Secure Broadcast Ambients

Elsa L. Gunter and Ayesha Yasmeen

Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL, USA
egunter@cs.uiuc.edu, yasmeen@uiuc.edu

Abstract. Broadcast mechanism is prevalent in many forms of electronic networks. Modeling broadcast protocols succinctly and reasoning about how secure these protocols are is gaining importance as society increasingly comes to depend on a wide variety of electronic communications. In this work we present a modified ambient calculus where the nature of communication is broadcast within domains. We allow reconfigurable configurations of communication domains, access restrictions to domains and the capability of modeling cryptographic communication protocols in broadcast scenarios.

Key words: ambient calculi, process calculi, broadcast, bisimulations, congruence, security

1 Introduction

In a world of increasing dependence on electronic communications between reconfigurable and mobile devices, there is a clear need for accurate formal systems to model these devices and their communications to facilitate guaranteeing such properties as functionality and security (especially privacy and integrity). Ambients [9], including boxed ambients [5,2], are formalisms that have been developed to model such mobile devices and their communication. Ambients have an associated topology that confines their movement and their communication options. This topology has traditionally been restricted to tree structures, and communication and movement have been restricted to adjacent ambients. The tree structure implies that an ambient can only be "in" one other ambient at a given time. This poses problems for modeling aspects of networks, such as routers. A router is most naturally modeled as being "in" multiple domains at once. Similarly, a laptop with an ethernet connection, a bluetooth connection and a dialup-modem connection, can be thought of as being "in" three different domains at once. The restriction of the topology to tree structures prevents modeling these devices that way. In this work, we loosen this constraint to allow the topology to be that of a dynamically reconfigurable directed acyclic graph, thus allowing one ambient to be in more than one other ambient at a given time, or possibly none at all.

In theoretical models of systems, and ambients in particular, communication is often modeled using point-to-point channels. Depending on the particular calculus, many processes may have access to a given channel, but each communication will have a unique recipient. Within such frameworks, modeling broadcast and multicast communications must be done using multiple unicasts. Let us consider a π-calculus [15] implementation of a message being broadcasted over a channel. In order to ensure that this message can be received by every possible recipient, the message sending action has to be replicated.

$$\text{Server} = !(\overline{c}\langle M\rangle.\mathsf{nil});\ \text{Client}_\text{i} = c(x_i).P_i;\ \text{Network} = \text{Server} \mid \text{Client}_1 \mid \text{Client}_2 \mid \ldots$$

This implementation of a broadcast communication using π-calculus is incorrect in the sense that it does not enforce receiving the message M simultaneously by every recipient. One recipient may receive it much later than another recipient. An adversary can choose to come in any time after the message has been sent to sniff the message. As a result, it allows for possible behaviors in the model that can not arise in practice.

Alternatives to this have been devised using broadcast communications. These include broadcast communication limited to a specified domain. However, in frameworks with broadcast within a domain, the domains are relatively static with only at most code moving among them. Because the code alone is mobile, it carries no identity with it, which limits the ability to concisely and accurately model the organization of the domains, and model the restriction of access to the domains by other domains. In our work, we have broadcast communication within ambients. Messages announced to the ambient are heard by the ambient and all ambients directly within it. Ambients may restrict access to themselves, and hence to the privilege of the communication within them, based on the identity of potential entrants, without requiring that their names be hidden.

Let us now consider a wifi network existing in a home. *The wireless network usually continually broadcasts its identity (SSID) to the rest of the world on its wavelength. Anyone capable of capturing wireless communication on that wavelength is capable of listening to this name. The wireless network gets to announce its name even when no one is listening for it. If the network does not secure itself, then typically anyone who knows its name, can enter it. Once a rogue computer enters this network, it can send and receive packets of its own and listen to all the communication occurring inside the network, as all packets are essentially broadcasted in a wireless network. In order to be able to prevent indiscriminate access to the home network, the network should not be allowing everyone to gain access to itself. The simplest method of access control is* M*achine* A*uthentication* C*ode (MAC) filtering, where a computer is allowed access to a network only if it has a network card with a pre-approved MAC. This can allow the home network to deny access to an unauthorized computer from the house next door. Also, the router of the wifi network connects the computers connected to the home wireless network with the outside world. This router is capable of directly communicating to the outside internet network and the home computers. Hence, virtually it is present in multiple communication domains simultaneously. Any computer such as a laptop belonging to a household member should be allowed access and also it should be allowed to leave the network whenever the user of the laptop simply* *shuts it off. Finally, each computer on the home network should be able to hide from all other computers whatever it is communicating with the rest of the world.*

This scenario leads to the necessity of a calculus which is able to model (i) distinct regions of computation; (ii) broadcast communication within the regions of communication, where all listeners get to listen to the communication simultaneously, irrespective of whether the speaker knows of the listeners' identities or not, and also irrespective to whether there is anyone actually listening for the communication or not; (iii) allowing indiscriminate access to a region; (iv) allowing access control via selective permission to enter a region; (v) allowing elegant representation of regions being inside multiple domains, for example, as said before, a laptop computer connected to the home wifi network, may also be connected to a workplace network via dialup connection; (vi) allowing the hierarchy of the regions to be a directed acyclic graph with the capability of modeling that a region can be a standalone region, that is, it may at some times be adjacent to no other region, (vii) allowing regions to leave other regions at their free will, without requiring any permission from the regions that it is trying to leave; (viii) allow regions to engage in cryptographic communication protocols to be able to secure their data. None of the existing process calculi, especially the ambient calculi and the broadcast calculi, are capable of modeling all these aspects of the home wireless network. We list what aspects the other existing related process calculi can encode in the related works section in Section 6. Our directed acyclic hierarchy of ambients and broadcast communication is however a novel combination.

2 Secure Broadcast Ambients

2.1 Syntax of Secure Broadcast Ambients

In order to define the syntax of Secure Broadcast Ambients we use the following categories of identifiers: ambient names: $n, m \in \mathit{Amb}$, message variables: $x \in \mathit{MessVar}$ and key variables: $k \in \mathit{Keys}$. The syntax of Secure Broadcast Ambients is presented in Table 1. *Messages*, *Processes* and *Systems* are the main syntactic categories. We only mention interesting aspects of the syntax. Apart from the variables, a message can be a pair of messages or an encrypted message. Each ambient contains a process. Processes are built from the usual constructs, such as replication. parallel combination, and prefixing with actions.

