A Calculus of Trustworthy Ad Hoc Networks ⋆

Massimo Merro Eleonora Sibilio

Dipartimento di Informatica, Università degli Studi di Verona, Italy

Abstract We propose a *process calculus* for *mobile ad hoc networks* which embodies a *behaviour-based multilevel decentralised trust model.* Our trust model supports both *direct trust*, by monitoring nodes behaviour, and *indirect trust*, by 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. We define a labelled *bisimilarity* 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 are proved *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 expressed in terms of information flow is also proved.

1 Introduction

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. A *Mobile ad hoc network* (MANET) is a self-configuring network of mobile devices (also called nodes) communicating with each other via radio transceivers without relying on any base station. Lack of a fixed networking infrastructure, high mobility of the devices, shared wireless medium, cooperative behaviour, and physical vulnerability are some of the features that make challenging the design of a *security scheme* for mobile ad hoc networks.

Access control is a well-established technique for limiting access to the resources of a system to authorised programs, processes, users or other systems. Access control systems typically authenticate principles and then solicit access to resources. They rely on the definition of specific permissions, called access policies, which are recorded in some data structure such as *Access Control Lists* (ACLs). ACLs work well when access policies are set in a centralised manners. However, they are less suited to ubiquitous systems where the number of users may be very large (think of sensor networks) and/or continuously changing. In these scenarios users may be potentially unknown and, therefore, untrusted. In order to overcome these limitations, Blaze et at. [1] have introduced the notion of *Decentralised Trust Management* as an attempt to define a coherent framework in which safety critical decisions are based on trust policies relying on partial knowledge.

Trust formalisation is the subject of several academic works. According to [2], trust is the quantified belief by a *trustor*, the trusting party, with respect to the

⋆ This work has been partially supported by the national MIUR Project SOFT.

competence, honesty, security and dependability of a *trustee*, the trusted party, within a specified context. Trust information is usually represented as a collection of assertions on the reliability of the parties. The trust establishment process includes specification of valid assertions, their generation, distribution, collection and evaluation. Trust assertions may be uncertain, incomplete, stable and long term. Trust evaluation is performed by applying specific policies to assertions; the result is a trust relation between the trustor and the trustee. According to their scope and kind of trust evidence, trust frameworks can be divided in two categories: *certificate-based* and *behaviour-based.* In the first ones, trust relations are usually based on certificates, to be spread, maintained and managed either independently or cooperatively. In behaviour-based frameworks, each node performs trust evaluation based on continuous monitoring of misbehaviours of neighbours (*direct trust*). Misbehaviours typically include dropping, modification, and misrouting of packets at network layer. However, trust evaluation may also depend on node *reputation*. Node reputation usually comes from other nodes (*indirect trust*) and does not reflect direct experience of the interested node. In history-based trust models, node reputation may also depend on past behaviours.

The characteristics of mobile ad hoc networks pose a number of challenges when designing an appropriate trust model for them. Due to the lack of a fixed network infrastructure, trust models for MANETs must be decentralised and should support cooperative evaluation, according to the diversity in roles and capabilities of nodes. There are various threats to ad hoc networks, 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 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, revoke other bad nodes, vote honestly or delete keys shared with revoked nodes. So, trust frameworks for ad hoc networks should support *node revocation* to isolate malicious nodes.

Another key feature of MANETs is their support for *node mobility*: devices move while remaining connected to the network, breaking links with old neighbours and establishing 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 in such a dynamic environment. Thus, a mobile node should acquire trust information on new neighbours, and remove trust information on old neighbours that cannot be monitored anymore.

