



Security and Trust in Peer to Peer Systems:

Risks and Countermeasures

Ling Liu

College of Computing

Georgia Institute of Technology



Motivation

- ◆ Internet P2P Applications
 - Large & dynamic population of users
- ◆ Representative Applications
 - ✓ Search Service – Gnutella, KaZaa, Limewire
 - ✓ File Service – Cooperative file system, Farsite, OceanStore
 - ✓ Publish-Subscribe Service – Siena, Scribe, Gryphon
 - ✓ VoIP – Skype



Motivation

- ◆ Applications Communicating Over the Internet Need Security Precautions to
 - Protect against attacks
 - Guarantee Data Confidentiality, Integrity, Availability, and Accountability
- ◆ P2P Applications
 - Additional aspect: who can we trust



Security and Trust in P2P applications

- ◆ Network Layer (TCP/IP):
 - Packet level confidentiality, integrity and message origin.
- ◆ Overlay Network Layer (Topology and Lookup protocol):
 - Infrastructure attacks and Defenses
- ◆ Application Layer:
 - Trust and Reputation Management.



Agenda

- ◆ A Brief Survey of Common P2P Applications
- ◆ Security Risks in P2P systems and applications
 - Example defense ideas or mechanisms
 - Why Trust Matters
- ◆ Dependable Trust Model and Trust Management
 - Vulnerabilities and Some Defense Mechanisms



P2P Basics

- ◆ All nodes serve as client, server, and router
- ◆ Network classification based on network connectivity
 - Exponential Networks:
Homogenous network, [average] node connectivity is equally distributed
 - Scale-free networks:
Follows power-law for connectivity, that is there are some highly connected nodes and many not highly connected nodes
- ◆ Current unstructured P2P systems are scale-free networks



Overlay Network Service Model

- **Application specific network topology**
 - Search Service: power-law topology
 - File Service: distributed hash tables based topology
 - Publish-Subscribe Service: tree-like topology
 - VoIP: geographical proximity based topology
- **Application specific interfaces**
 - Search Service: keyword based search
 - File Service: read, write, open and create
 - Publish-Subscribe Service: publish, subscribe, advertise, unsubscribe, unadvertise and route
 - VoIP: session init and session end



Security Issues

- ◆ **Denial of Service (DoS) Attacks**
 - TCP/IP layer: SYN flood, SYN+ACK flood
 - Overlay network layer: routing attack
 - Application layer: application-specific attacks
- ◆ **Authenticity Attacks**
 - Masquerading content provider
 - Incorrect or inaccurate application data / services
- ◆ **Confidentiality & Integrity**
 - Confidentiality & integrity from overlay network nodes
 - Access control on users
- ◆ **Key distribution & management**
 - Large and dynamic population of overlay nodes
 - Scalability with the number of overlay nodes



Denial of Service Attacks

◆ TCP/IP layer

- SYN flooding attack on TCP connection state
- SYN+ACK flooding attacks from large BotNets or zombie networks
- TCP connection reset attack
- Bandwidth exhaustion attack



Denial of Service Attacks

◆ Overlay Network Layer

- Routing attacks
 - Malicious node can drop/block messages routed through it
 - Malicious/sub-optimal routing decisions: increase the latency of routing algorithm
- Identity attacks
 - Spoof multiple identities: breaks threshold based fault-tolerant algorithms
 - Sybil attack
- Targeted attacks
 - Launch an DoS attack on small set of nodes hosting the target online service or application data
 - Individual nodes may be weaker than well-maintained web servers



Example Targeted Attacks

1. Attack a set of chosen (important) files

- File replication does not solve this problem if the location of file replicas are known
- CFS, FARSITE and OceanStore fail here

How?

- Say, file name is f
- R (# of replicas of file f) $\ll B$ (# bad nodes)
- Locations of these replicas are public
 - $hash(f \parallel 0), hash(f \parallel 1), \dots, hash(f \parallel R - 1)$
- Easy to perform DoS, host compromise attacks
- File encryption does not help if the adversary knows replica locations !

