A Formal Role-Based Access Control Model for Security Policies in Multi-Domain Mobile Networks

D. Unal\textsuperscript{a,}\textsuperscript{*}, M.U.Caglayan\textsuperscript{b,}\textsuperscript{**}

\textsuperscript{a}TUBITAK BILGEM, PO BOX 74, Gebze, Kocaeli, Turkey
\textsuperscript{b}Bogazici University TAM Research Center, Kandilli, Istanbul, Turkey

Abstract

Mobile users present challenges for security in multi-domain mobile networks. The actions of mobile users moving across security domains need to be specified and checked against domain and inter-domain policies. We propose a new formal security policy model for multi-domain mobile networks, called FPM-RBAC, Formal Policy Model for Mobility with Role Based Access Control. FPM-RBAC supports the specification of mobility and location constraints, role hierarchy mapping, inter-domain services, inter-domain access rights and separation of duty. Associated with FPM-RBAC, we also present a formal security policy constraint specification language for domain and inter-domain security policies. Formal policy constraint specifications are based on Ambient Logic and Predicate Logic. We also use Ambient Calculus to specify the current state of a mobile network and actions within security policies for evaluation of access requests according to security policies. A novel aspect of the proposed policy model is the support for formal and automated analysis of security policies related to mobility within multiple security domains.

Keywords: mobile network, multi-domain, security policy, access control, model checking

1. Introduction

The provision of services in networks with multiple administrative domains requires support for cross-domain security policy enforcement, management and verification. The multi-domain mobile network environment consists of multiple interconnected domains and mobile users, hosts and objects as sketched in Fig. 1. Inter-domain policies in such an environment need to support concepts such as mobility, inter-domain access rights, role mapping and separation of duty between domains.

An inter-domain security policy is based on a set of security agreements by participating organizations. The provision of inter-domain information sharing with mobile users call for an inter-domain policy model for mobile networks, which supports the concepts of inter-domain access rights,
role mapping, locations and mobility, as explained below:

1. **Inter-domain access rights** are access rights for roles of a foreign domain in a local domain and access rights for local roles when accessing from a foreign domain. These rights relate to inter-domain operations.

2. **Role mapping** maps user roles in one domain to another. e.g. a lecturer in one university may become a researcher in another.

3. **Locations and mobility** in multiple domains relate to object, host and user mobility across domains. A mobile user needs to be given access due to location and mobility constraints.

Current state-of-the-art in the area of multi-domain security policy management are mostly related to federated systems. The federated system approach [1, 2] requires a centralized knowledge of all system resources and multi-domain users, which are assumed to be static in the network. This approach is not suitable for multi-domain mobile networks where administration is distributed and also users and resources are mobile. Other studies such as [3, 4, 5, 6, 7, 8, 9] in the area of role based access control policies with location information is mostly based on location of users, not providing a general model for security policies in a multi-domain mobile network.

In this paper, we propose a new formal security policy model for multi-domain mobile networks, called FPM-RBAC, Formal Policy Model for Mobility with Role Based Access Control, for specification of domain and inter-domain security policies. The FPM-RBAC model is based on the well-known RBAC [10, 11] model. We use the RBAC model since roles provide flexibility in assignment and administration of permissions. In the context of multi-domain networks, roles are means of mapping permissions of a user in one domain to another domain. In the context of mobile networks, roles provide a user with the capability to carry all permissions associated with a role from one location to another. We augment the RBAC model by introducing services, inter-domain access rights, role hierarchy mapping, mobility and location constraints and separation of duty based on role mapping, locations and mobility.

Within the FPM-RBAC policy model, Ambient Logic [12, 13] is used to specify dynamic mobility and location constraints in security policy rules. Logical constructs based on Predicate Logic are used for specification of static constraints such as separation of duty. We use the Ambient Calculus [14] to specify the current state of a mobile network and actions within security policies for evaluation of access requests according to security policies. The matching of mobility and location constraints in policy rules is accomplished by checking the validity of Ambient Logic formulas against Ambient Calculus specifications based on the model checking algorithms presented in our previous work [15].

Our first contribution is the introduction of a formal inter-domain policy model for mobile networks. Second contribution is a process calculus based formal mobility model within security policies, capable of representing mobile network state as well as complex location and mobility constraints. The administration model is distributed, where policy rules for inter-domain access are defined by role hierarchy mapping between home, inter-domain and foreign roles. Therefore our model does not require the global knowledge of users and objects and does not introduce conflicts caused by inter-domain hierarchy mapping.

In this study, we present an example scenario where we demonstrate the concepts introduced above. In this scenario, we consider a university involved in a joint research project in the e-health area, with a hospital and an industrial partner. The project members have a need to access and share information both locally and remotely from different locations, possibly using mobile communication and computing devices. Roles in each individual domain may be mapped to another one through role mapping relations during the project. Policy rules for inter-domain access rights should be in place for accessing joint project information. Location and mobility of users and information also need to be restricted. An inter-domain security policy rule may state that database records from the hospital domain related to patients may not be accessible from the university or industrial partner domain and may not be written to the university domain.

Section 2 summarizes the related work. In Section 3, we present the formal role-based security policy model called FPM-RBAC. In Section 4, we present the verification of FPM-RBAC security policies by model checking. In Section 5, we present an algorithm for giving the permission or denial decisions for requested actions with respect to FPM-RBAC security policy specifications. Section 6 summarizes our current and future research work.
2. Related Work

Ambient calculus, proposed by Cardelli and Gordon [14], is a process calculus which is able to theorize about concurrent systems that include mobility and locations. Modal logics are used for expressing properties of models which cannot be expressed by the constructs of calculi. Ambient logic [12, 13] is a modal logic for expressing spatial and temporal properties of Ambient Calculus. All spatial and temporal constructs of Ambient Calculus are reflected in Ambient Logic. The main differences of Ambient Logic from latter logics are introduction of more expressive space modalities and simpler temporal connectives [16].

Reasoning about spatial configurations for application level security policies in ubiquitous environments is investigated by Scott [17]. BACI_R [18] is a boxed ambient calculus with RBAC mechanisms used to define access control policies for ambients. ACPEG proposed by Zhang et al. [19] is a tool for evaluating and generating access control policies based on first-order logic. In [20] a generic security policy specification language is used to express rules and manipulate locations of mobile code. There are various model checking algorithms [21, 22] for Ambient Calculus. In our previous work [15], we presented a model checking algorithm for Ambient Calculus.

There are various access control models that include temporal, location and other context-based spatial and temporal constraints based on the RBAC model. TRBAC by Bertino et al. [23] and GTRBAC by Joshi et al. [24], an extension of TRBAC, are temporal RBAC models. Some of the examples of access control models which include location and context-based constraints are, SRBAC by Hansen and Oleshchuk [3], LOT-RBAC by Chandran and Joshi [4], Spatio-Temporal RBAC by Ray and Toalchoodee [5], STRBAC by Kumar and Newman [6], GEO-RBAC by Damiani et al. [7], Context Aware RBAC by Kulkarni and Tripathi [8] and ESTARBAC by Aich et al. [9] and Mondal and Sural [25].

Former studies in the area of location based RBAC are more suitable for environments in which the location of users and objects are static in nature. The former models address locations, however they do not address mobility, which is a fundamental concept for application to mobile networks. Locations are defined with first-order relations over a set of logical locations. Some formalism is provided to specify complex spatial configurations; but no syntax or semantics are available for mobility of users or objects. The impact of role assumption, enabling, activation and permission assignment due to mobility are not considered. Additionally the prior location based RBAC models do not consider inter-domain aspects, such as role-mapping and inter-domain access. For the reasons discussed above, former models are not well-suited for multi-domain mobile networks.

Separation of duty (SOD) constraints based on time and location have been discussed in the Spatio-Temporal RBAC model [5]. This study presents a concept for SOD similar to our framework. However, due to the shortcomings in the location model, the expressiveness of SOD constraints are limited. First of all, location based SOD constraints in this study does not support location expressions or formulas and they are limited to a single location. In contrast, the FPM-RBAC model supports location formulas, which reflect the dynamic mobility aspects by representing the entire mobile network state. Second, sessions are statically bound to locations. When the location of a user is changed after logging into a session, the evaluation result of a SOD constraint may become invalid. In the FPM-RBAC model, this kind of SOD constraint is provided through Service Location based SOD. Third, location constraints are defined through predicates that need to be re-evaluated with each change in locations, which is not efficient in a mobile environment. In the FPM-RBAC model, location based SOD constraints are evaluated by spatio-temporal model checking, which generates all the possible location configurations based on actions that result in mobility. Finally, the model of Ray et al. does not support multiple domains, whereas in FPM-RBAC, SOD constraints based on inter-domain mappings are supported.

3. FPM-RBAC: Formal Policy Model for Mobility with Role Based Access Control

FPM-RBAC is a multi-domain access control model based on the concepts of domain and inter-domain policies and represents both static and dynamic aspects of security policies in multi-domain mobile networks. The presented security policy model supports the specification of Role-Based Access Control (RBAC) policies with Role Hierarchy, Object Type Hierarchy and Role Mapping in a multi-domain setting. We define a formal model for
network state related to locations and provide the mapping of actions to the formal model. Specification of location and mobility constraints as well as Separation-of-Duty (SOD) constraints is supported.

3.1. Overall Structure of FPM-RBAC

FPM-RBAC components are, a location and mobility model, a domain security policy model, an inter-domain security policy model, and a separation of duty model. The representation of location and mobility aspects for objects, users, hosts and domains is necessary in a multi-domain mobile network. Location and mobility model of FPM-RBAC is related with representation of location and mobility constraints in the security policy rules as well as the representation of actions in the security policy rules and the state of the network with respect to locations. The location and mobility model of FPM-RBAC is presented in Section 3.2.

The domain policy model of FPM-RBAC introduces the concepts of domain, service and Object Type Hierarchy to the RBAC model. The domain security policy model is presented in Section 3.3. An inter-domain security policy includes security requirements for mobility, role mapping and inter-domain access rights among multiple domains. The inter-domain security policy model is presented in Section 3.4.

Separation of Duty (SOD) constraints are related to assignment and activation of roles. In FPM-RBAC, we introduce additional classes of SOD constraints: (i) Service based SOD, (ii) Inter-Domain SOD and (iii) Location and Mobility Based SOD. SOD constraints in FPM-RBAC are presented in Section 3.5. The introduced SOD relation types in FPM-RBAC support specification of SOD relations for service-based multi-domain mobile networks.

3.2. Location and Mobility Model

Location and mobility model of FPM-RBAC is related with the representation of location and mobility constraints in the security policy rules as well as the representation of actions in the security policy rules and the location configuration of the system. Due to the dynamic nature of mobility, in mobile networks the location (spatial) configuration of the system changes with actions. Ambient Calculus is used for representation of actions in the security policy rules and the location configuration of the system. The location and mobility constraints in the security policy rules are specified using Ambient Logic. We provide a very brief overview of Ambient Calculus and Ambient Logic in the following section.

3.2.1. Ambient Calculus and Ambient Logic

For specifying dynamic properties of a mobile network and modeling the spatial and temporal effects of actions, we need to use an executable process calculus. We use the ambient calculus because it provides natural means to model location hierarchies and mobility. The fragment of ambient calculus used in this study is presented in Table 1. The semantics of ambient calculus is based on the structural congruence relation.

Properties of Ambient Calculus processes can be analyzed in two ways, spatial properties and temporal properties. The notion of Ambient is the basic element of the spatial properties of processes. Ambients are bounded places identified by a name where processes reside inside or outside. Ambients can be nested in other ambients. This provides an hierarchical organization of locations.

