Formal Analysis of Privacy for Anonymous Location Based Services

Morten Dahl[1], Stéphanie Delaune[2], and Graham Steel[2]

[1] Department of Computer Science, Aalborg University
[2] LSV, ENS Cachan & CNRS & INRIA Saclay Île-de-France

Abstract. We propose a framework for formal analysis of privacy in location based services such as anonymous electronic toll collection. We give a formal definition of privacy, and apply it to the VPriv scheme for vehicular services. We analyse the resulting model using the ProVerif tool, concluding that our privacy property holds only if certain conditions are met by the implementation. Our analysis includes some novel features such as the formal modelling of privacy for a protocol that relies on interactive zero-knowledge proofs of knowledge and list permutations.

1 Introduction

The sophistication and quantity of embedded devices in modern vehicles is growing rapidly. Ad-hoc wireless networking is envisioned as one of the next big steps, with various car-to-infrastructure and car-to-car communication applications planned [12, 14]. Many of these applications are location-based, and providing the precise position of the vehicle is essential to the quality of the service provided. As these applications are deployed, privacy concerns naturally emerge.

Some of the location-based services already in widespread use today, such as RFID tag based electronic toll collection systems, offer little privacy protection to drivers [16]. By using the same fixed identifying tag whenever they have to pay a toll fee, it becomes trivial to later trace the routes of any driver given the database of payments. Little is gained by using a fixed random tag instead of a real-world identifier such as the license plate. Although the tolling database may not be publicly available, the privacy of drivers is still at risk of exploitation from within the toll company.

The more widespread employment of such systems, combined with the possibility of moving them to the emerging general framework for network communication, increases the need for privacy oriented systems. In this paper, we bring the privacy analysis of location-based services into the world of formal methods, leveraging previous work on privacy for vehicular mix-zones [10], electronic voting [11, 15], and RFID tags [3, 8]. In particular, we concentrate on VPriv [7], a proposed scheme for building location-based services using zero-knowledge techniques, designed to ensure that the paths of drivers are not revealed to the service providers, while nonetheless preventing drivers from reporting fake paths. We use the formal notion of indistinguishability to formalise privacy and carry out the

analysis with the aid of the protocol analysis tool ProVerif [5]. In particular, we will use a notion of trace equivalence, after explaining why the more usual notion of observational equivalence is not suitable in this setting. To the best of our knowledge this is the first use of the tool for analysing a protocol that relies on interactive zero-knowledge proofs of knowledge. Note that contrary to non-interactive ones that can be abstracted by means of an appropriate equational theory (see *e.g.* [4]), interactive proofs cannot since the interactions between the participants reveal some information that has to be considered when we carry out the privacy analysis.

2 The VPriv Scheme

In this section we introduce the VPriv scheme [7], a protocol that offers a variety of location-based vehicular services such as "pay-as-you-go" insurance, electronic toll collection, etc. Its goal is to both protect the privacy of drivers whilst ensuring that they cannot cheat service providers by, for instance, paying a lower price.

2.1 Description

The participants are a set of users with vehicles and a service provider. We assume that time is split into periods. The following three phases detailed below are executed in order by each vehicle during each period. At the start of a period, the vehicle generates fresh random tags for the period and registers commitments to encrypted versions of these with the service provider (registration phase). Then, whenever the vehicle must emit a message containing an identifier during the period it will choose a new tag from its set of fresh tags. The tags are emitted in clear and the service provider records all tags v emitted by all vehicles together with the emission location l and a timestamp t, building a database containing a mixture of tuples (v, t, l) (driving phase). Finally, at the end of a period, each user initiates a protocol with the service provider in order to compute and settle the payable debts (reconciliation phase).

In the following, $[M]_d$ denotes a commitment to message M that can only be revealed with opening key d. Moreover, it is assumed to be a homomorphic commitment scheme, thus we have that:

$$[M_1]_{d_1} \cdot [M_2]_{d_2} = [M_1 + M_2]_{d_1+d_2}.$$

We further use $f_k(M)$ to denote a deterministic one-way function f that is parametrized by a key k. Note that knowing $f_k(M)$ and k does not allow one to retrieve M but only to compute a new encryption matching $f_k(M)$.

Phase 1 - Registration. Each vehicle generates a set V of fresh tags $v_1, \ldots, v_n$ and a set of fresh keys $k^1, \ldots, k^s$ for f. It furthermore generates opening keys $dk^1, \ldots, dk^s$ and $dv_1^1, \ldots, dv_n^s$. It then forms s *round packages*

$$\mathcal{V} \rightarrow \mathcal{S} : \; r^i = \left(id, i, [k^i]_{dk^i}, [f_{k^i}(v_1)]_{dv_1^i}, \ldots, [f_{k^i}(v_n)]_{dv_n^i}\right)$$

consisting of the *round number* $i \in [1; s]$, a commitment to the *round key* k^i, and commitments to encryptions of the vehicle's tags under the round key. The vehicle sends the round packages to the server together with a fixed identifier id for the user, such as the vehicle's license plate.

