Towards Role Based Trust Management without Distributed Searching of Credentials

Gang Yin[1], Huaimin Wang[1], Jianquan Ouyang[2], Ning Zhou[3], and Dianxi Shi[1]

[1] School of Computer, National University of Defense Technology, Changsha, China
jack.nudt@gmail.com, whm_w@163.com, dxshi@nudt.edu.cn
[2] College of Information Engineering, Xiangtan University, Xiangtan, China
oyjq@ict.ac.cn
[3] Institute of Electronic System Engineering of China, Beijing, China
humimi74@yahoo.cn

Abstract. Trust management systems enable decentralized authorization by searching distributed credentials from network. We argue that such distributed searching processes may encounter many technical or non-technical problems, and can be avoided by storing delegation credentials redundantly with acceptable costs. We propose a scoped-role based trust management system ScoRT, using a novel credential affiliation model to compute the credentials necessary for role membership decisions, which can be used to guide the storage, retrieval and revocation of credentials. The algorithm for distributed credential storage and retrieval is designed based on the model and its sound and complete properties are formally analyzed with respect to ScoRT semantics. Complexity analysis and estimation show that, by redundantly storing acceptable amount of delegation credentials, ScoRT enables more practical and automatic authorization without searching credentials from remote entities, and thus helps to overcome the deficiencies of existing approaches.

1 Introduction

Trust management (TM) systems use delegation credentials to realize flexible and scalable authorization across security domains. A number of TM systems have been proposed to enable various delegation mechanisms, such as [2, 4, 5, 7, 10, 14, 15, 16]. The primary idea of delegation is that one entity gives some of its authority to others to make authorizations on behalf of the former. Multi-steps of delegation among different entities may result in chains of credentials, which are prerequisite for making authorization decisions in TM systems. Some TM systems study the distributed storage and retrieval problems of credential chains, which can be mainly classified into logic-based approach [1, 12, 14] and graph-based approach [13, 16, 18].

The logic-based approach requires that each credential is defined with a specified location, and servers pull credentials from remote entities (usually one credential at a time) during the process of logic-based compliance-checking, such as QCM [12], SD3 [14] and Cassandra [2]. The graph-based approach retrieves the whole credential chains that delegate the privileges from authorizers to requesters, such as the discovery methods of certificate paths or credential chains [8, 9, 13, 16]. These two approaches

L. Chen, M.D. Ryan, and G. Wang (Eds.): ICICS 2008, LNCS 5308, pp. 222–237, 2008.

are mainly used to search credentials in dynamic and decentralized environments, which may face a lot of technical and non-technical problems. (1) They use the depth-first or breadth-first search process to find credential chains, which will retrieve a lot of useless credentials during back-tracking processes, and if the authorization queries have negative results, the credentials may become tremendous. (2) The issuers of the target credentials may not always trust or be trusted by the credential requesters, and thus the privacy of credentials may be breached by uncontrolled credential retrieval. (3) Entities holding the target credentials may not on-line all the time and thus the requests will be rejected if any credential in the chain can not be retrieved in time. These problems may breach the security and availability of systems.

The primary task of credential searching is to retrieve delegation credentials from remote entities. Above observations motivate us to revisit the problem of distributed management of credentials. Due to the balance of scalability and controllability of delegation mechanisms, we believe that delegation credentials should be used and configured as the backbone of collaboration networks. For example, M. Becker gives a comprehensive analysis of policies in EHR systems [3], the delegation credentials across security domains are only 8% of the total 375 credentials.

In this paper, we propose a novel credential distribution approach to store delegation credentials redundantly such that every entity can hold all necessary delegation credentials for making access decisions. To ensure the generality of our approach, we propose ScoRT, a role-based TM system which combines the primary capabilities of both RT [16] and SPKI [10]. The credential affiliation model is proposed to computes the affiliation graphs for roles, and each affiliation graph contains all the delegation chains starting from these roles. A credential distribution algorithm is designed based on the model to publish and retrieve the credentials among the entities in the network.

The rest of this paper is organized as follows. Section 2 defines ScoRT and the credential affiliation model. Section 3 introduces the credential management framework of ScoRT and proves the sound and complete properties of credential distribution algorithm. Section 4 estimates the complexity of our approach based on Aura's delegation network model. Section 5 analyzes the related works and section 6 concludes the paper.

2 Credential Affiliation in Trust Management

ScoRT is a role-based TM system combing the primary advantages of both role-based trust management [17] and SPKI [10]. ScoRT uses scoped roles to enable and control the delegation of role-based privileges.

2.1 ScoRT: Trust Management with Scoped-Roles

ScoRT introduces the notion of scoped roles, and a scoped role is a role appended with a trust scope tag, which is either ■ or □. Given an entity A and a role name r, the $A.r$ has two scoped roles $A.r.■$ and $A.r.□$. The scoped roles can be regarded as a kind of refined abstraction of principals. Intuitively, $A.r.■$ denotes the entities directly assigned with $A.r$ by A, while $A.r.□$ denotes the entities directly or indirectly assigned with $A.r$. ScoRT has three kinds of credentials:

- type-1: $A.r \leftarrow B$
- Entity B is a member of the role $A.r$.
- type-2: $A.r \leftarrow B.r_1.s$
- Members of the scoped role $B.r_1.s$ are members of the role $A.r$.
- type-3: $A.r \leftarrow B_1.r_1.s_1 \wedge B_2.r_2.s_2$
- Members of both $B_1.r_1.s_1$ and $B_2.r_2.s_2$ are members of the role $A.r$. We call $B_1.r_1.s_1 \wedge B_2.r_2.s_2$ a scoped role intersection or an intersection for short.

The type-1 credentials are authorization credentials, while the type-2 and type-3 credentials are delegation credentials. Given a credential c, the entity at left-side is called the issuer of c, and the entity at right-side is called subject of c. ScoRT only supports the intersections of two scoped roles, but the intersections of more scoped roles can be defined by introducing new intermediate credentials. For example, the intersection of $B_1.r_1.s_1$ and $B_2.r_2.s_2$ can be replaced by $C.r.\square$ by introducing the credential $C.r \leftarrow B_1.r_1.s_1 \wedge B_2.r_2.s_2$.

Example 1. The following ScoRT credentials have similar meanings to the sample credentials in [16], but these credentials can provide more refined delegation control.

(1) EPub.discount ← EOrg.preferred.□ ∧ ACM.member.■
(2) EOrg.preferred ← StateU.student.□ (3) StateU.student ← RegB.student.■
(4) ACM.member ← Alice (5) RegB.student← Alice

The credential (3) means that all students *directly* assigned by RegB can be the students of StateU, which *implicitly* defines the delegation from StateU to RegB of StateU.student within the depth 2. The credential (2) defines delegation from EOrg to StateU of EOrg.preferred without depth control. The semantics of ScoRT is defined by Datalog [19], which decides whether an entity is a member of a given role.

