PROPS : PRivacy-preserving lOcation Proof System

Sébastien Gambs[1], Marc-Olivier Killijian[2,3], Matthieu Roy[2,3], and Moussa Traoré[2,3⋆]

[1] Université de Rennes 1 - INRIA / IRISA
[2] CNRS, LAAS, 7 avenue du colonel Roche, F-31400 Toulouse, France
[3] Université de Toulouse, LAAS, F-31400 Toulouse, France

1 Introduction

Location-Based Services (*LBS*) personalize the service they provide or grant access to ressouces according to the current location of users [1]. They are used in a variety of contexts, such as geosocial network, real-time traffic monitoring, discount tied to the visit of a particular shop or local electronic election. In most of current schemes, the location of a user/device is determined by the device itself (*e.g.*, through GPS) and forwarded to the LBS provider. By doing so, a user can cheat by having his device transmitting a false location to gain access to unauthorized resources, thus raising the issue of verifying the position claimed by a particular user.

To counter this threat, a LBS should ideally require the requesting device to formally prove that it really is at the claimed location. This notion has been formalized through the concept of *location proof* [16], which is a digital certificate attesting that someone was at a particular location at a specific moment in time. A location proof architecture is a system by which users can obtain location proofs from neighboring witnesses (*e.g.*, trusted access points or other users) that can later be verified by other entities. In recent years, several location proof architectures have been proposed in the literature [19, 16, 9, 13, 20, 17, 6]. Most of these approaches require the users to disclose their identities or their positions to a centralized server, thus raising privacy issues such as the possibility of tracing the movements of users of the location proof architecture.

Therefore, a main challenge is to design a location proof system that can accommodate location requirements of different LBS while preserving location privacy of users. To address this challenge, we propose Props, a PRivacy-preserving lOcation Proof System aiming at preserving the privacy of all the participants involved in the system through a decentralized architecture.

The outline of the paper is the following. First in Section 2, we describe the entities participating to the location proof architecture and the desirable properties for a secure and privacy-aware location proof system before briefly presenting some of the building blocks upon which Props is constructed. Afterwards, we describe the architecture of PROPS in Section 3 and we give details about the protocols for the gathering of a location proof, its verification and the preliminaries results of our proof of concept implementation. Finally, we briefly conclude in Section 4.

2 Location Proof Specification and Building Blocks

2.1 Interacting Entities

Within Props, there are three types of participating entities: 1) Users, 2) the Certification Authority (CA) and 3) the Anonymity Lifter (AL). A user is an entity using the location proof service, and also refers at the same time to the device carried out by this individual. A user can take one or several of the following roles: prover, witness or verifier. More precisely, a *prover* denotes an individual who wants to collect anonymously location proofs from *witnesses*, which correspond to other users of the system that are within communication range. A *verifier* is a user of the system who wants to verify the validity of a location proof certificate released by a prover before granting him access to some resource. The Certification Authority (CA) is the trusted third party responsible for issuing the credentials to newly registered users. Finally, the Anonymity Lifter (AL) is a trusted third party that has the capacity to lift the anonymity of a particular user when needed (for instance upon request from a judge).

⋆ Contact author: mtraore@laas.fr

2.2 Security and Privacy Desiderata

In this section, we briefly define the fundamental properties that a secure and privacy-preserving location proof system should fulfill. We classified them in two categories, namely security requirements and privacy requirements.

Security requirements. A location proof should satisfy the *correctness* property, which we define by means of soundness and completeness properties. More precisely, a malicious prover should not be able to generate a proof for a location in which he has never been (*spatial soundness property*), or for a different instant that the one at which he was visiting this location (*temporal soundness property*). The correctness property has to remain valid despite possible attacks such as the *distance fraud* [2] or the *mafia fraud* [7] (*i.e.*, man-in-the-middle attack). Moreover, if the verifier is honest, he should always accept a location proof provided that the location proof is authentic (*completeness property*).

In addition, a location proof should be tied to his owner, in the sense that the genuine owner of the proof should be the only one able to convince a verifier that he was located at the position certified by the location proof (*non-transferability* and *ownership proof*). Moreover, it should also be impossible for a user to forge a fake location proof share (*unforgeability*), despite potential distance hijacking [5] or malicious users colluding together to generate a fake location proof. In addition of being unable to produce a fake location proof for one of them, a collusion of malicious users should not have the capacity to frame an honest (*i.e.*, non-colluding) user by generating a fake location proof that would make it appear as if this user was located at a particular location in which he was not.

