DOI: 10.15439/2015F95
ACSIS, Vol. 5

More Practical Application of Trust Management Credentials

Anna Felkner, Adam Kozakiewicz
NASK – Research and Academic Computer Network
Wawozowa 18, 02-796 Warsaw, Poland
Email: {anna.felkner, adam.kozakiewicz}@nask.pl

***Abstract*—Trust management is an approach to access control in distributed open systems, where access control decisions are based on policy statements made by multiple principals. The family of Role-based Trust management languages (RT) is an effective means for representing security policies and credentials in decentralized, distributed, large scale access control systems. It provides a set of role assignment credentials. A credential provides information about the privileges of users and the security policies issued by one or more trusted authorities.**

The main purpose of this paper is to show how extensions can make the RT family languages more useful in practice. It shows how security policies can be made more realistic by including timing information, maintaining the procedure or parameterizing the validity of credentials.

I. Introduction

THE modern human life is heavily reliant on trust. The complexity of tasks we are dealing with makes it practically impossible to succeed without delegating some of them. It is therefore not surprising that the issue of trust became a very active area of research quite long ago. The trust between humans has been thoroughly analysed both from the social sciences and economic point of view, as it is an important enabler of delegation. Attempts to transfer the concept of trust to different domains were made by computer scientists, including security, electronic commerce, semantic web or even social networks areas. In any network, be it computer or social one, trust remains an essential factor. The definition of trust is however not perfectly clear, different authors show slightly different definitions. Most often it is defined either on the basis of personal experience, reputation or recommendation.

The concept of trust is closely related to the notion of reputation – the opinion about a person held by others, e.g. the opinion about an Internet seller by his customers or the opinion about the behavior of a node in a wireless sensor network built by other nodes as a result of previous interactions. Reliability is another concept related to trust. Originally, this was a measure of the length of the period during which a machine can be considered trustworthy. In general, trust can be presented a a derivation of an entity's reputation.

Access control systems, based on traditional access control models (like Mandatory Access Control (MAC), Discretionary Access Control (DAC) and Role Based Access Control (RBAC)), are in essence identity based. Authorization decisions are based entirely on the role or – more directly – identity of the requesting party and can only be made if the requester is known to the owner of the resource. The decision is based on the relation of the only two entities involved – the owner of the protected resource, who is responsible for granting access, and the requester, who requires access.

In closed, centralized environments this approach is actually correct. As identity of the users of the system is established in advance, basing access decisions on it is a natural and easy to implement choice. Unfortunately this scenario does not scale well as the system becomes decentralized and highly distributed over a network. In such open systems the set of users is not only large, but may change dynamically, leading to entirely new challenges, such as the problem of propagation of information about the changes. A central database of user's identities is only a partial solution – although it enables the implementation of classic approaches to access control, it also introduces a single point of failure, where temporary lack of access to the central database makes any authorization decisions impossible system-wide. In absence of such a central identity repository it is no longer possible to assume that the identities of the requester and the resource owner are mutually recognized. A more flexible approach is needed, one which enables requesters to cross security domains and access resources owned by non-related entities. One such solution is called trust management.

Consider a simple example of a bookstore which offers special discounts to returning customers who are students. There is no obvious way to determine whether a given customer is eligible based only on his/her identity. Requesting a proof of identity will not resolve this problem. On the other hand, the problem can be solved efficiently using credentials, such as a bookstore card and a student card. The access rights are then determined not based on the users identity, but on the sufficiently documented information about the user's privileges assigned by other authorities and trust for those authorities. This requires a new, different approach to access control.

The rest of this paper is organized as follows. Trust management concept is shown in Section II and Role-based Trust management family of languages is shown in Section III. Section IV presents the Role-based Trust management language syntax with an example of RT^T credentials. Inference system over RT^T language is described in Section V. Section VI describes a few extensions of RT^T language (time validity and determination of the order). Section VII shows an inference system over new RT^T_+ language time constraints. Final remarks are given in the Conclusions.

II. Trust Management

Trust management was first introduced as a term by Blaze et al. [2] in 1996. The term was defined as a unified approach to specify and interpret security policies, credentials and trust relationships. The privileges of an entity in such a system are based not on its identity, but on its attributes. Multiple principals have the right to issue credentials which are then used to demonstrate the entity's attributes. The definition of a *credential* is an attestation of authority, competence or qualification of an individual issued by a third party. The information contained in a credential includes privileges of a given user and/or security policies issued by trusted authorities. Real life examples of credentials are easy to propose – all sorts of academic diplomas, driver's licenses, identification documents, certificates or membership cards are clearly credentials. In fact, a real-life credential does not need to be a document in the traditional sense of the world – e.g. keys can be treated as a form of credential. In a computer system credentials may either be records available from one or more repositories, or – more flexibly – digitally signed documents which can be provided by the requester on demand. In the first case trust for the credential follows from trust for the repository and its own access control, in the second case it is provided by the signature, obviously requiring a trusted method of propagation or verification of public keys.

In literature, the earliest described example of a trust management application was PolicyMaker [3]. An assertion language defined in this system was capable of expressing locally trusted policy statements, as well as credentials requiring a digital signature using a private key. The second generation of trust management languages includes SPKI/SDSI [6], an enhanced version of PolicyMaker called KeyNote [4] and several other languages [7]. In these languages the privileges were still assigned directly to entities and delegation of permissions through credentials was performed only directly from the issuer to the subject. This generation still did not provide any mechanism enabling delegation to be separated from identity of the entity. Introduction of delegation based on atributes (as opposed to identity) was introduced in the next generation, represented by a family of Role-based Trust management (RT) languages [8], [16], [17]. Security policies are represented by defining a formalism using credentials to establish trust in distributed, decentralized access control systems.

III. Role-based Trust Management Family Languages

The family of Role-based Trust management languages is used for representing security policies and credentials in decentralized, distributed access control systems. Several types of role assignment credentials are provided in RT languages, depending on the language. $\mathbf{RT_0}$ [17] forms the core of the family, providing basic abilities – localization of authority for roles, delegation of that authority, role hierarchies and role intersections. These features are available in all RT languages, who extend this set with new features. To represent relationships between entities, parametrized roles were introduced in $\mathbf{RT_1}$. To provide similar flexibility for resources as that provided for entities by roles, $\mathbf{RT_2}$ extends RT_1 with the notion of logical objects, enabling simple assignment of access rights for entire groups of logically related objects (resources). Note that both extentions presented so far do not actually change the expressive power of the language. They allow much more concise notation, but the same policy can in fact be expressed in RT_0, although with a much larger set of credentials, mapping each combination of parameters or each real instance of logical objects to a separate role. The first language actually adding new capabilities not present in other members of the family is the $\mathbf{RT^T}$ language, the main focus of this paper. The new capabilities include the ability to express agreement of multiple principals, even from disjoint sets, via manifold roles and separation of duties or threshold policies, via role-product operators.

A manifold role differs from a normal (singleton) role as instead of defining a set of principals, it defines a set of sets of principals. It is a wider term, since a singleton role can be expressed as a manifold role, whose principal sets are singletons, effectively meaning that cooperation is required from a group consisting of a single entity. Therefore, introduction of manifold roles does not affect the ability to express RT_0 credentials in the RT^T language.

A threshold policy is used to specify a common occurrence, where agreement of multiple principals is required to initiate a given action. More formally, at least k entities from a set satisfying certain conditions must agree on some fact. E.g. in banking certain transations require authorisation by two cashiers. Separation of duties policy is similar, but the agreeing entities fulfill different roles (e.g. in the banking example some transactions may also require authorisation by a controller). Both types of policies specify requirements that cannot be fulfilled by a single entity and therefore cannot be expressed in RT_0.

