CROSS: loCation pROof techniqueS for consumer mobile applicationS

Gabriel A. Maia[0000−0002−8691−6110] and Miguel L. Pardal[0000−0003−2872−7300]

Instituto Superior Técnico, Universidade de Lisboa, Portugal
{gabriel.maia,miguel.pardal}@tecnico.ulisboa.pt

Abstract. Many mobile applications are location-aware, but do not verify the location information they consume, making them vulnerable to location spoofing attacks. Location proof systems aim to solve this problem by allowing devices to interact with location-specific resources and issue proof that they have been at specific locations on specific times. In this paper we introduce CROSS, a system that performs location verification using techniques compatible with off-the-shelf Android smartphones. We present three strategies for the production of location proofs with increasing tamper-resistance, two of which are based on Wi-Fi and a third based on physical interaction with kiosk-like devices. Our system was designed with user privacy and security in mind, minimizing the amount of connections between devices. A prototype application was implemented to assess the feasibility and reliability of the architecture and location proof strategies. The application allows rewarding users who complete a touristic route, with proofs of visit collected along the way.

Keywords: Location Proof · Context-Awareness · Mobile Security · Internet of Things

1 Introduction

In the coming years, the amount of Internet-connected devices will increase by orders of magnitude. These sensors and actuators will connect the physical and virtual worlds, constituting an Internet of Things (IoT). Smartphones will play an important role as user interfaces between people and the IoT devices.

Many mobile IoT applications use location context to provide their core functionality or to augment their capabilities [1]. These systems typically do not verify the location information they use, and are susceptible to *location spoofing attacks*. Developing the means to validate location information is, therefore, of high importance. *Location proof systems* differ from location systems in that they focus on countering location spoofing, by providing verifiable location information. The methods that can be used to produce location proofs depend on the available information sources and on the intended uses cases. One of the possible use cases for location proofs is in *Smart Tourism* [8] where tourists can use their personal devices to interact with existing or newly-added infrastructure in emblematic city locations. These interactions can then be used to verify location information allowing, for example, the implementation of reward schemes.

Wi-Fi can be used as infrastructure for location because most urban environments in populated areas tend to have many Wi-Fi networks. Some of these are for private or institutional use, while others are open for the general public to use. Nevertheless, the overwhelming majority of these networks announce their presence and can be detected using commodity smartphones.

In this paper we propose CROSS, a system that uses the Wi-Fi networks present in a predefined set of locations, to both detect the presence of the user in these locations, and to verify that the user is not spoofing his location. This information is used, in the example application, to ascertain whether the user completed any predefined tourism routes. The smart tourism application runs on the smartphones of tourists. The system uses Wi-Fi to determine whether the user is present at a location, using techniques that allow the implementation of location proofs without degrading the user experience.

The rest of this paper is organized as follows. An overview of the system and its operation is presented in Section 2. In Section 3, we propose three different location proof strategies. Section 4 presents the prototype used to validate our proposal. The evaluation is presented in section 5. Section 6 presents a brief comparison with existing works on location proofs. Section 7 concludes the paper.

2 System overview

CROSS has four main components: client application, server accessed through API, Wi-Fi Access Point (for proof strategy described in 3.2), and Kiosk (for proof strategy described in 3.3). The CROSS system uses a client-server model with no peer-to-peer communication between clients. This has advantages from a security and user experience standpoint, which we will detail later.

The system operation is represented in Figure 1. A tourist installs the smartphone application and signs up for an account. Before starting the trip, the application downloads the catalog of locations. The application logs visits to locations, illustrated in the figure as points P1 through P4. The location sensing relies on Wi-Fi and takes advantage of the scans regularly performed by the mobile operating system. At the end of the trip, the application submits the collected information to the server, and rewards will be issued.

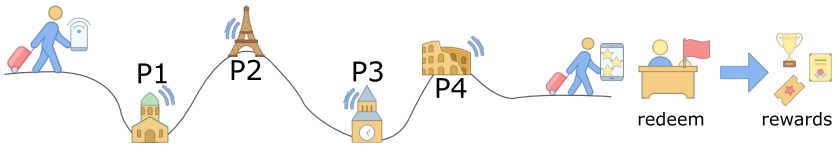

Fig. 1. User flow throughout a tourism route with four locations.

