Dynamic policy based model for evolution of trust in P2P applications

Madhumita Chatterjee, G.Sivakumar, and Menezes Bernard

Indian Institute of Technology, Mumbai, India
madhumita@cse.iitb.ac.in, siva@cse.iitb.ac.in, bernard@cse.iitb.ac.in

Abstract. The decentralized and anonymous characteristics of P2P environments, necessitates the need for an access control model, based on trust and recommendation. Since peers are heterogeneous, some peers might be malicious and in the absence of a centralized authority to punish such peers, mechanisms for trust and recommendation can help to distinguish between good and bad peers. In this paper we propose a dynamic multi-level access control framework for P2P applications which integrates policy based access control with trust based access control. We use a prolog like language to model the dynamic access control policies.

1 Introduction

Trust is an important component in a security infrastructure for communication between peers in P2P Collaborative applications like video/audio conferencing, IP telephony, file sharing, collaborative work spaces, and multi-user games, having varied security requirements.

The evolution of dynamic, self organizing groups like wikipedia, f/oss and other business community groups motivates new security requirements for peer to peer communication. Consider an example of a developer’s community for some critical security related open-source software. The communication forum could be either chat or e-mail. New users would be allowed into this forum to put in their suggestions, only if they are introduced by an existing member. Depending on the sensitivity level of the code being developed, there could be hierarchial levels for the members of this forum. New members and existing members could be periodically rated or evaluated by existing members based upon the quality of their contribution. A member receiving a high rating value could be elevated to a higher level in the forum. A peer or a member in any level need not always be evaluating other peers, i.e a member can have different roles in a group. For critical software like security applications, authentication of developing members would be important as well as deciding which member is allowed to play which role and participate in which level.

In the dynamic P2P communities where peers are unknown to each other and uncertain about each other’s reputation it is necessary to develop strategies for establishing trust among peers. Peers in self-organising groups with multiple levels, should have the right to collaboratively modify the access control policies governing their levels and the levels below them. Current solutions do not allow dynamic change of access levels or policies, with incremental building of trust during the communication. Nor do they allow dynamic authentication of participants who were previously part of the group and later wish to rejoin. In view of the varied security requirements for diverse peer to peer applications, we propose an integrated framework which has a flexible, dynamic and totally decentralized access control mechanism, integrated with authentication and dynamic rekeying.

A key concept of our framework is introduction of unequal roles for peers in a group. There can be different levels within a group and peers at each level can have different functionalities. Peers in one level can dynamically elevate their levels in a group. Group composition determines join priorities. We model a new framework for dynamic multi-level policies. We propose some metrics for trust evaluation in multiple context dimensions. We further propose to analyze the pros and cons of different trust settings and different policies and study their effect on the evolution of P2P groups. As a preliminary step we have modelled our protocols for join, leave and updation in a peer to peer Java based simulator Peersim, and are currently analyzing the effect of different policies in the presence of malicious peers in our framework.
2 Proposed Integrated Framework

In the decentralized and dynamic peer group scenario it is necessary to have an adaptive access control mechanism where peers have the right to collaboratively make access control decisions, modify these decisions and frame rules based on attributes as well as trust evolved in the group. We propose a hybrid access control framework which integrates policy based access control and trust based access control along with multi-level access control.

2.1 Model Overview

We consider a peer based model where every peer $P_i$ has a unique user identity $UUID_i$ associated with it. A group is a named set of peers or a subset of a community that is governed under a set of rules that describe the conditions required to be part of the group, and is formed based on a particular interest criterion. The dynamic membership of the group implies that no peer will be highly available. The challenges are rapid admission of new members with appropriate access rights, dynamic elevation of access levels of existing members and dynamic revocation of access rights of leaving or defaulting members.

Basic Assumptions The following are the assumptions made in our model.

– No centralized control: Traditional access control models such as ACL or RBAC [3] rely on central servers for authorization. In our model a peer has a high level of autonomy and can frame and manage his own policies.

– Peers in the P2P system are loosely coupled and interacting partners are unknown.

– Peers in a group can belong to different levels within the group and can perform several roles.

– A role of a peer is independent of the level which the peer is in.

– Peers are individually capable of performing tasks of authentication, voting, access control, key management etc.

– Peers can compute their own public-private key pair and provide self signed certificates, binding their identity with their public key.

