Available online at www.sciencedirect.com
ScienceDirect
Electronic Notes in Theoretical Computer Science
ELSEVIER
Electronic Notes in Theoretical Computer Science 229 (2009) 77–93
www.elsevier.com/locate/entcs

Verification of Correspondence Assertions in a Calculus for Mobile Ad Hoc Networks

Jens Chr. Godskesen[1]

The IT University of Copenhagen
Rued Langgaards Vej 7
2300 Copenhagen S
Denmark

Hans Hüttel[2] and Morten Kühnrich [3,4]

Distributed and Embedded Semantics
Aalborg University
9220 Aalborg, Denmark

Abstract

We introduce a novel process calculus called DBSPI (distributed broadcast SPI-calculus) which models mobile ad hoc networks (MANET). The calculus is a cryptographic broadcast calculus with locations and migration. Communication and migration are limited to neighborhoods. Neighborhood definitions are explicitly part of the syntax allowing dynamic extension using bound identifiers. In this semantic setting we study authentication of agents in MANET protocols. A safety property dealing with authentication correspondence assertions is defined. Later a dependent type and effect system is given and it is shown to be sound, i.e. protocols which are typeable are also safe. This result is lifted to open systems which involves Dolev-Yao attackers. Our Dolev-Yao attacker may use public keys for encryption and can attack any neighborhood it wishes. Our technique is applied to the Mobile IP registration protocol – a type check shows it is safe. To our knowledge this is the first type system for a MANET calculus doing that.

Keywords: Process calculus theory, Mobile Ad Hoc Networks, MANET, Type and Effect Systems, Broadcast Communication, Mobility

1 Introduction

The last decades have seen the rise of new technologies within the field of communication and networking. The development of new protocols has gone hand in hand with the request for new and more demanding services. One of the more

[1] Email: jcg@itu.dk
[2] Email: hans@cs.aau.dk
[3] Email: mokyhn@cs.aau.dk
[4] Technical report at http://www.cs.aau.dk/~mokyhn/publications/dbspirap.pdf

doi:10.1016/j.entcs.2009.06.030

recent advances is that of mobile ad-hoc networks (MANETs). A MANET is a self-organizing network that does not rely on the existence of central service providers. This means that route discovery and route maintenance are completely local and that mobile devices act as routers and servers. The mobility and limited send and receive range of such devices cause new challenges in protocol construction. Firstly, route discovery and multi-hop protocols are necessary. Secondly, topologies that change dynamically need to be captured. Thirdly, reliability and security are important issues. Finding the right attacker model is not trivial. The Dolev-Yao attacker seems unrealistic strong in a MANET setting because of the limited radio transmission range.

Several process calculi for MANETs have already been proposed. Many of these introduce broadcast communication. On the other hand, the treatment of connectivity and mobility is extremely varied, considering the different approach taken in the work on process calculi with locations, with Hennessy's Dπ calculus [8] and the calculus of Mobile Ambients [5] due to Cardelli and Gordon as prominent representatives.

1.1 Our work

In this paper we describe a process calculus that lets us reason about authenticity properties of MANET protocols that use cryptographic operations. We define an attacker model that allows the user of our calculus to choose the strength of the attacker. One example would be the attacker that may only communicate locally. Another would be the attacker capable of communicating everywhere.

The problem of authentication has been addressed for cryptographic protocols with point to point communication. In particular, the body of work by Gordon and Jeffrey [9,10,11,12] describes how one may analyze authenticity properties of a SPI-calculus process [4] using effect type systems and correspondence assertions in the style of Woo and Lam [15]. The type inference problem for assertion-check type and effect systems has recently been studied independently by Hüttel et. al. [7] and by Kobayashi et. al [13].

The message language of DBSPI contains cryptographic operations. We have locations (like in distributed π–calculus Dπ [8]) and objective migration. Furthermore DBSPI uses synchronous broadcast (with possible loss of packages) restricted to local neighborhoods. In [18] Ene and Muntean compare the expressive power of point-to-point communication and broadcast communication in the π–calculus. They show that it is impossible to encode broadcast communication using point-to-point communication uniformly. This together with the practical need for broadcasts in MANET protocol specifications makes it natural to have a true broadcast primitive in a MANET calculus. The notion of a neighborhood is explicit in the syntax of the calculus. It does not mean that the neighborhood is fixed, since neighbors can be added dynamically (See section 2.4 for a discussion on the expressive power). Furthermore we are able to model link failures and process crashes. Mobility and broadcasts are limited to locations which are neighbors.

On top of the calculus we give a dependent type and effect system for DBSPI

following the tradition of [9,10,11,12]. Using our type system we are capable of proving that the registration protocol of Mobile IP [14,19] is well–typed. This implies that the assertions of the protocol are not violated, hence it is shown that authentication happens correctly. If a network is well–typed, then it is also safe in presence of an attacker, lifting this result to open systems. The type system is tailored for the check of assertions. This means that we do not check for common type errors such as mismatching types in comparisons or arity mismatch. We believe that this is the first type system presented for a calculus within the setting of MANETs. Below we give an example of a DBSPI network. We shall use this as our running example in the text.

Example 1.1 (Mobile IP registration) The Mobile IP protocol [14,19] allows a mobile device (or mobile node, in short MN) to have one single long term IP-address, when connecting to the Internet from different access points. We will assume that a MANET is used to establish connection to MN. Each device has a home agent (in short HA) in some home network associated with it. The agent acting as the access point for MN (called the foreign agent, in short FA) is a router between MN and HA. Every communication with MN from the Internet goes through HA and FA.

The Mobile IP protocol proceeds in three phases: Agent discovery, registration and message tunneling. We will focus on the registration part and assume that agent discovery has already taken place. When MN has been authenticated as MN at HA, HA will serve MN in the manner described above. We shall assume that an asymmetric key pair has been distributed among FA and MN, i.e. FA uses the key pair F^+, F^-. Furthermore assume that MN and HA has agreed on a shared secret key $S_{\mathrm{MN-HA}}$. The messages of the protocol have been simplified focusing only at the cryptographic operations.

We shall write $\mathsf{decrypt}\ y\ \mathsf{is}\ \{y_1, y_2, y_3\}_K.P$ for the decryption of y using key K. If (M_1, M_2, M_3) is the resulting message from this operation then M_i will be substituted for the variable y_i in P.

