

# A Formal Policy Specification Language for an 802.11 WLAN with Enhanced Security Network

Handan Gül Çalkılı and Ufuk Çağlayan

Department of Computer Engineering, Boğaziçi University,  
Bebek, İstanbul 34342, Turkey  
{gul.calikli, caglayan}@boun.edu.tr

**Abstract.** In Wide Area Networks (WANs) inconsistencies among the security policies of Administrative Domains (ADs) may cause severe security flaws. Recently, security policies are written in natural language and as they get more complicated, even for an expert it might be impossible to detect such inconsistencies. However, when a formal language is used, it might be possible to make verification of security policies by automated theorem proving tools. Due to the existence of mobile devices such as laptops, PDAs and mobile agents, we need a formal language that is capable of defining the concept of mobility. In this paper, we extend  $Mob_{adtl}$  [1] according its refinement methodology to obtain a formal policy specification language for an 802.11 WLAN with enhanced security network.

**Keywords:** Administrative Domain, 802.11 Wireless Local Area Network (WLAN),  $Mob_{adtl}$ , Security Policy.

## 1 Introduction

In this paper we extend  $Mob_{adtl}$  [1] according to its refinement methodology to form a formal policy specification language for an 802.11 WLAN with enhanced security network. Refinements are made by adding axioms for system information, architectural operations and file access operations to existing axioms of  $Mob_{adtl}$ .

In the literature, there are network-aware formal models. Ambient Calculi are aimed to be used in the design process of programming languages for mobile environments. Among the Ambient Calculi we examined, there were Mobile Ambients [6] and its variants Mobile Safe Ambients [7], Secure Safe Ambients [8] and Boxed Ambients [9]. However, the semantics and syntax of Ambient Calculi are complicated. The model proposed by Cuppens and Saurel in [10] use deontic logic for modelling the concepts of permission, obligation and prohibition with a modal logic of action. In [11], Ryutov and Neuman propose a model that is capable of representing existing access control mechanisms such as *Access Control List* (ACL) and *Condition*.

Finally,  $Mob_{adtl}$  is an ensemble of a model, a logic to formalize the model and a methodology required to structure, refine and compose specifications to obtain other specifications [1]. Compared to Ambient Calculi, functional units of  $Mob_{adtl}$  remain relatively simple while enabling the specification of rich coordination policies in an incremental manner. In addition, unlike most models such as Boxed Ambients and Mobile Ambients,  $Mob_{adtl}$  does not base its model on a basic security mechanism. Thus each authority can define and modify its own security policy.

In order to form axioms for architectural operations and system information, necessary information is extracted from IEEE 802.11 [12] and IEEE 802.11i [13] standards, respectively.

The rest of this paper is structured as follows. Overview of security aspects of IEEE 802.11 and IEEE 802.11i standards is given in Section II followed by an overview of network-aware formal models in the literature and  $Mob_{adtl}$ . Examples for enforcement of proposed formal policy language are given in Section III. Finally, Section IV summarizes the work done and mentions future work.

## 2 Background

The formal policy specification language, we formed refining  $Mob_{adtl}$ , contains axioms for architectural operations and system information. Information required to form axioms for architectural operations and system information was extracted from IEEE 802.11 and IEEE 802.11i Standards, respectively. This section also includes an overview of  $Mob_{adtl}$  and other related network-aware formal models.

### 2.1 Overview of IEEE 802.11 and 802.11i Aspects

The most basic component of the 802.11 wireless network is station (STA) which is any device containing conformant medium access control (MAC) and physical layer (PHY) interface to the Wireless Medium (WM). The building block of an IEEE 802.11 WLAN is basic service set (BSS) and it consists of a group of any number of STAs. 802.11 wireless networks can exhibit two different basic architectures which are infrastructure basic service set and independent basic service set (IBSS). In an IBSS a STA communicates directly with one or more other STAs and a minimum IEEE 802.11 WLAN may consist of only two STAs. Mobile STAs must be within the range of each other to communicate. This type of operation is often referred to as an ad hoc network. An infrastructure basic service set is a BSS with a component called access point (AP). An AP is any entity that has STA functionality and provides access to the distribution services, via WM for associated STAs [12]. In an infrastructure BSS, STAs communicate via AP, but not directly with each other. Distribution System (DS) interconnects a set of BSSs and integrated local area networks (LANs) to create an extended service set (ESS) [12]. IEEE 802.11i [13] standard inherits basic security aspects from IEEE 802.1X standard which is the standard for port based network access control. The operation of port