Each group has a group policy which describes the roles permitted in the group and the rules associated with each role. The functions that a peer is capable of performing in a group are storing and verifying certificates, authentication, voting, updation of levels, key management etc. The extent to which a peer performs these tasks is decided by the role the peer wants to play in the group. We identify the following roles:

1. **Member peer** A peer in this role participates in the activities of the group like gaming, conferencing, file transfer etc, but does not contribute to the admission of new peers, nor updation of levels of existing peers. Thus this peer need not store certificates of every other peer and need not invoke voting either.

2. **Admission Control Peer** A peer with this role would participate in the voting algorithm, and would therefore store public keys of every other peer. However it need not store the updated ratings.

3. **Maximal role peer** A peer in this role would have all the functional components and would participate in updation levels of existing peers also.

The access policy would decide the role a peer assumes, based on the request of the peer and his credentials as well as rating calculated by the other peers.

3 Trust Management in P2P Groups

Trust management has been an important research area in the development of modern distributed systems, mainly P2P applications. There are different approaches to trust management namely policy based trust management and reputation based trust management. Policy based TM has been proposed as a solution to the problem of authentication and access control in P2P systems. The TM mechanisms employ different policy languages based on a set of credentials and a set of policies. It is possible to formalise trust with rule based policy languages. Policy based trust is typically involved in access control decisions.
3.1 Policy based framework for trust management

One major contribution of our work is our Policy Driven Trust Based Access control framework, wherein it is possible to deploy access control policies flexibly and dynamically. We consider groups having multiple levels where peers in each level can have different functionalities. Peers in one level are allowed to dynamically be updated to a higher level. A peer could also be collaboratively ejected from a higher level and forced to a lower level if his behavior in the group degrades. The group policy can also dynamically prioritize requests for join. Join priorities would depend on current group composition. Thus low level requests for join could be postponed in a group which already has a large number of members performing the lower roles. It is also possible for peers at the highest level to introduce a new level into the group if the situation so demands. To the best of our knowledge a group with such enhanced peer capabilities has not been discussed so far in the group communication systems.

3.2 Group composition and Policies

Our model focusses on the genetic evolvement of groups based on group composition. Multiple policies can apply to a group simultaneously, or the system can switch between policies over time, influenced by the system’s state or environment. Some policies can have higher priorities than others. Policies can apply to different levels of the system and can be global or apply to only parts of the system. Currently we define two levels of policies viz:

Global policy This would be the policy initiated by the creator of the group. The peers can contribute to policy rules by dynamically adding new rules. The meta policy would contain rules to specify the threshold of peers required and conditions and constraints to be satisfied. As an example the global policy could have rules like:

- A peer can join as a member peer at level 1 of the group if he has received votes greater than 50% and if his current rating is greater than some threshold x.
- A peer can join as control peer at level 1 of the group if he has received votes greater than 70% and if his current rating is greater than some threshold y.
- To update level a peer must have an accumulated trust value > x and should be an authenticated member at the lower level.

Domain specific meta policy

This set of policies would inherit the properties of the global policy and would contain application specific policies. So policy rules for an e-commerce scenario, online gaming scenario or online software development would be specified here depending on the application. The performance of the group would depend upon policy rules like:

- For a new policy to be framed at least 75% of existing peers with trust value > some threshold x must collaborate.
- Number of member level peers to be permitted to complete a specific task.
- Number of maximal role peers required.
- The group policy can also dynamically prioritize requests for join.
- Low level requests for join could be postponed in a group which already has a large number of members performing the lower roles.
- For a new level to be introduced at least y no. of peers must be present in the current highest level and all 100 % should collaborate.

3.3 Modelling Policies and their dynamic environment

In RBAC, each role has a set of privileges for operating on some resources. Access permissions are associated with roles rather than individual users. In policy driven RBAC, the policy description language besides being expressive and easy to use must be flexible enough to allow extension of a policy by adding or modifying its constraints easily, without affecting existing policies. We use a logic programming system to realize our access control policies. Sample policies for F/OSS require information such as assignment of projects to reviewers, assigning moderators to projects reviewed and so on. We capture the policies using Prolog rules. A prolog rule is an expression of the form
where $R_i$ are predicates and $u_i$ are (possibly empty) tuples of variables and constants. The head of the rule is $Ro(u_0)$ and the sequence of formulae on the R.H.S is the body of the rule. We chose Prolog as our policy description language because of the following features.

