A Calculus of Trustworthy Ad Hoc Networks [*]

Massimo Merro and Eleonora Sibilio

Dipartimento di Informatica, Università degli Studi di Verona, Italy

Abstract

We propose a *process calculus* for *mobile ad hoc networks* which relies on an abstract behaviour-based multilevel *trust model*. The operational semantics of the calculus is given in terms of a labelled transition system, where actions are executed at a certain security level. We define a *labelled bisimilarity* over networks parameterised on security levels. Our bisimilarity is a congruence and an efficient proof method for an appropriate variant of barbed congruence, a standard contextually-defined program equivalence. Communications in the calculus are safe with respect to the security levels of the involved parties. In particular, we ensure safety despite compromise: compromised nodes cannot affect the rest of the network. A *non-interference* result is also proved in terms of information flow. Finally, we illustrate the practical utility of our calculus by providing a formal description of trust-based versions of a routing protocol and a leader election protocol for ad hoc networks.

1 Introduction

Wireless communication has become very popular in industry, business, commerce and in everyday life. Wireless technology spans from user applications such as personal area networks, ambient intelligence, and wireless local area networks, to real-time applications, such as cellular and ad hoc networks.

The emerging mobile ad hoc and sensor networking paradigms usher in a new type of network: devices form multihop topologies in a self-organizing manner, relaying packets from other devices across multiple wireless links (hops), and essentially become the network. Several applications are enabled already by these developments or are expected in the near future. *Wireless sensor networks* are deployed for environmental and building monitoring. *Mobile ad hoc networks* are used in disaster relief operations, with rolled-in base stations and portable radios, as well as in tactical operations with a multitude of vehicle-, aircraft-, or personnel-borne wireless devices. *Static ad hoc*

[*] An extended abstract appears in the proceedings of the *6th International Workshop on Formal Aspects of Security & Trust* (FAST'09), Eindhoven, The Netherlands, *Lecture Notes in Computer Science*, volume 5983, pp. 157-172, Springer 2010.

or *mesh networks* are being formed by home computers with roof-top antennas. *Low-mobility ad hoc networks* will enable (often delay-tolerant) communication in urban environments; examples include networks of hand-held devices, wearable devices, and radio frequency identifiers (RFID).

In this paper we focus on mobile ad hoc networks. A Mobile ad Hoc NETwork (MANET) is a self-configuring network of mobile devices (also called nodes) communicating with each other via radio transceivers. Basically, wireless devices use radio frequency channels to broadcast messages to the other devices. Ad hoc networks may operate in a standalone fashion, or may be connected to the larger Internet. They can be used wherever a wired backbone is infeasible and/or economically inconvenient.

In MANETs, due to the limited transmission range of communications, each node seeks the assistance of its neighbouring nodes in forwarding packets. In order to establish routes between nodes which are further than a single hop, specially configured routing protocols are engaged. The unique feature of these protocols is their ability to trace routes in spite of a dynamic topology. *Routing protocols* for MANETs, such as AODV, DSR and TORA [47, 32, 46], require persistent cooperative behaviour from the intermediate nodes that primarily contribute to the route development. These protocols have been developed for networks where all nodes can faithfully execute them in a munificent manner. However, in real life, such an altruistic stance is difficult to achieve and, so, these protocols are more often executed by nodes that divert from the basic requirements of participation.

MANETs are generally established in open and physically insecure environments where nodes are exposed to various threats, of which the most interesting and important is *node subversion*. In this kind of attack, a node may be reverse-engineered, and replaced by a malicious node. A bad (or compromised) node can communicate with any other node, good or bad. Bad nodes may have access to the keys of all other bad nodes, whom they can impersonate if they wish. They do not execute the authorised software and thus do not necessarily follow protocols to identify misbehaviour, isolate other bad nodes, vote honestly or delete keys shared with compromised nodes.

Lack of a fixed infrastructure, shared wireless medium, cooperative behaviour, and physical vulnerability are some of the features that make particularly challenging the design of a *trust-based scheme* for mobile ad hoc networks. In a trust-based scheme, all nodes in the network independently execute a trust model and maintain their own assessment concerning other nodes in the network. Each node, based upon its personal experiences, rewards collaborating nodes for their benevolent behaviour and penalises malicious nodes for their malevolent conduct. Most of the events that are experienced by a node occur within the vicinity of its immediate neighbours. This helps to establish trust relationships between the neighbours. In contrast, very few events are directly experienced between nodes that are more than one hop away.

MANETs do not support stable hierarchies of trust relations because trust evidence may be uncertain and incomplete, and only sporadically collected and exchanged. In fact, when moving around, wireless devices break links with old neighbours and establish fresh links with new devices. This makes security even more challenging as the compromise of a legitimate node or the insertion of a malicious node may go unnoticed.

Contribution In this paper, we propose a *process calculus* for mobile ad hoc networks which relies on an abstract *behaviour-based trust model.* Our trust model supports both *direct trust*, to describe monitoring of neighbour nodes, and *indirect trust*, when collecting recommendations and spreading reputations. We model our networks as *multilevel systems* [5] where each device is associated with a security level depending on its behaviour.

In our calculus, each node is equipped with a local trust store containing a set of assertions. These assertions supply trust information about the other nodes, according to a local security policy. The calculus is not directly concerned with cryptographic underpinnings. However, we assume the presence of a hierarchical key generation and a distribution protocol [30, 56]. Thus, messages are transmitted at a certain security level by relying on an appropriate set of cryptographic keys.

We provide an operational semantics in terms of a *labelled transition system.* Transitions take the form

$$M \xrightarrow{\lambda}_{\rho} N$$

to indicate that the network M can perform the action λ, at security level ρ, evolving into the network N.

In our setting, communications are safe up to certain a security level. Thus, a node m transmitting at security level ρ may only synchronise with nodes at security level ρ or above, according to the local knowledge of both sender and receivers. We also ensure *safety despite compromise*, as bad nodes, once detected, may not interact with good ones. In this manner, bad nodes (recognised as such) are isolated from the rest of the network.

A central concern in process calculi is to establish when two terms have the same observable behaviour. Behavioural equivalences are fundamental for justifying program transformations. Our program equivalence is a security variant of (weak) barbed congruence, a branching-time contextually-defined program equivalence. Barbed equivalences [43] are simple and intuitive but difficult to use due to the quantification on all contexts. Simpler proof techniques are based on labelled bisimilarities [41], which are co-inductive relations that characterise the behaviour of processes using a labelled transition system. Along the lines of [14], we propose a labelled bisimilarity, called *δ-bisimilarity*, parameterised on security levels. Intuitively, two networks are δ-bisimilar if they cannot be distinguished by any observer that can only perform actions at security level at most δ. We prove that δ-bisimilarity represents an efficient proof method for our barbed congruence.

We use our notion of δ-bisimilarity to prove a *non-interference* result [26]. Formally, high-level behaviours can be arbitrarily changed without affecting low-level equivalences, that is equivalence parameterised on low security levels. Thus, a network is interference free if its low security level behaviour is not affected by any activity at high security level.

Finally, we show that our calculus represents a suitable formal language to describe *trust-based routing protocols* for MANETs [49, 50, 63]. Initial work on routing in ad hoc networks has considered only the problem of providing efficient mechanisms for finding paths, without considering security issues. Trust-based routing represents an emerging effective approach to improve the security of mobile ad hoc networks as opposed to *secure routing protocols*, such as SRP [45], Ariadne [29], endairA [2], SAODV [62], and ARAN [55], where paths are established by means of cryptographic communications among nodes. Trust-based schemes can be successfully used for a variety of other protocols. As as example, we provide a trust-based version of a leader election protocol for MANETs, proposed in [60].

Outline In Section 2, we introduce the concepts of trust and reputation, we describe the trust management systems in general and the trust management systems for MANETs. In Section 3, we describe our behaviour-based trust model. In Section 4, we provide both syntax and operational semantics of our calculus. In Section 5, we present our mobility model. In Section 6, we prove our safety properties. In Section 7, we propose a notion of observational equivalence along the lines of Milner and Sangiorgi's barbed congruence. In Section 8, we propose a labelled bisimilarity as a proof method for our observations equivalence. More precisely, we prove that our bisimilarity is a congruence and it implies our observational equivalence. In Section 9, we use our bisimilarity to prove a non-interference result. In Section 10, we use our calculus to formalise a trust-based version of the AODV routing protocol; we also provide a trust-based version of the leader election protocols for MANETs [60]. Finally, Section 11 is devoted to conclusions and related work.

2 Background on trust models

In this section, we discuss the notions of *trust* and *reputation* in information systems according to the literature. We then discuss the main key issues when designing *trust management systems* for mobile ad hoc networks. Finally, we present a number of existing trust-based schemes for MANETs.

2.1 Trust and reputation

The concepts of trust and reputation are firmly routed in sociology and psychology and they are widely used in computer science.

Trust enables a *trustor* to reduce uncertainty in its future interactions with a *trustee*, whose actions may affect the state of the trustor. Trust is usually represented as a binary relationship between trustor and trustee. Trust formalisation has been the subject of several academic works. For instance, in [20] trust is defined as the "belief or subjective possibility by which an individual A expects that another individual B performs a given action on which the welfare of A depends". The author has introduced the dependency of trust from the context, meaning that it applies to a specific purpose or domain of actions. In these terms, trust may be viewed as a quantitative value, that is a quantifiable relation between two entities. According to [27], trust is "the quantified belief by a trustor with respect to the competence, honesty, security and dependability of a trustee, within a specified context". In [33] the authors have distinguished between functional and referral trust, and between direct and indirect trust. *Functional trust* is the belief in an entity's ability and willingness to carry out or support a specific function on which the relying party depends. *Referall trust* is the belief in an entity's ability to recommend another entity with respect to functional trust.

Reputation is defined as the opinion held by the trustor towards the trustee, based both on its past experience and recommendations of other trustees. A recommendation is simply an attempt at communicating a party's reputation from one community context to another. Reputation is an important concept for the trust evaluation, but it is often confused with trust. Trust represents an active and decisive concept: if one entity trusts another entity then the latter is allowed to perform certain actions. Reputation may serve as a source of trust; however, it does not directly define allowed actions. Reputation usually comes from the context and it does not reflect personal experience of the interested party. As for trust, there are several possible definitions of reputation. For instance, in [1] reputation is defined as "the expectation about an individual's behaviour based on observations of its past behaviour". In [34] reputation is proposed as a meaning of building trust; one can trust another based on its good reputation.

2.2 Designing trust management systems for MANETs

Decentralised Trust Management Systems [7] define languages for expressing authorisations and access control policies, and provide trust management engines for determining when a particular request is authorised. Traditional *access control mechanisms* [54] are centralised and operate under a closed world assumption in which all of the parties are known. Trust management systems generalise access control mechanisms by operating in distributed systems and eliminating the closed world assumption.

Some of the features of MANETs, such as lack of a fixed infrastructure, node mobility, shared wireless medium, cooperative behaviour, and physical vulnerability, make particularly challenging the design of a trust management mechanism for them. In MANETs, node connectivity cannot be assured, and thus stable hierarchies of trust relations cannot be supported. More specifically, trust management systems for MANETs should:

- be *decentralised* and not based on online trusted parties; instead, they should support distributed, *cooperative evaluation*, based on uncertain evidence;
- support and exploit the *diversity in the roles* and the capabilities of the nodes in the deployments by allowing for flexibility in the trust establishment process;
- support *trust revocation* in a controlled manner;

- scale to *large deployments*, be flexible to membership changes and entail acceptable resource consumption.

Trust management systems can be classified into *credential-based* and *behaviour-based* systems. Credential-based trust management systems for MANETs aim at defining mechanisms for predeployment of knowledge on the trust relationships within the network, usually represented by certificates, to be spread, maintained and managed either independently or cooperatively by the nodes. Trust decisions are mainly based on the provision of a valid certificate which proves that the target node is considered trusted either by a certification authority or by other nodes that the issuer trusts. It is generally outside the scope of certificate-based frameworks to evaluate the behaviour of nodes taking trust decisions on that evaluation.

Behaviour-based systems are often called experience-based as in these models an entity A trusts another entity B based on its experience on B's past behaviour. In behaviour-based trust management systems for MANETs, each node comes together with an extra component called *trust manager*. A trust manager consists of two main components: the monitoring module and the reputation handling module. The first module monitors the behaviour of neighbours, while the second one collects/spreads recommendations and evaluates trust information about other nodes using a local security policy. The continuous work of the trust manager results in a local trust store T containing the up-to-date trust relations. Although a mechanism that determines the identities of the other nodes is usually assumed to exist, it is generally outside the scope of behaviour-based trust establishment models to authenticate other nodes and to determine whether they are legitimate members of the network. The main objective of behaviour-based models is to isolate those nodes that either act maliciously or selfishly.

2.3 Some trust-based schemes for MANETS

Comprehensive surveys of trust management systems for ad hoc networks can be found in [3, 52, 4]. Here, we provide a brief overview of some of these systems.

2.3.1 Watchdog and Pathrater

The Watchdog and Pathrater mechanism [37] has been specifically designed to optimise the forwarding mechanism in the DSR Routing protocol for MANETs [32]. The mechanism basically consists of two components: Watchdog and Pathrater. The Watchdog is responsible for detecting selfish nodes that do not forward packets. To do so, each node in the network buffers every transmitted packet for some time. During this interval, the node places its wireless interface into the promiscuous mode in order to overhear whether the next node has forwarded the packet or not. The Pathrater assigns different rating to the nodes based upon the feedback that it receives from the Watchdog. These ratings are then used to select routes consisting of nodes with the highest forwarding rate.

2.3.2 CONFIDANT

CONFIDANT (Cooperation Of Nodes, Fairness In Dynamic ad hoc NeTworks) [10] adds a trust manager and a reputation system to the Watchdog and Pathrater scheme. The trust manager evaluates the events reported by the Watchdog (monitor in this case) and issues alarms to warn other nodes regarding malicious nodes. The alarm recipients are maintained in a friends-list, which is configured through a user-to-user authentication mechanism [59]. To verify the source of alarms, a mechanism similar to Pretty Good Privacy [21] is employed. The reputation system maintains a black-list of nodes at each node and shares them with nodes in the friends-list. The CONFIDANT protocol implements a punishment-based scheme by not forwarding packets of nodes whose trust level drops below a certain threshold.

2.3.3 CORE

CORE (COllaborative REputation) [40] is similar to CONFIDANT, however, it employs a complicated reputation exchange mechanism. CORE divides the reputation of a node into three distinct components: *Subjective Reputation*, which is observed through own observations, *Indirect Reputation*, which is a positive report by another node, and *Functional Reputation*, which is based upon behaviour monitored during a specific task. These reputations are weighted for a combined reputation value. This combined reputation value is used to make decisions regarding the inclusion or isolation of another node. CORE makes use of two types of entities, a requestor and one or more providers, to support a collaborative reputation mechanism. The requestor asks the providers for reputation values and validates the obtained results with the expected results that have been derived using the Watchdog. Positive trust ratings are exchanged, while the negative ratings are locally derived using the Watchdog.

2.3.4 Terminodes

The TermiNodes project [11] makes use of a virtual currency called nuglets, which serves as a payment per forwarded packet. The nuglets are maintained by each node in a tamper-resistant security module. The project uses a cryptographic infrastructure to ensure accuracy in transactions and avoid misuse of nuglets. The number of nuglets held by a node increase with every forwarded packet and decrease with each originated packet. The project endeavours to encourage forwarding by introducing two charging models: Packet trade model (Recipient to pay) and Packet purse model (Sender to pay). In the first model, each intermediate node has to purchase the packet from the sender of the packet. This increases the overall price of a packet, which has to be paid by the destination. The advantage of this model is that the originator of a packet does not need to know in advance the exact amount of nuglets required to reach a particular destination and can, so, send the packet for free. The obvious disadvantage here is that it does not stop malicious nodes from superfluous flooding. In the Packet Purse Model, the sending node has to load each packet with sufficient nuglets so that it reaches the

Table 1 Trust Framework

$m, n \in \mathit{Nodes}$	node name
$\langle \mathcal{S}, < \rangle$	complete lattice
$\mathcal{S} = \{\mathsf{bad}, \mathsf{trust}, \mathsf{low}, \mathsf{high}\}$	security level
$A \in \mathit{Assertions} = \mathit{Nodes} \times \mathit{Nodes} \times \mathcal{S}$	assertion
$T \subseteq \wp(\mathit{Assertions})$	trust store
$\mathcal{P} : \wp(\mathit{Assertions}) \to \wp(\mathit{Assertions})$	policy function

destination. During transit of the packet, each intermediate node supposedly takes one nuglet out of the packet as its forwarding fee. The advantage of this model is that it is resilient to flooding as the number of nuglets is limited in each packet. The disadvantage of this scheme is that the sender has to know precisely the number of nuglets that are required to be loaded into each transmitted packet and to ensure that the intermediate nodes do not overcharge during the forwarding mechanism.

3 An abstract trust model for MANETs

In this section, we propose an abstract trust model for MANETs where trust information may change over time due to mobility, temporary disconnections, recommendations, etc. We support *node revocation* and spreading of reputations. Repudiable evidence allows bad nodes to falsely accuse good nodes. Thus, recommendations are always evaluated using a local security policy implementing some appropriate metric.

The basic components of our model are *nodes* (or principals), *security levels*, *assertions*, *policies* and *trust stores*. We use $k, l, m, n, \ldots$ to range over the set *Nodes* of node names. We assume a complete lattice $\langle \mathcal{S}, < \rangle$ of security levels: $\mathsf{bad} < \mathsf{trust} < \mathsf{low} < \mathsf{high}$. The bad security level is associated with a compromised behaviour. The lowest security level associated with a trusted behaviour is trust; the low level is associated with more trusted behaviour, i.e. for handling more sensible data; high stands for the highest trusted behaviour. The metavariable ρ ranges over security levels in $\mathcal{S}$.

Assertions are represented by triples contained in $\mathit{Nodes} \times \mathit{Nodes} \times \mathcal{S}$. An assertion (m, n, ρ) says that a node m trusts a node n at security level ρ. A local trust store T contains a set of assertions, formally $T \subseteq \wp(\mathit{Assertions})$. When a node m (the trustor) wants to know the security level of a node n (the trustee), it simply has to check its own trust store T. For convenience, we often use T as a partial function of type $\mathit{Nodes} \to \mathit{Nodes} \to \mathcal{S}$, writing $T(m, n) = \rho$ when m considers n as a node of security level ρ. If $\rho = \mathsf{bad}$ then m considers n to be a compromised node, and stops any interaction with it. A node can receive new assertions from its neighbours. These assertions will be opportunely stored in the local trust store by the trust manager, according to a local security policy $\mathcal{P}$. A *security policy* $\mathcal{P}$ is a function that evaluates the current information collected by a node and returns a set of assertions consistent

Table 2 The Syntax

Values			
u	$::=$	v	closed value
	$\mid$	x	variable
Networks:			
M, N	$::=$	$\mathbf{0}$	empty network
	$\mid$	$M \mid N$	parallel composition
	$\mid$	$n[P]_T$	node
Processes:			
P, Q	$::=$	nil	termination
	$\mid$	$\sigma!\langle\tilde{u}\rangle.P$	broadcast sender
	$\mid$	$\sigma!\langle\tilde{u}\rangle_u.P$	unicast sender
	$\mid$	$\sigma?(\tilde{x}).P$	receiver
	$\mid$	$P + Q$	nondeterministic choice
	$\mid$	$[\tilde{u} \mathrel{\mathsf{op}} \tilde{u}']P, Q$	matching
	$\mid$	$H\langle\tilde{u}\rangle$	recursion

with the local policy. Formally, $\mathcal{P} : \wp(Assertions) \rightarrow \wp(Assertions)$. For simplicity, we assume that all nodes have the same security policy $\mathcal{P}$. Notice that the outcome of the policy function could differ from one node to another as the computation depends on the local knowledge of nodes.

Messages exchanged among nodes are assumed to be encrypted using a hierarchical key generation and distribution protocol [30, 56]. The trust manager may determine a key redistribution when a security level is compromised. More generally, re-keying [15] allows to refresh a subset of keys when one or more nodes join or leave the network; in this manner nodes are enable to decrypt past traffic, while evicted nodes are unable to decrypt future traffic. As showed in [56] re-keying may be relatively inexpensive if based on “low-cost” hashing operators.