2. Risks in DHT routing

- Routing algorithm exposes the location of files
- Chord, CAN and most other routing algorithms fail here



LocationGuard Design

[SrivastaLiu. Usenix Security05]

◆ Three components

- Location Keys
 - effectively hide a file's location ?
- Routing Guard
 - lookup the location of a file without revealing the file's identity
- Location Inference Guards
 - counter inference attacks



LocationGuard Design

◆ Technical Challenges

- How to effectively hide a file's location ?
 - How to choose a location key lk ?
 - How to choose $A: (f, lk) \rightarrow loc$?
- How to secure the location of a file during lookup/routing ?
 - How to perform access control?
 - How to lookup the location of a file without revealing the file's identity?
- How to counter inference attacks ?
 - What is the hardness of breaking Location Keys?



LocationGuard: Summary

- ◆ LocationGuard: hiding objects on the overlay network
 - Location keys
 - Secure routing algorithm
 - Extensible set of inference guards
- ◆ Protects files against DoS, DDoS, and host compromise attacks
- ◆ Small performance (0.5%) and storage overhead (16 Bytes per file)



Other Security issues in P2P networks

- ◆ Inauthentic content
 - A malicious peer answering queries with fake files
- ◆ Peers not following the routing protocol
- ◆ Vulnerabilities due to Anonymity
 - malicious peers can answer to virtually any query providing tampered-with information
 - Denial of service attacks: spam, spoofing
 - Fake or dishonest feedback ratings (votes), fake voters etc



Popular Methods to P2P Security

- ◆ Countering Inauthentic Content
 - Content Encryption
- ◆ Countering the problem that peers do not following the routing protocol
 - Multiple alternate routes combined with routing correctness validation at protocol level
- ◆ Countering possible misuses and abuses due to peer anonymity
 - **Reputation-based trust** through peer review process
 - Peers' opinions (feedbacks) are used to establish a reputation for peers in the network
 - Method: Give each peer a *trust value* based on its previous behavior



P2P Reputation-based Trust Systems

- ◆ Attempt to reduce and avoid risks due to interacting with unknown and potentially malicious peer problems
- ◆ Peers receive reputations based on number of completed transactions (downloads received)
 - successful downloads receive higher reputations
- ◆ Reputations are used to calculate a trust score
- ◆ Advantage:
 - Self-regulating mechanism for P2P content sharing
 - popular in service oriented computing and eCommerce with multi-clients, multi-servers, and mutual anonymity support



Reputation Trust Management

- ◆ Reputation based trust models
 - Higher the reputation, more trustworthy the node is
- ◆ Feedback based reputation
 - Feedbacks are peer reviews and expressed as numerical ratings
 - A node's reputation is computed based on feedbacks
- ◆ Use trust values to avoid malicious nodes
 - Choose trustworthy nodes to perform transactions



Common Vulnerabilities in Reputation Models (1)

- ◆ Malicious peers can
 - modify the votes in transit
 - selectively discard low rating votes
- ◆ Popular Solutions → encryption of votes
 - Malicious nodes cannot modify the votes in transit because encrypted
 - Nodes will not be able to selectively discard votes since the recipient is not known (anonymity) and the content is not visible (encrypted)



Common Vulnerabilities in Reputation Models (2)

- ◆ Malicious peers can boost their ratings:
 - Create or forge a set of peers with the purpose of sending positive feedback (rating) to enhance their reputations.
 - Create **fake transactions** (interactions)
 - Remove their bad ratings or manipulating the reputation management schemes (**strategic oscillation**)
- ◆ Semi-honest peers
 - honest in serving the content but may lie when rating other peers in order to improve their trust ratings (**dishonest feedbacks**)



Security issues with P2P reputation systems: State of Art

Problems

◆ Confidentiality and Integrity

- Most addressed inauthentic content and confidentiality and integrity of reputation ratings through content encryption and votes encryption

◆ Very few addresses issues:

- Strategic malicious nodes
 - Alter node behavior strategically and dynamically
- Dishonest feedbacks
 - Differentiate between honest feedbacks and feedbacks from credible peers
- Fake transactions
 - Feedbacks on transactions that *never* actually happened



