Bluetooth Peer-to-Peer Location Certification with a Gamified Mobile Application

Ricardo Grade, Samih Eisa, Miguel L. Pardal
INESC-ID, Instituto Superior Técnico, Universidade de Lisboa, Portugal
{ricardo.grade, miguel.pardal}@tecnico.ulisboa.pt, samih.eisa@inesc-id.pt

***Abstract*—Nowadays, tourists turn to digital platforms to discover new places to explore. *CROSS City* is a smart tourism mobile application that enhances the user experience of tourists visiting points of interest in a route by rewarding them in the end, if they actually visited all locations. From a technical standpoint, the user location is certified resorting to strategies that take advantage of both the diversity of the existing Wi-Fi network infrastructure throughout the city, as well as the presence of other users at the same site using Bluetooth. This work developed a new, peer-to-peer location certification strategy and added gamification elements to encourage users to keep the wireless radios turned on and use the app more. This work was evaluated both in laboratory experiments and with users in a real-world scenario which demonstrated that the new Bluetooth peer-based strategy is both feasible and resistant to collusion attacks.**

I. Introduction

Mobile devices are increasingly equipped with various network interfaces, such as Wi-Fi and Bluetooth, which can be used to prove the device location, through detection of the surrounding context and in cooperation with nearby devices that act as *witnesses* [1]. Endorsement schemes [2] can be *infrastructure-based*, where the witnesses are elements of the reliable wireless infrastructure; or *peer-based*, where the witnesses are other users at the location. In this work, we focus on peer-based systems, such as: SureThing [1], LINK [3], APPLAUS [4], CREPUSCOLO [5], and PASPORT [2] because they are particularly useful in crowded locations where the wireless infrastructure is insufficient.

SureThing has recenly been expanded to produce a location certification framework, with common data structures and software components that can be reused to provide location certificates for different applications. CROSS City [6] is one such application, in this case, focused on smart tourism. It rewards users for actually visiting all the points of interest that make up a tourist route, and it relies on the location certificates to make sure the visits are not being forged. Before this work, all location certification strategies developed in CROSS only worked in locations with the necessary surrounding Wi-Fi network infrastructure. Moreover, location evidence relied on the reporting of volatile Wi-Fi networks to obtain location certification with temporal granularity, which could only be validated after a long period, up to 24 hours, making the users wait this time to get their rewards. The contribution of this work addressed these limitations by introducing a new peer-based location certification strategy that we named *P2P Witnessing*. The insight was to leverage the moments when user devices *see* each others to produce co-location evidence. This technique is especially useful in places with no surrounding infrastructure.

Furthermore, since the location certification strategy just introduced is peer-based, it means that user participation is now an even more critical factor for its proper functioning. To address this challenge, we fully re-implemented the app to deliver a better user interface and, more important, added *gamification* [7] elements to encourage users to further use the app and embrace the new peer-based strategy. This has been shown to be effective in both location-based applications [7] and in the tourism domain [8].

The *P2P Witnessing* was evaluated both in laboratory experiments and in a real-world setting. The results show that it is effective and responsive, that the gamification elements served their purpose, and, finally, that the produced solution is collusion-resilient, i.e., the location certificates are secure.

II. Background and Related Work

Peer-based location certification system leverage the existence of on-site users willing to endorse the location of their peers, issuing them endorsements via Bluetooth or another short-range wireless communication protocol. Systems such as SureThing [1], LINK [3], APPLAUS [4], CREPUSCOLO [5], and PASPORT [2], generically operate in two phases: *endorsement acquisition*, where the prover broadcasts claims and collects endorsements issued by nearby witnesses; followed by *claim validation*, where the verifier validates the endorsements supporting the claim.

Peer-based systems can suffer the following attacks: *endorsements transfer*, where a malicious prover uses endorsements issued to another; *endorsements forging*, where a malicious user can forge endorsements either for their own benefit or to defame another; *prover-witness collusion*, when the prover and witnesses are malicious and those witnesses endorse the false claim of that prover; *prover-prover collusion*, when two provers are malicious and the first channels a claim to the second who is at a different location, which in turn broadcasts it over Bluetooth, causing honest witnesses to endorse the false claim of the first; *location privacy breach*, where users often broadcast their identity to neighboring

devices or a third-party server, however if the user identity is publicly disclosed, their personal information would be leaked.

Next, we present the defenses against these attacks. *Ensuring nontransferability of endorsements*: in LINK [3], all endorsements have the sequence number associated with their claim built in. In APPLAUS [4] and CREPUSCOLO [5], each claim contains an associated nonce (number used only once) that is present in all endorsements issued for this claim. In PASPORT [2], two secret random numbers are generated for a claim, which are used to respond to the challenges that are part of the performed Distance Bounding (DB) protocol.

Ensuring unforgeability of endorsements: to avoid eavesdropping on messages, provers and witnesses encrypt them with the public key of the verifier. To avoid tampering with messages, users sign them with their private key.