- It is declarative. A rule in PROLOG defines a relationship between several objects
- It is based on a subset of First Order Logic, thus properties such as soundness and completeness can be guaranteed.
- It supports back tracking and can express non-deterministic constraints.
- It is a unification based language which allows writing policy templates.
- It is a productive modelling language supporting incremental policy writing and refinement.
- It is possible to reason from a set of Prolog rules, supports meta-level reasoning, thus making policy conflict detection possible.
- Dynamic rules can be modelled using the assert and retract clauses.

The Prolog inference engine provides a mechanism to derive consistent access control decisions at runtime. It may also be used to analyse the correctness and consistency of access control and other rules. Access control rules, defined in Prolog, can be more expressive than the traditional (subject, object, action) tuple. The access control policy is machine readable and directly under the control of the administrator. It also possesses dynamic updating capabilities. New facts from independent policy sources can thus be added to the policy base before decisions are made, ensuring dynamic decisions at runtime.

A policy interacts with its dynamic environment by consulting facts in the environment and constraining certain actions in the environment. We model such interactions through events that share same names as actions in policy requests. For e.g a transition labelled $join_{req}()$ would correspond to a join request sent to the policy. We define the following components in our policy language framework:

- Facts: These are represented as Prolog unit clauses.
- Rules: Rules are defined by a Prolog program which when presented with a goal, representing a regulated event triggered by request from a peer, is evaluated by the policy engine in appropriate context.
- Role Sets (RS): This comprises of the different roles permitted.
- Group Hierarchy (GH): These are the different levels possible in the group and would depend upon the attributes of the peers and the composition of the group.
- Service Set (SRVC): This is the possible set of operations permitted within the group.
- Events: Regulated events are subject to rules and are triggered by peers. For e.g there could be a join event or an update event. A $JOIN_{REQ}$ or an $UPDATE_{REQ}$ is defined as a triple $< role, level, srv >$ where role $\in$ RS, level $\in$ GH and srv $\in$ SRVC. Based on the join or update request, the system determines the applicable access policy for the requested service. The access policy for a role service pair is a set of clauses consisting of multiple access conditions.
- Control States. Each peer $i$ has an associated control state $CS_i$ consisting of a set of Prolog terms called attributes of $i$. The control state of a peer represents the role of the peer (maybe member-peer, admission peer or maximal role peer) and its level (maybe developer, reviewer or moderator for F/OSS application).

3.4 Goal reachability

Goal reachability is a very useful and a special case of first order model checking. A question in Prolog is a sequence of goals. Prolog tries to satisfy a question by trying to satisfy all goals in the question in left to right order and top to bottom. If a sub-goal cannot be satisfied this failure is passed back and prolog backtracks by trying to resatisfy the previous goal.

3.5 Dynamic Access Control Policies

Our hybrid access control model is based on an integration of collaborative roles and trust. A role represents a set of privileges and rights. It involves meta-level policies in regard to admission of users to the role. Roles are globally defined. Users can exist in different hierarchical levels in the group and can be assigned roles in these levels. Users assigned to
a role are dynamically assigned privileges and tasks based on the context. The privileges
assigned to a user in a role may change with time due to the behavior of peers in the
group. Thus permissions need to be assigned to roles dynamically based on the context.
Integration of the roles, tasks, privileges and security policies needs a proper framework for
policy specification. We are able to express dynamic rules at runtime, as well as change
existing policies depending on the status of the group at runtime by exploiting the assert
and retract clauses of Prolog.

Global Policies
An example of rules for peer to join as a member peer or admission peer in a group at level1
of a groups is

\[
\text{accept}(N\_peer, \text{join}, \text{Role}, \text{level1}) :\text{-} \text{join}\_\text{req}(N\_\text{peer}, \text{member}, \text{level1}), \text{num}\_\text{votes}(N\_\text{peer}, V), V > 50, \text{rating}(N\_\text{peer}, R), R > 2. \quad \text{Rule1}
\]

\[
\text{accept}(N\_\text{peer}, \text{join}, \text{Role}, \text{level1}) :\text{-} \text{join}\_\text{req}(N\_\text{peer}, \text{admission}, \text{level1}), \text{num}\_\text{votes}(N\_\text{peer}, V), V > 50, \text{rating}(N\_\text{peer}, R), R > 4. \quad \text{Rule2}
\]

Rule1 states that if a peer with some ID and initial rating 2 gives a request to join a group
in level1 as member peer and the number of peers who vote yes is greater than 50% then he
is allowed to join.