Definition 1 (Semantics of ScoRT). Given a credential set Σ, a role $A.r$ and an entity B, we use $\Sigma \hookrightarrow \mathrm{mem}(B, A.r)$ to denote that B is a member of $A.r$, and:

$$\Sigma \hookrightarrow \mathrm{mem}(B, A.r) \text{ iff } P_\Sigma \vdash \mathrm{m}(B, A.r, \square)$$

where P_Σ is the set of definite Datalog rules derived from Σ, $\vdash$ is the logical consequence relation. P_Σ contains a rule:

$$\mathrm{m}(x, y.r, \square) \leftarrow \mathrm{m}(x, y.r, \blacksquare). \quad \text{(R1)}$$

P_Σ also contains the rules obtained by a transform process. For each $A.r \leftarrow b \in \Sigma$, do the transform according to three rules:
(T1) if b is an entity, then $\mathrm{m}(b, A.r, \blacksquare) \in P_\Sigma$;
(T2) if b is $B.r_1.s$ then $\mathrm{m}(x, A.r, \square) \leftarrow \mathrm{m}(x, B.r_1, s) \in P_\Sigma$;
(T3) if b is $B_1.r_1.s_1 \wedge B_2.r_2.s_2$ then $\mathrm{m}(x, A.r, \square) \leftarrow \mathrm{m}(x, B_1.r_1, s_1)$, $\mathrm{m}(x, B_2.r_2, s_2) \in P_\Sigma$.

2.2 Credential Affiliation

The credential affiliation model is built upon ScoRT. Given a set Σ of ScoRT credentials, the authorization structures of Σ can be modeled by a partial weighed directed graph, in which the type-1 credentials are mapped to simple directed edges, while the type-2 and type-3 are mapped to weighed directed edges.

Given a set Σ of credentials, we use $\Sigma.\text{entities}$, $\Sigma.\text{roles}$ and $\Sigma.\text{scoints}$ to denote the set of all entities appeared in the right side of type-1 credentials in Σ, the set of all roles in the credentials in Σ, and all the intersections appeared in Σ.

Definition 2 (Credential Graph). Given a set Σ of credentials, the credential graph for Σ is denoted by a weighed directed graph G_Σ which has a node set N_Σ and an edge set E_Σ, defined as follows:

$$N_\Sigma = \Sigma.\text{entities} \cup \Sigma.\text{roles} \cup \Sigma.\text{scoints}$$
$$E_\Sigma \subseteq E_\Sigma.\text{AE} \cup E_\Sigma.\text{TE} \cup E_\Sigma.\text{IE}$$

where $E_\Sigma.\text{AE}$, $E_\Sigma.\text{TE}$ and $E_\Sigma.\text{IE}$ are the set of authorization edges, trust edges and intersection edges in E_Σ, which are further defined as follows:

$$E_\Sigma.\text{AE} \subseteq \Sigma.\text{roles} \times \Sigma.\text{entities}$$
$$E_\Sigma.\text{TE} \subseteq (\Sigma.\text{roles} \cup \Sigma.\text{scoints}) \times \Sigma.\text{roles} \times \text{TS}$$
$$E_\Sigma.\text{IE} \subseteq \Sigma.\text{roles} \times \Sigma.\text{scoints} \times \wp(\Sigma.\text{entities})$$

where TS={■, □} and $\wp(\Sigma.\text{entities})$ is the power set of $\Sigma.\text{entities}$. G_Σ uses a long arrow $\longleftarrow$ to denote a directed edge and a weighed long arrow $\xleftarrow{w}$ to denote a weighed directed edge with the weight *w*. If there is an edge from *n* to *n′*, then *n′* is a *successor* of *n*. Given a path ξ from node *n* to *n′* in G_Σ, $n \neq n'$ and *n″* is the successor of *n* in ξ, then ξ is a legal path denoted by $n' \twoheadleftarrow n$ if one of the following is satisfied:

- if *n* is a role, then the weight of each trust edge between *n″* and *n′* in ξ is □.
- if *n* is an entity, then the sub-path from *n″* to *n′* in ξ is a legal path and the weight of each intersection edge in the sub-path contains *n*.
- if ξ is a sub-path of a legal path, then ξ is a legal path.

We use $n' \twoheadleftarrow n \in G_\Sigma$ to denote that $n' \twoheadleftarrow n$ is a legal path in G_Σ. Given $n'' \xleftarrow{w} n'$ and $n' \twoheadleftarrow n$ in G_Σ, if the path formed by linking $n'' \xleftarrow{w} n'$ and $n' \twoheadleftarrow n$ is a legal path then $n'' \xleftarrow{w} n' \twoheadleftarrow n \in G_\Sigma$. Similarly, $n'' \twoheadleftarrow n' \xleftarrow{w} n \in G_\Sigma$ if the path formed by linking $n'' \twoheadleftarrow n'$ and $n' \xleftarrow{w} n$ is a legal path. The subsets of E_Σ can be constructed by the closure properties:

Closure Property 1: Given $B \in \Sigma.\text{entities}$, $A.r \twoheadleftarrow B \in \Sigma$, then $A.r \longleftarrow B \in E_\Sigma.\text{AE}$.

Closure Property 2: Given $A.r \twoheadleftarrow b \in \Sigma$ and *b* is not an entity, then for each scoped role *n.s* in *b*, $A.r \xleftarrow{s} n \in E_\Sigma.\text{TE}$.

Closure Property 3: Given $A.r \twoheadleftarrow b \in \Sigma$ and $b \in \Sigma.\text{scoints}$, if there is no edge from *b* to *A.r* then $A.r \xleftarrow{\varnothing} b \in E_\Sigma.\text{IE}$.

Closure Property 4: Given $B \in \Sigma.\text{entities}$, $A.r \twoheadleftarrow b \in \Sigma$ and $b \in \Sigma.\text{scoints}$, if $A.r \xleftarrow{w} b \in E_\Sigma.\text{IE}$ and $b \xleftarrow{w} n \twoheadleftarrow B \in G_\Sigma$ for each role *n* in *b* then $A.r \xleftarrow{w \cup \{B\}} b \in E_\Sigma.\text{IE}$.

The credential graph for the credentials in example 1 is shown in Figure 1. The directed edge ACM.member $\longleftarrow$ Alice is an authorization edge of the credential (4). The weighed directed edges EOrg.perferred $\xleftarrow{\square}$ StateU.student is a trust edge of the credential (2). Let α be the intersection EOrg.preferred.□∧ACM.member.■, the trust edges

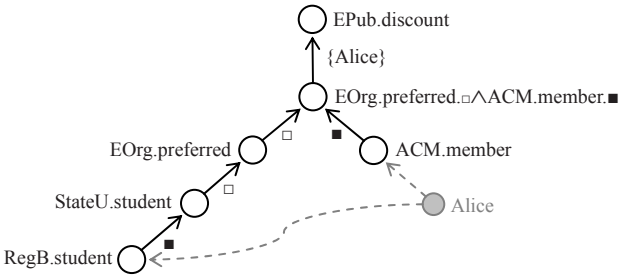

Fig. 1. Weighed Credential Graphs of ScoRT

$\alpha \xleftarrow{\square}$ EOrg.preferred and $\alpha \xleftarrow{\blacksquare}$ ACM.member are derived from credential (1). By definition 2, EOrg.perferred$\twoheadleftarrow$Alice and ACM.member$\twoheadleftarrow$Alice, and thus the edge EPub.discount $\xleftarrow{\{Alice\}} \alpha$ is the intersection edge of the credential (1).

