Safety of ABAC$_\alpha$ Is Decidable

Tahmina Ahmed(✉) and Ravi Sandhu

Institute for Cyber Security and Department of Computer Science,
University of Texas at San Antonio, One UTSA Circle, San Antonio, TX 78249, USA
tahmina.csebuet@gmail.com, ravi.sandhu@utsa.edu

Abstract. The ABAC$_\alpha$ model was recently defined with the motivation to demonstrate a minimal set of capabilities for attribute-based access control (ABAC) which can configure typical forms of the three dominant traditional access control models: discretionary access control (DAC), mandatory access control (MAC) and role-based access control (RBAC). ABAC$_\alpha$ showed that attributes can express identities (for DAC), security labels (for MAC) and roles (for RBAC). Safety analysis is a fundamental problem for any access control model. Recently, it has been shown that the pre-authorization usage control model with finite attribute domains (UCON$^{\text{finite}}_{\text{preA}}$) has decidable safety. ABAC$_\alpha$ is a pre-authorization model and requires finite attribute domains, but is otherwise quite different from UCON$^{\text{finite}}_{\text{preA}}$. This paper gives a state-matching reduction from ABAC$_\alpha$ to UCON$^{\text{finite}}_{\text{preA}}$. The notion of state-matching reductions was defined by Tripunitara and Li, as reductions that preserve security properties including safety. It follows that safety of ABAC$_\alpha$ is decidable.

Keywords: ABAC$_\alpha$ · Safety

1 Introduction

Attribute-Based Access Control (ABAC) is gaining attention in recent years for its generalized structure and flexibility in policy specification [2]. Considerable research has been done and a number of formal models have been proposed for ABAC [3–6,8,10]. Among them UCON$_{ABC}$ [6] and ABAC$_\alpha$ [4] are two popular ABAC models. UCON$_{ABC}$ has been defined to continuously control usage of digital resources which covers authorizations, obligations, conditions, continuity and mutability, while ABAC$_\alpha$ is defined to configure DAC, MAC and RBAC which shows that attributes can express identities, security labels and roles. UCON$^{\text{finite}}_{\text{preA}}$ is a member of UCON$_{ABC}$ family of models which covers attribute based pre-authorization usage control with finite attribute domains.

Safety is a fundamental problem for any access control model. Harrison et al. [1] introduced the *safety question* in protection systems, which asks whether or not a subject s can obtain right r for an object o. They showed this problem is undecidable in general. A safety analyzer can answer decidable safety questions. A recent result shows that safety of UCON$^{\text{finite}}_{\text{preA}}$ is decidable [7]. Since UCON$^{\text{finite}}_{\text{preA}}$

Z. Yan et al. (Eds.): NSS 2017, LNCS 10394, pp. 257–272, 2017.
DOI: 10.1007/978-3-319-64701-2_19

allows unbounded creation of subjects and objects, in general a $\text{UCON}_{\text{preA}}^{\text{finite}}$ system can grow without bound.

ABAC_α shares some characteristics with $\text{UCON}_{\text{preA}}^{\text{finite}}$. Both models restrict attributes to finite constant domains, and both allow unbounded creation of subjects and objects. Nonetheless there are significant differences between the two models, as discussed in Sects. 2 and 3. The central result of this paper is that the safety problem for ABAC_α can be reduced to that for $\text{UCON}_{\text{preA}}^{\text{finite}}$, and hence is decidable. Our reduction follows the notion of state-matching [9] and preserves security properties, including safety.

The rest of the paper is organized as follows. Section 2 reviews the ABAC_α model, and provides a slightly re-casted, but essentially identical, formal definition relative to its original definition [4]. Section 3 reviews the formal description of $\text{UCON}_{\text{preA}}^{\text{finite}}$ model. Section 4 presents a reduction from ABAC_α to $\text{UCON}_{\text{preA}}^{\text{finite}}$. Section 5 proves that the reduction of Sect. 4 is state-matching, from which decidability of ABAC_α follows. Section 6 concludes the paper.

2 The ABAC_α Formal Model (Review)

ABAC_α is an ABAC model that has "just sufficient" features to be "easily and naturally" configured to do DAC, MAC and RBAC [4]. The core components of this model are: users (U), subjects (S), objects (O), user attributes (UA), subject attributes (SA), object attributes (OA), permissions (P), authorization policy, creation and modification policy, and policy languages. The structure of ABAC_α model is shown in Fig. 1. Table 1 gives the formal definition of ABAC_α.

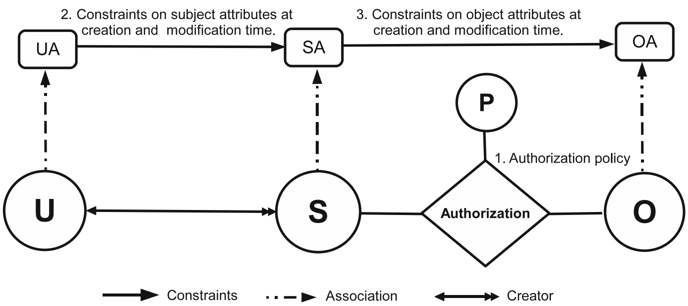

Fig. 1. ABAC_α model (adapted from [4])

2.1 Users, Subjects, Objects and Their Attributes

Users (U) represent human beings in an ABAC_α system who create and modify subjects, and access resources through subjects. **Subjects** (S) are processes created by users to perform some actions in the system. ABAC_α resources are represented as **Objects** (O). Users, subjects and objects are mutually disjoint

Table 1. ABAC$_\alpha$ formal model

Basic Sets and Functions

U, S, O are finite sets of existing users, subjects and objects

UA = $\{ua_1, ua_2, \ldots ua_l\}$, finite set of user attributes

SA = $\{sa_1, sa_2, \ldots sa_m\}$, finite set of subject attributes

OA = $\{oa_1, oa_2, \ldots oa_n\}$, finite set of object attributes

SubCreator: S $\rightarrow$ U. A system function, specifies the creator of a subject.

attType: UA $\cup$ SA $\cup$ OA $\rightarrow$ {set, atomic}

For each attribute att$\in$ UA $\cup$ SA $\cup$ OA:

SCOPE(att) denotes the finite set of atomic values for attribute att.

Range(att) represents a finite set of atomic or set values as the range of att.

$$\text{Range(att)} = \begin{cases} \text{SCOPE(att)} & \text{attType(att) = atomic.} \\ 2^{\text{SCOPE(att)}} & \text{attType(att) = set.} \end{cases}$$

ua_i: U $\rightarrow$ Range(ua_i), $ua_i \in$ UA

sa_j: S $\rightarrow$ Range(sa_j), $sa_j \in$ SA

oa_k: O $\rightarrow$ Range(oa_k), $oa_k \in$ OA

Tuple Notation

UAVT $\equiv \times_{i=1}^{l}$ Range(ua_i), set of all possible attribute value tuples for users

SAVT $\equiv \times_{j=1}^{m}$ Range(sa_j), set of all possible attribute value tuples for subjects

OAVT $\equiv \times_{k=1}^{n}$ Range(oa_k), set of all possible attribute value tuples for objects

uavtf: U $\rightarrow$ UAVT, current attribute value tuple for a user

savtf: S $\rightarrow$ SAVT, current attribute value tuple for a subject

oavtf: O $\rightarrow$ OAVT, current attribute value tuple for an object

Authorization Policy

P = $\{p_1, p_2, \ldots p_n\}$, a finite set of permissions.

