This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2016.2646639, IEEE Transactions on Information Forensics and Security

A Cross Tenant Access Control (CTAC) Model for Cloud Computing: Formal Specification and Verification

Quratulain Alam, Saif U. R. Malik, *Member, IEEE*; Adnan Akhunzada, Kim-Kwang Raymond Choo, *Senior Member, IEEE*; Saher Tabbasum, and Masoom Alam

***Abstract*—Sharing of resources on the cloud can be achieved on a large scale since it is cost effective and location independent. Despite the hype surrounding cloud computing, organizations are still reluctant to deploy their businesses in the cloud computing environment due to concerns in secure resource sharing. In this paper, we propose a cloud resource mediation service offered by cloud service providers, which plays the role of trusted third party among its different tenants. This paper formally specifies the resource sharing mechanism between two different tenants in the presence of our proposed cloud resource mediation service. The correctness of permission activation and delegation mechanism among different tenants using four distinct algorithms (Activation, Delegation,Forward Revocation and Backward Revocation) is also demonstrated using formal verification. The performance analysis suggest that sharing of resources can be performed securely and efficiently across different tenants of the cloud.**

***Index Terms*—Cross Tenant Access Control, Formal Specification and Verification, Cloud Computing**

I. Introduction

WHILE there are a number of benefits afforded by the use of cloud computing to facilitate collaboration between users and organizations, security and privacy of cloud services and the user data may deter some users and organizations from using cloud services (on a larger scale) and remain topics of interest to researchers [11], [8], [12], [14]. Typically, a cloud service provider (CSP) provides a web interface where a cloud user can manage resources and settings (e.g. allowing a particular service and/or data to selected users). A CSP then implements these access control features on consumer data and other related resources.

However, traditional access control models, such as role based access control [15], are generally unable to adequately deal with cross-tenant resource access requests. In particular, cross-tenant access requests pose three key challenges. Firstly, each tenant must have some prior understanding and knowledge about the external users who will access the resources. Thus, an administrator of each tenant must have a list of users to whom the access will be allowed. This process is static in nature. In other words, tenants cannot leave and join cloud as they wish, which is a typical setting for a real-world deployment. Secondly, each tenant must be allowed to define cross-tenant access for other tenants as and when needed. Finally, as each tenant has its own administration, trust management issue among tenants can be challenging to address, particularly for hundreds or thousands of tenants.

To provide a secure cross-tenant resource access service, a fine-grained cross-tenant access control model is required [21]. Thus, in this paper, we propose a cloud resource mediation service (CRMS) to be offered by a CSP, since the CSP plays a pivotal role managing different tenants and a cloud user entrusts the data to the CSP. We posit that a CRMS can provide the CSP competitive advantage, since the CSP can provide users with secure access control services in a cross-tenant access environment (hereafter, we referred to as cross tenant access control - CTAC). From a privacy perspective, the CTAC model has two advantages. The privacy of a tenant, say T_2, is protected from another tenant, say T_1, and the CRMS, since T_2's attributes are not provided to T_1. T_2's attributes attributes are evaluated only by the CRMS. Furthermore, a user does not provide authentication credentials to the CRMS. Therefore, the privacy of T_2 is also protected as the CRMS has no knowledge of the permissions that T_2 is requesting from T_1. The security policies defined by T_1 use pseudonyms of the permissions without revealing the actual information to the CRMS during publication of the policies.

To demonstrate the correctness and security of the proposed approach, we use model checking to exhaustively explore the system and verify the finite state concurrent systems. Specifically, we use High Level Petri Nets (HLPN) and Z language for the modeling and analysis of the CTAC model. HLPN provides graphical and mathematical representations of the system, which facilitates the analysis of its reactions to a given input [17], [13]. Therefore, we are able to understand the links between different system entities and how information is processed. We then verify the model by translating the HLPN using bounded model checking. For this purpose, we use Satisfiability Modulo Theories Library (SMT-Lib) and Z3 solver [19], [9]. We remark that such formal verification has previously been used to evaluate security protocols such as in [3], [2], [7].

We regard the key contributions of this paper to be as follows:

Quratulain Alam is with Department of Computer Sciences, Institute of Management Sciences Peshawar, Pakistan e-mail: quratul.alam30@gmail.com

Saif U. R. Malik, Adnan Akhunzada, Saher Tabbasum, and Masoom Alam are with the Department of Computer Sciences, COMSATS Institute of Information Technology, Islamabad, Pakistan, email: saif.rehmanmalik, a.qureshi, saher.tabbasum, and masoom.alam@comsats.edu.pk.

Kim-Kwang Raymond Choo is with the Department of Information Systems and Cyber Security, The University of Texas at San Antonio, San Antonio, TX 78249 USA, email: raymond.choo@fulbrightmail.org.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2016.2646639, IEEE Transactions on Information Forensics and Security

- We present a CTAC model for collaboration, and the CRMS to facilitate resource sharing amongst various tenants and their users.
- We also present four different algorithms in the CTAC model, namely: activation, delegation, forward revocation and backward revocation.
- We then provide a detailed presentation of modeling, analysis and automated verification of the CTAC model using the Bounded Model Checking technique with SMT-LIB and Z3 solver, in order to demonstrate the correctness and security of the CTAC model.

The rest of the paper is organized as follows. Section II discusses background materials and related work. In Sections III and IV, we respectively present our proposed CRMS and CTAC model. We then formally specify and model the model using HLPN and Z formal language, as well as verifying its correctness and security in Sections V and VI. The last section concludes the paper.

II. Background and Related Work

A. SMT-Lib and Z3 Solver

Boolean Satisfiability Solvers (SAT) have been used in automated reasoning and formal verification, and are propositional satisfiability solvers. Satisfiability Modulo Theories (SMT) [14], however, takes the decidability problem as first order logic formula and decides its satisfiability based on the decidable background theory. There are a number of theories supported by the SMT solvers, such as equality and uninterpreted functions, linear arithmetic over rationals, linear arithmetic over integers, non-linear arithmetic over reals, over arrays, bit vectors, and combinations. The SMT-Lib provides a common input platform for a number of solvers used in the verification of systems. Behavioral specifications of a system can also be represented using abstract models. The SMT solvers are then used to perform bounded model checking to explore a bounded symbolic execution of the model [19]. A number of solvers that support the SMT-Lib are available. Examples include BarceLogic [5], CVC4 [4], MathSAT [6], and Yices [10]. The differentiating features of solvers include the underlying logic, the underlying theories, the input formulas, and the interfaces. In this paper, we use the Z3 constraint solver, which is an efficient automated SMT solver by Microsoft Research Labs [12]. The Z3 solver has been used in the analysis and verification of software systems. The underlying verification theory for our system's model is the theory of array, which is used to prove the satisfiability of our model's logical formulae. The array theory is frequently used in the software modeling domain [9].

B. High-level Petri Nets (HLPN)

Petri Nets have been used in the modeling of a wide range of systems, such as asynchronous, concurrent, distributed, non-deterministic, parallel and stochastic systems [17]. However, there is a tradeoff between modeling generality and analysis capability in Petri Nets. Even an average model can become too large for analysis. In this work, we use HLPN for the formal verification of the proposed technique, which is a set of 7-tuple, $N = (P, T, F, \varphi, R, L, M_0)$:

1) P denotes a set of finite places;
2) T denotes a set of finite transitions, where both P and T are two distinct sets (i.e. $P \cap T = \phi$);
3) F represents the flow relation from place to transition or transition to place, such that $F \subseteq (P \times T) \cup (T \times P)$;
4) ϕ represents the mapping function that maps places to data types, such that ϕ: P $\rightarrow$ Data types;
5) R defines the set of rules that maps T to logical formulae, such that R: T $\rightarrow$ Formula;
6) L represents the labels that are mapped on each flow in F, such that L: F $\rightarrow$ Label; and
7) M_0 represents the initial state/marking where the flow can be initiated, such that M: P $\rightarrow$ Tokens [13].

Information about the structure of the Petri Net is captured in (P, T, F), and the static semantics of the Petri Net are captured in (ϕ, R, L).

