Available online at www.sciencedirect.com

ScienceDirect

Electronic Notes in Theoretical Computer Science 244 (2009) 53–65

Electronic Notes in Theoretical Computer Science

www.elsevier.com/locate/entcs

An Implementation of Role-Base Trust Management Extended with Weights on Mobile Devices

Davide Fais [1] Maurizio Colombo[2]

Istituto di Informatica e Telematica
Consiglio Nazionale delle Ricerche
Pisa, Italy

Aliaksandr Lazouski[3]

Dipartimento di Informatica
Universita di Pisa
Pisa, Italy

Abstract

This paper describes the implementation of a library for the management and evaluation of Role-based Trust Management (RT) credentials and policies written in RTML, also extended with weights, in mobile devices. In particular, it describes the implementation of the library in J2ME. It is worth noticing, that RTML credentials are XML-like documents and thus the capability of porting these features on mobile devices makes the overall framework very interoperable with other RT frameworks (as for GRID systems). As policy language, we use actually a variant of RTML, whose policies are added with weights and are able to express quantitative experience-based notions of trust. It allow also to encode certain reputation and recommendation models. The obtained results show how the implementation on mobile devices is feasible and the running time acceptable for several applications.

Keywords: role-based trust management, reputation, mobile devices, policy language, computer security.

1 Introduction

Trust management is a promising approach to authorize entities in open, distributed, heterogenous computer environments. Trustworthiness of the entity determines access permissions. Trust value is assigned to every entity and based on credentials the peer possesses and/or experience accumulated by previous interactions with

[1] Email: davide.fais@iit.cnr.it

[2] Email: maurizio.colombo@iit.cnr.it

[3] Email: lazouski@di.unipi.it

doi:10.1016/j.entcs.2009.07.038

the peer. A credential certifies some attributes (e.g. roles)issued to the peer by an attribute authority. Accessing the resource the peer usually needs to present a set of credentials signed by multiple decentralized authorities. Role-based trust management (RT) framework [8,7,6] is especially suitable for attribute-based access control in large-scale, distributed systems with decentralized attribute authorities. RT exploits a notion of a role as peer's attribute that can be delegated. We extended RT assigning a trust measure or weight to peer's roles in order to incorporate experience-based trust capabilities [9]. The prototype, RT's credentials and policies enhanced with trust weights and the evaluation engine, is described in this paper and was successfully implemented on mobile devices.

Several trust management frameworks supporting attribute-based authorization have been proposed last years, e.g. RT, PolicyMaker [1], KeyNote [2], SPKI [3], etc. We concentrate on RT as it has some specific advantages. RT constitutes a declarative, logic-based semantic foundation, support for vocabulary agreement, strongly-typed credentials and policies, more flexible authority delegation structures [7]. RT is a theoretically-based and practically-implementable approach which encompasses role-based access control (RBAC) and trust management. Flexibility and expressiveness of RT framework bases on the notion of role exploited in the assignment of access permissions to the role's holder. RT assumes delegated, linked and parameterized roles.

Heterogenous computer environment compounds various types of devices interacting with each other to accomplish some particular goal. To use RT authorization framework and to allow peers interoperate seamlessly we need implement RT on different devices and platforms. RT framework has been successfully implemented and deployed in several environments [4,7]. In this paper, we focus on the implementation of RT framework on mobile devices supported J2ME. Original semantics of RT policies and credentials written in RTML policy language [5] was extended by trust weight associated with the role. We also introduced an algorithm to compute trust weight's values for roles. As a matter of fact, the prototype integrated both credential-based and experience-based trust models in a single framework. We optimized the framework and did some assumptions during implementation taking into account power and computational limitations of mobile devices. The final prototype was able to verify, validate RTML with weight credentials pushed to or pulled by the mobile device. Applying inference rules to RT credentials and local access rules, the deduction engine returned the complete set of the requestor attributes with assigned trust values and consequently access permissions. We did some performance tests to measure applicability of the framework.

