Towards a Flexible Access Control Mechanism for E-Transactions

Vishwas Patil and R.K. Shyamasundar

School of Technology and Computer Science
Tata Institute of Fundamental Research
Homi Bhabha Road, Colaba, Mumbai - 400005, India

Abstract. Security over the Internet depends on a clear distinction between authorized and un-authorized principals. Discriminating between the two involves: identification (user identifies himself/herself), authentication (the system validates the user's identity) and authorization (specific rights granted). Thus, it is important to develop specifications for access control that realize the above properties with ease. Public Key Infrastructures (PKIs) provide a basis for specifying access-control to the users in a secure and non-reputable fashion. Some of the general deficiencies of PKIs are: (i) they are rigid and cannot scale across different PKI frameworks, (ii) due to efficiency reasons, PKIs are constrained to be just static data-structures shipped across domains and hence cannot carry any dynamic or state-based information, and (iii) for reasons of (ii) the recipients are not explicitly defined. In this paper, we shall argue that a judicious mix of digital certificates and authentication mechanisms would lead to a flexible security policy specification having both static and dynamic capabilities and lead to user-friendly mechanisms to achieve availability of secure services in e-commerce.

1 Introduction

Web has become an important medium for disseminating information, doing commerce and business. Organizations are making their whole work-flow system e-enabled, sharing their resources with clients, collaborators, out-stationed employees etc. It has become necessary to deploy mechanisms that would restrict access of such networked resources only to the authorized users. The main security threats are:

– Confidentiality: gets violated when unauthorized users get protected information,
– Integrity: gets violated when unauthorized user modifies protected information,
– Availability: gets violated when the system is prevented from doing its intended function.

Cryptographic frameworks like PKI [1] have become de facto standards for realizing such requirements since it provides authentication, authorization of users, data integrity, confidentiality, and non-repudiation of transactions.

Access control mechanisms based on digital certificates solely, help in authenticating and determining user's authorizations. However, it needs an extensive infrastructure, which is costly and as certificates are not modifiable once issued, the dynamic or context-based authorization policies cannot be embedded into them.

There are access control mechanisms that are based on the capability models, traditionally used in operating systems, in which each process originating from an authorized user inherits same power and this power has to be trimmed down or amplified based on the context/state of the system, as the case in capability based operating systems. This is not possible in a PKI based access control mechanisms because of the static nature of certificates that carry only authorization information. However, the rationale of a PKI cannot be achieved in capability models. Some of the recent work in capability based systems [2] tries to take the power of capability model to the applications in distributed environment. In this approach, users transfer their capabilities over resources/users to other using references, unlike authorization certificates in PKI based models. A user can execute a permission over a resource if it has a reference inherited from someone who already posses such authorization over the resource. Revocation of authority in such models is immediate since it is an on-line system. Being an on-line system (that are generally costly), it may hamper the access control decisions in case the underlying communication system fails.

Several other models exist that try to capture the requirements of an access control mechanism; PolicyMaker [3], KeyNote [4], RBAC [5]. Access control mechanisms for a small number of users with limited set of permissions under a single administrative domain are fairly straight forward. For example, an access control matrix in which each entry defines a specific set of permissions for a user; removing the entry will revoke user's authorizations. As the number of users grow, this approach becomes tedious in terms of management and performance due to the size of the access control matrix. RBAC etc. abstracts user permissions into roles, and the roles are assigned to users using digital certificates that can be used as a proof against the role authorizations.

In this paper, we show that a mix of an off-line framework such as SPKI/SDSI [6], with an *a priori* defined heterogeneous authentication mechanisms (on-line/off-line), on the resource controller's side to perform additional policy compliance checking, leads to an expressive policy specification framework. A large part of our framework has been the establishment of the need of such a mixed approach. Towards the end, we shall illustrate how the above concepts are being embedded in our effort on the design of *flexi*-ACL [7].

The rest of the paper is organized as follows: An overview of SPKI/SDSI is given in section 2. In sections 3 – 4, we will argue requirements of an access control framework in distributed environments and highlight the shortcomings of existing approaches. We briefly sketch our access control framework formalism in section 5 followed by discussions in section 6.

2 Background: SPKI/SDSI

In this section, we give an overview of underlying PKI framework, SPKI/SDSI, used in our framework. SPKI and SDSI [6, 8] were two separate efforts initiated to overcome the complexity, privacy and trust related issues faced by the highly centralized traditional PKIs [9] that have now been integrated under the name SPKI/SDSI or just SPKI.