Privacy requirements. From a privacy point of view, the location proof as well as the proof gathering and the proof verification protocols should not leak any information that can be used to identify the prover or his collaborating witnesses (*unlinkability and anonymity of prover and witnesses*). More precisely, it should not be possible to decide if two location proof requests have been sent by the same prover or to distinguish if two different proof shares have been issued by the same witness. Moreover, a witness participating to the generation of a location proof should not have to disclose his exact position (*witness location privacy*) and the messages exchanged between the entities of the system should be transmitted through a secure pairwise channel ensuring the confidentiality of the communications and the integrity of the data exchanged (*confidentiality and integrity*).

Additionally, a prover should be able to keep his location proofs under his full control and decide when he wants to show one of his location proof to a verifier. Moreover, he should be given the ability to control the granularity of the location information revealed during the verification process of the location proof (*location sovereignty*). Finally, the architecture should provide some mechanism to lift the anonymity of users in some specific situations upon a judge request (*optional anonymity removing*).

2.3 Building Blocks

PROPS relies on cryptographic primitives such as hash function, symmetric encryption and unique group signatures [8], which is an extension of group signature [4]. Within unique group signature, one can anonymously sign a message on behalf of the group (*i.e.*, his identity is hidden within the members of the group) unless he signs twice the same message in which case a detection algorithm can be used to link these two signatures. In PROPS, we assume the availability of a *proximity testing protocol* that can be used by a prover to convince a witness that he is located close to him in the same spirit as distance-bounding protocols [2, 10, 18, 11].

To implement the location sovereignty, we use hash chains [12] for the controlled release of the granularity of a location (which includes the spatial and temporal information) contained in a location proof. More precisely, the primitive based on hash chains is composed of three algorithms: 1) $\mathsf{Hide}(pos, seed)$, which is used by the prover to encode his precise location *pos* into a hash chain embedded in the location proof initialized with the random string *seed*, 2) $\mathsf{Reveal}(pos, p, seed)$ is used by the prover to reveal to a verifier a granularity p over his precise location *pos* included in the location proof, and finally 3) $\mathsf{Check}(K_{pos}, K_{pos}^{p}, pos|_p)$ is used by the verifier to verify if the location information disclosed by the prover matches the information encoded in the location. Finally, PROPS is also based on CL-signature [3] that allows a user to prove the knowledge of a certificate without revealing it.

3 PRivacy-preserving lOcation Proof System (PROPS)

3.1 Architecture

Thereafter, we will use the notation $\mathcal{UGS}$ [8] to refer to unique group signature scheme and $\mathcal{CLS}$ [3] to denote a CL-signature scheme, while $\bigwedge_{i=1}^{k} f_i(x)$ is used as a shortcut to $f_1(x) \wedge \ldots \wedge f_k(x)$ in which f_i is a boolean function and $\wedge$ represents the logic operator `AND`.

The architecture of PROPS consists of the following four different procedures: Init, Join, LocationProof-Gathering and LocationProofVerification. The procedure Init is run once by the Certification Authority (CA) in order to set up the global parameters of the location proof system. More precisely, the CA calls $\mathcal{UGS}.\mathsf{Init}(1^{\lambda})$, which corresponds to the initialization procedure of a unique group signature. This initialization procedure generates the credentials (sk, ik, ok, gpk) and (sk_{cert}, pk_{cert}) a private/public key compatible with a CL-signature. The public parameters of the system are defined as $pk = (gpk, pk_{cert})$. The CA keeps ik secret and will use it to register new users to the system during the Join procedure, while ok is sent to the Anonymity Lifter (AL) in order to be able to lift the anonymity of a user if needed.

Once the system has been correctly initialized, the CA can dynamically add users to PROPS using the Join procedure. At the end of this procedure, each user receives a tuple $(gsk[i], s_u, cert_u = \mathcal{CLS}.\mathsf{Sign}(s_u, sk_{cert}))$ in which $gsk[i]$ represents his private group signature key, s_u is his secret identifier and $cert_{s_u}$ corresponds to a CL-signature [3] over s_u done using sk_{cert}. The CL-signature will be used to enable a user to prove his possession of a signature without revealing the signature itself using a zero-knowledge proof.

