Formal Model based Specification of Authorization Framework for Ubiquitous Enterprise Computing Environment

Supreet Kaur
PURCITM, Punjabi University, Patiala
India

Kawaljeet Singh, Ph. D
UCC, Punjabi University, Patiala
India

ABSTRACT

With the emergence of new dynamic computing environments, the traditional identity based authorization models are unable to meet multiple attribute based policy requirements through single function based access control model. There is need for a flexible and scalable authorization model that can meet the different protection requirement of the computing system and adapt to the demand of real world security requirements. In this paper a formal authorization model for ubiquitous computing environment is proposed. Ubiquitous computing environment demands a dynamic access control mechanism that can adapt to the changing security requirement of the computing environment. The proposed security model has taken these factors into consideration and adopted a formal approach to design a flexible and scalable model to support intelligent authorization process in ubiquitous computing environment.

Keywords

Access Control, Authorization, Formal Methods, Security Model, Ubiquitous Computing.

1. INTRODUCTION

The use of formal methods based approach has become an essential part of the development process of secure systems keeping in view the increased complexity of the computing environments. In order to achieve high degree of safety, reliability and security, formal methods play a very important role in system development process. Formal methods refer to the set of activities that rely on mathematically-based languages and tools for specification, verification and systems requirement validation. The formal specification helps in presenting the system in a precise and unambiguous way. A formal specification has well defined syntax and semantics based mathematical concepts drawn from set theory and Logic. Moreover formal methods can help in reducing high validation costs in case of complex system by discovering specification errors at an early stage and hence reduce the rate of system failures.

In this paper formal methods based approach is used for the development of specification framework for authorization model defined for secure ubiquitous enterprise computing environment [1-2]. The main advantage of the formal specification approach is that it allows reasoning about the security properties of the system which is very important feature for the successful implementation of the security mechanism based on the proposed model. In the proposed work the model based approach to formal specification is used to write detail specification for the ubiquitous authorization model. In model based approach a model of the proposed system is built using state machine based approach and the system management operations are defined for performing system state transitions. In literature different languages exists like VDM, OCL, B and Z notation that have been developed to formally specify the systems. The formal specification language used for development in proposed work is Z formal specification notation [3] that is based on set theory, first order predicate logic and schema calculus. In Z, states, as well as operations, are described with a two-dimensional notation called a schema. The Formal notation of Z provides well defined semantics and complement informal requirements specifications with formal description. The Z language notation provides high degree of expressiveness and precise specification for development of state based model.
The proposed model specification is based on state based ubiquitous authorization model that has been designed to provide the reliable authorization service of ubiquitous computing environment. The formal description is presented as small modules called schemas. Schemas are used to describe the state variables and to define constraints and set of management operations on system state. The proposed model based specification includes schemas defined primarily for describing state variables and operation with respect to security aspect of the ubiquitous computing environment.

1.1 Background and Related Work

Formal specification approach has been widely applied for the development of security system to achieve higher rate of reliability and efficiency. In the context of information security systems, it has become a standard feature to use formal specification techniques and tools before the implementation of the system. In view of complexity and heterogeneity of ubiquitous computing environments, formal specification approach provides precise, unambiguous and explicit specification of access control framework in order to achieve the organizational security objectives.

Ubiquitous computing environments are physical environments saturated with computing and communication, yet gracefully integrated with human users [4].These computing environment are collection of heterogeneous entities which are computationally autonomous. These computational entities are embedded in physical environment, distributed over network and interact with each other to provide smart service to the users. In recent years, the rapid growth of networking technologies and influx of smart devices has significantly promoted level of interaction among the computing entities in computing environments. The interactions between these autonomous entities are adhoc in nature and can happen in anywhere anytime mode. This ubiquity property imposes a new set of security challenges in computing environment [4].In order to provide secure service, the computing environment should incorporate secure access control mechanism.

The literature review reveals that recent approaches are moving towards the development of flexible frameworks with the support of multiple parameters and policies as per the requirements of the specific application environment. In traditional access control models like DAC [5], MAC [6] and RBAC [7-8] authorization decisions are determined according to identities of subjects and objects, which are authenticated by a system completely. Given the complexity of the scenario, the simple authorization function triple (subject, object, operation) is no more sufficient. Modern access control practices for emerging computing requirements require flexible authorization policies. A survey of the literature in this direction shows significant amount of work done for the development of models for the emerging computing environments. Al-Muhtadi et al. [9] in their paper presented a ubiquitous security mechanism that integrates context-awareness with automated reasoning to perform authentication and access control in ubiquitous computing environments. Kim et al. [10] in their paper proposed an extended RBAC model to deal with context, which dynamically adjusts role assignments (UA) and permission assignments (PA). Sampemane et al. [11] in their paper studied the problem of access control for new emerging environments, which are called Active Spaces. Song-hwa et al. [12] in their paper proposed new access control model supporting time and location dimensions. The proposed access control model can effectively support various ubiquitous computing environments. Wang et al. [13] in their paper presented a usage control model to protect services and devices in ubiquitous computing environments, which allows the access restrictions directly on services and object documents. Lin at al. [14] in their paper presented a flexible, autonomous and non-redundancy access control model for ubiquitous computing environment which dynamically grants and adapts permissions to users based on context information including time, location and trust value. Hung et al. [15] in their paper proposed Activity-Oriented Access Control (AOAC) model, aiming to support user's activity in ubiquitous environments. Filho et al. [16]in their paper proposes a generalized context-based access control model for making access control decisions completely based on context information. Sejong et al. [17] in their study proposed a new access control model termed the Ubi-RBAC model. It is based on the RBAC model and adds new components such as space, space hierarchy, and context constraints.

