Tracking and Constraining Authorization Provenance

Jinwei Hu[1], Khaled M. Khan[1], Yun Bai[2], and Yan Zhang[2]

[1] Department of Computer Science and Engineering, Qatar University, Qatar
[2] School of Computing and Mathematics University of Western Sydney, Australia
{jinwei, k.khan}@qu.edu.qa {ybai, yan}@scm.uws.edu.au

Abstract. Authorization provenance concerns how an authorization is derived. It appears important to define authorization provenance to (1) analyze policy bases, (2) defend against a class of attacks, and (3) audit authorizations. In this paper, we study a notion of authorization provenance, based on a recently proposed logic in the literature. By examining a collection of properties, we show this definition captures the intuitions of authorization provenance. We also present an application of our notion of authorization provenance: specifying and enforcing a new type of security requirements.

1 Introduction

Authorization provenance is information about how an authorization is derived. Modelling authorization provenance is challenging in decentralized environments, as mechanisms like delegation make authorization provenance complex. Besides the resource guard who controls access to resources, other agents (e.g., delegatees) play a role in authorization decision-making as well.

Suppose for example that Alice is the warden of a building and that the request to access the building would be allowed only if "Alice believes $access$" can be proved. Consider the following cases:

CASE1	Alice believes $access$
CASE2	Alice trusts Bob on $access$ Bob trusts Cathy on $access$ Cathy believes $access$

Observe that Alice's belief in $access$ is concluded in different ways. In CASE1, it is because Alice herself, whereas Bob and Cathy also have an effect in CASE2.

Among others, provenance information helps enforce and analyze security. Putting restrictions on authorization provenance may prevent insiders' misuse of their privileges. Suppose that the management board of the building is composed of Alice and Bob, and that it is required that whether or not to allow access be determined only by the board members. In this case, in order to enter the building, one has to prove "due to Alice and Bob, Alice believes $access$" but not simply "Alice believes $access$". In CASE2, Cathy could not obtain the access, because her statement is indispensable to the conclusion that "Alice believes $access$". Hence, the delegation from Bob to Cathy is actually ignored, thus preventing Bob's misuse and neglect. As also pointed out in [6, 13], host security may be compromised if provenance is not taken into account when making authorization decisions.

In the literature, a logic DBT (Due to, Belief and Trust) is designed to represent belief, trust, provenance, and their relations [5]. DBT enables explicit representation of authorization provenance. This work follows the lines of [5].

In this paper, we attempt to track and constrain authorization provenance with respect to logic-based policy bases. Based on DBT, we define two forms of authorization provenance: *simple provenance* and *nested provenance*. We study their properties and thus show that the definition captures important intuitions of authorization provenance (Section 3). We present an example application of the definition of provenance: specification and enforcement of constraints on authorization provenance (Section 4). These constraints can model novel security requirements.

2 Background

We recall the syntax and the semantics of DBT. Consider a countable set of agents AG. DBT has three types of modal operators for each agent i: $\mathtt{B}_i$, $\mathtt{T}^i_j$, and $\mathtt{D}_i$. $\mathtt{B}_i\varphi$ means that agent i *believes* φ. $\mathtt{T}^i_j\varphi$ reads that agent i *trusts* agent j on φ. $\mathtt{D}_i\varphi$ means that "*due to* agent i, φ holds". A subset AE of AG is called an *agent expression*. Given an $AE \subseteq AG$, there is also an operator $\mathtt{D}_{AE}$ based on $\mathtt{D}_i$ for each $i \in AE$. $\mathtt{D}_{AE}\varphi$ means that, due to the set AE of agents together, φ holds. Let Prop be a set of primitive propositions. Given $p \in$ Prop, DBT formulas are inductively defined:

$$\varphi ::= p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \Rightarrow \varphi \mid \mathtt{B}_i\varphi \mid \mathtt{D}_i\varphi \mid \mathtt{D}_{AE}\varphi \mid \mathtt{T}^i_j\varphi.$$

The semantics of DBT formulas is defined based on *Kripke models*. A Kripke model $\mathcal{M}$ is a tuple $\langle W, \pi, \mathcal{B}_i, \mathcal{D}_i, \mathcal{T}^i_j\rangle$ $(i, j \in AG; i \neq j)$, where W is a set of states, $\pi : W \mapsto 2^{\text{Prop}}$ is a labeling function which maps each state to a subset P of Prop such that, in this state, any $p \in P$ is true and any $p \in \text{Prop}\backslash P$ is false, $\mathcal{B}_i \subseteq W \times W$ is a serial, transitive and Euclidean binary relation on W, $\mathcal{D}_i \subseteq W \times W$ is a binary relation on W, and $\mathcal{T}^i_j \subseteq W \times 2^W$ is a binary relation between W and its power set.

