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.

D. Basin and J.C. Mitchell (Eds.): POST 2013, LNCS 7796, pp. 1–20, 2013.

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 $ar(\mathsf{f}) = s_1 \times \ldots \times s_k \to s$ 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.

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

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:

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

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

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

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

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

$$\textsc{Read-Else}\ \ (\mathcal{E};\lfloor \mathsf{read}\ u[\Phi]\ \mathsf{then}\ P\ \mathsf{else}\ Q\rfloor_A\cup\mathcal{P};\mathcal{S};\sigma)\xrightarrow{\tau}_{\mathcal{T}}(\mathcal{E};\lfloor Q\rfloor_A\cup\mathcal{P};\mathcal{S};\sigma)$$
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

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

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

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

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

$$\textsc{New}\qquad (\mathcal{E};\lfloor \mathsf{new}\ n.P\rfloor_A\cup\mathcal{P};\mathcal{S};\sigma)\xrightarrow{\tau}_{\mathcal{T}}(\mathcal{E}\cup\{n'\};\lfloor P\{^{n'}/_n\}\rfloor_A\cup\mathcal{P};\mathcal{S};\sigma)$$
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:

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

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

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* $\mathsf{tr}_1 = K_1 \overset{s_1}{\Longrightarrow} (\mathcal{E}_1; \mathcal{P}_1; \mathcal{S}_1; \sigma_1)$ *and* $\mathsf{tr}_2 = K_2 \overset{s_2}{\Longrightarrow} (\mathcal{E}_2; \mathcal{P}_2; \mathcal{S}_2; \sigma_2)$ *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 $\langle\cdot,\cdot\rangle$.

$$A \to B : \langle \mathsf{req}, id, \mathsf{pub}(A), \{\mathsf{dest}, K_D\}_{K_T}, \{\mathsf{dest}\}_{K_D}, \{\mathsf{src}\}_{K_A}\rangle$$
$$B \to C : \langle \mathsf{req}, id, \mathsf{pub}(B), \{\mathsf{dest}, K_D\}_{K_T}, \{\mathsf{dest}\}_{K_D}, \{N_B, \{\mathsf{src}\}_{K_A}\}_{K_B}\rangle$$
$$C \to D : \langle \mathsf{req}, id, \mathsf{pub}(C), \{\mathsf{dest}, K_D\}_{K_T}, \{\mathsf{dest}\}_{K_D}, \{N_C, \{N_B, \{\mathsf{src}\}_{K_A}\}_{K_B}\}_{K_C}\rangle$$

$$D \to C : \langle \mathsf{rep}, \{K_{seedD}\}_{\mathsf{pub}(C)}, \{K_D, \{N_C, \{N_B, \{\mathsf{src}\}_{K_A}\}_{K_B}\}_{K_C}\}_{K_{seedD}}\rangle$$
$$C \to B : \langle \mathsf{rep}, \{K_{seedC}\}_{\mathsf{pub}(B)}, \{K_D, \{N_B, \{\mathsf{src}\}_{K_A}\}_{K_B}\}_{K_{seedC}}\rangle$$
$$B \to A : \langle \mathsf{rep}, \{K_{seedB}\}_{\mathsf{pub}(A)}, \{K_D, \{\mathsf{src}\}_{K_A}\}_{K_{seedB}}\rangle$$

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* $\ell_1, \ldots, \ell_n \neq \tau$

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

and for any $i \in \{1, \ldots, n\}$, $\mathsf{R}_i \in \mathsf{Roles}$ 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 $\{\mathsf{Src}\}$. 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 $\{\mathsf{Dest}\}$. 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 $\mathcal{P}^{\mathsf{simp}}_{ANODR}$ 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. Abadi, M., Fournet, C.: Mobile values, new names, and secure communication. In: Proc. 28th Symposium on Principles of Programming Languages, POPL 2001, pp. 104–115. ACM Press (2001)
2. Abadi, M., Gordon, A.: A calculus for cryptographic protocols: The spi calculus. In: Proc. 4th Conference on Computer and Communications Security, CCS 1997, pp. 36–47. ACM Press (1997)
3. Arapinis, M., Chothia, T., Ritter, E., Ryan, M.: Analysing unlinkability and anonymity using the applied pi calculus. In: Proc. 23rd Computer Security Foundations Symposium, CSF 2010, pp. 107–121. IEEE Computer Society Press (2010)
4. Armando, A., Carbone, R., Compagna, L., Cuéllar, J., Tobarra, M.L.: 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, pp. 1–10. ACM (2008)