When a user wants to obtain a proof of his current location, he needs to run the LocationProofGathering procedure with each of his neighboring users (*i.e.*, witnesses) to collect location proof shares. For security reasons, each witness is allowed to generate at most one location proof share in response to a prover's location proof request. After collecting at least k shares, the prover can produce a location proof certifying his position (*cf.* Section 3.2). Later, this location proof can then be used by its true owner to prove his location to a verifier by running the LocationProofVerification procedure.

The two sections detail the LocationProofGathering and LocationProofVerification procedures.

3.2 Location Proof Gathering

The location proof gathering protocol between a prover and a witness, summarized in Fig. 1, consists of the following six rounds.

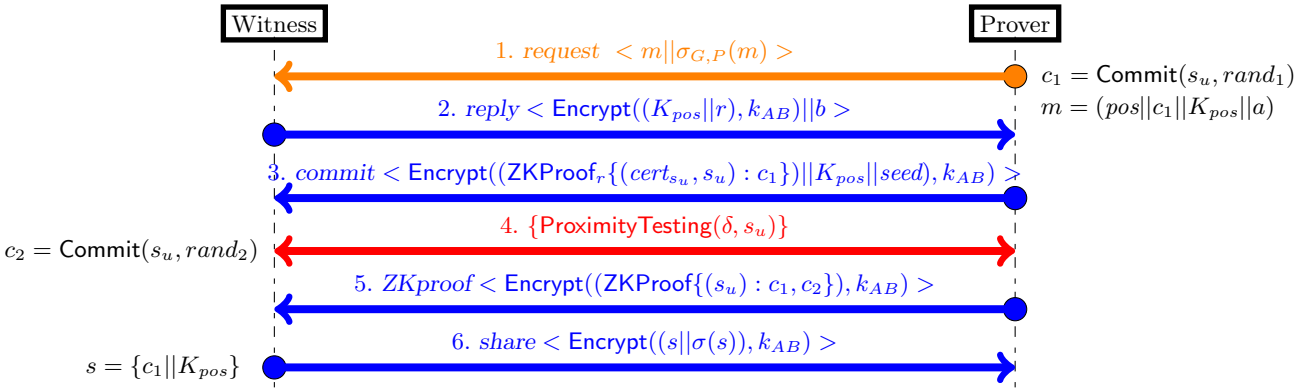

Fig. 1. Sketch of the location proof gathering protocol.

1. The prover P generates a random *seed* and computes $K_{pos} = \mathsf{Hide}(pos, seed)$. P also computes a, a random share for a Diffie-Hellman key agreement scheme. Finally, the prover P generates a random string $rand_1$ and computes $c_1 = \mathsf{Commit}(s_u, rand_1)$, which corresponds to the commitment of his secret identifier

s_u under $rand_1$. Then, P broadcasts locally a request for a location proof to neighboring users over an insecure (*i.e.*, unencrypted and unauthenticated) channel: $P \rightarrow * : \{m = (pos||c_1||K_{pos}||a)||\sigma_{G,P}(m)\}$. The request contains the following information: a message m composed of the concatenation of *pos*, the current position of the user (which also contains the current timestamp), c_1 a commitment of P on his secret string s_u, K_{pos} an encoding of the position (including the time information) of P via a hash chain, a the share of the Diffie-Hellman key agreement and finally $\sigma_{G,P}(m)$ a group signature on the message m. The variables c_1 and K_{pos} can be considered as being the identifiers of the location proof, thus these values will be common to the k shares collected by the prover.