The RT family includes one more important language, $\mathbf{RT^D}$, which provides mechanisms to describe delegation of role activations and selective use of role membership. However, this language is out of scope of this paper. More in-depth information about the RT family of languages can be found in paper [16].

The main advantage of trust management approach is the ability to use *delegation*. A principal's authority over a resource may be transferred in a limited fashion to other principals by simple means of a credential. The notion of ownership is no longer central – the access control strategy and all decisions on who is authorized to use which resource is defined by a set of credentials. One upside of such approach is that the authority is easy to transfer over a network – as long as a credential can be transferred and trusted in another location, there is no need to involve the identity of the resource owner. However, although such decentralization of credential storage is very useful, it does present a variety of new problems.

In the years since its creation, the concept of trust management has evolved significantly, made applicable to new, broader contexts, such as assessment of reliability or trustworthiness of systems and individuals [15]. In this paper we restrict the meaning of trust management only to its original context of access control. A lot of different work connected with the possibility of using RT in different types of network, eg. Role based Trust management model for Peer-to-Peer networks [5], a trust management system for Ad-Hoc Networks [1], or Wireless Sensor Networks [11], and also Role-based Trust Management Model in Multi-domain Environment [18] have been studied recently.

Our approach is another type of extension of trust management languages. It shows how to make RT family languages more useful in practice by including time validity constraints and order of entities which can appear in the execution context.

IV. The Syntax of RT Family Languages

RT languages use a set of basic elements, such as entities, role names, roles and credentials. *Entities* are principals controlling access to resources by defining roles and issuing credentialsas well as requesters willing to access resources. The entity may be a person, but it might just as well be an application, identifying itself in a computer system by a user name or a public key. *Role names* represent permissions and can be issued to entities or groups of them by other entities. *Roles* represent sets of entities for which the access control policies grant particular permissions. *Credentials* define roles by appointing a new member of the role or by delegating authority to members of other roles.

RT^T includes six different types of credentials, the first four in common with the less expressive RT languages:

$A.r \leftarrow B$	–	*simple membership*: entity B is a member of role $A.r$.
$A.r \leftarrow B.s$	–	*simple inclusion*: role $A.r$ includes (all members of) role $B.s$. This type of credential involves delegation of authority over role r, since by issuing new credentials defining $B.s$ B may add new members to role $A.r$. This type of credential is also used to define role hierarchies.
$A.r \leftarrow B.s.t$	–	*linking inclusion*: role $A.r$ includes role $C.t$ for each C, which is a member of role $B.s$. This is a delegation of authority from A to all the members of the role $B.s$. The expression $B.s.t$ is called a *linked role*.
$A.r \leftarrow B.s \cap C.t$	–	*intersection inclusion*: role $A.r$ includes all members of both roles $B.s$ and $C.t$. This is a partial delegation from A to B and C. The expression $B.s \cap C.t$ is known as an *intersection role*.
$A.r \leftarrow B.s \odot C.t$	–	role $A.r$ can be satisfied by a union set containing one member of each of the two roles ($B.s$ and $C.t$). A single entity being a member of both roles suffices.
$A.r \leftarrow B.s \otimes C.t$	–	role $A.r$ includes two *different* entities, one of which is a member of role $B.s$ and one a member of role $C.t$.

Since the models used in practice can be very complex, this paper uses some simplified examples. We focus on RT^T-specific credentials and intend to illustrate the basic notions and notation, not the full expressive power of the language.

Example 4.1 (Example of RT^T - subject):

Suppose that a university will only activate a subject if at least two of four students apply and among the applicants at least one is a PhD student. RT_0 would require a long list of credentials listing all possible satisfactory combinations, changed whenever the list of students changes. RT^T allows us to express this rule as just two policy credentials and a list of simple membership credentials, easy to manage along with the official list of students. The policy credentials are:

$$F.students \leftarrow F.student \otimes F.student \tag{1}$$

$$F.activeSubject \leftarrow F.students \odot F.phdStudent \tag{2}$$

Now, if the following membership credentials are added::

$$F.student \leftarrow \{Alex\} \tag{3}$$

$$F.student \leftarrow \{Betty\} \tag{4}$$

$$F.student \leftarrow \{David\} \tag{5}$$

$$F.student \leftarrow \{John\} \tag{6}$$

$$F.phdStudent \leftarrow \{John\} \tag{7}$$

$$F.phdStudent \leftarrow \{Emily\} \tag{8}$$

we can conclude that any pair of students from the set $\{Alex, Betty, David, John\}$ fulfills the role $F.students$ and that the subject can be activated if the pair includes John or if Emily is willing to attend.

Example 4.2 (Example of RT^T – signature): Suppose that we have a situation in which we need to collect the signatures of the requester, accountant, his official superior, the manager of financial department and the director of a company to accept some transaction. Such a policy can be described using the following credentials:

$$\begin{aligned} &Company.signature \leftarrow Company.requester \\ &\odot\, Company.accountant \odot Company.superior \\ &\odot\, Company.fdManager \odot Company.director \end{aligned} \tag{9}$$

Now suppose that we have such people, who play those roles, so the following credentials have been added to our security policy:

$$Company.requester \leftarrow \{Jacob\} \tag{10}$$

$$Company.accountant \leftarrow \{Jacob\} \tag{11}$$

$$Company.accountant \leftarrow \{Eliot\} \tag{12}$$

$$Company.accountant \leftarrow \{Alexander\} \tag{13}$$

$$Company.superior \leftarrow \{William\} \tag{14}$$

$$Company.superior \leftarrow \{Michael\} \tag{15}$$

$$Company.fdManager \leftarrow \{Jacob\} \tag{16}$$

$$Company.director \leftarrow \{William\} \tag{17}$$

As we can see, to complete the set of signatures we need just two people: $Jacob$, who can play a role of $requester$, $accountant$, $fdManager$ and $William$ who plays the role of $superior$ and $director$, but as may be required, groups of people $\{Jacob, Eliot, William\}$, $\{Jacob, Eliot, Michael, William\}$, $\{Jacob, Alexander, William\}$, and $\{Jacob, Alexander, Michael, William\}$ can also play a manifold role, and cooperatively complete the set of signatures.

V. Inference System over RT^T Credentials

RT^T credentials define roles, which in turn are used to represent permissions. The set of member entities for a role is defined by a set $\mathcal{P}$ of RT^T credentials. This set can be more conveniently calculated using an inference system, which defines an operational semantics of RT^T language. The system consists of a set of inference rules used to derive credentials from existing ones and an initial set of formulae considered true.

Let $\mathcal{P}$ be a set of RT^T credentials. The inference rules can be applied to create new credentials, derived from credentials of the set $\mathcal{P}$. A derived credential c will be denoted using a formula $\mathcal{P} \succ c$, meaning that credential c can be derived from a set of credentials $\mathcal{P}$.

