Accountability in Cloud Computing and Distributed Computer Systems

Hongda Xiao
Department of Electrical Engineering
Yale University

Thesis Defense, Sept. 17th, 2014
Advisor: Prof. Joan Feigenbaum
Accountability in Computer Science

- Preventive methods are inadequate
  - More and more online activities
  - Inter-domain business transactions and information exchanges
- Accountability is not a unified research area yet
  - Different researchers use the term to mean different things
  - Lack of practical accountability mechanisms for real systems
Accountability in Cloud Computing and Distributed Systems

● Cloud computing: different communication pattern

● Distributed systems: different cooperation pattern
Overview

- Systematization of accountability in computer science
- Cloud user infrastructure attestation
- On virtual machine reallocation in cloud-scale data centers
- Structural cloud audits that protect private information
Rest of Talk

• Briefly present three pieces of work
  • Systematization of Accountability
  • Cloud User Infrastructure Attestation
  • Virtual Machine Reallocation

• Go into the details of one
  • Structural Cloud Audits that Protect Private Information
Overview

- Systematization of accountability in computer science [Xiao, Feigenbaum, Jaggard, Wright; 2012]
- Cloud user infrastructure attestation
- On virtual machine reallocation in cloud-scale data centers
- Structural cloud audits that protect private information
Systematization of Accountability (1)

- Well established policies that need to be enforced
- Detect and punish violations of policies

E.g. PeerReview
- A distributed system to enforce a set of system policies
- Each node needs to respond to a message and send valid messages to other nodes
- A tamper-evident log to record actions of nodes and identify violators
Systematization of Accountability (2)

- High-level Perspective on Appropriate Focus of Accountability:
  - Enable violations to be tied to punishment

- Aspects of Accountability:
  - Time/Goals
    - Prevention, Detection, Evidence, Judgment, Punishment
  - Information
    - Identity of Participants, Violation Disclosure, Violator Identification
  - Action
    - Centralized vs. Decentralized
    - Automatic vs. Mediated
Systematization of Accountability (3)

- Conclusions
  - "Accountability" is used to mean different things
  - Accountability does not preclude anonymity or privacy
  - Accountability need not be mediated by a central authority
Overview

- Systematization of accountability in computer science

- Cloud user infrastructure attestation
  [Xiao, Szefer, Feigenbaum; 2014]

- On virtual machine reallocation in cloud-scale data centers

- Structural cloud audits that protect private information
Cloud User Infrastructure Attestation

- Cloud users need to verify the properties of cloud resources
  - Server
  - Topology
- Cloud providers are unwilling to reveal cloud infrastructure
- Objective: Cloud providers attest to cloud users without revealing their private infrastructure
Cloud User Infrastructure

- VMs and the hardware that supports them
  - Server Architecture
  - Topology Infrastructure
    - Virtual Network
    - Physical Network
Solution Overview

- Server Architecture (Trusted Computing)
  - Property-based Attestation (PBA)
  - Trusted Platform Module (TPM)

- Topology Infrastructure
  - Collect topology information
    - Secure hardware: Network TPM
  - Attestation protocols
    - Properties instead of infrastructure
    - Preserves cloud provider's privacy
    - Verifiable computation
Attestation Results

- A Novel Network TPM can collect topology information and serve as a security base
- A cloud provider attests to a cloud user that the cloud user infrastructure satisfies the requirements of the cloud user
- The cloud provider's infrastructure configurations remain private
Overview

- Systematization of accountability in computer science
- Cloud user infrastructure attestation
- On virtual machine reallocation in cloud-scale data centers [Xiao, Szefer; 2014]
- Structural cloud audits that protect private information
On VM Reallocation in Cloud-scale Data Centers

- **Motivation**
  - Unexpected events in data centers
  - Existing work on VM migration
- **We focus on migration-target selection**
- **NP-hard optimization problem**
- **Two-layer, heuristic algorithm**
  - Efficient with small optimality loss
Overview

• Systematization of accountability in computer science

• Cloud user infrastructure attestation

• On virtual machine reallocation in cloud-scale data centers

• Structural cloud audits that protect private information
  [Xiao, Ford, Feigenbaum; 2013]
Cloud Structural Audits that Protect Private Information

Xiao, Ford, Feigenbaum, "Structural Cloud Audits that Protect Private Information," CCSW 2013

- Cloud-service providers use redundancy to achieve reliability
- Redundancy can fail because of Common Dependencies

We need a systematic way to discover and quantify vulnerabilities resulting from common dependencies
Motivation

- This is a real problem
  - E.g.: A lightning storm in northern Virginia took out both the main power supply and the backup generator that powered all of Amazon EC2's data centers in the region

- Objective
  - Audit the cloud infrastructure to assess the reliability risk that results from common dependencies
  - Protect the private information of the cloud infrastructure providers

- Accountability Mechanism
  - Cloud users hold the cloud providers accountable for the reliability that they promise
Five Main Technical Ingredients of Our Approach
1. Structural Reliability Auditing