Preventing prover-witness collusion: SureThing [1] and LINK [3] devalue endorsements issued by the same witnesses to the claims of a given user, as they assign weighted reputations to witnesses, which are lower depending on the number of times they have endorsed claims of that user. In both APPLAUS [4] and CREPUSCOLO [5], they check how many other co-located and concurrent endorsements exist. In PASPORT [2], the verifier selects, from the range of witnesses that are in the same region as the prover, a subset of them who will be able to endorse a given claim.

Preventing prover-prover collusion: to circumvent this attack CREPUSCOLO [5] adds *Token Providers* to the infrastructure that act as trusted witnesses located in known locations that generate tokens which contain images from surveillance cameras. The issuance of endorsements in PASPORT [2] is carried out based on a DB protocol, where the maximum RTT between the prover and the witness is calculated based on the acceptable distance between them.

Preserving user location privacy: APPLAUS [4] and CREPUSCOLO [5], assign a set of pseudonyms to each user. In this way, the privacy of users is maintained even in scenarios where the server is compromised, as locations are associated with pseudonyms that only a trusted CA, that assigns them, knows who owns them. PASPORT [2] takes a different approach, the prover identifiers are not revealed to witnesses, as this is sent to them encrypted with the verifier public key, as well as witness identifiers (in endorsements).

III. Design and Implementation

In this work, we integrated the new peer-based collusion resilient location certification strategy – *P2P Witnessing* – with the gamification elements to encourage users to use it. We assume that: the users are willing to share a small portion of their mobile device power resources to endorse the claims of their peers; the mobile device supports BLE to execute the peer endorsement acquisition protocol; and the device will eventually have access to the Internet to validate the user visits.

A. Requirements

We define the following requirements for the P2P strategy:

- R1 The strategy must only successfully validate the visit if a considerable number of witnesses endorse it;
- R2 The strategy must be resilient against prover-witness collusion;
- R3 The peer endorsement validation protocol must provide responsive user feedback;
- R4 The strategy must be resilient against prover-prover collusion;
- R5 The peer endorsement acquisition protocol must not require users to be in close proximity for an extended period of time;
- R6 The strategy adoption impact on the mobile device power resources must not impact the user experience.

In addition, we defined additional requirements related to user participation:

- R7 The app should encourage users to adopt the P2P Witnessing strategy;
- R8 The app should motivate users to further use it;
- R9 The app must run on unmodified smartphones running the most popular OS to reach a large number of users.

As *threat model*, we assume that a malicious user U_α cannot: share private keys and other secrets, which would allow third parties to impersonate U_α; compromise the server and gain access to the data it stores; discover the server private key. On the other hand, we assume that a malicious user U_α can try to: T1 eavesdrop and tamper with sensitive messages; T2 submit endorsements issued on behalf of other users for U_α's own benefit; T3 submit endorsements issued in the past to try to certify the current location of U_α; T4 systematically collude with witnesses that endorse U_α's false claim; T5 collude with another prover, causing honest witnesses to endorse U_α's false claim; and T6 track app users eavesdropping the messages they exchange.

B. P2P Witnessing Strategy

We now present the details of the peer endorsement acquisition and validation.

1) Short-Range Wireless Technology: We chose BLE (Bluetooth Low Energy) as short-range wireless technology due to: low power consumption; connection speed; ability to enable device discoverability without user interaction; ability to establish a connection without having to pair devices; high throughput not required; and extensibility to limited devices.

To ensure that the devices they connect to are app users, the advertising data contains the identifier of the service it provides, which is unique to our protocol. When a device is scanned, if an instance of the protocol is already running, that device enters a queue (this is done so that there are not multiple protocol running instances, otherwise it can affect its performance and latency which is a critical factor for its

success), when the device turn comes, a connection request is sent to it. When the scanned device (witness) accepts the connection request and the connection is established, the client device (prover) requests a high-priority connection to keep the latency of subsequent messages to a minimum and negotiates an MTU of 515 bytes. The maximum message payload size of our protocol is composed by: the endorsement, which is an encrypted block with 512 bytes; and by the 3-byte header of messages exchanged via BLE. So, we require an MTU of at least $512 + 3 = 515$ bytes, which is below the MTU limit of 517 bytes in Android devices.

As soon as the connection is set up, the acquisition protocol execution itself begins. When the user collects or issues endorsements, this information is captured to be displayed on the end-of-visit screen as feedback to the user. To handle errors that might occur during connection establishment, we retry it under a linear backoff policy (i.e. the reconnect timeout increases linearly).

2) Cryptography Selection: The encryption function with the entity's public key uses the *RSA/ECB/PKCS1Padding* algorithm. The signature function with the entity's private key uses the *SHA256withRSA* algorithm. We opted for RSA with default options as it is still the most widely available public-key cryptographic scheme in Android.