SPKI emphasizes naming, groups, ease-of-use, and flexible authorizations. To access a protected resource, a client must present proof that it is authorized to the server; this proof takes the form of a "certificate chain" proving that the client's public key is in one of the groups on the resource's ACL. *In SPKI, the onus of identification and proof of authority to do something is left on the user*. Every principal can issue/define the key bindings locally using an identifier which is valid only locally but the underlying raw public key is valid globally. So, the SPKI principals can issue certificates binding other principal's keys or names (called extended names) at discretion. An authorization grant is made only locally. If a principal needs to grant authorization to someone beyond her locality, then she may (must) delegate that grant through a chain of local relationships. Formally SPKI certificates are defined below:

- A name certificate C is a signed four-tuple (K, A, S, V):
  - The $issuer\ K$ is a public key; the certificate is signed by K.
  - The $identifier\ A$ (together with the issuer) determines the local name "KA" that is being defined; the name belongs to the local name space of key K.
  - The $subject\ S$ is a key or a local name defined in the other key's name space to which $issuer$ of this certificate binds its local name.
  - The $validity\ specification\ V$ provides additional information allowing one to ascertain if the certificate is currently valid, beyond the obvious verification of the certificate signature.
- An authorization certificate C is a signed five-tuple (K, S, D, T, V): where,
  - $delegation\ bit\ D$, if true, grants the $subject$ permission to further delegate to others the authorization it is receiving via this certificate.
  - $authorization\ specification$ or $authorization\ tag$ $,T$, that specifies permissions being granted.

A brief functional outline of the authorization computation using rewrite rules is illustrated with the following scenario: *Let principal K serve a resource denoted by* RESOURCE *and specify the ACL for its access. K authorizes principals K_1, K_2, K_3 to act as retailers for its service. A principal K_s subscribes for the* RESOURCE *service via one of the retailers. Let us see how the subscriber K_s comes up with an authorization proof to access* RESOURCE*, and how the resource owner K makes use of group certificates and extended names to efficiently specify and manage the access to the resource.* Design for such a policy is given below, using short-hand denotations for the sake of simplicity;

- $K\ retailers \rightarrow \{K_1, K_2, K_3\}$ is a "$retailers$" group defined by principal K and it can enforce a common policy on all the three subject principals by just narrating the policy over the name definition $retailers$.
- $K\ customers \rightarrow \{K_A, K_B, K_3\ customers\}$ is another local group definition by principal K, where it has included another group i.e. K_3's $customers$ apart from K_A and K_B, where $K_3\ customers \rightarrow \{K_p, K_q, K_r, K_s\}$.
- Principal K consolidates its groups by issuing $K\ subscribers \rightarrow \{K\ retailers, K\ customers\}$ and empowers its members to access the RESOURCE by making an authorization definition, $K_{\text{RESOURCE}} \rightarrow K\ subscribers\ \Box$. The *live* delegation flag allows members of group $subscribers$ to further delegate the authority.

The sequel of messages, between the verifier (K) and the prover (K_s) is given below:

1. K_s sends an access request for RESOURCE. Controller K demands K_s to satisfy the access control policy enforced by the following rule:
   $K_{\text{RESOURCE}} \rightarrow K\ subscribers\ \Box$.
2. K_s requests for the definition of K's $subscribers$. (K_s should prove its membership to $subscribers$ group, defined by principal K)
3. K provides the definitions of its groups $subscribers$, $retailers$, and $customers$. In K's $customers$ definition, K_s finds the missing authorization link.

$K_{\text{RESOURCE}} \rightarrow K\ subscribers\ \Box$
$K_{\text{RESOURCE}} \rightarrow K\ customers\ \Box$; since
$\underline{K\ subscribers} \rightarrow \{K\ retailers, \underline{K\ customers}\}$
$K_{\text{RESOURCE}} \rightarrow K_3\ customers\ \Box$; since
$\underline{K\ customers} \rightarrow \{K_A, K_B, \underline{K_3\ customers}\}$ and
$\underline{K_3\ customers} \rightarrow \{K_p, K_q, K_r, \underline{K_s}\}$
$\therefore\ K_{\text{RESOURCE}} \rightarrow K_s\ \Box$

Fig. 1. K_s's Certificate Chain Discovery Composition

Fig. 1 is K_s's authorization proof inference ("certificate chain discovery"). In this manner, K_s proves its access credentials over RESOURCE and is capable of delegating the authority further. A *dead* delegation flag in the access control definition of RESOURCE will restrict K_s from further delegation. For complete certificate chain discovery algorithm (including extensions for threshold authorizations), readers are encouraged to refer [10, 6].

Such a distributed security infrastructure facilitates designing and efficiently managing complex security models. Its ability to allow users to locally define their own name and authorization binding helps achieving natural decentralized trust models, which are not rigidly dependent on global root CAs. Also, the separation of authorization from naming prevents unnecessary revelation of user's authorizations which are not required while executing a particular authority. Furthermore, provision of threshold certificates and group certificates allow a resource administrator to write the access control policies in a relatively manageable and editable fashion.