Phase 2 - Driving. Each vehicle emits its tags $v_1, \ldots, v_n$ in random order along its route. The server records these tags along with the location l where it was emitted and a timestamp t.

Phase 3 - Reconciliation. The server starts the reconciliation phase by sending the list W of all m tags w_j in its database together with their associated cost c_j, *i.e.* W contains all tags emitted by all vehicles during the period. Then the vehicle computes C as the sum of the costs of its own tags contained in W and sends this back to the server

$$\begin{aligned} &\mathcal{S} \to \mathcal{V} : W = \Big[(w_1, c_1), \ldots, (w_m, c_m)\Big] \\ &\mathcal{V} \to \mathcal{S} : id, C \end{aligned}$$

The remaining part of the protocol consists of several rounds. For each round i, the vehicle generates opening keys $dc_1^i, \ldots, dc_m^i$. Then, it processes all pairs in W by encrypting the tag w_j under its round key k^i and committing to the associated cost c_j using opening key dc_j^i. It permutes the pairs using a random permutation σ^i and sends the resulting list U^i to $\mathcal{S}$ together with its identifier. Then, the server decides to either verify that U^i is indeed the correct processing of W under k^i and $dc_1^i, \ldots, dc_m^i$ or to verify that the user has correctly calculated the cost C. In the former case it sends $b^i = 0$ to the vehicle and in the later case $b^i = 1$.

$$\begin{aligned} &\mathcal{V} \to \mathcal{S} : id, U^i = \Big[\big(f_{k^i}(w_{\sigma^i(1)}), [c_{\sigma^i(1)}]_{dc_1^i}\big), \ldots, \big(f_{k^i}(w_{\sigma^i(m)}), [c_{\sigma^i(m)}]_{dc_m^i}\big)\Big] \\ &\mathcal{S} \to \mathcal{V} : b^i \\ &\mathcal{V} \to \mathcal{S} : \begin{cases} id, dk^i, dc_1^i, \ldots, dc_m^i & \text{if } b^i = 0 \\ id, dv_1^i, \ldots, dv_n^i, D^i & \text{if } b^i = 1 \end{cases} \end{aligned}$$

If $b^i = 0$ the server receives $dk^i, dc_1^i, \ldots, dc_m^i$ from the vehicle. It can then obtain k^i from r^i and verify that U^i correctly follows from W. If $b^i = 1$ the server receives $dv_1^i, \ldots, dv_n^i$ to open the commitments in r^i to obtain the encrypted version of the vehicle's tags $f_{k^i}(v_1), \ldots, f_{k^i}(v_n)$. Knowing these it can pick out the pairs from U^i belonging to the vehicle (by the deterministic nature of f_{k^i}). It multiplies the cost commitments from these together and verifies that they indeed open to C under opening key D^i that is provided by the vehicle.

If the above mentioned checks pass for every round then the server accepts, the client is billed, and a new period begins. It can be argued that the probability of a cheating vehicle convincing the server of a false cost is 2^{-s}. This probability can hence be made arbitrarily low by choosing a large enough number of rounds.

Spot checks. Note that there is no mechanism in the protocol described above that prevents vehicles from cheating by not emitting the tags they have committed to in order to reduce the price they have to pay. To address this the protocol suggests random spot checks. A spot check consists of an enforcement device that secretly collects identifying data about passing vehicles at locations where they are supposed to emit tags, for instance by taking a photograph of the license plate. This way it will obtain a database DB of tuples (id, v, l, t) each with an identity id, a tag v, a location l and a timestamp t. Before the reconciliation phase the server will ask the vehicle about its identity id, challenge it with $\{(l, t) \mid (id, v, l, t) \in DB\}$, and *e.g.* fine the user if it fails to provide matching tags or the provided tags are not in the server's database. Since the location of the spot checks are assumed to be unknown to the vehicles, it can be argued that they do not know when they can safely avoid emitting a (valid) tag and hence must do so when they are supposed to. Note that some privacy is leaked due to the spot checks; it is argued that this is at an acceptable level.

2.2 Privacy

The privacy definition for the VPriv scheme as stated in [7] asks that the privacy guarantees from the system are the same as those of a system in which the server, instead of storing tuples (v, t, l), stores only tag-free path points (t, l). In other words, from the server's point of view, the tags might just as well be uncorrelated and random. This definition accounts for the fact that some privacy leaks are unavoidable and should not be blamed on the system. For instance, if one somehow learns that only a single vehicle was on a certain road at a particular time, then that vehicle's tags can of course be linked to the tags emitted along the road at that time.

3 Formal Model

The process calculus used as input to the ProVerif tool is a variant of the applied pi calculus [1], a process calculus for formally modelling concurrent systems and their interactions. We here recall the basic ideas and concepts of this calculus that are needed for our analysis.

