Limiting Data Exposure in Monitoring Multi-domain Policy Conformance

Mirko Montanari, Jun Ho Huh,
Rakesh B. Bobba, Roy H. Campbell
University of Illinois at Urbana-Champaign

TRUST 2013
June 17, 2013, London, UK

Information Trust
INSTITUTE

Assured Cloud Computing
University Center of Excellence
Sponsored By: AFRL/AFOSR

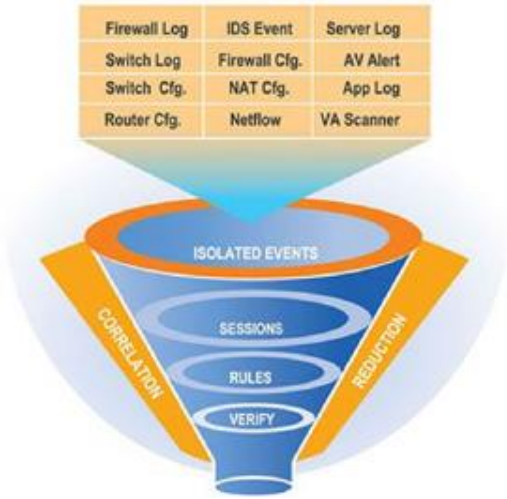
Multi-Organization Systems

2012: 44 million compromised records*
2005-2008 (US): estimate 227 million records**

Security Information and Event Management Systems (SIEM)

- 85+ products on the market in 2012

Gather, analyze, and present security relevant information collected from devices, applications, and users



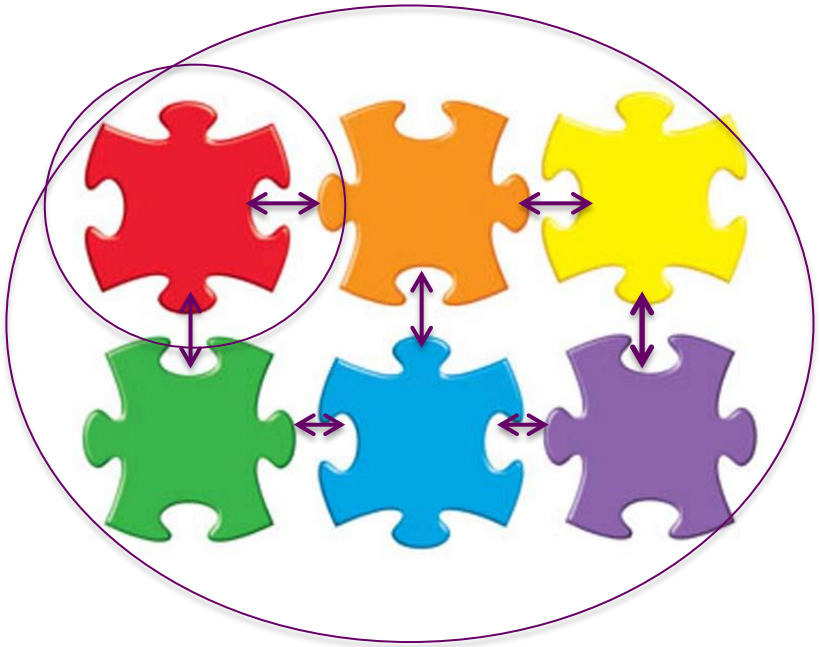
Cloud Computing

Cloud providers
Cloud users

Airport infrastructure

Airlines
Airport management
Maintenance contractor

Tradeoff: Confidentiality vs Detection



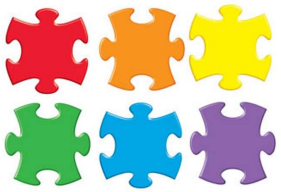
Events provide knowledge about:

- network topology
- network traffic
- configurations
- installed programs
- vulnerable programs
- user behaviors
- services
- critical machines
- …

Complete confidentiality — Complete openness

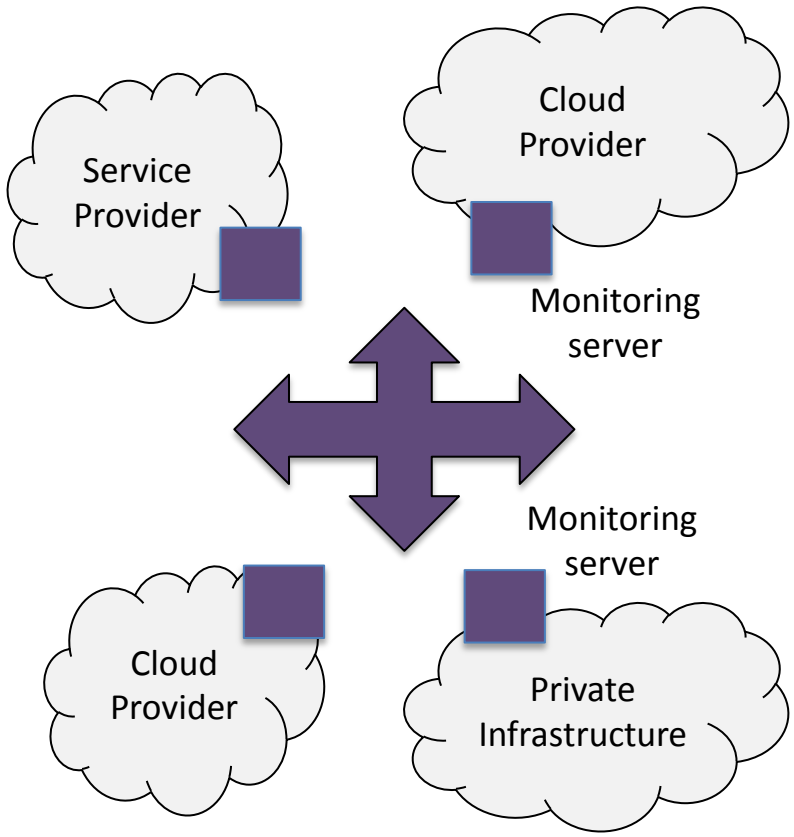
Only detection of **local** security concerns

Detection of **global** security concerns



Can we find a tradeoff?

Monitoring Architecture



Multi-organization event-based monitoring

- Built on top of current monitoring architecture
- Each organization detect problems in its infrastructure independently

Contributions:

- Minimum information sharing / need-to-know in multi-organization systems
- Distributed logic reasoning algorithm for policy compliance
- Minimal sharing obtainable for simple policies; reduces information exposure for more complex policies

Policy-based Approaches

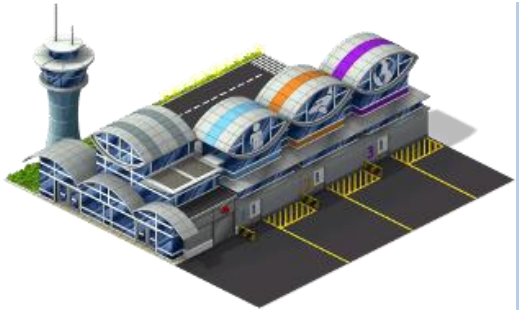
"1.3) Prohibit direct public access between the Internet and any system component in the cardholder data environment."

"6.1) Ensure that all system components and software are protected from known vulnerabilities by having the latest vendor-supplied security patches installed. Install critical security patches within one month of release."

96% of victims subject to PCI-DSS had not achieved compliance [Verizon Data Breach Investigation Report 2012]

> […] nearly every case that we have seen thus far has attributes of its breach that could have been prevented if the control requirements had been properly implemented. […]

Examples of Application Domain



Maintenance contractors ←→ airline

e.g., Maintenance crew and device must be located on airport tarmac when accessing external access point of aircraft

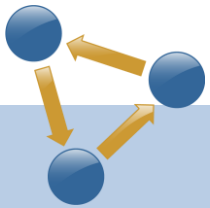
Cloud user ←→ Cloud provider

e.g., critical services *should not run on a* physical server *which is sending* malicious traffic *from one of its virtual machines*

Challenges

Discrete Events

- e.g., configuration changes, failures, audit logs
- Hard to summarize
- Current anonymization techniques focus on numeric data



Distributed architecture

- Cannot rely on a single entity to process information
  - Confidentiality of records; liability reasons
- Multiple monitoring systems interacting without a single point of aggregation

State-based Representation: Datalog

Monitoring Rule: A violation is detected if a critical service is running on a physical host which is sending malicious traffic

VM instance $inst_1$ is running a critical service "apache"

runsCriticalService($inst_1$, apache)

VM $instance_1$ is assigned to physical server ps_1

instanceAssigned($inst_1$, ps_1)

Malicious traffic detected from ps_1

badTraffic(ps_1)

runsCriticalService($inst_1$, apache), instanceAssigned($inst_1$, ps_1), badTraffic(ps_1)
→ $violation_A$($inst_1$, apache)