4 The calculus

In Table 2, we define the syntax of our calculus in a two-level structure, a lower one for *processes* and an upper one for *networks*. We use letters $k, l, m, n, \ldots$ for node names. The symbol σ ranges over the security levels low and high, the only ones which are directly used by programmers. We use letters x, y, z for *variables*, u for *values*, and v and w for *closed values*, i.e. values that do not contain free variables. We write $\tilde{u}$ to denote a tuple $u_1, \ldots, u_k$ of values. For convenience, we sometime use angled brackets to delimit tuples, by writing $\langle u_1, \ldots, u_k\rangle$ instead of $u_1, \ldots, u_k$.

Networks are collections of nodes (which represent devices) running in parallel and

Table 3 Structural Congruence

$m[P + Q]_T \equiv m[Q + P]_T$	(Struct Sum Comm)
$m[P + (Q + Q')]_T \equiv m[(P + Q) + Q']_T$	(Struct Sum Assoc)
$m[P + \mathsf{nil}]_T \equiv m[P]_T$	(Struct Sum Zero)
$M \mid N \equiv N \mid M$	(Struct Par Comm)
$(M \mid N) \mid M' \equiv M \mid (N \mid M')$	(Struct Par Assoc)
$M \mid \mathbf{0} \equiv M$	(Struct Par Zero)
$M \equiv M$	(Struct Refl)
$M \equiv N$ implies $N \equiv M$	(Struct Symm)
$M \equiv N \wedge N \equiv O$ implies $M \equiv O$	(Struct Trans)
$M \equiv N$ implies $M \mid M' \equiv N \mid M'$, for all M'	(Struct Cxt Par)

using channels at different security levels to communicate with each other. We use the symbol $\mathbf{0}$ to denote the empty network. We write $n[P]_T$ for a node named n (denoting its network address) executing the sequential process P, with a local trust store T. We write $M \mid N$ for the parallel composition of two sub-networks M and N.

Processes are sequential and live within the nodes. We write nil to denote the skip process. The multicast sender process $\sigma!\langle\tilde{v}\rangle.P$ transmits the message $\tilde{v}$ to all neighbours at security level σ, and then continues as P. The unicast sender process $\sigma!\langle\tilde{v}\rangle_n.P$ transmits the message $\tilde{v}$ to node n at security level σ, and then continues as P. The receiver process $\sigma?(\tilde{x}).P$ listens for incoming communications at security level σ. Upon reception, the receiver processes evolves into P, where the variables of $\tilde{x}$ are replaced with the message $\tilde{v}$. We write $\{^{\tilde{v}}/_{\tilde{x}}\}P$ for the substitution of variables $\tilde{x}$ with values $\tilde{v}$ in P, with $\mid \tilde{x} \mid = \mid \tilde{v} \mid$. In process $[\tilde{v}\,\mathsf{op}\,\tilde{w}]P, Q$ the metavariable op denotes a binary operator, returning a boolean, such as $=$, $<$, $>$, $\leq$, $\geq$, $\in$, $\subseteq$. The process $[\tilde{v}\,\mathsf{op}\,\tilde{w}]P, Q$ denotes the "if then else" construct: it behaves as P if $\tilde{v}\,\mathsf{op}\,\tilde{w} = \mathtt{true}$, and as Q otherwise. Process $P + Q$ denotes standard nondeterministic choice. In processes $\sigma?(\tilde{x}).P$, $\sigma!\langle\tilde{v}\rangle.P$ and $\sigma!\langle\tilde{v}\rangle_n.P$ the occurrence of process P is said to be *guarded*. We write $H\langle\tilde{v}\rangle$ to denote a process defined via a definition $H(\tilde{x}) \overset{\text{def}}{=} P$, with $\mid \tilde{x} \mid = \mid \tilde{v} \mid$, where $\tilde{x}$ contains all variables that appear free in P. Defining equations provide *guarded recursion*, since P may contain only guarded occurrences of process identifiers. In process $\sigma?(\tilde{x}).P$ variables $\tilde{x}$ are bound in P. This gives rise to the standard notion of α-conversion and free and bound variables. We assume there are no free variables in our networks. The absence of free variables in networks is trivially maintained as the network evolves. Given a network M, $\mathsf{nds}(M)$ returns the set of nodes which constitute the network M. Notice that, as networks addresses are unique, we assume that there cannot be two nodes with the same name in the same network. We write $\prod_i M_i$ to denote the parallel composition of all sub-networks M_i. We write $\sigma!\langle\tilde{v}\rangle$ and $\sigma!\langle\tilde{v}\rangle_n$ to mean $\sigma!\langle\tilde{v}\rangle.\mathsf{nil}$ and $\sigma!\langle\tilde{v}\rangle_n.\mathsf{nil}$, respectively. As usual in process calculi, in Table 3 we define *structural congruence*, written $\equiv$, to identify processes up to reordering of parallel and nondeterministic processes.

Table 4 LTS - Synchronisation

$$\text{(MCast)}\ \frac{\mathcal{D} := \{n : T(m,n) \geq \sigma\} \quad \mathcal{D} \neq \emptyset}{m[\sigma!\langle\tilde{v}\rangle.P]_T \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_{\sigma} m[P]_T} \qquad \text{(UCast)}\ \frac{T(m,n) \geq \sigma}{m[\sigma!\langle\tilde{v}\rangle_n.P]_T \xrightarrow{m!\tilde{v}\triangleright n}_{\sigma} m[P]_T}$$

$$\text{(Rcv)}\ \frac{T(n,m) \geq \sigma \quad \mid \tilde{x} \mid = \mid \tilde{v} \mid \quad m \neq n}{n[\sigma?(\tilde{x}).P]_T \xrightarrow{m?\tilde{v}\triangleright n}_{\sigma} n[\{^{\tilde{v}}/_{\tilde{x}}\}P]_T} \qquad \text{(RcvEnb)}\ \frac{m \notin \mathsf{nds}(M) \quad \rho \geq \mathsf{trust}}{M \xrightarrow{m?\tilde{v}\triangleright\emptyset}_{\rho} M}$$

$$\text{(RcvPar)}\ \frac{M \xrightarrow{m?\tilde{v}\triangleright\mathcal{D}}_{\rho} M' \quad N \xrightarrow{m?\tilde{v}\triangleright\mathcal{D}'}_{\rho} N'}{M \mid N \xrightarrow{m?\tilde{v}\triangleright\mathcal{D}\cup\mathcal{D}'}_{\rho} M' \mid N'}$$

$$\text{(Sync)}\ \frac{M \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_{\rho} M' \quad N \xrightarrow{m?\tilde{v}\triangleright\mathcal{D}'}_{\rho} N' \quad \mathcal{D}' \subseteq \mathcal{D}}{M \mid N \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_{\rho} M' \mid N'}$$

4.1 The operational semantics

We give the operational semantics of our calculus in terms of a *labelled transition system* (LTS). We have divided our LTS in two sets of rules. Table 4 contains the rules to model synchronisations between sender and receivers. Table 5 contains the rules to model trust management.

Our transitions take the form

$$M \xrightarrow{\lambda}_{\rho} N$$

to indicate that network M performs action λ, at security level ρ, evolving into network N. By construction, in any transition of this form ρ will be always different from bad. More precisely, ρ will be equal to low for low-level security transmissions, and equal to high for high-level security transmissions. If $\rho = \mathsf{trust}$ then the transition models some aspect of trust management, and involves only trusted nodes. The metavariable λ ranges over the labels $m!\tilde{v}\triangleright\mathcal{D}$, $m?\tilde{v}\triangleright\mathcal{D}$, and τ, where $\mathcal{D}$ is a set of nodes. We sometimes write $m!\tilde{v}\triangleright n$ and $m?\tilde{v}\triangleright n$ as an abbreviation for $m!\tilde{v}\triangleright\{n\}$ and $m?\tilde{v}\triangleright\{n\}$, respectively. The label $m!\tilde{v}\triangleright\mathcal{D}$ models the transmission of message $\tilde{v}$, originating from node m, and addressed to the set of nodes contained in $\mathcal{D}$. The label $m?\tilde{v}\triangleright\mathcal{D}$ represents the reception of a message $\tilde{v}$, sent by m, and received by the nodes contained in $\mathcal{D}$. The label τ models internal actions, which cannot be observed.

Remark 4.1 *Messages exchanged among nodes are assumed to be encrypted using a hierarchical key generation and distribution protocol [30, 56]. Thus, a message transmitted at security level ρ can be decrypted only by nodes at security level ρ or greater, according to the trust store of both sender and receiver. Moreover, we assume that messages are always signed by transmitters.*

Let us comment on the rules of Table 4. Rule (MCast) models a node m sending a message $\tilde{v}$ at security level σ; the set $\mathcal{D}$ contains the destination nodes with security level at least σ, according to the trust store of m.[1] Rule (UCast) models a unicast transmission of message $\tilde{v}$ from node m to node n, at security level σ. Rule (Rcv) models a node n receiving a message $\tilde{v}$, sent by node m, at security level σ. In this rule, n receives a message from m only if it trusts m at security level σ. Rule (RcvPar) serves to put together parallel nodes receiving from the same sender. Rule (RcvEnb) says that every node can synchronise with an external transmitter m. This rule, together with rule (RcvPar), serves to model message loss. If sender and receiver(s) trust each other then they may synchronise by applying rule (Sync). In this rule, the condition $\mathcal{D}' \subseteq \mathcal{D}$ ensures that only authorised recipients receive the transmitted value. Rule (Sync) has its symmetric counterpart.

Let us explain the rules in Table 4 with an example.

Example 4.2 *Let us consider the network:*

$$M \stackrel{\text{def}}{=} k[\sigma?(\tilde{x}).P_k]_{T_k} \;\; \big| \;\; l[\sigma?(\tilde{x}).P_l]_{T_l} \;\; \big| \;\; m[\sigma!\langle\tilde{v}\rangle.P_m]_{T_m} \;\; \big| \;\; n[\sigma?(\tilde{x}).P_n]_{T_n}$$

where $T_k(k,m) \geq \sigma$, $T_l(l,m) < \sigma$, $T_m(m,n) = T_m(m,l) \geq \sigma$, $T_m(m,k) < \sigma$ and $T_n(n,m) \geq \sigma$. In this configuration, node m transmit message $\tilde{v}$ at security level σ, being n and l the nodes allowed to receive that message at that security level. However, since l does not trust m, at security level σ, node n is the only node that may receive the message. Thus, by an applying rules (MCast), (Rcv), (RcvEnb), and (Sync) we have:

$$M \xrightarrow{m!\tilde{v}\triangleright\{l,n\}}_{\sigma} k[\sigma?(\tilde{x}).P_k]_{T_k} \;\; \big| \;\; l[\sigma?(\tilde{x}).P_l]_{T_l} \;\; \big| \;\; m[P_m]_{T_m} \;\; \big| \;\; n[\{^{\tilde{v}}/_{\tilde{x}}\}P_n]_{T_n} \;\; .$$

Now, let us comment on the rules of Table 5. We recall that each node comes with a trust manager component which is not specified in the syntax. Thus, Table 5 provide the transition rules to model the semantics of these components. The transmissions described in this table are addressed to all trusted nodes i.e. all nodes at security level trust. Rule (DTrust) models *direct trust*. This happens when the *monitoring module* of a node m, while monitoring the activity of a trusted node n, detects a misbehaviour of n. In this case, node m executes two operations: (i) it implements node revocation updating its trust store, according to its local policy; (ii) it broadcasts the corresponding information to inform all *trusted* nodes about the misbehaviour of n. Rule (SndRcm) describes *indirect trust* and models the sending of a recommendation. This may happen, for example, when a node moves and asks for recommendations on new neighbours. Again, recommendations are addressed to all trusted nodes, according to the trust knowledge of the recommender. Rule (RcvRcm) models the reception of a recommendation from a trusted node: a new trust table T' is calculated, applying the local policy to $T \cup (m, \tilde{v})$. Rule (Lose) models loss of trust information. This may happen, for instance, when a node moves, changing its neighbourhood. In this case,

[1] Rule (MCast) may recall the Directed Diffusion approach of [31].

Table 5 LTS - Trust Management

$$\text{(DTrust)}\ \frac{\begin{array}{c} T(m,n) \geq \mathsf{trust} \quad \tilde{v} := n, \mathsf{bad} \\ T' := \mathcal{P}(T \cup (m,\tilde{v})) \quad \mathcal{D} := \{k : T(m,k) \geq \mathsf{trust}\} \end{array}}{m[P]_T \xrightarrow{m!\tilde{v} \triangleright \mathcal{D}}_{\mathsf{trust}} m[P]_{T'}}$$

$$\text{(SndRcm)}\ \frac{T(m,n) = \rho \quad \tilde{v} := n, \rho \quad \mathcal{D} := \{n : T(m,n) \geq \mathsf{trust}\}}{m[P]_T \xrightarrow{m!\tilde{v} \triangleright \mathcal{D}}_{\mathsf{trust}} m[P]_T}$$

$$\text{(RcvRcm)}\ \frac{T(n,m) \geq \mathsf{trust} \quad \tilde{v} := l, \rho \quad T' := \mathcal{P}(T \cup (m,\tilde{v}))}{n[P]_T \xrightarrow{m?\tilde{v} \triangleright n}_{\mathsf{trust}} n[P]_{T'}}$$

$$\text{(Lose)}\ \frac{T' \subseteq T \quad T'' := \mathcal{P}(T')}{n[P]_T \xrightarrow{\tau}_{\mathsf{trust}} n[P]_{T''}}$$

trust information concerning old neighbours should be removed as it cannot be verified any longer.

Table 6 contains the standard rules for matching and recursion. It also contains the rule (TauPar) to propagate τ-actions over parallel components and the standard rule Rule (Sum) for nondeterministic choice. Rules (TauPar) and (Sum) have their symmetric counterparts.

Let us show how direct and indirect trust are modelled in our setting with an example.

Example 4.3 *Let us consider the network:*

$$M \stackrel{\mathrm{def}}{=} k[P_k]_{T_k} \;\;|\;\; l[P_l]_{T_l} \;\;|\;\; m[P_m]_{T_m} \;\;|\;\; n[P_n]_{T_n}$$

where $T_k(k,m) \geq \mathsf{trust}$*,* $T_l(l,m) = \mathsf{bad}$*,* $T_m(m,k) = T_m(m,l) = T_m(m,n) \geq \mathsf{trust}$*, and* $T_n(n,m) \geq \mathsf{trust}$*. Now, if node* m *observes that node* k *is misbehaving, then (i) it adds an assertion* (m,k,bad) *to its local knowledge; (ii) it broadcasts the information to its neighbours. Thus, by an application of rules (DTrust), (RcvRcm), (RcvEnb), and (Sync) we have*

$$M \xrightarrow{m!\langle k,\mathsf{bad}\rangle \triangleright \{k,l,n\}}_{\mathsf{trust}} k[P_k]_{T'_k} \;\;|\;\; l[P_l]_{T_l} \;\;|\;\; m[P_m]_{T'_m} \;\;|\;\; n[P_n]_{T'_n} \;\; .$$

Notice that since l *does not trust* m*, only node* n *and the bad node* k *receives* m*'s recommendation. Moreover the local knowledge of* m *and* n *will change, according to their local policy. This is a case of direct trust for* m*, and indirect trust for* n*.*

Table 6 LTS - Matching, recursion, parallel composition and summation.

$$\text{(Then)}\ \frac{n[P]_T \xrightarrow{\lambda}_{\rho} n[P']_{T'} \quad \tilde{v_1}\,\mathtt{op}\,\tilde{v_2} = \mathtt{true}}{n[[\tilde{v_1}\,\mathtt{op}\,\tilde{v_2}]P, Q]_T \xrightarrow{\lambda}_{\rho} n[P']_{T'}}$$

$$\text{(Else)}\ \frac{n[Q]_T \xrightarrow{\lambda}_{\rho} n[Q']_{T'} \quad \tilde{v_1}\,\mathtt{op}\,\tilde{v_2} = \mathtt{false}}{n[[\tilde{v_1}\,\mathtt{op}\,\tilde{v_2}]P, Q]_T \xrightarrow{\lambda}_{\rho} n[Q']_{T'}}$$

$$\text{(Rec)}\ \frac{n[\{\tilde{v}/\tilde{x}\}P]_T \xrightarrow{\lambda}_{\rho} n[P']_{T'} \quad H(\tilde{x}) \overset{\text{def}}{=} P}{n[H\langle\tilde{v}\rangle]_T \xrightarrow{\lambda}_{\rho} n[P']_{T'}}$$

$$\text{(TauPar)}\ \frac{M \xrightarrow{\tau}_{\rho} M'}{M \mid N \xrightarrow{\tau}_{\rho} M' \mid N} \qquad \text{(Sum)}\ \frac{m[P]_T \xrightarrow{\lambda}_{\sigma} m[P']_T}{m[P+Q]_T \xrightarrow{\lambda}_{\sigma} m[P']_T}$$

5 Node mobility

In wireless networks, node mobility is associated with the ability of a node to access telecommunication services at different locations from different nodes. Unlike wired networks, where the main security requirements are addressed by installing firewalls, in mobile ad hoc networks node mobility introduces new issues related to user credential management, indirect trust establishment and mutual authentication between previously unknown and hence untrusted nodes.

For these reasons, node mobility in ad hoc networks has turned to be a challenge for automated verification and analysis techniques. After the first work on model checking of (stationary) ad hoc networks [6], Nanz and Hankin [44] have proposed a process calculus where topology changes are abstracted into a *fixed* representation. This representation, called *network topology*, is essentially a set of *connectivity graphs* denoting the possible connectivities within the nodes of the network. Thus, in [44] the topology is not part of the syntax, but it is a parameter of the operational semantics. A similar approach has been proposed in [23], although the labelled transition systems and the equivalence relations proposed in the two papers are completely different. In [44] a transition to the next state is examined for all possible valid graphs (those contained in the network topology fixed a priori) whereas in [23] a transition is examined for all graphs containing the connections used in a communication. These connections are called *restrictions*.

As the reader may have noticed, our calculus does not directly model the network topology neither in the syntax nor in the semantics. However, it is very easy to modify our labelled transition system to add topology changes at semantics level. In Table 7 we rewrite the rules of Table 4 in the style of [23]. Rules take the form

$$M \xrightarrow{\lambda}_{\rho,C} M'$$

Table 7 LTS - Synchronisation with network restrictions

$$\text{(MCastR)}\ \frac{\mathcal{D}{:=}\{n : T(m,n) \geq \sigma\} \quad \mathcal{D} \neq \emptyset}{m[\sigma!\langle\tilde{v}\rangle.P]_T \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_{\sigma,\emptyset} m[P]_T}$$

$$\text{(UCastR)}\ \frac{T(m,n) \geq \sigma}{m[\sigma!\langle\tilde{v}\rangle_n.P]_T \xrightarrow{m!\tilde{v}\triangleright n}_{\sigma,\emptyset} m[P]_T}$$

$$\text{(RcvR)}\ \frac{T(n,m) \geq \sigma \quad \mid \tilde{x} \mid = \mid \tilde{v} \mid \quad m \neq n}{n[\sigma?(\tilde{x}).P]_T \xrightarrow{m?\tilde{v}\triangleright n}_{\sigma,(n,m)} n[\{^{\tilde{v}}/_{\tilde{x}}\}P]_T}$$

$$\text{(RcvEnbR)}\ \frac{m \notin \mathsf{nds}(M)}{M \xrightarrow{m?\tilde{v}\triangleright\emptyset}_{\rho,\emptyset} M}$$

$$\text{(RcvParR)}\ \frac{M \xrightarrow{m?\tilde{v}\triangleright\mathcal{D}}_{\rho,C_1} M' \quad N \xrightarrow{m?\tilde{v}\triangleright\mathcal{D}'}_{\rho,C_2} N'}{M \mid N \xrightarrow{m?\tilde{v}\triangleright\mathcal{D}\cup\mathcal{D}'}_{\rho,C_1\cup C_2} M' \mid N'}$$

$$\text{(SyncR)}\ \frac{M \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_{\rho,C_1} M' \quad N \xrightarrow{m?\tilde{v}\triangleright\mathcal{D}'}_{\rho,C_2} N' \quad \mathcal{D}' \subseteq \mathcal{D}}{M \mid N \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_{\rho,C_1\cup C_2} M' \mid N'}$$

indicating that network M performs the action λ, at security level ρ, under the network restriction C, evolving into network M'. Thus, a network restriction C keeps track of the connections which are necessary for the transition to fire. The rules in Table 5 (as well as those in Table 6) can be rewritten in a similar manner, except for rule (Lose) in which the network restriction must be the empty set.