3.1 Messages

To describe messages, we start with a set of *names* used to identify communication channels and other atomic data, a set of *variables* $x, y, \ldots$ and a signature Σ formed by a finite set of *function symbols* $\mathsf{f}, \mathsf{g}, \ldots$ each with an associated arity. Function symbols are distinguished into two categories: *constructors* and *destructors*. We use standard notation for function application, *i.e.* $\mathsf{f}(M_1, \ldots, M_n)$. Constructors are used for building messages. Destructors represent primitives for taking messages apart and can visibly succeed or fail (while constructors always

succeed). Messages $M, N, \ldots$ are obtained by repeated application of constructors on names and variables whereas a term evaluation D can also use destructors. The semantics of a destructor g of arity n is given by a set of rewrite rules of the form $\mathsf{g}(M_1, \ldots, M_n) \rightarrow M_0$ where $M_0, \ldots, M_n$ are messages that only contain constructors and variables. Given a term evaluation D, we write $D \Downarrow M$ when D can be reduced to M by applying some destructor rules.

In the following, we consider constructors to model commitments and the one-way function f. Since there is no destructor associated to f we have only one destructor whose semantics is given by the following rule:

$$\mathsf{open}(\mathsf{com}(x, y), y) \rightarrow x.$$

The applied pi calculus is quite general: it allows us, for instance, to model the homomorphism property of the commitment scheme by means of an equational theory containing, among others, the equation

$$\mathsf{com}(x_1, y_1) \times \mathsf{com}(x_2, y_2) = \mathsf{com}(x_1 + x_2, y_1 + y_2).$$

However, since ProVerif will not be able to reason with this equation, we will remove the homomorphic property in Section 5 and instead consider a simplified version of the protocol with no costs.

3.2 Processes

Processes are built from the grammar described below, where M is a message, D is a term evaluation, n and c are names, x is a variable, and i is a positive integer.

$P, Q, R ::=$	processes
0	null process
$P \mid Q$	parallel composition
$!P$	replication
new $n; P$	name restriction
let $M = D$ in P else Q	term evaluation
$\mathrm{in}(c, M); P$	message input
$\mathrm{out}(c, M); P$	message output
phase $i; P$	phase separation

The process "let $M = D$ in P else Q" tries to evaluate D; if this succeeds and if the resulting message matches the term M then the variables in M are bound and P is executed; if not then Q is executed. As explained in [5], the process phase $i; P$ indicates the beginning of phase i. Intuitively, the process first executes phase 0, that is, it executes all instructions not under phase $i \geq 1$. Then, when changing from phase i to phase $i + 1$, all processes that have not reached a phase $i' \geq i + 1$ instruction are discarded and the instructions under phase $i + 1$ are executed. The rest of the syntax is quite standard. To ease the presentation we will use tuples of messages, denoted by parentheses, while

keeping the reduction rules for these tuples implicit. We will omit "else Q" when Q is 0. In the remainder of the paper, we use the more intuitive notation "if $M = N$ then P else Q" instead of "let $M = N$ in P else Q".

An *evaluation context* C is a process with a hole, built from $[\_]$, $C \mid P$, $P \mid C$ and new $n; C$. We obtain $C[P]$ as the result of filling the hole in C with P. A process P is *closed* if all its variables are bound through an input or a let construction.

The vehicle process. To illustrate the calculus used throughout this paper, we give in Figure 1 a partial description of the vehicle process. It follows the description given in the previous section but is simplified in several aspects to keep this illustrative example as simple as possible. A more accurate model is described in Section 5.

Here we consider the case where a vehicle only has two tags v_1 and v_2, and where the reconciliation phase consists of only one round. We assume that during the driving phase the vehicle will visit only two locations and that the vehicle is spot checked at the second location. The vehicle receives a list of tags of size three (in reality, the length of the list is not known *a priori*), and instead of applying a random permutation, we only encode one particular permutation.

$$
\begin{array}{l}
\mathsf{Vehicle}(id, l_1, l_2) \stackrel{\mathsf{def}}{=} \\
\quad \text{phase } 1; (* \textit{ registration phase } *) \\
\quad \text{new } v_1;\ \text{new } v_2; \\
\quad \text{new } k;\ \text{new } dk;\ \text{new } dv_1;\ \text{new } dv_2; \\
\quad \text{out}(c, (id, \mathsf{com}(k, dk), \mathsf{com}(\mathsf{f}(v_1, k), dv_1), \mathsf{com}(\mathsf{f}(v_2, k), dv_2))); \\
\quad \text{phase } 2; (* \textit{ driving phase } *) \\
\quad \text{out}(c, (l_1, v_1));\ \text{out}(c, (l_2, v_2, id)); \\
\quad \text{phase } 3; (* \textit{ reconciliation phase } *) \\
\quad \text{in}(c, (x_1, x_2, x_3)); \\
\quad \text{out}(c, (id, \mathsf{f}(x_2, k), \mathsf{f}(x_1, k), \mathsf{f}(x_3, k))); \\
\quad \text{in}(c, y); \\
\quad \text{if } y = \mathsf{false} \text{ then out}(c, (id, dk)) \text{ else out}(c, (id, dv_1, dv_2))
\end{array}
$$

