Nonmonotonicity in Trust Management

Wojciech Pikulski

Nonmonotonicity in Trust Management

Pikulski Wojciech

Institute of Control and Computation Engineering
Warsaw University of Technology, Warsaw, Poland
w.pikulski@elka.pw.edu.pl

Abstract. The work discusses nonmonotonicity in terms of trust management systems and presents model allowing for credential revocation in the Role-based Trust-management Framework. A freshness constraints have been adopted into RT Framework in order to overcome nonmonotonicity and turn it to be temporarily monotonic. The proposed model allows for freshness requirements specification on policy level and utilises freshness graph in order to perform propagation of freshness requirements along credential chains. Finally, an evaluation of the model against real-life scenario has been performed.

Keywords: Software security, trust management, RT Framework, nonmonotonicity, credential chain, credential graph, freshness constraints

1 Introduction

Distributed systems have become very popular during the last decade. Due to the expansion of the Internet access in the last few years, they have also become open to the external environment. Global market and economy made companies set partnerships which often need infrastructure that can provide access to the shared resources and supply collaboration services.

Distributed design meets many issues that have to be solved in order to enable application collaboration. Some of the most significant are security of shared resources and provisioning of convenient access control mechanisms in distributed environment, where subjects and resources are situated and operate in different security domains also called administrative domains. Each of such domain controls its subjects and resources.

To solve problems listed above, various access control models have been proposed. Security mechanisms dedicated for utilisation in distributed environment are named Trust Management. Researchers have also focused on creating Automated Trust Negotiation mechanism that can establish trust between strangers in automated way.

This work is focused on nonmonotonicity issues in trust management models. The monotonicity of the security model means that when access decision evaluates to true at some point in time, it cannot evaluate to false in future when time lapses or new information is added to policy. Model can introduce nonmonotonicity when it allows for credential revocation or role negation. Currently, nonmonotonicity is regarded as

undesirable [4][11]. However, in [6] authors proved that certificate revocation, that used to be thought as nonmonotonic, can be implemented in monotonic manner by introducing freshness constraints. This paper analyses the nonmonotonicity caused by credential revocation. The work aims at introducing credential revocation in the Role-based Trust-management Framework. This goal is achieved by creating the model of freshness constraints propagation. The paper presents its formal description and evaluates the idea against real-life scenario.

Because certificates and credentials are electronically signed documents containing digital information, the freshness constraints can be implemented for both. As certificates contain binding of an entity name to its private key, they are entity's electronic identity. Credentials contain signed policy statements defining permissions.

The rest of the paper is outlined as follows. Section 2 describes related work. In section 3 trust management is defined. Section 4 presents business problem which would incorporate credential revocation. The RT Framework and freshness constraint propagation model is presented and discussed in Section 5. Finally, in section 6 the paper is summed up and future work is outlined.

2 Related Work

A significant effort has been carried out to address problem of trust in distributed systems. There were presented many models for authorisation in such environment. Few examples are KeyNote [2], PolicyMaker [1] and RT Framework [7]. Seamons et. al. in [11] and Chapin et. al. in [4] point out the requirements for trust management systems and perform survey of proposed models against them. One of the commonly agreed requisite is monotonicity of the model, which is regarded to be one of the factors allowing a practical implementation of the system. However, Li et. al. in [6] proved that certificate revocation, which was thought to cause system nonmonotonic, can be implemented in monotonic manner. Skalka et. al. [12] created RT^R language which takes into consideration a risk associated with each credential. The risk is being propagated along credential chains. The notion is conceptually similar to the idea of freshness constraints propagation presented in this work. Changyu et. al. in [15] introduce a non-monotonic trust management model called Shinren. The concept is different from RT framework analysed in the paper in the sense that Shinren utilises multi-value logic with negative assertions in contrast to RT, which uses classic logic with only positive statements. Furthermore, presented freshness constraints propagation focuses on allowing for credential revocation instead of expressing negative information in the security policy.