An ambient n can indicate its intention to move into another ambient by a $\mathsf{in}\ m$ prefix. However, this movement can only be successful if a corresponding permission is there to allow this move. The corresponding permission can be either $\overline{\mathsf{in}\ n}$ allowing specifically n to enter, or $\overline{\mathsf{in}\ \_}$ indicating permission for any ambient. As we shall see in Section 3, the only further restriction placed on entrance is that an ambient is not allowed to enter a descendant of itself. This interpretation of ambient movement leads to a directed acyclic graph structure for the hierarchy of ambients. In Secure Broadcast Ambients, an ambient can be in multiple ambients at the same time. An ambient may even fail to be in any ambient, for example, a laptop that has been turned off. An ambient n exits from

Messages:

$M, N ::=$	x	message ident
$\mid$	m	ambient name
$\mid$	(M, N)	pairing
$\mid$	k	key
$\mid$	$\{M\}_k$	encrypted message

Format:

$F ::=$	m	ambient name
$\mid$	$\{x\}_k$	decryption
$\mid$	(x, y)	pairs

Prefixes:

$\pi ::=$	$\overline{\mathsf{in}\, \mu}$	allow enter
$\mid$	$\mathsf{in}\, m$	enter
$\mid$	$\mathsf{out}\, m$	exit
$\mid$	$(x)^m$	input
$\mid$	νk	new key
$\mid$	$\langle M \rangle^m$	output

Processes:

$P, Q ::=$	nil	nil process
$\mid$	$P \mid Q$	composition
$\mid$	$!P$	replication
$\mid$	$\pi.P$	prefixing
$\mid$	$\mathsf{cond}\ M\ \mathsf{is}\ N\ \mathsf{in}\ P$	data comparison
$\mid$	$\mathsf{case}\ M\ \mathsf{of}\ F : P$	case analysis

Systems:

$\mathcal{S} ::=$	$\mathsf{nilsystem}$	empty system
$\mid$	$m[P]$	ambient
$\mid$	$(x)^m(\mathcal{S})$	broadcast receive
$\mid$	$\nu k.(\mathcal{S})$	key restrict
$\mid$	$\nu m :: \mathcal{L}.(\mathcal{S})$	ambient restrict
$\mid$	$\mathcal{S}_1 \,\|\, \mathcal{S}_2$	parallel

Ambient Pattern:

$\mu ::=$	$\_$	any ambient
$\mid$	m	ambient name

Table 1. Syntax of Broadcast Ambients

the ambient m by the $\mathsf{out}\, m$ action without requiring any permission from any other ambient, and without affecting the relationship of n to any other ambient.

The processes have sending and receiving prefixes for communication. We do not have channels to be used for inter-ambient communication. In our calculus the name of a parent ambient acts as the broadcast channel for both itself and its children. This way, any ambient can listen to any conversation that is going on between any of its parents and their children. Henceforth channels are synonymous with ambients.

A process can also perform matching, or case analysis on a message much in the manner of [1]. The case analysis patterns are given by the formats. In this work we only consider symmetric encryption and so the decryption key is the same as the encryption key for every encrypted message.

We introduce systems, which are collections of ambients. A system can be an empty system, an ambient, a system waiting to receive a message or multiple systems in parallel. Systems can create a new key or a new ambient name with a given parent list by restriction.

The free variables, free ambient names and free keys of messages, processes and systems will be denoted by the function fv(), fn() and fk() respectively and is defined the usual way. We use fi() for their union. Ambient and key restriction, message input and case analysis are the binding constructs.

We now encode the components in the wireless home network as described in the introduction using our syntax:

$$\mathsf{Wifi}[!\overline{\mathsf{in\ Laptop}}.\mathsf{nil}] \,\|\, \mathsf{Laptop}[\mathsf{in\ Wifi}.(x)^{\mathsf{Wifi}}.P] \,\|$$
$$\mathsf{Router}[!((x)^{\mathsf{Wifi}}.\langle x \rangle^{\mathsf{ISP}}.\mathsf{nil}) \mid !((x)^{\mathsf{ISP}}.\langle x \rangle^{\mathsf{Wifi}}.\mathsf{nil})]$$

The router, Router captures all outgoing packets in the home network, Wifi, and forwards them to the ISP (its code is omitted here) and vice versa. Currently, the wireless home network allows only the laptop, Laptop to enter the network. The laptop first tries to enter the network and then as an example code we make it listen on the network.

2.2 Structural equivalence of Broadcast Ambients

Structural equivalence is defined for each of processes and systems. It is the smallest equivalence relation containing the rules in Table 2, closed under alpha equivalence (where ν, message receipt and case analysis are the binding constructs), and the associativity and commutativity of parallel composition of each of processes and systems, with nil and nilsystem as the respective identities. We omit some rules that are very similar to rules for ambients and process calculi in general. Rules for restriction, permutation and pattern matching are very similar to rules provided in [1]. Decryption is also performed by pattern matching in an abstract fashion as in [1]. We use bi() to indicate the identifier that is being bound in a restriction. We sometimes use u as an abbreviation for $\nu m :: \mathcal{L}$ or νk.

The rule (StrSysPar) indicates that one ambient can have multiple scattered pieces with the same name. However, there will be only one ambient with a particular name. The last four rules in Table 2 are the rules that enable broadcast communication. The rule (StrBrdcstListen) allows an ambient to lift the receive action on a particular broadcast channel of a process within it up to the level of the ambient system. The rule (StrCombListen) can then be used to combine the listening systems. It is the principal rule that is used to model broadcast systems. This rule combines multiple ambients listening on the same channel so that later on only one transition is needed to send a message simultaneously to all the ambients listening on this ambient. In addition to being able to send a message to multiple parties simultaneously, in a broadcast scenario, the broadcaster can send out a message even if no one is listening on the broadcast channel being used. The rule (StrNoListen) enables us to model this scenario.

3 Operational Semantics

The operational semantics of Secure Broadcast Ambients relies heavily on the topological layout of the ambients under consideration. We now introduce *configurations*, which keep track of the topology of the ambients in a system.

3.1 Configuration

We first introduce the concept of a parent list, Π, for the ambients. The parent list keeps track of the ancestors for each ambient. A parent list is a finite ordered list of pairs of ambients and ambient lists, written as $[m_0 :: \mathcal{L}_0, m_1 ::$