C. Related Work

Role based access control (RBAC) enables fine-grained access control (and generally in a single domain). Different extensions of RBAC have been proposed in the literature to support multi-domain access control. These approaches rely on a single body responsible for maintaining cross-domain policies. However, in a cloud environment, each user (individual or organization) may have one or more tenants and have a separate management infrastructure. Therefore, it is likely that users are not able to agree on a single organization to manage access control on their behalf. With the increased trend of cloud services due to its various benefits (e.g. on-demand self-service model and resources sharing among tenants), it is essential for CSPs to provide mechanisms to segregate the data of the tenants.

An advanced Hierarchical Open Stack Access Control model was proposed in [22], which is designed to facilitate secure and effective management of information sharing in a community cloud for both routine and cyber incident response needs. A cross-tenant trust model and its RBAC extension was proposed in [20] for enabling secure cross-tenant communication. A multi-tenant authorization as a Service (MTAaaS) platform to enforce such cross-tenant trust model is also presented in the paper. In a separate work, an autonomous multi-tenant network security framework "Jobber" was proposed [18]. However, the security of the approaches in these three studies was not demonstrated.

As computing resources are being shared between tenants and used in an on-demand manner, both known and zero-day system security vulnerabilities could be exploited by the attackers (e.g. using side-channel and timing attacks) [1]. In [16], a fine grained data-level access control model (FDACM) designed to provide role-based and data-based access control for multi-tenant applications was presented. Relatively lightweight expressions were used to represent complex policy rules. Again, the security of the approach was not provided.

Zhao et al. [23] propose a cross-domain single sign on authentication protocol for cloud users, whose security was

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2016.2646639, IEEE Transactions on Information Forensics and Security

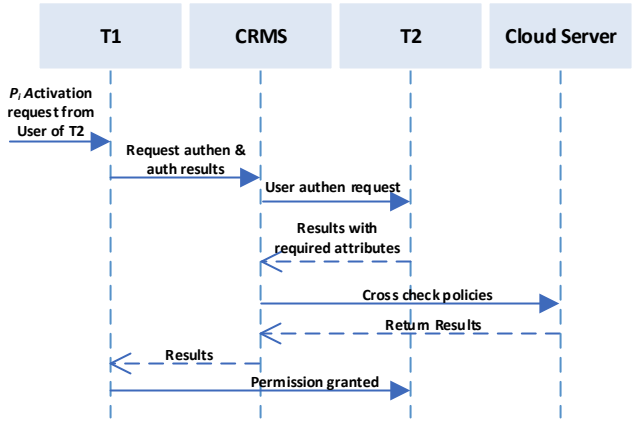

Fig. 1: Sequence diagram for permission request in the cloud

also proven mathematically. In the approach, the CSP is responsible for verifying the user's identity and making access control decisions. Specification level security is difficult to achieve at the user and provider ends.

III. Cloud Resource Mediation Service (CRMS)– A trusted third party for enabling cross tenant resource access

In this section, we describe our proposed CRMS designed to facilitate the CSPs in managing cross-tenant resource access requests for cloud users. To explain the service, we use an example involving two tenants, T_1 and T_2, where T_1 is the Service Provider (SP) and T_2 is the Service Requester (SR) (i.e. user). T_1 must own some permission p_i for which user of T_2 can generate a cross-tenant request. The resource request from a user of T_2 must be submitted to T_1, which then handovers the request to the CRMS for authentication and authorization decisions. The CRMS evaluates the request based on the security polices provided by T_1. The steps are represented in Figure 1 and explained in Section III-A.

A. Steps for permission activation request in the cloud

There are three main entities, namely: the SP (T_1), the SR (T_2), and the CRMS. The roles of these entities are described as follows:

a) Tenant T_1 responsibilities: T_1 is responsible for publishing cross tenant policies on the CRMS. T_1 receives access requests from T_2 and redirects the request to the CRMS for further processing.

b) Tenant T_2 responsibilities: The CRMS redirects access requests to T_2 for authentication. Once the redirected access request is received, the responsibility of T_2 is to authenticate the identity of particular user. In response, T_2 sends the user authentication response (valid or invalid) and tenant authentication response to the CRMS.

c) CRMS responsibilities: The CRMS receives the permission-activation request redirected from T_1. Once an access request is received, the CRMS evaluates the request on the pre-published policies and responds to T_1.

The steps for initiating a permission-activation request are as follows:

Step 1: Permission activation request: A user wishing to access a resource at T_1. The user will be presented a directory where a list of shared services along with their descriptions are present.

Step 2: Request redirection to the CRMS: Upon selection of a shared service the user wishes to access, the user is redirected to the CRMS site. On the site, the user will be asked for the parent tenant. The user selects the parent tenant and the CRMS redirects the user's request to the selected tenant (T_2 in this case).

Step 3: Tenant T_2 authentication: The user has to authenticate at her parent tenant, T_2. Upon successful authentication, the user will be redirected again to CRMS with the attributes requested by the CRMS for cross tenant policy execution.

Step 4: CRMS redirection to tenant T_1 & permission activation: The user's attributes are evaluated against the T_1 policy and if the policy criteria is successfully fulfilled, then the user is provided service access at T_1; otherwise, the access request is denied. The CRMS also takes into account any conflict of interest policies, such as Chinese Wall Policy.

IV. Cross-Tenant Access Control (CTAC) Model

To deal with cross-tenant resource requests in the presence of CRMS, we describe the components of the CTAC model whose key components are as follows:

- U_i represents a set of intra-tenant user. A user from this set can also act as the delegator of the permission.
- U_j and U_k represent the sets of cross-tenant users. A user from either of the sets can also act as delegator (in the event that further delegation of permission is required) or delegatee (in the event that the permission is delegated to these users).
- P_i represents a set of permissions.
- $UPA_i \subseteq (U_i \times P_i)$. A binary relation between the intra-tenant users and the set of permissions assigned to them in the i^{th} tenant.
- $UPA_a \subseteq (U_i \times P_i)$. A binary relation between the intra-tenant users and the set of permissions activated by them in the i^{th} tenant.
- $LEU_a \subseteq (U_i \times U_j \times P_i)$. A triple relation among the intra-tenant users, cross-tenant users and the set of permissions activated by them in the i^{th} tenant.
- $EEU_a \subseteq (U_j \times U_k \times P_i)$. A triple relation among the cross-tenant users and the set of permissions activated by them in the i^{th} tenant.
- $LED_a \subseteq (U_i \times t \times P_i)$. A triple relation among the intra-tenant users, the tenant and the set of permissions activated by the users of that tenant in the i^{th} tenant.
- $EED_a \subseteq (U_k \times t \times P_i)$. A triple relation among the cross-tenant users, the tenant and the set of permissions activated by the users of that tenant in the i^{th} tenant.
- $LEUD\_POL \subseteq (U_i \times U_j \times P_i \times C)$. A quadruple relation among the intra-tenant users, the cross-tenant users, the set of permissions, and the set of delegation constraints published on the CRMS for the evaluation of the requested resource.
- $EEUD\_POL \subseteq (U_j \times U_k \times P_i \times C)$. A quadruple relation among the cross-tenant users, the set of permissions, and

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2016.2646639, IEEE Transactions on Information Forensics and Security

the set of delegation constraints published on the CRMS for the evaluation of the requested resource.

- $LEDD\_POL \subseteq (U_i \times t \times P_i \times C)$. A quadruple relation among the intra-tenant users, the tenant, the set of permissions, and the set of delegation constraints published on the CRMS for the evaluation of the requested resource.
- $EEDD\_POL \subseteq (U_k \times t \times P_i \times C)$. A quadruple relation among the cross-tenant users, the tenant, the set of permissions, and the set of delegation constraints published on the CRMS for the evaluation of the requested resource.

Definition 1: Activation Query: An activation query defines a request from an intra-tenant/cross-tenant user for the activation of a permission p_i, where $p_i \subseteq P_i$. We formally define an activation query as :

$$Activation_Q{:}U_{t=i,j} \times t \times p_i \rightarrow \{UPA'_a, LEU'_a, EEU'_a, LED'_a, EED'_a\}$$

For a cross-tenant user to successfully activate a particular permission, one of the following must be fulfilled.

- An intra-tenant user, after the activation of a permission, has delegated the requested permission to the cross-tenant user. In other words, an approved delegation must exist for the cross-tenant user.
- An intra-tenant/cross-tenant user has delegated the requested permission to a tenant (i.e. an approved delegation must exist for a particular tenant).