Given a credential $A.r \leftarrow b$, we use Edges($A.r \leftarrow b$) to denote the edge set S, where S is $\{A.r \longleftarrow b\}$ if b is an entity, and S is $\{A.r \xleftarrow{s} B.r_1\}$ if b is a scoped role $B.r_1.s$, and S is $\{A.r \xleftarrow{\varnothing} b, b \xleftarrow{s_1} B_1.r_1, b \xleftarrow{s_2} B_2.r_2\}$ if b is an intersection $B_1.r_1.s_1 \wedge B_2.r_2.s_2$. We use the concept of affiliation graph to model the credentials which influence the decision on whether an entity is a member of a specified role.

Definition 3 (Affiliation Graph). Given a set Σ of credentials, a role or intersection n, the affiliation graph of n is denoted by G_Σ^n, and n is called the *root* of G_Σ^n. N_Σ^n and E_Σ^n are sets of nodes and edges in G_Σ^n, constructed by AffG (n, $N_\Sigma^n = \varnothing$, $E_\Sigma^n = \varnothing$):

AffG (n /*input node*/, ns /*node set*/, es /*edge set*/)

1. if n is marked then return;
2. add n into ns; mark n to be processed;
3. for each edge e of the form $n \xleftarrow{w} n'$ in G_Σ do
4. add e into es;
5. if w=□ then AffG (n', ns, es);
6. if w=∅ then for each scoped role $n''.s$ in n' do
7. add $n' \xleftarrow{s} n''$ into es;
8. if s=□ then AffG (n'', ns, es);
9. return;

Both space and time complexities of affiliation graphs are liner to the size of given credentials. By definition 3, type-1 credentials are not used when constructing affiliation graphs. Given a set Σ of non-type-1 credentials, N is the number of credentials in Σ, by definition 2, G_Σ has at most $4N$ nodes ($3N$ role nodes and N intersection nodes) and $4N$ edges ($3N$ trust edges and N intersection edges). Given a role $A.r$, $G_\Sigma^{A.r}$ is a sub-graph of G_Σ by definition 3. Therefore the space complexity of $G_\Sigma^{A.r}$ is $O(N)$.

By definition 2, constructing G_Σ only need one iteration step and time complexity of constructing G_Σ is $O(N)$. By definition 3, every processed node is marked and there are

at most $4N$ nodes in G_Σ, and thus lines 2, 4, 7 of AffG will be executed at most $4N$ times when constructing $G_\Sigma^{A.r}$. Therefore the time complexity of AffG is $O(N)$.

Lemma 1. Given a credential set Σ, a role $A.r$ and an entity D, then $\Sigma \hookrightarrow \text{mem}(D, A.r)$ if and only if $A.r \twoheadleftarrow D \in G_\Sigma$.

Given a set Σ of credentials, we use $G_\Sigma^{A.r:B}$ to denote an extension of $G_\Sigma^{A.r}$ which is called the specified affiliation graph of $A.r$ for B. The node set and edge set of $G_\Sigma^{A.r:B}$ are $N_\Sigma^{A.r:B} = N_\Sigma^{A.r} \cup \{B\}$ and $E_\Sigma^{A.r:B} = E_\Sigma^{A.r} \cup \Delta_B$, where Δ_B is $\{e \mid e \text{ is } n \longleftarrow B \text{ and } e \in E_\Sigma.\}$ containing all the authorization edges from B. The intuitionistic meaning of $G_\Sigma^{A.r:B}$ is that it contains all the credentials that maybe used to decide whether B is a member of $A.r$. The following lemmas show some basic properties of affiliation graphs.

Lemma 2. Given a set Σ of credentials, let $A.r$ be a role in $\Sigma.\text{roles}$ and n be an node in G_Σ, if $A.r \twoheadleftarrow n \in G_\Sigma$ then $A.r \twoheadleftarrow n \in G_\Sigma^{A.r}$.

Lemma 3. Given a set Σ of credentials, a role $A.r$ and an entity D, then $A.r \twoheadleftarrow D \in G_\Sigma$ if and only if $A.r \twoheadleftarrow D \in G_\Sigma^{A.r:D}$.

From lemma 1 and lemma 3, the affiliation graph is sound and complete with respect to the semantics of ScoRT.

Theorem 1 (Soundness and Completeness of Affiliation Graph). Given a set of Σ of credentials, a role $A.r$ and an entity D, then $\Sigma \hookrightarrow \text{mem}(D, A.r)$ if and only if $A.r \twoheadleftarrow D \in G_\Sigma^{A.r:D}$.

3 Credential Management Framework

ScoRT provides a framework for distributed storage, retrieval and revocation, which is mainly guided by the affiliation graph model. The framework uses different policies to handle authorization credentials and delegation credentials.

ScoRT uses credential distribution algorithm (CDA) to exchange credentials among collaborating entities. When a CDA algorithm is invoked, a CDA instance will be created by its local entity. Given a CDA instance α, if α is invoked by its local entity, then α is a *root* instance, otherwise α is called a *derived* instance ScoRT manages credentials based on the following policies.

Definition 4 (Credential Management Policies). ScoRT manages the credentials according to four general and intuitionistic policies:

Policy CMP1: Authorization credentials are stored at its issuers and subjects, and should be pushed to authorizers if required by authorization checking.

Policy CMP2: Delegation credentials are stored at their issuers and subjects; the affiliation graphs of each role should be stored at its defining entity.

Policy CMP3: Credentials can only be revoked by their issuers by sending revoking messages to related entities, or by setting expiration time for each credentials.

Policy CMP4: Root CDA instances run one by one serially in order to ensure the consistency of credential distribution.

Given a time τ and a credential set Σ, we use $\Sigma|_\tau$ and $G_\Sigma|_\tau$ to denote the snapshots of Σ and G_Σ at τ respectively. Given an entity A, we use Σ_A to denote the credentials stored at A, and use G_A to briefly denote G_{Σ_A}. Given a CDA instance α, we use α^- and α^+ to denote the moments when α just begins and ends.

3.1 Credential Distribution Algorithm

CDA exchanges the credentials among entities based on CMP1 and CMP2. When an entity A issues a credential $A.r \leftarrow b$, it creatse a root CDA instance by calling CDA($A.r \leftarrow b$, *nil*, *nil*). A root CDA instance may retrieve credentials from remote entities (lines 7 and 11), push credentials to remote entities (lines 4, 8 and 12), and invoke derived instances by calling CDA on remote entities (lines 16 and 17).

The statement "*n* contains *v*" in line 14 means that *v* is a role appearing in *n*. Given an entity B, we use $B.cda$ to denote the invocation of CDA on entity B, and thus a chain of CDA instances may be created with cascaded invocations. Given an entity B and a credential c, the statement "send c to B" means that after receiving c, B will store c at its local repository and add Edges(c) into G_B.