Major work in the literature has focused on role based access control model and proposed new model around it only. Less attention has been paid on the rest of approaches like attribute based approach or policy based approach. Recently attribute based approach has been standardized with the draft publication by NIST [18]. ABAC is a logical access control methodology where authorization to perform a set of operations is determined by evaluating attributes associated with the subject, object, requested operations, and, in some cases, environment conditions against policy, rules, or relationships that describe the allowable operations for a given set of attributes.

In our proposed work our target system is ubiquitous computing environment. The ubiquity property of the computing environment and anywhere anytime computing system complicate the implementation of reliable authorization service as it demands a dynamic and an intelligent approach towards modeling of authorization framework. Ubiquitous computing environment involves arbitrary attributes of system entities in authorization process. In this work attribute centric authorization framework for managing authorization in ubiquitous computing environment is proposed. The Z notation is found as most suitable choice for development of a complex authorization model specification.

2. FORMAL SPECIFICATION

Formal specification is an approach for description of relevant properties of a system using formal notation. The choice of specification notation used in our proposed work is based certain characteristics described below.

- **Generic Specification**

The specification notation should result in a generic specification and not inclined towards certain programming paradigm or tied to the specifics of a particular implementation language. It should complement informal requirements specification techniques

- **Precision and Unambiguous**

The specification notation should support precise and Unambiguous specification. The specification format should match naturally for expressing the target system with a minimum of needs to compromise. It should help in removing area of ambiguity in specification and avoid any instance of misinterpretation.

- **Abstraction and High degree of Expressiveness**

The specification notation should support high degree of expressiveness to specify the system details to the desired level. This would allow system components to be stated at multiple levels of abstraction and allow model components reusability. Good specification notation will simplify the system behavioral specification.

- **User friendly**

In general Formal specification is found to be difficult and tedious to read especially when specifications are purely based on mathematical constructs and formulae. To be user friendly formal specification must be supplemented by supporting informal text and description.

- **Support for Formal analysis and Validation.**

The principle advantage of using Formal method approach is that it forces an analysis of the system requirements at an early stage. The Formal specification notation should support interaction with computer tools for verifying notation, as a verified model specification reduces the system failure rate by highlighting errors at an early stage of system development and increases the level of assurance.

Based on the above considerations and criteria a survey of various specification notation like B, OCL, VDM and Z was performed. The standard Z notation is selected as specification notation due to the fact that it is generic, precise, user friendly and support model analysis and verification.

2.1 The Formal Specification Notation

Formal specification notation has been widely used for development of model based systems. Among various mature notations the Z formal specification notation is well established typed formal specification language and has been adopted as an international standard through the publication of ISO/IEC 13568:2002.

In Z, systems are modeled using concepts based on set theory and first order predicate calculus. It provides a precise syntax and semantics based on mathematical constructs for the abstract specification of system. The main feature of the Z is a way in which formal description is split into small pieces called schemas. Schemas are considered as building block of Z specification. These schemas are used to describe both static and dynamic components of the system. The static component defines state and the system invariant that system should maintain while transition from one state to another state. The dynamic component defines the set of system management operation. The structural representation of the elementary schema used in Z specification is as follow.

SchemaName

Signature

Predicates

Figure 1. The structure of the Z Schema.

The Z schema presentation provides structure to the specification. A schema has shown in Figure 1 has two sections. The Schema signature section defines the state elements used to define the state of system. The Schema predicates section defines the conditions that must always be satisfied for these state elements. The elements in the signature section are called components of the schema. A schema can be considered as a set of named components that constrained by predicates. This can be specified as follow.

$$SchemaName \triangleq [Signature \mid Predicates]$$

When Schema is used to define system operation , the predicate section may have pre-conditions and post-conditions which are used to define state before and after the operation. The 'before' state of the schema is indicated by unprimed variables, the 'after' by primed variables. The Table 1 briefly explains the Z notations used in our proposed model.

3. MODEL BASED SPECIFICATION

In this section the development of model oriented formal specification in Z for Ubiquitous Authorization model is presented.

3.1 Formal Model Framework

In this section a general schema of formal model for Ubiquitous computing system is described. The proposed model is referred as Ubiquitous authorization model (UAM). The proposed model captures the system state from secure authorization service perspective only. The state machine based approach is adopted to model the system state. The system is represented as collection of entities with properties relevant to describe the secure state of the system. The system state machine describes the instance of the system that should satisfy the system security invariant to qualify as secure state. The system undergoes transition whenever there is a change in the state security variables or request of new system operation is generated. The system after transition will be continue in secure state subject to fulfillment of the criteria of system security invariant.