The *catalog* is stored on the smartphone to allow offline operation. It contains information about the registered locations, routes and rewards. It also contains

the BSSIDs[1] for a subset of the Wi-Fi networks that can be found at each location. The application uses this subset, which we call *triggers*, to sense the location it is at, and set off the collection of Wi-Fi-based location proofs. The ability to operate offline is important, as the tourists may be roaming without a data plan, or the cellular coverage may be insufficient at some locations.

The server validates location proofs submitted by the client and issues rewards, if the user is entitled to them. For each claimed visit to a location, the server computes a *strength score* based on the proofs backing the visit. This value is calculated differently for each location, depending on the proof strategy used. This score is also modified according to the characteristics of the movement of the user, as additional security mechanism. A penalty is applied to trips where the user moved faster between locations than is humanly possible. The intention is to thwart attackers who are able to forge at least some of the elements used for proof production, such as Wi-Fi scan results. In the definition of a route, each location is associated with a minimum strength score and a minimum visit duration. The user will receive a reward for a given route if the collected proofs match or exceed the minimum values acceptable for each point in the route.

3 Location proof strategies

We propose three different strategies for location proof production and verification, with increasingly stronger guarantees. The first strategy, *scavenging*, relies solely on existing Wi-Fi networks. The second strategy, *TOTP* (Time-based One-time Password), relies on Wi-Fi infrastructure deployed and configured specifically for use with CROSS. The third strategy, *Kiosk*, requires users to physically interact with an electronic kiosk booth.

3.1 Scavenging strategy

The idea for this strategy is to harness the large amount of Wi-Fi networks installed by third parties in urban environments. These networks may appear and disappear at any time. In this strategy, represented in Figure 2, location proofs are produced simply by storing Wi-Fi scan results with associated timestamps. These results are then submitted as part of the trip log.

On the server side, the set of Wi-Fi networks present in the scan results is compared with a list of known networks for each location. This list is curated by the system operators. The server periodically analyzes past location proofs to suggest the addition and removal of certain Wi-Fi networks from the list. The *strength score* is the fraction of client-presented networks over the total number of server-known networks.

The main advantage of the scavenging strategy is its simplicity and reduced setup cost, as it just uses existing infrastructure. However, it is also the strategy that provides the weakest guarantees: as soon as the list of networks at a certain location is known, an attacker can forge trip logs.

[1] Basic Service Set Identifiers, normally the address of the radio of the Access Point

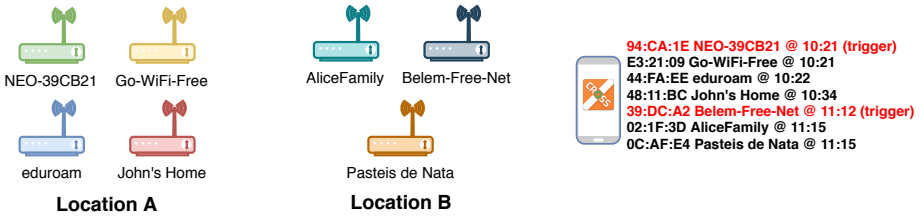

Fig. 2. Representation of the networks and logs in a visit to two locations, A and B, where the scavenging strategy is used. Locations are sensed using trigger networks.

3.2 TOTP strategy

This strategy allows for stronger proofs by deploying a customized Wi-Fi access point that is dynamically changing the broadcast SSID[2], depicted in Figure 3. The SSID is used as a low-bandwidth, unidirectional communication channel to transmit a changing value: the digits in the network name. This strategy is standards-compliant and compatible with existing devices. Note that the device observes the changing SSID values and does not need to connect to the network.

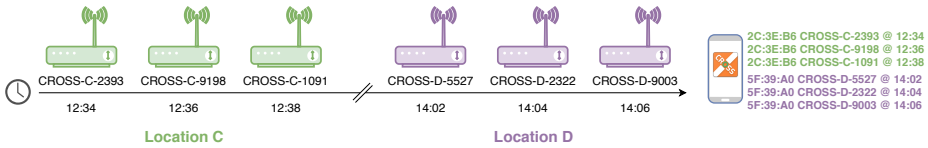

Fig. 3. Representation of the networks and logged information in a visit to two locations, C and D, where the TOTP strategy is used. There is one AP at each location.