Temporal constructs of process algebras represent the change of processes over time. The ambient calculus provides temporal constructs related to mobility in addition to communication primitives. The main temporal constructs for modeling entrance, exit and dissolution are expressed as \( \text{in} n, \text{out} n, \text{open} n \) respectively. Sequential execution is represented by a \textit{path}, parallel execution is represented by the parallel operator \((\|)\). A specification in the form of \(a[\text{in} n. \text{out} z.0]\) represents an ambient \(a\) which will enter ambient \(n\), exit ambient \(z\) and stop. Communication primitives enable processes within the same ambient to exchange messages.

Ambient logic has temporal and spatial modalities in addition to propositional logic elements. Semantics of the connectives of the ambient logic are defined through satisfaction relations. The satis-

<table>
<thead>
<tr>
<th>Process (P,Q)</th>
<th>Capability (M)</th>
<th>Inactivity (0)</th>
<th>Name (x)</th>
<th>In (M)</th>
<th>Can enter (n)</th>
<th>Out (M)</th>
<th>Can exit (n)</th>
<th>Open (M)</th>
<th>Can open (M)</th>
<th>Null ((\epsilon))</th>
<th>Path (M,M)</th>
</tr>
</thead>
<tbody>
<tr>
<td>P (</td>
<td>Q</td>
<td>) composition</td>
<td>M[P] ambient</td>
<td>in M can enter M</td>
<td>out M can exit M</td>
<td>open M can open M</td>
<td>(\epsilon) null</td>
<td>M,M path</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The ambient logic provides two main spatial and temporal constructs. The \( \text{Somewhere} \) connective, \( \exists \), is used for specifying nesting properties of processes. The formula \( \diamond \alpha \) is satisfied by processes which satisfies \( \alpha \) in some inner location. The \( \text{Sometime} \) connective, \( \lozenge \), is used for specifying temporal behavior of the processes on the basis of reduction relations (\( \rightarrow \)). Fragment of ambient logic used in this paper is shown in Table 2.

<table>
<thead>
<tr>
<th>( \eta )</th>
<th>a name ( n )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \mathcal{A} ), ( \mathcal{B} ), ( \mathcal{C} ) :=</td>
<td>composition</td>
</tr>
<tr>
<td>( T )</td>
<td>true</td>
</tr>
<tr>
<td>( \neg \mathcal{A} )</td>
<td>negation</td>
</tr>
<tr>
<td>( \mathcal{A} \lor \mathcal{B} )</td>
<td>disjunction</td>
</tr>
<tr>
<td>( \lozenge \mathcal{A} )</td>
<td>sometime</td>
</tr>
<tr>
<td>( 0 )</td>
<td>void</td>
</tr>
</tbody>
</table>

Table 2: Fragment of Ambient Logic used in the study

\( \eta \) a name \( n \)
\( \mathcal{A} \), \( \mathcal{B} \), \( \mathcal{C} \) := composition

\( T \) true
\( \neg \mathcal{A} \) negation
\( \mathcal{A} \lor \mathcal{B} \) disjunction
\( \lozenge \mathcal{A} \) sometime
\( 0 \) void somewhere

3.2.2. Representation of Location and Mobility in FPM-RBAC

The mobility model for a multiple domain mobile network consists of four basic elements: administrative domains, hosts, users and objects. We assume that all the elements reside in a system element called the World. In this model the locations and mobility of hosts, users and objects are formalized as ambient calculus processes and they have the following mobility capabilities:

- **Hosts**: Moving into a domain represents connecting a host to a domain. Moving out represents disconnecting.
- **Users**: If a user moves into a host, the user is logged into that host. If a user moves out of a host, the user is logged out. If the user moves into/out of a domain, this represents enrollment, logging into a domain or movement between domains depending on the context.
- **Objects**: Every object must reside in a host when not on the move. The movement of objects from one host to another represents communication.

In the formal specification, domains, hosts, users and objects are modeled as Ambients. We present some examples of object, host and user mobility specification as ambient calculus processes. Some known notation conventions are utilized, for example \( n[\cdot] \) means \( n[0] \). The symbol \( \rightarrow \) represents the reduction relation and \( \rightarrow^* \) represents a series of reductions.

**Object Mobility**: File \( f_1 \) is moved from directory \( \text{dir}_1 \) in host \( h_1 \) to directory \( \text{dir}_2 \) in host \( h_2 \).

\[
d_1[h_1[\text{dir}_1]\text{out}\ h_1;\text{in}\ h_2;\text{in}\ \text{dir}_2.f_1[]]] \rightarrow^* d_1[h_1[\text{dir}_1[]]h_2[\text{dir}_2.f_1[]]]
\]

**Host Mobility**: Portable host \( h_1 \) moves from Domain \( d_1 \) to Domain \( d_2 \).

\[
d_1[h_1[\text{out}\ d_1;\text{in}\ d_2.0][]d_2[]] \rightarrow^* d_1[[]]d_2[h_1[[]]]
\]

**User Mobility**: User \( u_1 \) logs into Host \( h_1 \), which contains File \( f_1 \).

\[
u_1[\text{in}\ h_1.0]h_1[f_1[]] \rightarrow h_1[u_1[]f_1[]]
\]

An example mobile network specification in Ambient Calculus is presented in (1). In this example there are two domains, \( d_1 \) and \( d_2 \). Mobile user \( u_2 \) has the right to move into either of the two domains, login to \( h_1 \) and \( h_2 \) and read files \( f_1 \) and \( f_2 \). The formal representations of multiple actions are combined with the parallel (\( | \) ) operator since any of these actions may be exercised independent from each other.

\[
LCONF = \begin{align*}
d_1[u_1[]h_1[f_1[\text{data}_1[\text{in}\ u_2.0]\text{out}\ u_2.0][]]] \\
|d_2[u_2[\text{out}\ h_2.0]\text{out}\ d_2.0][\text{in}\ d_1;\text{in}\ h_1.0] \\
|\text{out}\ h_1;\text{out}\ d_1.0][\text{in}\ d_2;\text{in}\ h_2.0][\text{in}\ f_1.0][\text{in}\ f_2.0] \\
|\text{out}\ f_1.0][\text{out}\ f_2.0][\text{data}_2[]]
\end{align*}
\]

3.2.3. Location Configuration of the System

The location configuration of the system \( LCONF \) holds the ambient calculus process specification for the current system. The initial state for a user is before a user logs into a service. In the initial state, \( LCONF_0 \) is derived from the initial state of the mobile network and the security policy specification. The domains, users, hosts and objects in the system and their locations are derived from defined data sets. The set of possible actions are derived from the security policy. When an action is requested by a user, the action is matched to its formal representation, and if allowed by the security policy, it is executed and the location configuration is updated accordingly. The state change between different location configurations takes place with a sequence of actions. Here is an example for modifying the location configuration with actions. The location configuration of the system at time \( T_0 \) and \( T_1 \) are depicted in Figure 2.

At the initial state the ambient calculus specification is as follows:

5
3.3. Domain Security Policy Model

A security administrative domain, or a domain in short, is a logical boundary for an information and communication system governed by a single administrative authority. A domain is defined by the data sets and relations associated with an administrative boundary. A security policy is defined by a set of authorization terms. Authorization terms include authorization subjects, authorization objects, actions and constraints. Services provided by the domain to its internal and external users are also included in the authorization model. Services are associated with a subset of authorization objects and actions. The set of enabled roles for a service is defined by the service access relation. For each service, a permission assignment relation is specified for association of enabled roles for the service with permissions. Constraints are specified by formal languages. Predicate calculus is used for the specification of generic constraints named as conditions and Ambient Logic is used for the specification of location and mobility constraints named as location formula, or formula in short. The single domain security policy model of FPM-RBAC is depicted in Figure 3. In the following sections we present data sets, domains, services, actions, constraints, relations, hierarchies of the domain security policy model.

$$LCONF_0 = D_1[u_1][h_1[f_1[dt_1[in u_2, 0][out u_2, 0]]]$$
$$D_2[h_2[u_2][out h_2, 0][out D_2, 0][in D_1, in h_1, 0][out h_1, out D_1, 0][in D_2, in h_2, 0][in f_1, 0][in f_2, 0][out f_1, 0][out f_2, 0][h_2][f_2[dt_2]]]$$

In Step (1), User $u_2$ logs out of Host $h_2$ with ambient calculus capability $u_2[out h_2...]:$

$$LCONF_1 = D_1[u_1][h_1[f_1[dt_1[in u_2, 0][out u_2, 0]]]$$
$$D_2[u_2[out D_2, 0][in D_1, in h_1, 0][out h_1, out D_1, 0][in D_2, in h_2, 0][in f_1, 0][in f_2, 0][out f_1, 0][out f_2, 0][h_2][f_2[dt_2]]]$$

3.3.1. Data sets

The data sets in the FPM-RBAC access control model is specified using First Order Set Theory. An Authorization Subject is an active entity that may conduct an Action on an Authorization Object. Since the RBAC model is adopted, roles are authorization subjects, however specification of users in this context are also supported. An authorization object is an entity upon which an action is conducted. An authorization object may be a domain, host, object or object type.

Where the constants $\eta, \upsilon, \rho, \beta, \nu, \sigma$ respectively denote the number of hosts, users, roles, objects, object types and services, the following data sets are defined:

$$H = \{h_1, h_2, ..., h_\eta\}$$  
$$U = \{u_1, u_2, ..., u_\upsilon\}$$  
$$R = \{r_1, r_2, ..., r_\rho\}$$  
$$O = \{o_1, o_2, ..., o_\sigma\}$$  
$$T = \{t_1, t_2, ..., t_\upsilon\}$$  
$$AS = U \cup R$$  
$$AO = O \cup T \cup H \cup \Delta$$

In the joint project scenario, three organizations are working on a joint e-health research project. Regarding this scenario, example definitions for data sets of University A (UniA), Commercial Company B (CorpB), and Hospital C (HosC) related to the joint project case are given below.

$$\Delta = \{UniA, CorpB, HosC\},$$
$$H_{UniA} = \{h_{11}, h_{12}, h_{13}\},$$
$$H_{CorpB} = \{h_{21}, h_{22}\}, H_{HosC} = \{h_{31}, h_{32}\},$$
$$H = H_{UniA} \cup H_{CorpB} \cup H_{HosC};$$
$$U_{UniA} = \{nmullis, jfrantz, cmiele\},$$


\[ U_{CorpB} = \{ \text{damentiola, mrundell} \}, \]
\[ U_{HosC} = \{ \text{fmcbride, aweathers} \}, \]

\[ U = U_{UniA} \cup U_{CorpB} \cup U_{HosC}, \]
\[ O = \{ \text{jrapp, unif, jrep, prd} \} \]
\[ T = \{ \text{Obj, App, File, Db, Message, Portable} \} \]

\[ R_{UniA} = \{ \text{Member, Teaching, Research, Admin, } \] Lecturer, Faculty, RAAssist, SysAdmin} \]
\[ R_{CorpB} = \{ \text{Member, Engineer, Research, Mgr, RnDEng, SwEng, PrjMgr} \}, \]
\[ R_{HosC} = \{ \text{Member, Medical, Doctor, SocialSec} \}, \]

The elements of the set of hosts represent the following in our scenario: \( h_{11} \): Research lab client of \( \text{jfrantz} \), \( h_{12} \): Application server used for joint research project of University A, \( h_{13} \): The portable PC of \( \text{nmullis} \), \( h_{21} \): The portable PC of \( \text{mrundell} \), \( h_{22} \): The file server of CorpB, which holds joint project reports, \( h_{31} \): File server of Hospital C containing patient records, \( h_{32} \): The portable PC of \( \text{fmcbride} \). The elements of the set of objects represent the following in our scenario: \( \text{jrapp} \): Joint research project web application, \( \text{unif} \): Joint research project work file, \( \text{jrep} \): Joint research project project report file, \( \text{prd} \): Patient health records database.