Example 5.1 *Consider the network appearing in Example 4.2. Then, by applying rules (MCastR), (RcvR), (RcvEnbR), and (SyncR) we have*

$$M \xrightarrow{m!\tilde{v}\triangleright\{l,n\}}_{\sigma,\{(n,m)\}} k[\sigma?(\tilde{x}).P_k]_{T_k} \;\;\Big|\;\; l[\sigma?(\tilde{x}).P_l]_{T_l} \;\;\Big|\;\; m[P_m]_{T_m} \;\;\Big|\;\; n[\{^{\tilde{v}}/_{\tilde{x}}\}P_n]_{T_n} \;\; .$$

The transition is tagged with the network restriction $\{(n,m)\}$, as only node n has synchronised with node m.

The reader may have noticed that the rules of Table 7 do not use network restrictions in the premises. As a consequence, there is a straightforward operational correspondence between a transition $\xrightarrow{\lambda}_{\rho}$ and one of the form $\xrightarrow{\lambda}_{\rho,C}$.

Proposition 5.2

1. *$M \xrightarrow{\lambda}_{\rho} M'$ with $\lambda \in \{m!\tilde{v}\triangleright\mathcal{D}, m?\tilde{v}\triangleright\mathcal{D}\}$ iff there exists a restriction C such that $M \xrightarrow{\lambda}_{\rho,C} M'$ and $C \subseteq \{(m,n)$ for all $n \in \mathcal{D}\}$.*

2. $M \xrightarrow{\tau}_{\rho} M'$ *iff* $M \xrightarrow{\tau}_{\rho,\emptyset} M'$.

Proof By transition induction. □

6 Safety properties

Access control [54] is a well-established technique to provide *safety properties* ensuring that only principals with appropriate access rights can access data. In distributed security systems, safety properties ensure that no forbidden interactions arise [19]. In our setting, safety properties involve security levels, as communications are parameterised on them. Thus, our safety properties aim at guaranteeing that only authorised nodes receive sensible information.

We introduce the notion of *safety up to a security level* to describe when a communication is safe up to a certain security level. Intuitively, a communication is said to be *safe up to a security level* ρ if a node transmitting at level ρ may only synchronise with nodes at level ρ or above, according to the local knowledge of sender and receivers. This means that all parties involved in a transmission safe up to security level ρ trust each other at that security level.

The next theorem states that only safe communications are allowed in our calculus.

Theorem 6.1 (Safety preservation)

1. Let $M \xrightarrow{m?\tilde{v} \triangleright \mathcal{D}}_{\rho} M'$ *with* $M \equiv \prod_i n_i[P_i]_{T_i}$ *and* $M' \equiv \prod_i n_i[P'_i]_{T'_i}$.

(a) If $P'_i \neq P_i$*, for some* i*, then* $n_i \in \mathcal{D}$ *and* $T_i(n_i, m) \geq \rho$.

(b) If $T'_i \neq T_i$*, for some* i*, then* $n_i \in \mathcal{D}$ *and* $T_i(n_i, m) \geq \rho$.

2. Let $M \xrightarrow{m!\tilde{v} \triangleright \mathcal{D}}_{\rho} M'$ *with* $M{\equiv}m[P]_T \mid \prod_i n_i[P_i]_{T_i}$ *and* $M'{\equiv}m[P']_{T'} \mid \prod_i n_i[P'_i]_{T'_i}$.

(a) If $P'_i \neq P_i$*, for some* i*, then* $n_i \in \mathcal{D}$*,* $T(m, n_i) \geq \rho$ *and* $T_i(n_i, m) \geq \rho$.

(b) If $T'_i \neq T_i$*, for some* i*, then* $n_i \in \mathcal{D}$*,* $T(m, n_i) \geq \rho$ *and* $T_i(n_i, m) \geq \rho$.

Proof By transition induction. See the Appendix for details. □

A similar conformance criterion, called *safety despite compromised principals* (SDCP), has been proposed in [19]. According to this criterion, an invalid authorisation decision at an uncompromised node m may arise when some decision of m logically depends on one or more compromised nodes. A node is said to be *compromised* (or *bad*) when its privileges can be exercised by the attacker. A realistic threat model for a distributed system, such as an ad hoc network, should include *partial compromise*, that is the possibility that some of the nodes in the system are compromised. Partial compromise covers deliberate insider attacks as well as external attackers taking ownership of insiders' assets.

In our setting, the SDCP property comes as a consequence of Theorem 6.1, for which trusted nodes never synchronise with untrusted ones. In this manner, bad nodes (recognised as such) are isolated from the rest of the network and they cannot affect communications.

Corollary 6.2 (Safety despite compromised nodes)

1. *Let* $M \xrightarrow{m?\tilde{v}\triangleright\mathcal{D}}_{\rho} M'$ *with* $M \equiv \prod_i n_i[P_i]_{T_i}$ *and* $M' \equiv \prod_i n_i[P_i']_{T_i'}$*. If* $T_i(n_i, m) = \mathsf{bad}$*, for some* i*, then* $P_i'{=}P_i$ *and* $T_i'{=}T_i$*.*

2. *Let* $M \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_{\rho} M'$ *with* $M \equiv m[P]_T \mid \prod_i n_i[P_i]_{T_i}$ *and* $M' \equiv m[P']_{T'} \mid \prod_i n_i[P_i']_{T_i'}$*. If* $T(m, n_i) = \mathsf{bad}$ *or* $T_i(n_i, m) = \mathsf{bad}$*, for some* i*, then* $P_i'{=}P_i$ *and* $T_i'{=}T_i$*.*

Proof See the Appendix. □

7 Behavioural semantics

Our main behavioural equivalence between networks is a variant of Milner and Sangiorgi's (weak) barbed congruence [43] parameterised overs security levels. Basically, two terms are barbed congruent if they have the same *observables* (called *barbs*) in all possible contexts, under all possible *evolutions*. For the definition of barbed congruence we need two crucial concepts: a reduction semantics to describe how a system evolves, and a notion of observable which says what the environment can observe in a system.

From the LTS given in Section 4.1 it is easy to see that a network may evolve either because there is a transmission at a certain security level or because a node loses some trust information. Thus, we define the reduction relation $\rightarrowtriangle$ between networks using the following inference rules:

$$\text{(Red1)}\ \frac{M \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_{\rho} M'}{M \rightarrowtriangle M'} \qquad\qquad \text{(Red2)}\ \frac{M \xrightarrow{\tau}_{\mathsf{trust}} M'}{M \rightarrowtriangle M'}$$

We write $\rightarrowtriangle^*$ to denote the reflexive and transitive closure of $\rightarrowtriangle$.

Let us focus on the definition of an appropriate notion of observable. In our calculus, as in CCS [41] and in π-calculus [42], we have both transmission and reception of messages, although only transmissions can be observed. In fact, in a broadcasting calculus an observer cannot see whether a given process actually receives a broadcast synchronisation. In particular, if the node $m[\sigma!\langle\tilde{v}\rangle.P]_T$ evolves into $m[P]_T$ we do not know whether some potential recipient has synchronised with m. On the other hand, if a node $n[\sigma?(\tilde{x}).P]_T$ evolves into $n[\{\tilde{v}/\tilde{x}\}P]_T$, then we can be sure that some trusted node has transmitted a message $\tilde{v}$ to n at security level σ.

Following Milner and Sangiorgi [43] we use the term "barb" as synonymous of observable.

Definition 7.1 (σ-Barb) *We write* $M \downarrow_n^{\sigma}$ *if either* $M \equiv m[\sigma!\langle\tilde{v}\rangle.P]_T \mid N$ *or* $M \equiv m[\sigma!\langle\tilde{v}\rangle_n.P]_T \mid N$*, for some* $m, N, \tilde{v}, P, T$ *such that* $n \notin \mathsf{nds}(M)$*, and* $T(m,n) \geq \sigma$*. We write* $M \Downarrow_n^{\sigma}$ *if* $M \rightarrowtriangle^* M' \downarrow_n^{\sigma}$ *for some network* M'*.*

The barb $M \Downarrow_n^\sigma$ says that there is a potential transmission at security level σ, originating from M, that may reach the node n of the environment.

In the sequel, we write $\mathcal{R}$ to denote binary relations over networks.

Definition 7.2 (σ-Barb preserving) *A relation $\mathcal{R}$ is said to be* σ-barb preserving *if whenever $M \mathrel{\mathcal{R}} N$ it holds that $M \downarrow_n^\sigma$ implies $N \Downarrow_n^\sigma$.*

Definition 7.3 (Reduction closure) *A relation $\mathcal{R}$ is said to be* reduction closed *if $M \mathrel{\mathcal{R}} N$ and $M \rightarrowtriangle M'$ imply there is N' such that $N \rightarrowtriangle^* N'$ and $M' \mathrel{\mathcal{R}} N'$.*

As we are interested in weak behavioural equivalences, our definition of reduction closure is given in terms of weak reductions.

Definition 7.4 (Contextuality) *A relation $\mathcal{R}$ is said to be* contextual *if $M \mathrel{\mathcal{R}} N$ implies that $M \mid O \mathrel{\mathcal{R}} N \mid O$, for all networks O.*

Finally, everything is in place to define our touchstone behavioural equivalence.

Definition 7.5 (σ-Reduction barbed congruence) *The* σ-reduction barbed congruence*, written $\cong_\sigma$, is the largest symmetric relation over networks which is σ-barb preserving, reduction closed and contextual.*

8 Bisimulation proof method

The definition of σ-reduction barbed congruence is simple and intuitive. However, due to the universal quantification on parallel contexts, it may be quite difficult to prove that two terms are equivalent. Simpler proof techniques are based on labelled bisimilarities. In this section, we define an appropriate notion of bisimilarity. As a main result, we prove that our labelled bisimilarity is a proof-technique for our σ-reduction barbed congruence.

In general, a labelled bisimilarity describes how two terms (in our case networks) can mimic each other's actions. As we are interested in weak behavioural equivalences, we have to distinguish between transmissions that can be observed and transmissions that cannot be observed by the environment. We do that by introducing two extra rules in the labelled transition system:

$$\text{(Shh)}\ \frac{M \xrightarrow{m!\tilde{v} \triangleright \mathcal{D}}_{\rho} M' \quad \mathcal{D} \subseteq \mathsf{nds}(M) \quad \rho' \geq \mathsf{trust}}{M \xrightarrow{\tau}_{\rho'} M'} \qquad \text{(Obs)}\ \frac{M \xrightarrow{m!\tilde{v} \triangleright \mathcal{D}}_{\rho} M' \quad \mathcal{D}' := \mathcal{D} \backslash \mathsf{nds}(M) \neq \emptyset}{M \xrightarrow{m!\tilde{v} \blacktriangleright \mathcal{D}'}_{\rho} M'}$$

Rule (Shh) models transmissions that cannot be observed because none of the potential receivers is in the environment. Notice that security level of silent actions is not related to the transmissions they originate from. Rule (Obs) models a transmission, at security level ρ, of a message $\tilde{v}$, from a sender m, that may be received, and hence observed, by the nodes of the environment (i.e. those in $\mathcal{D} \setminus \mathsf{nds}(M)$). Notice that the rule (Obs) can be applied in a derivation tree only at top-level.

In the sequel, we use the metavariable α to range over the following actions: τ, $m?\tilde{v}{\rhd}\mathcal{D}$ and $m!\tilde{v}{\blacktriangleright}\mathcal{D}$. Since we are interested in *weak behavioural equivalences*, that abstract over τ-actions, we introduce weak actions in a standard manner: we write $\Rightarrow_\rho$ to denote the reflexive and transitive closure of $\xrightarrow{\tau}_\rho$; the weak transition $\xRightarrow{\alpha}_\rho$ is an abbreviation for $\Rightarrow_\rho \xrightarrow{\alpha}_\rho \Rightarrow_\rho$, while $\xRightarrow{\hat{\alpha}}_\rho$ denotes $\Rightarrow_\rho$ if $\alpha = \tau$ and $\xRightarrow{\alpha}_\rho$ otherwise.

Definition 8.1 (δ-Bisimilarity) *The* δ-bisimilarity*, written $\approx_\delta$, is the largest symmetric relation over networks such that whenever $M \approx_\delta N$ and $M \xrightarrow{\alpha}_\rho M'$, with $\rho \leq \delta$, there exists a network N' such that $N \xRightarrow{\hat{\alpha}}_\rho N'$ and $M' \approx_\delta N'$.*

This definition is inspired by that proposed in [14]. Intuitively, two networks are δ-bisimilar if they cannot be distinguished by any observer that can perform actions at security level at most δ.

Remark 8.2 *Notice that we can redefine the δ-bisimilarity using a labelled transition system with network restrictions as suggested in Section 5. However, in Proposition 5.2 we already proved the operational correspondence between the two labelled transition systems. As a consequence, the resulting bisimilarity would not change.*

It is worth noticing that bisimilar networks must have the same set of nodes.

Proposition 8.3 *If $M \approx_\delta N$, for some δ, then $\mathsf{nds}(M) = \mathsf{nds}(M)$.*

Proof By contradiction. Suppose there is a node m such that $m \in \mathsf{nds}(M)$ and $m \notin \mathsf{nds}(N)$. Then, by an application of rule (RcvEnb) we have $N \xrightarrow{m?\tilde{v}{\rhd}\emptyset}_\rho N$, for all $\rho \geq \mathsf{trust}$. Since $M \approx_\delta N$ there must be M' such that $M \xRightarrow{m?\tilde{v}{\rhd}\emptyset}_\rho M'$, with $M' \approx N'$. However, since $m \in \mathsf{nds}(M)$, by inspection on the transition rules, there is no way to deduce such a weak transition for M. □

In the next result we show that our bisimilarity is a congruence. This result allows us to reason on networks in a modular fashion.

Theorem 8.4 ($\approx_\delta$ is contextual) *Let M and N be two networks such that $M \approx_\delta N$. Then $M \mid O \approx_\delta N \mid O$ for all networks O.*

Proof We prove that the relation

$$\mathcal{S} \stackrel{\text{def}}{=} \{\big(M \mid O \, , \, N \mid O\big) \text{ for all } O \text{ such that } M \approx_\delta N\}$$

is a δ-bisimulation. By case analysis on the transition $M \mid O \xrightarrow{\alpha}_\rho \widehat{M}$, with $\rho \leq \delta$. See the Appendix for details. □

The previous result is fundamental to prove that the labelled bisimilarity is a sound proof technique for the σ-reduction barbed congruence.

Theorem 8.5 (Soundness) *Let M and N be two networks such that $M \approx_{\delta} N$. Then $M \cong_{\sigma} N$, for $\sigma \leq \delta$.*

Proof We have to prove that the δ-bisimilarity is σ-barb preserving, reduction-closed, and contextual. The σ-barb preserving follows because bisimilar networks can mimic each other transmissions. The reduction-closure follows by definition, while contextuality follows by Theorem 8.4. For details see the Appendix. □

9 Non-interference

Information flow properties are a particular class of security properties for controlling the information flow among different entities. The seminal idea of *non-interference* proposed in [26] aims at assuring that "*variety in secret inputs should not be conveyed to public outputs*". In a multilevel system [5] this property says that information can only flow from low levels to higher ones. The first taxonomy of non-interference-like properties has been uniformly defined and compared in [16, 17] in the context of CCS-like process calculus. In [16, 17], processes are divided into high-level and low-level processes, according to the level of actions they can perform. To detect whether an incorrect information flow (i.e. from high-level to low-level) occurs, a particular non-interference-like property has been defined, the so-called *Non Deducibility on Composition* (NDC). This property basically says that a process is secure with respect to wrong information flows if its low-level behaviour is independent of changes to its high-level behaviour. Here, we prove a non-interference result using our notion of δ-bisimilarity. In Definition 9.1 we formalise the concept of high-level behaviour introducing the notion of high-level network. We recall that actions at security level trust do not depend on the syntax but they only depend on the trust management component; thus, these actions can fire at any moment of the computation.

Definition 9.1 (δ-high level network) *A network H is a δ-*high level network*, written $H \in \mathcal{H}_{\delta}$, if whenever $H \xrightarrow{\lambda}_{\delta'} H'$ then either (i) $\delta' = \mathsf{trust}$, or (ii) $\lambda = m?\tilde{v}\triangleright\emptyset$ and $H' = H$, or (ii) $\delta' > \delta$. Moreover, $H' \in \mathcal{H}_{\delta}$.*

The non-interference result is stated below. Intuitively, if two δ-bisimilar networks M and N run in parallel with two trust-bisimilar high-level networks H and K, then the resulting networks $M \mid H$ and $N \mid K$ are δ-bisimilar as well.

Theorem 9.2 (Non-interference) *Let M and N be two networks such that $M \approx_{\delta} N$. Let H and K be two networks such that: (i) $H, K \in \mathcal{H}_{\delta}$ and (ii) $H \approx_{\mathsf{trust}} K$. Then, $M \mid H \approx_{\delta} N \mid K$.*

Proof We prove that the relation

$$\mathcal{S} \stackrel{\mathrm{def}}{=} \{\big(M \mid H \,,\, N \mid K\big) : H, K \in \mathcal{H}_\delta,\ M \approx_\delta N \text{ and } H \approx_{\mathsf{trust}} K\}$$

is a δ-bisimulation. By case analysis on the transition $M \mid H \xrightarrow{\alpha}_\rho \widehat{M}$, with $\rho \leq \delta$. See the Appendix for details. □

10 Case studies

In this section, we use our calculus to specify a trust-based version of the AODV routing protocol [47], and a trust-based version of the leader election algorithm for MANETs proposed in [60]. In these two encodings, both routing paths and leader election are associated with at a security level σ of the nodes involved in the procedure. This is the essence of trust-based distributed algorithms in a multilevel network, where information at a given security level ρ may only travel along nodes with a security level greater or equal than ρ.

A subnetwork of a network M is said to be a *connected component* of M if all nodes are connected to each other via one or more hops. Since we are working in a multilevel scenario we define a *σ-connected component* as a connected component where neighbouring nodes trust each other at security level (at least) σ.

Definition 10.1 (σ-connected component) *Let M be a network. We say that N is a σ-*connected component *of M if*

- *$M \equiv N \mid M'$, for some network M';*
- *for all $m, n \in \mathsf{nds}(N)$ there is a sequence of nodes $m_1, \ldots, m_k \in \mathsf{nds}(N)$, with $N \equiv m_1[P_1]_{T_1} \mid \ldots \mid m_k[P_k]_{T_k} \mid N'$, such that $m = m_1$, $n = m_k$, $T_i(m_i, m_{i+1}) \geq \sigma$ and $T_i(m_{i+1}, m_i) \geq \sigma$, for $1 \leq i \leq k{-}1$.*

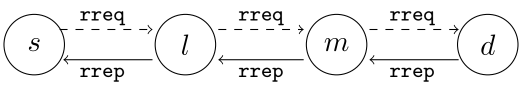
Our encodings are trust-based version of AODV and leader election as they both succeed only within a σ-connected component.

10.1 A trust-based variant of the AODV routing protocol

Ad hoc networks rely on multi-hop wireless communications where nodes have essentially two roles: (i) acting as end-systems, and (ii) performing routing functions. A routing protocol is used to determine the appropriate paths on which data should be transmitted in a network. Routing protocols for wireless systems can be classified into *topology-based* and *position-based.* Topology-based protocols rely on traditional routing concepts, such as maintaining routing tables or distributing link-state information. Position-based protocols use information about the physical locations of the nodes to route data packets to their destinations. Topology-based protocols can be divided into

Figure 1 The AODV routing protocol

$$\begin{array}{rcll} s & \longrightarrow & * & : \ \mathtt{rreq}, s, \mathit{Rid}, d, \mathit{Sseq}, \mathit{Dseq}, 0 \\ l & \longrightarrow & * & : \ \mathtt{rreq}, s, \mathit{Rid}, d, \mathit{Sseq}, \mathit{Dseq}, 1 \\ m & \longrightarrow & * & : \ \mathtt{rreq}, s, \mathit{Rid}, d, \mathit{Sseq}, \mathit{Dseq}, 2 \\ d & \longrightarrow & m & : \ \mathtt{rrep}, s, d, \mathit{Dseq}', 0 \\ m & \longrightarrow & l & : \ \mathtt{rrep}, s, d, \mathit{Dseq}', 1 \\ l & \longrightarrow & s & : \ \mathtt{rrep}, s, d, \mathit{Dseq}', 2 \end{array}$$



proactive and *reactive* protocols. Proactive routing protocols try to maintain consistent routing information within the system at any time. In reactive routing protocols, a route is established between a source and a destination only when it is needed. For this reason, reactive protocols are also called *on-demand* protocols. Examples of proactive routing protocols for MANETs are OLSR [13] and DSDV [48], while DSR [32] and AODV [47] are on-demand protocols.

In this section, we use our calculus to formalise a trust-based version of the AODV routing protocol.