based access control has the effect of creating two distinct points of access to the authenticator system's point of attachment to the LAN. One of these two distinct points of attachment is the uncontrolled port which allows port data unit (PDU) exchange regardless of the authorization state; whereas the other is the controlled port which requires port to be authorized for allowing PDU exchanges. The authenticator Port Access Entity (PAE) uses the uncontrolled port for the purposes of exchanging protocol information with the supplicant. If MAC is physically or administratively inoperable, then no protocol exchanges of any kind can occur on either the controlled port or uncontrolled port. A STA can be part of an ESS, only if it is associated to an AP. If 802.1X EAP authentication is used, for association it should be followed by four way handshake. 802.1X EAP authentication process in an ESS can be initiated either by the supplicant or authenticator. If the EAP authentication succeeds, IEEE 802.1X controlled port is blocked so that general data traffic is not allowed. Authenticator associated with AP exchanges authentication messages only with supplicant which made a request for authentication (or to which the authenticator has sent authentication request message). When 802.1X EAP authentication is used in an IBSS, each STA will need to include 802.1X authenticator and authentication server. During IEEE 802.1X EAP authentication, each STA's supplicant will send an EAPOL-Start message to every other STA to which it wants to authenticate and each STA's authenticator will respond the identity of the credential it wants to use. For a STA to be a member of an IBSS, it should authenticate to each STA in that IBSS. In addition, four way handshakes should be performed in a mutual manner.

## 2.2 Overview of Network-Aware Formal Models and $\text{Mob}_{adtl}$

In [10] Cuppens and Saurel model the concepts of permission, obligation and prohibition using deontic logic with a modal logic of action. Temporal representation is introduced to the model via boolean attributes *Before*, *After* and *During*. Ryutov and Neuman present in [11] a model based on set and function formalism. *Mobile Ambients* (MA) [6] describe the movement of processes and devices through and within ADs. In MAs, interaction is *one-sided* and one of the two partners simply undergoes the action of moving or opening an ambient. In *Mobile Safe Ambients* (SA) [7], which is a variant of MA, a *mutual agreement* between the two partners undergoing an action is required. *Secure Safe Ambients* (SSA) [8] is a typed variant of SA. This type system provides for static detection of security attacks such as Trojan Horses and other combinations of malicious agents. *Boxed Ambients* (BA) [9] is a variant of MA and it provides finer grained abstractions for resource protection and access control in systems of distributed and mobile agents. Finally  $\text{Mob}_{adtl}$  [1] is an ensemble of a model, a logic to formalize the model and a methodology required to structure, compose and refine specifications to obtain other specifications.  $\Delta\text{DSTL}(x)$  [2] is an asynchronous, distributed, temporal first order logic used to formalize the model of  $\text{Mob}_{adtl}$ . It adds a temporal structure and the concept of event on the top of  $\text{DSL}(x)$ , which is

a first order logic for reasoning on properties of distributed systems. The syntax of DSL(x) formulae defined in [3] is as  $F :: A \mid \perp \mid \neg F \mid F \wedge F' \mid \forall F \mid \exists F \mid m_i F$ .

In the syntax of DSL(x), formula A is an atomic first order formula,  $\perp$  is the constant *false*,  $m_i$  is a modality for locality built using the name of component  $m_i$ .  $\overline{m}_i$  is used to denote the dual of  $m_i$  satisfying the following equation  $\overline{m}_i = \neg m_i \neg F$ .

$\Delta$ DSTL(x) owns operators which relate conditions in different states of a computation. These operators and their functionalities are listed below:

- *STABLE* is used to state that a condition will keep staying *true* once it is established,
- *INIT* defines conditions holding in the initial state of a computation,
- *LEADS\_TO* expresses a *liveness* condition and defines the *sufficient* causes for a condition to hold,
- *BECAUSE* expresses a *safety* condition and defines the *necessary* causes for a condition to hold.

The concept of *event* (i.e. becoming *true* of a condition) is defined using the formula of the form  $\Delta F$  which is a part of  $\Delta$ DSTL(x) syntax.