5. Armando, A., Basin, D., Boichut, Y., Chevalier, Y., Compagna, L., Cuellar, J., Hankes Drielsma, P., Heám, P.C., Kouchnarenko, O., Mantovani, J., Mödersheim, S., von Oheimb, D., Rusinowitch, M., Santiago, J., Turuani, M., Viganò, L., Vigneron, L.: The AVISPA Tool for the Automated Validation of Internet Security Protocols and Applications. In: Etessami, K., Rajamani, S.K. (eds.) CAV 2005. LNCS, vol. 3576, pp. 281–285. Springer, Heidelberg (2005)
6. Arnaud, M., Cortier, V., Delaune, S.: Modeling and verifying ad hoc routing protocols. In: Proc. 23rd IEEE Computer Security Foundations Symposium, CSF 2010, pp. 59–74. IEEE Computer Society Press (July 2010)
7. Backes, M., Hritcu, C., Maffei, M.: Automated verification of remote electronic voting protocols in the applied pi-calculus. IEEE Comp. Soc. Press (2008)
8. Blanchet, B.: An efficient cryptographic protocol verifier based on prolog rules. In: Proc. 14th Computer Security Foundations Workshop, CSFW 2001. IEEE Comp. Soc. Press (2001)
9. Bruso, M., Chatzikokolakis, K., den Hartog, J.: Formal verification of privacy for RFID systems. In: Proc. 23rd Computer Security Foundations Symposium, CSF 2010, IEEE Computer Society Press (2010)
10. Chrétien, R., Delaune, S.: Formal analysis of privacy for routing protocols in mobile ad hoc networks. Research Report LSV-12-21, Laboratoire Spécification et Vérification, ENS Cachan, France, 24 pages (December 2012)
11. Cortier, V., Degrieck, J., Delaune, S.: Analysing Routing Protocols: Four Nodes Topologies Are Sufficient. In: Degano, P., Guttman, J.D. (eds.) Principles of Security and Trust. LNCS, vol. 7215, pp. 30–50. Springer, Heidelberg (2012)
12. Delaune, S., Kremer, S., Ryan, M.D.: Verifying privacy-type properties of electronic voting protocols. Journal of Computer Security (4), 435–487 (2008)
13. Garcia, F.D., Hasuo, I., Pieters, W., van Rossum, P.: Provable anonymity. In: Proc. ACM Workshop on Formal Methods in Security Engineering, FMSE 2005, pp. 63–72. ACM (2005)
14. Hu, Y.-C., Perrig, A., Johnson, D.: Ariadne: A Secure On-Demand Routing Protocol for Ad Hoc Networks. Wireless Networks 11, 21–38 (2005)
15. Kong, J., Hong, X.: 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 2003. ACM (2003)
16. Mauw, S., Verschuren, J.H.S., de Vink, E.P.: A Formalization of Anonymity and Onion Routing. In: Samarati, P., Ryan, P.Y.A., Gollmann, D., Molva, R. (eds.) ESORICS 2004. LNCS, vol. 3193, pp. 109–124. Springer, Heidelberg (2004)
17. Nanz, S., Hankin, C.: A Framework for Security Analysis of Mobile Wireless Networks. Theoretical Computer Science 367(1), 203–227 (2006)
18. Papadimitratos, P., Haas, Z.: Secure routing for mobile ad hoc networks. In: Proc. SCS Communication Networks and Distributed Systems Modelling Simulation Conference, CNDS (2002)
19. Serjantov, A., Danezis, G.: Towards an Information Theoretic Metric for Anonymity. In: Dingledine, R., Syverson, P.F. (eds.) PET 2002. LNCS, vol. 2482, pp. 41–53. Springer, Heidelberg (2003)
20. Song, R., Korba, L., Lee, G.: AnonDSR: Efficient anonymous dynamic source routing for mobile ad-hoc networks. In: Proc. ACM Workshop on Security of Ad Hoc and Sensor Networks, SASN 2005. ACM (2005)
21. Zapata, M.G., Asokan, N.: Securing ad hoc routing protocols. In: Proc. 1st ACM Workshop on Wireless SEcurity, WiSE 2002, pp. 1–10. ACM (2002)