For each p$\in$ P, Authorization$_p$(s:S,o:O) returns true or false.

Specified in language LAuthorization.

Creation and Modification Policy

Subject Creation Policy:

ConstrSub(u:U,s:NAME,$savt$:SAVT) returns true or false.

Specified in language LConstrSub.

Subject Modification Policy:

ConstrSubMod(u:U,s:S,$savt$:SAVT) returns true or false.

Specified in language LConstrSubMod.

Object Creation Policy:

ConstrObj(s:S,o:NAME,$oavt$:OAVT) returns true or false.

Specified in language LConstrObj.

Object Modification Policy:

ConstrObjMod(s:S,o:O,$oavt$:OAVT) returns true or false.

Specified in language LConstrObjMod.

Policy Languages

Each policy language is an instantiation of the Common Policy Language CPL that varies only in the values it can compare. Table 2 defines CPL for ABAC$_\alpha$.

Functional Specification

ABAC$_\alpha$ operations are formally specified in Table 3 in ABAC_α, and are collectively called entities. **NAME** is the set of all names for various entities in the system. **Attributes** are set-valued or atomic-valued functions which take an entity (user, subject or object) and return a value from a finite set of atomic values. Each user, subject, object is associated with a finite set of user attributes (UA), subject attributes (SA) and object attributes (OA) respectively. Each attribute is a set-valued or atomic-valued function. **attType** is a function that returns type of the attribute, i.e., whether it is set or atomic valued. **SCOPE** represents the domain of an attribute which is a finite set of atomic values. Potentially infinite domain attribute such as location, age are represented as large finite domains. For each attribute att, SCOPE(att) can be an unordered, a totally ordered or a partially ordered set. **Range**(att) is a finite set of all possible atomic or set values for attribute att. Each attribute takes a user or a subject or an object, and returns a value from its range. **SubCreator** is a system function which specifies the creator of a subject. SubCreator is assigned by the system at subject creation time, and cannot change. UAVT, SAVT, OAVT are sets of all possible **Attribute Value Tuples** for users, subjects and objects respectively. The functions uavtf, savtf and oavtf, return current attribute value tuples for a particular user, subject or object respectively.

2.2 Authorization Policy

ABAC_α authorization policy consists of a single authorization policy for each permission. **Permissions** are privileges that a user can hold on objects and exercise through subjects. It enables access of a subject on an object in a particular mode, such as read or write. $\text{P} = \{\text{p}_1, \text{p}_2, \ldots\ \text{p}_n\}$ is a finite set of permissions. Each **Authorization Policy** is a boolean function which is associated with a permission, and takes a subject and an object as input and returns true or false based on the boolean expression built from attributes of that subject and object.

2.3 Creation and Modification Policy

User creation, attribute value assignment of user at creation time, user deletion and modification of a user's attribute values is done by security administrator, and is outside the scope of ABAC_α. Subject creation and assigning attribute value to subject during creation time is constrained by the values of user attributes. Only creator is allowed to terminate and modify attributes of a subject. Modification of subject attributes is constrained by the creating user's attribute values, and existing and new attribute values of the concerned subject.[1] Objects are created by subjects. Object creation and attribute value assignment at creation time is constrained by creating subject's attribute values and proposed attribute value for the object. Modification of object attribute

[1] In the original definition of ABAC_α [4] subject creation and modification have identical policies. However, a correct configuration of MAC in ABAC_α requires different policies for these two operations. Hence, we define ABAC_α here to have separate policies for these two operations.

value is constrained by subject and object's existing attribute values and proposed attribute values for object. ABAC$_\alpha$ has subject deletion however there is no object deletion. An existing subject can be deleted only by its creator.

2.4 Policy Languages

Each policy is expressed using a specific language. CPL is the common policy language part for each language. Each language is a CPL instantiation with different values for *set* and *atomic*. CPL is defined in Table 2.

Table 2. Definition of CPL

CPL
φ ::= $\varphi \wedge \varphi \mid \varphi \vee \varphi \mid (\varphi) \mid \neg\varphi \mid \exists\, \mathrm{x} \in$ *set*.$\varphi \mid \forall\, \mathrm{x} \in$ *set*.$\varphi \mid$ *set* setcompare *set* $\mid$ *atomic* $\in$ *set* $\mid$ *atomic* atomiccompare *atomic*
setcompare ::= $\subset \mid \subseteq \mid \not\subseteq$
atomiccompare ::= $< \mid = \mid \leq$

Authorization Policy: The boolean expression of authorization policy is defined using the language LAuthorization which is a CPL instantiation where *set* and *atomic* refers to the set and atomic valued attribute of concerned subject and object.

Creation and Modification Policy: Subject creation, subject attribute modification, object creation and object attribute modification policies are all boolean expressions and defined using LConstrSub, LConstrSubMod, LConstrObj and LConstrObjMod respectively. LConstrSub is a CPL instantiation where *set* and *atomic* refers to the set and atomic valued attribute of creating user and proposed attribute values for subject being created. LConstrSubMod is a CPL instantiation where *set* and *atomic* refers to the set and atomic valued attribute value of concerned user and subject and proposed attribute value for subject. LConstrObj is a CPL instantiation where *set* and *atomic* refers to the set and atomic valued attribute value of creating subject and proposed attribute value for object being created. LConstrObjMod is a CPL instantiation where *set* and *atomic* refers to the set and atomic valued attribute value of concerned subject and object and proposed attribute values for the object.

2.5 Functional Specification

ABAC$_\alpha$ functional specification has six operations: access an object by a subject, creation of subject and object, deletion of subject, modification of subject and object attributes. Each ABAC$_\alpha$ operation has two parts: condition part and update part. Table 3 shows the specification of condition and update parts for ABAC$_\alpha$ operations.

Table 3. Functional specification of ABAC$_\alpha$ operations

Operations	Conditions	Updates
Access$_p$(s, o)	s∈ S ∧ o∈ O ∧ Authorization$_p(s, o)$	
CreateSubject (u, s: NAME, $savt$: SAVT)	u∈U ∧ s∉S ∧ ConstrSub$(u, s, savt)$	S′ = S∪{s} SubCreator(s) = u savtf(s) = $savt$
DeleteSubject (u, s: NAME)	s∈S ∧ u∈U ∧ SubCreator(s) = u	S′ = S\\{s}
ModifySubjectAtt (u, s: NAME, $savt$: SAVT)	u∈U ∧ s∈S ∧ SubCreator(s) = u ∧ConstrSubMod$(u, s, savt)$	savtf(s) = $savt$
CreateObject (s, o: NAME, $oavt$: OAVT)	s∈S ∧ o∉O ∧ ConstrObj(s, o, $oavt$)	O′ = O∪{o} oavtf(o) = $oavt$
ModifyObjectAtt (s, o: NAME, $oavt$: OAVT)	s∈S ∧ o∈O ∧ ConstrObjMod(s, o, $oavt$)	oavtf(o) = $oavt$

3 The UCON$^{\mathbf{finite}}_{\mathbf{preA}}$ Model (Review)

In usage control authorization model entities are subjects and objects, and subjects are a subset of objects. Each object has a unique identifier and a finite set of attributes. Attributes can be mutable or immutable. Usage control Pre-Authorization model (UCON$_{\mathrm{preA}}$) evaluates authorization decisions of permission prior to the execution of commands. Figure 2 shows the components of UCON$_{\mathrm{preA}}$ model.