Definition 2: Delegation Query: A delegation query defines a request in which an intra-tenant/cross-tenant user delegates a subset of the permissions p_i (where $p_i \subseteq P_i$) to a cross-tenant user/tenant associated with a delegation constraint. We formally define a delegation query as :

$$Delegation_Q : (U_{t=i,k} \times U_{t=j}/t \times p_i \times C) \rightarrow \{U_i \rightsquigarrow^{p_i} U_j\}' | \{U_j \rightsquigarrow^{p_i} U_k\}' | \{U_i \rightsquigarrow^{p_i} t\}' | \{U_k \rightsquigarrow^{p_i} t\}' | LEUD-POL' | EEUD-POL' | LEDD-POL' | EEDD-POL' | Error$$

In the above, p_i denotes a permission that can be delegated to the cross-tenant user or the tenant, and C a set of delegation constraints.

Definition 3: Forward Revocation Query: A forward revocation query defines a request in which an intra-tenant user revoke a permission or a set of permissions from a cross-tenant user/tenant along with the deactivation of the delegation policy. We formally define a forward revocation query as:

$$Forward-Revocation_Q : (U_{t=i,j} \times p_i) \rightarrow UPA'_i | \{U_i \rightsquigarrow^{p_i} U_j\}' | \{U_j \rightsquigarrow^{p_i} U_k\}' | \{U_i \rightsquigarrow^{p_i} t\}' | \{U_k \rightsquigarrow^{p_i} t\}' | \{Policies\}'$$

Definition 4: Backward Revocation Query: A backward revocation query defines a action that is triggered when the attributes of the delegatee mismatch. Thus, an intra-tenant user revokes a permission or a set of permissions from a cross-tenant user/tenant as well as deactivating the delegation policy. We formally define a backward revocation query as:

$$Backward-Revocation_Q : (U_{t=j,k} \times p_i \times Att_{usr} \mid Att_t) \rightarrow \{U_i \rightsquigarrow^{p_i} U_j\}' | \{U_j \rightsquigarrow^{p_i} U_k\}' | \{U_i \rightsquigarrow^{p_i} t\}' | \{U_k \rightsquigarrow^{p_i} t\}' | \{Policies\}'$$

Definition 5: Cross-tenant Delegation: We define cross-tenant delegation as a triple $(U_{t=i,j,k}, U_{t=j,k}/t, p_i)$, where $(U_{t=i,j,k})$ is a delegator, $(U_{t=j,k}/t)$ is a delegatee and p_i is a permission or a set of delegated permissions. We formally define a cross-tenant delegation as :

$$(U_{t=i,j,k}, U_{t=j,k}/t, p_i) \in \{U_{t=i,j,k} \rightsquigarrow^{p_i} U_{t=j,k}/t\}$$

In the above, $\{U_{t=i,j,k} \rightsquigarrow^{p_i} U_{t=j,k}/t\}$ denotes a set of delegation pairs.

Definition 6: Tenant-user Delegation: We define tenant-user delegation as a triple $(t, U_{t=j,k}, p_i)$, where t is a delegator, $U_{t=j,k}$ is a delegatee (an internal user of the tenant) and p_i is a permission or a set of delegated permissions. We formally define a tenant-user delegation as :

$$(t, U_{t=j,k}, p_i) \in \{t \rightsquigarrow^{p_i} U_{t=j,k}\}$$

In the above, $\{t \rightsquigarrow^{p_i} U_{t=j,k}\}$ denotes a set of delegation pairs.

V. Formal Specification and Modeling of CTAC flows using HLPN

A. Permission activation flow in the CTAC model

We now define four algorithms for the CTAC model. These algorithms capture the manner in which an intra-tenant user or a cross-tenant user activates, delegates or revokes a permission.

B. Activation Algorithm

The activation algorithm is based on the activation query defined in Section IV. It authenticates a user for the activation of a particular permission. As defined earlier, a permission activation request can be generated by an intra-tenant/cross-tenant user. For a cross-tenant user, a prior delegation of permission to cross-tenant user/tenant must exist according to Definition 2. The algorithm for activation of the permission under the proposed CRMS for CSPs is shown in Figure 2.

(1) $Activation_Q\ (u_i|u_j, t, p_i)$
(2) Output: $UPA_a{}', LEU_a{}', EEU_a{}', LED_a{}', EED_a{}'$
(3) if(i = t) then
(4) if$(u_i, p_i) \in UPA_i$ then
(5) $UPA_a{}' = UPA_a \cup (u_i, p_i)$
(6) else
(7) if$(u_i, u_j, p_i) \in \{U_i \rightsquigarrow^{p_i} U_j\}$ \\intra-tenant user to a cross-tenant user permission delegation set
(8) $LEU_a{}' = LEU_a \cup (u_i, u_j, p_i)$
(9) else
(10) if$(u_j, u_k, p_i) \in \{U_j \rightsquigarrow^{p_i} U_k\}$ \\cross-tenant user to a cross-tenant user permission delegation set
(11) $EEU_a{}' = EEU_a \cup (u_j, u_k, p_i)$
(12) else
(13) if$(u_i, t, p_i) \in \{U_i \rightsquigarrow^{p_i} t\} \wedge (p_k, p_i) \notin SMEP$ \\intra-tenant user to a tenant permission delegation set
(14) $\{t \rightsquigarrow^{p_i} U_j\}' = \{t \rightsquigarrow^{p_i} U_j\} \cup (t, u_j, p_i)$ \\activation set of a delegated permission by a cross-tenant user which is assigned to it by its tenant
(15) $LED_a{}' = LED_a \cup (u_i, t, p_i)$
(16) else
(17) if$(u_k, t, p_i) \in \{U_k \rightsquigarrow^{p_i} t\} \wedge (p_k, p_i) \notin SMEP$ \\cross-tenant user to a tenant permission delegation set
(18) $\{t \rightsquigarrow^{p_i} U_j\}' = \{t \rightsquigarrow^{p_i} U_j\} \cup (t, u_j, p_i)$
(19) $EED_a{}' = EED_a \cup (u_k, t, p_i)$
(20) else
(21) return false

Fig. 2: Activation Algorithm

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2016.2646639, IEEE Transactions on Information Forensics and Security

Using HLPN, we model the Activation Algorithm. The HLPN in Figure 3 illustrates the process of activating a permission p_i in a cross-tenant environment.

There are four different ways in which the permission p_i can be activated in a multi-tenant environment. When the condition for any of the cases is satisfied, subsequent permission p_i for the corresponding user will be activated. If none of the cases are satisfied, the algorithm terminates.

As stated in the formal definition of the Petri Net presented in [13], the HLPN is a 7-tuple $N = (P, T, F, \varphi, R, L, M_0)$. The first step to model a system in the HLPN is to define a set of P (Places) and the associated data types. As depicted in Figure 3, there are 21 places in the model. The names and the mappings of P are shown in Table I.

We will now model and analyze the granular level details of activating a permission p_i in a cross-tenant environment. We will also define the set of rules, preconditions, and post-conditions to map to T. The set of transitions T is : $T = \{$*Start-Act, L-D-Chk, Upa-Chk, L-Eu-Dchk, Leu-Act-C, Eeu-D-Chk, Eeu-Act-C, Led-Smp-Chk, Smp-Chk, Edu-Upd-C1, Eed-Smp-Chk, Edu-Upd-C2, Failure*$\}$. The flow begins at the transition *Start-Act*. There are no pre-conditions that need to be satisfied, as shown in $\boldsymbol{R}$ *(Start-Act)* = $\exists$ *s-act* $\in$ S-Act $\mid \bullet$ *s-act* = ϕ.