3) Peer Endorsement Acquisition Protocol: Let E_{entity} be the encryption function with the public key, S_{entity} the signature function with the private key, and $m_i||m_j$ the concatenation of m_i with m_j.

1) **Prover**: upon starting the visit, the prover generates two n-bit random numbers a and b (should not be reused);
2) **Prover**: upon crossing paths with a witness, the prover generates the claim $LC = a||b||ID_{prover}||SessionID_{prover}||PoI||Timestamp$ and sends its signature $e = S_{prover}(LC)$ to the witness;
3) **Witness**: upon receiving the claim signature, if the witness has not yet endorsed the prover [1] or the prover has not yet exhausted their attempts, the witness generates an n-bit random number h and sends it to the prover;
4) **Prover**: upon receiving h, the prover computes $z = b \oplus h$ and responds to the witness with an ack message;
5) **Witness**: upon receiving the acknowledgment, the witness generates an n-bit random number c and begins the DB protocol: For each challenge bit c_i:
   a) **Witness**: the witness sends c_i to the prover, starting a timer immediately thereafter to time the response;
   b) **Prover**: upon receiving c_i, the prover computes the response bit $r_i = a_i$ if c_i is 0, otherwise $r_i = z_i$, and sends r_i to the witness in response to the challenge;
   c) **Witness**: upon receiving r_i, the witness validates the response time to the challenge (validation which is explained in detail below in the DB protocol presentation), if it fails the claim is rejected.
6) **Witness**: upon successful completion of the DB protocol, the witness generates $m = r||c||h||e||ID_{witness}||SessionID_{witness}||PoI||Timestamp$ and sends the endorsement $LE = E_{server}(m||S_{witness}(m))$ to the prover.

[1]The MAC address is used to distinguish provers as the device name is not disclosed to avoid user tracking.

This protocol was inspired by the P-TREAD protocol introduced in PASPORT [2], in which the verifier is the server. The main change is that there is no first contact with the server to generate a claim identifier. This step was necessary for the PASPORT witness selection phase, which aims to reduce the effectiveness of prover-witness collusion, with only a few witnesses of the server choice able to issue endorsements to the prover in question. However this solution is not suitable for our case as users are not required to have Internet access during the visit and as such the server is unaware of which witnesses are at the same location as the prover. Instead, to be resilient to prover-witness collusion, we integrated a reputation system and a witness decay mechanism explained ahead.

4) Distance-Bounding Protocol: Also inspired by P-TREAD, we initially planned to define the maximum acceptable challenge response time $RTT_{max} = \frac{2D}{C} + t_0$ where: D is the maximum distance that can be found between the prover and the witness; C is the speed of light; and finally, t_0 is the computation overhead of the challenge response bit. However, when implementing it, we found that this threshold did not hold, the response times we were measuring (between about 20 and 40 milliseconds) were $>>$ than $RTT_{max} \approx 1$ millisecond (considering $D = 10$ meters and $t_0 \approx 1$ millisecond), the reason for this is that the propagation time represented in the RTT_{max} equation by $\frac{2D}{C}$ is on the order of nanoseconds, but unfortunately this does not take into account factors such as: the BLE connection interval (which delays packet propagation by a few milliseconds even when requesting a high-priority connection, the volatility of the Android system running other processes, or the network congestion. These problems are due to the fact that our DB protocol is fully implemented at the application layer, where we do not have full control over these factors, but we also cannot descend from this layer because we want our app to be usable on any unmodified Android device.

To work around the problem we experimented with other short-range wireless technologies, to check if this degree of latency in sending packets was general, or unique to BLE. We implemented the protocol in classic Bluetooth, Wi-Fi Direct, and Nearby Connections API (this interface uses a combination of BLE, classic Bluetooth, and Wi-Fi Direct). The implementations on top of classic Bluetooth and Nearby Connections API resulted in higher challenge response times than those achieved with the initial implementation of BLE with the high-priority connection. The only implementation that seemed to give better results beforehand ($\approx$ 3 milliseconds) was the one on top of Wi-Fi Direct, however, the solution did not scale when we did exhaustive tests with two devices exchanging endorsements every 5 seconds. The response times increased greatly over time, reaching times

longer than those obtained with the implementation of BLE. In addition, it forced the users to accept pairing with other devices to exchange endorsements, connection establishment errors were frequent and more battery was drained. As none of the alternative technologies solved this problem, we decided to stick with BLE because it is the technology that best suits our use case.