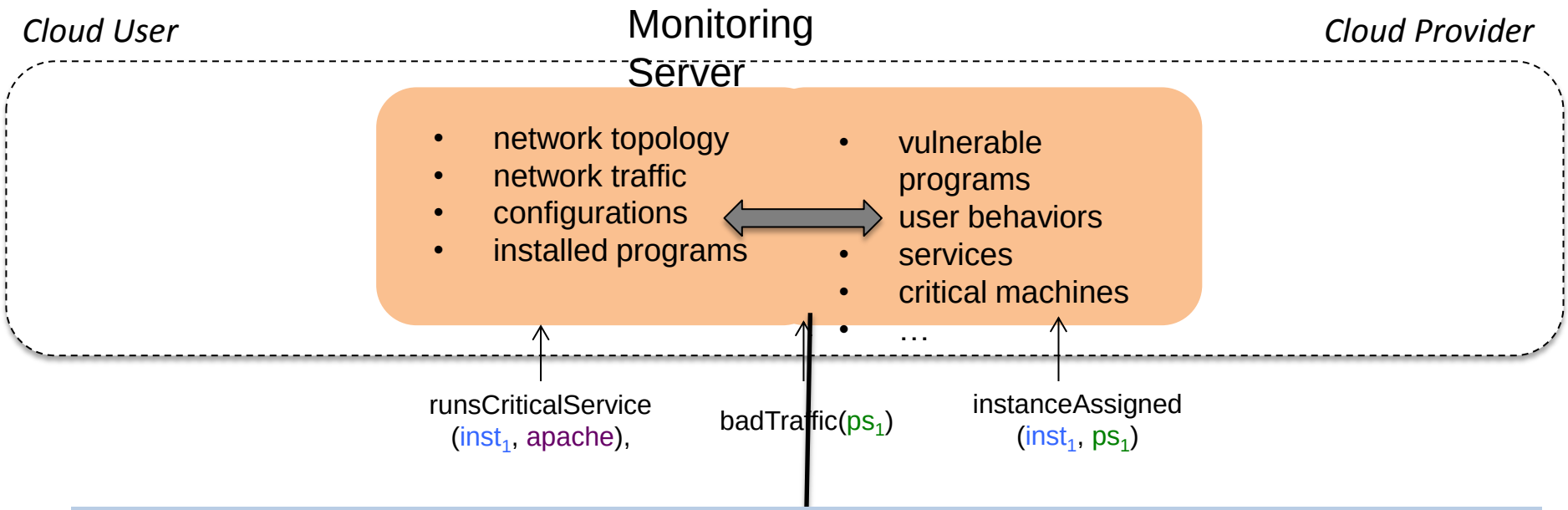
runsCriticalService(I, P),
instanceAssigned(I, S), badTraffic(S).
→ $violation_A$(I, P)

Correlation process is logic reasoning

I: VM instance
P: program
S: physical server

Event Aggregation

Event correlation: process of analyzing events for detecting complex conditions

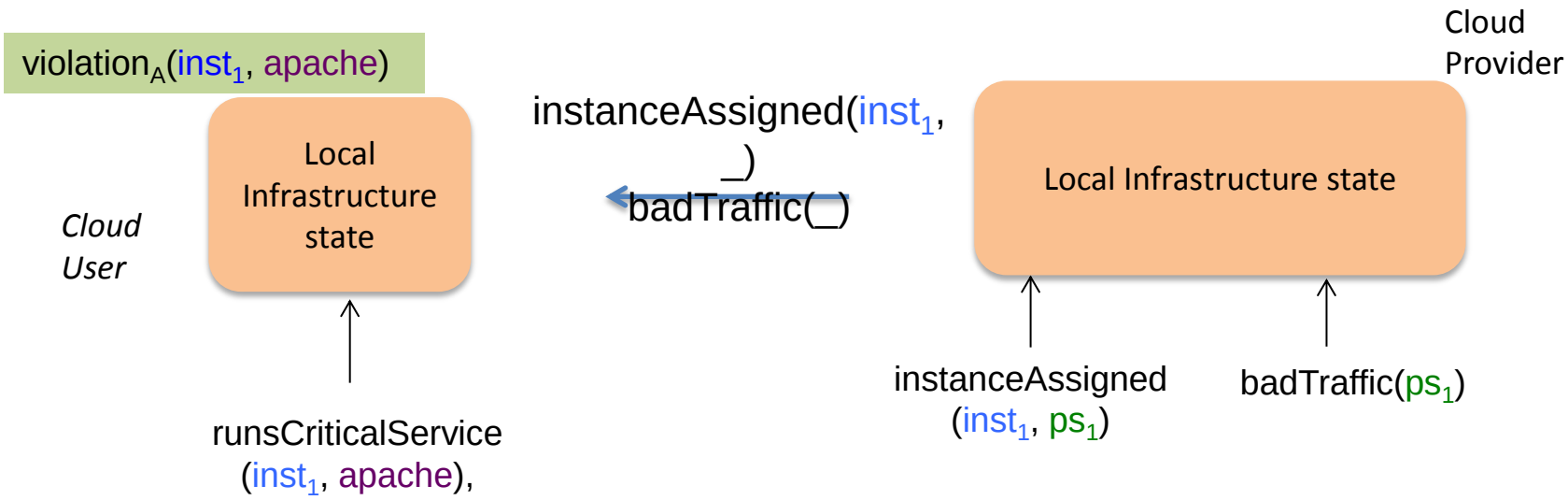


Need-to-know set: information needed for inferring the presence of a violation

Observation: If no violation, no need to share actual events

Minimal Sharing Example

Locality: classifying events into local and remote



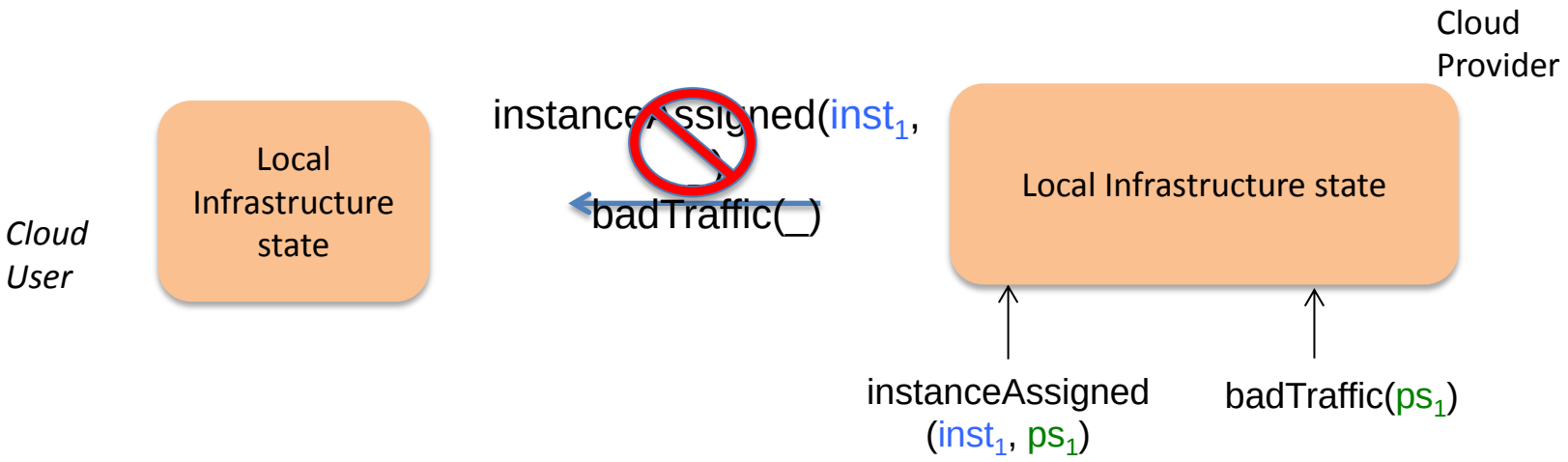
violation$_A$(I, P) ← runsCriticalService(I, P), **instanceAssigned(I, S), badTraffic(S)**.

external for cloud users

I: VM instance
P: program
S: physical server

Minimal Sharing Example (II)