(StrRepPar)	$!P \equiv P \mid !P$	(StrNil Sys)	$m[\mathsf{nil}] \equiv \mathsf{nilsystem}$
(StrProcSys)	$P \equiv Q \Rightarrow m[P] \equiv m[Q]$	(StrSysPar)	$m[P_1 \mid P_2] \equiv m[P_1] \parallel m[P_2]$
(StrSysKey)	$m[\nu k.P] \equiv \nu k.m[P]$	(StrKey)	$\nu u.\nu k.S \equiv \nu k.\nu u.S$
(StrPKey)	$\nu k_1.\nu k_2.P \equiv \nu k_2.\nu k_1.P$	(StrResNil)	$\nu u.\mathsf{nilsystem} \equiv \mathsf{nilsystem}$
(StrRes Par)	$\nu u.(S_1 \parallel S_2) \equiv S_1 \parallel \nu u.S_2,\ \text{if } \mathrm{bi}(u) \notin \mathrm{fi}(S_1)$		
(StrRes Rec)	$\nu u.(x)^n.S \equiv (x)^n.\nu u.S \text{ if } n \neq \mathrm{bi}(u)$		
(StrRes Res)	$\nu n :: \mathcal{L}_1.\nu m :: \mathcal{L}_2.S \equiv \nu m :: \mathcal{L}_2.\nu n :: \mathcal{L}_1.S,$ if $\mathrm{n} \neq \mathrm{m}$ and $\mathrm{m} \notin \mathcal{L}_1$ and $\mathrm{n} \notin \mathcal{L}_2$		
(StrBrdcstListen)	$\nu m :: \mathcal{L}.\nu n_1 :: \mathcal{L}'_1 \ldots \nu n_k :: \mathcal{L}'_k.m[(x)^n.P] \equiv \nu m :: \mathcal{L}.\nu n_1 :: \mathcal{L}'_1 \ldots \nu n_k :: \mathcal{L}'_k.((x)^n(m[P])),\ \text{if } m = n \vee (m \notin \{n_1, \ldots, n_k\} \wedge n \in \mathcal{L})$		
(StrNoListen)	$(x)^n(\mathsf{nilsystem}) \equiv \mathsf{nilsystem}$		
(StrNeutral)	$(x)^n.(S_1 \parallel S_2) \equiv S_1 \parallel ((x)^n.S_2) \ \text{ if } x \notin fv(S_1)$		
(StrCombListen)	$(x)^n.S_1 \parallel (y)^n.S_2 \equiv (z)^n.(S_1[z/x] \parallel S_2[z/y]),\ \ z \text{ fresh in } S_1, S_2$		

Table 2. Structural Equivalence

$\mathcal{L}_1, \ldots, m_n :: \mathcal{L}_n]$ where each $\mathcal{L}_i$ is the list of parent ambients of ambient m_i and for each i and j such that $0 \leq i, j \leq n$ we have that $m_i \neq m_j$ and if $i \leq j \leq n$ then $m_i \notin \mathcal{L}_j$.

We allow our ambients to be decomposed into several parallel pieces as the structural equivalence rules suggest. Hence, we need to consolidate the hierarchy information for each ambient, so that all pieces have the same view of their ancestry, and so that no cycle can arise in the hierarchical graph of the ambients. Hence we introduce the idea of a configuration, Δ. Configurations represent the partial ordering of the ambients by the parent-child relation, and are a linearization of that order with parents always occurring to the left.

Definition 1. *A configuration is a parent list $\Pi = [m_0 :: \mathcal{L}_0, m_1 :: \mathcal{L}_1, \ldots, m_n :: \mathcal{L}_n]$ with the additional constraint that $\forall i.0 \leq i \leq n.p \in \mathcal{L}_i \Rightarrow \exists j.0 \leq j < i.p = m_j$. The set of all first components in a configuration Δ will be denoted by $\mathbf{dom}(\Delta) = \{m \mid \exists \mathcal{L}.(m, \mathcal{L}) \in \Delta\}$. We define a function # which concatenates an ambient-parent list pair to a configuration as $[m_0 :: \mathcal{L}_0, m_1 :: \mathcal{L}_1, \ldots, m_n :: \mathcal{L}_n]\#(m :: \mathcal{L}) = [m_0 :: \mathcal{L}_0, m_1 :: \mathcal{L}_1; \ldots, m_n :: \mathcal{L}_n, m :: \mathcal{L}]$ if $m \notin \mathbf{dom}(\Delta)$ and $\mathcal{L} \subseteq \mathbf{dom}(\Delta)$, and otherwise it is undefined. We define $\Delta\#\Pi$ as the iterative folding of # over Π*

We shall refer to the pair of a configuration and a system, $\Delta \triangleright S$, as a *formation* where $\mathsf{ambsinsys}(S) \subseteq \mathbf{dom}(\Delta)$. $\mathsf{ambsinsys}(S)$ is a function that returns all ambients of the form $m[P]$ syntactically appearing in the system S (definition omitted in this work). The set of all formations will be denoted by $\mathcal{F}$. Notice that $\Delta\#(m :: \mathcal{L})$ is a configuration if it is defined. Also notice, $\Delta = [m_0 :: \mathcal{L}_0, m_1 :: \mathcal{L}_1, \ldots, m_n :: \mathcal{L}_n] = []\#m_0 :: \mathcal{L}_0\#m_1 :: \mathcal{L}_1\# \ldots \#m_n :: \mathcal{L}_n$. When looking an ambient up in a configuration, we will start at the right, moving to the left. Because the parents of any ambient are always to the left of this ambient in the configuration, we are ensured that a configuration always encodes a directed acyclic graph.

Barb In:

$$\frac{m \in \mathrm{dom}(\Delta) \wedge m \notin \mathcal{L}_n}{\mathit{\Delta} \;\#\; n :: \mathcal{L}_n \;\#\; \mathit{\Pi} \rhd n[\mathsf{in}\; m\, .P] \downarrow \mathsf{in}(n, m)}$$

Barb Co-In:

$$\frac{\mu = \_ \vee (\mathsf{match}(n, \mu) = \mathsf{false} \wedge \; \mu \notin \mathrm{dom}(\Delta) \cup \mathcal{L}_n)}{\mathit{\Delta} \;\#\; n :: \mathcal{L}_n \;\#\; \mathit{\Pi} \rhd n[\overline{\mathsf{in}\; \mu}\, .P] \downarrow \overline{\mathsf{in}}(n, \mu)}$$

Barb Send:

$$\frac{m \in (\{n\} \cup \mathit{\Delta}(n))}{\mathit{\Delta} \rhd n[\langle M \rangle^m .P] \downarrow \mathsf{send}\; m}$$

FormationStructBarb:

$$\frac{\mathit{\Delta}_1 \rhd S_1 \equiv \mathit{\Delta}_2 \rhd S_2 \quad \mathit{\Delta}_2 \rhd S_2 \downarrow \xi}{\mathit{\Delta}_1 \rhd S_1 \downarrow \xi}$$

SysParBarb:

$$\frac{\mathit{\Delta} \rhd S_1 \downarrow \xi}{\mathit{\Delta} \rhd (S_1 \| S_2) \downarrow \xi}$$

NewKeyBarb:

$$\frac{\mathit{\Delta} \rhd S \downarrow \xi}{\mathit{\Delta} \rhd \nu k.S \downarrow \xi}$$

NewAmbBarb:

$$\frac{\mathit{\Delta} \;\#\; n :: \mathcal{L} \rhd S \downarrow \xi \quad n \notin fn(\xi)}{\mathit{\Delta} \rhd \nu(n :: \mathcal{L}).S \downarrow \xi}$$

