Toward a Dynamic Trust Establishment Approach for Multi-provider Intercloud Environment

Canh Ngo, Yuri Demchenko, Cees de Laat
University of Amsterdam, Netherlands
Email:{t.c.ngo,y.demchenko,delaat}@uva.nl

***Abstract*—In Cloud computing, the data are not only managed by the data owner but also by Cloud providers. Sophisticated Clouds collaboration scenarios require that these data objects can be accessed distributively among Cloud providers, while still being under the control of data owners. It brings security challenges for distributed authorization and trust management in which existing proposed schemes have not fully solved. In this paper, we propose a Dynamic Trust Establishment approach which can incorporate into Cloud provisioning life-cycles for the multi-provider Intercloud environment. It relies on attribute-based policies as the mechanism for trust evaluation and delegation. The paper also presents a practical implementation approach for attribute-based policies using Multi-type Interval Decision Diagrams which has advantage in term of evaluation complexity.**

***Keywords*-Dynamic trust establishment; distributed authorization; trust delegation; attribute-based policy.**

I. Introduction

Based on principles of Clouds suggested by NIST [1], there's a tendency that Cloud providers will cooperate to bring composite Cloud services to customers. Such collaboration between providers forms daisy-chain Cloud services, in which providers in the chain leverage their services from preceding one. The Fig. 1 can illustrate this scenario. In this figure, an Infrastructure as a Service (IaaS) Cloud provider can aggregate individual virtualized resources from different Physical Providers to build up virtual infrastructures consisting of virtual computing nodes, virtual storages, reserved network capacities, etc. The Platform as a Service (PaaS) providers then may subscribe these infrastructures to run their own platform development which offers services to Software as a Service (SaaS) Providers. These Cloud providers can collaborate in a daisy-chain to form composited Cloud services distributed among Clouds to end-users. Prospective development for Cloud Computing, known as Intercloud as in [2], is that Cloud providers not only leverage their Cloud services from others vertically, but also can cooperate their Cloud services horizontally to obtain reliability, scalability and cost efficiency.

In Fig. 1 with multiple Cloud providers, the end-user, say Alice, has a business workflow that connect her subscribed services from providers $SaaS_1$ to a software system running on a virtual infrastructure provided by $IaaS_1$ and another service from $SaaS_2$ and so on. The lowest layer is the Physical Providers Layer, which consists of Physical Infrastructure Providers (PIPs) who physically own physical devices and offer virtualized primitive resources such as storage, computing and network. It requires interconnections between Cloud providers, even when they may not have any direct relationships such as subscription contracts or Service Level Agreements (SLAs): by vertically such as $PIP_2 - IaaS_1 - PaaS_1 - SaaS_1$, or by horizontally such as $SaaS1 - SaaS_2$, $SaaS_1 - IaaS_1 - PaaS_1$, etc.

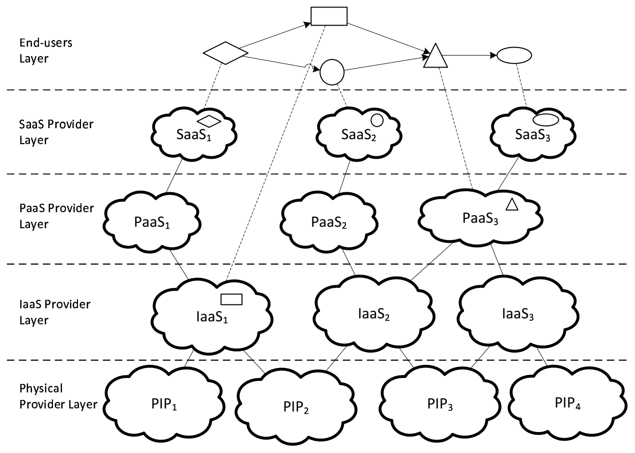

Figure 1. An Intercloud scenario

This scenario brings security challenges in the aspect of authorization and trust management. How a physical provider, say PIP_2, allows Alice to access its data when there's no direct relationship between her, a stranger, and the PIP_2. There's only have indirect relationships between them through intermediate providers, e.g. $SaaS_2 - PaaS_1 - IaaS_1 - PIP_2$. In the above workflow, services running $PaaS_3$ wants to access storage services provided by $IaaS_1$ on behave of the owner Alice. The challenges here are how to provide an effective, robust authorization mechanism in which an entities endow another strange entities to access its data in a distributed environment. Basic authorization mechanisms do not fit this situation, when they require all entities' identifiers are known by authorizer. In a distributed, open and dynamic environment with multiple administrative domains, it is challenge for authorization based on users' identities, since it requires a federated identity management system accessible by corresponding Clouds.