Conditional Sharing: events shared only if match found on the other side



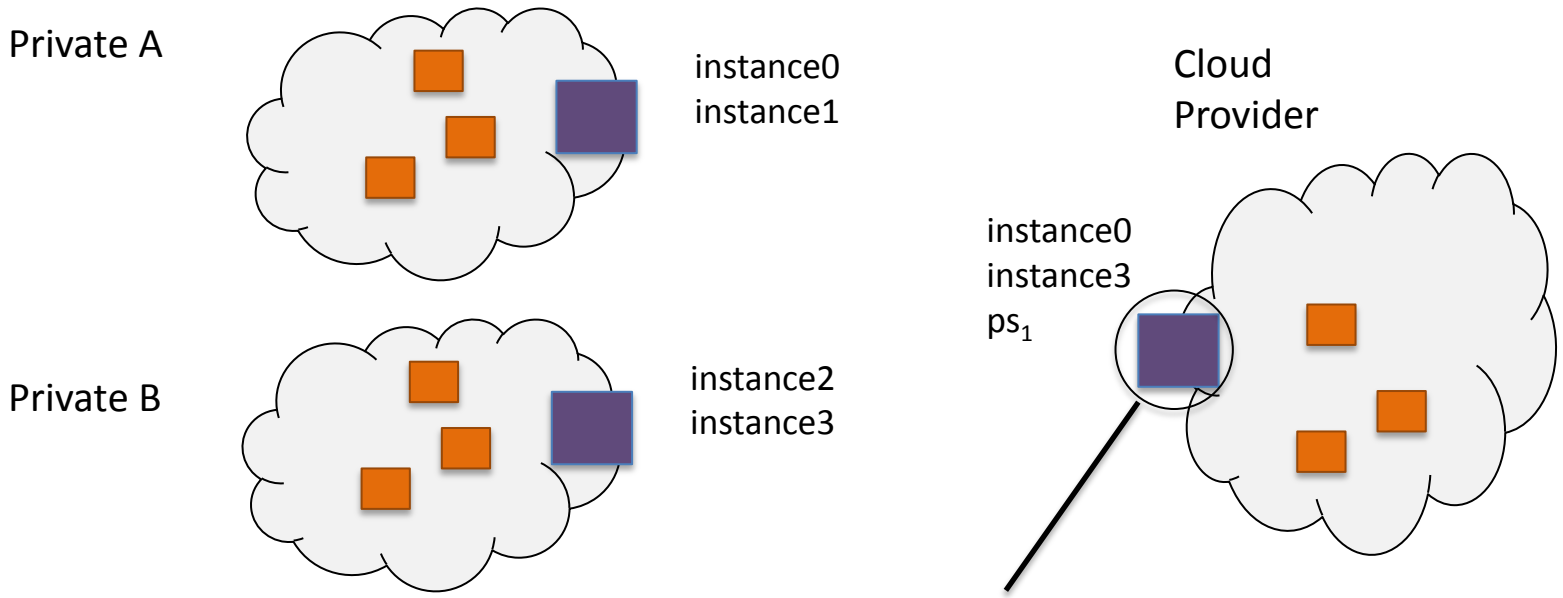
violation$_A$(I, P) ← runsCriticalService(I, P), **instanceAssigned(I, S), badTraffic(S)**.

external for cloud users

I: VM instance
P: program
S: physical server

Resource-based Overview

Resource: unique names for entities in the system. e.g., hosts, users, programs



violation(inst0, p) ← instAssigned(inst0, ps_1), badTraffic(ps_1).

Resource-data completeness

If a monitoring server receives all events regarding a particular resource *r*, rules which body include all events containing *r* can be processed locally

Intuition: Resource-based Rewrite

Complex policies rewritten to correlate events about a **single resource** at each step

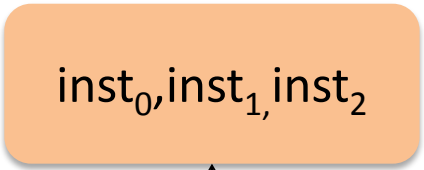
violation ($inst_0$, p) ← runsCritService ($inst_0$, p), instAssigned($inst_0$, ps_1), badTraffic(ps_1) .

violation($inst_0$, p) ← runsCritService ($inst_0$, p), partial($inst_0$).

partial($inst_0$) ← instAssigned ($inst_0$, ps_1), badTraffic(ps_1)

Distributed Correlation

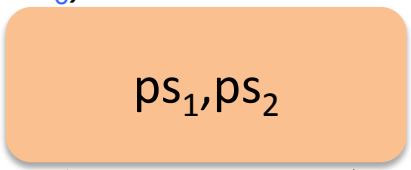
violation(I, P) ←
runsCritService (I, P), partial(I).
partial(I) ←
instAssigned(I, S), badTraffic(S)



runsCritService
(inst0, p)

violation(I, P) ←
runsCritService (I, P), partial(I).
partial(I) ←
instAssigned(I, S), badTraffic(S)

partial(inst0)

