DS^2: Declarative Secure Distributed Systems

Boon Thau Loo
University of Pennsylvania

Joint work with Wenchao Zhou, Bill Marczak, Micah Sherr, Mengmeng Liu, Matt Blaze, Zack Ives,

External collaborators: Martín Abadi (MSR), Yun Mao (AT&T), LogicBlox Inc.

This work is partially supported by NSF grant IIS-0812270 and CNS-0831376.

Motivation

- **Proliferation of new network architecture and protocols**
  - □ Overlay networks with new capabilities
    - Mobility, resiliency, anycast, multicast, anonymity, etc
  - □ Distributed data management applications
    - Network monitoring, publish-subscribe systems, content-distribution networks
- **Challenges - scalability and security threats**
- **Techniques proposed by security/networking community**
  - □ **Distributed debugging:** PIP [NSDI 06], FRIDAY [NSDI 07]
  - □ **Accountability:** IP traceback [SIGCOMM 00], IP forensics [ICNP 06], AIP [SIGCOMM 08]
  - □ **Distributed trust management:** SD3 [Oakland 01], Delegation Logic [TISSEC 03], Network capabilities [Hotnets'03]

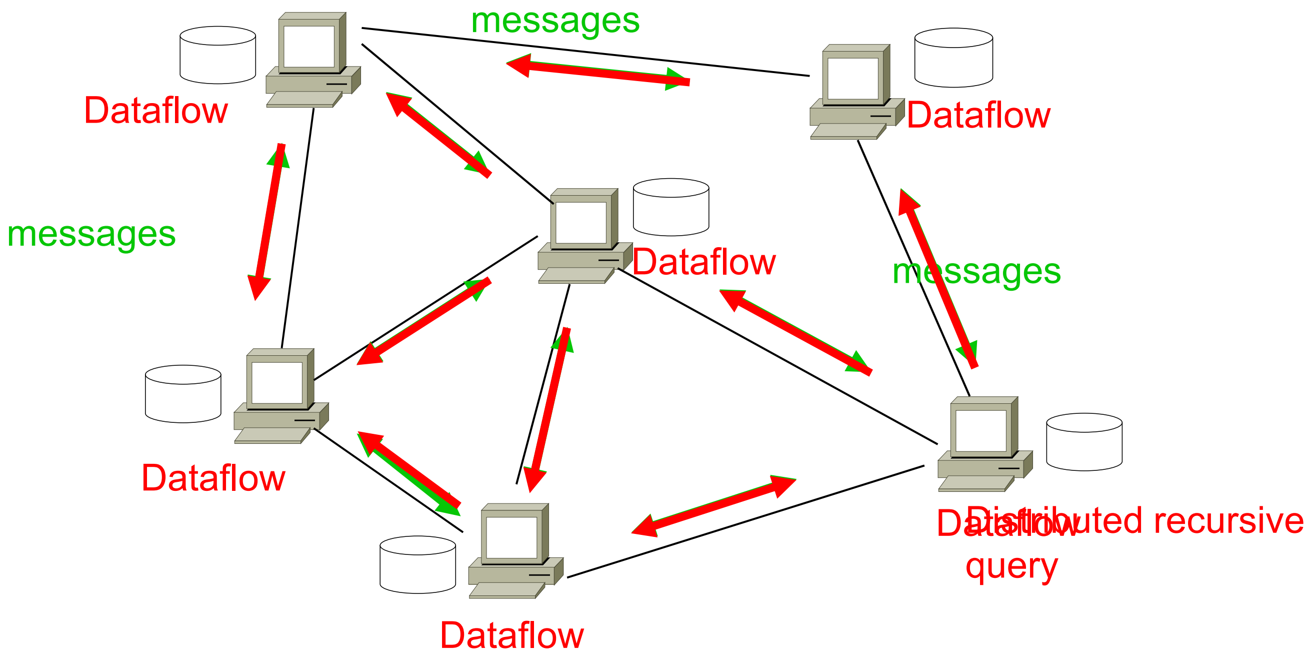
Motivation

- **Problem: lacking generalized framework**
  - Designed for specific security threats
  - Implemented and enforced in different languages and environments
  - Lack of cross-layer integration (networks and higher layers)
- **Overall goal:**
  - Extensible platform for specifying and implementing *distributed systems* and their *security policies*
  - Support for a variety of existing and enable new *analysis techniques*

Outline of Talk

- Background: declarative networking and access control logic
- Unified declarative platform for secure distributed systems [ICDE’09]
- Network provenance [NetDB’08, CCS ’09 submission]
- Reconfigurable trust management [CIDR ’09]
- Other research highlights (http://netdb.cis.upenn.edu/)

Background: Declarative Network



Traditional Networks	Declarative Networks
Network State	Distributed database
Network protocol	Recursive Query Execution
Network messages	Distributed Dataflow

Background: Declarative Networking

- **Declarative query language for network protocols**
  - □ Network Datalog (NDlog) – distributed Datalog [SIGCOMM ‘05, SIGMOD ‘06]
  - □ Compiled to distributed dataflows, executed by distributed query engine
  - □ *Location specifiers* (@ symbol) indicate the source/destination of messages
- **Example: Network Reachability**

r1: reachable(@S,D) :- link(@S,D)

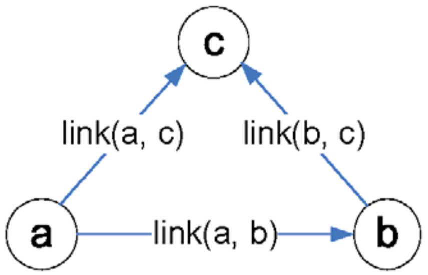
r2: reachable(@S,D) :- link(@S,Z), reachable(@Z,D)

link(@a,b) – “there is a link from node *a* to node *b*”

reachable(@a,b) – “node *a* can reach node *b*”

If there is a link from S to D, then S can reach D.