Pseudo-codes of Credential Distribution Algorithm

CDA ($A.r \leftarrow$ b /*input credential*/, g /*affiliation graph*/, v /*tracing role*/)

1. let c be $A.r \leftarrow b$; let *es* be Edges(c);
2. add *es* and g into G_{ld}; /*ld is the identity of local entity*/
3. if $A = ld$ then
4. if b is an entity D and $b = ld$ then
5. send c to D;
6. if b is a scoped role $B.r_1.\square$ and $B = ld$ then
7. pull $G_B^{B.r_1}$ from B;
8. send c to B;
9. if b is a scoped role intersection $B_1.r_1.s_1 \wedge B_2.r_2.s_2$ then
10. for each i in {1, 2} do if $s_i = \square$ and $B_i = ld$ then
11. pull $G_{B_i}^{B_i.r_i}$ from B_i;
12. send c to B_i;
13. for each edge $A'.r' \xleftarrow{w} n$ in G_{ld} do
14. if n *contains* v and $A' = ld$ then
15. if ($w = \square$) or ($w = \varnothing$ and $n \xleftarrow{\square} v \in G_{ld}$) then
16. if $A = ld$ then call $B.cda(c, G_{ld}^n, A'.r')$; /*calls remote CDA*/
17. else call $B.cda(c, g, A'.r')$; /*calls remote CDA*/
18. return;

CDA stores each credential at its issuers and subjects, because subjects must know their privileges being assigned, and issuers must know the security policies being configured. Technically, the credentials stored at subjects will enable the local backward tracking in delegation network.

Lemma 4. Let e be the edge $A.r \xleftarrow{\square} B.r_b$ and $e \in G_A|_\tau$, if $B.r_b \leftarrow C.r_c \in G_B|_\tau$ then $B.r \leftarrow C.r_c \in G_A|_\tau$, where no CDA instances are running on A and B at τ.

Lemma 5. Let v be an intersection contains $B.r_b.\square$ and there is an edge from v to $A.r$ in $G_{A|\tau}$, if $B.r_b \leftarrow C.r_c \in G_{B|\tau}$ then $B.r_b \leftarrow C.r_c \in G_{A|\tau}$, where no CDA instances are running on A and B at τ.

Given entities A and B, we use $\Sigma_{A:B}$ to denote the union of Σ_A and Δ_B, and $G_{A:B}$ to denote the credential graph of $\Sigma_{A:B}$. Lemma 4 and 5 can be used to prove the soundness and completeness of CDA, with respect to the semantics of ScoRT.

Theorem 2 (Soundness and Completeness of CDA). Given an entity D and a role $A.r$, then $\Sigma_{A:D}|\tau \hookrightarrow \text{mem}(D, A.r)$ if and only if $\Sigma|\tau \hookrightarrow \text{mem}(D, A.r)$, where Σ is the set of all credentials and no CDA instances are running on A and D at τ.

Given entities A and B, if B requests the resources controlled by A. By theorem 2, A need not search any delegation credentials to make sound and complete authorization decisions. But the theorem assumes that Δ_B (all type-1 credentials issued to B) should be available to A, and retrieval of Δ_B is beyond CDA.

3.2 Credential Revocation

Two kinds of revocation can be provided in ScoRT: revocation on expiration and revocation on demands. Expiration time is the most efficient method for credential revocation, especially for short-term credentials. ScoRT can use revocation messages to enable revocation on demands. Given a credential c of the form $A.r \leftarrow b$, its revocation message can be denoted as $A.r \nleftarrow b$, which is also signed by A. The entity who receives the message $A.r \nleftarrow b$ will delete the credential $A.r \leftarrow b$ from its local credential repository. Revocation of the credential c involves the following four operations:

1. ScoRT stores each authorization credential at its issuer and subject, so if b is an entity, A sends the message $A.r \nleftarrow b$ to A and b.
2. ScoRT stores each delegation credential at its issuer and subject, so if b is not an entity, A sends the message $A.r \nleftarrow b$ to A and the subject entities in b.
3. ScoRT stores the affiliation graphs of each role at its defining entity, so if b is not an entity, A sends the message $A.r \nleftarrow b$ to entities that uses $A.r$ to define credentials.
4. The affiliation graph should be deleted from the local credential repositories if its root is isolated from all roles defined by the local entity.

4 Complexity Estimation

Affiliation graphs will be transferred among entities in credential distribution processes. We have shown that the worst space complexity of affiliation graphs is liner to the number of delegation credentials. This section gives more practical analysis on complexities of affiliation graphs and CRA algorithm based on the branching matrix model of layered delegation networks [1], which can be formally defined as follows:

$$\text{DN} = (F, B, R)$$

where F is an n-dimensional forward branching matrix, B is an n-dimensional backward branching matrix, R is an n-dimensional vector and $R^{(i)} \cdot F^{(j,i)} = R^{(j)} \cdot B^{(i,j)}$. Let G_{DN} be a credential graph which complies with DN:

- $R^{(i)}$ is the number of entity nodes and role nodes at layer i in G_F;
- $F_{n-1}^{(i,j)}$ is the average of trust edges from a node in layer i to layer j, and $F^{(i,n)}$ is the average of authorization edges from a node in layer i to layer n;
- $B_{n-1}^{(i,j)}$ is the average of trust edges to a node in layer i from layer j, $B^{(n,j)}$ is the average of authorization edges to a node in layer n from layer j;

where F_{n-1} is a matrix with n-1 dimensions and $F_{n-1}^{(i,j)} = F^{(i,j)}$, $i, j \in [1, n\text{-}1]$. Let $A.r$ be the role node at layer 1 in G_{DN}, we use $G_{DN}^{A.r}$ to denote the credential graph for $A.r$ and sizeof($G_{DN}^{A.r}$) to denote the number of credentials in $G_{DN}^{A.r}$. By definition 3, sizeof($G_{DN}^{A.r}$) = $|E_{DN}^{A.r}.\mathrm{TE}| - |E_{DN}^{A.r}.\mathrm{IE}|$, where $G_{DN}^{A.r}$ is the edge set of $G_{DN}^{A.r}$. Apparently sizeof($G_{DN}^{A.r}$) is bounded by $|E_{DN}^{A.r}.\mathrm{TE}|$ which can be computed by the following equation:

$$|E_{DN}^{A.r}.\mathrm{TE}| = V_n \cdot (I_n + \Sigma_{k=1}^{n-1} F^k) \cdot U_n^T \tag{1}$$

where V_n is an n-dimensional unit row vector, I_n is an n-dimensional identity matrix, U_n is an n-dimensional row vector and $U_n^{(1)} = 1$ and $U_n^{(i)} = 0$ for $i \in [2, n]$. The equation shows that the worst complexity of affiliation graphs may increase exponentially on the scale of delegation networks. However, delegation networks in practical systems usually have specific structure models [1] with acceptable costs.

Now we analyze the space and communication costs based on two sample delegation networks derived from existing researches [1, 16]. Compared with the delegation structures in the EHR system [3], these sample networks seem quite complex. Given a delegation network DN and a role $A.r$ at layer one in G_{DN}, then $G_{DN}^{A.r}$ only contains the nodes at first n-1 layers. According to equation (1), we can compute the trust edges in $G_{DN}^{A.r}$ which can be reached within specified delegation steps: $\mathrm{AGN}(F_{n-1}, d)$ is the number of trust edges in $G_{DN}^{A.r}$ which can be reached by a legal path from $A.r$ within d steps. Similarly, we use $\mathrm{CGN}(F, d)$ to denote the number of both trust and authorization edges in G_{DN} which can be reached by a legal path from $A.r$ within d steps.

