Formalising privacy-type security properties using the applied pi calculus

Rémy Chrétien and Stéphanie Delaune

LSV, ENS Cachan & CNRS & INRIA Saclay Île-de-France

The results presented in this report are partially based on results that have been published in [4, 3] – works that have been supported by the VIP project and that have been added at the end of this report.

Abstract. Privacy is a general requirement that needs to be studied in different contexts. We identify some applications for wich privacy plays an important role, and with significant interest in terms of societal impact. Since each application leads to several definitions of privacy, and raise some particular modelling issues, we concentrate our efforts on the three following applications: electronic voting protocols, RFID tags, and routing protocols in mobile ad hoc networks. For each application, we show how to formalise different notions of privacy in the applied pi calculus (or extension of it).

1 Introduction

Formal methods have proved their usefulness for precisely analysing the security of protocols. However, most existing results focus on trace properties, that is, statements that something bad never occurs on any execution trace of a protocol. Secrecy and authentication, are typical examples of trace properties. There are however several security properties, which cannot be defined (or cannot be naturally defined) as trace properties and require a notion of *equivalence*. Intuitively, two processes P and Q are equivalent, denoted by $P \approx Q$, if for any process O (the observer) the processes $P \mid O$ and $Q \mid O$ are equally able to emit on a given channel. This means that the process O cannot observe any difference between the processes P and Q.

We focus here on the notion of static equivalence and trace equivalence proposed in the context of applied pi calculus [1], which is well-suited for the analysis of security protocols. First, we present the applied pi calculus in Section 2, and we formally define the notions of equivalence we are interested in.

Then, we consider privacy related properties involved in electronic voting protocols, in RFID protocols, or in routing protocols. We show how to formalise those properties, and each time we use an equivalence as a key notion. Those applications together with the privacy-type security properties are detailed in the remaining sections.

2 Applied pi calculus

The *applied pi calculus* [1] is a derivative of the pi calculus that is specialised for modelling cryptographic protocols. Participants in a protocol are modelled as processes, and the communication between them is modelled by means of message passing.

2.1 Syntax

We consider a set of *names*, which is split into the set $\mathcal{N} = \{a, b, k, n, \ldots\}$ of names of *base types* and the set $\mathcal{C}h$ of names of *channel type* (which are used to name communication channels). We also consider a set of *variables* $\mathcal{X} = \{x, y, \ldots\}$, and a *signature* $\mathcal{F}$ consisting of a finite set of *function symbols*. We rely on a sort system for terms. The details of the sort system are unimportant, as long as it distinguishes *base types* from the *channel type*. We suppose that function symbols only operate on and return terms of base types. *Terms* are defined as names, variables, and function symbols applied to other terms. We denote by $\mathcal{T}(\mathcal{F}, \mathcal{N} \cup \mathcal{X})$ the set of terms built on $\mathcal{F}$ and $\mathcal{N} \cup \mathcal{X}$. Of course function symbol application must respect sorts and arities.

Example 1. Let $\mathcal{F} = \{\mathsf{aenc}/2,\ \mathsf{adec}/2,\ \mathsf{pk}/1,\ \langle\ \rangle/2,\ \mathsf{proj}_1/1,\ \mathsf{proj}_2/1\}$ be a signature containing function symbols for asymmetric encryption, decryption and pairing, each of arity 2, as well as projection symbols and the function symbol pk, each of arity 1. The term $\mathsf{pk}(sk)$ represents the public counterpart of the private key sk.

In the applied pi calculus, one has *plain processes*, denoted P, Q, R and *extended processes*, denoted by A, B, C. Plain processes are built up in a similar way to processes in pi calculus except that messages can contain terms rather than just names. Extended processes add *active substitutions* and restriction on variables. In the grammar described below, M and N are terms, n is a name, x a variable and u is a *metavariable*, standing either for a name or a variable.

$P, Q, R :=$	plain processes	$A, B, C :=$	extended processes
0		P	
$P \mid Q$		$A \mid B$	
$!P$		$\texttt{new}\ n.A$	
$\texttt{new}\ n.P$		$\texttt{new}\ x.A$	
$\texttt{if}\ M = N\ \texttt{then}\ P\ \texttt{else}\ Q$		$\{^M/_x\}$	
$\texttt{in}(u, x).P$			
$\texttt{out}(u, N).P$			

$\{^M/_x\}$ is the active substitution that replaces the variable x with the term M. Active substitutions generalise the "let" construct: $\texttt{new}\ x.(\{^M/_x\} \mid P)$ corresponds exactly to "let $x = M$ in P". As usual, names and variables have scopes, which are delimited by restrictions and by inputs. We write $fv(A)$, $bv(A)$, $fn(A)$ and $bn(A)$ for the sets of *free* and *bound variables* and *free* and *bound names* of A, respectively. We say that an extended process is *closed* if all its variables

are either bound or defined by an active substitution. An *evaluation context* $C[\_]$ is an extended process with a hole instead of an extended process.

Active substitutions are useful because they allow us to map an extended process A to its *frame*, denoted $\phi(A)$, by replacing every plain process in A with 0. A frame is an extended process built up from 0 and active substitutions by parallel composition and restriction. The frame $\phi(A)$ accounts for the set of terms statically possessed by the intruder (but does not take into account for A's dynamic behaviour). The *domain* of a frame φ, denoted by $dom(\varphi)$, is the set of variables for which φ defines a substitution (those variables x for which φ contains a substitution $\{^M/_x\}$ not under a restriction on x).

Example 2. Consider the signature $\mathcal{F}$ of Example 1. Let A be the following process made up of three components in parallel:

$$A = \mathtt{new}\ s, sk, x_1.(\mathtt{out}(c_1, x_1) \mid \mathtt{in}(c_1, y).\mathtt{out}(c_2, \mathsf{adec}(y, sk)) \mid \{^{\mathsf{aenc}(s,\mathsf{pk}(sk))}/_{x_1}\}).$$

The first component publishes the message $\mathsf{aenc}(s, \mathsf{pk}(sk))$ stored in x_1 by sending it on c_1. The second receives a message on c_1, uses the secret key sk to decrypt it, and forwards the result on c_2. We have $\phi(A) = \mathtt{new}\ s, sk, x_1.\{^{\mathsf{aenc}(s,\mathsf{pk}(sk))}/_{x_1}\}$ and $dom(\phi(A)) = \emptyset$ (since x_1 is under a restriction).

2.2 Semantics

We briefly recall the operational semantics of the applied pi calculus (see [1] for details). First, we associate an *equational theory* E to the signature $\mathcal{F}$. The equational theory is defined by a set of equations of the form $M = N$ where $M, N \in \mathcal{T}(\mathcal{F}, \mathcal{X})$, and induces an equivalence relation over terms: $=_{\mathsf{E}}$ is the smallest congruence relation on terms, which contains all equations $M = N$ in E, and that is closed under substitution of terms for variables.

Example 3. Considering the signature $\mathcal{F}$ of Example 1 we define the equational theory $\mathsf{E}_{\mathsf{aenc}}$ by the equations $\mathsf{adec}(\mathsf{aenc}(x, \mathsf{pk}(y)), y) = x$ and $\mathsf{proj}_i(\langle x_1, x_2\rangle) = x_i$ for $i \in \{1, 2\}$.

Structural equivalence, noted $\equiv$, is the smallest equivalence relation on extended processes that is closed under α-conversion on names and variables, by application of evaluation contexts, and satisfying some further basic structural rules such as $A \mid 0 \equiv A$, associativity and commutativity of $\mid$, binding-operator-like behaviour of $\mathtt{new}$, and when $M =_{\mathsf{E}} N$ the equivalences

$$\mathtt{new}\ x.\{^M/_x\} \equiv 0 \qquad \{^M/_x\} \equiv \{^N/_x\} \qquad \{^M/_x\} \mid A \equiv \{^M/_x\} \mid A\{^M/_x\}$$

Example 4. Consider the following process:

$$P = \mathtt{new}\ s, sk.(\mathtt{out}(c_1, \mathsf{aenc}(s, \mathsf{pk}(sk))) \mid \mathtt{in}(c_1, y).\mathtt{out}(c_2, \mathsf{adec}(y, sk))).$$

The process P is structurally equivalent to the process A given in Example 2. We have also that $\phi(P) = 0 \equiv \phi(A)$.

The operational semantics of processes in the applied pi calculus is defined by rules defining two relations: *structural equivalence* (described above) and *internal reduction*, noted $\xrightarrow{\tau}$. Internal reduction $\xrightarrow{\tau}$ is the smallest relation on extended processes closed under structural equivalence and application of evaluation contexts such that:

$$\mathtt{out}(a,x).P \mid \mathtt{in}(a,x).Q \xrightarrow{\tau} P \mid Q \qquad \mathtt{if}\ M = M\ \mathtt{then}\ P\ \mathtt{else}\ Q \xrightarrow{\tau} P$$

$$\mathtt{if}\ M = N\ \mathtt{then}\ P\ \mathtt{else}\ Q \xrightarrow{\tau} Q \text{where } M, N \text{ are ground terms and } M \neq_{\mathtt{E}} N$$

The operational semantics is extended by a *labelled* operational semantics enabling us to reason about processes that interact with their environment. Labelled operational semantics defines the relation $\xrightarrow{\ell}$ where ℓ is either an input or an output. We adopt the following rules in addition to the internal reduction rules. Below, a is a channel name, u is a metavariable, and x is a variable of base type.

$$\textsc{In} \quad \mathtt{in}(a,x).P \xrightarrow{in(a,M)} P\{^{M}/_{x}\} \qquad \textsc{Scope} \quad \frac{A \xrightarrow{\ell} A' \quad u \text{ does not occur in } \ell}{\mathtt{new}\ u.A \xrightarrow{\ell} \mathtt{new}\ u.A'}$$

$$\textsc{Out-Atom} \quad \mathtt{out}(a,u).P \xrightarrow{out(a,u)} P \qquad \textsc{Par} \quad \frac{bn(\ell) \cap fn(B) = \emptyset \quad A \xrightarrow{\ell} A' \quad bv(\ell) \cap fv(B) = \emptyset}{A \mid B \xrightarrow{\ell} A' \mid B}$$

$$\textsc{Open-Atom} \quad \frac{A \xrightarrow{out(a,u)} A' \qquad u \neq a}{\mathtt{new}\ u.A \xrightarrow{\mathtt{new}\ u.out(a,u)} A'} \qquad \textsc{Struct} \quad \frac{A \equiv B \quad B \xrightarrow{\ell} B' \quad A' \equiv B'}{A \xrightarrow{\ell} A'}$$

Note that the labelled transition is not closed under application of evaluation contexts. Moreover the output of a term M needs to be made "by reference" using a restricted variable and an active substitution.

2.3 Equivalences

Let $\mathcal{A}$ be the alphabet of actions (in our case this alphabet is infinite) where the special symbol $\tau \in \mathcal{A}$ represents an unobservable action. For every $\alpha \in \mathcal{A}$ the relation $\xrightarrow{\alpha}$ has been defined in Section 2.2. We consider the relation $\overset{\alpha}{\mapsto}$ that is the restriction of $\xrightarrow{\alpha}$ on closed extended processes. For every $w \in \mathcal{A}^*$ the relation $\overset{w}{\mapsto}$ on closed extended processes is defined in the usual way. By convention $A \overset{\epsilon}{\mapsto} A$ where ϵ denotes the empty word. For every $s \in (\mathcal{A} \setminus \{\tau\})^*$, the relation $\overset{s}{\Mapsto}$ on extended processes is defined by: $A \overset{s}{\Mapsto} B$ if, and only if, there exists $w \in \mathcal{A}^*$ such that $A \overset{w}{\mapsto} B$ and s is obtained from w by erasing all occurrences of τ. Intuitively, $A \overset{s}{\Mapsto} B$ means that A transforms into B by experiment s.

Intuitively, two processes are *equivalent* if they cannot be distinguished by any active attacker represented by any context. Equivalences can be used to formalise many interesting privacy related properties. The universal quantification

over contexts makes equivalences difficult to verify. Hence, an alternative characterization, namely *trace equivalence*, is often used. This characterization, recalled in Definition 2, relies on a direct comparison of labelled transitions rather than on contexts. First, we introduce a notion of intruder's knowledge that has been extensively studied.

Definition 1 (static equivalence $\sim$). *Two terms M and N are* equal in the frame ϕ, *and we write $(M =_{\mathsf{E}} N)\phi$, if there exists $\tilde{n}$ and a substitution σ such that $\phi \equiv \mathtt{new}\, \tilde{n}.\sigma$, $\tilde{n} \cap (fn(M) \cup fn(N)) = \emptyset$, and $M\sigma =_{\mathsf{E}} N\sigma$. Two closed frames ϕ_1 and ϕ_2 are* statically equivalent, *$\phi_1 \sim_{\mathsf{E}} \phi_2$ (or simply $\phi_1 \sim \phi_2$ when E is clear from the context), when*

- *$dom(\phi_1) = dom(\phi_2)$, and*
- *for all terms M, N we have that $(M =_{\mathsf{E}} N)\phi_1$ if and only if $(M =_{\mathsf{E}} N)\phi_2$.*

Example 5. Consider $\phi_0 = \{^{\mathsf{aenc}(s_0,\mathsf{pk}(sk))}/_{x_1}\}$ and $\phi_1 = \{^{\mathsf{aenc}(s_1,\mathsf{pk}(sk))}/_{x_1}\}$. We have $(\mathsf{adec}(x_1, sk) =_{\mathsf{E}_{\mathsf{aenc}}} s_0)\phi_0$ whereas $(\mathsf{adec}(x_1, sk) \neq_{\mathsf{E}_{\mathsf{aenc}}} s_0)\phi_1$, thus $\phi_0 \not\sim \phi_1$. However, we have that $\mathtt{new}\ sk.\phi_0 \sim \mathtt{new}\ sk.\phi_1$. This is a non trivial equivalence. Intuitively, there is no test that allows one to distinguish the two frames since neither the decryption key, nor the encryption key are available.

For every closed extended process A, we define its set of traces, each trace consisting in a sequence of actions together with the sequence of sent messages

$$\mathsf{trace}(A) = \{(s, \phi(B)) \mid A \overset{s}{\Rightarrow} B \text{ for some closed extended process } B\}.$$

Definition 2 (trace equivalence $\approx$). *Let A and B be two closed extended processes, A and B are* trace equivalent, *denoted by $A \approx B$, if if for every $(s, \varphi) \in \mathsf{trace}(A)$ such that $bn(s) \cap fn(B) = \emptyset$, there exists $(s', \varphi') \in \mathsf{trace}(B)$ such that $s = s'$ and $\varphi \sim \varphi'$ (and conversely).*

Example 6. Consider the theory $\mathsf{E}_{\mathsf{aenc}}$, the processes $P_0 = \mathtt{out}(c, \mathsf{aenc}(s_0, \mathsf{pk}(sk)))$ and $Q_0 = \mathtt{out}(c, \mathsf{aenc}(s_1, \mathsf{pk}(sk)))$. We have $\mathtt{new}\ sk.P_0 \approx_\ell \mathtt{new}\ sk.Q_0$ whereas $P_0 \not\approx_\ell Q_0$. These results are direct consequences of the static (in)equivalence relations stated in Example 5.

In the remaning of the paper, we illustrate how privacy-type security properties can be expressed by the means of equivalences.

3 Application 1: electronic voting protocols

We report below on some of our efforts in using the equivalences of the applied pi calculus to model privacy-type properties of electronic elections [5, 6], and more recently [3] in which the notion of everlasting privacy is studied. Those properties can be expressed informally as follows:

- *Vote-privacy*: the fact that a particular voter voted in a particular way is not revealed to anyone.

– *Everlasting vote-privacy*: the fact that a particular voter voted in a particular way will be not revealed in 20 years (even if the strength of the cryptographic primitives that are used is eroded with the passage of time).
– *Receipt-freeness*: a voter does not gain any information (a *receipt*) which can be used to prove to a coercer that she voted in a certain way.
– *Coercion-resistance*: a voter cannot cooperate with a coercer to prove to him that she voted in a certain way.

3.1 Vote-privacy

The privacy property aims to guarantee that the link between a given voter V and his vote v remains hidden. A classical device for modelling anonymity is to ask whether two processes, one in which V_A votes and one in which V_B votes, are equivalent. However, such an equivalence does not hold here as the voters' identities are revealed (and they need to be revealed at least to the administrator to verify eligibility). In a similar way, an equivalence of two processes where only the vote is changed does not hold, because the votes are published at the end of the protocol. To ensure privacy we need to hide the *link* between the voter and the vote and not the voter or the vote itself.

In order to give a reasonable definition of privacy, we suppose that at least two voters are honest. We denote the voters V_A and V_B and their votes a and b. We say that a voting protocol respects privacy whenever a process where V_A votes a and V_B votes b is equivalent to a process where V_A votes b and V_B votes a. Formally, we have defined privacy as follows:

Definition 3 (vote-privacy). *[5] A voting protocol respects* vote-privacy *if*

$$S[V_A\{^a/_v\} \mid V_B\{^b/_v\}] \;\approx\; S[V_A\{^b/_v\} \mid V_B\{^a/_v\}]$$

for all possible votes a and b.

Note that this definition is robust even in situations where the result of the election is such that the votes of V_A and V_B are necessarily revealed *e.g.* if the vote is unanimous, or if all other voters reveal how they voted. In some protocols the vote-privacy property may hold even if authorities are corrupt, while other protocols may require the authorities to be honest. When proving privacy, we choose which authorities we want to model as honest, by including them in the context S.

3.2 Everlasting vote-privacy

Here, it is important to model that an attacker may interact with a protocol today and store some data which may be exploited in the future when his computational power has increased. The fact that the attacker's power may change will be modeled using two different equational theories:

– E_0 models the attacker's capabilities while interacting with the protocol;

– E_1 models his capabilities when exploiting the published data in the future.

It is also assumed that the attacker does not store all the data that was sent on the network. We will consider some channels $\mathcal{C}$ to be *everlasting*: data sent over such channels will remain in the attacker's knowledge for future analysis while other data will be "forgotten" and can only be used during the interaction with the protocol.

Two processes A and B are said to be *forward indistinguishable* (denoted with $\approx_{\mathsf{fwd}}$) if, informally, an attacker cannot observe the difference between A and B given the computational power modeled by E_1, but for executions that happened in the past, that is over E_0 and observing only the information that was passed through everlasting channels.

Definition 4 (everlasting vote-privacy). *[3] A voting protocol respects* everlating vote-privacy *w.r.t. a set of channels $\mathcal{C}$ and equational theories E_0 and E_1, if for all possible votes a and b, we have:*

1. *vote-privacy w.r.t. E_0, and*
2. *forward indistinguishability w.r.t. $\mathcal{C}$ and equational theories E_0 and E_1 for the following processes:*

$$S[V_A\{^a/_v\} \mid V_B\{^b/_v\}] \approx_{\mathsf{fwd}} S[V_A\{^b/_v\} \mid V_B\{^a/_v\}]$$

Note that everlasting vote-privacy is stricly stronger than vote-privacy. When $\mathcal{C}$ corresponds to all channels, we typically get a requirement which is too strong for practical purposes.

3.3 Receipt-freeness

Similarly to privacy, receipt-freeness may be formalised as an equivalence. However, we need to model the fact that V_A is willing to provide secret information, *i.e.* the receipt, to the coercer. We assume that the coercer is in fact the attacker who, as usual in the Dolev-Yao model, controls the public channels. To model V_A's communication with the coercer, we consider that V_A executes a voting process V_A^{ch} which has been modified: inputs and freshly generated names of base type (*i.e.* not channel type) are forwarded to the coercer on the channel ch. We do not forward restricted channel names, as these are used for modelling purposes, such as physically secure channels, *e.g.* the voting booth, or the existence of a PKI which securely distributes keys (the keys are forwarded but not the secret channel name on which the keys are received). The process $A^{\setminus out(ch,\cdot)}$ is as the process A, but hiding the outputs on the channel ch.

Intuitively, a protocol is receipt-free if, for all voters V_A, the process in which V_A votes according to the intruder's wishes is indistinguishable from the one in which she votes something else. As in the case of privacy, we express this as an equivalence between two processes. Suppose the coercer's desired vote is c. Then we define receipt-freeness as follows:

Definition 5 (Receipt-freeness). *[5] A voting protocol is* receipt-free *if there exists a closed plain process V' such that*

- $V'\backslash out(chc,\cdot) \approx V_A\{^a/_v\}$,
- $S[V_A\{^c/_v\}^{chc} \mid V_B\{^a/_v\}] \approx S[V' \mid V_B\{^c/_v\}]$,

for all possible votes a and c.

As before, the context S in the second equivalence includes those authorities that are assumed to be honest. V' is a process in which voter V_A votes a but communicates with the coercer C in order to feign cooperation with him. Thus, the second equivalence says that the coercer cannot tell the difference between a situation in which V_A genuinely cooperates with him in order to cast the vote c and one in which she pretends to cooperate but actually casts the vote a, provided there is some counterbalancing voter that votes the other way around. The first equivalence of the definition says that if one ignores the outputs V' makes on the coercer channel *chc*, then V' looks like a voter process V_A voting a.

The first equivalence of the definition may be considered too strong. Informally, one might consider that the equivalence should be required only in a particular S context rather than requiring it in any context (with access to all the private channels of the protocol). This would result in a weaker definition, although one which is more difficult to work with. In fact, the variant definition would be only slightly weaker. It is hard to construct a natural example which distinguishes the two possibilities, and in particular it makes no difference to the case studies we have performed. Therefore, we prefer to stick to Definition 5.

3.4 Coercion-resistance

Coercion-resistance is the third and strongest of the three privacy properties. Again, it says that the link between a voter and her vote cannot be established by an attacker, this time even if the voter cooperates with the attacker during the election process. Such cooperation can include giving to the attacker any data which the voter gets during the voting process, and using data which the attacker provides in return. When analysing coercion-resistance, we assume that the voter and the attacker can communicate and exchange data at any time during the election process. Coercion-resistance is intuitively stronger than receipt-freeness, since the attacker has more capabilities. The definition is more involved and can be found in [5].

We have proved the intuitive relationships between these properties:

Proposition 1. *[5, 3] Let V be a voting protocol.*

1. *If V is coercion-resistant (for a given set of honest authorities), then it also respects receipt-freeness (for the same set);*
2. *If V is receipt-free (for a given set of honest authorities), then it also respects privacy (for the same set);*

3. *If V respects everlasting privacy (for a given set of honest authorities), then it also respects privacy (for the same set).*

The definitions of privacy and receipt-freeness described above have also been reused and adapted to model privacy and receipt-freeness in on-line auction systems [7, 8].

4 Application 2: RFID protocols

We report below on some efforts in using the equivalences of the applied pi calculus to model privacy-type properties of RFID protocols. We present below, on a simple example, the definitions that have been proposed by M. Arapinis *et al.* in [2]. Those defintions have then been used to analyse the Basic Access Control protocol used in electronic passports.