Table 3. Barbs

Definition 2. *We define a permutation relation* perm *as the reflexive, symmetric, transitive closure of the following rule: if $m \notin \mathcal{L}_n$, $n \notin \mathcal{L}_m$, $m \neq n$ then perm$(\mathit{\Delta} \;\#\; m :: \mathcal{L}_m \;\#\; n :: \mathcal{L}_n \;\#\; \mathit{\Pi}, \mathit{\Delta} \;\#\; n :: \mathcal{L}_n \;\#\; m :: \mathcal{L}_m \;\#\; \mathit{\Pi})$.*

This permutation relation creates equivalence classes of configurations that have the same topological structure that is induced by the hierarchical relationship of the ambients in the domain of the configurations. Let us now extend the structural equivalence of systems, $\equiv$, to structural equivalence of formations, where two formations $\mathit{\Delta}_1 \rhd S_1$ and $\mathit{\Delta}_2 \rhd S_2$ are structurally equivalent if perm$(\mathit{\Delta}_1, \mathit{\Delta}_2)$ and $S_1 \equiv S_2$.

3.2 Barbs

Our aim is to provide co-inductive relations that relate two formations that are externally indistinguishable. However, since we are considering secrecy of confidential information contained in systems, we want to reason about systems whose actions are indistinguishable even while sending and receiving different secrets. In that regard, we now define a predicate traditionally referred to as *barbs* that describes the actions a system can be observed to take. These actions will not indicate what messages (if any) are involved in the actions. We define the set of barbs exhibited by a system using the rules in Table 3, where we denote S exhibits a barb ξ in the configuration $\mathit{\Delta}$ by $\mathit{\Delta} \rhd S \downarrow \xi$. We define the barbs by descending through the syntax of the systems. Systems are mainly composed of ambients in parallel. Each ambient contains a process and hence we consider all the visible actions such processes can take. The empty system does not exhibit any barb. For the process inside an ambient only movement and communication actions are observable; other process constructs like pattern matching are not observable. The barb $\mathsf{send}\; m$ indicates that an ambient wants to send some message over the channel m. However receiving a message is not a barb in our system. The reason is that in a broadcast system a broadcaster or sender of a message can send a message, but whether the message was actually heard by anyone is not observable. An ambient may even send a message in a channel to which no one is listening. Similarly, entering, but not exiting, an ambient is an observable action as exiting does not require any form of permission from the ambients involved. The barb $\mathsf{in}\ (n, m)$ indicates that the ambient n is trying to enter ambient m. Similarly, the barb $\overline{\mathsf{in}}(n, \mu)$ indicates that the ambient n is allowing some ambient to enter it. The *match* operator takes an ambient name and a μ and determines whether the μ in the provided permission matches the name of the ambient which is trying to enter, as match(n,μ) $= (\mu = \_ \vee \mu = n)$.

3.3 Labeled Transition Semantics

Using formations, we introduce the labeled transition relation $\xrightarrow{\mathcal{L}} \subseteq \mathcal{F} \times \mathcal{L} \times \mathcal{F}$ for Secure Broadcast Ambients, where the transitions take one formation to another with the label $\mathcal{L}$ where $\mathcal{L}$ can either be a barb or the silent unobservable action τ. The transition semantics is given in Table 4. We now describe some interesting transition rules. In the Broadcast Message rule, we have a number of recipients waiting to receive a message being sent to an ambient, n. When an ambient sends a message to n, the communication completes. However, the communication should only occur if the configuration implies that the sender is either n itself or a child of n and the recipients are all children of n or n itself. Hence, we impose some side conditions for this rule. The first condition ensures that all ambients mentioned are described by the configuration Δ. The second condition ensures that n is either the sender or the sender is a child of n. It also ensures that the recipients are either children of n or n itself. After the communication, the sender proceeds with the rest of its code and a message variable substitution occurs for the recipient system. We now consider the ambient entry rule. As mentioned earlier an ambient needs specific permission to be able to enter another ambient. If there is a matching permission then an enter action is performed. In the rule Enter, n wants to enter ambient m. Ambient m has a permission prefix which has to either allow any ambient access to enter m or specifically allow ambient n to enter m. We also impose restrictions so that an ambient cannot enter an ambient it is already in. The other condition for this rule is to ensure that adding m as a parent of n will not introduce a cycle in the ancestry of n. After gaining entry into the ambient m, the parent list of n is updated. For the exit action in rule Exit, we check that the ambient requesting to exit from an ambient is actually a child of that ambient. The unlabeled transition system, $\rightarrow \subseteq \mathcal{F} \times \mathcal{F}$ is obtained by simply removing the labels from the labeled transitions.

3.4 Testing Equivalence

We now define the concept of *testing* a system in the manner of [1]. Intuitively it represents all tests an external system can make on any context in order to gather information about it. A formation $\Delta \triangleright T$ is a *context* for a formation $\Delta' \triangleright S$ if $\Delta' \circ \Delta \triangleright (S \| T)$ is a formation. $\circ$ is a function which composes two configurations

Broadcast Message:

$$\frac{m_j \in \mathrm{dom}(\Delta), j = 0, \ldots, k \ \wedge \ n \in \bigcap_{j=0}^{k} (\{m_j\} \cup \Delta(m_j))}{\begin{array}{l}\Delta \triangleright (m_0[\langle M \rangle^n.P] \| \, (x)^n (m_1[P_1] \| \ldots \| \, m_k[P_k])) \xrightarrow{\mathsf{send}\ m_0} \\ \Delta \triangleright m_0[P] \| \, (m_1[P_1] \| \ldots \| \, m_k[P_k])[M/x]\end{array}}$$

Enter:

$$\frac{\mathrm{match}(n, \mu) \ \wedge \ m \notin \mathcal{L}_n \wedge \ m \in \mathrm{dom}(\Delta)}{\begin{array}{c}\Delta \ \# \ n :: \mathcal{L}_n \ \# \ \Pi \triangleright n[\mathsf{in}\, m\,.P] \| \, m[\overline{\mathsf{in}\, \mu}\,.Q] \\ \xrightarrow{\mathsf{in}(n,m)} \Delta \ \# \ n :: \mathcal{L}_n \cup \{m\} \ \# \ \Pi \triangleright n[P] \| \, m[Q]\end{array}}$$

Exit:

$$\frac{m \in \mathcal{L}_n}{\begin{array}{c}\Delta \ \# \ n :: \mathcal{L}_n \ \# \ \Pi \triangleright n[\mathsf{out}\, m\,.P] \\ \xrightarrow{\tau} \Delta \ \# n :: \mathcal{L}_n \setminus \{m\} \ \# \ \Pi \triangleright n[P]\end{array}}$$