### 3.3.2. Domains

A domain is defined by a set of hosts, users, objects, roles and object types associated with that particular domain. Each domain is associated with a domain administrator. A Domain \( D \) is defined as

\[ D = \{ H, U, O, R, T \}. \]

When there are multiple domains in the network, \( \delta \) represents the number of domains and \( \Delta \) defines the set of all domains in the network. In this case, the set of hosts, users, roles, objects, object types, authorization subjects and authorization objects associated with domain \( D_i \) are represented respectively with \( H_i, U_i, R_i, O_i, T_i, AS_i, AO_i \). The set of all domains is defined as

\[ \Delta = \{ D_1, D_2, ... D_\delta \}. \]

### 3.3.3. Services

The FPM-RBAC model includes a specific construct for services, which binds the dynamic user-role association to a subset of resources and actions. The decision to introduce services arises from the service-oriented nature of multi-domain networks with heterogeneous and interconnected services such as web services and network services. In FPM-RBAC, the concept of services is built on the W3C Web Services Architecture [26]. The W3C Web Service Glossary defines services as follows [27]: "A service is an abstract resource that represents a capability of performing tasks that form a coherent functionality from the point of view of providers entities and requesters entities". More specifically, a service is a resource that represents a person or organization in some collection of related tasks. A task is a set of actions associated with a service. The difference of a service from other kind of resources is that it is associated with a set of actions.

In the RBAC model, resources are regarded as a set of objects, which are passive entities. However, for supporting access control for services, there is a need for a construct that defines the functionality provided by resources. For example, a joint project service may be associated with actions on the set of resources which are utilized towards providing a functionality, such as shared access to joint project documents.

A policy is scoped by a domain, applies to a resource and constrains actions which target resources. For a domain service, the scope of the policy for the service is a single domain. For inter-domain services, the scope of the policy may be multiple domains. For example, the access control policy for a joint project service for the university applies to the resources within the university which provides the joint project service, such as a web application for editing shared documents. The policy constrains actions upon the resource, such as writing to a specific document.

A service in the FPM-RBAC model is associated with a set of resources. The set of resources associated with a specific service \( S_j \) is denoted as \( P_j \). \( P_j \) is a subset of authorization objects and it includes hosts, objects and object types. The relation called Service Resources (SR), includes the set of accessible resources by a service for provision of a capability. SR associates services with resources. The set of hosts associated with a service represents logical locations that a role may be enabled. The subset of objects and object types associated with a service are included in the Permission Assignment Matrix related with the service. A service is also associated with a set of actions. This set is called tasks. The Service Tasks (ST) relation maps services to tasks.

The set of Services \( \mathcal{V} \) defines the services within a single domain. If there are multiple domains, the services within a domain \( D_i \) is represented as \( \mathcal{V}_i \). The set of services \( \mathcal{S} \) and the set of Domain Services
The set of services is $S = \{S_1, S_2, \ldots, S_k\}$. The set of domain services is $S', V = V_1 \cup \ldots \cup V_8$. The set of resources for a specific service $S_j$ is $P_j \subset AO$. $P_j = H_j \cup O_j \cup T_j$ where $H_j^i = \{h_k, \ldots, h_l\}$ are hosts associated with service $S_j$, $H_j^i \subset H_i$, $O_j^i = \{o_m, \ldots, o_n\}$ are objects associated with service $S_j$, $O_j^i \subset O_i$, $T_j^i = \{t_x, \ldots, t_y\}$ are object types associated with service $S_j$, $T_j^i \subset T_i$. The set of tasks for a specific service is $K_j \subset \text{ACT}$. The Service Resources relation for a specific service $S_j$ is $SR \subset S \times P_j$. The Service Tasks relation for a specific service $S_j$ is $ST \subset S \times K_j$.

The activation of roles in FPM-RBAC Access Control Model is achieved through the use of Service Sessions. The concept of session in the original RBAC model represents a dynamic binding of users to a set of roles. Sessions are not sufficient to express the use of services. We introduce the concept of a service session, which is a ternary relationship between subjects and objects specified in the high-level specifications of security policy. Here we present template encodings for translation of actions to Ambient Calculus. The encoding is for a single domain. This list is not exhaustive since alternative encodings for the same actions in different contexts are possible.

**Definition 2.** Ambient calculus encoding for actions in security policy rules are specified as follows. In this encoding, $\alpha_{as, ao}$ represents an action with authorization subject as and authorization object $ao$. Where $a$, $newz \in U \cup H, u, u_1, u_2 \in U, d \in \Delta, h, h_1, h_2 \in H, l \in H \cup \Delta, m, prog, file, f, data \in O, and M, P$ are process specifications.

- $\text{Enrol}_{u,d} \triangleq \text{newz[in \ d.z]} | d[\text{open newz}].0$
- $\text{Login}_{u,l} \triangleq u[\text{in } l] | l[\text{in } 1]$
- $\text{Logout}_{u,d} \triangleq l[u[\text{out } l]]$
- $\text{Send}_{u,m} \triangleq d[h_1[u_1[m][M][\text{out } u_1[\text{out } h_1,in \ h_2,\text{in } u_2,0][h_2[u_2]]]]$
- $\text{Receive}_{u,m} \triangleq u[\text{open } m.(m.)[m][M]]$
- $\text{Execute}_{u,prog} \triangleq d[h[u[\text{open } prog] | prog[\text{in } u[P]]]$
- $\text{Read}_{u,\text{file}} \triangleq h[\text{file}[\text{data}[\text{in } u.0]]] | u[\text{in file.0}]$
- $\text{Write}_{u,\text{file}} \triangleq h[\text{file}[T] | u[\text{in file.data[\text{out } u.0]]}]$
- $\text{Delete}_{u,\text{file}} \triangleq h[\text{file}[u] | u[\text{open file.0}]]$
- $\text{Create}_{u,\text{file}} \triangleq h[u[\text{open f.f[\text{file}[\text{out } f.0]]}]$

### Constraints

A constraint determines the applicability of a security policy rule. The set of constraints is of the form $C = \{(co, fo) : co \in PL, fo \in AL\}$ where $PL$ denotes the language of predicate logic and $AL$ denotes the language of Ambient Logic. There are two types of constraints. Conditions $(co)$ define the generic constraints for a rule to be applicable. Location formula $(fo)$ defines the location and mobility constraints of the mobile network that must be satisfied by a security policy rule.

Generic constraints are logical pre-requisites for a rule to be applicable and are defined using predicate logic. They are specified by a predicate logic formula that defines the constraints and relationships between subjects and objects specified in the policy rules. In cases where rules are mentioned in a rule, generic constraints may apply to user-role assignment.

We define the following pre-defined predicates for

\[ V_i \text{ within a domain } D_i \text{ are defined as follows.} \]

\[ \textbf{Definition 1.} \text{ The set of services is } S = \{S_1, S_2, \ldots, S_k\}. \text{ The set of domain services is } S' = V_1 \cup \ldots \cup V_8 \text{. The set of resources for a specific service } S_j \text{ is } P_j \subset AO. P_j = H_j \cup O_j \cup T_j \text{ where } H_j^i = \{h_k, \ldots, h_l\} \text{ are hosts associated with service } S_j, \text{ and } H_j^i \subset H_i \text{. The set of tasks for a specific service is } K_j \subset \text{ACT}. \text{ The Service Resources relation for a specific service } S_j \text{ is } SR \subset S \times P_j. \text{ The Service Tasks relation for a specific service } S_j \text{ is } ST \subset S \times K_j. \]

3.3.4. Actions

The set of actions defines the operations by subjects on objects. For this study we assume a fixed set of actions $A = \{\text{Enrol, Login, Logout, Execute, Read, Write, Send, Receive, Delete, Create, Manage}\}$. The action Manage is an administrative action relating to the policy system. The set of permissions $P = \text{ACT} \times AO$ defines all possible actions on authorization objects for a given system.

The actions may also be signed. Positive authorizations are considered default and positive sign may be omitted within specifications. The following constructs are used to specify signed actions:

- **Signs**: $N = \{+, -\}$ represents permission or denial.
- **Signed Actions**: $\text{ACT} = N \times A$ represents permission or denial of an action. $(+, \text{read})$ denotes that read action is permitted.

The actions are modeled as Ambient Calculus capabilities as shown in Definition 2. The translation of actions in the security policy rules to Ambient Calculus is on a formal mapping. The translations are encoded as a template and are based on inference of specific subject and object names from the high-level specifications of security policy. Here we present template encodings for translation of actions to Ambient Calculus. The encoding is for a single domain. This list is not exhaustive since alternative encodings for the same actions in different contexts are possible.
definition of generic constraints. Additional predicates may be defined by the domain administrator.

1. **EnrolledDomainUser** \((D_i, u_j)\), where \(D_i \in \Delta, u_j \in U\), specifies whether a user \(u_j\) has been enrolled (registered) to domain \(D_i\). Abbreviated as \(EDU(D_i, u_j)\).

2. **EnrolledDomainHost** \((D_i, h_k)\), where \(D_i \in \Delta, h_k \in H\), specifies whether a host \(h_k\) has been enrolled (registered) to domain \(D_i\). Abbreviated as \(EDH(D_i, h_k)\).

3. **ActiveDomainUser** \((D_i, u_j)\), where \(D_i \in \Delta, u_j \in U\), specifies whether user \(u_j\) has been logged into domain \(D_i\). Abbreviated as \(ADU(D_i, u_j)\).

4. **RoleAssigned** \((u_j, r_l)\), where \(u_j \in U, r_l \in R\), specifies whether user \(u_j\) has rights to assume role \(r_l\). Abbreviated as \(RSG(u_j, r_l)\).

5. **RoleAssumed** \((u_j, r_l)\), where \(u_j \in U, r_l \in R\), specifies whether user \(u_j\) has actively assumed role \(r_l\). Abbreviated as \(RAS(u_j, r_l)\).

6. **RoleEnabled** \((r_l, S_m)\), where \(S_m \in V, r_l \in R\), specifies whether role \(r_l\) is enabled for service \(S_m\). Abbreviated as \(REN(r_l, S_m)\).

7. **DescendantRole** \((r_l, r_k)\), where \(r_l, r_k \in R\) specifies whether a given role is \(r_l\) descendant (or specialization) of another given role \(r_k\). Abbreviated as \(DR(r_l, r_k)\).

8. **ObjectIsType** \((o_n, t_m)\), specifies whether an object \(o_n \in O\) identified by its object name is of a given object type \(t_m \in T\). Abbreviated as \(OIT(o_n, t_m)\).

9. **Administrator** \((u_j, D_i)\), where \(u_j \in U, D_i \in \Delta\) specifies whether user \(u_j\) has administrative rights over domain \(D_i\). Abbreviated as \(ADM(u_j, D_i)\).

The location constraints in the security policy rules are specified using the Ambient Logic. The location formula \(fo\), which represents a location and mobility constraint, is a modal logic formula with spatial and temporal constructs. It is a formula in spatial logic that includes names of domains, authorization subjects and authorization objects. The Ambient Logic is used since it is possible to represent both time and location. We note here that the location in our model is a logical concept that shows the relative location of objects and users to hosts and domains rather than their physical location.

If the location formula contains the Somewhere \((\diamond)s\), parallel \((\|)p\) and ambient \((\langle\rangle)a\) formalizations of ambient logic, it is referred as a *location constraint*. Additionally, if it contains temporal constructs which are Sometime \((\diamond)s\) and Everyday \((\bigcirc)e\) modalities in ambient logic, it is referred as a *mobility constraint*. Location constraints are evaluated by spatial model checking, whereas mobility constraints are evaluated by a further step of temporal model checking.

