TOWARDS PEER-TO-PEER-BASED CRYPTANALYSIS

Matthäus Wander, Arno Wacker, Torben Weis
Oct 14th, 2010

6th LCN Workshop on Security in Communications Networks
Outline

• Introduction, Requirements

• System Model, Example Application

• Self-organizing job management

• Result verification

• Summary and Outlook
Introduction

• **Goal**
  – Run computationally intensive cryptanalytic job

• **Stakeholder**
  – Users with interest in cryptanalysis
  – But without distributed computing infrastructure

• **Job distribution**
  – Invite friends
  – Share CPU cycles with foreigners
Requirements

R1) CPU cycle sharing
R2) Ad-hoc setup
R3) Open & decentralized (participant-driven)
R4) Correctness
R5) Offline support
R6) Scalability & Efficiency
# Distributed Computing Paradigms

<table>
<thead>
<tr>
<th></th>
<th>Sharing (R1)</th>
<th>Ad-hoc (R2)</th>
<th>Open (R3)</th>
<th>Correct (R4)</th>
<th>Offline (R5)</th>
<th>Scales (R6)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Client/Server Computing</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Cloud Computing</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Cluster Computing</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Grid Computing</td>
<td>X</td>
<td>(x)</td>
<td>(x)</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Peer-to-Peer Computing</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>
Related Work

- **CoDiP2P [Castella2008]**
  - Manager hierarchy with job master

- **Organic Grid [Chakravarti2006]**
  - Manager hierarchy with job master

- **Jalapeno [Therning2005]**
  - Manager/worker groups

- **JNGI [Verbeke2002]**
  - Monitor/dispatcher/worker groups

→ Require **trusted** job managers
Job Management

• Management work
  – Divide job into tasks
  – Allocate tasks
  – Monitor status, track progress
  – Collect, verify and merge results

• Who manages the job?
  – Job Submitter
  – Elected workers
  – All workers (self-organization)
Requirements

R1) Cycle-sharing
R2) Ad-hoc setup
R3) Open & decentralized (participant-driven)
R4) Correctness
R5) Offline support
R6) Scalability & efficiency

Submitter manages job
Requirements

R1) Cycle-sharing
R2) Ad-hoc setup
R3) Open & decentralized (participant-driven)
R4) Correctness
R5) Offline support
R6) Scalability & efficiency

Elected workers manage job
Requirements

R1) Cycle-sharing
R2) Ad-hoc setup
R3) Open & decentralized (participant-driven)
R4) Correctness
R5) Offline support
R6) Scalability & efficiency

All workers manage job
Idea

- Share management burden among peers
- Do not use designated manager role
- Peers write job status into a distributed storage
- Resulting challenges
  - Organize data structure for efficient access
  - Ensure correctness of computation
  - Ensure correctness of stored data
System Model and Assumptions

- Scalable **peer-to-peer** overlay (Chord, Pastry)
- Unique participant ID [Wacker2008b]
- Secure message transport [Wacker2008b]
- NAT traversal [Wacker2008a]
- Accounting of work performed [Garcia2005] [Turner2004]
- Distributed storage (Distributed Hashtable)
Example Application

- Brute-force attack on **symmetric-key cipher**
- Represents class of search problems
- Input: ciphertext, cipher being used
- Solution space: all possible keys
  - Decrypt ciphertext
  - Rate result with score function
- Divide solution space into **task** blocks
Self-organizing Job Management

- Structure task list as tree
- Each object stored on different peer
Self-organizing Job Management (2)

- Participating peers traverse tree to get task
- Construct tree on demand
  - Divide
  - Compute
  - Merge
Self-organizing Job Management (2)

- Participating peers traverse tree to get task
- Construct tree on demand
  - Divide
  - Compute
  - Merge
Self-organizing Job Management (2)

- Participating peers traverse tree to get task
- Construct tree on demand
  - Divide
  - Compute
  - Merge
Self-organizing Job Management (2)