The properties we consider here can be expressed informally as follows:

- *anonymity*: the fact that a user may use a service or a resource without disclosing the user's identity.
- *unlinkability*: the fact that a user may make multiple uses of a service or a resource without others being able to link these uses together.

For sake of simplicity, we only present a strong version of each property so that the properties can be expressed without modifying the processes under study. Otherwise, it will be necessary to annotate protocols in such a way that in any trace each observable transition will be accompanied with information on its initiator. We consider the following example:

$$P : A \to S : \{A\}^r_{\mathsf{pk}(S)}$$

In protocol P, the agent A simply identifies himself to the server S by sending him his identity encrypted under S's public key (using a probabilistic encryption scheme).

4.1 Anonymity

Anonymity is informally defined by the ISO/IEC standard 15408 as the property ensuring that *a user may use a service or a resource without disclosing the user's identity.* Formally, strong anonymity has been defined to hold [2] when an outside observer cannot tell the difference between a system in which the user with a public known identity id_0 executes the analysed protocol, from the system where id_0 is not present at all.

Following this formal definition of anonymity, the protocol

$$P = \mathtt{new}\ r.\mathtt{out}(c, \mathsf{aenc}(\langle r, id\rangle, \mathsf{pk}(sk_S)))$$

is said to satisfy strong anonymity if the following equivalence holds:

$$\begin{array}{c} \texttt{new}\ sk_S.\ ((!\texttt{new}\ id.\ !P) \mid !P\{id_0/id\}) \\ \approx \\ \texttt{new}\ sk_S.(!\texttt{new}\ id.\ !P) \end{array}$$

In other words, anonymity is satisfied if an observer cannot tell if the user id_0 (known to the attacker) has been executing the protocol P or not.

4.2 Unlinkability

Unlinkability is informally defined by the ISO/IEC standard 15408 as the property ensuring that *a user may make multiple uses of a service or a resource without others being able to link these uses together*. Formally, strong unlinkability has been defined to hold [2] when a system in which the analysed protocol can be executed by each user multiple times looks the same to an outside observer that the system in which the analysed protocol can be executed by each user at most once.

Again, we can formalise this property for the protocol P using an equivalence:

$$\texttt{new}\ sk_S.\ (!\texttt{new}\ id.\ !P) \ \approx\ \texttt{new}\ sk_S.(!\texttt{new}\ id.\ P)$$

In other words, unlinkability is satisfied if an observer cannot tell if the users can execute multiple or at most once the protocol P.

5 Application 3: routing protocols

We report below on some efforts in using the equivalences of the applied pi calculus to model privacy-type properties of mobile ad-hoc network routing protocols. The definitions that have been proposed in [4] and used to analyse the ANODR routing protocol used for anonymous routing. In order to be expressed, we enrich our initial model to take into account the topology of the network and the traffic necessary to ensure the privacy properties to hold. These properties can be informally expressed as follows:

- *Indistinguishability:* the fact that a particular action, typically done by the source or the destination of a session of the protocol, can always be executed by another role of the protocol, and hence conceiling one's role in this session.
- *Unlinkability:* the inability for an attacker to link togother two messages of a same session of the protocol.
- *Anonimity:* the fact that the source or the destination of a route is never revealed to anyone.

5.1 Enriching the model

To analyse privacy properties of routing protocols for mobile ad-hoc networks, we focus only on the case of a passive attacker, *i.e.* an intruder who can listen to

some communications and hence learn information about the participants and the particular session of the protocol they are running. Because wireless communication has a limited range and the agents in a wireless routing protocol do not know *a priori* the topology of the network, we parametrize our definitions by a *topology*: an undirected graph whose nodes represent the agents of the protocol and edges indicate a neighbour relation between two agents. We moreover identify a subset of nodes, the *malicious nodes*, where the passive attacker is indeed able to listen to emitted messages. This set can be the entire network if needed.

The usual semantics of the applied pi-calculus need to be adapted to take into account the localization of a process and whether a communication between two nodes can happen. Dealing with both a passive attacker and localized process enables us to define a communication rule stating that two nodes can exchange a message if they are neighbours; and the attacker learns the message if the outputting node is malicious. A more detailed presentation of these semantics can be found in [4].

Privacy definition also rely on a number of information absent for the usual labeled transition systems, and thus unknown to the attacker, such as the role which executed a particular action, its location, the session of the protocols it belongs, the source and the intended destination of this instance of the protocol. This added information is introduced as *annotations* in the transition system to use them when specifying privacy properties.

As already illustrated in the previous sections, privacy-type properties are often formalised using a notion of equivalence. Here, we consider the notion of *equivalence between two traces*.

Definition 6 (equivalence of two traces). *Let* $\mathsf{tr}_1 = K_1 \overset{s_1}{\Rightarrow} (\mathcal{E}_1; \mathcal{P}_1; \mathcal{S}_1; \sigma_1)$ *and* $\mathsf{tr}_2 = K_2 \overset{s_2}{\Rightarrow} (\mathcal{E}_2; \mathcal{P}_2; \mathcal{S}_2; \sigma_2)$ *be two traces. They are* equivalent*, denoted* $\mathsf{tr}_1 \approx_{\mathsf{E}} \mathsf{tr}_2$*, if* $s_1 = s_2$ *and* $\mathit{new}\, \mathcal{E}_1.\sigma_1 \sim_{\mathsf{E}} \mathit{new}\, \mathcal{E}_2.\sigma_2$*.*

Finally, one cannot hope to achieve privacy on a mobile ad-hoc network without some traffic to create some confusion for the attacker. This added traffic is modelised through the notion of the *extension* of a trace: given a trace tr, we say that tr^+ is an *extension* of tr, denoted $\mathsf{tr} \preccurlyeq \mathsf{tr}^+$, if actions of tr are present in tr^+ up to some re-indexing, as well as some other actions, from other sessions of the protocol.

Given an indice i corresponding to an action in tr ($1 \leq i \leq \ell$), we denote by $\mathsf{ind}_i(\mathsf{tr}, \mathsf{tr}^+)$ the indice of the corresponding action in tr^+, i.e. $\mathsf{ind}_i(\mathsf{tr}, \mathsf{tr}^+) = k_i$.

5.2 Indistinguishability

Indistinguishability deals with the ability for a participant of the protocol to conceil the role he played in some executation. Formally, a routing protocol is broken into a set of roles, typically one the source, the destination, and two additional roles for the forwarding nodes during the request and reply phase. We say a protocol preserves the indistinguishability of a set of roles Roles if for any execution with enough traffic of the protocol where some action is executed

by a role in Roles there exists an equivalent execution where the said action is realized by another role, not in Roles.

Definition 7 (indistinguishability). *Let K_0 be an initial configuration associated to a routing protocol and a topology, and* Roles *be a set of roles. We say that K_0 preserves* indistinguishability w.r.t. Roles *if for any annotated trace* tr

$$\mathsf{tr} = K_0 \xrightarrow[A_1,\mathsf{R}_1]{\ell_1} K_1 \xrightarrow[A_2,\mathsf{R}_2]{\ell_2} \dots \xrightarrow[A_n,\mathsf{R}_n]{\ell_n} K_n$$

*and for any $i \in \{1,\dots,n\}$ such that $\mathsf{R}_i \in \mathsf{Roles}$ and $\ell_i \neq \tau$ (*i.e. *ℓ_i is an action observed by the attacker), there exist two annotated traces tr^+ and tr' such that:* $\mathsf{tr} \preccurlyeq \mathsf{tr}^+$, $\mathsf{tr}^+ \approx \mathsf{tr}'$, *and $\mathsf{R}'_{\mathsf{ind}_i(\mathsf{tr},\mathsf{tr}^+)} \notin \mathsf{Roles}$ where*

$$\mathsf{tr}' = K'_0 \xrightarrow[A'_1,\mathsf{R}'_1]{\ell'_1} K'_1 \xrightarrow[A'_2,\mathsf{R}'_2]{\ell'_2} K'_2 \dots \xrightarrow[A'_{n'},\mathsf{R}'_{n'}]{\ell'_{n'}} K'_{n'}.$$

The trace tr^+ enables us to deal with the aforementioned traffic needed to aim at preserving indistinguishability, and $\mathsf{ind}_i(\mathsf{tr},\mathsf{tr}^+)$ enables us to link actions for the original trace to its extension.

5.3 Unlinkability

Unlinkability relates to the possibility for the attacker to link together two messages belonging to the same session of the routing protocol. A session is defined as a (possibly partial) execution of the protocol initiated by a source to an intended destion, through a number of relaying nodes. To provide unlinkability, a protocol has to ensure that the relaying is done in a way which hide the link between incoming and outcoming messages. As with indistinguishability, to express unlinkability, we need to rely on some traffic and hence use the notion of extensions of traces. This definition is moreover parametrized by two sets of roles: the attacker should not be able to link a message emitted by an agent playing a role in the first set to another message emitted by a role in the second set.

Definition 8 (unlinkability). *Let K_0 be an initial configuration associated to a routing protocol and a topology, and* Roles_1, Roles_2 *be two sets of roles. We say that K_0 preserves* unlinkability w.r.t. $\mathsf{Roles}_1/\mathsf{Roles}_2$ *if for any annotated trace* tr

$$\mathsf{tr} = K_0 \xrightarrow[A_1,\mathsf{R}_1,[sid_1,S_1,D_1]]{\ell_1} K_1 \xrightarrow[A_2,\mathsf{R}_2,[sid_2,S_2,D_2]]{\ell_2} \dots \xrightarrow[A_n,\mathsf{R}_n,[sid_n,S_n,D_n]]{\ell_n} K_n$$

*and for any $i,j \in \{1,\dots,n\}$ such that $\mathsf{R}_i \in \mathsf{Roles}_1$, $\mathsf{R}_j \in \mathsf{Roles}_2$, $sid_i = sid_j$, and $\ell_i, \ell_j \neq \tau$ (*i.e. *ℓ_i, ℓ_j are actions observed by the attacker), there exist two annotated traces tr^+ and tr' such that:* $\mathsf{tr} \preccurlyeq \mathsf{tr}^+$, $\mathsf{tr}^+ \approx \mathsf{tr}'$, *and $sid'_{\mathsf{ind}_i(\mathsf{tr},\mathsf{tr}^+)} \neq sid'_{\mathsf{ind}_j(\mathsf{tr},\mathsf{tr}^+)}$ where*

$$\mathsf{tr}' = K'_0 \xrightarrow[A'_1,\mathsf{R}'_1,[sid'_1,S'_1,D'_1]]{\ell'_1} K'_1 \xrightarrow[A'_2,\mathsf{R}'_2,[sid'_2,S'_2,D'_2]]{\ell'_2} \dots \xrightarrow[A'_{n'},\mathsf{R}'_{n'},sid'_{n'},S'_{n'},D'_{n'}]{\ell'_{n'}} K'_{n'}.$$

Note that unlinkability is a distinct notion from indistinguishability as it is possible to design protocols satisfying one and note the other for various sets of role.

5.4 Anonymity

Finally, anonymity deals more traditionally with the fact that the source or the destination of a session of the routing protocol remains hidden to anyone but the source and destination of that said session. Once again, enough traffic is needed to achieve such a property.

Definition 9 (anonymity). *Let K_0 be an initial configuration associated to a routing protocol and a topology. We say that K_0 preserves source anonymity (resp. destination anonymity) if for any annotated trace* tr

$$\mathsf{tr} = K_0 \xrightarrow[A_1,\mathsf{R}_1,[sid_1,S_1,D_1]]{\ell_1} K_1 \xrightarrow[A_2,\mathsf{R}_2,[sid_2,S_2,D_2]]{\ell_2} \dots \xrightarrow[A_n,\mathsf{R}_n,[sid_n,S_n,D_n]]{\ell_n} K_n$$

*and for any $i \in \{1,\dots,n\}$ such that $\ell_i \neq \tau$ (*i.e. *ℓ_i is an action observed by the attacker), there exist two annotated traces tr^+ and tr' such that $\mathsf{tr} \preccurlyeq \mathsf{tr}^+$, $\mathsf{tr}^+ \approx \mathsf{tr}'$, and $S'_{\mathsf{ind}_i(\mathsf{tr},\mathsf{tr}^+)} \neq S_i$ (resp. $D'_{\mathsf{ind}_i(\mathsf{tr},\mathsf{tr}^+)} \neq D_i$) where*

$$\mathsf{tr}' = K'_0 \xrightarrow[A'_1,\mathsf{R}'_1,[sid'_1,S'_1,D'_1]]{\ell'_1} K'_1 \xrightarrow[A'_2,\mathsf{R}'_2,[sid'_2,S'_2,D'_2]]{\ell'_2} \dots \xrightarrow[A'_{n'},\mathsf{R}'_{n'},[sid'_{n'},S'_{n'},D'_{n'}]]{\ell'_{n'}} K'_{n'}$$

Anonymity is distinct from the notions of indistinguishability or unlinkability, as examples in [4] demonstrate. Nethertheless, for reasonable protocol, one could prove that source (resp. destination) anonymity is a stronger notion than indistinguishability for the corresponding roles.

Proposition 2. *Let K_0 be an initial configuration associated to a routing protocol and a topology. If K_0 preserves source (resp. destination) anonymity, then it preserves indistinguishability w.r.t. the set of roles acting as a source (resp. destination).*

6 Conclusion and perspectives

In this report, we gave a brief description of the applied pi calculus [1] which has been used by many researchers to model and analyse security protocols in a variety of areas. In such a calculus, the properties of the cryptographic primitives are modelled by means of an equational theory. This leads to a flexible calculus suitable to formalise a lot of security protocols. This calculus is however not sufficient for instance to capture some features needed to model routing protocols, but it can be easily extended to allow one to model new applications. Actually, this calculus is particularly suitable to model privacy-type properties that are encountered in many applications. We present some of them in this report.

References

1. M. Abadi and C. Fournet. Mobile values, new names, and secure communication. In *Proceedings of the 28th ACM Symposium on Principles of Programming Languages (POPL'01)*, pages 104–115, London (United Kingdom), 2001. ACM Press.
2. M. Arapinis, T. Chothia, E. Ritter, and M. Ryan. Analysing unlinkability and anonymity using the applied pi calculus. In *Proceedings of the 23rd IEEE Computer Security Foundations Symposium (CSF'10)*, pages 107–121. IEEE Computer Society Press, 2010.
3. M. Arapinis, V. Cortier, S. Kremer, and M. D. Ryan. Practical Everlasting Privacy. In D. Basin and J. Mitchell, editors, *Proceedings of the 2nd Conferences on Principles of Security and Trust (POST'13)*, Lecture Notes in Computer Science, Rome, Italy, Mar. 2013. Springer. To appear.
4. R. Chrétien and S. Delaune. Formal analysis of privacy for routing protocols in mobile ad hoc networks. In D. Basin and J. Mitchell, editors, *Proceedings of the 2nd International Conference on Principles of Security and Trust (POST'13*, Lecture Notes in Computer Science, Roma, Italy, Mar. 2013. Springer. To appear.
5. S. Delaune, S. Kremer, and M. D. Ryan. Verifying privacy-type properties of electronic voting protocols. *Journal of Computer Security*, 17(4):435–487, July 2009.
6. S. Delaune, S. Kremer, and M. D. Ryan. Verifying privacy-type properties of electronic voting protocols: A taster. In D. Chaum, M. Jakobsson, R. L. Rivest, P. Y. A. Ryan, J. Benaloh, M. Kutyłowski, and B. Adida, editors, *Towards Trustworthy Elections – New Directions in Electronic Voting*, volume 6000 of *Lecture Notes in Computer Science*, pages 289–309. Springer, May 2010.
7. N. Dong, H. Jonker, and J. Pang. Analysis of a receipt-free auction protocol in the applied pi calculus. In *Proceedings of the 7th International Workshop on Formal Aspects in Security and Trust (FAST'10)*, volume 6561 of *Lecture Notes in Computer Science*, pages 223–238, Pisa, Italy, 2010. Springer.
8. J. Dreier, P. Lafourcade, and Y. Lakhnech. Formal verification of e-auction protocols. In D. Basin and J. Mitchell, editors, *Proceedings of the 2nd Conferences on Principles of Security and Trust (POST'13)*, Lecture Notes in Computer Science, Rome, Italy, Mar. 2013. Springer. To appear.

Practical everlasting privacy

Myrto Arapinis[1], Véronique Cortier[2], Steve Kremer[2], and Mark Ryan[1]

[1] School of Computer Science, University of Birmingham
[2] LORIA, CNRS, France

Abstract. Will my vote remain secret in 20 years? This is a natural question in the context of electronic voting, where encrypted votes may be published on a bulletin board for verifiability purposes, but the strength of the encryption is eroded with the passage of time. The question has been addressed through a property referred to as *everlasting privacy*. Perfect everlasting privacy may be difficult or even impossible to achieve, in particular in remote electronic elections. In this paper, we propose a definition of *practical everlasting privacy*. The key idea is that in the future, an attacker will be more powerful in terms of computation (he may be able to break the cryptography) but less powerful in terms of the data he can operate on (transactions between a vote client and the vote server may not have been stored).
We formalize our definition of everlasting privacy in the applied-pi calculus. We provide the means to characterize what an attacker can break in the future in several cases. In particular, we model this for perfectly hiding and computationally binding primitives (or the converse), such as Pedersen commitments, and for symmetric and asymmetric encryption primitives. We adapt existing tools, in order to allow us to automatically prove everlasting privacy. As an illustration, we show that several variants of Helios (including Helios with Pedersen commitments) and a protocol by Moran and Naor achieve practical everlasting privacy, using the ProVerif and the AKiSs tools.

1 Introduction

Electronic voting schemes such as Helios [2], JCJ/Civitas [14, 8], and Prêt-à-Voter [7] aim simultaneously to guarantee *vote privacy* (that is, the link between the voter and her vote will not be revealed), and *outcome verifiability* (that is, voters and observers can check that the declared outcome is correct). A common way to achieve verifiability is to publish a "bulletin board" that contains all encrypted votes (indeed, this is the way it is done in the systems cited above). The strength and key-length of the encryption should be chosen so that decryption by an attacker is impossible for as long as the votes are expected to remain private. To prevent coercer reprisal not just to the voter but also to her descendants, one may want vote privacy for up to 100 years.

Unfortunately, however, it is not possible to predict in any reliable way how long present-day encryptions will last. Weaknesses may be found in encryption algorithms, and computers will certainly continue to get faster. A coercer can plausibly assert that a voter should follow the coercer's wishes because the bulletin board will reveal in (say) 10 years whether the voter followed the coercer's instructions. For this reason, systems with "everlasting privacy" have been introduced by [18]. These systems do not rely

on encryptions whose strength may be eroded, but on commitments that are *perfectly* or *information-theoretically hiding*. These systems have computational verifiability instead of perfect verifiability, and are considered less usable and computationally more expensive than systems relying on encryptions. More recently, schemes have been proposed with a weaker form of everlasting privacy (e.g., [10, 12]); they rely on encryptions for counting votes, but use commitments rather than encryptions for verifiability purposes. Thus, the bulletin board which only publishes the commitments does not weaken the privacy provided by the underlying scheme. Although the encrypted votes must be sent to the election administrators, it is assumed that these communications cannot be intercepted and stored *en masse*. We call this weaker form of everlasting privacy *practical everlasting privacy*.

Symbolic models for security protocol analysis have been used to model both privacy properties (e.g., [11, 3, 13]) and verifiability properties (e.g.,[16, 17]) of voting systems, but they are currently not capable of distinguishing *perfect* versus *computational* notions of privacy, or indeed, of verifiability. Our aim in this paper is to extend the model to allow these distinctions. We focus on practical everlasting privacy, and use our definitions to verify whether particular schemes satisfy that property.

Our contributions. Our first and main contribution is a general and practical definition of everlasting privacy. The key idea is that, in the future, an attacker will be more powerful in terms of computation (he may be able to break cryptography) but less powerful in terms of the data he can operate on (transactions between a vote client and the vote server may not have been stored). We therefore distinguish between standard communication channels (on which eavesdropping may be possible, but requires considerable effort) and *everlasting channels*, on which the information is intentionally published and recorded permanently (e.g. web pages that serve as a public bulletin board). Formally, we model everlasting privacy in the applied-pi calculus [1], a framework well-adapted to security protocols and already used to define privacy [11] and verifiability [16]. Our definitions apply not only to voting protocols but also to situations where forward secrecy is desirable, such as for instance untraceability in RFID protocols.

Modeling everlasting privacy also requires to precisely model what an attacker can break in the future. Our second contribution is a characterization, for several primitives, of what can be broken. The first natural primitive is encryption, for which we provide an equational theory that models the fact that private keys can be retrieved from public keys, or even from ciphertexts. Some other primitives have been primarily designed to achieve everlasting privacy. This is the case of *perfectly hiding* and *computationally binding* primitives, such as Pedersen commitments [19]. Intuitively, perfectly hiding means that the hidden secret cannot be retrieved even by a computationally unbounded adversary, while computationally binding means that, binding is ensured only for a (polynomially) limited attacker. We provide an equational theory that models such perfectly hiding and computationally binding primitives in general.

As an application, we study everlasting privacy for several variants of Helios [2], an e-voting protocol used for electing the president of the University of Louvain and board members of the IACR[3]. We study in particular its latest variants with Pedersen

[3] International Association for Cryptologic Research

commitments [12], designed to achieve everlasting privacy, still providing full verifiability. We also model and prove everlasting privacy of a (simplified) version of Moran and Naor's protocol [18]. Interestingly, we were able to adapt algorithms in existing tools to automate the verification of everlasting privacy and we use adapted versions of the AKisS [6] and ProVerif [4] tools to analyze everlasting privacy for half a dozen of protocols.

Outline. In the following section we recall the applied pi calculus and introduce notations and basic definitions. In Section 3 we define new equivalence relations, namely forward and everlasting indistinguishability. Then, in Section 4 we instantiate these equivalences to the case of voting protocols, define everlasting privacy and illustrate this property on several examples. In Section 5 we present a modeling of perfectly hiding and computationally binding (and vice-versa) primitives in the applied pi calculus. In particular we model Pedersen commitments, which are for studying two protocols that provide everlasting privacy. In Section 6 we discuss tool support for automatically proving everlasting indistinguishability before concluding.

2 The applied pi calculus

The applied pi calculus [1] is a language for modeling distributed systems and their interactions. It extends the pi calculus with an equational theory, which is particularly useful for modeling cryptographic protocols. The following subsections describe the syntax and semantics of the calculus.