Given the impossibility of using the P-TREAD RTT_{max} equation for our use case, we decided to empirically define our own challenge response time threshold. The value we determined was $T_{max} = 34,5$ milliseconds in our evaluation IV-C in order to keep the DB protocol execution success rate as low as possible for colluding users and as high as possible for honest users. Furthermore, to provide robustness to the DB protocol against observed response time fluctuation (induced by runtime volatility) without degrading its resilience against colluding provers, we designed the following refinements: *number of challenges definition* - we set the number of challenges that the protocol is composed of to 64, as this seemed to be enough to get an average that would mitigate the response time spikes induced by the Android runtime volatility without degrading its performance; *noise reduction* - to eliminate underlying implicit noise, we do not take into account the 10% worst response times recorded; *claim rejection* - instead of rejecting a claim as soon as we get a longer-than-expected response time, we sum up the fastest 90% and only if that sum is greater than $T_{max} \times numberOfChallenges \times 90\%$ does the witness reject the claim. In this way, the claim is only rejected if the average response time is or will be greater than T_{max}; *attempts* - each witness allows each prover 3 protocol execution attempts, in case of false negatives in claim rejection, this flexibility allows us to be more demanding in defining the T_{max} value in order to reach greater resilience against colluding provers.

5) Peer Endorsement Validation: When CROSS users end the visits, they submit all the location evidence collected during the visit to the server.

The first step to validate the evidence is to validate the claim, in this way if any of the conditions are met the prover is behaving maliciously and as such we automatically **reject** the visit. Two examples of these conditions are: – The prover named in the claim is not the visit validation requester (extracted from the JWT sent along with the request); – The point-of-interest identifier present in the claim is different from the one present in the visit.

Second, we move on to the endorsement validation, where if any of the following conditions are met, unlike claim validation, we are not sure who is behaving maliciously, whether the prover or the witness (because the endorsement may have been forged) and as such we simply **ignore** it. Two examples of these conditions are: – The endorsement issuance timestamp is not within the visit period or is less than the claim generation timestamp. *Note*: Since the endorsement issuance timestamp is assigned by the witness device and the claim generation timestamp are assigned by the prover device, we assume a maximum clock skew of 2 minutes; – The endorsement is not properly signed by the witness.

To conclude, if there is any response bit r_i that does not match the expected one, it means that either the user who answered the challenges was not in possession of the secret numbers a and b, which indicates that the prover was colluding with another, or the witness tampered with the challenge responses the prover gave, in either case, we **ignore** the endorsement as we fail to recognize its authenticity.

a) Witness Decay: To protect the system against systematic prover-witness collusion, we implemented a witness decay mechanism whose function is to reduce the weight that endorsements, issued by recurring witnesses of a given prover, have under the strategy confidence calculation. Thus, the weight of an endorsement is calculated as follows:

$$endorsementWeight(p, w) = \frac{Rw}{Npw + 1} \quad (1)$$

where Rw is the reputation of the witness w who issued the endorsement and Npw is the number of routes the prover p initiated or completed, which the witness w has already testified to.

b) Visit Confidence: The confidence assigned to the visit is the product of the confidence that the physical displacement is possible with the sum of the confidence acquired with the different strategies:

$$min(displCM \times (wiFiConf + peerConf), 1) \quad (2)$$

where the $displCM$ is the confidence multiplier that the displacement between the reported points of interest is physically plausible, the $wiFiConf$ is the confidence assigned by the existing Wi-Fi-based strategies, and the $peerConf$ is the confidence assigned by the P2P Witnessing strategy.

C. Threat Mitigation Assessment

We now explain how each aforementioned threat is mitigated by our design: to address T1, the sensitive messages, which are claims and endorsements, are not disclosed to eavesdroppers: the claim is not sent to the witnesses, only its signature; the endorsement is sent to the prover encrypted with the server public key; in addition, they are sent to the server through an HTTPS connection that uses the SSL protocol to protect the messages exchanged; for T2, users cannot take advantage of endorsements issued on behalf of other users as the endorsement contains the signature of the claim which in turn contains the prover identity; for T3, the endorsements are bound to a location and a timestamp, they can be submitted later, but there is nothing to be gained from keeping them for that purpose; in the case of T4, systematic prover-witness collusion is essentially mitigated by the witness decay mechanism in place, that devalues the weight of endorsements issued by recurring witnesses; but also by requiring a confidence threshold for the visit to be accepted, which is only possible

to obtain with a considerable number of witnesses; in the case of T5, prover-prover collusion is mitigated by the DB protocol, as it allows witnesses to verify if the prover they are communicating with is indeed on site; to thwart T6, the presented DB protocol strives to maintain the privacy of users when communicating with each other as all personal sensitive information (such as their username and location) is encrypted with the server public key.

D. Gamification

We developed the following features to enable gamification and encourage users to further use the app:

1) Scoring System and Scoreboard: To foster competition among app users and encourage the desire to climb to the top. The scoring system unit is the XP that is earned when the user successfully validates a visit or a visit from another, endorsed by the user, through the P2P Witnessing strategy, is successfully validated. The users are then ranked according to their all-time, seasonal and weekly score.

2) Badge System: To encourage app users to complete activities and trigger a sense of accomplishment when completing them. When a user reaches a certain condition for the first time, they receive the corresponding badge.