- Zhai, Wolinsky, Xiao, Liu, Su, Ford built a Structural Reliability Auditor (SRA)
  - collect comprehensive information from infrastructure providers
  - construct a service-wide fault tree
  - identify critical components; estimate likelihood of service outage
- An internal, structural cloud-audit system
  - Obtain the infrastructure information directly from the cloud provider, instead of from external interfaces of third parties
  - Evaluate the reliability of the cloud infrastructure by identifying the common dependencies – different from cloud diagnosis
2. Fault-tree analysis (FTA)

- FTA is a well established, classical deductive-reasoning technique for failure analysis
  - Occurrence of top-level failure event is a boolean combination of occurrence of lower-level events
- Fault "Tree" is actually a Directed Acyclic Graph (DAG)
  - Node: gate or event
  - Link: dependency information
Fault-Tree Example
3. Failure Sets and Failure-Sampling Algorithm

- Failure Set (FS)
  - a set of components whose simultaneous failure results in a cloud-service outage
  - Minimal FS: contains no proper subset that is also an FS
- Minimal-FS Algorithm
  - Finds all minimal FSes; exponential time in worst case
- Failure-Sampling Algorithm
  - Randomly assigns fail or not fail to the leaf-level events of the Fault Tree and computes whether the top-level event fails
  - If the top-level event fails, the failed leaf-level events are a FS
Fault-Tree Analysis Example

Cloud Storage Service1

CP1

{ R1, R2 }
{ DC1, DC2 }
{ DC1, P2 }
{ DC2, P1 }
{ P1, P2 }

CP2

DC1
Power 1

Power 2

DC2

Router 1

Router 2

Minimal-ES algorithm

Cloud Storage Service1

CP1

{} 

CP2

DC1
Power 1

Power 2

DC2

Router 1

Router 2

Failure-Sampling algorithm

{}
4. Secure Multiparty Computation (SMPC)

Each $i$ learns $y$.

No $i$ can learn anything about $x_j$ (except what he can infer from $x_i$ and $y$).

Very general positive results. Not very efficient.

In our work, $x_i$ is the cloud infrastructure of cloud provider $i$. 

$$y = F(x_1, \ldots, x_n)$$
5. Subgraph Abstraction

• Abstract the dependency information as a directed graph on macro-components; this will be the actual inputs to the SMPC
  - Macro-component: an abstracted (virtual) node in the dependency graph that can be considered an atomic unit for the purpose of structural-reliability analysis

• Key step in reducing the size of the input to the SMPC

• The choice of abstraction policy is flexible as long as it satisfies the requirements of ★.
Subgraph Abstraction: Example (1)

- Dependency Graph of Simple Data Center
  - A Storage Service
  - Two Data Centers, one for service and the other for back-up
- Inside the red frame is data center 1, which satisfies the abstraction policy in [XFF, CCSW 2013]
Subgraph Abstraction: Example (2)
System-Design Overview

- **P-SRA Client**
  - Data Acquisition Unit (DAU)
  - Local Execution Unit (LEU)
  - Secret Sharing Unit (SSU)

- **P-SRA Host**
  - Represents Cloud Users and Reliability Auditors
  - Does SMPC coordination

---

P-SRA Clients are the software installed on the “cloud providers,” instead of cloud users.
Algorithm Overview

- Step 1: Privacy-preserving dependency acquisition
- Step 2: Subgraph abstraction to reduce problem size
- Step 3: SMPC protocol execution and local computation
- Step 4: Privacy-preserving output delivery
Step 1: Privacy-preserving dependency acquisition

- The DAU of each cloud-service provider collects information about the components and dependencies of this provider
  - network dependencies
  - hardware dependencies
  - software dependencies
  - failure-probability estimates for components
- Stores the information in a local database for use by P-SRA's other modules.
Step 2: Subgraph Abstraction

- SSU computes the macro-components that satisfy the abstraction policy
- Prepares the abstracted dependency graph to be input to the SMPC. (Secret sharing is one of the steps in this process.)
- Gives the internal structure of the macro-components to LEU for local analysis
Data Acquisition and Subgraph Abstraction

DAU
Obtain the dependencies

Power 1
Router 1
Gateway 1
Core 1
Core 2
Agg 1
Agg 2
ToR 1
S 1
S 2
S 3
S 4
Storage

LEU
Give to LEU for local execution

Router 2
Gateway 2
Core 3
Core 4
Agg 3
Agg 4
ToR 1
S 5
S 6
S 7
S 8
Back-up

SSU
Prepare inputs to SMPC

Cloud Service 1
Data Center 1
Data Center 2

Power 1
Router 1
Power 2
Router 2

Prepare inputs to SMPC
Give to LEU for local execution
Obtain the dependencies
Step 3: SMPC and Local Computation

**SMPC**
- SSUs of P-SRA Clients send secret shares and scripts to P-SRA host
- Perform SMPC to identify common dependency and perform fault-tree analysis across cloud providers

**Local Computation**
- SSU passes the dependency information within macro-components to LEU
- LEU locally computes fault trees of macro-components
SMPC: Identify Common Dependencies