Another subject of research in the area is automated trust negotiation. Seamons et. al. in [11] present the requirements for languages used to specify policies and for compliance checkers which parse the policies and make access decisions. The idea of automated trust negotiation is elaborated in [10] where authors present negotiation strategies and policy graphs.

3 Access Control and Trust Management

The aim of access control is to prevent unwanted users from acquiring access to a shared resource [9]. Trust management is an evolution of access control for distributed systems. A decentralised design brings new problems and imposes additional requirements for access control mechanisms. The main difference is that subjects and resources can belong to various administrative domains. Another issue is that the credentials are decentralised and its number is virtually unlimited. The problem of creating access control for such environment is referred as trust management, and was first introduced in [3].

Trust management research mainly focuses on two problems: authorisation and trust negotiation. The former is answering a question whether a particular resource access request should be allowed or denied. To achieve this models utilise credentials defining subject permissions and inferencing rules that allow to evaluate an access decision. Trust negotiation is a newer problem, whose point is to establish trust between two stranger parties in automated manner. It is accomplished by a bidirectional conversation with gradual exchange of owned credentials. Each party has a policy defining for each credential what credentials have to be revealed by the negotiating partner in order to be disclosed. Details can be found in [3][10][11][13].

4 Business Problem

This paper describes a problem of nonmonotonocity caused by credential revocation. The issue is illustrated with a real-life scenario adopted from [5]. As work is focused on the RT Framework, policies are presented in form of RT credentials.

The example demonstrates how nonmonotonicity is introduced into RT Framework when it would allow for credential revocation. In such situation, each credential could be revoked before its expiry time passes. As a consequence an unaware acceptor would accept revoked credential. In the example, an eStore offers discount both to its long-standing customers and also to students who are simultaneously members of Superior Mountaineering Club (SMC). The student role definition is delegated to the Accrediting Board for Universities and Schools (ABUS) which defines it for universities and schools appropriately. Sample credentials are listed below:

$$\text{eStore.discount} \longleftarrow \text{eStore.discountEligible} . \quad (1)$$

$$\text{eStore.discountEligible} \longleftarrow \text{eStore.longStandingCustomer} . \quad (2)$$

$$\text{eStore.longStandingCustomer} \longleftarrow \text{John} . \quad (3)$$

$$\text{eStore.discountEligible} \longleftarrow \text{eStore.student} \cap \text{SMC.member} . \quad (4)$$

$$\text{eStore.student} \longleftarrow \text{ABUS.university.student} . \quad (5)$$

$$\text{eStore.student} \longleftarrow \text{ABUS.school.pupil} . \quad (6)$$

$$\text{ABUS.university} \longleftarrow \text{StateU} . \quad (7)$$

$$\text{StateU.student} \longleftarrow \text{StateU.faculty.student} . \quad (8)$$

$$\text{StateU.faculty} \longleftarrow \text{IT} . \quad (9)$$

$$\text{IT.student} \longleftarrow \text{Adam} . \quad (10)$$

$$\text{SMC.member} \longleftarrow \text{Adam} . \quad (11)$$

Evaluation of above credentials leads to conclusion that both Adam and John are eligible for discount. The evaluation process is described in Section 5 and [8].

In real-life scenario, it would be reasonable if security model would allow to revoke credentials. For example, if Adam does not meet semester requirements, his student credential can be revoked. Similar situation would happen if Adam resigns or is expelled from SMC. In any of both cases, eStore would unwittingly accept revoked credentials and grant discount.

5 Nonmonotonicity in Trust Management

Monotonicity of a security model is defined as a feature that if an access decision evaluates to true at some point in time, it should still be true if time lapses or additional credentials are introduced into the policy. There are two types of nonmonotonocity. If time lapse makes access decision false, system is nonmonotonic in "time domain". Such situation can happen when model allows for credential revocation, as revoking credential can cause that user will no longer be member of specific role. When addition of new credentials makes access decision false, system is nonmonotonic in "system size domain". This can happen when model allows to negate roles in policy. When user becomes a member of a negated role, he loses access to a resource. This paper presents the analysis of former type of nonmonotonicity leaving the latter for future research.