The initial set of formulae of an inference system over a set $\mathcal{P}$ of RT^T credentials are all the formulae: $c \in \mathcal{P}$ for each credential c in $\mathcal{P}$. The inference rules of the system are the following:

$$\frac{c \in \mathcal{P}}{\mathcal{P} \succ c} \tag{$\mathbf{W_1}$}$$

$$\frac{\mathcal{P} \succ A.r \leftarrow B.s \qquad \mathcal{P} \succ B.s \leftarrow X}{\mathcal{P} \succ A.r \leftarrow X} \tag{$\mathbf{W_2}$}$$

$$\frac{\begin{matrix}\mathcal{P} \succ A.r \leftarrow B.s.t \qquad \mathcal{P} \succ B.s \leftarrow C \\ \mathcal{P} \succ C.t \leftarrow X\end{matrix}}{\mathcal{P} \succ A.r \leftarrow X} \tag{$\mathbf{W_3}$}$$

$$\frac{\begin{matrix}\mathcal{P} \succ A.r \leftarrow B.s \cap C.t \qquad \mathcal{P} \succ B.s \leftarrow X \\ \mathcal{P} \succ C.t \leftarrow X\end{matrix}}{\mathcal{P} \succ A.r \leftarrow X} \tag{$\mathbf{W_4}$}$$

$$\frac{\begin{matrix}\mathcal{P} \succ A.r \leftarrow B.s \odot C.t \qquad \mathcal{P} \succ B.s \leftarrow X \\ \mathcal{P} \succ C.t \leftarrow Y\end{matrix}}{\mathcal{P} \succ A.r \leftarrow X \cup Y} \tag{$\mathbf{W_5}$}$$

$$\frac{\begin{matrix}\mathcal{P} \succ A.r \leftarrow B.s \otimes C.t \qquad \mathcal{P} \succ B.s \leftarrow X \\ \mathcal{P} \succ C.t \leftarrow Y \qquad X \cap Y = \phi\end{matrix}}{\mathcal{P} \succ A.r \leftarrow X \cup Y} \tag{$\mathbf{W_6}$}$$

The inference systems of a language may not be unique – many different systems may be defined. There are however two properties required for the system to be useful in practice – the system must be sound an complete. Soundness guarantees that any formula derived by the system must be valid with respect to the semantics of the language, while completeness ensures that any valid formula is derivable.

All the credentials, which can be derived in the system, either belong to set $\mathcal{P}$ (rule W_1) or are of the type: $\mathcal{P} \succ A.r \leftarrow X$ (rules W_2 through W_6). Proof of soundness of the inference system involves showing that for each new formula $\mathcal{P} \succ A.r \leftarrow X$, the triple (A, r, X) belongs to the semantics $S_{\mathcal{P}}$ of the set $\mathcal{P}$. Completeness is proved by showing that every formula $P \succ A.r \leftarrow X$ can be derived using inference rules for each element $(A, r, X) \in S_{\mathcal{P}}$. Both proofs can be found in [10], showing that the inference system is a valid alternative way of presenting the semantics of RT^T.

Example 5.1 (Inference system for Example 4.1):
We will now derive the set of entities that can cooperate to activate a subject using an inference system, using a limited set of credentials for brevity: ((1), (2), (4), (6), and (7)). Using credentials (1), (2), (4), (6), and (7) according to rule (W_1) we can infer:

$$\frac{F.students \leftarrow F.student \otimes F.student \in \mathcal{P}}{\mathcal{P} \succ F.students \leftarrow F.student \otimes F.student}$$

$$\frac{F.activeSubject \leftarrow F.students \odot F.phdStudent \in \mathcal{P}}{\mathcal{P} \succ F.activeSubject \leftarrow F.students \odot F.phdStudent}$$

$$\frac{F.student \leftarrow \{Betty\} \in \mathcal{P}}{\mathcal{P} \succ F.student \leftarrow \{Betty\}}$$

$$\frac{F.student \leftarrow \{John\} \in \mathcal{P}}{\mathcal{P} \succ F.student \leftarrow \{John\}}$$

$$\frac{F.phdStudent \leftarrow \{John\} \in \mathcal{P}}{\mathcal{P} \succ F.phdStudent \leftarrow \{John\}}$$

Then, using credentials (1), (6) and (4) and rule (W_6) we infer:

$$\frac{\begin{matrix}\mathcal{P} \succ F.students \leftarrow F.student \otimes F.student \\ \mathcal{P} \succ F.student \leftarrow \{John\} \\ \mathcal{P} \succ F.student \leftarrow \{Betty\} \\ \{John\} \cap \{Betty\} = \phi\end{matrix}}{\mathcal{P} \succ \mathbf{F.students} \leftarrow \{\mathbf{John}, \mathbf{Betty}\}}$$

These newly inferred credential and (2) and (7) with the rule (W_5):

$$\frac{\begin{matrix}\mathcal{P} \succ F.activeSubject \leftarrow F.students \odot F.phdStudent \\ \mathcal{P} \succ F.phdStudent \leftarrow \{John\} \\ \mathcal{P} \succ F.students \leftarrow \{John, Betty\}\end{matrix}}{\mathcal{P} \succ \mathbf{F.activeSubject} \leftarrow \{\mathbf{John}, \mathbf{Betty}\}},$$

we can show that the set of entities $\{John, Betty\}$ is sufficient to activate the subject.

Example 5.2 (Inference system for Example 4.2): We use the inference system to formally derive a set of entities who are essential to accept some transaction, i.e. the signatures of the requester, accountant, his official superior, the manager of financial department and the director of a company. To make the notation shorter, let us use C instead of $Company$.

Using credentials (9)-(17) according to the rule (W_1) we can infer:

$$\frac{\begin{array}{c} C.signature \leftarrow C.requester \\ \odot\ C.accountant\ \odot\ C.superior \\ \odot\ C.fdManager\ \odot\ C.director \in \mathcal{P} \end{array}}{\begin{array}{c} \mathcal{P} \succ C.signature \leftarrow C.requester \\ \odot\ C.accountant\ \odot\ C.superior \\ \odot\ C.fdManager\ \odot\ C.director \end{array}}$$

$$\frac{C.requester \leftarrow \{Jacob\} \in \mathcal{P}}{\mathcal{P} \succ C.requester \leftarrow \{Jacob\}}$$

$$\frac{C.accountant \leftarrow \{Jacob\} \in \mathcal{P}}{\mathcal{P} \succ C.accountant \leftarrow \{Jacob\}}$$

$$\frac{C.accountant \leftarrow \{Eliot\} \in \mathcal{P}}{\mathcal{P} \succ C.accountant \leftarrow \{Eliot\}}$$

$$\frac{C.accountant \leftarrow \{Alexander\} \in \mathcal{P}}{\mathcal{P} \succ C.accountant \leftarrow \{Alexander\}}$$

$$\frac{C.superior \leftarrow \{William\} \in \mathcal{P}}{\mathcal{P} \succ C.superior \leftarrow \{William\}}$$

$$\frac{C.superior \leftarrow \{Michael\} \in \mathcal{P}}{\mathcal{P} \succ C.superior \leftarrow \{Michael\}}$$

$$\frac{C.fdManager \leftarrow \{Jacob\} \in \mathcal{P}}{\mathcal{P} \succ C.fdManager \leftarrow \{Jacob\}}$$

$$\frac{C.director \leftarrow \{William\} \in \mathcal{P}}{\mathcal{P} \succ C.director \leftarrow \{William\}}$$

Then, using credentials (9), (10), (11), (14), (16) and (17) and rule (W_5) we infer:

$$\frac{\begin{array}{c} \mathcal{P} \succ C.signature \leftarrow C.requester \\ \odot\ C.accountant\ \odot\ C.superior \\ \odot\ C.fdManager\ \odot\ C.director \\ \mathcal{P} \succ C.requester \leftarrow \{Jacob\} \\ \mathcal{P} \succ C.accountant \leftarrow \{Jacob\} \\ \mathcal{P} \succ C.superior \leftarrow \{William\} \\ \mathcal{P} \succ C.fdManager \leftarrow \{Jacob\} \\ \mathcal{P} \succ C.director \leftarrow \{William\} \end{array}}{\mathcal{P} \succ \mathbf{C.signature} \leftarrow \{\mathbf{Jacob}, \mathbf{William}\}}$$

showing that the set of entities $\{Jacob, William\}$ is sufficient to complete the set of signatures.