If there is a link from S to Z, AND Z can reach D, then S can reach D.



Node a	Node b
link(@a, b)	link(@b, c)
link(@a, c)	reachable(@b, c)
reachable(@a, c)	

Path Vector in Network Datalog

R1: path(@S,D,P) ← link(@S,D), P=(S,D).

R2: path(@S,D,P) ← link(@S,Z), path(@Z,D,P_2), $P=S \bullet P_2$.

Query: path(@S,D,P)

Add S to front of P_2

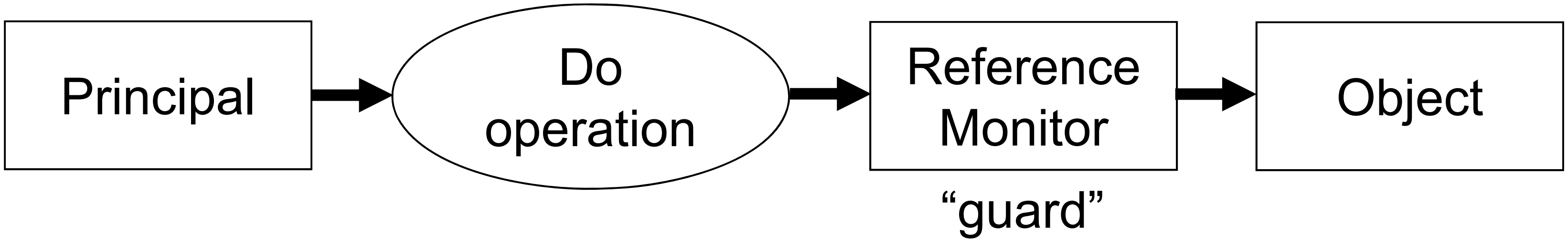
- Input: link(@source, destination)
- Query output: path(@source, destination, pathVector)

Large Library of Declarative Protocols

- Example implementations to date:
  - □ **Routing protocols:** DV, LS, DSR, AODV, OLSR, HSLS, etc.
  - □ **Overlay networks:** Distributed Hash Tables, Resilient overlay network (RON), Internet Indirection Infrastructure (i3), P2P query processing, multicast trees/meshes, etc.
  - □ **Network composition:** Chord over RON, i3+RON
  - □ **Hybrid protocols**: Combining LS and HSLS
  - □ **Others:** sensor networking protocols, replication, snapshot, fault tolerance protocols

Background: Access Control

- Central to security, pervasive in computer systems
- Broadly defined as:
  - Enforce security policies in a multi-user environment
  - Assigning credentials to principals to perform actions
  - Commonly known as ***trust management***
- Model:
  - objects, resources
  - requests for operations on objects
  - sources for requests, called principals
  - a reference monitor to decide on requests



Background: Access Control

- **Access control languages:**
  - *Analyzing* and *implementing* security policies
  - Several runtime systems based on distributed Datalog/Prolog
- **Binder** [Oakland 02]**: a simple representative language**
  - **Context:** each principal has its own context where its rules and data reside
  - **Authentication:** “says” construct (digital signatures)

```
At alice:
    b1: access(P,O,read) :- good(P).
    b2: access(P,O,read) :- bob says access(P,O,read).
```

  - “In alice's context, any principal P may access object O in read mode if P is good (b1) or, bob says P may do so (b2 - delegation)”
- Several languages and systems: Keynote [RFC-2704], SD3 [Oakland 01], Delegation Logic [TISSEC 03], etc.

Comparing the two

- Declarative networking and access control languages are based on logic and Datalog
- Similar observation:
  - Martín Abadi. "*On Access Control, Data Integration, and Their Languages*."
  - Comparing data-integration and trust management languages
- Both extends Datalog in surprisingly similar ways
  - Notion of context (location) to identify components (nodes) in a distributed system
  - Suggests possibility to unify both languages
  - Leverage ideas from database community (e.g. efficient query processing and optimizations) to enforce access control policies
- Differences
  - Top-down vs bottom-up evaluation
  - Trust assumptions

Outline of Talk

- Background: declarative networking and access control logic
- Unified declarative platform for secure distributed systems [ICDE’09]
- Network provenance [NetDB’08, CCS ’09 submission]
- Reconfigurable trust management [CIDR ’09]
- Other research highlights (http://netdb.cis.upenn.edu/)

Secure Network Datalog (SeNDlog)

- Rules within a context
  - □ Untrusted network
  - □ Predicates in rule body in local context
- Authenticated communication
  - □ “says” construct
  - □ *Export predicate:* “X says p@Y”
    - X exports the predicate p to Y.
  - □ *Import predicate:* “X says p”
    - X asserts the predicate p.

```
r1: reachable(@S,D) :- link(@S,D).
r2: reachable(@Z,D) :- link(@S,Z),
                reachable(@Z,D).
```