The set of system states of the system is represented as *UbSysState*. The set *UbSysState* describes the system security related state variables and relation of the system. The user access request is represented as *AccReq*. The system will be in secure state when all allowed access request are covered under the system authorized permissions represented as *UbAuthPerm*. The criteria for the secure state is defined as security invariant that need to be satisfied for any system state to be secure state. The transition function defines the state transition on execution of system operation on behalf of system users. The system operation defines the security criteria at operation level that should be satisfied for successful execution of the operation. The derived state after system transition is represented as *UbSysState′* and is evaluated against the security invariant to be an eligible secure state. Formally the generic schema of UAM is specified as follow.

UAMSM

$UbSysStates : \mathbb{P}\ State$

$AccReqs : \mathbb{P}\ UbACCREQ$

$AccPerms : \mathbb{P} UbACCPERM$

$Tf : \mathbb{P} UbMOp \times UbSysState \rightarrow UbSysState'$

$UbIntState : \mathbb{P}\ State$

$Dom\ Tf \subseteq \mathbb{P} UbMOp \times UbSysState$

$Ran\ Tf \subseteq UbSysState$

The set *UbMOp* describes the system management operations related to access control and system administration. The transition function *Tf* describes the transition from one state to another state by applying one or a sequence of operations from the set *UbMOp*. After defining the generic schema of the proposed model the development process is initiated. The formal development process is divided into the following phases.

1. Formal definition of Model Basic Entities of System: In this phase a precise definition of model basic entities are specified. These definitions will be used to build the formal specification for the proposed model.
2. Formal definition of Model State Security Variables: In this phase a precise definition of model state security variables are specified. These definitions are used to describe the security information in terms of security function for the state machine model specification.
3. Formal definition of Abstract State: In this phase a precise definition of model abstract state is specified. Abstract state specification describes the relationship between the various components of the model and is used represent the system state at abstract level.
4. Formal definition of Initial State: In this phase a precise definition of model initial state is specified.
5. Formal Definition of System Management Operations: In this phase a precise definition of model system management operation is specified. These functions control the changes of model state variables as per constraints defined under system operations.
6. Formal Definition of Model Secure State Invariant: In this phase secure state invariant for the model is developed. In order to implement a secure authorization system, the proposed model should maintain secure state of the system. To maintain the secure state of the system a security criteria as system security invariant is defined.

7. Model Implementation, Analysis and Verification: In this phase formal model analysis and verification is performed. The Formal specification notation developed in previous phases can be analyzed and verified using computer tools designed for model verification. A verified model specification reduces the system failure rate by highlighting errors at an early stage of system development and increases the level of assurance.

3.2 UAM Basic Elementary Sets

In this section the elementary sets used to describe the ubiquitous authorization model is introduced.

- **Subject:** Subject refers to an entity, which can be a user or a process on behalf of users. A user is an entity that generates an access request for accessing resources in the ubiquitous computing environment. The set of all subjects is represented as UbSUB. Formally, a subject is associated to different sets through the association functions.

- **Object:** This set includes the set of all entities designated as object in ubiquitous computing system. The set of all objects is represented as *UbOBJ.*

- **Environment Domain:** Environment domain represents the computing environment or location or surroundings in which an object resource is being accessed by the subject. The set of environment domains for ubiquitous computing environment can be represented as UbENVDOM.

- **System Entities:** The collection of all active computing system elements like subject , objects and Computing domains can be represented under universal set called Entities. This set of all computing elements is represented as UbSYSENTITY.

3.3 UAM State Security Elements

- **Subject Attribute**: Subject Attribute represent characteristic relevant to subject/user like subject id, subject name etc. The set of Subject attribute is represented as UbSUBATT. Formally subjects are associated with a set of attributes through many to many relation ship.The function *AssignAttS* returns the list of subjects attributes to which a given subject is associated

  $$\text{AssignSAtt} : \text{UbSUB} \rightarrow \mathbb{P}\ \text{UbSUBATT}.$$

- **Object Attribute**: Object Attribute represent characteristic relevant to subject/user like subject id, subject name etc. The set of Subject attribute is represented as UbOBJATT. Formally a subject is associated with a set of attributes through a function

  $$\text{AssignOAtt} : \text{UbOBJ} \rightarrow \mathbb{P}\ \text{UbOBJATT}.$$

- **Environment Domain Attribute**: Ubiq-Env domain attribute represent characteristic relevant to Environment domain under consideration during active access operation. The set of Environment domain attribute is represented as UbENVDOMATT. Formally a environment domain is associated with a set of attributes through a function

  $$\text{AssignEDAtt} : \text{UbENVDOM} \rightarrow \mathbb{P}\ \text{UbENVDOMATT}.$$

- **Contextual Attribute:** Contextual Attribute represents an additional constraint as a part of applicable policy rule or as security parameter that can be used to determine the applicability of the access control policy in particular state. Contextual Attribute is dynamic and can be associated with subject, object or environment. The set of Context Attribute is represented as UbCTXATT. Formally a subject ,object or any system entity is associated with a set of contextual attributes through a function

  $$\text{AssignCtxAtt} : \text{UbSUB} \rightarrow \mathbb{P}\ \text{UbCTXATT}.$$

- **Trust Attribute:** Trust attributes represents the trust oriented constraints. Trust oriented constraints are represented as an edge in an Attribute based Trust relation graph and assigned a trust value. The set of Trust Attribute is represented as UbTRUSTATT. Formally a subject ,object or any system entity is associated with a set of trust attributes through a function

  $$\text{AssignTrAtt} : \text{UbSUB} \rightarrow \mathbb{P}\ \text{UbTRUSTATT}$$

- **System Attributes:** The collection of all computing system elements properties like attributes of subject, objects, computing domains, context attributes and trust attributes can be represented under universal set called System Attributes. This set of all computing elements properties is represented as UbSYSATT.