TrustGuard: Three-Tiered Framework [Srivatsa, Li, Liu, WWW05]

◆ Upper tier: strategic malicious nodes

- Reputation history
- Sudden fluctuations

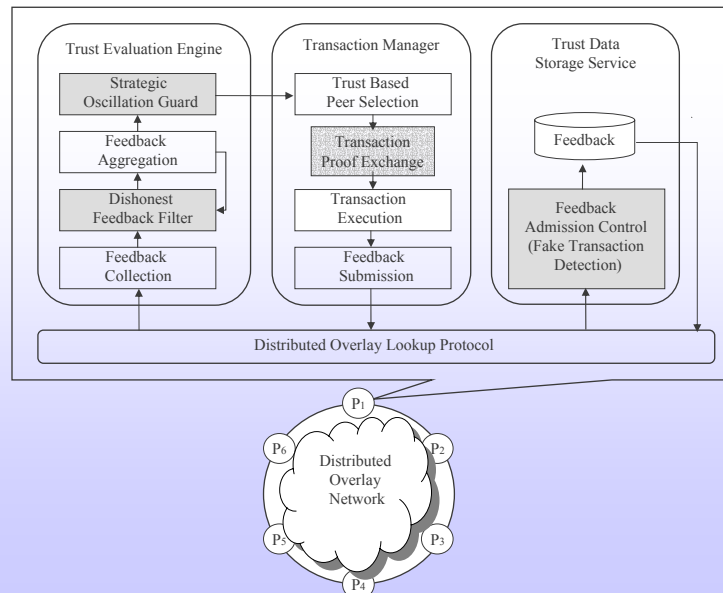
◆ Middle tier: dishonest feedbacks

- Weighted filtering of dishonest feedbacks

◆ Lower tier: fake transactions

- Light-weight fair-exchange protocol

TrustGuard Architecture



Guarding from Strategic Malicious Nodes

- ◆ Who are strategic malicious nodes?
- ◆ Game theoretic definition
 - Strategic nodes adapt its behavioral pattern to maximize their *malicious* goals
 - Good nodes are long standing and consistently behave well
- ◆ For example:
 - Misbehave only after earning high reputation
 - Alternate between good and bad behavior at regular or arbitrary frequencies



Strategic dynamic behavior

◆ Issues:

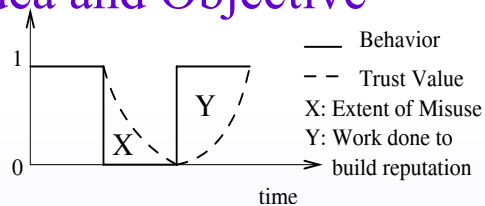
- Misbehave after earning high reputation
- Alternate between good and bad behavior at regular or arbitrary frequencies

◆ Desired properties:

- Reflect the dynamic behavior of peers quickly
- Hard to build, easy to drop – differentiate improvement and decrease of behavior
- Reflect consistent behavior of peers
- Tolerate occasional unintentional errors



Key Idea and Objective



- Fact: perform non-malicious behavior over an extended period of time => high reputation
- Observation: cost of increasing reputation should depend on the extent of past misbehavior

◆ Objective:

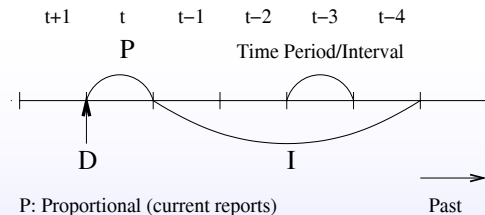
- For all $g \in G$, $TV_g(t) \approx 1$, For all $b \in B$, **Maximize** $cost(b)$

$$cost(b) = \lim_{t \rightarrow \infty} 1/t * \int_0^t BH_b(x) - TV_b(x) dx$$

Objective: for all $b \in B$, **Maximize** $cost(b)$



Strategic Oscillation Guard



P: Proportional (current reports)