$$\mathrm{AGN}(F_x, d) = V_x \times (I_x + \Sigma_{k=1}^{d} F_x^k) \times U_x^T - 1 \tag{2}$$

$$\mathrm{CGN}(F, d) = V_n \times (I_n + \Sigma_{k=1}^{d} F^k) \times U_n^T \tag{3}$$

where $x \in [0, n\text{-}1]$, $d \in [0, n\text{-}1]$, and because credentials issued to layer 5 are authorization credentials, we have $\mathrm{AGN}(F_{n-1}, n\text{-}1) = \mathrm{AGN}(F_{n-2}, n\text{-}2)$. Usually, practical applications only permit delegation paths with small depths, such as 2 and 3, and the depths of more than 4 are rarely considered [15]. Let u be the upper bound of depths, the length of legal paths in affiliation graphs is u-1 at most. By line 17 in CDA, each affiliation graph being transferred contains at most $\mathrm{AGN}(F_{n-1}, u\text{-}2)$ trust edges. Given a root CDA instance α, by lines 14~16 of CDA, α transfers the affiliation graphs over the edges covered by the backward searching processes in [1], which is $\mathrm{BGN}(B_{n-1}, u\text{-}1)$. Therefore the worst communication cost between $[\alpha^-, \alpha^+]$ is $\mathrm{CDN}(F, B)$:

$$\mathrm{CDN}(F, B) = \mathrm{AGN}(F_{n-1}, u\text{-}2) \times \mathrm{BGN}(B_{n-1}, u\text{-}1) \tag{4}$$

$$\mathrm{BGN}(B_x, d) = V_d \times (I_d + \Sigma_{k=1}^{d} B_x^k) \times Z_d^T - 1 \tag{5}$$

where Z_d is an d-dimensional row vector, $Z_d^{(d)}=1$ and $Z_d^{(i)}=0$ for $i\in[1, d\text{-}1]$. Now we estimate the complexities of affiliation graphs and CDA communication costs based on two sample delegation networks where the upper bound of delegation depth is 4.

CASE1: Consider the sample delegation network defined in [1], as shown in the first three columns of Table 1. The nodes at layer 1 ~ 4 are roles, while the nodes at layer 5 are entities. The credentials from layer i to layer j ($1 \le i \le j \le 4$) are delegation credentials. The credentials issued to the entities at layer 5 are authorization credentials.

Table 1. Cost Estimation based on the Delegation Network in [1]

Amt. of credentials per role (F)		From layer 1	2	3	4	5	Amt. of roles(R)	AGN, CGN, %	CDN(F, B)
To layer	1	0	0	0	0	0	10	0, 0, -	1722 (21×82)
	2	1	2	0	0	0	5	4, 14, 29%	
	3	1	2	2	0	0	2	21, 86, 24%	
	4	2	2	5	2	0	20	83, 418, 20%	
	5	10	5	10	20	0	2000	83, 1678, 4.9%	

By definition of delegation network, the backward branching matrix B is [0, 2, 5, 1, 0.05; 0, 2, 5, 0.5, 0.0125; 0, 0, 2, 0.5, 0.01; 0, 0, 0, 2, 0.2; 0, 0, 0, 0, 0]. Therefore CDN(F, B) = AGN(F_4, 2)×BGN(B_4, 3)=1722. According to the 4th column in table 1, AGN(F_i, i) is relatively small compared with CGN(F, i), $i\in[0, 4]$. The largest affiliation graph in credential repositories is AGN(F_3, 3) with only 83 credentials. The largest affiliation graph to be transferred on network contains only 21 credentials and the upper bound of total credentials to be transferred in one root CDA instance is 1722.

CASE2: Consider the credentials in Example 1. The estimated delegation network is given in Table 2. The layer model is: EPub.discount is at layer 1; EOrg.preferred and ACM.member are at layer 2; StateU.student, RegB.student and Alice are at layer 3, 4, 5 respectively. Similar to the computation process in case 1, CDN(F, B) = 2772. But in case 2, AGN(F_i, i) is a much small portion of CGN(F, i), $i\in[0, 4]$. This is because that the number of authorization credentials in case 2 is practically increased. The worst complexity of affiliation graphs does not exceed 466.

Table 2. Cost Estimation based on the Delegation Network for Example 1

Amt. of credentials per role (F)		From layer 1	2	3	4	5	Amt. of roles (R)	AGN / CGN / %	CDN(F, B)
To layer	1	0	0	0	0	0	10	0, 0, -	2772 (231×12)
	2	5	0	0	0	0	5	46, 146, 3.2%	
	3	20	10	0	0	0	50	231, 22831, 1.0%	
	4	20	15	2	1	0	500	466, 163066, 0.29%	
	5	100	100	100	1000	0	20000	466, 398301, 0.12%	

In practical systems, a credential mainly contains two keys (1024 bits each) and at most three role names (256 bits each). Together with accessories such as time and tags, the size of one credential can be around 1K bytes. Therefore, the worst communication costs in above two cases are about 1,722K bytes and 2,772K bytes respectively.

5 Related Work

Delegation-based authorization complicates the distribution and retrieval of credentials in decentralized TM systems. This section gives more detailed analysis and comparison of related works in this area.

RT [16, 17] is an influential role-based TM system and deeply studies the credential discovery problem. Li et al designs a graph-based distributed discovery algorithm and a type system to define storage policies. However, the credentials storage policies are difficult to configure which can only be tackled by experts, and the linked roles in RT make the algorithm log-space P-complete. The graph model in [16] does not define edges between intersections and roles, and can not be used to define affiliation graphs in ScoRT. The derived intersection edges in [16] are denoted by the weights of intersections edges in our graph model, which provides more intuitionistic graphical interfaces for security administrators. Furthermore, ScoRT provides delegation control with depth 1, 2 and ∞, where the depth 2 is enabled by the trust scope tag $\blacksquare$, which covers a wide range of practical delegation scenarios.

Aura firstly studies the graph-based methods for credential searching in SPKI certificate databases [1] and suggests that backward searching usually perform much faster than forward searching. Aura proposes the branch matrices model for practical delegation structures, which are used in this paper to evaluate the complexity of our approach. Based on Aura' work, a DNS-based storage scheme for SPKI certificates and certificate retrieval algorithms are proposed [13]. They distinguish four kinds of SPKI certificates and the certificates are stored by issuer sites or subject sites according to their types.

QCM [12] was the first TM system to consider automated credential retrieval: if QCM engine is not given the required credential, the system will retrieve it from the specified server. It provides a language-based framework for automatic retrieval of certificates and distribution of revocation information. SD3 [14] is a successor of QCM based on distributed Datalog which supports certified evaluation and recursive policies. Cassandra [2] borrows the credential retrieval mechanism in SD3 and provides various trust negotiation and credential retrieval strategies.

Table 3 shows an overall comparison with existing approaches. Here clients are resource requesters, and servers are authorizers that provide resources. We distinguish four kinds of credential retrieval models. The client-push and server-pull models are similar to the traditional push and pull models in PMI [11]. ScoRT introduces the server-push model in which servers push credentials to other servers according to credential storage policies. Some TM systems support client-pull model where clients retrieve credential chains from network before push them to servers.

Table 3. Credential Retrieval Models for Distributed Authorization

Retrieval Models	SPKI-DNS	QCM	RT	Cassandra	ScoRT
client-pull	Yes/AC		Yes/AC		
client-push	Yes/AC		Yes/AC		Yes/AC
server-pull	Yes/AC	Yes/AC	Yes/AC	Yes/AC	Yes/SA
server-push				Yes/AC	Yes/SA

We argue that the authorization tasks can be divided into two stages: access control (AC) stages and security administration (SA) stages. AC stages are usually performance sensitive because AC decisions are used to reply login requests. While at SA stages, the reasonable delay caused by security configuration are commonly accepted. ScoRT moves the credential retrieval tasks from AC stages to SA stages and helps to lift the overall system performance. Furthermore, ScoRT can be extended to enforce various credential privacy policies in CDA at SA stages.