5.1 Time Domain Nonmonotonicity

Li and Feigenbaum in [6] proved that the certificate revocation can be implemented in monotonic manner. Thus, time domain nonmonotonicity is turned to be monotonic. Typically, certificates are interpreted as: "valid from their issue time t_0 to their expiry time t_{ex}". When system supports certificate revocation, a certificate can be cancelled before its expiry time. Such action introduces nonmonotonicity as acceptors unaware of certificate revocation will still accept it. The nonmonotonicity is temporal, because when certificate expiry time passes no authoriser will accept the certificate anymore, even when it was revoked.

In order to solve this issue, certificate meaning should be modified to "at the time of issuing t_0, certificate is valid from t_1 to t_2". This interpretation is true any time after t_0 and it does not change even when certificate is revoked. The introduction of t_1 parameter is not crucial, but increases expressivity. Apart from new certificate interpretation, a notion of fresh time (t_f) of a certificate is introduced. Initially its value is set to certificate issue time ($t_f=t_o$). When acceptor ensures that certificate is

still valid at a later time t_x, its fresh time is changed to it ($t_f=t_x$). Each acceptor defines a parameter Δt which states the maximum age of accepted certificates. That is, if $t_f \geq t_{now}-\Delta t$, then certificate is regarded as valid. If the condition does not hold, authoriser needs to reject certificate or verify its validity.

In this paper, the Δt parameter is called a freshness constraint or requirement. Setting Δt to small value implies frequent certificates validity checks. Defining big values limits validity verification but introduces risk of accepting revoked credential. The exact value of Δt depends on application and level of risk it can accept.

Freshness Constraints. After closer look at the idea of fresh time, one can find that Δt parameter should be specified not globally, but on more grained plane such as policy level. For example, some certificates can be treated as more vulnerable to revocation than others. Moreover, Δt value can also be defined based on contextual information available during policy evaluation, e.g. number of revoked certificates from given issuer or customer order amount. Another issue is whether Δt value for a credential should be defined by its issuer or authoriser. This is not a trivial problem. The issuer is aware of credential meaning and circumstances under whose it can be revoked. On the other hand, a freshness constraint can be regarded by acceptor as a part of access policy and therefore should be specified by himself. To link two solutions, system would allow issuers to specify hints about Δt, but let acceptor define final value used in access decision evaluation. The paper focuses on freshness constraints propagation leaving hints specification for future research.

5.2 Role-based Trust-management Framework

The RT Framework is a trust management model. Users, applications and resource holders are called entities. Each entity can define roles and its members. To access a resource, user must be a member of role that represents a shared resource. Entity can also delegate authority over role to other users who will be able to define members of the delegated role. The policy is expressed with a set of RT credentials. At the basis of the RT Framework are sets:

- Credentials or C - a set of RT credentials.
- Entities - a set of RT entities for a given set of RT credentials C.
- RoleNames - a set of names of roles that can be used by entities to define roles for a given set of RT credentials C.

Each RT credential has a form: C:head⟵body. Head contains a credential issuer with role name. Body contains a role expression which depends on credential type. In the framework, there are four types of credentials:

1. Simple membership: A.r ⟵ D. With this statement A asserts that D is a member of role A.r.
2. Simple inclusion: A.r ⟵ B.s. Issuer A asserts that all members of B.s are also members of A.r. This is a simple role delegation, since B can add new entities to A.r role by adding them to B.s role.

3. Linking inclusion: $A.r \longleftarrow A.s$.t. Issuer A asserts that A.r includes all members of B.t role for each B that is member of A.s. This type of credential allows for authority delegation over A.r to members of A.s.
4. Intersection: $A.r \longleftarrow C.s \cap D.t$. This credential allows A to assert that a member of A.r is any entity that simultaneously is member of C.s and D.t.