The paper is structured as follows. In Section 2, we recall the role-based trust management framework. In Section 3 we show its flexible extension with weights and the corresponding evaluations algorithm. We then show the prototype implementation on mobile devices in Section 4. Example of credentials, policy statements and some performance evaluation are given in Section 5. Finally, Section 6 contains our conclusions and future work.

2 Role-based Trust Management

The *RT Role-based Trust-management* framework for access control provides policy language, semantics, deduction engine, and concrete tools such as *application domain specification documents* which help distributed users to maintain consistent use of policy terms (e.g., see [7,11]). *RT* purpose is to manage access control and authorization problems in large-scale and decentralized system. Such problems came out when independent and autonomous organizations, whose membership change very rapidly, wish to share their resources. *RT* combines the strength of Role-Based Access Control (RBAC) and trust-management (TM). RBAC was developed for access control in a single organization in which the control of role membership and role permissions is relatively centralized in a few users, *RT* takes from it the notion of role as instrument to assign permissions to users. TM is an approach to distributed access control and authorization in which access control decision are taken on the base of *policy statements* made by multiple principals. From TM, *RT* takes the principles of managing distributed authority through the use of credentials, as well as some notation denoting relationships between those authorities. The main concept in *RT* is the notion of *roles*, each *RT* principal has its own name space for roles, and each role is compounded by the principal name and a *role term*. For example, if K_A is a principal and R is a role term, then $K_A.R$ is the role R defined by principal K_A and can be read as K_A's R role. Only K_A has the authority to issue policy statements defining which are the members for the role $K_A.R$. A role in RBAC can be viewed as a set of principals who are members of this role. Granting a permission to a principal means to making the principal a member of the set corresponding to the permission, granting a permission to a role implies the assertion that the set corresponding to the permission includes as a subset the set corresponding to the role.

2.1 Parametric Roles

Roles in RBAC are atomic strings, sometimes this fact should be too limited. Could be desirable to use the same name for a large numbers of roles with few differences among them. To address this feature *RT* introduced the notion of parameterized role with the following data types:

- **Integer types.** An integer type is ordered.
- **Closed enumeration types.** Could be both ordered and unordered.
- **Open enumeration types.** An Open enumeration type is unordered.
- **Float types.** A float type is ordered.
- **Date and time types.** There are predefined types and they are ordered.

A role term takes the form $r(p_1, ..., p_n)$, in which r is a role name, and each p_j takes one of the following three forms: $name = c$, $name = X$ and $name \in S$, where $name$ is the name of a parameter of r, c is a constant of the appropriate type, X is a variable and S is a value set of the appropriate type. Variables make equal two parameters in the same role definition.

2.2 RTML credentials

An *RT* credential has an head and a body. The *head* of a credential has the form $A.r(p_1, ..., p_n)$, in which A is an entity, and $r(p_1, ..., p_n)$ is a role name. For each i in $1...n$, p_i is a data term having the type of the ith parameter of r. A data term could be either a constant or a variable. A credential with the head $A.r(h_1, ...h_n)$ *defines* the role $A.r(h_1, ...h_n)$. A is the issuer of the credential. In the following are presented into an abstract syntax six types of credentials, each having a different form of body corresponding to a different way of defining role membership.

- **Simple Member** $A.r(p =' value') \leftarrow D$
  The credential issuer is A and the role membership includes the D principal (A and D are possibly the same). In general a Simple Member represents a certification about which roles has been acquired by a user, it is signed by the entity who released this roles. The body part is a *Principal* identifier.
- **Simple Containment** $A.r \leftarrow B.r_1$
  An *ExternalRole* is the body part of the credential. A defines that the role membership includes all entities included in the external r_1 role issued by the B principal (A and B are possibly the same). The dimension of $A.r$ should be no less than that of $B.r_1$.
- **Linked Containment** $A.r \leftarrow A.r_1.r_2$
  The body part consists of a *LinkedRole* element, which contains two or more elements. The credential issuer is A and the role membership includes all entities from the external r_2 role issued by the P principal where P is a member of r_1 of the default principal A. The dimension of $A.r$ should be no less than that of $P.r_2$
- **Intersection Containment** $A.r \leftarrow A.r_1 \cap B.r_2$
  The body part consists of an *Intersection* element, which contains two or more roles; it could be an external role if the referred role is issued by an external entity or it could issued by A itself. The role membership includes all entities which are simultaneously members from the internal r_1 role and the external r_2 role issued by the B principal.