The user-permission activation flow starts by evaluating the user against the user's own tenant. If the user belongs to his/her own tenant, then a result is produced as shown in R 1.

$$\begin{aligned}\boldsymbol{R}\ \textit{(Lu-D-Chk)} = \forall \textit{a-q} \in A - Q, \forall \textit{l-dset} \in L - Dset, \forall \\ \textit{l-e-rlt} \in L - E - Rlt \mid \textit{a-q}[1] = \textit{l-dset}[1], \textit{a-q}[3] \\ = \textit{l-dset}[2] \rightarrow \textit{l-e-rlt}[1] := CRT - RLT(\textit{a-q}[1], \textit{a-q}[3]) \\ L - E - Rlt' = \{L - E - Rlt\} \cup \{\textit{l-e-rlt}[1]\} \quad (1)\end{aligned}$$

Assuming the result is true, the set of user-permission assignment UPA_i is evaluated for the user. If the permission p_i is already assigned to the user in this set, then the algorithm activates the subsequent permission p_i for the user in the tenant, updates its user-permission activation set UPA_a as shown in R 2 prior to exiting.

$$\begin{aligned}\boldsymbol{R}\ \textit{(Upa-Chk)} = \forall \textit{s-l-erlt} \in S - L - Erlt, \\ \forall \textit{upa-set} \in Upa - Set, \forall \textit{a-q} \in A - Q, \forall \textit{l-act-t} \in L - Act - T \mid \\ \textit{s-l-erlt}[1] = True,\ \textit{a-q}[1] = \textit{upa-set}[1], \textit{a-q}[2] = \\ \textit{upa-set}[2] \rightarrow \textit{l-act-t}[1] := \textit{a-q}[1], \textit{l-act-t}[2] := \textit{a-q}[2] \\ L - Act - T' = \{L - Act - T\} \cup \{\textit{l-act-t}[1], \textit{l-act-t}[2]\} \quad (2)\end{aligned}$$

In the event that the user and the permission do not exist in the intra-tenant user-permission assignment set UPA, the next case for consideration is the delegation of permission p_i to some cross-tenant user in cross-tenant. The cross-tenant permission delegation is of the following types:

- An intra-tenant user to cross-tenant user permission delegation; or
- A cross-tenant user to cross-tenant user permission delegation.

To check whether the user from a tenant has delegated the permission p_i to some other user(s) (i.e. cross-tenant user belonging to some other tenant), the algorithm evaluates the intra-tenant to cross-tenant user delegation set. If the intra-tenant user has previously delegated the permission p_i to the cross-tenant user and it does exists in that set, then a result will be produced as shown in R 3.

TABLE I: Places and Mapping of the Activation Algorithm

Places	Mapping	Description
φ(ACT_QUERY)	$\wp$(luser× permission× domain× euser)	Holds activation query attributes
φ(U_D_SET)	$\wp$(luser×domain)	Stores user against its own tenant attributes
φ(U_D_RLT)	$\wp$(ud-rlt)	Holds a result attribute by evaluating the user against its own tenant
φ(UPA)	$\wp$(luser× permission)	Captures user to permission assignment attributes
φ(LUa)	$\wp$(upa-result)	Holds upa result attribute
φ(L_EX_ DEL_SET)	$\wp$(luser× permission× euser)	Holds local user to cross-tenant user delegation attributes
φ(L_EX_ DEL_RLT)	$\wp$(leu-rlt)	Holds local to cross-tenant user delegation result attribute
φ(LEUa)	$\wp$(luser× permission× euser)	Holds activated local to cross-tenant user delegation attributes
φ(EX_EX_ DEL_SET)	$\wp$(euser1 ×permission ×euser2)	Holds cross-tenant user to cross-tenant user delegation attributes
φ(EX_EX_ DEL_RLT)	$\wp$(eeu-rlt)	Holds cross-tenant to cross-tenant user delegation result attribute
φ(EEUa)	$\wp$(euser1× permission× euser2)	Holds activated cross-tenant to cross-tenant user delegation attributes
φ(SMEP)	$\wp$(cpermission× cuser)	Holds statically mutually exclusive permission (SMEP) attributes
φ(PRMISSIONS)	$\wp$(smp-per-rlt)	Holds SMEP result attribute
φ(DOM_LOCL_ EX_SET)	$\wp$(luser×domain ×permission)	Holds local user to cross-tenant delegation attributes
φ(DOM_LEX_ DELE_RLT)	$\wp$(leu-rlt-d)	Holds local user to cross-tenant delegation result attribute
φ(LEDa)	$\wp$(luser×domain ×permission)	Holds activated local user to cross-tenant delegation attributes
φ(DOM_EXX_ DELE_SET)	$\wp$(euser×domain ×permission)	Holds cross-tenant user to cross-tenant delegation attributes
φ(DOM_EEX_ DELE_RLT)	$\wp$(eeu-rlt-d)	Holds cross-tenant user to cross-tenant delegation result attribute
φ(EEDa)	$\wp$(euser×domain ×permission)	Holds activated cross-tenant user to cross-tenant delegation attributes
φ(DOM_LOCL_ EX_ASSIGN)	$\wp$(domain×duser ×permission)	Holds cross-tenant to its own user delegation attributes
φ(Error)	$\wp$(fail)	Holds failure attributes

$$\begin{aligned}\boldsymbol{R}\ \textit{(L-Eu-Dchk)} = \forall \textit{a-q} \in A - Q, \\ \forall \textit{f-l-edrlt} \in F - L - Edrlt, \forall \textit{leu-set} \in Leu - Set, \\ \forall \textit{leu-rlt} \in Leu - Rlt \mid \textit{f-l-edrlt}[1] = False, \\ \textit{a-q}[1] = \textit{leu-set}[1], \textit{a-q}[4] = \textit{leu-set}[3], \\ \textit{a-q}[2] = \textit{leu-set}[2] \rightarrow \textit{leu-rlt}[1] := CRT - RLT \\ (\textit{a-q}[1], \textit{a-q}[2], \textit{a-q}[4]) \\ Leu - Rlt' = \{Leu - Rlt\} \cup \{\textit{leu-rlt}[1]\} \quad (3)\end{aligned}$$

If the result is true, then the algorithm activates the permission p_i for the cross-tenant user, updates its intra-tenant-cross-

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2016.2646639, IEEE Transactions on Information Forensics and Security

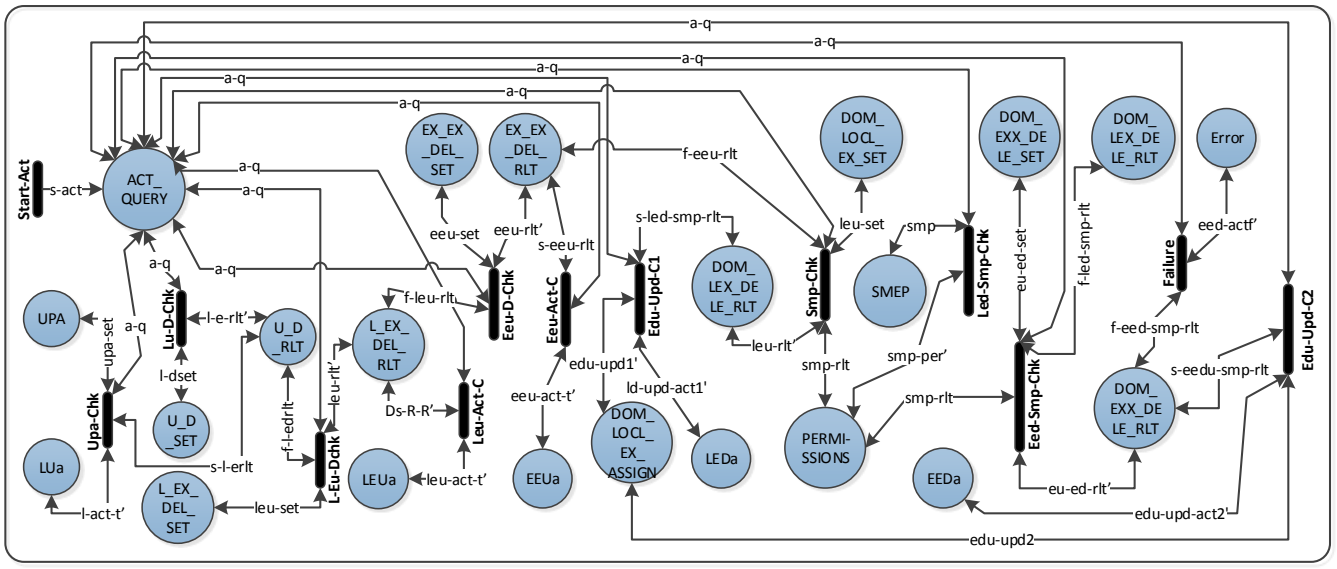

Fig. 3: A HLPN Model for the process of activating a permission p_i in a cross-tenant environment

tenant user activation set LEU_a as shown in R 4, and exits.