2. Upon reception of the previous request, a witness W first verifies if the request sent by the prover P seems to be valid by checking the authenticity of the group signature $\sigma_{G,P}(m)$ and the plausibility to the location *pos* claimed in the request. If this verification succeeds, then the witness W generates a random share b of a Diffie-Hellman key agreement and combines it with a in order to generate a session key k_{AB}. This session key k_{AB} will then be used to encrypt all subsequent communications between P and W by relying on a symmetric cryptosystem, thus ensuring the confidentiality and integrity of communications (*cf.* Section 2.2). Afterwards, the witness W sends to the prover P the encryption under the session key k_{AB} of K_{pos} concatenated with a freshly generated nonce r, as well as in clear the value of b, the second share of the Diffie-Hellman key agreement: $W \rightarrow P : \{\mathsf{Encrypt}((K_{pos}||r), k_{AB})||b\}$.
3. The prover P can combine b with a in order to reconstruct the session key k_{AB} of the Diffie-Hellman key agreement. Afterwards, P decrypts the message sent by W at the previous round in order to learn r while also checking that W has been able to encrypt K_{pos} with the session key. Finally, P sends to W the following message, which corresponds to the encryption of the commitment c_1 concatenated with a zero-knowledge proof and the information needed to verify that K_{pos} is a correct encoding of *pos*:
$P \rightarrow W : \{\mathsf{Encrypt}((\mathsf{ZKProof}\{(cert_{s_u}, s_u) : c_1\})||K_{pos}||seed), k_{AB})\}$. More precisely, the zero-knowledge proof contained in the message is a Groth-Sahai proof that the commitment c_1 contains a secret value s_u for which P possesses a valid signature (*i.e.*, certificate) over the secret value s_u.

$$\begin{aligned}\mathsf{ZKProof}\{(cert_{s_u}, s_u) : c_1\} \leftarrow \mathsf{ZKProof}\big\{&(cert_{s_u}, s_u) : \\ &\mathsf{VerifyCommit}(c_1, s_u, rand_1) = 1 \ \wedge \\ &\mathcal{CLS}.\mathsf{VerifySign}(s_u, cert_{s_u}, pk_{cert}) = 1\big\}.\end{aligned}$$

   In order to ensure the freshness of the proof, the zero-knowledge proof will also depend on r, the nonce generated by the witness at the previous step, which is made possible by the zero-knowledge proofs based on CL-signatures.
4. Then, P starts the proximity testing tool with W, denoted by $P \leftrightarrow W : \{\mathsf{ProximityTesting}(\delta, s_u)\}$. At the end of this procedure, beside being convinced that the prover is close to him, the witness also receives as output a new commitment $c_2 = \mathsf{Commit}(s_u, rand_2)$, a commitment on the identity s_u of the prover under a new random string $rand_2$ only known by the prover.The distance threshold δ is a global parameter of the system.
5. If the $\mathsf{ProximityTesting}(\delta, s_u)$ procedure outputs accept, W is now convinced that P is located at the claimed position *pos* and receives as output c_2 at the end of the protocol. Otherwise, if the proximity testing procedure outputs reject, then the location proof gathering protocol is aborted. If the protocol proceeds, P sends a zero-knowledge proof that the value s_u contained in the committed value contained in the commitments c_1 and c_2 is the same by calling the $\mathsf{EqualityCommitment}$ functionality on c_1, c_2, $rand_1$ and $rand_2$.
6. Afterwards, W verifies the correctness of both zero-knowledge proofs (*i.e.*, the one that proved that c_1 is a commitment on a secret value for which P possesses a certificate and the one proving the equality of the committed value in c_1 and c_2) and that K_{pos} is a correct encoding value of *pos* using *seed*. If all these steps succeed, W creates a share $s = \{c_1||K_{pos}\}$ as well as $\sigma_j(s) = \mathcal{UGS}.\mathsf{SignGroup}(s, gsk[j])$, which corresponds to a unique group signature of witness (*i.e.*, user) j on the share s using the private signature key $gsk[j]$ of W. Finally, the witness sends to the prover: $W \rightarrow P : \{\mathsf{Encrypt}((s||\sigma(s)), k_{AB})\}$.

3.3 Location Proof Verification

After collecting at least k different shares coming from unique witnesses, the prover P can aggregate them into a location proof that we denote thereafter as $certificate = \{s, \sigma_1(s), \ldots, \sigma_k(s)\}$. Before showing a location proof to a verifier, the prover and the verifier have to agree on the granularity of the location information that has to be revealed in order to benefit from the service. For instance, this could be done by running a service level agreement protocol between the prover and the verifier. The location proof verification protocol consists of the following consecutive steps. If one of these steps fails, the protocol aborts.