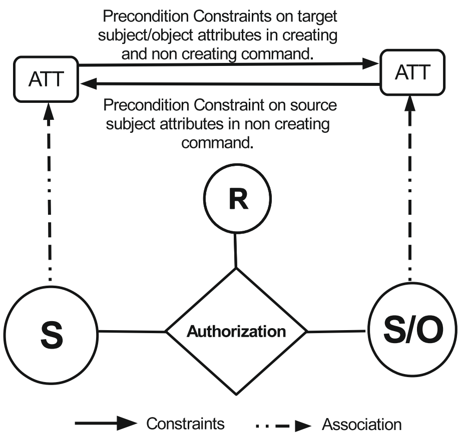

Fig. 2. UCON$_{\mathrm{preA}}$ model.

The $\mathrm{UCON}^{\mathrm{finite}}_{\mathrm{preA}}$ model, i.e., pre-authorization UCON with finite attributes, is defined through a usage control scheme [7], as follows.

1. Object schema OS_Δ, is of the form $\{\mathrm{a}_1\colon \sigma_1, \ldots, \mathrm{a}_n\colon \sigma_n\}$ where each a_i is the name of an attribute and σ_i is a finite set specifying a_i's domain. $\mathrm{UCON}^{\mathrm{finite}}_{\mathrm{preA}}$ considers single object schema for different objects and considers only atomic values for each domain σ_i.
2. $\mathrm{UR} = \{\mathrm{r}_1, \mathrm{r}_2, \ldots\ \mathrm{r}_k\}$, a set of usage rights, where r_i defines a permission enabled by a usage control command.
3. $\mathrm{UC} = \{\mathrm{UC}_1, \mathrm{UC}_2, \ldots\ \mathrm{UC}_l\}$, a set of usage control commands.
4. $\mathrm{ATT} =\{\mathrm{a}_1, \mathrm{a}_2, \ldots\ \mathrm{a}_n\}$, a finite set of object attributes.
5. $\mathrm{AVT} = \sigma_1 \times \ldots \times \sigma_n$, set of all possible attribute value tuples.
6. avtf: $\mathrm{O} \rightarrow \mathrm{AVT}$, returns existing attribute value tuple of an object.
7. Each command in UC is associated with a right and has two formal parameters s and o, where s is a subject trying to access object o with right r. A single right can be associated with more than one command. Number of commands (l) $\geq$ number of rights (k). There are two types of usage control commands, Non-Creating Command and Creating Command. Each command has a precondition part and an update part. Table 4 shows the structure of non-creating and creating command of $\mathrm{UCON}^{\mathrm{finite}}_{\mathrm{preA}}$.
   (a) In $\mathrm{UCON}^{\mathrm{finite}}_{\mathrm{preA}}$ non-creating command, $f_b(\mathrm{s}, \mathrm{o})$ is a boolean function which takes the attribute values of s and o and returns true or false. If the result is true then the PreUpdate is performed with zero or more attributes of s and o independently updated to new values computed from their attribute values prior to the command execution. Also the usage right r is granted. Otherwise the command terminates without granting r. f_1 and f_2 are the computing functions for new values.
   (b) In $\mathrm{UCON}^{\mathrm{finite}}_{\mathrm{preA}}$ creating command, $f_b(\mathrm{s})$ is a boolean function which takes the attribute values of s and returns true or false. If the result is true then

Table 4. $\mathrm{UCON}^{\mathrm{finite}}_{\mathrm{preA}}$ command structure

Non-Creating Command	Creating Command
Command_Name$_\mathrm{r}$(s,o) **PreCondition:**f_b(s,o)$\rightarrow$ {true,false}; **PreUpdate:** $\mathrm{s.a}_{i_1} := \mathrm{f}_{1,a_{i_1}}(\mathrm{s,o})$; $\vdots$ $\mathrm{s.a}_{i_p} := \mathrm{f}_{1,a_{i_p}}(\mathrm{s,o})$; $\mathrm{o.a}_{j_1} := \mathrm{f}_{2,a_{j_1}}(\mathrm{s,o})$; $\vdots$ $\mathrm{o.a}_{j_q} := \mathrm{f}_{2,a_{j_q}}(\mathrm{s,o})$;	**Command_Name**$_\mathrm{r}$(s,o) **PreCondition:**f_b(s)$\rightarrow$ {true,false}; **PreUpdate:** create o; $\mathrm{s.a}_{i_1} := \mathrm{f}_{1,a_{i_1}}(\mathrm{s})$; $\vdots$ $\mathrm{s.a}_{i_p} := \mathrm{f}_{1,a_{i_p}}(\mathrm{s})$; $\mathrm{o.a}_{j_1} := \mathrm{f}_{2,a_{j_1}}(\mathrm{s})$; $\vdots$ $\mathrm{o.a}_{j_q} := \mathrm{f}_{2,a_{j_q}}(\mathrm{s})$;

the PreUpdate is performed with zero or more attributes of s updated to new values computed from the attribute values of s. All attributes of the newly created object o are assigned computed attribute values. Also the usage right r is granted. Otherwise the command terminates without granting r. f_1 and f_2 are the computing functions for new values.

4 Reduction from $\mathbf{ABAC}_{\boldsymbol{\alpha}}$ to $\mathbf{UCON}_{\mathbf{preA}}^{\mathbf{finite}}$

In this section we define a reduction from ABAC_α to $\text{UCON}_{\text{preA}}^{\text{finite}}$. For convenience we introduce policy evaluation functions and sets of eligible attribute value tuples for creation and modification of subjects and objects of ABAC_α. We also introduce the PreCondition evaluation functions of $\text{UCON}_{\text{preA}}^{\text{finite}}$ which we will use in the next section. These additional notations enable us to relate the machinery of these two models.

4.1 Policy Evaluation Functions for $\mathbf{ABAC}_{\boldsymbol{\alpha}}$

Each Policy evaluation function evaluates corresponding policy and returns true or false.

Authorization Policy Evaluation Function: ChkAuth(p, $savtf(s)$, $oavtf(o)$) returns true or false. This function evaluates the authorization policy Authori-zation$_p(s, o)$ to determine whether a subject s is allowed to have permission p on object o.

Creation and Modification Policy Evaluation Functions:

– ChkConstrSub($uavtf(u)$,$savt$) returns true or false. It evaluates the subject creation policy ConstrSub$(u, s, savt)$ as to whether a user u with attribute value tuple $uavtf(u)$ is allowed to create a subject s with attribute value tuple $savt$.
– ChkConstrSubMod$(uavtf(u), savtf(s), savt)$ returns true or false. It evaluates the subject modification policy ConstrSubMod$(u, s, savt)$ as to whether a user u with attribute value tuple uavtf(u) is allowed to modify a subject s with attribute value tuple savtf(s) to $savt$.
– ChkConstrobj$(savtf(s), oavt)$ returns true or false. It evaluates the object creation policy ConstrObj$(s, o, oavt)$ as to whether a subject s with attribute value tuple $savt$ is allowed to create an object o with attribute value tuple $oavt$.
– ChkConstrobjMod$(savtf(s), oavtf(o), oavt)$ returns true or false. It evaluates the object modification policy ConstrObjMod$(s, o, oavt)$ as to whether a subject s with attribute value tuple savtf(s) is allowed to modify an object o with attribute value tuple oavtf(o) to $oavt$.