This motivates the application of trust management for authorization purpose in which the authorizer can decide permissions based on principle's attributes that are distributed at different locations and does not need principles' identities. Moreover, the data/resource ownership transferable property in the Clouds demands the distributed authorization system using trust management should provide chain of delegation in multiple levels. In this paper, we propose a dynamic trust establishment with distributed authorization in multi-provider Intercloud environment. The contributions are as follows. First, our proposed scheme, based on the basic attribute-based access control model (ABAC), presents an attribute-based trust model approach proved by logic formulas. The trust model is then applied to propose the dynamic trust establishment mechanism which is the part of a Dynamic Access Control Infrastructure for On-demand provisioned Clouds [3]. After all, we propose a practical implementation of the attribute-based trust by using Multi-type Intervals Decision Diagrams which has substantial performance comparing to other basic ABAC implementations.

The paper is organized as follows. Section II describes the trust model for entities in the Intercloud. It also introduces attribute-based trust policy concepts as mechanisms for applying trust model to distributed authorization in Intercloud. In section III, we incorporate the trust model into Cloud provisioning life-cycles to provide dynamic trust establishments among entities in the Intercloud. Section IV presents adopted method to implement the attribute-based policies evaluation in section II with Multi-types Interval Decision Diagram. Section V describes the state of the art of existing trust management mechanisms for distributed authorization and identifies their limitations when applying to architectures of Cloud echo system. Finally, section VI summarizes work done and points out our future works in trust management, as a part of a security infrastructure for Clouds.

II. Trust Model

This section analyzes trust model in the multi-provider Intercloud environment, which includes entities, trust relationships and approaches to establish trust relationships.

A. Definitions

1) Entities: A sophisticated Cloud scenario may involve number of different entities, which can be categorized into one or several below types.

- Cloud providers and Cloud clients: When entities in the Cloud eco-system can offer services to others, they are called Cloud providers. The subscribers are called Cloud clients. Some entities may have two roles, both provider role and client role.
- End-users: End-users are the last endpoint in the chain of Clouds. They can be employees using cloud services of a company that subscribing IaaS from an IaaS Cloud providers. Or they can be any individual users using services from an Cloud providers.
- Physical Cloud providers: they are entities that hold physical resources such as storage, computing, connectivity, etc. By applying various virtualization technologies, they can split their physical resources and platforms into slices of virtual resources, e.g. virtual machines (VMs), cloud storage, virtual networks, etc. These virtualized resources can be consumed by other entities. In the chain of Cloud providers, Physical Cloud providers stand at the beginning. This type of Cloud providers can be seen as Amazon with EC2, S3 services.
- Intermediate Cloud providers: are providers who must not own physical facilities, but by subscribing virtualized resources from Physical Cloud providers, they can build services on top these resources and offer new products to customers. When the service is a complete Cloud Infrastructure including storage, computing and network connectivity, we call these providers as IaaS Cloud providers. If services are development and runtime environments for developers, these providers are called PaaS Cloud providers. If services are software applications then they are SaaS Cloud providers. In practical, some Cloud providers may have several roles, such as Google with Google Drive service and Google Apps services; Microsoft with their own physical facilities to provide Azure platform as PaaS and Office365 application services as SaaS. A Cloud image processing provider consuming stored images at a storage providers can be seen as an intermediate Cloud provider.

2) Trust: In the context of authorization for Clouds, we define trust of an entity (trustor) to another entity (trustee) as the belief of trustor on the trustee that the trustee can behave reliably, dependably and securely in some specific contexts. It can be seen that trust is the basis for authorization, an entity only grant permission on another one only if it trusts the other, not for everything, all the time, but on a specific situation and limited time, or a specific context. For example, entity A is an expert in finance, entity B who trusts A's capability, will take consideration from A's comments on finance issues, but may not listen to A's ideas on other, such as medical.

This example mentions an important feature, that is how the trust is established. In the above example, we can assume after seeing A's certificates on finance, or A's experience history, B will trust A on finance area. In most abstraction, the trustor trust the trustee on a specific context when the trustee can show enough his attributes that satisfy trustor's criteria, and the trustor, by some mechanisms, makes sure that these attributes are validated. This is the basic formation of attribute-based trust establishment.

3) Trust relationships: The trust relationships in Cloud Computing has following properties:

- Asymmetric: the relationship has direction, which is A trusts B does not mean B trusts A.
- Contextual: the trust often specify on a particular context. For example, A trusts B as a Cloud provider providing network service, but not on storage service.
- Time-constraint: The trust should have limited lifetime. In Cloud Computing, the lifetime of the trust between Cloud providers and Cloud clients could be depend on subscription contracts to provide services between them.