3) In-app Currency: To increase user in-app loyalty through the potential to save money by completing previous activities. The in-app currency is named gems, and they are assigned to users in the same cases as the XP cases.

IV. Evaluation

First, we evaluate the *P2P Witnessing* strategy in laboratory experiments to assess its effectiveness, responsiveness and collusion-resilience; second, we evaluate its feasibility in a real-world scenario and the impact of the gamification features.

A. Prover-Witness Collusion-Resilience

Prover-witness collusion could be carried out by a prover that has available a set of witnesses who can endorse the visit, even if they are not physically at the point of interest.

To assess the resilience of the system against systematic prover-witness collusion, we need to answer the following question: *How many successful collusions can a prover perform with N newly created witnesses?* To find an answer, we compared the acquired confidence with the confidence threshold necessary to obtain for the visit to be accepted, attempting to carry out up to 10 collusions with different sizes of witness sets. The confidence acquired in the first collusion attempt can be derived from Equation 1 as $min(\sum_w Rw, 1)$, so that the greater the number of witnesses, the greater the initial acquired confidence. On subsequent attempts it drops sharply until the claim is no longer accepted, while the confidence threshold increases linearly with the number of rejected claims. This drop in the confidence acquired results from the witness decay mechanism in place.

From this analysis, we concluded that if the size of the set of witnesses at the disposal of the prover is less than or equal to 4, no collusion can be performed, if the set size is between 5 and 8, 1 collusion can be performed, if the set size is between 9 and 10, 2 collusions can be performed. This is, if the set size is greater than 4, the prover will be able to carry out the attack, but only a reduced number of times. Thus, we conclude that the system meets the R2 requirement, being resilient against systematic prover-witness collusion. Even if witnesses have a track record of good behavior, since the decay applied to the reputation of the witness is independent of their behavior.

B. Validation Protocol Responsiveness

We now evaluate the responsiveness of the peer endorsement validation protocol to assess whether the server is able to provide real-time feedback to the user when submitting a visit.

The computer on which the server was running was a GL75 Leopard 10SEK model equipped with an Intel® Core™ i7-10750H CPU Hexa-Core 2.60GHz, with 16GiB of RAM. The computer running the simulation is a GL553VW model equipped with an Intel® Core™ i7-6700HQ CPU Quad-Core 2.60GHz, with 16GiB of RAM. Both running on top of Ubuntu 20.04.4 OS. The times were recorded in the testing client.

To answer the question: *How long does it take the server to validate a visit endorsed by N witnesses?*, we performed the experiments and the results are shown in Figure 1 where we can see the evolution of the visit validation time in the server, varying the number of witnesses who endorse the visit.

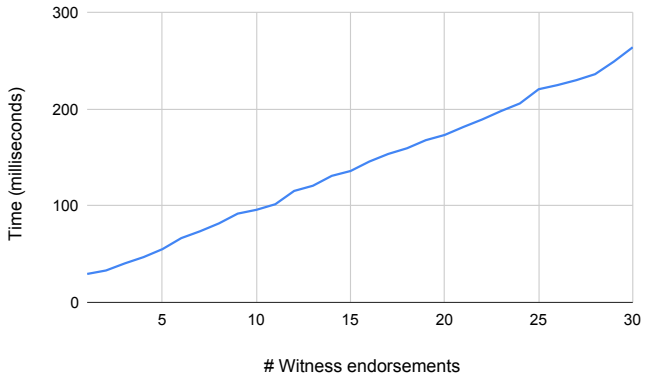

Figure 1. Visit validation time varying the number of witnesses who endorse the visit.

From this plot, we conclude that the visit validation time *varies linearly* with the number of peer endorsements present in it. We can deduce from the plot a validation time function $t(w) \approx 8,09w + 29,43$, where $t(w)$ is the expected validation milliseconds of a visit endorsed by w witnesses. Considering that the protocol provides a responsive feedback experience to the user, if the validation response time is less than 1 second, according to $t(w)$ this would only happen for $w \geq 120$, being

quite unlikely that this number of witnesses would be obtained we conclude that in this scenario the protocol is responsive.

To answer the question: *What is the respective overhead and likelihood of the occurrence of N simultaneous visit submissions?*, we performed the experiments and the results are shown in Figure 2 where we vary the number of simultaneous submissions of visits endorsed by 15 witnesses.