4.2 Sets of Eligible Attribute Value Tuples

Using the policy evaluation functions for ABAC$_\alpha$ we define 4 eligible sets for attribute value tuples as follows.

Definition 1. *set of user-subject-creatable-tuples*
$UAVTCrSAVT \subseteq UAVT \times SAVT$
$UAVTCrSAVT = \{\langle i, j\rangle \mid i \in UAVT \wedge j \in SAVT \wedge ChkConstrSub(i,j)\}$

Definition 2. *set of user-subject-modifiable-tuples*
$UAVTModSAVT \subseteq UAVT \times SAVT \times SAVT$
$UAVTModSAVT = \{\langle i, j, k\rangle \mid i \in UAVT \wedge j \in SAVT \wedge k \in SAVT \wedge ChkConstrSubMod(i, j, k)\}$

Definition 3. *set of subject-object-creatable-tuples*
$SAVTCrOAVT \subseteq SAVT \times OAVT$
$SAVTCrOAVT = \{\langle i, j\rangle \mid i \in SAVT \wedge j \in OAVT \wedge ChkConstrObj(i, j)\}$

Definition 4. *set of subject-object-modifiable-tuples*
$SAVTModOAVT \subseteq SAVT \times OAVT \times OAVT$
$SAVTModOAVT = \{\langle i, j, k\rangle \mid i \in SAVT \wedge j \in OAVT \wedge k \in OAVT \wedge ChkConstrObjMod(i,j,k)\}$

4.3 PreCondition Evaluation Functions for $\mathbf{UCON_{preA}^{finite}}$

PreCondition evaluation functions of UCON$_{\text{preA}}^{\text{finite}}$ check the PreConditions of UCON$_{\text{preA}}^{\text{finite}}$ commands and return true or false.

- **CheckPCNCR(**$uc_r, avtf(s), avtf(o), avt_1, avt_2$**)** returns true or false. It evaluates the PreCondition $f_b(s, o)$ and PreUpdate of non-creating command $uc_r(s, o)$ as to whether a subject s is allowed to execute command uc_r on object o and if allowed whether it modifies s's attribute value tuple from avtf(s) to avt_1 and o's attribute value tuple from avtf(o) to avt_2.
- **CheckPCCR(**$uc_r, avtf(s), o, avt_1, avt_2$**)** returns true or false. It evaluates the PreCondition $f_b(s)$ and PreUpdate of creating command $uc_r(s, o)$ as to whether a subject s is allowed to execute the command uc with right r and if allowed whether it creates object o with attribute value tuple to avt_2 and modifies s's own attribute value tuple from avtf(s) to avt_1.

4.4 Reduction from $\mathbf{ABAC_\alpha}$ to $\mathbf{UCON_{preA}^{finite}}$

The reduction is presented showing the configuration of UCON$_{\text{preA}}^{\text{finite}}$ object schema, rights and commands to do ABAC$_\alpha$. Table 5 shows the reduction.

Object Schema of UCON$_{\text{preA}}^{\text{finite}}$**:** Every ABAC$_\alpha$ entity (user, subject, object) is represented as a UCON$_{\text{preA}}^{\text{finite}}$ object and the attribute entity_type specifies

Table 5. Reduction from ABAC_α to $\mathrm{UCON}_{\mathrm{preA}}^{\mathrm{finite}}$

Object Schema($\mathbf{OS}_\Delta$):
[entity_type:{user, subject, object}, user_name: $\mathrm{U}^{\mathrm{ABAC}_\alpha}$, SubCreator: $\mathrm{U}^{\mathrm{ABAC}_\alpha}$, isDeleted: {true,false}, ua_1:Range(ua_1), ..., ua_m:Range(ua_m), sa_1:Range(sa_1), ..., sa_n:Range(sa_n), oa_1:Range(oa_1), ..., oa_p: Range(oa_p)]
Attributes:
ATT = {entity_type, user_name, SubCreator, isDeleted} $\cup\ \mathrm{UA}^{\mathrm{ABAC}_\alpha} \cup \mathrm{SA}^{\mathrm{ABAC}_\alpha} \cup \mathrm{OA}^{\mathrm{ABAC}_\alpha}$
Usage Rights:
UR= $\mathrm{P}^{\mathrm{ABAC}_\alpha} \cup \{\mathrm{d}\}$
Commands:
$\mathrm{UCON}_{\mathrm{preA}}^{\mathrm{finite}}$ commands are defined in Tables 6 and 7

whether a particular $\mathrm{UCON}_{\mathrm{preA}}^{\mathrm{finite}}$ object is ABAC_α user, subject or object. User, subject and object attributes of ABAC_α are represented as $\mathrm{UCON}_{\mathrm{preA}}^{\mathrm{finite}}$ object attributes. There is no user creation in ABAC_α so $\mathrm{U}^{\mathrm{ABAC}_\alpha}$ is a finite set. ABAC_α function SubCreator is configured here with a mandatory $\mathrm{UCON}_{\mathrm{preA}}^{\mathrm{finite}}$ object attribute whose domain would be finite set of users ($\mathrm{U}^{\mathrm{ABAC}_\alpha}$). To determine which user is the creator of an ABAC_α subject, $\mathrm{UCON}_{\mathrm{preA}}^{\mathrm{finite}}$ object needs to have another mandatory attribute user_name whose range is also finite set of users ($\mathrm{U}^{\mathrm{ABAC}_\alpha}$). ABAC_α has a subject deletion operation. In [7] it is shown that deletion of a subject can be simulated by using a special boolean attribute isDeleted which has a boolean domain. We consider "NULL" as a special attribute value for any atomic or set valued attribute. It is assigned to an attribute which is not appropriate for a particular entity. We need to add "NULL" in the range of UA, SA and OA for this reduction. As there is no user deletion and object deletion in ABAC_α, isDeleted would be "NULL" for both users and objects. $\mathrm{UCON}_{\mathrm{preA}}^{\mathrm{finite}}$ attribute set ATT = {entity_type, user_name, SubCreator, isDeleted} $\cup\ \mathrm{UA}^{\mathrm{ABAC}_\alpha} \cup \mathrm{SA}^{\mathrm{ABAC}_\alpha} \cup \mathrm{OA}^{\mathrm{ABAC}_\alpha}$.

$\mathrm{UCON}_{\mathrm{preA}}^{\mathrm{finite}}$ **usage rights UR:** In this reduction each ABAC_α permission is considered as a usage right in $\mathrm{UCON}_{\mathrm{preA}}^{\mathrm{finite}}$ and additionally a dummy right d is introduced. Each $\mathrm{UCON}_{\mathrm{preA}}^{\mathrm{finite}}$ command associates with a right. We use dummy right d for association with the commands which are defined to configure ABAC_α operations. Usage Right $\mathrm{UR}^{\mathrm{UCON}_{\mathrm{preA}}^{\mathrm{finite}}} = \mathrm{P}^{\mathrm{ABAC}_\alpha} \cup \{d\}$.