In the AODV protocol each node maintains a *routing table* (*RT*) containing informations about the routes to be followed when sending messages to the other nodes of the network. In particular, for each destination node n a routing table should provide an entry containing the following information:

- the name of the destination node (i.e. n)
- the name of the neighbour node to send messages addressed to n
- the number of hops necessary to reach n
- the destination sequence number associated with n, to determine whether the path is up-to-date
- the expiration time for that entry.

Each node maintains also a *local history table* (*HT*) containing pairs of the form (*source-name*, *request-id*) to discard request packets which have already been processed.

In Figure 1, we report a scheme of the AODV protocol with four nodes: a source s, a destination d and two intermediate nodes l and m. We also provide a graphical representation of the flow of messages: dashed arrows denote the broadcast of *route request packets* (`rreq`), while continuous arrows denote the unicast sending of *route reply packets* (`rrep`). More precisely, suppose the source node s wishes to send a message to the destination node d. In order to perform the sending, s will look up an entry for d in its routing table. If there is no such an entry it will launch a route discovery procedure to find a route to d. The protocol works as follows:

- The source s broadcasts a route request packet of the form

  $$\langle \texttt{rreq}, s, \mathit{Rid}, d, \mathit{Sseq}, \mathit{Dseq}, \mathit{hc} \rangle \ .$$

  Here, the fields s and d denote the IP addresses of source and destination, respectively. The field *Rid* denotes a *request-id*, that is a sequence number uniquely identifying the request. The *Sseq* field contains the *source sequence number*, i.e. the current sequence number to be used in routing table entries pointing towards the source node s. The *Dseq* field is the *destination sequence number* containing the latest sequence number received in the past by the source node s for any route towards the destination d; this number is 0 if d is unknown to s. The *hop-count* field *hc* keeps track of the number of hops from the source node to the node handling the request. Initially, this field is set to 0.

- When the intermediate node l receives the route request, it acts as follows:

  - It looks up the pair (s, *Rid*) in its local history table to verify whether the request has already been processed. If this is the case, the request is discarded and the processing stops. Otherwise, the pair is entered into the local history table, so that future requests from s with the same *Rid* will be discarded.
  - Then, l looks up an entry for d in its routing table. If there is such an entry, with destination sequence number greater than or equal to the *Dseq*, then a route reply packet is sent back to the source saying to use l itself to get to the destination d. Otherwise, it re-broadcasts the route request packet with the *hc* field incremented by one.
  - In any case, l compares the source sequence number *Sseq* contained in the request with the one appearing in its routing table associated with node s. If *Sseq* is more recent (i.e. greater) than the one in the table, l updates its routing table entry associated with s.

- Node m will repeat the same steps executed by node l and re-broadcasts the route request packet.

- Whenever the destination d receives the route request, it sends to m a unicast reply packet of the form

  $$\langle \texttt{rrep}, s, d, \mathit{Dseq}', \mathit{hc}, \mathit{lt} \rangle \ .$$

Here, the source address and the destination address are copied from the incoming request, while the destination sequence number is that associated with the route. The *hop-count* field is set to 0. The *lifetime* field contains the time in milliseconds for which nodes receiving the `rrep` consider the route to be valid.

- The reply packet then follows the reverse path towards node s increasing the hc field at each hop. Each node receiving the reply packet will update the routing table entry associated with d if one of the following conditions is met:

  - No route to d is known;
  - The sequence number for d in the route reply packet is greater than that stored in the routing table;
  - The sequence numbers are equal but the new route is shorter.

  In this way, nodes on the reverse route learn the route to d.

Our encoding is a *trust-based variant* of the original protocol as paths are associated with security levels, and they are composed only by trusted nodes. As consequence, entries of routing tables and history tables also contains the security level of paths and requests, respectively.

Trust-based routing schemes generally separate nodes into two possible states: benevolent and malevolent nodes. In our model, there is no such discrete segregation and all nodes in the network are considered potential routing candidates based upon their current trust level. Thus, our scheme facilitates best-effort delivery even in the presence of malicious and selfish nodes. According to our operational semantics, before sending or receiving a message, a node verifies the security level of the participants. In practise, a node also checks the security level of the next hop. This ensures that both request messages and reply messages are forwarded on paths of trusted nodes, up to a certain security level.

In our trust-based encoding of AODV, we adopt a few simplifications. More specifically, we do not consider lifetime fields and route error messages. Lifetime would require a notion of time, while error messages could be easily modelled. Moreover, node disconnections are modelled using the choice operator of our calculus.

For convenience, we generalise the matching construct as follows:

$$[(\tilde{u}_1 \,\mathsf{op}\, \tilde{u}'_1)\mathtt{lc}\ldots\mathtt{lc}(\tilde{u}_k \,\mathsf{op}\, \tilde{u}'_k)]P, Q$$

where `lc` is a binary logical operator. Its operational semantics is straightforward. We also assume the following two functions: $\mathsf{Dseq}(\cdot,\cdot)$ and $\mathsf{Hcnt}(\cdot,\cdot)$. These functions take in input a routing table RT and a node identifier id. In particular, $\mathsf{Dseq}(RT, id)$ returns the destination sequence number of the entry for id in RT, whereas $\mathsf{Hcnt}(RT, id)$ returns the next hop node. In all cases, if there is no entry in RT associated with id, both functions return the undefined value $\perp$. In the sequel, we will use the routing table RT and the history table HT as partial mappings from node identifiers to tuples of data. We sometimes write $RT\{id \mapsto Dseq, hc, nh, \rho\}$ to denote the routing table RT in which the entry associated with the node id is updated with the information contained in $Dseq$, hc, nh and ρ. Similarly, we will write $HT\{id \mapsto Rid, \rho\}$ to update a history table.

In Figure 2 we provide an encoding of a trust-based variant of the AODV protocol in our calculus. Let us explain it in some detail.

At the beginning, each node can be in one of these two states:

- $\textsc{Source}(id_s, id_d, Sseq, Dseq, Rid, RT_s)$, when a source node initiates the protocol; in this state the node broadcasts a request message;
- $\textsc{Node}(id, RT, HT)$, when a node is ready to receive a request or a reply message.

While executing the protocol nodes may evolve into one of the following states:

- $\textsc{AwaitReply}(id_s, id_d, Sseq, Dseq, Rid, RT_s)$, the source node waits for the reply message;
- $\textsc{RouteSuccess}(id_s, id_d, RT)$, the source node has accepted the route;
- $\textsc{RRequest}(req, id_s, id_d, Sseq, Dseq, Rid, hc, id_p, id, RT, HT)$, an intermediate node receives a request message;
- $\textsc{RReply}(rep, id_s, id_d, Dseq, hc, nh, id_p, id, RT, HT)$, an intermediate node receives a reply message;
- $\textsc{SndReply}(id_s, id_d, Dseq, hc, id_p, id, RT, HT)$, a recipient node (a node that knows a route to the destination) or the destination node sends the reply message.

The source node s begins in state $\textsc{Source}\langle id_s, id_d, Sseq, Dseq, Rid, RT_s\rangle$ where id_s and id_d are the node-ids of the source and the destination, respectively; $Sseq$ and $Dseq$ are the source and destination sequence numbers, respectively, and RT_s is the routing table of s. In this state, the source node broadcasts a route request message of the form $\langle \texttt{rreq}, id_s, id_d, Sseq{+}1, Dseq, Rid{+}1, 0, id_s\rangle$. For convenience, the message contains also the node-id (the last element of the message) of the source in order to facilitate the storing of a new entry in the reverse routing table. After this transmission, the node evolves into the state $\textsc{AwaitReply}\langle id_s, id_d, Sseq, Dseq, Rid, RT_s\rangle$, waiting for a reply message of the form $\langle rep, id_s, id_d, Dseq', hc, nh\rangle$.

When the source receives a reply message for its request it checks whether the destination sequence number of the reply packet is greater than the one stored in its routing table (or the sequence numbers are equal and the hop counter is smaller). If this is the case the source node accepts the route, evolving into the state

$$\textsc{RouteSuccess}\langle id_s, id_d, RT_s\{id_d \mapsto Dseq', hc, nh, \sigma\}\rangle$$

meaning that the route from id_s to id_d has been accepted and the routing table is updated accordingly. Otherwise, the reply is dropped and the node returns into the

Figure 2 A trust-based encoding of the AODV protocol at security level σ

/* *Source node broadcasts a request message and evolves into the* AwaitReply *state waiting for replies.**/

$$\begin{array}{l}
\textsc{Source}(id_s, id_d, Sseq, Dseq, Rid, RT_s) \stackrel{\text{def}}{=} \\
\quad \sigma!\langle \mathtt{rreq}, id_s, id_d, Sseq{+}1, Dseq, Rid{+}1, 0, id_s\rangle.\textsc{AwaitReply}\langle id_s, id_d, Sseq, Dseq, Rid, RT_s\rangle
\end{array}$$

/* *Source node waits for reply messages; if a reply is successfully received the node accepts the route, evolving into the state* RouteSuccess. *Otherwise, it continues waiting.* */

$$\begin{array}{l}
\textsc{AwaitReply}(id_s, id_d, Sseq, Dseq, Rid, RT_s) \stackrel{\text{def}}{=} \\
\qquad \sigma?(rep, id'_s, id'_d, Dseq', hc, nh).[(rep = \mathtt{rrep}) \wedge (id_s = id'_s) \wedge (id_d = id'_d)] \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad [(Dseq' > Dseq) \vee (Dseq'{=}Dseq \ \wedge \ hc < \mathsf{Hcnt}(RT_s, id_d))] \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \textsc{RouteSuccess}\langle id_s, id_d, RT_s\{id_d \mapsto Dseq', hc, nh, \sigma\}\rangle, \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad \textsc{Source}\langle id_s, id_d, Sseq{+}1, Dseq, Rid{+}1, RT_s\rangle, \\
\qquad\qquad\qquad\qquad\qquad\qquad \textsc{AwaitReply}\langle id_s, id_d, Sseq, Dseq, Rid, RT_s\rangle \\
\qquad + \ \textsc{Source}\langle id_s, id_d, Sseq{+}1, Dseq, Rid{+}1, RT_s\rangle
\end{array}$$

/* *Intermediate nodes may receive either a request message or a reply message.* */

$$\begin{array}{l}
\textsc{Node}(id, RT, HT) \stackrel{\text{def}}{=} \\
\qquad \sigma?(req, id_s, id_d, Sseq, Dseq, Rid, hc, id_p).\textsc{RRequest}\langle req, id_s, id_d, Sseq, Dseq, Rid, hc, id_p, id, RT, HT\rangle \\
\qquad + \\
\qquad \sigma?(rep, id_s, id_d, Dseq, hc, nh).\textsc{RReply}\langle rep, id_s, id_d, Dseq, hc, nh, id_p, id, RT, HT\rangle
\end{array}$$

/* *An intermediate node receiving a request message check whether it can serve the request, by sending a reply message, or it should re-broadcast the request.* */

$$\begin{array}{l}
\textsc{RRequest}(req, id_s, id_d, Sseq, Dseq, Rid, hc, id_p, id, RT, HT) \stackrel{\text{def}}{=} \\
\qquad [(req = \mathtt{rreq}) \wedge (HT(id_s) \neq \langle Rid, \sigma\rangle)] \\
\qquad\qquad\quad [(id_d = id) \vee (\mathsf{Dseq}(RT, id_d) \geq Dseq)] \\
\qquad\qquad\qquad [(Sseq > \mathsf{Dseq}(RT, id_s)) \vee (\mathsf{Dseq}(RT, id_s){=}Sseq \wedge hc < \mathsf{Hcnt}(RT, id_s))] \\
\qquad\qquad\qquad\quad \textsc{SndReply}\langle id_s, id_d, \mathsf{Dseq}(RT, id_d), \mathsf{Hcnt}(RT, id_d), id_p, id, \\
\qquad\qquad\qquad\qquad\qquad RT\{id_s \mapsto \mathsf{Dseq}(RT, id_d), hc{+}1, id_p, \sigma\}, HT\{id_s \mapsto Rid, \sigma\}\rangle, \\
\qquad\qquad\qquad \textsc{SndReply}\langle id_s, id_d, \mathsf{Dseq}(RT, id_d), \mathsf{Hcnt}(RT, id_d), id_p, id, RT, HT\{id_s \mapsto Rid, \sigma\}\rangle, \\
\qquad\qquad\quad \sigma!\langle \mathtt{rreq}, id_s, id_d, Sseq, Dseq, Rid, hc{+}1, id\rangle. \\
\qquad\qquad\qquad [(Sseq > \mathsf{Dseq}(RT, id_s)) \vee (\mathsf{Dseq}(RT, id_s){=}Sseq \wedge hc < \mathsf{Hcnt}(RT, id_s))] \\
\qquad\qquad\qquad\quad \textsc{Node}\langle id, RT\{id_s \mapsto Sseq, hc{+}1, id_p\}, HT\{id_s \mapsto Rid, \sigma\}\rangle, \\
\qquad\qquad\qquad \textsc{Node}\langle id, RT, HT\{id_s \mapsto Rid, \sigma\}\rangle, \\
\qquad \textsc{Node}\langle id, RT, HT\rangle
\end{array}$$

/* *A intermediate node receiving a reply message checks whether it should propagate the reply towards the source.**/

$$\begin{array}{l}
\textsc{RReply}(rep, id_s, id_d, Dseq, hc, nh, id_p, id, RT, HT) \stackrel{\text{def}}{=} \\
\qquad [rep = \mathtt{rrep}] \\
\qquad\quad [(Dseq > \mathsf{Dseq}(RT, id_d)) \vee (\mathsf{Dseq}(RT, id_d){=}Dseq \wedge hc < \mathsf{Hcnt}(RT, id_d))] \\
\qquad\qquad \textsc{SndReply}\langle id_s, id_d, Dseq, hc, id_p, id, RT\{id_d \mapsto Dseq, hc, nh, \sigma\}, HT\rangle, \\
\qquad\quad \textsc{Node}\langle id, RT, HT\rangle, \\
\qquad \textsc{Node}\langle id, RT, HT\rangle
\end{array}$$

/* *After receiving a request, a node replies back along the reverse path to the source.* */

$$\begin{array}{l}
\textsc{SndReply}(id_s, id_d, Dseq, hc, id_p, id, RT, HT) \stackrel{\text{def}}{=} \\
\quad \sigma!\langle \mathtt{rrep}, id_s, id_d, Dseq, hc{+}1, id\rangle_{id_p}.\textsc{Node}\langle id, RT, HT\rangle + \textsc{Node}\langle id, RT, HT\rangle
\end{array}$$

state $\textsc{Source}\langle id_s, id_d, Sseq+1, Dseq, Rid+1, RT_s\rangle$. In any case, the source node may nondeterministically return into the state $\textsc{Source}\langle id_s, id_d, Sseq{+}1, Dseq, Rid{+}1, RT_s\rangle$, starting again the protocol (in case the reply message gets lost).

The other nodes in the protocol may be either intermediate nodes or the destination node. In both cases, they begin the protocol in the state $\textsc{Node}\langle id, RT, HT\rangle$, where id is the identity of the node, RT is the routing table of node id, and HT is the history table of the node. In this state, the node expects to receive either a route request message or a route reply message.

When a node receives a route request message, it evolves into the state

$$\textsc{RRequest}\langle req, id_s, id_d, Sseq, Dseq, Rid, hc, id_p, id, RT, HT\rangle \ .$$

In this state, it first checks its history table to verify whether the request is fresh. Then, it checks whether it is the destination of the request or a potential recipient of the request, because it knows a "better" route to the destination. In doing so, it also profits to check the source sequence number of the request to possibly update its routing table entry pointing to the source node. Then, the node moves into a state $\textsc{SndReply}$ to send a route reply packet. If the node is not the destination and it does not know a "better" path to reach the destination, then it re-broadcasts the request message, after adding 1 to the hc field. Then, the node moves into the state $\textsc{Node}\langle id, RT, HT\rangle$.

When a node in state $\textsc{Node}\langle id, RT, HT\rangle$ receives a route reply message it evolves into the state $\textsc{RReply}\langle rep, id_s, id_d, Dseq, hc, nh,\ id_p, id, RT, HT\rangle$. Here, it carries out checks similar to those appearing in state $\textsc{AwaitReply}$.

In state $\textsc{SndReply}\langle id_s, id_d, Dseq, hc, id_p, id, RT, HT\rangle$, the node sends the unicast reply message $\langle \mathtt{rrep}, id_s, id_d, Dseq, hc{+}1, id\rangle$ to the previous node in the route path, that is id_p. The choice operator allows to avoid deadlocks in case the previous node becomes suddenly disconnected.

Here, we report an example to explain how the protocol works.

Example 10.2 *Let M be the following network:*

$$\begin{array}{rcl} M & \stackrel{\text{def}}{=} & l[\textsc{Source}\langle l, n, Sseq, Dseq, Rid, RT_l\rangle]_{T_l} \mid \\ & & m[\textsc{Node}\langle m, RT_m, HT_m\rangle]_{T_m} \mid \\ & & n[\textsc{Node}\langle n, RT_n, HT_n\rangle]_{T_n} \end{array}$$

with $T_l(l,m) = T_l(l,n) = T_m(m,l) = T_m(m,n) = T_n(n,m) \geq \sigma$. For convenience we define:

- $v_1 = \langle \mathtt{rreq}, l, n, Sseq{+}\mathit{1}, Dseq, Rid{+}\mathit{1}, \mathit{0}, l\rangle$,
- $v_2 = \langle \mathtt{rreq}, l, n, Sseq{+}\mathit{1}, Dseq, Rid{+}\mathit{1}, \mathit{1}, m\rangle$,
- $v_3 = \langle \mathtt{rrep}, l, n, Dseq', \mathit{1}, m\rangle$,
- $v_4 = \langle \mathtt{rrep}, l, n, Dseq', \mathit{2}, l\rangle$,

where $Dseq' = \mathsf{Dseq}(RT_n, n)$.

Here, we report the evolution of M while running the AODV protocol.

$$\begin{aligned}
M \xrightarrow{l!v_1 \rhd \{m,n\}}_{\sigma} \;& l[\textsc{AwaitReply}\langle l, n, Sseq, Dseq, Rid, RT_l\rangle]_{T_l} \mid \\
& m[\textsc{RRequest}\langle \texttt{rreq}, l, n, Sseq{+}1, Dseq, Rid{+}1, 1, l, m, RT_m, HT_m\rangle]_{T_m} \mid \\
& n[\textsc{Node}\langle n, RT_n, HT_n\rangle]_{T_n} \\
\stackrel{\text{def}}{=} \;& M_1 \ .
\end{aligned}$$

Node l starts the protocol broadcasting the message v_1 and evolving into the state $\textsc{AwaitReply}\langle l, n, Sseq, Dseq, Rid, RT_l\rangle$*, waiting for a reply. Only m receives the message evolving into* $\textsc{RRequest}\langle \texttt{rreq}, l, n, Sseq{+}1, Dseq, Rid{+}1, 1, l, m, RT_m, HT_m\rangle$*.*

$$\begin{aligned}
M_1 \xrightarrow{m!v_2 \rhd \{l,n\}}_{\sigma} \;& l[\textsc{AwaitReply}\langle l, n, Sseq, Dseq, Rid, RT_l\rangle]_{T_l} \mid \\
& m[\textsc{Node}\langle m, RT_m, HT_m\{l \mapsto Rid{+}1\}\rangle]_{T_m} \mid \\
& n[\textsc{SndReply}\langle l, n, \mathsf{Dseq}(RT_n, n), \mathit{0}, m, n, \\
& \quad RT\{l \mapsto \mathsf{Dseq}(RT_n, n), \mathit{1}, m, \sigma\}, HT\{l \mapsto Rid{+}\mathit{1}, \sigma\}\rangle]_{T_n} \\
\stackrel{\text{def}}{=} \;& M_2 \ .
\end{aligned}$$

We suppose that node m does not have a route entry for the destination n and its source sequence number is up-to-date. Thus, m broadcasts the message v_2 and evolves into the state $\textsc{Node}\langle m, RT_m, HT_m\{l \mapsto Rid{+}1, \sigma\}\rangle$*, waiting for a reply. Node l ignores the message sent by m and remains in the state* $\textsc{AwaitReply}\langle l, n, Sseq,\ Dseq, Rid, RT_l\rangle$*. Node n receives the request, it verifies to be the destination node and that its entry pointing to the source node is not up-to-date. In this case, it evolves into the state* $\textsc{SndReply}\langle l, n, Dseq', 0, m, n,\ RT\{l \mapsto Dseq', \mathit{1}, m, \sigma\}, HT\{l \mapsto Rid{+}\mathit{1}, \sigma\}\rangle$*.*

$$\begin{aligned}
M_2 \xrightarrow{n!v_3 \rhd m}_{\sigma} \;& l[\textsc{AwaitReply}\langle l, n, Sseq, Dseq, Rid, RT_l\rangle]_{T_l} \mid \\
& m[\textsc{SndReply}\langle l, n, Dseq', 0, m, n, RT\{l \mapsto Dseq', 1, m, \sigma\}, \\
& \qquad\qquad HT\{l \mapsto Rid{+}\mathit{1}, \sigma\}\rangle]_{T_m} \mid \\
& n[\textsc{Node}\langle n, RT_n, HT_n\rangle]_{T_n} \\
\stackrel{\text{def}}{=} \;& M_3 \ .
\end{aligned}$$

Node n sends the reply message v_3 back to m. We assume that node m correctly receives this message and it evolves into the state

$\textsc{SndReply}\langle l, n, \mathsf{Dseq}(RT_m, n), 0, m, n, RT\{l \mapsto \mathsf{Dseq}(RT_m, n), 1, m, \sigma\}, HT\{l \mapsto Rid{+}\mathit{1}, \sigma\}\rangle$*.*

$$\begin{aligned} M_3 \xrightarrow{m!v_4 \triangleright l}_{\sigma} \; & l[\textsc{RouteSuccess}\langle l, n, RT_l\{n \mapsto Dseq', 2, m, \sigma\}\rangle]_{T_l} \mid \\ & m[\textsc{Node}\langle m, RT_m, HT_m\{l \mapsto Rid{+}1, \sigma\}\rangle]_{T_m} \mid \\ & n[\textsc{Node}\langle n, RT_n, HT_n\rangle]_{T_n} \; . \end{aligned}$$

Finally, node m sends the reply message v_4 back to l. Node l receives the message and verifies that the node-id of the source is the expected ones. Then it accepts the route and evolves into the state $\textsc{RouteSuccess}\langle l, n, RT_l\{n \mapsto Dseq', 2, m, \sigma\}\rangle$.

