Accountability in Cloud Computing and Distributed Computer Systems

Hongda Xiao
Department of Electrical Engineering
Yale University

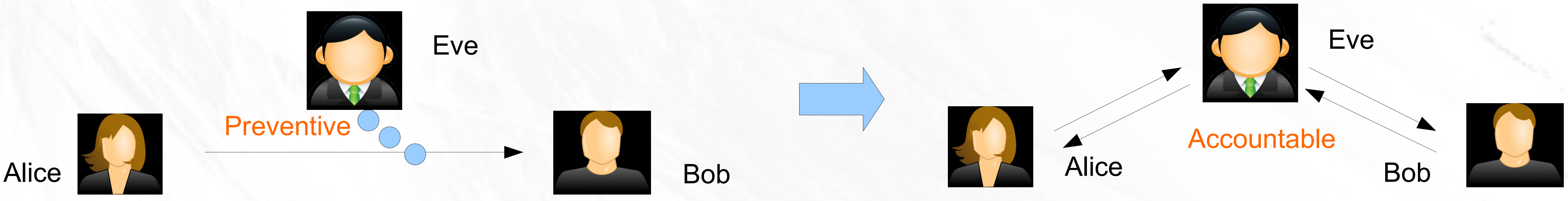
Thesis Defense, Sept. 17th, 2014
Advisor: Prof. Joan Feigenbaum

Accountability in Computer Science

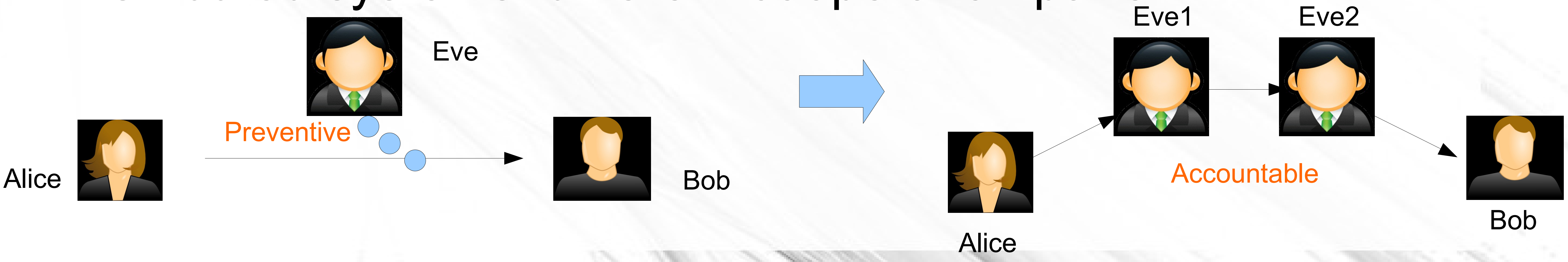
- Preventive methods are inadequate
  - More and more online activities
  - Inter-domain business transactions and information exchages
- Accountability is not a unified research area yet
  - Different researchers use the term to mean different things
  - Lack of practical accountability mechanisms for 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 Discloure, Violator Identification
  - Action
    - Centralized vs. Decentralized
    - Automatic vs. Mediated

Systematization of Accountability (3)

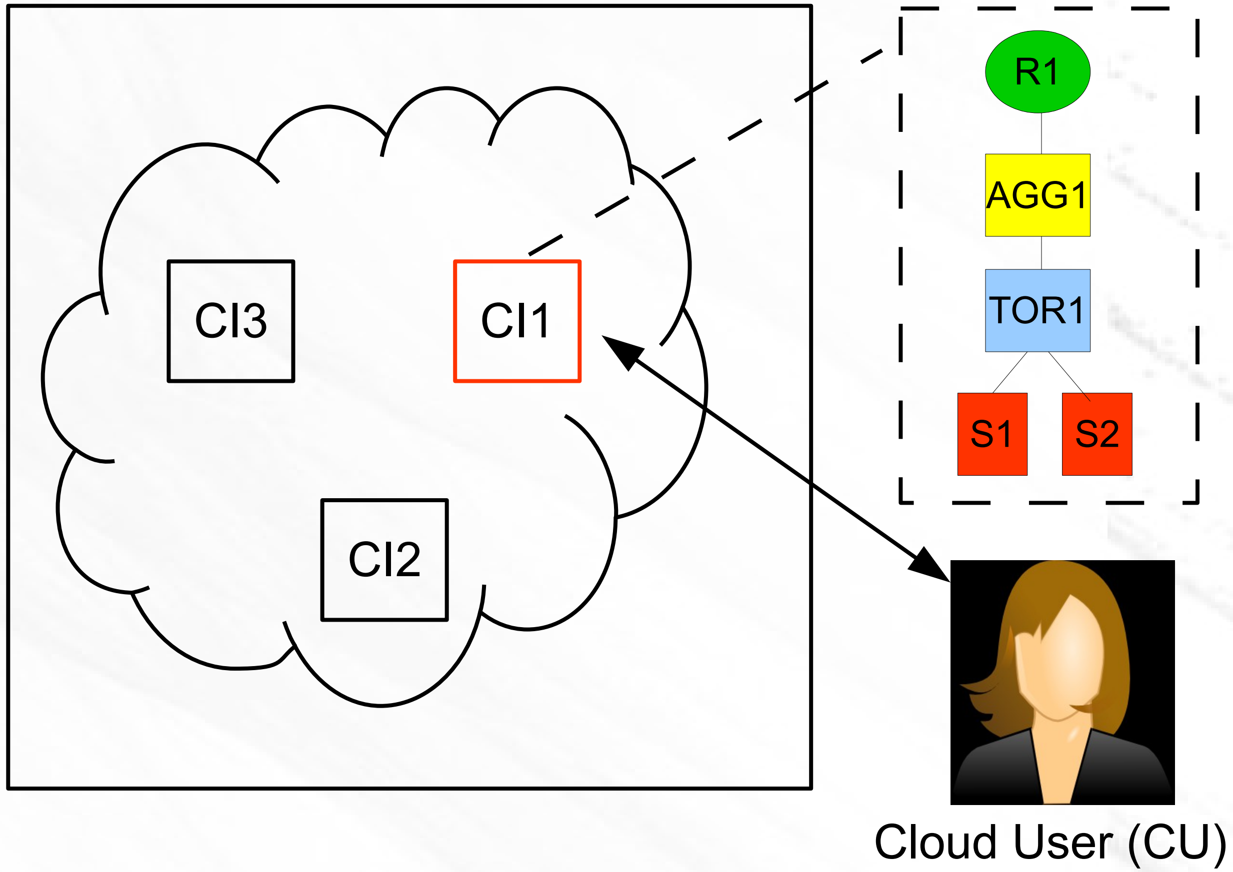
- Conclusions
  - "Accountability" is used to mean different things
  - Accountability does not preclude anonymity or privacy
  - Accountability need not be mediated by a central authority