We classify trust relationships into two following types, depending on their lifetimes and how they are established:

- Direct Trust relationship: is the trust relationship between a Cloud provider and its direct clients. It is a bilateral relationship in a long-term period which is based on subscription contracts and usually is enforced by SLAs.
- Indirect Trust relationship: is the trust relationship between a Cloud provider and client through one or several intermediate Cloud providers. This is a dynamic, ad-hoc trust relationship forming during service consumption in a short time (compare to the lifetime in SLAs). This relationship is often established based on existing several direct trust relationships.

B. Trust Policies

In the previous section, we defines trusts and related trust relationships. However, the model needs mechanisms to decide the trust in different situations. In the multi-providers of Intercloud environment, we propose to use attributes as the primitive data to evaluate trust.

1) Basic trust policy: The trust between two entities should have a specific semantic meaning, or trust context. The trust statement between two entities is defined as: ”Alice trusts Bob on context X”. Alice is called the actor of trust, Bob is the target of trust and the trust relationship is limited by context X.

When the context can be described by attributes, the trust statement can be formulated as a set of logic conditions on attributes which combined by Boolean operators *and* ($\wedge$), *or*($\vee$), *not*($\neg$) and the context X is defined as a vector consisting of n attributes $X = (x_1, x_2...x_n)$, each attribute x_i in the context has their domain values P_i.

The trust statement is analogized as logic conditions expression over the vector variable X and the actor of context. We call it as the attribute-based trust policy:

$$f_{actor}(target, X) \rightarrow trust \tag{1}$$

In the trust policy (1), t is the target of the context X, and a is the actor of the following trust statement: The $actor$ trusts the $target$ on context X.

For example, Alice store her personal finance data in a Cloud storage services provided by P_1. She asks Bob, a finance consultant, to analyze her finance status and give advices. Thus, Alice grants Bob to read her finance data in the P_1. For simplification, we denote this context X as $X^* \equiv (data_{Alice}, read)$. The trust policy defined at P_1 is:

$$f_{Alice}(Bob, (data_{Alice}, read)) \rightarrow trust$$

Now Bob wants to use a finance expert system provided by a Cloud provider P_2 in which he subscribes this service. The outcome of this expert service then is analyzed by Bob's to produce report for his customer, Alice. The scenario here becomes complicated when Bob should have permission to allow services in P_2 can access Alice's data at P_1. In other words, Bob wants to delegate his permission to P_2 system. In next section, we define policies for such scenario.

2) Policies for delegation: The indirect trust relationship in the model is based on the concept of conditional trust transitivity. In this concept, entity C may trust A by an indirect trust relationship when existing an intermediate entity B, plays as the trust recommender. They need to satisfy following conditions:

- The recommender B trusts A and recommends it to C
- The trustor C trusts B as the recommender.
- On receiving recommendation from B, C will count it in the trust evaluation of A.

In attribute-based trust model, these conditions are described as follows:

- B trusts A based on B's policy on context X_A, then B issues a recommendation in the form of a trust credential tc_B.
- C asserts that B is a legit recommender for the trust context X_B.
- With recommendation tc_B and the context X_A, C uses a recommendation policy to decide if it can trust A.

These conditions are described by following notations:

- Attribute issuing policy for B: it's similar like trust policy, but the result is an issued credential as the approval of B for context X.

  $$f_B(A, X_A) \rightarrow tc_B^{X_A} \tag{2}$$

  in which $tc_B^{X_A}$ is the recommendation of B on the context X_A.
- Delegation policy for C: it defines set of targets which are eligible as recommenders for context X.

  $$f_C^D(X) \rightarrow \{targets\} \tag{3}$$

- C will evaluate the recommendation of B by using below conditions:

$$f^R_C(tc^{X_A}_B, X_A) = (B \in f^D_C(X_A)) \wedge valid(tc^{X_A}_B, X_A) \rightarrow trust \quad (4)$$

The recommendation $tc^X_{recommender}$ is an attribute issued by the recommender. It can be shown as a trust credential exchanging between entities. The implementation of this trust credential should guarantee the authenticity of the recommender that issuing the attribute, and the integrity of the trust context that it conveys. The function $valid$ in the above policy has the purpose to check the integrity of trust certificate against the context. In the section III, we propose a scheme that provides these properties.

