Designing and verifying core protocols for location privacy

David von Oheimb and Jorge Cuellar

Siemens Corporate Technology, Munich, Germany

Abstract. Geographic privacy services provide location information on roaming targets to location recipients via location servers, in a way that protects the privacy of the individuals involved. In this paper we propose and discuss new protocols representing the core of Geopriv, with particular focus on the security requirements stated in the IETF's RFC 3693. Using the AVISPA tool, we check that these requirements, namely anonymity against the location server, as well as confidentiality, integrity, and authenticity of the location information, are actually met. In the design phase of such protocols, numerous variants are to be considered and evaluated. Here the use of model checkers turns out to be very helpful in exploring the security implications quickly and precisely.

Keywords: Geopriv, location information, privacy, model checking, exploration

1 Introduction

With the widespread use of mobile devices like mobile phones and GPS receivers, *Location Based Services* offer convenient and commercially attractive ways to solve issues like "Please direct me to the nearest shopping mall.", "Which time zone is my boss currently traveling in?", or "Has the kid reached home safely?" However, location information needs to be gathered and transfered securely, protecting the privacy of the individuals involved.

This paper explores some of the basic protocols that can be used to transfer location data, respecting the authorization, integrity and privacy requirements. See the RFCs 3693 [CMM+04] and 3694 [DMMP04] for more background on requirements and threat analysis.
The IETF working group Geopriv [GWG06] has focused itself on

- the format of the *Location Information* to be sent ("*Location Object*"),
- the format of the *Privacy Rules* describing policies to be applied,
- particular cases of so-called "using" protocols, that is, protocols that carry Location Information about a mobile user (the "*Target*") from a *Location Server* to a *Location Recipient*.

Nevertheless, to understand the requirements and goals of Geopriv, one needs to consider also protocols that are out of scope at the IETF. The pieces missing are protocols used to

– agree on pseudonyms and/or passwords for the Target and the Location Recipient, which are to be used by the policies,
– request, using the credentials just mentioned, the Location Information from a Location Server,
– transfer credentials to a Location Server,
– transfer the Privacy Rules to a Location Server,
– transfer the Location Information to an initial Location Server.

The main goals of this paper are to model a reference implementation of the overall system and to evaluate its security in a systematic and holistic way. This can be done only by including the just mentioned components.

The work described here has been done in an industrial context, at the intersection of standardization bodies and commercial implementors. The main principles of our design are:

– The most important requirements are to guarantee the correctness (integrity) and confidentiality of the Location Information. This requires authenticating the main entities of the protocol and securing the exchanged messages.
– A central role is played by user-controlled policies, which describe the permissions (or consent) given by the Target. The policies specify not only the time and place when Location Information may be released to whom, but also which component (or derived measure) of the information is to be released and in which granularity or accuracy.
– Whenever possible, the Location Information should not be linked to the identity of the user. Rather, the user is able to specify which local identifier, pseudonym, private identifier, or token is to be used instead.
  Although complete anonymity may not be appropriate because of legal constraints or because some location services do in fact need the explicit identification of the user, we argue that in most cases the location services may only need some type of authorization information and/or perhaps an anonymous identifier of the users, that may change as often as needed.
– To ease comprehendability and implementation, our reference model should be as concise as possible. It shall clarify any issues left open by the RFCs. Particular solutions and alternatives shall be motivated and explained.

Note that the main challenge and novelty aspect here is the anonymity goal (identity protection), which inherently opposes the authentication requirements and makes the use of existing protocols with standard certificates etc. impossible.

Given the recent advances in protocol analysis by the project AVISPA [AH-03] and the availability of their tools, we have done the modeling in the High Level Protocol Specification Language HLPSL [CCC+04] and have conducted our analysis with the tools contained in the AVISPA package [AT-05].

For reasons of space and to avoid confusion by two different syntactic levels, we have decided to use for our presentation a pseudo-mathematical "Alice-Bob"-style notation that seems to be widely accepted or at least easily understandable without much further explanation. For more information on specifications in HLPSL and on how to check them with the AVISPA Tool, please refer to the HLPSL Tutorial [AVI05a] and the AVISPA User Manual [AVI05b].

2 General design

As already mentioned, typical Geopriv protocols involve a *Target (T)*, whose *Location Information (LI)* is to be conveyed to a *Location Recipient (LR)* by a *Location Server (LS)*. A *Privacy Rule (PR)* defines the policy: who is allowed to learn whose location, under which circumstances, with which *Granularity (GR)*. The Granularity may be, for instance, the complete street address, the GPS coordinates up to a given precision, or just the time zone.

Although it is assumed that Location Servers adhere to the protocol and cannot be compromised by an attacker, at least some types of Location Servers are considered untrusted in the sense that they should not learn the real identities of the Location Recipient and of the Target. So when communicating with the Location Server, these parties do not use their real names but just pseudonyms and/or passwords. Each party can authenticate itself to the Location Server with the help of such a password or a signature related to the respective pseudonym. The pseudonyms and passwords, if any, form part of the Privacy Rules. As opposed to passwords, pseudonyms in general need not be kept secret.