$$\begin{aligned}\textbf{R}\ \textit{(Leu-Act-C)} = \forall \textit{a-q} \in A - Q,\\ \forall \textit{s-leu-rlt} \in S - Leu - Rlt,\\ \forall \textit{leu-act-t} \in Leu - Act - T \mid\\ \textit{s-leu-rlt}[1] = True, \textit{leu-act-t}[1] := \textit{a-q}[1],\\ \textit{leu-act-t}[2] := \textit{a-q}[4], \textit{leu-act-t}[3] := \textit{a-q}[2]\\ Leu - Act - T' = \{Leu - Act - T\}\\ \cup \{\textit{leu-act-t}[1], \textit{leu-act-t}[2], \textit{leu-act-t}[3]\} \quad (4)\end{aligned}$$

Alternatively, the algorithm evaluates the next option, which is checking the user against cross-tenant to cross-tenant user permission delegation set. If the result turns out to be false, then the algorithm checks for the presence of input query parameters in the cross-tenant to cross-tenant delegation set. If such a delegation has previously been been performed, then a result will be produced as shown in R 5.

$$\begin{aligned}\textbf{R}\ \textit{(Eeu-D-Chk)} = \forall \textit{a-q} \in A - Q, \forall \textit{f-leu-rlt} \in F - Leu - Rlt,\\ \forall \textit{eeu-set} \in Eeu - Set, \forall \textit{eeu-rlt} \in Eeu - Rlt \mid\\ \textit{f-leu-rlt}[1] = False, \textit{a-q}[1] = \textit{eeu-set}[1],\\ \textit{a-q}[2] = \textit{eeu-set}[2], \textit{a-q}[4] = \textit{eeu-set}[3]\\ \rightarrow \textit{eeu-rlt}[1] := CRT - RLT(\textit{a-q}[1], \textit{a-q}[2], \textit{a-q}[4])\\ Eeu - Rlt' = \{Eeu - Rlt\} \cup \{\textit{eeu-rlt}[1]\} \quad (5)\end{aligned}$$

If the result is true, then the algorithm activates the permission p_i for the user, updates its cross-tenant-to-cross-tenant user activation set EEU_a set as shown in R 6, before exiting.

$$\begin{aligned}\textbf{R}\ \textit{(Eeu-Act-C)} = \forall \textit{a-q} \in A - Q, \forall \textit{eeu-act-t}\\ \in Eeu - Act - T, \forall \textit{s-eeu-rlt} \in S - Eeu - Rlt \mid\\ \textit{s-eeu-rlt}[1] = True, \textit{eeu-act-t}[1] = \textit{a-q}[1],\\ \textit{eeu-act-t}[2] = \textit{a-q}[2], \textit{eeu-act-t}[3] = \textit{a-q}[4] \rightarrow\\ Eeu - Act - T' = \{Eeu - Act - T\} \cup \{\textit{eeu-act-t}[1],\\ \textit{eeu-act-t}[2], \textit{eeu-act-t}[3]\} \quad (6)\end{aligned}$$

In the event that the user level delegation does not exist for a cross-tenant user, then the algorithm will check for intra-tenant user to cross-tenant permission delegation. For a permission to be delegated to a tenant, there is an additional check known as Statically Mutually Exclusive Permissions (SMEP). This check will evaluate the permission p_i against the set of permissions p_k for any conflicts, as shown in R 7.

$$\begin{aligned}\textbf{R}\ \textit{(Led-Smp-Chk)} = \forall \textit{a-q} \in A - Q,\\ \forall \textit{smp-per} \in Smp - Per \mid \textit{a-q}[4] = \textit{smp}[2] \rightarrow\\ \textit{smp-per}[1] := CHK - CONFLICT(\textit{a-q}[2])\\ Smp - Per' = \{Smp - Per\} \cup \{\textit{smp-per}[1]\} \quad (7)\end{aligned}$$

If the result is false (i.e. there does not exist a conflict between the permissions), then the algorithm will check for intra-tenant to cross tenant permission delegation. If such a delegation has previously been performed, then a result will be produced as shown in R 8.

$$\begin{aligned}\textbf{R}\ \textit{(SMP-Chk)} = \forall \textit{a-q} \in A - Q, \forall \textit{smp-rlt} \in Smp - Rlt,\\ \forall \textit{leu-set} \in Leu - Set, \forall \textit{f-eeu-rlt} \in F - Eeu - Rlt,\\ \forall \textit{leu-rlt} \in Leu - Rlt \mid \textit{f-eeu-rlt}[1] = False,\\ \textit{smp-rlt}[1] = False \rightarrow \textit{a-q}[1] = \textit{leu-set}[1],\\ \textit{a-q}[3] = \textit{leu-set}[2], \textit{a-q}[2] = \textit{leu-set}[3]\\ \textit{leu-rlt}[1] := CRT - RLT(\textit{a-q}[1], \textit{a-q}[2],\\ \textit{a-q}[3]) Leu - Rlt' = \{Leu - Rlt\} \cup \{\textit{leu-rlt}[1]\} \quad (8)\end{aligned}$$

If the result is true, the algorithm activates the permission p_i for the user, updates its local-cross tenant activation set LED_a set along with assigning the permission p_i to user internal to the tenant as shown in R 9.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2016.2646639, IEEE Transactions on Information Forensics and Security

$$\begin{aligned}\textbf{R}\ \textit{(Edu-Upd-C1)} = \forall \textit{a-q} \in A-Q, \forall \textit{s-led-smp-rlt} \in \\ S-Led-Smp-Rlt, \forall \textit{ld-upd-act1} \in Ld-Upd-Act1, \\ \forall \textit{edu-upd1} \in Edu-Upd1 \mid \textit{s-led-smp-rlt}[1] = True \rightarrow \\ \textit{edu-upd1}[1] := \textit{a-q}[1], \textit{edu-upd1}[2] := \textit{a-q}[3], \\ \textit{edu-upd1}[3] := \textit{a-q}[2], \textit{ld-upd-act1}[1] := \textit{a-q}[1], \\ \textit{ld-upd-act1}[2] := \textit{a-q}[2], \textit{ld-upd-act1}[3] := \textit{a-q}[3] \\ Edu-Upd1' = \{Edu-Upd1\} \cup \{\textit{edu-upd1}[1], \\ \textit{edu-upd1}[2], \textit{edu-upd1}[3]\} Ld-Upd-Act1' = \\ \{Ld-Upd-Act1\} \cup \{\textit{ld-upd-act1}[1], \\ \textit{ld-upd-act1}[2], \textit{ld-upd-act1}[3]\} \end{aligned} \quad (9)$$

The same process will be followed by the algorithm to check cross-tenant user to cross-tenant permission delegation. After checking the SMEP check, a result will be produced. If the result is true, then the algorithm activates the permission p_i for the user, updates its cross-cross tenant activation set EED_a along with assigning the permission p_i to user internal to the tenant. After evaluating all the cases for permission activation and if none of them are satisfied, then the algorithm terminates the process.

C. Delegation Algorithm

The delegation algorithm is based on the delegation query defined in Section IV. The delegation algorithm enables an intra-tenant user to generate a delegation request for the permission which the user can activate. The algorithm for delegation of the permission under the proposed CRMS for CSPs is shown in Figure 4.