  The following two definitions can be express using the previous rules:

- **Simple Delegation** $A.r \Leftarrow B[: r_2]$
  The credential issuer is A who delegates its authority over r to B, in other words, A trusts B's judgment on assigning members to r. When r_2 is present it works as a sort of control; B can only assign members of $A.r_2$ to $A.r$. A *Simple Delegation* could be expressed using a *Simple Containment* and an *Intersection Containment*: $A.r \leftarrow B.r \cap A.r_2$
- **Advanced Delegation** $A.r \Leftarrow r_1[: B.r_2]$
  The credential issuer is A who delegates its authority over r to members of $A.r_1$. When $B.r_2$ is present it works as a sort of control; each member of $A.r_1$ can

only assign members of $B.r_2$ to $A.r$. In other words an *Advanced Delegation* could be expressed using a *Linked Containment* and an *Intersection Containment*: $A.r \leftarrow A.r.r_1 \cap B.r_2$

2.3 XML syntax for credentials

RTML is an XML-based data representation for *RT* policies and credentials which implements the RT framework. It includes data types to encode permissions, a mechanism for identifying principals, data-structures to define a common vocabulary and a semantic relation that determines when, given a set of policy statements and a set of user credentials, a query is true. RTML uses three types of document:

- **ApplicationDomainSpecification**
  Defines a suite of related data types and role names, called a *vocabulary.* The use of a role name needs to refer to the ADSD in which the role name is declared.
- **Credentials**
  Defines one or more credentials issued by an entity, this document should be signed by the entity who released these credentials.
- **AccessRules**
  Defines the rules which control the access to a role or group.

In the following we describe these documents.

2.3.1 Application Domain Specification Document (ADSD)

As previously disclosed, the use of a shared vocabulary constitutes a critical point in those systems which represent the rights as attributes, such as the Grid environment. To delegate permissions on resources, all the principals involved in the chain need to use consistent terminology to specify resource permissions and delegation conditions. If some of them use incompatible schemes, their credentials cannot be meaningfully combined and some intended permissions cannot be granted. The definition of a role is meaningful only if all the parts involved are allowed to access the role structure and possibly the permissions granted to it. An ADSD defines a vocabulary which implements this structure using standard data type or defining new ones, this document has to be public. This grants *RT* to have a strongly typed credentials and policies. A credential which uses a role term needs to refer the particular ADSD which defines the role name and all its data types.

2.3.2 Credential documents

In a "trust-management" approach, a requester sends a request to an *authorizer* who specifies an access-policy, expressed as a set of *access rules*, which govern the accesses to protect resources. The requester adds a set of credentials to the request and the authorizer decides whether to authorize this request by answering the question: "Do the access rules and credentials authorize the request?".

Credential and *AccessRules* have the same structure; they define one or more rule contained into a root-element called *CredentialStore* tag:

```
<CredentialStore >

       <Credential id='#1'>. . .</Credential>

             . . .

       <Credential id='#n'>. . .</Credential>

</CredentialStore>
```

Each rule definition is compound of three elements.

(i) **Prologue**
It contains all the informations used afterwords in the document, it counts a *DefaultDomain* and zero or more *ImportDomain* which refers all the ADSDs necessary to recover the structure of all the roles involved, one or more *Principal* containing the public key of the entities in the credential and a *Issuer* that points to the Principal who issued the role, it could be an IntegerValue or a StringValue but it has to be a KeyValue as defined in the XML Signature Standard to offer guarantee of security.