Definition 1. *($\models$) Given a model $\mathcal{M} = \langle W, \pi, \mathcal{B}_i, \mathcal{D}_i, \mathcal{T}^i_j\rangle$, $w \in W$, and a formula φ, let $\mathcal{D}_{AE} = \bigcap_{i\in AE}\mathcal{D}_i$. We define the satisfaction relation $\models$ as follows: (1) $\langle\mathcal{M}, w\rangle \models p$ iff $p \in \pi(w)$, (2) $\langle\mathcal{M}, w\rangle \models \neg\varphi$ iff $\langle\mathcal{M}, w\rangle \not\models \varphi$, (3) $\langle\mathcal{M}, w\rangle \models \varphi_1\wedge\varphi_2$ iff $\langle\mathcal{M}, w\rangle \models \varphi_1$ and $\langle\mathcal{M}, w\rangle \models \varphi_2$, (4) $\langle\mathcal{M}, w\rangle \models \varphi_1 \Rightarrow \varphi_2$ iff $\langle\mathcal{M}, w\rangle \not\models \varphi_1$, or $\langle\mathcal{M}, w\rangle \models \varphi_2$, (5) $\langle\mathcal{M}, w\rangle \models \mathtt{B}_i\varphi$ iff $\langle\mathcal{M}, v\rangle \models \varphi$ for all v such that $(w, v) \in \mathcal{B}_i$, (6) $\langle\mathcal{M}, w\rangle \models \mathtt{D}_i\varphi$ iff $\langle\mathcal{M}, v\rangle \models \varphi$ for all v such that $(w, v) \in \mathcal{D}_i$, (7) $\langle\mathcal{M}, w\rangle \models \mathtt{D}_{AE}\varphi$ iff $\langle\mathcal{M}, v\rangle \models \varphi$ for all v such that $(w, v) \in \mathcal{D}_{AE}$, and (8) $\langle\mathcal{M}, w\rangle \models \mathtt{T}^i_j\varphi$ iff $(w, \lceil\varphi\rceil) \in \mathcal{T}^i_j$, where $\lceil\varphi\rceil = \{v \in W \mid \langle\mathcal{M}, v\rangle \models \varphi\}$.*

We define a decentralized authorization system as a tuple $\langle AG, LPB, L, Q, M\rangle$. AG is the set of agents involved in the system. For example, an agent could be a resource-requester, a process running one behalf of a user, and an organization (e.g., a university). We assume that AG includes a member agent called *resource guard*, denoted as G. The guard, as the root of trust, makes authorization decisions.

Credentials are stored in a decentralized way. Each agent maintains a local policy base, which may store credentials that the agent signed and credentials that concern the agent. LPB maps each agent in AG to a set of credentials. We refer to the set of credentials that an agent i maintains as i's *local policy base*. For instance; Alice's local policy base includes a credential that Alice trusts Bob on $access$.

L is the authorization logic used to represent the credentials. In this paper L is DBT. We assume that credentials only encode agents' beliefs and trust. For example, we write $\mathtt{T}_{Bob}^{Alice} access$ for the above-mentioned credential. Q is a set of queries of the form $\mathtt{D}_{AE_n} \cdots \mathtt{D}_{AE_1} \mathtt{B}_i p$. A query usually asks whether an access is allowed. A query for the building-access example is $\mathtt{D}_{\{Alice,Bob\}} \mathtt{B}_{Alice} access$.

M is a mechanism searching for proofs of queries. We assume M is the proof-carrying authorization mechanism [1]. In this case, the resource guard would not permit access unless it is provided with a valid proof which concludes the access is entailed by the policy. The proof is usually submitted by the agent who requests the access. We assume that M is able to access local policy bases to compose proofs. In this case, the union of local policy bases could be treated as one monolithic policy base, thus eliminating the needs of LPB and M. In other words, we abstract the decentralized authorization system as $\langle AG, \text{DBT}, Q, \text{PB} \rangle$, where PB is the monolithic policy base. Formally, a policy base PB is a finite set of $\mathtt{WFF}_{pa}$ formulas defined as below.

$$\phi ::= \mathtt{B}_i p \mid \mathtt{T}_j^i p$$

3 Defining Authorization Provenance

In this section, we examine a simple form of authorization provenance, which is abstracted as the set of agents whose statements are referenced in the deduction of the authorization. The intended function of $\mathtt{D}_{AE}$ is to record the agents who affect agents' beliefs. It appears plausible to define provenance in terms of $\mathtt{D}_{AE}$.

