Policy Based Framework for Trust Management and Evolution of Peer to Peer Groups.

Madhumita Chatterjee

Advisors:

Prof G. Sivakumar &
Prof Menezes. Bernard
Presentation Outline

- Overview of Collaborative Groups
- Challenges and Issues
- Goal of Thesis/Problem Definition
- Thesis Contributions
- StateChart Model
- Dynamic Policy Driven Trust Based Access Control
- Simulations and Analysis
- Adaptive Multi-dimensional Trust Model
- Simulations and Results
- Conclusion and Future Scope
- Responses to examiner reviews.
Collaborative Groups

F/OSS

collaborative work-space

Formation

distance learning

online gaming

Member join

Member leave

Group partition

Group merge
Challenges and issues in collaborative peer groups

- The group should have an admission and access control framework which can
  - Control admission of peers.
  - Support different functional roles for peers in self-organizing groups.
  - Permit peers to collaboratively frame modify policies for group management based on group composition and evolving trust.
  - Permit peers to collaboratively re-organize the group structure and change the system model to optimize group performance.
  - Reduce the impact of malicious peers on the system by using an adaptive trust model.
- Motivate non-malicious peers to provide best possible service
Goals

- To design an integrated system with dynamic policy based access control model with adaptive trust mechanism.
- To provide an adaptive framework that supports re-organization of group system model.
Global meta polices for multilevel access control

Trust Engine
- Weighted Context Specific Parameter
- Peer Credibility
- Self Trust
- Feedback by Direct interaction
- Indirect Recommendation
- Incentives for rating

Trust Computation

Context based policy
- E-commerce Domain
- Open Source Software Development
- Online gaming
- Policy 1
- Policy 2
- Policy 3

Evolution of P2P Groups
- Dynamic Policies
  - Adaptive Trust
  - Malicious Peers

Peersim Simulator
Global policy

- Peer is allowed to join at level 1 of group if
  - 50% of existing peers agree and
  - his current rating is greater than some threshold $x$.

- For a new level to be introduced in group
  - at least $y$ no. of peers with rating greater than $x$ should collaborate.

- To update to a higher level
  - peer should be authenticated member of lower level and should have trust value $> x$
Domain specific policies

- A sample policy for FOSS
  - Developer d in member role can be elevated to reviewer r if no. of codes submitted is > t, and % code accepted is > a
  - Reviewer r may be updated to role of moderator if he has completed x reviews successfully.
Thesis contributions

- A integrated framework for collaborative groups which has a novel policy based method for membership control, access control and trust management.

- Main features are
  - Dynamic policy driven trust based access control model.
    - Dynamic Policies are written using Prolog, so that they can be easily modified without changing the code.
  - Dynamic adaptation of policies and re-organization of group structure by switching between system models.
  - Designed a tool capable of achieving this dynamic policy adaptation and optimization
Thesis contributions......

- A statechart model to capture the dynamic adaptation of varying system models and policies.
- An adaptive multi-dimensional trust model for P2P applications.
- We are able to evaluate the effect of different dynamic policies on groups and assist the users to adapt to the right set of policies for evolution of the group.
- We have also designed a GUI based converter for converting text based input to Prolog format to assist the user for writing the policies.
Summary of contributions

Motivation and Context
- Collaborative peer groups - decentralized, dynamic, self-organized.
- Need for admission control, access control, trust.
- Dynamic Policy based and reputation based access control
- Adaptive and evolving trust for self-organising groups
- Tool to be able to select optimum policy for a specific application

Contributions
- Framework and protocols for secure communication in collaborative peer groups
- Dynamic Policy based model for multi-level access control integrated with trust.
- Optimization of group performance by switching between different system models and state chart representation of the same.
- Adaptive trust mechanism capable of handling malicious peers in self-organising groups.