2.1 Syntax

Terms. The calculus assumes an infinite set of names $\mathcal{N} = \{a, b, c, \ldots\}$, an infinite set of variables $\mathcal{V} = \{x, y, z, \ldots\}$ and a finite signature Σ, that is, a finite set of function symbols each with an associated arity. We use meta-variables u, v, w to range over both names and variables. Terms $M, N, T, \ldots$ are built by applying function symbols to names, variables and other terms. Tuples $M_1, \ldots, M_l$ are occasionally abbreviated $\tilde{M}$. We write $\{^{M_1}/_{u_1}, \ldots, ^{M_l}/_{u_l}\}$ for substitutions that replace $u_1, \ldots, u_l$ with $M_1, \ldots, M_l$. The applied pi calculus relies on a simple type system. Terms can be of sort $\mathsf{Channel}$ for channel names or Base for the payload sent out on these channels. Function symbols can only be applied to, and return, terms of sort Base. A term is ground when it does not contain variables.

The signature Σ is equipped with an equational theory E, that is a finite set of equations of the form $M = N$. We define $=_{\mathsf{E}}$ as the smallest equivalence relation on terms, that contains E and is closed under application of function symbols, substitution of terms for variables and bijective renaming of names.

Example 1. A standard signature for pairing and encryption is:

$$\Sigma_{\mathsf{enc}} = \{0, 1, \langle \_, \_ \rangle, \mathsf{fst}(\_), \mathsf{snd}(\_), \mathsf{pk}(\_), \mathsf{aenc}(\_, \_, \_), \mathsf{adec}(\_, \_), \mathsf{senc}(\_, \_, \_), \mathsf{sdec}(\_, \_)\}$$

The term $\langle m_1, m_2 \rangle$ represents the concatenation of m_1 and m_2, with associated projectors $\mathsf{fst}(\_)$ and $\mathsf{snd}(\_)$. The term $\mathsf{aenc}(k, r, m)$ represents the asymmetric encryption of

$P, Q, R ::=$	processes
0	null process
$P \mid Q$	parallel composition
$!P$	replication
$\nu n.P$	name restriction
$u(x).P$	message input
$\overline{u}\langle M\rangle.P$	message output
if $M = N$ then P else Q	conditional

$A, B, C ::=$	extended processes
P	plain process
$A \mid B$	parallel composition
$\nu n.A$	name restriction
$\nu x.A$	variable restriction
$\{^M/_x\}$	active substitution

where u is either a name or variable of channel sort.

Fig. 1. Applied pi calculus grammar

message m with public key k and randomness r while the associated decryption operator is adec. Similarly, $\mathsf{senc}(k, r, m)$ represents the symmetric encryption of message m with key k and randomness r. The associated decryption operator is sdec. The properties of these primitives are modeled by the following standard equational theory $\mathsf{E_{enc}}$:

$$\mathsf{E_{enc}} = \left\{ \begin{array}{r} \mathsf{fst}(\langle x, y\rangle) = x \\ \mathsf{snd}(\langle x, y\rangle) = y \\ \mathsf{adec}(x, \mathsf{aenc}(\mathsf{pk}(x), y, z)) = z \\ \mathsf{sdec}(x, \mathsf{senc}(x, y, z)) = z \end{array} \right\}$$

Processes. The grammar for processes is shown in Figure 1. Plain processes are standard. Extended processes introduce *active substitutions* which generalize the classical let construct: the process $\nu x.(\{^M/_x\} \mid P)$ corresponds exactly to the process $\mathrm{let}\ x = M\ \mathrm{in}\ P$. As usual names and variables have scopes which are delimited by restrictions and by inputs. All substitutions are assumed to be cycle-free.

The sets of free and bound names, respectively variables, in process A are denoted by $\mathrm{fn}(A)$, $\mathrm{bn}(A)$, $\mathrm{fv}(A)$, $\mathrm{bv}(A)$. We also write $\mathrm{fn}(M)$, $\mathrm{fv}(M)$ for the names, respectively variables, in term M. An extended process A is *closed* if it has no free variables. A *context* $C[\_]$ is an extended process with a hole. We obtain $C[A]$ as the result of filling $C[\_]$'s hole with A. An *evaluation context* is a context whose hole is not under a replication, a conditional, an input, or an output.

Example 2. Throughout the paper we illustrate our definitions with a simplified version of the Helios voting system [2]. Two techniques can be used for tallying in Helios: either a homomorphic tally based on El Gamal encryption, or a tally based on mixnets. We present here the version with mixnets.

1. The voter V computes her ballot by encrypting her vote with the public key $\mathsf{pk}(skE)$ of the election. The corresponding secret key is shared among several election authorities. Then she casts her ballot together with her identity on an authenticated channel. Upon receiving the ballot, the administrator simply publishes it on a public web page (after having checked that V is entitled to vote).
2. Once the voting phase is over, the votes are shuffled and reencrypted through mixnets. The permuted and rerandomized votes are again published on the public web page (together with a zero knowledge proof of correct reencryption and mixing).
3. Finally, the authorities decrypt the rerandomized votes and the administrator publishes the decrypted votes (with a zero knowledge proof of correct decryption).

The process representing the voter is parametrized by her vote v, and her identity id.

$$V(auth, id, v) \stackrel{\mathsf{def}}{=} \nu r.\overline{auth}\langle\langle id, \mathsf{aenc}(\mathsf{pk}(skE), r, v)\rangle\rangle$$

The administrator BB receives votes on private authenticated channels and publishes the votes. It is parametrized by the authenticated channels of the voters. Then the ballots are forwarded to the tally T over the private channel c. The tally consists in decrypting the vote. The shuffle through mixnets is modeled simply, by non deterministic parallel composition after all ballots have been received. For simplicity, we consider here an election system for three voters.

$$BB(a_1, a_2, a_3) \stackrel{\mathsf{def}}{=} \nu c.\ a_1(x).\ \overline{bb}\langle x\rangle.\ \overline{c}\langle x\rangle \mid a_2(y).\ \overline{bb}\langle y\rangle.\ \overline{c}\langle y\rangle \mid a_3(z).\ \overline{bb}\langle z\rangle.\ \overline{c}\langle z\rangle \mid T$$

$$T \stackrel{\mathsf{def}}{=} c(x').c(y').c(z').$$
$$(\overline{bb}\langle\mathsf{adec}(skE, \mathsf{snd}(x'))\rangle \mid \overline{bb}\langle\mathsf{adec}(skE, \mathsf{snd}(y'))\rangle \mid \overline{bb}\langle\mathsf{adec}(skE, \mathsf{snd}(z'))\rangle)$$

The process H then represents the whole Helios system with two honest voters and one dishonest voter (which does therefore not need to be explicitly specified and whose authenticated channel is public).

$$H \stackrel{\mathsf{def}}{=} \nu skE.\ \nu auth_1.\ \nu auth_2.$$
$$\overline{bb}\langle\mathsf{pk}(skE)\rangle.\ (V(auth_1, id_1, a) \mid V(auth_2, id_2, b) \mid BB(auth_1, auth_2, auth_3))$$

The first honest voter casts the vote a while the second honest voter casts the vote b.

2.2 Semantics

The operational semantics of the applied pi calculus is defined by the means of two relations: structural equivalence and internal reductions. *Structural equivalence* ($\equiv$) is the smallest equivalence relation closed under α-conversion of both bound names and variables and application of evaluation contexts such that:

$$A \mid 0 \equiv A \qquad \nu n.0 \equiv 0$$
$$A \mid (B \mid C) \equiv (A \mid B) \mid C \qquad \nu u.\nu w.A \equiv \nu w.\nu u.A$$
$$A \mid B \equiv B \mid A \qquad A \mid \nu u.B \equiv \nu u.(A \mid B) \text{ if } u \notin \mathrm{fn}(A) \cup \mathrm{fv}(A)$$
$$!P \equiv P \mid !P \qquad \{^M/_x\} \equiv \{^N/_x\} \text{ if } M =_{\mathsf{E}} N$$
$$\nu x.\{^M/_x\} \equiv 0$$
$$\{^M/_x\} \mid A \equiv \{^M/_x\} \mid A\{^M/_x\}$$

$$a(x).P \xrightarrow{a(M)} P\{^M/_x\} \qquad \frac{A \xrightarrow{\alpha} A' \qquad u \text{ does not occur in } \alpha}{\nu u.A \xrightarrow{\alpha} \nu u.A'}$$

$$\overline{a}\langle u\rangle.P \xrightarrow{\overline{a}\langle u\rangle} P \qquad \frac{A \xrightarrow{\alpha} A' \quad \mathrm{bv}(\alpha) \cap \mathrm{fv}(B) = \mathrm{bn}(\alpha) \cap \mathrm{fn}(B) = \emptyset}{A \mid B \xrightarrow{\alpha} A' \mid B}$$

$$\frac{A \xrightarrow{\overline{a}\langle u\rangle} A' \qquad u \neq a}{\nu u.A \xrightarrow{\nu u.\overline{a}\langle u\rangle} A'} \qquad \frac{A \equiv B \quad B \xrightarrow{\alpha} B' \quad A' \equiv B'}{A \xrightarrow{\alpha} A'}$$

Fig. 2. Labelled reductions.

Internal reduction ($\rightarrow$) is the smallest relation closed under structural equivalence, application of evaluation contexts satisfying the following rules:

COMM	$\overline{c}\langle x\rangle.P \mid c(x).Q \rightarrow P \mid Q$
THEN	if $N = N$ then P else $Q \rightarrow P$
ELSE	if $L = M$ then P else $Q \rightarrow Q$ for ground terms L, M where $L \neq_{\mathsf{E}} M$

Labelled reduction ($\xrightarrow{\alpha}$) extends the internal reduction and enables the environment to interact with the processes as defined in Figure 2. The label α is either an input, or the output of a channel name or a variable of base type.

We write $\Rightarrow$ for an arbitrary (possibly zero) number of internal reductions and $\stackrel{\alpha}{\Rightarrow}$ for $\Rightarrow\xrightarrow{\alpha}\Rightarrow$. Whenever the equational theory is not clear from the context we annotate the above relations by the equational theory and write e.g. $\rightarrow_{\mathsf{E}}$.

A *trace* of a process is the sequence of actions (i.e. labels) together with the corresponding sent messages. Formally, the set of traces of a process A is defined as follows. Note that it depends on the underlying equational theory E.

$$\mathsf{trace}_E(A) = \{(\alpha_1 \cdot \alpha_2 \cdot \ldots \cdot \alpha_n, \varphi(B)) \mid A \stackrel{\alpha_1}{\Longrightarrow}_{\mathsf{E}} A_1 \stackrel{\alpha_2}{\Longrightarrow}_{\mathsf{E}} \cdots A_{n-1} \stackrel{\alpha_n}{\Longrightarrow}_{\mathsf{E}} B\}$$

Example 3. Consider the process H representing the Helios protocol as defined in Example 2. A possible execution for H is:

$$H \xrightarrow{\nu xk.\ \overline{bb}\langle xk\rangle} H_1 \xrightarrow{\nu x.\ \overline{bb}\langle x\rangle} \xrightarrow{\nu y.\ \overline{bb}\langle y\rangle} \xrightarrow{auth_3(\langle id_3,x\rangle)} \xrightarrow{\nu z.\ \overline{bb}\langle z\rangle} \xrightarrow{\nu x'.\ \overline{bb}\langle x'\rangle} \xrightarrow{\nu y'.\ \overline{bb}\langle y'\rangle} \xrightarrow{\nu z'.\ \overline{bb}\langle z'\rangle} H_2$$

where H_1 and H_2 are defined below (we omit the other intermediate processes). Note that H_2 is simply an active substitution.

$$H_1 = \nu skE.\ \nu auth_1.\ \nu auth_2.\ \nu r_1.$$
$$(\{\mathsf{pk}(skE)/_{xk}\} \mid V(auth_1, id_1, a) \mid V(auth_2, id_2, b) \mid\ BB(auth_1, auth_2, auth_3))$$

$$H_2 = \nu skE.\ \nu auth_1.\ \nu auth_2.\ \nu r_1.\ \nu r_2.\{\mathsf{pk}(skE)/_{xk}\} \mid \{a/_{x'}, b/_{y'}, a/_{z'}\} \mid$$
$$\{\langle id_1, \mathsf{aenc}(\mathsf{pk}(skE), r_1, a)\rangle/_x, \langle id_2, \mathsf{aenc}(\mathsf{pk}(skE), r_2, b)\rangle/_y, \langle id_3, \mathsf{aenc}(\mathsf{pk}(skE), r_1, a)\rangle/_z\}$$

This execution trace corresponds to the case where the two honest voters cast their vote as expected, while the dishonest voter replays the first voter's ballot. As we shall see in Example 5, this corresponds to the attack on privacy discovered in [9].

2.3 Equivalence Relations for Processes

Privacy is often stated in terms of equivalence [11]. We recall here the definitions of static and trace equivalence.

Sequences of messages are often stored as *frames*. Formally, a frame is an extended process built from 0 and active substitutions $\{^{M}/_{x}\}$, and closed by parallel composition and restriction. The *domain* of a frame $\phi = \nu\tilde{n}.\ \{^{M_1}/_{x_1}, \ldots, ^{M_n}/_{x_n}\}$ such that $x_i \notin \tilde{n}$ is $\mathrm{dom}(\phi) = \{x_1, \ldots, x_n\}$. Every extended process A can be mapped to a frame $\varphi(A)$ by replacing every plain process in A with 0. The frame $\varphi(A)$ represents the static knowledge output by a process to its environment.

Two frames are indistinguishable to an attacker if it is impossible to build a test that allows to differentiate between the two.

Definition 1 (Static equivalence). *Given an equational theory* E *two frames* ϕ *and* ψ *are are statically equivalent, denoted* $\phi \sim_{\mathsf{E}} \psi$*, if* $\mathrm{dom}(\phi) = \mathrm{dom}(\psi)$ *and there exist* $\tilde{n}, \sigma, \tau$ *such that* $\phi \equiv \nu\tilde{n}.\sigma$*,* $\psi \equiv \nu\tilde{n}.\tau$ *and for all terms* M, N *such that* $\tilde{n} \cap (\mathrm{fn}(M) \cup \mathrm{fn}(N)) = \emptyset$*, we have* $M\sigma =_{\mathsf{E}} N\sigma$ *if and only if* $M\tau =_{\mathsf{E}} N\tau$*. By abuse of notation, we may write* $M\phi$ *instead of* $M\sigma$ *when* σ *is clear from the context.*

Example 4. Let $\mathsf{E}_{\mathsf{enc}}$ be the equational theory defined at Example 1. Let H_2 be the process/frame defined in Example 3. Let $\phi = \varphi(H_2)$ $(= H_2$ actually). Consider the following frame ψ.

$$\psi = \nu skE.\ \nu r_1.\ \nu r_2.\{^{\mathsf{pk}(skE)}/_{xk}\} \mid \{^{a}/_{x'}, ^{b}/_{y'}, ^{b}/_{z'}\} \mid$$
$$\{\langle id_1, \mathsf{aenc}(\mathsf{pk}(skE), r_1, b)\rangle/_{x}, \langle id_2, \mathsf{aenc}(\mathsf{pk}(skE), r_2, a)\rangle/_{y}, \langle id_3, \mathsf{aenc}(\mathsf{pk}(skE), r_1, b)\rangle/_{z}, \}$$

The two frames ϕ and ψ are not statically equivalent for the equational theory $\mathsf{E}_{\mathsf{enc}}$. Indeed, consider for example $M = z'$ and $N = a$, we have $M\phi = a = N\phi$ but $M\psi = b \neq N\psi$. Therefore, $\phi \not\sim_{\mathsf{E}_{\mathsf{enc}}} \psi$.

The active counterpart of static equivalence is trace equivalence. Intuitively, two processes A and B are indistinguishable to an attacker if any execution of A can be matched to an execution of B that is equal for their observable actions and such that the corresponding sequences of sent messages are statically equivalent.

Definition 2 (Trace equivalence). *Given an equational theory* E *two processes* A *and* B *are trace equivalent, denoted* $A \overset{\mathsf{tr}}{\approx}_{\mathsf{E}} B$*, if for any trace* $(tr_A, \phi_A) \in \mathsf{trace}_{\mathsf{E}}(A)$ *there is a corresponding trace* $(tr_B, \phi_B) \in \mathsf{trace}_{\mathsf{E}}(B)$ *such that* $tr_A = tr_B$ *and* $\phi_A \sim_{\mathsf{E}} \phi_B$ *(and reciprocally).*

Example 5. We consider the Helios system H' with two honest voters and one dishonest voter where one honest voter casts the vote b while the other one casts the vote a.

$$H' \stackrel{\mathsf{def}}{=} \nu skE.\, \nu auth_1.\, \nu auth_2.$$
$$\overline{bb}\langle \mathsf{pk}(skE)\rangle.\, (V(auth_1, id_1, b) \mid V(auth_2, id_2, a) \mid BB(auth_1, auth_2, auth_3))$$

Let (tr, ϕ) be the trace corresponding to the execution of H described in Example 3 where $\phi = \varphi(H_2) = H_2$ (as defined in Example 3) and $tr = \nu xk.\, \overline{bb}\langle xk\rangle \cdot \nu x.\, \overline{bb}\langle x\rangle \cdot \nu y.\, \overline{bb}\langle y\rangle \cdot auth_3(\langle id_3, x\rangle) \cdot \nu z.\, \overline{bb}\langle z\rangle \cdot \nu x'.\, \overline{bb}\langle x'\rangle \cdot \nu y'.\, \overline{bb}\langle y'\rangle \cdot \nu z'.\, \overline{bb}\langle z'\rangle$. Then $(tr, \phi) \in \mathsf{trace}_{\mathsf{E_{enc}}}(H)$ and for any $(tr, \phi') \in \mathsf{trace}_{\mathsf{E_{enc}}}(H')$, it is easy to check that we have $\phi \not\sim_{\mathsf{E_{enc}}} \phi'$. (In fact, $\phi' = \psi$ from Example 4.) Therefore, $H \overset{\mathsf{tr}}{\not\approx}_{\mathsf{E_{enc}}} H'$

Intuitively, if the dishonest voter's strategy is to replay the first voter's vote, then he would cast a vote of the form $\langle id_3, \mathsf{aenc}(\mathsf{pk}(skE), r_1, a)\rangle$ in the system H while he would cast a vote of the form $\langle id_3, \mathsf{aenc}(\mathsf{pk}(skE), r_1, b)\rangle$ in the system H'. Once the result is published, an attacker would be then able to distinguish H from H' since the tally in H is $\{a, b, a\}$ while it is $\{b, a, b\}$ in H'. This corresponds exactly to the replay attack against Helios, explained in [9].

3 Forward and everlasting indistinguishability

In this section we introduce and illustrate our definitions of forward and everlasting indistinguishability. In the next section we will show how the here presented definitions can be used to define practical everlasting privacy in electronic voting.

3.1 Definitions of forward and everlasting indistinguishability

From now on we suppose that Σ is a signature and that E_0 and E_1 are equational theories over Σ. We want to model that an attacker may interact with a protocol today and store some data which may be exploited in the future when his computational power has increased. We model the fact that the attacker's computational power may change by using two different equational theories: E_0 models the attacker's capabilities while interacting with the protocol at the time of the election, while E_1 models his capabilities when exploiting the published data in the future when the strength of cryptography has been eroded.

We also argue that in many situations it is reasonable to suppose that the attacker does not store all of the data that was sent over the network. We will therefore consider some channels to be *everlasting*: data sent over such channels will remain in the attacker's knowledge for future analysis while other data will be "forgotten" and can only be used during the interaction with the protocol. Typically, everlasting channels are media such as web-pages published on the Internet (that can easily be accessed by anyone, for a rather long period of time) while public but non-everlasting channels can be communications over the Internet, which can be recorded only by the active and costly involvement of an attacker.

In order to reason about data that has been sent on certain channels we introduce the following notation. Let P be a process, $\mathcal{C}$ a set of channels (corresponding to the

everlasting channels), and $tr = (\alpha_1 \cdot \alpha_2 \cdot \ldots \cdot \alpha_n, \varphi) \in \mathsf{trace}_E(P)$ a trace of P. We define the set of variables in the domain of φ corresponding to terms sent on channels in $\mathcal{C}$ as $\mathcal{V}_{\mathcal{C}}(\alpha_1 \cdot \alpha_2 \cdot \ldots \cdot \alpha_n) = \{x \mid c \in \mathcal{C},\ 1 \leq i \leq n,\ \alpha_i = \nu x.\, \overline{c}\langle x\rangle\}$ and denote by $\phi_{\mathcal{V}}(P_n)$ the substitution $\phi(P_n)$ whose domain is restricted to the set of variables $\mathcal{V}$.

Two processes A and B are said to be forward indistinguishable if, informally, an attacker cannot observe the difference between A and B being given the computational power modeled by E_1 (where it can break keys for example), but for executions that happened in the past, that is over E_0 (the standard theory) and observing only the information that was passed through everlasting channels.

Definition 3 (Forward indistinguishability). *Let A and B be two closed extended processes and $\mathcal{C}$ a set of channels. We define $A \overset{\mathsf{fwd}}{\sqsubseteq}^{\mathcal{C}}_{\mathsf{E}_0,\mathsf{E}_1} B$, if for every trace $(\alpha_1 \cdot \alpha_2 \cdots \alpha_n, \varphi_A) \in \mathsf{trace}_{\mathsf{E}_0}(A)$ there exists φ_B s.t. $(\alpha_1 \cdot \alpha_2 \cdots \alpha_n, \varphi_B) \in \mathsf{trace}_{\mathsf{E}_0}(B)$*

$$\textit{and} \quad \phi_{A\mathcal{V}} \ \sim_{\mathsf{E}_1} \ \phi_{B\mathcal{V}}.$$

where $\mathcal{V} = \mathcal{V}_{\mathcal{C}}(\alpha_1 \cdot \alpha_2 \cdots \alpha_n)$. A and B are forward indistinguishable w.r.t. *$\mathcal{C}$, E_0 and E_1, denoted $A \overset{\mathsf{fwd}}{\approx}^{\mathcal{C}}_{\mathsf{E}_0,\mathsf{E}_1} B$, if $A \overset{\mathsf{fwd}}{\sqsubseteq}^{\mathcal{C}}_{\mathsf{E}_0,\mathsf{E}_1} B$ and $B \overset{\mathsf{fwd}}{\sqsubseteq}^{\mathcal{C}}_{\mathsf{E}_0,\mathsf{E}_1} A$.*

Note that in the above definition we only check equivalence of messages that were sent on channels in the set $\mathcal{C}$. We may also require that A and B are indistinguishable in the standard way (over E_0). Standard indistinguishability and forward indistinguishability yield *everlasting indistinguishability*.