Definition 2 (Authorization Provenance). *Given PB and a query $q = \mathtt{D}_{AE_n} \cdots \mathtt{D}_{AE_1} \mathtt{B}_i p$ such that PB $\models q$, we say $\langle AE_n, \cdots, AE_1 \rangle$ is the* provenance *of $\mathtt{B}_i p$ in PB, denoted as $\mathsf{prov}_{PB}[\mathtt{B}_i p]$. We say $\mathsf{prov}_{PB}[\mathtt{B}_i p]$ is a* simple provenance *if $\mathsf{prov}_{PB}[\mathtt{B}_i p] = \langle AE \rangle$, and a* nested provenance *if $\mathsf{prov}_{PB}[\mathtt{B}_i p] = \langle AE_n, \cdots, AE_1 \rangle$, where $n > 1$.*

Simple provenance is useful when defending against attacks that utilize systems' neglect of privilege source [6, 13]. It has been shown that simply tracking provenance as a set of agents is the key to defend against trojan horses in discretionary access control [6]. Also, the security problem of delegation can be resolved by monitoring delegation's provenance [13]. We now show some properties of simple provenance; nested provenance will be discussed in Section 4.

Distinguishability Suppose that there is more than one way to conclude φ: in addition to AE, AE' could be the provenance of φ. One should be able to express and query φ with these two provenance (i.e., $\mathtt{D}_{AE}\varphi$ and $\mathtt{D}_{AE'}\varphi$) separately. Provenance distinguishability is the basic motivation and design objective of DBT. With the operator $\mathtt{D}_{AE}$, one is able to express and query provenance.

Traceability Any agent contributing to the derivation of φ, should be included in the simple provenance of φ. Re-delegations and attribute-based delegations can easily result in nested provenance of a belief. With this property, we are able to collapse the accumulated provenance into a simple one - a set of agents, no matter how complicated the derivation is. Proposition 1 implies that we can keep provenance concise and simple.

Proposition 1. $\models \mathtt{D}_{AE_n} \cdots \mathtt{D}_{AE_1} \varphi \Rightarrow \mathtt{D}_{AE_1 \cup \cdots \cup AE_n} \varphi$.

With simple provenance, the following property shows that provenance is correctly recorded during the applications of delegations.

Proposition 2. *1.* $\models \mathtt{D}_{AE_1} T_j^i \varphi \wedge \mathtt{D}_{AE_2} B_j \varphi \Rightarrow \mathtt{D}_{AE_1 \cup AE_2 \cup \{j\}} B_i \varphi$.
2. $\models \mathtt{D}_{AE_1} T_j^i \varphi \wedge \mathtt{D}_{AE_2} T_k^j \varphi \Rightarrow \mathtt{D}_{AE_1 \cup AE_2 \cup \{j\}} T_k^i \varphi$.

Proposition 2 shows that, even though delegations and beliefs come along with their own provenance, these provenance would be recorded and merged with new ones in the event of delegations taking effect.

Cooperation Agents may cooperate to finish a task. If a task ψ is divided into n sub-tasks $\varphi_1, \cdots, \varphi_n$, then the union of agent sets, each of which finishes one sub-task, are responsible for the original task.

Proposition 3. $\models (\varphi_1 \wedge \cdots \wedge \varphi_n \Rightarrow \psi) \wedge (\mathtt{D}_{AE_1} \varphi_1 \wedge \cdots \wedge \mathtt{D}_{AE_n} \varphi_n) \Rightarrow \mathtt{D}_{AE_1 \cup \cdots \cup AE_n} \psi$.

Example 1. Suppose a warehouse $task$ is composed of four steps: $prepare$, $payment$, $issue$, and $invoice$. That is, we have $(prepare \wedge payment \wedge issue \wedge invoice) \Rightarrow task$. Suppose further that each of the agents A, B, C, and D accomplish each of the steps, respectively. That is, we also have $\mathtt{D}_A prepare \wedge \mathtt{D}_B payment \wedge \mathtt{D}_C issue \wedge \mathtt{D}_D invoice$. According to Proposition 3, it holds that $\mathtt{D}_{\{A,B,C,D\}} task$; namely, we know it is because $\{A, B, C, D\}$ that the task is done.