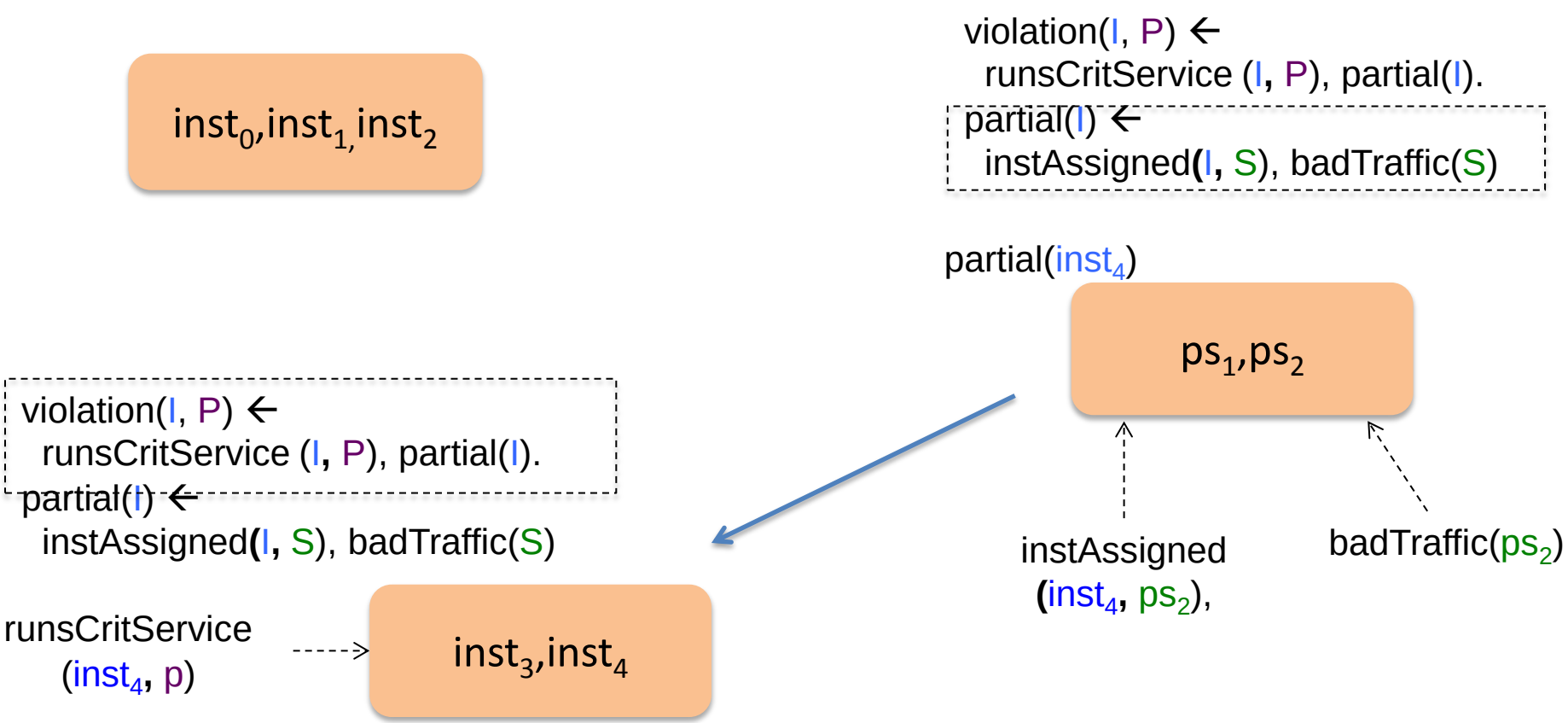


instAssigned
(inst0, ps1),

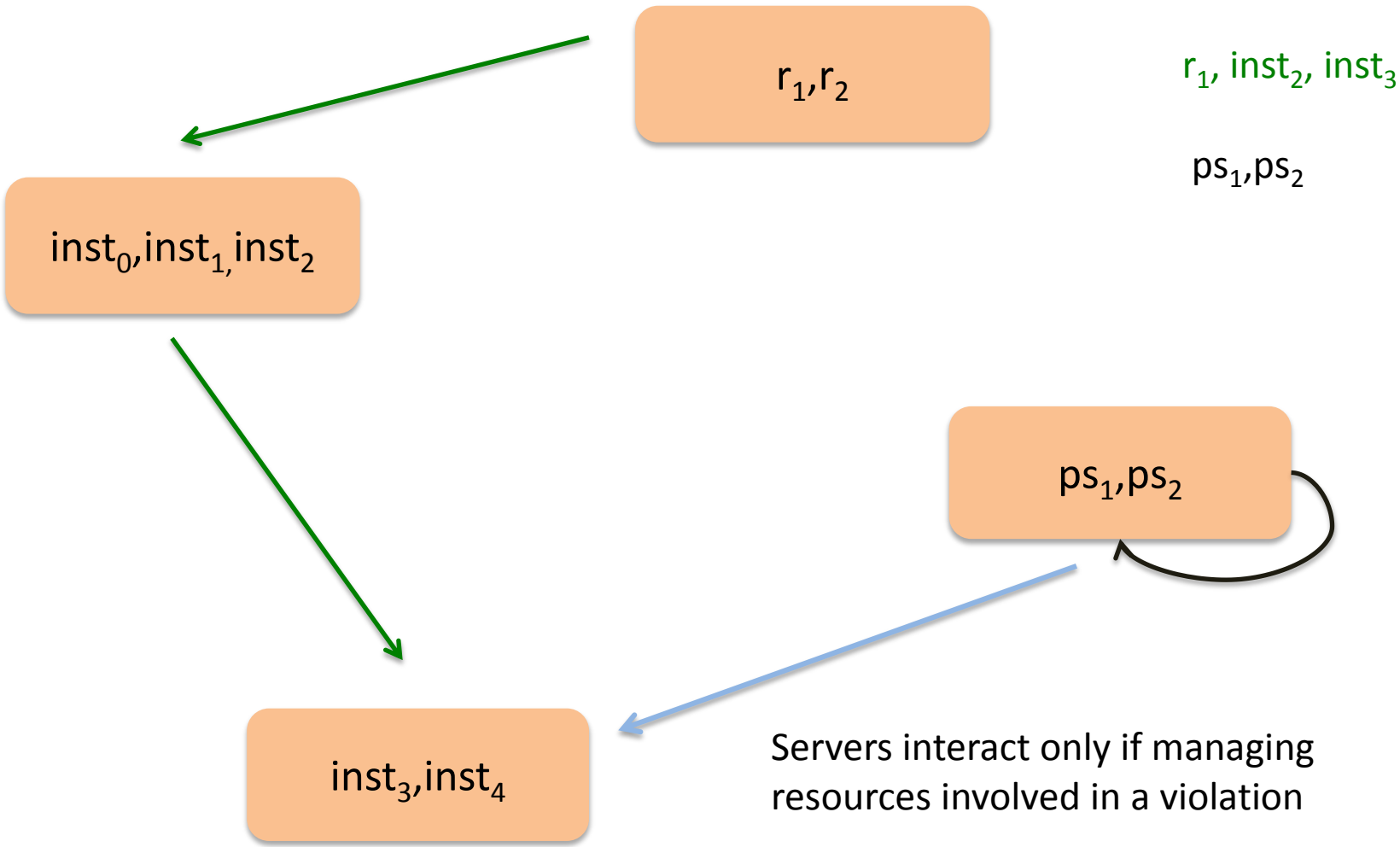
badTraffic(ps1)

Process locally, send to the next monitoring system

Distributed Correlation



Distributed Correlation



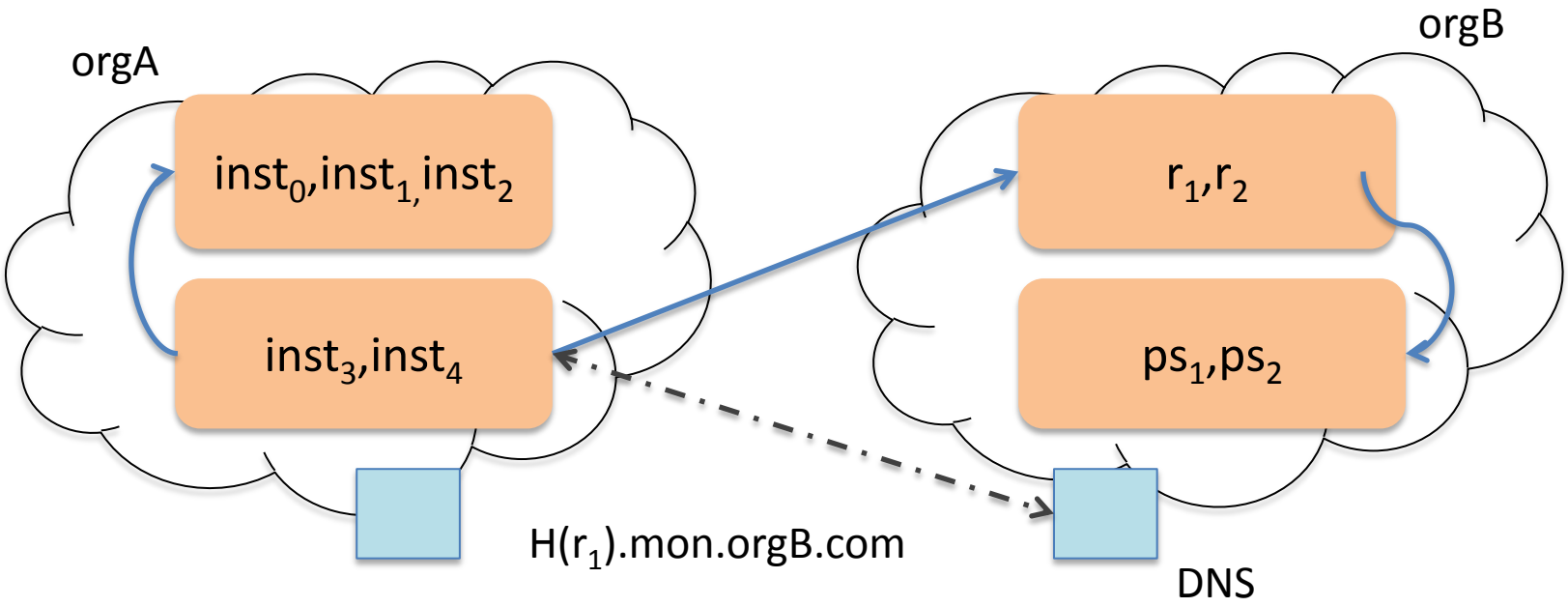
Servers interact only if managing resources involved in a violation

Local detection of all-local violations

Resource-based Processing- Naming

Multiple monitoring servers within each domain

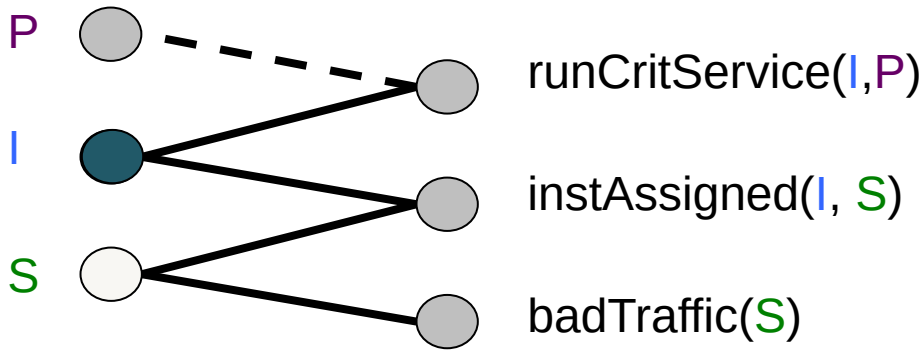
- Distributing load / information across multiple servers

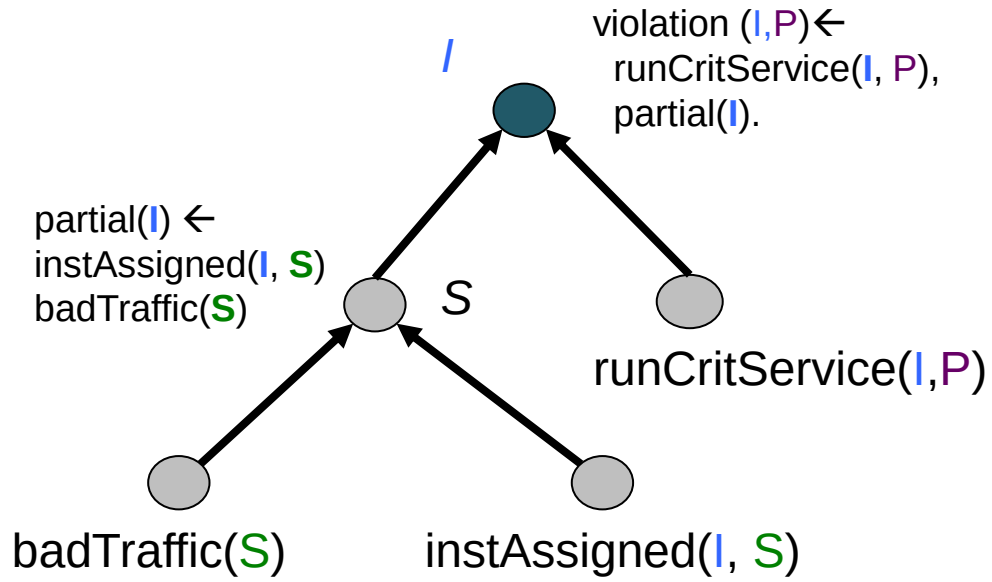


DNS-based naming system to specify managed resources

Event Correlation Trees

violation(I, P) ← runCritService(I, P), instAssigned(I, S), badTraffic(S) .