Or, using credentials (9), (10), (12), (14), (16) and (17) and rule (W_5) we infer:

$$\frac{\begin{array}{c} \mathcal{P} \succ C.signature \leftarrow C.requester \\ \odot\ C.accountant\ \odot\ C.superior \\ \odot\ C.fdManager\ \odot\ C.director \\ \mathcal{P} \succ C.requester \leftarrow \{Jacob\} \\ \mathcal{P} \succ C.accountant \leftarrow \{Eliot\} \\ \mathcal{P} \succ C.superior \leftarrow \{William\} \\ \mathcal{P} \succ C.fdManager \leftarrow \{Jacob\} \\ \mathcal{P} \succ C.director \leftarrow \{William\} \end{array}}{\mathcal{P} \succ \mathbf{C.signature} \leftarrow \{\mathbf{Jacob}, \mathbf{Eliot}, \mathbf{William}\}}$$

showing that here we need more people to complete the set of signatures, i.e. $\{Jacob, Eliot, William\}$.
Or, using credentials (9), (10), (13), (15), (16) and (17) and rule (W_5) we infer:

$$\frac{\begin{array}{c} \mathcal{P} \succ C.signature \leftarrow C.requester \\ \odot\ C.accountant\ \odot\ C.superior \\ \odot\ C.fdManager\ \odot\ C.director \\ \mathcal{P} \succ C.requester \leftarrow \{Jacob\} \\ \mathcal{P} \succ C.accountant \leftarrow \{Alexander\} \\ \mathcal{P} \succ C.superior \leftarrow \{Michael\} \\ \mathcal{P} \succ C.fdManager \leftarrow \{Jacob\} \\ \mathcal{P} \succ C.director \leftarrow \{William\} \end{array}}{\mathcal{P} \succ \mathbf{C.signature} \leftarrow \{\mathbf{Jacob}, \mathbf{Alexander}, \mathbf{Michael}, \mathbf{William}\}}$$

showing that here we need four people to complete the set of signatures, i.e. $\{Jacob, Alexander, Michael, William\}$.

Depending on which credentials at the moment we have (because not all the credentials are always available), we can determine the sets of people who can cooperatively sign the document.

VI. Credential extensions

This section shows a few extensions of RT^T languages which make it more useful in practice.

A. Time Validity in RT^T

Real security policies involve time restrictions. Allowing the credentials to have limited time validity can make the RT^T language more useful in practice. Inference rules with time validity for RT_0 were originally introduced in a slightly different way in [14]. In [13] we tried to extend the potential of RT^T language by adding time validity constraints. Most permissions are in fact given for fixed periods of time, permanent permissions are less common. Time dependent credentials take the form: c **in** v, meaning "the credential c is available during the time v". Finite sets of time dependent credentials are denoted by $\mathcal{CP}$ and the new language is called RT^T_+. c is used to denote "c **in** $(-\infty,\ +\infty)$" to make notation lighter.

Most trust management languages are monotonic: adding new assertion to a query can never result in canceling an action, which was accepted before [9]. Therefore, each policy statement or credential added to the system may only increase the capabilities and privileges granted to others, making revocation of rights impossible. Introduction of time constraints does not invalidate the monotonicity of the system, but achieves some of the utility of negation.

Time validity can be denoted as follows:

$[\tau_1, \tau_2]; [\tau_1, \tau_2); (\tau_1, \tau_2]; (\tau_1, \tau_2); (-\infty, \tau]; (-\infty, \tau);$
$[\tau, +\infty); (\tau, +\infty); (-\infty,\ +\infty); v_1 \cup v_2; v_1 \cap v_2; v_1 \backslash v_2$

and v_1, v_2 of any form in this list, with τ ranging over time constants.

Example 6.1 (Time validity for Example 4.1):
Assuming that $Alex$, $Betty$, $David$ and $John$ in our scenario will not be student forever seems quite natural. $John$ and $Emily$'s PhD student status is similar. Thus, credentials (3)–(8) should be generalized to:

$$F.student \leftarrow \{Alex\} \textbf{ in } v_1 \tag{18}$$

$$F.student \leftarrow \{Betty\} \textbf{ in } v_2 \tag{19}$$

$$F.student \leftarrow \{David\} \textbf{ in } v_3 \tag{20}$$

$$F.student \leftarrow \{John\} \textbf{ in } v_4 \tag{21}$$

$$F.phdStudent \leftarrow \{John\} \textbf{ in } v_5 \tag{22}$$

$$F.phdStudent \leftarrow \{Emily\} \textbf{ in } v_6 \tag{23}$$

stating that (3) – (8) are only available during v_1, v_2, v_3, v_4, v_5, and during v_6, respectively. The policy itself, described by credentials (1) and (2) may however be permanent. By using (1), (2) and (18)–(23), we want to be able to derive that for example the set $\{Alex, Betty, John\}$ can cooperatively activate the subject during all of the period: $v_1 \cap v_2 \cap v_5$ or $\{Betty, John\}$ during the time $v_2 \cap v_4 \cap v_5$ or $\{Alex, David, Emily\}$ during the time intersection $v_1 \cap v_3 \cap v_6$. Another set of people can cooperatively activate the subject (depending of the time).

Example 6.2 (Time validity for Example 4.2): In our scenario, it is quite natural to assume that $Jacob$ are a requester only for a fixed period of time. The same with $Jacob$, $Eliot$ and $Alexander$ as a company accountants, also $Wiliam$ and $Michael$ as a superior, and $Jacob$ as a financial department manager, as well as $William$ as a director. Thus, credentials (10)–(17) should be generalized to:

$$Company.requester \leftarrow \{Jacob\} \textbf{ in } v_1 \tag{24}$$

$$Company.accountant \leftarrow \{Jacob\} \textbf{ in } v_2 \tag{25}$$

$$Company.accountant \leftarrow \{Eliot\} \textbf{ in } v_3 \tag{26}$$

$$Company.accountant \leftarrow \{Alexander\} \textbf{ in } v_4 \tag{27}$$

$$Company.superior \leftarrow \{William\} \textbf{ in } v_5 \tag{28}$$

$$Company.superior \leftarrow \{Michael\} \textbf{ in } v_6 \tag{29}$$

$$Company.fdManager \leftarrow \{Jacob\} \textbf{ in } v_7 \tag{30}$$

$$Company.director \leftarrow \{William\} \textbf{ in } v_8 \tag{31}$$

stating that (10) – (17) are only available during v_1, v_2, v_3, v_4, v_5, v_6, v_7, and during v_8, respectively. On the other hand, credential (9) can be always valid, as it expresses some time-independent fact. Now, by using (9) and (24)–(31), we want to be able to derive that for example the set $\{Jacob, William\}$ can cooperatively sign the document during all of the period: $v_1 \cap v_2 \cap v_5 \cap v_7 \cap v_8$, where $Jacob$ plays the role of *requester*, *accountant* and *financial department manager* and $William$ acts as a *superior* and *director of the company*. But during the time $v_1 \cap v_3 \cap v_6 \cap v_7 \cap v_8$ it has to be the set consisting of $\{Jacob, Eliot, Michael, William\}$.