(1) $Delegation_Q\ (u_i|u_j, t, p_i, C)$
(2) output: $\{U_i \rightsquigarrow^{p_i} U_j\}', \{U_j \rightsquigarrow^{p_i} U_k\}', \{U_i \rightsquigarrow^{p_i} t\}', \{U_k \rightsquigarrow^{p_i} t\}',$ $ledpolicy', eedpolicy', ledompolicy', eedompolicy', error$
(3) for all p_k in P_k {
(5) for all p_i in P_i {
(7) if (p_k,p_i) $\in SMEP$ then
(8) return error;
(9) else {
(11) $if(u_i, u_j, p_i) \in \{U_i \rightsquigarrow^{p_i} U_j\} \vee (u_j, u_k, p_i) \in \{U_j \rightsquigarrow^{p_i} U_k\}$
(12) return error
(13) else {
(15) if(i=t) then {
(17) $\{U_i \rightsquigarrow^{p_i} U_j\}' = \{U_i \rightsquigarrow^{p_i} U_j\} \cup (u_i, u_j, p_i)$
(18) $ledpolicy' = ledpolicy \cup (u_i, u_j, p_i, C)$ }
(20) else {
(22) $\{U_j \rightsquigarrow^{p_i} U_k\}' = \{U_j \rightsquigarrow^{p_i} U_k\} \cup (u_j, u_k, p_i)$
(23) $eedpolicy' = eedpolicy \cup (u_j, u_k, p_i, C)$
(24) }}}}}
(25) $if(u_i, t, p_i) \in (U_i \rightsquigarrow^{p_i} t) \vee (u_k, t, p_i) \in (U_k \rightsquigarrow^{p_i} t) then$
(26) return error
(27) else {
(29) if(i=t) then {
(31) $\{U_i \rightsquigarrow^{p_i} t\}' = \{U_i \rightsquigarrow^{p_i} t\} \cup (u_i, t, p_i)$
(32) $ledompolicy' = ledompolicy \cup (u_i, t, p_i, C)$ }
(34) else {
(36) $\{U_k \rightsquigarrow^{p_i} t\}' = \{U_k \rightsquigarrow^{p_i} t\} \cup (u_k, t, p_i)$
(37) $eedompolicy' = eedompolicy \cup (u_k, t, p_i, C)$ } }

Fig. 4: Delegation Algorithm

Using HLPN, we model the Delegation Algorithm. The HLPN in Figure 5 illustrates the process of delegating a permission p_i in a cross-tenant environment.

There are four different ways in which the permission p_i can be delegated. When the condition of any of the cases is satisfied, subsequent permission p_i for the corresponding user will be delegated. There are two types of delegation that exist in the system, namely: user-level delegation and tenant-level delegation. Failure of one of these two cases will result in the checking of the other case. If none of the cases are satisfied, then the algorithm terminates and the permission delegation for the corresponding cross-tenant user/ cross-tenant fails.

The flow starts at the transition *Start-Dele*. There are no pre-conditions which need to be satisfied as shown in ***R*** *(Start-Dele)* = $\exists$ *s-dele* $\in$ S-Dele $\mid \bullet$ *s-dele* = ϕ. The delegation query takes the subsequent parameters as input, user $u_i \mid u_j$, tenant j, permission p_i, and constraint C. The output of the algorithm is a successful delegation of permission to cross-tenant user/cross-tenant. The permission delegation flow moves further by evaluating the permission coming from the delegation input query against the set of Statically Mutually Exclusive Permissions (SMEP). As defined earlier, SMEP is a check that evaluates the permission for conflicts with other delegated permissions as shown in R 10.

$$\begin{aligned}\textbf{R}\ \textit{(Smp-Chk)} = \forall \textit{smp-per} \in SMP-PER, \forall \textit{d-q} \in D-Q, \\ \forall \textit{smp-set} \in Smp-Set \mid \textit{d-q}[4] = \textit{smp-set}[2] \rightarrow \textit{smp-per}[1] \\ := CHK-CONFLICT(\textit{d-q}[4]) SMP-PER' \\ = \{SMP-PER\} \cup \{\textit{smp-per[1]}\} \end{aligned} \quad (10)$$

For brevity, the remaining rules for delegation algorithm are not presented in this paper although they were considered in our verification process in Section VI.

D. Revocation Algorithm

There are two ways in which we can revoke a previously granted permission from the cross-tenant user/cross-tenant.

1) Revoke the permission from the service provider's side; or
2) Revoke the permission when the attributes of the user change and no longer match the published security policies on the CRMS.

To achieve the permission revocation, we introduce the Forward Revocation Algorithm and the Backward Revocation Algorithm.

1) Forward Revocation Algorithm: The forward revocation algorithm is based on the forward revocation query defined in Section IV. This algorithm enables an intra-tenant user to generate a permission revocation request for the permission that is delegated to a cross-tenant user/cross-tenant. The permission is revoked at both user and tenant levels. All subsequent delegations also need to be revoked along with deactivating/invalidating the security policy for the said permission on the CRMS. The algorithm for forward revocation of the permission under the proposed CRMS for CSPs is shown in Figure 6. Using HLPN, we model the forward revocation Algorithm. The HLPN in Figure 7 illustrates the process of revoking a permission p_i in a cross-tenant environment.

2) Backward Revocation Algorithm: The backward revocation algorithm is based on the backward revocation query defined in Section IV. The backward revocation algorithm is invoked on the CRMS when the attribute of the delegatee (cross-tenant user/tenant) does not match the delegation

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2016.2646639, IEEE Transactions on Information Forensics and Security

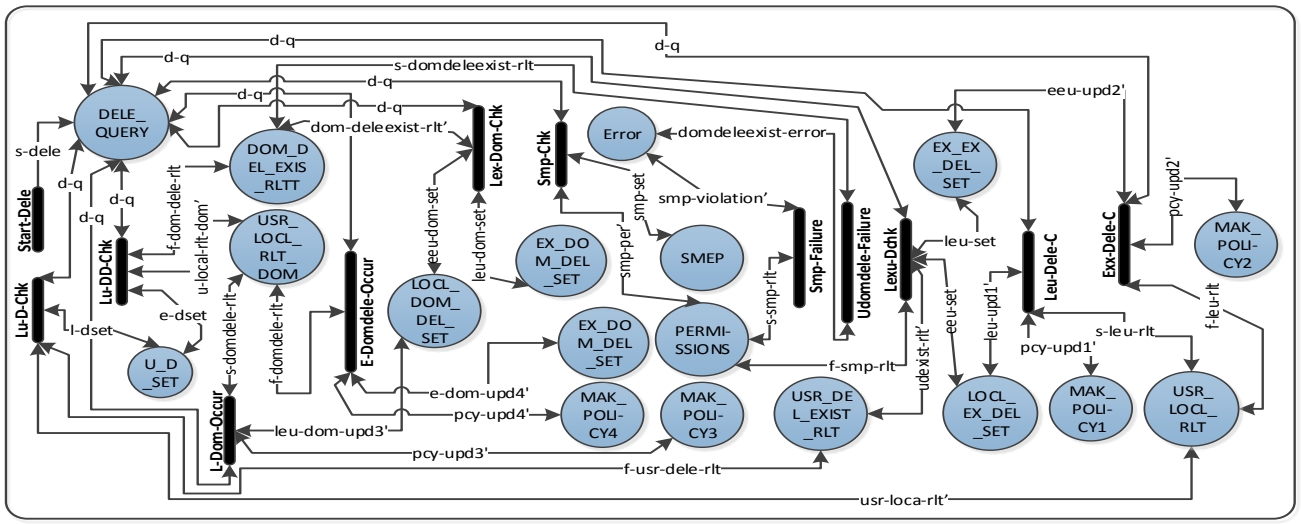

Fig. 5: A HLPN Model for the process of delegating a permission p_i in a cross-tenant environment