In this paper, we propose a *process calculus* for *mobile ad hoc networks* which embodies a *behaviour-based multilevel trust model.* Our trust model supports both *direct trust*, by monitoring nodes behaviour, and *indirect trust*, by collecting recommendations and spreading reputations. No information on past behaviours of nodes is recorded. We model our networks as *multilevel systems* where each device is associated to a security level depending on its role [3]. Thus, trust relations associate security levels to nodes. Process calculi have been recently used to model different aspects of wireless systems [4,5,6,7,8,9,10]. However, none of these papers address the notion of trust. 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. Our calculus is not directly concerned with cryptographic underpinnings. However, we assume the presence of a hierarchical key generation and distribution protocol [11]. Thus, messages are transmitted at a certain security level relying on an appropriate set of cryptographic keys. We provide the operational semantics of our calculus in terms of a *labelled transition system.* Our transitions are of the form

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

indicating that the network M can perform the action λ, at security level ρ, evolving into the network N. For simplicity, our operational semantics does not directly express mobility. However, we can easily adapt the approach proposed in [9] to annotate our labelled transitions with the necessary information to represent node mobility.

Our calculus enjoys two desirable security properties: *safety up to a security level* and *safety despite compromise.* Intuitively, the first property means that only *trusted* nodes, i.e. with an appropriate security level, may synchronise with other nodes. The second property says that bad (compromised) nodes, once detected, may not interact with good nodes.

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) reduction barbed congruence, a branching-time contextually-defined program equivalence. Barbed equivalences [12] are simple and intuitive but difficult to use due to the quantification on all contexts. Simpler proof techniques are based on labelled bisimilarities [13], which are co-inductive relations that characterise the behaviour of processes using a labelled transition system. We define a labelled bisimilarity parameterised on security levels proving that it represents an efficient proof method for our reduction barbed congruence.

We apply our notion of bisimilarity to prove a *non-interference* property for our networks. Intuitively, a network is *interference free* if its low security level behaviour is not affected by any activity at high security level.

2 A behaviour-based multilevel decentralised trust model

In our framework each node comes together with an extra component called *trust manager*. A trust manager consists of two main modules: the *monitoring module* and the *reputation handling module*. The first one 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.

Trust information may change over time due to mobility, temporary disconnections, recommendations, etc. As a consequence, trust knowledge may be uncertain and incomplete. The main objective of the model is to isolate *bad nodes*, i.e. nodes which do not behave as expected. For this reason, we support *node revocation*. This may happens when a node detects a misbehaviour of another node, and spreads this information to its neighbours. Repudiable evidences enable bad nodes to falsely accuse good nodes. Hence, it would be foolish to design a simple decision mechanism that revokes any node accused of misbehaviour. Thus, recommendations are always evaluated using a local security policy implementing an appropriate metric.

The basic elements 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: $\mathtt{bad} < \mathtt{trust} < \mathtt{low} < \mathtt{high}$. We use the Greek letter ρ for security levels belonging to $\mathcal{S}$. The set of *assertions* is defined as $\mathit{Assertions} = \mathit{Nodes} \times \mathit{Nodes} \times \mathcal{S}$. Thus, an assertion $\langle m, n, \rho \rangle$ says that a *node m trusts a node n at security level ρ*. A local trust store T contains a set of assertions, formally $T \subseteq \wp(\mathit{Assertions})$. 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 consistent assertions, formally $\mathcal{P} : \wp(\mathit{Assertions}) \rightarrow \wp(\mathit{Assertions})$. 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. Thus, when a node m (the trustor) wants to know the security level of a node n (the trustee), it has to check its own trust store T. For convenience, we often use T as a partial function of type $\mathit{Nodes} \rightarrow \mathit{Nodes} \rightarrow \mathcal{S}$, writing $T(m, n) = \rho$ if m considers n as a node of security level ρ. If $\rho = \mathtt{bad}$ then m considers n a *bad* (unreliable) node and stops any interaction with it.

Messages exchanged among nodes are assumed to be encrypted using a hierarchical key generation and distribution protocol [14]. 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 [14] re-keying may be relatively unexpensive if based on "low-cost" hashing operators.

3 The Calculus

In Table 1, we define the syntax of our calculus in a two-level structure, a lower one for *processes* and a upper one for *networks*. We use letters $k, l, m, n, \ldots$ for node names. The Greek symbol σ ranges over the security levels $\mathtt{low}$ and $\mathtt{high}$, 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.