Based on presented definitions, four sets can be defined:

- $Roles = \{ A.r: A \in Entities,\ r \in RoleNames \}$.
- $LinkedRoles = \{ A.r.s: A \in Entities,\ r,s \in RoleNames \}$.
- $Intersections = \{ f_1 \cap \ldots \cap f_i: f_i \in Entities \cup Roles \cup LinkedRoles \}$.
- $RoleExpressions = Entities \cup Roles \cup LinkedRoles \cup Intersections$.

Authorisation procedure in the RT Framework utilises a credential graph, which is built based on RT credentials. Each vertex corresponds to a role expression, while edges represent credentials. If access should be granted, there should be a path linking node corresponding to resource with vertex representing user. Such path is called a credential chain.

Credential graph is defined below [5]. Notation $e_1 \Leftarrow e_2 \in E_C$ denotes a graph edge and $e_1 \overset{*}{\Leftarrow} e_2 \in E_C$ represent path in a graph.

Definition 1. Let C be a set of RT_0 credentials. The basic credential graph G_C relative to C is defined as follows: the set of nodes N_C=RoleExpressions and the set of edges E_C is the least set of edges over N_C that satisfies the following three closure properties:

1. If $A.r \longleftarrow e \in C$ then $A.r \overset{*}{\Leftarrow} e \in E_C$. $A.r \Leftarrow e$ is called a credential edge.
2. If there exists a path $A.r \Leftarrow B \in G_C$, then $A.r_1.r_2 \overset{*}{\Leftarrow} B.r_2 \in E_C$. A $A.r_1.r_2 \Leftarrow B.r_2$ is called a derived link edge, and the path $A.r \overset{*}{\Leftarrow} B$ is a support set for this edge.
3. If $D, B_1.r_1 \cap B_2.r_2 \in N_C$, and there exist paths $B_1.r_1 \overset{*}{\Leftarrow} D$, and $B_2.r_2 \overset{*}{\Leftarrow} D$ in G_C, then $B_1.r_1 \cap B_2.r_2 \Leftarrow D \in E_C$. This is called a derived intersection edge, and $\{B_1.r_1 \Leftarrow D, B_2.r_2 \Leftarrow D\}$ is a support set for this edge.

To illustrate the authorisation process of RT Framework Figure 1 presents a credential graph corresponding to scenario described in Section 4. There exist credential chains that link John and Adam entities with role representing accessed resource, i.e. eStore.discount. This implies that those users are eligible for a discount. Path for John contains only normal credential edges, whereas path for Adam also contains a derived intersection edge. Not derived edges are labelled with corresponding credentials numbers presented in example scenario policy. Derived link and intersection edges are dashed and dotted respectively and are annotated with their support sets names.

5.3 Freshness Constraints

The paper discusses the possibility to allow for credential revocation in the RT Framework by introducing fresh time notion to it. The idea is to check during access decision evaluation whether all credentials are fresher than a specified value.

Because RT Framework allows for authority delegation, the mechanism should ensure that users to whom the authority over particular role has been delegated does not grant access to entities whose credentials fresh time has exceeded requirements defined in the policy. For instance, a freshness of credentials defining ABUS.university.student members should not exceed freshness constraint defined for eStore.discount role.