Return with previous example, the attribute issuing policy of Bob to grant permission for P_2 to access Alice's data on behaving of him:

$$f_{Bob}(P_2, X^*) \rightarrow tc^{X^*}_{Bob}$$

Alice has a delegation policy:

$$f^D_{Alice}(X^*) \rightarrow \{Bob\}$$

The trust of Alice to P_2 is setup when following condition is fulfilled:

$$f^C_{Alice}(tc_{Bob}, X^*) := (Bob \in f^D_{Alice}(X^*)) \wedge valid(tc_{Bob}, X^*) = true$$

3) Delegation trust chain: When Cloud resources are composed from stack of Cloud providers, for example a SaaS provider P_1 subscribed PaaS from P_2. In turn P_2 runs on a virtual infrastructure provided by IaaS provider P_3. In this situation, providers P_1, P_2 and P_3 need to use policies for delegation to establish a trust between endpoints of the chain.

In general, given a Cloud supply chain of providers $P_1 \rightarrow P_2 ... \rightarrow P_k$ where the provider P_i subscribes Cloud resources from provider P_{i-1} as in Fig. 2.

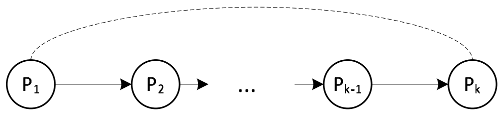

Figure 2. Chain of Cloud providers

The trust chain from P_1 to P_k is done when delegations are setup along the path, which is:

$$\bigwedge_{i=k-1}^{1} (P_{i+1} \in f^D_{P_i}(X)) \wedge valid_{P_i}(tc^X_{P_{i+1}}, X) = true \quad (5)$$

III. Dynamic Trust Establishment Mechanism for Multi-level Clouds

This section will analyze challenges on building up a trust establishment protocol for multi-provider Intercloud environment. Our approach shows that it can be solved by using attribute-based trust polices in section II in applying to Cloud provisioning life-cycles.

A. Challenges

1) Distributed of policies and attributes: In practical of Intercloud, trust policies are distributed and under controls and configurations of different entities with their own security domains. To evaluate the formula (5), we need to collect decisions and attributes from these entities. We propose to use the Pull and Push sequences as in [4] for distributed policy evaluation. The Pull sequence is illustrated in the section III-B. The remain Push sequence is then deduced as well.

2) Local name spaces: Each Cloud provider has its own name space, so understanding attribute-based context crossing domains is a challenge. It's obvious that with the direct relationship, the Cloud client knows the direct Cloud provider name space, because they have SLAs: the Cloud client receives the Cloud provider's resource ontology after SLA negotiation, this ontology then is to describe a trust context X which is understandable by the provider.

To overcome this challenge, we suggest applying semantic techniques to transform contexts between name spaces. We assume that Cloud providers has their own ontologies to describe their resources and attribute profiles, e.g. Infrastructure and Network Description Language (INDL) [5] for virtual infrastructures provisioning. When Cloud clients consume Cloud resources from a Cloud provider P, they are provided the provider's ontology, say O_P. When a Cloud client C_1 plays as the intermediate Cloud provider by offering its Cloud resources, it also has its ontology, say O_{C_1}. By default, a request context X comes from end-user U to C_1 is described based on concepts of O_{C_1}. Due to some Cloud computing operations requirements, P may need to communicate directly with U. In this case, C_1 needs to provide a semantic inference engine that can infer concepts between ontologies O_P and O_{C_1}, used to transform the request context of end-user U before sending to P. This is an open research direction on the Cloud semantics which its results can be applied into distributed authorization for Intercloud.

3) Dynamic trusts relationships: Direct trust and indirect trust relationships in the model are not static and cannot be implemented using trusted certificate list (TAL) mechanisms as in Public-key Infrastructures or PGP systems. These two relationships has their lifetimes binding with Cloud resources. The direct trust relationships with their trust anchors are established in the Cloud services provisioning phase and terminated at the end of Cloud services' lifetimes. The indirect trust relationships are formed during the operation phase of the Cloud resources, when Cloud clients, through daisy-chain of intermediate Cloud providers, want to access the Cloud resources from the original Cloud providers.

We propose a dynamic trust establishment mechanism, in which the direct trust relationships are provisioned during Cloud resources provisioning which was the part of the architecture in [3]. The indirect trust relationships are established by the below distributed trust chain discovery by using attribute-based trust policies for enforcement.

B. Dynamic Trust Establishments

The direct trust relationship between Cloud client and its Cloud provider are setup during the deployment phase of the Cloud security services provisioning life-cycles in [6]. The establishment workflow is described as below:

Context:

- A Cloud client C wants to subscribe Cloud resources from the Cloud provider P.
- P has a Trust Management repository for storing subscribed client identifier (GRI) along with its related security parameters: public key, policies (including trust policies, delegation policies and recommendation-based trust policies) that binds with GRI.

