Proving Your Location Without Giving up Your Privacy

Wanying Luo & Urs Hengartner
Cheriton School of Computer Science
University of Waterloo
{w8luo,uhengart}@cs.uwaterloo.ca

ABSTRACT

Although location-based applications have existed for several years, verifying the correctness of a user's claimed location is a challenge that has only recently gained attention in the research community. Existing architectures for the generation and verification of such location proofs have limited flexibility. For example, they do not support the proactive gathering of location proofs, where, at the time of acquiring a location proof, a user does not yet know for which application or service she will use this proof. Supporting proactive location proofs is challenging because these proofs might enable proof issuers to track a user or they might violate a user's location privacy by revealing more information about a user's location than strictly necessary to an application. We present six essential design goals that a flexible location proof architecture should meet. Furthermore, we introduce a location proof architecture that realizes our design goals and that includes user anonymity and location privacy as key design components, as opposed to previous proposals. Finally, we demonstrate how some of the design goals can be achieved by adopting proper cryptographic techniques.

1. INTRODUCTION

Mobile devices, such as smartphones and PDAs, are playing an increasingly important role in people's lives. Location-based applications take advantage of user location information and provide mobile users with a unique style of resource and service offerings. Today, it typically is a user's mobile device that determines the device's location (e.g., using GPS) and that sends the location to an application. This approach makes it possible for a user to cheat by having his device transmit a fake location, which might let the user access a protected resource erroneously. Therefore, an application might ask a device to prove that the device really is or was at the claimed location. For example, a hospital might limit access to patient information to doctors or nurses who can prove that they are in (maybe a particular area of) the hospital. Content that is generated by mobile users might be geotagged, and people who download the content should be able to verify the location claim associated with the content [6]. A company might want proof from its repairmen that they really followed a prescribed route during the day. An insurance company might hand out discounts to drivers who can prove that they take non–accident-prone routes for their daily commutes. A person accused of committing a crime is very much interested in being able to prove to the police that he was somewhere other than the crime scene at the time the crime was committed [7].

A *location proof* is an electronic form of document that certifies someone's presence at a certain location at some point in time. A *location proof architecture* is a mechanism with which mobile users can obtain location proofs from proof issuers and with which applications can verify the validity of these proofs. In order to be truly useful, a location proof architecture must be flexible. For example, in some application scenarios, such as the insurance or police scenarios mentioned above, users might not know while being at a particular location that they will need a proof for having been at this location later on. Therefore, it must be possible for users to gather location proofs *proactively*. However, the proactive gathering of location proofs must be done carefully, otherwise proof issuers can track users and people's privacy will be in jeopardy. Moreover, different applications have different requirements for the contents of a location proof, such as the granularity of the certified location. For example, an insurance company might want to know only that a client drives around mainly in sedate Waterloo (as opposed to busy Toronto), but not where exactly in Waterloo. When a user does not know about the application that a location proof will be used for, she also does not know about the location granularity that will be required by the application. Including fine-grained location information in any location proof would solve this problem. However, presenting such a proof to an application might reveal more information than necessary about the user, and her privacy would get violated.

Previous research has recognised the need for location proof architectures (e.g., [6, 7]). However, none of the existing architectures is flexible enough to support all envisioned use cases. We make the following contributions:

- We put forward six design goals for a flexible location proof architecture. For each design goal, we elaborate on its importance, how it contributes to the practicality and usefulness of the architecture, and how it potentially conflicts with other design goals.
- We present a location proof architecture that realizes

HotMobile 2010, February 22–23, 2010, Annapolis, Maryland, USA.

our design goals. The architecture can be deployed on existing WiFi access points (APs) and is usable by regular mobile users. Moreover, the architecture demonstrates how cryptography can improve user privacy and system security. We are implementing and deploying our architecture on a building-wide scale.

The rest of the paper is organised as follows: We first motivate potential applications that benefit from location proofs in section 2. Our design goals are specified in section 3. We present our threat model in section 4, which is followed by our location proof architecture in section 5. Section 6 briefly discusses our implementation. We summarise related work in section 7 and conclude in section 8.