- **Access Operation:** An access operation is defined as action that subject is authorized to perform on an object as per access policy. The type of operation will depend upon the protected system entity and computing environment. The set of Access Operations is represented as UbACCOP. The Access List represents the operation that can be performed on an object.

  $$\text{UbACCLIST} : \text{UbACCOP} \times \mathbb{P}\text{UbOBJ}$$

- **Authorization Policy:** The authorization policy represents a function that maps set of subject attribute, a set of object attributes and set of environment attributes to set of access operations. Formally the authorization policy function can be specified as follow.

  $$\text{AUTHPOL}: \mathbb{P}\text{UbSUBATT} \times \mathbb{P}\ \text{UbOBJATT} \times \mathbb{P}\text{UbENVATT} \times \mathbb{P}\ \text{UbCOND} \rightarrow \mathbb{P}\ \text{UbACCOP}$$

- **System Access Permission:** The access permission allows subject with specific attributes to perform specific operation on an object. The semantics of the permission *p(Sub,Obj,AccOp)* specifies the prerequisites that subject should satisfy before being allowed to exercise the set of privileged operation over resource object. Such permission is called conditional permission. The conditions of such permission are represented as referenced security parameters RSP(). These referenced parameters can be authorization policy, attribute constraint, contextual constraint, trust based constraint, semantic property constraint or any other constraint defined in the model. Formally the Access permission function can be specified as follow.

  AccessPerm(P): $\mathbb{P}\ RSP_1 \times \mathbb{P}\ RSP_i \times \ldots \times \mathbb{P}\ RSP_n \rightarrow \mathbb{P}\text{UbAccOp}$

- **System Management Operation:** The UAM Management Operation set MOp_{UAM} is a collection of Administrative and User operations for management of the UAM model components. These functions control the changes of model state variables as per constraints defined under system operations. The set of operation can be represented as UbMOP. The set of user access request operation UbACCREQ is considered as subset of System Management operations.

3.4 Abstract State

After defining basic state variables and generic schema of UAM model, the formal specification of the Abstract State of the UAM model is presented in this section. Abstract state precisely represents state variables and invariants on them. The following schema captures the abstract state of the UAM model.

UAMS

$Subs : \mathbb{P}\ UbSUB; Objs : \mathbb{P}\ UbOBJ; EnvDoms : \mathbb{P} UbENVDOM$

$SubAtts : \mathbb{P}\ UbSUBATT; ObjAtts : \mathbb{P}\ UbOBJATT$

$EnvDomAtts : \mathbb{P}\ UbENVDOMATT; CtxAtts : \mathbb{P}\ UbCTXATT$

$TrustAtts : \mathbb{P}\ UbTRUSTATT; SysEnitites : \mathbb{P} UbSYSENTITY$

$SysAtts : \mathbb{P} UbSYSATT; AccOps : \mathbb{P}\ UbACCOP$

$AttConsts : \mathbb{P} ATTCONSTCLAUSE;$

$AccPermPols : \mathbb{P}_1\ UbACCPERMPOL;$

$SysAuthPols : \mathbb{P}\ UbPOLRULES$

$AuthStatePerms : \mathbb{P} UbSTATEPERM ; Rsps : \mathbb{P}_1\ UbRSP$

$AssignSAtt : UbSUB \rightarrow \mathbb{P}\ UbSUBATT$

$AssignOAtt : UbOBJ \rightarrow \mathbb{P}\ UbOBJATT$

$AssignEDAtt : UbENVDOM \rightarrow \mathbb{P} UbENVDOMATT$

$AssignCtxAtt : UbSYSENTITY \rightarrow \mathbb{P}\ UbCTXATT$

$AssignTrAtt : UbSYSENTITY \rightarrow \mathbb{P}\ UbTRUSTATT$

$AuthStatePerms : \mathbb{P} UbSUB \times \mathbb{P} UbOBJ \times \mathbb{P} UbACCOP$