Transferability Provenance has a flavor of responsibility in certain cases. Responsibility may be transferred from one agent to another.

Proposition 4. *If* $\models \mathtt{D}_{AE_1 \cup AE_2} \varphi$ *and* $\models \mathtt{D}_{AE_2} \varphi \Rightarrow \mathtt{D}_{AE_3} \varphi$, *then* $\models \mathtt{D}_{AE_1 \cup AE_3} \varphi$.

Suppose that a software manufacturer Mf releases a software called Sw, and that A installs Sw on her computer. For some reasons, Sw automatically executes some malicious script downloaded from a web-site Wb and the script did some damages to her computer (e.g., files being destroyed). Naturally, the responsibility of Sw could be transferred to Mf, as Mf is the producer of Sw. Then, besides Wb, A may complain about Mf. Put formally, from $\models \mathtt{D}_{\{Wb,Sw\}} damage$ and $\models \mathtt{D}_{Sw} damage \Rightarrow \mathtt{D}_{Mf} damage$, we have $\models \mathtt{D}_{\{Wb,Mf\}} damage$ by the transferability.

Proposition 4 implies that the responsibility on the set AE_2 of agents can be transferred to the agent set AE_3. This kind of transferability often happens between agents who are related with each other, like "be a supervisor of".

Limited Responsibility As mentioned before, provenance can be used to trace responsibility: If i delegates the judgement of φ to j, then when j utters her belief in φ, j is the provenance of, and is also responsible for, i's belief in φ. Note that the responsibility is assigned from i's viewpoint. However, j may not speak of φ directly. Suppose that i delegates only the judgement of φ to j, but that j utters her belief in ψ, which is more informative than φ. In this case, j is only responsible for i's belief in φ. In other words, j's responsibility is limited to $\mathtt{B}_i\varphi$.

Proposition 5. *If* $\models \psi \Rightarrow \varphi$*, then* $\models \mathtt{T}^i_j\varphi \wedge \mathtt{B}_j\psi \Rightarrow \mathtt{D}_j\mathtt{B}_i\varphi$.

4 Constraints on Authorization Provenance

While simple provenance suffices in some cases, more informative provenance comes in handy. [5] presents a class of provenance-aware queries with nested provenance; one can syntactically extract useful information about the delegation in PB from a query's provenance, if the query is entailed by the PB. A query $q = \mathtt{D}_{AE_n}\cdots\mathtt{D}_{AE_1}\mathtt{B}_ip$ is *provenance-aware*, if $AE_n = \{i_n,\cdots,i_1\}$, $AE_t = AE_n\backslash\{i_n,\cdots,i_{t+1}\}$ $(n-1 \leq t \leq 1)$, and $i \notin AE_n$, where $\{i_n,\cdots,i_1\} \subseteq AG$. We utilize the following theorem to enforce constraints on authorization provenance.

Theorem 1 ([5]). *If PB* $\models \mathtt{D}_{\{i_l,\cdots,i_l\}}\cdots\mathtt{D}_{\{i_2,i_1\}}\mathtt{D}_{i_1}\mathtt{B}_ip$*, there is a delegation of* p *from* i *to* i_l*, from* i_l *to* i_{l-1}*,* $\cdots$*, and finally from* i_2 *to* i_1*.*

4.1 Definition

Several attacks are found related to authorization provenance. For example, Wang et al. [13] point out users may abuse delegations to circumvent security policies (in particular, *separation of duty* policies).

Example 2. Consider a task of issuing checks [13]: In a company, the task of issuing checks is modeled by two authorizations pre and app, which stand for "prepare check" and "approve check", respectively. In order to prevent fraudulent transactions, a separation of duty policy $sod\langle pre, app\rangle$ that pre and app must be performed by two *different* treasurers is required. Also, for the sake of resiliency, the company allows a treasurer to delegate her authority to a clerk in case she is not able to work.

Suppose that a treasurer, A, and a clerk of the company, B, decide to collude to issue checks for themselves. They can accomplish in three steps: A delegates the authority pre to B; B performs pre to prepare a check for A; and A performs app to approve the check prepared by B.