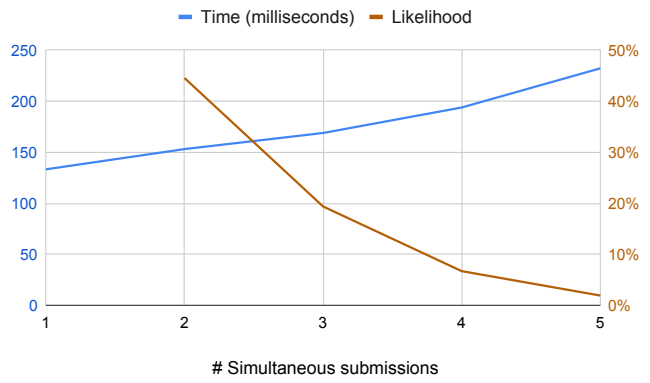

Figure 2. Visit validation time varying the number of simultaneous visit submissions.

In Figure 2, we can see that, as expected, the visit validation time increases with the number of simultaneous submissions, however, given an increase ratio of only $\frac{232,28}{133,25} = 1,74$ (comparing the validation response times of 1 and 5 simultaneous submissions) and the likelihood of that number of simultaneous submissions occurring, we conclude that submission concurrency is not a factor of concern for the protocol responsiveness. With this, we conclude that the system is responsive meeting the R3 requirement, even in cases where multiple visits are submitted simultaneously.

C. Prover-Prover Collusion-Resilience

Prover-prover collusion could be carried out by two provers that are far from each other and one of them forwards the DB protocol challenges to the one that is elsewhere, managing to gather endorsements that certify that the other prover is on site, when in fact it is not. We are preventing this by restricting the challenge response times that are acceptable for a witness to endorse a claim to be less than the time required for the $witness \leftrightarrow prover_\alpha$ communication over BLE plus the $prover_\alpha \leftrightarrow prover_\beta$ communication over Wi-Fi.

For these tests, two different mobile devices were used, which communicate with each other over BLE. The results of both devices were collected to observe their discrepancies. These mobile devices were: a Samsung Galaxy S9 model equipped with an Exynos 9810 CPU Quad-Core 2.8GHz + Quad-Core 1.7GHz, 4GiB of RAM, on top of Android 10 OS; and a Xiaomi POCO X3 Pro model equipped with a Qualcomm® Snapdragon™ 860 CPU Octa-Core up to 2.96GHz, 8GiB of RAM, on top of Android 11 OS.

To assess the resilience of the system against prover-prover collusion, we answer the following question: *What is the DB protocol acceptance rate with honest provers versus colluding provers?* To look for an answer, we start by monitoring the average challenge response times between the two mobile devices in three scenarios: when separated by 1 meter, 10 meters, and in adjacent rooms with the two doors closed. We concluded that regardless of the arrangement of the two mobile devices, the times oscillate between approximately the same values, i.e. the message propagation time does not play a major role in the challenge response time fluctuation. To visualize the distribution of the average challenge response times recorded, we gathered all the times obtained across all scenarios and inserted them in the plot in Figure 3.

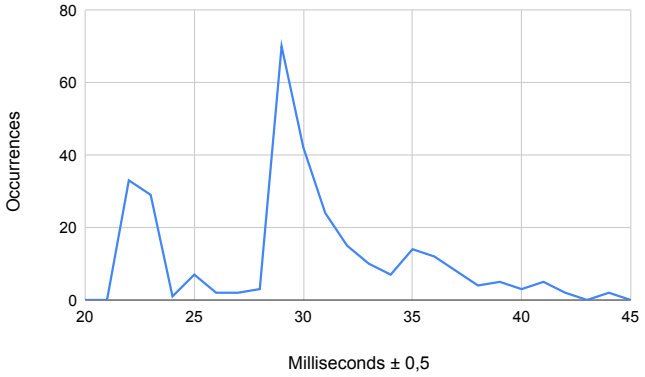

Figure 3. Frequency of average challenge response times.

We consider that the probability of a user responding to challenges in time T is given by the function $P(t \leq T) = \frac{\sum_t (1 \textbf{ if } t \leq T \textbf{ else } 0)}{N}$ such that t are the times collected from the sample collected above and $N = 300$ the number of samples. This probability function will be used next to calculate the *acceptance* and *rejection rate* as a function of challenge response times.

Now that we have measured the time distribution of the $witness \leftrightarrow prover_\alpha$ communication over BLE, let us estimate the $prover_\alpha \leftrightarrow prover_\beta$ communication time over Wi-Fi. In other words, let us measure the latency introduced in challenge responses by a user attempting to perform prover-prover collusion. To do so, we recorded the average ping time from Cascais (Portugal) to some locations [2] in Portugal and Spain, Portugal's neighboring country, at different times of the day, on both mobile devices in parallel. We concluded that they vary depending on the mobile device, but not on the time of the day, as their standard deviation recorded in each iteration is small.