As an example to location and mobility constraints, we present the location formula in (2) associated with the ambient calculus specification \(LCONF\) presented in (1). This is an ambient logic specification stating that Domain \(d_2\) should never contain data1 and data2 at the same time. Combined with specification \(LCONF\) which specifies that data1 resides in \(d_1\) and data2 resides in \(h_2\) inside \(d_2\), the interpretation of this mobility constraint is that domain \(d_2\) data should not be copied to domain \(d_1\).

\[
fo = \Box(\neg\diamond\{\diamond d_2[\diamond\{data1[T] \| data2[T]\}]\} \| \top)
\]  

(2)

The possibility of conflicts arising from conflicting actions may be resolved using the theorem proving approach, as presented in our previous work in [28]. Using the formalization methodology described above, some example security policy definitions with location constraints are presented below. The authorization term structure, which is the basis of the security policy rules, will be introduced in Section 3.3.8.

- All users can read files in folder `Project_Folder`, if they are in a location that contains this folder: \((as = user, ao = Project_Folder, sa = + read, fo = o(as\| ao), co = user \in U \cap OIT(ao, Project_Folder))\)

- All users can send `E-mail` between the `UniA` and `CorpB` domains: \((as = user, ao = E-mail, sa = + send, fo = UniA [\diamond ao]\| CorpB [\diamond ao]\) \lor CorpB [\diamond ao]\| UniA[ao]], co = user \in U \cap OIT(ao, E-mail))

3.3.6. Relations and System Functions

Relations in the FFM-RBAC model are specified in Table 3. The `HD` relation maps hosts to domains, whereas `UD` relation maps users to their
Table 3: Relations in the FPM-RBAC model.

<table>
<thead>
<tr>
<th>Relation Name</th>
<th>Relation</th>
<th>Specification</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>HD</td>
<td>$H \times \Delta$</td>
<td>$HD(h_k, D_i)$</td>
<td>$h_k$ is enrolled to Domain $D_i$.</td>
</tr>
<tr>
<td>UD</td>
<td>$U \times \Delta$</td>
<td>$UD(u_r, D_i)$</td>
<td>$u_r$ is enrolled to Domain $D_i$.</td>
</tr>
<tr>
<td>OD</td>
<td>$O \times \Delta$</td>
<td>$OD(o_n, D_i)$</td>
<td>$o_n$ is within Domain $D_i$.</td>
</tr>
<tr>
<td>UA</td>
<td>$U \times R$</td>
<td>$UA(u_j, r_l)$</td>
<td>Role $r_l$ assigned to the user $u_j$.</td>
</tr>
</tbody>
</table>
| PA            | $R \times ACT \times AO$ | $PA(r_l, act, ao)$ | $r_l$ has permission $(act, ao)$.
| SA            | $R \times V$ | $SA(r_l, S_m)$ | $r_l$ is enabled for service $S_m$. |

The Service Access Matrix (SAM) binds roles to home domains. The $UA$ relation specifies the assignment of users to roles. The Permission Assignment ($PA$) relation specifies the assignment of roles to permissions. Finally, the Service Assignment ($SA$) relation specifies the set of enabled roles for services.

The $PA$ and $SA$ relations are specified in terms of matrices. The dynamic binding of users to roles and permissions is achieved through services. Permissions are assigned to roles based on PAMs defined for each service and authorization object type.

The Service Access Matrix (SAM) binds roles to services. When a user logs into a service, the roles associated with that service are enabled. Then the role may act only on the subset of authorization objects referred by the service. The SAM matrix denotes whether role $r_l$ is allowed to use service $S_j$.

**Definition 3.** The Service Access Matrix is $SAM : R \times V$ where $SAM[i, j] = True$ | False, $1 \leq i \leq \rho, 1 \leq j \leq \sigma$. $SAM[i, j] = True \rightarrow SA(r_l, S_j)$.

A Permission Assignment Matrix (PAM) is defined for each service, which also includes Location and Mobility constraints. This design decision simplifies the specification of permissions for a network with multiple domains and different types of services. PAM binds roles to permissions. Permissions are tuples of the form $(act, ao)$ where $act \in ACT, ao \in AO$. For each service there is a PAM that includes permissions for that service. $PAM[i, j, k]$ determines whether Role $r_l$ is allowed to conduct Action $act_j$ on Authorization Object $ao_k$.

**Definition 4.** The Permission Assignment Matrix is $PAM : R \times ACT \times AO$ where $PAM[i, j, k] = True$ | False, $1 \leq i \leq \rho, 1 \leq j \leq |ACT|, 1 \leq k \leq |AO|$. $PAM[i, j, k] = True \rightarrow PA(r_l, act_j, ao_k)$.

The system functions in the FPM-RBAC model are defined in line with the standard RBAC model [10, 11]. Additional functions are provided due to the introduction of services and object type hierarchies. The functions in FPM-RBAC are presented in Table 4.

### 3.3.7. Role and Object Hierarchies in FPM-RBAC

Hierarchies in the FPM-RBAC model consist of Role Hierarchy (RH) and Object Type Hierarchy (OTH). The precedence relationship of roles are defined in the Role Hierarchy (RH) relation. The Role Hierarchy is compatible with the Role-Based Access Control [10, 11] model. The role hierarchy for a domain $D_i$ is represented as $RH_i$.

The mapping of objects to object types are defined in the Object Type Hierarchy (OTH) relation. Object-Type Hierarchies were introduced by Jajodia et al. in the Flexible Authorization Framework (FAF) [29]. In a multi-domain setting, the knowledge of objects across domains is not desirable because of distributed administration. Through the use of Object Types the permissions may be mapped across multiple domains without the need of global knowledge of individual objects. The object hierarchy for a domain $D_i$ is represented as $OTH_i$.

**Definition 5.** Hierarchy. A hierarchy is a triple $(A, B, \leq)$ where

1. $A$ and $B$ are disjoint sets;
2. $\leq$ is a partial order on $A \cup B$ s.t. every element $a \in A$ is said to be minimal in $A \cup B$. Minimal elements have no elements below themselves in the hierarchy, i.e. $a \in A$ and $\forall b \in A \cup B, b \leq a \Rightarrow b = a$.

For the OTH relation, $a \leq b$ means that object $a$ is of object type $b$. For the RH, $a \leq b$ means that role $a$ is a specialization of role $b$.

**Definition 6.** Object Type Hierarchy. $A=O, B=T$ $a \leq b \Rightarrow (a, b) \in OTH$ where $a \in O, b \in T$. 


Table 4: System functions in the FPM-RBAC model.

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Specification</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>services</td>
<td>$U \rightarrow 2^V$</td>
<td>Gives the services accessible by a user.</td>
</tr>
<tr>
<td>service_users</td>
<td>$V \rightarrow 2^U$</td>
<td>Maps each service to its users.</td>
</tr>
<tr>
<td>enabled_roles</td>
<td>$V \rightarrow 2^R : { r \in R</td>
<td>(v, r) \in SA }$</td>
</tr>
<tr>
<td>enabled_roles*</td>
<td>$V \rightarrow 2^R : { r \in R</td>
<td>(v, r) \in SA }$</td>
</tr>
<tr>
<td>assigned_users</td>
<td>$R \rightarrow 2^U : { u \in U</td>
<td>(u, r) \in UA }$</td>
</tr>
<tr>
<td>authorized_users</td>
<td>$R \rightarrow 2^U : { u \in U</td>
<td>(u, r) \in UA }$</td>
</tr>
<tr>
<td>object_type</td>
<td>$O \rightarrow T$</td>
<td>Gives the type of an object.</td>
</tr>
<tr>
<td>roles</td>
<td>$U \cup P \cup V \rightarrow 2^R$</td>
<td>Maps users, permissions and services to roles.</td>
</tr>
<tr>
<td>roles*</td>
<td>$U \cup P \cup V \rightarrow 2^R$</td>
<td>Maps users, permissions and services to roles in presence of role hierarchy.</td>
</tr>
<tr>
<td>permissions</td>
<td>$R \rightarrow 2^P$</td>
<td>Maps roles to permissions.</td>
</tr>
<tr>
<td>permissions*</td>
<td>$R \rightarrow 2^P$</td>
<td>Maps roles to permissions in presence of role hierarchy.</td>
</tr>
</tbody>
</table>

**Definition 7. Role Hierarchy.** $A = R$, $B = R$ $a \leq b \rightarrow (a, b) \in RH$ where $a \in R, b \in R$.

For the joint project scenario, example role hierarchies for University A $RH_{UniA}$, B Corporation $RH_{CorpB}$, Hospital C $RH_{HostC}$ are presented below.

$RH_{UniA} = \{ \text{Member} \prec \text{Teaching}, \text{Member} \prec \text{Research}, \text{Member} \prec \text{Lecturer}, \text{Research} \prec \text{Faculty}, \text{Research} \prec \text{ResAssist}, \text{Admin} \prec \text{SysAdmin} \}$

$RH_{CorpB} = \{ \text{Member} \prec \text{Engineer}, \text{Member} \prec \text{Research}, \text{Member} \prec \text{Mgr}, \text{Engineer} \prec \text{SwEng}, \text{Research} \prec \text{RnDEng}, \text{Mgr} \prec \text{PrjMgr} \}$

$RH_{HostC} = \{ \text{Member} \prec \text{Medical}, \text{Medical} \prec \text{Doctor}, \text{Member} \prec \text{SocialSec} \}$

The types for Services may be defined as sub-types for Application object type in the Object Type hierarchy. The use of services together with Object-Type Hierarchies enable the use of FPM-RBAC model in specification of security policies for software, network and security services. An example object type hierarchy for the joint project scenario is $OTH = \{ (jprapp, App), (unif, File), (jrep, File), (prd, Db) \}$.

### 3.3.8. Authorization Terms

An authorization term $at$ is the basic formal construct used to specify security policy rules. $AT$ is the set of all authorization terms and $AT_r$ is the set of authorization terms for Domain $D_r$. The Permission Assignment relations together with Generic Constraints and Location and Mobility Constraints are formalized by authorization terms. Set of authorization terms $AT$ is defined as: $AT \subset PA \times C$ where $at \in AT$, $AT = \{(as, ao, act, co, fo) : as \in AS, ao \in AO, act \in ACT, (co, fo) \in C\}$

The formal specification of security policy rules is based on predicate logic and ambient modal logic within the authorization term structure. Here, we present the formalization process through realistic security policy examples. The policy statement in verbal form is followed by the corresponding formally specified authorization term. There are two types of policy specification. The first one is generic policy statements valid for all network models. The second one is network specific policy statements defined by a system or security administrator specific to a network. First, we list examples of generic policy statements that can be defined by following our method. In these generic statements, $host \in H$, $object \in O$, $user \in U$, $domain \in \Delta$.

- Users must be enrolled to a domain before they can login.: $(as = user, ao = domain, sa = login, fo = \diamond(domain[T]|user[T]|T), co = \forall user \in U, \exists domain \in \Delta, EDR(domain, user))$
- Users must be logged into a host before any other action can be conducted by a user in a domain: $(as = user, ao = domain, sa = A, \{login\}, fo = \diamond(domain|host[|user[T]|]|T)|T), co = \forall user \in U, \exists domain \in \Delta, \exists host \in H)$
Enrolling any entity to a domain can be done by a user with administrative rights:

\[(as = \text{user}, \ ao = \text{domain}, \ sa = \text{enrol}, \ fo = \text{T}, \ co = \forall \text{domain} \in \Delta, \exists \text{user} \in U, \text{ADM}(\text{user}, \text{domain}))\]

Below are some examples for formal specification of network specific policy statements with location and mobility constraints for the joint project scenario.