The observation is that both requests to pre and app are (in part) from A. Similarly, A can create dummy agents (e.g., processes) or manipulate behind the scene (e.g., trojan horses) [6]. Unfortunately, systems that employ logic-based policy bases are also vulnerable to these attacks. The defense mechanism proposed in [13] addresses the problem in work-flow systems; it is unclear how to adapt that approach in face of logic-based policies. Moreover, in spite of the importance of separation of duty policies, other compromises involving provenance deserve investigation in their own right.

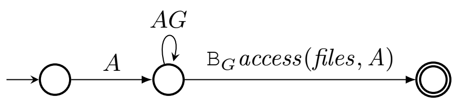

Fig. 1. The BP-NFA γ_A.

The attacks are mainly caused by the ignorance of how an authorization is derived. This implies that we may resolve them by putting constraints on authorization provenance. We follow the convention to model constraints as finite state automata [9]. We work on non-deterministic finite automata (NFA). Let the alphabet of the automaton be $\Sigma = AG \cup \mathtt{WFF}_{pa} \cup \{\epsilon\}$. An NFA γ is a tuple $(V, \Sigma, \delta, v_0, F)$, where V is a finite set of states, Σ is an alphabet, $\delta : V \times \Sigma \mapsto 2^V$ is a transition function, $v_0 \in V$ is a start state, and $F \subseteq V$ is a set of accept states [10].

Definition 3 (Basic Provenance NFA (BP-NFA)). *A* basic provenance NFA *is an NFA* $(V, \Sigma, \delta, v_0, F)$ *that meets the two conditions: (1)* $|F| = 1$*, and (2) for any state* $v \in V$ *and any* $\sigma \in \Sigma$*,* $\delta(v, \sigma) \subseteq F$ *if and only if* σ *is of the form* $\mathtt{B}_i p$*. Suppose that* $F = \delta(v, \phi)$*; we say that* γ *ends with* ϕ*, denoted as* $\textsf{end}_\gamma[\phi]$*.*

The intuition of the two conditions is that each BP-NFA recognizes only one query at a time. Given a provenance aware query $q = \mathtt{D}_{AE_n} \cdots \mathtt{D}_{AE_1} \mathtt{B}_i p$, we define a string over Σ as $s_n s_{n-1} \cdots s_1 s_0$, where, for $1 \leq t \leq n$, $s_t = AE_t \backslash AE_{t-1}$ and $s_0 = \mathtt{B}_i p$; denote the string as *str*$[q]$. We say a BP-NFA γ *recognizes* q if γ accepts *str*$[q]$.

Take the BP-NFA γ_A in Fig. 1 for example. Consider the access to A's files, which is authorized if one can prove $\mathtt{D}_{AG} \mathtt{B}_G access(\mathit{files}, A)$. γ_A ends at $\mathtt{B}_G access(\mathit{files}, A)$. γ_A recognizes $\mathtt{D}_{\{A,C\}} \mathtt{D}_C \mathtt{B}_G access(\mathit{files}, A)$ but not $\mathtt{D}_{\{B,A,C\}} \mathtt{D}_{\{A,C\}} \mathtt{D}_C \mathtt{B}_G access(\mathit{files}, A)$, for the string of the latter does not start with A.

Some constraints are put on a combination of authorization provenance but not a single one. For example, the separation of duty policy *sod*$\langle pre, app\rangle$ forbids any user from executing both pre and app.

Definition 4 (Concatenated Provenance NFA (CP-NFA)). *Given two BP-NFA,* $\gamma_1 = (V_1, \Sigma, \delta_1, v_{0,1}, F_1)$ *and* $\gamma_2 = (V_2, \Sigma, \delta_2, v_{0,2}, F_2)$*, define the concatenation of* γ_1 *and* γ_2 *as* $\gamma_1 \circ \gamma_2 = (V, \Sigma, \delta, v_0, F)$ *such that* $V = V_1 \cup V_2$*,* $v_0 = v_{0,1}$*,* $F = F_2$*, and*

$$\delta(v, \sigma) = \begin{cases} \delta_1(v, \sigma) & \textit{if } v \in V_1 \textit{ and } v \notin F_1 \\ \delta_1(v, \sigma) & \textit{if } v \in F_1 \textit{ and } \sigma \neq \epsilon \\ \delta_1(v, \sigma) \cup \{v_{0,2}\} & \textit{if } v \in F_1 \textit{ and } \sigma = \epsilon \\ \delta_2(v, \sigma) & \textit{if } v \in F_2. \end{cases}$$

CP-NFA are used to recognize a set of queries. Denote the concatenation of two strings *str*$_1$ and *str*$_2$ as *str*$_1 \diamond$ *str*$_2$. Given a set Q of provenance aware queries and a CP-NFA γ, we say γ *recognizes* Q if $Q = \{q_1, \cdots, q_n\}$ and there exists a sequence $\langle q_1, \cdots, q_n\rangle$ such that γ accepts the string *str*$[q_1] \diamond \cdots \diamond$ *str*$[q_n]$.