Definition 4 (Everlasting indistinguishability). *Let A and B be two closed extended processes, $\mathcal{C}$ a set of channels. A and B are* everlasting indistinguishable w.r.t. *$\mathcal{C}$, E_0 and E_1, denoted $A \overset{\mathsf{ev}}{\approx}^{\mathcal{C}}_{\mathsf{E}_0,\mathsf{E}_1}$ if*

1. *$A \overset{\mathsf{tr}}{\approx}_{\mathsf{E}_0} B$,* i.e. *$A$ and B are trace equivalent* w.r.t. *E_0; and*
2. *$A \overset{\mathsf{fwd}}{\approx}^{\mathcal{C}}_{\mathsf{E}_0,\mathsf{E}_1} B$,* i.e. *$A$ and B are forward indistinguishable* w.r.t. *$\mathcal{C}$, E_0 and E_1.*

3.2 Examples

We illustrate the above definitions on a simple RFID protocol. In the context of RFID systems, forward secrecy is often a desired property: even if an RFID tag has been tampered with, and its secrets have been retrieved by a malicious entity, its past transactions should remain private. This can be seen as a form of everlasting security requirement. Indeed, RFID tags being devices vulnerable to tampering, one would like to make sure that when an intruder gains access to an honest device, he is not able to trace back the movements of the tag. Such tampering can be modelled by the following equational theory $\mathsf{E}_{\mathsf{break}}$, that gives direct access to keys.

$$\mathsf{E}_{\mathsf{break}} = \left\{ \begin{array}{r} \mathsf{break}_{\mathsf{aenc}}(\mathsf{aenc}(\mathsf{pk}(x), y, z)) = x \\ \mathsf{break}_{\mathsf{senc}}(\mathsf{senc}(x, y, z)) = x \end{array} \right\}$$

We also use this equational theory later to model that in 20 or 30 years an adversary will be able to break nowadays encryption keys.

Consider the following toy RFID protocol

$$\begin{aligned} session &= \nu r.\, \overline{c}\langle \mathsf{enc}(k, r, id)\rangle \\ tag &= \nu k.\, \nu id.\, !session \\ system &= !tag \end{aligned}$$

where a tag identifies itself to a reader by sending its tag identifier id encrypted with a long-term symmetric key shared between the tag and the reader.

We can model unlinkability as being the property that an attacker cannot distinguish the situation where the same tag is used in several sessions from the situation where different tags are used. Formally unlinkability is modelled as the trace equivalence:

$$system \overset{\mathsf{tr}}{\approx}_{\mathsf{E_{enc}}} system'$$

where

$$system' = !\nu k.\nu id.\ session.$$

Intuitively, this protocols satisfies unlinkability only as long as the keys are not leaked. Indeed, since each identification uses a different random in the encrypted message sent to the reader, each of the sent messages is different and looks like a random message to the intruder. However, $system$ and $system'$ are not forward indistinguishable when considering a theory E_1 which allows to break keys, i.e.,

$$system \not\overset{\mathsf{fwd}}{\approx}^{\{c\}}_{\mathsf{E_{enc}}, \mathsf{E_{enc}} \cup \mathsf{E_{break}}} system'$$

where $\mathsf{E_{enc}}$ is the equational theory introduced in Example 1. Indeed, once the key k of a tag is obtained by the intruder, he can retrieve the identity behind each blob he has seen on channel c, and thus distinguish the set of messages obtained by an execution of $system$ where the same tag executes at least two sessions, from the set of messages obtained by the corresponding execution of $system'$ where each tag has executed at most one session. Thus this protocol does not satisfy the stronger requirement of everlasting indistinguishability either:

$$system \not\overset{\mathsf{ev}}{\approx}^{\{c\}}_{\mathsf{E_{enc}}, \mathsf{E_{enc}} \cup \mathsf{E_{break}}} system'$$

4 Application to practical everlasting privacy

We model a practical version of everlasting privacy in voting protocols based on everlasting indistinguishability.

4.1 Definition of practical everlasting privacy

We first recall the definition of vote privacy introduced in [15].

Definition 5 (Vote privacy). *Let* $\mathsf{VP}(\mathsf{v}_1, \mathsf{v}_2)$ *be an extended process with two free variables* v_1, v_2. $\mathsf{VP}(\mathsf{v}_1, \mathsf{v}_2)$ *respects vote privacy for an equational theory* E *if*

$$\mathsf{VP}(a, b) \overset{\mathsf{tr}}{\approx}_{\mathsf{E}} \mathsf{VP}(b, a)$$

Intuitively, the free variables refer to the votes of two honest voters id_1 and id_2. Then this equivalence ensures that an attacker cannot distinguish the situations where voter id_1 voted for candidate a and voter id_2 voted for candidate b, from the situation where the voters swapped their votes, i.e., voter id_1 voted for candidate b and voter id_2 voted for candidate a.

Example 6. Let $\mathsf{Helios}(v_1, v_2)$ be the process

$$\begin{array}{l}\nu skE.\ \nu auth_1.\ \nu auth_2.\\ \overline{bb}\langle \mathsf{pk}(skE)\rangle.\ (V(auth_1, id_1, v_1) \mid V(auth_2, id_2, v_2) \mid BB(auth_1, auth_2, auth_3))\end{array}$$

where processes V and BB are defined in Example 2.

In Example 5, when we illustrated trace equivalence we showed that Helios does not satisfy vote privacy due to a vote replay attack discovered in [9].

A simple fix of the attack consists in weeding duplicates. The corresponding tally is

$$\begin{array}{l}T' \stackrel{\mathsf{def}}{=} c(x').c(y').c(z').\\ \quad \text{if } \mathsf{snd}(x') \neq \mathsf{snd}(y') \ \wedge\ \mathsf{snd}(x') \neq \mathsf{snd}(z') \wedge \mathsf{snd}(y') \neq \mathsf{snd}(z') \text{ then}\\ \quad\quad \overline{bb}\langle \mathsf{adec}(skE, \mathsf{snd}(x'))\rangle \mid \overline{bb}\langle \mathsf{adec}(skE, \mathsf{snd}(y'))\rangle \mid \overline{bb}\langle \mathsf{adec}(skE, \mathsf{snd}(z'))\rangle\end{array}$$

In other words, the tally is performed only if there are no duplicates amongst the cast votes. We define the voting protocol $\mathsf{Helios}^{\mathsf{noreplay}}$ as Helios but with the revised version T' of the tally. Using the tools ProVerif and AKISS we have shown that this protocol satisfies vote privacy.

$$\mathsf{Helios}^{\mathsf{noreplay}}(a, b) \stackrel{\mathsf{tr}}{\approx}_{\mathsf{E}_{\mathsf{enc}}} \mathsf{Helios}^{\mathsf{noreplay}}(b, a)$$

The above definition of vote privacy does however not take into account that most cryptographic schemes rely on computational assumptions and may be broken in the future. In order to protect the secrecy of votes against such attacks in the future we propose a stronger definition based on forward indistinguishability.

Definition 6 (Everlasting vote privacy). *Let* $\mathsf{VP}(\mathsf{v}_1, \mathsf{v}_2)$ *be an extended process with two free variables* v_1, v_2. $\mathsf{VP}(v_1, v_2)$ *satisfies* everlasting privacy *w.r.t. a set of channels* $\mathcal{C}$ *and equational theories* E_0 *and* E_1*, if*

$$\mathsf{VP}(a, b) \stackrel{\mathsf{ev}}{\approx}^{\mathcal{C}}_{\mathsf{E}_0, \mathsf{E}_1} \mathsf{VP}(b, a)$$

We note that everlasting vote privacy is strictly stronger than vote privacy as it requires trace equivalence of $\mathsf{VP}(a, b)$ and $\mathsf{VP}(b, a)$ (which is exactly vote privacy) and additionally forward indistinguishability of these processes. Our definition is parametric with respect to the equational theories and the channels we suppose to be everlasting. The equational theory E_1 allows us to exactly specify what a future attacker may be able to break. The set of everlasting channels $\mathcal{C}$ allows us to specify what data a future attacker has access to. When $\mathcal{C}$ corresponds to all channels we typically get a requirement which is too strong for practical purposes. We argue that it is reasonable to suppose that in, say 50 years, an attacker does not have access to the transmissions between individual voters and the system while a bulletin board published on the Internet could easily have been stored.

4.2 Examples

Helios with identities As discussed In Example 6, $\mathsf{Helios}^{\mathsf{noreplay}}$ does satisfy vote privacy. However, this protocol does not satisfy everlasting vote privacy with $\mathsf{E}_0 = \mathsf{E}_{\mathsf{enc}}$, $\mathsf{E}_1 = \mathsf{E}_{\mathsf{enc}} \cup \mathsf{E}_{\mathsf{break}}$ and $\mathcal{C} = \{bb\}$. Intuitively, this is due to the fact that a future attacker can break encryption and link the recovered vote to the identity submitted together with the cast ballot. Formally, we can show that

$$\mathsf{Helios}^{\mathsf{noreplay}}(a,b) \overset{\mathsf{fwd}}{\not\approx} \mathsf{Helios}^{\mathsf{noreplay}}(b,a)$$

Consider the trace $(\nu xk.\ \overline{bb}\langle xk\rangle \cdot \nu x.\ \overline{bb}\langle x\rangle \cdot \nu y.\ \overline{bb}\langle y\rangle, \varphi_A) \in \mathsf{trace}_{\mathsf{E}_{\mathsf{enc}}}(\mathsf{Helios}^{\mathsf{noreplay}}(a,b))$ where

$$\varphi_A = \nu skE, r_1, r_2.\{\ \begin{array}{l}\mathsf{pk}(skE)/_{xk},\\ \langle id_1, \mathsf{aenc}(\mathsf{pk}(skE), r_1, a)\rangle/_x,\\ \langle id_2, \mathsf{aenc}(\mathsf{pk}(skE), r_2, b)\rangle/_y\}\end{array}$$

Traces $(\nu xk.\ \overline{bb}\langle xk\rangle \cdot \nu x.\ \overline{bb}\langle x\rangle \cdot \nu y.\ \overline{bb}\langle y\rangle, \varphi_B) \in \mathsf{trace}_{\mathsf{E}_{\mathsf{enc}}}(\mathsf{Helios}^{\mathsf{noreplay}}(b,a))$ are either such that

$$\varphi_B \equiv \nu skE, r_1, r_2.\{\ \begin{array}{l}\mathsf{pk}(skE)/_{xk},\\ \langle id_1, \mathsf{aenc}(\mathsf{pk}(skE), r_1, b)\rangle/_x,\\ \langle id_2, \mathsf{aenc}(\mathsf{pk}(skE), r_2, a)\rangle/_y\}\end{array}$$

or

$$\varphi_B \equiv \nu skE, r_1, r_2.\{\ \begin{array}{l}\mathsf{pk}(skE)/_{xk},\\ \langle id_2, \mathsf{aenc}(\mathsf{pk}(skE), r_1, a)\rangle/_x,\\ \langle id_1, \mathsf{aenc}(\mathsf{pk}(skE), r_2, b)\rangle/_y\}\end{array}$$

In both cases we have that $\varphi_A \not\sim_{\mathsf{E}_{\mathsf{enc}} \cup \mathsf{E}_{\mathsf{break}}} \varphi_B$. In the first case this is witnessed by the test $M = a$ and $N = \mathsf{break}_{\mathsf{aenc}}(\mathsf{snd}(x))$ as

$$M\varphi_A = a =_{\mathsf{E}_{\mathsf{enc}} \cup \mathsf{E}_{\mathsf{break}}} N\varphi_A \quad \text{but} \quad M\varphi_B = a \neq_{\mathsf{E}_{\mathsf{enc}} \cup \mathsf{E}_{\mathsf{break}}} b =_{\mathsf{E}_{\mathsf{enc}} \cup \mathsf{E}_{\mathsf{break}}} N\varphi_B$$

In the second case non equivalence is witnessed by the test $M = id_1$ and $N = \mathsf{fst}(x)$.

Helios without identities As we just saw $\mathsf{Helios}^{\mathsf{noreplay}}$ does not satisfy everlasting privacy. This is due to the fact that encrypted votes are published together with the identity of the voter on the bulletin board. A simple variant (used e.g. in Louvain for student elections) consists in publishing the encrypted vote without the identity of the voter. We define $\mathsf{Helios}^{\mathsf{noid}}$ as $\mathsf{Helios}^{\mathsf{noreplay}}$ but redefining the process BB' as

$$\begin{array}{l}BB'(a_1,a_2,a_3) \overset{\mathsf{def}}{=} \nu c.\ a_1(x).\ \overline{bb}\langle \mathsf{snd}(x)\rangle.\ \overline{c}\langle x\rangle \mid a_2(y).\ \overline{bb}\langle \mathsf{snd}(y)\rangle.\ \overline{c}\langle y\rangle \mid\\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad a_3(z).\ \overline{bb}\langle \mathsf{snd}(z)\rangle.\ \overline{c}\langle z\rangle \mid T'\end{array}$$

where T' is as defined at Example 6. As we shall see in Section 6, we prove everlasting privacy of $\mathsf{Helios}^{\mathsf{noid}}$ w.r.t $\mathsf{E}_{\mathsf{enc}}, \mathsf{E}_{\mathsf{break}}$ and everlasting channel bb, using (adaptations of) ProVerif and AKISS.

5 Modeling commitments

Commitment schemes allow a sender to commit to a value v while keeping this value hidden until an 'opening' phase, where the sender reveals v to the receiver of the commitment $\mathsf{commit}(v)$. The receiver should then be able to verify that the revealed value is indeed the one used to compute $\mathsf{commit}(v)$, and in that sense that the sender had indeed committed to the revealed value. The two main security properties of such schemes are *binding* (the sender can't claim that $\mathsf{commit}(v)$ is a commitment to some $v' \neq v$), and *hiding* (the receiver can't deduce v from $\mathsf{commit}(v)$). These two properties can hold 'perfectly' or merely 'computationally'. It is known that there are no commitment schemes which are both perfectly hiding and perfectly binding, so one has to choose between perfectly hiding and computationally binding (PHCB) and perfectly binding and computationally hiding (PBCH). In this section, we characterize in our formal model what it means for a primitive to be PHCB and PBCH. We also give equational theories to model such primitives, which we then use for the verification of two voting protocols that rely on such primitives to ensure everlasting vote privacy.

5.1 Modeling hiding and binding cryptographic primitives

PBCH primitives Informally, an n-ary function f is *perfectly binding* if the inputs are totally determined by the output. In other words, f is perfectly binding if it admits no collisions. It is *computationally hiding* if it is hard to retrieve the inputs from the output.

To model a PBCH primitive f using the applied pi calculus, we introduce two equational theories $\mathsf{E}_0^{\mathsf{f}}$ and $\mathsf{E}_1^{\mathsf{f}}$ over the signature $\Sigma = \{\mathsf{f}, \mathsf{break}_{\mathsf{f}}^1, \ldots, \mathsf{break}_{\mathsf{f}}^n\}$, such that no equation of the form

$$\mathsf{f}(M_1, \ldots, M_n) =_{\mathsf{E}} \mathsf{f}(N_1, \ldots, N_n)$$

is derivable, where $(M_1, \ldots, M_n) \neq_{\mathsf{E}} (N_1, \ldots, N_n)$ and $\mathsf{E} \in \{\mathsf{E}_0^{\mathsf{f}}, \mathsf{E}_1^{\mathsf{f}}\}$; and that the equation

$$\mathsf{break}_{\mathsf{f}}^i(\mathsf{f}(v_1, \ldots, v_n)) =_{\mathsf{E}_1^{\mathsf{f}}} v_i.$$

is derivable. As before, $\mathsf{E}_0^{\mathsf{f}}$ models the capabilities of a computationally bounded attacker interacting with the protocol, while $\mathsf{E}_1^{\mathsf{f}}$ models the capabilities of a computationally unbounded attacker in the future.

Example 7. A trivial example of a perfectly binding function is the identity function id. However, id is not hiding.

Example 8. An example of a PBCH primitive is the ElGamal public key derivation function. Given multiplicative cyclic group G of order q with generator g, to generate a private and public key pair Alice does the following:

1. she chooses at random her private key $sk \in \{1, \ldots, q-1\}$,
2. she computes and publishes her public key $\mathsf{pk}_{G,g,q}(sk) = g^{sk}$.

The secret key sk is totally determined by the public key $\mathsf{pk}_{G,g,q}(sk) = g^{sk}$. It is however as hard to find sk from $\mathsf{pk}_{G,g,q}(sk)$ as it is to compute discrete logarithms.

Thus, to reason about protocols relying on ElGamal encryption we consider the following equational theories over the signature $\{\mathsf{aenc}_{G,g,q}, \mathsf{adec}_{G,g,q}, \mathsf{pk}_{G,g,q}, \mathsf{break}_{\mathsf{pk}_{G,g,q}}\}$ (we omit the subscripts G, g, q for readability):

$$\mathsf{E}_0^{\mathsf{ElGamal}} = \{\mathsf{adec}(xk, \mathsf{aenc}(\mathsf{pk}(xk), xr, xm)) = xm\}$$

$$\mathsf{E}_1^{\mathsf{ElGamal}} = \left\{ \begin{array}{l} \mathsf{adec}(xk, \mathsf{aenc}(\mathsf{pk}(xk), xr, xm)) = xm \\ \mathsf{break}_{\mathsf{pk}}(\mathsf{pk}(xk)) = xk \end{array} \right\}$$

The function $\mathsf{pk}_{G,g,q}$ is PBCH. Note however that the encryption algorithm $\mathsf{aenc}_{G,g,q}$ is not PBCH, since it is not perfectly binding. Indeed, given the parameters G, q, and g, to encrypt the message m with the public key g^{sk}, Alice would

1. pick a random $r \in \{0, \ldots, q-1\}$ and comput $c_1 = g^r$;
2. compute the secret shared key $s = (g^{sk})^r$; and
3. computer $c_2 = m.s$

The computed ciphertext is then $\mathsf{aenc}(\mathsf{pk}(sk), r, m) = (c_1, c_2) = (g^r, m.(g^{sk})^r)$. Hence, for any public key $\mathsf{pk}(sk') = g^{sk'}$, there exists a message $m' = m.(g^{sk})^r/(g^{sk'})^r$ such that $\mathsf{aenc}(\mathsf{pk}(sk), r, m) = \mathsf{aenc}(\mathsf{pk}(sk'), r, m')$. Thus, ElGamal encryption is not perfectly binding.

PHCB primitives Informally, an n-ary function f is perfectly hiding if given the output, it is impossible to retrieve any of the inputs. So even enumerating all the possible inputs shouldn't allow one to retrieve the inputs from the output of the function. But this implies that f should admit collisions for each possible input. On the other hand, f is computationally binding if it is computationally not feasible to find such collisions.

Example 9. Any constant function $f(x_1, \ldots, x_n) = c$ is obviously perfectly hiding but not computationally binding. The $\oplus$ function is also perfectly hiding since for all $z = x \oplus y$

- for all x', we have that $y' = z \oplus x'$ is such that $x \oplus y = x' \oplus y'$; and
- for all y'', we have that $x'' = z \oplus y''$ is such that $x \oplus y = x'' \oplus y''$.

However, it is not computationally binding since it is easy to compute x'' and y'.

Example 10. Pedersen commitments are PHCB. The Pedersen commitment over a cyclic group G of order q and two generators $h, g \in G$ such that $\log_g h$ is not known is the function $P_{h,g}^G(x, y) = h^x \cdot g^y (\bmod q)$. Pedersen commitments are perfectly hiding since for all $z = P_{h,g}^G(x, y)$,

- for all x', we have that $y' = y + (x - x') \cdot \log_g h \bmod q$ is such that $P_{h,g}^G(x, y) = P_{h,g}^G(x', y')$;
- for all y'', we have that $x'' = x + (y - y'') \cdot \log_h g \bmod q$ is such that $P_{h,g}^G(x, y) = P_{h,g}^G(x'', y'')$.

but they are computationally binding because finding these x'' and y' is as hard as computing discrete logarithms.

To reason about protocols relying on Pedersen commitments using the applied pi calculus, we introduce the function symbols $\mathsf{forge}^1_{\mathsf{Ped}}$, and $\mathsf{forge}^2_{\mathsf{Ped}}$ and the two following equational theories

$$\mathsf{E}_0^{\mathsf{Ped}} = \emptyset$$

$$\mathsf{E}_1^{\mathsf{Ped}} = \left\{ \begin{array}{l} \mathsf{Ped}(\mathsf{forge}^1_{\mathsf{Ped}}(v, y'), y') = v \\ \mathsf{Ped}(x', \mathsf{forge}^2_{\mathsf{Ped}}(v, x')) = v \\ \mathsf{forge}^1_{\mathsf{Ped}}(\mathsf{Ped}(x, y), y) = x \\ \mathsf{forge}^2_{\mathsf{Ped}}(\mathsf{Ped}(x, y), x) = y \\ \mathsf{forge}^1_{\mathsf{Ped}}(v, \mathsf{forge}^2_{\mathsf{Ped}}(v, x)) = x \\ \mathsf{forge}^2_{\mathsf{Ped}}(v, \mathsf{forge}^1_{\mathsf{Ped}}(v, y)) = y \end{array} \right\}$$

For the first equation, suppose $v = \mathsf{Ped}(x, y)$, and we have some y'; then $\mathsf{forge}^1_{\mathsf{Ped}}$ allows us to compute a value $x' = \mathsf{forge}^1_{\mathsf{Ped}}(v, y')$ such that $v = \mathsf{Ped}(x', y')$. The second equation is similar. The third and fourth equation allow us to recover one of the arguments, given that the other argument is known. In other words the third equation expresses that when forging $x' = \mathsf{forge}^1_{\mathsf{Ped}}(v, y)$ and $v = \mathsf{Ped}(x, y)$ then we must have that $x' = x$, and similarly for the fourth equation. The fifth and sixth equations are also seen to be mathematically valid, given that $\mathsf{forge}^1_{\mathsf{Ped}}(v, y)$ and $\mathsf{forge}^2_{\mathsf{Ped}}(v, x)$ respectively model the terms $\log_g(v/h^y)$ and $\log_h(v/g^x)$.

5.2 Applications: electronic voting protocols and everlasting privacy

Pedersen commitments have been used in several voting protocols for achieving everlasting privacy. In particular we study the protocol by Moran and Naor [18] and a recent version of Helios [12] based on Pedersen commitments.

Moran-Naor protocol Moran and Naor [18] designed a protocol to be used with voting machines in a polling station. The protocol aims to achieve both verifiability and everlasting privacy. From a high level point of view the protocol works as follows.

1. The voter enters his vote into the voting machine inside the voting booth. The machine then computes a Pedersen commitment to this vote and provides a zero knowledge proof to the voter that the computed value commits to the voter's choice. The commitment is then published on a bulletin board so that the voter can verify the presence of his ballot.
2. After all ballots have been cast, the votes are published (in random order) on the bulletin board together with a zero knowledge proof asserting that the published votes correspond to the votes of the published commitments.