Networks are collections of nodes (which represent devices) running in parallel and using channels at different security levels to communicate with each other. We use the symbol $\mathbf{0}$ to denote an empty network. We write $M \mid N$ for the parallel composition of two sub-networks M and N. 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. Processes are sequential and live within the nodes. We write nil

Table 1 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
	$\mid$	$\sigma?(\tilde{x}).P$	receiver
	$\mid$	$[\tilde{u} = \tilde{u'}]P, Q$	matching
	$\mid$	$H\langle\tilde{u}\rangle$	recursion

to denote the skip process. The sender process $\sigma!\langle\tilde{v}\rangle.P$ can broadcast the value $\tilde{v}$ at security level σ, continuing as P. 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. The receiver process $\sigma?(\tilde{x}).P$ listens on the channel for incoming communications at security level σ. Upon reception, the receiver process 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. Process $[\tilde{v} = \tilde{w}]P, Q$ is the standard "if then else" construct: it behaves as P if $\tilde{v} = \tilde{w}$, and as Q otherwise. We write $H\langle\tilde{v}\rangle$ to denote a process defined via a definition $H(\tilde{x}) \stackrel{\text{def}}{=} P$, 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 the names of those 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. Finally, we define *structural congruence*, written $\equiv$, as the smallest congruence which is a commutative monoid with respect to the parallel operator.

3.1 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 2 contains the rules to model the synchronisation between sender and receivers. Table 3

Table 2 LTS - Synchronisation

$$(\text{Snd})\ \frac{\mathcal{D} := \{n : T(m,n) \geq \sigma\}}{m[\sigma!\langle\tilde{v}\rangle.P]_T \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_\sigma m[P]_T} \qquad (\text{Rcv})\ \frac{T(n,m) \geq \sigma \quad |\,\tilde{x}\,| = |\,\tilde{v}\,|}{n[\sigma?(\tilde{x}).P]_T \xrightarrow{m?\tilde{v}\triangleright n}_\sigma n[\{^{\tilde{v}}/_{\tilde{x}}\}P]_T}$$

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

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

$$(\text{Par})\ \frac{M \xrightarrow{\lambda}_\rho M' \quad \mathsf{sender}(\lambda) \notin \mathsf{nds}(N)}{M \mid N \xrightarrow{\lambda}_\rho M' \mid N}$$

contains the rules to model trust management, i.e. the actions of the trust manager components.

Our transitions are of the form $M \xrightarrow{\lambda}_\rho M'$, indicating that the network M can perform the action λ, at security level ρ, evolving into the network M'. By construction, in such a transition, 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 $\rho =$ `trust` then the transition models some aspects of trust management and involves all trusted nodes. The label λ ranges over the actions $m!\tilde{v}\triangleright\mathcal{D}$, $m?\tilde{v}\triangleright\mathcal{D}$, and τ. The action $m!\tilde{v}\triangleright\mathcal{D}$ models the transmission of message $\tilde{v}$, originating from node m, and addressed to the set of nodes in $\mathcal{D}$. The action $m?\tilde{v}\triangleright\mathcal{D}$ represents the reception of a message $\tilde{v}$, sent by m, and received by the nodes in $\mathcal{D}$. We sometimes write $m?\tilde{v}\triangleright n$ as an abbreviation for $m?\tilde{v}\triangleright\{n\}$. The action τ models silent actions, as usual. The function $\mathsf{sender}(\cdot)$ applied to an action returns the name of the sender, thus $\mathsf{sender}(m!\tilde{v}\triangleright\mathcal{D}) = \mathsf{sender}(m?\tilde{v}\triangleright\mathcal{D}) = m$, whereas $\mathsf{sender}(\tau) = \perp$.