Implementation:

- In the reservation phase, Cloud client C and Cloud provider P exchange security parameters:
    1) C generate a pair of its public key PK_C and equivalent secret key SK_C. The PK is then sent to Cloud provider P.
    2) Based on negotiated SLA, P creates a subscribed resources description following its Cloud resource ontology definition. It then generates attribute-based policies in which the trust context is derived from the subscribed resource description, e.g. in the space of virtual infrastructure provisioning, it can be done by using INDL [5]. P also initializes the delegation trust policy for the subscribed client C. All these information is stored as a security parameters for C, which is indexed by the GRI value in the repository.
    3) After the deployment phase, Cloud client C holds its secret key SK_C, public key of the provider PK_P, the subscribed identifier GRI and description of subscribed resources. The Cloud provider P holds PK_C, trust policies and delegation policies.
- In the operation phase, the trust relationships between C and P are set up depending on resource requests coming to P:
    1) If a specific request X comes directly from the Cloud client C, signed by SK_C, the Cloud provider P will validate its origin by the PK_C stored in the security parameter store and perform evaluating its context against trust policies. If the decision is $trust$, then P allows C to operate on resource described in X. This is the direct trust relationship from C to P.
    2) If a request X comes from an external entity, say E, recommended by C, the interaction between these entities are illustrated in Fig. 3. In this case, the indirect trust relationship between E and P is established based on policies managed by C.
    3) For a chain of Cloud providers, the Pull model can be applied as in Fig. 4. In this figure, trust contexts X_i are transformed between Cloud name spaces; the trust credential tc_i denotes the recommendation of the provider P_i for the context X_i to the successor provider.
- In the Decommissioning phase, related parameters and policies binding to GRI of C is released from the repository, along with subscribed Cloud resources.

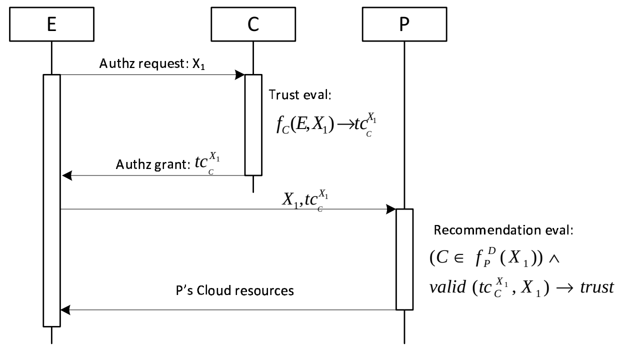
E
C
P
Authz request: X_1
Trust eval:
$f_C(E,X_1) \rightarrow tc_C^{X_1}$
Authz grant: $tc_C^{X_1}$
$X_1, tc_C^{X_1}$
Recommendation eval:
$(C \in f_P^D(X_1)) \wedge$
$valid\ (tc_C^{X_1}, X_1) \rightarrow trust$
P's Cloud resources

Figure 3. Indirect Trust Establishment Protocol Flow

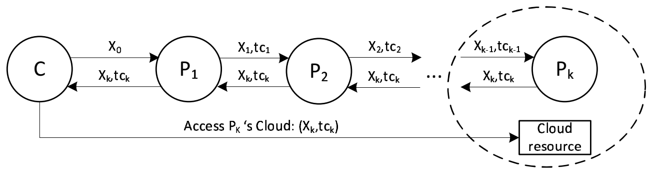

Figure 4. Indirect Trust Establishment Protocol Flow with Pull Model

C. Trust Credential Validation

As mentioned in section II, the trust credential $tc^X_{Recommender}$ should have authenticity of the recommender, context X integrity and limited lifetime. These requirements can be implemented by using cryptographic techniques as follows:

$$\begin{aligned} tc^X_{Recommender} &:= (issuer, s_X, t_X); \\ issuer &:= Recommender; \\ s_X &:= sign(SK, H(X)|t_X); \end{aligned}$$

in which $H(X)$ is the one-way hash function of the trust context and t_X is the lifetime value of the recommendation. SK is the secret key of the recommender in which its public key is in security parameters store of the target provider. The signature is to protect integrity of the context's content and validity of trust credential's lifetime.