(ii) **Credential**
RTML defines a set of Credential/AccessRule definition, each containing a *HeadRoleTerm* and a *body*. Different kinds of definitions contain different elements as the body part. We adopt an abstract syntax in describing these definitions; r represents the *HeadRoleTerm*, r_1 and r_2 represents other role terms, we assume A as the credential issuer while B and D represent generic issuers.

(iii) **VerificationData**
It contains a *ValidityTime* element, and an optional signature part consisting of a *Signature* element as specified in the XML Signature Standard.

2.3.3 AccessRule Documents

As already expressed in 2.3.2, *Credential* and *AccessRule* documents have the same structure. The main difference consists that the *Issuer* value is not present in the *AccessRule* document; it defines a policy and the issuer of these rules is the policy issuer itself. Furthermore the *Signature* element is missed too, the issuer does not need to verify its policy.

3 Extension with weights

In our framework, reputation of users can be calculated according to properties that a user possesses with respect to services, rather than a role that he/she covers. As an example, reputation of a user can be calculated based on past experiences of other services with respect to that user. The more the user has been well-behaved with that service, the more the service will positively recommend interactions with that user. Thus, by rephrasing what stated above, a simple member credential $A.r(v) \leftarrow D$ can be read as A *asserts that* D *has property* r *with degree (weight)* v.

Services emit two kinds of credentials, either expressing trust towards good behaviors of users, or recommending someone able to express that a user has a good behavior. The first kind of credentials expresses trust towards a functionality (*e.g.,* towards good behaviors), and we denote them by $A.f(v) \leftarrow D$, *i.e.,* A trusts D for performing functionality f with degree v. The latter are credentials of recommendation, denoted as $A.rf(v) \leftarrow D$, and they express the fact that A trusts D as a recommender able to suggest someone for performing f.

Recommendations can be transitive. The transitivity step is encoded into *RT* by a linking containment of the form $A.rf \leftarrow A.rf.rf$. This statement says that if A defines B to have property $A.rf$, and B defines D to have property $B.rf$, then A defines D to have role $A.rf$, *i.e.,* D is trusted to act as a recommender according to A.

The indirect functional trust step is encoded as $A.f \leftarrow A.rf.f$. This statement says that if B has role $A.rf$ and C has role $B.f$ then C has role $A.f$. B, that has the role $A.rf$, is the recommender, *i.e.,* A trust B for choosing someone that is trusted for performing f. C, that has role $B.f$, is trusted to perform f by B. Hence, C is indirectly trusted to perform f by A.

Also, one can define a set of functionalities, *e.g.,* a range of possible values for f, and the relative simple containment credentials are, *e.g.,* as follows:

- $A.files(p, v) \leftarrow$ D. A trusts user D with degree v for *operating on a file.*
- $A.socket(p, v) \leftarrow$ D. A trusts user D with degree v for *operating on a socket.*

There must be explicit rules for combining trust weights. Thus, we consider two operators, namely the link operator $\otimes$ and the aggregation operator $\odot$, for combining the trust measures. Generally speaking, the former is used to compose trust weights, while the latter is used to compare, select or aggregate trust weights. We give some specific examples of semiring-based trust measures. Assume that we want to consider the weight with maximal trust, then:

- $\otimes$ is the multiplication on real numbers (for instance between 0 and 1);
- $\odot$ is the maximum between two real numbers.

Here, we recall the language given in [10], for enriching part of RTML with trust measures.

- $A.r(p, v) \leftarrow D$. The role $A.r(p)$ is covered with weight v.
- $A.r(p) \leftarrow_{v_2} A_1.r_1(p_1)$. According to A, all members of role $A_1.r_1(p_1)$ with weight v_1 are members of role $A.r(p)$ with weight $v = v_1 \otimes v_2$.
- $A.r(p) \leftarrow A.r_1(p_1).r_2(p_2)$. If B has role $A_1.r_1(p_1)$ with weight v_1 and D has role $B.r_2(p_2)$ with weight v_2, then D has role $A.r(p)$ with weight $v = v_1 \otimes v_2$.
- $A.r(p) \leftarrow A_1.r_1(p_1) \cap A_2.r_2(p_2)$. This statement defines that if D has both roles

$A_1.r_1(p_1)$ with weight v_1 and $A_2.r_2(p_2)$ with weight v_2, then D has role $A.r(p)$ with weight $v = v_1 \odot v_2$.