While in both examples the policy defining credentials were assummed to be permanently valid, this is not required. Some policies are naturally time-limited (eg. seasonal sales).

B. Determination of the order

Another powerful feature which would be useful to model more realistic policies is the ability to determine the order in which a member of a role or an entity (entities) can appear.

If we want to maintain a procedure, we have to add two new types of credentials at the syntax level. These are:

$A.r \leftarrow B.s \overset{\odot}{\rightarrow} C.t$	–	role $A.r$ is satisfied by a union set of one member of role $B.s$ and one member of role $C.t$ in this exact order or by one entity satisfying the intersection role $B.s \cap C.t$.
$A.r \leftarrow B.s \overset{\otimes}{\rightarrow} C.t$	–	role $A.r$ is satisfied by a set of two different entities: one member of role $B.s$ and one member of role $C.t$ in this order.

In our Example 4.1 we can want to have such situation:

$$F.activeSubject \leftarrow F.students \overset{\odot}{\rightarrow} F.phdStudent$$

which means that the order is important. First we need to have two students and just after that one PhD student.

That extension can be extremely useful in a large variety of situations. For example, if we have a situation, when one person is a member of a few roles, it can be useful to have some restrictions connected with appearing in particular roles during the execution context when the credential is used.

Example 6.3 (The right order of signature):
When we have a situation in which we need to collect the signatures of people who are essential to accept some transaction, we can imagine at least a few scenarios in our security policy.

Suppose that we need a signature of the requester, accountant, his official superior, the manager of financial department and the director of a company, in such order.

Now we can use the data from *Example* 4.2 and we can have three different scenarios:

1) The order is strictly obeyed and it is important that an accountant can give his signature after having

received the signature of a requester, and accountant's superior can give his signature after having received the signature of an accountant, even if it is one person. This means that in a first step $Jacob$ can sign the document as a $requester$, in a second step as an $accountant$, after that $William$ can give his signature as a $Jacob's$ official superior. In the next step $Jacob$ can sign the document as a financial department manager, and at the end $William$ can sign the document as a director of the company. Table I presents the signature order in our first scenario.

It can be important in some situation to strictly keep the order of signatures, but in a huge implementation it can be a little bit inefficient. That is why we can propose two other scenarios.

2) We can allow signing the document by one person who plays a few roles at once if the roles appear in credentials successively without any role between. In our example it can look like in the Table II (to make our example easier, we can use just credentials (10), (11), (14), (16), (17)).
In such a simple example, we have one step less than in the previous scenario. It shows how such change can be useful in real large systems.
3) In our third scenario we can allow that one person, who plays more than one role, can give all the signatures at once. It can be very useful in an automatic implementation. Table III shows how it can look in our third scenario.
That situation means that $Jacob$ accepts his signature as a financial department manager if $William$ signs the document as his official superior and $William$ accepts his signature as a $director$ if $Jacob$ signs the document as a financial department manager. We have to have a possibility to accept or not accept our signature, which is dependent on another person's signature.

If we want to mandate that the entity can appear in particular roles during the execution context exactly when the credential is used, we can put a new type of role denoted by underlined identifiers (e.g. $\underline{r}, \underline{s}, \underline{t}$). In such situation, when we change the credential:

$$\begin{gathered}Company.signature \leftarrow Company.requester\\ \overset{\odot}{\to} Company.accountant \overset{\odot}{\to} Company.superior\\ \overset{\odot}{\to} Company.fdManager \overset{\odot}{\to} Company.director\end{gathered}$$

into:

$$\begin{gathered}Company.signature \leftarrow Company.requester\\ \overset{\odot}{\to} Company.accountant \overset{\odot}{\to} Company.superior\\ \overset{\odot}{\to} Company.fdManager \overset{\odot}{\to} Company.\underline{director}\end{gathered}$$

in our third scenario we will have the situation described in Table IV, meaning that $William$ has to wait with his signature as a $director$ untill the time $Jacob$ approves his signature as $fdManager$.

All the semantics previously defined for RT^T_+, set-theoretic (which maps roles to a set of entity names), operational semantics (where credentials can be derived from the initial set of credentials using a set of inference rules [13]), and logic-programming (where credentials are translated into a logic program [12]), are still valid, meaning that proofs of the soundness and the completeness of that semantics are also valid.

This section shows how we can explore the potential of RT languages. It shows how security policies can be made more realistic by including timing information or maintaining the procedure.

VII. Inference System for RT^T Credentials with Time Validity

All the semantics previously defined for RT languages are still valid for determined order, but they have to be changed for credentials with time validity. Because of that reason we have to take on it. This section is showing an inference system for RT^T credentials with time validity.

We will now adapt the inference system over RT^T credentials to respect time validity. Let $\mathcal{CP}$ be a set of RT^T credentials, from which new credentials may be derived. A derived credential c valid in time τ will be denoted using a formula $\mathcal{CP} \succ_\tau c$, meaning that the credential c can be derived from a set of credentials $\mathcal{CP}$ during the time τ. The initial set of formulae of an inference system over a set $\mathcal{CP}$ of RT^T_+ credentials are all the form: c **in** $v \in \mathcal{CP}$ for each credential c valid in time v in $\mathcal{CP}$. The inference rules of the system are the following:

$$\frac{c \textbf{ in } v \in \mathcal{CP} \quad \tau \in v}{\mathcal{CP} \succ_\tau c} \qquad (\mathbf{CW_1})$$

$$\frac{\mathcal{CP} \succ_\tau A.r \leftarrow B.s \quad \mathcal{CP} \succ_\tau B.s \leftarrow X}{\mathcal{CP} \succ_\tau A.r \leftarrow X} \qquad (\mathbf{CW_2})$$

$$\frac{\begin{gathered}\mathcal{CP} \succ_\tau A.r \leftarrow B.s.t \qquad \mathcal{CP} \succ_\tau B.s \leftarrow C\\ \mathcal{CP} \succ_\tau C.t \leftarrow X\end{gathered}}{\mathcal{CP} \succ_\tau A.r \leftarrow X} \qquad (\mathbf{CW_3})$$

$$\frac{\begin{gathered}\mathcal{CP} \succ_\tau A.r \leftarrow B.s \cap C.t \qquad \mathcal{CP} \succ_\tau B.s \leftarrow X\\ \mathcal{CP} \succ_\tau C.t \leftarrow X\end{gathered}}{\mathcal{CP} \succ_\tau A.r \leftarrow X} \qquad (\mathbf{CW_4})$$

$$\frac{\begin{gathered}\mathcal{CP} \succ_\tau A.r \leftarrow B.s \odot C.t \qquad \mathcal{CP} \succ_\tau B.s \leftarrow X\\ \mathcal{CP} \succ_\tau C.t \leftarrow Y\end{gathered}}{\mathcal{CP} \succ_\tau A.r \leftarrow X \cup Y} \qquad (\mathbf{CW_5})$$

$$\frac{\begin{gathered}\mathcal{CP} \succ_\tau A.r \leftarrow B.s \otimes C.t \qquad \mathcal{CP} \succ_\tau B.s \leftarrow X\\ \mathcal{CP} \succ_\tau C.t \leftarrow Y \qquad X \cap Y = \phi\end{gathered}}{\mathcal{CP} \succ_\tau A.r \leftarrow X \cup Y} \qquad (\mathbf{CW_6})$$