It is assumed that Location Recipients do not abuse the Location Information they obtain, e.g. by publishing it. Yet in our analysis, we will consider not only standard sessions with honest Targets and Location Recipients, but also problematic sessions where the intruder is allowed to assume the role of either of these two. This means that his initial knowledge is augmented with the private keys that a legitimate player of the role usually has, enabling him to play that role properly. Such sessions themselves do not make much sense because the intruder is made a legitimate source or receiver of the Location Information. It is interesting, however, to see if such degenerate sessions can have a bad effect on standard sessions.

Usually, the Target subsumes the role of *Rule Maker*. In our presentation we further assume that the Target is also the *Location Generator*.[1] We also assume that the Location Server is the *Rule Holder*.

The exchange between the Geopriv entities can be divided into the following phases (or, sub-protocols).

Agreement. The Location Recipient and Target, who usually know and trust each other, exchange credentials like pseudonyms and passwords. Typically, this includes mutual authentication, as well as authorization of the Location Recipient.

Policy. The Privacy Rule is transferred from the Target to the Location Server, potentially via a separate Rule Holder. The main issue here is authorization of the Target and authentication and integrity of the Privacy Rule.

Location. The Location Server learns the current location of the Target, potentially via a separate Location Generator. The main issue here is secrecy, authentication, and integrity of the Location Information.

[1] A simpler — but in many cases not possible — alternative is that the Location Server senses the presence of the Target directly (still without knowing T's identity).

Information. The Location Recipient requests the location of the Target and receives an answer from the Location Server. The main issues here are authentication and authorization of the Location Recipient, as well as the secrecy, authentication, and integrity of the Location Information, this time from the perspective of the Location Recipient.

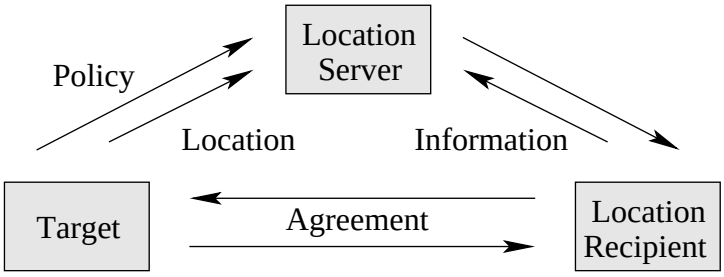

Fig. 1. Geopriv Structure

The general Geopriv structure is depicted in Fig. 1. The order of the first three of the four sub-protocols may vary, whereas — without loss of generality — we assume the order just given.

For each of the four phases, there are various ways to implement them. In this paper we describe in detail two Geopriv protocol variants that exhibit four differences distributed over all these phases. These differences are orthogonal to each other, such that in effect we implicitly cover $2^4 = 16$ variants.

3 Variant with two self-signatures

In our first variant, both the Location Recipient and the Target use a pseudonym and a self-signature. A *self-signature* is a digital signature where the sender's public key (or a hash of it) is included in the part signed with the corresponding private key. No certificate is used to link the sender's identity with the public key, but still the receiver can check that the sender holds the corresponding private key and therefore should be the owner of the public key. Self-signatures are used here to protect the anonymity of the Location Recipient and the Target when communicating with the Location Server. To this end, each of the two parties $X \in \{LR, T\}$ creates a public/private key pair (P_X, P_X^{-1})[2] and uses the hash of the public key $h(P_X)$ as its pseudonym Ψ_X. We use the letter 'P' rather than the usual 'K' for the these keys to emphasize that they are related to a

[2] We use the handy notion X^{-1} to refer to the private key related to the public key X, which does not imply that the former can be derived from the latter, but alludes to the fact that encryption using X can be inverted by decryption using X^{-1}, and signatures (by decryption) using X^{-1} can be checked (by encryption) using X.

pseudonym. The purpose of hashing is just to compress the relatively large public key values to a short and fixed-length field. A message including a self-signature typically has the format $\{M.h(P_X)\}_{P_X^{-1}}.P_X$, i.e. consists of some payload M and $\Psi_X = h(P_X)$ jointly signed[3] with the private key P_X^{-1} and concatenated with P_X. In this way, the receiver (which is the Location Server here) can use the public key provided to check the signature and see if the hash of the public key matches the corresponding value in the signed part. If so, he can be sure that the sender holds the private key related to both the public key and the pseudonym, without learning the identity of the sender. Of course, authentication can only be ensured in combination with other means. If self-signatures are used in isolation, any party can produce them and then mount, e.g., denial-of-service attacks.

Furthermore, this variant uses public key cryptography for securing the messages of the agreement phase and signing the location information message sent by the Location Server.

3.1 Protocol specification

The overall protocol, in the usual Alice-Bob notation, reads as follows.