Now that we know about the distribution of challenge response times and the average time a packet takes to reach different locations on different mobile devices, we can now

[2]The IPs used to ping each location were: 193.136.128.169 (Lisbon, Portugal); 193.137.35.140 (Porto, Portugal); 5.134.119.53 (Madrid, Spain); 185.166.215.231 (Barcelona, Spain).

derive the DB protocol acceptance rate in scenarios where the prover is either honest or colluding. Remember that if something goes wrong and the endorsement gets rejected users have up to 3 attempts (empirically defined) to request an endorsement from each of the witnesses around them. To calculate the first attempt acceptance rate, this is equal to $AR_1 = P(t \leq T_{max} - L)$ (P being the probability function presented above) given that AR_i is the acceptance rate of attempt i, T_{max} is the threshold value of the average challenge response times, and L is the communication latency between the two colluding provers ($L = 0$ if the prover is honest). We set $T_{max} = 34,5$ milliseconds, which seemed the most appropriate, comparing the acceptance rate of honest and colluding provers, and L is the ping time from Cascais (Portugal) to the mentioned locations [2]. The acceptance rate of subsequent attempts follows a binomial distribution with a success probability $p = AR_1$.

We concluded that with just 2 attempts, an honest prover is practically guaranteed to get the endorsement successfully (acceptance rate $\approx 97\%$), while in a scenario of prover-prover collusion where one is in Cascais (Portugal) and the other in Lisbon (Portugal), if they are connected over Wi-Fi, the colluding prover on average will only be able to get endorsements from ≈ 1 out of 3 witnesses (acceptance rate for this average case with 3 attempts is $\approx 32\%$); if connected over 4G the colluding prover will not be able to get any. In the scenario in which the colluding provers are further away, such as one in Cascais (Portugal) and the other in Porto (Portugal), even connected over Wi-Fi, it is practically impossible (max acceptance rate with 3 attempts is 1%) for the colluding prover to be able to deceive any witness into issuing an endorsement.

We conclude that the system meets the R4 requirement, being resilient against prover-prover collusion attacks in the case where the prover is connected over 4G; or over Wi-Fi where they are far from each other (as from Cascais to Porto $\approx$ 300 km). Even in cases where the prover is connected over Wi-Fi and both are close to each other, the DB protocol substantially decreases the effectiveness of this attack.

D. Acquisition Protocol Performance

To assess the performance and feasibility of the peer endorsement acquisition protocol, we answer the following question: *How long does the peer endorsement acquisition protocol take to be executed, accounting for connection time, connection setup time, and endorsement time (from the claim is sent until the endorsement is received)?* To investigate it, we monitored the execution time of the peer endorsement acquisition protocol on two mobile phones that periodically issued endorsements to each other.

The samples collected on Samsung Galaxy S9 are shown in the plot in Figure 4, those collected on Xiaomi POCO X3 Pro are not shown here because they are similar. We can see that the average execution time of the peer endorsement acquisition protocol is around 4,5 seconds on both mobile devices, and

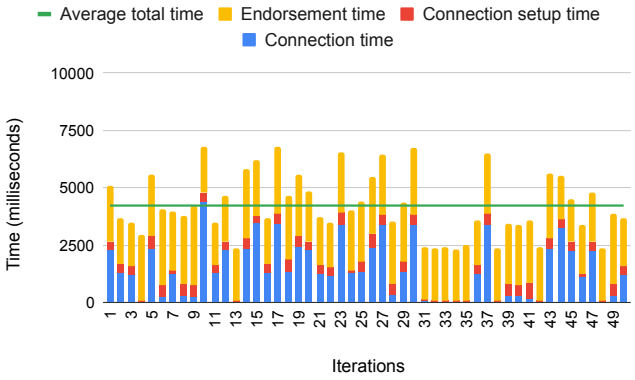

Figure 4. Execution time of the endorsement acquisition protocol on Samsung Galaxy S9.

that the endorsement time is relatively constant, in contrast to the connection establishment and setup time where volatility is most noticeable.

Overall, the times obtained appear to be extensible to similar mobile devices and indicate that the peer endorsement acquisition protocol is responsive and does not require the prover and witness to stay close to each other for a long time, fulfilling the R5 requirement.

E. Protocol Impact on the Mobile Device Power Resources

We observed that the battery consumed by the app in both scenarios on both mobile devices is around the same percentage $\approx 0.15\%$, which allows us to conclude that the computation necessary for the exchange of peer endorsements, being performed every 30 seconds, does not represent a noticeable factor in battery consumption. We conclude, therefore, that the overhead induced by the addition of the P2P Witnessing strategy is not a factor of concern for the user experience, thus fulfilling the R6 requirement.

F. User Tests

We also did a real-world deployment on our university campus. The route was made up of three pavilions, about 200 meters apart; with each visit lasting 5 minutes. We went to these visit points the day before to scan the access points there and include them in the server database. On the day of the tour, August 2nd of 2022, we deployed the server and the reputation system on Google Cloud, generated the app APK and made it available to a group of 7 users. They were able to install the app on their unmodified Android device, fulfilling R9. We instrumented the code to print relevant metrics, which were collected at the end of the experiments.