Note that weights must not explicitly appear in the simple, linking and intersection containment statements. Indeed, these statements combine basic credentials (the simple member ones) and they determine how weights from the basic credentials must be combined too.

3.1 An implementation of RTML with trust measures

We present an algorithm for calculating a set of simple member credentials with trust measures on $RTML$. Figure 1 shows algorithm's details. It takes as input the available credentials, split in two sets, the set of basic credentials and the others. If one does not consider trust measures, the algorithm basically builds the minimal set of simple member credentials by iteratively applying the inference rules for each kind of credential. If the inferred credential does not belong yet to the set of computed basic credentials, then it is added to this set. The procedure is iterated until no new credentials are found. When the algorithm is applied to a finite set of credentials, it correctly terminates.

Adding weights is possible. Indeed, due to the specific nature of c-semirings we are going to apply, it can be also seen as a variant of the Floyd algorithm for calculating minimal/maximal weighted paths among all the nodes in a graph. Indeed, a simple member credential, say $A.r(v) \leftarrow C$, states that between the node A and the node C there is an arc labeled r and with measure v. If we assume the order $\leq_w$ among weights defined as $v_1 \leq_w v_2$ iff $v_1 \odot v_2 = v_2$, then the algorithm computes the greatest weighted path ($w.r.t.$ $\leq_w$) in the graph. We remind that in c-semiring $\otimes$ is an inclusive operation.

4 Prototype Implementation on Mobile Devices

This section outlines the ongoing implementation of the RT with trust weights on mobile devices supported J2ME platform and CLDC1.1/MIDP2.0 profiles. The extended version of the prototype was initially implemented as a part of the authorization framework for GRID services [4]. The prototype was modified due to minimize of the size of the final application and taking into account J2ME limitations. Java language was chosen for its suitability in developing and testing mobile applications. Moreover, several simulation environments are freely available (e.g. NetBeans), and a wide set of mobile devices like Smartphones and PDAs are able to execute MIDlets.

RT framework consists of (i) XML-based user's credentials expressed in RTML policy language formats; (ii) local XML-based access rules or policy statements saved on a mobile device and managing an access to some particular resource; (iii) a deduction engine. The deduction engine discovers new user's attributes or roles inferred from access rules and credentials. Applying inference rules to every

```
Trust Calculations (basic creds, rules)= {
   Results:=basic creds; Changed := true;
   While(Changed) {
     Changed:=false;
     For each credential $A.r \leftarrow_{v_2} A_1.r_1$ in rules and for each credential
     $A.r_1(v_1) \leftarrow C$ in basic creds
       if $A.r \leftarrow C$ not in basic creds, or $A.r(v) \leftarrow C$ in basic creds with
       not $v_1 \otimes v_2 \leq_w v$
          then {remove from basic creds all the creds like $A.r \leftarrow C$;
          insert $A.r(v_1 \otimes v_2) \leftarrow C$ in basic creds; Changed:=true};
     For each credential $A.r \leftarrow A.r_1.r_2$ in rules and for each credential
     $A.r_1(v_1) \leftarrow B$, $B.r_2(v_2) \leftarrow C$ in basic creds
       if $A.r \leftarrow C$ not in basic creds, or $A.r(v) \leftarrow C$ in basic creds with
       not $v_1 \otimes v_2 \leq_w v$
          then {remove from basic creds all the creds like $A.r \leftarrow C$;
          insert $A.r(v_1 \otimes v_2) \leftarrow C$ in basic creds; Changed:=true};
     For each credential $A.r \leftarrow A_1.r_1 \cap A_2.r_2$ in rules and for each credential
     $A_1.r_1(v_1) \leftarrow C$, $A.r_2(v_2) \leftarrow C$ in basic creds
       if $A.r \leftarrow C$ not in basic creds, or $A.r(v) \leftarrow C$ in basic creds with
       not $v_1 \odot v_2 \leq_w v$
          then {remove from basic creds all the creds like $A.r \leftarrow C$;
          insert $A.r(v_1 \odot v_2) \leftarrow C$ in basic creds; Changed:=true};
}
```

Fig. 1. Algorithm for credential inference.

