

Formal Analysis of Privacy for Anonymous Location Based Services

Stéphanie Delaune, Morten Dahl and **Graham Steel**

University of Aalborg

and

LSV, INRIA & CNRS & ENS-Cachan

Context: Technological

- Location based services increasingly popular
- Privacy fears growing

We take vehicle toll collection as a case study

- Current systems not privacy-friendly, e.g. fixed static identifier
- More widespread deployment + VANET infrastructure has increased privacy concerns
- VPriv [Blumberg et al., USENIX '09] proposes a solution using zero-knowledge techniques

Context: Scientific

We are interested in verifying privacy-like properties of protocols in symbolic models

Privacy properties naturally expressed as indistinguishability from an attacker's point of view

Indistinguishability not as straightforward as trace properties (secrecy, authentication) to specify and verify

VPriv

Blumberg, Balakrishnan, Popa, USENIX 2009.

- Applications: insurance, tolling, etc.
- Aim: protect privacy of drivers while preventing cheating
- Uses interactive zero-knowledge proofs
- Privacy threat scenario: “honest but curious” central server

VPriv: How it works

Phase 1: Registration

- Driver generates n fresh random tags v_1, \dots, v_n
- Sends commitments to encrypted versions of these tags to server

VPriv: How it works

Phase 1: Registration

- Driver generates n fresh random tags v_1, \dots, v_n
- Sends commitments to encrypted versions of these tags to server

Phase 2 - Driving

- Tags emitted in a random order as vehicle drives about

VPriv: How it works

Phase 1: Registration

- Driver generates n fresh random tags v_1, \dots, v_n
- Sends commitments to encrypted versions of these tags to server

Phase 2 - Driving

- Tags emitted in a random order as vehicle drives about

Phase 3 - Reconciliation

- Server sends all tags and costs, vehicles computes and returns his cost

Then for s rounds:

- Vehicle sends permuted list of commitments to all tag, cost pairs in list
- Server demands proof either that all tags are in list, or that cost was correctly calculated

VPriv: Registration Phase

Each vehicle generates a set V of fresh tags v_1, \dots, v_n and a set of fresh keys k^1, \dots, k^s for f .

Also generates opening keys dk^1, \dots, dk^s and dv_1^1, \dots, dv_n^s .

Then forms s round packages

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

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.

VPriv: Driving Phase

Each vehicle emits its tags v_1, \dots, 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 .

(in our models, cars will visit either the 'left' location or the 'right' location)

VPriv: Reconciliation Phase (1)

$$\begin{aligned} \mathcal{S} \rightarrow \mathcal{V} : \quad W &= \left[(w_1, c_1), \dots, (w_m, c_m) \right] \\ \mathcal{V} \rightarrow \mathcal{S} : \quad id, C \end{aligned}$$

For each round i , the vehicle generates opening keys dc_1^i, \dots, 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.

VPriv: Reconciliation Phase (2)

Server decides to either verify that U^i is indeed the correct processing of W under k^i and dc_1^i, \dots, 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 $b^i = 1$.

$$\mathcal{V} \rightarrow \mathcal{S} : id, U^i = \left[(\text{enc}_{w_{\sigma^i(1)}}^{k^i}, [c_{\sigma^i(1)}]_{dc_1^i}), \dots, (\text{enc}_{w_{\sigma^i(m)}}^{k^i}, [c_{\sigma^i(m)}]_{dc_m^i}) \right]$$

$$\mathcal{S} \rightarrow \mathcal{V} : b^i$$

$$\mathcal{V} \rightarrow \mathcal{S} : \begin{cases} id, dk^i, dc_1^i, \dots, dc_m^i & \text{if } b^i = 0 \\ id, dv_1^i, \dots, dv_n^i, D^i & \text{if } b^i = 1 \end{cases}$$

VPriv: Reconciliation Phase (3)

If $b^i = 0$ the server receives $dk^i, dc_1^i, \dots, dc_m^i$ 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, \dots, dv_n^i to open the commitments in r^i to obtain the encrypted version of the vehicle's tags $enc_{v_1}k^i, \dots, enc_{v_n}k^i$.

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.

Applied Pi Syntax

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

Cryptographic Functions and Equational Theory

Homomorphic commitment scheme

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

Cryptographic Functions and Equational Theory

Homomorphic commitment scheme

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

Proverif will not terminate using this, hence assume uniform costs

Destructor symbol for commitments

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

Cryptographic Functions and Equational Theory

Homomorphic commitment scheme

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

Proverif will not terminate using this, hence assume uniform costs

Destructor symbol for commitments

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

Deterministic one-way function $f_k(X)$

No destructor symbol required

Security Properties: Equivalences

Intuition is that processes should be indistinguishable to an active attacker.