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

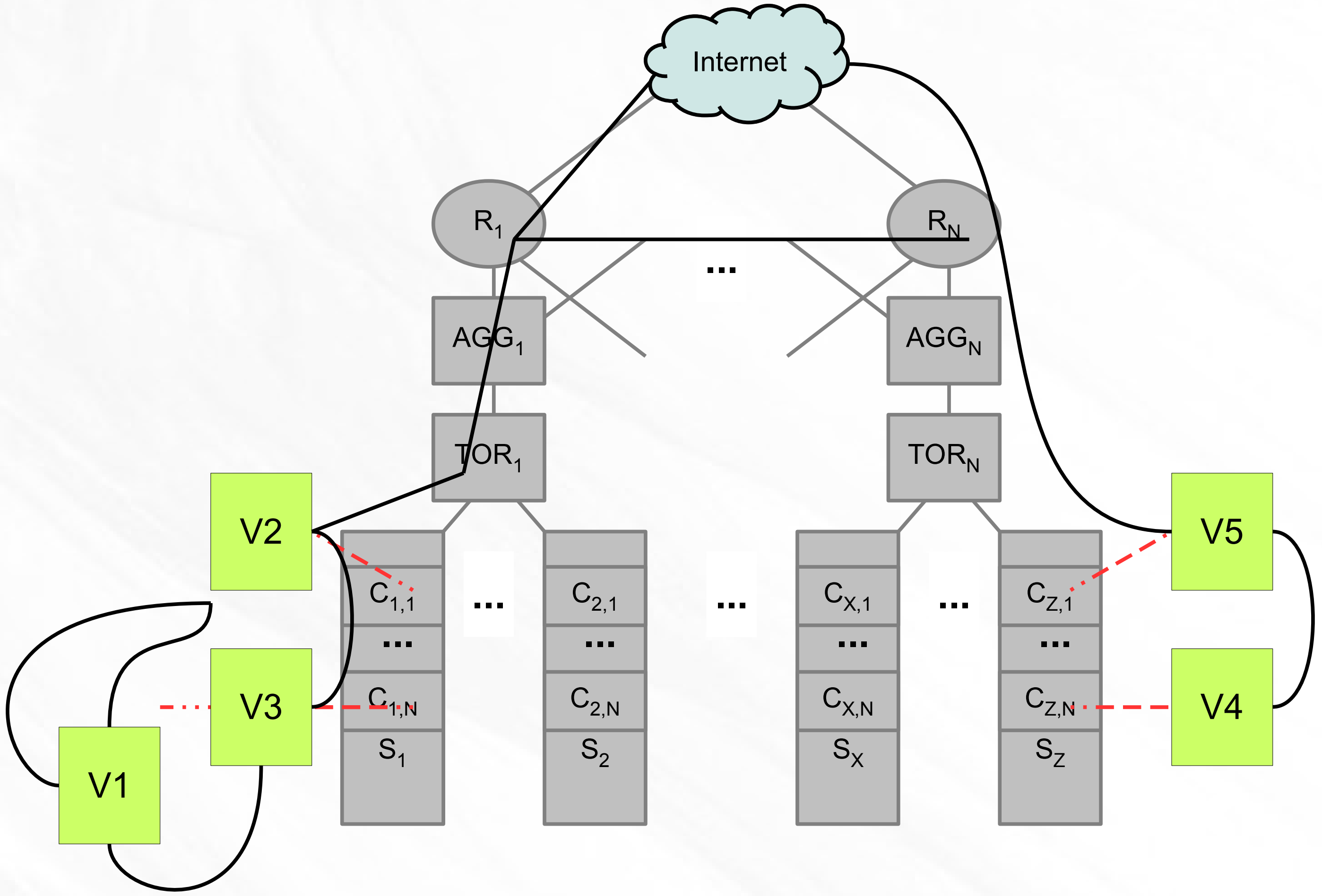
Cloud User Infrastructure Attestation

- 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



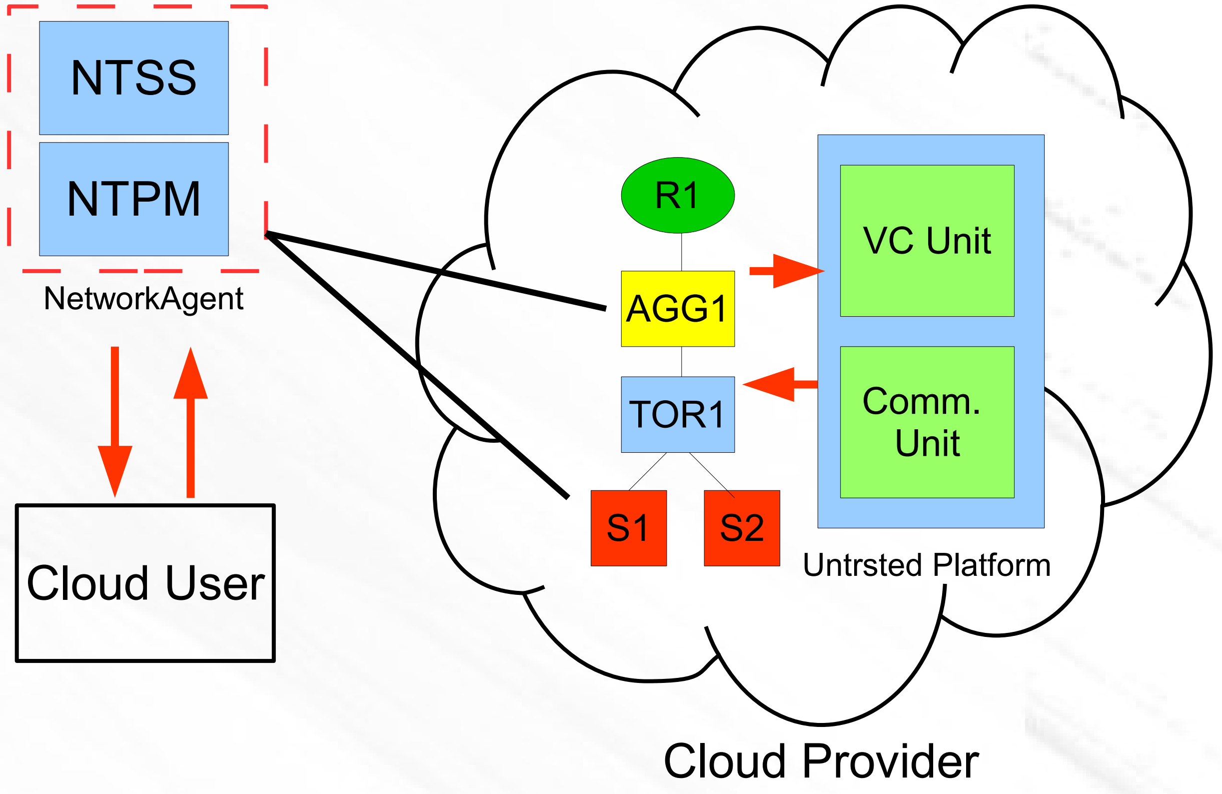
Cloud User Infrastructure

- 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

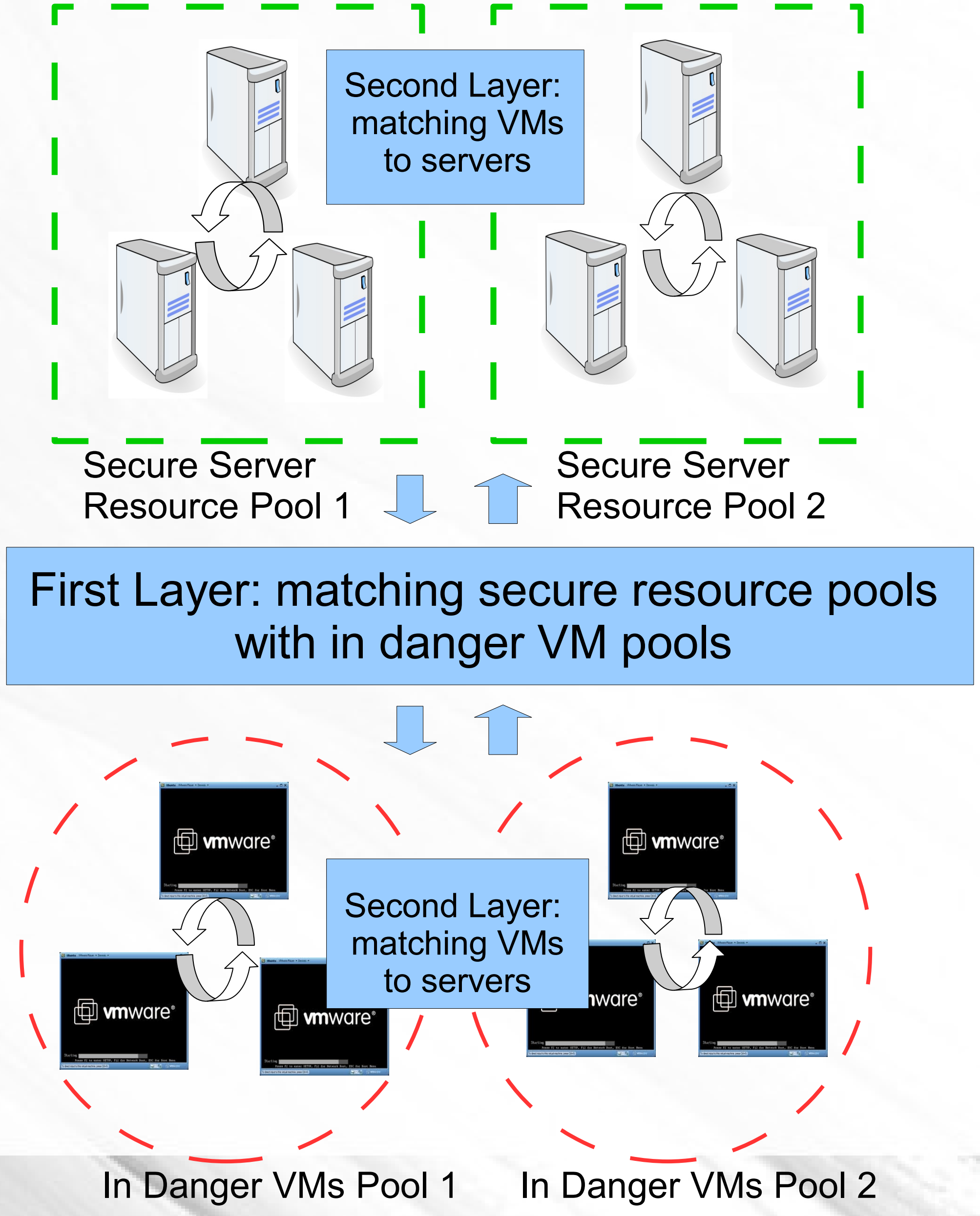
- A Novel Network TPM can collect topology information and serve as a security base
- A cloud provider attests to a cloud user that the cloud user infrastructure satisfies the requirements of the cloud user
- The cloud provider's infrastructure configurations remain private