3 Modeling Security Requirement Specifications

In this section, we shall illustrate how security requirements for a wide range of applications cannot be met merely either by a PKI framework or an OS-based mechanism; but needs a mix of static and dynamic (e.g. on-line authentication) schemes. In the following section, we provide a number of examples, where we start from a simple scenario and gradually refine the requirements – formulating solutions in our envisaged framework, and the solutions possible in existing frameworks. The presentations are kept informal for want of space.

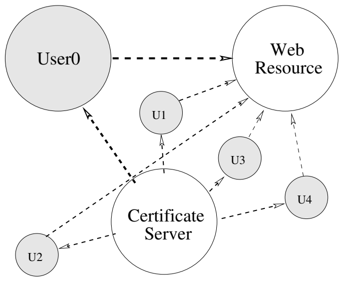

Fig. 2. Access Control with X.509

Scenario 0: Consider a *Certificate server* that can issue certificates and rights for services on a *Web Resource*, as shown in Fig. 2. Users of this setup should be able to securely authenticate themselves to the *Web Resource* and perform authorizations as specified inside the certificate. The dashed lines indicate *off-line* communication links.
Solution: A simple solution using certificates is: Users obtain digital certificate from the *Certificate Server* for accessing the *Web Resource*. Users provide the credentials acquired from *Certificate Server* as a signed access request to the *Web Resource*, which simply verifies *Certificate Server*'s digital signature over the requester's credentials and its absence in the CRL (Certificate Revocation List) provided by the *Certificate Server*.

A centralized PKI framework with a single certificate for name and authorization bindings introduces issues in scalability, hierarchical trust, and privacy. These issues are brought out in the examples below where we gradually add constraints or demand flexibility.

3.1 Scenario 1: *Is it possible to delegate authorizations and restrict the number?*

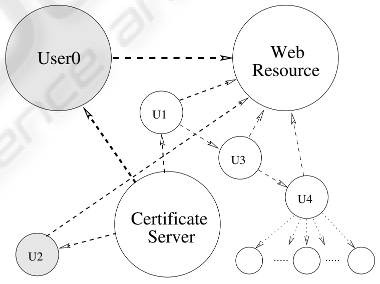

Fig. 3. Access Control with SPKI

The facility of delegation is a natural requirement in any distributed decision making framework. Fig. 3, shows the modified access control setup using SPKI (delegation is supported), where principal U_1 is acting as an authorization proxy for the *Certificate*

Server. The authorization has been further delegated to U$_3$, U$_4$, ... without any limit, unlike U$_2$ (hence shown in shade). In SPKI, the source authority cannot control the distribution of authority once it issues an authorization certificate with delegation facility to a subject. Thus, without a central database

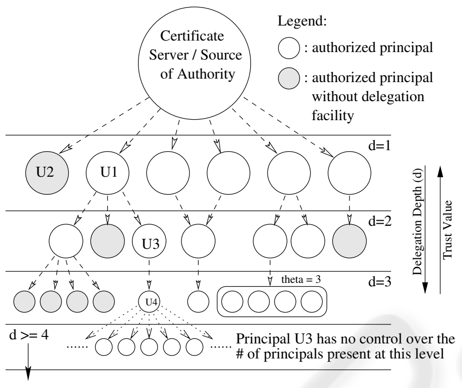
Certificate Server / Source of Authority
Legend:
: authorized principal
: authorized principal without delegation facility
d=1
U2
U1
d=2
U3
d=3
theta = 3
U4
d >= 4
Principal U3 has no control over the # of principals present at this level
Delegation Depth (d)
Trust Value

Fig. 4. Authorization Flow under SPKI

it is not possible to enforce limits on the number of authorized users in the system.
Plausible Solution: One can restrict the number of delegations in a setup through the use of an on-line central repository which must be updated upon each new assertion made by any authorized agent of the system. But such an approach will transform the setup into on-line mode and the role of sub-authority will be reduced to a proxy to the central authority. Obviously, such model is not appropriate for distributed environment like Internet, since communication failures are intrinsic to it.

3.2 Scenario 2: *Is it possible to restrict the depth of authorization delegation?*

Strictly speaking, it is not possible to restrict the depth of authorization delegation with the help of digital certificates alone (discussed below).

A principal (requester) can form an authorization flow graph (analogous to the one shown in Fig. 4) from the set of certificates it possesses, in which each vertex is a principal (either issuer and/or subject) and each directed edge between two vertices represents an authorization flow from the issuer to the subject vertex. By performing depth-first-search on such a graph [10], a principal can discover several paths originating in source authority vertex and terminating in the vertex represented by the principal itself. One of these paths can be presented as proof of that principal's authorization over the resource under consideration. And each vertex present in the proof path essentially represents one depth of authority delegation. Thus, the verifier of the authorization proof can keep a tab on the maximum depth permissible in the construction of the proof.