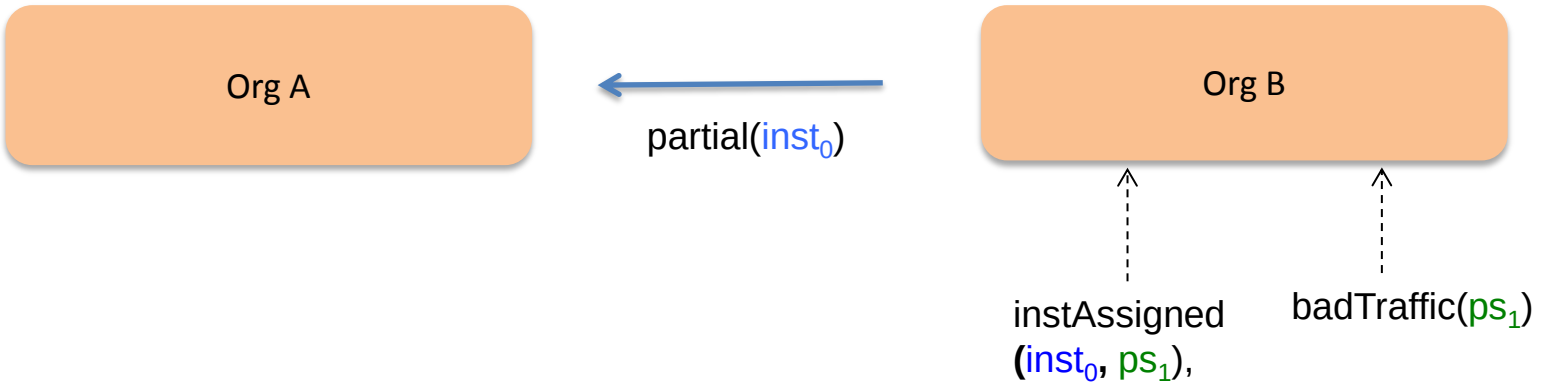


I: VM instance
P: program
S: physical server

Problem: Unilateral Sharing

violation(I, P) ←
runsCritService (I, P), partial(I).

partial(I) ←
instAssigned(I, S), badTraffic(S)



When a rule is satisfied on a monitoring server, the resulting event is shared unilaterely, without checking if it is relevant to a violation

Conditional Sharing

An event is shared only if there is a matching event on the remote server

Secure Two-Party Computation

Conditional Sharing

r=sharing if events a,b match the policy

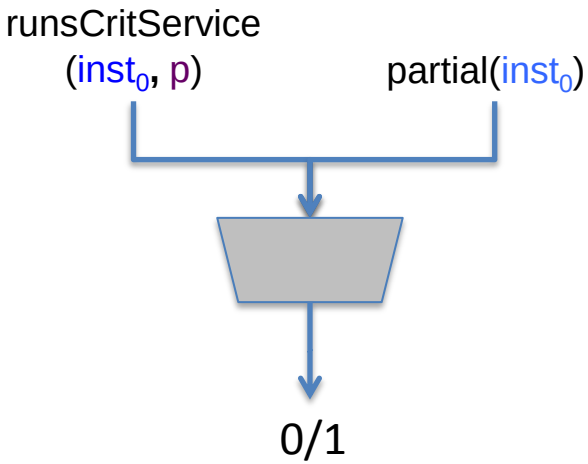
- Event a known only by org A
- Event b known only by org B

Determine if the two events match without revealing them to the other party

Garbled Circuits [Yao, 1986; Huang, 2012]

- Fast secure two-party computation

1. Encode each resource-based rule as a combinatorial circuit
2. Event parameters as input from each organization
3. If result is true, the event is shared
   - If not, almost no information is leaked
4. Repeat for each couple of private events



Event-based Representation

Alternative (more powerful) representation of policies and events

- Temporal conditions (e.g., before, precedes, overlaps)

violation(I, P) ← runsCritService (I, P), partial(I,S)

*critical operation **overlaps** a component failure*

*malicious traffic detected **during** execution of vulnerable software*

violation(I, P) ←
E1 type runsCriticalService
E1 instance I
E1 program P
partial(I, S); **E1 during E2**

Condition	Description
precedes	$x^+ < y^-$
meets	$x^+ == y^-$
overlaps	$x^- < y^- < x^+, x^+ < y^+$
during	$x^- > y^-, x^+ < y^+$
starts	$x^- == y^-, x^+ < y^+$
finishes	$x^+ == y^+, x^- > y^-$

partial(I,S) ← instanceAssigned(I, S), badTraffic(S).

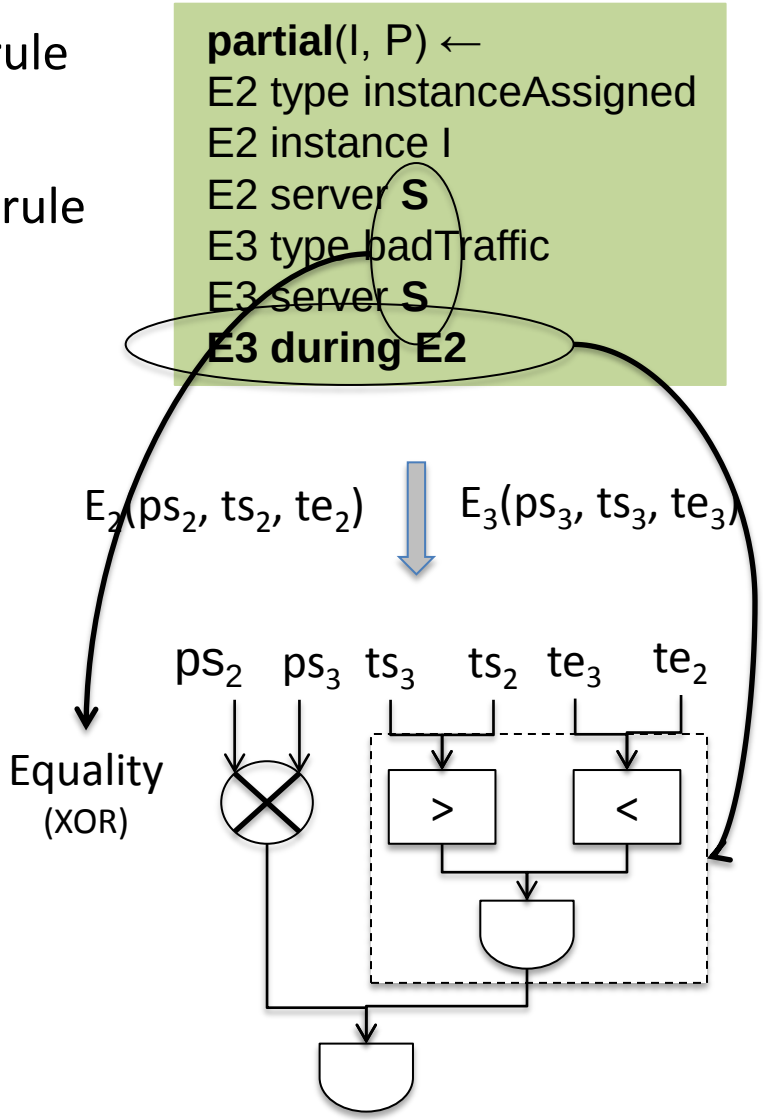
partial(I, P) ←
E2 type instanceAssigned
E2 instance I
E2 server S
E3 type badTraffic
E3 server S; **E3 during E2**

Creating the Circuit

Create a circuit for each resource-based rule

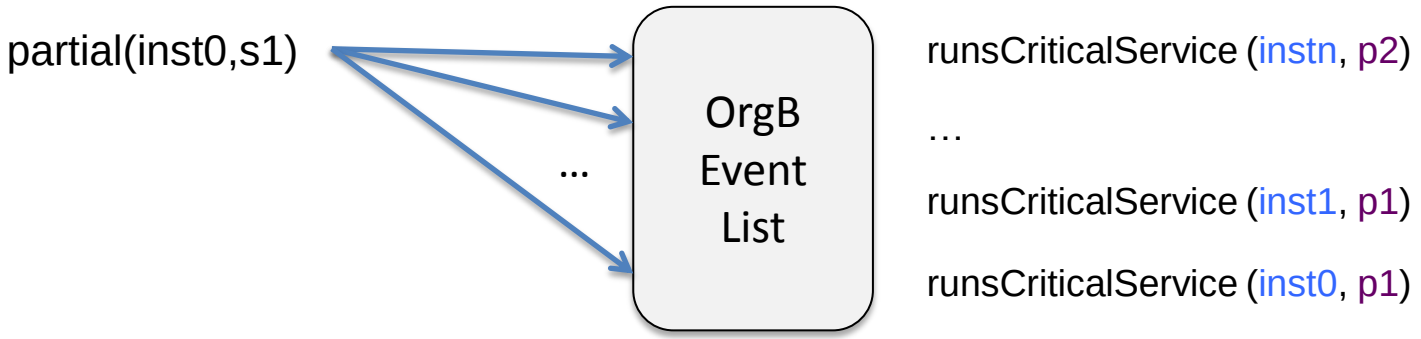
The circuit encodes the conditions in the rule