1. The prover P chooses the precision p for which he accepts to reveal his position and computes the pair $(pos|_p, K^p_{pos}) = \mathsf{Reveal}(pos, p, seed)$ in which $pos|_p$ denotes the position pos partially blinded up to granularity p and K^p_{pos} represents the value of the hash chain corresponding to the $d - p$ right digits of pos. Then, P sends to the verifier V the location proof $certificate$ as well as $pos|_p$ and K^p_{pos}: $P \rightarrow V : \{certificate||pos|_p||K^p_{pos}\}$.
2. The verifier V checks for each location share $\sigma_i(s)$ in $certificate$ whether $\mathcal{UGS}.\mathsf{VerifySignGroup}(s, \sigma_i(s), gpk)$ returns accept. More precisely, the function $\mathcal{UGS}.\mathsf{VerifySignGroup}(s, \sigma_i(s), gpk)$ verifies the validity of the group signature of each of the k share $\sigma_i(s)$:
$\bigwedge_{i=1}^{k}(\mathcal{UGS}.\mathsf{VerifySignGroup}(s, \sigma_i(s), gpk)) = \mathsf{true}$.
3. The verifier V validates the uniqueness of the shares in $certificate$ by verifying that *all these shares* have been generated by different witnesses:
$\bigwedge_{i=1}^{k}\bigwedge_{j=i+1}^{k}(\mathcal{UGS}.\mathsf{Detect}(s, \sigma_i(s), \sigma_j(s))) = \mathsf{false}$.
4. The verifier V anonymously authenticates the prover P as the legal owner of $certificate$ by running a zero-knowledge proof protocol with him. At the end of this step, V should be convinced that P knows the secret s_u used to generate the commitment c contained in s.
5. Finally, using the information $(pos|_p, K^p_{pos})$, the verifier V computes $\mathsf{Check}(K_{pos}, K^p_{pos}, pos|_p)$ in order to verify the validity of the location claimed by the prover.

3.4 Proof-of-concept Implementation

In this section, we briefly report on the current proof-of-concept implementation of PROPS. Our main objective is to demonstrate that most of the architecture of PROPS can be implemented with currently available technology. Our implementation relies on Idemix[4] version 2.3.4, a Java library containing advanced cryptographic primitives such as CL-signatures, commitment schemes and zero-knowledge proofs. Moreover, we have implemented unique group signatures by relying on the concept of domain pseudonym offered by Idemix. In a nutshell, a domain pseudonym can be used to link all the group signatures that are performed by a user within the same domain. Within the context of PROPS, we set the domain to be equal to the value r included in the location proof, thus allowing a verifier to check if several signatures have been issued by the same user on a specific domain (*i.e.*, location proof).

Our proof-of-concept implementation of PROPS consists of Java classes modeling the different participants of the location proof system, namely the CA, a prover, a witness, a verifier and the AL. Currently, we have implemented the registration phase, the location proof gathering protocol and the location proof verification. The measurements that we report for each phase have been computed by averaging over 100 independent trials run on an Intel i5-2435M dual-core processor at 2.4 Ghz with 4 GB of 1333 Mhz DDR3 SDRAM running OSX 10.8.3. The registration phase requires on average 0.27 ± 0.04 s from the prover while the interaction between a prover and a witness have an average running time of 0.75 ± 0.05 s in order to produce a location share of 3444 bytes. The verification of a location share takes around 0.46 ± 0.03 s to be computed by a verifier. We have not been able to test the proximity testing protocol as current implementations of distance-bounding (*e.g.*, [15, 14]) consist only of hardware proof-of-concepts, and are not mature enough to be used directly on standard devices.

[4] `http://www.zurich.ibm.com/security/idemix/`

4 Conclusion

In this paper, we introduced PROPS, a novel privacy-preserving location proof system based on a collaborative and distributed architecture. To the best of our knowledge, this protocol is the first to ensure the privacy all participating users, from the location proof gathering up to the verification phase. Moreover, due to the use of unique group signatures, PROPS can detect, though not identify, a malicious user that tries to generate several location shares under the same identity. Finally, PROPS is secure against classical localization attacks such as the distance, mafia and terrorist frauds. While our proof-of-concept implementation is based on off-the-shelf software components, we are currently developing a prototype running on devices carried by users (*e.g.*, smartphones) in order to validate the approach in real-life. In the long term, our aim is to use the proposed architecture as a building block to provide secure, privacy-aware and fully distributed location-based services.