Our example system is composed of three processes MN, FA, and HA representing the mobile node, foreign agent, and home agent respectively. They communicate via channels m, f and h respectively. We will compose MN, FA and HA in the following network:

$$N = \mathsf{new}\ S_{\mathrm{MN-HA}} : T; A[\mathrm{HA} \mid \mathrm{MN}] \mid \mathrm{B}[\mathrm{FA}] \mid \mathrm{A} \frown \mathrm{B} \mid \mathrm{B} \frown \mathrm{C} \mid \mathrm{A} \frown \mathrm{D} \mid \mathrm{C} \frown \mathrm{D}.$$

In the above A, B, C, and D are so called *locations*. The expression $\mathrm{A} \frown \mathrm{B}$ means that locations A and B are *neighbors* (likewise for B and C etc.). Agents MN and HA are collocated in the beginning.

The protocol is a request/reply protocol. The request message M_{Req} is $\mathrm{Req}, N_{MN}, w, \{\mathrm{Req}, N_{MN}, w\}_{S_{\mathrm{MN-HA}}}$ and the reply message M_{Rep} is given by $\mathrm{Rep}, N'_{HA}, y_2, \{\mathrm{Rep}, N'_{HA}, y_2\}_{S_{\mathrm{MN-HA}}}$. Names w and y_2 are bound in the process definitions below.

$\mathrm{MN} = \mathsf{new}\ N_{MN} : \mathsf{Un};$

$\mathrm{m}(w)$.

begin N_{MN} | // Start authentication of MN
$\bar{\mathrm{f}}(M_{\mathrm{Req}})$. // Send request message
$\mathrm{m}(x)$. // Receive result

decrypt x is $\{y_1, y_2, y_3, y_4\}_{F^+}$. // Forwarder integrity check
decrypt y_4 is $\{z_1, z_2, z_3\}_{S_{\mathrm{MN-HA}}}$.
if $(y_1, y_2, y_3) = (z_1, z_2, z_3)$ then
if $N_{MN} = z_3$ then // Check nonce
end z_2 // End authentication of HA

FA $= !\mathrm{f}(x).\bar{\mathrm{h}}(\{x\}_{F^-}).\mathrm{f}(y).\overline{\mathrm{m}}(\{y\}_{F^-})$ // Router

HA $=$ new N_{HA} : Un; $\overline{\mathrm{m}}(N_{\mathrm{HA}})$.
$\mathrm{h}(x)$. // Receive request
decrypt x is $\{y_1, y_2, y_3, y_4\}_{F^+}$.
decrypt y_4 is $\{z_1, z_2, z_3\}_{S_{\mathrm{MN-HA}}}$.
if $(y_1, y_2, y_3) = (z_1, z_2, z_3)$ then
if $N_{HA} = z_3$ then // Check nonce
new N'_{HA} : Un;
end z_2 | // End authentication of MN
begin N'_{HA} | // Begin authentication of HA
$\bar{\mathrm{f}}(M_{\mathrm{Rep}})$ // Send reply

The protocol starts by HA sending it's initial nonce N_{HA} to MN at location A. The mobile node MN might then start traveling e.g. to location D and from there to location C. Communication with the home agent is possible through the foreign agent FA which is located at B. FA routes registration messages back and forth between the mobile and the home address. If nonce, decryption and validity checks succeed we have a successful registration. This enables the home agent to verify that it was indeed the node that started the authentication. On the other hand, the mobile node can be assured that it was indeed the home agent that replied. We shall prove this fact later when we give types and effects for this example.

1.2 Related work

In [16] Maffeis et al. give a type system involving logic-based policies for correct authorization in a point-to-point communication semantics. Their type system generalizes the work of Gordon et. al. mentioned earlier. Instead of using names as effects they use logical formulæ. That allows formulation of effects not expressible in the type systems of Gordon et. al. It is unclear how their work relates to MANET-calculi.

In [21] following earlier work on the CBS calculus [20], Prasad presents the outline of a broadcast calculus called MBS, where synchronous communication is used locally and asynchronous communication globally. We are capable of simulat-

ing such behavior under the use of our neighborhood relation and an encoding of asynchronous communication. A similar calculus, CBS#, introduced by Nanz and Hankin in [6], uses an explicit connectivity graph and memory stores. Stores can be encoded using the π–calculus fragment of our calculus. The connectivity graphs of CBS# are closely related to our neighborhood relations. Using a control flow analysis they check security properties defined using an equivalence-based approach. In this domain it is unclear, which methodology is stronger; type theory or control flow theory.

In [22] Sangiorgi and Mezzetti introduce a calculus for wireless systems called CWS. The objectives of this work are different from ours. Sangiorgi and Mezzetti study semantic properties of collisions in radio broadcasts. Our work is targeted at a higher level of abstraction where problems related to collisions have already been dealt with.

In [17] Merro describes the calculus CMN, a broadcast calculus with locations and mobility but without cryptography. Here, the connectivity of nodes is described using a distance function and physical radii. It is an interesting and fundamental discussion whether topologies should be given using radii or neighborhood relations. Where Merro gives simulations we use an approximative verification technique.

In [23] Smolka et al. present the ω-calculus which is a conservative extension of the semantics for the π-calculus without cryptographic primitives. Both broadcast and point-to-point communication is supported as well as mobility. Neighbor topologies are called groups, and the use of a special non-standard restriction construct for group names allows one to describe mobility. We think our representation of neighborhoods is more concise - though it might be a matter of taste.

In [3] Godskesen describes a calculus with a rich term language á la Applied π [2] for mobile ad hoc networks called CMAN. The calculus supports locations and neighborhoods and permits dynamic change of the topology. It seems that our concept of neighborhoods in this work is stronger due to dynamic extension using e.g. communication as seen in Example 2.5. Something similar is not possible in Godskesen's calculus.

1.3 Outline of the paper

Section 2 contains the formal syntax and semantics of the DBSPI calculus. Furthermore a Dolev-Yao attacker and authenticity in a MANET is defined. Section 3 defines a dependent type and effect system which ensures that well–typed processes are safe. Furthermore we prove subject reduction. We apply our type system to the Mobile IP Registration protocol and show it is typeable. Section 4 contains our conclusions and directions for further research.

Messages			Processes		
C, K, L, M	::=	$a, b, \ldots$	P, Q	::=	0
	|	$K^+ \mid K^-$		| $P \mid Q$
	|	(M, M')		| $\mathsf{new}\ n : T; P$
	|	$\mathsf{fst}(M) \mid \mathsf{snd}(M)$		| $C(x).P$
	|	$\{M\}_K$		| $\overline{C}(L).P$
	|	ok		| $!P$
				|	$\mathtt{if}\ M = L\ \mathtt{then}\ P\ \mathtt{else}\ Q$
Networks				|	$\mathtt{let}\ x = M\ \mathtt{in}\ P$
N	::=	0		|	$\mathsf{decrypt}\ M\ \mathsf{is}\ \{y\}_K.P$
	|	$N \mid N'$		|	$l \rhd m$
	|	$\mathsf{new}\ n : T; N$		|	$\mathsf{begin}\ L$
	|	$l[P]$		|	$\mathsf{end}\ L$
	|	$l \rhd m$		|	$\mathsf{release}\ L; P$

Table 1
Syntax of the DBSPI–calculus.