- SSUs and P-SRA Host cooperate to identify common dependency
  - Multiple (privacy-preserving) set intersections, followed by one (privacy-preserving) union

- Strict security requires doing so without conditional statements
  - Translate conditional statements into arithmetic computation

\[ \text{If (element } a \text{ in Set } B) \]
\[ \text{Int } a\text{\_in}\_B = \sum_{b \in B} (a = b) \]
SMPC: Fault-Tree Analysis

- Fault-tree construction
  - Represent the fault tree as "topology paths form with types" data structure
  - Eliminates use of conditionals
  - Cost: may be exponentially larger than DAG in worst case :(  
- Calculate failure sets from the topology paths form with types
  - Minimal FSes Algorithm
  - Failure-Sampling Algorithm
Step 4: Privacy-preserving Output Delivery

- Output for Cloud-Service Providers
  - Common dependency
  - Partial failure sets
- Output for Cloud-Service Users
  - Common-dependency ratio
  - Overall failure probabilities of cloud services
  - Top-ranked failure sets (a little information leakage)
Partial Failure Set

• Tell cloud provider S all the components in a (minimal) FS that belong to S

• Informs cloud provider S where to increase redundancy to avoid an outage regardless of what happens outside of S
Common-Dependency Ratio

- Common-Dependency Ratio of cloud provider S is defined as the fraction of components in S that are shared with at least one other cloud provider.
- The larger the ratio, the higher the risk of failure.
Top-Ranked Failure Sets

- Rank the (minimal) FSes based on user-defined rules, e.g.:
  - Failure probability
  - Size
- Help focus attention on most likely source of failure
- May release some private information, but this may be tolerable in some markets
Implementation

- Sharemind SecreC
  - C-like SMPC programming language
  - Specialized assembly to execute the code
Simulation: SMPC

- Practical as an offline service

- Used a low-end laptop – performance would improve on a workstation
Simulation: Local Execution

- Practical as an offline service
- Local Execution is not the bottle-neck.

Table 2: Performance of the LEU of a P-SRA client

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Case 1</th>
<th>Case 2</th>
<th>Case 3</th>
<th>Case 4</th>
<th>Case 5</th>
</tr>
</thead>
<tbody>
<tr>
<td># of switch ports</td>
<td>4</td>
<td>8</td>
<td>16</td>
<td>24</td>
<td>48</td>
</tr>
<tr>
<td># of core routers</td>
<td>4</td>
<td>16</td>
<td>64</td>
<td>144</td>
<td>576</td>
</tr>
<tr>
<td># of agg switches</td>
<td>8</td>
<td>32</td>
<td>128</td>
<td>288</td>
<td>1152</td>
</tr>
<tr>
<td># of ToR switches</td>
<td>8</td>
<td>32</td>
<td>128</td>
<td>288</td>
<td>1152</td>
</tr>
<tr>
<td># of servers</td>
<td>16</td>
<td>128</td>
<td>1024</td>
<td>3456</td>
<td>13824</td>
</tr>
<tr>
<td>Total # of components</td>
<td>40</td>
<td>216</td>
<td>1360</td>
<td>4200</td>
<td>16752</td>
</tr>
<tr>
<td>Running time (minutes)</td>
<td>&lt; 0.7</td>
<td>&lt; 0.7</td>
<td>&lt; 0.7</td>
<td>&lt; 0.7</td>
<td>&lt; 0.7</td>
</tr>
<tr>
<td>FS round 10^3</td>
<td>0.7</td>
<td>0.7</td>
<td>1.7</td>
<td>2.3</td>
<td>6.9</td>
</tr>
<tr>
<td>FS round 10^4</td>
<td>0.8</td>
<td>0.9</td>
<td>5.3</td>
<td>28.1</td>
<td>6.9</td>
</tr>
<tr>
<td>FS round 10^5</td>
<td>1.7</td>
<td>4.5</td>
<td>65.0</td>
<td>243.5</td>
<td>462.9</td>
</tr>
<tr>
<td>FS round 10^6</td>
<td>28.3</td>
<td>56.6</td>
<td>512.1</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>FS round 10^7</td>
<td>0.8</td>
<td>14.8</td>
<td>309.7</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>Minimal FS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Summary

- Systematization of accountability in computer science
- Accountability mechanisms for cloud-computing and distributed systems
  - Designed a framework to enable a cloud-service provider to attest to the properties of a cloud user infrastructure. Proposed a novel secure-hardware component, the "Network TPM," and a new attestation protocol
  - Formulated the VM Reallocation Problem, which is NP-hard, and provided an efficient, "two-layer" heuristic solution
  - Designed P-SRA, a private, structural-reliability auditor for cloud services based on SMPC. Prototyped it using the Sharemind SecreC platform
Future Work

- Cloud User Infrastructure Attestation
  - Build a Network TPM
  - More carefully evaluate the memory requirements of the attestation protocol
- VM reallocation
  - Integrate the algorithms into a standard cloud-management framework, such as OpenStack
- P-SRA
  - Measure the cost of audits and seek more efficient algorithms
  - Generalize the notion of common dependency
Thank you
Any Questions?