- Files in the patient records server \(h_{11}\) cannot be read by any user from portable hosts: \((as = \text{user}, \ ao = \text{File}, \ sa = (-) \text{read}, \ fo = \circ(\text{host}[\text{user}[T][T][h_{11}[\text{file}]])), \ co = \forall \text{user} \in U, (\text{OIT}(\text{file}, \text{File}) \land \text{OIT}(\text{host}, \text{Portable}))\)

- The user \(\text{nmullis}\) may send messages to the user \(jfrantz:\ (as = \text{nmullis}, \ ao = \text{Message}, \ sa = \text{send}, \ fo = \circ(\text{nmullis}[m][][T]) \land \circ(jfrantz[m][][T])), \ co = \text{OIT}(m, \text{Message}))\)

- The portable host \(h_{13}\) can be connected to the CorpB domain by users of CorpB: \((as = \text{user}, \ ao = h_{13}, \ sa = \text{connect}, \ fo = \circ(h_{13}[T][\text{CorpB}[T]]), \ co = \exists \text{user} \in U, \exists h_{13} \in H, \exists \text{CorpB} \in \Delta, \text{EDR}(\text{user}, \text{CorpB}) \land \text{OIT}(h_{13}, \text{Portable}))\)

- User \(\text{cmiele}\) can login to the joint project application server \(h_{12}\): \((as = \text{cmiele}, \ ao = h_{12}, \ sa = \text{login}, \ fo = \circ(h_{12}[T][\text{cmiele}[T]]), \ co = \text{T})\)

### 3.3.9. Domain Security Policy

The security policy \(P_i\) for a domain \(D_i\) includes the set of defined services, authorization terms and hierarchies. The data sets that are referred by the domain security policy are included in the specification of \(D_i\). The domain security policy is defined as \(P_i = \{V_i, AT_i, RH_i, OTH_i\}, 1 \leq i \leq \delta\).

Since the FPM-RBAC model is multi-domain, there may be multiple domain security policy definitions. Each domain security policy defines its access rules for local users to access local resources within the domain. The access rules for users to access inter-domain resources are specified by an inter-domain security policy. As a result of this administration method, the knowledge of local users and resources remains within the administrative boundary of a domain. The set of Domain Security Policies in a multi-domain environment is defined by the set \(\Omega\). The set of domain security policies is defined as \(\Omega = \{P_1, \ldots, P_\delta\}\).

### 3.4. Inter-Domain Security Policy Model

An *inter-domain security policy* is the set of access control rules, attributes and mapping for entities and information in multiple domains to be able to securely access or exchange information. It facilitates the provision of services between domains by providing a common set of security policies for cross-domain services. Use of inter-domain policies adds a layer of abstraction to security management by enabling the security management of cross-domain services independently from domain services.

The constructs specified in the Domain Security Policy Model presented in Section 3.3 are also valid in the Inter-Domain Security Policy Model (hereafter referred as inter-domain model). The inter-domain model adds constructs for *inter-domain services*, *inter-domain roles*, *inter-domain role hierarchy*, role maps, *inter-domain service access* and *inter-domain permission assignment*. Inter-domain security policy is defined over inter-domain authorization terms which are specified on the basis of these constructs.

The inter-domain model introduces the concepts of *Home Domain*, *Foreign Domain* and *Inter-Domain*. Home Domain is the boundary of security administration for a domain which provides services to other domains. Foreign domains are domains whose users access inter-domain services through inter-domain roles. Inter-domain policy includes rules for foreign users to access home domain objects through inter-domain roles, as well as the mapping of home and foreign domain roles to inter-domain roles.

As far as domain security administration is concerned, home domain administrator is only informed about the inter-domain role hierarchy. It is not necessary for the foreign domain role hierarchy to be communicated to the home domain. The home domain administrator defines rules for the foreign users to access home domain objects using inter-domain roles. In this manner, the principle of autonomous administration is preserved without the need of global distribution of knowledge of objects and users.

The overall view for access control policies in a multi-domain environment is depicted in Figure 4. The inter-domain access control model of FPM-RBAC is detailed in Figure 5. Here, we introduce the concept of Inter-Domain Roles and Inter-Domain Role Hierarchies. Roles are mapped to
Inter-Domain Roles with a role mapping function. An inter-domain service is accessible by home or foreign users through inter-domain roles. It is associated with actions and authorization objects from the home domain. The access rules for inter-domain services by foreign roles is achieved through the \textit{Inter-Domain Permission Assignment} relation.

### 3.4.1. Inter-Domain Services

\textit{Inter-domain services} are services which are associated with authorization objects and actions within a set of domains $\Gamma$. One of the elements of $\Gamma$ is named as \textit{home domain} and the others are named as \textit{foreign domains}. Inter-domain services are accessed by inter-domain roles. The set of inter-domain services $I_{\Gamma}$ is defined as follows.

**Definition 8.** The set of inter-domain services $I_{\Gamma}$ among a set of domains $\Gamma$, where $D_\alpha$ is the Home Domain and $D_\beta, D_\gamma, \ldots$ are foreign domains, is specified as $I_{\Gamma} = \{S_j : 1 \leq j \leq \gamma\}$ where $\gamma$ is the number of inter-domain services.

The subset of authorization objects associated with an inter-domain service is called \textit{inter-domain resources}. The subset of actions associated with an inter-domain service is called \textit{inter-domain tasks}. The associations are defined through the Service Resources ($SR_\Gamma$) and Service Tasks ($ST_\Gamma$) relations. The mentioned sets and relations are defined as follows.

**Definition 9.** Inter-domain service resources for a specific inter-domain service $S_j$ is $\hat{P} = \{\Gamma, H_j, O_a, T_a, P_j\} \subseteq AO$. $H_j = \{h_k, \ldots, h_l\}$ are hosts associated with $S_j$, $O_a = \{o_m, \ldots, o_n\}$ are objects associated with $S_j$, $O_a \subseteq O_a$, $T_a = \{t_x, \ldots, t_y\}$ are object types associated with $S_j$, $T_a \subseteq T_a$. The set of all inter-domain service resources is $\hat{P} = \bigcup_{j=1}^{\gamma} \hat{P}_j$. The set of inter-domain service tasks is $\hat{K} = \bigcup_{j=1}^{\gamma} \hat{K}_j$. The inter-domain service resources relation is $SR_\Gamma \subseteq I_{\Gamma} \times \hat{P}$. The inter-domain service tasks relation is $ST_\Gamma \subseteq I_{\Gamma} \times \hat{K}_j$.

Regarding the joint project scenario, we consider two inter-domain services: The Joint Research Project service and the Patient Health Records service. The resources associated with the Joint Research Project service include the application server, joint research web application, project work files and project report files. The tasks associated with this service include actions to login and execute the application as well as to read and write project files. The resources associated with the Patient Health Records service include the patient records database and the file server of Hospital C containing patient records. The tasks associated with this service include actions to login to the file server as well as to read and write database records.

The definitions for inter-domain services are as follows. Here, $\hat{S}_1$ represents the Joint Research Project inter-domain service and $\hat{S}_2$ represents the Patient Health Records inter-domain service.

\begin{align*}
I_{\Gamma} & = \{\hat{S}_1, \hat{S}_2\} \\
\hat{P}_1 & = \{UniA, CorpB, HosC, jrapp, unif, jrep, App, File, h_{11}, h_{12}, h_{13}, h_{21}, h_{22}, h_{31}, h_{32}\}, \\
\hat{P}_2 & = \{UniA, HosC, prd, App, Db, File, h_{12}, h_{31}\} \\
\hat{P} & = \hat{P}_1 \cup \hat{P}_2 \\
\hat{K}_1 & = \{Login, Logout, Execute, Read, Write\}
\end{align*}
\[
\hat{K}_2 = \{Login, Logout, Read, Write\} \\
\hat{K} = \hat{K}_1 \cup \hat{K}_2 \\
SR_T = \hat{S}_1 \times P_1 \cup \hat{S}_2 \times P_2 \\
ST_T = \hat{S}_1 \times \hat{K}_1 \cup \hat{S}_2 \times \hat{K}_2
\]

3.4.2. Inter-Domain Roles

An inter-domain role is a role which does not apply to a particular domain and is used for mapping of roles between domains. A direct map between roles of multiple domains is avoided to provide more flexibility of administration and enforcement. Inter-domain services can be directly specified according to the needs of information sharing between organizations and enforced with respect to inter-domain roles. The set of Inter-Domain Roles \( R_T \) is defined as \( R_T = \{r_1^1, \ldots, r_g^g\} \) where \( g \) is the number of inter-domain roles. The definition of inter-domain role hierarchy \( RH_T \) in Definition 10 is similar to RH relation.

Definition 10. Inter-Domain Role Hierarchy \( RH_T \triangleq a \leq b \rightarrow (a, b) \in RH_T, a, b \in R_T \).

3.4.3. Role Map

A role map \( RM \) is a many-to-one relation from a set of roles \( R \) to inter-domain roles \( R_T \). The choice of many-to-one relations for defining role maps eliminates conflicts in role mapping. A home or foreign role may not map to multiple inter-domain roles associated with a possibly conflicting set of permissions. Furthermore a direct map between roles of multiple domains is avoided by introduction of home and foreign role maps. This decision facilitates independent administration of inter-domain roles and inter-domain role hierarchies. The role map from home roles to inter-domain roles is represented as \( RM_h \) and the role map from foreign roles to inter-domain roles is represented as \( RM_f \).

Definition 11. Role Maps are specified as
\[
RM_h = \{(r_i, r_1^j) : r_i \in R_a, r_1^j \in R_T\} \quad \exists(r_x, r_y) \in RM_h \land \exists(r_x, r_z) \in RM_h \rightarrow y = z \\
RM_f = \{(r_k, r_1^j) : r_k \in R_b, r_1^j \in R_T\} \quad \exists(r_x, r_y) \in RM_f \land \exists(r_x, r_z) \in RM_f \rightarrow y = z, \text{ where } D_a \text{ is the home domain and } D_b \text{ is a foreign domain.}
\]

Consider the example presented in Figure 6. In this example the role hierarchies are assumed to be tree structures for the sake of simplicity. Here,

\[
R_1 = \{r_{11}, r_{12}, r_{13}, r_{14}, r_{15}, r_{16}\}, \\
R_2 = \{r_{21}, r_{22}, r_{23}, r_{24}, r_{25}, r_{26}\}, \\
R_T = \{r_{121}, r_{122}, r_{123}, r_{124}, r_{125}\}, \\
RM_h = \{(r_{12}, r_{125}), (r_{13}, r_{124}), (r_{15}, r_{124})\}, \\
RM_f = \{(r_{24}, r_{124}), (r_{26}, r_{125})\}.
\]

3.4.4. Inter-Domain Relations and Authorization Terms

The relation \( PA_T \) is the inter-domain permission assignment relation, used for the assignment of inter-domain roles to permissions. \( SA_T \) is the inter-domain service assignment relation used for assignment of inter-domain roles to inter-domain services. \( AT_T \) is the set of inter-domain authorization terms specified as inter-domain permission assignments associated with constraints. \( C_T \) is the set of inter-domain constraints. Inter-domain permission assignment, inter-domain service access relation and inter-domain authorization terms are respectively specified as follows: \( PA_T \subset R_T \times CT \times AO \), \( SA_T \subset R_T \times ST_T \), \( AT_T \subset PA_T \times CT \), \( AT_T = \{(as, ao, sa, co, fo) : as \in R_T, ao \in AO, sa \in ACT, (co, fo) \in CT\} \).

Regarding the joint research project case, assume that the "project manager" (ResProjMgr) may use the Joint Project service, but may not access patient health records. In contrast "Supervisor" (Supervisor) may access health records but not Joint Project resources. "System administrator" (SysAdmin) may perform administrative duties on Joint Project service, for example to manage and upgrade the joint project web application. A service access matrix for the joint project inter-domain access is given in Table 5.
3.4.5. Inter-Domain Security Policy

An inter-domain security policy $W_I$ is defined among a set of domains $\Gamma$, where $D_a$ is the Home Domain and $D_b, D_c, \ldots$ are foreign domains. $W_I$ includes inter-domain services $T_I$, inter-domain role hierarchy $RH_I$, home domain to inter-domain role-mapping $RM_h$, foreign domain to inter-domain role mapping $RM_f$ and inter-domain authorization terms $AT_I$. The formal representation of Inter-Domain Security Policy is $W_I = \{ T_I, RH_I, AT_I, RM_h, RM_f \}$. 