Suppose that the request to pre (respectively, app) is accompanied with a proof whose conclusion is $\mathtt{D}_{AE_n} \cdots \mathtt{D}_{AE_1} \mathtt{B}_G pre$ (respectively, $\mathtt{D}_{AE'_n} \cdots \mathtt{D}_{AE'_1} \mathtt{B}_G app$). Let X

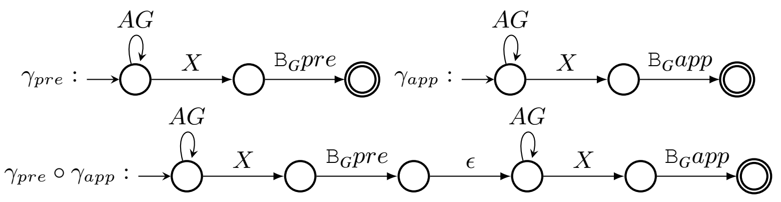

Fig. 2. $\gamma_{sod} = \gamma_{pre} \circ \gamma_{app}$.

be a meta-variable over AG. The CP-NFA γ_{sod} in Fig. 2 recognizes a combination of requests of pre and app which are both issued by X, regardless of intermediate delegation. For instance, γ_{sod} recognizes $\{\mathtt{D}_{\{A,X\}}\mathtt{D}_X\mathtt{B}_G pre, \mathtt{D}_{\{B,C,X\}}\mathtt{D}_{\{C,X\}}\mathtt{D}_X\mathtt{B}_G app\}$.

Definition 5 (Constraints on Authorization Provenance). *A constraint on authorization provenance is a tuple $\langle \gamma, sign \rangle$, where γ is either a BP-NFA or a CP-NFA and $sign \in \{+, -\}$. Given a set Q of provenance aware queries, we say Q* satisfies $\langle \gamma, sign \rangle$ *if γ recognizes Q when $sign = +$, and γ does not recognize Q when $sign = -$. Given a set C of constraints on provenance, we say Q satisfies C if for all $c \in C$, Q satisfies c.*

A constraint $\langle \gamma, + \rangle$ requires that the provenance of the involved authorizations matches the pattern specified by γ, whereas $\langle \gamma, - \rangle$ means that the provenance must not be recognized by γ.

4.2 Example Constraints

Discretionary access control safety We can express discretionary access control safety by a machine which accepts the provenance starting with the owner of the object. Consider again the BP-NFA γ_A in Fig. 1. Then the constraint $\langle \gamma_A, + \rangle$ requires the provenance of $\mathtt{B}_G access(\mathit{files}, A)$ starts with A; that is, every access to A's files originates from A or her delegation.

Group-related constraints We restrict an authorization to a group of users by $\langle \gamma_{group}, + \rangle$, where γ_{group} is shown in Fig. 3. The BP-NFA γ_{group} specifies that only if the access is directly requested by $group$ members (in the sense that a member is at the end of the provenance) would it be allowed.

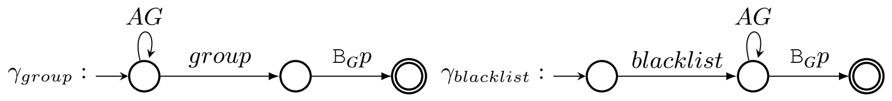

Fig. 3. Example BP-NFA related to groups.

Also, blacklist can be enforced via a constraint $\langle \gamma_{blacklist}, - \rangle$. By this constraint, any authorization with provenance beginning with a blacklisted user would be declined.

Traditional approaches to blacklisting may fall short in face of delegations, for they do not take authorization provenance into account.