Time-based SSID setting The SSID should change in a way that is unpredictable to an observer, but which can be verified by the server. We achieve this by including a Time-based One-Time Password (TOTP), similar to the proposed in RFC 6238 [9], in the SSID value. Only the Wi-Fi Access Point (AP) and the CROSS server know the TOTP secret, to produce and validate OTPs, respectively. Each AP should use a different secret key, and only the server should know the keys used by all APs. The APs and server must have synchronized clocks with minute granularity, but both components do not need to communicate, which means APs can function as stand-alone beacons in locations without Internet access.

Our solution uses a carefully selected time-step size and hash algorithm, that are different from those recommended in RFC 6238, as our use case is different from the typical TOTP use case where the one-time password acts as

[2] Service Set Identifier, the user-facing name for a Wi-Fi network

a second authentication factor. We use a time-step size of 120 seconds, sufficient to provide enough resolution during proof verification, while still fitting within the constraints of most Wi-Fi Stations when it comes to updating scan results. We chose SHA-512 HMAC as the TOTP hash algorithm, with keys as long as the HMAC output, instead of the typically used SHA-1 HMAC. This allows the use of longer keys for additional security. This algorithm was selected to ensure that it is computationally complex to perform a key-recovery attack, i.e., infer the secret TOTP key by continuously observing the different SSIDs assumed by the AP. To the best of our knowledge, such an attack against SHA-512 HMAC is yet to be achieved [5], unlike HMAC using weaker hash algorithms [4].

Proof collection and validation Clients are programmed to log all the different SSIDs a Wi-Fi network assumes during their visit to a location, along with the observation timestamp. Clients do not know whether each Wi-Fi network is part of the infrastructure for this strategy, as that is irrelevant to how they collect proofs; only the server needs to know this, to select the correct proof validation strategy. In other words, as far as the client implementation is concerned, the scavenging strategy and the TOTP strategy are the same.

The TOTP strategy, unlike the scavenging one, allows for attesting not just that the user was present at a certain location, but also that he did so at a certain point in time. Therefore, this strategy can verify the visit duration. Here, the strength score corresponds to the fraction of visit time that could be verified, in relation to the total time the client claims to have been present at the location. For example, if the client claims to have been present at a location for 20 minutes, but only 7 OTPs could be verified, corresponding to a total of 14 minutes within the claimed 20 minutes period, the strength score will be 70%.

Whenever the TOTP strategy is set up at a location, it supersedes the scavenging one, as it provides stronger guarantees. This way, updating the list of networks is not a concern for locations where custom APs are installed.

Validating the authenticity of Wi-Fi and Bluetooth devices is hard because the hardware identifiers can be spoofed. Because this solution does not involve bi-directional communication with other devices or networks, as in many witness-based proof strategies [12], it minimizes user exposure to attacks. This also protects their privacy, as only the entity operating the CROSS server will be able to know the locations visited by the user.

3.3 Kiosk strategy

The TOTP strategy prevents the attacker from creating new proofs on the fly, but not from replaying proofs from a legitimate visit under a different user account, or tunneling the information to a distant user. The kiosk strategy counters the possibility of claiming multiple rewards for a single trip, by preventing variants of Sybil attacks, where a malicious visitor creates multiple user accounts and runs them in parallel using one or more smartphones.

This strategy requires the tourist to interact with a machine present at the location - the *kiosk* - in order to prove his presence. In CROSS, the main function

of the kiosk is to sign a message for the CROSS client, logged on the account of the user and running on his smartphone. The kiosk can have other functionality unrelated to CROSS, including showing advertising or information about the location. Existing tourism information kiosks can be adapted for this purpose.

Proof production and validation Similarly to Wi-Fi APs in the TOTP strategy, kiosks are required to have their clocks synchronized with the server, also with minute granularity. Each kiosk keeps a private key, used to sign information. The server has the corresponding public key. Kiosks do not need to have a network connection to the server.