3.3 Scenario 3: *Can each user have different accessing rights on the web resource?*

An authorization certificate *speaks for* user's access rights. One has to revoke the authorization certificate even if a single right among the set of user's rights has to be revoked. Thus, on the basis of PKIs such provision becomes cumbersome and unmanageable over a period of time. But it is possible to provide different access rights to the user if the resource controller makes use of certificates only to perform correct user authentication and maintains a separate database (access control matrix) for each user. Such a scheme is not scalable if there are large number of users with frequent profile updates.
Solution 1: A practical approach is to categorize users into different roles and authorizing them against the rights honored against respective roles. This way the resource controller has to maintain less information in a manageable way (as done in work-flow management systems, RBAC [5]).
Solution 2: Another approach could be to explicitly specify the rights of each user into the user's certificate itself. This is suitable for a setup where users exercise rights from a well-defined fixed set of rights. Since inclusion of all rights information into a certificate will limit the scope of certificates to particular domain. Furthermore if one uses X.509 [1] to enlist user's authorizations into the certificate itself; revocation decision on a single user right will require re-issuance of the certificate apart from the privacy aspects involved in such an approach. SPKI uses separate certificates for each authorization and hence, reduces the impact of revocation to that particular authority. Further, it uses separate certificates for naming purpose, they are shielded from any authorization revocation decisions and their scope is not limited to a particular application.

3.4 Scenario 4: *Is it possible to restrict an authorized user from acting as a service proxy for others?*

In the access control setup, a user requires a signed statement from the *Certificate Server* to access the protected resource. The type of certification scheme used in the underlying PKI framework has direct effect on the setup. Adding too little information about the recipient of the authorization in the authorization statements boils down to a scheme where: *whoever produces a signed authorization request, will get an access*. Such a setup can only distinguish between authorized and un-authorized access requests. Whereas, adding too much of user information will invade user's privacy. So, the underlying scheme should be able to provide a trade-off between above two options suitable to users and the system so that the system can identify authorized users. Also, it should provide effective means of authorization communication among the users so that they can authorize other users for certain task.

- **Case 1:** *Authorizing users to act as an authorization/service proxy*: Sometimes it is required to allow the authorized users to act as proxy, as is the case of *auction robots* bidding on behalf of the user. The underlying PKI framework should be able to provide authorization delegation facility to its users, unlike hierarchical PKIs where it is only given to CAs/sub-CAs.
- **Case 2:** *Use of authorization certificates containing no additional information other than the* issuer *and* subject *public keys*: Under such setups, users enjoy full anonymity and can run *laundry service* [11] for others.

– **Case 3:** *Use of authorization certificates containing additional information e.g. user's name, address of delivery etc.*: Specifying service access point details inside the users' authorization certificate can be a solution for the scenario mentioned in Case 1. Inclusion of such details either reduces the scope of the underlying certification scheme or poses privacy threats. Hence, such an approach is not feasible. But such a setup will certainly differentiate between authorized and un-authorized users. In communications that are not face-to-face, remote lending cannot be prevented, regardless of whether privacy-protecting certificates or fully traceable identity certificates are used. Indeed, the "lender" might as well perform the entire showing protocol execution and simply relay the provided service or goods to the "borrower" [12]. In user's negative spirit, it is difficult to stop such behavior.

Further, in a centralized authorization setup (Fig. 2), as the number of users grows the load on *Certificate Server* increases. It will be of great assistance, if it could designate few other agents as a proxy for its' authorization issuance service.

An important observation is that certificates are static data-structures with cryptographic properties to convey the assertions in provable and off-line manner. To increase their scope across different applications that use them, one cannot put application-specific information in them. Also, we have seen that such an effort acts against user's privacy. The inability of these static data-structures to convey the dynamic state changes of applications that use them demands non-certificate based methods in the access control framework to express context-aware access policies. In the sequel, we will discuss solutions to the above discussed scenarios and some new scenarios that explicitly demand book keeping of state change.

4 Modeling Static and Dynamic Security Requirements

In the earlier sections, we have argued that the existing frameworks do not support the following features:

1. Constraints and flexibilities required for specifying proxies by users,
2. Variable access rights for the users,
3. Emergency access requirements, and
4. Robustness/Fault-tolerance and immediate revocation of authority.

As highlighted already, to cater to the above needs, we should have a judicious mix of certificates and on-line schemes such as authentications. Conceptually, our approach or solution addresses these through the following abstractions:

1. Abstract out the core access control across scenarios as a global policy specification that can and will be handled through certificates,
2. Specify refinements that may require on-line schemes as local policies, and
3. The overall policy is then is then obtained through a merger (e.g., intersection is an example of such merging operation)) of the global and local policies.