2 Syntax and Semantics of the DBSPI–calculus

2.1 Syntax

We will assume the existence of a countable infinite set of names. The syntax of DBSPI which is given in Table 1 consists of *messages*, *processes* and *networks*. The symbol T denotes a type annotation which will be defined in Section 3.

Networks are built from *locations* $l[P]$. The notation $l[P]$ means that process P is at location l. A location is to be thought of as a physical area on a land map which allows inhabitation of processes P. Locations are related through neighborhoods using a special construct $\rhd$. If we write $l[P] \mid m[Q] \mid l \rhd m$ it means that P may send to Q (but not the other way) and P can migrate from l to m. We will write $l \frown m$ for $l \rhd m \mid m \rhd l$ which is a neighborhood allowing bidirectional communication and migration. The empty network is denoted 0, and $\mathsf{new}\ n : T; N$ means that name n of type T is private in network N. Networks N and N' are composed by writing $N \mid N'$. **Processes** are built from synchronous broadcast based communication and the constructs explained in the following. An input prefix $C(x).P$ listens on channel C. When a message M is received $C(x).P$ is rewritten to P where M is substituted for all free occurrences of variable x in P. An output prefix $\overline{C}(M).P$ broadcasts message M on channel C and proceeds as P.

A name n is restricted to process P by the prefix $\mathsf{new}\ n : T; P$. We require that type T is one of the allowed types for names, i.e. it is *generative* (see Section 3). Replication $!P$ allows replicated instantiation of finitely many copies of process P. If two messages M, M' are equal up to simple reductions, then process $\mathtt{if}\ M = M'\ \mathtt{then}\ P\ \mathtt{else}\ Q$ evolves to P otherwise the process evaluates to Q. We will write $\mathtt{if}\ M = M'\ \mathtt{then}\ P$ as shorthand for the process $\mathtt{if}\ M = M'\ \mathtt{then}\ P\ \mathtt{else}\ 0$. Decrypting an encrypted message M using key K is done with $\mathsf{decrypt}\ M\ \mathsf{is}\ \{y\}_K.P$. If message M actually is encrypted by key K's inverse i.e. $\overline{K}$, we continue as P

where y is instantiated with the content of M. Otherwise the process is stuck. Local binding of a message M for a free name x in P is written $\texttt{let}\ x = M\ \texttt{in}\ P$. The constructs $\mathsf{begin}\ L$ and $\mathsf{end}\ L$ are assertions i.e. protocol points. The process $\mathsf{release}\ M; P$ is an annotation used with latent effects in the type system later on. Their meaning will become clearer later. We will write $\prod_\phi P$ for the parallel composition of a finite set of processes, which satisfy the logical predicate ϕ.

Messages are built from names, tuples of names and cryptographic constructors for symmetric and asymmetric keys and encryption. Messages M and M' are paired by (M, M'). Components can be extracted by projections $\mathsf{fst}(M)$ (the first) and $\mathsf{snd}(M)$ (the second).

We have private K^- and public K^+ keying. Symmetric keys are not annotated. We define the following operator $\overline{K}$ on keys by the equations $\overline{K} = K$, $\overline{K^+} = K^-$ and finally $\overline{K^-} = K^+$. Key $\overline{K}$ is called the inverse of K. Encryption of message M under key K is written $\{M\}_K$.

We will use the message ok as a carrier of effects in our static analysis given later on. We transfer effects using the ok-message, instead of using channels with latent effects. Notice that ok is a symbol which is not a name.

2.2 Free names, substitution and structural congruence

Restriction, decryption, let and input prefixes are the name binding constructs of DBSPI. The notations $\mathrm{bn}(P)$ and $\mathrm{bn}(N)$ denote the set of bound names for process P or network N respectively. Function $\mathrm{fn}(\cdot)$ denotes the free names analogously. A name n is said to be *bound* in a process P or in network N if $n \in \mathrm{bn}(P)$ or $n \in \mathrm{bn}(N)$ respectively. A name n is said to be *free* in process P or network N if $n \in \mathrm{fn}(P)$ or $n \in \mathrm{fn}(N)$ respectively. The set of all identifiers is $\mathsf{n}(M)$ or $\mathsf{n}(P)$ or $\mathsf{n}(N)$ for messages, processes and networks respectively. Since we are working with a dependent type system alpha conversions are also defined on types occurring in N and N'. The substitution of a message M for a name v in message L written $L[M/v]$ is defined in the standard way. The substitution of a message M for a free name or variable v in a process P or network N written $P[M/v]$ and $N[M/v]$ respectively, is also defined in the standard way with the added clause $(\mathsf{new}\ a : T; P)[M/v] = \mathsf{new}\ a : T[M/v]; P[M/v],\ a \notin \mathsf{n}(M)$.

Definition 2.1 Structural equivalence on networks is the least reflexive, symmetric and transitive binary relation, closed under alpha conversion, satisfying the rules given below treating parallel composition $\mid$ as a commutative and associative construct with 0 as neutral element.

(S1) $l[\mathsf{new}\ n : T; P] \equiv \mathsf{new}\ n : T; l[P],\ l \neq n$ (S2) $l[P \mid Q] \equiv l[P] \mid l[Q]$
(S3) $N \mid \mathsf{new}\ n : T; N' \equiv \mathsf{new}\ n : T; (N \mid N'),\quad n \notin \mathsf{n}(N)$
(S4) $l[P \mid m \rhd n] \equiv l[P] \mid m \rhd n$

Rules (S1-S3) are standard and inspired from Dπ. Rule (S4) describes how local neighborhood declarations can be pushed to network level.