Publications
- Integrated Framework for Authentication and Access control in peer to peer groups
  - WISA 2007 (Korea)
- Dynamic Policy based model for trust based access control in peer to peer applications
  - ICC (IEEE)2009 (Germany)
- Dynamic Policy Adaptation for Collaborative Groups.
  - CNSA 2010 (Chennai)
  - Published in Springer (Recent Trends in Network Security)
Group Activities and Behavior

- Membership control
  - Join
  - Role_Set
  - P2P Group
  - Update
  - Rating
  - Leave

- Task Scheduling
  - Concurrent
  - Access control

- Peer Behavior
  - Existing peer

- Group Metrics
  - New Peer

- Rating
  - Existing peer
**F/OSS Model**

\[
\text{Role\_Set} = \{\text{GR U DR}\}
\]

\[
\text{GR} = \{\text{MP, AP, CP}\}
\]

\[
\text{DR} = \{\text{Developer, Reviewer, Moderator}\}
\]

\[
\text{F/OSS group}
\]

**Tasks**
- development of software modules of different difficulty levels
- content distribution
- resource sharing
- publish/subscribe postings news-group.

\[
\text{Task-Set} = \{\text{GTi, ATi}\}
\]

**Events**
- Join,
- Leave,
- Update,
- Rate,
- task scheduling,

\[
\text{Task\_attrib} = \{\text{DiffLevel, Time0fEntry, ExpectedTi}-
\text{meOfCompletion, Cost, ErrorCount, SecurityClassification, TaskStatus}\}
\]
F/OSS Model

\[
\text{Role_SET} = \{GR \cup DR\}
\]
\[
\text{GR} = \{MP, AP, CP\}
\]
\[
\text{DR} = \{Developer, Reviewer, Moderator\}
\]

\[
\text{F/OSS group}
\]

\[
\text{Events} - \text{Join, Leave, Update, Rate, task scheduling,}
\]

\[
\text{Task_SET} = \{GTi, ATi\}
\]

\[
\text{Tasks}
\]
- development of software modules of different difficulty levels
- content distribution
- resource sharing
- publish/subscribe
- postings
- news-group.

\[
\text{Task_attrib} = \{\text{DiffLevel, Time0fEntry, ExpectedTimeOfCompletion, Cost, ErrorCount, SecurityClassification, TaskStatus}\}
\]
FOSS policy changes

- More stringent join policy as RateOfJoin and #MPs increases.
- More stringent update policy if RateOfJoin and #CPs increases
- Change in job allocation policy if RateOfCompletion or RateOfAcceptance decreases.
FOSS system model

- 3 domain roles say developer, reviewer moderator
- When code is submitted by developer it is forwarded to single reviewer
- Moderator accepts code if rating is say 7/10
System model change in FOSS

- 4 domain roles instead of 3
  - add an additional reviewer role and submit intermediate code to Reviewer 1.
- Different workflow
  - submit code to 3 reviewers in parallel instead of a single reviewer
  - allocate same task to 3 coders instead of one.
- Different trust model
  - adapt a new trust model to calculate ratings of peers.
Architectural Model
Statechart Model

- Statecharts used to reflect the behavior of the system and specify the control flow between the activities.
- Transitions between states are triggered by events (E) and guarded by conditions (C).
- Events can be external
  - (generated by elements outside the statechart)
- or internal
  - (generated by elements inside the statechart).

Transitions are driven by E[C]A triple.
State Chart Model…..

- **Components are:**
- **States, Events, Transitions, Conditions/Guards, Actions, Expressions, Variables.**

- **State is an observable condition that the system can be in**
  - Action Set
  - Policy Set
  - Global State Variables—modified on occurrence of events and actions
Group Health measures the Group Performance Index for a particular state of the system.

\[ GHP = \{\text{AvgTrustRateOfPeers, AvgSkillSetOfPeers, RateOfCompletion, RateOfUpdate, RateOfJoin, RateOfLeave, RateOfAllocation, SizeOfGrp}\} \]

These parameters cannot be known apriori and have to be periodically monitored by the group peers.

\[ SM = \{\{\text{Peers}\}, \{\text{Events}\}, \{\text{Tasks}\}, \{\text{Policies}\}, \{\text{Roles}\}, \{\text{GHP}\}\} \]
Group Tasks

- Information/Resource Request (T1)
- Information/Resource Response (T2)
- Feedback/Rating Request (T3)
- Feedback/Rating Response (T4)
- Voting Request (T5)
- Voting Response (T6)
- Record Maintenance (certificates/keys/ratings) (T7)
- Monitoring Group Performance Index GPI (T8)
- Providing incentives for correct ratings (T9)
- Punish actions for incorrect ratings (T10)
Guards and conditions