Employing the underlying PKI framework for secure authentication of users to the system and thereafter to establish identity, through a challenge-response based negotiation allows to filter out the efforts of authorized users to run laundry service for others. The inherent feature of delegation comes handy for proxy authorization when required. Furthermore, to derive fine-grained access control decisions, the requester negotiates with the resource controller for a list of local policies that needs to be adhered to. The resource controller may also specify policies for auxiliary rights for users based on the state information associated with that specific user/process. Integration of capability based mechanisms at this level of implementation helps us modeling capability systems for distributed applications.

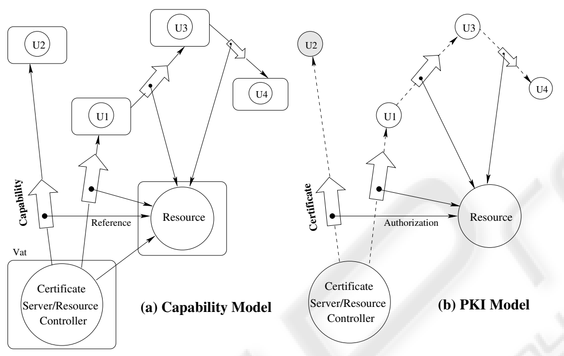

Fig. 5. Granovetter Diagrams for Capability, PKI Models

A comparative distinction between access control system under capability model and PKI model is depicted in Fig. 5. Access control system based totally on capability model is an on-line system as opposed to the off-line feature of PKI model that can turn on-line during user's negotiation phase with the resource controller for fine-grained access controller decisions. Thus, it is important to find a fine trade-off between off-line and on-line mechanisms in the design of a scalable, fault-tolerant access control system.

It is also important to see that a skillful execution of user rights in a particular sequence should not land up the system in unsafe or inconsistent state. By isolating unsafe execution of sequences of access rights from users and encapsulating a desired sequence of rights under a newly defined policy, the system can maintain safe protection state. The encapsulation of rights is achieved by restricting the communication interface of these access right mechanisms to the domain of encapsulation. A user can only interact with the interface provided by this newly defined access rights mechanism by the system. A meta-level sequencing is possible using cryptographic tokens/cookies as a communication parameter between the sequenced chain of access rights.

Immediate revocation of authority, provisions for alternative authorization in emergency, robustness, and fault-tolerance are other important aspects required in an access control mechanism. Due to lack of space, we shall not describe formal notations of flexi-acl to handle these aspects. A brief informal description is given in the next section using S-expression like notations.

5 Towards an Integrated Formalism Specifying Static and Dynamic Security Requirements

```
(acl
(subject (name
  (hash sha1 |tY8plW2ry0yQdtgk3l9oQtR=|)
   my-group))
 (delegate)
 (tag (ftp://mil.gov (read write))))
```

Fig. 6. ACL For Resource `ftp://mil.gov`

In general, the access control frameworks abstract the users of the system into groups or roles for the sake of brevity while expressing the access control policies. In doing so they lose the ability to address the user-specific policies, which is essential in the long term because *Resource Administrator* should have methods to immediately revoke only the non-conforming users from the system, instead of revoking the whole group to which the non-conforming user belongs. We have elaborated few such scenarios below. Before that let us give the templates, on which our description is based, describing the access control policy for resource `ftp://mil.gov` and an authorization certificate that allows its recipient to access this resource.

```
(cert
 (issuer (name
  (hash sha1 |tY8plW2ry0yQdtgk3l9oQtR=|)
   my-group))
(subject
  (hash sha1 |isEf64Sf5JpqasB4DCsR6Bn=|))
(not-before "2004-04-01_00:00:00")
(not-after "2005-05-01_00:00:00"))
```

Fig. 7. Name/Group Certificate For Resource `ftp://mil.gov`

Fig. 6 is the access control policy for `ftp://mil.gov` with two distinct permissions (`read`, `write`). The requester for this resource must prove that it belongs to `my-group`, a group defined by the `subject` principal, listed inside the ACL. Many such local or extended name definitions may be listed in the ACL.

Fig. 7 shows a membership certificate empowering its subject to `ftp://mil.gov` with full permissions automatically. Inheritance of these permissions allow the members of `my-group` to delegate the authorizations to others due to presence of `(delegate)` flag in the ACL of resource. Partial delegation of authorization is shown in Fig. 8 (only `read` permission is delegated to the `subject`). Authorization certificate is used to transfer authorizations (permission set) and as authorizations are delegated further, the permission set can only reduce.

```
(cert
 (issuer
  (hash sha1 |isEf64Sf5JpqasB4DCsR6Bn=|))
(subject
  (hash sha1 |ZdtyR5TuOr2sXgtyuoqGbET=|)
 (delegate) (tag (ftp://mil.gov (read)))
 (not-before "2004-04-01_00:00:00")
 (not-after "2005-05-01_00:00:00"))
```

Fig. 8. Authorization Certificate For Resource `ftp://mil.gov` With `read` Permission