The implementation of the trust credential can utilize SAML standard [7] by deriving the SAML *TrustStatement* from the SAML abstract statement.

```
<?xml version="1.0" encoding="UTF-8"?>
<saml:Assertion>
  <saml:Issuer>trustAuthority-P1</saml:Issuer
      >
  <saml:Subject>
    <saml:NameID>u1@companyC</saml:NameID>
    <saml:SubjectConfirmation Method="
        urn:oasis:names:tc:SAML:2.0:cm:sender-
        vouches"/>
  </saml:Subject>
  <saml:Conditions>
    <saml:AudienceRestriction>
      <saml:Audience>IaaS-P2</saml:Audience>
    </saml:AudienceRestriction>
  </saml:Conditions>
  <saml:TrustStatement TrustInstant="
      2012-06-21T16:11:41.392Z"
      SessionNotOnOrAfter="2012-06-24
      T16:11:41.392Z">
    <saml:TrustContext xmlns:indl="
        urn:names:IaaS-P2:ontologies:indl">
      <!-- trust context in XML format is
          inserted or referred here-->
    </saml:TrustContext>
    <dsig:Signature xmlns="http://www.w3.org
        /2000/09/xmldsig#">
      <!-- a XML signature to protect trust-
          context integrity-->
    </dsig:Signature>
  </saml:TrustStatement>
</saml:Assertion>
```

IV. Attribute-based Policies Implementation

In this section, we present the Multi-type Interval Decision Diagram which are extended from the Interval Decision Diagram in [8]. It represents a multi-variable logic function as an acyclic, direct graph which is practical for implementation of attribute-based policy model in section II.

A. *Multi-type Interval Decision Diagrams*

The policies in section II can be seen as a multivalued function with signature:

$$f : P_1 \times P_2 \ldots \times P_n \to R \tag{6}$$

Let a data interval $I \subset P_i$ is a range of values in the domain P_i. Define a Boolean function $h_{x_i}(I)$ as:

$$h_{x_i}(I) = \begin{cases} 0 & \text{if } x_i \notin I \\ 1 & \text{if } x_i \in I \end{cases}$$

Function f in (6) is called independent with a variable x_i in the interval I when:

$$\forall x_i^1, x_i^2 \in I : f_{x_i^1} = f_{x_i^2}$$

We denote this function as $f_{x_i^I}$.

Set of interval $I(P_i) = \{I_1, I_2, ..., I_{P_i}\}$ is called cover the domain set P_i of the variable x_i when:

$$P_i = \bigcup_{I \in I(P_i)} I$$

The cover $I(P_i)$ is disjoint if:

$$\forall i, j \in [1, p_i], i \neq j : I_i \cap I_j = \emptyset$$

According to Boole-Shannon expansion, a function f can be decomposed to several partial functions in respect of variable x_i against a disjoint, covered partition $I(P_i)$

$$f(X) = \bigvee_{I \in I(P_i)} h_{x_i}(I) \wedge f_{x_i^I} \tag{7}$$

Each partial function $f_{x_i^I}$ which is independent with variable x_i, can also be decomposed in respect to other variable x_j. The decomposition continues until the function is free from all variables. Then we can symbolize function f as a decision diagram $G(V, E)$ that:

- G is a rooted, directed acyclic graph with a node set V having two types of nodes: terminal node and non-terminal node.
- A terminal node $v \in V$ has the value $r \in R$.
- A non-terminal node $v \in V$ is a variable $x_i \in P_i$ of the function f in which its disjoint, covered data interval partition is $I(P_i) = \{I_1, I_2, ..., I_{P_i}\}$. Each interval $I \in I(P_i)$ is equivalent to an out going edge $e_v^I \in E$ from the node v.
- The sub graph of the outgoing edge of non-terminal node $v \in V$ is a partial function described by Boole-Shannon expansion in equation (7).

The formula (7) then can be represented as a decision diagram in following figure:

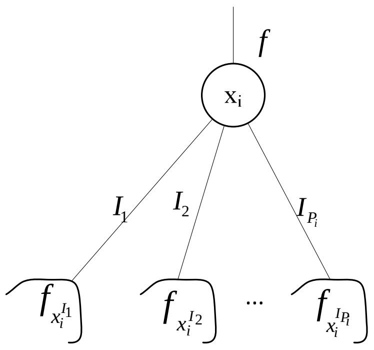

Figure 5. Decision diagram illustration for logic function decomposition

An example of a decision diagram is shown in Fig. 6. It represents following policy:

$$f(x_1, x_2, x_3) = \begin{cases} trust \quad \text{if}((x_1 \in \{Bob, Carol\} \wedge \\ \quad (x_2 = report_1) \wedge \\ \quad (x_3 \in read, write)) \vee \\ \quad ((x_1 = Dave) \wedge \\ \quad ((x_2 = report_2) \vee \\ \quad ((x_2 = report_1) \wedge \\ \quad (x_3 = read)))) \\ distrust \quad \text{otherwise} \end{cases}$$