Agreement. $T \leftarrow \{LR\}_{K_T}.\{\{T.N_1\}_{K_T}.\Psi_{LR}\}_{K_{LR}^{-1}} - LR$

$T \dashrightarrow \{\{N_1\}_{K_{LR}}.\Psi_T\}_{K_T^{-1}} \rightarrow LR$

This phase is secured using ordinary public-key encryption and signatures of T and LR, with the public keys K_T and K_{LR} as well as their private counterparts K_T^{-1} and K_{LR}^{-1}. In order to register with T and obtain T's pseudonym, LR sends to T its own identity LR, T's identity (as a redundancy that can be checked by T), a nonce N_1 (a "random" value used to ensure freshness for the authentication of T), as well as its own pseudonym $\Psi_{LR} = h(P_{LR})$. The name LR, used by T to identify LR, is encrypted with T's public key such that LS has no chance to learn the association of Ψ_{LR} with the real identity LR. Note that we cannot move the name LR inside the signed part, because T first needs to derive LR's public key from this name before being able to decrypt the signed part.

If T is willing to share its location (up to some granularity) with LR, it answers with the nonce just received and the pseudonym $\Psi_T = h(P_T)$. Note that the pseudonyms do not need to be encrypted, but just signed.

Policy. $T \dashrightarrow \{GR.\Psi_{LR}.\Psi_T\}_{P_T^{-1}}.P_T \longrightarrow LS$

T sends to LS its policy aka Privacy Rule, comprising the granularity GR and the pseudonyms of LR and T, in a self-signed way.

[3] In practice, a message $\mathcal{M}$ is signed with a key Y^{-1} by appending $\mathcal{M}$ with the hash of $\mathcal{M}$ decrypted with Y^{-1}, which we would denote by the term $\mathcal{M}.\{h(\mathcal{M})\}_{Y^{-1}}$, but for our purposes (assuming $\mathcal{M}$ contains enough redundancy, and abstracting from other issues like computational efficiency) it suffices to use the simpler term $\{\mathcal{M}\}_{Y^{-1}}$ that avoids repeating $\mathcal{M}$.

Location. $T \;—\!\!—\; \{TS.\{LI.\Psi_T\}_{K_{LS}}\}_{P_T^{-1}}.P_T \longrightarrow LS$

T informs LS about its current location, by sending, again in a self-signed way, a timestamp TS (which can be used to avoid replay attacks) as well as the Location Information LI (with maximal accuracy) and its pseudonym Ψ_T, both encrypted with the public key K_{LS}.

Information. $LS \longleftarrow\!\!—\!\!— \{\Psi_{LR}.\Psi_T.N_2\}_{P_{LR}^{-1}}.P_{LR} - LR$

$LS \;—\!\!—\; \{\{GR(LI)\}_{P_{LR}}.N_2\}_{K_{LS}^{-1}} \longrightarrow LR$

Finally, LR requests T's current location by sending to LS, also in a self-signed way, its own pseudonym, T's pseudonym, and a nonce N_2. The nonce ensures freshness for the authentication of LS and identifies the answer expected from LS. Therefore LS does not need to echo Ψ_{LR} or Ψ_T.

LS matches the pseudonyms with its Privacy Rule table, which is used also for looking up the granularity GR specified by T. If found, LS replies with a message containing the location data and the nonce, where the location data is LI projected to the granularity GR and encrypted with P_{LR}. Since the encryption key P_{LR} and the nonce N_2 are not secret, such that anyone could construct such a message, LS has to sign the message with its private key K_{LS}^{-1} in order to authenticate itself to LR.

Our model does not support location updates by re-sending the Location message with new data. Therefore, replay protection for the authenticity of $GR(LI)$ is simple. If location updates are possible, one must do more to prevent replay attacks, namely use a timestamp. Since HLPSL does not support time, we include a pseudo-timestamp TS just as a reminder.

Furthermore, we do not explicitly model authorization (as opposed to authentication) of the Location Recipient in the Agreement sub-protocol, or authorization of the Target in the Policy sub-protocol. We simply take the worst-case assumption that they are authorized unconditionally.

3.2 Requirements specification

The protocol has been designed to enjoy the following security properties, which (apart from the last two ones) are immediate formalizations of the requirements stated in RFC3693 [CMM+04].

- secrecy of LI and of the filtered version $GR(LI)$
- LR strongly authenticates LS on N_2
- LS weakly authenticates LR on P_{LR}
- LS weakly authenticates T on GR
- LR strongly authenticates T on $GR(LI)$
- LR strongly authenticates T on N_1
- T weakly authenticates LR on Ψ_{LR}

The phrase "X weakly authenticates Y on Z" means that X can be sure that its peer is indeed Y and the two parties use the same value Z. This the same

as Lowe's notion of *non-injective agreement* [Low97]. Strong authentication additionally entails replay protection, i.e. freshness of the agreement (or session) between the two, and directly corresponds to Lowe's *injective agreement*. The term "LR strongly authenticates T" appears twice — the first instance referring to the authenticity of $GR(LI)$ in the Information phase, the second one referring to the authentication of T in the Agreement phase.