2. MOTIVATION

Location-based access control: Role-based access control is not always sufficient in restricting access to confidential information. For example, a doctor or a nurse should never be allowed to access patients' medical records outside hospital buildings. In this kind of scenario, information access should also take users' locations into account. Similarly, a company that stores confidential customer information or a crucial trade secret may forbid its employees from accessing the company's database anywhere other than inside the company premises, in fear that its employees may deliberately or accidentally leak sensitive information.

Location-based social networking: Online social networks allow users to form social groups based on their geographical locations. Users are often bombarded by too many unsolicited add-friend requests. In order to be sure that an add-friend request comes from a person in the required geographical region, the user can ask for a location proof from the request sender and accepts the request only if the sender is able to present a valid location proof.

Related work [4, 5, 6, 7, 8, 9] lists some more applications that use location proofs.

3. DESIGN GOALS

In order for a location proof architecture to be useful and deployable in practice, the design of such an architecture should meet the following requirements.

3.1 Scalability

Whereas some earlier work has suggested to let a central entity issue location proofs [6], this approach does not scale and makes deployment difficult. Instead, similar to Saroiu and Wolman [7], we envision that individual APs or cell towers issue location proofs. However, an AP or cell tower usually needs to serve many users. If issuing a location proof demands hefty computation, the performance will decline as the number of users increases, and the architecture will not scale. Furthermore, mobile devices have far less processing power than a desktop computer, and processing uses up battery power. Therefore, a location proof architecture that involves intensive computation is impractical for both APs and mobile devices. Although adopting certain cryptographic operations may be necessary in a location proof architecture, we should minimise their usage.

3.2 Application-Agnostic Location Proofs

A number of mobile applications are available nowadays, and each location-proof-based application might use location proofs in different ways. If a single location proof can only be used for a specific application, a user has to obtain several location proofs when she wants to prove her location to many applications. Each proof would be crafted to follow the requirements of a particular application. Whereas this approach is adopted in some earlier work [9], we believe that location proofs should be produced in an application-agnostic manner such that the same proof can be repeatedly used by the user and accepted by a variety of applications. This reduces the workload for proof issuers dramatically and benefits users for two reasons: a general location proof reduces storage space that a user has to spend on storing location proofs; moreover, a general proof format relieves users of the burden of having to constantly interact with location proof issuers, and allows them to spend more battery power and time on other business.

3.3 Proactive Location Proofs

Most previous research assumes that there are three parties involved in a location proof architecture: a user who intends to prove her location, a verifier (or an application in other words) that offers a certain service to users who possess a valid location proof, and a location proof issuer who gives out location proofs. But more often than not, users may not have a target application in mind and simply want to collect location proofs for future use, that is, a verifier will be involved only later. For example, a spouse or an employee might get wrongly accused by her partner or employer, respectively, of having been at a certain location; here, location proofs gathered during the day can help resolve these accusations. A shopper might learn only after having left a shop that the shop offered discount coupons for future purchases to people in the store; here, with the help of a location proof gathered while being in the store, the shopper can still retrieve these coupons. In all these scenarios, users have an incentive to collect location proofs continuously, in particular, before using an application and maybe even before knowing about the application. Therefore, we distinguish between proactive and retroactive location proofs:

- **A retroactive location proof** is a location proof that is requested by a user to interact with a target application.
- **A proactive location proof** is a location proof that is collected by a user for future purposes, without having a target application in mind.

Proactive location proofs tie in with application-agnostic proofs; since a user will not know for which application(s) a proof will be used for, location proofs should follow a general format and contain no application-specific data.

3.4 User Anonymity

An important factor that concerns many users before they adopt a new technology is whether their privacy will be undermined by using the technology. To be widely adopted, location proof architectures must invest every effort to guarantee user privacy. In particular, a user should be able to remain anonymous while acquiring location proofs. Anonymity becomes particularly important for proactive location proofs. Here, a user continuously gathers location proofs. If the (maybe colluding) parties that issue location proofs learn the user's identity, the user would become trackable.