Fig. 1. Illustrative vehicle process

The operational semantics of processes is essentially defined by two relations, namely *structural equivalence* and *reduction*. Structural equivalence, denoted by $\equiv$, is the smallest equivalence relation on processes that is closed under application of evaluation contexts and standard rules such as associativity and commutativity of the parallel operator. Moreover, in order to deal with the phase

construct, we have also the following rules (see [6]):

$$\begin{aligned} \text{new } n; \text{phase } i; P &\equiv \text{phase } i; \text{new } n\ ; P \\ \text{phase } i; (P \mid Q) &\equiv \text{phase } i; P \mid \text{phase } i; Q \\ \text{phase } i; \text{phase } i'; P &\equiv \text{phase } i'; P \quad \text{if } i < i' \end{aligned}$$

Reduction at phase i, denoted by $\rightarrow_i$, is the smallest relation closed under structural equivalence and application of evaluation contexts such that:

RED I/O	$\text{phase } i; (\text{out}(c, M).Q \mid \text{in}(c, N).P) \rightarrow_i \text{phase } i; (Q \mid P\sigma)$
RED FUN 1	$\text{phase } i; \text{let } N = D \text{ in } P \text{ else } Q \rightarrow_i \text{phase } i; P\sigma \quad \text{if } D \Downarrow M$
RED FUN 2	$\text{phase } i; \text{let } N = D \text{ in } P \text{ else } Q \rightarrow_i \text{phase } i; Q$ if there is no M such that $D \Downarrow M$
REPL	$\text{phase } i; !P \rightarrow_i \text{phase } i; (P \mid !P)$

where σ is the substitution defined on the variables that occur in N and such that $M = N\sigma$. In case such a substitution does not exist, the resulting process will be $Q \mid \text{in}(c, N).P$ for the RED I/O rule and Q for the RED FUN 1 rule. We denote $\rightarrow = \bigcup_{i \geq 0} \rightarrow_i$ and we write $\rightarrow^*$ for the reflexive and transitive closure of reduction.

4 Privacy for Interactive Zero-Knowledge Protocols

We will define privacy using indistinguishability, which in turn will be formalized by a notion of equivalence. Equivalences have already been used to model privacy properties in formal analysis for *e.g.* vehicular mix-zones [10] and electronic voting [11, 15]. The precise notion used is often *observational equivalence* but as we will explain, it happens that this notion is too strong to analyse interactive zero-knowledge protocols. So, we will rely on *trace equivalence* to formalize our notion of privacy in Section 4.2. However, the only equivalence relation supported by ProVerif is a stronger notion called *diff-equivalence*, and thus we explain in Section 4.3 how to use this tool to analyse trace equivalence-based properties.

4.1 Equivalences

One equivalence notion for formalizing indistinguishability is observational equivalence [17]. Here we write $P \Downarrow_c$ when P can send an observable message on the channel c; that is, when $P \rightarrow^* C[\text{phase } i; \text{out}(c, M); Q]$ for some evaluation context C that does not bind c, some message M, some process Q, and some integer i.

Definition 1 (Observational equivalence). Observational equivalence, *denoted* $\sim_o$, *is the largest symmetric relation* $\mathcal{R}$ *on closed processes* P *and* Q *such that* $P\ \mathcal{R}\ Q$ *implies:*

1. *if* $P \Downarrow_c$ *then* $Q \Downarrow_c$*;*
2. *if* $P \to P'$ *then there exists* Q' *such that* $Q \to^* Q'$ *and* $P' \mathcal{R} Q'$*;*
3. $C[P] \mathcal{R} C[Q]$ *for all evaluation contexts* C*.*

Intuitively, a context may represent an attacker, and two processes are observationally equivalent if they cannot be distinguished by any attacker at any step: every output step in an execution of process P must have an indistinguishable equivalent output step in the execution of process Q. If not then there exists a context that 'breaks' the equivalence.

In the case of privacy for the VPriv protocol, we will see that this notion is too strong (see the discussion in Section 4.2). Instead, we will use the notion of *trace equivalence* (also called *testing equivalence* in some other contexts [2]).

Definition 2 (Trace equivalence). Trace equivalence $\sim_t$ *is the largest symmetric relation on closed processes* P *and* Q *such that for all evaluation contexts* C *we have* $C[P] \Downarrow_c$ *if and only if* $C[Q] \Downarrow_c$*.*

This is a strictly weaker notion than observational equivalence (see *e.g.* [9]) but intuitively it captures the equivalence upon which we can *a priori* hope to base our privacy property, as we explain below.