From this setup, it should be clear that users belonging to the group definition `my-group` can access the resource with full permissions (both `read` and `write`); however, the user authorized by the certificate shown in Fig. 8 can access the resource only with `read` permission. The *Resource Administrator* is free to update the ACL as the policy changes over the period of time. Therefore, the actual permissions available to a requester has to be computed accordingly, which is done by taking the intersection of permissions listed inside `(tag)` fields of authorization certificates present in the "certificate chain". The `subject` principal listed in Fig. 8 deduces its authorization proof with the help of certificates shown in Fig. 6, 7, and 8 accessed in the same order. The actual set of permissions available to this principal are computed as described below using `AIntersect (t1, t2)` function, where `t1` and `t2` are `(tag)` fields from authorization certificates. Therefore,

```
AIntersect ( (tag (ftp://mil.gov (read write)))), (tag (ftp://mil.gov (read))) ) =
(tag (ftp://mil.gov (read)))
```

Intersection of tag sets:

– basic: if `t1 == t2`, then the result is `t1`
– basic: if `t1 != t2` and neither has a `*`-form, then the result is "`null`".
– `(tag (*))`: if `t1 == (tag (*))`, then the result is `t2`.
  If `t2 == (tag (*))`, then the result is `t1`.
– `(* set ...)`: if some `(tag)` S-expression contains a `(* set )` construct, then one expands the set and does the intersection of the resulting simpler S-expression.
– `(* prefix ...)`: if some `(tag)` field compares a `(* prefix )` to a byte string, then the result is the explicit string if the test string is a prefix of it and otherwise "`null`".

Let us study the behavior of *Resource Administrator* as different policy requirements gradually arise in the setup.

– **Scenario 1:** Stop the service to members of `my-group` defined by public-key `(hash sha1 |tY8plW2ry0yQdtgk3l9oQtR=|)`.
  Solution: The *Resource Administrator* removes the rule that authorizes members of `my-group`, from the ACL.
– **Scenario 2:** Resource `ftp://mil.gov` not in a position to provide `write` permission.
  Solution: The *Resource Administrator* updates the `tag` field of corresponding ACL entry to `(tag (ftp://mil.gov (read)))`

- **Scenario 3:** Introduce a new permission `foo` over the resource `ftp://mil.gov`.
  Solution: The *Resource Administrator* modifies the `(tag)` field of corresponding ACL entry to `(tag (read write foo))`; provided that the `(tag)` field of user's authorization certificate is `(tag (*))` so that upon taking intersection the effective set of permissions include `foo`.
- **Scenario 4:** Restrict the resource service only to the members of `my-group`.
  Solution: The *Resource Administrator* removes the delegation flag `(delegate)` from the ACL rule for group `my-group`.

In the above scenarios, the actions of *Resource Administrator* are enforced on a set of users in a generalized fashion. Such an approach falls short of expectations when the change in policy is directed towards a subset of a well-defined group or an individual. The main reason behind this shortcoming is the inability of the underlying framework to efficiently capture the system's state information through just certificates. One solution is to issue new certificates as and when the state constraint changes. However, this is too costly and does not serve the rationale of PKIs and further, one need to handle the problem of revocations.

In our approach towards a flexible access control framework, we integrate *a priori* defined heterogeneous e-authentication mechanisms to attain the goals of capturing state information, instant revocation of authority, fine-grained access control, and provision for alternate methods of authorization. We also pursue a new approach in certification mechanism; the authorization certificates under our framework include more general (and relatively static) permissions inside the `(tag)` field followed by `(*)`, i.e. `(tag (resource (foo1 bar1)(*)))`. Whereas the `(tag)` structure of ACL contains a `(*)` followed by the set of frequently up-datable permissions over the resource, i.e. `(tag (resource (*)(foo2 bar2)))`. This approach segregates the more general access policies (global policy) imbibed into the authorization certificates of the users and access permissions that are subject to change depending upon state of the system or context (local policy) are expressed inside the ACL entries. The effective policy for resource access is a combination of general policy, specified inside the requester's credentials, and local policy specified at the resource in conjunction with the e-authentication mechanisms for conditional decision making. Another tool in our integrated approach for flexible access control framework is the tokenization of user credentials. Such an abstraction of user's credentials makes the framework modular and efficient, if the trust among the different verifiers in the setup is high. Also, the abstraction induces a level of indirection in user's authorization proof; thus providing the much needed property for e-transactions – *privacy.*

5.1 Access Control Policy Using e-authentication Mechanisms

The communication between the resource requester and the resource is an interactive process during which the resource controller challenges the requester with access control rules for the resource. Fig. 9 shows a typical access control policy designed using

```
(flexiacl
  (quorum 1
    (acl-block B1
      (rule X1) AND (rule X2) AND (rule X3))
    (acl-block B2
      (rule Y3) OR (rule Y4) AND (rule Y6))))
```

Fig. 9. *flexi*-ACL: Typical Structure

heterogeneous e-authentication mechanisms (like `pamd`, `RSA SecurID`, `biometric`, `TCP/IP wrapper` etc.) as basic access control rules[1]. The heterogeneous authentication mechanisms are glued together using the boolean `AND`, `OR` operators.