(1) $ForwardRevocation_Q(u_i|u_j\ ,\ p_i\ ,t,\ Att_{usr})$
(2) output: $UPA', \{U_i \rightsquigarrow^{p_i} U_j\}', \{U_j \rightsquigarrow^{p_i} U_k\}', \{U_i \rightsquigarrow^{p_i} t\}',$
$\{U_k \rightsquigarrow^{p_i} t\}', ledpolicy', eedpolicy', ledompolicy', eedompolicy'$
(3) if$(u_i, p_i) \in UPA$
{
(4)$UPA' = UPA - (u_i, p_i)$
(5) $if(u_i, u_j, p_i) \in \{U_i \rightsquigarrow^{p_i} U_j\}$
(6) $\{U_i \rightsquigarrow^{p_i} U_j\}' = \{U_i \rightsquigarrow^{p_i} U_j\}' - (u_i, u_j, p_i)$
(7) $ledpolicy' = ledpolicy - (u_i, u_j, p_i, C)$
(8) $if(u_j, u_k, p_i) \in \{U_j \rightsquigarrow^{p_i} U_k\}$
(9) $\{U_j \rightsquigarrow^{p_i} U_k\}' = \{U_j \rightsquigarrow^{p_i} U_k\}' - (u_j, u_k, p_i)$
(10) $eedpolicy' = eedpolicy - (u_j, u_k, p_i, C)$
(11) $if(u_i, t, p_i) \in \{U_i \rightsquigarrow^{p_i} t\}$
(12) $\{U_i \rightsquigarrow^{p_i} t\}' = \{U_i \rightsquigarrow^{p_i} t\} - (u_i, t, p_i)$
(13) $\{t \rightsquigarrow^{p_i} U_j\}' = \{t \rightsquigarrow^{p_i} U_j\} - (t, u_j, p_i)$
(14) $ledompolicy' = ledompolicy - (u_i, t, p_i, C)$
(15) $if(u_k, t, p_i) \in \{U_k \rightsquigarrow^{p_i} t\}$
(16) $\{U_k \rightsquigarrow^{p_i} t\}' = \{U_k \rightsquigarrow^{p_i} t\} - (u_k, t, p_i)$
(17) $\{t \rightsquigarrow^{p_i} U_j\}' = \{t \rightsquigarrow^{p_i} U_j\} - (t, u_j, p_i)$
(18) $eedompolicy' = eedompolicy - (u_k, t, p_i, C)$
(19) }
(20) else
(21) do nothing

Fig. 6: Forward Revocation Algorithm

constraint defined in the security policy. In this scenario, it is necessary to remove the corresponding delegation triples from user-level delegation sets. We will also revoke the tenant level delegations of this permission along with deactivating/invalidating the corresponding policy on the CRMS. Again for brevity, we will not discuss the rules for forward and backward revocation algorithms although they were considered in the verification process discussed next.

VI. Verification of the CTAC model

a) The System verification: The correctness of a system is demonstrated by the verification process. To prove the correctness of the system under consideration, the system is verified on the system specifications, and the system properties.

b) The CTAC model verification using the Z3 constraint solver: We verified the CTAC model by proving the correctness of activation algorithm, delegation algorithm, forward revocation algorithm, and backward revocation algorithm. Each algorithm was modeled, analyzed, and verified. Specifically, the algorithm was modeled using HLPN, and the Z formal language was used to define transition rules. The array theory of SMT-Lib was then used to transform such rules. Finally, the properties of the algorithm were verified using the Z3 solver. The properties of algorithm were also checked for satisfiability of logical formulae over the algorithm specification. The Z3 solver performed the computations and generated result as satisfiable *sat* or unsatisfiable *unsat*. If the generated output is *sat*, then this indicates that there is a violation of the asserted property and the solver will generate a counter example. Alternatively, if the generated output is *unsat*, then this shows that the property holds and implies the correctness of the algorithm.

We defined the following algorithm specific security properties for each model. The properties were translated in array theory and then the Z3 solver examined them to determine the correctness of each algorithm.

c) Security Properties for Activation Algorithm: The activation algorithm specific properties are as follows:

- **(Activation.a) Intra-tenant user to permission assignment property** holds when the intra-tenant user and the permission both exist in the intra-tenant-permission assignment set.
- **(Activation.b) Intra-tenant user to cross-tenant user activation property** holds when the intra-tenant user, cross-tenant user and the permission match the intra-cross-tenant-user-permission set.
- **(Activation.c) Cross-tenant user to cross-tenant user activation property** holds when the two cross-tenant users and the permission match the cross-cross-tenant-user-permission set.
- **(Activation.d) Intra-tenant user to cross-tenant acti-**

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2016.2646639, IEEE Transactions on Information Forensics and Security

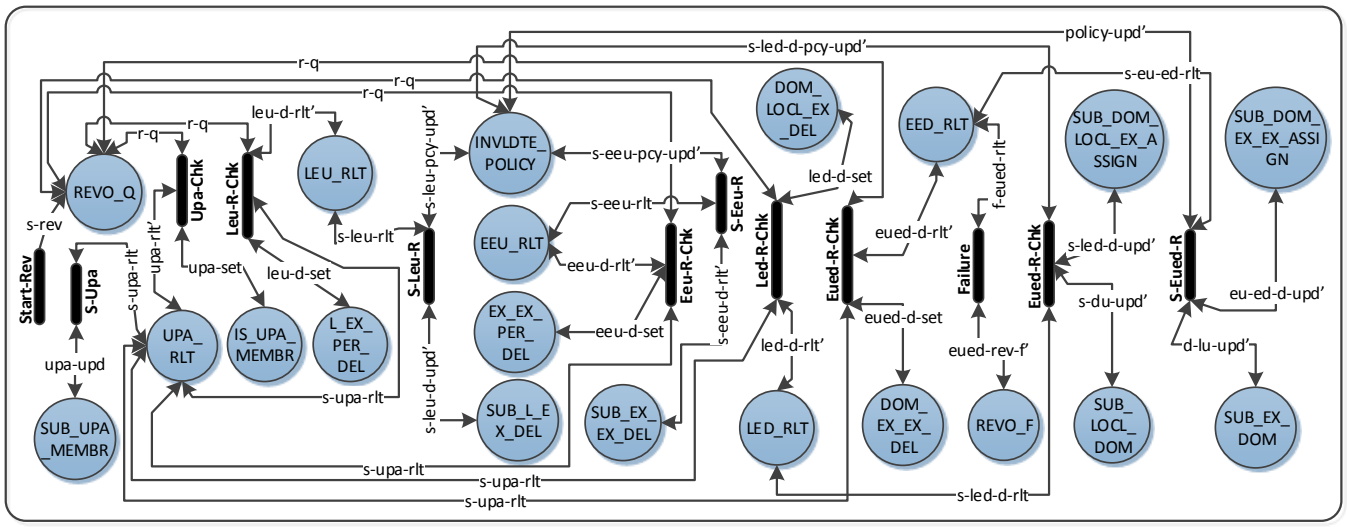

Fig. 7: A HLPN Model for the process of revoking a permission p_i in a cross-tenant environment

vation property holds when the intra-tenant user, cross-tenant and the permission match the intra-user cross-tenant-permission set.

- **(Activation.e) Cross-tenant user to cross-tenant activation property** holds when the cross-tenant user, cross-tenant and the permission match the cross-user-cross-tenant-permission set.

d) Correctness of Activation Algorithm: In the activation algorithm, at the time of activation, an intra-tenant user or a cross-tenant user activates a permission after fulfilling some conditions. To prove the correctness of the activation algorithm, the activation algorithm must fulfill one of the following logical formulas.

- (Activation.a) OR (Activation.b) OR (Activation.d) ***(For an intra-tenant user)***
- (Activation.c) OR (Activation.e) ***(For a cross-tenant user)***

Similarly, we defined the algorithm specific properties for the delegation, forward revocation and the backward revocation algorithms. Although these properties are not presented here, they were considered in the verification process.

e) Results: We transformed each algorithm with their properties to the SMT solver, so that we could verify them. Z3 solver considers both the model and its characteristics and checks the properties of the model fulfill the required level of satisfaction. In the execution of the CTAC model with the stated characteristics in the Z3 solver, the model of CTAC performed well and produced the results as expected. F QF_AUFLIA logic used for closed quantifier-free linear formulas over the theory of integer arrays extended with free sort and function symbols, was also used in implementation under SMT-LIB.

Since our aim is to verify the correctness of the proposed algorithms under the CTAC model, by executing the formal models of the algorithms with the asserted properties in the Z3 solver, we obtained the required results (see Figure 8). The results demonstrate that the CTAC model is able to accomplish the tasks in a finite time (i.e. any cross-tenant request initiated by a user ends up with a result). Moreover, in this case, the execution time indicates that the specific time taken by the Z3 solver in order to measure the satisfiability of the properties. This demonstrated both preciseness and correctness. The execution time consumed by Z3 solver on the respective security property of the algorithms is depicted in Figure 8.