In the model of  $\text{Mob}_{adtl}$  [1] *neighborhood* is a bounded environment where both stationary and mobile components live. *Components* have unique names determined by their initial neighborhood. Each neighborhood is associated with a stationary component called the *guardian*. The components are location aware due to the knowledge of their guardians. Communications are based on asynchronous message passing and they occur between components via guardians. A guardian intercepts messages and decides which of them can enter or leave the neighborhood it controls. Similarly, the guardians intercept components and decide which of them can enter or leave their neighborhood. Inter-operability of  $\text{Mob}_{adtl}$  provides enforcement and implementation of the security policies by different security mechanisms. Most models for secure mobile systems base their models on a specific security mechanism. For instance *Mobile Ambients*(MA) [6] specifies and enforces access control policies of mobile agents via capability based type system. In *Boxed Ambients* (BA), the resource access control framework defined is an instance of the standard *Mandatory Access Control* policies in multi-level security environments.  $\text{Mob}_{adtl}$  provides each guardian(authority) with the ability to define its own security policy. It also allows each authority to modify its own security policy by adding new restrictions. Since  $\text{Mob}_{adtl}$  separates functionality from coordination, it can specify both component functionalities and the security requirements independently at the abstract level. Mobility in  $\text{Mob}_{adtl}$  is subjective, in other words mobile units can control their location. In Seal Calculus, seals are moved by their parents, thus mobility is objective and cannot be controlled. Ambient Calculus is a hybrid; ambients can decide to move, but they carry their sub-ambients with themselves so that sub-ambients are moved in an objective way. The fact that MA supports both subjective and objective mobility can be seen as a source of interferences and wrong behaviors. *Safe Ambients* (SA) [7] introduces a form of control centralization with the notion of single thread and constrains subjective and objective moves via

co-actions and synchronization, and thus it has similarities with  $Mob_{adtl}$ . In SA, only one process in each ambient can have the capability of making a move. Such notion of authority is similar to that of guardians in  $Mob_{adtl}$ . Secondly, co-action mechanism is similar to the interaction protocol between an agent and its guardian. Finally, reasoning on policies in  $Mob_{adtl}$  is automatized giving us the opportunity to make verifications in order to detect the security flaws in the overall system using MaRK [1].

### 3 $Mob_{adtl}$ Specialized for 802.11 WLANs with Enhanced Security Network

In our work, refinements and additions are made to the theory for mobile devices which was given as an example for usage of  $Mob_{adtl}$  in [1]. *Neighborhood* is used to model ADs. An AD may consist of a hybrid of IBSS and ESS topologies. Sometimes an AD can consist of two or more mobile devices, as in ad-hoc networks. Moreover, when a mobile device has no connection with a mobile device or an AP, then its neighborhood is itself. The *guardian* concept of  $Mob_{adtl}$  model is used to model the central authority of a neighborhood which decides to forward or veto mobility, message and file access requests according to that neighborhood's security policy. In addition, a *guardian* is able to detect approaching and leaving STAs as well as reachability of a STA [1]. A *component* can be both a mobile and stationary device.

The axiom set for architectural operations consists of axioms for ESS/BSS and IBSS topologies. These axioms are encodings of the necessary extracts from IEEE 802.11 standard and are given in Table 1. Axioms for system information include EAP 802.1X authentication, and four way handshake procedures extracted from IEEE 802.11i standard as shown in Table 2. Axioms in Table 3, concern the basic file access operations which are *read*, *write* and *execute* respectively and data flow.