Observational equivalence, \sim_o

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

Proverif proves ‘diff equivalence’ - processes with same structure

– stronger than observational equivalence, and often too strong

We propose that trace equivalence, even weaker, is enough

Equivalences: Examples

$$P = \text{out}(a) \mid \text{out}(b), Q = \text{out}(b) \mid \text{out}(a)$$

Written in Proverif as:

$$\text{out}(\textit{choice}[a, b]) \mid \text{out}(\textit{choice}[b, a])$$

Equivalences: Examples

$$P = \text{out}(a) \mid \text{out}(b), Q = \text{out}(b) \mid \text{out}(a)$$

Written in Proverif as:

$$\text{out}(\textit{choice}[a, b]) \mid \text{out}(\textit{choice}[b, a])$$

- can use *swap* operator [Delaune, Ryan, Smyth'08] to help Proverif in this case

Equivalences: Examples

$$P = \text{out}(a) \mid \text{out}(b), Q = \text{out}(b) \mid \text{out}(a)$$

Written in Proverif as:

$$\text{out}(\text{choice}[a, b]) \mid \text{out}(\text{choice}[b, a])$$

- can use *swap* operator [Delaune, Ryan, Smyth'08] to help Proverif in this case

$$P = \begin{array}{c} a \rightarrow b \\ \searrow \\ c \end{array}, \quad Q = \begin{array}{c} \rightarrow a \rightarrow b \\ \searrow \\ a \rightarrow c \end{array}$$

Equivalences: Examples

$$P = \text{out}(a) \mid \text{out}(b), Q = \text{out}(b) \mid \text{out}(a)$$

Written in Proverif as:

$$\text{out}(\text{choice}[a, b]) \mid \text{out}(\text{choice}[b, a])$$

- can use *swap* operator [Delaune, Ryan, Smyth'08] to help Proverif in this case

$$\begin{array}{ccc} P = & a & \rightarrow & b \\ & \searrow & & \\ & & c & \end{array} , \quad \begin{array}{ccc} Q = & \rightarrow & a & \rightarrow & b \\ & \searrow & & \\ & & a & \rightarrow & c \end{array}$$

- not observational equivalent: P must 'commit' before Q

- but are *trace equivalent*

For all evaluation contexts C we have $C[P] \Downarrow_c$ if and only if $C[Q] \Downarrow_c$.

Modelling Reconciliation

The choice of b by the attacker means that observational equivalence can always be broken

– processes must commit to a list permutation before receiving b , and the relationship between the lists of the two processes depends on b .

We therefore model separate reconciliation processes for $b = 0$ and $B = 1$, and they deadlock if the wrong b is later received.

Security Property

$$C_T [V_A(\textit{route}_{\textit{left}}) \mid V_B^{\textit{dri}}(\textit{route}_{\textit{right}})]$$
$$\sim_t$$
$$C_T [V_A(\textit{route}_{\textit{right}}) \mid V_B^{\textit{dri}}(\textit{route}_{\textit{left}})]$$

If context C_T is arbitrary, represents active attacker

We add a ‘bulletin board’ process to restrict attacker behaviour in reconciliation phase

Summary of Simplifications

- One 'real car', one dummy car, two locations, two tags
- Uniform unit cost for tolls
- Fixed length of tag list
- Fixed permutations for cases $b = 0, 1$

Results

We progressively increase checks in bulletin board process

Results

We progressively increase checks in bulletin board process

- With no checks, attacker send only one real tag, finds attack

Results

We progressively increase checks in bulletin board process

- With no checks, attacker send only one real tag, finds attack
- With check for real tags, attacker sends duplicate tag, makes attack

Results

We progressively increase checks in bulletin board process

- With no checks, attacker send only one real tag, finds attack
- With check for real tags, attacker sends duplicate tag, makes attack
- With check for real tags, no duplicates, Proverif can prove equivalence

Evaluation

The VPriv Protocol

- No attacks found for the 'honest but curious' attacker
- 'Duplicate' attack easily detected, 'fake tags attack' less easy

Evaluation

The VPriv Protocol

- No attacks found for the ‘honest but curious’ attacker
- ‘Duplicate’ attack easily detected, ‘fake tags attack’ less easy

Analysis Approach

- Considerable work needed to make suitable model for Proverif
- Identified areas for improvement: lists and permutations, homomorphic schemes, weaker equivalences

Further Work

- Soundness of abstractions
- Dedicated procedures for proving symbolic equivalences
- More privacy protocols

<http://www.cs.aau.dk/~dahl/vpriv/>