FormationEquiv:

$$\frac{\begin{array}{c}\Delta_1 \triangleright S_1 \equiv \Delta_1' \triangleright S_1' \\ \Delta_1' \triangleright S_1' \xrightarrow{\xi} \Delta_2' \triangleright S_2' \\ \Delta_2' \triangleright S_2' \equiv \Delta_2 \triangleright S_2\end{array}}{\Delta_1 \triangleright S_1 \xrightarrow{\xi} \Delta_2 \triangleright S_2}$$

AmbRestrict:

$$\frac{\begin{array}{c}\Delta \ \# \ n :: \mathcal{L} \triangleright S_1 \xrightarrow{\xi} \\ \Delta' \ \# \ n :: \mathcal{L}' \triangleright S_2\end{array}}{\begin{array}{c}\Delta \triangleright \nu n :: \mathcal{L}.S_1 \xrightarrow{\xi} \\ \Delta' \triangleright \nu n :: \mathcal{L}'.S_2\end{array}}$$

Parallel:

$$\frac{\begin{array}{c}\Delta \triangleright S_1 \xrightarrow{\xi} \\ \Delta' \triangleright S_1'\end{array}}{\begin{array}{c}\Delta \triangleright (S_1 \| \, S_2) \xrightarrow{\xi} \\ \Delta' \triangleright (S_1' \| \, S_2)\end{array}}$$

Table 4. Labeled Transition System

and returns another configuration. Its formal definition is omitted here. First we define barb convergence in the manner of [1]. The predicate $\Delta \triangleright S \Downarrow \xi$ is true when S is a system which can exhibit the barb ξ after zero or more transitions under the configuration Δ. The rules for barb convergence are as follows:

Barb:

$$\frac{\Delta \triangleright S \downarrow \xi}{\Delta \triangleright S \Downarrow \xi}$$

Reduct:

$$\frac{\Delta \triangleright S \rightarrow \Delta' \triangleright S' \quad \Delta' \triangleright S' \Downarrow \xi}{\Delta \triangleright S \Downarrow \xi}$$

Definition 3. *A test of $\Delta \triangleright S$ is a pair containing a formation $\Delta' \triangleright T$ and a barb ξ. $\Delta \triangleright S$ passes the test $(\Delta' \triangleright T, \xi)$ if and only if $(\Delta \circ \Delta') \triangleright (S \| \, T) \Downarrow \xi$.*

Definition 4. *A testing preorder, $\sqsubseteq$ for two formations $\Delta_1 \triangleright S_1$ and $\Delta_2 \triangleright S_2$ is defined as follows: $\Delta_1 \triangleright S_1 \sqsubseteq \Delta_2 \triangleright S_2$ if for any test $(\Delta \triangleright T, \xi)$, we have that if $\Delta_1 \triangleright S_1$ passes the test $(\Delta \triangleright T, \xi)$ then $\Delta_2 \triangleright S_2$ passes the test $(\Delta \triangleright T, \xi)$.*

Definition 5. *Two formations $\Delta_1 \triangleright S_1$ and $\Delta_2 \triangleright S_2$ are testing equivalent, denoted by $\Delta_1 \triangleright S_1 \simeq \Delta_2 \triangleright S_2$, if $\Delta_1 \triangleright S_1 \sqsubseteq \Delta_2 \triangleright S_2$ and $\Delta_2 \triangleright S_2 \sqsubseteq \Delta_1 \triangleright S_1$.*

3.5 Barbed Equivalence and Barbed Congruence

As mentioned in [1], testing equivalence, though elegant in concept, is hard to deal with. Hence we need another co-inductive relation that is easier to deal with to reason about systems being equivalent as to externally observable actions. Such a relation will be useful if it implies testing equivalence thereby removing the need to come up with all possible tests to determine testing equivalence. Let us define barbed simulation as a binary relation $\mathcal{R} \subseteq \mathcal{F} \times \mathcal{F}$ such that for two formations F_1 and F_2, $F_1 \ \mathcal{R} \ F_2$ implies that,

– if $F_1 \downarrow \xi$ then $F_2 \downarrow \xi$
– if $F_1 \rightarrow F_1'$ then there exists F_2' such that $F_2 \rightarrow F_2'$ where $F_1' \, \mathcal{R} \ F_2'$

A *barbed bisimulation* is a relation $\mathcal{R}$ such that both $\mathcal{R}$ and $\mathcal{R}^{-1}$ are barbed simulations. *Barbed equivalence*, written $\dot{\sim}$, is the greatest barbed bisimulation.

Definition 6. *Two formations $\Delta_1 \triangleright S_1$ and $\Delta_2 \triangleright S_2$ are barbed congruent, denoted by $\Delta_1 \triangleright S_1 \sim \Delta_2 \triangleright S_2$, if for any context $\Delta \triangleright T$, we have that $(\Delta_1 \circ \Delta) \triangleright (S_1 \| T) \dot{\sim} (\Delta_2 \circ \Delta) \triangleright (S_2 \| T)$*

Proposition 1. – *Barbed congruence is reflexive, transitive and symmetric.*
– *Barbed congruence is a congruence on closed systems.*
– *Structural equivalence implies barbed congruence.*
– *Barbed congruence implies testing equivalence.*

4 Secrecy

Secrecy of confidential information is a big issue in computer networks. The confidential information can be login information for an online banking facility, medical information of a patient or grades of a student in a university online grade processing system. The importance of confidentiality of these type of information is tremendous. In this section we focus on whether, during the execution of a communication protocol, any message from a set of (secret) messages is ever disclosed to ambients who are not trusted or authorized to receive that message. We will use behavioral congruence to characterize processes with nondisclosure assurance.

In real life scenarios, it is typical to have that some of the agents under consideration are trusted. For example, in a WPA-enabled Wi-Fi network the RADIUS authentication server is considered to be trusted; in an online banking scenario, any authorized client and the server at the bank are considered trusted and the login information is assumed to be known to both these parties. Hence we wish to determine whether a system will ever reveal a set of confidential information to distrusted agents in any possible context. In order to do that we will introduce the concept of safe contexts. The behavioral congruence that we have introduced before considers all possible contexts. But in order to determine whether a system can keep some confidential information secret, we have to impose the restriction that the context does not already contain the secrets, because that would imply that the secret has already been revealed to the world and is no longer a secret. We will concentrate our focus on whether contexts that do not already know about secrets can possibly distinguish among systems using different secrets at different times. Our analysis is an adaptation of the work done by Abadi *et al.* in [1].