$PermSub : UbACCPERMPOL \rightarrow \mathbb{P} UbSUB$

$PermObj : UbACCPERMPOL \rightarrow \mathbb{P} UbOBJ$

$PermAccOp : UbACCPERMPOL \rightarrow \mathbb{P} UbACCOP$

$PermCondCompl : UbACCPERMPOL \rightarrow \mathbb{P} ATTCONSTCLAUSE$

$PermSEAtt : UbACCPERMPOL \times UbSYSENTITY \rightarrow \mathbb{P} UbSYSATT$

$PermRsps : UbACCPERMPOL \rightarrow \mathbb{P} RSP$

$SysOp : \mathbb{P} UbMOP$

$Dom\ AssignSAtt = Subs; Ran\ AssignSAtt \subseteq Subatts$

$Dom\ AssignOAtt = Objs; Ran\ AssignOAtt \subseteq Objatts$

$Dom\ AssignEDAtt = EnvDoms; Ran\ AssignEDAtt \subseteq EnvDomatts$

$Dom\ AssignCtxAtt = SysEntities; Ran\ AssignCtxAtt \subseteq CtxAtts$

$Dom\ AssignTrAtt = SysEntities; Ran\ AssignTrAtt \subseteq TrustAtts$

Figure 2. Abstract state of UAM model

The schema in Figure 2 captures the abstract state of UAM model. The abstract schema defines the state of the system with basic state variables like Subject, Object, Environment Domain, Basic Attributes, Context Attributes and Trust Attributes. The Subject represents an entity that initiates access request to access a resource of the system. Object represents an entity that is designated as a resource in the system and accessed by the other entities designated as subjects. Environment domain represents the computing environment or location or surroundings in which an object resource is being accessed by the subject. Attributes are security relevant characteristics. Subject Attribute represent characteristic relevant to subject/user like subject id, subject name etc. Object Attribute represent characteristic relevant to object/resource like object id, object name etc. Ubiquitous environment domain attribute represent characteristic relevant to Environment domain under consideration during active access operation. Contextual Attribute represents the state conditions that can be used to determine the applicability of the access permission in particular state or as an additional constraint as a part of applicable policy rule. Trust attributes represents the trust oriented constraints that can be used as an additional constraints as part of authorization decision process. Security Policy rules represents collection of finite set of security policy rules built over security parameter associated with system entities. An access operation is defined as action that subject is authorized to perform on an object as per access policy. The type of operation will depend upon the protected system entity and computing environment.

The function Assign*SAtt* returns the list of attributes associated with a subject. The function Assign*OAtt* returns the list of attributes associated with an object. The function Assign*EDAtt* returns the list of attributes associated with an environment domain. The function Assign*CtxAtt* returns the list of contextual attributes associated with a system entity. The function Assign*TrAtt* returns the list of trust attributes associated with a system entity. The *Acclists* represents set of access lists in terms of allowed access operation on system object entities.

The Access Permission Policy represents a set of permissions where individual permission comprises of reference security parameters as Boolean expression to decide the access requests for system resources. The structure of Access permission can be specified as follow.

The Access permission is identified with unique ID defined as permission ID.The access permission has *Permproperty* that represents a set of attribute clauses that restricts the applicability of the access permission. Further the access permission has a component defined as reference security parameters. The referenced security parameter can be an access control policy, knowledge based rules, facts, attribute constraints or any other constraint expressed as Boolean attribute expression. For each access permission there is non-empty set of *rsp* statements that may evaluate to true to allow access request or false to deny the access request.

AccessPermissionPol

$Id : PermissionID$

$permproperty : \mathbb{P}_1 AttConstClause$

$rsps : \mathbb{P}_1 RSP$

The *rsp* is also identified with unique id defined as *RSP ID.*Each *rsp* also has component defines as *Rspproperty* that represents a set of clauses that restricts the applicability of the rsp statement. The RSP also has a precondition component that is used to add granularity in the process of authorization. The structure of RSP can be represented as follow.

RSP

$Id : RSPID$

$rspval : RSPVal$

$rspproperty : \mathbb{P}_1 AttConstClause$

$preconditons : \mathbb{P}\ Precondition$

The *AttConstClause* contain multiple system attribute where each attribute can have single or set of values as a part Boolean expression. The structure of *AttConstClause* can be represented as follow.

AttConstClause

$sysAtts : \mathbb{P} UbSYSATT$

$attvals : \mathbb{P}_1 AttVal$

The Precondition component has domain that defines precondition type. The Precondition has a component *CondAttProperty* that is represented as set of clauses. The clause is composed of attributes relevant to the type of condition domain. These attributes can have single or set of values that are evaluated as a part of decision making process. The structure of the Precondition can be represented as follow.

Precondition

$Dom : PreCondDom$

$CondAttproperty : \mathbb{P}_1 AttConstClause$

The Access Permissions implemented in UAM based system defines the authorization matrix for the system, based on which all access requests for the system resources is decided. In UAM based system the access rights are derived from access permission defined as attribute constrained policies.

3.5 Initial State of the Model

After defining Abstract state of the model in the previous section, the UAM model initialization is presented in this section. The following schema captures the initial state of the UAM model. The following model initialization instance is an example where each model elementary set is initialized with empty set. As a result, the relationship between all elementary sets is also represented as null set. For secure system, initial state should also be a secure state i.e. initial state should also satisfy the secure state invariant. In this case the initial state can be assumed to be secure state as all its state variables initialized to null value i.e. with the condition $Subs = \phi$, it is sufficient for the state to be declared as secure state irrespective of the status of the other state variables .In order to make transition from initial state to next secure state administrative function is required to add subject into the system. In the case of initial state where $Subs \neq \phi$, the state s_0 is said to be secure if it comply with all the conditions of as defined for secure state invariant.