- $c_1 = \text{AvgTrustRateOfPeers}$
- $c_2 = \text{AvgSkillSetOfPeers}$
- $c_3 = \text{RateOfCompletion}$
- $c_4 = \text{RateOfUpdate}$
- $c_5 = \text{RateOfJoin}$
- $c_6 = \text{RateOfLeave}$
- $c_7 = \text{RateOfAllocation}$
- $c_8 = \text{SizeOfGrp}$
- $c_9 = \text{RateOfAccept}$
State Variables

- \( S_v = \{\# \text{Roles}, \# \text{Levels}, \# \text{GroupSize}, \# \text{MPs}, \# \text{APs}, \# \text{CPs} \} \)

- **Specific combination of event and guard trigger a change in state variables which in turn triggers an action which could be policy change or system model change**
Conditions that Trigger Policy change

- Combination of group metrics /predicates
- $c_1 \land c_2$
- Case
  - If e and
    
    if $c_1 < t_1 \land c_5 > t_2$, then $P_1 \rightarrow P_2$
StateChart Representation
StateChart Representation for FOSS

S1 – SM1
S2 – SM2
S3 – SM3

S21 → S22 (change in join/update policy)

c5 >> t1 && c4 << t2, && c8 > 100 && # MPS > 50

c1 = AvgTrustRateOfPeers
c8 = SizeOfGroup
c3 = RateOfCompletion
c5 = RateOfJoin
c4 = RateOfAccept
**StateChart Representation for FOSS**

S1 → SM1
S2 → SM2
S3 → SM3

S21 → S23 (change in job_alloc policy) (c3 ∧ c9 << t2)

S2 → S3 (change in system model)
(c1 ∧ c3) << t1, GH << t

c1 = AvgTrustRateOfPeers
c2 = AvgSkillSetOfPeers
c3 = RateOfCompletion
c4 = RateOfUpdate
c9 = RateOfAcceptance
System Model Change

- \( SM_i = \{P_i, GR_i, DR_i, Pol_i, GHP_i\} \)
- \( SM_j = \{P_j, GR_j, DR_j, Pol_j, GHP_j\} \)

where

- \( P_i \subseteq P_j \), \( DR_i \subseteq DR_j \) or \( DR_i \supseteq DR_j \),
- \( GR_i \subseteq GR_j \) or \( GR_i \supseteq GR_j \)
- \( Pol_i \subseteq Pol_j \)

When the system is in any sub-state of state \( SM_i \), all the State Variables are modified except \#Roles and \#Levels.
Formal System Model

\[ \text{SM} = (S, T^0, T, T', E, C, A, i, f, \psi) \text{ where} \]

- S is a finite set of States
- E is a finite set of events
- C is a finite set of conditions or guards
- A is a finite set of actions
- \( T \subseteq S \times E \times C \times 2^A \times S \) is a finite set of transitions where \( 2^A \) denotes the power set of A.
- \( T': S_i \rightarrow S_j \) is a set of transitions from one OR-state to another OR-state
- $T^0$ is a transition to the same state.
- $i \in S$ is an initial state.
- $f \in S$ is a final state.
- $\psi : S \rightarrow \text{BASIC, AND, OR}$ is a function that defines state decomposition.
DPDTBAC

- Global Policy
- Domain Specific Policy
- Declarative Language Prolog to express the policies
- A prolog rule is an expression of the form
  - Ro(uo):-R1(u1).......Rn(un)
  - Assert and Retract clauses of Prolog help to achieve dynamism
Prolog facts

- Events, group roles and domain roles in the system are defined as prolog **facts**.
  - grouprole(member).
  - grouprole(admission).
  - grouprole(control).
  - level1(developer)
  - level2(reviewer).
  - level3(moderator).
Facts and Rules

- **Facts**
  - req(join).
  - req(update).
  - req(leave).

- **Join Rules:**
  - `verify(UUID, Request, GR, Level, SPR) :- join_req(Request), grouprole(GR), GR=member, level1(Level), SPR > 20, SPR < 40, assert(assignRole(UUID, GR)), assert(assignLevel(UUID, Level)).`
Join and update rules

- \texttt{verify(UUID,Request,GR,Level,SPR)} :- \texttt{join\_req(Request)}, \texttt{grouprole(GR)}, \texttt{GR=control}, \texttt{level1(Level)}, \texttt{SPR >60}, \texttt{assert(assignRole(UUID,GR))}, \texttt{assert(assignLevel(UUID,Level))}.