Let us comment on the rules of Table 2. Rule (Snd) models a node m which broadcasts a message $\tilde{v}$ at security level σ; the set $\mathcal{D}$ contains the nodes at security level at least σ, according to the trust store of m. Rule (Rcv) models a node n receiving a message $\tilde{v}$, sent by node m, at security level σ. Node n receives the 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. If sender and receiver(s) trust each other there will be a synchronisation.[1] Rule (Sync) serves to synchronise the components of a network with a broadcast communication; the condition $\mathcal{D}' \subseteq \mathcal{D}$ ensures that only authorised recipients can receive the transmitted value. Rule (Par) is standard in process calculi. Notice that using rule (Par) we can model situations where potential receivers do not necessarily receive the message, either because they are not in the transmission range of the

[1] Here, we abstract on the actual behaviour of receivers as they verify the identity of the sender and discard unauthorised messages.

Table 3 LTS - Trust Management

$$(\text{Susp})\ \frac{T(m,n) > \texttt{bad} \quad \tilde{v} := n, \texttt{bad} \qquad T' := \mathcal{P}(T \cup \langle m, \tilde{v}\rangle) \quad \mathcal{D} := \{n : T(m,n) > \texttt{bad}\}}{m[P]_T \xrightarrow{m!\tilde{v} \triangleright \mathcal{D}}_{\texttt{trust}} m[P]_{T'}}$$

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

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

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

transmitter or simply because they loose the message. Rules (Sync), (RcvPar) and (Par) have their symmetric counterparts.

Example 1. Let us consider the network:

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

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 broadcasts message $\tilde{v}$ at security level σ, knowing that the nodes allowed to receive the message at that security level are n and l. However, node l does not trust m at security level σ. Thus, n is the only node that may receive the message. By an application of rules (Snd), (Rcv), (Par), and (Sync) we have:

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

Now, let us comment on the rules of Table 3 modelling trust management. Rule (Susp) 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. Notice that this transmission is not under the control of the code of m but it rather depends on the *reputation handling module*. Notice also that the transmission is addressed to all trusted nodes, that's why the transmission fires at security level `trust`. Rule (SndRcm) models *indirect trust* by sending 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 $T \cup \langle m, \tilde{v}\rangle$. Rule (Loss) models loss of trust

Table 4 LTS - Matching and recursion

$$\text{(Then)}\ \frac{n[P]_T \xrightarrow{\lambda}_\rho n[P']_{T'}}{n[[\tilde{v}=\tilde{v}]P,Q]_T \xrightarrow{\lambda}_\rho n[P']_{T'}} \qquad \text{(Else)}\ \frac{n[Q]_T \xrightarrow{\lambda}_\rho n[Q']_{T'} \quad \tilde{v_1} \neq \tilde{v_2}}{n[[\tilde{v_1}=\tilde{v_2}]P,Q]_T \xrightarrow{\lambda}_\rho n[Q']_{T'}}$$

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

information. This happens, for instance, when a node moves, changing its neighbourhood. In this case, assertions concerning old neighbours must be deleted as they cannot be directly verified. The consistency of the remaining assertions must be maintained by applying the security policy.

Example 2. Let us show how direct and indirect trust work. Let us consider the network:

$$M \overset{\text{def}}{=} k[P_k]_{T_k} \mid l[P_l]_{T_l} \mid m[P_m]_{T_m} \mid n[P_n]_{T_n}$$

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

$$M \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_{\texttt{trust}} k[P_k]_{T'_k} \mid l[P_l]_{T_l} \mid m[P_m]_{T'_m} \mid n[P_n]_{T'_n} \ .$$

Notice that since l does not trust m, only node n (but also the bad node k) will receive m's recommendation. Moreover the local knowledge of m and n will change, accordingly to the local policy. This is a case of direct trust for m, and indirect trust for n. The security level that n will assign to k will actually depend the local policy of n.

Finally, Table 4 contains the standard rules for matching and recursion.