References

[1] Claudio A. Ardagna, Marco Cremonini, Ernesto Damiani, Sabrina De Capitani di Vimercati, and Pierangela Samarati. Supporting location-based conditions in access control policies. In *ACM ASIACCS*, 2006.

[2] Stefan Brands and David Chaum. Distance-bounding protocols. In *Proceedings of EUROCRYPT '93*, 1993.

[3] Jan Camenisch and Anna Lysyanskaya. A signature scheme with efficient protocols. In *Security in Communication Networks*, Lecture Notes in Computer Science, 2003.

[4] David Chaum and Eugene van Heyst. Group signatures. In *EUROCRYPT*. LNCS 547, 1991.

[5] Cas Cremers, Kasper Bonne Rasmussen, and Srdjan Čapkun. Distance hijacking attacks on distance bounding protocols. Technical report, Cryptology ePrint Archive: 2011/129, 2011.

[6] Benjamin Davis, Hao Chen, and Matthew Franklin. Privacy-preserving alibi systems. In *7th ACM Symposium on Information, Computer and Communications Security, ASIACCS*, Seoul, South Korea, 2012.

[7] Yvo Desmedt. Major security problems with the 'unforgeable' (feige)-atshamir proofs of identity and how to overcome them. In *Proceedings of SecuriCom '88*, 1988.

[8] Matthew Franklin and Haibin Zhang. Unique group signatures. In Sara Foresti, Moti Yung, and Fabio Martinelli, editors, *ESORICS*, pages 643–660. LNCS 7459, 2012.

[9] Michelle Graham and David Gray. Protecting privacy and securing the gathering of location proofs - the secure location verification proof gathering protocol. *LNICST*, 17, 2009.

[10] G.P. Hancke and M.G. Kuhn. An rfid distance bounding protocol. In *Procs of SecureComm 2005*, pages 67 – 73, sept. 2005.

[11] Chong Hee Kim, Gildas Avoine, François Koeune, François-Xavier Standaert, and Olivier Pereira. Information security and cryptology. In *Information Security and Cryptology*, chapter The Swiss-Knife RFID Distance Bounding Protocol, pages 98–115. Springer-Verlag, Berlin, Heidelberg, 2009.

[12] Gabriele Lenzini, Sjouke Mauw, and Jun Pang. Selective location blinding using hash chains. In *Security Protocols Workshop*, LNCS 7114, 2011.

[13] Wanying Luo and Urs Hengartner. Proving your location without giving up your privacy. In *ACM HotMobile*, 2010.

[14] Aanjhan Ranganathan, NilsOle Tippenhauer, Boris Skoric, Dave Singelee, and Srdjan Capkun. Design and implementation of a terrorist fraud resilient distance bounding system. In *Computer Security - ESORICS 2012*, Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2012.

[15] Kasper Bonne Rasmussen and Srdjan Čapkun. Realization of rf distance bounding. In *Proceedings of the USENIX Security Symposium*, 2010.

[16] S. Saroiu and A. Wolman. Enabling new mobile applications with location proofs. In *ACM HotMobile*, 2009.

[17] Manoop Talasila, Reza Curtmola, and Cristian Borcea. Link: Location verification through immediate neighbors knowledge. *Springer, LNICST 73*, 2012.

[18] Rolando Trujillo-Rasua, Benjamin Martin, and Gildas Avoine. The Poulidor distance-bounding protocol. In Siddika Ors Yalcin, editor, *Radio Frequency Identification: Security and Privacy Issues*, volume 6370 of *Lecture Notes in Computer Science*, pages 239–257. Springer Berlin / Heidelberg, 2010.

[19] Brent Waters and Edward Felten. Secure, private proofs of location. Technical report, Department of Computer Science, Princeton University, Tech. Rep. TR-667-03, 2003.

[20] Zhichao Zhu and Guohong Cao. Applaus: A privacy-preserving location proof updating system for location-based services. In *INFOCOM*, pages 1889–1897, 2011.