As we are only interested in privacy and not verifiability we ignore the zero knowledge proofs in our modeling and simply represent the protocol by the process

$$\begin{aligned} \mathsf{MoranNaor}(v_1, v_2) \stackrel{\mathsf{def}}{=} \nu priv_1.\, \nu priv_2.& \\ V(priv_1, v_1) \mid V(priv_2, v_2) \mid \nu c.(DRE(priv_1, priv_2, priv_3) \mid T)& \end{aligned}$$

where

$$\begin{aligned} V(priv, v) &\stackrel{\mathsf{def}}{=} \overline{priv}\langle v\rangle \\ DRE(p_1, p_2, p_3) &\stackrel{\mathsf{def}}{=} p_1(x_1).\nu r_1.\overline{bb}\langle \mathsf{Ped}(x_1, r_1)\rangle.\overline{c}\langle x_1\rangle \mid \\ &\quad\; p_2(x_2).\nu r_2.\overline{bb}\langle \mathsf{Ped}(x_2, r_2)\rangle.\overline{c}\langle x_2\rangle \mid \\ &\quad\; p_3(x_3).\nu r_3.\overline{bb}\langle \mathsf{Ped}(x_3, r_3)\rangle.\overline{c}\langle x_3\rangle \\ T &= c(y_1).\overline{bb}\langle y_1\rangle \mid c(y_2).\overline{bb}\langle y_2\rangle \mid c(y_3).\overline{bb}\langle y_3\rangle \end{aligned}$$

As the voter enters his vote in a private ballot booth, we have modelled this communication on a private channel. We have been able to show that MoranNaor verifies everlasting privacy with respect to the channel bb and the equational theories introduced for Pedersen commitments.

Helios with Pedersen commitments In [12], the authors propose a version of the Helios voting system that provides everlasting vote privacy *w.r.t.* the bulletin board. They rely for this on Pedersen commitments. In this section, we present this new version of the Helios system.

1. The voter V chooses her candidate v and commits to it by generating a random number r and computing the Pedersen commitment $\mathsf{Ped}(r, v)$. She then separately encrypts the decommitment values r and v using the public key $\mathsf{pk}(skE)$ of the election; and casts her commitment together with the encrypted decommitment values and her identity on a private authenticated channel. Upon reception of the ballot, the Bulletin Board (BB) publishes on a public web page the commitment $\mathsf{Ped}(r, v)$ (after having checked that V is entitled to vote).
2. Once the voting phase is over, the ballots (*i.e.* the commitments together with the encrypted decommitment values) are shuffled and rerandomized through mixnets. The random permutation of the rerandomized ballots is published on the public webpage (together with a zero knowledge proof of correct reencryption and mixing).
3. Finally, the authorities decrypt the rerandomized and shuffled decommitment values and the BB publishes them.

The voter can be modelled by the following process:

$$\begin{aligned} V(id, auth, v) &\stackrel{\mathsf{def}}{=} \nu s.\nu rv.\nu rs. \\ &\quad \overline{auth}\langle\langle id, \langle \mathsf{Ped}(s, v), \langle \mathsf{aenc}(\mathsf{pk}(skE), rv, v), \mathsf{aenc}(\mathsf{pk}(skE), rs, s)\rangle\rangle\rangle\rangle \end{aligned}$$

She sends to the BB on the private authenticated channel $authCh$, her commitment $\mathsf{Ped}(s, v)$ to vote v, together with her identity and the encrypted decommitment values $\mathsf{aenc}(\mathsf{pk}(skE), rv, v)$, $\mathsf{aenc}(\mathsf{pk}(skE), rs, s)$.

The ballot box publishes her commitment for verifiability purposes. After having received all votes, the BB publishes the votes in a random order through the process T.

$$\begin{aligned} BB(a_1, a_2, a_3) &\stackrel{\mathsf{def}}{=} a_1(x).\ \overline{bb}\langle\langle \mathsf{fst}(x), \mathsf{fst}(\mathsf{snd}(x))\rangle\rangle.\ \overline{c}\langle x\rangle \mid \\ &\quad\; a_2(y).\ \overline{bb}\langle\langle \mathsf{fst}(x), \mathsf{fst}(\mathsf{snd}(x))\rangle\rangle.\ \overline{c}\langle x\rangle \mid \\ &\quad\; a_3(z).\ \overline{c}\langle z\rangle \mid T \end{aligned}$$

$$
\begin{aligned}
T \stackrel{\mathsf{def}}{=}\ & c(x).\ c(y).\ c(z).\mathrm{if}\ \mathsf{fst}(\mathsf{snd}(\mathsf{snd}(x))) \neq \mathsf{fst}(\mathsf{snd}(\mathsf{snd}(z))) \\
& \quad \wedge\ \mathsf{fst}(\mathsf{snd}(\mathsf{snd}(y))) \neq \mathsf{fst}(\mathsf{snd}(\mathsf{snd}(z))) \\
& \quad \wedge\ \mathsf{fst}(\mathsf{snd}(\mathsf{snd}(x))) \neq \mathsf{fst}(\mathsf{snd}(\mathsf{snd}(y)))\ \mathrm{then} \\
& \overline{bb}\langle \mathsf{adec}(skE, \mathsf{fst}(\mathsf{snd}(\mathsf{snd}(x))))\rangle \mid \\
& \overline{bb}\langle \mathsf{adec}(skE, \mathsf{fst}(\mathsf{snd}(\mathsf{snd}(y))))\rangle \mid \\
& \overline{bb}\langle \mathsf{adec}(skE, \mathsf{fst}(\mathsf{snd}(\mathsf{snd}(z))))\rangle
\end{aligned}
$$

Finally we can define the voting protocol $\mathsf{Helios}^{\mathsf{Ped}}$ as

$$
\begin{aligned}
& \mathsf{Helios}^{\mathsf{Ped}}(v1, v2) \stackrel{\mathsf{def}}{=} \nu skE.\ \nu auth_1.\ \nu auth_2. \\
& \quad \overline{bb}\langle \mathsf{pk}(skE)\rangle.\ (V(auth_1, id_1, v_1) \mid V(auth_2, id_2, v_2) \mid BB(auth_1, auth_2, auth_3))
\end{aligned}
$$

which verifies everlasting privacy with respect to the channel bb and the previously introduced equational theories.

6 Tool support for everlasting indistinguishability

In order to verify everlasting indistinguishability on the examples presented in the previous section we have adapted two tools for automated verification of equivalence properties, AKISS [6] and ProVerif [5]. The two tools have shown themselves to be complementary and the results obtained using the tools are summarized in Figure 3.

AKISS. AKISS is a recent tool that has been designed to automatically prove trace equivalence by translating processes into Horn clauses and using a dedicated resolution algorithm. More precisely it can both under- and over-approximate trace equivalence in the case of a bounded number of sessions, i.e. for processes without replication. The tool has currently two limitations: it does not support private channels, or else branches in conditionals. However, it is able to deal with a wide range of equational theories, including the theory for Pedersen commitments introduced in the previous section.

We have adapted the tool in order to check forward indistinguishability and adapted the syntax to declare everlasting channels and an everlasting equational theory. More precisely we implemented an algorithm to check an under-approximation of forward indistinguishability, yielding a proof of forward indistinguishability whenever the tool responds positively. While false attacks are possible, we did not encounter any in our case studies.

Absence of support for private channels and else branches required us to adapt some of the examples. In particular we rewrote the processes by directly *inlining* private communications, which in the examples maintained the same set of traces, hence preserving everlasting indistinguishability. The weeding operation in $\mathsf{Helios}^{\mathsf{noreplay}}$, $\mathsf{Helios}^{\mathsf{noid}}$ and $\mathsf{Helios}^{\mathsf{ped}}$ requires the use of an else branch. We encoded a different weeding procedure using cryptographic proofs of knowledge. While the vote replay attack on the simple Helios protocol is found in less than 10 seconds, the verification of other examples ranged from a few minutes to several hours. While attempting to verify $\mathsf{Helios}^{\mathsf{ped}}$ the tool ran out of memory and we were only able to verify a version of $\mathsf{Helios}^{\mathsf{ped}}$ with two honest voters and no dishonest voter. As the tool is still in a prototype status

we are confident that future optimizations will allow the tool to scale up to this kind of protocols. The tool and example files are available at `https://github.com/ciobaca/akiss`.

ProVerif. The ProVerif tool [4] is an automatic cryptographic protocol verifier. It is based on the representation of protocols by Horn clauses and relies on several approximations. ProVerif can handle several types of properties and in particular equivalence based properties [5] like the privacy-type ones which we are interested in this work. Moreover, ProVerif can handle many different cryptographic primitives, including Pedersen commitments as our case studies show.

The ProVerif tool works by translating biprocesses into Horn clauses built over the two predicates attacker2 and message2. For equivalence checking, biprocess is used to represent the pair of processes for which ProVerif is called to check equivalence. The fact $\mathsf{attacker2}(M, M')$ means that the attacker can learn the value M (*resp.* M') from the first (*resp.* second) process encoded by the biprocess. The fact $\mathsf{message2}(M, N, M', N')$ means that the message N (*resp.* N') has appeared on the channel M (*resp.* M') while executing the first (*resp.* second) process encoded by the biprocess.

As for the AKISS tool, our extension of ProVerif consists in the addition of constructs for declaring *everlasting channels* and a *future equational theory* (different from the *present* one). We introduce the extra binary predicate attacker2_ev for the generation of Horn clauses from biprocesses of our extended ProVerif language. The fact $\mathsf{attacker2\_ev}(M, M')$ means that in the future, the attacker will either remember or be able to compute from messages he remembers, the value M (*resp.* M'). The declaration of an everlasting channel c generates the following *inheritance* Horn clause:

$$\mathsf{message2} : c[], xm, c[], ym \ \rightarrow \ \mathsf{attacker2\_ev} : xm, ym$$

This clause transports messages sent on the everlasting channel to the "future". The declaration of future equations generates the same equations as present ones but using our new attacker2_ev predicate. For example, the everlasting equation

$$\mathsf{break}(\mathsf{aenc}(\mathsf{pk}(xk), xr, xm)) = xk$$

will generate the two following clauses

$$\begin{array}{l} \mathsf{attacker2\_ev} : x, \mathsf{aenc}(\mathsf{pk}(xk), xr, xm) \ \rightarrow \ \mathsf{attacker2\_ev} : \mathsf{break}(x), xk \\ \mathsf{attacker2\_ev} : \mathsf{aenc}(\mathsf{pk}(xk), xr, xm), x \ \rightarrow \ \mathsf{attacker2\_ev} : xk, \mathsf{break}(x) \end{array}$$

These clauses model the "future" ability of the attacker to recover the decryption key of ciphertexts he remembers.

Using our extension of the ProVerif tool, we managed to find the attack on $\mathsf{Helios}^{\mathsf{noreplay}}$ presented in section 4.2, but also to prove that $\mathsf{Helios}^{\mathsf{noid}}$, $\mathsf{Helios}^{\mathsf{pedersen}}$ and that Moran–Naor satisfy everlasting vote privacy. However, because of the abstractions made by ProVerif, we had to adapt our models of $\mathsf{Helios}^{\mathsf{noid}}$ and $\mathsf{Helios}^{\mathsf{pedersen}}$ in order for ProVerif to succeed in proving that satisfy everlasting privacy. Indeed, these two protocols do not satisfy uniformity under reductions, and ProVerif reported false attacks on these

two protocols. To overcome this limitation of ProVerif, we fixed the order in which the three voters cast their votes.

The tool and example files are available at `http://markryan.eu/research/EverlastingPrivacy/`.

	AKiSs	ProVerif
Helios	attack on privacy	attack on privacy
Helios$^{\text{noreplay}}$	proof of privacy attack on everlasting privacy	proof of privacy attack on everlasting privacy
Helios$^{\text{noid}}$	proof of everlasting privacy	proof of everlasting privacy (voters casting their votes in fixed order)
Helios$^{\text{ped}}$	proof of everlasting privacy (2 honest voters only)	proof of everlasting privacy (voters casting their votes in fixed order)
Moran-Naor	proof of everlasting privacy	proof of everlasting privacy

Fig. 3. Automated verification using AKiSs and ProVerif.

7 Conclusion

The key idea of "practical" everlasting privacy is that in the future, an attacker will be more powerful in terms of computation (he may be able to break the cryptography) but less powerful in terms of the data he can operate on (transactions between a vote client and the vote server may not have been stored). We realized this idea in the "symbolic" model by allowing different equational theories in different phases, and restricting the information flow from the earlier phase to the later one. We modified ProVerif and AKiSs to verify our examples automatically.

We foresee to apply our results to more evolved case studies, e.g. taking into account the zero knowledge proofs that we omitted here for simplicity. Our case studies also show the limitations of the tools for checking equivalence properties which motivates further work to increase their efficiency and scope. Finally, the ability to model different equational theories with restricted information passing between them opens up possibilities for modeling breakable cryptography and other kinds of forward security. In particular it would be interesting to apply the notion of everlasting security to other flavors of anonymity and untraceability.

Acknowledgements. The research leading to these results has received funding from the European Research Council under the European Unions Seventh Framework Programme (FP7/2007-2013) / ERC grant agreement no 258865, project ProSecure, the ANR projects ProSe (decision ANR 2010-VERS-004) and JCJC VIP (decision ANR-11-JS02-006). We also acknowledge funding from EPSRC projects EP/G02684X/1 "Trustworthy Voting Systems" and EP/H005501/1 "Analysing Security and Privacy Properties".

References

1. M. Abadi and C. Fournet. Mobile values, new names, and secure communication. In *28th Symposium on Principles of Programming Languages (POPL'01)*. ACM Press, 2001.
2. B. Adida. Helios: web-based open-audit voting. In *17th conference on Security symposium (SS'08)*. USENIX Association, 2008.
3. M. Backes, C. Hritcu, and M. Maffei. Automated verification of remote electronic voting protocols in the applied pi-calculus. In *21st IEEE Computer Security Foundations Symposium (CSF'08)*. IEEE, 2008.
4. B. Blanchet. An efficient cryptographic protocol verifier based on Prolog rules. In *14th Computer Security Foundations Workshop (CSFW'01)*. IEEE Comp. Soc. Press, 2001.
5. B. Blanchet, M. Abadi, and C. Fournet. Automated verification of selected equivalences for security protocols. *Journal of Logic and Algebraic Programming*, 75(1), 2008.
6. R. Chadha, Ş. Ciobâcă, and S. Kremer. Automated verification of equivalence properties of cryptographic protocols. In *21th European Symposium on Programming (ESOP'12)*, volume 7211 of *LNCS*. Springer, 2012.
7. D. Chaum, P. Ryan, and S. Schneider. A practical, voter-verifiable election scheme. In *10th European Symposium On Research In Computer Security (ESORICS'05)*, volume 3679 of *LNCS*. Springer, 2005.
8. M. Clarkson, S. Chong, and A. Myers. Civitas: Toward a secure voting system. In *29th IEEE Symposium on Security and Privacy (S&P'08)*, 2008.
9. V. Cortier and B. Smyth. Attacking and fixing helios: An analysis of ballot secrecy. In *24th IEEE Computer Security Foundations Symposium (CSF'11)*, June 2011.
10. E. Cuvelier, T. Peters, and O. Pereira. Election verifiabilty or ballot privacy: Do we need to choose? SecVote, Dagstuhl, 2012. `secvote.uni.lu/slides/opereira-verif-or-priv.pdf`.
11. Stéphanie Delaune, Steve Kremer, and Mark D. Ryan. Verifying privacy-type properties of electronic voting protocols. *Journal of Computer Security*, 17(4):435–487, July 2009.
12. D. Demirel, J. Van De Graaf, and R. Araújo. Improving helios with everlasting privacy towards the public. In *International conference on Electronic Voting Technology/Workshop on Trustworthy Elections (EVT/WOTE'12)*. USENIX Association, 2012.
13. J. Dreier, P. Lafourcade, and Y. Lakhnech. Defining privacy for weighted votes, single and multi-voter coercion. In *17th European Symposium on Research in Computer Security (ESORICS 2012)*, volume 7459 of *LNCS*. Springer, 2012.
14. A. Juels, D. Catalano, and M. Jakobsson. Coercion-resistant electronic elections. In *ACM workshop on Privacy in the electronic society (WPES'05)*. ACM, 2005.
15. S. Kremer and M. D. Ryan. Analysis of an electronic voting protocol in the applied pi-calculus. In *14th European Symposium on Programming (ESOP'05)*, volume 3444 of *LNCS*. Springer, 2005.
16. S. Kremer, M. D. Ryan, and B. Smyth. Election verifiability in electronic voting protocols. In *15th European Symposium on Research in Computer Security (ESORICS'10)*, volume 6345 of *LNCS*. Springer, 2010.
17. R. Küsters, T. Truderung, and A. Vogt. Accountability: definition and relationship to verifiability. In *ACM Conference on Computer and Communications Security (CCS 2010)*, 2010.
18. T. Moran and M. Naor. Receipt-free universally-verifiable voting with everlasting privacy. In *Advances in Cryptology - CRYPTO 2006*, volume 4117 of *LNCS*. Springer, 2006.
19. T. P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *Advances in Cryptology - CRYPTO '91*, volume 576 of *LNCS*. Springer, 1991.

Formal analysis of privacy for routing protocols in mobile ad hoc networks*

Rémy Chrétien and Stéphanie Delaune

LSV, ENS Cachan & CNRS & INRIA Saclay Île-de-France

Abstract. Routing protocols aim at establishing a route between distant nodes in ad hoc networks. Secured versions of routing protocols have been proposed to provide more guarantees on the resulting routes, and some of them have been designed to protect the privacy of the users.
In this paper, we propose a framework for analysing privacy-type properties for routing protocols. We use a variant of the applied-pi calculus as our basic modelling formalism. More precisely, using the notion of equivalence between traces, we formalise three security properties related to privacy, namely *indistinguishability*, *unlinkability*, and *anonymity*. We study the relationship between these definitions and we illustrate them using two versions of the ANODR routing protocol.

1 Introduction

Mobile ad hoc networks consist of mobile wireless devices which autonomously organise their communication infrastructure. They are being used in a large array of settings, from military applications to emergency rescue; and are also believed to have future uses in *e.g.* vehicular networking. In such a network, each node provides the function of a router and relays packets on paths to other nodes. Finding these paths is a crucial functionality of any ad hoc network. Specific protocols, called *routing protocols*, are designed to ensure this functionality known as *route discovery*.

Since an adversary can easily paralyse the operation of a whole network by attacking the routing protocol, substantial efforts have been made to provide efficient and secure routing protocols [21, 14, 18]. For instance, in order to prevent a malicious node to insert and delete nodes inside a path, cryptographic mechanisms such as encryption, signature, and MAC are used. However, there is a privacy problem related to the way routes are discovered by those routing protocols. Indeed, most routing protocols (*e.g.* [14, 18]) flood the entire network with a route request message containing the names of the source and the destination of the intended communication. Thus, an eavesdropper can easily observe who wants to communicate with whom even if he is not on the route between the communicating nodes. Since then, in order to limit privacy issues, several anonymous routing protocols have been developed, *e.g.* ANODR [15], AnonDSR [20] to resist against passive adversaries showing no suspicious behaviours.

* This work has been partially supported by the project JCJC VIP ANR-11-JS02-006.

Because security protocols are in general notoriously difficult to design and analyse, formal verification techniques are particularly important. For example, a flaw has been discovered in the Single-Sign-On protocol used *e.g.* by Google Apps [4]. It has been shown that a malicious application could very easily gain access to any other application (*e.g.* Gmail or Google Calendar) of their users. This flaw has been found when analyzing the protocol using formal methods, abstracting messages by a term algebra and using the AVISPA platform [5].

Whereas secrecy and authentication are well-understood notions, anonymity itself is ill-defined: behind the general concept lie distinct considerations which share the general idea of not disclosing any crucial information to an attacker on the network. Thus, formalizing privacy-type properties is not an easy task and has been the subject of several papers in the context of electronic voting (*e.g.* [12, 7]), RFID systems (*e.g.* [3, 9]), or anonymizing protocols (*e.g.* [16, 13]). Whereas some of them rely on a probabilitistic notion of anonymity (*e.g.* [19]), we focus on deterministic ones, for which formal analysis appears more natural. All these definitions share a common feature: they are based on a notion of equivalence that allows one to express the fact that two situations are similar, *i.e.* indistinguishable from the point of view of the attacker.

Our contributions. In this paper, we propose a formal framework for analyzing privacy-type properties in the context of routing protocols. We use a variant of the applied-pi calculus as our basic modeling formalism [1], which has the advantage of being based on well-understood concepts and to allow us to model various cryptographic primitives by the means of an equational theory (see Sections 2 and 3). However, in order to model route discovery protocols, we have to adapt it to take into account several features of those protocols, *e.g.* the topology of the network, broadcast communication, internal states of the nodes, *etc*

Then, we investigate the different properties a routing protocol could achieve to be considered indeed anonymous in presence of a passive attacker. We propose three different families of such properties: *indistinguishability*, which deals with the possibility to distinguish some external action undertaken by an agent from another (see Section 4); *unlinkability*, which is related to the ability for the attacker to link certain actions together (see Section 5); and finally *anonymity* which concerns the disclosure of information such as the identity of the sender, or the identity of the receiver (see Section 6). We formalise those properties using a notion of equivalence between traces. Some difficulties arise due to the application under study. In particular, to achieve those security properties, we have to ensure that the network is *active enough*, and thus we have to provide a formal definition of this notion. We study the relationship between these privacy-type properties and we illustrate our definitions on two versions of the ANODR routing protocol [15].

Related work. Notions of privacy have been studied for RFID protocols [3] such as the key establishment protocol used in the electronic passport application. Similarly, formal definitions and proofs of anonymity for anonymizing protocols, like the onion routing, were proposed in [16, 13]. Nevertheless these formalisms do

not allow one to freely specify network topologies, a crucial feature for mobile ad-hoc routing. Moreover, as an extension of the applied pi-calculus, our formalism is not bound to a fixed set of primitives but make our definition usable for a large class of routing protocols. A more detailed version of this paper is available in [10].

2 Messages and attacker capabilities

As often in protocol analysis, cryptographic primitives are assumed to work perfectly. However, we do *not* consider an active attacker who controls the entire network as generally done when analyzing more classical protocols. We will consider an eavesdropper who listens to some nodes of the network or even all of them. Basically, he can see messages that are sent from locations he is spying on, and can only encrypt, decrypt, sign messages or perform other cryptographic operations if he has the relevant keys.