For another more realistic initial state where $Subs \neq \phi, Objs \neq \phi$ an example initial state is considered with assumption that the initial system state s_0 is defined in such a way that it satisfies all the conditions of the system secure state. For state initialization of the UAM to be correct and secure, the Subjects, Objects and EnvDom should be valid entities created by the system administrator as per organization policy. The system attributes that includes attributes of subjects, objects, environment domain, context and trust should be valid attribute as per entity attribute relations and should be defined for the state. The list of access operation should be the valid operations as per organization security policy and should be defined as a part of access operation set. The association of attributes with the system entities should be though entity attributes association function and all attributes should be assigned a value from a valid attribute value range. The list of access operation that system user intend to perform on system entities should be defined as a part of access list. The reference security policy defined for access permission should be valid policy defined under reference security policy set. The access permission under which system user is authorized to perform access operations on object entities should be defined as part of Access Permission set. The Access permission set should be defined as per organization policy and security requirements. The example initial state is described through following schema.

4. SYSTEM OPERATION

System administrative operations are used for the management of UAM State security variables. In this section the administrative operations for the management of basic sets of UAM model are described along with security predicate. The system operations are described using schema notation of Z.

4.1 System Admin Level Operation

Add Subject: This is an operation used to create a new system entity subject *Sub*. The Add Subject system operation with security predicate is statically represented as *AddSubject* schema.

AddSubject

$\Delta UAMS$

$sub? : Subs$

$sub? \notin Subs\ ;\ Subs' = Subs \cup \{sub?\}$

$AssignSAtt' = AssignSAtt \cup \{sub? \rightarrow \varnothing\}$

$AssignCtxAtt' = AssignCtxAtt \cup \{sub? \rightarrow \varnothing\}$

$AssignTrAtt' = AssignTrAtt \cup \{sub? \rightarrow \varnothing\}$

Add Subject Attribute: This is an operation used to create a new system entity subject attribute *SubAtt.* The Add Subject Attribute operation with security predicate is statically represented as *AddSubAtt* schema.

AddSubAtt

$\Delta UAMS$

$sub?: Subs$

$sa? : SubAtts$

$sa? \notin SubAtts\ ;\ SubAtts' = SubAtts \cup \{sa?\}$

$\forall sub \in Subs \bullet\ sa? \notin AssignSAtt'(sub?)$

Add Contextual Attribute: This is an operation used to create a new system entity contextual attribute *CtxAtt.* The Add Contextual Attribute system operation with security predicate is statically represented as *AddCtxAtt* schema.

AddCtxAtt

$$\begin{array}{l}
\Delta UAMS \\
se?: SysEntities \\
ca?: CtxAtts \\
\hline
ca? \notin CtxAtts\ ; CtxAtts' = CtxAtts \cup \{ca?\} \\
\forall se \in SysEntities \bullet\ ca? \notin AssignCtxAtt'(se?)
\end{array}$$

Add Trust Attribute: This is an operation used to create a new system entity trust attribute *TrAtt.* The Add Trust Attribute system operation with security predicate is statically represented as *AddTrustAtt* schema.

AddTrustAtt

$$\begin{array}{l}
\Delta UAMS \\
se?: SysEntities \\
ta?: TrustAtts \\
\hline
ta? \notin TrustAtts\ ; TrustAtts' = TrustAtts \cup \{ta?\} \\
\forall se \in SysEntities \bullet\ ta? \notin AssignTrAtt'(se?)
\end{array}$$

Add Access Permission Policy: This is an operation used to create a new access permission policy. The Add Access Permission Policy system operation with security predicate is statically represented as *AddAccPermPol* schema.

AddAccPermPol

$$\begin{array}{l}
\Delta UAMS \\
p? : AccPermPols \\
rsp? : Rsps \\
\hline
p? \notin AccPermPols\ ;\ AccPermPols' = AccPermPols \cup \{p?\} \\
PermRsp' = PermRsp \cup \{p? \rightarrow rsp?\}
\end{array}$$

4.2 System User Level Operation

In the following section system user level operations are described using schema notation. The purpose of user level system operation is to allow user interact with the system in secure manner. The security conditions of the user level operation are defined to allow only authorized transactions in the system and secure transition from one state to another state.

Access Request Evaluation: This is an operation used to evaluate the access request generated by the system user. The Access Request Evaluation system operation with security predicate is represented as A*ccReqEval.*

After defining the model components, initial state, abstract state and set of operation the next section describes model secure state invariant.