4.2 Formal Definition of Privacy

In our formal privacy definition we will assume that we have at least two honest vehicles called A and B. As we are interested in studying privacy guarantees for A, the process V_A for this vehicle will consist of all three phases of the protocol (registration, driving, and registration). We assume that vehicle A has three tags, one of which is emitted at one of the two locations $route_{left}$ and $route_{right}$, one which is 'leaked', *i.e.* given to the server along with the vehicle's identity to model the spot-check procedure, and one which is not emitted. On the other hand, vehicle B is only needed to counterbalance the effect of the tag emitted by A at a *route* location. Thus, we will consider a vehicle B that only executes its driving phase, denoted V_B^{dri} in the equivalence below, by emitting its tag at the *route* not visited by vehicle A.

We say that privacy holds if the following equivalence holds:

$$C_T\big[\, V_A(route_{left}) \mid V_B^{\mathsf{dri}}(route_{right}) \big] \sim_t C_T\big[\, V_A(route_{right}) \mid V_B^{\mathsf{dri}}(route_{left}) \big]$$

where C_T is an evaluation context modelling additional assumptions that may have to be made for the privacy property to hold (*e.g.* that the server is curious but otherwise assumed to be honest and following the protocol, or the existence of a trusted third party helping vehicles ensure that the list of tags received from the server contains tags from both vehicles). The next section presents the

analysis we have performed, including the definition of the vehicles processes and the different contexts C_T within which we have performed the analysis.

Note that observational equivalence would be too strong for this property to hold. This is due to the interactive zero-knowledge subprotocol that occurs in the reconciliation phase. Consider the two slightly different processes $V_A(route_{left})$ and $V_A(route_{right})$ in our privacy definition and assume that the two processes have reached the reconciliation phase. At this point, the server will send a list of tags to vehicle A. Then one of the two processes, say the former one, will commit to a permuted list. To mimic this step, the latter process has also to commit to a permuted list. However, no matter what list it commits to, this list will not mimic the former process' list for either $b = 0$ or $b = 1$ because of the slight difference between them. In other words, the choice of a list to mimic the former process depends on the challenge bit b that has not yet been received from the server. Thus observational equivalence is impossible to achieve. However, moving from observation equivalence to trace equivalence allows us to choose the mimic trace only after the challenge bit has been learned. Intuitively, this captures privacy: if an attacker observes a trace of registration, tag emission and reconciliation, and then guesses that vehicle A took a particular route, then there is an equivalent trace where vehicle A takes a different route. The fact that we cannot specify the equivalent trace until we have seen the whole of the first trace does not seem to lead to any loss of privacy. In fact, from the definition of zero-knowledge protocols in the computational model [13, §4] we see that the protocol is actually designed to support only trace equivalence and furthermore, that soundness contradicts observational equivalence.

4.3 Checking Privacy with ProVerif

The basic idea behind equivalence checking in ProVerif is to overlap the two processes that are supposedly equivalent, thereby forming a *biprocess* B. To achieve this, the syntax of ProVerif contains a $choice[M, M']$ operator which allows us to model a pair of processes that have the same structure and differ only in the choice of terms. Given a biprocess B, the process $P = fst(B)$ is obtained by replacing all occurrences of $choice[M, M']$ in B with M. Similarly, $Q = snd(B)$ is obtained by replacing $choice[M, M']$ with M'. When ProVerif is able to conclude positively on B, this implies that $P \approx_o Q$. However, ProVerif checks a stronger equivalence than observational equivalence and hence it fails on some simple examples of processes that are equivalent, but whose equivalence cannot be simulated by the moves of a single biprocess.

We will use two transformations in order to use ProVerif to check the trace equivalence defining our privacy property. The first arises from recent work which shows how to use ProVerif to prove observational equivalence for a wider class of processes [11]. Additionally, we also transform our biprocess B into another biprocess B' that preserves the traces of each underlying process, *i.e.* $fst(B)$ and $fst(B')$ will produce the same traces, and likewise for $snd(B)$ and $snd(B')$. This ensures that our transformation preserves trace equivalence. In our case study

this transformation consists of guessing b in advance and deadlocking the process if it later turns out that the guess was wrong.

5 Privacy Analysis

The purpose of our analysis is to investigate the privacy guarantees provided for an honest user in the VPriv protocol. We do not attempt to analyse whether users can cheat the server nor whether the server will accuse an honest user of cheating.

Section 5.1 contains a description of the simplifications we had to make in order to carry through the analysis in ProVerif. In Section 5.2 we describe our formal model of the VPriv protocol using the applied pi calculus from the previous section. We give the results of our analysis in Section 5.3.

5.1 Simplifications

The following simplifications were necessary in order to carry through the analysis in ProVerif.