Meta Policies
new\_join\_rule(N\_peer, \text{join}, \text{member}, \text{level2}) :\text{-} \text{join}\_\text{req}(N\_\text{peer}, \text{member}, \text{level2}), \text{num}\_\text{member}\_\text{peers}(MP), MP > 30,
retract((\text{accept}(N\_\text{peer}, \text{join}, \text{Role}, \text{level2}) :\text{-} \text{join}\_\text{req}(N\_\text{peer}, \text{member}, \text{level2}), \text{num}\_\text{votes}(N\_\text{peer}, V), V > 60, \text{rating}(N\_\text{peer}, R), R > 6))
asserta((\text{accept}(N\_\text{peer}, \text{join}, \text{member}, \text{level2}) :\text{-} \text{join}\_\text{req}(N\_\text{peer}, \text{member}, \text{level2}), \text{num}\_\text{votes}(N\_\text{peer}, V), V > 50, \text{num}\_\text{member}\_\text{peers}(MP), MP < 20, \text{rating}(N\_\text{peer}, R), R > 6)). \quad \text{Rule3}

Rule 3 is one of the rules which expresses dynamism. If a peer requests to join the group as
a member peer after the group has evolved for some time and the no.of member peers in the
group is greater than 50% of current group size then the previous join rule is deleted from
the database and a new join rule is added, thereby postponing low-level requests for join.

verify(N\_peer, Request, Rl, Level, Rate, Vote) :- Level1 is Level-1,
belongs(N\_peer, Level1), update(Request), member(Rl), Level =:= 2,
Rate >= 4, retract(belongs(N\_peer, Level1)), assert(belongs(N\_peer, Level)).
update\_engine(Levels, MPs, APs, CPs):- MP$s > 10,
retract((\text{verify}(N\_\text{peer}, \text{Request}, Rl, \text{Level}, \text{Rate}, \text{Vote}) :\text{-} \text{Level1 is Level-1,}
belongs(N\_\text{peer}, \text{Level1}), update(\text{Request}), member(Rl), Level =:= 2, \text{Rate} >= 4, retract(belongs(N\_\text{peer}, \text{Level1})),
assert(belongs(N\_\text{peer}, Level))))
assert((\text{verify}(N\_\text{peer}, \text{Request}, Rl, \text{Level}, \text{Rate}, \text{Vote}) :\text{-} \text{Level1 is Level-1,}
belongs(N\_\text{peer}, \text{Level1}), update(\text{Request}), member(Rl), Level =:= 2, \text{Rate} >= 5, retract(belongs(N\_\text{peer}, \text{Level1})),
assert(belongs(N\_\text{peer}, Level))))).

4 Trust evaluation for multiple context dimensions in
Incentive based system
In an incentive based reputation system different context categories can be incorporated into
the basic trust metric with a weight function. For example for an e-commerce transaction,
peers can submit feedback about each other with different parameters such as quality of product, delivery time, payment reliability, etc. The transaction cost could also be taken into account. The final trust would then be a weighted aggregation of feedback along these different parameters multiplied by the transaction cost for high-value transactions. Similarly for a software development community peers would submit feedback along dimensions such as quality of software, Lines of Code, completion time and reliability.

We propose a context based trust model that determines the weight function for such different parameters and integrates the different extensions into our previous trust metric.

4.1 Context Sensitive Trust Model

We identify different factors for evaluating trust of a peer in different application scenarios
1. Feedback in terms of reputation of a peer as computed by peers with whom it had direct interactions.
2. Reputation computed on the basis of indirect recommendations given by unknown peers.
3. Total number of transactions and cost of transaction in e-commerce scenario.
4. Credibility of peer giving the feedback.
5. Context specific attributes associated with the transactions and current trust value of peer u as calculated by peer v based on the attributes
6. Trust value of peer u based on past history
7. Weightage to be given to recent interactions.

Weights and attributes
We assign weights with different attributes depending on the type of application. Let x be the set consisting of the different attributes. \( x = \{x_1, x_2, \ldots, x_n\} \)

Relative importance assigned to each attribute can be modelled as weight \( w_{x_i} \) such that \( \sum w_{x_i} = 1 \).

Let \( I_u \) denote the total number of transactions performed by peer u and \( P \) denote the set of peers.