2.3 Reduction semantics of DBSPI

The reduction rules of the semantics are given in Table 2. The rules define a state transition system, where states are networks and the relation $\rightarrow$ describes reductions between networks. *Reduction is closed under parallel composition, name restriction and structural equivalence.*

Definition 2.2 Reduction of a message M with respect to projection on pairs written $[\![M]\!]$ is defined in the following. We define $[\![\gamma]\!] = \gamma$ when $\gamma \in \{a, \mathsf{ok}\}$ and $[\![K^i]\!] = [\![K]\!]^i$ for $i \in \{+,-\}$. Furthermore $[\![(M_1, M_2)]\!] = ([\![M_1]\!], [\![M_2]\!])$ and $[\![\mathsf{fst}(M)]\!] = M_1$ if $[\![M]\!] = (M_1, M_2)$. If $[\![M]\!]$ is not a pair then $[\![\mathsf{fst}(M)]\!] = \mathsf{fst}(M)$. The second projection $\mathsf{snd}(M)$ is defined as above with M_2 instead of M_1. Two messages M and M' are equal up to projections written $\vdash M = M'$ if $[\![M]\!]$ is syntactically equal to $[\![M']\!]$.

(R-Move)	$l[P] \mid l \rhd m \rightarrow m[P] \mid l \rhd m$
(R-BCom)	$l[\overline{C}(L).P] \mid \prod_{i\in I} m_i[C(x_i).Q_i] \mid l \rhd m_i \rightarrow$ $l[P] \mid \prod_{i\in I} m_i[Q_i[L/x_i]] \mid l \rhd m_i$
(R-If True)	$l[\mathtt{if}\ M = M'\ \mathtt{then}\ P\ \mathtt{else}\ Q] \rightarrow l[P],\ \text{if}\ \vdash M = M'$
(R-If False)	$l[\mathtt{if}\ M = M'\ \mathtt{then}\ P\ \mathtt{else}\ Q] \rightarrow l[Q],\ \text{if}\ \vdash M \neq M'$
(R-Let)	$l[\mathtt{let}\ x : T = M\ \mathtt{in}\ P] \rightarrow N$, where $l[P[[\![M]\!]/x]] \rightarrow N$
(R-Decrypt)	$\mathsf{decrypt}\ \{M\}_N\ \mathsf{is}\ \{y\}_{\overline{N}}.P \rightarrow P[M/y]$
(R-Replication)	$l[!P] \rightarrow l[P] \mid l[!P]$
(R-Release)	$l[\mathsf{release}\ \mathsf{ok}; P] \rightarrow l[P]$

Table 2
Reduction semantics of DBSPI defined on networks

Reduction rule (R-Move) moves process P from location l to m if l and m are neighbors. Rule (R-BCom) defines broadcast communication between neighbors. Since I may be a smaller set than the total set of listeners rule (R-BCom) permits loss of messages. Rules (R-If) are standard. Rule (R-Let) permits the use of a local declaration and (R-Decrypt) describes decryption of an encrypted message. Replication is defined by (R-Replication). The annotation $\mathsf{release}\ \mathsf{ok}; P$ can be reduced using (R-Release). Begin- and end-assertions do not add anything to the semantics.

Example 2.3 (Revisiting the Mobile IP example). Using (R-BCom) HA can send nonce N_{HA} to MN at location A. By rule (R-Move) the mobile node can move to location D and from there to location C. FA routes messages back and forth using rule (R-BCom). Decryption and validity checks are performed using rules (R-Let), (R-If) and (R-Decrypt). We have not yet defined safety formally but we will mention what the idea is informally. The protocol we are working with is safe, since the assertions $\mathsf{end}\ N_{\mathrm{MN}}$ and $\mathsf{end}\ N'_{\mathrm{HA}}$ never occur without corresponding expressions $\mathsf{begin}\ N_{\mathrm{MN}}$ and $\mathsf{begin}\ N'_{\mathrm{HA}}$. Intuitively this means that if an authentication succeeds (marked with an end-assertion) then it was actually initiated by the agent intended (marked with a begin-assertion). The reason is the protecting cryptographic operations involved. Since the attacker has no access to the secret key $S_{\mathrm{MN\text{-}HA}}$ he will not be able to send interfering messages that ultimately (after sanity checks) would create

and unmatched end-assertion say end bogus. This means that the authentication of MN at HA (and vice versa) cannot be interfered by an attacker.

2.4 Considerations on the expressive power of DBSPI

In the examples below we will show typical MANET features expressed in DBSPI. We will start this section by noting that migration is our main semantic tool for what sometimes is called joining and splitting of MANET. If we have the following topology $A \frown B \mid B \frown C \mid B \frown D$ we see that processes at location A and location C may only communicate via a forward node at B. A split of the groups connected to A and C (if A and C has B as their only intermediate node) occurs if B migrates to A. A join would be the opposite situation where a process acting as a forwarder enter location B, thereby interconnecting A and C.

Example 2.4 Link failures occurring now and then are possible to model using reduction rule (R-BCom). Permanent link failures can be modeled by the introduction of a remote "prison location" i.e. a location which allows agents to enter but not to exit. n would be such a location in the network $m \rhd n \mid n[0] \mid m[P]$. If P moves to location n, P will be unable to communicate with other processes. It is not the same as a process crash since P may still perform internal reductions.

Example 2.5 Dynamic extension of neighborhoods are possible since input prefixes may bind names occurring in neighborhood relations. For example $l[C(x).l \rhd x] \mid m[\overline{C}(n).P] \mid m \rhd l$ can result in the network $l \rhd n \mid m \rhd l \mid m[P]$. By (R-Move) used twice this can be rewritten to $l \rhd n \mid m \rhd l \mid n[P]$ which shows that dynamic creation of locations is possible as well.

2.5 Safety and Robust Safety of Networks

Safety is a correspondence between protocol points: Whenever an end L is reached there has already been one or more begin L. This is useful for specification of properties when causality of events matter e.g. when dealing with authentication issues. In that case the protocol designer includes information in L determining the identity of the agent who starts/ends the authentication. In Example 1.1 nonces plays this role. If the authentication ends successfully then it is the same agent that started it (and not a hostile attacker). Before giving the safety property we notice that any network N can be put on the form $N' = \mathsf{new}\ a_1 : T_1; \ldots; \mathsf{new}\ a_n : T_n; N'',\ n \geq 0$ such that $N \equiv N'$ using structural congruence and alpha conversion.

Definition 2.6 (Safety) A network N annotated with begin and end-assertions is safe when the following condition is true: for all reductions on the form $N \rightarrow^* \mathsf{new}\ a_1 : T_1; \ldots; \mathsf{new}\ a_n : T_n; (l[\mathsf{end}\ L] \mid N'),\ n \geq 0$ it holds that $N' \equiv \mathsf{begin}\ L \mid N''$ for some N''.

Example 2.7 The networks end a and begin $a \mid$ end b are unsafe whereas begin $a \mid$ end a and begin $a \mid$ end $a \mid$ begin a are safe.