Removing Costs. In the extreme case where a unique price is used for every tag, the system cannot protect the privacy of users. It seems reasonable however, to assume that the information leaked by costs will in practice not affect the privacy of users. Forcing a uniform cost for every tag seems to be the only solution if we want to carry out our analysis with ProVerif. Furthermore, while we could model the homomorphic commitment scheme and its arithmetic properties by means of an equational theory, we know that ProVerif will not be able to deal with it properly in that it will not terminate. Thus, we remove prices and costs and proceed with a simplified version of the VPriv protocol. This change only affects the reconciliation phase where the list W sent by the server is now simply

$$\mathcal{S} \to \mathcal{V} : W = \left[w_1, \ldots, w_m\right]$$

and the round subprotocol as described in Figure 2.

$$\begin{aligned}
&\mathcal{V} \to \mathcal{S} : id, U^i = \left[f_{k^i}(w_{\sigma^i(1)}), \ldots, f_{k^i}(w_{\sigma^i(m)})\right] \\
&\mathcal{S} \to \mathcal{V} : b^i \\
&\mathcal{V} \to \mathcal{S} : \begin{cases} id, dk^i & \text{if } b^i = 0 \\ id, dv_1^i, \ldots, dv_n^i & \text{if } b^i = 1 \end{cases}
\end{aligned}$$

Fig. 2. Reconciliation round protocol without cost.

Fixing the length of W. It turns out that privacy can be violated if the list of tags sent by the server is blindly accepted by the vehicles without any scrutiny. Some sanity conditions must be fulfilled in order to guarantee privacy. Furthermore, implementing these sanity checks together with the random permutation would lead us to consider a complex model that ProVerif is not able to handle. So instead, we fix *a priori* the length of the list expected by the vehicle to a size of three. This will allow us to easily encode the sanity checks and the random permutation, and despite its simplicity, still allow us to discover a number of issues to which attention should be paid when implementing the protocol. Note that with the sanity conditions discovered we can argue that fixing the length to three does not weaken the attacker.

5.2 Analysis Model

The model is represented by the biprocess B_S defined in Figure 3. In the following we show only the main details of this process and refer the interested reader to the full model available at `http://www.cs.aau.dk/~dahl/vpriv/`.

$$
\begin{array}{l}
B_S(id^A, v_1^A, v_2^A, v_3^A, v_1^B, v_l, v_r) \stackrel{\mathsf{def}}{=} \\
\quad \text{new } pc; \\
\qquad \text{phase } 2; B_{\mathsf{dri}} \\
\qquad \mid \text{phase } 3; B_{\mathsf{BB}} \\
\qquad \mid !\ \text{new } k, dk, dv_1, dv_2, dv_3\ \left(\text{phase } 1; V_{\mathsf{reg}}; \text{phase } 3; V_{\mathsf{rec}}^{b=0}\right) \\
\qquad \mid !\ \text{new } k, dk, dv_1, dv_2, dv_3\ \left(\text{phase } 1; V_{\mathsf{reg}}; \text{phase } 3; V_{\mathsf{rec}}^{b=1}\right)
\end{array}
$$

Fig. 3. System Biprocess

The system B_S consist of five parts: B_{dri}, V_{reg}, $V_{\mathsf{rec}}^{b=0}$, $V_{\mathsf{rec}}^{b=1}$, and B_{BB}. The first four of these together make up the behaviour of vehicle A and vehicle B. Using the choice operator the emitter biprocess B_{dri} outputs the tags of both vehicles while V_{reg} and the two V_{rec} are responsible for performing registration and reconciliation, respectively, for vehicle A. By splitting up vehicle A in this way we accurately model an unbounded number of reconciliation rounds while only emitting tags once. The bulletin board B_{BB} is responsible for performing sanity checks on W. It receives a list of tags on a public channel and forwards the list to the V_{rec} biprocesses on the private channel pc an unbounded number of times if the checks succeed. Note that to avoid trivial false attacks, any checks against v_1^A and v_1^B must use the choice operator and hence the bulletin board is a biprocess. Finally, we use ProVerif's phases to orchestrate the processes so that they follow the order dictated by the protocol.

As discussed in Section 4, in order to establish the equivalence between the two cases, the selection of permutation for U^i depends upon the bit b^i that will be send by the server. We have two separate reconciliation processes $V_{\mathsf{rec}}^{b=0}$ and $V_{\mathsf{rec}}^{b=1}$

to model this. They guess that $b = 0$ and $b = 1$ will be sent, respectively, and permute accordingly. If the guess was correct the process proceeds as dictated by the protocol, otherwise it comes to a deadlock. Formally, let process P_{xyz} be defined by

$$P_{xyz} = \text{in}(s_1, \cdot); \text{out}\Big(c, \big(id^A, f(w_x, k), f(w_y, k), f(w_z, k)\big)\Big); \text{out}(s_2, \cdot); 0$$