⇩ *localization rewrite*

```
At S:
  s1: reachable(@S,D) :- link(@S,D).
  s2: linkD(D,S)@D :- link(S,D).
  s3: reachable(Z,D)@Z :- linkD(@S,Z),
              reachable(@S,D).
```

⇩ *authenticated communication*

```
At S:
  s1: reachable(@S,D) :- link(@S,D).
  s2: S says linkD(D,S)@D :- link(S,D).
  s3: S says reachable(Z,D)@Z :-
             Z says linkD(@S,Z),
             W says reachable(@S,D).
```

Example Protocols in SeNDlog

- **Secure network routing**
  - Nodes import/export signed route advertisements from neighbors
  - Advertisements include signed sub-paths (*authenticated provenance*)
  - Building blocks for secure BGP
- **Secure packet forwarding**
- **Customizable anonymous routing**
  - Path selection and setting up “onion paths” with layered encryption
  - Application-aware Anonymity (http://a3.cis.upenn.edu)
- **Secure DHTs**
  - Chord DHT – authenticate the node-join process
  - Signed node identifiers to prevent malicious nodes from joining the DHT
- **P2P query processing – application layer**
  - PIER - built upon Chord DHT
  - Capability of *layered authentication*

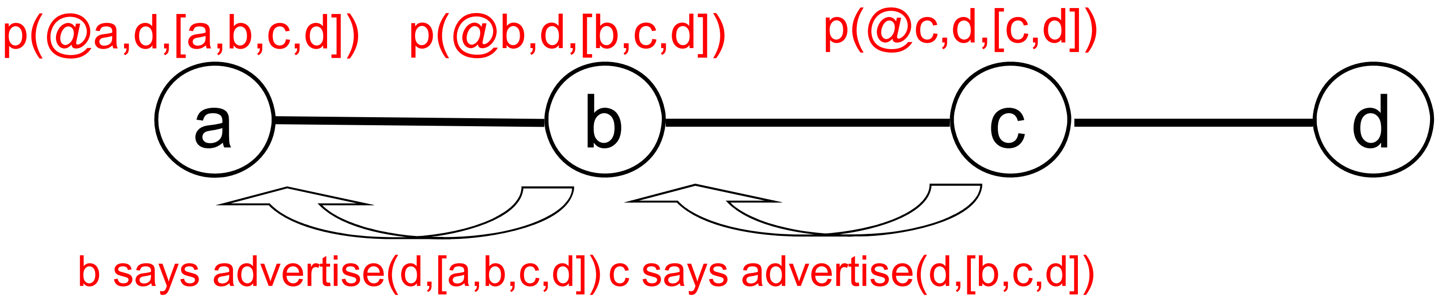
Authenticated Path Vector Protocol

```
At Z,
   z1 route(Z,X,P) :- neighbor(Z,X), P=f_initPath(Z,X).
   z2 route(Z,Y,P) :- X says advertise(Y,P), acceptRoute(Z,X,Y).
   z3 advertise(Y,P1)@X :- neighbor(Z,X), route(Z,Y,P),
                          carryTraffic(Z,X,Y), P1=f_concat(X,P).
```

- Import and export policies
- Basis for Secure BGP
  - Authenticated advertisements
  - Authenticated subpaths (provenance)
  - Encryption (for secrecy) with cryptographic functions

Authenticated Path Vector Protocol

```
At Z,
    z1 route(Z,X,P) :- neighbor(Z,X), P=f_initPath(Z,X).
    z2 route(Z,Y,P) :- X says advertise(Y,P), acceptRoute(Z,X,Y).
    z3 advertise(Y,P1)@X :- neighbor(Z,X), route(Z,Y,P),
                            carryTraffic(Z,X,Y), P1=f_concat(X,P).
```



Authenticated Query Processing

- **Semi-naïve Evaluation**
  - Standard technique for processing recursive queries
  - Synchronous rounds of computation
- **Pipelined Semi-naïve Evaluation** [SIGMOD 06]
  - Asynchronous communication in distributed setting
  - No requirement on expensive synchronous computation
- **Authenticated Semi-naïve Evaluation**
  - Modification for “says” construct, in p’s context:

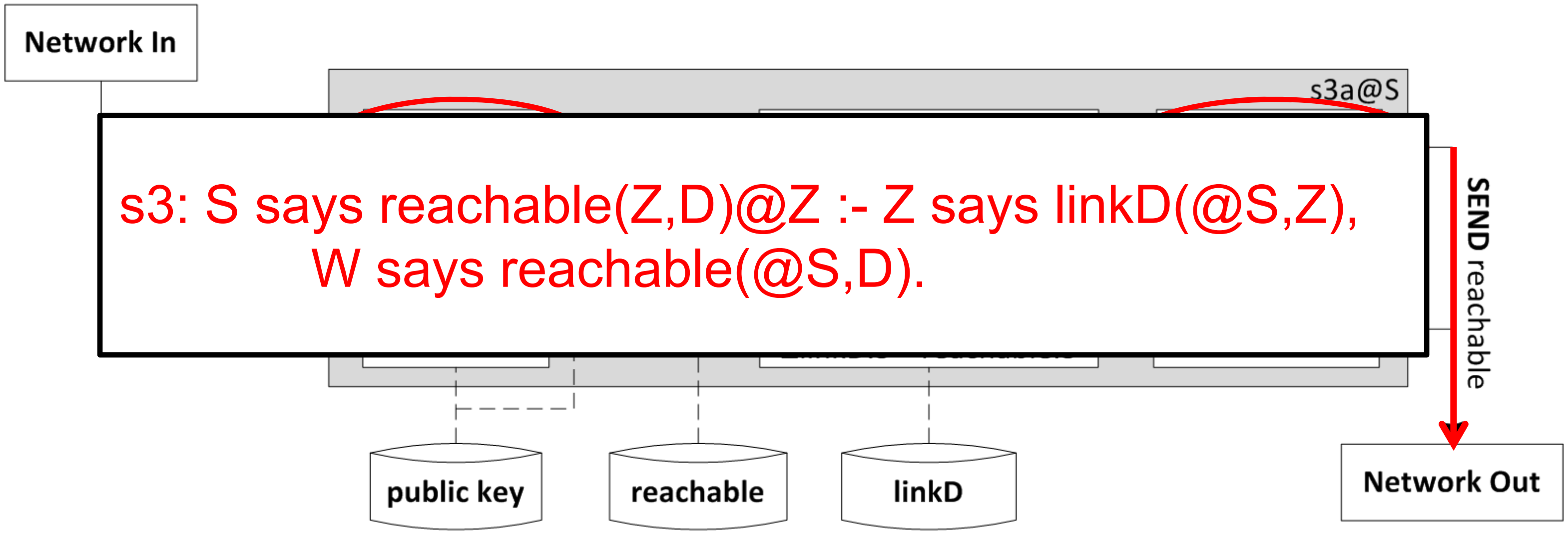
    a :- d_1, ..., d_n, b_1, ..., b_m, p_1 says a_1, ..., p_k says a_k, ..., p_o says a_o.

    for kth *import predicate*, an authenticated delta rules is generated:

    p says Δa :- d_1, ..., d_n, b_1, ..., b_m, p_1 says a_1, ..., p_k says Δa_k, ..., p_o says a_o.

Architectural Overview of Dataflow

- **Dataflow Architecture**
  - Based on the P2 declarative networking system
  - Additional modules to support authenticated communication



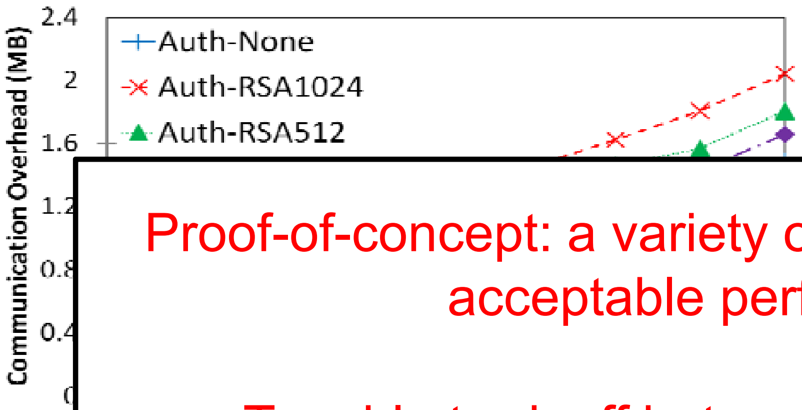
s3: S says reachable(Z,D)@Z :- Z says linkD(@S,Z),
W says reachable(@S,D).

Experimental Setup

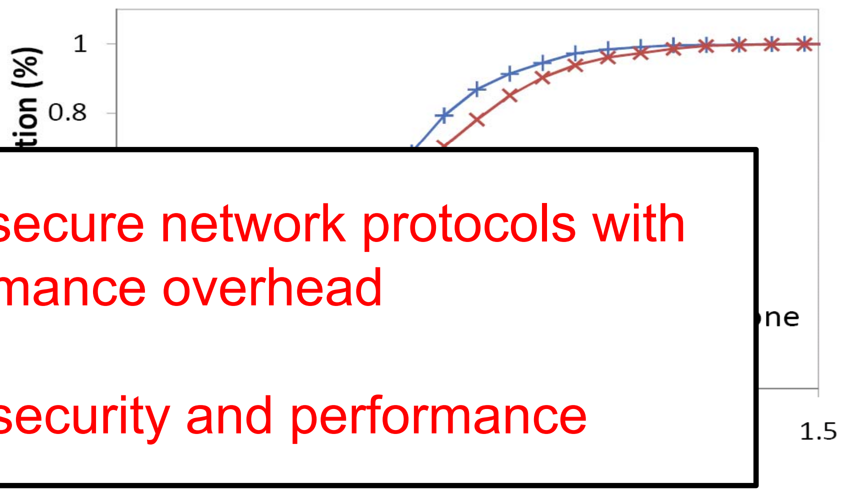
- **P2 declarative networking system**
  - Extensions for security and provenance support
- **Workload**
  - Path-vector – network routing
  - Chord – distributed hash table
  - PIER – p2p query processing
- **Test-bed**
  - A local cluster with 16 quad-core machines
  - Planetlab testbed with 80 nodes
- **Metrics**
  - Communication overhead
  - Query completion time / lookup latency

Authentication Overheads

Path-vector on cluster



Lookup latency for Chord on cluster



Proof-of-concept: a variety of secure network protocols with acceptable performance overhead