$\mathrm{UCON}_{\mathrm{preA}}^{\mathrm{finite}}$ **commands:** ABAC_α operations are reduced to specific $\mathrm{UCON}_{\mathrm{preA}}^{\mathrm{finite}}$ commands. We use the sets of eligible attribute value tuples to define $\mathrm{UCON}_{\mathrm{preA}}^{\mathrm{finite}}$ commands. It defines a creating command for each element of UAVTCrSAVT and SAVTCrOAVT and a non-creating command for each element of UAVT-ModSAVT and SAVTModOAVT. For example consider an ABAC_α subject creation policy where a user u with attribute value tuple *uavt* is allowed to create a subject s with attribute value tuple $savt$, so by definition $\langle uavt, savt\rangle \in$ UAVTCrSAVT. For each element $\langle i, j\rangle \in$ UAVTCrSAVT this reduction has a command named CreateSubject_ij(s, o) which creates an object o with

Table 6. $\text{UCON}_{\text{preA}}^{\text{finite}}$ non-creating commands

for each r $\in \text{UR}^{\text{UCON}_{\text{preA}}^{\text{finite}}} \setminus \{\text{d}\}$ **Access**$_r(s, o)$ PreCondition: ChkAuth(r,avtf(s),avtf(o)) PreUPdate: N/A	**DeleteSubject**$_d(s, o)$ PreCondition: s.entity_type =user $\wedge$ o.entity_type = subject $\wedge$ o.SubCreator = s.user_name $\wedge$ o.isDeleted = false PreUpdate: o.isDeleted = true
For each $\langle i, j, k\rangle \in$ UAVTModSAVT **ModifySubjectAtt_ijk**$_d$**(s,o)** PreCondition: s.entity_type = user $\wedge$ o.entity_type = subject $\wedge$ o.isDeleted = false $\wedge$ o.SubCreator = s.user_name $\wedge \langle s.ua_1, \ldots, s.ua_m\rangle = \langle i_1, \ldots, i_m\rangle$ $\wedge \langle o.sa_1, \ldots, s.sa_n\rangle = \langle j_1, \ldots, j_n\rangle$ PreUpdate: o.sa$_1$ = k$_1$ $\vdots$ o.sa$_n$ = k$_n$	For each $\langle i, j, k\rangle \in$ SAVTModOAVT **ModifyObjectAtt_ijk**$_d$**(s,o)** PreCondition: s.entity_type = subject $\wedge$ o.entity_type = object $\wedge$ s.isDeleted = false $\wedge \langle s.sa_1, \ldots, s.sa_n\rangle = \langle i_1, \ldots, i_n\rangle$ $\wedge \langle o.oa_1, \ldots, s.oa_p\rangle = \langle j_1, \ldots, j_p\rangle$ PreUpdate: o.oa$_1$ = k$_1$ $\vdots$ o.oa$_p$ = k$_p$

Table 7. $\text{UCON}_{\text{preA}}^{\text{finite}}$ creating commands

For each $\langle i, j\rangle \in$ UAVTCrSAVT **CreateSubject_ij**$_d(s, o)$ PreCondition: s.entity_type = user $\wedge \langle s.ua_1, \ldots, s.ua_m\rangle = \langle i_1, \ldots, i_m\rangle$ PreUpdate: create o o.entity_type = subject o.user_name = NULL o.SubCreator = s.user_name o.isDeleted = false o.ua$_1$ = NULL $\vdots$ o.ua$_m$ = NULL o.sa$_1$ = j$_1$ $\vdots$ o.sa$_n$ = j$_n$ o.oa$_1$ = NULL $\vdots$ o.oa$_p$ = NULL	For each $\langle i, j\rangle \in$ SAVTCrOAVT **CreateObject_ij**$_d$**(s,o)** PreCondition: s.entity_type = subject $\wedge$ s.isDeleted = false $\wedge \langle s.sa_1, \ldots, s.sa_n\rangle = \langle i_1, \ldots, i_n\rangle$ PreUpdate: create o o.entity_type = object o.user_name = NULL o.SubCreator = NULL o.isDeleted = NULL o.ua$_1$ = NULL $\vdots$ o.ua$_m$ = NULL o.sa$_1$ = NULL $\vdots$ o.sa$_n$ = NULL o.oa$_1$ = j$_1$ $\vdots$ o.oa$_p$ = j$_p$

entity_type = subject. Each $\text{Access}_r^{\text{UCON}_{\text{preA}}^{\text{finite}}}(s,o)$ configures $\text{Access}_p^{\text{ABAC}_\alpha}(s,o)$ where r = p. Here $\text{Access}_r^{\text{UCON}_{\text{preA}}^{\text{finite}}}$ is a non-creating command with PreCondition part only and PreCondition checks the authorization evaluation function of ABAC_α. Each $\text{DeleteSubject}_d^{\text{UCON}_{\text{preA}}^{\text{finite}}}(s,o)$ configures $\text{DeleteSubject}^{\text{ABAC}_\alpha}(u,s)$ which is also a non-creating command and sets o.isDeleted = true. Tables 6 and 7 show the configuration of non-creating and creating commands for this construction.

5 Safety of ABAC_α

In this section we show that safety of ABAC_α is decidable. We prove that the reduction provided in the previous section is state matching, so it preserves security properties including safety. Decidable safety for ABAC_α then follows from decidable safety for $\text{UCON}_{\text{preA}}^{\text{finite}}$. Tripunitara and Li [9] define an access control model as a set of access control schemes. An access control scheme is a state transition system $\langle \Gamma, \Psi, Q, \vdash \rangle$, where Γ is a set of states, Ψ is a set of state transition rules, Q is a set of queries and $\vdash$: $\Gamma \times$ Q $\rightarrow \{true, false\}$ is the entailment relation. The notion of state-matching reduction is defined as follows.

Definition 5. *State Matching Reduction:*
Given two schemes A and B and a mapping A to B, $\sigma : (\Gamma^A \times \Psi^A) \cup Q^A \rightarrow (\Gamma^B \times \Psi^B) \cup Q^B$, we say that the two states γ^A and γ^B are equivalent under the mapping σ when for every $q^A \in Q^A, \gamma^A \vdash^A q^A$ if and only if $\gamma^B \vdash^B \sigma(q^A)$. A mapping σ from A to B is said to be a state-matching reduction if for every $\gamma^A \in \Gamma^A$ and every $\psi^A \in \Psi^A, \langle \gamma^B, \psi^B \rangle = \sigma(\langle \gamma^A, \psi^A \rangle)$ has the following two properties:

1. *For every γ_1^A in scheme A such that $\gamma^A \xrightarrow{*}_\psi \gamma_1^A$, there exists a state γ_1^B such that $\gamma^B \xrightarrow{*}_\psi \gamma_1^B$ and γ_1^A and γ_1^B are equivalent under σ.*
2. *For every γ_1^B in scheme B such that $\gamma^B \xrightarrow{*}_\psi \gamma_1^B$, there exists a state γ_1^A such that $\gamma^A \xrightarrow{*}_\psi \gamma_1^A$ and γ_1^B and γ_1^A are equivalent under σ.*