The formalization of both the protocol and its security requirements in the specification language HLPSL can be found in the appendix as well as on-line at `http://www.avispa-project.org/library/self-signatures.html`.

3.3 Design process and analysis results

In designing the protocol, we have been careful to fulfill the anonymity requirements, which unfortunately cannot be checked by the tools at hand. Moreover, we aimed at minimizing the number of message fields and the use of cryptographic operations, and we have tried to get as far as possible wrt. replay protection without adding extra message exchanges. For evaluating the large number of intermediate protocol versions wrt. structural correctness, confidentiality and authentication, the AVISPA Tool [AT-05] has proved very useful – we have been able to check these versions in an exploratory way, very quickly and easily.

For instance, in this way we have found that in the Location message, if we encrypt LI only (resulting in the message $\{TS.\{LI\}_{K_{LS}}.\Psi_T\}_{P_T^{-1}}.P_T$), we get a man-in-the-middle attack against both authentication of LR and secrecy of $GR(LI)$, the trace[4]of which is the following:

$$
\begin{array}{rcl}
 & & i \leftarrow \{LR\}_{K_T}.\{\{T.N_1\}_{K_T}.\Psi_{LR}\}_{K_{LR}^{-1}} \text{ - } LR \\
T \leftarrow \{LR\}_{K_T}.\{\{T.N_1\}_{K_T}.\Psi_{LR}\}_{K_{LR}^{-1}} \text{ - } i & & \\
T \quad \text{------} \{\{N_1\}_{K_{LR}}.\Psi_T\}_{K_T^{-1}} \text{------}\!\!\rightarrow \quad i & & \\
T \quad \text{------} \{GR.\Psi_{LR}.\Psi_T\}_{P_T^{-1}}.P_T \longrightarrow \quad i & & \\
T \quad \text{– } \{TS.\{LI\}_{K_{LS}}.\Psi_T\}_{P_T^{-1}}.P_T \longrightarrow \quad i & & \\
 & i(T) \text{ ------ } \{GR.h(P_j).h(P_i)\}_{P_i^{-1}}.P_i & \rightarrow \; LS \\
 & i(T) \text{ - } \{TS.\{LI\}_{K_{LS}}.h(P_i)\}_{P_i^{-1}}.P_i & \rightarrow \; LS \\
 & i(LR) \text{ ------ } \{h(P_j).h(P_i).N_2\}_{P_j^{-1}}.P_j & \rightarrow \; LS \\
 & i(LR) \leftarrow\!\!\text{------ } \{\{GR(LI)\}_{P_j}.N_2\}_{K_{LS}^{-1}} \text{ ------} & LS
\end{array}
$$

The essence of this attack is that after the Target has sent its messages, the intruder can re-use the encrypted value $\{LI\}_{K_{LS}}$ intercepted from T and pose

[4] In our setting, a *trace* is a kind of message sequence chart describing which messages are sent between which parties in which order. Due to the standard Dolev-Yao intruder model [DY83] which we employ, messages between honest parties are always sent via the intruder, denoted by i, who may decide to suppress, modify, or forward them to any party. Where the intruder poses as role R, we write $i(R)$. An *attack trace* is a trace leading to an *attack state*, i.e. a state of the parties involved (including the intruder) where one of the desired security properties is violated.

towards LS as both a Target and a Location Recipient, hijacking the original session. In this way, he can trick the Location Server to send to him, rather than to the legitimate Location Receiver, a copy of $GR(LI)$ encrypted with a public key P_j. Since P_j can be chosen freely by the intruder, he can decrypt the location data. To prevent this attack, we simply move Ψ_T inside the encryption, arriving at the message $\{TS.\{LI.\Psi_T\}_{K_{LS}}\}_{P_T^{-1}}.P_T$. Now any attempt to replay $\{LI.\Psi_T\}_{K_{LS}}$ in a different context does not fit because Ψ_T does not match with the hash of any public key for which the intruder has the corresponding private key needed for producing the self-signature for the Location message.