Location proofs are produced as follows. The client application sends the username of the logged in user to the kiosk, by displaying a QR code that is scanned by the kiosk. The latter, using its private key, signs a message containing the kiosk ID, the username of the user, the current date and time, and a randomly generated large number (a nonce). This message and respective signature is sent back to the client, again using a QR code, which is scanned by the smartphone built-in camera.

The smartphone stores this data as a visit proof. When the trip log is submitted to the server, it verifies this proof by checking the signed message using the public key associated with the kiosk at the visit location; the username matches the user account submitting the proof; the date and time is contained within the period of the visit; the nonce was not reused from any other visit proof submitted in the past. The signed timestamp allows for limiting the validity of the proofs, eliminating the need to store nonces for a long period.

By eliminating the remote network connection to the kiosk, an attacker must be physically present at the location to interact with it using QR codes. This physical interaction is essential to prevent Sybil attacks [6]. It can easily be inspected by a bystander, e.g. a tourist attraction staff member, who can check the behavior of the users for any suspicious activity, e.g. attempting to check-in with more than one device, or using the same device to check in multiple times, using different user accounts. An additional security mechanism could be the collection of ad-hoc witness reports from users in the same location, as described in [7]. However, this will not be necessary for this use case because kiosk devices can act as trusted witnesses.

The inconvenience for the user, and the kiosk setup cost for the system operators, can be greatly minimized by limiting the use of this strategy to a few locations per trip, where there are already tourist support infrastructures.

4 Prototype

To validate our solution, we developed prototypes of the client, server and trusted Wi-Fi AP components to evaluate the scavenging and TOTP strategies.

The client prototype is an Android application written in Java, compatible with off-the-shelf smartphones running Android from 4.4 up to 8.1. The client uses a SQLite database to store the catalog for offline operation, and to store trip

logs and respective location proofs, for opportunistic submission on the server. The server is written in Go and uses a PostgreSQL database to store information about locations, tourism routes, rewards, and the Wi-Fi networks present at each location, including TOTP secrets for trusted APs. Most importantly, the database is used to store user credentials and trip logs including the respective location proofs. The server exposes a REST API, with JSON payloads, which the client uses to obtain the catalog, and to submit trip logs. The Wi-Fi AP component was implemented using a ESP8266 board, a low-cost Wi-Fi microchip with full TCP/IP stack. The firmware was written in C++ using the Arduino environment for this microchip.

5 Evaluation

Preliminary versions of the prototype have been tested with a limited amount of users. A brief demonstration of the system to the general public was conducted in a public event, where users could install the client application on their own smartphones, and receive a reward for completing a short route. This route made use of both the scavenging and TOTP strategies, both of which proved to be viable.

5.1 Location detection accuracy and power consumption

Our system relies exclusively on Wi-Fi to detect its proximity to each location, so it is limited by the ability of the devices to accurately detect Wi-Fi networks. There are many factors that reduce the accuracy of the system, among them: AP transmit power, receiver sensitivity, amount of networks and interference sources in an area, signal propagation patterns, and the ability of the Wi-Fi station to display scan results in real-time (a minority of phones have delays presenting updated scan results). To ensure correct operation, the visits to locations should have a minimum duration of five minutes, which is perfectly suitable for the tourism use case. However, currently our solution is not well-suited for other applications that require shorter duration of visits.

In regard to power consumption, the prototype shows that our application can mostly piggyback on Wi-Fi scans already performed by Android and, as such, its overhead is negligible.

5.2 Security assessment

We consider an attacker model given the following attacker goals:

- *Tamper*: obtain more rewards than he is entitled to, considering the routes actually completed;
- *DoS* (Denial-of-Service): Disrupt the CROSS system (denial of service), e.g., preventing other users from receiving rewards;
- *Hijack*: Attack CROSS users through the CROSS infrastructure, by e.g. using it to spread malware.

The security properties we want to ensure are the *confidentiality* of the proofs and trip logs, which should only be accessible by their respective authors and by the system operators, the *integrity* of this information, which must not be tampered with by other users, and the *availability* of the system, so that all users can be rewarded by their visits.