2.1 Messages

For modeling messages, we consider an arbitrary term algebra, which provides a lot of flexibility in terms of which cryptographic primitives can be modelled. In such a setting, messages are represented by terms where cryptographic primitives such as encryption, signature, and hash function, are represented by *function symbols*. More precisely, we consider a *signature* $(\mathcal{S}, \Sigma)$ made of a set of sorts $\mathcal{S}$ and a set of *function symbols* Σ together with arities of the form $ar(\mathsf{f}) = s_1 \times \ldots \times s_k \rightarrow s$ where $\mathsf{f} \in \Sigma$, and $s, s_1, \ldots, s_k \in \mathcal{S}$. We consider an infinite set of *variables* $\mathcal{X}$ and an infinite set of *names* $\mathcal{N}$ which are used for representing keys, nonces, *etc* We assume that names and variables are given with sorts. *Terms* are defined as names, variables, and function symbols applied to other terms. Of course function symbol application must respect sorts and arities. For $\mathcal{A} \subseteq \mathcal{X} \cup \mathcal{N}$, the set of terms built from $\mathcal{A}$ by applying function symbols in Σ is denoted by $\mathcal{T}(\Sigma, \mathcal{A})$.

We write $vars(u)$ (resp. $names(u)$) for the set of variables (resp. names) that occur in a term u. A term u is said to be a *ground* term if $vars(u) = \emptyset$. Regarding the sort system, we consider a special sort agent that only contains names and variables. These names represent the names of the agents, also called the nodes of the network. We assume a special sort msg that subsumes all the other sorts, *i.e.* any term is of sort msg.

For our cryptographic purposes, it is useful to distinguish a subset Σ_{pub} of Σ, made of *public symbols*, *i.e.* the symbols made available to the attacker. A *recipe* is a term in $\mathcal{T}(\Sigma_{\mathsf{pub}}, \mathcal{X} \cup \mathcal{N})$, that is, a term containing no private (non-public) symbols. Moreover, to model algebraic properties of cryptographic primitives, we define an *equational theory* by a finite set E of equations $u = v$ with $u, v \in \mathcal{T}(\Sigma, \mathcal{X})$ (note that u, v do not contain names). We define $=_{\mathsf{E}}$ to be the smallest equivalence relation on terms, that contains E and that is closed under application of function symbols and substitutions of terms for variables.

Example 1. A typical signature for representing secured routing protocols is the signature $(\mathcal{S}, \Sigma)$ defined by

- $\mathcal{S} = \{\mathsf{agent}, \mathsf{msg}\}$, and
- $\Sigma = \{\langle\rangle, \mathsf{proj}_1, \mathsf{proj}_2, \mathsf{senc}, \mathsf{sdec}, \mathsf{aenc}, \mathsf{adec}, \mathsf{pub}, \mathsf{prv}, \mathsf{req}, \mathsf{rep}, \mathsf{src}, \mathsf{dest}, \mathsf{key}\}$

with the following arities:

$$\mathsf{senc}, \mathsf{sdec}, \mathsf{aenc}, \mathsf{adec}, \langle\,\rangle : \mathsf{msg} \times \mathsf{msg} \to \mathsf{msg} \qquad \mathsf{pub}, \mathsf{prv} : \mathsf{agent} \to \mathsf{msg}$$
$$\mathsf{req}, \mathsf{rep}, \mathsf{src}, \mathsf{dest}, \mathsf{key} : \to \mathsf{msg} \qquad \mathsf{proj}_1, \mathsf{proj}_2 : \mathsf{msg} \to \mathsf{msg}$$

The constants req and rep are used to identify the request phase and the reply phase, src, dest, and key are some other public constants. The function symbols sdec, senc (resp. adec and aenc) of arity 2 represent symmetric (resp. asymmetric) decryption and encryption. Pairing is modelled using a symbol of arity 2, denoted $\langle\,\rangle$, and projection functions are denoted proj_1 and proj_2. We denote by $\mathsf{pub}(A)$ (resp. $\mathsf{prv}(A)$) the public key (resp. the private key) associated to the agent A. Moreover, we assume that $\mathsf{prv} \notin \Sigma_{\mathsf{pub}}$. Then, we consider the equational theory E, defined by the following equations ($i \in 1, 2$):

$$\mathsf{sdec}(\mathsf{senc}(x, y), y) = x \quad \mathsf{adec}(\mathsf{aenc}(x, \mathsf{pub}(y)), \mathsf{prv}(y)) = x \quad \mathsf{proj}_i(\langle x_1, x_2\rangle) = x_i$$

For sake of clarity, we write $\langle t_1, t_2, t_3\rangle$ for the term $\langle t_1, \langle t_2, t_3\rangle\rangle$.

Substitutions are written $\sigma = \{x_1 \triangleright u_1, \ldots, x_n \triangleright u_n\}$ where its *domain* is written $dom(\sigma) = \{x_1, \ldots, x_n\}$, and its *image* is written $img(\sigma) = \{u_1, \ldots, u_n\}$. We only consider *well-sorted* substitutions, that is substitutions for which x_i and u_i have the same sort. The application of a substitution σ to a term u is written $u\sigma$. A *most general unifier* of two terms u and v is a substitution denoted by $mgu(u, v)$. We write $mgu(u, v) = \bot$ when u and v are not unifiable.

2.2 Attacker capabilities

To represent the knowledge of an attacker (who may have observed a sequence of messages $t_1, \ldots, t_\ell$), we use the concept of *frame*. A frame $\phi = \mathsf{new}\ \tilde{n}.\sigma$ consists of a finite set $\tilde{n} \subseteq \mathcal{N}$ of *restricted* names (those unknown to the attacker), and a substitution σ of the form $\{y_1 \triangleright t_1, \ldots, y_\ell \triangleright t_\ell\}$ where each t_i is a ground term. The variables y_i enable an attacker to refer to each t_i. The *domain* of the frame ϕ, written $dom(\phi)$, is $dom(\sigma) = \{y_1, \ldots, y_\ell\}$.

In the frame $\phi = \mathsf{new}\ \tilde{n}.\sigma$, the names $\tilde{n}$ are bound in σ and can be renamed. Moreover names that do not appear in ϕ can be added or removed from $\tilde{n}$. In particular, we can always assume that two frames share the same set of restricted names. Thus, in the definition below, we will assume w.l.o.g. that the two frames ϕ_1 and ϕ_2 have the same set of restricted names.

Definition 1 (static equivalence). *We say that two frames $\phi_1 = \mathsf{new}\ \tilde{n}.\sigma_1$ and $\phi_2 = \mathsf{new}\ \tilde{n}.\sigma_2$ are* statically equivalent, *$\phi_1 \sim_{\mathsf{E}} \phi_2$, when $dom(\phi_1) = dom(\phi_2)$, and for all recipes M, N such that $names(M, N) \cap \tilde{n} = \emptyset$, we have that: $M\sigma_1 =_{\mathsf{E}} N\sigma_1$ if, and only if, $M\sigma_2 =_{\mathsf{E}} N\sigma_2$.*

Intuitively, two frames are equivalent when the attacker cannot see the difference between the two situations they represent, *i.e.*, his ability to distinguish whether two recipes M, N produce the same term does not depend on the frame.

Example 2. Let $\phi_{\mathsf{req}} = \mathsf{new}\, n.\{y_1 \triangleright \mathsf{senc}(\langle \mathsf{req}, n\rangle, k)\}$ and $\phi_{\mathsf{rep}} = \mathsf{new}\, n.\{y_1 \triangleright \mathsf{senc}(\langle \mathsf{rep}, n\rangle, k)\}$ be two frames. Considering the equational theory E introduced in Example 1, we have that $\phi_{\mathsf{req}} \not\sim_{\mathsf{E}} \phi_{\mathsf{rep}}$ since the recipes $M = \mathsf{proj}_1(\mathsf{sdec}(y_1, k))$ and $N = \mathsf{req}$ allow one to distinguish the two frames. However, we have that $\mathsf{new}\, k.\phi_{\mathsf{req}} \sim_{\mathsf{E}} \mathsf{new}\, k.\phi_{\mathsf{rep}}$. Indeed, without knowing the key k, the attacker is unable to observe the differences between the two messages. This is a non-trivial equivalence that can be established using an automatic tool (*e.g.* ProVerif [8]).

3 Models for protocols

In this section, we introduce the cryptographic process calculus that we will use for describing protocols. Several well-studied calculi already exist to analyse security protocols and privacy-type properties (*e.g.* [2, 1]). However, modelling ad-hoc routing protocols requires several additional features. Our calculus is actually inspired from some other calculi (*e.g.* [17, 6, 11]) which allow mobile wireless networks and their security properties to be formally described and analysed. We adapt those formalisms in order to be able to express privacy-type properties such as those studied in this paper.

3.1 Syntax

The intended behavior of each node of the network can be modelled by a *process* defined by the grammar given below (u is a term that may contain variables, n is a name, and Φ is a formula). Our calculus is parametrized by a set $\mathcal{L}$ of formulas whose purpose is to represent various tests performed by the agents (*e.g.* equality tests, neighbourhood tests). We left this set unspecified since it is not relevant for this work. For illustration purposes, we only assume that the set $\mathcal{L}$ contains at least equality and disequality tests.

$P, Q :=$	0	null process
	$\mathsf{in}(u).P$	reception
	$\mathsf{out}(u).P$	emission
	$\mathsf{if}\ \Phi\ \mathsf{then}\ P\ \mathsf{else}\ Q$	conditional $\Phi \in \mathcal{L}$
	$\mathsf{store}(u).P$	storage
	$\mathsf{read}\ u[\Phi]\ \mathsf{then}\ P\ \mathsf{else}\ Q$	reading
	$P \mid Q$	parallel composition
	$!P$	replication
	$\mathsf{new}\ n.P$	fresh name generation

The process "$\mathsf{in}(u).P$" expects a message m of the form u and then behaves like $P\sigma$ where σ is such that $m = u\sigma$. The process "$\mathsf{out}(u).P$" emits u, and then behaves like P. The variables that occur in u will be instantiated when the

evaluation will take place. The process $\mathsf{store}(u).P$ stores u in its storage list and then behaves like P. The process $\mathsf{read}\ u[\Phi]\ \mathsf{then}\ P\ \mathsf{else}\ Q$ looks for a message of the form u that satisfies Φ in its storage list and then, if such an element m is found, it behaves like $P\sigma$ where σ is such that $m = u\sigma$. Otherwise, it behaves like Q. The other operators are standard.

Sometimes, for the sake of clarity, we will omit the null process. We also omit the else part when $Q = 0$. We write $\mathit{fvars}(P)$ for the set of *free variables* that occur in P, *i.e.* the set of variables that are not in the scope of an input or a read. We consider *ground processes*, *i.e.* processes P such that $\mathit{fvars}(P) = \emptyset$, and *parametrized* processes, denoted $P(z_1, \ldots, z_n)$ where $z_1, \ldots, z_n$ are variables of sort agent, and such that $\mathit{fvars}(P) \subseteq \{z_1, \ldots, z_n\}$. A *routing protocol* is a set of parametrized processes.

3.2 Example: ANODR

ANODR is an anonymous on-demand routing protocol that has been designed to prevent traffic analysis in ad hoc networks [15]. We consider a simplified version of this protocol, denoted $\mathcal{P}^{\mathsf{simp}}_{\mathsf{ANODR}}$. For sake of readability, we give below an Alice and Bob version of this two-phase protocol where we omit some $\langle \cdots, \cdot \rangle$ and we use $\{\cdot\}_\cdot$ instead of senc and aenc.

$$
\begin{array}{l}
S \to V_1 : \langle \mathsf{req}, id, \{D, chall\}_{\mathsf{pub}(D)}, \{S, \mathsf{src}\}_{k_S} \rangle \\
V_1 \to V_2 : \langle \mathsf{req}, id, \{D, chall\}_{\mathsf{pub}(D)}, \{V_1, \{S, \mathsf{src}\}_{k_S}\}_{k_1} \rangle \\
V_2 \to D : \langle \mathsf{req}, id, \{D, chall\}_{\mathsf{pub}(D)}, \{V_2, \{V_1, \{S, \mathsf{src}\}_{k_S}\}_{k_1}\}_{k_2} \rangle \\
\\
D \to V_2 : \langle \mathsf{rep}, N_D, chall, \{V_2, \{V_1, \{S, \mathsf{src}\}_{k_S}\}_{k_1}\}_{k_2} \rangle \\
V_2 \to V_1 : \langle \mathsf{rep}, N_2, chall, \{V_1, \{S, \mathsf{src}\}_{k_S}\}_{k_1} \rangle \\
V_1 \to S : \langle \mathsf{rep}, N_1, chall, \{S, \mathsf{src}\}_{k_S} \rangle
\end{array}
$$

Request phase. The source initiates route discovery by locally broadcasting a request. The constant req is used to identify the request phase whereas id is an identifier of the request. The third component of the request is a cryptographic trapdoor that can only be opened by the destination; and the last one is a cryptographic onion that is used for route establishment. At this stage, the onion built by the source contains only one layer.

Then, intermediate nodes relay the request over the network, except if they have already seen it. However, contrary to what happen in many routing protocols, the names of the intermediate nodes are not accumulated in the route request packet. This is important to prevent traffic analysis.

Reply phase. When the destination D receives the request, it opens the trapdoor and builds a route reply.

During the reply phase, the message travels along the route back to S. The intermediary node decrypt the onion using its own key which has been generated during the request phase. If its own identity does not match the first field of the decrypted result, it then discards the packet. Otherwise, the node is on the anonymous route. It generates a random number (namely N_D, N_1, or N_2), stores

the correspondence between the nonce it receives and the one it has generated. It peels off one layer of the onion, replaces the nonce with its own nonce, and then locally broadcasts the reply packet.

Formally, this protocol is composed of four parametrized processes that can be modelled using the signature given in Example 1. Let id be a name, z_S, z_V, z_D be variables of sort agent, and x_N, x_{id}, x_{tr} and x_{onion} be variables of sort msg.

The process executed by the agent z_S initiating the search of a route towards a node z_D is:

$$P_{\mathsf{src}}(z_S, z_D) = \mathsf{new}\ id.\mathsf{new}\ chall.\mathsf{new}\ k_S.\mathsf{out}(u_1).\mathsf{in}(u_2).\mathsf{store}(\langle z_D, x_N\rangle)$$

$$\text{where } \begin{cases} u_1 = \langle \mathsf{req}, id, \mathsf{aenc}(\langle z_D, chall\rangle, \mathsf{pub}(z_D)), \mathsf{senc}(\langle z_S, \mathsf{src}\rangle, k_S)\rangle \\ u_2 = \langle \mathsf{rep}, x_N, chall, \mathsf{senc}(\langle z_S, \mathsf{src}\rangle, k_S)\rangle \end{cases}$$

The source z_S builds a request message and sends it. Then, the source is waiting for a reply containing the same cryptographic onion as the one used in the request, a proof of global trapdoor opening (here modelled as a nonce $chall$), and a locally unique random route pseudonym N. Lastly, the source will store that destination D can be reached using the route pseudonym N as the next hop.

The process executed by an intermediary node z_V during the request phase is described below. For sake of simplicity, we did not model the fact that a duplicated request message is directly discarded.

$$P^{\mathsf{req}}_{\mathsf{int}}(z_V) = \mathsf{in}(w_1).\mathsf{if}\ \neg\Phi_{\mathsf{req}}\ \mathsf{then}\ (\mathsf{new}\ k_V.\mathsf{store}(\langle \mathsf{key}, k_V\rangle).\mathsf{out}(w_2))$$

$$\text{where } \begin{cases} w_1 = \langle \mathsf{req}, x_{id}, x_{tr}, x_{onion}\rangle \qquad \Phi_{\mathsf{req}} = \mathsf{proj}_1(\mathsf{adec}(x_{tr}, \mathsf{prv}(z_V))) = z_V \\ w_2 = \langle \mathsf{req}, x_{id}, x_{tr}, \mathsf{senc}(\langle z_V, x_{onion}\rangle, k_V)\rangle \end{cases}$$

The process executed by the destination node z_D is the following:

$$P_{\mathsf{dest}}(z_D) = \mathsf{in}(v_1).\ \mathsf{if}\ \Phi_{\mathsf{dest}}\ \mathsf{then}\ (\mathsf{new}\ N.\mathsf{out}(v_2))$$

$$\text{where } \begin{cases} v_1 = \langle \mathsf{req}, x_{id}, x_{tr}, x_{onion}\rangle \qquad \Phi_{\mathsf{dest}} = \mathsf{proj}_1(\mathsf{adec}(x_{tr}, \mathsf{prv}(z_D))) = z_D \\ v_2 = \langle \mathsf{rep}, N, \mathsf{proj}_2(\mathsf{adec}(x_{tr}, \mathsf{prv}(z_D))), x_{onion}\rangle \end{cases}$$

The process executed by an intermediary node z_V during the reply phase is as follows:

$$P^{\mathsf{rep}}_{\mathsf{int}}(z_V) = \mathsf{in}(w'_1).\mathsf{read}\ \langle \mathsf{key}, y\rangle\ [\Phi_{\mathsf{rep}}]\ \mathsf{then}\ (\mathsf{new}\ N'.\mathsf{store}(\langle x_N, N'\rangle).\mathsf{out}(w'_2))$$

$$\text{where } \begin{cases} w'_1 = \langle \mathsf{rep}, x_N, x_{pr}, x_{onion}\rangle \qquad \Phi_{\mathsf{rep}} = \mathsf{proj}_1(\mathsf{sdec}(x_{onion}, y)) = z_V \\ w'_2 = \langle \mathsf{rep}, N', x_{pr}, \mathsf{proj}_2(\mathsf{sdec}(x_{onion}, y))\rangle \end{cases}$$

Once, a route between S and D has been established using this protocol, a data packet can then be sent from S to D using the route pseudonyms that nodes have stored in their storage list.

3.3 Configuration and topology

Each process is located at a node of the network, and we consider an eavesdropper who observes messages sent from particular nodes. More precisely, we assume that the *topology* of the network is represented by a pair $\mathcal{T} = (G, \mathcal{M})$ where:

- $G = (V, E)$ is an undirected finite graph with $V \subseteq \{A \in \mathcal{N} \mid A \text{ of sort } \mathsf{agent}\}$, where an edge in the graph models the fact that two agents are neighbors.
- $\mathcal{M}$ is a set of nodes, the *malicious nodes*, from which the attacker is able to listen to their outputs.

We consider several malicious nodes, and our setting allows us to deal with the case of a global eavesdropper (*i.e.* $\mathcal{M} = V$). A *trivial topology* is a topology $\mathcal{T} = (G, \mathcal{M})$ with $\mathcal{M} = \emptyset$.

A *configuration* of the network is a quadruplet $(\mathcal{E}; \mathcal{P}; \mathcal{S}; \sigma)$ where:

- $\mathcal{E}$ is a finite set of names that represents the names restricted in $\mathcal{P}$, $\mathcal{S}$ and σ;
- $\mathcal{P}$ is a multiset of expressions of the form $\lfloor P \rfloor_A$ that represents the process P executed by the agent $A \in V$. We write $\lfloor P \rfloor_A \cup \mathcal{P}$ instead of $\{\lfloor P \rfloor_A\} \cup \mathcal{P}$.
- $\mathcal{S}$ is a set of expressions of the form $\lfloor u \rfloor_A$ with $A \in V$ and u a ground term. $\lfloor u \rfloor_A$ represents the term u stored by the agent $A \in V$.
- $\sigma = \{y_1 \triangleright u_1, \ldots, y_n \triangleright u_n\}$ where $u_1, \ldots, u_n$ are ground terms (the messages observed by the attacker), and $y_1, \ldots, y_n$ are variables.

3.4 Execution model

Each node broadcasts its messages to all its neighbors. The communication system is formally defined by the rules of Figure 1. They are parametrized by the underlying topology $\mathcal{T}$. The Comm rule allows nodes to communicate provided they are (directly) connected in the underlying graph, without the attacker actively interfering. We do not assume that messages are necessarily delivered to the intended recipients. They may be lost. The exchange message is learnt by the attacker as soon as the node that emits it is under its scrutiny.

The other rules are quite standard.

We write $\rightarrow$ instead of $\rightarrow_{\mathcal{T}}$ when the underlying network topology $\mathcal{T}$ is clear from the context. Let $\mathcal{A}$ be the alphabet of actions where the special symbol $\tau \in \mathcal{A}$ represents an unobservable action. For every $\ell \in \mathcal{A}$, the relation $\xrightarrow{\ell}$ has been defined in Figure 1. For every $w \in \mathcal{A}^*$ the relation $\xrightarrow{w}$ on configurations is defined in the usual way. By convention $K \xrightarrow{\epsilon} K$ where ϵ denotes the empty word. For every $s \in (\mathcal{A} \setminus \{\tau\})^*$, the relation $\stackrel{s}{\Longrightarrow}$ on configurations is defined by: $K \stackrel{s}{\Longrightarrow} K'$ if, and only if, there exists $w \in \mathcal{A}^*$ such that $K \xrightarrow{w} K'$ and s is obtained from w by erasing all occurrences of τ. Intuitively, $K \stackrel{s}{\Longrightarrow} K'$ means that K transforms into K' by experiment s.