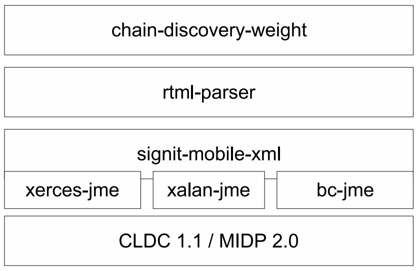

Fig. 2. RT framework with weights for mobile devices

credential and access rule the deduction engine returns the complete set of the user's attributes. We did assumption about centralized credential's repository from which credentials can be pushed or pulled to mobile device over Bluetooth connection link.

The complete software packages structure of the implemented prototype shown in the figure 2. The packages supplied RT framework functionality by verifying, validating, parsing signed XML-based user's credentials and access rules and evaluating new credentials by the deduction engine.

Any user's credential could be presented as a single XML file. To ensure integrity all credentials were digitally signed. The XML Digital Signature standard [4] with some extensions (RFC-4050) was used for credentials signing. Cryptographic algorithms for J2ME were imported from bouncy castle project [5]. The signature was kept into the Signature Value tag of XML encoded credential representation. Public key of the credential holder was settled at KeyInfo tag. *Xerces-jme* [6], *xalan-jme* [7], *bc-jme* and *signit-mobile-xml* packages prepared the input to *rtml-parser* by verifying the credentials signature and expiration time, extracting the related XML code, and saving credentials in a local KeyStore holding access rules and protected by the password.

The *rtml-parser* is a DOM-based parser which converted received credentials and access rules from XML-format into a complex data structure and passed it to the deduction engine implemented by *chain-discovery-weight* package.

Chain-discovery-weight evaluated new credentials based on the initials credentials and access rules. It also implemented the algorithm described in section 3.1 assigning properly a trust weight to every new credential. *Chain-discovery-weight* maintained the final set of credentials wit trust measures. Initially, weights were assigned to credentials and access rules by random values before sending to the deduction engine. We did so in order to keep the original RTML format of the credentials. The new user's credentials denoted a complete set of roles a user held on the mobile device. This information might be used to grant to the user some access permissions.

5 Performance evaluation

We performed some experiments on Nokia E-61 Smartphone to measure the performance of the prototype.

Working algorithms of the prototype, as mentioned before, included several steps and was as follows: (i) to upload a set of user's credentials to the mobile device via Bluetooth; (ii) to validate and verify a signature and time expiration of each credential on the mobile device; (iii) to parse credentials and corresponding access rules from XML-based format to a logical structure understandable by the deduction engine; (iv) to push credentials and access rules to the deduction engine to evaluate a set of all user's credentials and trust weights assigned to it.

We measured time consumptions for step (ii) till (iv) varying the number of credentials and access rules. We used combinations of 4 user's credentials and 3 policy rules to test the prototype. Due to space limitations we do not present here XML version of credentials, its definition and assigned weights. We refer a reader to [5], where the similar example was considered in details. The credentials and access rules used in the prototype are enlisted below:
Credentials:

[4] http://www.ietf.org/rfc/rfc3275.txt
[5] http://www.bouncycastle.org/
[6] http://xerces.apache.org/xerces-j/
[7] http://xml.apache.org/xalan-j/