To model different types of attackers, we considered the capabilities presented in Table 1, divided in sets **A** through **D**. **A** focuses on server control, **B** affects the connection between clients and server, **C** consists on client control and client-side tampering, and **D** are generic infrastructure attacks.

Table 1. Attacker capabilities considered in our security assessment.

A1	Read all the information in the server database.
A2	Take control over the server.
B1	Record all communication between the server and his client.
B2	Record all communication between the server and any client.
B3	Send requests to the server, posing as a client, using alternative software.
B4	Suppress communication between the server and any client, or redirect the client to a rogue server.
C1	Make the client application believe any Wi-Fi network is nearby.
C2	Know all the networks at a given location, at a certain point in the past.
C3	Know all the networks at a given location, at present time, without being at location.
C4	Set up rogue Wi-Fi Access Points.
C5	Know the OTP secret key of a AP used in the TOTP strategy.
C6	Pretend to be multiple users when at a kiosk.
C7	Know the private key of a kiosk.
C8	Set up fake kiosks.
D1	Send Wi-Fi deauthentication packets.
D2	Attack devices connected to the same network.

Capability A1 can be acquired by discovering, for example, a vulnerability in the server REST API that lets the attacker perform arbitrary SQL read-only queries. A1 attackers are not able to change the data associated with other user accounts or impersonate users (passwords and API tokens are hashed and salted) but they will immediately gain capabilities C2, C5, and, partially, B2, as the database data allows for inferring much of the communication with other clients. Because the server does not have the private key for each kiosk, it is still impossible for A1 attackers to break the third proof strategy.

A2 attackers essentially become as powerful as the system operators, and can manipulate the system to their own will, and are able to achieve all three attacker goals. In the CROSS security model, the server is the trust anchor, and therefore, it is not designed to counter attackers with capabilities A1 or A2. Attackers with capability A2 are not able to produce kiosk proofs, so while they achieve the *Tamper* goal, this tampering will be obvious if the proofs are independently audited.

Capability B1 can be acquired by compromising the connection of a single client with a man-in-the-middle, and B2 by compromising the server connection. As long as B1 or B2 attackers are not able to obtain the clear-text content of the HTTPS connections between the server and the client, they will only be able to violate the user privacy by inferring usage patterns. They will not be able to achieve any of the three goals.

Capability B3 can be obtained relatively easily, by reverse-engineering the client application to obtain the API address and request format. By itself, this capability does not let the attacker achieve any goals, but it is useful in conjunction with some of the C capabilities.

Capability B4 will let the attacker perform selective DoS, as well as obtain the credentials of clients connecting to the rogue server. To set up a rogue server, the attacker will need to obtain a valid TLS certificate for the fully-qualified domain name of the server that matches the pinned certificate in the client application. The most likely way to do this is to compromise the server and obtain the private key of the certificate, i.e. obtain capability A2, at which point setting up a rogue server is a pointless exercise.

C1 can be used in conjunction with C2, C3 and C5 in order to produce illegitimate visit proofs. With capability C2, an attacker can produce valid proofs for any location that uses the scavenging strategy, and can produce valid TOTP strategy proofs for the limited period in time when the list of APs was obtained.

With capability C3, which can be acquired using signal amplification or by tunneling information from the attraction location to the attacker's location, an attacker can produce valid proofs for both the scavenging and TOTP strategies. In conjunction with capability B3 or C1, they will be able to achieve the *Tamper* goal with routes that do not use the kiosk strategy.

An attacker with capability C4 can achieve the *DoS* goal by faking triggers, making client applications believe they are at a different location, breaking the scavenging strategy. They can also break the TOTP strategy by making clients collect SSIDs with nonsense OTPs, producing invalid proofs that will be rejected by the server. This capability does not help attackers achieve the *Hijack* goal, as clients never automatically connect to any of the CROSS Wi-Fi networks.

An attacker that acquires capability C5 will be able to fraudulently prove his presence at the location of the AP, at any past and future point in time, until a new key is generated for that AP. This allows him to achieve the *Tamper* goal, only if the stronger kiosk strategy is not used in some other point of the route.