Table 1. `AND`, `OR` Operators

Expression	Description
`(rule X1) AND (rule X2)`	returns 1; when *evaluate* `(rule X1)` = 1 and *evaluate* `(rule X2)` = 1, else returns 0
`(rule Y3) OR (rule Y4)`	returns 0; when *evaluate* `(rule Y3)` = 0 and *evaluate* `(rule Y4)` = 0, else returns 1

The heterogeneous primitive access control rules can be aggregated into an `acl-block`. Such blocks can be placed under another construct called `quorum` to achieve a very flexible access control policy expression. Such an unique, modular method of expressing access control policies allows the *Resource Administrator* to establish a comfortable trust or credibility level before making access control decisions. Hence, for the policy specified in Fig. 9, the requester has to conform to any of the two `acl-block`s to get resource access, since the `quorum` is set to 1.
In the following, we show features like instant authority revocation, state-based access decisions, rights amplification, and fine-grained access control that can be effectively and elegantly addressed in our approach; note that these features cannot be handled in the existing access control frameworks.

- **Scenario 5:** Stop providing service to non-conforming users.
  Solution: The *Resource Administrator* simply revokes non-conforming users from the authentication mechanism. Since the resource has been conditionally integrated with *a priori* defined e-authentication mechanism, a non-conforming user can produce certificate based authorization proof but could not authenticate itself against the authentication mechanism.
- **Scenario 6:** A particular user `U` should not access the resource more than *n* times.
  Solution: For such scenarios, the *Resource Administrator* may integrate the resource with an e-authentication mechanism like “one-time password” and issue *n* passwords to the user.

[1] The complete details of *flexi*-ACL specifications are available in [7]

- **Scenario 7:** Provide more permissions to the users who can satisfy the additional policy requirements. (rights amplification)
  Solution: The *Resource Administrator* can put the additional policy requirements in conjunction with the necessary e-authentication mechanism as a conformance check.
- **Scenario 8:** Introduce a new permission `foo` over the resource only to certain users.
  Solution: The *Resource Administrator* will create a new rule inside the `(acl-block)` in which permission `foo` is introduced as a new permission under local policy but availed to the users who can authenticate themselves against the e-authentication mechanism integrated with that rule using `AND` operator.

More complex and flexible policies can be envisaged using our approach. For lack of space, we will informally describe additional properties like sequential access and alternative/emergency access provisions using an abstract comprehensive scenario in the next subsection.

5.2 A Comprehensive Scenario

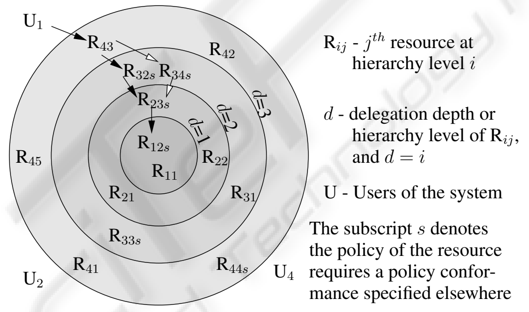
U_1
R_{43}
R_{32s} R_{34s}
R_{42}
R_{23s}
d=3
d=2
d=1
R_{12s}
R_{22}
R_{11}
R_{45}
R_{21}
R_{31}
R_{33s}
R_{41}
R_{44s}
U_2
U_4
R_{ij} - j^{th} resource at hierarchy level i
d - delegation depth or hierarchy level of R_{ij}, and $d = i$
U - Users of the system
The subscript s denotes the policy of the resource requires a policy conformance specified elsewhere

Fig. 10. Layered Security Infrastructure

In an environment where the resources are widely distributed and have varying levels of sensitivity towards unauthorized resource access, a common access control policy cannot be envisaged due to various reasons. Let us explain a scenario using which we will demonstrate our framework.

Consider an organization that has installed its resources (R_{11}, R_{22}, R_{33s}, R_{43} etc.) in hierarchical layers closely resembling the descending organization hierarchy. The first subscript of a resource R denotes its placement in hierarchy level, second subscript denotes its identity number and the third optional subscript indicates sequential access constraints over the resource (i.e. access request to such a resource should be made after complying to the policies specified at the resource prior/lower to it in the hierarchy).

Members of the organization are authorized to access the resources lying in their hierarchical level; conditional access to resources at higher hierarchical levels can be granted. Also, it should not be possible for a member to lend out its credentials to others. Access to certain resources must start at the lowest level of hierarchy and should proceed further in strict sequence. The internal network of the organization is assumed to be a trusted and resource access requests originating from outside the network will not be honored with full authorization capability. Also, the resources should have the facility to implement context-based access control policies.