4.1 Safe Contexts and Secrecy in Safe Contexts

Let us now define a formation in which a set of keys are secret from ambients not in a given set of trusted ambients. A formation $(\Delta \triangleright T)$ is safe with respect to a set of ambients $\mathcal{A}$ and a set of secret keys $\mathcal{K}$ if ambients not in $\mathcal{A}$ only see confidential messages encrypted with the keys in $\mathcal{K}$ which are unknown to them. This allows us to specify restrictions that will be imposed upon the environments that will be composed with systems whose secrecy property we wish to check.

Definition 7. *Let $\mathcal{A}$ be a set of ambients, $\mathcal{K}$ be a set of keys, X be a set of variables and $\Delta \triangleright T$ be a formation. $\Delta \triangleright T$ is safe with respect to $\mathcal{A}$, X and $\mathcal{K}$ denoted by $\clubsuit_{\mathcal{A},X,\mathcal{K}}(\Delta \triangleright T)$ if $\Delta \triangleright T$ is a well formed formation, $\mathsf{fv}_{\bar{\mathcal{A}}}(T) \subseteq X$ and $\mathsf{fk}_{\bar{\mathcal{A}}}(T) \cap \mathcal{K} = \{\}$.*

Here by $\mathsf{fv}_{\bar{\mathcal{A}}}(T)$ and $\mathsf{fk}_{\bar{\mathcal{A}}}(T)$ we indicate the free variables and free keys appearing in the ambients not in the set $\mathcal{A}$ appearing in the form $m[P]$ in the system T. We now define substitutions for the free variables appearing in safe contexts where the substitution functions will only replace each free variable with messages encrypted with the keys that are secret from untrusted ambients. We call them *safe* substitutions.

Definition 8. *A substitution function $\sigma : X \longrightarrow M$ is a safe substitution with respect to a set of variables X and a set of keys $\mathcal{K}$ denoted by $\Upsilon_{X,\mathcal{K}}(\sigma)$ if $\mathsf{dom}(\sigma) = X$, σ is injective and for all $x \in X$, $\sigma(x)$ is of the form $\{M\}_k$ where $k \in \mathcal{K}$.*

The following lemma states that a safe formation is structurally equivalent to safe formations no matter what safe substitution is applied to it. Similarly a safe formation only transitions to safe formations irrespective of the safe substitution applied to it.

Lemma 1. *Let $\mathcal{A}$ be a set of ambients, X be a set of variables and $\mathcal{K}$ be a set of keys and $\Delta \triangleright T$ be a formation where $\clubsuit_{\mathcal{A},X,K}(\Delta \triangleright T)$. Let σ be a substitution such that $\Upsilon_{X,\mathcal{K}}(\sigma)$.*

- *If $\Delta \triangleright \sigma(T) \equiv \Delta' \triangleright T'$ then there exists a system S such that $\clubsuit_{\mathcal{A},X,\mathcal{K}}(\Delta' \triangleright S)$, $\mathrm{fv}(S) \subseteq \mathrm{fv}(T)$, $\mathrm{fk}(S) \subseteq \mathrm{fk}(T)$, and $T' = \sigma(S)$ such that whenever $\Upsilon_{X,\mathcal{K}}(\sigma')$, $\Delta \triangleright \sigma'(T) \equiv \Delta'' \triangleright \sigma'(S)$, where $perm(\Delta', \Delta'')$.*
- *If $\Delta \triangleright \sigma(T) \xrightarrow{\xi} \Delta' \triangleright T'$ then there exists a system S such that $\clubsuit_{\mathcal{A},X,\mathcal{K}}(\Delta' \triangleright S)$, $\mathrm{fv}(S) \subseteq \mathrm{fv}(T)$, $\mathrm{fk}(S) \subseteq \mathrm{fk}(T)$, and $T' = \sigma(S)$ such that whenever $\Upsilon_{X,\mathcal{K}}(\sigma')$, $\Delta \triangleright \sigma'(T) \xrightarrow{\xi} \Delta'' \triangleright \sigma'(S)$, where $perm(\Delta', \Delta'')$.*

Proof. By induction on the structure of T.

The next lemma states that the observable behaviors of a safe formation is indistinguishable under different safe substitutions.

Lemma 2. *Let $\mathcal{A}$ be a set of ambients, X be a set of variables and $\mathcal{K}$ be a set of keys. Let σ and σ' be two substitutions such that $\Upsilon_{X,\mathcal{K}}(\sigma)$ and $\Upsilon_{X,\mathcal{K}}(\sigma')$. Then $\{(\Delta \triangleright \sigma(T)), (\Delta \triangleright \sigma'(T)) | \clubsuit_{\mathcal{A},X,\mathcal{K}}(\Delta \triangleright T)\}$ is a barbed bisimulation.*

This lemma along with Proposition 1 allows us to reason about untrusted ambient systems without having to figure out all possible tests the untrusted ambients can perform on a system.

5 Example in Secure Broadcast Ambients

We first show how to impose configurations on systems by adding a configuration on top of part of our encoding for the home network. Assume Router is already in the Wifi ambient and Laptop and Wifi are not inside any ambient. A possible configuration can be $[\mathsf{Laptop} :: \mathsf{empty} \# \mathsf{Wifi} :: \mathsf{empty} \# \mathsf{Router} :: (\mathsf{Wifi}; \mathsf{empty})]$. Now the first action Laptop attempts is entering Wifi. However, the formation $F_0 = [\mathsf{Laptop} :: \mathsf{empty} \# \mathsf{Wifi} :: \mathsf{empty} \# \mathsf{Router} :: (\mathsf{Wifi}; \mathsf{empty})] \triangleright \mathsf{Wifi}[!\overline{\mathsf{in\ Laptop}}.\mathsf{nil}] \| \mathsf{Laptop}[\mathsf{in\ Wifi}.(x)^{\mathsf{Wifi}}.P]$ will not allow this transition as ambient Laptop is trying to enter an ambient which appears later than itself in the configuration. In order to enable this transition, we permute the configuration to $[\mathsf{Wifi} :: \mathsf{empty} \# \mathsf{Laptop} :: \mathsf{empty} \# \mathsf{Router} :: (\mathsf{Wifi}; \mathsf{empty})]$. This configuration is going to allow the ambient Laptop to enter the ambient Wifi. Here is the transition:

$$
\begin{array}{l}
F_0 = [\mathsf{Laptop} :: \mathsf{empty} \# \mathsf{Wifi} :: \mathsf{empty} \# \mathsf{Router} :: (\mathsf{Wifi}; \mathsf{empty})] \triangleright \\
\mathsf{Wifi}[!\overline{\mathsf{in\ Laptop}}.\mathsf{nil}] \| \mathsf{Laptop}[\mathsf{in\ Wifi}.(x)^{\mathsf{Wifi}}.P] \\
\equiv F_1 = [\mathsf{Wifi} :: \mathsf{empty} \# \mathsf{Laptop} :: \mathsf{empty} \# \mathsf{Router} :: (\mathsf{Wifi}; \mathsf{empty})] \triangleright \\
\mathsf{Wifi}[!\overline{\mathsf{in\ Laptop}}.\mathsf{nil}] \| \mathsf{Laptop}[\mathsf{in\ Wifi}.(x)^{\mathsf{Wifi}}.P] \rightarrow \\
F_2 = [\mathsf{Wifi} :: \mathsf{empty} \# \mathsf{Laptop} :: (\mathsf{Wifi}; \mathsf{empty}) \# \mathsf{Router} :: (\mathsf{Wifi}; \mathsf{empty})] \triangleright \\
\mathsf{Wifi}[!\overline{\mathsf{in\ Laptop}}.\mathsf{nil}] \| \mathsf{Laptop}[(x)^{\mathsf{Wifi}}.P]
\end{array}
$$