All the derived credentials either belong to set $\mathcal{CP}$ (rule CW_1) or are of the type: $\mathcal{CP}_\tau \succ A.r \leftarrow X$ (rules CW_2 through CW_6). This new inference system is based on an extension of the inference rules from section V, where rules

TABLE I
SIGNATURE ORDER IN THE FIRST SCENARIO

Step	**requester**	**accountant**	**superior**	**fdManager**	**director**
1	$Jacob$	ϕ	ϕ	ϕ	ϕ
2	$Jacob$	$Jacob, Eliot, Alexander$	ϕ	ϕ	ϕ
3	$Jacob$	$Jacob, Eliot, Alexander$	$William, Michael$	ϕ	ϕ
4	$Jacob$	$Jacob, Eliot, Alexander$	$William, Michael$	$Jacob$	ϕ
5	$Jacob$	$Jacob, Eliot, Alexander$	$William, Michael$	$Jacob$	$William$

TABLE II
SIGNATURE ORDER IN THE SECOND SCENARIO

Step	**requester**	**accountant**	**superior**	**fdManager**	**director**
1	$Jacob$	$Jacob$	ϕ	ϕ	ϕ
2	$Jacob$	$Jacob$	$William$	ϕ	ϕ
3	$Jacob$	$Jacob$	$William$	$Jacob$	ϕ
4	$Jacob$	$Jacob$	$William$	$Jacob$	$William$

TABLE III
SIGNATURE ORDER IN THE SECOND SCENARIO

Step	**requester**	**accountant**	**superior**	**fdManager**	**director**
1	$Jacob$	$Jacob$	ϕ	$Jacob$	ϕ
2	$Jacob$	$Jacob$	$William$	$Jacob$	$William$

TABLE IV
SIGNATURE ORDER IN THE THIRD "UNDERLINED" SCENARIO

Step	**requester**	**accountant**	**superior**	**fdManager**	**director**
1	$Jacob$	$Jacob$	ϕ	$Jacob$	ϕ
2	$Jacob$	$Jacob$	$William$	$Jacob$	ϕ
3	$Jacob$	$Jacob$	$William$	$Jacob$	$William$

(W_i) are replaced with (CW_i) and only valid time-dependent credentials from $\mathcal{CP}$ are considered.

The proof of soundness of the inference system requires showing that for each new formula $\mathcal{CP}_\tau \succ A.r \leftarrow X$, the triple (A, r, X) belongs to the semantics $S_{\mathcal{CP}}$ of the set $\mathcal{CP}$. All the formulae $\mathcal{CP}_\tau \succ A.r \leftarrow X$, such that $A.r \leftarrow X \in \mathcal{CP}$ are sound, as shown in [12].

Completeness of the inference system over a set $\mathcal{CP}$ of RT_+^T credentials can be proved by showing that a formula $\mathcal{CP} \succ A.r \leftarrow X$ can be derived using inference rules for each element $(A, r, X) \in S_{\mathcal{CP}}$. The proof is presented in [12].

A. Inferring time validity of credentials

The proposed inference system can also derive the maximal time validity of a credential c from $\mathcal{CP}$. Formula $\mathcal{CP} \succ_\tau c$ is modified to $\mathcal{CP} \succ\!\succ_v c$, meaning that at any time $\tau \in v$ in which $\mathcal{CP}$ has a semantics, it is possible to infer the credential c from $\mathcal{CP}$. The inference rules of the system are the following:

$$\frac{c \textbf{ in } v \in \mathcal{CP}}{\mathcal{CP} \succ\!\succ_v c} \qquad (\mathbf{CWP_1})$$

$$\frac{\mathcal{CP} \succ\!\succ_{v_1} A.r \leftarrow B.s \quad \mathcal{CP} \succ\!\succ_{v_2} B.s \leftarrow X}{\mathcal{CP} \succ\!\succ_{v_1 \cap v_2} A.r \leftarrow X} \qquad (\mathbf{CWP_2})$$

$$\frac{\begin{array}{c}\mathcal{CP} \succ\!\succ_{v_1} A.r \leftarrow B.s.t \\ \mathcal{CP} \succ\!\succ_{v_2} B.s \leftarrow C \quad \mathcal{CP} \succ\!\succ_{v_3} C.t \leftarrow X\end{array}}{\mathcal{CP} \succ\!\succ_{v_1 \cap v_2 \cap v_3} A.r \leftarrow X} \qquad (\mathbf{CWP_3})$$

$$\frac{\begin{array}{c}\mathcal{CP} \succ\!\succ_{v_1} A.r \leftarrow B.s \cap C.t \\ \mathcal{CP} \succ\!\succ_{v_2} B.s \leftarrow X \quad \mathcal{CP} \succ\!\succ_{v_3} C.t \leftarrow X\end{array}}{\mathcal{CP} \succ\!\succ_{v_1 \cap v_2 \cap v_3} A.r \leftarrow X} \qquad (\mathbf{CWP_4})$$

$$\frac{\begin{array}{c}\mathcal{CP} \succ\!\succ_{v_1} A.r \leftarrow B.s \odot C.t \\ \mathcal{CP} \succ\!\succ_{v_2} B.s \leftarrow X \quad \mathcal{CP} \succ\!\succ_{v_3} C.t \leftarrow Y\end{array}}{\mathcal{CP} \succ\!\succ_{v_1 \cap v_2 \cap v_3} A.r \leftarrow X \cup Y} \qquad (\mathbf{CWP_5})$$

$$\frac{\begin{array}{c}\mathcal{CP} \succ\!\succ_{v_1} A.r \leftarrow B.s \otimes C.t \\ \mathcal{CP} \succ\!\succ_{v_2} B.s \leftarrow X \quad \mathcal{CP} \succ\!\succ_{v_3} C.t \leftarrow Y \\ X \cap Y = \phi\end{array}}{\mathcal{CP} \succ\!\succ_{v_1 \cap v_2 \cap v_3} A.r \leftarrow X \cup Y} \qquad (\mathbf{CWP_6})$$

$$\frac{\mathcal{CP} \succ\!\succ_{v_1} c \quad \mathcal{CP} \succ\!\succ_{v_2} c}{\mathcal{CP} \succ\!\succ_{v_1 \cup v_2} c} \qquad (\mathbf{CWP_7})$$

Rule is (CWP_1) claims that $\mathcal{CP}$ can be used whenever it is valid. Rules (CWP_2) - (CWP_6) simply claim that inference rules can be used iff all their premises are true. Finally, the rule (CWP_7) is used to join validity periods, meaning that if c can be inferred both with validity v_1 and validity v_2, then it can also be inferred with validity $v_1 \cup v_2$. $\mathcal{CP} \succ\!\succ_v$ generalizes $\mathcal{CP} \succ_\tau$. They are both equivalent whenever $v = [\tau, \tau]$. Note that inferring a certain c from $\mathcal{CP}$ may be possible in several different ways, resulting in different validity periods. Rule (CWP_7) can then be used as many times as necessary to broaden c's validity.

Maximal inference is the process, where in each step we infer with maximal time validity.

An inference terminating in $\mathcal{CP} \succ\!\succ_v c$ is called maximal if and only if:

1) there exists no $v' \supset v$ such that $\mathcal{CP} \succ\!\succ_{v'} c$, and
2) every its sub-inference terminating in $\mathcal{CP} \succ\!\succ_{v''} c'$, for $c' \neq c$ is maximal.

The first condition ensures that further use of rule (CWP_7) will not extend the validity of c. The second condition ensures that this property is propagated through the whole inference tree. Maximal inferences guarantee that v in (CWP_1) is the maximal time validity for $A.r \leftarrow X$.