4 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. Node mobility in ad hoc networks introduces new security issues related to user credential management, indirect trust establishment and mutual authentication between previously unknown and hence untrusted nodes. Thus, mobile ad hoc networks has turned to be a challenge for automated verification and analysis techniques. After the first works on model checking of (stationary) ad hoc networks [16], Nanz and Hankin [5] have proposed a process calculus where topology changes are abstracted into a fixed representation. This representation, called *network*

Table 5 LTS - Synchronisation with network restrictions

$$\text{(SndR)}\ \frac{\mathcal{D} := \{n : T(m,n) \geq \sigma\}}{m[\sigma!\langle\tilde{v}\rangle.P]_T \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_{\sigma,\emptyset} m[P]_T} \quad \text{(RcvR)}\ \frac{T(n,m)\geq\sigma \quad |\tilde{x}|=|\tilde{v}| \quad P':=\{^{\tilde{v}}/_{\tilde{x}}\}P}{n[\sigma?(\tilde{x}).P]_T \xrightarrow{m?\tilde{v}\triangleright n}_{\sigma,(n,m)} n[P']_T}$$

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

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

$$\text{(ParR)}\ \frac{M \xrightarrow{\lambda}_{\rho,C} M' \quad \mathsf{sender}(\lambda) \notin \mathsf{nds}(N)}{M \mid N \xrightarrow{\lambda}_{\rho,C} M' \mid N}$$

topology, is essentially a set of *connectivity graphs* denoting the possible connectivities within the nodes of the network.

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 add topology changes at semantics level, so that each state represents a set of valid topologies, and a network can be at any of those topologies at any time [9]. In Table 5 we rewrite the rules of Table 2 in the style of [9]. Rules are of the form $M \xrightarrow{\lambda}_{\rho,C} M'$, indicating that the network M can perform the action λ, at security level ρ, under the network restriction C, evolving into the network M'. Thus, a network restriction C keeps track of the connections which are necessary for the transition to fire. The rules in Table 3 can be rewritten in a similar manner, except for rule (Loss) in which the network restriction is empty i.e. $C = \emptyset$.

Example 3. Consider the same network given in the Example 1. Then by applying rules (SndR), (RcvR), (ParR), and (SyncR) we have

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

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

Notice that the rule (Loss) in Table 3 may indirectly affect future communications. In fact, if a trust information is lost then certain nodes may not be able of communicating anymore.

The reader may have noticed that the rules of Table 5 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 1.

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. □

5 Safety properties

In this section, we show how to guarantee in our setting that only authorised nodes receive sensible information. We define a notion of *safety up to a security level* to describe when a communication is safe up to a certain security level.

Definition 1 (Safety up to a security level). *A node m transmitting at level ρ may only synchronise with a node n receiving at level ρ or above, according to the local knowledge of m and n, respectively.*

Intuitively, Definition 1 says that a synchronisation at a certain security level ρ is safe if the involved parties trust each other at that security level.

The safety property is then preserved at run time.

Theorem 1 (Safety preservation). *Let $M \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_{\rho} M'$ with*

$$M \equiv m[P]_T \mid \textstyle\prod_i n_i[P_i]_{T_i} \text{ and } M' \equiv m[P']_{T'} \mid \textstyle\prod_i n_i[P'_i]_{T'_i} \ .$$

1. If $P'_i \neq P_i$, for some i, then $T(m,n_i) \geq \rho$ and $T_i(n_i,m) \geq \rho$.
2. If $T'_i \neq T_i$, for some i, then $T(m,n_i) \geq \rho$ and $T_i(n_i,m) \geq \rho$.

Proof By induction on the transition $M \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_{\rho} M'$. □

A consequence of Theorem 1, is that (trusted) nodes never synchronise with untrusted nodes. In this manner, bad nodes (recognised as such) are isolated from the rest of the network.

Corollary 1 (Safety despite compromise). *Let $M \xrightarrow{m!\tilde{v}\triangleright\mathcal{D}}_{\rho} M'$ such that*

$$M \equiv m[P]_T \mid \prod_i n_i[P_i]_{T_i} \text{ and } M' \equiv m[P']_{T'} \mid \prod_i n_i[P'_i]_{T'_i} \ .$$

If $T(m,n_i)=\mathtt{bad}$ or $T_i(n_i,m)=\mathtt{bad}$, for some i, then $P'_i=P_i$ and $T'_i=T_i$.