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



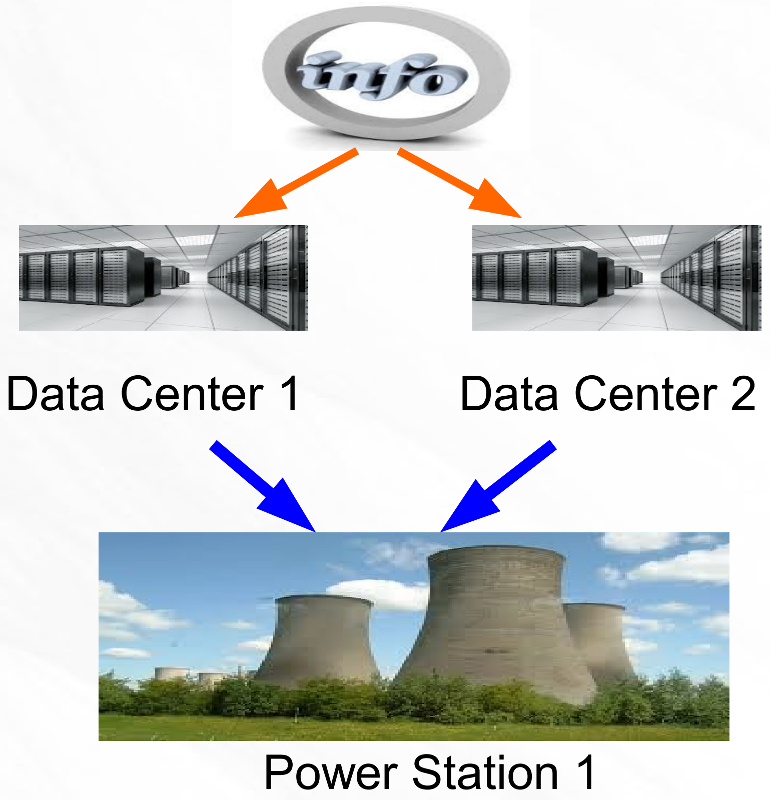
Overview

- 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

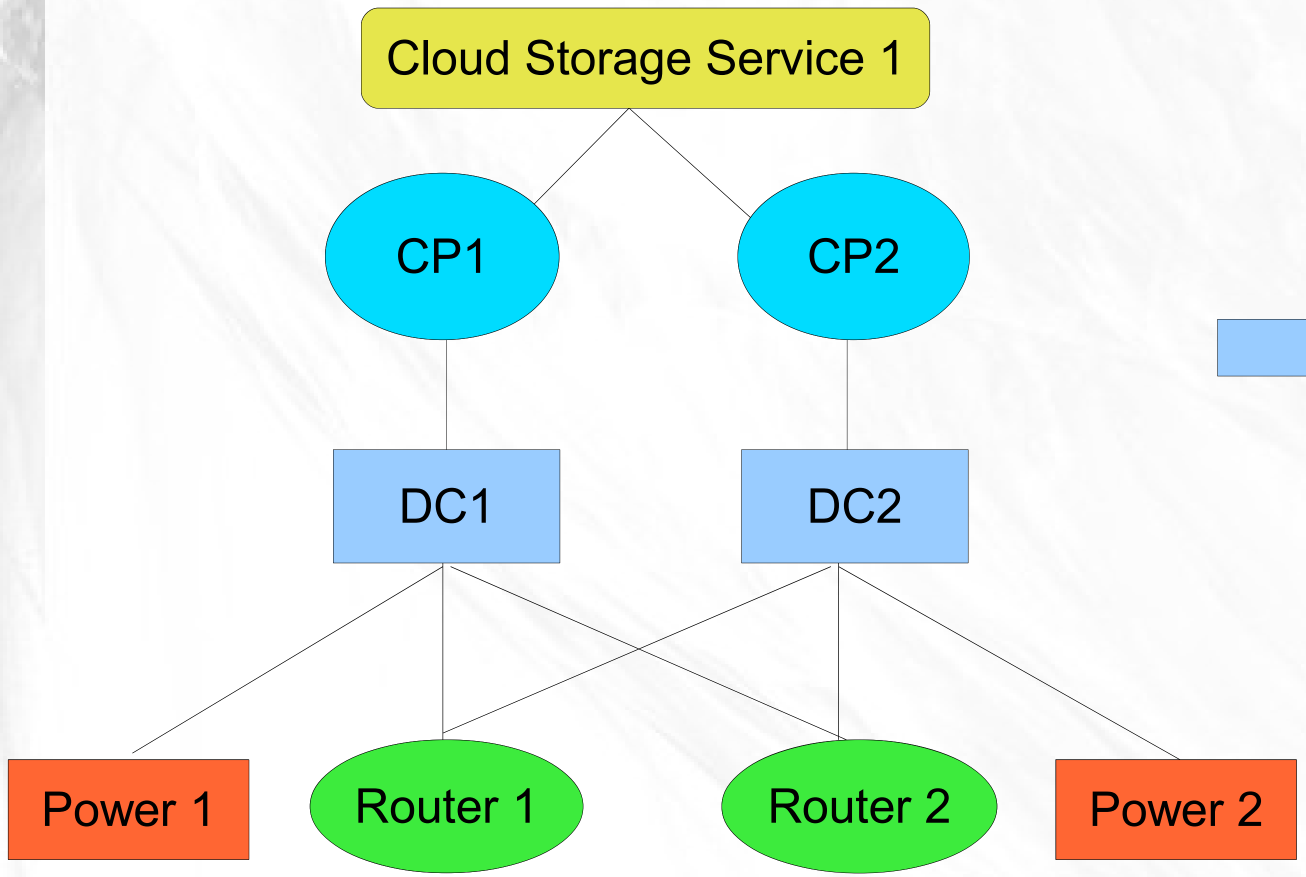
1. Structural Reliability Auditing

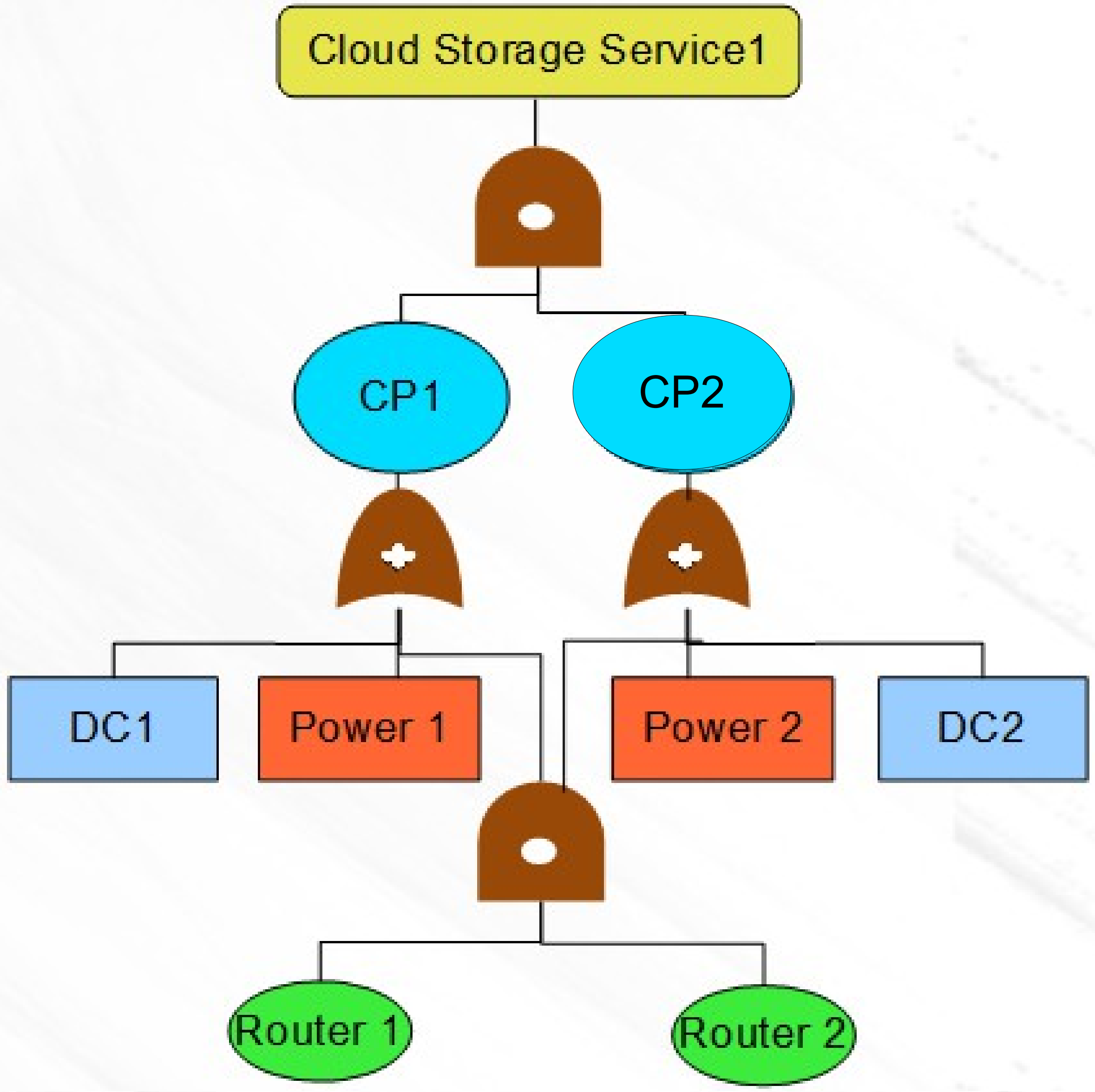
- Zhai, Wolinsky, Xiao, Liu, Su, Ford built a Structural Reliability Auditor (SRA)
  - collect comprehensive information from infrastructure providers