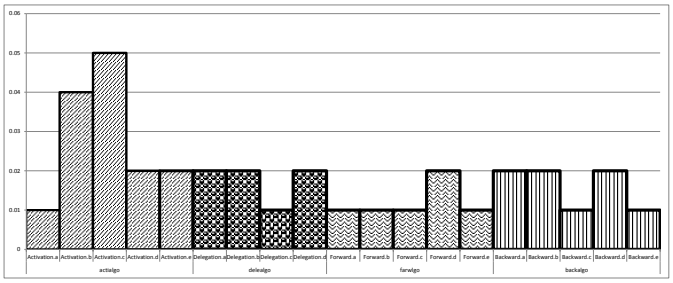

Fig. 8: Verification results of the CTAC model

VII. Conclusion

In this paper, we proposed a cross-tenant cloud resource mediation service (CRMS), which can act as a trusted-third party for fine-grained access control in a cross-tenant environment. For example, users who belong to an intra-tenant cloud can allow other cross-tenant users to activate a permission in their tenant via the CRMS. We also presented a formal model CTAC with four algorithms designed to handle the requests for permission activation. We then modeled the algorithms using HLPN, formally analyzed these algorithms in Z language, and verified them using Z3 Theorem Proving Solver. The results obtained after executing the solver demonstrated that the asserted algorithm specific access control properties were satisfied and allows secure execution of permission activation on the cloud via the CRMS.

Future work will include a comparative analysis of the proposed CTAC model with other state-of-the-art cross domain access control protocols using real-world evaluations. For example, one could implement the protocols in a closed or small scale environment, such as a department within a university. This would allow the researchers to evaluate the performance, and potentially (in)security, of the various approaches under different real-world settings.

REFERENCES

[1] Akhunzada, A., Gani, A., Anuar, N. B., Abdelaziz, A., Khan, M. K., Hayat, A., & Khan, S. U. (2016). Secure and dependable software defined networks. Journal of Network and Computer Applications, 61, 199-221.

[2] Alam, Q., Tabbasum, S., Malik, S., Alam, M., Tanveer, T., Akhunzada, A., Khan, S., Vasilakos, A. and Buyya, R., (2016). Formal Verification of the xDAuth Protocol. IEEE Transactions on Information Forensics and Security, 11(9), pp. 1956-1969.

[3] Ali, M., Malik, S. and Khan, S., DaSCE: Data Security for Cloud Environment with Semi-Trusted Third Party.

[4] Barrett, C., Conway, C.L., Deters, M., Hadarean, L., Jovanovi, D., King, T., Reynolds, A. and Tinelli, C., 2011, July. Cvc4. In International Conference on Computer Aided Verification (pp. 171-177). Springer Berlin Heidelberg.

[5] Bofill, M., Nieuwenhuis, R., Oliveras, A., Rodrguez-Carbonell, E. and Rubio, A., 2008, July. The barcelogic SMT solver. In International Conference on Computer Aided Verification (pp. 294-298). Springer Berlin Heidelberg.

[6] Bruttomesso, R., Cimatti, A., Franzn, A., Griggio, A. and Sebastiani, R., 2008, July. The mathsat 4 smt solver. In International Conference on Computer Aided Verification (pp. 299-303). Springer Berlin Heidelberg.

[7] Choo, K.K., 2006. Refuting security proofs for tripartite key exchange with model checker in planning problem setting. In 19th IEEE Computer Security Foundations Workshop (CSFW'06) (pp. 12-pp). IEEE.

[8] Choo, K.-K. R., Domingo-Ferrer, J. and Zhang, L., 2016. Cloud Cryptography: Theory, Practice and Future Research Directions. Future Generation Computer Systems, 62, pp. 51-53.

[9] De Moura, L. and Bjørner, N., 2011. Satisfiability modulo theories: introduction and applications. Communications of the ACM, 54(9), pp.69-77.

[10] Dutertre, B. and De Moura, L., 2006. The yices smt solver. Tool paper at http://yices. csl. sri. com/tool-paper. pdf, 2(2).

[11] Heiser, J., 2009. What you need to know about cloud computing security and compliance. Gartner, Research, ID, (G00168345).

[12] Jung, T., Li, X. Y., Wan, Z. and Wan, M., 2015. Control Cloud Data Access Privilege and Anonymity With Fully Anonymous Attribute-Based Encryption. IEEE Transactions on Information Forensics and Security, 10(1), (pp. 190-199).

[13] Lin, Y., Malik, S.U., Bilal, K., Yang, Q., Wang, Y. and Khan, S.U., 2016. Designing and Modeling of Covert Channels in Operating Systems. IEEE Transactions on Computers, 65(6), pp.1706-1719.

[14] Liu, J. K., Au, M. H., Huang, X., Lu, R., and Li, J., 2016. Fine-Grained Two-Factor Access Control for Web-Based Cloud Computing Services. IEEE Transactions on Information Forensics and Security, 11(3), (pp. 484-497).

[15] Liu, X., Deng, R. H., Choo, K.-K. R. and Weng, J., 2016. An Efficient Privacy-Preserving Outsourced Calculation Toolkit With Multiple Keys. IEEE Transactions on Information Forensics and Security, 11(11), pp. 2401-2414.

[16] Ma, K., Zhang, W. and Tang, Z., 2014. Toward Fine-grained Data-level Access Control Model for Multi-tenant Applications. International Journal of Grid and Distributed Computing, 7(2), pp.79-88.

[17] Murata, T., 1989. Petri nets: Properties, analysis and applications. Proceedings of the IEEE, 77(4), pp.541-580.

[18] Sayler, A., Keller, E. and Grunwald, D., 2013. Jobber: Automating inter-tenant trust in the cloud. In Presented as part of the 5th USENIX Workshop on Hot Topics in Cloud Computing.

[19] SMT-Lib. http://smtlib.cs.uiowa.edu/, 2015.

[20] Tang, B. and Sandhu, R., 2013, August. Cross-tenant trust models in cloud computing. In Information Reuse and Integration (IRI), 2013 IEEE 14th International Conference on (pp. 129-136). IEEE.

[21] Yang, Y., Zhu, H., Lu, H., Weng, J., Zhang, Y. and Choo, K.-K. R., 2016. Cloud based data sharing with fine-grained proxy re-encryption. Pervasive and Mobile Computing, 28, pp. 122-134.

[22] Zhang, Y., Patwa, F., Sandhu, R. and Tang, B., 2015, August. Hierarchical secure information and resource sharing in openstack community cloud. In Information Reuse and Integration (IRI), 2015 IEEE International Conference on (pp. 419-426). IEEE.

[23] Zhao, G., Ba, Z., Wang, X., Zhang, Y., Huang, C. and Tang Y., 2016. Constructing Authentication Web in Cloud Computing. Security and Communication Networks, 9(15), pp. 2843-2860.

Q.Alam is currently undertaking a Ph.D. in Information Security and Formal Methods. Her research interests include cross domain access control frameworks and formal verification.

Saif U. R. Malik received his Ph.D. from North Dakota State University, USA. He is currently an Assistant Professor in the Department of Computer Sciences at COMSATS Institute of IT, Islamabad, Pakistan. His research interests include the application of formal methods in large scale computing systems, software engineering, and security protocols.

Adnan Akhunzada received the Ph.D. in Network Security in 2016 from University of Malaya, Malaysia. Currently, he is working as an Assistant Professor in the Department of Computer Sciences at COMSATS Institute of IT, Islamabad, Pakistan. His research interests include secure and dependable software defined networks (SDNs), lightweight cryptography, attacker attribution and profiling, remote data auditing, and modeling and designing cross-domain access control protocols.

Kim-Kwang Raymond Choo received the Ph.D. in Information Security in 2006 from Queensland University of Technology, Australia. He currently holds the Cloud Technology Endowed Professorship at The University of Texas at San Antonio. He is the recipient of various awards including ESORICS 2015 Best Paper Award, Winning Team of the Germany's University of Erlangen-Nuremberg (FAU) Digital Forensics Research Challenge 2015, and 2014 Highly Commended Award by the Australia New Zealand Policing Advisory Agency, Fulbright Scholarship in 2009, 2008 Australia Day Achievement Medallion, and British Computer Society's Wilkes Award in 2008. He is also a Fellow of the Australian Computer Society, and a Senior Member of IEEE.

S.Tabbasum is currently undertaking a Masters degree in Software Engineering. Her research interests include software engineering and formal verification.

M.Alam received the Ph.D. in Computer Sciences from University of Innsbruck, Austria. He is currently an Associate Professor in the Department of Computer Sciences at COMSATS Institute of IT, Islamabad, Pakistan. His interests include access control systems, model driven architecture, and work flow management systems.