\( R_L_u \) is the global rating of peer u.
\( R_L_v \) is the rating of peer v.
\( T_{uv} \) is the trust value of u as computed by v
\( C(u,i) \) is the cost associated with the ith transaction of peer u

Basic Trust Metric

In the basic trust metric we compute peer u’s trust as evaluated by every peer v with whom u has had direct interactions as well as all peer k who recommend peer u based on indirect interactions.

\[
R_L_u = \frac{\sum_{v \in P, v \neq u} T_{uv} \cdot R_L_v}{I_u} + \sum T_{uk} \cdot R_L_k
\]

Peer v evaluates peer u against attributes \( x_i \) and the rating thus computed is multiplied by \( w_{x_i} \), which is the weight associated with attribute \( x_i \).

Thus \( \sum_{v} T_{uv} = \sum R_{x_i} \cdot w_{x_i} \)

For e-commerce domain

In this domain we identify the attribute set \( x = \{P, D, Q, S, R, T\} \)

where

1. P — price of product.
2. D — Delivery time
3. R — Reliability of payment
4. S — prompt shipment
5. Q — Quality of product
6. T — quantum of trust in unknown

To account for high value transactions we add the cost of the transaction and fine tune its weightage with a constant.

\[
R_L_u = \frac{\sum_{v \in P, v \neq u} T_{uv} \cdot R_L_v \cdot \alpha \cdot C(u,i)}{I_u} + \beta \cdot \sum T_{uk} \cdot R_L_k
\]
For F/oss domain

The attribute set for this domain would consist of parameters like...

1. P → price of software.
2. D → Delivery time
3. R → Lines of Code
4. Q → Quality

Adding incentives for rating

\[ RI_u = \frac{\sum_{v \in P, v \neq u} T_{uv} \ast RI_v \ast \alpha \ast C(u, i)}{I_u} + \beta \ast \sum T_{uk} \ast RI_k + \gamma \ast \frac{F_u}{I_u} \]

where \( F_u \) is the total number of feedback ratings given by peer \( u \). \( \alpha \) and \( \gamma \) are the fine tuning constants to control the amount of weightage to be given to transaction cost and amount of reputation gained by rating others respectively. \( \beta \) is the fine tuning constant to control the quantum of trust to be placed in indirect recommendations.

5 Simulation and Experiments

We have currently modelled our protocols using a simulator in order to test the performance. The protocols for join, leave, updation of peers have been successfully implemented in the Java based simulator for peer to peer groups viz. Peersim. Experimental runs were done with different policy rules and by varying the no of peers in the group and the composition of the group. We are working in event-driven mode, where controls have to be scheduled explicitly since there are no cycles. The policy rules are specified in Prolog and we are using SWI-Prolog for the interaction between Java and Prolog.

6 Conclusion and Future Work

Currently there are some integrated solutions for secure group communication, but very few allow the development of trust and switching of users between hierarchial levels in an incremental fashion. We have modeled an integrated framework for decentralized groups, which allows us to implement secure communication between different peers using admission and trust based multi-level access control along with key management. Our dynamic access control policy permits peers to collaboratively frame, and modify policies at various levels. The policy also permits peers to update their access rights within a group. Join priorities can also be decided by the group policy and depend upon current group composition. Idea of per-session authentication is also proposed which depends on the incremental trust factor and eliminates the need of the voting process for a old member of the group who wishes to re-join.

Though we have not done a detailed security analysis of our protocols, we have taken care to see that the necessary authentication is done when a new member wishes to join (section x x) or an existing member requests for level updation (section x x). Secrecy in different levels is maintained by an efficient key management algorithm. Peers in different levels maintain different sets of keys and encrypt data with an appropriate group key. Backward secrecy is maintained by doing rekeying at appropriate places.

A more detailed security analysis would involve taking cognizance of the threats to this model. Common threats from malicious peers, free-riders and whitewashers have been discussed by Jongpil Yoon and A. Ravichandran [17]. These could easily be handled in our framework. However threats posed by collusion of existing peers needs detailed analysis and is in the scope of our future work.

We would like to focus on the policies for composition of groups in the dynamic scenario. Future work would also include measuring the behavior of the group, with varying group compositions. Measuring intra-group trust level and a focus on the behavior of the group in presence of malicious peers are the other directions of work proposed.
References


7. Thomas Repantis, Vana Kalogeraki.: Decentralized Trust Management for AdHoc Peer-to-Peer Networks. MPAC, November 27-December 1, ACM 2006