- Participating peers traverse tree to get task
- Construct tree on demand
  - Divide
  - Compute
  - Merge

Diagram of a tree structure with nodes labeled 0-7, 0-3, 0-1, 0, and 1.
Self-organizing Job Management (2)

- Participating peers traverse tree to get task
- Construct tree on demand
  - Divide
  - Compute
  - Merge
Self-organizing Job Management (2)

- Participating peers traverse tree to get task
- Construct tree on demand
  - Divide
  - Compute
  - Merge
Self-organizing Job Management (2)

- Participating peers traverse tree to get task
- Construct tree on demand
  - Divide
  - Compute
  - Merge
Self-organizing Job Management (3)

• **Task allocation**
  – Peers mark tasks as allocated
  – But allocations are not exclusive locks
  – Other peers may ignore allocations

• **Tree traversal**
  – Avoid querying occupied objects
  – Avoid unnecessary large trees
  – Depth-first traversal with frequent left turns
Result Verification

• Cheat attempts for search problems
  – Peer claims to have found solution (false positive)
  – Peer claims subspace does not contain solution (false negative)

• Find opportunistic cheaters efficiently

Task block

Result
[] Hit: ____
[] Miss

Toplist
1st: _____
2nd: _____
3rd: _____

Slices
Result Verification (2)

- c=10 cheated slices
- c=100 cheated slices
- c=500 cheated slices
- c=900 cheated slices

Probability of detection vs. Tested slices, v
Removal of Partial Results

- Large tree requires bandwidth for maintenance
- Tree size would scale with number of peers
  – ... if unneeded subtrees were removed
- Peers not allowed to remove unneeded subtrees
  – Too risky to lose progress
  – Even with verification
- Job submitter removes junk from time to time
  – If she doesn’t: some maintenance overhead
  – Still allowed to go offline
Distributed Storage

• Special requirements not provided by plain DHT
• Adaptive replication
  – Ensure consistency
  – Scale with number of read operations [Knoll2008]
• Access model: read all, append all, modify own
• Prevent unauthorized modifications
• Soft state: remove data if not refreshed
Summary and Outlook

• Peer-to-peer computing for CPU sharing (R1)
• **Self-organizing** without infrastructure setup (R2)
• Without provider or administration (R3)
• Deals with opportunistic peers (R4 partly)
• Allows job submitter to go offline (R5)
• Considers scalability & efficiency so far (R6)
• Future work
  – Large-scale evaluation
  – More complex applications
References

• [Chakravarti2006]: Organic Grid

• [Castella2008]: CoDiP2P
References

- **[Therning2005]: Jalapeno**
  - Niklas Therning, Lars Bengtsson: Jalapeno: secentralized grid computing using peer-to-peer technology. 2nd conference on Computing frontiers, 2005

- **[Verbeke2002]: JNGI**
References

• [Knoll2008]: Replication (Repl1, Repl2)
  – Mirko Knoll, Haitham Abbadi, Torben Weis: Replication in Peer-to-Peer Systems. 3rd Int. Workshop on Self-Organizing Systems (IWSOS), 2008

• [Wacker2008b]: Authentication (Auth)
References

• [Garcia2005]: Currency

• [Turner2004]: Currency
References

• [Wacker2008a]: NAT traversal (NAT)
  – Arno Wacker, Gregor Schiele, Sebastian Holzapfel, and Torben Weis: (Demo) A NAT Traversal Mechanism for Peer-To-Peer Networks. 8th Int. Conference on Peer-to-Peer Computing (P2P), 2008

• [Knoll2009]: Bootstrapping (IRC)
  – Mirko Knoll, Matthias Helling, Arno Wacker, Sebastian Holzapfel and Torben Weis: Bootstrapping Peer-to-Peer Systems Using IRC. 5th Int. Workshop on Collaborative Peer-to-Peer Systems (COPS), 2009