The safety property we are interested in is safety in presence of an attacker. This is what is called *robust safety*. In this respect it is important to settle an appropriate opponent model. We will use the Dolev-Yao model [25] extended with concepts related to MANET. Choosing the right concepts is not a trivial task. However there are some basic limitations. The opponent is not allowed to use assertions - since this would let the attacker violate safety by creating unmatched end-assertion like $l[\mathsf{new}\ a : \mathsf{Un}; \mathsf{end}\ a]$.

Many attacks in the literature assumes that the attacker can receive messages at one location and e.g. resend them at another location. It seems reasonable to let the attacker choose whatever neighborhood relation he wishes. Rule (R-Move) and the extension of neighborhoods will enable an attacker to visit any location and overhear and send modified messages. If a system is robust with a super attacker like that it will also be robust with a weaker attacker not capable of extending the neighborhood relation. One might argue that this assumption is too strong – leading to unnecessary over-approximations. We will leave the study of weaker assumptions as future work.

Having settled the core of our attacker model we give the last details which are all tied to the type system given later. We will assume that bounded names occurring in the attacker are of type Un (i.e. the type for untrusted data, see section 3). Furthermore we do not allow the attacker to use self created public/private key pairs – this will be left as future work. However he is allowed to use shared public key parts. This is a stronger attacker compared to e.g. [10].

Definition 2.8 (Opponents and robust safety) An opponent O is an assertion free network where all bound names are of type Un. The opponent is not allowed to perform encryption or decryption using secret keys K^-. A network N is robustly safe if $N \mid O$ is safe for all opponents O.

3 A Type and Effect System for DBSPI

In the following we give a dependent type and effect system that approximates robust safety. Well–typed processes are safe. Our effects are assertions and types are used in two ways: they ensure correct use of keys in encryption and decryption and they keep track of effects. We will determine whether end-assertions are correctly matched by preceding begin-assertions under the use of types.

3.1 Effects and Types

Definition 3.1 Effects are finite sets of assertions. The operator begins$(\cdot)$ defined on processes and networks returns top level begin-assertions.

$$\begin{array}{llllll} \mathrm{begins}(\mathsf{begin}\ L) & = & \{L\} & \mathrm{begins}(P \mid Q) & = & \mathrm{begins}(P) \cup \mathrm{begins}(Q) \\ \mathrm{begins}(l[P]) & = & \mathrm{begins}(P) & \mathrm{begins}(N \mid N') & = & \mathrm{begins}(N) \cup \mathrm{begins}(N') \\ \mathrm{begins}(!P) & = & \mathrm{begins}(P) & & & \end{array}$$

$\mathrm{begins}(\mathsf{new}\ a : T; \gamma) = \{L \in \mathrm{begins}(\gamma) \mid a \notin \mathsf{n}(L)\}$, where $\gamma = P$ or $\gamma = N$

$\mathrm{begins}(\cdot) = \emptyset$, for all other P or N.

Definition 3.2 Types T are defined by the following:

$$T ::= \quad \mathsf{Un} \mid \mathsf{SymKey}(T) \mid \mathsf{KeyPair}(T) \mid \mathsf{SecKey}(T) \mid \mathsf{PubKey}(T) \mid \mathsf{Pair}(x : T, T') \mid \mathsf{Ok}(\{L_1, \ldots, L_n\})$$

The type Un (untrusted) is given to public messages and $\mathsf{SymKey}(T)$ is given to names which are used for symmetric encryption and decryption of messages of type T. The type $\mathsf{KeyPair}(T)$ is given to a name k which acts as a seed for public k^+ and private k^- key parts. Types $\mathsf{SecKey}(T)$ and $\mathsf{PubKey}(T)$ are types for private and public key-parts respectively. The type $\mathsf{Ok}(\{L_1, \ldots, L_n\})$ is used to keep track of effects $L_1, \cdots, L_n$. The type $\mathsf{Pair}(x : T, T')$ is used to type dependent pairs where the first component has type T and the second has type $T'[T/x]$.

The types $\mathsf{Un}, \mathsf{SymKey}(T), \mathsf{SecKey}(T), \mathsf{PubKey}(T)$ are *generative*, i.e. used for names. The typing of a name with a non generative type makes no sense. We will identify pair types up to alpha conversion of bound names, i.e. $\mathsf{Pair}(x : T, T') = \mathsf{Pair}(y : T, T'[y/x])$ where y is a fresh name not occurring in T or T'. The domain of the free names operator $\mathrm{fn}(\cdot)$ is extended to types considering binders inside dependent pairs as binders.

Example 3.3 Given the network

$$l[\mathsf{begin}\ b \mid \overline{a}(\{(b, \mathsf{ok})\}_k)] \mid m[a(x).\mathsf{decrypt}\ x\ \mathsf{is}\ \{y\}_k.\mathsf{release}\ \mathsf{snd}(y); \mathsf{end}\ \mathsf{fst}(x)] \mid l \rhd m$$

We will type a and b with types Un, and let the symmetric key k be of type $\mathsf{SymKey}(\mathsf{Pair}(z : \mathsf{Un}, \mathsf{Ok}(\{z\})))$ because k is a symmetric key. Now (b, ok) was given type $\mathsf{Pair}(z : \mathsf{Un}, \mathsf{Ok}(\{z\})$ which means that the message occurring at the first component is of type Un. The second component expresses that there has been observed a begin-assertion of the form $\mathsf{begin}\ z$ for whatever z might be (could be b). ok-messages and dependent types are used to capture that effects has been observed at some points of the protocol. It is hence safe when receiving a message with an ok (at some other point of the protocol) to assume the same effects. This is what the expression release does. It allows effects generated at one place in the protocol to be unleashed into the type environment at another point. The protocol designer will have to insert release-expressions preceding end-expressions. In this way he can use the type system checking that each end-expression will be matched by one or more begin-expressions.