The trust-based nature of our variant of the AODV protocol can be summarised in the following statement.

Proposition 10.3 *If a path p is established in a network M by applying the AODV protocol at some security level σ, then the nodes of p constitute a σ-connected component of M.*

Proof It follows by Theorem 6.1. □

10.2 A trust-based leader election protocol

In [60] the authors propose a leader election protocol to elect as a leader a node of a connected component of a MANET. The algorithm operates by first "growing" and then "shrinking" a spanning tree rooted at the node that initiates the election algorithm. We refer to this computation-initiating node as the source node. As we will see, after the spanning tree shrinks completely, the source node will have adequate information to determine the most-valued-node and will then broadcast its identity to the rest of the nodes in the network. The algorithm uses three kinds of messages, viz. *Election*, *Ack* and *Leader*. Election messages are used to grow the spanning tree. When election is triggered at a source node s (for instance, upon departure of its current leader), the node broadcasts an *election message*. Each node, i, other than the source s, designates the neighbour from which it first receives an election message as its parent in the spanning tree. Then, each node i broadcasts the received election message. After sending an election message, a node awaits ack messages from its children in the spanning tree, before sending an ack message to its parent. The ack messages sent to the parents contains leader-election information based on the ack messages received from children.

Once the spanning tree has completely grown, it shrinks back toward the source. Specifically, once all of i's outgoing election messages have been acknowledged, i sends its pending ack message to its parent node. Tree shrinkage begins at the leaves of the spanning tree, which are parents to no other node. Eventually, each leaf receives ack messages for all election messages it has sent. As a consequence, leaves send their pending ack messages to their respective parents, who in turn send their pending ack messages to their own parents, and so on, until the source node receives all of its pending ack messages. In a ack message, a node announces to its parent the node-id and the value of the most-valued-node among all its downstream nodes. Hence, the source node eventually has sufficient information to determine the most-valued-node from among all nodes in the network. Once the source node for a computation has received ack messages from all of its children, it then broadcasts a *leader message* to all nodes announcing the node-id of the most-valued-node.

In Figure 3 we provide our trust-based encoding of the leader election protocol for MANETs [60]. For simplicity, in sub-terms of the form $\sigma?(\tilde{x}).P$ we write x_i in P to mean the i-th component of $\tilde{x}$, if this component is defined, and $\perp$ (undefined) otherwise. In the algorithm, nodes periodically use *probe* and *reply* messages to keep track of their neighbours. We do not consider probe and reply messages in our encoding as we can model the effect of disconnection between nodes using the choice operator. Let us explain more in detail the encoding of Figure 3. At the beginning, each node can be in one of these two states:

- $\textsc{Source}(\mathit{id}, \mathit{elec}, \mathit{lid})$, if the node initiates the protocol;
- $\textsc{Node}(\mathit{id}, \mathit{elec}, \mathit{lid})$, otherwise.

While the protocol is executed, nodes may evolve into one of the following states:

- $\textsc{AwaitAckInit}(\mathit{id}, \mathit{elec}, \mathit{lid})$, a starting node waits for ack messages;
- $\textsc{SendLeader}(\mathit{id}, \mathit{elec}, \mathit{lid})$, a node broadcasts a leader message;
- $\textsc{ElectionProcess}(\mathit{id}, \mathit{elec}, \mathit{lid}, \mathit{id}_p)$, a node rebroadcasts the election message previously received by its parent;
- $\textsc{AwaitAck}(\mathit{id}, \mathit{elec}, \mathit{lid}, \mathit{id}_p)$, a node waits for ack messages;
- $\textsc{SendAck}(\mathit{id}, \mathit{elec}, \mathit{lid}, \mathit{id}_p)$, a node sends an ack message to its parent;
- $\textsc{LeaderProcess}(\mathit{id}, \mathit{elec}, \mathit{lid}, \mathit{maxid})$, a node sets its leader parameter to the value received.

The meaning of the parameters of the above states is the following: *id* is the name of the node; *elec* indicates whether the node is part of the election process, thus, $\mathit{elec} = 1$ if the node is participating in the election process, $\mathit{elec} = 0$ otherwise; *lid* represents the node's knowledge of the leader; id_p is the name of the parent node; *maxid* is maximum node-id in the spanning tree rooted at *id*.

A node may send and/or receive election, ack or leader messages. These messages are pairs of the following shape:

- `elecMsg`, *id* meaning an election message sent by node *id*;
- `ackMsg`, *lid* meaning an ack message where *lid* is the current leader at that stage;
- `ldrMsg`, *lid* meaning a leader message where *lid* is the current node's knowledge of the leader.

Figure 3 An trust-based encoding of the leader election protocol for MANETs at security level σ.

/* *Starting node broadcasts election message and evolves into the* AwaitAckInit *state waiting for ack.* */

$$\textsc{Source}(id, elec, lid) \stackrel{\text{def}}{=} \sigma!\langle \texttt{elecMsg}, id\rangle.\textsc{AwaitAckInit}\langle id, \mathit{1}, lid\rangle$$

/* *In the* AwaitAckInit *state the initiator node receives ack messages (other messages are ignored) and store the maximum node-id received; eventually the process evolves into the* SendLeader *state.* */

$$\begin{array}{lll}\textsc{AwaitAckInit}(id, elec, lid) & \stackrel{\text{def}}{=} & \sigma?(\tilde{x}).[x_1 = \texttt{ackMsg}] \\ & & \qquad\qquad [x_2 \geq lid]\textsc{AwaitAckInit}\langle id, elec, \mathbf{snd}(\tilde{x})\rangle, \\ & & \qquad\qquad \textsc{AwaitAckInit}\langle id, elec, lid\rangle, \\ & & \qquad\quad \textsc{AwaitAckInit}\langle id, elec, lid\rangle \\ & + & \textsc{SendLeader}\langle id, elec, lid\rangle\end{array}$$

/* *In the* SendLeader *state the node broadcasts a leader message.* */

$$\textsc{SendLeader}(id, elec, lid) \stackrel{\text{def}}{=} \sigma!\langle \texttt{ldrMsg}, lid\rangle.\textsc{Node}\langle id, \mathit{0}, lid\rangle$$

/* *A node which did not initiate the protocol may receive either an election or a leader message, evolving into an* ElectionProcess *or a* LeaderProcess *state, respectively.* */

$$\begin{array}{ll}\textsc{Node}(id, elec, lid) \stackrel{\text{def}}{=} & \sigma?(\tilde{x}).[x_1 = \texttt{ldrMsg}] \\ & \qquad \textsc{LeaderProcess}\langle id, elec, lid, \mathbf{snd}(\tilde{x})\rangle, \\ & [x_1 = \texttt{elecMsg}]\textsc{ElectionProcess}\langle id, \mathit{1}, lid, x_2\rangle, \textsc{Node}\langle id, elec, lid\rangle\end{array}$$

/* *A node in the* LeaderProcess *state basically propagates leader messages containing the maximum between its lid and the maxid received in the leader messages. The most interesting case in when maxid < lid. This means that either the node was not part of the election process or the node did not report the ack to its parent nodes, for example because it was disconnected. In both cases, it broadcasts its lid as the maximum node-id.* */

$$\begin{array}{ll}\textsc{LeaderProcess}(id, elec, lid, maxid) \stackrel{\text{def}}{=} & [maxid = lid] \\ & \quad [elec = 0]\textsc{Node}(id, \mathit{0}, lid), \sigma!\langle \texttt{ldrMsg}, lid\rangle.\textsc{Node}\langle id, \mathit{0}, lid\rangle, \\ & \quad [maxid > lid]\sigma!\langle \texttt{ldrMsg}, maxid\rangle.\textsc{Node}\langle id, \mathit{0}, maxid\rangle, \\ & \quad [maxid < lid]\sigma!\langle \texttt{ldrMsg}, lid\rangle.\textsc{Node}\langle id, \mathit{0}, lid\rangle\end{array}$$

/* *In the* ElectionProcess *state a node broadcasts the election message received by its parent and evolves into an* AwaitAck *state, waiting for ack messages.* */

$$\textsc{ElectionProcess}(id, elec, lid, id_p) \stackrel{\text{def}}{=} \sigma!\langle \texttt{elecMsg}, id\rangle.\textsc{AwaitAck}\langle id, elec, lid, id_p\rangle$$

/* *In the* AwaitAck *state the node receives ack messages and update the maximum node-id as in the* AwaitAckInit *state; the only difference is that when no more ack arrives, the process evolves into the* SendAck *state.* */

$$\begin{array}{lll}\textsc{AwaitAck}(id, elec, lid, id_p) & \stackrel{\text{def}}{=} & \sigma?(\tilde{x}).[x_1 = \texttt{ackMsg}] \\ & & \qquad\qquad [x_2 \geq lid]\textsc{AwaitAck}\langle id, elec, x_2, id_p\rangle, \\ & & \qquad\qquad \textsc{AwaitAck}\langle id, elec, lid, id_p\rangle, \\ & & \qquad\quad \textsc{AwaitAck}\langle id, elec, lid\rangle \\ & + & \textsc{SendAck}\langle id, elec, lid, id_p\rangle\end{array}$$

/* *In a* SendAck *state a node may either send (unicast transmission) to its parent node an ack message, with the current maximum node-id, or evolve into a* SendLeader *state if the node disconnects from its parent node. In this case, it reports its current leader.* */

$$\textsc{SendAck}(id, elec, lid, id_p) \stackrel{\text{def}}{=} \sigma!\langle \texttt{ackMsg}, lid\rangle_{id_p}.\textsc{Node}\langle id, elec, lid\rangle + \textsc{SendLeader}\langle id, elec, lid\rangle$$

A starting node begins the protocol in the $\textsc{Source}\langle id, 0, lid\rangle$ state, with $lid = id$, and broadcasts the message $\langle \texttt{elecMsg}, lid\rangle$ moving into the $\textsc{AwaitAckInit}\langle id, 1, lid\rangle$ state, waiting for the ack message. In this state the starting node may receive ack messages of the form $\langle \texttt{ackMsg}, maxid\rangle$. When this happens, the node checks the *maxid* variable contained in the ack message. If $maxid \geq lid$ then the node evolves into the $\textsc{AwaitAckInit}\langle id, elec, maxid\rangle$ state to record the *maxid* value, otherwise it remains in $\textsc{AwaitAckInit}\langle id, elec, lid\rangle$, waiting for other ack messages. In the state $\textsc{AwaitAckInit}\langle id, elec, lid\rangle$ the node may also nondeterministically evolves into the $\textsc{SendLeader}\langle id, elec, lid\rangle$ when it has received all ack messages from its neighbours. In the $\textsc{SendLeader}\langle id, elec, lid\rangle$ state a node broadcasts a leader message $\langle \texttt{ldrMsg}, lid\rangle$ and evolves into the $\textsc{Node}\langle id, 0, lid\rangle$ state, waiting for other leader messages sent by its neighbours.

All the other nodes in the network begin the protocol in the $\textsc{Node}\langle id, 0, lid\rangle$ state. In this state, they may receive an election message $\langle \texttt{elecMsg}, id_p\rangle$ or a leader message $\langle \texttt{leaderMsg}, maxid\rangle$. In the first case, they evolve into $\textsc{ElectionProcess}\langle id, 1, lid, id_p\rangle$, where id_p records the id of their parent node, contained in the election message. In the second case, they evolve into the $\textsc{LeaderProcess}\langle id, elec, lid, maxid\rangle$ state, where *maxid* is the leader id contained in the leader message. When a node arrives in the $\textsc{ElectionProcess}\langle id, elec, lid, id_p\rangle$ state, it broadcasts an election message containing its node-id and then evolves into the state $\textsc{AwaitAck}\langle id, elec, lid, id_p\rangle$, waiting for ack messages. A node in the $\textsc{AwaitAck}\langle id, elec, id, id_p\rangle$ state may either receive ack messages of the form $\langle \texttt{ackMsg}, maxid\rangle$ or evolve into the $\textsc{SendAck}\langle id, elec, lid, id_p\rangle$ state to model that all ack messages have been received. When the node receives an ack, it stores the maximum node-id received with the ack, checking if $maxid \geq lid$. A node in the $\textsc{SendAck}\langle id, elec, lid, id_p\rangle$ state may send to its parent node id_p an ack message of the form $\langle \texttt{ackMsg}, lid\rangle$ with the current maximum node-id, and goes into the $\textsc{Node}\langle id, elec, lid\rangle$ state; otherwise the node may evolve into the $\textsc{SendLeader}\langle id, elec, lid\rangle$ state. This last state models the case when a node is disconnected from its parent node. In this case, the node reports its current leader.

A node in the $\textsc{LeaderProcess}\langle id, elec, lid, maxid\rangle$ basically propagates the received leader message by setting its *lid* parameter to the *maxid* values received in the leader message. The most interesting case is when $maxid < lid$. In this case, either the node was not part of the election process or it did not report the ack message to its parent nodes, for example because it was disconnected. In both case it broadcasts its *lid* as the maximum node-id sending a leader message $\langle \texttt{ldrMsg}, lid\rangle$ and evolves into $\textsc{Node}\langle id, 0, lid\rangle$.

Here, we report a running example of the protocol.

Example 10.4 *Let M be the following network:*

$$M \stackrel{\text{def}}{=} l[\textsc{Source}\langle l, 0, l\rangle]_{T_l} \mid m[\textsc{Node}\langle m, 0, m\rangle]_{T_m} \mid n[\textsc{Node}\langle n, 0, n\rangle]_{T_n}$$

with $l > m > n$ and $T_l(l,m){=}T_m(m,l){=}T_m(m,n){=}T_n(n,m){\geq}\sigma$. Here, we report the *evolution M while running the protocol.*

$$\begin{aligned} M \quad & \xrightarrow{l!\langle \mathtt{elecMsg}, l\rangle \triangleright m}_{\sigma} \quad l[\textsc{AwaitAckInit}\langle l,1,l\rangle]_{T_l} \mid \\ & \qquad\qquad m[\textsc{ElectionProcess}\langle m,1,m,l\rangle]_{T_m} \mid \\ & \qquad\qquad n[\textsc{Node}\langle n,0,n\rangle]_{T_n} \\ & \stackrel{\text{def}}{=} \qquad M_1 \ . \end{aligned}$$

Node l starts the protocol broadcasting the election message $\langle \mathtt{elecMsg}, l\rangle$, and evolving into the state $\textsc{AwaitAckInit}\langle l,1,l\rangle$*. Only node m receives the election message and evolves into the state* $\textsc{ElectionProcess}\langle m,1,m,l\rangle$*. Node m has marked l as its parent node.*

$$\begin{aligned} M_1 \quad & \xrightarrow{m!\langle \mathtt{elecMsg}, m\rangle \triangleright \{l,n\}}_{\sigma} \quad l[\textsc{AwaitAckInit}\langle l,1,l\rangle]_{T_l} \mid m[\textsc{AwaitAck}\langle m,1,m,l\rangle]_{T_m} \mid \\ & \qquad\qquad n[\textsc{ElectionProcess}\langle n,1,n,m\rangle]_{T_n} \\ & \stackrel{\text{def}}{=} \qquad M_2 \ . \end{aligned}$$

Node m broadcasts the message $\langle \mathtt{elecMsg}, m\rangle$, and evolves into $\textsc{AwaitAck}\langle m,1,m,l\rangle$*, waiting for ack messages. Node l ignores the message and remains in* $\textsc{AwaitAckInit}\langle l,1,l\rangle$*, whereas n receives the message and evolves into* $\textsc{ElectionProcess}\langle n,1,n,m\rangle$*. Again the last parameter m indicates the parent node of n.*

$$\begin{aligned} M_2 \quad & \xrightarrow{n!\langle \mathtt{elecMsg}, n\rangle \triangleright m}_{\sigma} \quad l[\textsc{AwaitAckInit}\langle l,1,l\rangle]_{T_l} \mid m[\textsc{AwaitAck}\langle m,1,m,l\rangle]_{T_m} \mid \\ & \qquad\qquad n[\textsc{AwaitAck}\langle n,1,n,m\rangle]_{T_n} \\ & \stackrel{\text{def}}{=} \qquad M_3 \ . \end{aligned}$$

Node n broadcasts the election message $\langle \mathtt{elecMsg}, n\rangle$ and then it evolves into the state $\textsc{AwaitAck}\langle n,1,n,m\rangle$*, waiting for ack messages. Node m ignores the message and remains in its state.*

$$\begin{aligned} M_3 \quad & \xrightarrow{n!\langle \mathtt{ackMsg}, n\rangle \triangleright m}_{\sigma} \quad l[\textsc{AwaitAckInit}\langle l,1,l\rangle]_{T_l} \mid m[\textsc{AwaitAck}\langle m,1,m,l\rangle]_{T_m} \mid \\ & \qquad\qquad n[\textsc{Node}\langle n,1,n\rangle]_{T_n} \\ & \stackrel{\text{def}}{=} \qquad M_4 \ . \end{aligned}$$

As n has no children, it will not receive ack messages. Thus, it will eventually evolve into the state $\textsc{SendAck}\langle n,1,n,m\rangle$ *sending the ack message $\langle \mathtt{ackMsg}, n\rangle$ to its parent node m. This message contains the current leader of node n, that is n itself. After the sending, n evolves into the state* $\textsc{Node}\langle n,1,n\rangle$*, waiting for leader messages.*

$$\begin{aligned} M_4 \quad & \xrightarrow{m!\langle \mathtt{ackMsg}, m\rangle \triangleright l}_{\sigma} \quad l[\textsc{AwaitAckInit}\langle l,1,l\rangle]_{T_l} \mid m[\textsc{Node}\langle m,1,m\rangle]_{T_m} \mid \\ & \qquad\qquad n[\textsc{Node}\langle n,1,n\rangle]_{T_n} \\ & \stackrel{\text{def}}{=} \qquad M_5 \ . \end{aligned}$$

When m receives the message $\langle \mathtt{ackMsg}, n\rangle$, it checks whether n is greater than its current leader m. As $m > n$, m sends the ack message $\langle \mathtt{ackMsg}, m\rangle$ to its parent l and evolves into the state $\textsc{Node}\langle m,1,m\rangle$. *The message $\langle \mathtt{ackMsg}, m\rangle$ contains the current leader of m, that is m itself.*

$$\begin{aligned} M_5 \quad & \xrightarrow{l!\langle \mathtt{ldrMsg}, l\rangle \triangleright m}_{\sigma} \quad l[\textsc{Node}\langle l,0,l\rangle]_{T_l} \mid m[\textsc{LeaderProcess}\langle m,1,m,l\rangle]_{T_m} \mid \\ & \qquad\qquad n[\textsc{Node}\langle n,1,n\rangle]_{T_n} \\ & \stackrel{\text{def}}{=} \qquad M_6 \ . \end{aligned}$$

When l receives the message $\langle \mathtt{ackMsg}, n\rangle$, it checks whether m is greater than its current leader l. As $l > m$, node l broadcasts the leader message $\langle \mathtt{ldrMsg}, l\rangle$ and evolves into the state $\textsc{Node}\langle l,0,l\rangle$. *Node m receives the leader message and evolves into the state* $\textsc{LeaderProcess}\langle m,1,m,l\rangle$.

$$\begin{aligned} M_6 \quad & \xrightarrow{m!\langle \mathtt{ldrMsg}, l\rangle \triangleright \{l,n\}}_{\sigma} \quad l[\textsc{Node}\langle l,0,l\rangle]_{T_l} \mid m[\textsc{Node}\langle m,0,l\rangle]_{T_m} \mid \\ & \qquad\qquad n[\textsc{LeaderProcess}\langle n,1,n,l\rangle]_{T_n} \\ & \stackrel{\text{def}}{=} \qquad M_7 \ . \end{aligned}$$

Node m checks whether l is greater than its current leader m. As $l > m$, then m broadcasts the leader message $\langle \mathtt{ldrMsg}, l\rangle$ with l as its current leader and evolves into the state $\textsc{Node}\langle m,0,l\rangle$. *When l receives this message, as it is in* $\textsc{Node}\langle l,0,l\rangle$ *state, it has simply to check whether the received leader corresponds to its current leader. This is the case, then it remains into* $\textsc{Node}\langle l,0,l\rangle$ *state. When n receives the leader message $\langle \mathtt{ldrMsg}, l\rangle$, it evolves into the state* $\textsc{LeaderProcess}\langle n,1,n,l\rangle$.

$$M_7 \quad \xrightarrow{n!\langle \mathtt{ldrMsg}, l\rangle \triangleright m}_{\sigma} \quad l[\textsc{Node}\langle l,0,l\rangle]_{T_l} \mid m[\textsc{Node}\langle m,0,l\rangle]_{T_m} \mid n[\textsc{Node}\langle n,0,l\rangle]_{T_n} \ .$$

Finally, node n verifies that l is greater than its current leader n. Thus, l becomes the current leader of n that broadcasts the leader message $\langle \mathtt{ldrMsg}, l\rangle$ with current leader l, evolving into the state $\textsc{Node}\langle n,0,l\rangle$. *The leader message sent by n is received by m that verifies that l is already its leader. So, it remains in the state* $\textsc{Node}\langle m,0,l\rangle$.