Below we present examples for inter-domain security policy rules. The formal specification includes the mobility and access control aspects of the inter-domain policy. Example policy statements for mobility and inter-domain access as a verbal form of policy rule are given followed by formal specification of authorization terms and an interpretation of the location and mobility constraints.

### Table 6: A Part of Permission Assignment Relation for Joint Project service relating to object jrapp.

<table>
<thead>
<tr>
<th>Inter-Domain Role</th>
<th>Permission</th>
<th>Object</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>ResPrjMgr</td>
<td>jrapp</td>
<td>+execute</td>
<td></td>
</tr>
<tr>
<td>ResGrpMgr</td>
<td>jrapp</td>
<td>+execute</td>
<td></td>
</tr>
<tr>
<td>SysAdmin</td>
<td>jrapp</td>
<td>+manage, +write</td>
<td></td>
</tr>
<tr>
<td>Researcher</td>
<td>jrapp</td>
<td>+execute</td>
<td></td>
</tr>
<tr>
<td>Supervisor</td>
<td>jrapp</td>
<td>-execute</td>
<td></td>
</tr>
</tbody>
</table>

Detailed access permissions may be granted to roles based on the services. An excerpt of a permission assignment relation related to the object jrapp is given in Table 6.

1. "Users with Research Project Manager role from Corporation B are allowed to login with portables to University A."

2. "Software Engineers of Corporation B can read data from files on the project server in University A domain from within University A."

3. "Researchers of University A with a Portable host within Hospital C can not write files to the patient records database in servers of Hospital C."

Interpretation of the constraints: There is a user, who has assumed the role of a Research Project Manager, which is logged into a host of object type Portable and is not yet connected to either UniA or CorpB, in a location that can be connected to either of the domains UniA and CorpB.

Interpretation of the constraints: UniA contains a project server including a file with data and the user who has assumed the role Software Engineer, is an active domain user of UniA and an enrolled domain user of CorpB, is located somewhere inside UniA.

Interpretation of the constraints: The user who has assumed the role Researcher, is an enrolled domain user of UniA, and an active domain user of HosC, who is logged onto a Portable host with a file, is within UniA domain which contains a Server $h_{31}$ with a Database $prd$.

3.5. Separation of Duty (SOD) Constraints in FPM-RBAC

Static SOD enforces constraints on the assignment of Users to Roles. If a user is authorized as a member of one role, the user may be prohibited...
to become a member of another (conflicting) role. Dynamic SOD enforces constraints on the activation of Roles by Users. A user may be assigned to two or more conflicting roles but may activate only one of them at any specific session.

In FPM-RBAC, novel classes of SOD constraints are identified: (i) Service based SOD, (ii) Inter-Domain SOD and (iii) Location and Mobility Based SOD. Service-based SOD is defined on the basis of a set of conflicting services CS. CS is a set of services, where two or more of its members may not be assigned to a single role. For example, a role may not be assigned to services "Secret" and "Unclassified". Inter-Domain SOD may be based on inter-domain role assignment, home role mapping or foreign role mapping. Location based SOD may be based on service location, role location or permission location. The classification of SOD in FPM-RBAC is presented in Figure 7. Static or dynamic aspects are represented by time of evaluation. Static SOD is evaluated statically against initial system state, while Dynamic SOD is evaluated at runtime against current system state.

In [30], the SOD constraints are specified within the Role-Based Constraints Language using the sets of conflicting roles, permissions and user sets in addition to the standard RBAC model. Mutually disjoint roles are defined as conflicting roles set CR. The concept of conflicting permissions defines conflict in terms of permissions rather than roles. Two permissions may be considered in conflict independent from role assignment relation. For example, permissions to write grades to student files and to approve grades of students may be declared as conflicting permissions. Role and Permission based SOD are included as Single-Domain constraints (SCR and SCP) in this classification. An interplay of the three classes of SOD constraints are also possible however a detailed analysis is out of scope of this study.

3.5.1. Single Domain SOD

The Single Domain SOD is defined based on the static separation of duty definition in the standard RBAC model. We differentiate between role-based, permission-based and service-based SOD in FPM-RBAC model. Service Based SOD is a novel concept defined in the FPM-RBAC model. The sets of roles, permissions and services which are the basis for mutual exclusion are defined as CR (conflicting roles), CP (conflicting permissions) and CS (conflicting services) respectively. The Static Separation of Duty based on roles follows the standard RBAC model definitions [11]. Additionally in FPM-RBAC, SCS (SOD based on Conflicting Services) and SCP (SOD based on Conflicting Permissions) are defined as follows:

**Static Separation of Duty based on Roles:**

The Static Separation of Duty based on Roles: $$SCR \subseteq (2^R \times N)$$ is a collection of pairs where $$CR = \{r_x, \ldots, r_y : r_x, \ldots, r_y \in R\}$$ is a set of conflicting roles, $$t$$ a subset of roles in CR, and $$n$$ a natural number $$n \geq 2$$, s.t. no user is assigned to $$n$$ or more roles from the set CR in each $$(t, n) \in SCR$$. $$SCR$$ is defined in (3). In presence of a hierarchy, the function $$\text{assigned\_users}(r)$$ in (3) is replaced with the function $$\text{authorized\_users}(r)$$.

$$\forall(t, n) \in SCR, \forall t \subseteq CR \mid t \geq n$$

$$\rightarrow \bigcap_{r \in t} \text{assigned\_users}(r) = \emptyset \quad (3)$$

**Static Separation of Duty based on Services:**

$$SCS \subseteq (2^V \times N)$$ is a collection of pairs where $$CS = \{v_x, \ldots, v_y : v_x, \ldots, v_y \in V\}$$ is a set of conflicting services, $$s$$ a subset of services in CS, and $$n$$ a natural number $$n \geq 2$$, s.t. no role is assigned to $$n$$ or more services from the set CS in each $$(s, n) \in SCS$$. $$SCS$$ is defined in (4). In presence of a hierarchy, the function $$\text{enabled\_roles}(v)$$ in (4) is replaced with the function $$\text{enabled\_roles}^*(v)$$.
\[ \forall (s, n) \in SCS, \forall s \subseteq CS : | s | \geq n \]
\[ \rightarrow \bigcap_{v \in s} enabled\_roles(v) = \emptyset \quad (4) \]

*Static Separation of Duty based on Permissions:* Where \( CP = \{ cp_i : cp_i \in ACT \times AO \} \) is a set of conflicting permissions, the set of conflicting inter-domain roles is interpreted as \( CR = \{ r_x : permissions^\ast(r_x) \cap cp_i \neq \emptyset \} \). Specification (3) is applicable with the derived CR set.

### 3.5.2. Inter-Domain SOD

The introduction of role maps arise additional concerns about separation of duty. Due to multiple sets of conflicting roles for home, foreign and inter-domain roles, new types of SOD need to be defined. First we define the set of conflicting inter-domain roles, \( CR_T = \{ r_1^T, \ldots, r_n^T : r_1^T, \ldots, r_n^T \in R_T \} \). We identify the following types of static SOD related with inter-domain role mapping:

1. **SICR:** No user can be assigned to a set of roles which map to \( n \) or more conflicting inter-domain roles.
2. **SRM_h:** No user can be assigned by mapping to \( n \) or more inter-domain roles which are mapped by a set of conflicting home domain roles.
3. **SRM_f:** No user can be assigned by mapping to \( n \) or more inter-domain roles which are mapped by a set of conflicting foreign domain roles.

We introduce a function \( rmap(r) : R \times R_T \) which gives the set of mapped inter-domain roles for a set of home or foreign domain roles. The formal definitions for the newly introduced static SOD constraints **SICR, SRM_h, SRM_f** are respectively defined in (5),(6) and (7).

\[ \forall (\alpha, n) \in SICR, \forall \alpha \subset R, | rmap(\alpha) | \geq n, \]
\[ \bigcap_{r \in \alpha} assigned\_users(r) \neq \emptyset \rightarrow rmap(\alpha) \notin CR_T \quad (5) \]

\[ \forall (\alpha, n) \in SRM_h, \forall \alpha \subset R, | rmap(\alpha) | \geq n, \]
\[ \bigcap_{r \in \alpha} assigned\_users(rmap(r)) \neq \emptyset \rightarrow \alpha \notin CR_h \quad (6) \]

As an example of static SOD relations for inter-domain role mapping, consider the example depicted in Figure 8. Let \( CR_h = \{ r_{12}, r_{13}, r_{15} \}, CR_f = \{ r_{124}, r_{125} \}, CR_f = \{ r_{23}, r_{24}, r_{26} \} \). The home domain mapping is \( RM_h = \{ (r_{13}, r_{124}), (r_{15}, r_{122}) \} \). The foreign domain mapping is \( RM_f = \{ (r_{25}, r_{124}), (r_{26}, r_{125}) \} \). The role assignment \{ (u, r_{122}), (u, r_{124}) \} is conflicting according to \( SRM_h \) and \{ (u, r_{25}), (u, r_{26}) \} is conflicting according to **SICR**.

**SOD relations may be defined in an alternative way. One may define sets of conflicting permissions instead of conflicting roles. In this case, the conflicting roles can be interpreted as the roles that have conflicting permissions assigned to them. In this case, the set of conflicting inter-domain permissions can be defined by the administrator as \( CP_T = \{ cp_i : cp_i \in ACT \times AO \} \) and the set of conflicting inter-domain roles is interpreted by the system as \( CR_T = \{ r_x^T : permissions^\ast(r_x^T) \cap cp_i \neq \emptyset \} \). The inter-domain SOD constraints presented in this section are then applicable with the derived \( CR_T \) set.

### 3.5.3. SOD for Location and Mobility Constraints

We identify new aspects about the specification of SOD constraints together with location and mobility constraints. Two or more roles or permissions may be considered in conflict if one of these permissions relates to an action which involves mobility. For example, the Guest Lecturer role may be in conflict with the Lecturer role if the person is visiting another university. As an example to permission conflicts arising from mobility, consider a requests to send a student record file to another institution.
and at the same time to change the records. In this case, the records which have been sent would be in conflict with the ones changed.

In the FPM-RBAC model, Static and Dynamic SOD constraints may be used in accordance with Location and Mobility Constraints. The generic form of location and mobility based SOD constraints based on set \( C \in \{ CR, CP, CS \} \) is \((C, n, sfo)\) where \( n \) is a natural number and \( sfo \) is a (set of) location constraint(s) for separation of duty. Set \( C \) is specified depending on whether the constraint is role-based, permission-based or service-based. We associate the SOD constraints with the Location and Mobility constraints for conflicting roles and conflicting services as follows:

1. **Conflicting Roles with Location Constraints (SLCR):** \( n \) or more roles may not be assigned to a user if these roles are within the conflicting roles set \( CR \) and if the Location constraint \( sfo \) is valid in the initial location configuration of the system.

2. **Conflicting Services with Location Constraints (SLCS):** \( n \) or more services from the set of conflicting services \( CS \) may not be assigned to a role if the location constraint \( sfo \) is valid in the initial location configuration of the system.

Static location constraints are evaluated against the initial location configuration \((LCONF_0)\), which depicts the state of the network before any action has been executed. The formal definitions of SLCR, location and mobility SOD based on conflicting roles, and SLCS, location and mobility SOD based on conflicting services, are in (8) and (9).

\[
\forall (t, n) \in SLCR, \forall t \subseteq CR : | t | \geq n \rightarrow \bigwedge_{r \in t} \text{assigned}_\text{users}(r) = \emptyset \land LCONF_0 \models sfo \quad (8)
\]

\[
\forall (s, n) \in SLCS, \forall s \subseteq CS : | s | \geq n \rightarrow \bigwedge_{v \in s} \text{enabled}_\text{roles}(v) = \emptyset \land LCONF_0 \models sfo \quad (9)
\]

In the presence of hierarchies, SLCR is obtained by replacing the function \( \text{assigned}_\text{users} \) in (8) with the function \( \text{authorized}_\text{users} \). Similarly, SLCS in the presence of hierarchies is obtained by replacing \( \text{enabled}_\text{roles}(r) \) in (9) with \( \text{enabled}_\text{roles}^*(r) \).