Definition 3.4 Letting S range over sets of messages we will use the following type and effect environment:

$E ::= \emptyset$		Empty environment
$\mid$	$E, u : T$	Identifier u of type T
$\mid$	E, S	Environment containing the effects S

An environment E is well–formed if $E \vdash \diamond$ can be derived using the rules in figure 1 and using the following definitions. The domain of an environment is defined by $\mathrm{dom}(\emptyset) = \emptyset$, $\mathrm{dom}(E, u : T) = \mathrm{dom}(E) \cup \{u\}$, $\mathrm{dom}(E, S) = \mathrm{dom}(E)$. An effect extracting function $\mathrm{effect}(\cdot)$ is defined by $\mathrm{effect}(\emptyset) = \emptyset$, $\mathrm{effect}(E, u : T) = \mathrm{effect}(E)$,

$$\frac{}{\emptyset \vdash \diamond} \qquad \frac{E \vdash \diamond \quad u \notin \mathrm{dom}(E) \quad \mathrm{fn}(T) \subseteq \mathrm{dom}(E)}{E, u : T \vdash \diamond} \qquad \frac{E \vdash \diamond \quad \mathsf{n}(S) \subseteq \mathrm{dom}(E)}{E, S \vdash \diamond}$$

Fig. 1. Well–defined environments

$\mathrm{effect}(E, S) = \mathrm{effect}(E) \cup S$. Furthermore extend the names operator $\mathsf{n}(\cdot)$ to sets of messages by $\mathsf{n}(\{L_1, \ldots, L_n\}) = \cup_{1 \leq i \leq n} \mathsf{n}(L_i)$.

Definition 3.5 (Type judgments) $E \vdash \diamond$ means that environment E is well–formed. $E \vdash M : T$ means that message M is well–formed with type T. The form $E \vdash P$ means that process P is well–typed in environment E and finally $E \vdash N$ means that network N is well–typed in environment E.

3.2 Type rules

All the following type rules will be given using the same scheme. One set of rules for each judgment form involving types which are not public and one set of rules for forms involving the public type Un. We shall comment on the type rules in the following. We omit explanation of trivial rules for brevity. Typing rules for **messages** are given in Figure 2 (rules for **untrusted message** types are given in Figure 3 describing that untrusted subcomponents of a message makes the entire message untrusted). Rule (M-Ok) states that the effect transfered by an ok cannot be more than the effects occurring in the environment. Rule (M-Pair) is the typing rule for pairs. The parameter x is used to describe effects, which depend on the value of x. Rule (M-Snd) likewise makes this dependency clear by the substitution of $\mathsf{fst}(M)$ for x in T_2.

Rules (M-PubKey), (M-SecKey) describe how to derive symmetric and asymmetric keys types from the key seed type $\mathsf{KeyPair}(T)$. Rule (M-SymEnc) describes encryption of a message M under a symmetric key y. The result of encryption is a cipher text which receives type Un. This is safe because the message is only readable to those who have the symmetric key. Rule (M-SecEnc) is the analogous rule using secret keys. Rule (M-Publish) (and the fact that the opponent only use public keys) makes it possible for the opponent to do public key encryption and decryption. In this way public key parts can be used by the opponent via a type environment with names of type $\mathsf{KeyPair}(T)$.

Typing rules for **networks** are given in Figure 4. Notice rule (N-Par). It states that the sub components of a network will be type checked using the effects occurring in the other sub network. This is essential since an end expression occurring in N' may have its matching begin expression in N and vice versa. Typing rules for **processes** are given in Figure 5 (see Figure 6 for the untrusted processes). The rules for input and output are (P-Un-In) and (P-Un-Out). Rule (P-Let) treats local binding of a type. The rule (P-Sym) describes symmetric decryption where a message of untrusted type Un is turned into a concrete type T. This is due to the assumption that opponents cannot access secret keys. Rule (P-Asym) is for the asymmetric case.

$$\text{(M-Id)}\frac{E', u : T, E'' \vdash \diamond}{E', u : T, E'' \vdash u : T} \qquad \text{(M-Ok)}\frac{E \vdash \diamond \quad S \subseteq \mathrm{effect}(E)}{E \vdash \mathsf{ok} : \mathsf{Ok}(S)}$$

$$\text{(M-Pair)}\frac{E \vdash M_1 : T_1 \quad E \vdash M_2 : T_2[M_1/x]}{E \vdash (M_1, M_2) : \mathsf{Pair}(x : T_1, T_2)} \qquad \text{(M-Fst)}\frac{E \vdash M : \mathsf{Pair}(x : T_1, T_2)}{E \vdash \mathsf{fst}(M) : T_1}$$

$$\text{(M-Snd)}\frac{E \vdash M : \mathsf{Pair}(x : T_1, T_2)}{E \vdash \mathsf{snd}(M) : T_2[\mathsf{fst}(M)/x]} \qquad \text{(M-PubKey)}\frac{E \vdash y : \mathsf{KeyPair}(T)}{E \vdash y^+ : \mathsf{PubKey}(T)}$$

$$\text{(M-SecKey)}\frac{E \vdash y : \mathsf{KeyPair}(T)}{E \vdash y^- : \mathsf{SecKey}(T)} \qquad \text{(M-Publish)}\frac{E \vdash y : \mathsf{KeyPair}(T)}{E \vdash y : \mathsf{Un}}$$

$$\text{(M-SecEnc)}\frac{E \vdash M : T \quad E \vdash y^- : \mathsf{SecKey}(T)}{E \vdash \{M\}_{y^-} : \mathsf{Un}}$$

$$\text{(M-SymEnc)}\frac{E \vdash M : T \quad E \vdash y : \mathsf{SymKey}(T)}{E \vdash \{M\}_y : \mathsf{Un}}$$

Fig. 2. Typing of messages

$$\text{(M-Un-Pair)}\frac{E \vdash M_1 : \mathsf{Un} \quad E \vdash M_2 : \mathsf{Un}}{E \vdash (M_1, M_2) : \mathsf{Un}}$$

$$\text{(M-Un-Fst)}\frac{E \vdash M : \mathsf{Un}}{E \vdash \mathsf{fst}(M) : \mathsf{Un}} \qquad \text{(M-Un-Snd)}\frac{E \vdash M : \mathsf{Un}}{E \vdash \mathsf{snd}(M) : \mathsf{Un}}$$

$$\text{(M-Un-Enc)}\frac{E \vdash M : T_1 \quad E \vdash N : T_2}{E \vdash \{M\}_N : \mathsf{Un}},\ \mathsf{Un} \in \{T_1, T_2\} \qquad \text{(M-Un-Ok)}\frac{E \vdash \diamond}{E \vdash \mathsf{ok} : \mathsf{Un}}$$

Fig. 3. Typing of untrusted messages

$$\text{(N-Nil)}\frac{}{E \vdash 0} \qquad \text{(N-Location)}\frac{E \vdash P}{E \vdash l[P]} \qquad \text{(N-Res)}\frac{E, n : T \vdash N}{E \vdash \mathsf{new}\ n : T; N}$$

$$\text{(N-Neighbor)}\frac{}{E \vdash l \rhd m} \qquad \text{(N-Par)}\frac{E, \mathrm{begins}(N') \vdash N \quad E, \mathrm{begins}(N) \vdash N'}{E \vdash N \mid N'}$$