Actually, due to anonymity, the Location Server cannot authenticate the Target and the Location Receiver individually but only check the consistency of the self-signed messages received from the two. This also implies that the intruder can always pose as both these parties simultaneously, provoking the following degenerate exchange:

$$
\begin{array}{lll}
i(T) & \text{---------} \{GR.h(P_j).h(P_i)\}_{P_i^{-1}}.P_i \rightarrow & LS \\
i(T) & \text{------} \{TS.\{LI.h(P_i)\}_{K_{LS}}\}_{P_i^{-1}}.P_i \rightarrow & LS \\
i(LR) & \text{---------} \{h(P_j).h(P_i).N_2\}_{P_j^{-1}}.P_j \rightarrow & LS \\
i(LR) & \leftarrow\text{------} \{\{GR(LI)\}_{P_j}.N_2\}_{K_{LS}^{-1}} \text{------} & LS
\end{array}
$$

Yet this is a useless attack (apart from wasting the Location Server's resources) because the intruder does not obtain anything interesting — he receives just the location data that he has provided himself. As soon as a faithful Location Recipient or Target is involved, correct authentication is guaranteed because the Location Server can check the self-signed messages of these two parties for consistency of the pseudonyms involved. This consistency, in turn, is guaranteed by the mutual authentication of the Location Recipient and the Target. Hence if one of these is authentic, then the other is, too.

4 Variant with password and certificate

Our second variant uses a pseudonym only for the Target and uses password for the Location Recipient. The Target authenticates itself to the Location Server not with a self-signature, but more stringently, with a certificate issued by a trusted third party. The Agreement phase is secured using a shared secret key (or any other form of a secure channel) rather than with public-key cryptography, such that both the Target and the Location Recipient depend on a public-key infrastructure (PKI) only for sending messages to the Location Server. In the Information phase, the Location Recipient provides a (secret) temporary key to be used for encrypting the location data. All this amounts to four major differences to our first variant.

4.1 Protocol specification

The protocol is specified as follows.

Agreement. $T \longleftarrow LR.\{T.N_1\}_{K_{TLR}} \longleftarrow LR$
$T \longrightarrow \{PW_T.\Psi_T.N_1\}_{K_{TLR}} \longrightarrow LR$

This phase is secured with a secret key K_{TLR} shared between T and LR. In order to register with T and obtain the password, LR sends to T its own identity LR, T's identity and a nonce N_1. The name LR is sent in the clear in this case, because T needs a way to tell who it is talking to and select the right key K_{TLR}. This is fine as long as LS cannot intercept and link the first message with the anonymous request it receives in the Information phase.

If T is willing to share its location with LR, it answers with the nonce just received, its pseudonym Ψ_T, and the password PW_T. Actually, signing the pseudonym would be sufficient, rather than encrypting it.

Policy. $T - \{GR.\{PW_T\}_{K_{LS}}.\Psi_T\}_{K_T^{-1}}.\{\Psi_T.K_T\}_{K_{CA}} \rightarrow LS$

T sends to LS its Privacy Rule, comprising the granularity GR, the password (in encrypted form), and its pseudonym. All this is signed with T's private key. The receiver LS does not know T's identity, yet in order to check the signature, it is sufficient that T encloses a certificate — signed by a trusted third party called CA — stating the relation between the pseudonym and T's public key. Of course, this means an extra overhead, which can be reduced in cases where LS is allowed to know the identity of the Target.

Location. $T \longrightarrow \{TS.\{LI\}_{K_{LS}}.\Psi_T\}_{K_T^{-1}} \rightarrow LS$

T conveys to LS its current location, by sending, again signed with K_T^{-1}, its pseudonym Ψ_T and a timestamp TS along with the Location Information LI. Here only LI needs to be encrypted with the public key K_{LS} because LS can authenticate T independently of LR, using the certificate received in the Policy phase. Therefore the two attacks explained in section 3.3 are a priori not possible.

Information. $LS \longleftarrow \{K_{LR}.PW_T.N_2\}_{K_{LS}} \longleftarrow LR$
$LS \longrightarrow \{GR(LI).N_2\}_{K_{LR}} \longrightarrow LR$

Finally, LR requests T's current location by sending to LS a temporary key K_{LR}, the password related to T (which actually renders sending along T's pseudonym unnecessary) and a nonce N_2. All this is encrypted with LS's public key.

LS matches the password (and T's pseudonym, if sent nevertheless) with its Privacy Rule table. If found, LS replies with a message containing the location data and the nonce. Since the encryption key K_{LR} and the nonce N_2 have been kept secret, LR just needs to encrypt the location data and the nonce with K_{LR} in order to protect the location data and to authenticate itself to LR.

The two notes wrt. location updates and authorization that we have given for the other variant, apply also for this variant: location updates are not modeled, which simplifies replay protection, and concerning authorization we have taken the most pessimistic and simple assumption that any access is granted.

4.2 Security Properties

This protocol variant has security properties very similar to the previous one:

- secrecy of LI and of the filtered version $GR(LI)$
- secrecy of PW_T and K_{LR}
- LR strongly authenticates LS on N_2
- LS weakly authenticates T on GR
- LR strongly authenticates T on $GR(LI)$
- LR strongly authenticates T on N_1
- T weakly authenticates LR on LR

Note that there are additional secrecy goals, namely for PW_T and K_{LR}.