To ensure user anonymity, some previous work [5] uses a broadcast-based approach to distribute location proofs. Anyone who captures a token broadcast by an AP can use the token as a location proof. However, captured tokens can be redistributed at will, which opens the door to abuse. Instead, we require that a location proof is tied to a specific user and can be used only by this user.

This problem is challenging in that it poses a dilemma to the architecture design: in order to vouch for a person's presence at a certain location, the location proof issuer has to know who it is vouching for; however, by knowing the identity of the person, the location proof issuer can track that person by her location, thus compromising her privacy.

3.5 Location Privacy

Intuitively, a location proof has to contain location information of some form. The location information in the proof not only vouches for a person's location, but might also indirectly reveal sensitive information about that person (e.g., her interests) to a verifier, and the person's location privacy could get violated. For example, if a service is offered only to mobile users in Waterloo, users who present a proof that shows that they are at the headquarter of the Liberal party in Waterloo not only prove their qualification to the service, but also reveal more location and personal information than necessary to the service provider. Therefore, the user should be given the ability to control how much location information to disclose in response to the location requirements of different applications and services. For retroactive location proofs, a user can inform the proof issuer of the required granularity for the location information contained in the proof. For proactive location proofs, this is not possible since the user does not yet know about the granularity required by an application.

Not only the location information contained in a location proof can reveal a person's location; so can the identity of the proof issuer. We assume that APs (or cell towers) issue location proofs. The location of APs is often available on the Internet. As a consequence, even if a proof contains only coarse-grained location information, it might still be possible to learn a user's fine-grained location simply by looking up the location of the AP issuing the proof on the Internet.

In summary, to maintain a user's location privacy, location proofs need to support granularity, where the required amount of granularity might be unknown at proof issuing time, and location proofs should not reveal the issuer's identity.

3.6 No Dedicated Hardware

One of the most significant limitations in many previous proposals of location proof architectures is their reliance on dedicated hardware. For example, the location proof protocol described by Sastry et al. [8] relies on the fact that nothing is faster than the speed of light in order to compute an upper-bound of a user's distance. This technique is called distance bounding [2]. Here, dedicated hardware is required to measure signal round trip time with very high precision and negligible processing delay. Putting dedicated hardware into users' cellphones is costly. While it is possible that future cellphones will have this hardware, we want our location proof architecture to be deployable immediately. Moreover, it is unrealistic to expect operators of existing APs to purchase and configure additional hardware to provide location proof services. Therefore, a location proof architecture should not rely on any dedicated hardware.

4. THREAT MODEL

We consider the following threats in our architecture:

- **Dishonest users**. A dishonest user tries to obtain location proofs that certify her presence at some place at a particular time even if she was not there. Dishonest users may achieve this goal by colluding with malicious intruders.
- **Malicious intruders**. A malicious intruder is not interested in obtaining location proofs for her own use but offers to help other users to get location proofs on their behalf in exchange for other benefits like money. We assume malicious intruders will not collude with dishonest users to launch wormhole attacks, which we address in detail below.
- **Curious APs and applications**. A curious AP tries to learn a user's identity while the user is acquiring a location proof from the AP. Similarly, a curious application tries to learn more location information from a location proof than it really needs.
- **Malicious applications**. A malicious application obtains location proofs from its users and then tries to take advantage of these proofs to get unauthorised access to other applications.
- **Active and passive eavesdroppers**. An eavesdropper records and maybe modifies communication between users, proof issuers, or applications.

We do not consider the following threats:

- **Wormhole attacks**. A wormhole attack takes place when a malicious party records network traffic in a region of the wireless network and replays it in another region. For example, suppose two wireless devices A and B are invisible to each other. A malicious device C within the transmission range of device A tunnels traffic from A to device B, which makes device B appear visible to A. This kind of attack is extremely hard to detect. In a location proof architecture, by launching wormhole attacks, a user may collude with several remote malicious intruders to simultaneously obtain location proofs from APs in different places. The only way to defeat wormhole attacks is to rely on dedicated hardware and distance bounding techniques as discussed in section 3.6. Since it violates our sixth design principle, we do not consider this option.
- **Weak identities**. Device carriers are not always the actual device owners. For example, a device may be stolen or lent to a friend by the owner. Therefore, our use of users' public keys as their identities, as suggested in section 5, may not be reliable, and this form of user identity is deemed a weak identity. Saroiu and Wolman [7] propose several alternatives to achieve stronger identities, but none of them is foolproof.