Tunable tradeoff between security and performance

- Path-vector protocol
  - 128 nodes, 6 neighbors per node
  - Auth-HMAC – 10% increase
  - Auth-RSA512 – 20% increase
  - Auth-RSA1024 – 40% increase

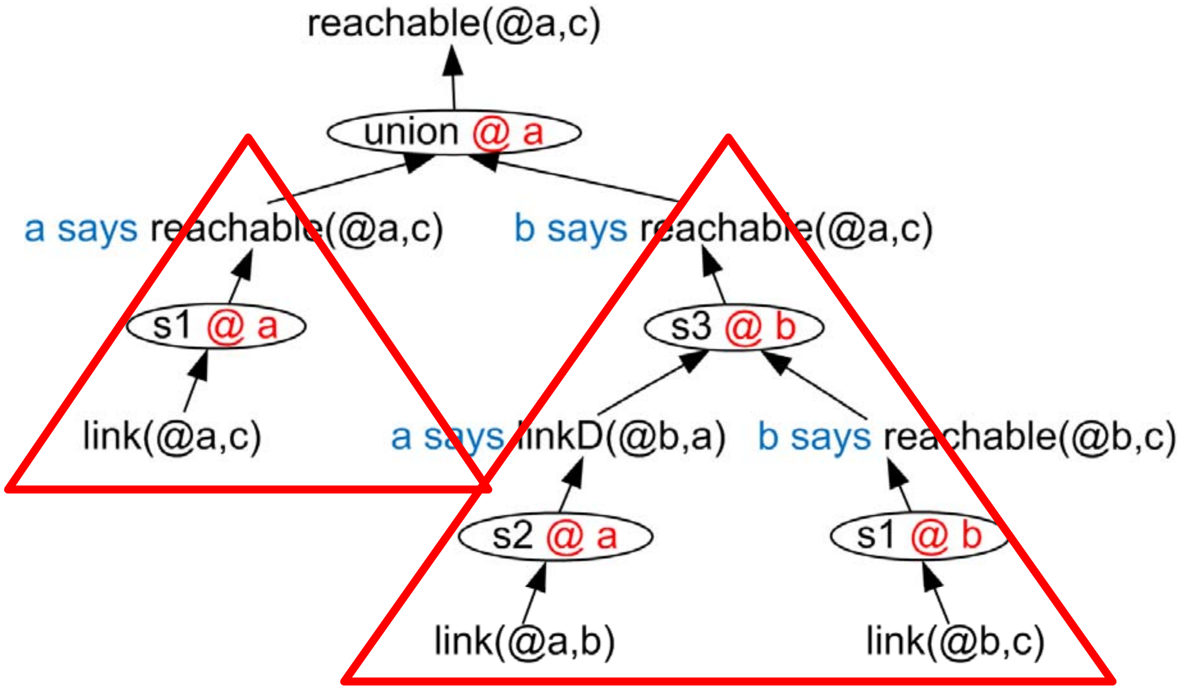
- Chord DHT protocol
  - 128 Chord nodes, random lookups
  - Auth (with RSA1024) – less than 10% increase to finish 50% lookups

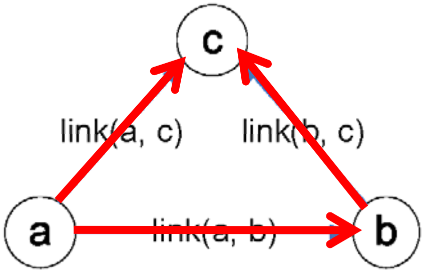
Outline of Talk

- Background: declarative networking and access control logic
- Unified declarative platform for secure distributed systems [ICDE’09]
- Network provenance [NetDB’08, CCS ’09 submission]
- Reconfigurable trust management [CIDR ’09]
- Other research highlights (http://netdb.cis.upenn.edu/)

Network Provenance

- Naturally captured within declarative framework
- Explain the existence of any network state
- Similar notion in security community: *proof-trees*





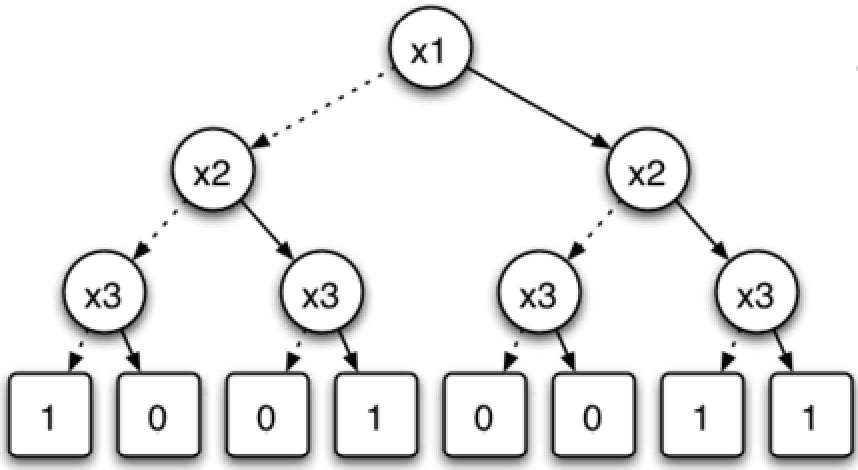
Optimizing network provenance

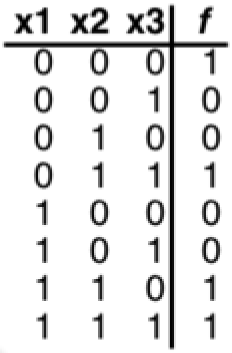
- Two types of provenance: *local* and distributed
- Local provenance is expensive to maintain, relatively cheap to query
  - Tag entire derivation with each tuple
  - Can we make it more bandwidth efficient?
- Distributed provenance is expensive to query, cheap to maintain
  - Ongoing work: query on-demand and caching