Now in the manner of [1], we show how to reason about ambient systems to determine whether they are distinguishable from the outside world as to what messages are being communicated in an encrypted form among them. If the components of the home network encrypt their messages then usually the router decrypts them and then forwards them to the worldwide internet network outside the home network. The corresponding code for the router and the Laptop can look like:

$$
\begin{array}{l}
\mathit{\Delta} = [\mathsf{Wifi} :: \mathsf{empty} \# \mathsf{Laptop} :: (\mathsf{Wifi}; \mathsf{empty}) \# \mathsf{Router} :: (\mathsf{Wifi}; \mathsf{empty})] \\
S_1(M) = \mathsf{Laptop}[\langle M \rangle^{\mathsf{Wifi}}.nil] \\
S_2 = \mathsf{Router}[(x)^{\mathsf{Wifi}}.\ \mathsf{case}\ x\ \mathsf{of}\ \{y\}_{k_{\mathsf{Wifi\_Laptop}}} : S(y)] \\
F(M) = \mathit{\Delta} \triangleright (\nu k_{\mathsf{Wifi\_Laptop}}.(S_1(M) \| S_2)) \\
S_{2spec}(M) = \mathsf{Router}[(x)^{\mathsf{Wifi}}.\ \mathsf{case}\ x\ \mathsf{of}\ \{y\}_{k_{\mathsf{Wifi\_Laptop}}} : S(M)] \\
F_{spec}(M) = \mathit{\Delta} \triangleright (\nu k_{\mathsf{Wifi\_Laptop}}.(S_1(M) \| S_{2spec}(M)))
\end{array}
$$

Now what we would like to have is that for any closed term M, $F(M) \simeq F_{spec}(M)$, that is for any closed term, the external world will not be able to distinguish among whether a system sending a specific message was being considered or whether a system handling any message was being considered. In other words the outside world will not be able to distinguish among systems handling different set of messages, and will get same set of responses from all possible tests. However, as we have Proposition 1, we do not need to test the two systems with all possible tests, we only need to determine whether they are barb congruent or not:

$$
\mathit{\Delta} \triangleright (\nu k_{\mathsf{Wifi\_Laptop}}.(S_1(M) \| S_{2spec}(M))) \sim \mathit{\Delta} \triangleright (\nu k_{\mathsf{Wifi\_Laptop}}.(S_1(M) \| S_2)
$$

By the definition of $\sim$, we know that we need to show that for all contexts $\varDelta' \triangleright T$,

$$\varDelta \circ \varDelta' \triangleright (\nu k_{\mathsf{Wifi\_Laptop}}.(S_1(M) \| S_{2spec}(M))) \| T \dot{\sim} \varDelta \circ \varDelta' \triangleright (\nu k_{\mathsf{Wifi\_Laptop}}.((S_1(M) \| S_2) \| T)$$

Without loss of generality we can assume that $k_{\mathsf{Wifi\_Laptop}} \notin \mathrm{fk}(T)$. Then we will need to show that for all contexts,

$$\varDelta \circ \varDelta' \triangleright \nu k_{\mathsf{Wifi\_Laptop}}.(S_1(M) \| S_{2spec}(M) \| T) \dot{\sim} \varDelta \circ \varDelta' \triangleright \nu k_{\mathsf{Wifi\_Laptop}}.(S_1(M) \| S_2 \| T)$$

Since barb equivalence is preserved by restriction (proof not given in this work), we only need to show that

$$\varDelta \circ \varDelta' \triangleright (S_1(M) \| S_{2spec}(M) \| T) \dot{\sim} \varDelta \circ \varDelta' \triangleright (S_1(M) \| S_2 \| T)$$

To prove that, we use a substitution function $\sigma = \{x \mapsto \{M\}_{k_{\mathsf{Wifi\_Laptop}}}\}$ where $\varUpsilon_{\{x\},\{k_{\mathsf{Wifi\_Laptop}}\}}(\sigma)$. We then introduce a relation $\mathcal{S}$: $F_1 \mathcal{S} F_2$ if and only if $F_1 = \varDelta \circ \varDelta' \triangleright (S_{2spec}(M) \| T_1\sigma)$ and $F_2 = \varDelta \circ \varDelta' \triangleright (S_2(M) \| T_1\sigma)$ for some T_1 such that $\clubsuit_{\mathcal{A},X,\mathcal{K}}(\delta \circ \varDelta' \triangleright T_1)$, where $\mathcal{A} = \{\mathsf{Router}, \mathsf{Laptop}\}$, $X = \{x\}$ and $\mathcal{K} = k_{\mathsf{Wifi\_Laptop}}$. Here, $\delta \circ \varDelta' \triangleright T_1\sigma$ represents both $S_1(M)$ and contexts that they do not already know of the key being used between the Router and the Laptop, that is $k_{\mathsf{Wifi\_Laptop}}$ is not known by any ambient other than Router and Laptop in them. The rest of the proof consists of proving that $\mathcal{S} \cup \dot{\sim}$ is a barbed bisimulation.

6 Related work

Mobile ambients have been extended in various ways since their introduction. In [3], *boxed ambients* removed the ability of ambients to open, and added the ability for parent and child ambients to communicate by communication channels. *Safe ambients* [12] and subsequently *NBA(New Boxed Ambients)*[4] allowed the ambient being entered or exited the power to grant (or by omission, deny) the requested entrance or egress. We removed the egress permission to be able to model situations where a mobile device is simply turned off.