I: Integral (history)

D: Derivative (fluctuations)

- ◆ $R_n(t)$: Reputation-based trust value of node n at time t computed using feedback ratings
- ◆ $TV_n(t) = \alpha * R_n(t) + \leftarrow \text{current}$
 $\beta * 1/t * \int_0^t R_n(x) dx + \leftarrow \text{history}$
 $\gamma * R'_n(t) \leftarrow \text{fluctuations}$
- ◆ Map PID to discrete domain



Computing $R_n[i]$

- ◆ Divide time into intervals of period T
- ◆ $TV_n[i]$: dependable trust value of node n in interval i
- ◆ $R_n[i]$: reputation of node n computed as an aggregation of feedbacks received in interval i
- ◆ Assume feedbacks are honest and transactions are not faked
- ◆ $R_n[i] = \text{Average over all feedback ratings}$



Incorporating History $H_n[i]$

- ◆ Assume trust value of node n is available for the last $maxH$ intervals
- ◆ $H_n[i] = \sum_{k=1}^{maxH} R_n[i-k] * w_k / \sum_{k=1}^{maxH} w_k$
- ◆ Optimistic weighting
 - $w_k = \rho^{k-1}$ (exponentially weighted sum)
- ◆ Pessimistic weighing
 - $w_k = 1/R_n[i-k]$ (inverse trust value weighted sum)



Incorporating Fluctuations

- ◆ $D_n[i] = R_n[i] - H_n[i]$
- ◆ $TV_n[i] =$
 $\alpha * R_n[i] + \beta * H_n[i] + \gamma(D_n[i]) * D_n[i]$
 - $\gamma(x) = \gamma_1$ if $x \geq 0$, γ_2 otherwise
 - Choose $\gamma_1 < \beta < \gamma_2$: increase derivative strength (with respect to history component) when node misbehaves and vice-versa
- ◆ $TV_n[i]$ can now handle **steady** and **sudden** behavioral changes
- ◆ Space & Time complexity = $O(maxH)$

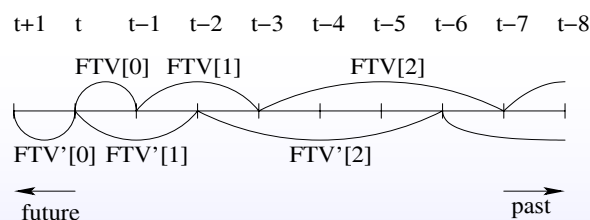


Optimization using Fading Memories

- ◆ History size = $maxH \Rightarrow$ wrong-doings will be **forgotten** in $maxH$ intervals
- ◆ Large history size not feasible
 - Too many trust values to store
 - Algorithm space & time complexity
- ◆ Fading memories: maintain **more detailed** information about **recent** trust values and only **fading memories** (less detailed) about **older** trust values



Implementing Fading Memories

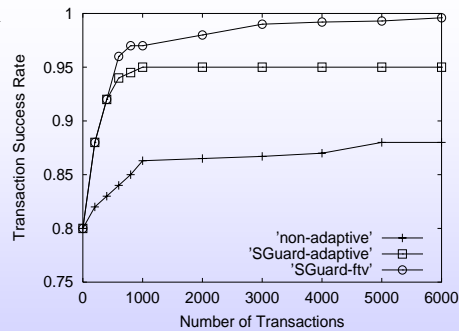


- ◆ Compressing 2^m intervals to m trust values
- ◆ Updating FTV : approximate trust value at interval $i-k$ by $FTV[\text{floor}(\log_2 k)]$
- ◆ $FTV^{i+1}[j] = (FTV^i[j] * (2^j - 1) + FTV^i[j-1]) / 2^j$
- ◆ Advantages: human experiences, extended bad behavior and space & time complexity



Evaluation: How much can Fading Memories help increase Transaction Success Rate?