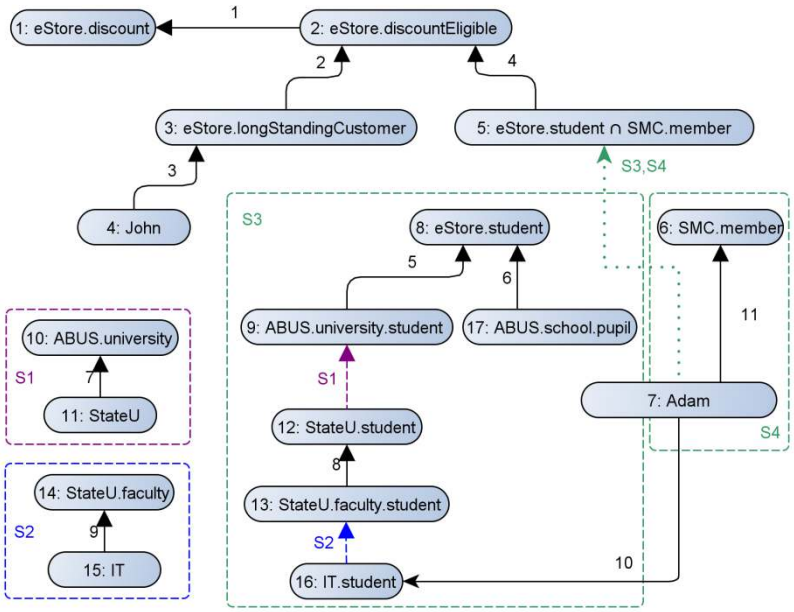

Fig. 1. Credential graph for example scenario

The idea of the propagation mechanism is based on a freshness graph, which is constructed based on RT credential graph. The first step of the procedure is to calculate a freshness constraint for node representing accessed resource. Then, this value is propagated along edges of the freshness graph. If authorisation procedure encounters a credential whose freshness exceeds the propagated constraint, the credential validity is checked. Depending of the verification result, the credential is accepted or rejected.

For the sake of easier understanding, a freshness graph is presented before the formal model of freshness constraints propagation. The graphs for John and Adam entities are depicted on Figure 2. They are based on credential graph presented on Figure 1. In freshness graph, each node has associated a freshness constraint which is denoted by f_n. During credential validity verification this value is used as a freshness requirement.

Definition 2. A freshness graph FG_C is based on RT Framework credential graph G_C. Its set of nodes $FN_C=N_C$, and set of edges FE_C is constructed as follows:

1. If $A.r \Leftarrow e \in E_C$, then $A.r \Rightarrow e \in FE_C$, and it is called a freshness edge.
2. If $A.r_1.r_2 \Leftarrow B.r_2 \in E_C$, then $A.r_1.r_2 \Rightarrow A.r_1 \in FE_C$, $B \Rightarrow B.r_2 \in FE_C$, and they are called linked freshness edges.
3. If $B_1.r_1 \cap B_2.r_2 \Leftarrow D \in E_C$, then $B_1.r_1 \cap B_2.r_2 \Rightarrow B_1.r_1 \in FE_C$ and $B_1.r_1 \cap B_2.r_2 \Rightarrow B_2.r_2 \in FE_C$, and they are called intersection freshness edges.