In order to show that a reduction from ABAC_α and $\text{UCON}_{\text{preA}}^{\text{finite}}$ is state matching, we have to show the following:

1. Represent ABAC_α and $\text{UCON}_{\text{preA}}^{\text{finite}}$ models as ABAC_α and $\text{UCON}_{\text{preA}}^{\text{finite}}$ schemes
2. Construct a mapping $\sigma^{\text{ABAC}_\alpha}$ that maps ABAC_α to $\text{UCON}_{\text{preA}}^{\text{finite}}$
3. Prove that $\sigma^{\text{ABAC}_\alpha}$ mapping from ABAC_α to $\text{UCON}_{\text{preA}}^{\text{finite}}$ satisfies the following two requirements for state matching reduction:
   (a) for every state $\gamma_1^{\text{ABAC}_\alpha}$ reachable from $\gamma^{\text{ABAC}_\alpha}$ under the mapping $\sigma^{\text{ABAC}_\alpha}$ there exists a reachable state in $\text{UCON}_{\text{preA}}^{\text{finite}}$ scheme that is equivalent (answers all the queries in the same way)
   (b) for every state $\gamma_1^{\text{UCON}_{\text{preA}}^{\text{finite}}}$ reachable from $\gamma^{\text{UCON}_{\text{preA}}^{\text{finite}}}$ under the mapping $\sigma^{\text{ABAC}_\alpha}$ there exists a reachable state in ABAC_α scheme that is equivalent (answers all the queries in the same way)

5.1 $\mathbf{ABAC}_\alpha$ Scheme

An ABAC_α scheme consists of $\langle \varGamma^{\mathrm{ABAC}_\alpha}, \varPsi^{\mathrm{ABAC}_\alpha}, Q^{\mathrm{ABAC}_\alpha}, \vdash^{\mathrm{ABAC}_\alpha}\rangle$. Where

- $\varGamma^{\mathrm{ABAC}_\alpha}$ is the set of all states. Where each state $\gamma^{\mathrm{ABAC}_\alpha} \in \varGamma^{\mathrm{ABAC}_\alpha}$ is characterized by $\langle \mathrm{U}_\gamma$, S_γ, O_γ, UA, SA, OA, uavtf, savtf, oavtf, P, SubCreator$\rangle$ where U_γ, S_γ, O_γ are set of users, subjects objects respectively in state γ.
- $\varPsi^{\mathrm{ABAC}_\alpha}$ is the set of state transition rules which are all ABAC_α operations defined in Table 3.
- $\mathrm{Q}^{\mathrm{ABAC}_\alpha}$ is the set of queries of type:
  1. $\mathrm{Authorization}_p(s, o)$ for $\mathrm{p} \in \mathrm{P}^{\mathrm{ABAC}_\alpha}$, $\mathrm{s} \in \mathrm{S}^{\mathrm{ABAC}_\alpha}$, $\mathrm{o} \in \mathrm{O}^{\mathrm{ABAC}_\alpha}$.
  2. $\mathrm{ConstrSub}(u, s, savt)$ for $u \in \mathrm{U}^{\mathrm{ABAC}_\alpha}$, $s \notin \mathrm{S}^{\mathrm{ABAC}_\alpha}$, $savt \in \mathrm{SAVT}^{\mathrm{ABAC}_\alpha}$.
  3. $\mathrm{ConstrSubMod}(u, s, savt)$ for $u \in \mathrm{U}^{\mathrm{ABAC}_\alpha}$, $s \in \mathrm{S}^{\mathrm{ABAC}_\alpha}$, $savt \in \mathrm{SAVT}^{\mathrm{ABAC}_\alpha}$.
  4. $\mathrm{ConstrObj}(s, o, oavt)$ for $s \in \mathrm{S}^{\mathrm{ABAC}_\alpha}$, $o \notin \mathrm{O}^{\mathrm{ABAC}_\alpha}$, $oavt \in \mathrm{OAVT}^{\mathrm{ABAC}_\alpha}$.
  5. $\mathrm{ConstrObjMod}(s, o, oavt)$ for $s \in \mathrm{S}^{\mathrm{ABAC}_\alpha}$, $o \in \mathrm{O}^{\mathrm{ABAC}_\alpha}$, $oavt \in \mathrm{OAVT}^{\mathrm{ABAC}_\alpha}$.
- Entailment $\vdash$ specifies that given a state $\gamma \in \varGamma^{\mathrm{ABAC}_\alpha}$ and a query $\mathrm{q} \in \mathrm{Q}^{\mathrm{ABAC}_\alpha}$, $\gamma \vdash \mathrm{q}$ if and only if q returns true in state γ.

5.2 $\mathbf{UCON}^{\mathbf{finite}}_{\mathbf{preA}}$ Scheme

An $\mathrm{UCON}^{\mathrm{finite}}_{\mathrm{preA}}$ scheme consists of $\langle \varGamma^{\mathrm{UCON}^{\mathrm{finite}}_{\mathrm{preA}}}, \varPsi^{\mathrm{UCON}^{\mathrm{finite}}_{\mathrm{preA}}}, Q^{\mathrm{UCON}^{\mathrm{finite}}_{\mathrm{preA}}}, \vdash^{\mathrm{UCON}^{\mathrm{finite}}_{\mathrm{preA}}}\rangle$, as follows.

- $\varGamma^{\mathrm{UCON}^{\mathrm{finite}}_{\mathrm{preA}}}$ is the set of all states. Where each state $\gamma^{\mathrm{UCON}^{\mathrm{finite}}_{\mathrm{preA}}} \in \varGamma^{\mathrm{UCON}^{\mathrm{finite}}_{\mathrm{preA}}}$ is characterized by $\langle OS^\gamma_\Delta, UR, ATT, AVT, avtf\rangle$. Here $\mathrm{OS}^\gamma_\Delta$ is the object schema in state γ.
- $\varPsi^{\mathrm{UCON}^{\mathrm{finite}}_{\mathrm{preA}}}$ is set of state transition rules which are the set of creating and non-creating commands of $\mathrm{UCON}^{\mathrm{finite}}_{\mathrm{preA}}$ defined in Tables 6 and 7.
- $\mathrm{Q}^{\mathrm{ABAC}_\alpha}$ is the set of queries and of following types:
  1. $\mathrm{CheckPCNCR}(uc_r, avtf(s), avtf(o), avt_1, avt_2)$ for $uc_r \in \mathrm{UC}$, $r \in \mathrm{UR}$, s and o are $\mathrm{UCON}^{\mathrm{finite}}_{\mathrm{preA}}$ objects.
  2. $\mathrm{CheckPCCR}(uc_r, avtf(s), avt_1, avt_2)$ for $uc_r \in \mathrm{UC}$, $r \in \mathrm{UR}$, s is an $\mathrm{UCON}^{\mathrm{finite}}_{\mathrm{preA}}$ object.
- Entailment $\vdash$ specifies that given a state $\gamma \in \varGamma^{\mathrm{UCON}^{\mathrm{finite}}_{\mathrm{preA}}}$ and a query $\mathrm{q} \in \mathrm{Q}^{\mathrm{UCON}^{\mathrm{finite}}_{\mathrm{preA}}}$, $\gamma \vdash \mathrm{q}$ if and only if q returns true in state γ.