- \texttt{Update\_role(UUID,Request,GR,Level,Rating)}:- \texttt{assignRole(UUID,member)}, \texttt{update\_req(Request)}, \texttt{admission(GR)}, \texttt{level1(level)}, \texttt{Rating > 5}, \texttt{retract(assignRole(UUID,member))}, \texttt{assert(assignRole(UUID,GR))}.
Domain Specific Rule for FOSS

- \texttt{Update\_level(UUID,Request,CodesSubmitted,Percent Accept,Rating)} :-
  \texttt{assignLevel(UUID,developer),update\_req(update),}
  \texttt{CodesSubmitted > 20, PercentAccept > 50, Rating > 6,}
  \texttt{retract(assignLevel(UUID,developer)),}
  \texttt{assert(assignLevel(UUID,reviewer)).}
Dynamic Rules for Join

- Update_engine(Tot, MPs, APS, CPs):- MPs > 50, Tot ≥ 200, retract(old_join_rule), assert(new_join_rule).

- Update_engine(Tot, MPs, APS, CPs):- MPs > 50, RateOfJoin < t, retract(old_join_rule), assert(new_join_rule).
Dynamic Rules

- \textit{Update\_engine}(\textit{Tot},\textit{MPs},\textit{APS},\textit{CPs}) :- \textit{MPs} > 50, \textit{Tot} > 200, \textbf{retract}((\textit{verify}(\textit{UUID},\textit{Request},\textit{GR},\textit{Level},\textit{SPR}) :-\textit{join\_req}(\textit{Request}), \textit{grouprole}(\textit{GR}), \textit{GR} = \text{member}, \textit{level1}(\textit{Level}), \textit{SPR} > 20, \textit{SPR} < 40, \textbf{assert}(\text{assignRole}(\textit{UUID},\textit{GR})), \textbf{assert}(\text{assignLevel}(\textit{UUID},\textit{Level}))),\textbf{assert}((\textit{verify}(\textit{UUID},\textit{Request},\textit{GR},\textit{Level},\textit{SPR}) :-\textit{join\_req}(\textit{Request}), \textit{grouprole}(\textit{GR}), \textit{GR} = \text{member}, \textit{level1}(\textit{Level}), \textit{SPR} > 40, \textit{SPR} < 50, \textbf{assert}(\text{assignRole}(\textit{UUID},\textit{GR})), \textbf{assert}(\text{assignLevel}(\textit{UUID},\textit{Level})))).
Dynamic Rule to Relax Join

- \text{update engine}(\text{Levels}, \text{MPs}, \text{APs}, \text{CPs}, \text{Total}, \text{RateOfJoin}) :- \text{Total} > 100, \text{RateOfJoin} < t1,
- \text{retract}(\text{verify}(\text{Request}, \text{RI}, \text{Level}, \text{Rate}, \text{SPR}, \text{Expert}, \text{Tot}) :- \text{join}(\text{Request}), \text{admission}(\text{RI}), \text{Level} =: 1, \text{SPR} > 40, \text{Expert} > 20,
- \text{assert}(\text{verify}(\text{Request}, \text{RI}, \text{Level}, \text{Rate}, \text{SPR}, \text{Expert}, \text{Tot} :- \text{join}(\text{Request}), \text{admission}(\text{RI}), \text{Level} =: 1, \text{SPR}>20, \text{Tot}<200).
Dynamic Rule for System Model change

- \textit{Introduce\_level}(\text{Tot}, \text{CPs}, \text{CP\ vote}, \text{RateAccept}) :-

  \text{RateAccept} < 30, \text{Tot} > 400, \text{CPs} > 30, \text{CP\ vote} > 90,

  \text{retract}(\text{level2}(\text{reviewer})), \text{retract}(\text{level3}(\text{moderator})), \text{assert}(\text{level2}(\text{reviewer1})), \text{assert}(\text{level3}(\text{reviewer2})), \text{assert}(\text{level4}(\text{moderator})).
Task allocation policies