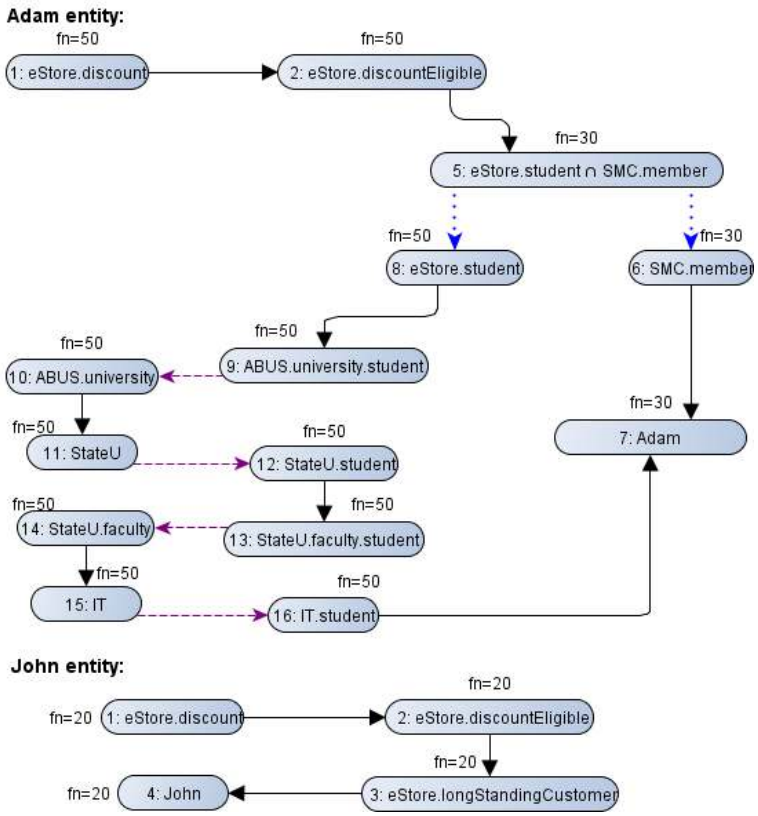

Fig. 2. Freshness constraints for Adam and John entities.

Defining Freshness Requirements. Policy creators should have possibility to define freshness constraints on different levels of granularity:

- role definition level – allows to define constraint value per role definition (e.g. eStore.discount, ABUS.student);
- entity level – defines freshness requirement value for specific entity and any roles defined by it (e.g. ABUS, eStore);
- global level – global value used when no other freshness requirement is defined.

In the paper a role representing shared resource to which user tries to get access is called a root role. In the example, a root role is an eStore.discount role. Presented levels are used in freshness constraint computation only when evaluating constraints

for a root role. This value is used as a starting point of the propagation process. Thus, it is called an initial freshness constraint.

A global freshness constraint Δt_g defines global level restriction. To define a freshness constraints for entities and roles, a f_c function is introduced. To allow for using contextual information during defining freshness requirements, a set of predicates $P=\{p_1,...,p_n\}$ is defined. Each predicate corresponds to a contextual condition. For example, freshness constraints in eStore depend on order amount. Therefore, P has one element, $P=\{p_1$: order.amount>\$100$\}$.

Definition 3. Function f_c defines freshness requirements on entities and roles level: f_c: $D\times P\rightarrow <0, \infty)$, where D=Entities $\cup$ Roles and $P=[p_1,...,p_n]$ is a predicate vector. Values of $p_i \in \{0,1\}$, i=1...n define logical values of predicates.

In the real world scenarios, freshness constraints will be specified only for a subset of the D and not all predicates will be used in definition for given D member. Therefore, f_c can be defined in simplified form presented in Table 1. It contains only a subset of D for whom freshness constraints are explicitly defined. For each member of D, a list of optionally negated predicates is constructed. If all elements in the list evaluate to true, then given Δt value is used. If table does not contain row for given role or entity, an infinite value is assigned. When the predicate list is empty, it means that freshness constraint does not depend on logical values of any predicates.

Table 1. Sample freshness constraints definition matrix

D	predicates	Δt
A	p_1	5
A	$-p_1$	10
B.r		40

Freshness Constraints Propagation. The propagation process starts with calculation of initial freshness constraint. This value is propagated along freshness graph edges.

To compute a propagated freshness constraints, a propagation operator ∇ is introduced. It can be defined in many ways, but the function should be commutative, monotonically decreasive and associative. In this paper a minimum function is used as a propagation operator: $x \nabla y = \min(x,y)$.

A final freshness constraint for each node of freshness graph is stored in f_n function. This value is used during credential freshness verification. To propagate freshness requirements from root role to all nodes of freshness graph, two steps have to be performed. Firstly, a freshness constraint for role expression that is represented by processed node has to be evaluated. This is achieved by calc function. It utilises constraints defined by policy creator in form of f_c function.