6 Behavioural Semantics

Our main behavioural equivalence is *σ-reduction barbed congruence*, a variant of Milner and Sangiorgi's (weak) barbed congruence [12] which takes into account 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 3.1 it is easy to see that a network may evolves either because there is a transmission at a certain security level or because a node looses some trust information. Thus, we can define the reduction relation $\rightarrowtriangle$ between networks using the following inference rules:

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

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

In our calculus, we have both transmission and reception of messages although only transmissions may 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.

Definition 2 (σ-Barb). *We write $M \downarrow_n^\sigma$ if $M \equiv m[\sigma!\langle\tilde{v}\rangle.P]_T \mid N$, 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, and that may reach the node n in the environment. In the sequel, we write $\mathcal{R}$ to denote binary relations over networks.

Definition 3 (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 4 (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, the definition of reduction closure is given in terms of weak reductions.

Definition 5 (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 σ-reduction barbed congruence.

Definition 6 (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.*

7 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 barbed congruent. Simpler proof techniques are based on labelled bisimilarities. In the sequel we define an appropriate notion of bisimulation. As a main result, we prove that our labelled bisimilarity is a proof-technique for our σ-reduction barbed congruence.

In general, a bisimulation describes how two terms (in our case networks) can mimic each other actions. First of all we have to distinguish between transmissions which may be observed and transmissions which may not be observed by the environment.

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

Rule (Shh) models transmissions that cannot be observed because none of the potential receivers are in the environment. Notice that security levels of τ-action are 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 by the nodes of the environment contained in $\widehat{\mathcal{D}}$. Notice that the rule (Obs) can only be applied at top-level of a derivation tree. In fact, we cannot use this rule together with rule (Par) of Table 2, because λ does not range on the new action.

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

Definition 7 (δ-Bisimilarity). *The* δ-bisimilarity*, written $\approx_{\delta}$, is the largest symmetric relation over networks such that whenever $M \approx_{\delta} N$ if $M \xrightarrow{\alpha}_{\rho} M'$, with $\rho \leq \delta$, then 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 [17]. Intuitively, two networks are δ-bisimilar if they cannot be distinguished by any observers that cannot perform actions at security level greater than δ.

Theorem 2 ($\approx_{\delta}$ 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

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

is a δ-bisimulation. □

Theorem 3 (Soundness). *Let M and N be two networks such that $M \approx_{\delta} N$. Then $M \cong_{\sigma} N$, for $\sigma \leq \delta$.*

Proof It is easy to verify that δ-bisimilarity is σ-barb preserving and reduction closed, by definition. Contextuality follows by Theorem 2. □

Remark 1. For the sake of analysis, we can define the δ-bisimilarity using the labelled transition system with network restrictions of Table 5. However, by Proposition 1 the resulting bisimilarity would not change.

8 Non-interference

The seminal idea of *non interference* [18] aims at assuring that "*variety in a secret input should not be conveyed to public output*". In a multilevel computer system [3] 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 in a CCS-like process calculus with high-level and low-level processes, according to the level of actions that can be performed [19]. To detect whether an incorrect information flow (i.e. from high-level to low-level) has occurred, 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 as process equivalence the notion of δ-bisimilarity previously defined. Formally, high-level behaviours can be arbitrarily changed without affecting low-level equivalences.

Definition 8 describes what high-level behaviour means in our setting. We recall that we assumed the presence of a trust manager component for each node to manage trust information. As a consequence, actions at security level `trust` do not depend on the syntax of the processes as they depend on the trust manager. These actions can fire at any step of the computation and cannot be predicted in advance.

Definition 8 (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 $\delta' = \texttt{trust}$ or $\delta' > \delta$. Moreover, $H' \in \mathcal{H}_\delta$.*

The non-interference result can be stated as follows.