Let us now consider the existing broadcast calculi like CBS [20] and $b\pi$[7]. They overcome the limitations of CCS and π-calculus respectively, by allowing one-to-many communication. However, these calculi were not designed to model regions of computation where the regions are capable of restricting access to themselves. Hennessy *et al.* gives a theory of bisimulation equivalence and equational theory and finitary proof system for both the strong and weak versions of the bisimulation equivalences for CBS in [10]. The calculus for Higher Order Broadcast System, HOBS is presented in [19], which is a higher order extension of CBS. Nanz *et al.* extend CBS to CBS# in [17], where they confine broadcast communication to local rooms and impose graphical restrictions on being able to listen in a particular region. However, their regions are not actively mobile. Moraru *et al.* suggests in [16] how the broadcast communication mechanism of CBS can be extended to the ambient calculus as given in [9]. Prasad suggests ways of modeling "globally asynchronous, locally synchronous" broadcast communication scenarios using Mobile Broadcasting Systems, MBS in [21]. Mezzetti *et al.* present Calculus of Wireless Systems (CWS) in [13], where they extend CBS to accommodate nonatomic communication action of wireless networks. Their focus is on modeling wireless communication protocols among usually immobile domains like sensor networks. They use locations and radius to define transmission regions instead of connectivity graphs like CBS#. Their regions, like CBS# are also immobile.

Among the process calculi where the method of communication is not broadcast, a related work is the work on distributed π-calculus [11] by Hennessy *et al.* where we have locations and code mobility among locations. The topology of locations is flat in their calculus and they have code movement but not location movement. Another related calculus is the m-calculus as given in [22] which is a higher-order extension of [8]. Their location topology is tree-structured, with the programmable access control to locations. Another related calculus is the seal calculus given in [6]. However, seals are different from ambients in the structure of their hierarchy and because seals are not subjectively mobile like ambients. *Discretionary ambients* by Nielson *et al.* [18] equip safe ambients with discretionary and mandatory access control.

A related work is that of bigraphs by Milner presented in [14]. Bigraphs are actually a pair of two graphs, a topograph, describing actual physical location and a link graph which describes communication links which does not care about the physical positions. Place graphs are forests of unordered trees whereas our topology is a collection of directed acyclic graphs. Link graphs allow channels between arbitrary pair of agents. They do not have to be adjacent in any way. They lack any mechanism of enforcing access restrictions.

7 Conclusion

We defined Secure Broadcast Ambients which is capable of modeling broadcast communication within a domain. Our calculus allows a directed acyclic graph topology of the location of our agents, hence our topology is more flexible than the usual tree structured one. Our domains or systems are also capable of restricting access to themselves. We then provide co-inductive relations to reason about whether an ambient system keeps a set of messages secret by using encryption. We provide a scenario which our calculus can model more succinctly than other calculi. We present the modeling of this scenario in our calculus to demonstrate the strengths and capabilities of our calculus.

Our calculus however only handles symmetric encryption at this point. Hence we are not capable of modeling communication protocols requiring asymmetric encryption. Developing a complete theory for handling asymmetric encryption is a future goal for us. Also, we intend to apply our calculus for modeling and reasoning about workflows specially workflows where confidentiality and integrity of the data involved is a very significant concern. We have modeled the Automated Identification and Data Capture (AIDC) workflow for hospital data management, being developed at UIUC, using our calculus. We are working towards proving security properties of such scenarios.

References

1. Martín Abadi and Andrew D. Gordon. A Calculus for Cryptographic Protocols: The Spi Calculus. *Inf. Comput.*, 148(1):1–70, 1999.
2. Michele Bugliesi, Giuseppe Castagna, and Silvia Crafa. Reasoning about security in mobile ambients. In *CONCUR '01: Proceedings of the 12th International Conference on Concurrency Theory*, pages 102–120, London, UK, 2001. Springer-Verlag.
3. Michele Bugliesi, Giuseppe Castagna, and Silvia Crafa. Access Control for Mobile Agents: The Calculus of Boxed Ambients. *ACM Transactions on Programming Languages and Systems*, 26(1):57–124, 2004.
4. Michele Bugliesi, Silvia Crafa, Massimo Merro, and Vladimiro Sassone. Communication and mobility control in boxed ambients. *Inf. Comput.*, 202(1):39–86, 2005.
5. Luca Cardelli and Andrew D. Gordon. Mobile Ambients. *Theoretical Computer Science*, 240(1):177–213, 2000. Special Issue on Coordination, Daniel Le Métayer Editor.
6. Giuseppe Castagna, Jan Vitek, and Francesco Zappa Nardelli. The seal calculus. *Inf. Comput.*, 201(1):1–54, 2005.
7. Cristian Ene and Traian Muntean. A broadcast-based calculus for communicating systems. In *IPDPS*, page 149. IEEE Computer Society, 2001.
8. Cédric Fournet and Georges Gonthier. The join calculus: A language for distributed mobile programming. In *APPSEM*, pages 268–332, 2000.
9. Andrew D. Gordon and Luca Cardelli. Equational properties of mobile ambients. *Mathematical Structures in Computer Science*, 13(3):371–408, 2003.
10. Matthew Hennessy and Julian Rathke. Bisimulations for a calculus of broadcasting systems. *Theor. Comput. Sci.*, 200(1-2):225–260, 1998.
11. Matthew Hennessy, Julian Rathke, and Nobuko Yoshida. safeDpi: a language for controlling mobile code. *Acta Inf.*, 42(4-5):227–290, 2005.
12. Francesca Levi and Davide Sangiorgi. Controlling Interference in Ambients. *Transactions on Programming Languages and Systems*, 25(1):1–69, 2003.
13. Nicola Mezzetti and Davide Sangiorgi. Towards a calculus for wireless systems. *Electr. Notes Theor. Comput. Sci.*, 158:331–353, 2006.
14. Robin Milner. Bigraphical reactive systems. In *CONCUR*, pages 16–35, 2001.
15. Robin Milner, Joachim Parrow, and David Walker. A Calculus of Mobile Processes. *Information and Computation*, 100:1–40, 1992.
16. Victor Moraru, Traian Muntean, and Emilian Gutuleac. Towards a model for broadcasting secure mobile processes. In *ISPDC*, pages 168–172, 2006.
17. Sebastian Nanz and Chris Hankin. A framework for security analysis of mobile wireless networks. *Theor. Comput. Sci.*, 367(1-2):203–227, 2006.
18. Hanne Riis Nielson, Flemming Nielson, and Mikael Buchholtz. Security for mobility. In *FOSAD*, volume 2946 of *LNCS*, pages 207–265. Springer, 2002.
19. Karol Ostrovsky, K. V. S. Prasad, and Walid Taha. Towards a primitive higher order calculus of broadcasting systems. In *PPDP*, pages 2–13, 2002.
20. K. V. S. Prasad. A calculus of broadcasting systems. *Sci. Comput. Program.*, 25(2-3):285–327, 1995.
21. K. V. S. Prasad. A prospectus for mobile broadcasting systems. *Electr. Notes Theor. Comput. Sci.*, 162:295–300, 2006.
22. Alan Schmitt and Jean-Bernard Stefani. The m-calculus: a higher-order distributed process calculus. In *POPL*, pages 50–61, 2003.