Definition 4. A calc function calculates freshness constraint for a given role expression:

$$\text{calc:RoleExpression} \longrightarrow <0,\infty) \; . \tag{12}$$

$$calc(A)=f_c(A)\ . \tag{13}$$

$$calc(A.r)=f_c(A.r)\ \nabla\ f_c(A)\ . \tag{14}$$

$$calc(A.r.s)=calc(A.r)\ \nabla\ f_c(A.r.s)\ . \tag{15}$$

$$calc(A.r\cap B.s)=calc(A.r)\ \nabla\ calc(B.s)\ . \tag{16}$$

The second step of propagation is to take into consideration a freshness constraint of processed node's predecessors. The result is stored in f_n function.

Definition 5. A f_n function represents a freshness constraint for each node of FG_C.

Each node $e \in FN_C$ that is a root role, has a freshness constraint of value:

$$f_n(e)=\Delta t_g\ \nabla\ calc(e)\ . \tag{17}$$

Each node $e_2 \in FN_C$ that is vertex of a freshness edge $e_1 \Rightarrow e_2 \in FE_C$ and is not a root role, has a freshness constraint of value:

$$f_n(e)=\begin{cases} f_n(e_0)\ \nabla\ calc(e) & pre(e)\in Intersections \\ \left(\nabla_{\varepsilon\in pre(e)} f_n(\varepsilon)\right)\ \nabla\ calc(e) & pre(e)\notin Intersections \end{cases} \tag{18}$$

$$e_0=pre\big(pre(e)\big)$$

$pre(e)=\{\ \varepsilon \in FN_C : \varepsilon \Rightarrow e \in FE_C\ \}$ is set of predecessors of e

The f_n definition for node that does not represent a root role contains a case for situation when node's predecessor is an intersection. In such situation, an freshness constraints of intersection's predecessors are taken into consideration. This strategy separates freshness constraints propagation of each intersection element. For example, in Figure 2 if freshness constraint calculation for node 8 would include intersection present in node 5, then f_n=30 value would be propagated to node 8. This is undesirable as this value is defined for SMC.member role but not eStore.student role.

Freshness Constraints Interpretation. Freshness constraints are propagated along edges of FG_C. At the end of the process, each node contains associated freshness requirement. Because credential head contains its issuer, during its validity verification system should use a freshness constraint defined for node $e \in FN_C$ corresponding to credential head. Credential containing an entity in its body states that an entity is a member of role defined in credential head. Therefore, freshness constraint associated with node representing this entity should be used during verification of user public key certificate binding user identity to an entity.

5.4 Example Scenario Analysis

Table 2 contains freshness constraints defined in eStore policy. Constraint for eStore.discount depends on order amount. Table also contains constraints for roles defined by ABUS and SMC. As eStore delegates authority to these entities, it may decide to define freshness constraints for their roles. Since eStore and SMC are

partners, eStore is aware that SMC collects fees on monthly basis and sets constraint for SMC.member role to this period.

Table 2. Freshness constraints for example scenario

global constraint: $\Delta t_g=100$

f_c function matrix		
D	predicates	Δt [days]
eStore		70
eStore.discount	order.amount > $100	20
eStore.discount	− (order.amount > $100)	50
ABUS.university.student		180
SMC.member		30

Figure 2 presents freshness graphs for John and Adam entities with propagated freshness constraints. It was assumed that John has made an order of amount greater than $100, whereas Adam has purchased goods for less than that amount.

The situation with John entity is straightforward. A freshness constraint for root role is 20 and the value is propagated along all freshness edges. Finally, node representing John has freshness constraint $f_n=20$. This value should be used for verification of John public key certificate binding this user to John entity.

Freshness graph for Adam entity contains freshness link edges and freshness intersection edges. They are a consequence of link and intersection derived edges in the RT credential graph. Because Adam entity is a solution to intersection node eStore.student∩SMC.member, it has two predecessors, whose freshness constraints are combined using propagation operator. Final Adam' freshness constraint is $f_n=30$ and this value should be used for verification of his public key certificate.