Capability C6 will let an attacker produce valid proofs for the strongest strategy. In conjunction with capability C4 and C1, this lets the attacker achieve the *Tamper* goal with any route. We believe that acquiring this capability is complex, as long as users are monitored by a supervisor when operating the kiosks. An alternative capability that nets the same results is C7, which can only be acquired by tampering with the kiosk to extract its key, or compromising the key when it was configured in the kiosk.

Similarly to C4 with regards to the TOTP strategy, an attacker with capability C8 will achieve the *DoS* goal by tricking clients into collecting invalid proofs which will then cause the rejection of the trip log.

Capabilities D1 and D2 do not affect any of the proof strategies, as they do not require the user to be connected to a network. This is one of the advantages of not using peer-to-peer communication between devices: because no connections are established, the potential for attacks and privacy invasion is much smaller. However, D1 and D2 can help the attacker achieve the *DoS* goal, if they prevent other users from submitting their trip logs.

Regarding privacy, the client application only collects proofs when enabled by the user, users can only see their own trip log, and because proof production methods do not involve communication between users, there are less channels through which privacy could be compromised. System operators can see the trip logs of all users, but the only location information in them is relative to the predefined set of tourist attractions. It is not possible for the operators to follow the movements of the users as they go home, for example.

Overall, we can conclude that the security mechanisms in place for the smart tourism application are well suited. The cost of attacks with the *Tamper* goal exceeds the value of the intended rewards, which are small (e.g. a value of up to EUR 2) and will likely consist on discount coupons whose full redemption will require a purchase. The cost of *DoS* is high and there is no clear benefit for a specific attacker. The benefit of *Hijack* for an attacker is limited, because most times there will be no direct device interaction – the device sees the networks but does not connect to them – or is limited to the scanning of a data QR code that cannot be interpreted as something different inside the application.

5.3 Limitations

The scavenging and TOTP strategies are limited by the Wi-Fi capabilities of each device, as detailed in section 5.1. The scavenging strategy provides weak security guarantees, as its proofs can be forged. The TOTP strategy is stronger, but still allows proofs for each time period to be reused by different user accounts. It is also vulnerable to denial of service attacks, where clients collect invalid SSIDs broadcast by impostor Access Points. The kiosk strategy overcomes these limitations and provides much stronger guarantees, but it is still vulnerable to DoS attacks, even if they will require much more effort from the attacker with no clear benefit for him.

6 Related work

Most works in the field of location proofs focus on providing strong guarantees, often using complex cryptography schemes for proof production and verification. These can be used, for example, to implement authentication schemes, to limit the geographical availability of services, to aid in identity verification or to combat tax evasion [10]. However, these systems can sometimes be obtrusive,

requiring the user to perform unnatural actions when using their software and hardware. This is undesirable in a tourism application, which should be able to work using the platforms available today, without impairing the user experience.

Witness-based systems such as APPLAUS [12], LINK [11] and SureThing [7] typically use peer-to-peer communication between witnesses. However, peer-to-peer is increasingly hampered by current consumer-oriented mobile operating systems (iOS and Android), which are heavily oriented towards client-server communication models, as they place few restrictions on Internet access while forbidding or requiring special permissions to access the peer-to-peer features of Wi-Fi and Bluetooth radios, ultimately resulting in a poor user experience if one wishes to use these capabilities.

Systems which rely solely on mobile witnesses, without fixed infrastructure, require a minimum amount of diverse users at each location to work. The CREPUSCOLO [3] system solves this problem by introducing trusted witnesses that are installed on specific locations. The custom Wi-Fi APs and kiosks, presented in our second and third strategies, play a similar role to these trusted witnesses.

Distance-bounding protocols, as introduced in [2], are often used to counter tunneling and signal amplification attacks. These are unsuitable for our use case, as Android is not a real-time operating system and does not provide low-level hardware access, making it impossible to distinguish between random scheduler delays and those caused by such attacks.

7 Conclusion and future work

In this paper we presented CROSS, a system that uses three different location proof strategies using Wi-Fi networks, with increasing tamper-resistance. The system demonstrates the feasibility of location proofs in consumer-oriented mobile applications, running in current mobile operating systems and hardware without special privileges or configurations. This is a novel contribution that allows trade-offs between strong security guarantees and easier user experience and the ability to work without witnesses or peer-to-peer connections.