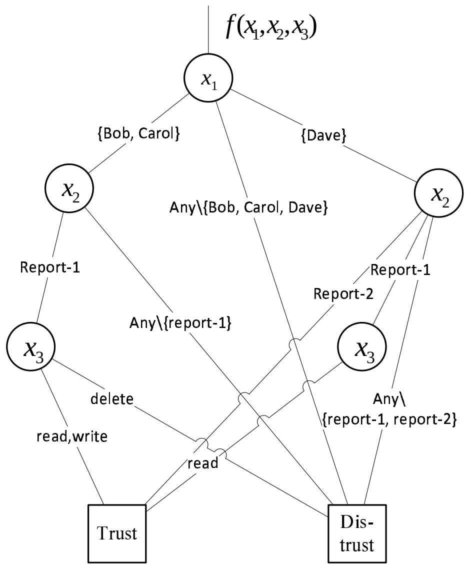

Figure 6. A Multi-type Interval Decision Diagram example

B. Authorization Complexity

In this section, we estimate the complexity of delegated trust chain in (5).

Any attribute-based policy with n attributes in section II can be transformed into a n-level MIDD described in 6. The evaluation complexity of this policy is $n \cdot \log(m_P)$ with $m = \max(P_i); \forall i \in [1, n]$, that is the maximal number of intervals for all attribute x_i.

Given a set of k multi-level Cloud providers which form the supply chain for composite Cloud resources, the complexity, the complexity of the delegation trust chain establishment in (5) will be $k \cdot n \cdot \log(m_P)$. In theory, this complexity does not depend on number of policies but only on number of attributes, the number of data intervals and the length of delegation chain. However, practice shows that increasing number of policies also affect the number of data interval, because each policy often defines different intervals.

One drawback of the MIDD is the memory space, which is depend on number of nodes in the diagram. A n-level MIDD with on average $\overline{m}$ edges of each node could cost up to $n^{\overline{m}} \cdot sizeof(node)$ memory. However, the number of nodes in MIDD depends heavily on whether the logical function in (6) is optimized or not [8]. So it's possible to apply different implementation techniques in to mitigate this problem by optimizing logical functions and using *unique table* implementation in [9].

V. Related Work

The problem of trust management for authorization in distributed, decentralized environment was initially investigated by Blaze et al. [10]. Subsequent work represented Datalog trust policy languages by Li and Michell [11] and then Role-based trust management language [12], in which trust policies map subjects to roles based on attributes in their credentials, then decisions were given from roles. Because of distributed properties of attributes in decentralized environment, they developed a credential chain discovery algorithm to retrieve and collect credentials. Such these algorithms belonged to trust negotiation process aware of privacy of sensitive attribute information such as automated trust negotiation of Li et al. [13] or the Privacy-aware role-based access control framework by Ni et al. [14]. These approaches has some difficult applying to Intercloud when they do not have efficient mechanisms to deal with local name spaces issue. Our direction, in other hand, uses attributes as the primitives for trust evaluation, which can be transformed among Clouds by semantic techniques to transform attributes between ontologies of local name spaces.

OAuth 2.0 authorization framework [15] enables a third-party to access data by HTTP service on approval of the data owner by an HTTP service. It provides workflow protocol for distributed authorization that is currently applied in various Cloud-based services such as Google API [16]. However, OAuth authorization framework does not mention authorization evaluation mechanisms and how to dynamically setup trust anchors for provisioned Cloud resources as well as establishment trusts through chain of entities.

VI. Conclusion

In this paper, we have identified challenges of trust management regarding distributed authorization for multi-provider Intercloud environment. We then propose a formal trust model that use attribute-based trust policies. The proposed model is also applied in Cloud provisioning life-cycles to provide the dynamic trust establishment mechanism. Furthermore, we presents a practical implementation of the attribute-based trust policies evaluation by using Multi-type Intervals Decision Diagrams which has substantial performance comparing to other basic ABAC implementations.

In future, for attribute resolutions among Clouds' name spaces, we plan to apply semantic techniques to transform attributes ontologies from Cloud to Cloud. For integration of proposed protocol to the Dynamic Access Control Infrastructure, we are developing the attribute-based trust policy mechanisms engine using MIDD as the back-end, while any attribute-base policy languages such as XACML [17] can be used at front-end for administration. The dynamic trust establishment protocol implementation should support existing standards such as OAuth [16] and SAML [7] to communicate between Cloud providers.