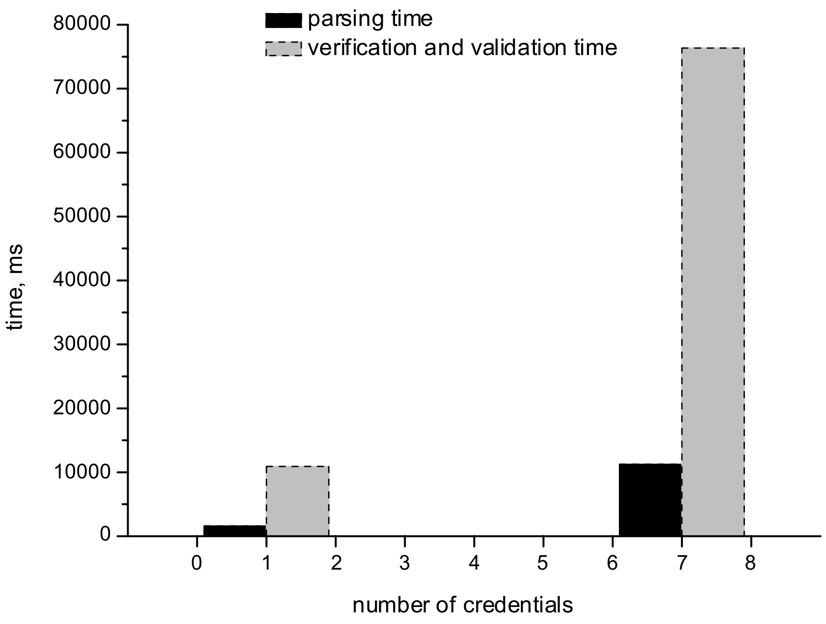

Fig. 3. Credentials management performance

(0) $K_{StateU}.stagist('BobSmith','StateU') \leftarrow K_{Bob}$
(1) $K_{StateU}.student('StateU','InformaticScience','123456789','BobSmith') \leftarrow K_{Bob}$
(2) $K_{Acm}.acmmember('BobSmith','Professional','UJ11111') \leftarrow K_{Bob}$
(3) $K_{Abu}.university('StateU') \leftarrow K_{StateU}$
Access rules:
(4) $K_{EPub}.epubRole1() \leftarrow K_{Acm}.acmmember(name,-,-) \cap K_{EPub}.student(-,'InformaticScience',-,name)$
(5) $K_{EPub}.university(uniName) \Leftarrow K_{Abu}$
(6) $K_{EPub}.student(uniName,'InformaticScience','123456789',-) \Leftarrow K_{EPub}.university(uniName)$

For example, passing (3) and (5), the deduction engine generates the new credential:
(7) $K_{EPub}.university('StateU') \leftarrow K_{StateU}$

One set of experiments was done to evaluate time costs of credentials verification and validation and parsing. Obtained results for 1 and 7 credentials is presented in figure 3. As shown in the figure, time for validation-verification of 1 credential was almost 11.0 seconds while parsing only 1.6 seconds. The time had a linear dependence on the number of credentials to be processed.

Figure 4 presents performance of the deduction engine for the case of 2 (1 credential and 1 access rule) and for the case of 7 (4 credentials and 3 access rules) policy statements. It shows that evaluation time almost remained on the same value and formed 28 seconds delay for 7 policies statements.