The trust-based nature of our variant of the leader election protocol can be summarised in the following statement.

Proposition 10.5 *If a node n is elected in a network M by applying the leader election protocol at some security level σ, then the nodes which have partecipated to its election constitute a σ-connected component of M.*

Proof It follows by Theorem 6.1. □

11 Conclusions and related work

We have proposed a process calculus for mobile ad hoc networks which relies on an abstract behaviour-based multilevel trust model. Our trust model supports both direct trust, to describe monitoring of neighbour nodes, and indirect trust, when collecting recommendations and spreading reputations. The operational semantics of the calculus is given in terms of a labelled transition system, where actions are executed at a certain security level. As to the behavioural semantics we focus on a trust-based variant of Milner and Sangiorgi's barbed congruence, a standard contextually-defined program equivalence. We then define a labelled bisimilarity over networks parameterised over security levels. We have proved that communications in our setting are safe, in a precise sense, with respect to the security levels of the involved parties. In particular, we guarantee safety despite compromised nodes, meaning that compromised nodes cannot affect the rest of the network. A *non-interference* result is also proved in terms of information flow. Finally, we have demonstrated the practical utility of our calculus by providing a formal description of *trust-based* versions of a *routing protocol* and a *leader election protocol* for ad hoc networks.

The problem of protecting information and resources in multilevel systems [5] has been extensively studied using different approaches. For instance, Bodei et al. [8] have applied flow analysis techniques, Reitman and Andrews [51] have used axiomatic logic, while Smith and Volpano in [61], Boudol and Castellani [9] and Heintz and Riecke in [28] have focused on type systems for prototypical programming languages.

Excellent surveys about information flow properties can be found in [18, 53].

In the context of multilevel systems, Crafa and Rossi [14] have introduced a notion of *controlled information release* for a typed version of the π-calculus extended with *declassified actions*. They have provided various characterisations of *controlled release property*, based on typed behavioural equivalence, parameterised on security levels, to model observers at a certain security level. Our notion of bisimilarity, parameterised on security levels, is inspired by theirs. Hennessy has proposed a typed version of the asynchronous π-calculus with I-O types associated with security levels. Typed versions of *may* and *must* equivalences are then used to prove a non-interference results.

Let us examine now the most relevant related work on process calculi for wireless systems. Nanz and Hankin [44] have introduced a calculus for Mobile Wireless Networks (CBS$^\sharp$), relying on graph representation of node localities. The main goal of the paper is to present a framework for specification and security analysis of communication protocols for mobile wireless networks. Merro [38] has proposed a process calculus for Mobile Ad Hoc Networks with a labelled characterisation of reduction barbed congruence. Godskesen [24] has proposed a calculus for mobile ad hoc networks (CMAN). The paper proves a characterisation of reduction barbed congruence in terms of a contextual bisimulation. It also contains a formalisation of an attack on the cryptographic routing protocol ARAN. Singh, Ramakrishnan, and Smolka [57] have proposed the ω-calculus, a conservative extension of the π-calculus. A key feature of the ω-calculus is the separation of a node's communication and computational behaviour from the description of its physical transmission range. The authors provide a labelled transition semantics and a bisimulation in "open" style. The ω-calculus is then used for modelling both the AODV routing protocol and the leader election protocol of [60], in an untrusted setting. Ghassemi et al. [22] have proposed a process algebra for mobile ad hoc networks (RBPT) where, topology changes are implicitly modelled in the (operational) semantics rather than in the syntax. The authors propose a notion of bisimulation for networks parameterised on a set of topology invariants that must be respected by equivalent networks. This work in then refined in [23] where the authors propose an equational theory for an extension of RBPT. Godskesen and Nanz [25] have proposed a simple timed calculus for wireless systems to express a wide range of mobility models. All the previous calculi abstract from the presence of interferences. Lanese and Sangiorgi [36] have instead proposed the CWS calculus, a lower level untimed calculus to describe interferences in wireless systems. More recently, Song and Godskesen [58], have proposed a probabilistic calculus for wireless systems modelling unreliable connection. They have characterised a notion of weak bisimilarity by a variant of PCTL.

None of the calculi mentioned above deal with trust. Carbone et al. [12] have introduced *ctm*, a process calculus which embodies the notion of trust for ubiquitous systems. In *ctm* each principal is equipped with a *policy*, which determines its legal behaviour, formalised using a Datalog-like logic, and with a *protocol*, in the process algebra style, which allows interactions between principals and the flow of information from principals to policies.

References

[1] Alfarez Abdul-Rahman and Stephen Hailes. Supporting Trust in Virtual Communities. In *HICSS*, volume 6, pages 6007–6016. IEEE Computer Society, 2000.

[2] Gergely Ács, Levente Buttyán, and István Vajda. Provably Secure On-Demand Source Routing in Mobile Ad Hoc Networks. *IEEE Transaction on Mobile Computing*, 5(11):1533–1546, 2006.

[3] Efthimia Aivaloglou, Stefanos Gritzalis, and Charalabos Skianis. Trust Establishment in Sensor Networks: Behaviour-based, Certificate-based and a Combinational Approach. *International Journal of System of Systems Engineering*, 1(1–2):128–148, 2008.

[4] Venkat Balakrishnan, Vijay Varadharajan, and Uday Tupakula. *Trust Management in Mobile Ad Hoc Networks*, pages 473–500. Computer Communications and Networks. Springer, 2009.

[5] David Elliott Bell and Leonard J. LaPadula. Secure Computer System: Unified Exposition and Multics Interpretation. Technical Report MTR-2997, MITRE Corporation, 1976.

[6] Karthikeyan Bhargavan, Davor Obradovic, and Carl A. Gunter. Formal Verification of Standards for Distance Vector Routing Protocols. *Journal of the ACM*, 49(4):538–576, 2002.

[7] Matt Blaze, Joan Feigenbaum, and Jack Lacy. Decentralized Trust Management. In *Symposium on Security and Privacy*, pages 164–173. IEEE Computer Society, 1996.

[8] Chiara Bodei, Pierpaolo Degano, Flemming Nielson, and Hanne Riis Nielson. Static Analysis for the pi-Calculus with Applications to Security. *Information and Computation*, 168(1):68–92, 2001.

[9] Gérard Boudol and Ilaria Castellani. Noninterference for Concurrent Programs and Thread Systems. *Theoretical Computer Science*, 281(1-2):109–130, 2002.

[10] S. Buchegger and J. Le Boudec. Performance analysis of the confidant protocol. In *MobiHoc*, pages 226–236, 2002.

[11] L. Buttyán and J.P. Hubaux. Enforcing service availability in mobile ad-hoc wans. In *MobiHoc*, pages 87–96, 2000.

[12] Marco Carbone, Mogens Nielsen, and Vladimiro Sassone. A Calculus for Trust Management. In *FSTTCS*, volume 3328 of *Lecture Notes in Computer Science*, pages 161–173. Springer, 2004.

[13] Tim Clausen and Philipp Jacquet. Optimized Link State Routing Protocol (OLSR), 2003. RFC 3626.

[14] Silvia Crafa and Sabina Rossi. Controlling Information Release in the π-calculus. *Information and Computation*, 205(8):1235–1273, 2007.

[15] Robero Di Pietro, Luigi V. Mancini, Yee W. Law, Sandro Etalle, and Paul J. M. Havinga. LKHW: A Directed Diffusion-Based Secure Multicast Scheme for Wireless Sensor Networks. In *ICPP Workshops*, pages 397–413. IEEE Computer Society, 2003.

[16] Riccardo Focardi and Roberto Gorrieri. A Classification of Security Properties for Process Algebras. *Journal of Computer Security*, 3(1):5–33, 1995.

[17] Riccardo Focardi and Roberto Gorrieri. The Compositional Security Checker: A Tool for the Verification of Information Flow Security Properties. *IEEE Transactions on Software Engineering*, 27(3):550–571, 1997.

[18] Riccardo Focardi and Roberto Gorrieri. Classification of Security Properties (Part I: Information Flow). In *FOSAD*, volume 2171 of *Lecture Notes in Computer Science*, pages 331–396. Springer, 2001.

[19] Cédric Fournet, Andrew D. Gordon, and Sergio Maffeis. A Type Discipline for Authorization in Distributed Systems. In *CSF*, pages 31–48. IEEE computer Society, 2007.

[20] Diego Gambetta. *Trust: Making and Breaking Cooperative Relations.* Blackwell Publishers, 1988.

[21] S. Garfinkel. *PGP: Pretty Good Privacy.* O'Reilly and Assoc., 1995.

[22] Fatemeh Ghassemi, Wan Fokkink, and Ali Movaghar. Restricted Broadcast Process Theory. In *SEFM*, pages 345–354. IEEE Computer Society, 2008.

[23] Fatemeh Ghassemi, Wan Fokkink, and Ali Movaghar. Equational Reasoning on Ad Hoc Networks. In *FSEN*, volume 5961 of *Lecture Notes in Computer Science*, pages 113–128. Springer, 2009.

[24] Jens Chr. Godskesen. A Calculus for Mobile Ad Hoc Networks. In *COORDINATION*, volume 4467 of *Lecture Notes in Computer Science*, pages 132–150. Springer, 2007.

[25] Jens Chr. Godskesen and Sebastian Nanz. Mobility Models and Behavioural Equivalence for Wireless Networks. In *COORDINATION*, volume 5521 of *Lecture Notes in Computer Science*, pages 106–122. Springer, 2009.

[26] Joseph A. Goguen and José Meseguer. Security Policies and Security Models. In *Security and Privacy*, pages 11–20. IEEE Computer Society, 1982.

[27] Tyrone W. A. Grandison. *Trust Management for Internet Applications.* PhD thesis, Department of Computing, University of London, 2003.

[28] Nevin Heintze and John G. Riecke. The SLam Calculus: Programming with Secrecy and Integrity. In *POPL*, pages 365–377. ACM Press, 1998.

[29] Yih-Chun Hu, Adrian Perrig, and David B. Johnson. Ariadne: A Secure On-Demand Routing Protocol for Ad Hoc Networks. *Wireless Networks*, 11(1-2):21–38, 2005.

[30] Dijiang Huang and Deep Medhi. A Secure Group Key Management Scheme for Hierarchical Mobile Ad Hoc Networks. *Ad Hoc Networks*, 6(4):560–577, 2008.

[31] Chalermek Intanagonwiwat, Ramesh Govindan, Deborah Estrin, John Heidemann, and Fabio Silva. Directed Diffusion for Wireless Sensor Networking. *IEEE/ACM Transactions on Networking*, 11(1):2–16, 2003.

[32] David B. Johnson and David A. Maltz. *Dynamic Source Routing in Ad Hoc Wireless Networks.* Kluwer Academic Publishers, 1996.

[33] Audun Jøsang, Elizabeth Gray, and Michael Kinateder. Simplification and Analysis of Transitive Trust Networks. *Web Intelligence and Agent Systems*, 4(2):139–161, 2006.

[34] Audun Jøsang, Roslan Ismail, and Colin Boyd. A Survey of Trust and Reputation Systems for Online Service Provision. *Decision Support Systems*, 43(2):618–644, 2006.

[35] Maryna Komarova and Miguel Riguidel. Adjustable Trust Model for Access Control. In *ATC*, volume 5060 of *Lecture Notes in Computer Science*, pages 429–443. Springer, 2008.

[36] Ivan Lanese and Davide Sangiorgi. An operational semantics for a calculus for wireless systems. *Theoretical Computer Science*, 411(19):1928–1948, 2010.

[37] S. Marti, T. Giuli, K. Lai, and M Baker. Mitigating routing misbehavior in mobile ad hoc networks. In *MOBICOM*, pages 255–265, 2000.

[38] Massimo Merro. An Observational Theory for Mobile Ad Hoc Networks (full paper). *Information and Computation*, 207(2):194–208, 2009.

[39] Nicola Mezzetti and Davide Sangiorgi. Towards a Calculus For Wireless Systems. *Electronic Notes in Theoretical Computer Science*, 158:331–353, 2006.

[40] P. Michiardi and R. Molva. CORE: A collaborative reputation mechanism to enforce node cooperation in mobile ad hoc networks. In *Communications and Multimedia Security*, pages 107–121, 2002.

[41] Robin Milner. *Communication and Concurrency.* Prentice Hall, 1989.

[42] Robin Milner, Joachim Parrow, and David Walker. A Calculus of Mobile Processes, (Parts I and II). *Information and Computation*, 100:1–77, 1992.

[43] Robin Milner and Davide Sangiorgi. Barbed bisimulation. In *ICALP*, volume 623 of *Lecture Notes in Computer Science*, pages 685–695. Springer Verlag, 1992.

[44] Sebastian Nanz and Chris Hankin. A Framework for Security Analysis of Mobile Wireless Networks. *Theoretical Computer Science*, 367(1-2):203–227, 2006.

[45] Panagiotis Papadimitratos and Zygmunt J. Haas. Secure Link State Routing for Mobile Ad Hoc Networks. In *SAINT Workshops*, pages 379–383. ACM, 2003.

[46] V. PArk and S. Corson. Temporally Ordered Routing algorithm (tora) version 1 functional specification. IETF MANET, Internet Draft, 2001.

[47] Charles E. Perkins and Elizabeth M. Belding-Royer. Ad-hoc On-Demand Distance Vector Routing. In *WMCSA*, pages 90–100. IEEE Computer Society, 1999.

[48] Charles E. Perkins and Pravin Bhagwat. Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers. In *SIGCOMM*, pages 234–244, 1994.

[49] A.A. Pirzada, A. Datta, and McDonald C. Trust Boased Routing for Ad Hoc Wireless Networks. In *ICON*. IEEE Computer Society, 2004.

[50] A.A. Pirzada, C. McDonald, and A. Datta. Performance comparison of trust-based reactive routing protocols. *IEEE Transactions on Mobile Computing*, 5(6):695–710, 2006.

[51] Richard P. Reitman and Gregory R. Andrews. An Axiomatic Approach to Information Flow in Programs. *ACM Transactions on Programming Languages and Systems*, 2(1):56–76, 1980.

[52] Rodrigo Roman, M.Carmen Fernandez-Zago, Javier Lopez, and Chen Hsiao-Hwa. *Trust and Reputation Systems for Wireless Sensor Networks*, pages 105–127. Security and Privacy in Mobile and Wireless Networking. Troubador, 2009.

[53] Peter Y. A. Ryan and Stanley A. Schneider. Process Algebra and Non-Interference. In *CSFW*, pages 214–227. IEEE Computer Society, 1999.

[54] Ravi S. Sandhu and Pierangela Samarati. Access Control: Principles and Practice. *IEEE Communications Magazine*, 32:40–48, 1994.

[55] Kimaya Sanzgiri, Daniel LaFlamme, Bridget Dahill, Brian Neil Levine, Clay Shields, and Elizabeth M. Belding-Royer. Authenticated Routing for Ad Hoc Networks. *IEEE Journal on Selected Areas in Communication, special issue on Wireless Ad Hoc Networks*, 23(3):598–610, 2005.

[56] Mohamed Shehab, Elisa Bertino, and Arif Ghafoor. Efficient Hierarchical Key Generation and Key Diffusion for Sensor Networks. In *SECON*, pages 76–84. IEEE Communications Society, 2005.

[57] Anu Singh, C. R. Ramakrishnan, and Scott A. Smolka. A Process Calculus for Mobile Ad Hoc Networks. *Science of Computer Programming*, 75:440–469, 2010.

[58] L. Song and J.C. Godskesen. Probabilistic mobility models for mobile and wireless networks. In *IFIP TCS*, 2010.

[59] F. Stajano and R. Anderson. The resurrecting duckling: Security issues for ad-hoc wireless networks. In *Security Protocols Workshop*, pages 172–194, 1999.

[60] Sudarshan Vasudevan, Jim Kurose, and Don Towsley. Design and Analysis of a Leader Election Algorithm for Mobile Ad Hoc Networks. In *ICNP*, pages 350–360. IEEE Computer Society, 2004.

[61] Dennis M. Volpano and Geoffrey Smith. Secure Information Flow in a Multi-Threaded Imperative Language. In *POPL*, pages 355–364. ACM Press, 1998.

[62] Manel Guerrero Zapata and N. Asokan. Securing Ad Hoc Routing Protocols. In *WiSe*, pages 1–10. ACM, 2002.

[63] C. Zhang, X. Zhu, Y. Song, and Y. Fang. A formal study of trust-based routing in wireless ad hoc networks. In *INFOCOM*, pages 2838–2846, 2010.

A Proofs

Proof of Theorem 6.1

1. Let us prove the first part of the statement.

   (a) The proof is by induction on why $M \xrightarrow{m?\tilde{v} \triangleright \mathcal{D}}_{\sigma} M'$. The base cases are when the transition $M \xrightarrow{m?\tilde{v} \triangleright \mathcal{D}}_{\sigma} M'$ is obtained by an application of one of the following rule: (Rcv), (RcvEnb). The most interesting case is the first one. Thus, $M = n[\sigma?(\tilde{x}).P]_T$, $M' = n[\{^{\tilde{v}}/_{\tilde{x}}\}P]_T$, $\mathcal{D} = n$ and $T(n,m) \geq \sigma$, as required.

   As to the inductive case, let us suppose the transition $M \xrightarrow{m?\tilde{v} \triangleright \mathcal{D}}_{\sigma} M'$ id derived by an application of one of the following rules: (Sum), (RcvPar). We show details only for (RcvPar); the other case is similar. Let be $M \xrightarrow{m?\tilde{v} \triangleright \mathcal{D}}_{\sigma} M'$ by an application of rule (RcvPar) with $M = M_1 \mid M_2$ and $M' = M_1' \mid M_2'$, for some M_1, M_2, M_1' and M_2', because $M_1 \xrightarrow{m?\tilde{v} \triangleright \mathcal{D}'}_{\sigma} M_1'$ and $M_2 \xrightarrow{m?\tilde{v} \triangleright \mathcal{D}''}_{\sigma} M_2'$, where $\mathcal{D} := \mathcal{D}' \cup \mathcal{D}''$. More precisely, let

   $$M \equiv \prod_i n_i[P_i]_{T_i} = \prod_k n_k[P_k]_{T_k} \mid \prod_j n_j[P_j]_{T_j}$$

   and

   $$M' \equiv \prod_i n_i[P_i']_{T_i'} = \prod_k n_k[P_k']_{T_k} \mid \prod_j n_j[P_j']_{T_j}$$

   for appropriate processes and tags, where

   $$\begin{array}{ll} M_1 = \prod_k n_k[P_k]_{T_k} & M_1' = \prod_k n_k[P_k']_{T_k} \\ M_2 = \prod_j n_j[P_j]_{T_j} & M_2' = \prod_j n_j[P_j']_{T_j}. \end{array}$$

   If $P_k' \neq P_k$, for some k or if $P_j' \neq P_j$, for some j, the result follows by inductive hypothesis.