We do not require that LS authenticates LR (e.g. on PW_T) in the usual sense because the LS can only check that the agent requesting the location information knows the correct password PW_T. That is, we should require authentication only modulo the group of agents allowed to know PW_T. Yet if we consider the (degenerate) session where the intruder legitimately plays the role of LR, he is allowed to learn PW_T in this session and obtain the location data from LS. On the other hand, the model checkers would report an authentication failure because the intruder can use PW_T to pose as LR in a different (standard) session where a honest agent has the role of LR and wants to talk to LS:

$$
\begin{array}{llll}
T & \longleftarrow i.\{T.N_1\}_{K_{Ti}} \longrightarrow & i(i) & \\
T & \text{---} \{PW_T.\Psi_T.N_1\}_{K_{Ti}} \longrightarrow & i(i) & \\
T & - \{GR.\{PW_T\}_{K_{LS}}.\Psi_T\}_{K_T^{-1}}. & & \\
 & \qquad \{\Psi_T.K_T\}_{K_{CA}^{-1}} \rightarrow & i & \\
 & & i & - \{GR.\{PW_T\}_{K_{LS}}.\Psi_T\}_{K_T^{-1}}. \\
 & & & \qquad \{\Psi_T.K_T\}_{K_{CA}^{-1}} \rightarrow LS \\
T & \text{---} \{TS.\{LI\}_{K_{LS}}.\Psi_T\}_{K_T^{-1}} \rightarrow & i & \\
 & & i & \text{---} \{TS.\{LI\}_{K_{LS}}.\Psi_T\}_{K_T^{-1}} \rightarrow LS \\
 & & i(LR) & \text{---} \{K_i.PW_T.\Psi_T.N_2\}_{K_{LS}} \longrightarrow LS \\
 & & i(LR) & \longleftarrow \{GR(LI).N_2\}_{K_i} \text{---} LS
\end{array}
$$

In this "attack", the intruder forwards the messages from T to LS (without modification, as intended by T), but then he poses as the Location Recipient of a different[5] session with the same LS. Although this is harmless in the given scenario (where the intruder can legitimately learn $GR(LI)$ anyway), the issue cannot be properly expressed in HLPSL and therefore confuses the analysis tools.

To avoid getting reported this spurious attack, we do not check that LS authenticates LR on PW_T but resort to checking the secrecy of the password and the location data, which is handled by the tools in an adequate way.

No other (spurious or actual) attacks on this protocol have been found.

---

[5] We do not reproduce the slightly obscure session indicators in the attack trace given by the tools. The fact that two different sessions are involved in the above trace can be inferred from the difference between the term $i(i)$ where the intruder represents himself and $i(LR)$ where he poses a LR.

The formalization of both the protocol and its security requirements in HLPSL are available on-line at http://www.avispa-project.org/library/password.html.

5 Conclusion

We have proposed protocols that form the core of Geopriv services, meeting the challenge of anonymity despite authentication. This is the main novelty of our protocol design. We have made explicit the design considerations that we took, such that the protocols can easily be evaluated and re-used by others. We have emphasized that many variations of the protocols are possible and have exemplified two of them. Others would be interesting, too, for instance combining the strong certificate-based authentication of the Target given in our second variant with the self-signed pseudonym approach for the Location Recipient given in our first variant (which is more strict than the password-based approach). Implementors are free to choose among all those variations according to their preferences and side-conditions imposed by the application context.

During the protocol design, it proved very helpful to formalize the various versions of the protocol, as well as their intended properties, in a designated protocol specification language and to check with automatic tools whether the given requirements are fulfilled. Since this approach greatly reduces both time and effort, we believe that it should — and soon will — become the standard approach for crypto protocol design, be it academic or industrial.

Apart from the issues discussed, the model checkers do not find attacks. Since we have carefully reviewed our formalizations to validate that they faithfully describe the protocol design, and since the tools used are quite mature, we can be confident that in the proposed Geopriv core protocols there are no design flaws that can lead to attacks on confidentiality and authentication. Of course, vulnerabilities at the cryptographic or implementation level cannot be excluded with this approach, and the anonymity aspect has been checked only informally.

6 Outlook

As mentioned, for each of the four Geopriv sub-protocols, there are various orthogonal ways to implement them. Therefore, it would be nice to reflect the inherent modularity also during formal analysis, allowing to verify each of the variants of the different phases separately and then perform some compositional reasoning to arrive at the overall security properties, combining the common properties of the variants for each of the phases involved.

Moreover, there are generalizations where several Geopriv instances are combined on demand at runtime, to build a chain of Location Servers with the policies and the Location Information flowing along a chain of trust among them. For instance, a Local Location Server may immediately sense the location, pass it on via an intermediate Remote Location Server to a Home Location Server, which is the only one the Target trusts. In this way, three Geopriv instances form a

single higher-level instance. For analyzing such a scenario, a compositional reasoning technique is not only desirable, but actually indispensable because of combinatorial explosion.

Further complications can arise from a different sort of privacy: when policies depend on the Location Server and, for a given untrusted Location Server L, the part of their contents that is not relevant to L should not be visible for L.