**Table 1.** Axioms for Architectural Operations

- EL1:**  $S(\text{associatedTo}(AP)) \wedge AP(\text{guardedby}(G)) \text{LEADS\_TO } G(\text{guarding}(S))$
- EL2:**  $\overline{S}(\text{associatedTo}(AP) \wedge \text{associated to}(AP')) \rightarrow AP \neq AP'$
- EIBL3:**  $G(\text{reachable}(S) \wedge \text{satisfies}(S,P)) \text{LEADS\_TO } G(\text{guarding}(S))$
- EL4:**  $G(\Delta \text{exit}(S,P)) \text{LEADS\_TO } G(\text{moving}(S))$
- ES1:**  $\overline{AP}(\text{guardedby}(G))$
- EIBS2:**  $S(\text{guardedby}(G)) \text{BECAUSE } S(\text{associatedTo}(AP)) \vee (S(\text{ibssAuthTo}(S')) \wedge S'(\text{ibssAuthTo}(S)))$
- ES3:**  $S(\text{associatedTo}(AP)) \text{BECAUSE } S(\text{mutualAuthTo}(AP))$
- IBL1:**  $S(\text{ibssAuthTo}(S')) \text{LEADS\_TO } S'(\text{ibssAuthTo}(S))$
- IBL2:**  $S(\text{ibssAuthTo}(S')) \text{LEADS\_TO } G(\text{guarding}(S) \wedge \text{guarding}(S'))$
- IBS1:**  $S(\text{ibssAuthTo}(S')) \text{BECAUSE } S(\text{mutualAuthTo}(S')) \wedge S'(\text{mutualAuthTo}(S))$

**Table 2.** System Information Axioms

- SIL1:**  $S(\Delta EAPAuthenticateTo(A)) \text{ LEADS\_TO } A(\Delta protocol\_msg(EAPOLStart, S, A))$
- SIL2:**  $A(\Delta EAPAuthenticate(S)) \text{ LEADS\_TO } S(\Delta protocol\_msg(EAPRequest, A, S))$
- SIL3:**  $A(\Delta protocol\_msg(EAPOLStart, S, A)) \text{ LEADS\_TO } S(\Delta EAPAuthenticatingTo(A) \vee toBeVetoed(protocol\_msg(EAPOLStart, S, A), D))$
- SIL4:**  $A(\Delta protocol\_msg(EAPOLStart, S', A) \wedge EAPAuthenticating(S) \wedge S \neq S') \text{ LEADS\_TO } S'(\Delta toBeVetoed(protocol\_msg(EAPOLStart, S', A), D))$
- SIL5:**  $S(\Delta EAPAuthTo(A)) \text{ LEADS\_TO } A(\Delta portAuthorized) \wedge \Delta block(ControlledPort, S)$
- SIL6:**  $S(\Delta mutualAuthTo(A)) \text{ LEADS\_TO } A(\neg block(controlledPort, S))$
- SIS1:**  $G(\Delta DataIn(D, G') \vee G(\Delta DataOut(D, G')) \text{ BECAUSE } A(\neg block(controlledPort, S) \wedge guardedby(G)) \wedge S(guardedby(G))$
- SIS2:**  $S(mutualAuthTo(A)) \text{ BECAUSE } S(EAPAuthTo(A))$
- SIS3:**  $G(exit(S, P)) \text{ BECAUSE } (S(\neg mutualAuthTo(AP)) \wedge AP(guardedby(G))) \vee (S(\neg ibssAuthTo(S')) \wedge S'(guardedby(G)))$
- SIS4:**  $A(\neg macEnabled) \text{ BECAUSE } A(Disconnect)$
- SIS5:**  $A(Disconnect) \text{ BECAUSE } A(EAPAuth(S) \wedge (\neg KeyAvailable \vee TimeOut))$

**Table 3.** Axioms for File Access Operations, Mobility, Communication and Data Flow

- FL1:**  $S(\Delta Read(F, O) \wedge guardedby(G)) \text{ LEADS\_TO } G(\Delta read(F, S, O))$
- FL2:**  $G(\Delta read(F, S, O) \wedge guarding(S)) \text{ LEADS\_TO } S(\Delta toBeVetoed(ftpConnection, D)) \vee G'(\Delta read(F, S, O) \wedge guarding(O))$
- FL3:**  $G(\Delta read(F, S, O) \wedge guarding(O)) \text{ LEADS\_TO } S(\Delta toBeVetoed(read(F, S, O), D)) \vee S(\Delta readIn(F, O))$
- FL4:**  $S(\Delta Write(F, S, O) \vee \Delta Execute(F, S, O)) \wedge S(guardedby(G)) \text{ LEADS\_TO } G(\Delta write(F, S, O))$
- FL5:**  $G(write(F, S, O) \vee execute(F, S, O)) \text{ LEADS\_TO } G(read(F, S, O))$
- FS1:**  $P(\Delta ReadIn(F, S)) \text{ BECAUSE } G1(guarding(S) \wedge allow(P, Connect, ftpPort)) \wedge G1(allow(P, Read, F) \vee allow(P, Write, F) \vee allow(P, Execute, F))$
- FS2:**  $S(Read(F, O) \wedge \Delta failure\_msg) \text{ BECAUSE } S(guardedby(G)) \wedge G(\neg reachable(O))$
- MS5:**  $G2(\Delta moving(S)) \text{ BECAUSE } G1(guarding(S) \wedge allow(S, moveTo, G2))$
- CS1:**  $P(\Delta in(M, S)) \text{ BECAUSE } G1(guarding(S) \wedge allow(S, Send, (M, P)))$
- DF1:**  $\forall G1, G2, G3, F. G2(\Delta DataIn(F, G1)) \wedge G3(\Delta DataIn(F, G2)) \rightarrow G3(\Delta DataIn(F, G2))$