A number \( n \) or more authorization terms are in conflict if among all authorization terms assigned to the same authorization subject, \( n \) or more permissions are within the conflicting permissions set \( CP = \{ cp_i : cp_i \in ACT \times AO \} \) and if the Location constraints associated with the permissions are all valid in the initial location configuration of the system. In this case the location formula for separation of duty is defined by the set \( \{ f_{o1}, \ldots, f_{on} \} \) associated with authorization terms \( \{ at_1, \ldots, at_n \} \). The formal definition for \( SLCP \), SOD based on Conflicting Permissions with Location Constraints, is given in (10).

\[
\forall (p, n) \in SLCP, \forall \alpha \subseteq AT, \forall p \subseteq CP : p \geq n, \\
\alpha = \{ (as, p_{o1}, f_{o1}), \ldots (as, p_{on}, f_{on}, c_{on}) \}, \\
\text{permissions}(as) \subseteq p \rightarrow LCONF_0 \models \{ f_{o1}, \ldots, f_{on} \} \\
(10)
\]

In presence of an hierarchy, the assignment of permissions to authorization subjects are evaluated according to the role hierarchy. In this case, the function \( \text{permissions}(as) \) in (10) is replaced by the function \( \text{permissions}^*(as) \).

### 3.5.4. Dynamic SOD

Here we define the dynamic SOD constraints in the FPM-RBAC model. Similar to static SOD constraints, dynamic SOD constraints also have three different types, namely, role based, service based and location/mobility based dynamic SOD. The main differentiation between Static and Dynamic SOD constraints are: (i) dynamic constraints are based on activation of roles, which occurs when a user logs into a service, rather than assignment or roles and (ii) dynamic constraints are evaluated at run-time against the current state of the system, in which a set of actions has already been executed. The spatial and temporal location and mobility constraints enable the specification of complex temporal and spatial restrictions on separation of duty. The concept of services, which binds a dynamic login session to a set of authorization objects, replaces the concept of sessions in standard RBAC. Since the user may login to one service at a time, the DSOD definition based on services is related to the set of enabled roles within a service.

**Dynamic Separation of Duty based on Roles:** \( DCR \subseteq (2^R \times N) \) is a collection of pairs where \( CR = \{ r_x, \ldots, r_y : r_x, \ldots, r_y \in R \} \) is a set of conflicting roles, \( t \) a subset of roles in \( CR \), and \( n \) a natural number \( n \geq 2 \), s.t. no user has actively
assumed $n$ or more roles from the set $CR$ in each $(t, n) \in DCR$.

$$\forall(t, n) \in DCR, \forall u \in U, \forall t \subseteq CR : |t| \geq n \rightarrow \bigwedge_{r \in t} RAS(u, r) = \bot \quad (11)$$

**Dynamic Separation of Duty based on Services:**

$$\forall(t, n) \in DCS, CR \subset 2^R, \forall t \subseteq CR,$$

$$\forall v \in V, t \subseteq enabled\_roles(v) \land |t| \geq n$$

$$\rightarrow \bigwedge_{r \in t} REN(v, r) = \bot \quad (12)$$

**Dynamic Separation of Duty based on Locations and Mobility:**

Location and mobility based dynamic SOD constraints are evaluated against the location configuration of the system at time $\tau$ ($LCONF_\tau$), after a set of actions has already been executed by users. The dynamic constraints take hierarchies into account since the predicates RAS and REN take into account the effect of descendant roles. Where $CR = \{r_x, \ldots, r_y : r_x, \ldots, r_y \in R\}$ is a set of conflicting roles, the formal definitions of dynamic location and mobility SOD constraints based on conflicting roles $DLCR$ and conflicting services $DLCS$ are given in (13) and (14).

$$\forall(t, n, \tau, sfo) \in DLCR, \forall u \in U, \forall t \subseteq CR : |t| \geq n$$

$$\rightarrow \bigwedge_{r \in t} RAS(u, r) = \bot \land LCONF_\tau \models sfo \quad (13)$$

$$\forall(t, n, \tau, sfo) \in DLCS, CR \subset 2^R, \forall t \subseteq CR,$$

$$\forall v \in V, t \subseteq enabled\_roles(v) \land |t| \geq n$$

$$\rightarrow \bigwedge_{r \in t} REN(v, r) = \bot \land LCONF_\tau \models sfo \quad (14)$$

4. Verification of FPM-RBAC Security Policies

Use of the FPM-RBAC policy model enables formal and automated verification of inter-domain security policies. A novel aspect of FPM-RBAC is that location and mobility of users, objects and hosts may be included in verification and mobility within multiple domains may be addressed. In this section, we present the summary of the spatio-temporal model checking algorithm for checking satisfaction of location and mobility constraints.

The Ambient Calculus Model Checker is a realization of the spatio-temporal model checking algorithm. The details of this algorithm and the model checker may be found in our previous work [15, 31]. We also present two examples, first concerning two domains and the second concerning multiple domains.

4.1. Spatial and Temporal Model Checking for Security Policies

We divide our problem into two sub problems; namely temporal model checking and spatial model checking. The temporal model checker is used for carrying out satisfaction process for the Sometime and Everytime connectives of ambient logic. The proposed model checking method generates all possible future states and build a state transition system based on the Ambient Calculus process specification. After evaluation of Ambient Logic formula in each state, this state transition system is processed into a Kripke Structure which is then given to temporal model checker. NuSMV [32] is used as a temporal model checker.

Outline of the algorithm for the model checking is presented below.

1. Define atomic propositions with respect to spatial properties of ambient logic formula and register the (atomic proposition-spatial modality) couples.
2. Reduce ambient logic formula to temporal logic formula (CTL) by replacing spatial modalities with atomic propositions.
3. Generate state transition system of the ambient calculus specification with respect to reduction relations. This involves generation of initial state from given ambient calculus specification, generation of new states by applying available capabilities with respect to ambient calculus reduction relations and addition of new states to state transition system with transition relation.
4. Generate Kripke Structure from state transition system. This step involves the assignment
Table 7: Part of output generated by the spatial model checker for an example policy.

<table>
<thead>
<tr>
<th>State</th>
<th>Spatial state of World</th>
<th>AP</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>(d_1[u_1</td>
<td>h_1[f_1[{\text{data}_1}]]]) (&amp;) (d_2[h_2]</td>
<td>u_2[{\text{data}_2}]) (\perp) (u_2[\text{out}\ h_2])</td>
</tr>
<tr>
<td>52</td>
<td>(d_1[u_1</td>
<td>h_1[f_1]</td>
<td>d_2[h_2]</td>
</tr>
<tr>
<td>53</td>
<td>(d_1[u_1</td>
<td>h_1[f_1]</td>
<td>d_2[h_2]</td>
</tr>
</tbody>
</table>

of the values of the atomic propositions for each state of state transition system (labeling) by applying model checking for spatial modalities on ambient topology of the related state and the addition of a new state with its label (values of atomic propositions) to the Kripke Structure.

5. Generate NuSMV code from Kripke Structure and CTL Formula.

As an example, let’s consider the scenario and policy example presented in Sections 3.2.2 and 3.3.5. When the Ambient Calculus specification is input to the model checker, a total of 53 states are generated. One Atomic Proposition (AP) is generated, where \(AP = \circ\{\circ h_2[\circ\{\text{data}_1[\top]\}][\text{data}_2[\top]]\}\). A part of the execution of the algorithm is presented in Table 7. Only the initial and the last two states are shown. For each state an action is executed to produce a new spatial state. For state 53 the spatial model checking algorithm matches the spatial formula AP to the current state of World.

A detailed analysis for the complexity and performance of the model checking algorithm may be found in [31]. According to this analysis, the time complexity of state transition system generation is dependent on the number of capabilities. Where \(n\) is the number of capabilities in the Ambient Calculus specification, the time complexity of generating the state transition system in the worst case is \(O(n!)\). The time complexity of the spatial match process is exponential with the number of ambient states. The space complexity of the model checking algorithm is linear with the number of capabilities and the number of connectives in the location formulas. The time complexity and performance of temporal model checking depends on the number of states generated with respect to an ambient calculus specification. Research on partial order reduction techniques to reduce the number of states generated by ambient calculus model checking, is in progress.

4.2. Example for Verification of Inter-Domain Security Policy between Two Domains

In the first example there are two domains, UniA and HosC. Users may roam between domains, but may be logged onto one domain. Assume that a security policy rule states that data related to other research projects of University A should not be in the same location with patient health records. This is formalized with the mobility constraint (ILFormula). Joint researchers are allowed to login to servers and read files and databases within UniA and HosC.

In Figure 9, an information leakage breach in this setting is shown. The set of possible actions allowed by the security policy is modeled as an ambient calculus specification (ILSpec) and the location and mobility constraint is modeled as an ambient logic formula (ILFormula) in (15). In order to find security breaches, the specification includes all possible actions of the user and possible movements of the patient health record, encoded as \(\text{pdata}\) within the database \(\text{prec}\). Here the users \(\text{fmcbride}\) and \(\text{frintz}\) with mobile devices have been encoded respectively as \(u_1\) and \(u_2\).

![Figure 9: An example mobile user inter-domain access leading to an inter-domain security policy violation](image)

\[
\text{ILSpec}_1 := \text{HosC}[u_1]|h_{31}|\text{prec}|\text{pdata}|
\text{in}\ u_2.0|\text{out}\ u_2.0]\} \text{UniA}[h_{11}|u_2|\text{out}\ h_{11.0}]
\text{out}\ \text{UniA.0}|\text{in}\ \text{HosC.in}\ h_{31.0}|\text{out}\ h_{31.0out}
\text{HosC.0}|\text{in}\ \text{UniA.in}\ h_{11.0}|\text{in}\ \text{unif.0}|\text{in}\ \text{prec.0}|
\text{out}\ \text{prec.0}|\text{out}\ \text{unif.0}|\text{in}\ \text{unif.0}|\text{in}\ \text{pdata0}]
\text{ILFormula} := \square\{\neg\circ \{\circ \text{pdata}[\top]\} \cup \text{unif}[\top] \top\} \} \}
\] (15)
The model checker finds the sequence of actions that violate the security policy rule ILFormula. The following action sequence formalized by ambient calculus capabilities leads to a violation of inter-domain security policy:

\[
LCONF_0 \xrightarrow{u_2[\text{out } h_{11}.\text{out } UniA.\text{in } HosC.\text{in } h_{31}]} \quad \text{pdata}[\text{in } u_2][\text{out } h_{31}.\text{in } UniA.\text{in } h_{11}]^LCONF_1
\]

The interpretation of these actions is, the university researcher \(u_2\) (jrantz) moves from UniA domain to HosC domain with a mobile device, reads patient health records from HosC file server to his mobile device, moves it to the university domain and instead of writing it to the joint research application server, writes it to the research lab client where it can be read by other UniA users. Although all of the actions in the specification are allowed individually by the security policy, a particular sequence of actions may lead to a security breach as shown in this example.

4.3. Example for Verification of Inter-Domain Security Policy among Multiple Domains

Let’s assume that the there are security policy rules stating that patient records database \(prd\) may be read directly by users of UniA and HosC but not by CorpB users. The joint project users in UniA may also write records to joint project web application \(jrapp\) of UniA. Assume that the security policy allows \(jrapp\) in Host \(h_{12}\), to be accessible and readable by Software Engineers in CorpB (dmendiola is encoded as \(u_4\)). Also assume that patient records server Host \(h_{31}\) in Domain HosC is accessible by Domain UniA user \(nmullis\) (encoded as \(u_2\)). These policy rules result in the formalization of security policy \(LCONF\) and the mapping of permissions to the formal specification as detailed in Table 8.