- Modularization:
  - Combine common subtrees within a single provenance tree or across trees
- Store a compressed provenance structure
  - Binary decision diagrams (BDDs)
  - Sufficient for certain types of queries
  - Sacrifices some information for compactness

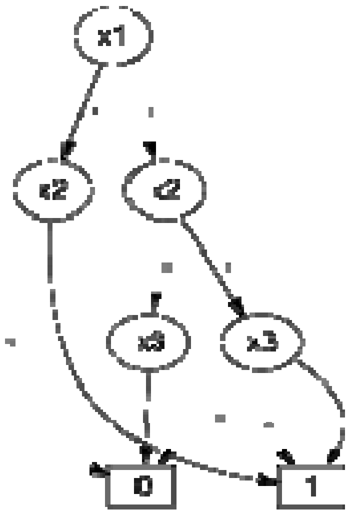
Binary Decision Diagrams

- Binary Decision Diagrams [Bryant 86]
  - Highly optimized libraries available: e.g. JavaBDD.





x1	x2	x3	*f*
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1



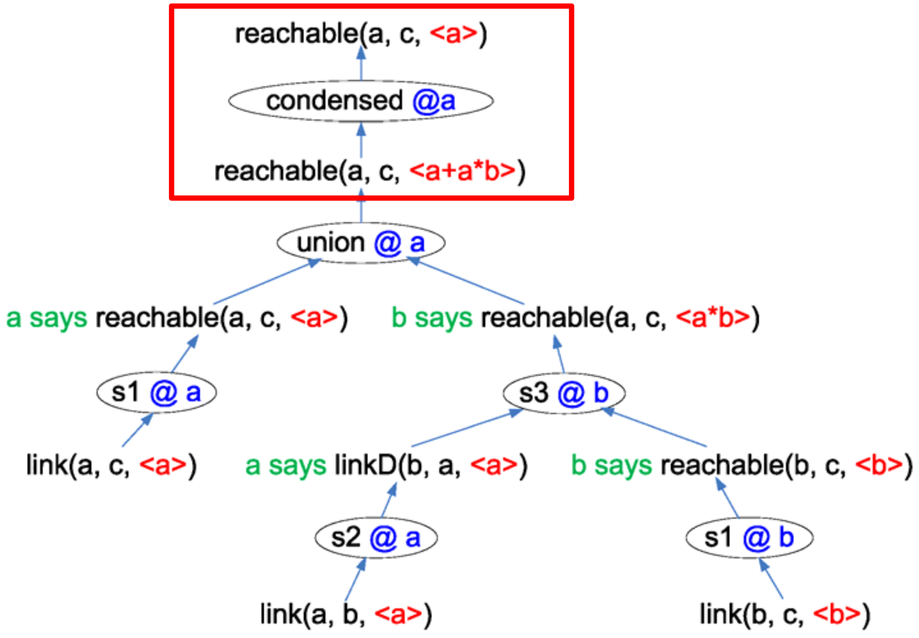
Boolean expression: $\overline{x1}\ \overline{x2}\ \overline{x3} + \overline{x1}\ x2\ x3 + x1\ x2$

Compressed Provenance

- Compress the size of local provenance
  - *Provenance semirings [PODS'07]* annotates provenance in Boolean expressions
  - + means union, * means join
  - BDD encodings for compression

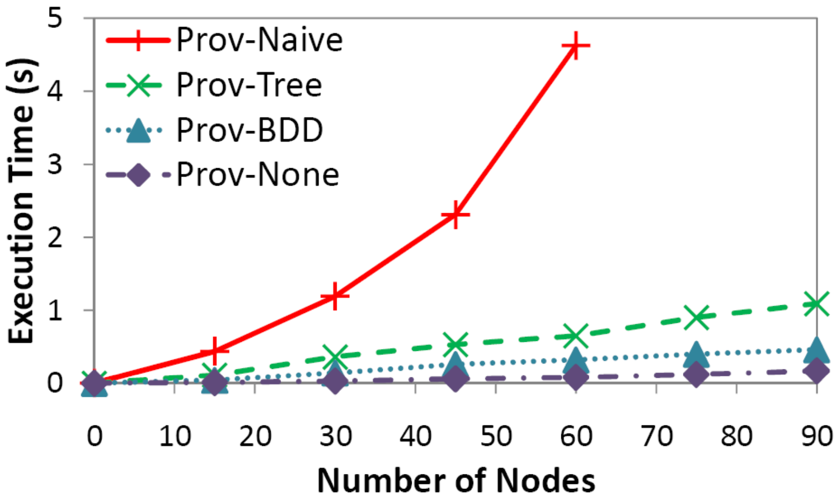
- Compressed:
  - Retain sufficient information for trust management.
  - Node-level provenance
  - Consider <a+a*b>, derivation reachable (a,c) is accepted as long as principal a is trusted
  - Principal b is inconsequential