6 Conclusion

This paper initiates the research on decentralized authorization that eliminates the process of searching credentials from networks when making access control decisions. The main contributions of our work include: (1) a scoped-role based TM system named ScoRT with more refined delegation control is proposed, which can be regarded as a generalized extension of both role-based and capability-based TM systems; (2) a novel weighed graph-based credential affiliation model is proposed to guide the distributed storage, retrieval and revocation of credentials, which can provide more friendly graphical UIs; (3) a distributed credential management framework is proposed to enable decentralized authorization without searching credentials from network when making access decisions, and thus greatly increases the system performance; (4) first attempt to use Aura's delegation network model to analyze the space complexities of distributed credential management processes, and the preliminary analysis results show that the costs of our approach are mainly acceptable. Further research on ScoRT are necessary, such as fault tolerance, retrieval of authorization credentials, privacy in credential retrieval, and more effective revocation mechanism that can work consistently with CDA.

Acknowledgement

This research is supported by the National 863 Grand project 2007AA010301 and the National 973 Basic Research program 2005CB321800.

References

1. Aura, T.: Fast access control decisions from delegation certificate databases. In: Boyd, C., Dawson, E. (eds.) ACISP 1998. LNCS, vol. 1438, pp. 284–295. Springer, Heidelberg (1998)

2. Becker, M.Y., Sewell, P.: Cassandra: Flexible Trust Management, Applied to Electronic Health Records. In: Proceedings of the 17th IEEE Computer Security Foundations Workshop (2004)
3. Becker, M.Y.: A formal security policy for an NHS electronic health record service. UCAM-CL-TR 628, University of Cambridge, Computer Laboratory, p. 81 (March 2005)
4. Becker, M.Y., Fournet, C., Gordon, A.D.: Design and Semantics of a Decentralized Authorization Language. In: 20th IEEE Computer Security Foundations Symposium, pp. 3–15 (2007)
5. Blaze, M., Feigenbaum, J., Lacy, J.: Decentralized trust management. In: Proceedings of the 1996 IEEE Symposium on Security and Privacy, pp. 164–173. IEEE Computer Society Press, Los Alamitos (1996)
6. Blaze, M., Feigenbaum, J., Strauss, M.: Compliance-checking in the PolicyMaker trust management system. In: Hirschfeld, R. (ed.) FC 1998. LNCS, vol. 1465, pp. 254–274. Springer, Heidelberg (1998)
7. Blaze, M., Feigenbaum, J., Ioannidis, J., Keromytis, A.D.: The KeyNote trust-management system, version 2. IETF RFC 2704 (September 1999)
8. Clarke, D., Elien, J.E., Ellison, C., Fredette, M., Morcos, A., Rivest, R.L.: Certificate chain discovery in SPKI/SDSI. Journal of Computer Security 9(4), 285–322 (2001)
9. Elley, Y., Anderson, A., Hanna, S., Mullan, S., Perlman, R., Proctor, S.: Building certification paths: Forward vs. reverse. In: Proceedings of the 2001 Network and Distributed System Security Symposium (NDSS 2001), pp. 153–160. Internet Society (February 2001)
10. Ellison, C., Frantz, B., Lampson, B., Rivest, R., Thomas, B., Ylonen, T.: SPKI certificate theory. IETF RFC 2693 (September 1999)
11. Farrell, S., Housley, R.: An Internet Attribute Certificate Profile for Authorization, RFC3281 (April 2002)
12. Gunter, C., Jim, T.: Policy-directed certificate retrieval. Software: Practice & Experience 30(15), 1609–1640 (2000)
13. Hasu, T., Kortesniemi, Y.: Implementing an SPKI Certificate Repository within the DNS. In: International Workshop on Public-Key Cryptography, PKC (2000)
14. Jim, T.: SD3: A trust management system with certified evaluation. In: Proceedings of the 2001 IEEE Symposium on Security and Privacy, pp. 106–115. IEEE Computer Society Press, Los Alamitos (2001)
15. Li, N.: Delegation Logic: A Logic-based Approach to Distributed Authorization. PhD thesis, New York University, New York (2000)
16. Li, N., Winsborough, W.H., Mitchell, J.C.: Distributed credential chain discovery in trust management (extended abstract). In: Proceedings of the Eighth ACM Conference on Computer and Communications Security (CCS-8), pp. 156–165. ACM Press, New York (2001)
17. Li, N., Mitchell, J.C., Winsborough, W.H.: Design of a role-based trust management framework. In: Proceedings of the 2002 IEEE Symposium on Security and Privacy, pp. 114–130. IEEE Computer Society Press, Los Alamitos (2002)
18. Mao, Z., Li, N., Winsborough, W.H.: Distributed Credential Chain Discovery in Trust Management with Parameterized Roles and Constraints. In: Ning, P., Qing, S., Li, N. (eds.) ICICS 2006. LNCS, vol. 4307, pp. 159–173. Springer, Heidelberg (2006)
19. Nilsson, U., Małuszyński, J.: Logic, Programming and Prolog, 2nd edn. John Wiley & Sons Ltd., Chichester (1995)

Appendix A

A. 1 Proof of Lemma 1

Proof. We first prove the if part. Do induction on the length k of $A.r \twoheadleftarrow D$. If k is one, then $A.r \longleftarrow D \in G_\Sigma$ and $A.r \leftarrow D \in \Sigma$. By ScoRT semantics, $\Sigma \hookrightarrow \text{mem}(D, A.r)$ is true. If k is greater than one, suppose n is the node in $A.r \twoheadleftarrow D$, then there is an edge $A.r \xleftarrow{w} n$ in $A.r \twoheadleftarrow D$ and the length of $n \twoheadleftarrow D$ is k-1. By definition of credential graph, n is either a role or a intersection. If n is a role, by induction assumption, $\Sigma \hookrightarrow \text{mem}(D, n)$ and thus $P_\Sigma \vdash \text{m}(D, n, \square)$. Because there is an edge $A.r \xleftarrow{w} n$ in G_Σ, the credential $A.r \leftarrow n.s$ must be in Σ and thus P_Σ contains the rule $\text{m}(x, A.r, \square) \leftarrow \text{m}(x, n, w)$. If w is $\square$ then $P_\Sigma \vdash \text{m}(D, A.r, \square)$ and $\Sigma \hookrightarrow \text{mem}(D, A.r)$; otherwise if w is $\blacksquare$, because $A.r \twoheadleftarrow D$ is a path in G_Σ, then k must equal to 2, therefore $\Sigma \hookrightarrow \text{mem}(D, A.r)$ follows too. If n is a intersection $n_1 \wedge n_2$, then G_Σ contains the edge $A.r \xleftarrow{w} n$ and the paths $n_1 \twoheadleftarrow D$ and $n_2 \twoheadleftarrow D$ where $D \in w$, and thus P_Σ contains the rule $\text{m}(x, A.r, \square) \leftarrow \text{m}(x, n_1, s_1), \text{m}(x, n_2, s_2)$. By the induction assumption, $\Sigma \hookrightarrow \text{mem}(D, n_1)$ and $\Sigma \hookrightarrow \text{mem}(D, n_2)$. Similar to the case when n is a role, $\Sigma \hookrightarrow \text{mem}(D, A.r)$ can be proved.