  - construct a service-wide fault tree
  - identify critical components; estimate likelihood of service outage
- An internal, structural cloud-audit 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 identifiying 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

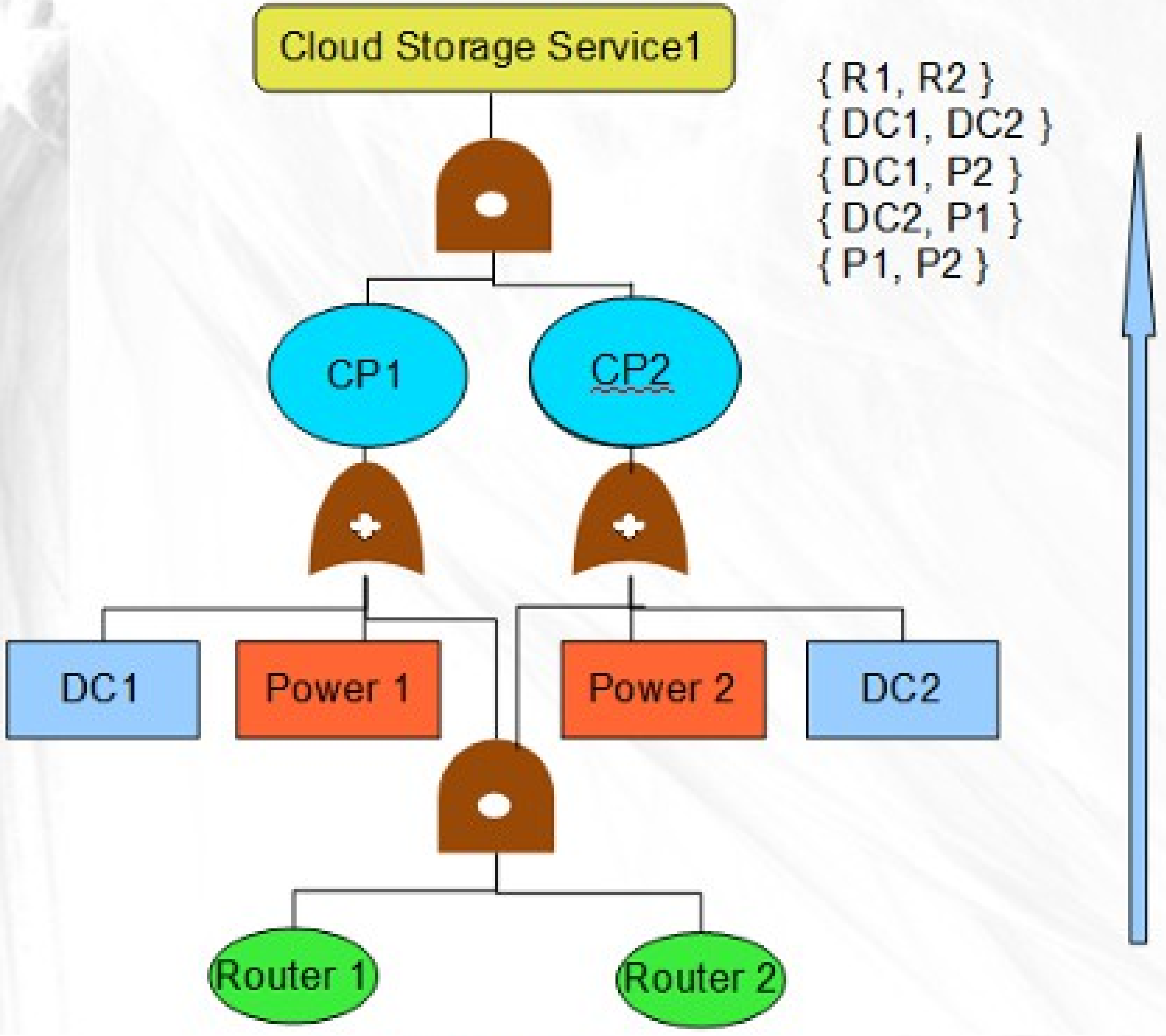




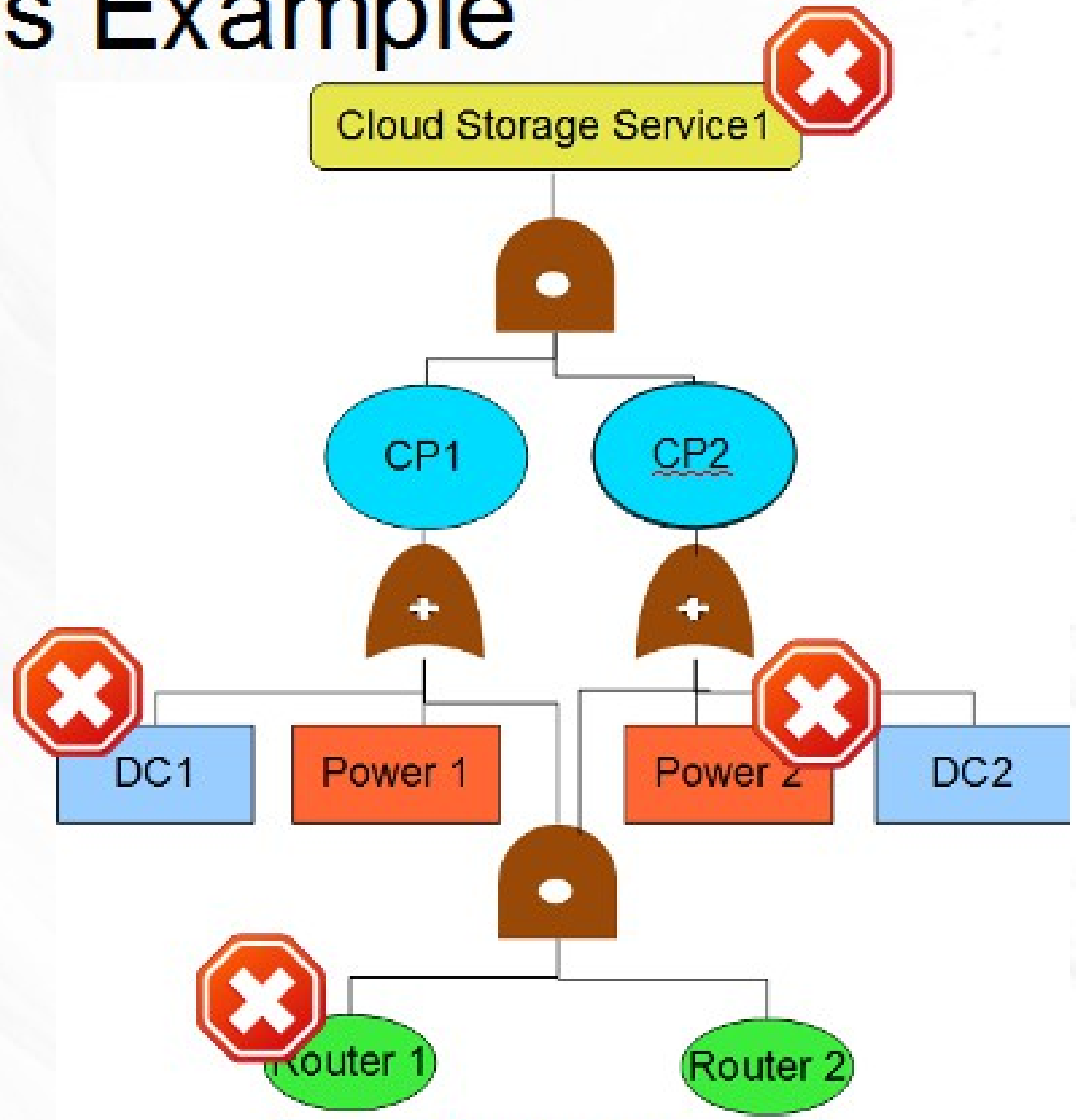
3. Failure Sets and Failure-Sampling Algorithm