During RT credentials validity verification, a f_n value associated with role expression of credential head should be used. For example, during verifying eStore.student←ABUS.university.student a f_n(eStore.student)=50 is utilised.

6 Summary

In the paper an analysis of time domain nonmonotonicity has been performed. It was explained how credential revocation causes it. Afterwards, a freshness constraints have been introduced and it was pointed out that credentials are in fact certificates but they convey different type of information. A formal model of freshness requirements propagation has been proposed. It was implemented in the RT Framework to allow for credential revocation. The model allows to define freshness constraints in the policy on different levels of granularity. The solution constructs a freshness graph that is based on credential graph. Freshness requirements are propagated along freshness graph edges. Finally, the model has been evaluated against to real-life example.

In future work, a verification of proposed freshness constraints propagation model will be performed. In order to process authorisation requests search algorithms have been created. Further research will focus on modifying those algorithms in order to

supplement them with freshness constraint propagation. Additionally, a possibility for credential issuers to specify freshness requirements hints will be analysed. The work focused on time domain nonmonotonicity. Future work will also include an analysis of system size domain nonmonotonicity.

References

1. Blaze, M., Feigenbaum, J., Strauss, M.: Compliance Checking in the PolicyMaker Trust Management System. In: 2nd International Conference on Financial Cryptography, pp. 254-274 (1998)
2. Blaze, M., Feigenbaum J., Ioannidis J., and Angelos D. Keromytis. The KeyNote trust-management system, version 2. IETF RFC 2704, September 1999.
3. Blaze, M.; Feigenbaum, J. & Lacy, J. Decentralized trust management Proc. IEEE Symp Security and Privacy, 164-173 (1996)
4. Chapin, P. C.; Skalka, C. & Wang, X. S. Authorization in trust management: Features and foundations ACM Comput. Surv., 2008, 40
5. Czenko, M., Etalle, S., Li, D., Winsborough, W.: An Introduction to the Role Based Trust Management Framework RT. LNCS vol. 4677, pp. 246--281, Springer, Heidelberg (2007)
6. Li, N., Feigenbaum, J. Syverson, P. F. (Ed.) Nonmonotonicity, User Interfaces, and Risk Assessment in Certificate Revocation Financial Cryptography, Springer, 2339, 157-168 (2001)
7. Li, N., Mitchell, J., Winsborough, W.: Design of a Role-Based Trust-Management Framework. In: IEEE Symposium on Security and Privacy, IEEE Computer Society Press, pp. 114-130 (2002)
8. Li, N., Winsborough, W., Mitchell, J.: Distributed Credential Chain Discovery in Trust Management. J. Computer Security 1, pp. 35-86 (2003)
9. National Information Assurance (IA) Glossary, CNSS Instruction No. 4009, 26 April (2010)
10. Seamons, K. E.; Winslett, M. & Yu, T. Limiting the Disclosure of Access Control Policies during Automated Trust Negotiation NDSS, The Internet Society (2001)
11. Seamons, K. E.; Winslett, M.; Yu, T.; Smith, B.; Child, E.; Jacobson, J.; Mills, H. & Yu, L. Requirements for policy languages for trust negotiation Proc. Third Int Policies for Distributed Systems and Networks Workshop, 68-79 (2002)
12. Skalka, C.; Wang, X. S. & Chapin, P. C. Risk management for distributed authorization Journal of Computer Security, 15, 447-489 (2007)
13. Winsborough, W. H. & Jacobs, J. Automated trust negotiation technology with attribute-based access control Proc. DARPA Information Survivability Conf. and Exposition, 60-62 (2003)
14. Yuan, E. & Tong, J. Attributed based access control (ABAC) for Web services Proc. IEEE Int. Conf. Web Services ICWS (2005)
15. Changyu D.; Naranker D., Shinren: Non-monotonic Trust Management for Distributed Systems; Proc. IFIP Advances in Information and Communication Technology (2010)