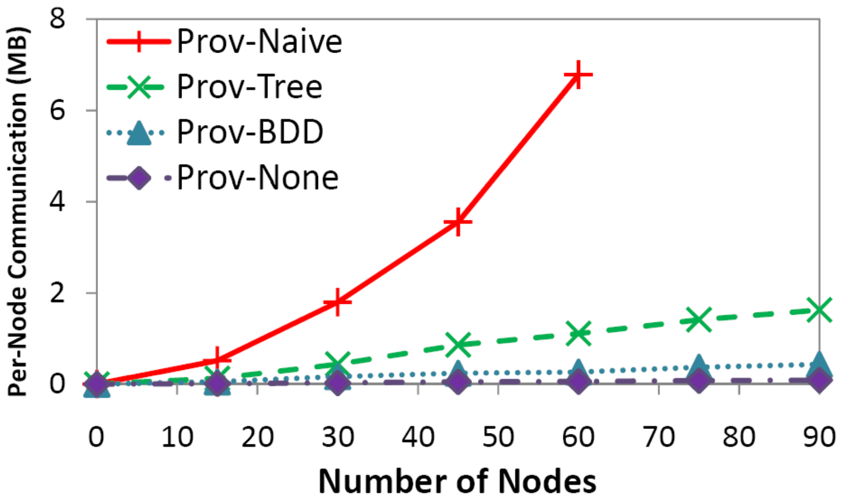
Compression <a+a*b> to <a>



Liu, Taylor, Zhou, Ives, Loo. Recursive Computation of Regions and Connectivity in Networks. [ICDE '09]

Experimental Results





- Computing all-pairs shortest path cost.
- Modularization (Prov-Tree): 90% reduction in execution time over Prov-Naïve
- BDD (Prov-BDD): Additional 60% reduction in execution time

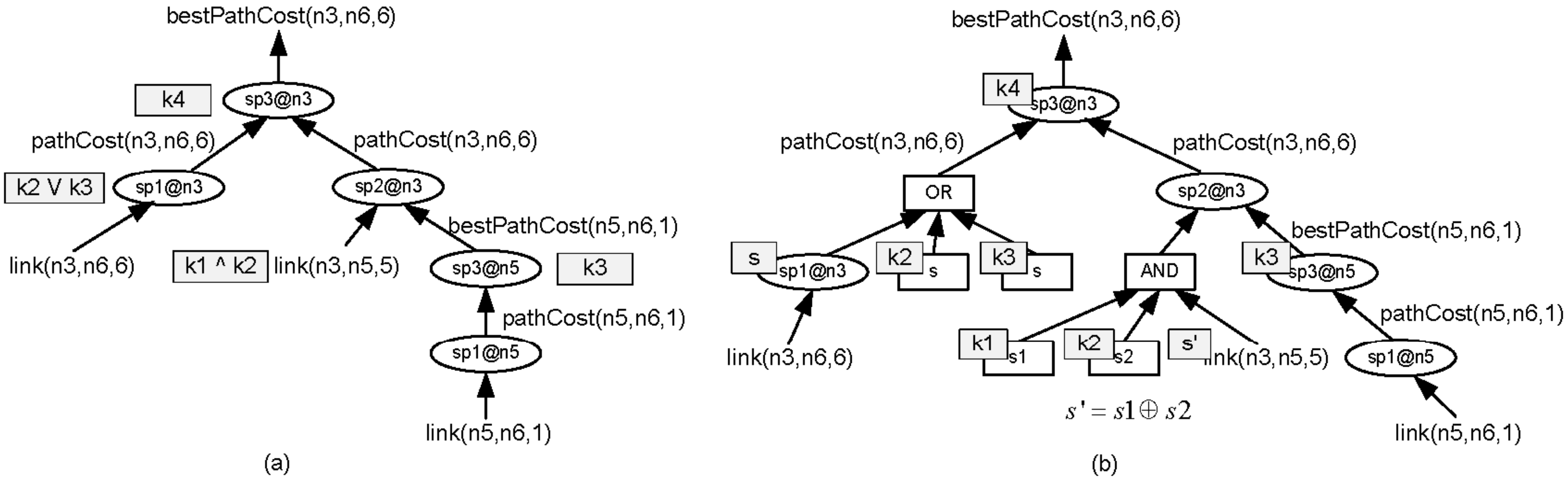
Wide Application of Network Provenance

Provenance Taxonomy	Distributed Debugging	Accountability	Trust Management
Derivation Tree / Algebra Expr.	Tree	Tree	Both
Local / Distributed	Both	Both	Local
Boolean/ Quantifiable	Both	Boolean	Both

- Distributed debugging: PIP [NSDI 06], FRIDAY [NSDI 07]
- Accountability: IP traceback [SIGCOMM 00], AIP [SIGCOMM 08], IP forensics [ICNP 06]
- Distributed trust management: SD3 [Oakland 01], Delegation Logic [TISSEC 03]

Provenance-aware Secure Networks. Zhou, Cronin and Loo. 4th International Workshop on Networking meets Databases (NetDB), 2008

Information hiding in provenance



At Z,

sp1 pathCost(S,D,C) :- link(S,D,C).

sp2 pathCost(D,Z,C1+C2)@D :- link(S,D,C1), bestPathCost(S,Z,C2).