Orthogonally, there is the issue of dynamic policies in the sense of Privacy Rules evolving over time, granting additional access and/or revoking access to location data of various targets to various receivers. This may involve additional pitfalls that would better be checked with the meticulousness of formal methods.

In a project related to AVISPA, we plan to do further research that will tackle all the issues mentioned above.

Acknowledgments. Initial versions of the protocol models described here were done by Lan Liu in her master's thesis [Liu05]. We thank Hariharan Rajasekaran and some anonymous referees for their comments on earlier versions of this paper.

References

AH-03. The AVISPA project homepage. http://www.avispa-project.org/, 2003.

AT-05. The AVISPA Tool. Available at http://www.avispa-project.org/, 2005.

AVI. The AVISPA Project. http://www.avispa-project.org/.

AVI05a. HLPSL Tutorial: A Beginner's Guide to Modelling and Analysing Internet Security Protocols, 2005. Available at [AH-03].

AVI05b. AVISPA User Manual, 2005. Available at [AH-03].

CCC+04. Yannick Chevalier, Luca Compagna, Jorge Cuellar, Paul Hankes Drielsma, Jacopo Mantovani, Sebastian Mödersheim, and Laurent Vigneron. *A High Level Protocol Specification Language for Industrial Security-Sensitive Protocols*, volume 180 of *Automated Software Engineering*, pages 193–205. Austrian Computer Society, Austria, September 2004.

CMM+04. J. Cuellar, J. Morris, D. Mulligan, J. Peterson, and J.Polk. RFC 3693: geopriv requirements, 2004. http://www.faqs.org/rfcs/rfc3693.html.

DMMP04. M. Danley, D. Mulligan, J. Morris, and J. Peterson. RFC 3694: Threat Analysis of the Geopriv Protocol, 2004. http://www.faqs.org/rfcs/rfc3694.html.

DY83. D. Dolev and A. Yao. On the Security of Public-Key Protocols. *IEEE Transactions on Information Theory*, 2(29), 1983.

GWG06. IETF Working Group: Geographic location/privacy (geopriv), 2006. http://www.ietf.org/html.charters/geopriv-charter.html.

Liu05. Lan Liu. Analyzing web service protocols with the AVISPA approach. M.Sc. thesis, Universität Karlsruhe and Siemens, 2005.

Low97. Gavin Lowe. A hierarchy of authentication specifications. In *Proceedings of the 10th IEEE Computer Security Foundations Workshop (CSFW'97)*, pages 31–43. IEEE Computer Society Press, 1997.