Now we consider the only if part. By $\Sigma \hookrightarrow \text{mem}(D, A.r)$, $P_\Sigma \vdash \text{m}(D, A.r, \square)$ follows. There is a sequence of proof steps for $\text{m}(D, A.r, \square)$. Do induction on the length s of the proof steps. If s equals one, then the rule R1 is used in the proof and the edge $A.r \longleftarrow D$ is in G_Σ. $A.r \twoheadleftarrow D \in G_\Sigma$ follows. If s is greater than one, suppose α is the rule used at the last proof step which is generated by T2 or T3. If α is a T2 rule, i.e. α is $\text{m}(x, A.r, \square) \leftarrow \text{m}(x, B.r_1, s)$, then $P_\Sigma \vdash \text{m}(x, B.r_1, s)$ and by induction assumption $B.r_1 \twoheadleftarrow D \in G_\Sigma$. If s is $\square$, then $A.r \xleftarrow{\square} B.r_1$ is in G_Σ and $A.r \twoheadleftarrow D \in G_\Sigma$. Otherwise, if s is $\blacksquare$ then $A.r \xleftarrow{\blacksquare} B.r_1$ is in G_Σ. Because $P_\Sigma \vdash \text{m}(D, A.r, \square)$, then there must be a rule $\text{m}(D, B.r_1, \blacksquare)$ in P_Σ and the edge $B.r_1 \xleftarrow{\blacksquare} D$ is in G_Σ. Therefore $A.r \twoheadleftarrow D \in G_\Sigma$. If α is a T3 rule, i.e. α is $\text{m}(x, A.r, \square) \leftarrow \text{m}(x, B_1.r_1, s_1), \text{m}(x, B_2.r_2, s_2)$, then $P_\Sigma \vdash \text{m}(x, B_1.r_1, s_1)$ and $P_\Sigma \vdash \text{m}(x, B_2.r_2, s_2)$. By induction assumption, $B_1.r_1 \twoheadleftarrow D \in G_\Sigma$ and $B_2.r_2 \twoheadleftarrow D \in G_\Sigma$ are true. Similar to the case of T2 rule, $A.r \twoheadleftarrow D \in G_\Sigma$ follows.

A. 2 Proof of Lemma 2

Proof. Let ξ be the legal path $A.r \twoheadleftarrow n$ in G_Σ. Do induction on the length s of ξ. When s is one, ξ is an edge. By line 3 and 4 of AffG, ξ will be added into $G_\Sigma^{A.r}$. When s is greater than one, then there is a node n' in ξ. Let e be the edge $n' \xleftarrow{w} n$ in ξ. Consider the path $A.r \twoheadleftarrow n'$ which is a sub-path of ξ. By induction assupmtion, all edges in $A.r \twoheadleftarrow n'$ are in $G_\Sigma^{A.r}$. Now we prove e is also in $G_\Sigma^{A.r}$. Let e' be $n'' \xleftarrow{w'} n'$ which is the last edge in $A.r \twoheadleftarrow n'$. Because $\xi \in G_\Sigma$, w' is either $\square$ or $\varnothing$. By definition of AffG, e' can only be added by line 4 or line 7. In the first case, if w' is $\square$, then AffG(n', ns, es) will add e into $G_\Sigma^{A.r}$; if w' is $\varnothing$, then e will also be added by line 7. In the second case, w' must be $\square$ and AffG(n', ns, es) will add e into $G_\Sigma^{A.r}$.

A. 3 Proof of Lemma 3

Proof. The if part is obvious because $G_{\Sigma}^{A.r:D}$ is a sub-graph of G_{Σ}. We now prove the only if part. For each edge e in $A.r \twoheadleftarrow D$ where e is $n \xleftarrow{w} n'$, if n is a role, by definition of legal paths, $A.r \twoheadleftarrow n \in G_{\Sigma}$. By lemma 2, e is in $G_{\Sigma}^{A.r}$. If n is a intersection $n_1 \wedge n_2$, there must be a role n'' that $n'' \xleftarrow{w'} n$ is in $A.r \twoheadleftarrow D$ and $A.r \twoheadleftarrow n'' \in G_{\Sigma}$. By definition of legal paths, $A.r \twoheadleftarrow n_1 \in G_{\Sigma}$ and $A.r \twoheadleftarrow n_1 \in G_{\Sigma}$ are true. By lemma 2, the edges $n'' \xleftarrow{w'} n$, $n \xleftarrow{s_1} n_1$ and $n \xleftarrow{s_2} n_2$ are in $G_{\Sigma}^{A.r}$, and e is in $G_{\Sigma}^{A.r}$ because e is either $n \xleftarrow{s_1} n_1$ or $n \xleftarrow{s_2} n_2$. Therefore $A.r \twoheadleftarrow D \in G_{\Sigma}^{A.r:D}$ is true.

A. 4 Proof of Lemma 4

Proof. Let ξ be the path $B.r_b \twoheadleftarrow C.r_c$ and $n_1, \ldots, n_s$ are nodes appearing in ξ along the reverse direction of the path where n_1 is $B.r_b$, n_s is $C.r_c$ and s is greater than one. Consider two CDA instances α and β: α is a root instance where $e \notin G_A|_{\alpha^-}$ and $e \in G_A|_t$, $\alpha^+ \le t \le \tau$; β is a root instance where $\xi \notin G_B|_{\beta^-}$ and $\xi \in G_B|_t$, $\beta^+ \le t \le \tau$. By definition of β, there exits a path $n_1 \twoheadleftarrow n_i$ in $G_B|_{\beta^-}$ and there is no edge from n_{i+1} to n_i, $0 < i < s$. Therefore β stores $n_i \xleftarrow{w} n_{i+1}$ into $G_B|_{\beta^+}$ and n_i is a role[1]. If α is the same instance as β, α will retrieve $n_i \xleftarrow{w} n_{i+1}$ from B by line 6. which is in contradiction to $n_i \xleftarrow{w} n_{i+1} \in G_B|_{\beta^+}$, therefore α is not β. Now we need to prove that $B.r \twoheadleftarrow C.r_c \in G_A|_\tau$ is true in both cases: α runs before β and α runs after β.

Let c be the credential $A.r \leftarrow B.r_b.\square$. Do induction on s and consider the base case when s is 2, then ξ is a trust edge and let c' be the credential $B.r_b \leftarrow C.r_c.s_c$. In the first case, α stores c at A and send it to B by line 8. And then β stores c' at B and uses line 16 to send c' to A, and thus Edges(c') is at A at time τ. In the second case, β firstly stores Edges(c') at B, then α stores Edges(c) at A and use line 7 to pull $G_B^{B.r_b}$ from B. By lemma 2, Edges(c')$\in G_B^{B.r_b}$. Therefore, $B.r \twoheadleftarrow C.r_c \in G_A|_\tau$ in both cases.