Fig. 4. Typing of networks

$$\text{(P-Nil)}\frac{}{E \vdash 0} \quad \text{(P-Par)}\frac{E, \text{begins}(P_2) \vdash P_1 \quad E, \text{begins}(P_1) \vdash P_2}{E \vdash P_1 \mid P_2} \quad \text{(P-Rep)}\frac{E \vdash P}{E \vdash !P}$$

$$\text{(P-Res)}\frac{E, n : T \vdash P}{E \vdash \mathsf{new}\ n : T; P} \quad \text{(P-If)}\frac{E \vdash M_1 : T_1 \quad E \vdash M_2 : T_2 \quad E \vdash P \quad E \vdash Q}{E \vdash \mathtt{if}\ M_1 = M_2\ \mathtt{then}\ P\ \mathtt{else}\ Q}$$

$$\text{(P-Let)}\frac{E \vdash M : T \quad E, x : T \vdash P}{E \vdash \mathtt{let}\ x = M\ \mathtt{in}\ P} \quad \text{(P-Begin)}\frac{E \vdash \diamond \quad \mathsf{n}(L) \subseteq \text{dom}(E)}{E \vdash \mathsf{begin}\ L}$$

$$\text{(P-Neighbor)}\frac{}{E \vdash l \rhd m} \quad \text{(P-Sym)}\frac{E \vdash M : \mathsf{Un} \quad E \vdash y : \mathsf{SymKey}(T) \quad E, x : T \vdash P}{E \vdash \mathsf{decrypt}\ M\ \mathsf{is}\ \{x\}_y.P}$$

$$\text{(P-Asym)}\frac{E \vdash M : \mathsf{Un} \quad E \vdash y^+ : \mathsf{PubKey}(T) \quad E, x : T \vdash P}{E \vdash \mathsf{decrypt}\ M\ \mathsf{is}\ \{x\}_{y^+}.P}$$

$$\text{(P-End)}\frac{E \vdash \diamond \quad \mathsf{n}(L) \subseteq \text{dom}(E) \quad L \in \text{effect}(E)}{E \vdash \mathsf{end}\ L} \quad \text{(P-Rel)}\frac{E \vdash M : \mathsf{Ok}(S) \quad E, S \vdash P}{E \vdash \mathsf{release}\ M; P}$$

Fig. 5. Typing of processes

$$\text{(P-Un-In)}\frac{E \vdash C : \mathsf{Un} \quad E, x : \mathsf{Un} \vdash P}{E \vdash C(x).P} \quad \text{(P-Un-Out)}\frac{E \vdash L : \mathsf{Un} \quad E \vdash C : \mathsf{Un} \quad E \vdash P}{E \vdash \overline{C}(L).P}$$

$$\text{(P-Un-Dec)}\frac{E \vdash M : \mathsf{Un} \quad E \vdash K : \mathsf{Un} \quad E, x : \mathsf{Un} \vdash P}{E \vdash \mathsf{decrypt}\ M\ \mathsf{is}\ \{x\}_K.P} \quad \text{(P-Un-Rel)}\frac{E \vdash M : \mathsf{Un} \quad E \vdash P}{\mathsf{release}\ M; P}$$

Fig. 6. Typing of untrusted processes

Rule (P-Rel) allows effects generated (by begin assertions) at some place in the protocol, to be unleashed into the environment at this very point. The protocol designer inserts appropriate ok-expressions in messages and release-expressions in front of occurrences of end-expressions. In this way we can use the type system to check that each end-expression is matched, by one or more begin-expressions, occurring at an earlier stage at the protocol.

When a begin-assertion is found the names used in the assertion L must be defined in the type environment. Rules (P-Par) and (N-Par) does the actual extraction of begin-assertions (using the operator begins$(\cdot)$). An end-assertion is correct if it is guarantied that there exists a corresponding begin-assertion elsewhere. This is checked using a lookup in the environment.

3.3 Subject reduction, safety and robust safety

Typability is decidable, sound (well–typed networks are safe) but not complete. A safe (assuming a, b and k are private names) but not typable network: $l[\overline{a}(\{(b,\mathsf{ok})\}_k).\mathsf{begin}\ b] \mid m[a(x).\mathsf{decrypt}\ x\ \mathsf{is}\ \{y\}_k.\mathsf{release}\ \mathsf{snd}(y);\mathsf{end}\ \mathsf{fst}(x)] \mid l \rhd m$. Effect $\{b\}$ is not present when the encryption $\{(b,\mathsf{ok})\}_k$ is performed. On the other hand example 3.3 is typable since the effect $\{b\}$ occurs before encryption.

Well–typedness is preserved under reduction, also in presence of a Dolev-Yao attacker.

Theorem 3.6 (Safety and subject reduction) *Network N is safe if effect$(E) = \emptyset$ and $E \vdash N$. If $E \vdash N$ and $N \rightarrow^* N'$ then $E \vdash N'$.*

Theorem 3.7 (Main theorem) *If $x_1 : T_1, \ldots, x_n : T_n \vdash N$ and $\forall i \exists T_i' : T_i \in \{\mathsf{Un}, \mathsf{KeyPair}(T_i')\}$ then it holds that network N is robustly safe, where* $\mathrm{fn}(N) \subseteq \{x_1, \ldots, x_n\}$.

Example 3.8 (Typing the Mobile IP example) First annotate the protocol with ok-messages and appropriate release-expressions making latent effects transferable. Add ok-messages to request and reply messages as follows: M_{Req} becomes $\mathrm{Req}, N_{MN}, w, \{\mathrm{Req}, N_{MN}, w, \mathsf{ok}\}_{S_{\mathrm{MN-HA}}}$ and M_{Rep} becomes $\mathrm{Rep}, N'_{HA}, y_2, \{\mathrm{Rep}, N'_{HA}, y_2, \mathsf{ok}\}_{S_{\mathrm{MN-HA}}}$. Furthermore one extra parameter is added in decryption: Expression $\mathsf{decrypt}\ y_4\ \mathsf{is}\ \{z_1, z_2, z_3\}_{S_{\mathrm{MN-HA}}}$ of MN becomes $\mathsf{decrypt}\ y_4\ \mathsf{is}\ \{z_1, z_2, z_3, z_4\}_{S_{\mathrm{MN-HA}}}$ and **end** z_2 becomes $\mathsf{release}\ z_4; \mathbf{end}\ z_2$. The same is done analogously for HA.