- 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

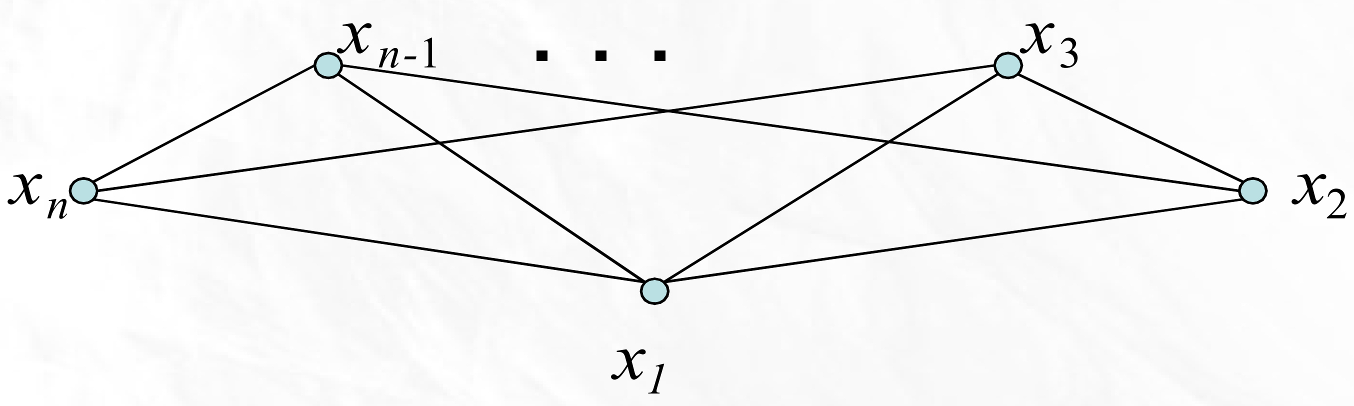


Minimal-FS algorithm



Failure-Sampling algorithm

4. Secure Multiparty Computation (SMPC)



$$y = F(x_1, \ldots, x_n)$$

- 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.

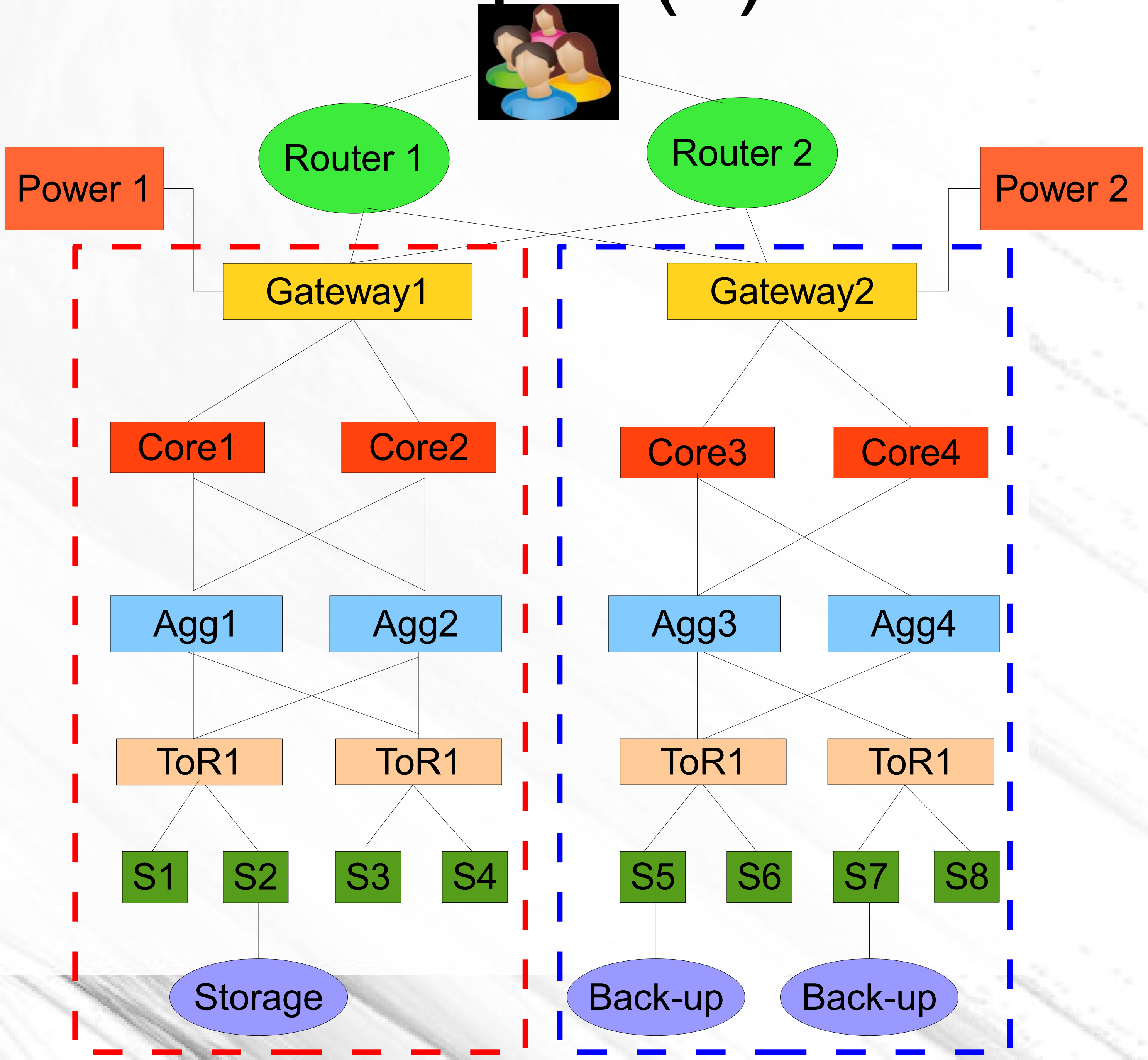
In our work, x_i is the cloud infrastructure of cloud provider i