(b) This case applies only for transitions at level trust. It is proved by induction on $M \xrightarrow{m?\tilde{v}\triangleright\mathcal{D}}_{\mathsf{trust}} M'$. The proof is similar to the previous case.

2. Let us prove the second part of the statement.

(a) The proof is by induction on the transition $M \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_{\sigma} M'$. The base cases are when $M \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_{\sigma} M'$ is obtained by an application of one of the following rules: rules (MCast), (UCast), (DTrust), or (SndRcm). These cases are immediate.

As to the inductive case, let us suppose that $M \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_{\sigma} M'$ is derived by the application of one of the following rules: (Sum), (Sync). We show details only for rule (Sync); the other case is similar. Thus, let $M \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_{\sigma} M'$ by an application of rule (Sync), with $M = M_1 \mid M_2$ and $M' = M_1' \mid M_2'$, for some M_1, M_2, M_1' and M_2', because $M_1 \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_{\sigma} M_1'$ and $M_2 \xrightarrow{m?\tilde{v}\triangleright\mathcal{D}'}_{\sigma} M_2'$ (the converse is similar), with $\mathcal{D} := \{n : T(m,n) \geq \sigma\}, \mathcal{D}' \subseteq \mathcal{D}$. More precisely, let

$$M \equiv m[P]_T \mid \prod_i n_i[P_i]_{T_i} = m[P]_T \mid \prod_k n_k[P_k]_{T_k} \mid \prod_j n_j[P_j]_{T_j}$$

and

$$M' \equiv m[P']_{T'} \mid \prod_i n_i[P_i']_{T_i'} = m[P']_{T'} \mid \prod_k n_k[P_k']_{T_k'} \mid \prod_j n_j[P_j']_{T_j'}$$

for appropriate processes and tags, where

$$\begin{array}{ll} M_1 = m[P]_T \mid \prod_k n_k[P_k]_{T_k} & M_1' = m[P']_{T'} \mid \prod_k n_k[P_k']_{T_k'} \\ M_2 = \prod_j n_j[P_j]_{T_j} & M_2' = \prod_j n_j[P_j']_{T_j'} \ . \end{array}$$

By inductive hypothesis, if $P_k' \neq P_k$, for some k, then $T(m, n_k) \geq \sigma$ and $T_k(n_k, m) \geq \sigma$. Similarly, by inductive hypothesis, if $T_k' \neq T_k$, for some k, then $T(m, n_k) \geq \sigma$ and $T_k(n_k, m) \geq \sigma$. By applying the first part of this theorem, if $P_j' \neq P_j$, for some j, then $T_j(n_j, m) \geq \sigma$. It remains to prove that $T(m, n_j) \geq \sigma$. Again, by applying the first part of this theorem, we have $n_j \in \mathcal{D}'$. As $\mathcal{D}' \subseteq \mathcal{D}$ and $\mathcal{D} := \{n : T(m,n) \geq \sigma\}$ it holds that $T(m, n_j) \geq \sigma$, as required.

(b) This case applies only for transitions at level trust. The proof is by induction on the transition $M \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_{\mathsf{trust}} M'$. The proof is similar to the previous case.

□

Proof of Corollary 6.2

Let us prove the second part of the statement. The first part is simpler. We proceed by contradiction. We prove that if $P'_i \neq P_i$ or $T'_i \neq T_i$, for some i, then $T(m, n_i) \neq \mathsf{bad}$ and $T_i(n_i, m) \neq \mathsf{bad}$. Indeed, by Theorem 6.1(2a) if $P'_i \neq P_i$ it holds that $T(m, n_i) \geq \rho$ and $T_i(n_i, m){\geq}\rho$ and by Theorem 6.1(2b) if $T'_i \neq T_i$ it holds that $T(m, n_i) \geq \rho$ and $T_i(n_i, m) \geq \rho$. By construction we know that $\rho \neq \mathsf{bad}$. This contradicts the hypotheses. □

Proof of Theorem 8.4

We prove that the relation

$$\mathcal{S} \overset{\text{def}}{=} \{\big(M \mid O \,,\, N \mid O\big) \text{ for all } O \text{ such that } M \approx_\delta N\}$$

is a δ-bisimulation. We proceed by case analysis on why $M \mid O \xrightarrow{\alpha}_\rho \widehat{M}$, with $\rho \leq \delta$.

- Let $M \mid O \xrightarrow{m!\tilde{v}\blacktriangleright\mathcal{D}}_\rho \widehat{M}$ by an application of the transition rule (Obs), because $M \mid O \xrightarrow{m!\tilde{v}\triangleright\widehat{\mathcal{D}}}_\rho \widehat{M}$, with $\mathcal{D} = \widehat{\mathcal{D}} \setminus \mathsf{nds}(M \mid O) \neq \emptyset$. There are the following sub-cases.

  – Let $M \mid O \xrightarrow{m!\tilde{v}\triangleright\widehat{\mathcal{D}}}_\rho \widehat{M}$ by an application of rule (Sync) because $M \xrightarrow{m!\tilde{v}\triangleright\widehat{\mathcal{D}}}_\rho M'$ and $O \xrightarrow{m?\tilde{v}\triangleright\mathcal{D}''}_\rho O'$, with $\widehat{M} = M' \mid O'$, $\mathcal{D}'' \subseteq \widehat{\mathcal{D}}$ and $\mathcal{D}'' \subseteq \mathsf{nds}(O)$. Let $\mathcal{D}' = \widehat{\mathcal{D}} \setminus \mathsf{nds}(M)$. As $\mathcal{D} = \widehat{\mathcal{D}} \setminus \mathsf{nds}(M \mid O) \neq \emptyset$ it follows that $\mathcal{D}' \neq \emptyset$. By an application of rule (Obs) we can derive $M \xrightarrow{m!\tilde{v}\blacktriangleright\mathcal{D}'}_\rho M'$. As $M \approx_\delta N$ there is N' such that $N \xRightarrow{m!\tilde{v}\blacktriangleright\mathcal{D}'}_\rho N'$ with $M' \approx_\delta N'$. Since the action $m!\tilde{v}\blacktriangleright\mathcal{D}'$ can be generated only by an application of rule (Obs) this implies that there are N_1 and N_2 such that

  $$N \Rightarrow_\rho N_1 \xrightarrow{m!\tilde{v}\triangleright\widehat{\mathcal{D}'}}_\rho N_2 \Rightarrow_\rho N'$$

  with $\mathcal{D}' = \widehat{\mathcal{D}'} \setminus \mathsf{nds}(N) \neq \emptyset$. We recall that $\mathcal{D}'' \subseteq \widehat{\mathcal{D}}$ and $\mathcal{D}'' \subseteq \mathsf{nds}(O)$. By node-uniqueness, $\mathsf{nds}(M) \cap \mathsf{nds}(O) = \emptyset$. As a consequence,

  $$\begin{aligned} \mathcal{D}'' &= \mathcal{D}'' \setminus \mathsf{nds}(M) \\ &\subseteq \widehat{\mathcal{D}} \setminus \mathsf{nds}(M) \\ &= \mathcal{D}' \\ &= \widehat{\mathcal{D}'} \setminus \mathsf{nds}(N) \\ &\subseteq \widehat{\mathcal{D}'} \ . \end{aligned}$$

  Thus, by several applications of rule (TauPar) and one application of rule (Sync) we have

  $$N \mid O \Rightarrow_\rho N_1 \mid O \xrightarrow{m!\tilde{v}\triangleright\widehat{\mathcal{D}'}}_\rho N_2 \mid O' \Rightarrow_\rho N' \mid O'.$$

It holds that

$$\begin{aligned}\widehat{\mathcal{D}'} \setminus \mathsf{nds}(N \mid O) &= \widehat{\mathcal{D}'} \setminus \mathsf{nds}(N) \setminus \mathsf{nds}(O)\\ &= \mathcal{D}' \setminus \mathsf{nds}(O)\\ &= \widehat{\mathcal{D}} \setminus \mathsf{nds}(M) \setminus \mathsf{nds}(O)\\ &= \widehat{\mathcal{D}} \setminus \mathsf{nds}(M \mid O)\\ &\neq \emptyset\ .\end{aligned}$$

Thus, by one application of rule (Obs) we have $N \mid O \xRightarrow{m!\tilde{v}\blacktriangleright\mathcal{D}}_{\rho} N' \mid O'$, with $(M' \mid O', N' \mid O') \in \mathcal{S}$, as required.

– Let $M \mid O \xrightarrow{m!\tilde{v}\triangleright\widehat{\mathcal{D}}}_{\rho} \widehat{M}$ by an application of rule (Sync) because $M \xrightarrow{m?\tilde{v}\triangleright\mathcal{D}''}_{\rho} M'$ and $O \xrightarrow{m!\tilde{v}\triangleright\widehat{\mathcal{D}}}_{\rho} O'$, with $\widehat{M} = M' \mid O'$, $\mathcal{D}'' \subseteq \widehat{\mathcal{D}}$ and $\mathcal{D}'' \subseteq \mathsf{nds}(M)$. As $M \approx_{\delta} N$ then there is N' such that $N \xRightarrow{m?\tilde{v}\triangleright\mathcal{D}''}_{\rho} N'$ with $M' \approx_{\delta} N'$. This implies that there are N_1 and N_2 such that

$$N \Rightarrow_{\rho} N_1 \xrightarrow{m?\tilde{v}\triangleright\mathcal{D}''}_{\rho} N_2 \Rightarrow_{\rho} N'.$$

Then by several applications of (TauPar) and one application of rule (Sync), as $\mathcal{D}'' \subseteq \widehat{\mathcal{D}}$, we have

$$N \mid O \Rightarrow_{\rho} N_1 \mid O \xrightarrow{m!\tilde{v}\triangleright\widehat{\mathcal{D}}}_{\rho} N_2 \mid O' \Rightarrow_{\rho} N' \mid O'.$$

Since $M \approx_{\delta} N$, by Proposition 8.3 it follows that $\mathsf{nds}(M) = \mathsf{nds}(N)$. As a consequence,

$$\begin{aligned}\widehat{\mathcal{D}} \setminus \mathsf{nds}(N \mid O) &= \widehat{\mathcal{D}} \setminus \mathsf{nds}(N) \setminus \mathsf{nds}(O)\\ &= \widehat{\mathcal{D}} \setminus \mathsf{nds}(M) \setminus \mathsf{nds}(O)\\ &= \widehat{\mathcal{D}} \setminus \mathsf{nds}(M \mid O)\\ &\neq \emptyset\ .\end{aligned}$$

Thus, by one application of rule (Obs) we have $N \mid O \xRightarrow{m!\tilde{v}\blacktriangleright\mathcal{D}}_{\rho} N' \mid O'$, with $(M' \mid O', N' \mid O') \in \mathcal{S}$, as required.

- Let $M \mid O \xrightarrow{\tau}_{\rho} \widehat{M}$ by an application of rule (Shh), because $M \mid O \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_{\rho'} \widehat{M}$, with $\mathcal{D} \subseteq \mathsf{nds}(M \mid O)$ and $\mathsf{trust} < \rho' \leq \delta$. There are two sub-cases.

  – Let $M \mid O \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_{\rho'} \widehat{M}$ by an application of rule (Sync) because $M \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_{\rho'} M'$ and $O \xrightarrow{m?\tilde{v}\triangleright\mathcal{D}''}_{\rho'} O'$, with $\widehat{M} = M' \mid O'$, $\mathcal{D}'' \subseteq \mathcal{D}$, $\mathcal{D}'' \subseteq \mathsf{nds}(O)$ and $\mathcal{D} \subseteq \mathsf{nds}(M \mid O)$. There are two sub-cases.

    ∗ Let $\mathcal{D} \subseteq \mathsf{nds}(M)$. Then by an application of rule (Shh) we have $M \xrightarrow{\tau}_{\rho} M'$. As $M \approx_{\delta} N$, there is N' such that $N \Rightarrow_{\rho} N'$, with $M' \approx_{\delta} N'$. By several applications of rule (TauPar) we have $N \mid O \Rightarrow_{\rho} N' \mid O'$, with $(M' \mid O', N' \mid O') \in \mathcal{S}$, as required.

∗ Let $\mathcal{D} \not\subseteq \mathsf{nds}(M)$. By an application of rule (Obs) we have $M \xrightarrow{m!\tilde{v}\blacktriangleright\widehat{\mathcal{D}}}_{\rho'} M'$, with $\widehat{\mathcal{D}} = \mathcal{D} \setminus \mathsf{nds}(M) \neq \emptyset$. As $M \approx_\delta N$, there is N' such that $N \xRightarrow{m!\tilde{v}\blacktriangleright\widehat{\mathcal{D}}}_{\rho'} N'$, with $M' \approx_\delta N'$. Since the action $m!\tilde{v}\blacktriangleright\widehat{\mathcal{D}}$ can be generated only by an application of rule (Obs) this implies that there are N_1 and N_2 such that

$$N \Rightarrow_{\rho'} N_1 \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}'}_{\rho'} N_2 \Rightarrow_{\rho'} N'$$

with $\widehat{\mathcal{D}} = \mathcal{D}' \setminus \mathsf{nds}(N) \neq \emptyset$. We recall that $\mathcal{D}'' \subseteq \mathcal{D}$ and $\mathcal{D}'' \subseteq \mathsf{nds}(O)$. By node-uniqueness, $\mathsf{nds}(M) \cap \mathsf{nds}(O) = \emptyset$. As a consequence,

$$\begin{aligned} \mathcal{D}'' &= \mathcal{D}'' \setminus \mathsf{nds}(M) \\ &\subseteq \mathcal{D} \setminus \mathsf{nds}(M) \\ &= \widehat{\mathcal{D}} \\ &= \mathcal{D}' \setminus \mathsf{nds}(N) \\ &\subseteq \mathcal{D}' \ . \end{aligned}$$

Thus, by several applications of rule (TauPar) and by one application of rule (Sync) we have:

$$N \mid O \Rightarrow_{\rho'} N_1 \mid O \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}'}_{\rho'} N_2 \mid O' \Rightarrow_{\rho'} N' \mid O'.$$

Notice that

$$\begin{aligned} \mathcal{D}' \setminus \mathsf{nds}(N \mid O) &= \mathcal{D}' \setminus \mathsf{nds}(N) \setminus \mathsf{nds}(O) \\ &= \widehat{\mathcal{D}} \setminus \mathsf{nds}(O) \\ &= \mathcal{D} \setminus \mathsf{nds}(M) \setminus \mathsf{nds}(O) \\ &= \mathcal{D} \setminus \mathsf{nds}(M \mid O) \\ &= \emptyset \ . \end{aligned}$$

Since $\mathcal{D}' \subseteq \mathsf{nds}(N \mid O)$, by one application of rule (Shh) we have $N \mid O \Rightarrow_\rho N' \mid O'$, with $(M' \mid O', N' \mid O') \in \mathcal{S}$, as required.

– Let $M \mid O \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_{\rho'} \widehat{M}$ by an application of rule (Sync) because $M \xrightarrow{m?\tilde{v}\triangleright\mathcal{D}''}_{\rho'} M'$ and $O \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_{\rho'} O'$, with $\widehat{M} = M' \mid O'$ and $\mathcal{D}'' \subseteq \mathcal{D}$. As $M \approx_\delta N$ there is N' such that $N \xRightarrow{m?\tilde{v}\triangleright\mathcal{D}''}_{\rho'} N'$ with $M' \approx_\delta N'$. This implies that there are N_1 and N_2 such that

$$N \Rightarrow_{\rho'} N_1 \xrightarrow{m?\tilde{v}\triangleright\mathcal{D}''}_{\rho'} N_2 \Rightarrow_{\rho'} N'.$$

Then, by several applications of rule (TauPar) and one application of rule (Sync), as $\mathcal{D}'' \subseteq \mathcal{D}$, we have

$$N \mid O \Rightarrow_{\rho'} N_1 \mid O \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_{\rho'} N_2 \mid O' \Rightarrow_{\rho'} N' \mid O'.$$

Since $M \approx_\delta N$, by Proposition 8.3 it follows that $\mathsf{nds}(M) = \mathsf{nds}(N)$. As a consequence,

$$\mathcal{D} \setminus \mathsf{nds}(N \mid O) = \mathcal{D} \setminus \mathsf{nds}(M \mid O) = \emptyset \ .$$

Thus, by one application of rule (Shh) we have $N \mid O \Rightarrow_\rho N' \mid O'$ and $(M' \mid O', N' \mid O') \in \mathcal{S}$, as required.

- Let $M \mid O \xrightarrow{\tau}_\rho \widehat{M}$ by an application of rule (TauPar). This case is easy.

- Let $M \mid O \xrightarrow{m?\tilde{v} \triangleright \mathcal{D}}_\rho \widehat{M}$ by an application of the transition rule (RcvPar) because $M \xrightarrow{m?\tilde{v} \triangleright \mathcal{D}'}_\rho M'$ and $O \xrightarrow{m?\tilde{v} \triangleright \mathcal{D}''}_{\mathsf{trust}} O'$, with $\mathcal{D} := \mathcal{D}' \cup \mathcal{D}''$, $\widehat{M} = M' \mid O'$. This case is easy.

□

In order to prove Theorem 8.5 we use the following auxiliary lemmas.

Lemma A.1

1. *If $M \xrightarrow{m!\tilde{v} \triangleright \mathcal{D}}_\sigma M'$, where $\mathcal{D}$ contains more than one node, then there are N, P, T such that $M \equiv m[\sigma!\langle\tilde{v}\rangle.P]_T \mid N$ and $\mathcal{D} = \{n : T(m,n) \geq \rho\}$.*

2. *If $M \xrightarrow{m!\tilde{v} \triangleright \mathcal{D}}_\sigma M'$, with $\mathcal{D} = n$, for some n, then there are N, P, T such that $M \equiv m[\sigma!\langle\tilde{v}\rangle.P]_T \mid N$ or $M \equiv m[\sigma!\langle\tilde{v}\rangle_n.P]_T \mid N$ with $T(m,n) \geq \rho$.*

Proof

1. By induction on why $M \xrightarrow{m!\tilde{v} \triangleright \mathcal{D}}_\sigma M'$. The base case is when $M \xrightarrow{m!\tilde{v} \triangleright \mathcal{D}}_\sigma M'$ is obtained by an application of one of the following rules: (MCast), (DTrust), (SndRcm). We show the details only for rule (MCast); the other cases are similar. In this case, $M = m[\sigma!\langle\tilde{v}\rangle.P]_T \equiv m[\sigma!\langle\tilde{v}\rangle.P]_T \mid \mathbf{0}$, for some P and T, with $\mathcal{D} = \{n : T(m,n) \geq \rho\}$. As to the inductive case, let us suppose that $M \xrightarrow{m!\tilde{v} \triangleright \mathcal{D}}_\sigma M'$ is obtained by an application of one of the following rules: (Sum), (Sync). We only consider the case for rule (Sync); the other case is similar. Let $M \xrightarrow{m!\tilde{v} \triangleright \mathcal{D}}_\sigma M'$ by an application of rule (Sync), because $M_1 \xrightarrow{m!\tilde{v} \triangleright \mathcal{D}}_\sigma M_1'$ and $M_2 \xrightarrow{m?\tilde{v} \triangleright \mathcal{D}'}_\sigma M_2'$, and $M = M_1 \mid M_2$ and $M' = M_1' \mid M_2'$ for some M_1, M_1', M_2, M_2' and $\mathcal{D}'$. By inductive hypothesis it holds that $M_1 \equiv m[\sigma!\langle\tilde{v}\rangle.P]_T \mid N$, for some N, P, T with $\mathcal{D} = \{n : T(m,n) \geq \rho\}$. Thus, $M \equiv m[\sigma!\langle\tilde{v}\rangle.P]_T \mid N \mid M_2$, as required.

2. By induction on why $M \xrightarrow{m!\tilde{v} \triangleright n}_\sigma M'$. This case is similar to the previous one.