Theorem 4 (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) $H, K \in \mathcal{H}_\delta$, (ii) $H \approx_{\texttt{trust}} K$, and (iii) $\mathsf{nds}(H) = \mathsf{nds}(K)$. Then, $M \mid H \approx_\delta N \mid K$.*

Proof We prove that the relation

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

is a δ-bisimulation. □

9 Related work

Formal methods have been successfully applied for the analysis of network security (see, for instance, [20,21,22,23,24]).

Komarova and Riguidel [25] have proposed a centralised trust-based access control mechanism for ubiquitous environments. The goal is to allow a service provider for the evaluation of the trustworthiness of each potential client. Crafa and Rossi [17] have introduced a notion of *controlled information release* for a typed version of the π-calculus extended with *declassified actions*. The controlled information release property scales to non interference when downgrading is not allowed. They provide various characterisations of *controlled release*, based on typed behavioural equivalence, parameterised on security levels, to model observers at a certain security level. Hennessy [26] has proposed a typed version of the asynchronous π-calculus in which I-O types are associated to security levels. Typed equivalences are then used to prove a non interference result.

As regards process calculi for wireless systems, Mezzetti and Sangiorgi [4] have proposed calculus to describe interferences in wireless systems. Nanz and Hankin [5] have introduced a calculus for mobile wireless networks for specification and security analysis of communication protocols. Merro [7] has proposed a behavioural theory for MANETs. Godskesen [8] has proposed a calculus for mobile ad hoc networks with a formalisation of an attack on the cryptographic routing protocol ARAN. Singh et al. [6] have proposed the ω-calculus for modelling the AODV routing protocol. Ghassemi et al. [9] have proposed a process algebra where topology changes are implicitly modelled in the semantics. Merro and Sibilio [27] have proposed a timed calculus for wireless systems focusing on the notion of communication collision. In trust models for ad hoc networks, the timing factor is important because more recent trust informations should have more influence on the trust establishment process. More generally, a notion of time would allow to record past behaviours. Finally, Godskesen and Nanz [10] have proposed a simple timed calculus for wireless systems to express a wide range of mobility models.

None of the calculi mentioned above deal with trust. Carbone et al. [28] 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. In [29] Martinelli uses a cryptographic variant of CCS to describe and analyse different access control policies.

References

1. Blaze, M., Feigenbaum, J., Lacy, J.: Decentralized Trust Management. In: Symposium on Security and Privacy, IEEE Computer Society (1996) 164–173
2. Grandison, T.W.A.: Trust Management for Internet Applications. PhD thesis, Department of Computing, University of London (2003)
3. Bell, D.E., LaPadula, L.J.: Secure Computer System: Unified Exposition and Multics Interpretation. Technical Report MTR-2997, MITRE Corporation (1975)
4. Mezzetti, N., Sangiorgi, D.: Towards a Calculus For Wireless Systems. Electronic Notes in Theoretical Computer Science **158** (2006) 331–353
5. Nanz, S., Hankin, C.: A Framework for Security Analysis of Mobile Wireless Networks. Theoretical Computer Science **367**(1-2) (2006) 203–227
6. Singh, A., Ramakrishnan, C.R., Smolka, S.A.: A Process Calculus for Mobile Ad Hoc Networks. In Lea, D., Zavattaro, G., eds.: COORDINATION 2008. Volume 5052 of LNCS., Springer (2008) 296–314
7. Merro, M.: An Observational Theory for Mobile Ad Hoc Networks (full paper). Information and Computation **207**(2) (2009) 194–208
8. Godskesen, J.: A Calculus for Mobile Ad Hoc Networks. In Murphy, A.L., Vitek, J., eds.: COORDINATION 2007. Volume 4467 of LNCS., Springer (2007) 132–150