Both wormhole attacks and attacks that exploit weak identities can be instantiated in real wireless networks. Unfortunately there are no easily deployable solutions against either of them. Applications that need to protect access to highly valued resources should therefore not rely only on our location proof architecture for security. For other applications, these threats can be ignored since the attacks tend to be expensive to instantiate, and their cost might be higher than the value of the protected resource.

Granularity	1	2	3	4	5
Location	country	province	city	street	building

Table 1: Location granularity

5. LOCATION PROOF ARCHITECTURE

In this section we first discuss how we achieve our design goals. We then combine the pieces and present our location proof architecture, followed by a security analysis.

5.1 User Anonymity

Our solution for user anonymity is based on cryptographic hashes and digital signatures. Before issuing a location proof, the issuer generates a nonce and sends it to the user. The user concatenates the nonce with a nonce of her own and signs them. The user's nonce protects against attacks where the proof issuer chooses non-random nonces in order to try to make the user commit to a certain action.

Next, the user sends the hash of the signature and of her nonce to the issuer. The hash in combination with the user's nonce serves two purposes: First, they act as a commitment by the user to her signature. Once the commitment has been released, it becomes computationally infeasible for the user to find another signature/nonce combination with the same hash value. Second, it hides the user's signature and therefore her identity from the proof issuer. Without the user's nonce, the proof issuer cannot infer any information about the user's signature and therefore her identity.

The proof issuer then creates a location proof. A location proof is a statement signed by the issuer that includes the location of the proof issuer (e.g., an AP), the current time, the hash value received from the user, and the issuer's nonce. See Section 5.4 for the detailed protocol.

Later when the user submits the proof to an application and is required to prove that she is the intended recipient of the proof, she reveals her signature and her nonce. The application first checks whether the hash value of the signature and of the nonce matches the hash value in the proof. Then, it checks whether the signature covers the nonce in the proof and the nonce from the user. See Section 5.5 for the detailed protocol.

5.2 Location Privacy

To maintain a user's location privacy, a location proof issuer first defines several location granularity levels, such as the five levels shown in table 1. When a user sends a request for a retroactive location proof to a proof issuer, the user should also specify a granularity level so that the proof issuer can write location information of appropriate granularity into the proof. (We discuss changes required for proactive location proofs in Section 5.3.)

The location proof must be signed by the proof issuer. As mentioned in Section 3.5, unless a user asks for a proof listing her fine-grained location information, the proof issuer cannot use the issuer's private key for signing the proof. Otherwise, an application could infer the identity of the issuer and look up the issuer's location on the Internet. Therefore, we must hide the issuer's identity from the application. The immediate question is how can the application verify the signature on the proof if the application does not know who issued it? Group signature schemes [3] come to the rescue.

A group signature scheme allows a member of a group to sign a message on behalf of the group while staying anonymous. Anyone can verify the validity of the signature without learning who signed it. Only the group manager can learn the signer's identity. In practice, proof issuers belonging to a certain corporation or geographical region would form a group. For example, AT&T as a provider of WiFi hotspots may deploy a group signature system on all its APs. A proof issued by any member of the group will be verifiable by the application, but the application has no way of knowing which issuer created the proof. This way, AP's identities are hidden from the application.

5.3 Proactive Location Proofs

In the previous subsection, we assume that a user knows which application to interact with and thus knows which granularity to ask for. We defined this situation as a retroactive location proof in section 3.3. But sometimes the user may try to collect location proofs for future use, which means she is unlikely to have a target application in mind and thus is not clear what granularity to ask for. To support proactive location proofs, we modify the table in the previous section by adding a wildcard granularity level, *. If the user requests location proofs in a proactive manner, she should specify * as the desired granularity. On receiving such a request, the proof issuer encrypts the five granularity representations of the location with five different symmetric keys, and includes the five ciphertexts in the location proof. It then sends the signed proof with the five decryption keys to the user. When the user finally wants to present the proof to an application, she reveals the appropriate key to the application, which can decrypt location information of a specific granularity level, as required by the application.