- ◆ Transaction is successful if both the parties cooperate
- ◆ Non-adaptive pays for inability to adapt
- ◆ Fading memories has an edge over basic dependable trust model
 - it encodes exponentially larger information in a given space



Fraction of malicious nodes $p=20\%$



Dishonest Feedback Guard

- ◆ Algorithms to filter out dishonest feedbacks
- ◆ Filter: credibility factor

$$R_n = \sum_{u \in I(n)} F_n(u) * CR_n(u)$$

Where:

- $I(n)$: interactions performed by node n
- $F(u)$: feedback rating for interaction u
- $CR(u)$: credibility of $F(u)$
- R_n : trust value of node n
 - devoid of history (integral) and fluctuations (derivative) components



Trust Value based credibility Measure (TVM)

- ◆ $u.x$: node that provides feedback for interaction u
- ◆ $CR_n^{TVM}(u) = TV_{u.x} / \sum_{u \in I(n)} TV_{u.x}$
- ◆ Assumptions
 - Untrustworthy nodes are likely to submit false feedbacks
 - Trustworthy nodes are likely to be more honest
- ◆ Two Problems with the TVM approach
 - Problems with the second assumption
 - peers may be honest in serving content but lie about some other peers when providing feedbacks
 - Problems with large population of malicious nodes and collusions



Personalized Similarity based credibility Measure (PSM)

- ◆ Node n weights feedback given by node m based on its similarity with node m
 - through node n 's personalized experience

$$CR_n^{PSM}(u) = Sim(n, u.m) / \sum_{u \in I(n)} Sim(n, u.m)$$

where

$$Sim(n, m) = 1 - \sqrt{\frac{\sum_{r \in IJS(n, r)} (\sum_{v \in I(n, r)} F_n(v) / |I(n, r)| - \sum_{v \in I(m, r)} F_x(v) / |I(m, r)|)^2}{|IJS(n, m)|}}$$

- $IJS(n, m)$: common nodes with whom both node n and m have interacted
- Dissimilarity based on root mean square of differences in feedbacks over $IJS(n, x)$

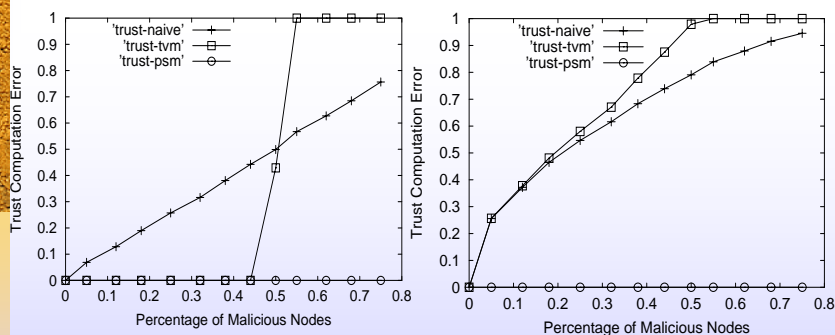


Dishonest Feedback Guard: Measuring Credibility Factor

- ◆ (1) Trust Value based Metric (TVM):
 - CR is proportional to the trust value of the node that provides the feedback
- ◆ (2) Personalized Similarity based Metric (PSM):
 - Node n weighs the feedbacks given by node m based on its *similarity* with node m .
- ◆ TVM is vulnerable to malicious collusion
 - Malicious collusion:
 - Boost the trust value of colluders
 - Bad mouth non-colluders
- ◆ PSM is personalized and thus resilient to collusions



Evaluation: Dishonest Feedback Guard



Trust computation error under non-collusive setting and collusive settings

- PSM is effective for even large malicious cliques
- TVM breaks down when $p > 50\%$ and in collusive setting



Guarding from Fake Transactions

- ◆ Need to bind transactions to feedbacks
- ◆ **Secure & Fairness** in transaction proofs
- ◆ Security is *easy*
- ◆ Proof of transaction
 - Signing transactions using node's public key
 - $\langle PK_n, RK_n \rangle$ public key pair of node n
 - $PI_n = RK_n(I)$, where $I = \langle Txn Descr \rangle \parallel \langle time stamp \rangle$ for some transaction u
- ◆ Use PI to claim to any third party that transaction u indeed happened



