: leveraging 3-cliques to enhance infiltration attacks in online social networks. In *Proceedings of the 17th ACM conference on Computer and communications security (CCS) poster*, 2010.
[22] R. Potharaju, B. Carbunara, and C. Nita-Rotaru. 3-clique attacks in online social networks. Technical report, CERIAS, August 2011. Unpublished manuscript. Available for download at www.cs.fiu.edu/~carbunar/morphx.pdf.
[23] Audun Jøsang, Roslan Ismail, and Colin Boyd. A survey of trust and reputation systems for online service provision. *Decis. Support Syst.*, 43, March 2007.
[24] Kevin Hoffman, David Zage, and Cristina Nita-Rotaru. A survey of attack and defense techniques for reputation systems. *ACM Comput. Surv.*, 42, December 2009.
[25] Stefan Saroiu and Alec Wolman. Enabling New Mobile Applications with Location Proofs. In *Proceedings of HotMobile*, 2009.
[26] W. Luo and U. Hengartner. VeriPlace: A Privacy-Aware Location Proof Architecture. In *Proceedings of ACM SIGSPATIAL GIS*, 2010.