5.4 Generation of Location Proof

Our protocol relies on cryptographic hashes, digital signatures, and symmetric-key encryption. We use $S_T(m)$ to represent message m signed with T's private key. As usual, we assume a hybrid signature scheme, where a message is first cryptographically hashed and the signature is computed for the hash value. Furthermore, we define $E_k(m)$ to represent message m encrypted using a probabilistic symmetric-key encryption algorithm with key k. We assume that all parties are in the possession of a public/private key pair, where the public key is signed by a Certification Authority (CA). Moreover, we assume that the private key is used for various purposes by a user, not only for location proofs, which increases the user's incentive to protect her private key and stops her from giving it to colluding users. Finally, we assume communication is secured against passive and active eavesdroppers with the help of TLS/SSL.

When a user is nearby an AP, she may execute the protocol in figure 1 to obtain location proofs from the AP. We use the user's public key as her identity, i.e. $\mathrm{ID}_{\mathrm{user}} = P_{\mathrm{user}}$, where P_{user} is the user's public key signed by the CA.

1. The user sends a location proof request to the AP. The request should contain a desired granularity g, where $g = 1, \ldots, 5$ (for retroactive proofs) or $g = *$ (for proactive proofs).
2. The AP generates nonce n_{AP} and sends it to the user.
3. The user generates nonce n_{user} and sends the following hash value to the AP:

$$H = \mathrm{hash}\left(S_{\mathrm{user}}\left(n_{\mathrm{user}} \parallel n_{\mathrm{AP}}\right) \parallel n_{\mathrm{user}}\right)$$

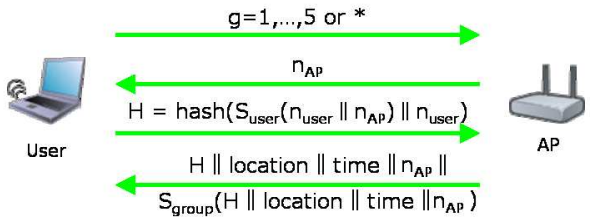

Figure 1: Location proof protocol

4. Finally, the AP creates a location proof with a group signature and sends the proof to the user. The proof is of the following format:

$$H \parallel \text{location} \parallel \text{time} \parallel n_{\text{AP}} \parallel$$
$$S_{\text{group}}\,(H \parallel \text{location} \parallel \text{time} \parallel n_{\text{AP}})$$

The "location" part of the proof is worth further clarification. If the user asks for a retroactive location proof by specifying a g value between one and five, the AP simply includes location information of that particular granularity. If the user requests a proactive proof, the "location" part becomes a concatenation of five ciphertexts, each of which is the encrypted form of location information of a particular granularity. That is

$$\text{location} = E_{k_1}\,(L_1) \parallel \cdots \parallel E_{k_5}\,(L_5)$$

where L_i is location information of granularity i, where $i = 1, \ldots, 5$. Moreover, the AP should also send five decryption keys, $k_1, \ldots, k_5$, to the user in this case.

5.5 Verification of Location Proof

The user submits $S_{\text{user}}\,(n_{\text{user}} \parallel n_{\text{AP}})$, n_{user}, P_{user} and the location proof to the application. The application needs to verify whether the user is in the possession of the private key corresponding to P_{user}, for example, by setting up a SSL/TLS connection with client authentication. The application then computes the following hash value:

$$H' = \text{hash}\,(S_{\text{user}}\,(n_{\text{user}} \parallel n_{\text{AP}}) \parallel n_{\text{user}})$$

If H' equals hash value H contained in the location proof, the application continues by verifying $S_{\text{user}}\,(n_{\text{user}} \parallel n_{\text{AP}})$ using P_{user} and n_{AP}, where the latter value can be retrieved from the location proof. Finally, the application checks the group signature in the location proof, where we assume that the application is in the possession of the required public key. In general, we assume that an application knows the public keys of organisations that are trusted by the application to issue location proofs. Only if all checks succeed, the proof will be accepted by the application.