### 3.1 Examples of Enforcement of Formal Policy Specification Language for 802.11 WLANs with Enhanced Security Network

In this section, examples for the enforcement of the axioms we formed as an extension of  $Mob_{adtl}$  to make verifications for 802.11 WLANs with enhanced security network are given.

For instance, a component of an AD may want to read a file F owned by entity O. If O or (and) S does (do) has no connection with a STA, then response

of the query requesting read access to file F will be a failure message. This is stated in axiom FS3 of file operations axiom set in Table 3. If a STA is not part of an BSS/ESS, then it is not mutually authenticated to an AP. On the other hand, if a station is not part of an BSS/ESS, then there is not a station S' so that S and S' are mutually authenticated to each other. These facts are stated in axiom SIS3 in Table 2. If S and S' are not part of the same IBSS, then this means that either (or both) of them did not mutually authenticate to each other as stated in axiom IBS1 in Table 1. Thus, if S is not mutually authenticated to a station, the reason for this might be,

- S might have performed 802.1X EAP authentication successfully. However, a problem may have occurred during four way handshake. This might be due to the fact that either Pairwise Master Key (PMK for ESS/BSS architecture, and it is PSK for IBSS architecture) is not received by authenticator (i.e.  $A(\neg KeyAvailable)$ ) or after supplicant receives the message with Anonce (Anonce is a random number generated by authenticator during four way handshake in order to be used in derivation of PTK) from the authenticator, it sends no response to the authenticator (i.e.  $A(\Delta TimeOut)$ ), or
- Neither authenticator nor supplicant wanted to perform 802.1X EAP authentication, (i.e.  $S(\neg EAPAuthenticateTo(A)) \wedge A(\neg EAPAuthenticate(S))$ ) which is supposed to be followed by four way handshake for the mutual authentication, or
- The supplicant wanted to initiate 802.1X EAP authentication and it received a veto message, (i.e.  $S(\Delta veto(protocol\_msg(EAPOL\_Start, S, A)))$ ) or
- The authenticator wanted to initiate 802.1X EAP authentication and it received a veto message, (i.e.  $A(\Delta veto(protocol\_msg(EAPRequest, A, S)))$ ) or
- Supplicant has not yet finished 802.1X EAP authentication and it is authenticating to the authenticator, (i.e.  $S(EAPAuthenticatingTo(A))$ ), or
- Supplicant has successfully performed 802.1X EAP authentication and has just started four way handshake. Thus, controlled port of the authenticator is blocked for data exchange, (i.e.  $A(\Delta block(controlledPort, S))$ ) and only messages related to four way handshake are allowed.

The possible reasons for why S is not mutually authenticated to a station can be written in  $Mob_{adt1}$  as follows:

$$\begin{aligned}
& S(\neg mutualAuthTo(A)) \text{ BECAUSE } A(\neg KeyAvailable) \vee A(\Delta TimeOut) \vee \\
& (S(\neg EAPAuthenticateTo(A)) \wedge A(\neg EAPAuthenticate(S))) \vee \\
& S(\Delta veto(protocol\_msg(EAPOLStart, S, A))) \vee \\
& A(\Delta veto(protocol\_msg(EAPRequest, A, S))) \vee \\
& S(EAPAuthenticatingTo(A)) \vee A(\Delta block(ControlledPort, S))
\end{aligned}$$

As mentioned before, serious security flaws arise as a result of the inconsistencies among the policy elements of different ADs. Detection of such inconsistencies is possible by using a verifiable formal language.