- 3 job allocation policies defined
- In Job_alloc 1 if SPR < 40 then easy job allocated, 40 < SPR < 60 medium job allocated and SPR > 60 diff job allocated
  - Peer not kept idle, so if easy job not avail then next level complex job allocated.
- In Job_alloc 2 if peer with SPR < 40 enters and no easy job avail then peer is kept idle, i.e strict job allocation done.
- In Job_alloc 3, peer with less SPR is not allocated a diff job, but peer with high SPR maybe allocated easy job
Simulation Results and analysis

- Switching between two system models.
  - We took runs for 2 different system models with single level of reviewing and double level of reviewing respectively.
  - Tested the performance in presence of varying job compositions for each policy.
  - For 100 projects, model with single review took more time as compared to double review model.
  - For 200 projects, with policy 2 single review level took longer time, but with policy 1, single review model converged faster.
  - Dynamically switching between the two models gave better performance.
Model with single review level takes more time to converge for both easy and difficult jobs.
Single review model takes more time but has lesser cost.
Dynamic switching between two system models
Dynamic switching between two system models
Single and double review model

Varying runs for difficult and easy jobs (200 projects). Double review converges faster in presence of all easy jobs.
Cost metric vs time metric for 200 projects with policy2

Single review takes more time to converge
Cost metric vs time metric for 200 projects with policy1

Double review takes more time to converge with policy 1 and difficult jobs
Simulation Results and Analysis for dynamic policies

- Simulator settings
  - Defined a project of 1000 modules of diff job complexities
  - Join policy maintains group composition as 30:40:30
  - Gave runs for varying job compositions and varying job_alloc policies

- Observation:
  - When maximum difficult jobs entered the system policy 1 gave better results whereas when maximum easy jobs entered policy 3 gave better results
  - Dynamism in job allocation policy helped the group to perform better.
Varying policies for easy jobs

Job_alloc 3 works better when maximum easy jobs enter the system
Varying policies for difficult jobs

Job_alloc_1 works better when maximum diff jobs enter the system

![Graph showing comparison of time taken and percentage completed for different job allocation policies](image-url)
Static vs dynamic policy

Dynamic policy helps group to converge faster.
Features of our Adaptive trust model

- Weighted sum of direct trust, indirect trust, credibility, transaction context and incentives for ratings
- Differentiate between peers who give poor service (incapable peers) and malicious peers giving varying levels of service
  - By measuring variation of service among random windows in the history of a peer.
Malicious behavior

- Maliciousness in providing service
  - Measured by random window method
- Maliciousness in providing recommendations
  - Similarity algorithm for measuring credibility of a peer
Detecting and penalizing malicious behaviors

- The ratings of the transactions are weighed by the value of the transaction
- The variance in the level of service is detected and penalized by the Variable window method.
- Cartelization of users is discouraged by the use of the similarity method.
- The effect of giving false ratings to peers results in decrease of indirect trust value.
Variable window method

- Diff between maximum and minimum rating acquired by a peer for service given is monitored within a window of length 1.
- If rating diff is > than a certain threshold, a violation is added to peer’s record and direct trust is reduced.
- Maximum variation is measured in various random window sizes.
AdaptiveTrust Model

- **Feedback**
  - $f_E(x, y)$ is the feedback given by peer $x$ for peer $y$ for a transaction.
  - $ai = a1, a2, a3, ......an$ is the set of attributes used for computing the feedback.

\[
 f_E(x, y) = \sum_{i=1}^{n} f_{ai} \ast w_{ai} 
\]

is the feedback score given by peer $x$ about peer $y$ for attribute $ai$. 

---

28-07-2013  
Madhumita Chatterjee
Reputation

- Assume that peer x stores up to n feedback ratings of previous transactions with peer y.
- Then reputation of peer y is calculated by peer x as

\[ R(x, y) = \frac{\sum_{i=1}^{n} \alpha^i \cdot f_{Ei}(x, y)}{\sum_{i=1}^{n} \alpha^i} \]

\( f_{Ei}(x, y) \) denotes the ith feedback given by peer x to y and \( \alpha \in [0, 1] \) is a decay factor indicating importance of recent transactions.
Direct Trust

- derived from a peer’s all Direct transaction experience, their credibility and the number of times the peer has been interacted with them.