For these inferences we can prove soundness and completeness of $\mathcal{CP} \succ\!\succ_v$, as shown in [12].

Example 7.1 (Time validity in inference system for Example 4.1): Let us get back to our example and to make long example shorter, let us use less credentials: (1), (2), (19), (21), and (22). According to rule (CWP_1) we can infer:

$$\frac{F.students \leftarrow F.student \otimes F.student \in \mathcal{CP}}{\mathcal{CP} \succ\!\succ F.students \leftarrow F.student \otimes F.student}$$

$$\frac{F.activeSubject \leftarrow F.students \odot F.phdStudent \in \mathcal{CP}}{\mathcal{CP} \succ\!\succ F.activeSubject \leftarrow F.students \odot F.phdStudent}$$

$$\frac{F.student \leftarrow \{Betty\} \textbf{ in } v_2 \in \mathcal{CP}}{\mathcal{CP} \succ\!\succ_{v_2} F.student \leftarrow \{Betty\}}$$

$$\frac{F.student \leftarrow \{John\} \textbf{ in } v_4 \in \mathcal{CP}}{\mathcal{CP} \succ\!\succ_{v_4} F.student \leftarrow \{John\}}$$

$$\frac{F.phdStudent \leftarrow \{John\} \textbf{ in } v_5 \in \mathcal{CP}}{\mathcal{CP} \succ\!\succ_{v_5} F.phdStudent \leftarrow \{John\}}$$

When we want to check when two different students can cooperate, from credentials (1), (19), (21) and rule (CWP_6) we infer:

$$\frac{\begin{array}{c}\mathcal{CP} \succ\!\succ F.students \leftarrow F.student \otimes F.student \\ \mathcal{CP} \succ\!\succ_{v_2} F.student \leftarrow \{Betty\} \\ \mathcal{CP} \succ\!\succ_{v_4} F.student \leftarrow \{John\} \\ \{Betty\} \cap \{John\} = \phi\end{array}}{\mathcal{CP} \succ\!\succ_{v_2 \cap v_4} \mathbf{F.students} \leftarrow \{\mathbf{Betty}, \mathbf{John}\}}$$

In next step we use it and additionally credentials (2), (22) and rule (CWP_5):

$$\frac{\begin{array}{c}\mathcal{CP} \succ\!\succ F.activeSubject \leftarrow F.students \odot F.phdStudent \\ \mathcal{CP} \succ\!\succ_{v_5} F.phdStudent \leftarrow \{John\} \\ \mathcal{CP} \succ\!\succ_{v_2 \cap v_4} F.students \leftarrow \{Betty, John\}\end{array}}{\mathcal{CP} \succ\!\succ_{v_2 \cap v_4 \cap v_5} \mathbf{F.activeSubject} \leftarrow \{\mathbf{Betty}, \mathbf{John}\}}$$

showing that the set of entities that can cooperatively activate a subject is: $\{Betty, John\}$ during the time: $v_2 \cap v_4 \cap v_5$.

Example 7.2 (Time validity in inference system for Example 4.2): Let us get back to our example and use credentials (9) and (24)–(31) (to make the notation shorter, let use C instead of $Company$). According to rule (CWP_1) we can infer:

$$\frac{\begin{array}{c}C.signature \leftarrow C.requester \\ \odot\ C.accountant \ \odot\ C.superior \\ \odot\ C.fdManager \ \odot\ C.director \in \mathcal{CP}\end{array}}{\begin{array}{c}\mathcal{CP} \succ\!\succ C.signature \leftarrow C.requester \\ \odot\ C.accountant \ \odot\ C.superior \\ \odot\ C.fdManager \ \odot\ C.director\end{array}}$$

$$\frac{C.requester \leftarrow \{Jacob\} \textbf{ in } v_1 \in \mathcal{CP}}{\mathcal{CP} \succ\!\succ_{v_1} C.requester \leftarrow \{Jacob\}}$$

$$\frac{C.accountant \leftarrow \{Jacob\} \textbf{ in } v_2 \in \mathcal{CP}}{\mathcal{CP} \succ\!\succ_{v_2} C.accountant \leftarrow \{Jacob\}}$$

$$\frac{C.accountant \leftarrow \{Eliot\} \textbf{ in } v_3 \in \mathcal{CP}}{\mathcal{CP} \succ\!\succ_{v_3} C.accountant \leftarrow \{Eliot\}}$$

$$\frac{C.accountant \leftarrow \{Alexander\} \textbf{ in } v_4 \in \mathcal{CP}}{\mathcal{CP} \succ\!\succ_{v_4} C.accountant \leftarrow \{Alexander\}}$$

$$\frac{C.superior \leftarrow \{William\} \textbf{ in } v_5 \in \mathcal{CP}}{\mathcal{CP} \succ\!\succ_{v_5} C.superior \leftarrow \{William\}}$$

$$\frac{C.superior \leftarrow \{Michael\} \textbf{ in } v_6 \in \mathcal{CP}}{\mathcal{CP} \succ\!\succ_{v_6} C.superior \leftarrow \{Michael\}}$$

$$\frac{C.fdManager \leftarrow \{Jacob\} \textbf{ in } v_7 \in \mathcal{CP}}{\mathcal{CP} \succ\!\succ_{v_7} C.fdManager \leftarrow \{Jacob\}}$$

$$\frac{C.director \leftarrow \{William\} \textbf{ in } v_8 \in \mathcal{CP}}{\mathcal{CP} \succ\!\succ_{v_8} C.director \leftarrow \{William\}}$$

Now, when we want to check when $Jacob$ and $William$ are the only people, who are necessary to cooperatively sign the document we use credentials (9), (24), (25), (28), (30), (17) and rule (CWP_5):

$$\frac{\begin{array}{c}\mathcal{CP} \succ\!\succ C.signature \leftarrow C.requester \\ \odot\ C.accountant \ \odot\ C.superior \\ \odot\ C.fdManager \ \odot\ C.director \\ \mathcal{CP} \succ\!\succ_{v_1} C.requester \leftarrow \{Jacob\} \\ \mathcal{CP} \succ\!\succ_{v_2} C.accountant \leftarrow \{Jacob\} \\ \mathcal{CP} \succ\!\succ_{v_5} C.superior \leftarrow \{William\} \\ \mathcal{CP} \succ\!\succ_{v_7} C.fdManager \leftarrow \{Jacob\} \\ \mathcal{CP} \succ\!\succ_{v_8} C.director \leftarrow \{William\}\end{array}}{\begin{array}{c}\mathcal{CP} \succ\!\succ_{v_1 \cap v_2 \cap v_5 \cap v_7 \cap v_8} \mathbf{C.signature} \\ \leftarrow \{\mathbf{Jacob}, \mathbf{William}\}\end{array}}$$

showing that the set of entities $\{Jacob, William\}$ is sufficient to complete the set of signatures during the time $v_1 \cap v_2 \cap v_5 \cap v_7 \cap v_8$.