An *initial configuration associated to a topology* $\mathcal{T} = (G, \mathcal{M})$ *and a routing protocol* $\mathcal{P}_{\mathsf{routing}}$ is a configuration $K_0 = (\mathcal{E}_0; \mathcal{P}_0; \mathcal{S}_0; \sigma_0)$ such that:

$$\mathcal{P}_0 = \bigcup_{\substack{P \in \mathcal{P}_{\mathsf{routing}} \\ A, B_1, \ldots, B_k \in V}} \lfloor !P(A, B_1, \ldots, B_k) \rfloor_A.$$

$$\textsc{Comm}\quad (\mathcal{E};\lfloor \mathsf{out}(t).P\rfloor_A \cup \{\lfloor \mathsf{in}(u_j).P_j\rfloor_{A_j} \mid mgu(t,u_j)\neq \bot \wedge (A,A_j)\in E\}\cup\mathcal{P};\mathcal{S};\sigma) \xrightarrow{\ell}_{\mathcal{T}} (\mathcal{E};\{\lfloor P_j\sigma_j\rfloor_{A_j}\}\cup\lfloor P\rfloor_A\cup\mathcal{P};\mathcal{S};\sigma')$$

$$\text{where } \begin{cases} \sigma_j{=}mgu(t,u_j) \\ \sigma' = \sigma\cup\{y\triangleright t\} \text{ where } y \text{ is a fresh variable and } \ell=(\mathsf{out}(y),A) \text{ if } A\in\mathcal{M}; \\ \sigma'=\sigma \text{ and } \ell=\tau \text{ otherwise}\end{cases}$$

$$\textsc{Store}\quad (\mathcal{E};\lfloor \mathsf{store}(t).P\rfloor_A\cup\mathcal{P};\mathcal{S};\sigma)\xrightarrow{\tau}_{\mathcal{T}}(\mathcal{E};\lfloor P\rfloor_A\cup\mathcal{P};\lfloor t\rfloor_A\cup\mathcal{S};\sigma)$$

$$\textsc{Read-Then}\quad (\mathcal{E};\lfloor \mathsf{read}\ u[\Phi]\ \mathsf{then}\ P\ \mathsf{else}\ Q\rfloor_A\cup\mathcal{P};\lfloor t\rfloor_A\cup\mathcal{S};\sigma)\xrightarrow{\tau}_{\mathcal{T}}(\mathcal{E};\lfloor P\lambda\rfloor_A\cup\mathcal{P};\lfloor t\rfloor_A\cup\mathcal{S};\sigma)$$
when $\lambda = mgu(t,u)$ exists and $\Phi\lambda$ is evaluated to true

$$\textsc{Read-Else}\quad (\mathcal{E};\lfloor \mathsf{read}\ u[\Phi]\ \mathsf{then}\ P\ \mathsf{else}\ Q\rfloor_A\cup\mathcal{P};\mathcal{S};\sigma)\xrightarrow{\tau}_{\mathcal{T}}(\mathcal{E};\lfloor Q\rfloor_A\cup\mathcal{P};\mathcal{S};\sigma)$$
if for all t such that $\lfloor t\rfloor_A\in\mathcal{S}$, $mgu(t,u)=\bot$ or $\Phi mgu(t,u)$ is evaluated to false

$$\textsc{If-Then}\quad (\mathcal{E};\lfloor \mathsf{if}\ \Phi\ \mathsf{then}\ P\ \mathsf{else}\ Q\rfloor_A\cup\mathcal{P};\mathcal{S};\sigma)\xrightarrow{\tau}_{\mathcal{T}}(\mathcal{E};\lfloor P\rfloor_A\cup\mathcal{P};\mathcal{S};\sigma)$$
if Φ is evaluated to true

$$\textsc{If-Else}\quad (\mathcal{E};\lfloor \mathsf{if}\ \Phi\ \mathsf{then}\ P\ \mathsf{else}\ Q\rfloor_A\cup\mathcal{P};\mathcal{S};\sigma)\xrightarrow{\tau}_{\mathcal{T}}(\mathcal{E};\lfloor Q\rfloor_A\cup\mathcal{P};\mathcal{S};\sigma)$$
if Φ is evaluated to false

$$\textsc{Par}\quad (\mathcal{E};\lfloor P_1 \mid P_2\rfloor_A\cup\mathcal{P};\mathcal{S};\sigma)\xrightarrow{\tau}_{\mathcal{T}}(\mathcal{E};\lfloor P_1\rfloor_A\cup\lfloor P_2\rfloor_A\cup\mathcal{P};\mathcal{S};\sigma)$$

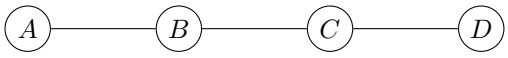
$$\textsc{Repl}\quad (\mathcal{E};\lfloor !P\rfloor_A\cup\mathcal{P};\mathcal{S};\sigma)\xrightarrow{\tau}_{\mathcal{T}}(\mathcal{E};\lfloor P\rfloor_A\cup\lfloor !P\rfloor_A\cup\mathcal{P};\mathcal{S};\sigma)$$

$$\textsc{New}\quad (\mathcal{E};\lfloor \mathsf{new}\ n.P\rfloor_A\cup\mathcal{P};\mathcal{S};\sigma)\xrightarrow{\tau}_{\mathcal{T}}(\mathcal{E}\cup\{n'\};\lfloor P\{^{n'}/_n\}\rfloor_A\cup\mathcal{P};\mathcal{S};\sigma)$$
where n' is a fresh name

Fig. 1. Transition system.

Such a configuration represents the fact that each node can play any role of the protocol an unbounded number of times. Moreover, the agent who executes the process is located at the right place. A typical initial configuration will consist of $\mathcal{E}_0=\mathcal{S}_0=\sigma_0=\emptyset$, but depending on the protocol under study, we may want to populate the storage lists of some nodes.

Example 3. Let $\mathcal{T}_0=(G_0,\mathcal{M}_0)$ be a topology where G_0 is described below, and consider a global eavesdropper, *i.e.* $\mathcal{M}_0=\{A,B,C,D\}$.



We consider the execution of the protocol $\mathcal{P}^{\mathsf{simp}}_{\mathsf{ANODR}}$ where B acts as a source to obtain a route to D. Receiving this request, and not being the destination, its neighbour C acts as a request forwarding node. We have that:

$$\mathsf{tr}=K_0\xrightarrow{\tau}\xrightarrow{\tau}\xrightarrow{\tau}\xrightarrow{\mathsf{out}(y_1),B}(\mathcal{E}_1;\mathcal{P}_1;\mathcal{S}_1;\sigma_1)\xrightarrow{\tau}\xrightarrow{\tau}\xrightarrow{\tau}\xrightarrow{\mathsf{out}(y_2),C}(\mathcal{E}_2;\mathcal{P}_2;\mathcal{S}_2;\sigma_2)$$

where: $\begin{cases} K_0 = (\emptyset; \mathcal{P}_0; \emptyset; \emptyset) \text{ initial configuration associated to } \mathcal{T}_0 \text{ and } \mathcal{P}^{\mathsf{simp}}_{\mathsf{ANODR}}. \\ \mathcal{E}_1 = \{id, chall, k_B\} \quad \mathcal{E}_2 = \{id, chall, k_B, k_C\} \\ \mathcal{S}_1 = \emptyset \qquad\qquad\quad \mathcal{S}_2 = \{\lfloor \langle \mathsf{key}, k_C \rangle \rfloor_C\} \\ \sigma_1 = \{y_1 \triangleright u\} \qquad\quad \sigma_2 = \{y_1 \triangleright u, y_2 \triangleright v\} \\ u = \langle \mathsf{req}, id, \mathsf{aenc}(\langle D, chall \rangle, \mathsf{pub}(D)), \mathsf{senc}(\langle B, \mathsf{src} \rangle, k_B) \rangle \\ v = \langle \mathsf{req}, id, \mathsf{aenc}(\langle D, chall \rangle, \mathsf{pub}(D)), \mathsf{senc}(\langle C, \mathsf{senc}(\langle B, \mathsf{src} \rangle, k_B) \rangle, k_C) \rangle \end{cases}$

The process $\lfloor P_{\mathsf{src}}(B, D) \rfloor_B$ that occurs in K_0 will first follow the rule New three times to generate the nonces id, $chall$ and k_B leading to a new set of restricted names $\mathcal{E}_1$. The rule Comm is then applied between nodes B and C. As $B \in \mathcal{M}_0$, the message is included in σ_1 to represent the knowledge gained by the attacker. As the node C is not the destination, $\lfloor P^{\mathsf{req}}_{\mathsf{int}}(C) \rfloor_C$ can evolve (rule If-Then). It generates a key (rule New) added in $\mathcal{E}_2$, and stores it in $\mathcal{S}_2$ (rule Store) and finally it uses Comm to broadcast the resulting message, which is also added to current substitution σ_2. Actually, in case we are only interested by the visible actions, this trace tr could also be written as follows:

$$\mathsf{tr} = K_0 \xRightarrow{\mathsf{out}(y_1), B} (\mathcal{E}_1; \mathcal{P}_1; \mathcal{S}_1; \sigma_1) \xRightarrow{\mathsf{out}(y_2), C} (\mathcal{E}_2; \mathcal{P}_2; \mathcal{S}_2; \sigma_2).$$

3.5 Extension and equivalence of traces

We cannot expect that privacy-type properties hold in any situation. We have to ensure that the traffic is sufficient. For this we need to introduce the notion of *extension of a trace*. Roughly, we say that a trace tr^+ is an extension of a trace tr if tr^+ contains at least all the actions that are exhibited in tr. In order to track of the actions, we consider annotated traces. This need comes from the fact that our calculus (and many others cryptographic calculi) does not provide us with information that allow us to retrieve who performed a given action.

We will denote $K \xrightarrow[A,\mathsf{R}]{\tau} K'$ (resp. $K \xrightarrow[A,\mathsf{R}]{\mathsf{out}(y),A} K'$) instead of $K \xrightarrow{\tau} K'$ (resp. $K \xrightarrow{\mathsf{out}(y),A} K'$) to explicit the annotations. We have that $A \in V$ and R is a constant. Intuitively A is the node that performs the action (resp. the output) whereas R is a constant that represents the role who is responsible of this action (resp. output). Thus, to formalise this notion of annotated trace, we associate a constant to each parametrized process part of the routing protocol under study. Theses annotations are nonetheless invisible to the attacker: she has only access to the labels of the transitions defined in our semantics. Annotations are meant to be used to specify privacy properties.

Example 4. Going back to our running example, $\mathcal{P}^{\mathsf{simp}}_{\mathsf{ANODR}}$ is made up of 4 roles and we associate a constant to each of them, namely Src, Req, Dest, and Rep. The annotated version of the trace tr described in Example 3 is:

$$K_0 \xrightarrow[B,\mathsf{Src}]{\tau} \xrightarrow[B,\mathsf{Src}]{\tau} \xrightarrow[B,\mathsf{Src}]{\tau} \xrightarrow[B,\mathsf{Src}]{\mathsf{out}(y_1),B} K_1 \xrightarrow[C,\mathsf{Req}]{\tau} \xrightarrow[C,\mathsf{Req}]{\tau} \xrightarrow[C,\mathsf{Req}]{\tau} \xrightarrow[C,\mathsf{Req}]{\mathsf{out}(y_2),C} K_2$$

with $K_1 = (\mathcal{E}_1; \mathcal{P}_1; \mathcal{S}_1; \sigma_1)$ and $K_2 = (\mathcal{E}_2; \mathcal{P}_2; \mathcal{S}_2; \sigma_2)$.

Given two configurations $K = (\mathcal{E}; \mathcal{P}; \mathcal{S}; \sigma)$ and $K^+ = (\mathcal{E}^+; \mathcal{P}^+; \mathcal{S}^+; \sigma^+)$, we write $K \subseteq K^+$ if $\mathcal{E} \subseteq \mathcal{E}^+$, $\mathcal{P} \subseteq \mathcal{P}^+$, $\mathcal{S} \subseteq \mathcal{S}^+$, and $\sigma^+_{|dom(\sigma)} = \sigma$.

Definition 2 (extension of a trace). *Let* tr^+ *be an annotated trace:*

$$\mathsf{tr}^+ = K_0 \xrightarrow[A_1,\mathsf{R}_1]{\ell_1} K_1^+ \xrightarrow[A_2,\mathsf{R}_2]{\ell_2} \ldots \xrightarrow[A_n,\mathsf{R}_n]{\ell_n} K_n^+.$$

We say that tr^+ *is an* extension of tr*, denoted* $\mathsf{tr} \preccurlyeq \mathsf{tr}^+$*, if*

$$\mathsf{tr} = K_0 \xrightarrow[A_{k_1},\mathsf{R}_{k_1}]{\ell_{k_1}} K_{k_1} \xrightarrow[A_{k_2},\mathsf{R}_{k_2}]{\ell_{k_2}} \ldots \xrightarrow[A_{k_\ell},\mathsf{R}_{k_\ell}]{\ell_{k_\ell}} K_{k_\ell}$$

where $0 < k_1 < k_2 < \ldots < k_\ell \leq n$*, and* $K_{k_i} \subseteq K^+_{k_i}$ *for each* $i \in \{1, \ldots, \ell\}$.

Given an indice i *corresponding to an action in* tr *(*$1 \leq i \leq \ell$*), we denote by* $\mathsf{ind}_i(\mathsf{tr}, \mathsf{tr}^+)$ *the indice of the corresponding action in* tr^+*, i.e.* $\mathsf{ind}_i(\mathsf{tr}, \mathsf{tr}^+) = k_i$.

Example 5. An extension of the trace tr described in Example 3 could be to let A initiate a new session before B tries to discover a route to D. Such an execution is formalised by the trace tr^+ written below:

$$K_0 \xrightarrow[A,\mathsf{Src}]{\tau} \xrightarrow[A,\mathsf{Src}]{\tau} \xrightarrow[A,\mathsf{Src}]{\tau} \xrightarrow[A,\mathsf{Src}]{\mathsf{out}(y_0),A} K_0^+ \xrightarrow[B,\mathsf{Src}]{\tau} \xrightarrow[B,\mathsf{Src}]{\tau} \xrightarrow[B,\mathsf{Src}]{\tau} \xrightarrow[B,\mathsf{Src}]{\mathsf{out}(y_1),B} K_1^+$$
$$\xrightarrow[C,\mathsf{Req}]{\tau} \xrightarrow[C,\mathsf{Req}]{\tau} \xrightarrow[C,\mathsf{Req}]{\tau} \xrightarrow[C,\mathsf{Req}]{\mathsf{out}(y_2),C} K_2^+.$$

where the configurations are not detailed, but $(\mathcal{E}_i; \mathcal{P}_i; \mathcal{S}_i; \sigma_i) \subseteq K_i^+$ $(i \in \{1, 2\})$.

Privacy-type security properties are often formalised using a notion equivalence (see *e.g.* [12, 3, 9]). Here, we consider the notion of *equivalence between two traces*.

Definition 3 (equivalence of two traces). *Let* $\mathsf{tr}_1 = K_1 \stackrel{s_1}{\Longrightarrow} (\mathcal{E}_1; \mathcal{P}_1; \mathcal{S}_1; \sigma_1)$ *and* $\mathsf{tr}_2 = K_2 \stackrel{s_2}{\Longrightarrow} (\mathcal{E}_2; \mathcal{P}_2; \mathcal{S}_2; \sigma_2)$ *be two traces. They are* equivalent*, denoted* $\mathsf{tr}_1 \approx_{\mathsf{E}} \mathsf{tr}_2$*, if* $s_1 = s_2$ *and* $\mathsf{new}\, \mathcal{E}_1.\sigma_1 \sim_{\mathsf{E}} \mathsf{new}\, \mathcal{E}_2.\sigma_2$.

Note that only observable actions are taken into account in the definition of equivalence between two traces. Roughly, two traces are equivalent if they process the same sequence of visible outputs. The two sequences may differ (we do *not* require the equality between σ_1 and σ_2) but they should be indistinguishable from the point of view of the attacker.

Example 6. In the execution tr^+ provided in Example 5 one could hope to hide the fact that the node B is initiating a route discovery and let the attacker think A is the actual source. Let tr' be the execution below where A initiates a route discovery towards D, while nodes B and C act as forwarders.

$$K_0 \xrightarrow[A,\mathsf{Src}]{\tau} \xrightarrow[A,\mathsf{Src}]{\tau} \xrightarrow[A,\mathsf{Src}]{\tau} \xrightarrow[A,\mathsf{Src}]{\mathsf{out}(y_0),A} K_0' \xrightarrow[B,\mathsf{Req}]{\tau} \xrightarrow[B,\mathsf{Req}]{\tau} \xrightarrow[B,\mathsf{Req}]{\tau} \xrightarrow[B,\mathsf{Req}]{\mathsf{out}(y_1),B} K_1'$$
$$\xrightarrow[C,\mathsf{Req}]{\tau} \xrightarrow[C,\mathsf{Req}]{\tau} \xrightarrow[C,\mathsf{Req}]{\tau} \xrightarrow[C,\mathsf{Req}]{\mathsf{out}(y_2),C} K_2'.$$

where the configurations are not detailed.
Unfortunately the attacker is able to tell the difference between tr^+ and tr'. Indeed, we have $\mathsf{tr}^+ \not\approx_{\mathsf{E}} \mathsf{tr}'$ since the test $\mathsf{proj}_2(\mathsf{proj}_1(y_0)) \stackrel{?}{=} \mathsf{proj}_2(\mathsf{proj}_1(y_1))$ can be used to distinguish the two traces. The equality test will hold in tr' and not in tr^+. Note that, as the annotations are invisible to the attacker, she cannot know *a priori* that B is playing a forwarder in tr'.

4 Indistinguishability

Intuitively, indistinguishability deals with the ability for the attacker to distinguish a specific action from another. For a routing protocol such actions take the form of the various roles of the protocol. In particular we could hope, in an execution of the protocol, to make actions of the initiator or recipient indistinguishable from actions of forwarding nodes. Our definition of indistinguishability, and later of other privacy properties, depends on the network topology we are considering. Incidentally, when designing anonymous protocols, these properties should hold for large enough classes of topologies.

4.1 Formalizing indistinguishability

Let Roles be a set of roles for which indistinguishability has to be preserved. A very naive definition would be to ensure that for any annotated trace tr issued from K_0 (the initial configuration associated to the protocol under study) where at some position i the role $\mathsf{R} \in \mathsf{Roles}$ is played and observed by the attacker, there exists an equivalent annotated trace tr' where the role played at position i is not in the set Roles. However, without appropriate traffic on the network, this definition is far too strong. Indeed, as soon as the source role is the only role able to spontaneously start a session, we will have no hope to achieve indistinguishability.

Definition 4 (indistinguishability). *Let K_0 be an initial configuration associated to a routing protocol and a topology, and* Roles *be a set of roles. We say that K_0 preserves* indistinguishability w.r.t. Roles *if for any annotated trace* tr

$$\mathsf{tr} = K_0 \xrightarrow[A_1,\mathsf{R}_1]{\ell_1} K_1 \xrightarrow[A_2,\mathsf{R}_2]{\ell_2} \dots \xrightarrow[A_n,\mathsf{R}_n]{\ell_n} K_n = (\mathcal{E};\mathcal{P};\mathcal{S};\sigma)$$

*and for any $i \in \{1,\dots,n\}$ such that $\mathsf{R}_i \in \mathsf{Roles}$ and $\ell_i \neq \tau$ (*i.e. *ℓ_i is an action observed by the attacker), there exist two annotated traces tr^+ and tr' such that: $\mathsf{tr} \preccurlyeq \mathsf{tr}^+$, $\mathsf{tr}^+ \approx \mathsf{tr}'$, and $\mathsf{R}'_{\mathsf{ind}_i(\mathsf{tr},\mathsf{tr}^+)} \notin \mathsf{Roles}$ where*

$$\mathsf{tr}' = K'_0 \xrightarrow[A'_1,\mathsf{R}'_1]{\ell'_1} K'_1 \xrightarrow[A'_2,\mathsf{R}'_2]{\ell'_2} K'_2 \dots \xrightarrow[A'_{n'},\mathsf{R}'_{n'}]{\ell'_{n'}} K'_{n'}.$$

The trace tr^+ enables us to deal with the aforementioned traffic needed to aim at preserving indistinguishability. Indeed rather than imposing the equivalence of tr with another trace, indistinguishability will be achieved if there exist two other traces tr^+ and tr' which look the same to the attacker, and in which the action of interest is played by a different role.

4.2 Analysis of ANODR

Now, we apply our formalisation of indistinguishability to the ANODR protocol.

Proposition 1. *Let $\mathcal{T}$ be a topology with a malicious node that has only malicious neighbours, and K_0 be an initial configuration associated to $\mathcal{P}^{\mathsf{simp}}_{ANODR}$ and $\mathcal{T}$. We have that K_0 does not preserve indistinguishability w.r.t.* Src *(resp.* Dest*).*

Indeed, given a node A which is, together with its neighbors, under the scrutiny of the attacker, consider a situation, *i.e.* a trace tr, where the node A starts a new session by acting as a source. Of course, if this action is the only activity of the network, there is no hope to confuse the attacker. The idea is to see whether the attacker can be confused when the traffic is sufficient. In particular, we may want to consider a situation, *i.e.* a trace tr^+, where a request also arrives at node A at the same time, so that the node A has also the possibility to act as a forwarder. However, since a request conveys a unique identifier id, it will be easy for the attacker to observe whether A is acting as a source (the request will contain a fresh identifier) or as a forwarder (the request will contain an identifier that has been previously observed by the attacker). Actually, the same reasoning allows us to conclude that indistinguishability is not preserved w.r.t. the role Dest: a reply conveys a globally unique nonce (namely *chall*).

The updated version of ANODR proposed in [15] and informally described below (see the appendice for a formal description) fixes the issue regarding indistinguishability w.r.t. $\mathsf{Roles} = \{\mathsf{Dest}\}$. In this version, K_T is a symmetric encryption key shared between the source A and the destination D; K_A, K_B and K_C are symmetric keys known only to their owners A, B, C, whereas K_{seedB}, K_{seedC}, K_{seedD} are fresh keys shared between consecutive nodes on the reply route. The key K_D is generated by A and will be known by every node on the route by the end of a session. The routes are stored as a correspondence between route pseudonyms (the N_i) by each intermediate node. The proof of opening takes the form of the key K_D which is embedded in an onion which is different from the onions used during the request phase. For sake of clarity, we use $\{\cdot\}_\cdot$ instead of senc and aenc, and we omit some $\langle\cdot,\cdot\rangle$.

$$
\begin{array}{l}
A \to B : \langle \mathsf{req}, id, \mathsf{pub}(A), \{\mathsf{dest}, K_D\}_{K_T}, \{\mathsf{dest}\}_{K_D}, \{\mathsf{src}\}_{K_A} \rangle \\
B \to C : \langle \mathsf{req}, id, \mathsf{pub}(B), \{\mathsf{dest}, K_D\}_{K_T}, \{\mathsf{dest}\}_{K_D}, \{N_B, \{\mathsf{src}\}_{K_A}\}_{K_B} \rangle \\
C \to D : \langle \mathsf{req}, id, \mathsf{pub}(C), \{\mathsf{dest}, K_D\}_{K_T}, \{\mathsf{dest}\}_{K_D}, \{N_C, \{N_B, \{\mathsf{src}\}_{K_A}\}_{K_B}\}_{K_C} \rangle \\
D \to C : \langle \mathsf{rep}, \{K_{seedD}\}_{\mathsf{pub}(C)}, \{K_D, \{N_C, \{N_B, \{\mathsf{src}\}_{K_A}\}_{K_B}\}_{K_C}\}_{K_{seedD}} \rangle \\
C \to B : \langle \mathsf{rep}, \{K_{seedC}\}_{\mathsf{pub}(B)}, \{K_D, \{N_B, \{\mathsf{src}\}_{K_A}\}_{K_B}\}_{K_{seedC}} \rangle \\
B \to A : \langle \mathsf{rep}, \{K_{seedB}\}_{\mathsf{pub}(A)}, \{K_D, \{\mathsf{src}\}_{K_A}\}_{K_{seedB}} \rangle
\end{array}
$$

Considering a topology $\mathcal{T}$ such that any malicious node has at least two distinct neighbours other than itself, and an initial configuration K_0 associated to the updated version of ANODR and $\mathcal{T}$, we have that K_0 preserves indistinguishability w.r.t. $\mathsf{Roles} = \{\mathsf{Dest}\}$, according to Definition 4.

Intuitively, for each trace tr in which the node A (under the scrutiny of the attacker) acts as a destination, we will consider a trace tr^+ which extends tr and such that the node A has at least two reply to treat (one as a destination and

one as a forwarder). Since the proof of opening and the onion are modified at each hop of the route, the attacker will not be able to observe whether two reply packets come from the same session or not. Thus, he can not be sure that the action of interest has been done by the role Dest.