Second, assign types to names of N. Key $S_{\text{MN-HA}}$ gets type $\mathsf{SymKey}(\mathsf{Pair}(x : \mathsf{Un}, \mathsf{Pair}(y : \mathsf{Un}, \mathsf{Pair}(z : \mathsf{Un}, \mathsf{Ok}(y)))))$ and the type environment E for free names is defined by:

$$E = A : \mathsf{Un}, B : \mathsf{Un}, C : \mathsf{Un}, D : \mathsf{Un}, \mathrm{f} : \mathsf{Un}, \mathrm{h} : \mathsf{Un}, \mathrm{m} : \mathsf{Un},$$
$$\mathrm{Req} : \mathsf{Un}, \mathrm{Rep} : \mathsf{Un}, F : \mathsf{KeyPair}(\mathsf{Un}), H : \mathsf{KeyPair}(\mathsf{Un})$$

Third, under the use of the type system it can be shown that $E \vdash N$, and by our main result N is robustly safe. When HA receives an authentication request it is indeed from the mobile node (and vice versa). All begin-assertions are typable by (Begin). The crucial point is that all end-assertions are well typable by (End). This is the case due to (P-Rel) and type rules (M-Fst) and (M-Snd) for projection on messages.

4 Conclusion and Future work

We defined a process calculus for MANET called DBSPI which is a distributed cryptographic π–calculus. Following previous work on type systems by Gordon and Jeffrey [9,10,11,12] we analyzed authenticity properties using correspondence assertions. Using a stronger attacker model than in [10] we proved subject reduction, type safety and robust safety. Finally we applied our techniques to an example

inspired from the Mobile IP registration protocol. We believe this type system is one of the first for a MANET process calculus.

Future work includes behavioral studies by definition of suitable equivalences. The study of weaker attacker models seems important. A sub-typing relation á la [10] in the type system would result in a more precise approximation of safety.

Finally it would be interesting to compare the expressive power of DBSPI with CMAN [3]. The CMAN calculus can model all the cryptographic operations from DBSPI and broadcast communication. It seems plausible that DBSPI networks with a fixed neighbor topology and bidirectional neighbor relations could be simulated in CMAN.

References

[1] Bruno Blanchet: Computationally Sound Mechanized Proofs of Correspondence Assertions. In 20th IEEE Computer Security Foundations Symposium (CSF'07), pages 97-111, Venice, Italy, July 2007. IEEE.

[2] Martin Abadi and Cedric Fournet: Mobile values, new names, and secure communication. SIGPLAN Not., vol. 36, number 3, 2001.

[3] Jens Chr. Godskesen: A Calculus for Mobile Ad Hoc Networks (Extended Abstract). To appear in Proceedings of Coordination'07, Paphos, Cyprus. June 2007.

[4] Martín Abadi and Andrew D. Gordon: A Calculus for Cryptographic Protocols: The Spi Calculus. Fourth ACM Conference on Computer and Communications Security. ACM Press, pp. 36–47, 1997.

[5] Luca Cardelli and Andrew D. Gordon. Mobile Ambients. Foundations of Software Science and Computation Structures: First International Conference, FOSSACS '98.

[6] Sebastian Nanz and Chris Hankin: A framework for security analysis of mobile wireless networks. Theor. Comput. Sci. Vol. 367, number 1. 2006.

[7] Andrew D. Gordon, Hans Hüttel and René Rydhof Hansen: *Inferring Correspondence Types in a Polarized Pi-Calculus*, Submitted to SECCO 2008.

[8] Matthew Hennessy: *A Distributed Pi-Calculus*, Cambridge University Press 2007.

[9] Andrew D. Gordon and Alan Jeffrey: Typing One-to-One and One-to-Many Correspondences in Security Protocols. Software Security – Theories and Systems, Mext-NSF-JSPS International Symposium, ISSS 2002, Tokyo, Japan, November 8-10, 2002.

[10] Andrew D. Gordon and Alan Jeffrey: Types and Effects for Asymmetric Cryptographic Protocols. Proc. CSFW 15, 2002.

[11] Andrew D. Gordon and Alan Jeffrey: Authenticity by Typing for Security Protocols. Journal of Computer Security, Volume 11, number 4, 2003.

[12] Andrew D. Gordon and Alan Jeffrey: Typing correspondence assertions for communication protocols. Theor. Comput. Sci., Vol 300, number 1-3, 2003.

[13] Daisuke Kikuchi and Naoki Kobayashi. *Type-based verification of correspondence assertions for communication protocols*. In Zhong Shao, editor, APLAS, volume 4807 of Lecture Notes in Computer Science, pages 191-205. Springer, 2007.

[14] Sufatrio and Kwok-Yan Lam: Mobile IP Registration Protocol: A Security Attack and New Secure Minimal Public-Key Based Authentication, 1999 International Symposium on Parallel Architectures, Algorithms and Networks (ISPAN '99), 23-25 June 1999, Fremantle, Australia.

[15] Thomas Y.C. Woo and Simon S. Lam: A Semantic Model for Authentication Protocols. Research in Security and Privacy, Oakland, CA, USA. 1993.

[16] Fournet, Cedric; Gordon, Andrew; Maffeis, Sergio: A Type Discipline for Authorization in Distributed Systems. Computer Security Foundations Symposium, 2007. CSF '07. 20th IEEE, pp.31-48, 6-8 July 2007.

[17] Massimo Merro: An Observational Theory for Mobile Ad Hoc Networks, ENCS. Vol 173, 2007.

[18] Christian Ene and Traian Muntean: Expressiveness of point-to-point versus broadcast communications. FCT '99 : fundamentals of computation theory (30 August - 3 September 1999). LNCS vol. 1684.

[19] Perkins C. ed.: IP Mobility Support. IETF RFC 2002, October 1996.

[20] K. V. S. Prasad: A Calculus of Broadcasting Systems. Sci. Comput. Program. Vol 25, Number 2–3, pp. 285–327. 1995.

[21] K. V. S. Prasad: A Prospectus for Mobile Broadcasting Systems. Electr. Notes Theor. Comput. Sci., Vol 162, pp. 295–300. 2006.

[22] Nicola Mezzetti and Davide Sangiorgi: Towards a Calculus For Wireless Systems. Electr. Notes Theor. Comput. Sci., vol 158, pages 331–353. 2006.

[23] Anu Singh, C.R.Ramakrishnan, and Scott A. Smolka: A Process Calculus for Mobile Ad Hoc Networks.

[24] Davide Sangiorgi and David Walker. *The π-calculus: A Theory of Mobile Processes*, Cambridge University Press 2001.

[25] Danny Dolev and Andrew C. Yao: On Security of Public Key Protocols, IEEE trans. on Information Theory, IT-29, 198-208. 1983.