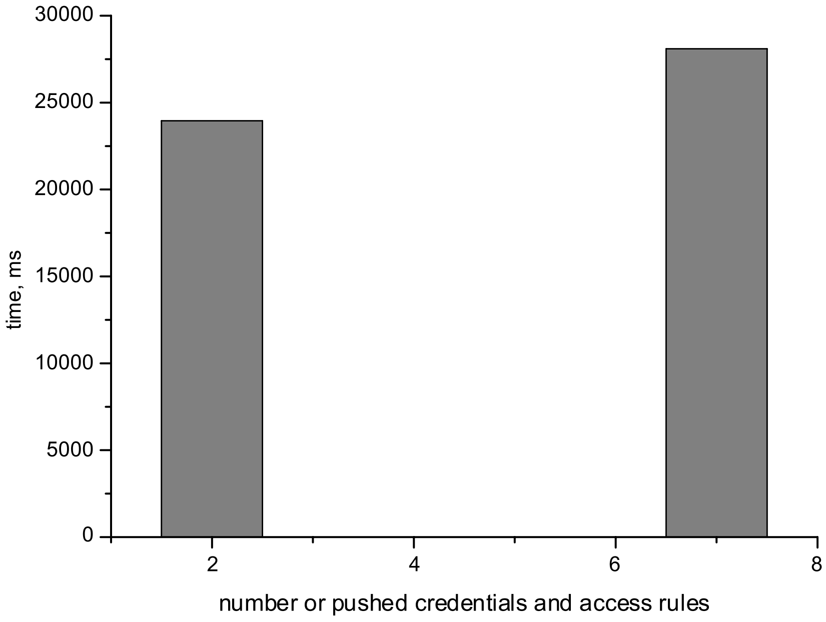

Fig. 4. The deduction engine performance

Running several experiments on a real devices we inferred that the credential verification and validation time expenses are very sensitive. The overall time expenses are acceptable only for specific application but can not be used, for example, to monitor run-time executions.

6 Conclusions and future work

In this paper, we have introduced implementation of RT framework on mobile devices. Trust weights were embedded into the RT credentials to supply certain reputation and recommendation models. We presented the algorithm to compute weights and ran some experiments to test the prototype performance. Obtained results showed that credentials verification and validation were the most time consuming comparing to credentials and access rules parsing and evaluation.

We plan to reduce time expenses by using light cryptographic signing algorithms and developing an adoptive caching model for credentials verification and validation. Finally, we concern about a complex case study expressing benefits of RT with weights. We suspect a Grid scenario to examine interoperability of RT framework deployed on various devices.

References

[1] Blaze, M., J. Feigenbaum and M. Strauss, *Compliance checking in the policymaker trust management system*, in: *Financial Cryptography*, 1998, pp. 254–274.
URL `citeseer.ist.psu.edu/blaze98compliance.html`

[2] Blaze, M., J. Ioannidis and A. D. Keromytis, *Experience with the keynote trust management system: Applications and future directions.*
URL `citeseer.ist.psu.edu/641873.html`

[3] Clarke, D. E., *Spki/sdsi http server / certificate chain discovery in spki/sdsi* (2001).
URL citeseer.ist.psu.edu/clarke01spkisdsi.html

[4] Colombo, M., F. Martinelli, P. Mori, M. Petrocchi and A. Vaccarelli, *Fine grained access control with trust and reputation management for globus*, in: *OTM Conferences (2)*, 2007, pp. 1505–1515.

[5] et al, N. L., *Rtml: A role-based trust-management markup language*, Technical report, Purdue University, cERIAS TR 2004-03.

[6] Li, N. and J. Mitchell, *Rt: A role-based trust-management framework* (2003).
URL citeseer.ist.psu.edu/li03rt.html

[7] Li, N., J. C. Mitchell and W. H. Winsborough, *Design of a role-based trust management framework*, in: *Proc. IEEE Symposium on Security and Privacy, Oakland*, 2002.
URL citeseer.ist.psu.edu/533810.html

[8] Li, N., W. H. Winsborough and J. C. Mitchell, *Distributed credential chain discovery in trust management: extended abstract*, in: *ACM Conference on Computer and Communications Security*, 2001, pp. 156–165.
URL citeseer.ist.psu.edu/article/li01distributed.html

[9] Liu, J. and V. Issarny, *Enhanced reputation mechanism for mobile ad hoc networks*, in: *iTrust*, 2004, pp. 48–62.

[10] Martinelli, F. and M. Petrocchi, *On relating and integrating two trust management frameworks*, Electr. Notes Theor. Comput. Sci. **168** (2007), pp. 191–205.

[11] Winsborough, W. H. and N. Li, *Safety in automated trust negotiation*, in: *S&P*, IEEE, 2004 .