REFERENCES

[1] Hogan, M. D, Liu, F., Sokol, A. W., and Jin, T., "NIST-SP 500-291, NIST cloud computing standards roadmap," NIST, Tech. Rep. NIST SP 500-291, 2011.

[2] Y. Demchenko, C. Ngo, M. Makkes, R. Strijkers, and C. de Laat, "Defining inter-cloud architecture for interoperability and integration," in *Proceedings of the 3rd International Conference on Cloud Computing, GRIDs, and Virtualization*, ser. CLOUD COMPUTING 2012. IARIA, 2012, pp. 174–180.

[3] C. Ngo, P. Membrey, Y. Demchenko, and C. de Laat, "Security framework for virtualised infrastructure services provisioned on-demand," in *Cloud Computing Technology and Science (CloudCom), 2011 IEEE Third International Conference on*, 29 2011-dec. 1 2011, pp. 698 –704.

[4] J. Vollbrecht, P. Calhoun, S. Farrell, L. Gommans, G. Gross, B. de Bruijn, C. de Laat, M. Holdrege, and D. Spence, "RFC 2904 - AAA authorization framework," Tech. Rep. RFC 2094, aug 2000. [Online]. Available: http://tools.ietf.org/html/rfc2904

[5] M. Ghijsen, J. van der Ham, P. Grosso, and C. de Laat, "Towards an infrastructure description language for modeling computing infrastructures," in *Parallel and Distributed Processing with Applications (ISPA), 2012 IEEE 10th International Symposium on*, july 2012.

[6] Y. Demchenko, C. Ngo, C. de Laat, T. Wlodarczyk, C. Rong, and W. Ziegler, "Security infrastructure for on-demand provisioned cloud infrastructure services," in *Cloud Computing Technology and Science (CloudCom), 2011 IEEE Third International Conference on*, 29 2011-dec. 1 2011, pp. 255 –263.

[7] "Assertions and protocols for the OASIS security assertion markup language (SAML) v2.0, OASIS standard," OASIS, SAML, Mar. 2005. [Online]. Available: http://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf

[8] K. Strehl and L. Thiele, "Interval diagrams for efficient symbolic verification of process networks," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 19, no. 8, pp. 939 –956, aug 2000.

[9] K. S. Brace, R. L. Rudell, and R. E. Bryant, "Efficient implementation of a bdd package," in *Proceedings of the 27th ACM/IEEE Design Automation Conference*, ser. DAC '90. New York, NY, USA: ACM, 1990, pp. 40–45. [Online]. Available: http://doi.acm.org/10.1145/123186.123222

[10] M. Blaze, J. Feigenbaum, and J. Lacy, "Decentralized trust management," in *In Proceedings of the 1996 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, 1996, pp. 164–173.

[11] N. Li and J. C. Mitchell, "Datalog with constraints: A foundation for trust management languages," in *Proceedings of the 5th International Symposium on Practical Aspects of Declarative Languages*, ser. PADL '03. London, UK, UK: Springer-Verlag, 2003, pp. 58–73. [Online]. Available: http://dl.acm.org/citation.cfm?id=645773.667961

[12] N. Li, J. C. Mitchell, and W. H. Winsborough, "Design of a role-based trust-management framework," in *Proceedings of the 2002 IEEE Symposium on Security and Privacy*, ser. SP '02. Washington, DC, USA: IEEE Computer Society, 2002, pp. 114–. [Online]. Available: http://dl.acm.org/citation.cfm?id=829514.830539

[13] J. Li, N. Li, and W. H. Winsborough, "Automated trust negotiation using cryptographic credentials," in *Proceedings of the 12th ACM conference on Computer and communications security*, ser. CCS '05. New York, NY, USA: ACM, 2005, pp. 46–57. [Online]. Available: http://doi.acm.org/10.1145/1102120.1102129

[14] Q. Ni, A. Trombetta, E. Bertino, and J. Lobo, "Privacy-aware role based access control," in *Proceedings of the 12th ACM symposium on Access control models and technologies*, ser. SACMAT '07. New York, NY, USA: ACM, 2007, pp. 41–50. [Online]. Available: http://doi.acm.org/10.1145/1266840.1266848

[15] E. D. Hardt and D. Recordon, "The oauth 2.0 authorization framework, draft-ietf-oauth-v2-30," Tech. Rep., July 2012. [Online]. Available: http://tools.ietf.org/html/draft-ietf-oauth-v2

[16] *Using OAuth 2.0 to Access Google APIs*. [Online]. Available: https://developers.google.com/accounts/docs/OAuth2

[17] OASIS, "XACML v3.0: Core specification," OASIS, Tech. Rep., Aug. 2010. [Online]. Available: http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-cs-01-en.pdf