Condition	Description
equality	E1.s == E2.s
less-than	E1.s < E2.s
precedes	$x^+ < y^-$
meets	$x^+ == y^-$
overlaps	$x^- < y^- < x^+, x^+ < y^+$
during	$x^- > y^-, x^+ < y^+$
starts	$x^- == y^-, x^+ < y^+$
finishes	$x^+ == y^+, x^- > y^-$



Multi-event Matching Protocol

violation(I, P) ← runsCriticalService (I, P), partial(I,S)



partial(inst0, s1), runsCriticalService (instn, p1)	no match	*parallel computation*
…		
partial(inst0, s1), runsCriticalService (inst1, p1)	no match	
partial(inst0, s1), runsCriticalService (inst0, p1)	match found	

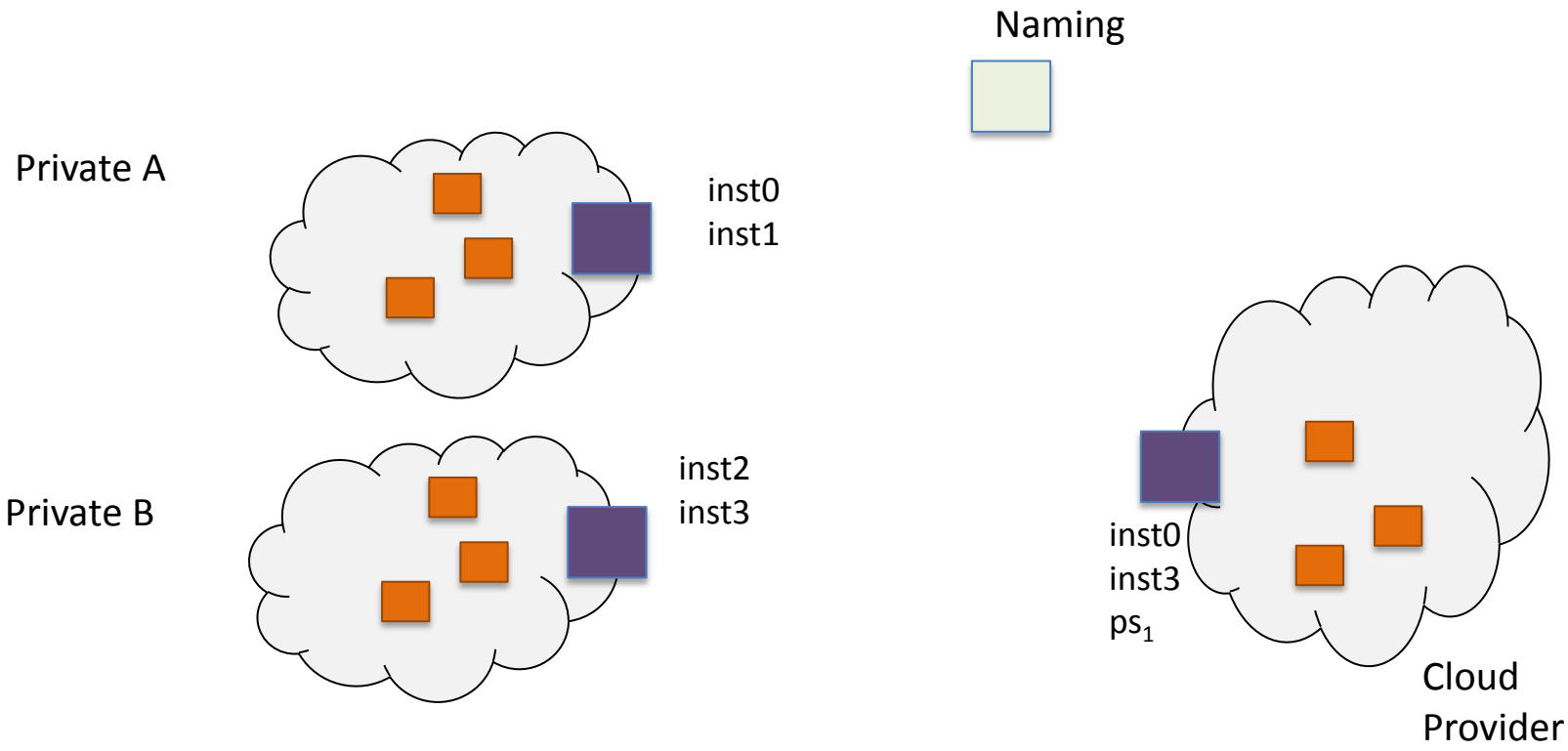
Information is shared only if there is a match of the policy
For two-event policies, this is **the minimal need-to-know**

Distributed Algorithm: Rewrite

violation(I, P) ←
runsCritService (I, P),
instAssigned(I, S)
badTraffic(S)

partial(I,S) ← instanceAssigned(I, S), badTraffic(S).

violation(I, P) ← runsCriticalService (I, P), partial(I,S)

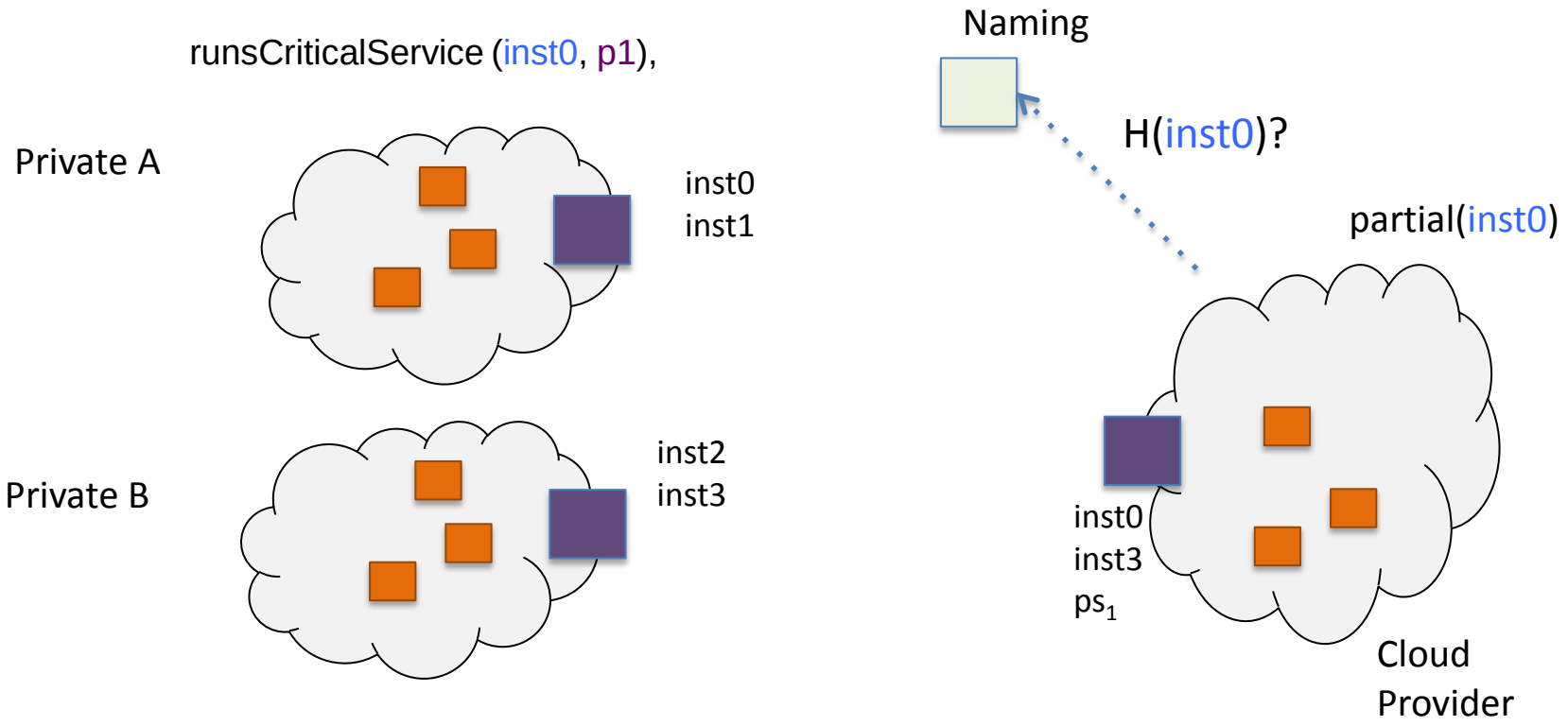


Distributed Algorithm: Naming Resolution

violation(I, P) ←
runsCritService (I, P),
instAssigned(I, S)
badTraffic(S)

partial(I,S) ← instanceAssigned(I, S), badTraffic(S).