5.3 Mapping from $\mathbf{ABAC}_\alpha$ to $\mathbf{UCON}^{\mathbf{finite}}_{\mathbf{preA}}$ ($\sigma^{\mathbf{ABAC}_\alpha}$)

- Mapping of $\varGamma^{\mathrm{ABAC}_\alpha}$ to $\varGamma^{\mathrm{UCON}^{\mathrm{finite}}_{\mathrm{preA}}}$
  - Mapping of Object Schema(OS_Δ), ATT and UR is provided in Table 5
- Mapping of $\varPsi^{\mathrm{ABAC}_\alpha}$ to $\varPsi^{\mathrm{UCON}^{\mathrm{finite}}_{\mathrm{preA}}}$
  - $\sigma(\mathrm{Access}_p) = \mathrm{Access}_r^{\mathrm{UCON}^{\mathrm{finite}}_{\mathrm{preA}}}$ where $\mathrm{r} = \mathrm{p}$.

- $\sigma(\text{CreateSubject}(u, s, savt))$ = CreateSubject_ij$_d$(s, o), i = uavtf(u) and j = $savt$.
- $\sigma(\text{DeleteSubject}(u, s))$ = DeleteSubject$_d$(s, o).
- $\sigma(\text{ModifySubjectAtt}(u, s, savt))$ = ModifySubjectAtt_ijk$_d$(s, o), i = uavtf(u) and j = savtf(s) and k = $savt$.
- $\sigma(\text{CreateObject}(s, o, oavt))$ = CreateObject_ij$_d$(s, o), i = savtf(s) and j = $oavt$.
- $\sigma(\text{ModifyObjectAtt}(s, o, oavt))$ = ModifyObjectAtt_ijk$_d$(s, o), i = savtf(s) and j = oavtf(o) and k = $oavt$.

– Mapping of Q^{ABAC_α} to $Q^{\text{UCON}^{\text{finite}}_{\text{preA}}}$ is provided below
- $\sigma(\text{Authorization}_p(s, o))$ = CheckPCNCR(Access$_p$, avtf(s), avtf(o), avtf(s), avtf(o)).
- $\sigma(\text{ConstrSub}(u, s, savt))$ = CheckPCCR(CreateSubject_ij$_d$, avtf(s), o, avtf(s), ⟨subject, NULL, u, false, NULL,..., NULL, savt$_1$, ... savt$_n$, NULL,..., NULL⟩) where i = uavtf(u) and j = savt.
- $\sigma(\text{ConstrSubMod}(u, s, savt))$= CheckPCNCR(ModifySubjectAtt_ijk$_d$, avtf(s), avtf(o), avtf(s), ⟨savt$_1$, ... savt$_n$⟩) where i = uavtf(u), j = savtf(s) and k = $savt$.
- $\sigma(\text{ConstrObj}(s, o, oavt))$ = CheckPCCR(CreateObject_ij$_d$, avtf(s), o, avtf(s), ⟨object, NULL, NULL, NULL, NULL,..., NULL, NULL,..., NULL, oavt$_1$, ... oavt$_p$⟩) where i = savtf(s) and j = oavt.
- $\sigma(\text{ConstrObjMod}(s, o, oavt))$=CheckPCNCR(ModifyObjectAtt_ijk$_d$, avtf(s), avtf(o), avtf(s), ⟨oavt$_1$, ... oavt$_p$⟩) where i = savtf(s), j = oavtf(o) and k = $oavt$.

5.4 Proof that $\sigma^{\text{ABAC}_\alpha}$ Is State-Matching

The proof that the mapping provided above is a state matching reduction is lengthy and tedious. Here we present an outline of the main argument.

Lemma 1. $\sigma^{\text{ABAC}_\alpha}$ *satisfies assertion 1 of the state matching reduction of Definition 5.*

Proof. (Sketch): Assertion 1 requires that, for every $\gamma^{\text{ABAC}_\alpha} \in \Gamma^{\text{ABAC}_\alpha}$ and every $\psi^{\text{ABAC}_\alpha} \in \Psi^{\text{ABAC}_\alpha}$, $\langle \gamma^{\text{ABAC}_\alpha}, \psi^{\text{ABAC}_\alpha} \rangle = \sigma\ (\langle \gamma^{\text{ABAC}_\alpha}, \psi^{\text{ABAC}_\alpha} \rangle)$ has the following property:

For every $\gamma_1^{\text{ABAC}_\alpha}$ in scheme ABAC$_\alpha$ such that $\gamma^{\text{ABAC}_\alpha} \xrightarrow{*}_{\psi^{\text{ABAC}_\alpha}} \gamma_1^{\text{ABAC}_\alpha}$, there exists a state $\gamma_1^{\text{UCON}^{\text{finite}}_{\text{preA}}}$ such that

1. $\gamma^{\text{UCON}^{\text{finite}}_{\text{preA}}}(=\sigma(\gamma^{\text{ABAC}_\alpha})) \xrightarrow{*}_{\psi^{\text{UCON}^{\text{finite}}_{\text{preA}}}(=\sigma(\psi^{\text{ABAC}_\alpha}))} \gamma_1^{\text{UCON}^{\text{finite}}_{\text{preA}}}$.
2. for every query $q^{\text{ABAC}_\alpha} \in Q^{\text{ABAC}_\alpha}$, $\gamma_1^{\text{ABAC}_\alpha} \vdash^{\text{ABAC}_\alpha} q^{\text{ABAC}_\alpha}$ if and only if $\gamma_1^{\text{UCON}^{\text{finite}}_{\text{preA}}} \vdash^{\text{UCON}^{\text{finite}}_{\text{preA}}} \sigma(q^{\text{ABAC}_\alpha})$. It can be decomposed into two directions:
   (a) The "if" direction:
   $\gamma_1^{\text{UCON}^{\text{finite}}_{\text{preA}}} \vdash^{\text{UCON}^{\text{finite}}_{\text{preA}}} \sigma(q^{\text{ABAC}_\alpha}) => \gamma_1^{\text{ABAC}_\alpha} \vdash^{\text{ABAC}_\alpha} q^{\text{ABAC}_\alpha}$.

(b) The "only if" direction:
$\gamma_1^{\mathrm{ABAC}_\alpha} \vdash^{\mathrm{ABAC}_\alpha} q^{\mathrm{ABAC}_\alpha} => \gamma_1^{\mathrm{UCON}_{\mathrm{preA}}^{\mathrm{finite}}} \vdash^{\mathrm{UCON}_{\mathrm{preA}}^{\mathrm{finite}}} \sigma(q^{\mathrm{ABAC}_\alpha})$.

The proof is by induction on number of steps n in $\gamma^{\mathrm{ABAC}_\alpha} \xrightarrow{*}_{\psi^{\mathrm{ABAC}_\alpha}} \gamma_1^{\mathrm{ABAC}_\alpha}$.