Separation of duty With γ_{sod} in Fig. 2, $\langle\gamma_{sod}, -\rangle$ enforces *sod*$\langle pre, app\rangle$ in the traditional sense. However, it fails to enforce *sod*$\langle pre, app\rangle$ in the presence of delegation. For example, $\{\mathtt{D}_{\{A,B\}}\mathtt{D}_B\mathtt{B}_G pre, \mathtt{D}_A\mathtt{B}_G app\}$ satisfies $\langle\gamma_{sod}, -\rangle$, but does not complies with *sod*$\langle pre, app\rangle$. With the BP-NFA $\gamma_{gen-pre}$ and $\gamma_{gen-app}$ in Fig. 4, the CP-NFA $\gamma_{gen-sod} = \gamma_{gen-pre} \circ \gamma_{gen-app}$ checks if the requests of pre and app are both from X, either directly or indirectly via delegation. The constraint $\langle\gamma_{gen-sod}, -\rangle$ prevents users from circumventing *sod*$\langle pre, app\rangle$ with the help of delegation.

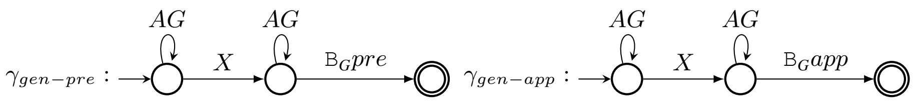

Fig. 4. $\gamma_{gen-sod} = \gamma_{gen-pre} \circ \gamma_{gen-app}$.

Consumable resources. Some resources are consumable in the sense that the times of usage are limited. For example, a member of $board$ can invite at most two agents to a conference by means of delegation; that is, her usage of invitation is limited. Let X be a meta-variable over the members of $board$; the BP-NFA $\gamma_{consume}$ in Figure 5 recognizes one consumption of X's invitation. Hence, $\langle\gamma_{consume} \circ \gamma_{consume} \circ \gamma_{consume}, -\rangle$ forbids X from inviting more than two users.

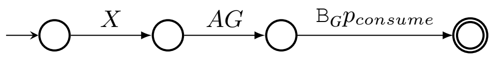

Fig. 5. $\gamma_{consume}$.

4.3 Enforcement

Suppose that the resource guard G needs to enforce a set C of constraints. We assume G maintains a history $\mathcal{H}$ of provenance aware queries that correspond to previous authorizations. Each time an authorization is granted, the associated query is added to $\mathcal{H}$. When a new access request arrives, G checks if the policy base entails the corresponding query q; since proof-carrying authorization is used, this amounts to checking the proof. If so, G proceeds to verify if C is satisfied by Algorithm 1. The request is allowed if an answer "true" is returned; and $\mathcal{H}$ is updated to $\mathcal{H} \cup \{q\}$. Otherwise, the requested is denied.

The algorithm can be optimized in several ways. First, when only constraints involving BP-NFA are required, the history is not needed. Each time a request comes,

Algorithm 1: Algorithm for enforcing constraints on authorization provenance.

```
begin
    foreach $\langle \gamma, sign \rangle \in C$ do
        if $\gamma$ is a BP-NFA then
            if $\gamma$ recognizes $q$ then
                if $sign = -$ then return false;
            else
                if $sign = +$ then return false;
        if $\gamma$ is a CP-NFA of the form $\gamma_1 \circ \cdots \circ \gamma_m$ then
            if $end_{\gamma_1}[B_G p] \vee \cdots \vee end_{\gamma_m}[B_G p]$ then
                for each $1 \leq t \leq m$, let $\mathcal{H}(\gamma_t) = \{\mathtt{D}_{AE_n} \cdots \mathtt{D}_{AE_1} \mathtt{B}_G p \in \mathcal{H} \cup \{q\} \mid \mathsf{end}_{\gamma_t}[\mathtt{B}_G p]\}$;
                foreach sequence $\langle q_1, \cdots, q_m \rangle$ such that, for $1 \leq t \leq m$, $q_t \in \mathcal{H}(\gamma_t)$ do
                    if $\gamma$ recognizes $\langle q_1, \cdots, q_m \rangle$ then
                        if $sign = -$ then return false;
                    else
                        if $sign = +$ then return false;
    return true;
```

we check the corresponding query against BP-NFA. BP-NFA concerns a single agent's authorizations. On the other hand, CP-NFA often involves collusion among multiple agents, which is more costly. Violations against BP-NFA seem more common. Second, one may index historical queries with respect to CP-NFA; this is likely to avoid searching through the history. Finally, constraints based on CP-NFA are usually temporary. For example, the constraint $\langle \gamma_{consume} \circ \gamma_{consume} \circ \gamma_{consume}, - \rangle$ is only effective during the conference; related queries can be discarded after the conference. This helps reduce $\mathcal{H}$'s size. We leave it to future work to investigate the optimization in detail.