In the experiment with 7 users, everything went as expected as well, with the exception of the following 2 anomalies: 1) $User\_\alpha$ was the one who stood out for the negative regarding the non-endorsement of their peers. We assessed that this was not due to the DB protocol failure, but to systematic

errors in the establishment of incoming connections, which could be explained by the malfunction of the Bluetooth chip in the mobile device of the user in question. 2) $User\_\beta$ failed to collect endorsements from the totality of the other users, even not considering $User\_\alpha$. Through the instrumentalization of the app, we noticed that the claims were being rejected by the DB protocol, by the witnesses, as the prover $User\_\beta$ was taking, on average, longer than acceptable time to respond to their challenges. This can be explained by realizing the device used by the user in question was a very outdated model with poor specs for the current date. Nevertheless, even accounting for the failures experienced, the peer endorsement acquisition success rate of a given peer was still high: 89.7%.

G. Gamification Impact

We sent to the participants a form to assess the impact of gamification focusing on the P2P Witnessing strategy.

1) Gamification elements benefits: From the users point of view, the gamification elements can drive them to explore further because they are rewarded for doing so. It can also motivate them to further use the app because they want to get badges to share with friends, climb up the scoreboard, and use the gems they have collected, which in doing so will earn more, creating the so-called *game loops*. An interesting factor that was assessed was that users also report that they feel more motivated to visit points of interest at the busiest hours, to issue more endorsements and thus be better rewarded, which is quite convenient for the P2P Witnessing strategy.

2) User adherence to the P2P Witnessing strategy: When the users were asked if they would voluntarily enable the P2P Witnessing strategy, all answered yes (fulfilling R7), as it improves their chances of seeing their visit successfully validated, but also allows additionally rewards for something already being down, with a negligible downside. They also acknowledged that the P2P Witnessing strategy can be essential for those who want to: compete for the first places on the scoreboard; maximize their gem earnings to save on future visits; and get badges for being a good comrade endorsing peer visits.

3) User adherence to the app: When users were asked if they would use the app again for future visits, 6 out of 7 users answered yes (meeting R8), because of the incentive to be rewarded for something they would already be doing and the possibility to save on future visit tickets, in addition, they also highlighted the ease of use and simplicity of the UI. An interesting justification for an affirmative answer was: *"because now I have tons of gems to spend"* (which the user collected on the route traveled) corroborating the creation of the game loop already foreseen.

V. Conclusion

This work successfully developed and evaluated *P2P Witnessing* that extends the CROSS City operation to places with no surrounding infrastructure. The app is now able to validate user visits in a more timely way, with temporal granularity, and prevents location evidence transfer between users. In addition, we used gamification to encourage users to use the app and to adopt the P2P strategy. With this work, we have shown how location certificates can go beyond a security credential and become part of engaging consumer applications.

Regarding the P2P Witnessing strategy, in future work we can explore the possibility of implementing it in a layer below the application layer, to obtain a more accurate measurement of the DB challenge response times, sacrificing, however, its extensibility and ease of installation on the user device, in favor of stricter location certification.

VI. Acknowledgements

This work was supported by national funds through Fundação para a Ciência e a Tecnologia (FCT) with reference UIDB/50021/2020 (INESC-ID) and through project with reference PTDC/CCI-COM/31440/2017 (SureThing).

References

[1] J. Ferreira and M. L. Pardal, "Witness-based location proofs for mobile devices," in *17th International Symposium on Network Computing and Applications (NCA)*. IEEE, 2018, pp. 1–4.

[2] M. R. Nosouhi, K. Sood, S. Yu, M. Grobler, and J. Zhang, "PASPORT: A Secure and Private Location Proof Generation and Verification Framework," *Transactions on Computational Social Systems*, vol. 7, no. 2, pp. 293–307, 2020.

[3] M. Talasila, R. Curtmola, and C. Borcea, "LINK: Location verification through immediate neighbors knowledge," in *International Conference on Mobile and Ubiquitous Systems: Computing, Networking, and Services*. Springer, 2010, pp. 210–223.

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

[5] E. S. Canlar, M. Conti, B. Crispo, and R. Di Pietro, "CREPUSCOLO: A collusion resistant privacy preserving location verification system," in *International Conference on Risks and Security of Internet and Systems (CRiSIS)*. IEEE, 2013, pp. 1–9.

[6] G. A. Maia, R. L. Claro, and M. L. Pardal, "CROSS City: Wi-Fi Location Proofs for Smart Tourism," in *International Conference on Ad-Hoc Networks and Wireless*. Springer, 2020, pp. 241–253.

[7] S.-K. Thiel and P. Fröhlich, "Gamification as motivation to engage in location-based public participation?" in *Progress in location-based services*. Springer, 2017, pp. 399–421.

[8] F. Xu, J. Weber, and D. Buhalis, "Gamification in tourism," in *Information and communication technologies in tourism*. Springer, 2013, pp. 525–537.