A Formalization of the first protocol variant in HLPSL

```
role target(
        T, LS, LR       : agent,
        K_T, K_LS, K_LR : public_key,
        H               : hash_func,
        Snd_LR, Snd_LS, Rcv: channel(dy)) played_by T def=

local
        State   : nat,
        N1      : text,
        P_T     : public_key,
        Psi_LR  : hash(public_key),
        LI, TS  : text,
        GR      : hash_func

init State := 1

transition

        1. State  = 1 /\ Rcv({LR}_K_T.{{T.N1'}_K_T.Psi_LR'}_inv(K_LR))
       =|> State':= 3 /\ P_T'  := new()
                      /\ Snd_LR({{N1'}_K_LR.H(P_T')}_inv(K_T))
                      /\ witness (T, LR, lr_T_N1, N1')
                      /\ wrequest(T, LR, t_LR_Psi_LR, Psi_LR')
% could be new transition here, but not done for efficiency
                      /\ GR' := new()  % chooses some granularity (accuracy)
                      /\ Snd_LS({GR'.Psi_LR'.H(P_T')}_inv(P_T').P_T')
                      /\ witness(T, LS, ls_T_GR, GR')
% could be new transition here, but not done for efficiency
                      /\ LI' := new()
                      /\ secret(LI', li, {T, LS, LR})
                      /\ secret((GR'(LI')), filtered_LI, {T, LS, LR})
                      /\ TS' := new()
                      /\ Snd_LS({TS'.{LI'.H(P_T')}_K_LS}_inv(P_T').P_T')

                      /\ witness (T, LR, lr_T_filtered_LI, (GR'(LI')))
                      /\ witness(LS, LR, ls_LR_P_LR, LS)

end role
```

```
role locationServer(
        T, LS, LR: agent, % but LS does not actually use identity of T and LR
        Psi_Table: (hash(public_key).hash(public_key).hash_func) set,
        K_LS     : public_key,
        H        : hash_func,
        Snd, Rcv : channel(dy)) played_by LS def=

local   State    : nat,
        P_T,P_LR : public_key,
        N2       : text,
        Psi_LR   : hash(public_key),
        LI, TS   : text,
        GR       : hash_func

init   State := 5

transition

        5.  State = 5 /\ Rcv({GR'.Psi_LR'.H(P_T')}_inv(P_T').P_T')
       =|> State':= 7 /\ Psi_Table':= cons(Psi_LR'.H(P_T').GR', Psi_Table)

        7.  State = 7 /\ Rcv({TS'.{LI'.H(P_T)}_K_LS}_inv(P_T).P_T)
       =|> State':= 9
```

```
        9. State  = 9 /\ Rcv({H(P_LR').H(P_T).N2'}_inv(P_LR').P_LR')
                      /\   in(H(P_LR').H(P_T).GR', Psi_Table)
                         % uses Psi_LR and Psi_T to look up GR in the table
       =|> State':=11 /\ Snd({{(GR'(LI))}_P_LR'.N2'}_inv(K_LS))
                      /\ wrequest(LS, T , ls_T_GR, GR')  % delayed
                      /\ wrequest(LS, LR, ls_LR_P_LR, P_LR')
                      /\ witness (LS, LS, lr_LS_N2, N2') % to any LR!

end role
```

```
role locationRecipient(
        T, LS, LR       : agent,
        K_T, K_LS, K_LR : public_key,
        H               : hash_func,
        Snd, Rcv        : channel(dy)) played_by LR def=

local
        State           : nat,
        N1, N2          : text,
        Psi_T           : hash(public_key),
        P_LR            : public_key,
        Filtered_LI     : hash(text)

init State := 0

transition

        0. State  = 0 /\ Rcv(start)
       =|> State':= 2 /\ N1'   := new()
                      /\ P_LR' := new()
                      /\ Snd({LR}_K_T.{{T.N1'}_K_T.H(P_LR')}_inv(K_LR))
                      /\ witness(LR, T, t_LR_Psi_LR, H(P_LR'))

        2. State  = 2 /\ Rcv({{N1}_K_LR.Psi_T'}_inv(K_T))
       =|> State':= 8 /\ N2' := new()
                      /\ Snd({H(P_LR).Psi_T'.N2'}_inv(P_LR).P_LR)
                      /\ witness(LR, LS, ls_LR_P_LR, P_LR)
                      /\ request(LR, T , lr_T_N1, N1)
                      /\ witness(LS, T , ls_T_GR, LS)

        8. State  = 8 /\ Rcv({{Filtered_LI'}_P_LR.N2}_inv(K_LS))
       =|> State':= 10/\ request(LR, T, lr_T_filtered_LI, Filtered_LI')
                      /\ request(LS, LS, lr_LS_N2, N2)

end role
```

```
role session(T, LS, LR          : agent,
             K_T, K_LS, K_LR    : public_key,
             H                  : hash_func,
             Psi_Table          : (hash(public_key).hash(public_key).hash_func) set) def=

local STLR, STLS, RT, SLR, RLR, SLS, RLS: channel(dy)

composition

           target           (T, LS, LR,       K_T, K_LS, K_LR, H, STLR, STLS, RT)
        /\ locationServer   (T, LS, LR, Psi_Table, K_LS,       H, SLS, RLS)
        /\ locationRecipient(T, LS, LR,       K_T, K_LS, K_LR, H, SLR, RLR)

end role
```

```
role environment() def=

local   Psi_Table: (hash(public_key).hash(public_key).hash_func) set
        % shared between all instances of LS

const   li, filtered_LI,
        ls_T_GR,
        lr_T_N1,
        t_LR_Psi_LR,
        ls_LR_P_LR,
        lr_LS_N2,
        lr_T_filtered_LI        : protocol_id,
        t, ls, lr               : agent,
        k_T, k_LS, k_LR, k_i    : public_key,
        h                       : hash_func

init    Psi_Table := {}

intruder_knowledge = {t, ls, lr, k_T, k_LS, k_LR, k_i, inv(k_i), h}

composition

           session(t, ls, lr, k_T, k_LS, k_LR, h, Psi_Table)
%       /\ session(t, ls, lr, k_T, k_LS, k_LR, h, Psi_Table)
        % repeat session to check for replay attacks

        /\ session(i, ls, lr, k_i, k_LS, k_LR, h, Psi_Table)
%       It does not make much sense to let the intruder play the role of T
%       since then the intruder knows its location information anyway.

        /\ session(t, ls, i , k_T, k_LS, k_i , h, Psi_Table)
%       It does not make much sense to let the intruder play the role of LR
%       since then the intruder is allowed to know the (secret) location of T.

end role
```

```
goal

        secrecy_of li, filtered_LI

      % strong authentication and integrity of the Location Information,
      % (including replay protection):
        authentication_on lr_T_filtered_LI

      % the Location Recipient Location authenticates the Location Server:
        authentication_on lr_LS_N2

      % the Location Server (weakly) authenticates the Location Recipient:
        weak_authentication_on ls_LR_P_LR

      % weak authentication and integrity of Granularity
        weak_authentication_on ls_T_GR

      % additional authentication goals, not in RFC3693:
             authentication_on lr_T_N1
        weak_authentication_on t_LR_Psi_LR


end goal
```

```
environment()
```