Or, using credentials (9), (24), (26), (28), (30) and (31) and rule (CWP_5) we infer:

$$\frac{\begin{array}{c} \mathcal{CP} \succ\!\succ C.signature \leftarrow C.requester \\ \odot\ C.accountant\ \odot\ C.superior \\ \odot\ C.fdManager\ \odot\ C.director \\ \mathcal{CP} \succ\!\succ_{v_1} C.requester \leftarrow \{Jacob\} \\ \mathcal{CP} \succ\!\succ_{v_3} C.accountant \leftarrow \{Eliot\} \\ \mathcal{CP} \succ\!\succ_{v_5} C.superior \leftarrow \{William\} \\ \mathcal{CP} \succ\!\succ_{v_7} C.fdManager \leftarrow \{Jacob\} \\ \mathcal{CP} \succ\!\succ_{v_8} C.director \leftarrow \{William\} \end{array}}{\begin{array}{c} \mathcal{CP} \succ\!\succ_{v_1 \cap v_3 \cap v_5 \cap v_7 \cap v_8} \mathbf{C.signature} \\ \leftarrow \{\mathbf{Jacob}, \mathbf{Eliot}, \mathbf{William}\} \end{array}}$$

showing that here we need more people to complete the set of signatures, i.e. $\{Jacob, Eliot, William\}$ during the time $v_1 \cap v_3 \cap v_5 \cap v_7 \cap v_8$.

Or, using credentials (9), (24), (27), (29), (30) and (31) and rule (CWP_5) we infer:

$$\frac{\begin{array}{c} \mathcal{CP} \succ\!\succ C.signature \leftarrow C.requester \\ \odot\ C.accountant\ \odot\ C.superior \\ \odot\ C.fdManager\ \odot\ C.director \\ \mathcal{CP} \succ\!\succ_{v_1} C.requester \leftarrow \{Jacob\} \\ \mathcal{CP} \succ\!\succ_{v_4} C.accountant \leftarrow \{Alexander\} \\ \mathcal{CP} \succ\!\succ_{v_6} C.superior \leftarrow \{Michael\} \\ \mathcal{CP} \succ\!\succ_{v_7} C.fdManager \leftarrow \{Jacob\} \\ \mathcal{CP} \succ\!\succ_{v_8} C.director \leftarrow \{William\} \end{array}}{\begin{array}{c} \mathcal{CP} \succ\!\succ_{v_1 \cap v_4 \cap v_6 \cap v_7 \cap v_8} \mathbf{C.signature} \\ \leftarrow \{\mathbf{Jacob}, \mathbf{Alexander}, \mathbf{Michael}, \mathbf{William}\} \end{array}}$$

showing that here we need four people to complete the set of signatures, i.e. $\{Jacob, Alexander, Michael, William\}$ during the time $v_1 \cap v_4 \cap v_6 \cap v_7 \cap v_8$.

VIII. Conclusions

In the paper we model the use of trust management systems in decentralized and distributed environments. The modelling framework is a family of Role-based Trust management language RT^T. The core part of the paper is introduction of time validity constraints and especially maintaining the procedure – modifications aimed at making the RT^T language more realistic. While the inference systems presented in this paper are simple, they are well-founded theoretically. The utility of the proposed extentions is most visible in large-scale distributed systems, where users have only partial view of their execution context.

References

[1] R. Akbani, T. Korkmaz, G.V.S. Raju, "Mobile Ad-Hoc Networks Security", Z. Qian et al. (Eds.): *Recent Advances in in Computer Science and Information Engineering*, Springer-Verlag Berlin Heidelberg 2012, pp. 659–666. http://dx.doi.org/10.1007/978-3-642-25769-8_92

[2] M. Blaze, J. Feigenbaum, J. Lacy, "Decentralized Trust Management", *Proc. 17th IEEE Symposium on Security and Privacy,* Oakland CA, 1996, pp. 164–173. http://dx.doi.org/10.1109/SECPRI.1996.502679

[3] M. Blaze, J. Feigenbaum, and M. Strauss, "Compliance checking in the PolicyMaker trust management system", in Proc. 2nd Int. Conf. Financial Cryptogr., London, UK, 1998, pp. 254–274. http://dx.doi.org/10.1007/BFb0055488

[4] M. Blaze, J. Feigenbaum, and A. D. Keromytis, "The role of trust management in distributed systems security" in Secure Internet Programming, J. Vitek, C. Damsgaard Jensen, Eds. London: Springer, 1999, pp. 185–210. http://dx.doi.org/10.1007/3-540-48749-2_8

[5] S. Chithra, "A Role Based Trust Model for Peer to Peer Systems Using Credential Trees", *International Journal of Computer Theory and Engineering*, Vol.3, No.2, April 2011, ISSN: 1793-8201, pp. 234–239. http://dx.doi.org/10.7763/IJCTE.2011.V3.310

[6] D. Clarke et al., "Certificate chain discovery in SPKI/SDSI", J. Comp. Secur., vol. 9, pp. 285–322, 2001.

[7] P. Chapin, C. Skalka, and X. S. Wang, "Authorization in trust management: Features and foundations", ACM Comput. Surv., vol. 3, pp. 1–48, 2008. http://dx.doi.org/10.1145/1380584.1380587

[8] M. R. Czenko, S. Etalle, D. Li, and W. H. Winsborough, "An Introduction to the Role Based Trust Management Framework RT", Tech. Rep. TR-CTIT-07-34, Centre for Telematics and Information Technology University of Twente, Enschede, The Netherlands, 2007. http://dx.doi.org/10.1007/978-3-540-74810-6_9

[9] M. R. Czenko et al., "Nonmonotonic Trust Management for P2P Applications", in Proc. 1st Int. Worksh. Secur. Trust Manag. STM 2005, Milan, Italy, 2005. http://dx.doi.org/10.1016/j.entcs.2005.09.037

[10] A. Felkner, K. Sacha, "Deriving RT^T Credentials for Role-Based Trust Management", *e-Informatica Software Engineering Journal*, Volume 4, No 1, 2010, pp. 9–19.

[11] A. Felkner, "How the Role-based Trust Management Can be Applied to Wireless Sensor Nnetworks", *Journal of Telecommunications and Information Technology,* Volume 4, 2012, pp.70–77.

[12] A. Felkner, A. Kozakiewicz, "RT^T_+-Time Validity Constraints in RT^T Language", *Journal of Telecommunications and Information Technology,* Volume 2, 2012, pp. 74–82.

[13] A. Felkner, A. Kozakiewicz, "Time Validity in Role-based Trust Management Inference System", *Secure and Trust Computing, Data Management, and Applications Communications in Computer and Information Science,* Volume 187, 2011, pp. 7–15. http://dx.doi.org/10.1007/978-3-642-22365-5_2

[14] D. Gorla, M. Hennessy, V. Sassone, "Inferring Dynamic Credentials for Role-Based Trust Management", *Proc. 8th Conference on Principles and Practice of Declarative Programming*, ACM, 2006, pp. 213–224. http://dx.doi.org/10.1145/1140335.1140361

[15] W. M. Grudzewski, I.K. Hejduk, A.Sankowska, M. Wańtuchowicz, "Trust Management in Virtual Work Environments: A Human Factors Perspective", *CRC Press Taylor & Francis Group*, 2008.

[16] N. Li, J. Mitchell, W. Winsborough, "Design of a Role-Based Trust-Management Framework". Proc. IEEE Symposium on Security and Privacy. IEEE Computer Society Press, Oakland CA (2002), pp. 114–130. http://dx.doi.org/10.1109/SECPRI.2002.1004366

[17] N. Li, W. Winsborough, J. Mitchell, "Distributed Credential Chain Discovery in Trust Management". J. Comput. Secur. 1 (2003), pp. 35–86.

[18] H. Liu, Q. Zhang, J. Zheng, X. Guo, "Role-based Trust Management Model in Multi-domain Environment", *TELKOMNIKA Indonesian Journal of Electrical Engineering*, Vol 11, No 1: January 2013, pp. 417–424