Fair Exchange of Secure Transaction Proofs

- ◆ Simply exchanging PI s is flawed
- ◆ Malicious node m does not provide PI_m to node n , but it accepts $PI_n \Rightarrow$ node m can claim it performed transaction u with node n
- ◆ Problem: **Exchange Atomicity**
- ◆ Solution: Fair exchange protocols
 - Example:
 - Fair contract signing protocol by Micali



Fair Exchange

- ◆ Fair-Exchange: *“Protocol that ensures that no player in an electronic commerce transaction can gain an advantage over the other player by misbehaving, misrepresenting or by prematurely aborting the protocol”*
- ◆ Trivial Solution: trusted third party
 - Not scalable
 - Single point of failure
 - Administrative overhead
 - Susceptible to DoS and host compromise attacks



XChange Protocol: High Level Properties

[Srivatsa, Li, Liu, IPTPS 2005]

- ◆ Fair-Exchange using untrusted servers
- ◆ Tolerates Byzantine failures of up to 1/3rd of the untrusted servers
- ◆ Satisfies effectiveness and timeliness properties
- ◆ Close to strong fairness property
 - When the Fair-Exchange fails, the honest party has a proof of malicious behavior of the fraudulent party
- ◆ XChange Protocol: distributed, decentralized, scalable, good load balancing, tolerance to Crash and Byzantine failures, and free of administrative costs



TrustGuard: Summary

- ◆ TrustGuard: Three-tiered framework building dependable reputation management system
 - Strategic oscillation guard
 - Dishonest feedback guard
 - Fake transaction guard
- ◆ Experimental results
 - Demonstrate the vulnerabilities in reputation management
 - Efficacy of TrustGuard in defending against them
- ◆ Componentized architecture, stack structured: suitable for replacing any component
 - Say, a different algorithm for the strategic oscillation guard



Summary: Possible Attack Targets

- ◆ Underlying protocol layers
- ◆ P2P routing mechanism
- ◆ Nodes themselves
- ◆ Trust system
- ◆ Applications
- ◆ Users



Attack Classification

◆ Infrastructure Attacks:

- Attacks aimed at disabling p2p system
e.g: eliminating nodes, attacks on routing protocols

◆ Application Targeted Attacks:

- Attacks aimed at making users abandon the system
e.g: bad content, asymmetric consumption

- ◆ Both attacks are equally effective because p2p is a “community based and social system”



Failure vs. Attack

◆ Failure:

- Random failure of nodes and/or infrastructure elements

◆ Attack:

- Systematic failure of nodes and/or infrastructure elements

- ◆ Most P2P systems give priority for failure-tolerance over attack-tolerance



Attacks & Defenses

◆ Infrastructure Attacks

- Attacks on nodes
- Attacks on routing mechanism
- Defense: topology guard, node guard, routing guard

◆ Targeted Attacks

- Storage & Retrieval Attacks
- Flooding
- Face/Off
- Defense: Location guard, Event Guard, Application Guard, etc.



Attacks & Defenses

◆ Attack detection & recovery involves...

- Identifying Invariants in the System
- Monitoring the Invariants
- Proactive combined with Reactive Defense Mechanisms

◆ Proactive Defense:

- continuously guarding the topology, routing, and application running on top of the overlay

◆ Reactive Defense:

- Detecting/Ascertain Attacks
- Triggering Recovery Procedure



Secure P2P Design Principles

- ◆ Define verifiable system invariants
- ◆ Verify system invariants during operation
- ◆ Allow the querier to observe lookup progress
- ◆ Assigns keys to nodes in a verifiable way
- ◆ Server selection in routing may be abused
- ◆ Cross-check routing tables using random queries
- ◆ Avoid single points of responsibilities



Acknowledgement:

Sponsor: NSF ITR, IBM, DARPA.

Graduate students involved in the P2P Security/Trust Research Projects: Mudhakar Srivatsa, Aameek Singh, James Caverlee, Li Xiong, Sunkeun Park.

Questions?