5. Subgraph Abstraction

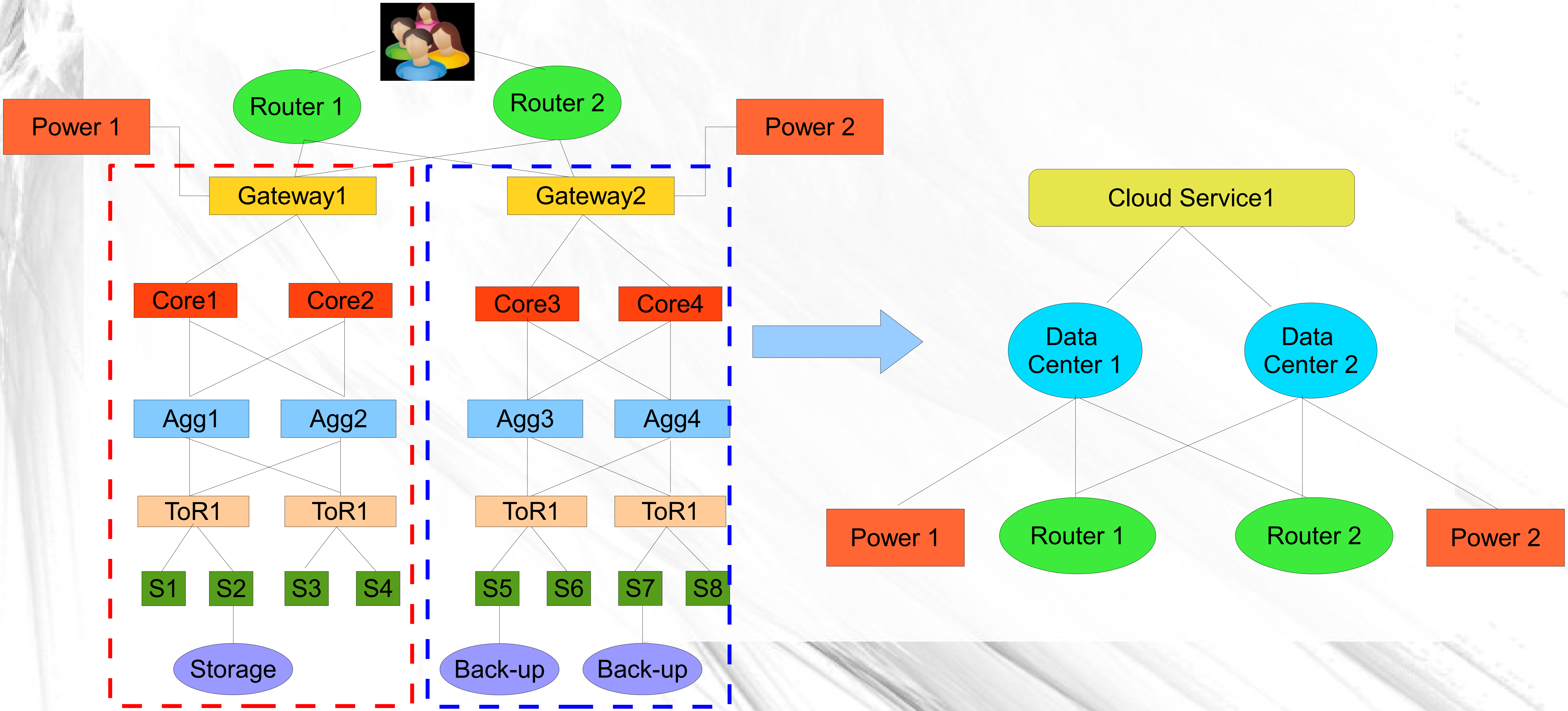
- Abstract the dependency information as a directed graph on macro-components; this will be the actual inputs to the SMPC
  - ✯ Macro-component: an abstracted (virtual) node in the dependency graph that can be considered an atomic unit for the purpose of structural-reliability analysis
- Key step in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

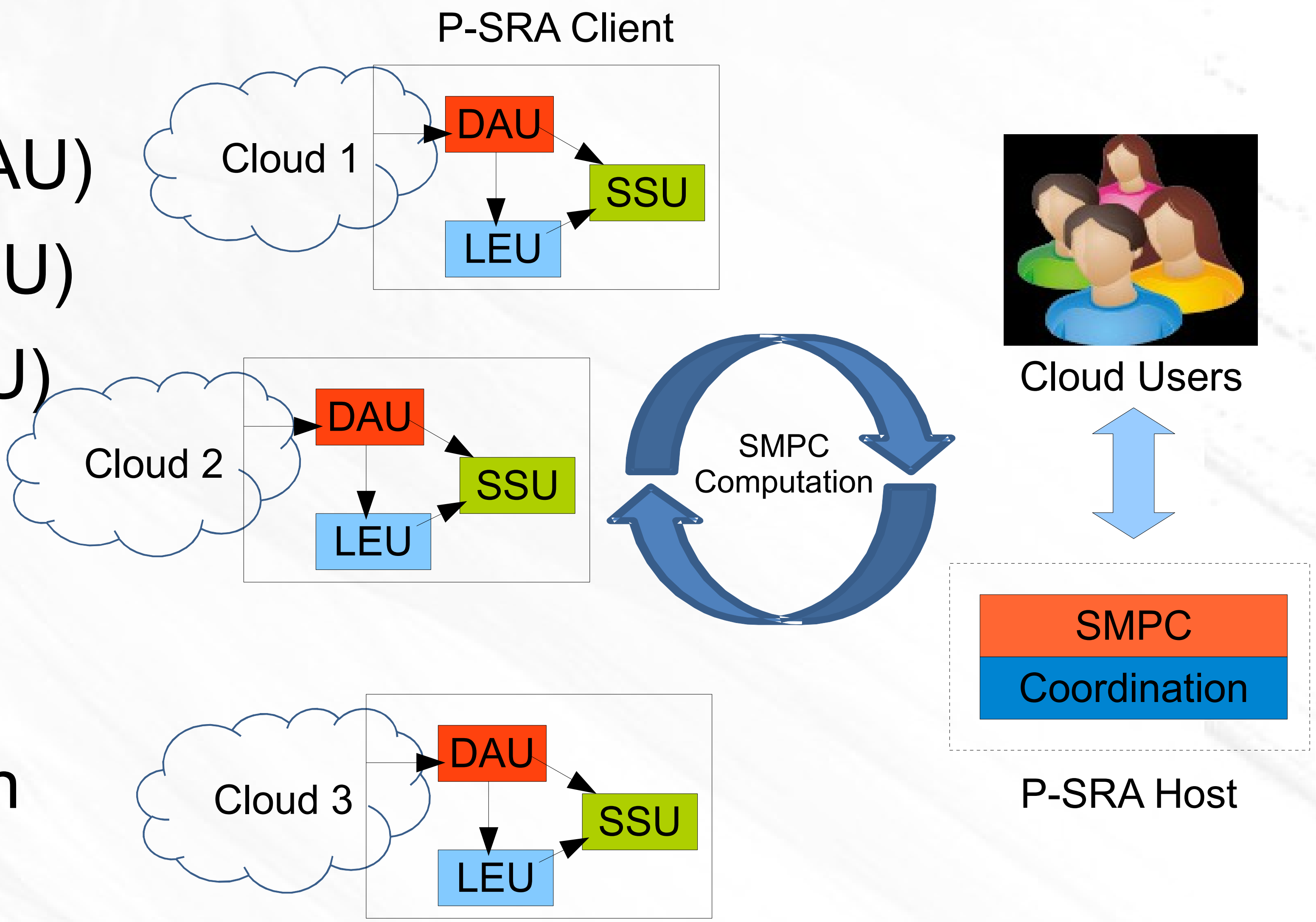


Subgraph Abstraction: Example (2)