AccReqEval

$$\begin{array}{l}
\Xi UAMS \\
ar? : AccReqs \\
sub? : Subs \\
obj? : Objs \\
accop? : AccOps \\
effect? : PolEffect \\
p? : AccPermPols \\
c? : Preconditions \\
result! : DECISION \\
\hline
\exists\ p : UbACCPERMPOL \bullet\ p? \in AccPermPols \wedge ar.Sub? \in PermSub(p) \\
\wedge\ ar.obj? \in PermObj(p) \wedge ar.accop? \in PermAccOp(p) \\
\wedge\ ar.c\ ? \in PermCondCompl(p) \\
\wedge\ PermEffect(p) = Permit\ \Rightarrow Result! = Allow \\
\exists\ p : UbACCPERMPOL \bullet\ p? \in AccPermPols \wedge ar.Sub? \in PermSub(p) \\
\wedge\ ar.obj? \in PermObj(p) \wedge ar.accop? \in PermAccOp(p) \\
\wedge\ ar.c\ ? \in PermCondCompl(p) \\
\wedge\ PermEffect(p) = Deny\ \Rightarrow Result! = NotAllowed
\end{array}$$

5. UAM MODEL STATE INVARIANT

In the previous sections the abstract state along with example initial state of the model is defined. The abstract state of the model captures the basic model sets along with model state security variables. In order to implement a secure authorization system, the proposed model should maintain secure state of the system. To maintain the secure state of the system a security criteria as system security invariant is defined. The security invariant is defined at two levels. First level is system state level and second level is system transition function level. In this section security invariant at system state level is described followed by description of security invariant at transition function level.

5.1 System State Level Invariant

In this section system state level invariant is defined that is used to characterize the secure state of the system. To formulate the criteria for secure state for authorization model there is a need to identify and consider all the security properties with respect to authorization service that must hold for state confirmation as secure state. In the following the process of identifying the security properties that will form the criteria for secure state under ubiquitous authorization model is initiated.

1. The first phase of user/subject interaction with the ubiquitous computing environment is considered initially. In this phase it is assumed that user with basic credentials has been successfully authenticated by system authentication service. In secure state perspective, all active subjects/users of the system are assumed to belong to authenticated subject/user list. To check the compliance in this context, secure state property termed as Secure System Authentication Service Property can be defined.

2. In Ubiquitous computing environment the authorization service works on the basis of the criteria defined in terms of system entity attributes. In secure state perspective all assignment of the system entity attributes are assumed to be

valid. To check the compliance in this case, secure state property as Secure System Attribute Assignment Property can be defined.

3. In the next phase user submit a request to access a particular service/resource to the system authorization service. Based on the subject/user credentials, the relevant access permission policy will be assigned. In secure state perspective all access permission of the system are assumed to be valid permissions. To check the compliance in this case, secure state property as Secure System Access Permission Validity Property can be defined.

4. The access permission will define reference security policies and conditions under which the authorized access to system resources will be allowed. In secure state perspective all current accesses are assumed to be covered under valid system access permission. To check the compliance in this context, secure state property termed as Secure System Resource Access Property can be defined.

After defining security properties for a given state, system state level security invariant can be characterize by the following security properties.

- System Authentication Service Compliance Property
- System Attribute Assignment Compliance Property
- System Access Permission Compliance Property
- System Authorized Resource Access Compliance Property

These security properties collectively define the criteria that must hold for a state to be a secure state. In the following the specification of properties using Z Notation is presented.

After defining the security properties, the secure state of the system can be defined in terms relationship between secure state and security properties that system should satisfy. Formally this can be represented as follow.

$$\text{UAMSS} \triangleq \text{AuthServProp} \wedge \text{AttrAssignProp} \wedge \text{AccPermProp} \wedge \text{AuthResAccProp}.$$

5.2 System Transition function Level Invariant

In this section a system transition function level invariant that is used to characterize the secure state of the system is presented. To formulate the criteria at the system transition function level, the preconditions of all the system operation those are required to maintain the secure state are identified and represented as predicate in the schema representation of each operation.

The UAM system operation set is a collection of Administrative and User operations for management of the UAM model components. These operations control the changes of model state variables as per precondition defined under system operations. The precondition of each system operation that is part of system transition function together constitutes system transition function level security invariant. Each system transition should satisfy the system transition function level invariant to maintain secure states.

6. UAM MODEL IMPLEMENTATION

In the previous section, an authorization model is proposed for ubiquitous computing environment that provides the formal security framework for implementation secure authorization system. In this section the process of model implementation and verification to complete the proposed development process is presented. In order to implement the model the following components are required

- Model Abstract State
- Model Initial State
- Model State Transition Function
- Model System Management Operations
- Model State Security Invariant.

The state security invariant described as a part of formal model defines the security criteria for implementation of secure authorization system. The section 5.4 provides the detail description of all the components required for implementation. The model implementation can be represented as follow.

$$\Delta\text{UAMS} \triangleq [\text{UAMS}, \text{UAMS}' \mid \text{Tf}\,]$$

Where transition function $Tf : \mathbb{P}UbMOp \times UAMS \rightarrow UAMS'$ .The transition function is composed of system management operation performed in a particular sequence and it allows the system to transit from one state to another state. The transition function for the proposed model can be represented as follow.

$$\begin{array}{|l}
\Delta UAMS \\
\hline
uams : \mathbb{P}\ State \\
uams' : \mathbb{P}\ State \\
ss : \mathbb{P}SecState \\
mop : \mathbb{P}UbMOp \\
tf : \mathbb{P}Tf \\
\hline
\forall\, mop \in tf\ ;\ uams \in ss \bullet\ \ tf(uams) \in ss \\
\hline
\end{array}$$

After defining the model components, the security theorem for model can be defined as follow.