\[
T(D)(y) = \frac{\sum I(x, y) \cdot R(x, y) \cdot Cr(x)}{\sum I(x, y) \cdot Cr(x)}
\]
Indirect Trust

- Assume that each user x receives job ratings for peer y from n references $k = (1, 2, ... n)$

$$T_{ID}(y) = \frac{\sum_{k=1}^{n} I(x, y) \cdot R(k, y) \cdot Cr(k)}{\sum_{k=1}^{n} I(x, y) \cdot Cr(k)}$$

Motivation based Trust Factor

To add incentives for rating we add the factor

$$\frac{F_x}{I_{x, y}}$$
Feedback Similarity Method for Credibility

\[ Cr = 1 - \sqrt{\frac{\sum [T(D)(R, X_j) - T(D)(i, X_j)]^2}{IR(X)}} \]
Global Trust value

\[ T = \alpha \cdot T_D(x, y) + \beta \cdot T_{ID}(x, y) + \gamma \cdot Context\ factor \]

- \( \alpha \) is the weight associated with direct experience,
- \( \beta \) with indirect reputation
- \( \gamma \) is a fine tuning constant to control the amount of reputation gained by rating others.

Proposed Adaptive Model:

Weights can be dynamically adapted and tuned
Simulations

- Run 1
  - simulation was conducted with 1000 peers and 600 feedback raters and a collusive group among the feedback raters of size 100. The simulation was run for 1000000 transactions

- Run II
  - Simulation run for 1000 transactions with 20% malicious peers
Trust values of normal vs malicious peers

![Graph showing the comparison of average trust values between normal and malicious peers over iterations.](image)
Credibility of collusive vs normal peers
We performed a comparative analysis on a network of 100 nodes with 800 transactions in each simulation. The parameters used for comparison were:

- Average trust value of good peers
- Average trust value of malicious peers
- Transaction success ratio in presence of varying percentage of malicious peers
- No of times the malicious peers are selected for transacting with increasing percentage of malicious peers.
Aggregate trust values of peers with 20 percent malicious peers
Successful Transaction Ratio

![Graph showing Successful Transaction Ratio against Percent Malicious Peer]
Service Selected of Good Peer vs Malicious peer

No of times Service Selected vs No. of Transactions (10% Malicious Peers)
Transaction success ratio with malicious peers as service providers
Conclusion

- Novel policy based method for membership control, access control and trust management of collaborative groups.
- We are able to express dynamic policies and varying system models in our framework.
- It supports dynamic adaptation of policies and dynamic re-organization of group structure by switching between system models.
- We give a statechart formalism for the same.
Dynamic policies are written using a declarative language so they can be easily modified without changing the code.

Built a tool and did experiments to validate our framework.

Tool assists users to evaluate the effect of different policies on groups and adapt to the right set of policies for evolution.

We give an adaptive multi-dimensional trust model for P2P applications.

Experiments show that the model is able to detect malicious peers and mitigate their effect on group performance.
Future Scope

- Use of collaborative reinforcement learning to optimise group behavior
- System should be able to suggest new policies for optimizing behavior
- Trust model can be made more adaptive by dynamically assigning weights to each trust parameter.
- Improved model for malicious behavior.
Response to reviews from examiner 2.

- Clearly show interplay between declarative language prolog and reactive model statecharts.

- (Addressed in section 5.4.1, 5.4.2 and 5.5.1, pages 82-84, in thesis)
Interplay between Prolog and Statecharts

System model change (using assert-retract)

- Yes: Continue in Same state (GH > t1)
  - Yes: GH << t1
  - No: GH > t1

- No: GH <= t1
  - Yes: State maintained
  - No: Update state

Prolog engine policy enforcement

Discrete event simulator

E/[C]/A
Statecharts

EVT queue

Local policy change (assert-retract)

Update state

(Update state, update Variables)

(Check, guards, conditions)

External events
(transitions, actions)

Internal events

Update state

28-07-2013
Madhumita Chatterjee
Provide a comparative evaluation (qualitative comparison) of Prolog with other Policy specification frameworks