9. Ghassemi, F., Fokkink, W., Movaghar, A.: Equational Reasoning on Ad Hoc Networks. In Arbab, F., Sirjani, M., eds.: FSEN 2009. Volume 5961 of LNCS., Springer (2010)
10. Godskesen, J.C., Nanz, S.: Mobility Models and Behavioural Equivalence for Wireless Networks. In Field, J., Vasconcelos, V.T., eds.: COORDINATION 2009. Volume 5521 of LNCS., Springer (2009) 106–122
11. Huang, D., Medhi, D.: A Secure Group Key Management Scheme for Hierarchical Mobile Ad Hoc Networks. Ad Hoc Networks **6**(4) (2008) 560–577
12. Milner, R., Sangiorgi, D.: Barbed Bisimulation. In Kuich, W., ed.: ICALP 1992. Volume 623 of LNCS., Springer (1992) 685–695
13. Milner, R.: Communication and Concurrency. Prentice Hall (1989)
14. Shehab, M., Bertino, E., Ghafoor, A.: Efficient Hierarchical Key Generation and Key Diffusion for Sensor Networks. In: SECON, IEEE Communications Society (2005) 76–84
15. Di Pietro, R., Mancini, L.V., Law, Y.W., Etalle, S., Havinga, P.J.M.: LKHW: A Directed Diffusion-Based Secure Multicast Scheme for Wireless Sensor Networks. In: ICPP Workshops 2003, IEEE Computer Society (2003) 397–413
16. Bhargavan, K., Obradovic, D., Gunter, C.A.: Formal Verification of Standards for Distance Vector Routing Protocols. Journal of the ACM **49**(4) (2002) 538–576
17. Crafa, S., Rossi, S.: Controlling Information Release in the π-calculus. Information and Computation **205**(8) (2007) 1235–1273
18. Goguen, J.A., Meseguer, J.: Security Policies and Security Models. In: IEEE Symposium on Security and Privacy. (1982) 11–20
19. Focardi, R., Gorrieri, R.: A Classification of Security Properties for Process Algebras. Journal of Computer Security **3**(1) (1995) 5–33
20. Reitman, R., Andrews, G.: An Axiomatic Approach to Information Flow in Programs. ACM Transactions on Programming Languages and Systems **2**(1) (1980) 56–76
21. Smith, G., Volpano, D.: Secure Information Flow in a Multi-threaded Imperative Language. In: Proc. 25th POPL, ACM Press (1998) 355–364
22. Heintz, N., Riecke, J.G.: The SLam Calculus: Programming with Secrecy and Integrity. In: Proc. 25th POPL, ACM Press (1998) 365–377
23. Bodei, C., Degano, P., Nielson, F., Nielson, H.R.: Static Analysis for the pi-Calculus with Applications to Security. Information and Computation **168**(1) (2001) 68–92
24. Boudol, G., Castellani, I.: Noninterference for Concurrent Programs and Thread Systems. Theoretical Computer Science **281**(1-2) (2002) 109–130
25. Komarova, M., Riguidel, M.: Adjustable Trust Model for Access Control. In Rong, C., Jaatun, M.G., Sandnes, F.E., Yang, L.T., Ma, J., eds.: ATC 2008. Volume 5060 of LNCS., Springer (2008) 429–443
26. Hennessy, M.: The Security pi-calculus and Non-Interference. Journal of Logic and Algebraic Programming **63**(1) (2005) 3–34
27. Merro, M., Sibilio, E.: A Timed Calculus for Wireless Systems. In Arbab, F., Sirjani, M., eds.: FSEN 2009. Volume 5961 of LNCS., Springer (2010)
28. Carbone, M., Nielsen, M., Sassone, V.: A Calculus for Trust Management. In Lodaya, K., Mahajan, M., eds.: FSTTCS 2004. Volume 3328 of LNCS., Springer (2004) 161–173
29. Martinelli, F.: Towards an Integrated Formal Analysis for Security and Trust. In Steffen, M., Zavattaro, G., eds.: FMOODS 2005. Volume 3535 of LNCS., Springer (2005) 115–130