Consider the induction step where s is greater than 2. In the first case, $\alpha^+ < \beta^- \le \tau$, α stores c at A and send it to B by line 8, i.e., $e \in G_A|_{\alpha^+}$. By $n_1 \twoheadleftarrow n_i \in G_B|_{\beta^-}$ and $e \in G_A|_{\beta^-}$, $B.r \twoheadleftarrow n_i \in G_A|_{\beta^-}$ is true by induction assumption. Because β stores $n_i \twoheadleftarrow n_s$ into $G_B|_{\beta^-}$, there must be a instance β' running on B that uses line 1 to add $n_i \twoheadleftarrow n_s$ into $G_B|_{\beta^+}$. Because $e \in G_A|_{\alpha^+}$, β' will use line 16 or 17 to send $G_B^{B.r_b}$ to A. By lemma 2, $n_i \twoheadleftarrow n_s \in G_B^{B.r_b}$. In the second case, $\beta^+ < \alpha^- \le \tau$, β firstly stores $n_1 \twoheadleftarrow n_s$ at B, then α uses line 7 to pull $G_B^{B.r_b}$ from B. By lemma 2, $n_1 \twoheadleftarrow n_s \in G_A^{B.r_b}|_{\alpha^+}$. Therefore $B.r \twoheadleftarrow C.r_c \in G_A|_\tau$ is true in both cases.

A. 5 Proof of Lemma 5

Proof. Let e be the intersection edge from v to $A.r$. Let ξ be the path $B.r_b \twoheadleftarrow C.r_c$ and $n_1, \ldots, n_s$ are nodes appearing in ξ along the reverse direction of the path where n_1 is $B.r_b$, n_s

[1] Otherwise n_i is an intersection $e_{i1}.s_{i1} \wedge e_{i2}.s_{i2}$, and n_{i+1} is either e_{i1} or e_{i2}; by lines 1 and 2 of CDA, there are two edges from e_{i1} and e_{i2} to n_i in $G_B|_{\beta^-}$ which is paradoxical because there is no edges from n_{i+1} to n_i.

is $C.r_c$ and s is greater than one. Consider two CDA instances α and β: α is a root instance where $e \notin G_A|_{\alpha^-}$ and $e \in G_A|_t$, $\alpha^+ \le t \le \tau$; β is a root instance where $\xi \notin G_B|_{\beta^-}$ and $\xi \in G_B|_t$, $\beta^+ \le t \le \tau$. By definition of β, there exits a path $n_1 \leftarrow n_i$ in $G_B|_{\beta^-}$ and there is no edge from n_{i+1} to n_i, $0<i<s$. Therefore $n_i \xleftarrow{w} n_{i+1} \in G_B|_{\beta^+}$. By similar analysis in lemma 4, n_i must be a role and α is the different instance from β. Now we need to prove that $B.r \leftarrow C.r_c \in G_A|_\tau$ is true in both cases: α runs before β and α runs after β.

Let c be the credential for e. Do induction on s and consider the base case when s is 2, then ξ is a trust edge and let c' be the credential $B.r_b \leftarrow C.r_c.s_c$. In the first case, α stores c at A and send it to B by line 12. And then β stores Edges(c) at B and uses line 16 to send c to A, and thus Edges(c') is at A at time τ. In the second case, β firstly stores Edges(c') at B, then α stores c' at A and use line 11 to pull $G_B^{B.r_b}$ from B. By lemma 2, Edges(c')$\in G_B^{B.r_b}$. Therefore, $B.r \leftarrow C.r_c \in G_A|_\tau$ in both cases.

Consider the induction step where s is greater than 2. In the first case, α stores Edges(c) at A and send c to B by line 12. By $n_1 \leftarrow n_i \in G_B|_{\beta^-}$ and $e \in G_A|_{\beta^-}$, $B.r \leftarrow n_i \in G_A|_{\beta^-}$ is true by induction assumption. Because β stores $n_i \leftarrow n_s$ into $G_B|_{\beta^-}$, there must be a instance β' running on B that uses line 1 to add $n_i \leftarrow n_s$ into $G_B|_{\beta^+}$. Because $e \in G_A|_{\alpha^+}$, β' will use line 16 or 17 to send $G_B^{B.r_b}$ to A. By lemma 2, $n_i \leftarrow n_s \in G_B^{B.r_b}$. In the second case, $\beta^+ < \alpha^- \le \tau$, β firstly stores $n_1 \leftarrow n_s$ at B, then α uses line 7 to pull $G_B^{B.r_b}$ from B. By lemma 2, $n_1 \leftarrow n_s \in G_A^{B.r_b}|_{\alpha^+}$. Therefore $B.r \leftarrow C.r_c \in G_A|_\tau$ is true in both cases.

A. 6 Proof of Theorem 2

Proof. $\Sigma|_\tau \hookrightarrow \mathrm{mem}(D, A.r)$ follows from $\Sigma_A|_\tau \hookrightarrow \mathrm{mem}(D, A.r)$ since $\Sigma_A \subseteq \Sigma$. We only need to prove the if part. By lemma 1 we need to prove $A.r \leftarrow D \in G_{A:D}|_\tau$ if $A.r \leftarrow D \in G_\Sigma|_\tau$.

Do induction on the length k of $A.r \leftarrow D$. If k equals one, then $A.r \leftarrow D$ is $A.r \longleftarrow D$. $A.r \leftarrow D \in G_{A:D}|_\tau$ follows. If k equals two, $A.r \leftarrow D$ can be denoted as $A.r \xleftarrow{w} B.r_b \longleftarrow D$. By $A.r \leftarrow D \in G_\Sigma|_\tau$, there must be a root CDA instance that adds $A.r \xleftarrow{w} B.r_b$ into G_A before τ. Therefore $A.r \leftarrow D \in G_{A:D}|_\tau$ is true.

Consider the induction step when k is greater than two. Suppose n and n' are nodes in $A.r \leftarrow D$, where $A.r \xleftarrow{w} n$ and $n' \longleftarrow D$ are two edges in $G_A|_\tau$, then the length of $n \leftarrow D$ is k-1. By definition of legal paths, w must be $\Box$. There must be a root CDA instance that uses line 2 to add Edges($A.r \leftarrow n$) into $G_A|_\tau$ and thus $A.r \xleftarrow{w} n$ is in $G_A|_\tau$. By definition of credential graph, n can be a role or an intersection. If n is a role $B.r_b$, by induction assumption we have $B.r_b \leftarrow D \in G_{B:D}|_\tau$. By lemma 3, $B.r \leftarrow n' \in G_A|_\tau$ is true and therefore $A.r \leftarrow D \in G_{A:D}|_\tau$. If n is an intersection $B.r_b \wedge B'.r_{b'}$, then there are two edges $n \xleftarrow{w_1} B.r_b$ and $n \xleftarrow{w_2} B'.r_{b'}$ in $G_\Sigma|_\tau$. Consider paths $B.r_b \leftarrow n'$ and $B'.r_{b'} \leftarrow n'$ in $G_\Sigma|_\tau$, by induction assumption, $B.r_b \leftarrow D \in G_{B:D}|_\tau$ and $B'.r_{b'} \leftarrow D \in G_{B':D}|_\tau$. By lemma 4, we can easily prove that both $B.r_b \leftarrow D$ and $B'.r_{b'} \leftarrow D$ are in $G_{A:D}|_\tau$. Therefore $A.r \leftarrow D \in G_{A:D}|_\tau$ follows.