(Addressed in Chapter 6, pp 105-110)
<table>
<thead>
<tr>
<th>Policy Framework Features</th>
<th>IETF</th>
<th>Ponder</th>
<th>KAoS</th>
<th>Rei</th>
<th>Our framework</th>
</tr>
</thead>
<tbody>
<tr>
<td>Policy specification Language</td>
<td>Specific policy language not defined</td>
<td>Ponder-Declarative specification</td>
<td>DAML/OWL-semantic language</td>
<td>Prolog like syntax + RDFDeclarative plus semantic</td>
<td>Prolog (Declarative)</td>
</tr>
<tr>
<td>Tools for policy specification</td>
<td>nil</td>
<td>Graphical editor and compiler</td>
<td>KPAT graphical editor for ontology and policy management</td>
<td>Under development</td>
<td>Simple graphical editor and converter from text to Prolog</td>
</tr>
<tr>
<td>Reasoning support</td>
<td>---</td>
<td>Event calculus</td>
<td>Java Theorem prover</td>
<td>Prolog engine</td>
<td>Prolog engine</td>
</tr>
<tr>
<td>Meta policies</td>
<td>No direct support</td>
<td>Yes</td>
<td>Partial</td>
<td>Yes</td>
<td>yes</td>
</tr>
<tr>
<td>Policy Framework Features</td>
<td>IETF</td>
<td>Ponder</td>
<td>KAoS</td>
<td>Rei</td>
<td>Our framework</td>
</tr>
<tr>
<td>---------------------------</td>
<td>-------</td>
<td>--------</td>
<td>-------</td>
<td>-------</td>
<td>---------------</td>
</tr>
<tr>
<td>Dynamic policy update</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Lang formal semantics and extensibility</td>
<td>Nil</td>
<td>Partial</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Domains and other forms of grouping</td>
<td>Yes</td>
<td>Yes group and role constructs</td>
<td>Partial</td>
<td>Partial-no distinction betn grp and role</td>
<td>Yes</td>
</tr>
<tr>
<td>Express dynamic system model</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Ease of use</td>
<td>---</td>
<td>Lang specifically designed</td>
<td>Need specialized tools</td>
<td>Need specialized tools</td>
<td>Simple policy specification</td>
</tr>
</tbody>
</table>
Relative evaluation of trust metric with prevailing trust models.

(Addressed in section 7.6, pages 130-132 in thesis)
## Comparative evaluation of different trust metrics

<table>
<thead>
<tr>
<th>Metric Features</th>
<th>Average Trust Model</th>
<th>Peertrust Model</th>
<th>Our Trust Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sensitivity to new experiences</td>
<td>Less sensitive</td>
<td>More sensitive</td>
<td>More sensitive</td>
</tr>
<tr>
<td>Sensitivity does not depend on total no of experiences</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Incentives provided for giving accurate ratings</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Long term behavior taken into account</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Metric Features</td>
<td>Average Trust Model</td>
<td>Peertrust Model</td>
<td>Our Trust Model</td>
</tr>
<tr>
<td>--------------------------------------------------------------------------------</td>
<td>---------------------</td>
<td>-----------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>Collusive behavior Detected and penalized</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Detects and penalizes sudden errant behavior</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Detects and penalizes long term oscillatory behavior</td>
<td>No unable to detect</td>
<td>No unable to detect</td>
<td>Yes Partially</td>
</tr>
<tr>
<td>Credibility measure distinguishes between confidence placed for services and recommendations</td>
<td>No, only false ratings are detected by similarity algorithm</td>
<td>Yes, random window method</td>
<td></td>
</tr>
</tbody>
</table>
Clarifications required by examiner 1

Figure 3.4: Member level update
Message details

1. \( P_i \rightarrow P_j : \{\text{UpdateREQ}\}_{SK_i}, \text{Cert}_i \)

2. Peer j verifies authenticity and returns computed trust value.

3. Peer i requests for indirect trust values

4. CP or update manager gives global trust value to Policy manager.

5. Verifies against join policy and gives input to key management system for computation of group key.

6. Group key given to Peer i.
Publications....


- “Dynamic Policy Based Model for Trust Based Access Control in P2P Applications”

- “Dynamic Policy Adaptation for Collaborative Groups”
  CNSA 2010, Chennai, India, Published in Springer.
Acknowledgements

- Prof G. Sivakumar
- Prof Bernard Menezes
- RPC Committee members