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

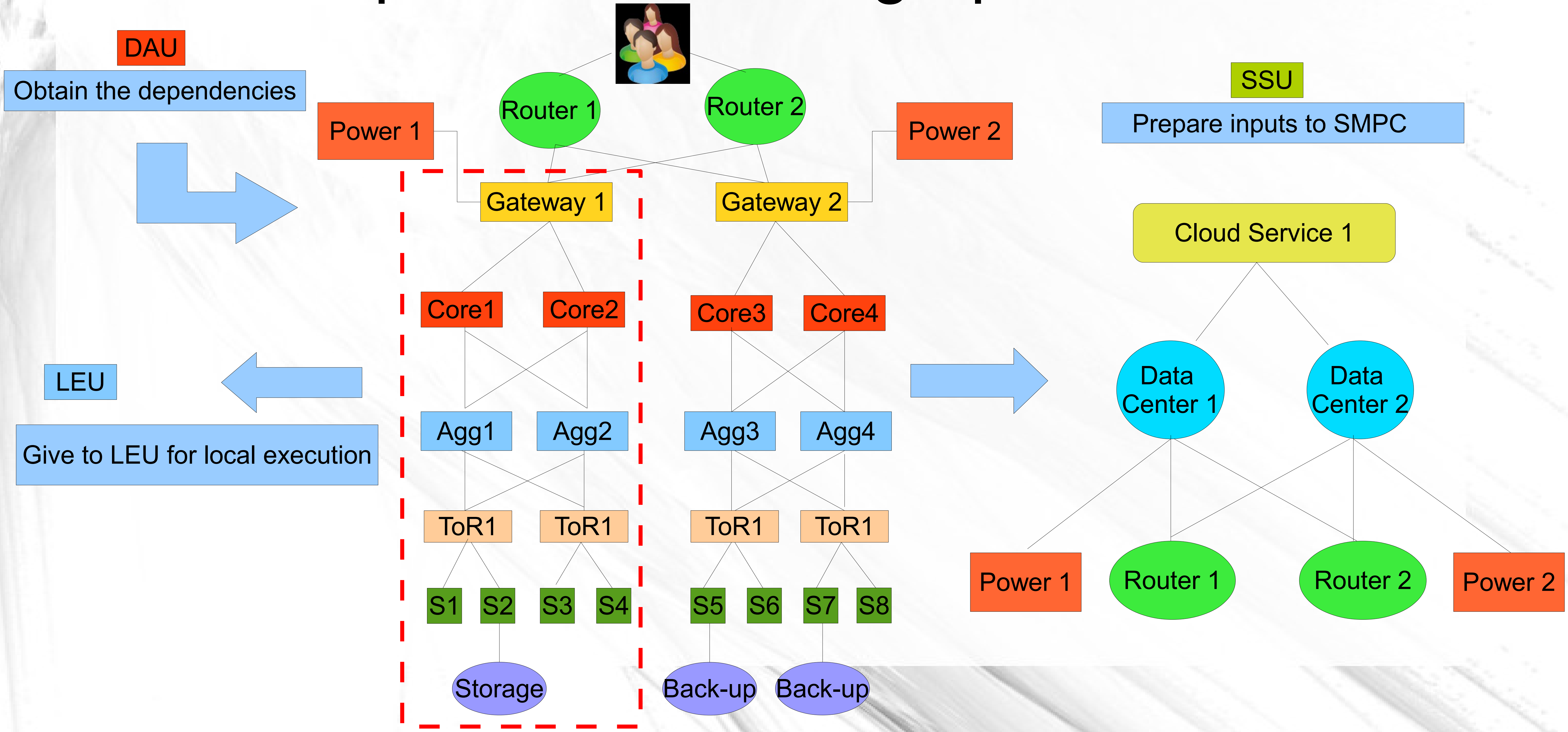
Step 1: Privacy-preserving dependency acquisition

- 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



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 informaiton 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

If (element a in Set B)

↓

Int a_in_B = $\Sigma_{b \in B}(a = b)$

SMPC: Fault-Tree Analysis

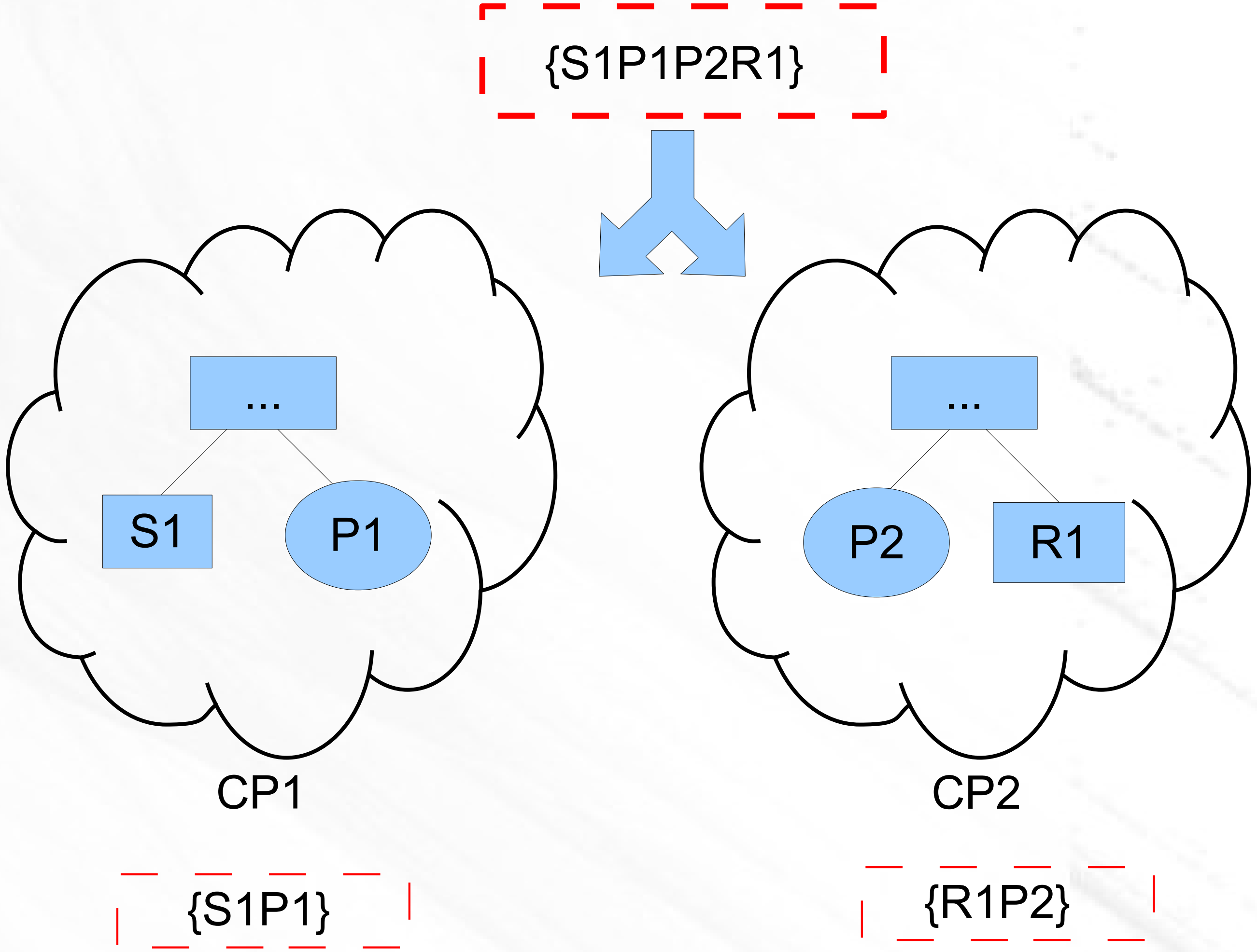
- Fault-tree construction
  - Represent the fault tree as "topology paths form with types" data structure
  - Eliminates use of conditionals
  - Cost: may be exponentially larger than DAG in worst case :(
- Calculate failure sets from the topology paths form with types
  - Minimal FSes Algorithm
  -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