which outputs the encrypted tags w_1, w_2 and w_3 permuted according to xyz. The initial input on s_1 is used to ensure that only a single permutation is selected and the final output on s_2 to indicate that the output was completed. The $\cdot$ stands for any name never used after it is bound. Using this process we then define $V_{\mathsf{rec}}^{b=0}$ as shown in Figure 4 and $V_{\mathsf{rec}}^{b=1}$ as shown in Figure 5. We note that because of the diff-equivalence that ProVerif is actually checking (see Section 4.3) it will only try to match the permutations at the same syntactical position. This means that we have to specify to ProVerif how permutations should be matched. For $V_{\mathsf{rec}}^{b=0}$ we can choose the same permutation in the two cases and hence no further modelling is needed and $V_{\mathsf{rec}}^{b=0}$ is actually just a process. However, this is not true for $V_{\mathsf{rec}}^{b=1}$ where we have to move the processes P_{xyz} around depending on which case we are in. We do this using the choice operator and hence $V_{\mathsf{rec}}^{b=1}$ is a biprocess. Let v_l be the tag emitted at $route_{left}$ and v_r the tag emitted at $route_{right}$. We have then chosen to arrange the permutations based on $w_1 = v_l$ and $w_3 = v_r$ and hence need to enforce this in the bulletin board.

$$\begin{aligned}
&V_{\mathsf{rec}}^{b=0}(id^A, pc, k, dk) \stackrel{\mathsf{def}}{=} \\
&\qquad \text{new } s_1, s_2; \\
&\qquad \text{in}(pc, (w_1, w_2, w_3)); \\
&\qquad\quad \text{out}(s_1, \cdot); 0 \\
&\qquad\quad \mid P_{123} \mid P_{132} \mid P_{213} \mid P_{231} \mid P_{312} \mid P_{321} \\
&\qquad\quad \mid \text{in}(s_2, \cdot); \text{in}(c, b);\ \text{if } b = 0 \text{ then out}(c, (id^A, dk))
\end{aligned}$$

Fig. 4. Reconciliation process for $b = 0$

$$\begin{aligned}
&V_{\mathsf{rec}}^{b=1}(id^A, pc, k, dv_1, dv_2, dv_3) \stackrel{\mathsf{def}}{=} \\
&\qquad \text{new } s_1, s_2; \\
&\qquad \text{in}(pc, (w_1, w_2, w_3)); \\
&\qquad\quad \text{out}(s_1, \cdot); 0 \\
&\qquad\quad \mid P_{123} \mid P_{132} \mid P_{213} \mid P_{231} \mid P_{312} \mid P_{321} \\
&\qquad\quad \mid \text{in}(s_2, \cdot); \text{in}(c, b);\ \text{if } b = 1 \text{ then out}(c, (id^A, dv_1, dv_2, dv_3))
\end{aligned}$$

Fig. 5. Reconciliation process for $b = 1$

5.3 Analysis Results

Unsurprisingly, it turns out that we have no privacy if W only contains tags of a single vehicle. It is necessary to ensure that the tags of both of the two honest vehicles are included in W, *i.e.* that W contains at least v_1^A (the tag emitted by vehicle A at its *route* location), and v_1^B (the tag emitted by the vehicle B at its *route* location). Actually, this is not sufficient since the server can still break privacy by sending a list with duplicates. An attack using this trick was reported by ProVerif.

With the above model P_S we performed several analyses by varying the sanity checks performed by the bulletin board. For the simplest case *without any checks* on W an attack is reported where arbitrary values are sent for $w_1 = w_2$ and w_3. This is a false attack caused by the way we match permutations in $V_{\mathsf{rec}}^{b=1}$. We can investigate the need for checks by removing all $V_{\mathsf{rec}}^{b=1}$. Then a real attack is reported: by sending arbitrary values for w_1 and w_2 and $w_3 = v_l$ the server can tell the cases apart when it sends $b = 0$.

In the case with W subject to *inclusion checks only* the attacker is allowed to choose w_2 but must send $w_1 = v_l$ (i.e. whichever tag was emitted in the left location) and $w_3 = v_r$ (whichever tag was emitted on the right). An attack is found when $w_2 = v_l$ by a comparison of the encrypted elements of U^i.

Finally, for the case with W subject to *inclusion checks and no duplicates* ProVerif is unable to conclude when *no duplicates* is interpreted as $w_2 \neq v_l \wedge w_2 \neq v_r$. However, if we interpret this as $w_2 = v_2^A \vee w_2 = v_3^A$, i.e. rather than using an arbitrary tag not equal to v_l or v_r, the attacker must specifically use one of the unused registered tags, ProVerif is able to prove the equivalence and thus the privacy property for our model.

5.4 Evaluation

We evaluate first the VPriv protocol, then our analysis. Results on the privacy-preserving properties of the protocol are largely positive, at least in our abstract model. We discovered only privacy breaches that are possible for an active attacker who can tamper with the list, not for an 'honest but curious' attacker who merely inspects the protocol trace. We proposed some checks that could be made on the list W to thwart even an active attacker. The check for no duplicates is easy enough for a single vehicle to apply, but the check that the list really contains the tags of other vehicles is less easy and may require a trusted third party.