Figure 1 shows a system consisting of three ADs together with their security policies. In the security policy representation of Figure 1, we used the model

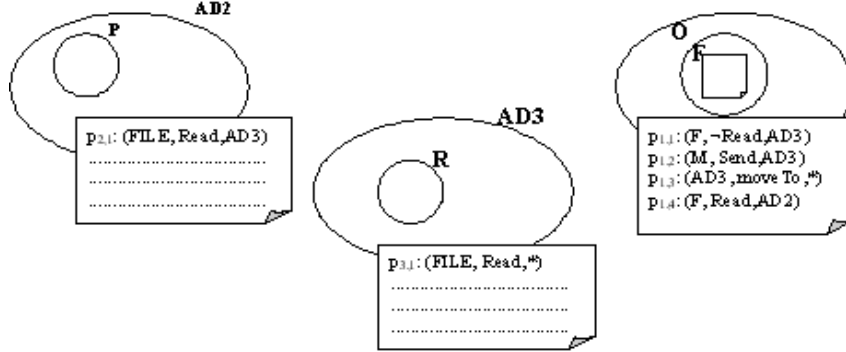


Fig. 1. A system consisting of three administrative domains

based on *Set and Function Formalism* proposed in [11] by Ryutov and Neuman. Thus, security policy of each AD consists of *Elementary Policy Statements*. As it is stated in [11], an *Elementary Policy Statement* consists of an object component, a positive or negative access component and zero or more conditions. In our representation an object component can be a *file* (F), a *message* (M), an AD, a *port*, etc. On the other hand, positive access right component can be *Read*, *Write*, *Execute*, (for a file F), *Connect* (for a port), *MoveTo* (for an AD), *Send* (for a message M), etc. Negative access right components indicate that an entity is not to allowed for an action related to an access right (i.e.  $\neg Read$ ,  $\neg Write$ ,  $\neg Execute$ ,  $\neg Connect$ ,  $\neg MoveTo$ ,  $\neg Send$  ). Finally, the condition is the AD in which the entity, that wants to access the object, resides for the case when access right is Read/Write/Execute (a file F) or target AD to which an entity wants to send a message or wants to move . The sign " \* " is used denote any AD.

In Figure 1, security policy of AD1 contains elementary policy statement  $p_{1,1}$  which restricts the components of AD3 to read file F. However, the same security policy contains the elementary policy statements  $p_{1,2}$  and  $p_{1,3}$  which allow every component of AD1 to send message to the components of AD3 and to move to AD3, respectively. Since a component of AD1 can attach file F to the message it sends to a component in AD3, there is an inconsistency between  $p_{1,1}$  and  $p_{1,2}$ . Moreover, since  $p_{1,3}$  allows every component of AD1 to move to AD3 and thus S which owns file F can move to AD3 and become a member of AD3, there is also inconsistency between  $p_{1,1}$  and  $p_{1,3}$ . These are the examples of inconsistencies among the elementary policy statements of a security policy of a single AD and the basic operations that can be performed for the information contained in file F to be transmitted from one AD to another (i.e.  $G2(DataIn(F,G1))$ ) are as follows:

- A component in an AD downloads file F owned by a component of another AD (i.e.  $P(\Delta ReadIn(F,S) \wedge guardedby(G2)) \wedge S(guardedby(G1))$  ) to perform one of the file operations which are read, write and execute, respectively

- A component in an AD sends a message M, with file F attached to it, to a component in another domain (i.e.  $P(\Delta in(M,S) \wedge guardedby(G2)) \wedge S(guardedby(G1) \wedge attached(F,M))$  ).
- A component in an AD, which owns file F (i.e.  $S(owns(F))$  ), moves to another AD (i.e.  $G2(moving(S))$  ) and becomes the component of that AD.