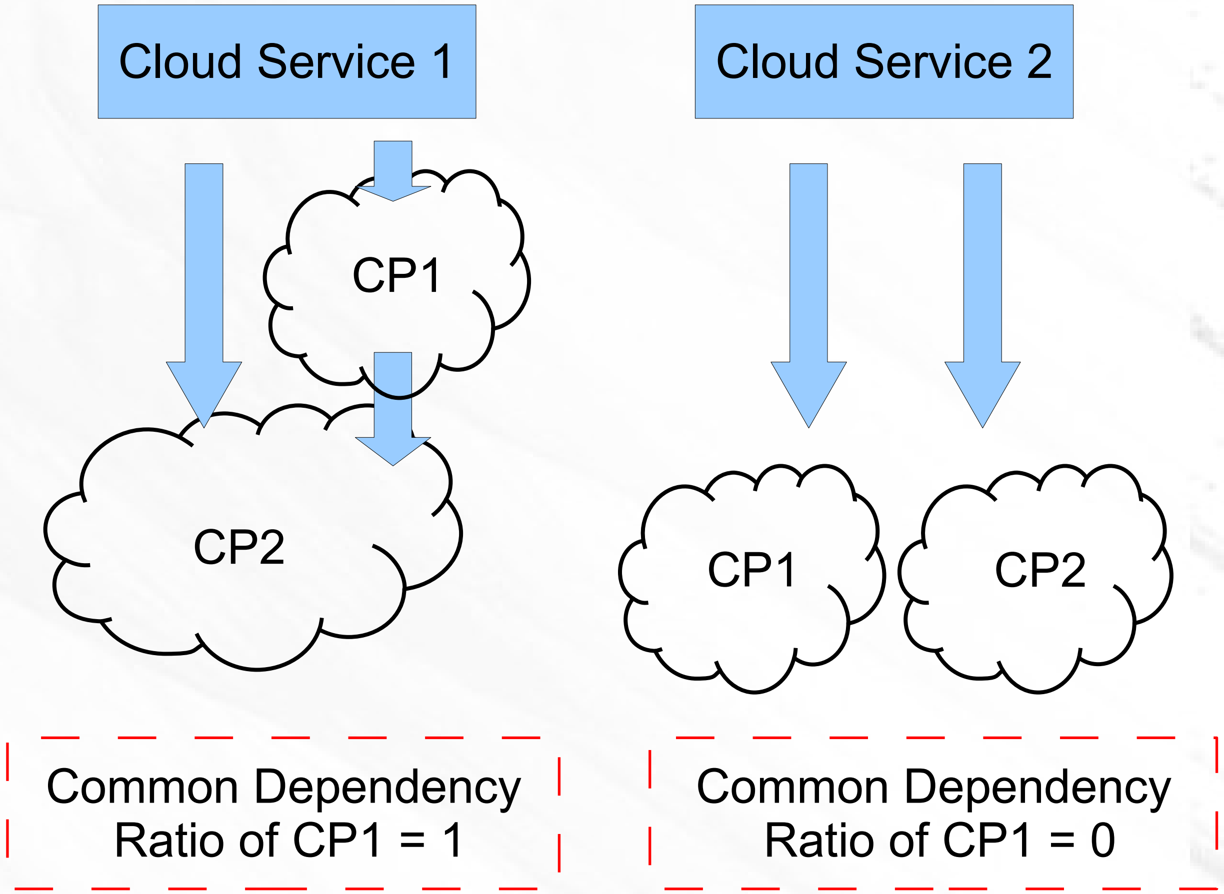
Partial Failure Set

- 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

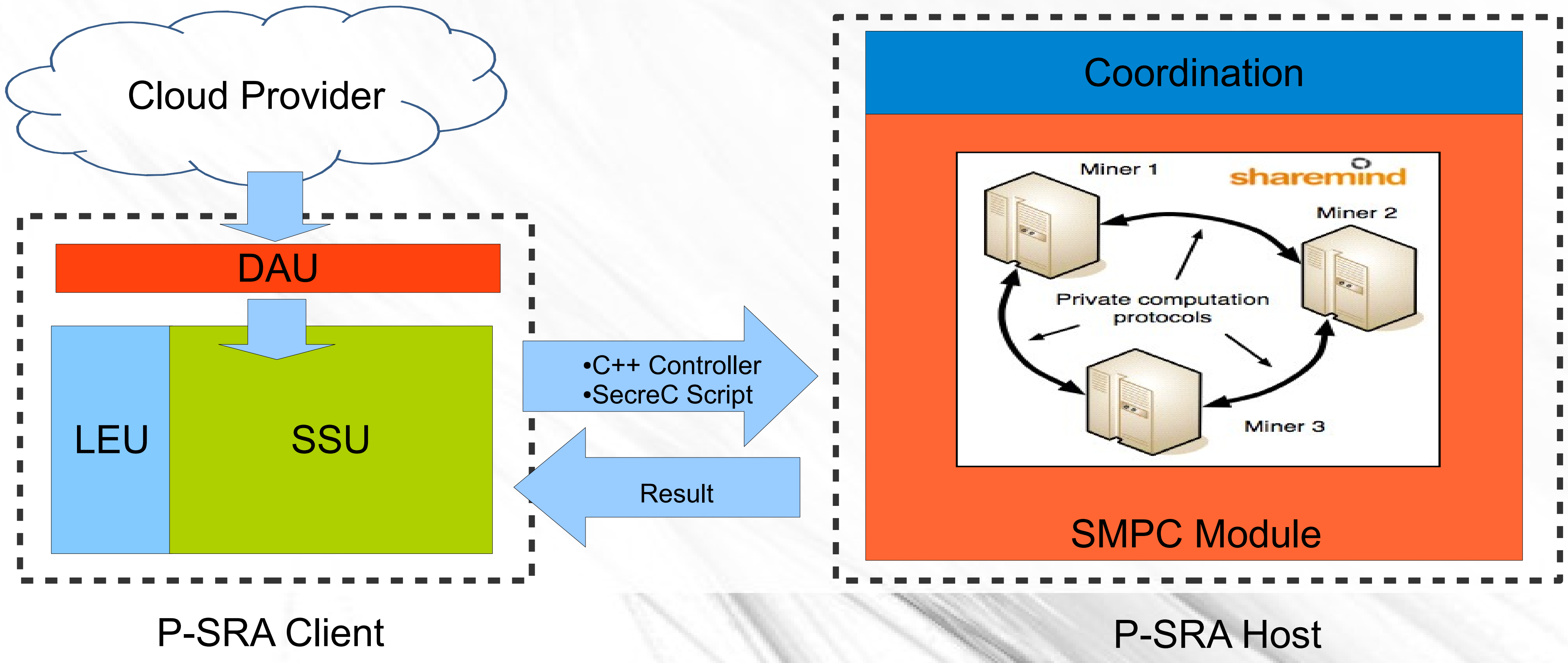


Top-Ranked Failure Sets

- 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

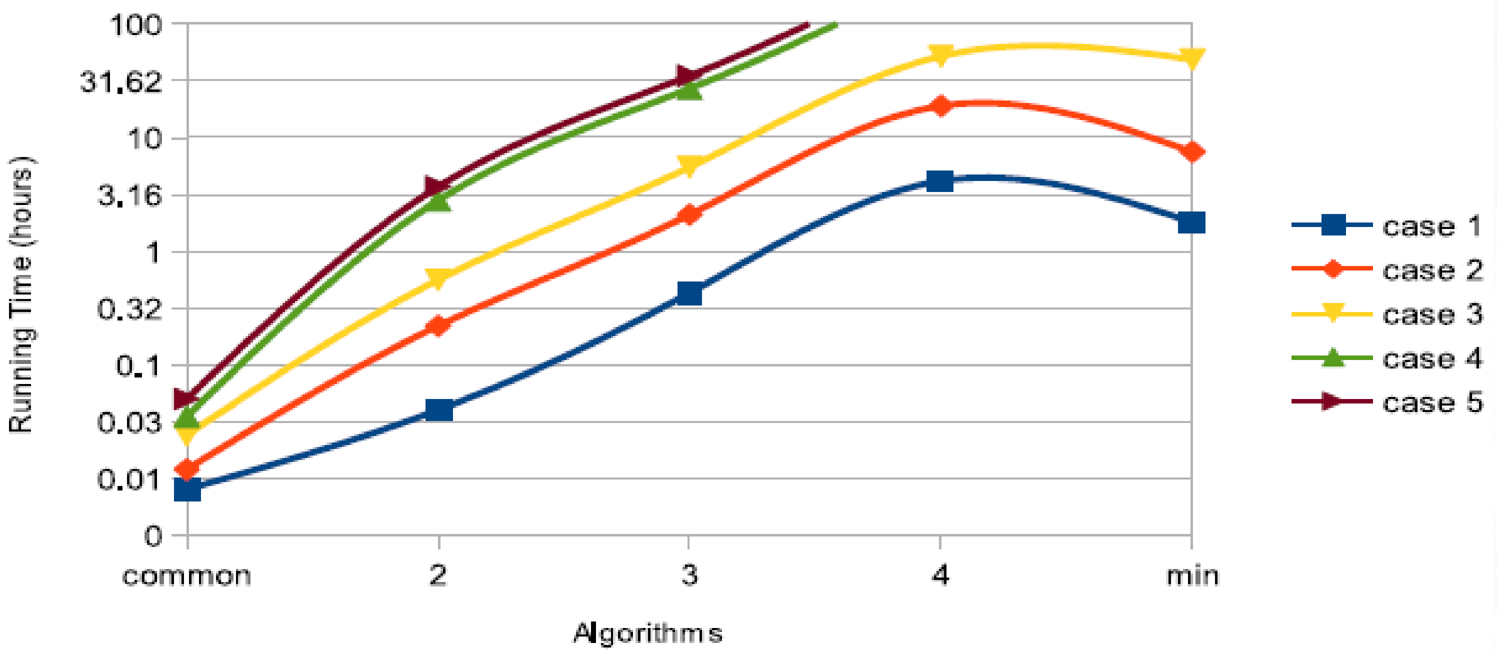
- Sharemind SecreC
  - C-like SMPC programming language
  - Specialized assembly to execute the code



Simulation: SMPC

	Case 1	Case 2	Case 3	Case 4	Case 5
# of cloud providers	2	2	3	3	2
# of data center	1	3	8	10	3
# of internet router	3	5	10	15	5
# of power stations	1	2	3	5	2
ratio of common dep.	0.8	0.2	0.2	0.2	0.2
ratio of padding	0.0	0.0	0.0	0.0	0.5

Table 1: Configuration of Test Data Sets



- Practical as an offline service
- Used a low-end laptop – performance would improve on a workstation

Simulation: Local Execution

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

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

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

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?