Future work will focus on extending the evaluation of the prototype. We will collect more measurements in a diverse sample of smartphone devices, and survey end-users about its usability. We will also collect data to further assess the utility of the scavenging strategy. Specifically, we will collect Wi-Fi data in the top Lisbon attractions at different times. The data set will be open and shared with the research community.

The kiosk strategy will be fully implemented and will have a mutual authentication scheme between clients and legitimate kiosks. The system will provide integration interfaces to allow it to be coupled with tourist ticket offices, that will play the role of kiosk, further leveraging existing infrastructure and personnel.

We will also investigate other location proof strategies for different use cases. An alternative to requiring a physical interaction would be to require a strong identity link between CROSS accounts and a real-world identity, for example, by requiring accounts to have a phone number associated, which would greatly

increase the barrier to creating multiple accounts. However, this approach has privacy drawbacks that will have to be mitigated.

Acknowledgements

This work was supported by national funds through Fundação para a Ciência e a Tecnologia (FCT) with reference UID/CEC/50021/2019 (INESC-ID) and through project with reference PTDC/CCI-COM/31440/2017 (SureThing).

References

1. Baldauf, M., Dustdar, S., Rosenberg, F.: A survey on context-aware systems. International Journal of Ad Hoc and Ubiquitous Computing **2**(4), 263 (2007). https://doi.org/10.1504/ijahuc.2007.014070
2. Brands, S., Chaum, D.: Distance-bounding protocols. In: Advances in Cryptology — EUROCRYPT '93, pp. 344–359. Springer Berlin Heidelberg (1993). https://doi.org/10.1007/3-540-48285-7_30
3. Canlar, E.S., Conti, M., Crispo, B., Pietro, R.D.: CREPUSCOLO: A collusion resistant privacy preserving location verification system. In: 2013 International Conference on Risks and Security of Internet and Systems (CRiSIS). IEEE (oct 2013). https://doi.org/10.1109/crisis.2013.6766357
4. Contini, S., Yin, Y.L.: Forgery and partial key-recovery attacks on HMAC and NMAC using hash collisions. In: Advances in Cryptology – ASIACRYPT 2006, pp. 37–53. Springer Berlin Heidelberg (2006). https://doi.org/10.1007/11935230_3
5. Dobraunig, C., Eichlseder, M., Mendel, F.: Security evaluation report on SHA-224, SHA-512/224, SHA-512/256, and the six SHA-3 functions. Tech. rep., CRYPTREC (2015)
6. Douceur, J.R.: The sybil attack. In: Revised Papers from the First International Workshop on Peer-to-Peer Systems. pp. 251–260. IPTPS '01, Springer-Verlag, London, UK, UK (2002), http://dl.acm.org/citation.cfm?id=646334.687813
7. Ferreira, J., Pardal, M.L.: Witness-based location proofs for mobile devices. In: 17th IEEE International Symposium on Network Computing and Applications (NCA) (Nov 2018)
8. Gretzel, U., Sigala, M., Xiang, Z., Koo, C.: Smart tourism: foundations and developments. Electronic Markets **25**(3), 179–188 (aug 2015). https://doi.org/10.1007/s12525-015-0196-8
9. M'Raihi, D., Machani, S., Pei, M., Rydell, J.: TOTP: Time-Based One-Time Password Algorithm. RFC 6238, RFC Editor (May 2011), https://www.rfc-editor.org/rfc/rfc6238.txt
10. Saroiu, S., Wolman, A.: Enabling new mobile applications with location proofs. In: Proceedings of the 10th workshop on Mobile Computing Systems and Applications - HotMobile '09. ACM Press (2009). https://doi.org/10.1145/1514411.1514414
11. Talasila, M., Curtmola, R., Borcea, C.: LINK: Location verification through immediate neighbors knowledge. In: Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, pp. 210–223. Springer Berlin Heidelberg (2012). https://doi.org/10.1007/978-3-642-29154-8_18
12. Zhu, Z., Cao, G.: APPLAUS: A privacy-preserving location proof updating system for location-based services. In: 2011 Proceedings IEEE INFOCOM. IEEE (apr 2011). https://doi.org/10.1109/infcom.2011.5934991