violation(I, P) ← runsCriticalService (I, P), partial(I,S)

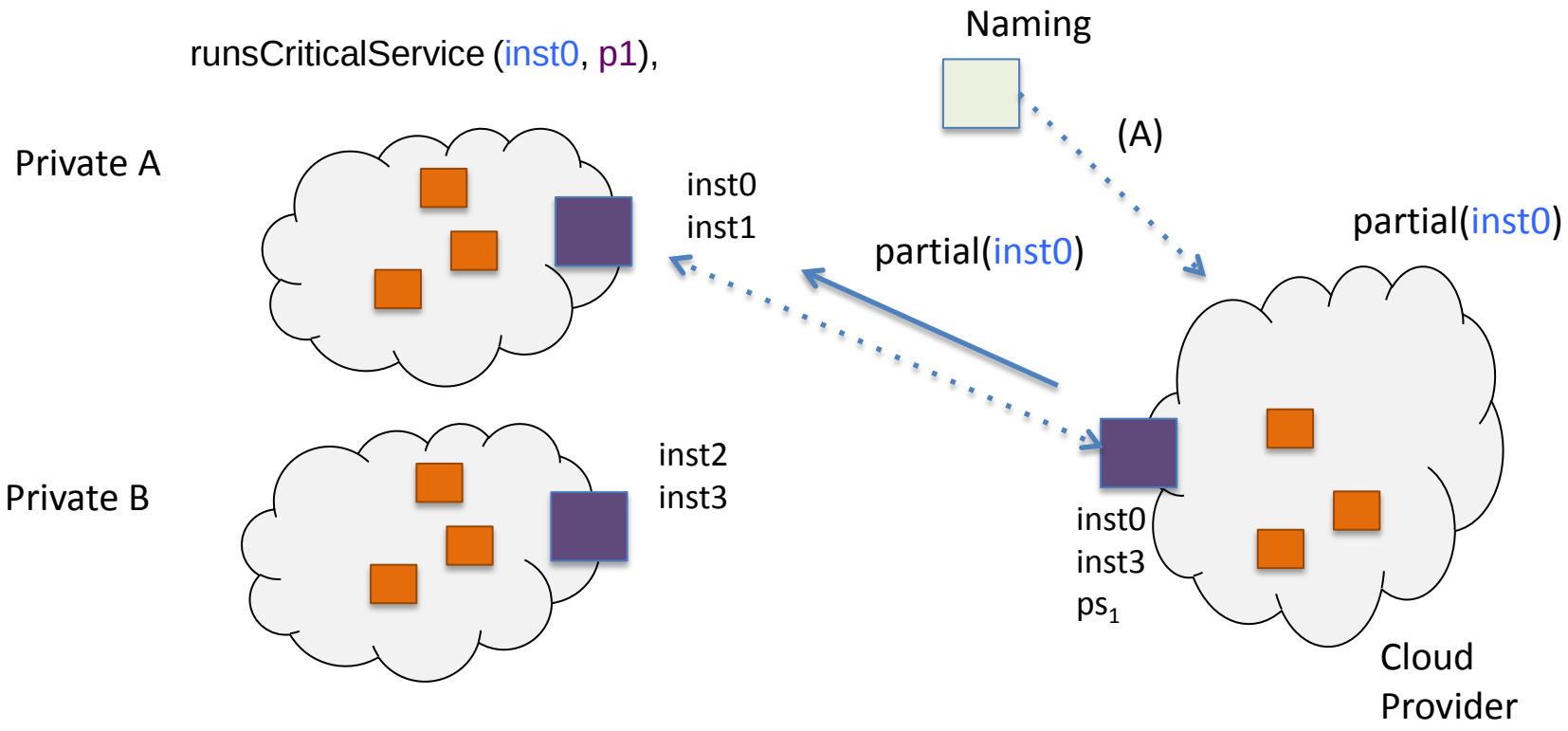


Distributed Algorithm

violation(I, P) ←
runsCritService (I, P),
instAssigned(I, S)
badTraffic(S)

partial(I,S) ← instanceAssigned(I, S), badTraffic(S).

violation(I, P) ← runsCriticalService (I, P), partial(I,S)



Evaluation

Quantitative measures: Shared events; Event throughput

Qualitative evaluation of other information leaks

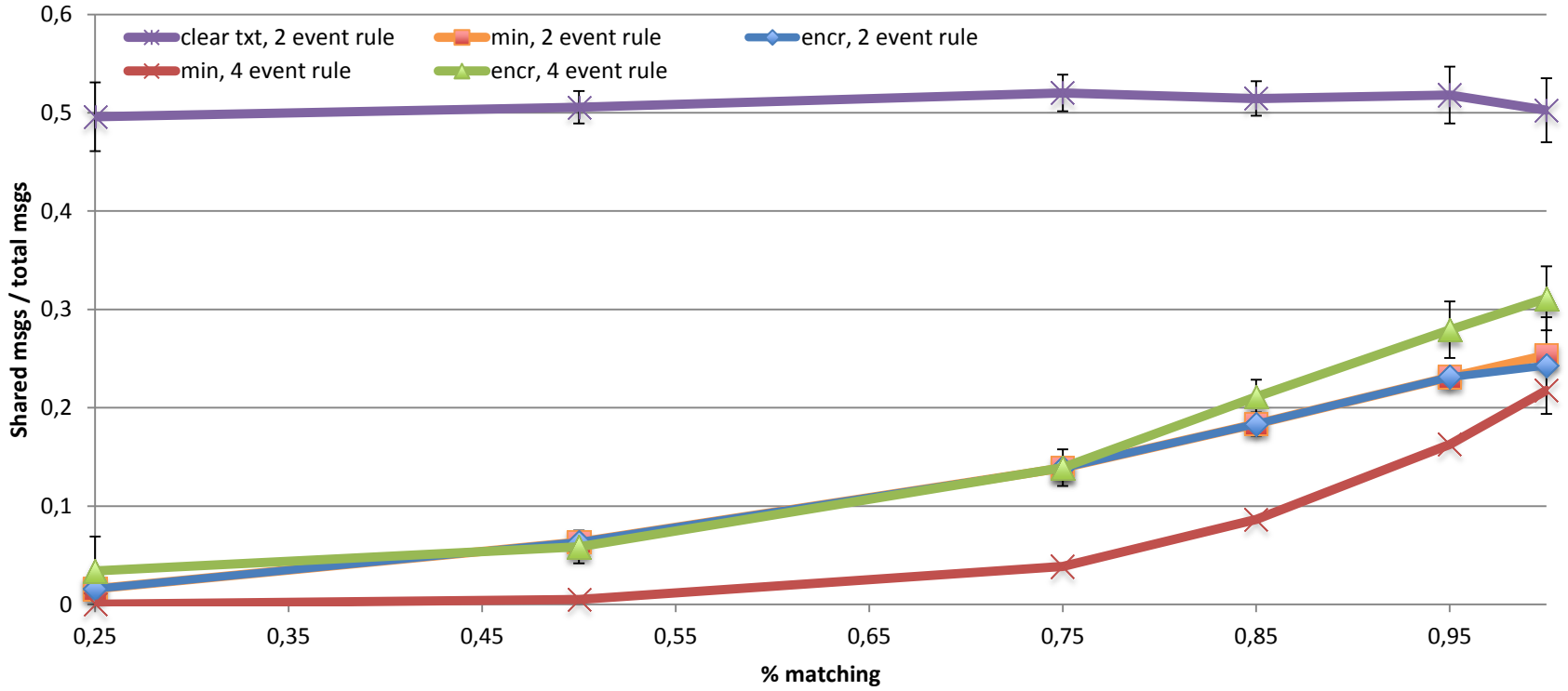
Experimental Setup

- Evaluated on a system running on 2-20 servers
- Parameters of event datasets generated to analyze specific behaviors of the system
  - Evaluation not specific to a single application domain
- Garbled circuit implementation from Huang, Evans, Katz (NDSS 2012)
  - Improvements for parallel computation