5 Related Work and Conclusions

Various mechanisms have been devised to track, store, and query provenance in database and file systems [7, 11]. While one may borrow ideas from these techniques, it is not clear how to adapt them into distributed authorization arena. Actually, authorization provenance deserves investigation in its own right. For example, of importance to authorization provenance are agents who affect the authorization, instead of the data inputs and the processes that the inputs undergo. Meanwhile, the security of provenance attracts considerable efforts [2, 4, 8]; the aim is to protect the confidentiality and integrity of provenance information itself and to provide access control to provenance. This paper, however, focuses on exploring provenance of authorization and its potential usage in access control area, but not on the security of provenance. Provenance is related to the notion of *causality*. van der Meyden [12] discusses the relation between causality and distributed knowledge. He argues that only a predefined set of agents may cause an agent to know a proposition (i.e., certain information), otherwise the system is considered insecure. Our work differs from [12] in two aspects. First, in distributed authorization, the set of agents who may affect the knowledge of an agent can not always be predefined; this uncertainty is a sacrifice for the flexibility via delegation. Second, the emphasis of [12] is on defining secure systems in terms of information flow policies, which defines the causal relation allowed among agents. It is hard to conceive of how to define authorization provenance using the framework in [12]. The following on work by Chong and van der Meyden [3] studies information flow properties using epistemic logic under the similar framework. They put no emphasis on authorization provenance.

In this paper, we defined a notion of authorization provenance, based on the logic DBT. We showed that this notion possesses a collection of interesting properties and thus captures the intuitions of authorization provenance. As an application of authorization provenance, we also illustrated the specification and enforcement of a new type of security requirements. There are several avenues for future work. On the one hand, more efficient algorithms for enforcing constraints are worth pursuing; on the other hand, we will consider the notion of roles when tracking provenance.

Acknowledgment

This publication was made possible by the support of an NPRP grant (NPRP 09-079-1-013) from the Qatar National Research Fund (QNRF). The statements made herein are solely the responsibility of the authors.

References

1. A. W. Appel and E. W. Felten. Proof-carrying authentication. In *ACM Conference on Computer and Communications Security*, pages 52–62, 1999.
2. U. Braun, A. Shinnar, and M. Seltzer. Securing provenance. In *Proc. of the 3rd USENIX Workshop on Hot Topics in Security (HotSec)*, July 2008.
3. S. Chong and R. van der Meyden. Deriving epistemic conclusions from agent architecture. In *TARK*, July 2009.
4. R. Hasan, R. Sion, and M. Winslett. The case of the fake picasso: Preventing history forgery with secure provenance. In *Proceedings of the 7th USENIX Conference on File and Storage Technologies (FAST)*, 2009.
5. J. Hu, Y. Zhang, R. Li, and Z. Lu. A logic for authorization provenance. In *Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security*, pages 238–249. 2010.
6. Z. Mao, N. Li, H. Chen, and X. Jiang. Trojan horse resistant discretionary access control. In *ACM symposium on access control models and technologies*, 2009.
7. K.-K. Muniswamy-Reddy, D. A. Holland, U. Braun, and M. I. Seltzer. Provenance-aware storage systems. In *Proc. of the USENIX Annual Technical Conference*, pages 43–56, 2006.
8. Q. Ni, S. Xu, E. Bertino, R. Sandhu, and W. Han. An access control language for a general provenance model. In *Proceedings of the 6th VLDB Workshop on Secure Data Management*, 2009.
9. F. B. Schneider. Enforceable security policies. *ACM Trans. Inf. Syst. Secur.*, 3(1):30–50, 2000.
10. M. Sipser. *Introduction to the Theory of Computation*. 2005.
11. W. C. Tan. Provenance in databases: Past, current, and future. *IEEE Data Eng. Bull.*, 30(4):3–12, 2007.
12. R. van der Meyden. On notions of causality and distributed knowledge. In *International Conference on Principles of Knowledge Representation and Reasoning*, pages 209–219, 2008.
13. Q. Wang, N. Li, and H. Chen. On the security of delegation in access control systems. In *European Symposium on Research in Computer Security*, pages 317–332, 2008.