Turning to our analysis, it should be clear that a reasonable amount of work was required to develop an abstract model suitable for ProVerif whilst preserving the features of the protocol. However, it was not our aim to formalise the protocol just to exemplify the use of ProVerif but rather to push the boundaries of the tool in terms of protocol features. As such we have succeeded in identifying several features that a future version of the tool might handle better, namely lists, permutations, and homomorphic encryption schemes.

6 Conclusion

We have presented a privacy analysis of the VPriv scheme for anonymous location-based vehicular services. We have shown how a notion of trace equivalence captures the privacy notion the protocol is intended to provide, and have formally verified this property, albeit for an abstract model of the protocol. During our analysis we uncovered a number of areas where special attention needs to be paid when implementing such a protocol. We also introduced novel features into formal privacy modelling such as random list permutations and reasoning about interactive zero-knowledge protocols.

In future work we plan to investigate proofs of soundness for abstractions in the context of privacy properties, and apply our method to other privacy-enhancing protocols. In particular, it would be interesting to investigate a more general approach to reasoning about zero-knowledge protocols using the ProVerif tool set.

References

1. M. Abadi and C. Fournet. Mobile values, new names, and secure communication. In *Proc. 28th ACM Symposium on Principles of Programming Languages (POPL'01)*, pages 104–115, New York, USA, 2001. ACM Press.
2. M. Abadi and A. Gordon. A calculus for cryptographic protocols: The spi calculus. In *Proc. 4th ACM Conference on Computer and Communications Security*, pages 36–47, Zurich (Switzerland), 1997. ACM Press.
3. M. Arapinis, T. Chothia, E. Ritter, and M. Ryan. Analysing unlinkability and anonymity using the applied pi calculus. In *Proc. 23rd IEEE Computer Security Foundations Symposium (CSF'10)*, pages 107–121. IEEE Computer Society Press, 2010.
4. M. Backes, M. Maffei, and D. Unruh. Zero-knowledge in the applied pi-calculus and automated verification of the direct anonymous attestation protocol. In *Proc. Symposium on Security and Privacy (S&P'08)*, pages 202 –215. IEEE Computer Society Press, 2008.
5. B. Blanchet. *Cryptographic Protocol Verifier User Manual*, 2004. `http://www.di.ens.fr/~blanchet/crypto/proverif-manual.ps.gz`.
6. B. Blanchet, M. Abadi, and C. Fournet. Automated verification of selected equivalences for security protocols. *Journal of Logic and Algebraic Programming*, 75(1):3–51, 2008.
7. A. J. Blumberg, H. Balakrishnan, and R. Popa. VPriv: Protecting privacy in location-based vehicular services. In *Proc. 18th Usenix Security Symposium*, 2009.
8. M. Bruso, K. Chatzikokolakis, and J. den Hartog. Formal verification of privacy for RFID systems. In *Proc. 23rd IEEE Computer Security Foundations Symposium (CSF'10)*. IEEE Computer Society Press, 2010.
9. V. Cortier and S. Delaune. A method for proving observational equivalence. In *Proc. 22nd IEEE Computer Security Foundations Symposium (CSF'09)*, pages 266–276, Port Jefferson, NY, USA, 2009. IEEE Computer Society Press.
10. M. Dahl, S. Delaune, and G. Steel. Formal analysis of privacy for vehicular mix-zones. In *Proc. 15th European Symposium on Research in Computer Security (ESORICS'10)*, volume 6345 of *LNCS*, pages 55–70. Springer, 2010.

11. S. Delaune, M. D. Ryan, and B. Smyth. Automatic verification of privacy properties in the applied pi-calculus. In *Proc. 2nd Joint iTrust and PST Conferences on Privacy, Trust Management and Security (IFIPTM'08)*, volume 263 of *IFIP Conference Proceedings*, pages 263–278. Springer, 2008.
12. M. D. Dikaiakos, S. Iqbal, T. Nadeem, and L. Iftode. VITP: an information transfer protocol for vehicular computing. In *Proc. 2nd International Workshop on Vehicular Ad Hoc Networks (VANET'05)*, pages 30–39, 2005.
13. O. Goldreich. *The Foundations of Cryptography*, volume 1. Cambridge University Press, 2001.
14. IEEE. IEEE standard. IEEE Trial-Use Standard for Wireless Access in Vehicular Environments – Security Services for Applications and Management Messages, Approved 8 June 2006.
15. S. Kremer and M. D. Ryan. Analysis of an electronic voting protocol in the applied pi-calculus. In *Proc. 14th European Symposium on Programming Languages and Systems (ESOP'05)*, volume 3444 of *LNCS*, pages 186–200. Springer, 2005.
16. N. Lawson. Highway to hell: Hacking toll systems. Presentation at Blackhat, 2008. Slides available from `http://rdist.root.org/2008/08/07/fastrak-talk-summary-and-slides/`.
17. R. Milner. A calculus of communicating systems. *Lecture Notes in Computer Science*, 92, 1980.