Definition: The proposed model is said to be secure if and only if

1. The initial state is a secure state.
2. The transition function satisfies all the security invariant defined at the system operation level for all member system operation.
3. All the states reachable from initial state satisfy the system level security invariant and are secure state.

The implementation of authorization system based on proposed model is said to be model compliant if it satisfies all the conditions of model security theorem.

7. CONCLUSION

In this paper, the key components of Network Access Control Model are formalized in order to be sharp, precise and prevent their multiple interpretations. A state based approach to the model is adopted, and precisely represented the operations involved in the access control domain. The model is represented using the well known formal notation, Z. Most of the schemas used in the model were between 5 to 12 lines long. The schema describing the basic system elements was large due to multiple security constraints of network computing environment. The model proposed in this paper addresses only the access control aspect of security for the network system and is intended to act as baseline for this area. Since multiple aspects of access control like discretionary control, mandatory control, multiple level security and role base access control had to be integrated and expressed in terms of target structure, level of complexity was a significant issue. The level of complexity was controlled by using Z notation to provide a simplified exposition of integrated rules, without allowing the formal notation to add to the complexity. In future work the use of logical formalism based approach is planned to produce an animation of the formal specification to further refine the framework. Because both Z and Symbolic computation languages are based on predicate logic this would be easier and straightforward task to carry out further refinements in the model to meet the new challenges of the information security systems and improve upon the existing framework.

8. REFERENCES

[1] M. Weiser, "The Computer for the Twenty-First Century," in Scientific American, vol. 265, 1991, pp. 94-104.

[2] M. Weiser, Ubiquitous Computing, Computer, v.26 n.10, p.71-72, October 1993

[3] Z-FSN. (2002).Information technology — Z formal specification notation.Syntax, type system and semantics, ISO/IEC 13568:2002(E), International Standard

[4] Lyytinen, K., Yoo, Y. 2002. Issues and Challenges in Ubiquitous Computing. Communications of the ACM45:62-65

[5] Lampson, Butler W. (1971). "Protection". Proceedings of the 5th Princeton Conference on Information Sciences and Systems. p. 437.

[6] Ravi S. Sandhu. Lattice-based access control models. IEEE Computer, 1993.

[7] David F. Ferraiolo and D. Richard Kuhn, "Role-Based Access Controls," Proceedings of the 15th NIST-NSA National Computer Security Conference, Baltimore, Maryland, October 13-16, 1992

[8] Ravi S. Sandhu and P. Samarati. Access control: Principles and practice. IEEE Com. Mag., 1994.

[9] Al-Muhtadi, J., Ranganathan, A., Campbell, R., Mickunas, M.D. (2003).Cerberus: a context-aware security scheme for smart spaces," Pervasive Computing and Communications, 2003. Proceedings of the First IEEE International Conference on, vol., no., pp. 489-496.

[10] Kim, Y., Moon, C.,Jeong, D., Lee,J., Song,C. and Baik, D. (2005). Context-aware access control mechanism for ubiquitous applications. In AWIC, pp. 236–242.

[11] Sampemane, G. (2005). Access Control for Active Spaces. Doctoral Thesis. UMI Order Number: AAI3199131., University of Illinois at Urbana-Champaign.

[12] Song-hwa, C.,Wonil, K. and Dong-kyoo, K.(2006). Role-Based Access Control Model for Ubiquitous Computing Environment, LNCS, Volume 3786,pp. 354-363.

[13] Wang, H.,Zhang, Y. and Cao, J.(2008). Access control management for ubiquitous computing.Future Generation Computer, pp. 870-878.

[14] Lin L. and Tianjie C., 2008. A Flexible, Autonomous and Non-redundancy Access Control for Ubiquitous Computing Environment. In Proceedings of the 2008 International Symposium on Information Science and Engieering - Volume 01 (ISISE '08), Vol. 1. IEEE Computer Society, Washington, DC, USA, 446-450

[15] Hung, L., Shaikh, A., Jameel, H., Raazi, S., Yuan, W., Ngo, T., Truc, P., Sungyoung, L., Heejo L.,Yuseung, S.,Fernandes, M.(2009).Activity-Oriented Access Control for Ubiquitous Environments. 6th IEEE Consumer Communications and Networking Conference, pp.1-5.

[16] Filho, J., Martin, H.(2008).Using Context Quality Indicators for Improving Context-Based Access Control in Pervasive Environments. IEEE/IFIP International Conference on Embedded and Ubiquitous Computing, pp. 285-290.

[17] Sejong, O.(2010). New role-based access control in ubiquitous e-business environment. *Journal of Intelligent Manufacturing,pp.* 607-612.

[18] Hu, T.C., Ferraiolo, D., Kuhn, R., Friedman, A.R., Lang, A.J., Cogdell, M.M., Schnitzer, A., Sandlin, K.,Miller, R., Scarfone, K(2013).Guide to Attribute Based Access Control (ABAC) Defnition and Considerations (Draft) ,NIST Special Publication 800-162.