Now we specify an inter-domain security policy rule that patient records in HosC domain should not ever occur within CorpB file server \(h_{22}\). This results in a security policy rule with spatial and temporal formula specified as \(fo = \square(\neg \diamond \{\circ [h_{22}[\circ [\text{prd}][T]][T]]\})\). The specifications are given to the model checker to find a counterexample which satisfies \(LCONF \models fo\).

The model checker generates a total of 81782 states for this scenario. The flow of states leading to a state in which \(fo\) is invalidated denotes the sequence of actions leading to the unintended violation of the inter-domain security policy. The sequence of actions, corresponding to the states leading to a counterexample discovered by the model checker is outlined below.

\[
LCONF_0 \xrightarrow{u_3[\text{read}, \text{prd}]} LCONF_1
\]
\[
\xrightarrow{u_2[\text{login}, jrapp]} u_2[\text{write}, \text{prec}] LCONF_2
\]
\[
\xrightarrow{u_4[\text{login}, jrapp]} LCONF_3
\]
\[
\xrightarrow{u_4[\text{read}, jrapp]} LCONF_4
\]

According to this sequence of actions, User \(nmullis\) \((u_2)\), who is logged into \(h_{11}\) in UniA, reads patient records \(prec\) from patient records database \(prd\) in \(h_{31}\) \((LCONF_0 \rightarrow LCONF_1)\). \(u_2\) logs into application \(jrapp\) and writes \(prec\) to application \(jrapp\) in \(h_{12}\) \((LCONF_1 \rightarrow LCONF_2)\). Then software engineer \(dmendiola\) \((u_4)\) of CorpB logs into \(jrapp\) \((LCONF_2 \rightarrow LCONF_3)\). Then \(u_4\) reads and discloses the database records from within file server \(h_{22}\) in CorpB \((LCONF_3 \rightarrow LCONF_4)\). The state \(LCONF_4\) violates the inter-domain security policy rule.


The algorithm presented in this section is utilized for making the permission or denial decisions for requested actions against security policy specifications. Determination of whether the actions in the multi-domain mobile network are permitted requires a formal linkage of access requests to the security policy model. Checking the satisfaction of current state of the system to constraints in the security policy is essential to match rules to policies. The location configuration of the system \(LCONF\) holds the ambient calculus process specification for the current system. With each action this configuration is updated. The location constraints which are enabled in the current location configuration of the system is determined by spatial model checking. This computation is decidable when using the fragments of ambient calculus and logic presented in this paper. The set of functions and predicates in domain policies and the inter-domain policies are defined respectively by the set of security policies \(\Omega\) and \(W\). The state of the logical functions and predicates are tracked by the system. The Generic
constraints as well as SOD constraints are evaluated according to the current state of the logical functions and predicates.

Matching actions to policy rules requires the following steps:

1. Determine the applicable authorization terms based on the service activated by the user,
2. Map the authorization subjects in the policy rule to roles in the access request through role hierarchy definitions,
3. Map the authorization objects in the policy rule to the object on which an access is requested, through object hierarchy definitions,
4. Determine whether the location and mobility constraints are satisfied by the location configuration,
5. Determine whether the Generic constraints and SOD constraints are satisfied.
6. Determine whether the action is permitted or denied through the Access Control Function.

The steps for checking the satisfaction of Location constraints and Generic constraints may be computed off-line during state changes. The evaluation of the Access Control function needs to be computed at the time of the request. In the following paragraphs, we investigate the functions and algorithms necessary for the execution of the steps mentioned above.

5.1. Evaluation of Access Requests according to Services

An access request \( ar \) is specified as \( ar = (u, serv, ru, l_r, act, obj) \) where \( serv \in \mathcal{V} \cup \mathcal{I}_r \), \( ru \in R \), \( l_r \in H \cup \Delta \), \( act \in A \), \( obj \in O \). Here, \( serv \) denotes the active service that the user has activated including inter-domain services, \( ru \) is the role assumed by the user, \( l_r \) is the location of the user (a host or domain), \( act \) is the requested action and \( obj \) is the object upon which an action is requested.

The subset of domain security policy for domain \( i \) pertaining to the service activated by the user, \( P'_i \), is specified as \( P'_i \subseteq P_i : \{ v \in \mathcal{V}, serv = v \} \).

Given an authorization term in of the form \( at = (as, ao, sa, co, fo) \), the set of available authorization terms to a role within a service \( AT'_i \subset AT_i \) is specified in (16).

\[
AT'_i = \bigcup_{at \in AT} RAS(as, ru) \land REN(r_u, serv) \tag{16}
\]

The calculation is straightforward since the PAM already defines permissions for each service. The authorization terms are the rules in the permission access matrix for the service which relate to the enabled roles.

5.2. Evaluation of Hierarchies

First, the system checks whether the requested role in the access request may be assumed by the user. This is equivalent to \( RAS(u, ru) = \top \).

Second, the set of authorization terms which apply to the role and object hierarchies defined in the security policy needs to be calculated. This takes four steps, to find the set of authorization terms \( at = (as, ao, sa, co, fo) \), \( at \in AT'_i \), where:

1. the user specified in the authorization term (if any) is equal to the user requesting the service, \( AT'_u = \{ at | at \in AT'_i, as = u \} \)
2. the role specified in the authorization term is equal or a descendant role of the role specified in the access request, \( AT'_r = \{ at | at \in AT'_i, as = ru \lor as \prec ru \} \)
3. the object specified in the authorization term is equal to the object upon which the access is requested, $AT^o_i = \{at | at \in AT^i_i, ao = obj\}$

4. the object specified in the access request is of object type specified in the authorization term, or the object type of the object is a descendant of the object type specified in the authorization term, $AT^{ot}_i = \{at | at \in AT^i_i, OIT(obj, ao) \lor object\_type(obj) < ao\}$

The set of derived authorization terms in presence of role and object type hierarchies $AT''$ is $AT'' = AT^u_i \cup AT^r_i \cup AT^o_i \cup AT^{ot}_i$.

5.3. Checking Satisfaction of Location and Mobility Constraints

To check location constraints in security policy, we apply model checking of Ambient Calculus specifications against Ambient Logic formulas, as presented in Section 4. The calculation of satisfaction relation for location configuration against location and mobility constraints is formalized as follows: for each authorization term $(as, ao, sa, co, fo) \in AT''$, where $LCONF$ is an ambient process specification for the current location configuration and $fo$ is an ambient logic formula within the authorization term, determine whether $LCONF \models fo$.

The set of authorization terms applicable for the current access request is further reduced according to the satisfaction of location constraints by the current location configuration of the system. The set of derived authorization terms which satisfy location constraints is $AT''^{loc} = \{(as, ao, sa, co, fo) \in AT'' | LCONF \models fo\}$.

5.4. Checking Generic and SOD Constraints Specified by Conditions

The method for checking conditions in the security policy rules is as follows: Where $PRED$ is the set of satisfied predicates in the current state of the system and $co$ is a condition, for each authorization term $(as, ao, sa, co, fo) \in AT''$, determine whether $PRED \models co$.

Since Predicate Calculus is utilized for specification of predicates and conditions in the security policy rule, the satisfaction relation is a logical entailment problem. We utilize an automated theorem prover for specification and verification of conditions. A detailed description of theorem prover support is out of scope of this paper. The calculation of SOD relations are presented in detail in Section 3.5.

The set of authorization terms applicable for the current access request, which satisfy conditions is specified as $AT^{co} = \{(as, ao, sa, co, fo) \in AT'' | PRED \models co\}$.

5.5. Evaluation of the Access Control Function

The access control function (ACF) matches a requested permission against the set of permissions assigned to roles in a certain location. The set $permissions^*(ru)$, denoting permissions for a role, derived from the set of authorization terms $at = (as, ao, sa, co, fo), at \in AT''$ evaluated against hierarchies, can be calculated as:

$permissions^*(ru) = (obj, act) \in \bigcup_{(ao, sa) \in P} at \in AT''$

When an access request $ar$ by a user $u$ with role $ru$ in a specific location $l_r$ to conduct action $act$ on object $obj$ is received, ACF $(ar)$ may return two values, allowed or denied. The definitions of the access control function is as follows.

**Definition 12.** Access Control Function ACF is:

$ACF(ar) = allowed \iff (l_r \in serv) \land (AT'' \neq \emptyset) \land (AT^{co} \cup AT^{loc} \neq \emptyset) \implies \exists (+, act) \in permissions^*(ru)$

$ACF(ar) = denied \iff (l_r \notin serv) \lor (AT'' = \emptyset) \lor \exists (-, act) \in permissions^*(ru)$

The interpretation of the access control function is, if the location of the user is within locations associated with the service, if there is any derived authorization terms according to evaluation of user name, role, role hierarchy, object or object hierarchy, if the location and separation-of-duty constraints are applicable to the set of derived authorization terms, then there is at least one rule with a permission that has the same object and action as those within the access request. The denial takes place if the request is placed from a location outside the scope of the service, there is no derived rule for the request, or there is at least one derived rule with a specific denial (negative sign). The order of evaluation of denials versus allowed accesses may depend on precedence of signs (denials take precedence, allowances take precedence).
6. Conclusions and Future Work

In this study, we presented a new formal security policy model for multi-domain mobile networks, called FPM-RBAC (Formal Policy Model for Mobility with Role Based Access Control). FPM-RBAC supports the specification of mobility and location constraints, role hierarchy mapping, inter-domain services, inter-domain access rights and separation of duty. Role hierarchy mapping, inter-domain services, inter-domain access rights are defined by a formal inter-domain security policy model. Within this model we have introduced the concept of Inter-Domain Roles and Inter-Domain Role Hierarchies. Roles are mapped to Inter-Domain roles with a role mapping function. An inter-domain service is accessible by home or foreign users through inter-domain roles and includes objects from home domain. The access rules for inter-domain services by foreign roles is achieved through the Inter-Domain Permission Assignment relation. This approach provides more flexible administration where a domain administrator does not need to know the roles and identities of foreign users or entities who are accessing resources in a domain and are enrolled in other domains. Also, need-to-know principle of security is preserved while information is shared between multiple domains.

Associated with FPM-RBAC, we also present a formal security policy constraints specification language for domain and inter-domain security policies. Formal policy constraint specifications are based on Ambient Logic and Predicate Logic. Ambient Calculus is used to specify the current state of a mobile network and actions within security policies for evaluation of access requests according to security policies. We take a formal process calculus approach because it is inefficient to represent mobility with relational models which is an approach in recent literature in the area of location based access control models. The formal location and mobility model within FPM-RBAC is capable of modelling actions in security policy rules as well as providing a dynamic spatial and temporal mobile network model. It provides a formal link between security policies and the mobile network.

As a result of this link, the use of the proposed formal policy model enables specification and analysis of inter-domain security policies. Formal analysis of security breaches arising from mobility within multiple security domains is possible. This analysis uses the model checking algorithm we have presented in our previous work. We have presented two examples of formal analysis in mobile networks, for two domains and multiple domains. By use of formal analysis, it is possible to construct a set of security policies towards prevention of security breaches arising from mobility.

In our ongoing research work, we are constructing automated tools for extracting formal specifications from XML-based security policies and automated verification of security policies for multiple-domain mobile networks. We aim in providing a complete policy framework for FPM-RBAC with an integrated specification, enforcement and verification environment.

Acknowledgements

This work is partially supported by the State Planning Organization of Turkey, TAM Project (2007K120610). We also acknowledge the comments by Dr. Tuna Tugcu.