5.6 Security Analysis

By including only a commitment to a user's signature in a location proof, but not the actual signature, a user remains anonymous while acquiring the proof. The commitment also prevents a malicious application from re-using a valid location proof obtained from a user for its own purposes. In particular, the application cannot find a signature/nonce combination with the same cryptographic hash value. The signature needs to be issued by the application and needs to cover nonce n_{AP}, as included in the location proof, and a nonce n_{user} of the application's choice. The only way for the malicious application to succeed here is to generate a signature of its own that is identical to the user's signature. However, this would imply that the application can forge signatures, which it cannot.

A dishonest user may collude with a malicious intruder to launch a replay attack in an attempt to acquire location proofs for a place where the dishonest user is no longer located. Suppose a dishonest user successfully and legitimately obtains a location proof for a particular location. The user then gives nonce n_{AP} and $S_{\text{user}}\,(n_{user} \parallel n_{AP})$ to a malicious intruder that is in the same area. The task of the malicious intruder is to acquire further location proofs from the same proof issuer on behalf of the dishonest user, who has moved away. The only way for the malicious intruder to succeed is to hope that the proof issuer is going to re-use nonce n_{AP}. However, since each nonce is used only once, the malicious intruder cannot succeed.

In the above attack, it is also impossible for the malicious intruder to sign a fresh nonce n_{AP} herself, because under the trust assumptions in section 5.4, she does not have the private key of the dishonest user. The malicious intruder may also try to set up a communication channel through which she can send a fresh nonce n_{AP} to the remote dishonest user to have him sign the nonce in real time. This is a wormhole attack, which we do not address in this paper.

Several APs could collude and track a user based on the MAC address of the user's device. However, for many devices, it is possible to change their MAC address, which makes tracking hard. Moreover, even if APs are able to track a user, our protocol still prevents them from learning the user's identity.

6. IMPLEMENTATION

We have implemented our architecture and are deploying it in a WiFi testbed [1]. The testbed covers two floors of the Davis Centre at the University of Waterloo. It consists of 38 APs that are connected to a central controller, which also issues location proofs in our implementation. This approach reduces load on the APs and simplifies deployment. Considering that in many business environments APs and authentication are centrally managed, it makes sense to have the controller also issue location proofs. Users request location proofs directly from a daemon running on the controller. Each AP in the testbed is also an IP router and implements NAT, which makes it easy for the daemon to learn the identity of the AP that a user is connecting to.

7. RELATED WORK

Denning and MacDoran [4] present a location-based authentication system where a location signature sensor (LSS) creates location signatures that describe the physical location of the LSS at a particular time. A user carrying an LSS can hand a location signature to an application, which learns the user's location based on the location signature. This system lacks a strong binding between the location signature and the user identity. Therefore, a user can sell location signatures to anyone. Moreover, the system relies on dedicated hardware and supports only reactive location proofs. In comparison, our architecture ties location proofs to specific users, does not require dedicated hardware, and supports also proactive location proofs.

Waters and Felten [9] introduce a system that allows a

device to obtain location proofs from a location manager (LM) and submit proofs to a verifier. A device requests location proofs by sending its device ID encrypted with the public key of the verifier. The LM then sends the device a nonce which the device immediately sends back upon reception. The LM measures the round-trip delay and sends the device a location proof containing the measured delay and the encrypted device ID. A drawback of this system is that a malicious intruder can craft fake device IDs and collect location proofs on behalf of dishonest users, which is not possible in our system. Moreover, Waters and Felten's system requires the device to know verifiers in advance and does not support proactive location proofs.

Faria and Cheriton [5] design a location-based authentication architecture for wireless LANs. A centralised wireless appliance (WA) controls a group of APs and broadcasts a set of random nonces through its controlled APs. To prove its closeness, a client must capture and send all the received nonces back to the WA. As argued in section 3.4, this broadcast-based approach for distributing location proofs is problematic, because no identity information about the client is included in a proof, which is why we do not pursue this approach in our solution.