Lemma 2. $\sigma^{\mathrm{ABAC}_\alpha}$ *satisfies assertion 2 of the state matching reduction of Definition 5.*

Proof. (Sketch): Assertion 2 requires that, for every $\gamma^{\mathrm{ABAC}_\alpha} \in \Gamma^{\mathrm{ABAC}_\alpha}$ and every $\psi^{\mathrm{ABAC}_\alpha} \in \Psi^{\mathrm{ABAC}_\alpha}$, $\langle \gamma^{\mathrm{ABAC}_\alpha}, \psi^{\mathrm{ABAC}_\alpha} \rangle = \sigma\ (\langle \gamma^{\mathrm{ABAC}_\alpha}, \psi^{\mathrm{ABAC}_\alpha} \rangle)$ has the following property:

For every $\gamma_1^{\mathrm{UCON}_{\mathrm{preA}}^{\mathrm{finite}}}$ in scheme $\mathrm{UCON}_{\mathrm{preA}}^{\mathrm{finite}}$ such that $\gamma^{\mathrm{UCON}_{\mathrm{preA}}^{\mathrm{finite}}}$ $(=\sigma(\gamma^{\mathrm{ABAC}_\alpha})) \xrightarrow{*}_{\psi^{\mathrm{UCON}_{\mathrm{preA}}^{\mathrm{finite}}}(=\sigma(\psi^{\mathrm{ABAC}_\alpha}))} \gamma_1^{\mathrm{UCON}_{\mathrm{preA}}^{\mathrm{finite}}}$, there exists a state $\gamma_1^{\mathrm{ABAC}_\alpha}$ such that

1. $\gamma^{\mathrm{ABAC}_\alpha} \xrightarrow{*}_{\psi^{\mathrm{ABAC}_\alpha}} \gamma_1^{\mathrm{ABAC}_\alpha}$.
2. for every query $q^{\mathrm{ABAC}_\alpha} \in Q^{\mathrm{ABAC}_\alpha}$, $\gamma_1^{\mathrm{ABAC}_\alpha} \vdash^{\mathrm{ABAC}_\alpha} q^{\mathrm{ABAC}_\alpha}$ if and only if $\gamma_1^{\mathrm{UCON}_{\mathrm{preA}}^{\mathrm{finite}}} \vdash^{\mathrm{UCON}_{\mathrm{preA}}^{\mathrm{finite}}} \sigma(q^{\mathrm{ABAC}_\alpha})$.
   It can be decomposed into two directions:
   (a) The "if" direction:
   $\gamma_1^{\mathrm{UCON}_{\mathrm{preA}}^{\mathrm{finite}}} \vdash^{\mathrm{ABAC}_\alpha} \sigma(q^{\mathrm{ABAC}_\alpha}) => \gamma_1^{\mathrm{ABAC}_\alpha} \vdash^{\mathrm{ABAC}_\alpha} q^{\mathrm{ABAC}_\alpha}$.
   (b) The "only if" direction:
   $\gamma_1^{\mathrm{ABAC}_\alpha} \vdash^{\mathrm{ABAC}_\alpha} q^{\mathrm{ABAC}_\alpha} => \gamma_1^{\mathrm{UCON}_{\mathrm{preA}}^{\mathrm{finite}}} \vdash^{\mathrm{UCON}_{\mathrm{preA}}^{\mathrm{finite}}} \sigma(q^{\mathrm{ABAC}_\alpha})$.

The proof is by induction on number of steps n in $\gamma^{\mathrm{UCON}_{\mathrm{preA}}^{\mathrm{finite}}}$ $(= \sigma(\gamma^{\mathrm{ABAC}_\alpha}))$ $\xrightarrow{*}_{\psi^{\mathrm{UCON}_{\mathrm{preA}}^{\mathrm{finite}}}(=\sigma(\psi^{\mathrm{ABAC}_\alpha}))} \gamma_1^{\mathrm{UCON}_{\mathrm{preA}}^{\mathrm{finite}}}$.

Theorem 1. $\sigma^{\mathrm{ABAC}_\alpha}$ *is a state matching reduction.*

Proof. Lemma 1 shows that $\sigma^{\mathrm{ABAC}_\alpha}$ satisfies assertion 1 of Definition 5 and Lemma 2 shows that $\sigma^{\mathrm{ABAC}_\alpha}$ satisfies assertion 2 of Definition 5. According to the Definition 5, $\sigma^{\mathrm{ABAC}_\alpha}$ is a state matching reduction.

Theorem 2. *Safety of* ABAC_α *is decidable.*

Proof. Safety of $\mathrm{UCON}_{\mathrm{preA}}^{\mathrm{finite}}$ is decidable [7]. Theorem 1 proved there exists a state matching reduction from ABAC_α to $\mathrm{UCON}_{\mathrm{preA}}^{\mathrm{finite}}$. A state matching reduction preserves security properties [9] including safety.

6 Conclusion

This paper gives a state matching reduction from ABAC_α to $\mathrm{UCON}_{\mathrm{preA}}^{\mathrm{finite}}$. Safety of $\mathrm{UCON}_{\mathrm{preA}}^{\mathrm{finite}}$ is decidable [7] and state matching reduction preserves security properties including safety [9]. It follows that safety of ABAC_α is decidable.

Acknowledgments. This research is partially supported by NSF Grants CNS-1111925, CNS-1423481, CNS-1538418, and DoD ARL Grant W911NF-15-1-0518.

References

1. Harrison, M.A., Ruzzo, W.L., Ullman, J.D.: Protection in operating systems. Commun. ACM **19**(8), 461–471 (1976). http://doi.acm.org/10.1145/360303.360333
2. Hu, V.C., Ferrariolo, D., Kuhn, R., Schnitzer, A., Sandlin, K., Miller, R., Karen, S.: Guide to attribute based access control (ABAC) definitions and considerations. 2014 NIST Special Publication 800–162
3. Jin, X.: Attribute-Based Access Control Models and Implementation in Cloud Infrastructure as a Service. Ph.D. thesis, UTSA (2014)
4. Jin, X., Krishnan, R., Sandhu, R.: A unified attribute-based access control model covering DAC, MAC and RBAC. In: Cuppens-Boulahia, N., Cuppens, F., Garcia-Alfaro, J. (eds.) DBSec 2012. LNCS, vol. 7371, pp. 41–55. Springer, Heidelberg (2012). doi:10.1007/978-3-642-31540-4_4
5. Kolter, J., Schillinger, R., Pernul, G.: A privacy-enhanced attribute-based access control system. In: Barker, S., Ahn, G.-J. (eds.) DBSec 2007. LNCS, vol. 4602, pp. 129–143. Springer, Heidelberg (2007). doi:10.1007/978-3-540-73538-0_11
6. Park, J., Sandhu, R.: The UCONabc usage control model. ACM TISSEC **7**, 128–174 (2004)
7. Rajkumar, P., Sandhu, R.: Safety decidability for pre-authorization usage control with finite attribute domains. IEEE Trans. Dependable Secure Comput. **13**(5), 582–590 (2016)
8. Shen, H.: A semantic-aware attribute-based access control model for web services. In: Hua, A., Chang, S.-L. (eds.) ICA3PP 2009. LNCS, vol. 5574, pp. 693–703. Springer, Heidelberg (2009). doi:10.1007/978-3-642-03095-6_65
9. Tripunitara, M.V., Li, N.: A theory for comparing the expressive power of access control models. J. Comput. Secur. **15**(2), 231–272 (2007)
10. Yuan, E., Tong, J.: Attributed based access control (ABAC) for web services. In: Proceedings of the IEEE International Conference on Web Services, ICWS 2005, pp. 561–569 (2005). http://dx.doi.org/10.1109/ICWS.2005.25