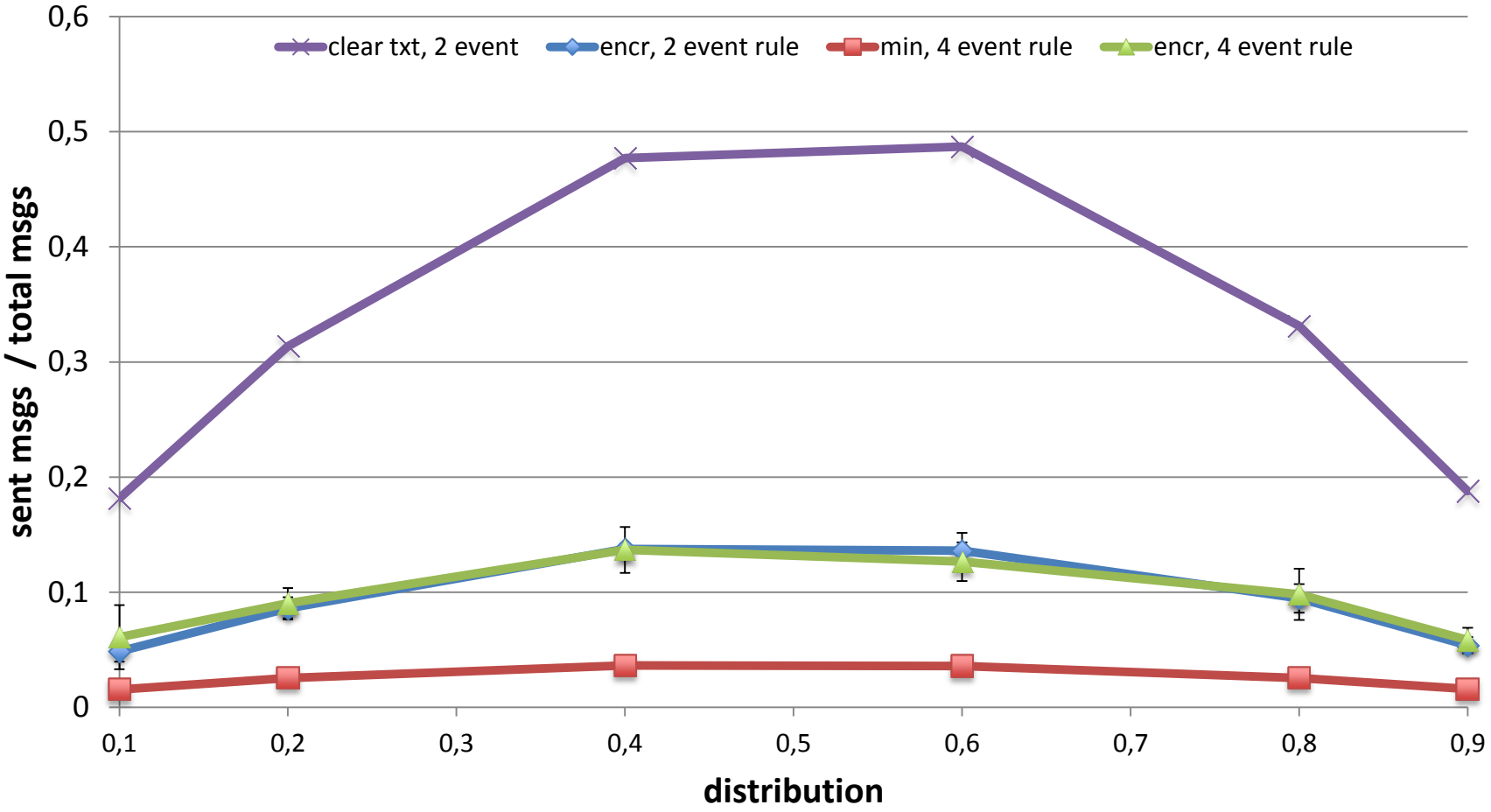
Event Shared

Complex policies

- Approach optimal for 2 event policies, more complex policies require sharing intermediate data



Resource Distribution



Fraction of resources allocated to a monitoring server. 2 servers

Information Leaks

Naming system

- Requests for resolution reveals that an organization has control of a resource
  - Short hash of resources reduces the information leaked
  - Potential of conflicts hides information about specific resources

Requests

- The presence of a request might imply the presence of a local sequence of events matching the policy
  - Add random requests

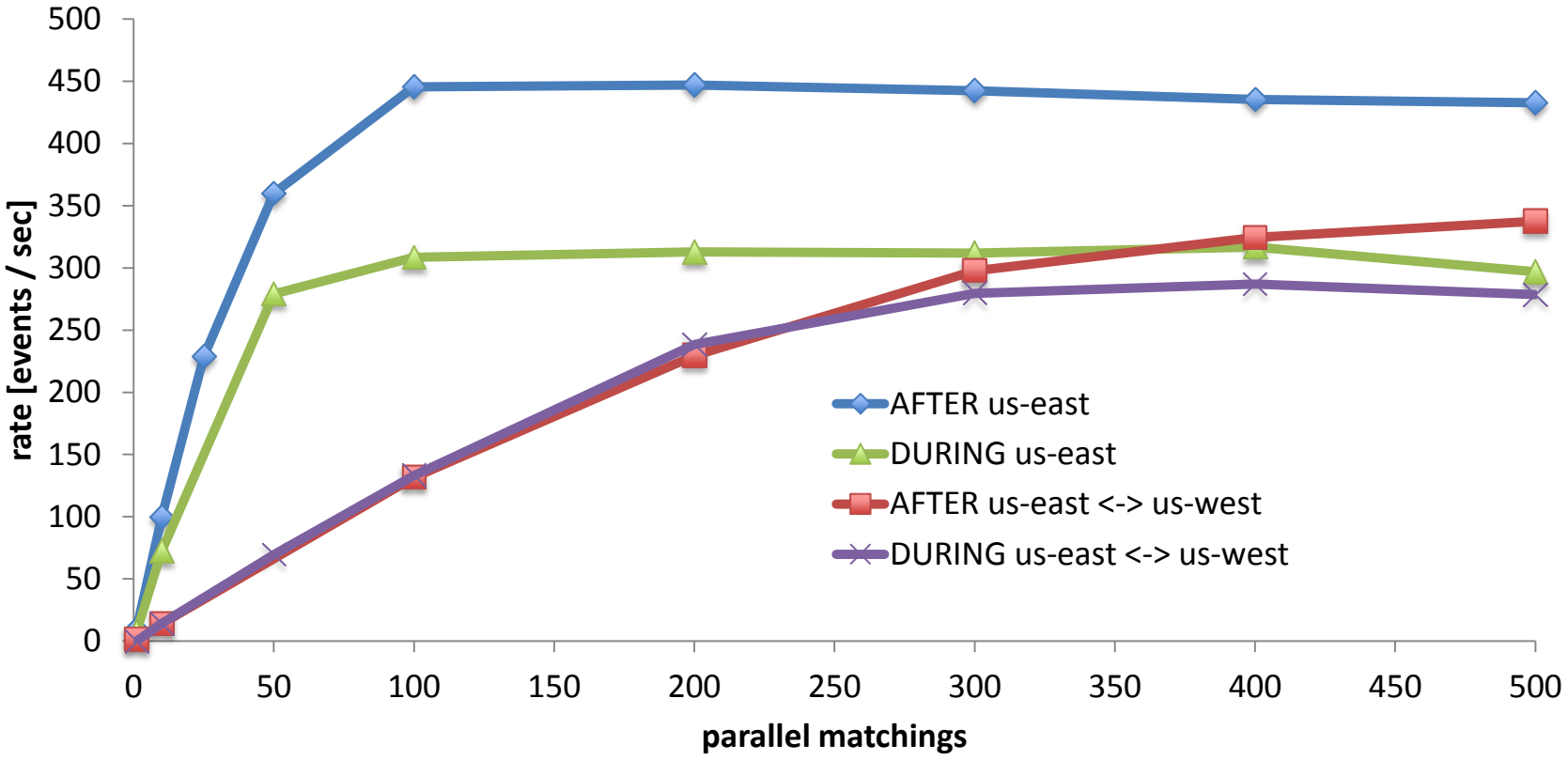
Number of events

- Repeating the process multiple times reveals the number of matching events
  - Add unmatchable events to hide the real event count

Performance Evaluation of GC

Performance: Delay in the processing of an event as a function of the level of concurrency in the server

- Executed within and across geographical regions (us-east, us-west)



Conclusions

- Policy-based approaches are applied widely in industry
- **Goal:** Extend approaches to multi-organization systems

Contributions

- Distributed reasoning algorithm for detecting violations when information is spread across multiple organizations
- Application of secure two-party computation to event correlation to reduce information sharing to minimum need-to-know for simple policies
- Evaluated the approach in multiple conditions
  - Significant reduction of information sharing; acceptable performance for configuration monitoring

Future Work

- Optimize policy-rewrite to reduce sharing in complex policies
- Allow multiple level of confidentiality in information, and reduce sharing of critical data