□

Lemma A.2

1. *If* $M \xrightarrow{m!\tilde{v}\blacktriangleright\mathcal{D}}_{\sigma} M'$ *then* $M \downarrow^{\sigma}_{n}$*, for all* $n \in \mathcal{D}$*.*
2. *If* $M \downarrow^{\sigma}_{n}$ *then there is a value* $\tilde{v}$ *and a set of nodes* $\mathcal{D}$*, with* $n \in \mathcal{D}$*, such that* $M \xrightarrow{m!\tilde{v}\blacktriangleright\mathcal{D}}_{\sigma} M'$*,*

Proof By Lemma A.1 and by Definition 7.1. □

Proof of Theorem 8.5
The preservation of σ-barbs follows by Lemma A.2; the reduction-closure follows by definition, while contextuality follows by Theorem 8.4. □

Proof of Theorem 9.2
We prove that the relation

$$\mathcal{S} \overset{\text{def}}{=} \{\big(M \mid H\, ,\, N \mid K\big) : H, K \in \mathcal{H}_\delta,\, M \approx_\delta N \text{ and } H \approx_{\mathsf{trust}} K\}$$

is a δ-bisimulation. By case analysis on the transition $M \mid H \xrightarrow{\alpha}_{\rho} \widehat{M}$, with $\rho \leq \delta$.

- Let $M \mid H \xrightarrow{m!\tilde{v}\blacktriangleright\mathcal{D}}_{\rho} \widehat{M}$ by an application of rule (Obs), with $\rho > \mathsf{trust}$, because $M \mid H \xrightarrow{m!\tilde{v}\triangleright\widehat{\mathcal{D}}}_{\rho} \widehat{M}$. Since $H \in \mathcal{H}_\delta$, the transition $M \mid H \xrightarrow{m!\tilde{v}\triangleright\widehat{\mathcal{D}}}_{\rho} \widehat{M}$ can be derived only by an application of rule (Sync) because $M \xrightarrow{m!\tilde{v}\triangleright\widehat{\mathcal{D}}}_{\rho} M'$ and $H \xrightarrow{m?\tilde{v}\triangleright\emptyset}_{\rho} H$, with $\mathcal{D} = \widehat{\mathcal{D}} \setminus \mathsf{nds}(M \mid H) \neq \emptyset$ and $\widehat{M} = M' \mid H$. Let $\mathcal{D}' = \widehat{\mathcal{D}} \setminus \mathsf{nds}(M)$. As $\mathcal{D} \neq \emptyset$ it follows that $\mathcal{D}' \neq \emptyset$. Thus, we can apply rule (Obs) to derive $M \xrightarrow{m!\tilde{v}\blacktriangleright\mathcal{D}'} M'$. As $M \approx_\delta N$ there is N' such that $N \xRightarrow{m!\tilde{v}\blacktriangleright\mathcal{D}'}_{\rho} N'$ with $M' \approx_\delta N'$. Since the action $m!\tilde{v}\blacktriangleright\mathcal{D}'$ can be generated only by an application of rule (Obs) there are N_1 and N_2 such that

  $$N \Rightarrow_\rho N_1 \xrightarrow{m!\tilde{v}\triangleright\widehat{\mathcal{D}'}}_{\rho} N_2 \Rightarrow_\rho N'$$

  with $\mathcal{D}' = \widehat{\mathcal{D}'} \setminus \mathsf{nds}(N) \neq \emptyset$. By node uniqueness, since $m \in \mathsf{nds}(N)$ it follows that $m \notin \mathsf{nds}(K)$. Thus, $K \xrightarrow{m?\tilde{v}\triangleright\emptyset} K$. By several applications of rule (TauPar) and one application of rule (Sync) we have:

  $$N \mid K \Rightarrow_\rho N_1 \mid K \xrightarrow{m!\tilde{v}\triangleright\widehat{\mathcal{D}'}}_{\rho} N_2 \mid K \Rightarrow_\rho N' \mid K.$$

  Since $H \approx_{\mathsf{trust}} K$, by Proposition 8.3 it follows that $\mathsf{nds}(H) = \mathsf{nds}(K)$. Hence,

  $$\begin{array}{rcl} \mathcal{D} & := & \widehat{\mathcal{D}} \setminus \mathsf{nds}(M \mid H) \\ & = & \widehat{\mathcal{D}} \setminus \mathsf{nds}(M) \setminus \mathsf{nds}(H) \\ & = & \mathcal{D}' \setminus \mathsf{nds}(H) \\ & = & \widehat{\mathcal{D}'} \setminus \mathsf{nds}(N) \setminus \mathsf{nds}(K) \\ & = & \widehat{\mathcal{D}'} \setminus \mathsf{nds}(N \mid K) \ . \end{array}$$

As $\mathcal{D} \neq \emptyset$, by one application of rule (Obs) we have $N \mid K \xRightarrow{m!\tilde{v}\blacktriangleright\mathcal{D}}_{\rho} N' \mid K$, with $\big(M' \mid H \,,\, N' \mid K\big) \in \mathcal{S}$, as required.

- Let $M \mid H \xrightarrow{m!\tilde{v}\blacktriangleright\mathcal{D}}_{\mathsf{trust}} \widehat{M}$ by an application of the transition rule (Obs) because $M \mid H \xrightarrow{m!\tilde{v}\triangleright\widehat{\mathcal{D}}}_{\mathsf{trust}} \widehat{M}$, with $\mathcal{D} = \widehat{\mathcal{D}} \setminus \mathsf{nds}(M \mid H) \neq \emptyset$. We have the following possibilities:

  - Let $M \mid H \xrightarrow{m!\tilde{v}\triangleright\widehat{\mathcal{D}}}_{\mathsf{trust}} \widehat{M}$ by an application of the transition rule (Sync) because $M \xrightarrow{m!\tilde{v}\triangleright\widehat{\mathcal{D}}}_{\mathsf{trust}} M'$ and $H \xrightarrow{m?\tilde{v}\triangleright\mathcal{D}''}_{\mathsf{trust}} H'$, with $\widehat{M} = M' \mid H'$, $\mathcal{D}'' \subseteq \widehat{\mathcal{D}}$, $\mathcal{D}'' \subseteq \mathsf{nds}(H)$ and $H' \in \mathcal{H}_{\delta}$. Let $\mathcal{D}' = \widehat{\mathcal{D}} \setminus \mathsf{nds}(M)$. As $\mathcal{D} = \widehat{\mathcal{D}} \setminus \mathsf{nds}(M \mid H) \neq \emptyset$ it follows that $\mathcal{D}' \neq \emptyset$. As a consequence, we can apply rule (Obs) to derive $M \xrightarrow{m!\tilde{v}\blacktriangleright\mathcal{D}'}_{\mathsf{trust}} M'$. As $M \approx_{\delta} N$, there is N' such that $N \xRightarrow{m!\tilde{v}\blacktriangleright\mathcal{D}'}_{\mathsf{trust}} N'$ with $M' \approx_{\delta} N'$. Since the action $m!\tilde{v}\blacktriangleright\mathcal{D}'$ can be generated only by an application of rule (Obs) there are N_1 and N_2 such that

    $$N \Rightarrow_{\mathsf{trust}} N_1 \xrightarrow{m!\tilde{v}\triangleright\widehat{\mathcal{D}'}}_{\mathsf{trust}} N_2 \Rightarrow_{\mathsf{trust}} N'$$

    with $\mathcal{D}' = \widehat{\mathcal{D}'} \setminus \mathsf{nds}(N) \neq \emptyset$. As $H \approx_{\mathsf{trust}} K$ and $H \xrightarrow{m?\tilde{v}\triangleright\mathcal{D}''} H'$, there is K' such that $K \xRightarrow{m?\tilde{v}\triangleright\mathcal{D}''}_{\mathsf{trust}} K'$ with $H' \approx_{\mathsf{trust}} K'$ and $K' \in \mathcal{H}_{\delta}$. This means there are K_1 and K_2 such that

    $$K \Rightarrow_{\mathsf{trust}} K_1 \xrightarrow{m?\tilde{v}\triangleright\mathcal{D}''}_{\mathsf{trust}} K_2 \Rightarrow_{\mathsf{trust}} K' \ .$$

    We recall that $\mathcal{D}'' \subseteq \widehat{\mathcal{D}}$ and $\mathcal{D}'' \subseteq \mathsf{nds}(H)$. By node-uniqueness, $\mathsf{nds}(M) \cap \mathsf{nds}(H) = \emptyset$. As a consequence,

    $$\begin{aligned} \mathcal{D}'' &= \mathcal{D}'' \setminus \mathsf{nds}(M) \\ &\subseteq \widehat{\mathcal{D}} \setminus \mathsf{nds}(M) \\ &= \mathcal{D}' \\ &= \widehat{\mathcal{D}'} \setminus \mathsf{nds}(N) \\ &\subseteq \widehat{\mathcal{D}'} \ . \end{aligned}$$

    Thus, by several applications of rule (TauPar) and one application of rule (Sync) we have

    $$N \mid K \Rightarrow_{\mathsf{trust}} N_1 \mid K_1 \xrightarrow{m!\tilde{v}\triangleright\widehat{\mathcal{D}'}}_{\mathsf{trust}} N_2 \mid K_2 \Rightarrow_{\mathsf{trust}} N' \mid K'.$$

    Since $H \approx_{\mathsf{trust}} K$, by Proposition 8.3 it follows that $\mathsf{nds}(K) = \mathsf{nds}(H)$. As a consequence,

    $$\begin{aligned} \mathcal{D} &= \widehat{\mathcal{D}} \setminus \mathsf{nds}(M \mid H) \\ &= \widehat{\mathcal{D}} \setminus \mathsf{nds}(M) \setminus \mathsf{nds}(H) \\ &= \mathcal{D}' \setminus \mathsf{nds}(H) \\ &= \widehat{\mathcal{D}'} \setminus \mathsf{nds}(N) \setminus \mathsf{nds}(K) \\ &= \widehat{\mathcal{D}'} \setminus \mathsf{nds}(N \mid K) \quad . \end{aligned}$$

As $\mathcal{D} \neq \emptyset$, by an application of rule (Obs) we can derive $N \mid K \xRightarrow{m!\tilde{v}\blacktriangleright\mathcal{D}}_{\rho} N' \mid K'$, with $(M' \mid H', N' \mid K') \in \mathcal{S}$, as required.

- Let $M \mid H \xrightarrow{m!\tilde{v}\triangleright\widehat{\mathcal{D}}}_{\mathsf{trust}} \widehat{M}$ by an application of the transition rule (Sync) because $M \xrightarrow{m?\tilde{v}\triangleright\mathcal{D}''}_{\mathsf{trust}} M'$ and $H \xrightarrow{m!\tilde{v}\triangleright\widehat{\mathcal{D}}}_{\mathsf{trust}} H'$, with $\widehat{M} = M' \mid H'$, $\mathcal{D}'' \subseteq \widehat{\mathcal{D}}$, $\mathcal{D}'' \subseteq \mathsf{nds}(M)$ and $H' \in \mathcal{H}_{\delta}$. As $M \approx_{\delta} N$ then there is N' such that $N \xRightarrow{m?\tilde{v}\triangleright\mathcal{D}''}_{\mathsf{trust}} N'$ with $M' \approx_{\delta} N'$. This implies that there are N_1 and N_2 such that
$$N \Rightarrow_{\mathsf{trust}} N_1 \xrightarrow{m?\tilde{v}\triangleright\mathcal{D}''}_{\mathsf{trust}} N_2 \Rightarrow_{\mathsf{trust}} N'.$$
Let $\mathcal{D}' = \widehat{\mathcal{D}} \setminus \mathsf{nds}(H)$. As $\mathcal{D} = \widehat{\mathcal{D}} \setminus \mathsf{nds}(M \mid H) \neq \emptyset$ it follows that $\mathcal{D}' \neq \emptyset$. By an application for rule (Obs) we have $H \xrightarrow{m!\tilde{v}\blacktriangleright\mathcal{D}'}_{\mathsf{trust}} H'$, with $\mathcal{D}' = \widehat{\mathcal{D}} \setminus \mathsf{nds}(H) \neq \emptyset$. As $H \approx_{\mathsf{trust}} K$ there is K' such that $K \xRightarrow{m!\tilde{v}\blacktriangleright\mathcal{D}'}_{\mathsf{trust}} K'$, with $H' \approx_{\mathsf{trust}} K'$ and $K' \in \mathcal{H}_{\delta}$. This implies that there are K_1 and K_2 such that
$$K \Rightarrow_{\mathsf{trust}} K_1 \xrightarrow{m!\tilde{v}\triangleright\widehat{\mathcal{D}'}}_{\mathsf{trust}} K_2 \Rightarrow_{\mathsf{trust}} K'$$
with $\mathcal{D}' = \widehat{\mathcal{D}'} \setminus \mathsf{nds}(K)$. We recall that $\mathcal{D}'' \subseteq \widehat{\mathcal{D}}$ and $\mathcal{D}'' \subseteq \mathsf{nds}(M)$. By node-uniqueness, $\mathsf{nds}(M) \cap \mathsf{nds}(H) = \emptyset$. As a consequence,
$$\begin{aligned} \mathcal{D}'' &= \mathcal{D}'' \setminus \mathsf{nds}(H) \\ &\subseteq \widehat{\mathcal{D}} \setminus \mathsf{nds}(H) \\ &= \mathcal{D}' \\ &= \widehat{\mathcal{D}'} \setminus \mathsf{nds}(K) \\ &\subseteq \widehat{\mathcal{D}'}\ . \end{aligned}$$
Thus, by several applications of rule (TauPar) and one application of rule (Sync), as $\mathcal{D}'' \subseteq \widehat{\mathcal{D}}$, we have
$$N \mid K \Rightarrow_{\mathsf{trust}} N_1 \mid K_1 \xrightarrow{m!\tilde{v}\triangleright\widehat{\mathcal{D}'}}_{\mathsf{trust}} N_2 \mid K_2 \Rightarrow_{\mathsf{trust}} N' \mid K'.$$
Since $M \approx_{\rho} K$, by Proposition 8.3 it follows that $\mathsf{nds}(M) = \mathsf{nds}(N)$. As a consequence,
$$\begin{aligned} \mathcal{D} &= \widehat{\mathcal{D}} \setminus \mathsf{nds}(M \mid H) \\ &= \widehat{\mathcal{D}} \setminus \mathsf{nds}(M) \setminus \mathsf{nds}(H) \\ &= \widehat{\mathcal{D}} \setminus \mathsf{nds}(H) \setminus \mathsf{nds}(M) \\ &= \mathcal{D}' \setminus \mathsf{nds}(M) \\ &= \mathcal{D}' \setminus \mathsf{nds}(N) \\ &= \widehat{\mathcal{D}'} \setminus \mathsf{nds}(K) \setminus \mathsf{nds}(N) \\ &= \widehat{\mathcal{D}'} \setminus \mathsf{nds}(N \mid K)\ . \end{aligned}$$
As $\mathcal{D} \neq \emptyset$, by one application of rule (Obs) we have $N \mid K \xRightarrow{m!\tilde{v}\blacktriangleright\mathcal{D}}_{\rho} N' \mid K'$, with $(M' \mid H', N' \mid K') \in \mathcal{S}$, as required.

- $M \mid H \xrightarrow{\tau}_{\rho} \widehat{M}$ by an application of rule (Shh), because $M \mid H \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_{\rho'} \widehat{M}$, with $\mathcal{D} \subseteq \mathsf{nds}(M \mid H)$. Let us suppose $\rho' > \mathsf{trust}$. Since $H \in \mathcal{H}_{\rho}$, the only possibility is that $M \mid H \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_{\rho'} \widehat{M}$ by an application of rule (Sync) because $M \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_{\rho'} M$ and $H \xrightarrow{m?\tilde{v}\triangleright\emptyset} H$, with $\widehat{M} = M' \mid H$ and $m \notin \mathsf{nds}(H)$. There are two sub-cases.
  - Let $\mathcal{D} \subseteq \mathsf{nds}(M)$. Then by an application of rule (Shh) we have $M \xrightarrow{\tau}_{\rho} M'$. As $M \approx_{\delta} N$, there is N' such that $N \Rightarrow_{\rho} N'$, with $M' \approx_{\delta} N'$. By several applications of rule (TauPar) we have $N \mid K \Rightarrow_{\rho} N' \mid K$, with $(M' \mid H, N' \mid K) \in \mathcal{S}$, as required.
  - Let $\mathcal{D} \not\subseteq \mathsf{nds}(M)$. Then by an application of rule (Obs) we have $M \xrightarrow{m!\tilde{v}\blacktriangleright\widehat{\mathcal{D}}}_{\rho'} M'$, with $\widehat{\mathcal{D}} = \mathcal{D} \setminus \mathsf{nds}(M) \neq \emptyset$. As $M \approx_{\delta} N$, there is N' such that $N \xRightarrow{m!\tilde{v}\blacktriangleright\widehat{\mathcal{D}}}_{\rho'} N'$, with $M' \approx_{\delta} N'$. Since the action $m!\tilde{v}\blacktriangleright\widehat{\mathcal{D}}$ can be generated only by an application of rule (Obs) this implies that there are N_1 and N_2 such that
  $$N \Rightarrow_{\rho'} N_1 \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}'}_{\rho'} N_2 \Rightarrow_{\rho'} N'$$
  with $\widehat{\mathcal{D}} = \mathcal{D}' \setminus \mathsf{nds}(N) \neq \emptyset$. By node-uniqueness $m \notin \mathsf{nds}(K)$. By definition of rule (RcvEnb) it follows that $K \xrightarrow{m?\tilde{v}\triangleright\emptyset} K$. By several applications of rule (TauPar) and by one application of rule (Sync) we have:
  $$N \mid K \Rightarrow_{\rho'} N_1 \mid K \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}'}_{\rho'} N_2 \mid K \Rightarrow_{\rho'} N' \mid K.$$
  Since $H \approx_{\mathsf{trust}} K$, by Proposition 8.3 it follows that $\mathsf{nds}(K) = \mathsf{nds}(H)$. As a consequence,
  $$\begin{aligned} \mathcal{D}' \setminus \mathsf{nds}(N \mid K) &= \mathcal{D}' \setminus \mathsf{nds}(N) \setminus \mathsf{nds}(K) \\ &= \widehat{\mathcal{D}} \setminus \mathsf{nds}(K) \\ &= \widehat{\mathcal{D}} \setminus \mathsf{nds}(H) \\ &= \mathcal{D} \setminus \mathsf{nds}(M) \setminus \mathsf{nds}(H) \\ &= \mathcal{D} \setminus \mathsf{nds}(M \mid H) \\ &= \emptyset\ . \end{aligned}$$
  Thus, by one application of rule (Shh) we have $N \mid K \Rightarrow_{\rho} N' \mid K$, with $(M' \mid H, N' \mid K) \in \mathcal{S}$, as required.
- $M \mid H \xrightarrow{\tau}_{\rho} \widehat{M}$ by an application of rule (Shh), because $M \mid H \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_{\mathsf{trust}} \widehat{M}$, with $\mathcal{D} \subseteq \mathsf{nds}(M \mid H)$. This case is similar to the previous one.
- Let $M \mid O \xrightarrow{\tau}_{\rho} \widehat{M}$ by an application of rule (TauPar). This case is easy.

- Let $M \mid H \xrightarrow{m?\tilde{v} \triangleright \mathcal{D}}_{\rho} \widehat{M}$ with $\rho > \mathsf{trust}$. Since $H \in \mathcal{H}_{\rho}$, the only possibility is that $M \mid H \xrightarrow{m?\tilde{v} \triangleright \mathcal{D}}_{\rho} M' \mid H$ by an application of rule (RcvPar) because $M \xrightarrow{m?\tilde{v} \triangleright \mathcal{D}}_{\rho} M'$ and $H \xrightarrow{m?\tilde{v} \triangleright \emptyset}_{\rho} H$ (by an application of rule (RcvEnb)) with $\widehat{M} = M' \mid H$. This case is easy.

- Let $M \mid H \xrightarrow{m?\tilde{v} \triangleright \mathcal{D}}_{\mathsf{trust}} \widehat{M}$ by an application of the transition rule (RcvPar) because $M \xrightarrow{m?\tilde{v} \triangleright \mathcal{D}'}_{\mathsf{trust}} M'$ and $H \xrightarrow{m?\tilde{v} \triangleright \mathcal{D}''}_{\mathsf{trust}} H'$, with $\mathcal{D} := \mathcal{D}' \cup \mathcal{D}''$, $\widehat{M} = M' \mid H'$ and $H' \in \mathcal{H}_{\delta}$. This case is easy.

□