All these above statements can be written as one specification in  $Mob_{adtl}$  as follows:

$$G2(\Delta DataIn(F,G1)) \text{ BECAUSE } (P(\Delta ReadIn(F,S) \wedge guardedby(G2)) \wedge S(guardedby(G1))) \vee (P(\Delta in(M,S) \wedge guardedby(G2)) \wedge S(guardedby(G1) \wedge attached(F,M))) \vee (G2(moving(S)) \wedge S(owns(F)))$$

We use  $ReadIn(F,S)$  because we assume that if a component has the right to execute a file (i.e.  $G1(allow(P, Execute, F))$  ) then it also has the right to write to it (i.e.  $G1(allow(P, Write, F))$ ) and any component which has the right to write to a file has the right to read it (i.e.  $G1(allow(P, Read, F))$ ). In addition, for a component to read (or write or execute) a file owned by another component, it should be allowed to access the ftp port by the security policy of AD where file owner resides (i.e.  $G1(guarding(S) \wedge allow(P, Connect, ftpPort))$ ). This is stated in axiom FS2 in Table 3. Similarly, if a component P receives the message M sent by another component S (i.e.  $P(\Delta in(M,S))$ ), it is required that the security policy of AD of which S is a component, allows S to send message M to P (i.e.  $G1(guarding(S) \wedge allow(S, Send, (M,P)))$ ). Axiom CS1 in Table 3 states this fact. Finally for a component S to move from one AD to another (i.e.  $G2(\Delta moving(S))$  ), it is required that security policy of AD, to which S belongs, allows S to leave (i.e.  $G1(guarding(S) \wedge allow(S, MoveTo, G2))$ ) as explained in axiom MS5 in Table 3.

In Figure 1, there is also an example of an inconsistency among two elementary policy statements which belong to security policies of different ADs. One can see that in the system shown in Figure 1 components in AD2 are allowed to read file F due the existence of elementary policy statement  $p_{1,4}$ . Once P reads file F owned by O, it owns a copy of file F and since elementary policy statement  $p_{2,1}$  of security policy of AD2 allows the components of AD3 to read any file owned by the components of AD2,  $p_{1,1}$  is violated by allowing the flow of information contained in file F from AD1 to AD3. In order to prevent such violation, axiom DF1 in Table 3 is added to the set of axioms.

## 4 Conclusion

In this paper, we formed a formal policy specification language for an 802.11 WLAN with enhanced security network adding axioms and making refinements to  $Mob_{adtl}$ [1] axioms. In the long run, a verification system that will employ the policy specification language formed in this work can be constructed. The verification system can be used to design security policies, which are for ADs inside the boundaries of the same domain (i.e. domain of a university, hospital, firm, etc.), free of security flaws.

## Acknowledgements

This work has been partially supported by Boğaziçi University Research Fund.

## References

1. Semprini, S.: Specification and Verification of Mobile Systems, PhD Thesis, Department of Informatics, University of Pisa, (2004)
2. Montangero, C., Semini, L.: Distributed States Temporal Logic, <http://arxiv.org/abs/cs.LO/0304046s>, (2003)
3. Montangero, C., Semini, L.: Distributed States Logic, Proceedings of Ninth International Symposium on Temporal Representation and Reasoning (2002), Manchester
4. Semprini, S.: Mark:  $Mob_{adtl}$  Reasoning Kit, <http://www.sra.itc.it/people/semprini>
5. Ferrari, G., Montangero, C., Semini, L., Semprini, S.: Multiple Security Policies in  $Mob_{adtl}$ , Proc. Workshop on Issues in the Theory of Security (2000)
6. Cardelli, L.: Mobility and Security, Lecture Notes for Marktoberdorf Summer School, (1999)
7. Levi, F., Sangiorgi, D.: Controlling Interference in Ambients, Proceedings of POPL (2000), 352–364
8. Bugliesi, M., Castagna, G.: Secure Safe Ambients, Proceedings of the 28<sup>th</sup> ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (2001), London
9. Bugliesi, M., Crafa, S., Castagna, G.: Boxed Ambients, Proceedings of Fourth International Symposium on Theoretical Aspects of Computer Software (2001), Sendai, Japan
10. Cuppens, F., Saurel, C.: Specifying a Security Policy: A Case Study, 9<sup>th</sup> IEEE Computer Security Foundations Workshop (1996), 123–134, Kenmare, Ireland
11. Ryutov, T., Neuman, C.: The Set and Function Approach to Modeling Authorization in Distributed Systems, Workshop on Mathematical Methods, Models and Architecture for Comp. Networks Security (2001), Russia
12. ANSI/IEEE 802.11 Standard, (1999)
13. IEEE P802.11i/D9.0 Standard, (2004)