Lenders et al. [6] describe a geotagging service that allows a content creator to obtain a location/time certificate for the content. Such a certificate proves the generation location and time of the content. There are two major drawbacks of this system compared to ours. First, their system relies on a single trusted location/time verification party, which could become a single point of failure, whereas in our architecture APs become proof issuers, thus making the architecture more robust. Second, the location/time certificates do not bind the content and the certificate to the content originator; in other words, anyone can claim or disclaim ownership of the content and its certificate. Our architecture achieves a strong binding between a location proof and its owner, while still keeping the anonymity of the owner to the proof issuer.

Saroiu and Wolman [7] propose a mechanism for mobile devices to acquire location proofs from APs. APs broadcast beacons to announce their presence. A user wishing to obtain a location proof must capture a beacon and extract a sequence number from it. The user then signs and returns the sequence number to the AP, which sends a location proof to the user. This design is problematic from a privacy point of view. First, to reduce the chance of being tracked, users are responsible for deciding when to request location proofs. But requiring a user to keep an eye on her own privacy is not only unrealistic but also error-prone. Moreover, this design makes the proactive collection of location proofs impossible due to privacy concerns. In our architecture, privacy protection is a fundamental component, which enables proactive location proofs. Second, Saroiu and Wolman's system always reveals a user's fine-grained location to an application, which can violate a user's location privacy.

8. CONCLUDING REMARKS

We formalised six essential goals that should govern the design of location proof architectures. We elaborated on the importance of these design goals with regard to their roles in system functionality and their implications on user privacy.

In addition, we put forward a location proof architecture that meets all design goals. We illustrated how cryptographic techniques can aid in our design.

As part of the deployment of our proposed architecture, we are going to evaluate it with several prototype applications, which will allow us to gain insights about the performance of the proposed architecture. Furthermore, the security of the architecture lacks formal proofs, which are important, but left for future work since they are out of scope here. Finally, finding defences against wormhole attacks that do not rely on dedicated hardware is another area of future work.

Acknowledgements

We thank the anonymous reviewers and our shepherd Ramón Cáceres for their helpful comments. This work is supported by the Natural Sciences and Engineering Research Council of Canada.

9. REFERENCES

[1] N. Ahmed and U. Ismail. Designing a high performance wlan testbed for centralized control. In *TridentCom 2009: Proc. of 5th International Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities*, pages 1–6, 2009.

[2] S. Brands and D. Chaum. Distance-bounding protocols. In *EUROCRYPT '93: Workshop on the theory and application of cryptographic techniques on Advances in cryptology*, pages 344–359, 1994.

[3] D. Chaum and E. van Heyst. Group signatures. In *EUROCRYPT '91: Workshop on the theory and application of cryptographic techniques on Advances in cryptology*, pages 257–265, April 1991.

[4] D. E. Denning and P. F. MacDoran. Location-based authentication: grounding cyberspace for better security. In *Internet besieged: countering cyberspace scofflaws*, pages 167–174, New York, NY, USA, 1998. ACM Press/Addison-Wesley Publishing Co.

[5] D. Faria and D. Cheriton. No long term secrets: Location based security in over provisioned wireless lans. In *HotNets-III: Proc. of the Third ACM Workshop on Hot Topics in Networks*, November 2004.

[6] V. Lenders, E. Koukoumidis, P. Zhang, and M. Martonosi. Location-based trust for mobile user-generated content: applications, challenges and implementations. In *HotMobile '08: Proc. of the 9th workshop on Mobile computing systems and applications*, pages 60–64, 2008.

[7] S. Saroiu and A. Wolman. Enabling new mobile applications with location proofs. In *HotMobile '09: Proc. of the 10th workshop on Mobile Computing Systems and Applications*, pages 1–6, 2009.

[8] N. Sastry, U. Shankar, and D. Wagner. Secure verification of location claims. In *WiSe '03: Proc. of the 2nd ACM workshop on Wireless security*, pages 1–10, 2003.

[9] B. Waters and E. Felten. Secure, private proofs of location. Technical Report TR-667-03, Department of Computer Science, Princeton University, January 2003.