Let the head of the organization be the source of authority. It empowers the subordinates in the immediate hierarchy for accessing the resources and allows further delegation of the authorization. This way the authorization flows from top level of hierarchy to lower levels. Now let us analyze the constraints and options available to principal U_1 for accessing the resource R_{12s}. As mentioned earlier, the resource subscript s denotes that it can only be accessed in a particular sequence. The directed paths shown in Fig. 10 are such sequences. Therefore, in the process of accessing resource R_{12s}:

– principal U_1 has to first comply with policy defined at resource R_{43}, which is installed at third level. R_{43} issues a token for policy conformance to U_1
– The signed tokens issued by R_{43} are acceptable as part of authorization proof at resource R_{32s} and R_{34s}
– Similarly, tokens issued by R_{32s} and R_{34s} are accepted at R_{23s}
– And resource R_{12s} accepts only the tokens issued by R_{23s}

In this fashion, one can deploy a layered access control policies over a large setup. The concept of tokenization should work across the administrative domains, based upon the level of trust between the tokenizer and the acceptor of such tokens.

6 Conclusion

In this paper, we have argued the need for a hybrid of digital certificates and other state based schemes to arrive at flexible distributed access control specifications. We have designed and experimenting the notations for flexible access control mechanism, called *flexi*-ACL [7], to achieve the above mentioned dynamic access control decisions in distributed systems.

Our access control model allows to make a judicious mix of on-line decision making protocols with the underlying off-line framework to capture the dynamics of modern access control requirements. Inclusion of external authentication mechanisms into the PKI framework empowers the resource controller to provide fine grained access control. The ability of resource controller to enforce local access control policies helps the resource owner in granting discretionary auxiliary rights to users. Such provisions also allow the resource controller to do rights amplification after a user achieves credibility by following the respective local policies. This scheme can easily express the complex access control requirements of applications like e-government, work-flow etc. in a manageable way.

We also observe that in the financial transactions, technical requirements form only a part of the overall requirements for security. Non-technical aspects include the reputation of the parties involved, liability for misuse, (legal action and stiff penalties for wrong doing), etc.

Acknowledgments

We are grateful to the Ministry of Information Technology, Government of India, for the support.

References

1. Warwick Ford and Michael S. Baum. *Secure Electronic Commerce: Building the Infrastructure for Digital Signatures and Encryption, Second Ed.* Prentice Hall, 2002.
2. Mark Miller and Jonathan Shapiro. Paradigm Regained: Abstraction Mechanisms for Access Control. *Advances in Computing Science, ASIAN 2003 Programming Languages and Distributed Computation*, LNCS 2896:224–242, 2003.
3. Matt Blaze, Joan Feigenbaum, and Martin Strauss. Compliance Checking in the PolicyMaker Trust Management System. *Financial Cryptography, FC 1998*, LNCS 1465:254–274, 1998.
4. Matt Blaze, Joan Feigenbaum, John Ioannidis, and Angelos Keromytis. The KeyNote Trust-Management System Version 2. RFC 2704, Internet Engineering Task Force, 1999.
5. David F. Ferraiolo, Ravi Sandhu, Serban Gavrila, D. Richard Kuhn, and R. Chandramouli. A Proposed Standard for Role-Based Access Control. Technical report, NIST, Dec 2000.
6. Carl Ellison, Bill Frantz, Butler Lampson, Ronald Rivest, Brian Thomas, and Tatu Ylonen. SPKI Certificate Theory. RFC 2693, Internet Engineering Task Force, Sep. 1999.
7. Vishwas Patil and R.K. Shyamasundar. Notations for Flexible Access Control System: *flexi*-ACL. Technical report, Tata Institute of Fundamental Research, 2003.
8. Ronald Rivest and Butler Lampson. SDSI – A Simple Distributed Security Infrastructure. Presented at CRYPTO'96 Rumpsession, 1996.
9. Carl Ellison. SPKI/SDSI Certificate Documentation, 2002. `http://world.std.com/~cme/html/spki.html`.
10. Dwaine Clarke, Jean-Emile Elien, Carl Ellison, Matt Fredette, Alexander Morcos, and Ronald Rivest. Certificate Chain Discovery in SPKI/SDSI. *Journal of Computer Security*, 9(4):285–322, 2001.
11. Mark Miller, Chip Morningstar, and Bill Frantz. Capability-Based Financial Instruments. *Financial Cryptography, FC 2000*, LNCS 1962:349–378, 2001.
12. Stefan Brands. *Rethinking Public Key Infrastructures and Digital Certificates; Building in Privacy*. MIT Press, 2000. ISBN 0-262-02491-8.

[2] This document is a draft of the work undergoing on Notations for Flexible Access Control System: *flexi*-ACL