5 Unlinkability

We focus here on a different kind of anonymity: the (un)ability for the attacker to determine whether two messages belong to the same session. Note that an attacker able to determine whether two reply messages belong to the same session will gain valuable information about the route being established.

5.1 Augmented process

To define unlinkability, we need a notion of session. Note that, in our setting, a session may involve an arbitrary number of actions since we do not know in advance the length of the path from the source to the destination. In order to define this notion formally, we need to be able to track an execution of the process through the entire network, goal which is achieved through a notion of *augmented processes*. Thus, given a routing protocol $\mathcal{P}_{\mathsf{routing}}$, we define its augmentation $\tilde{\mathcal{P}}_{\mathsf{routing}}$ and modify the operational semantics accordingly to trace an execution of one session of the protocol. We also add some information about the source and the destination. This information will be useful later on to define our notion of anonymity (see Section 6).

For sake of simplicity, we consider a routing protocol that is made up of parametrized processes of two different kinds. Even if these syntactic restrictions seem to be very specific, our definition actually captures most of the routing protocols and are quite natural.

Initiator: a parametrized process with two parameters $P(z_S, z_D)$ such that its first communication action is an output possibly followed by several inputs. In such a case, its *augmentation* $\tilde{P}(z_S, z_D)$ is obtained from $P(z_S, z_D)$ by adding the prefix $\mathsf{new}\, sid.$ to it, by replacing the action $\mathsf{out}(u)$ with $\mathsf{out}(\langle u, \langle sid, z_S, z_D\rangle\rangle)$, and replacing each action $\mathsf{in}(u)$ with $\mathsf{in}(\langle u, \langle x_1, x_2, x_3\rangle\rangle)$ where x_1, x_2, x_3 are fresh variables.

Responder: a parametrized process with one parameter $P(z_V)$ such that its first communication action is an input possibly followed by several outputs. In such a case, its *augmentation* $\tilde{P}(z_V)$ is obtained from $P(z_V)$ by replacing the action $\mathsf{in}(u)$ with $\mathsf{in}(\langle u, \langle x_1, x_2, x_3\rangle\rangle)$ where x_1, x_2, x_3 are fresh variables, and each action $\mathsf{out}(u)$ with $\mathsf{out}(\langle u, \langle x_1, x_2, x_3\rangle\rangle)$.

Now, to prevent the additional information that is conveyed by the messages to occur in the frame, we need to adapt our operational semantics. Basically, when we perform a communication, we only add the first projection of the outputted term in the frame. The second projection of the outputted term is added under the arrow as an annotation.

Example 7. Back to Example 3, the counterpart of the trace tr, where only visible actions have been exhibited, is succinctly depicted below:

$$\tilde{K}_0 \xRightarrow[B,\mathsf{Src},sid,B,D]{\mathsf{out}(y_1),B} \tilde{K}_1 \xRightarrow[C,\mathsf{Req},sid,B,D]{\mathsf{out}(y_2),C} \tilde{K}_2$$

where the configurations $\tilde{K}_0$, $\tilde{K}_1$ and $\tilde{K}_2$ are the counterpart of K_1, K_2, and K_3. The annotations under the arrows witness the fact that the two messages come from the same session *sid* which was initiated by B to obtain a route towards D.

Note that only observable action will benefit from this annotation. For sake of simplicity, we write $K \xrightarrow[A,\mathsf{R},[sid,S,D]]{\ell} K'$ even in presence of an unobservable action ℓ (*i.e.* when $\ell = \tau$) and we add the brackets to emphasize the fact that $[sid, S, D]$ is optional. Actually, the annotation is undefined in this case.

5.2 Formalising unlinkability

Intuitively, unlinkability means that an attacker cannot tell whether two visible actions of a trace tr belong to the same session. As it was the case for indistinguishability, one cannot expect to achieve this goal without any sufficient traffic on the network. Moreover, due to the globally unique identifier that occur for efficiency purposes in many routing protocols (*e.g.* the nonce *id* in ANODR), there is no hope to achieve unlinkability for request messages. However, this is not a big issue since these messages are flooded in the network and thus tracking them is useless. We may want to study unlinkability for particular sets of roles, and our definition allows one to do that.

Definition 5 (unlinkability). *Let K_0 be an initial configuration associated to a routing protocol and a topology, and Roles_1, Roles_2 be two sets of roles. We say that K_0 preserves* unlinkability w.r.t. $\mathsf{Roles}_1/\mathsf{Roles}_2$ *if for any annotated trace* tr

$$\mathsf{tr} = K_0 \xrightarrow[A_1,\mathsf{R}_1,[sid_1,S_1,D_1]]{\ell_1} K_1 \xrightarrow[A_2,\mathsf{R}_2,[sid_2,S_2,D_2]]{\ell_2} \cdots \xrightarrow[A_n,\mathsf{R}_n,[sid_n,S_n,D_n]]{\ell_n} K_n$$

*and for any $i, j \in \{1, \ldots, n\}$ such that $\mathsf{R}_i \in \mathsf{Roles}_1$, $\mathsf{R}_j \in \mathsf{Roles}_2$, $sid_i = sid_j$, and $\ell_i, \ell_j \neq \tau$ (*i.e. *ℓ_i, ℓ_j are actions observed by the attacker), there exist two annotated traces tr^+ and tr' such that: $\mathsf{tr} \preccurlyeq \mathsf{tr}^+$, $\mathsf{tr}^+ \approx \mathsf{tr}'$, and $sid'_{\mathsf{ind}_i(\mathsf{tr},\mathsf{tr}^+)} \neq sid'_{\mathsf{ind}_j(\mathsf{tr},\mathsf{tr}^+)}$ where*

$$\mathsf{tr}' = K'_0 \xrightarrow[A'_1,\mathsf{R}'_1,[sid'_1,S'_1,D'_1]]{\ell'_1} K'_1 \xrightarrow[A'_2,\mathsf{R}'_2,[sid'_2,S'_2,D'_2]]{\ell'_2} \cdots \xrightarrow[A'_{n'},\mathsf{R}'_{n'},sid'_{n'},S'_{n'},D'_{n'}]{\ell'_{n'}} K'_{n'}.$$

Unlinkability versus indistinguishability. Note that unlinkability is a distinct notion from the indistinguishability notion exposed in Section 4. Protocols unlinkable w.r.t. any reasonable topology can be designed so as not to be indistinguishable for any role. An example of such a protocol would be $\mathcal{P} = \{P_1(z_S, z_D), P_2(z_V)\}$ defined as follows:

$$P_1(z_S, z_D) = \mathsf{out}(\mathsf{src}).\mathsf{in}(x) \qquad P_2(z_V) = \mathsf{in}(x).\mathsf{out}(\mathsf{dest})$$

where src and dest are two constants. The unlinkability is a consequence of emitting the same messages for every session, whereas the indistinguishability fails as the constant src (resp. dest) identifies the role P_1 (resp. P_2).

Reciprocally one can design protocols preserving indistinguishability for certain roles but not unlinkability for any two subsets of roles. The protocol $\mathcal{P}'$ made up of the three roles described below fails clearly at preserving the unlinkability w.r.t. any non-trivial topology for any sets of roles Roles_1 and Roles_2 as it mimicks the session identifiers introduced formerly.

$$P'_1(z_S, z_D) = \mathsf{new}\ n.\mathsf{out}(n).\mathsf{in}(x) \qquad P'_2(z_V) = \mathsf{in}(x).\mathsf{out}(x)$$
$$P'_3(z_V) = \mathsf{in}(x).\mathsf{store}(x).\mathsf{out}(x)$$

On the other hand, the indistinguishability w.r.t. any topology for either P'_2 or P'_3 is trivially preserved as the roles are essentially the same.

5.3 Analysis of ANODR

As discussed at the beginning of Section 5.2, ANODR, as many other routing protocols, does not preserve unlinkability (as soon as the underlying topology is non-trivial topology) for sets $\mathsf{Roles}_1 = \mathsf{Roles}_2 = \{\mathsf{Src}, \mathsf{Req}\}$ due to the forwarding of the same *id* by every intermediate node during the request phase. Actually, the simplified version of ANODR presented in Section 3.2 does not preserve unlinkability for sets $\mathsf{Roles}_1 = \mathsf{Roles}_2 = \{\mathsf{Dest}, \mathsf{Rep}\}$ due to the forwarding of the nonce *chall* by every intermediate node during the reply phase. This version does not preserve unlinkability for sets $\{\mathsf{Src}, \mathsf{Req}\}/\{\mathsf{Dest}, \mathsf{Rep}\}$ either. Indeed, during the request phase, the nodes will emit a message containing an onion, and during the reply phase, they are waiting for a message that contains exactly the same onion. This allows the attacker to link a request message with a reply message and to identify them as coming from the same session.

The updated version of ANODR (see Section 4.2) actually fixes the two last issues. Again, we need for this to consider topologies $\mathcal{T}$ for which any malicious node has at least two distinct neighbours other than itself. In such a situation, following the same ideas as the one used to establish indistinguishability, we can show that an initial configuation K_0 preserves unlinkability w.r.t. $\{\mathsf{Dest}, \mathsf{Rep}\}/\{\mathsf{Dest}, \mathsf{Rep}\}$, and $\{\mathsf{Src}, \mathsf{Req}\}/\{\mathsf{Dest}, \mathsf{Rep}\}$ (according to Definition 5).

6 Anonymity

Anonymity is concerned with hiding who performed a given action. Here, we are not concerned by hiding the identity of the sender (or the receiver) of a given message, but we would like to hide the identity of the source (or the destination) of the request/reply message. When the identity of the source is hidden, we speak about *source anonymity*. Similarly, when the identity of the destination is

hidden, we speak about *destination anonymity*. Again, we consider both types of anonymity with respect to an external eavesdropper that is localised to some nodes (possibly every one of them) of the network.

As in Section 5, to define the anonymity, we need to link messages occurring at various places in the network to their respective source and destination, thus we consider the augmented version of the protocol as in Section 5.1

6.1 Formalising anonymity

Intuitively, source (resp. destination) anonymity can be achieved if the attacker is unable to tell the source (resp. the destination) of an observed message. This idea can actually be interpreted as the existence of anonymity sets of cardinal greater or equal than two. As for the previous privacy-type notions, one cannot expect to hide the source (resp. destination) of an action in a trace tr without any sufficient traffic as it would be easy for an attacker to observe the first node to output a request (resp. a reply) and deduce the source (resp. destination) of this execution. For this reason, anonymity will be achieved if there exist two other traces tr^+ and tr' of the system which look the same to the attacker, and in which the corresponding transitions have different sources (resp. destinations).

Definition 6 (anonymity). *Let K_0 be an initial configuration associated to a routing protocol and a topology. We say that K_0 preserves source anonymity (resp. destination anonymity) if for any annotated trace* tr

$$\mathsf{tr} = K_0 \xrightarrow[A_1,\mathsf{R}_1,[sid_1,S_1,D_1]]{\ell_1} K_1 \xrightarrow[A_2,\mathsf{R}_2,[sid_2,S_2,D_2]]{\ell_2} \dots \xrightarrow[A_n,\mathsf{R}_n,[sid_n,S_n,D_n]]{\ell_n} K_n$$

*and for any $i \in \{1,\dots,n\}$ such that $\ell_i \neq \tau$ (*i.e. *ℓ_i is an action observed by the attacker), there exist two annotated traces tr^+ and tr' such that $\mathsf{tr} \preccurlyeq \mathsf{tr}^+$, $\mathsf{tr}^+ \approx \mathsf{tr}'$, and $S'_{\mathsf{ind}_i(\mathsf{tr},\mathsf{tr}^+)} \neq S_i$ (resp. $D'_{\mathsf{ind}_i(\mathsf{tr},\mathsf{tr}^+)} \neq D_i$) where*

$$\mathsf{tr}' = K'_0 \xrightarrow[A'_1,\mathsf{R}'_1,[sid'_1,S'_1,D'_1]]{\ell'_1} K'_1 \xrightarrow[A'_2,\mathsf{R}'_2,[sid'_2,S'_2,D'_2]]{\ell'_2} \dots \xrightarrow[A'_{n'},\mathsf{R}'_{n'},[sid'_{n'},S'_{n'},D'_{n'}]]{\ell'_{n'}} K'_{n'}$$

6.2 Anonymity versus indistinguishability/unlinkability.

The notions of source and destination anonymity are distinct from indistinguishability for a set of roles and unlinkability of two sets of roles. The protocol $\mathcal{P} = \{P_1(z_S, z_D), P_2(z_V)\}$ where $P_1(z_S, z_D) = \mathsf{out}(z_S).\mathsf{in}(x)$, and $P_2(z_V) = \mathsf{in}(x).\mathsf{out}(x)$ preserves both the indistinguishability of P_1 (a node can play P_2 as a response to a session it initiated previously as P_1) and the unlinkability of any two subsets of $\{P_1, P_2\}$ (as every session with the same node as a source will generate the exact same messages) but not source anonymity as the identity of the source is obvious for any attacker along the route. A symmetrical protocol can be built by replacing z_S with z_D in P_1 to disclose the destination of a session without breaking the indistinguishability.

Conversely, the protocol $\mathcal{P} = \{P_1(z_S, z_D), P_2(z_V)\}$ defined as

$$P_1(z_S, z_D) = \mathsf{new}\ n.\mathsf{out}(\langle \mathsf{src}, n\rangle).\mathsf{in}(x) \quad P_2(z_V) = \mathsf{in}(\langle x, y\rangle).\mathsf{out}(\langle \mathsf{dest}, y\rangle)$$

preserves destination anonymity as any node can play P_2 in response to a request, whatever the original destination was. Indeed, given such a topology $\mathcal{T}$, a trace tr of the protocol, and a visible action $\ell_i = (\mathsf{out}(y), A)$ associated to a a source $S_i = S$ and a destination $D_i = A$, we can let tr^+ be equal to tr and define tr' to be the trace mimicking tr but with S as the source and destination of the request associated to ℓ_i. The equivalence of tr and tr' comes from the content of their frames which is limited to the names of the request sources, identical in both cases. On the other hand, $\mathcal{P}$ does not preserve indistinguishability of P_1 or P_2, nor unlinkability of any two subsets of $\{P_1, P_2\}$ as session identifiers and constants to distinguish roles are embedded in the protocol.

However, intuitively, there is a relation between source anonymity (resp. destination anonymity) and indistinguishability of the role source (resp. destination). Indeed, source anonymity seems to imply that the action of interest can be mimicked by someone different from source, and thus who should not play the role source. Thus, restricting ourselves to "reasonable" routing protocols, we are indeed able to establish this relation. For this, we define *source* and *destination roles* as roles which are only used by nodes acting as sources or destinations.

Definition 7 (acting as a source (resp. destination)). *Let K_0 be an initial configuration associated to a routing protocol and a topology. We say that* Roles *is the set of roles* acting as a source *(resp.* acting as a destination*) if for any annotated trace* tr *with* $\ell_1, \ldots, \ell_n \neq \tau$

$$\mathsf{tr} = K_0 \xrightarrow[A_1,\mathsf{R}_1,sid_1,S_1,D_1]{\ell_1} K_1 \xrightarrow[A_2,\mathsf{R}_2,sid_2,S_2,D_2]{\ell_2} \ldots \xrightarrow[A_n,\mathsf{R}_n,sid_n,S_n,D_n]{\ell_n} K_n$$

and for any $i \in \{1, \ldots, n\}$, $\mathsf{R}_i \in$ Roles *if and only if* $A_i = S_i$ *(resp. if and only if* $A_i = D_i$*).*

In case of ANODR (both versions), the set of roles acting as a source is {Src}. This is the only role able to spontaneously start a session and it is unable to reply to a request. The set of roles acting as a destination is limited to {Dest}. The proof of opening prevents any node other than the destination to play it and, conversely, a destination node can only play the role Dest as a response to such a request. Note that, for some badly designed routing protocols, it may happen that the set of roles acting as a source (resp. destination) is empty. In such a case, the proposition below is trivially true.

Proposition 2. *Let K_0 be an initial configuration associated to a routing protocol and a topology. If K_0 preserves source (resp. destination) anonymity, then it preserves indistinguishability w.r.t. the set of roles acting as a source (resp. destination).*

6.3 Analysis of ANODR

In this section, we apply our formalisation of anonymity to the ANODR routing protocol. As a consequence of Propositions 1 and 2, we have the following result.

Corollary 1. *Let $\mathcal{T}$ be a topology with a malicious node that has only malicious neighbours, and K_0 be an initial configuration associated to $\mathcal{P}^{\mathsf{simp}}_{\mathsf{ANODR}}$ and $\mathcal{T}$. We have that K_0 preserves neither source nor destination anonymity.*

For the updated version of ANODR, similarly, we can show that it does not preserve source anonymity. However, this protocol seems to have been designed to achieve destination anonymity. Indeed, considering topologies for which any malicious node has at least one neighbour other than itself, we can show that any trace tr can be extended to tr^+ so that the node of interest has at least two reply to treat (one as the destination of the request, and the other one as the forwarder). This is actually sufficient to confuse the attacker who observes the network, and to establish anonymity of the destination according to Definition 6.

7 Conclusion

We have defined a framework for modeling routing protocols in ad hoc networks in a variant of the applied pi-calculus. Within this framework we can stipulate which agents are subject to the attention of a global eavesdropper. We were able to propose several definitions for privacy-type properties that encompass the specificity of a given network topology. We illustrate these definitions on the anonymous routing protocol ANODR, considered in two versions, and thus provide a partial formal security analysis of its anonymity.

As future work, it would be interesting to have a more general model of protocols to represent high-level operations in routing protocols (*e.g.* reversing a list). However, since our definitions are expressed in terms of traces, this should not impact so much the privacy definitions proposed in this paper. Another direction is the enrichment of our attacker model, so as to model fully compromised nodes which disclose their long-term keys or fresh nonces generated during the execution of the protocols, and active attackers able to forge messages and interact with honest agents. Finally, from the point of view of the verification, a reduction result on network topologies as presented in [11] would make the perspective of automated proofs of anonymity easier.

References

1. M. Abadi and C. Fournet. Mobile values, new names, and secure communication. In *Proc. 28th Symposium on Principles of Programming Languages (POPL'01)*, pages 104–115. ACM Press, 2001.
2. M. Abadi and A. Gordon. A calculus for cryptographic protocols: The spi calculus. In *Proc. 4th Conference on Computer and Communications Security (CCS'97)*, pages 36–47. ACM Press, 1997.
3. M. Arapinis, T. Chothia, E. Ritter, and M. Ryan. Analysing unlinkability and anonymity using the applied pi calculus. In *Proc. 23rd Computer Security Foundations Symposium (CSF'10)*, pages 107–121. IEEE Computer Society Press, 2010.

4. A. Armando, R. Carbone, L. Compagna, J. Cuéllar, and M. L. Tobarra. Formal analysis of SAML 2.0 web browser single sign-on: breaking the SAML-based single sign-on for google apps. In *Proc. of the 6th ACM Workshop on Formal Methods in Security Engineering (FMSE 2008)*, pages 1–10. ACM, 2008.
5. A. Armando et al. The AVISPA Tool for the automated validation of internet security protocols and applications. In *Proc. 17th International Conference on Computer Aided Verification, CAV'2005*, volume 3576 of *LNCS*. Springer, 2005.
6. M. Arnaud, V. Cortier, and S. Delaune. Modeling and verifying ad hoc routing protocols. In *Proc. 23rd IEEE Computer Security Foundations Symposium (CSF'10)*, pages 59–74. IEEE Computer Society Press, July 2010.
7. M. Backes, C. Hritcu, and M. Maffei. Automated verification of remote electronic voting protocols in the applied pi-calculus. IEEE Comp. Soc. Press, 2008.
8. B. Blanchet. An efficient cryptographic protocol verifier based on prolog rules. In *Proc. 14th Computer Security Foundations Workshop (CSFW'01)*. IEEE Comp. Soc. Press, 2001.
9. M. Bruso, K. Chatzikokolakis, and J. den Hartog. Formal verification of privacy for RFID systems. In *Proc. 23rd Computer Security Foundations Symposium (CSF'10)*. IEEE Computer Society Press, 2010.
10. R. Chrétien and S. Delaune. Formal analysis of privacy for routing protocols in mobile ad hoc networks. Research Report LSV-12-21, Laboratoire Spécification et Vérification, ENS Cachan, France, Dec. 2012. 24 pages.
11. V. Cortier, J. Degrieck, and S. Delaune. Analysing routing protocols: four nodes topologies are sufficient. In *Proc. of the 1st International Conference on Principles of Security and Trust (POST'12)*, LNCS, pages 30–50. Springer, Mar. 2012.
12. S. Delaune, S. Kremer, and M. D. Ryan. Verifying privacy-type properties of electronic voting protocols. *Journal of Computer Security*, (4):435–487, July 2008.
13. F. D. Garcia, I. Hasuo, W. Pieters, and P. van Rossum. Provable anonymity. In *Proc. ACM workshop on Formal methods in security engineering (FMSE'05)*, pages 63–72. ACM, 2005.
14. Y.-C. Hu, A. Perrig, and D. Johnson. Ariadne: A Secure On-Demand Routing Protocol for Ad Hoc Networks. *Wireless Networks*, 11:21–38, 2005.
15. J. Kong and X. Hong. ANODR: anonymous on demand routing with untraceable routes for mobile ad-hoc networks. In *Proc. 4th ACM Interational Symposium on Mobile Ad Hoc Networking and Computing, (MobiHoc'03)*. ACM, 2003.
16. S. Mauw, J. Verschuren, and E. P. de Vink. A formalization of anonymity and onion routing. In *Proc. 9th European Symposium on Research Computer Security (ESORICS'04)*, volume 3193 of *LNCS*, pages 109–124. Springer, 2004.
17. S. Nanz and C. Hankin. A Framework for Security Analysis of Mobile Wireless Networks. *Theoretical Computer Science*, 367(1):203–227, 2006.
18. P. Papadimitratos and Z. Haas. Secure routing for mobile ad hoc networks. In *Proc. SCS Communication Networks and Distributed Systems Modelling Simulation Conference (CNDS)*, 2002.
19. A. Serjantov and G. Danezis. Towards an information theoretic metric for anonymity. In *Privacy Enhancing Technologies*, pages 41–53, 2002.
20. R. Song, L. Korba, and G. Lee. AnonDSR: Efficient anonymous dynamic source routing for mobile ad-hoc networks. In *Proc. ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN'05)*. ACM, 2005.
21. M. G. Zapata and N. Asokan. Securing ad hoc routing protocols. In *Proc. 1st ACM workshop on Wireless SEcurity (WiSE'02)*, pages 1–10. ACM, 2002.