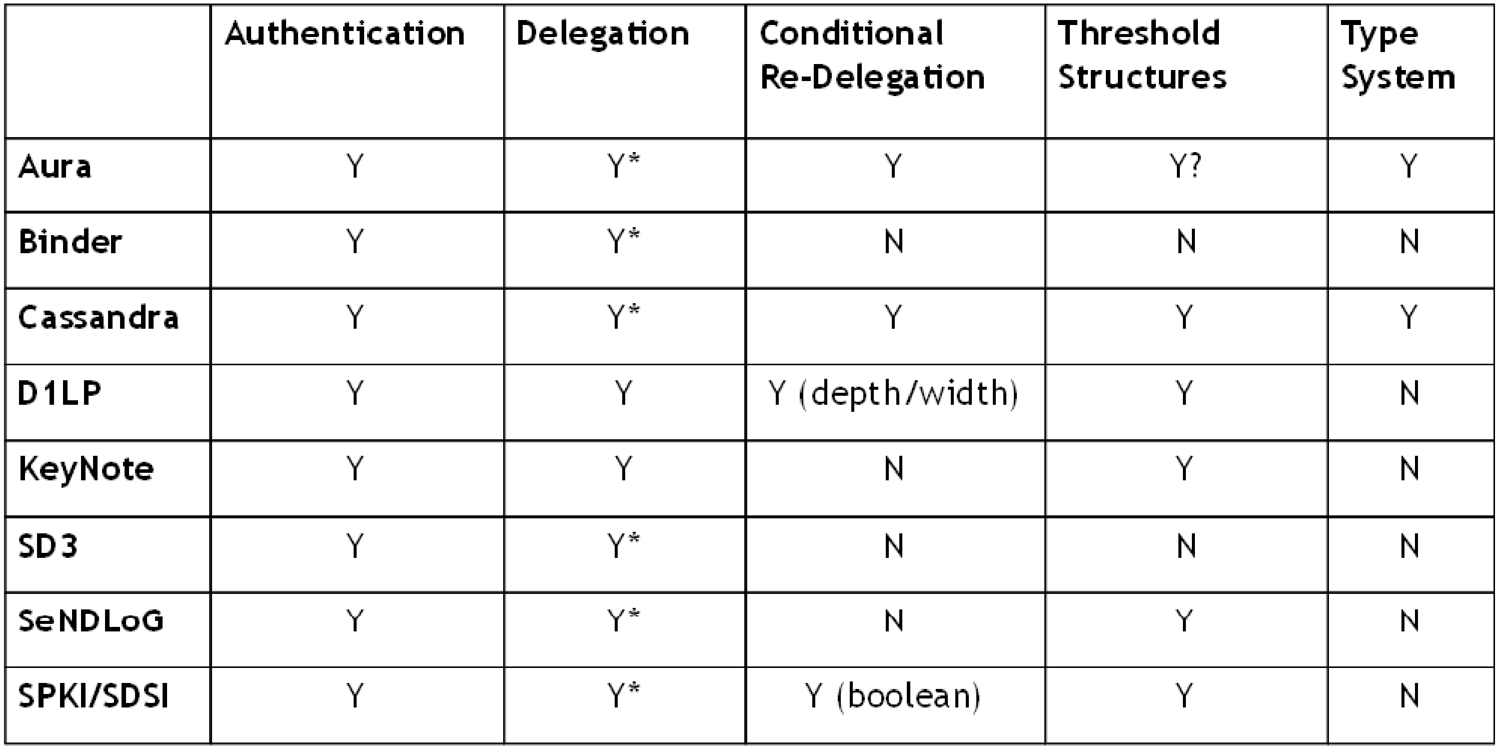
sp3 bestPathCost(S,D,min<C>) :- W says pathCost(S,D,C).

Miklau and Suciu. Controlling access to published data using cryptography. VLDB 03.

Outline of Talk

- Background: declarative networking and access control logic
- Unified declarative platform for secure distributed systems [ICDE'09]
- Network provenance [NetDB'08, CCS '09 submission]
- Reconfigurable trust management [CIDR '09]
- Other research highlights (http://netdb.cis.upenn.edu/)

(Non-Exhaustive) Survey of Trust Management Languages



	Authentication	Delegation	Conditional Re-Delegation	Threshold Structures	Type System
Aura	Y	Y*	Y	Y?	Y
Binder	Y	Y*	N	N	N
Cassandra	Y	Y*	Y	Y	Y
D1LP	Y	Y	Y (depth/width)	Y	N
KeyNote	Y	Y	N	Y	N
SD3	Y	Y*	N	N	N
SeNDLoG	Y	Y*	N	Y	N
SPKI/SDSI	Y	Y*	Y (boolean)	Y	N

- Problem: many languages, features, separate runtime systems, hard to compare and reuse
- Our goal: A unified declarative framework to enable all of these languages

LBTrust: Reconfigurable Trust Management

- Constraints: type safety, program correctness, security
- Meta-programmability.
  - Meta-model: rules as data [VLDB 08]
  - Meta-rules (code generation)
  - Meta-constraints (constraint + reflection)
- Customizable partitioning, distribution, and communication
- Extensible predicates for cryptographic primitives
- Developed using LogicBlox (http://www.logicblox.com), a commercial Datalog engine

Constraints and Types

fail() ← access(P,O,M), !principal(P).

↑

negation

“let fail() whenever access(P,O,M) and not principal(P)”

access(P,O,M) → principal(P).

“whenever access(P,O,M), require principal(P)”

access(P,O,M) → principal(P), object(O), mode(M).

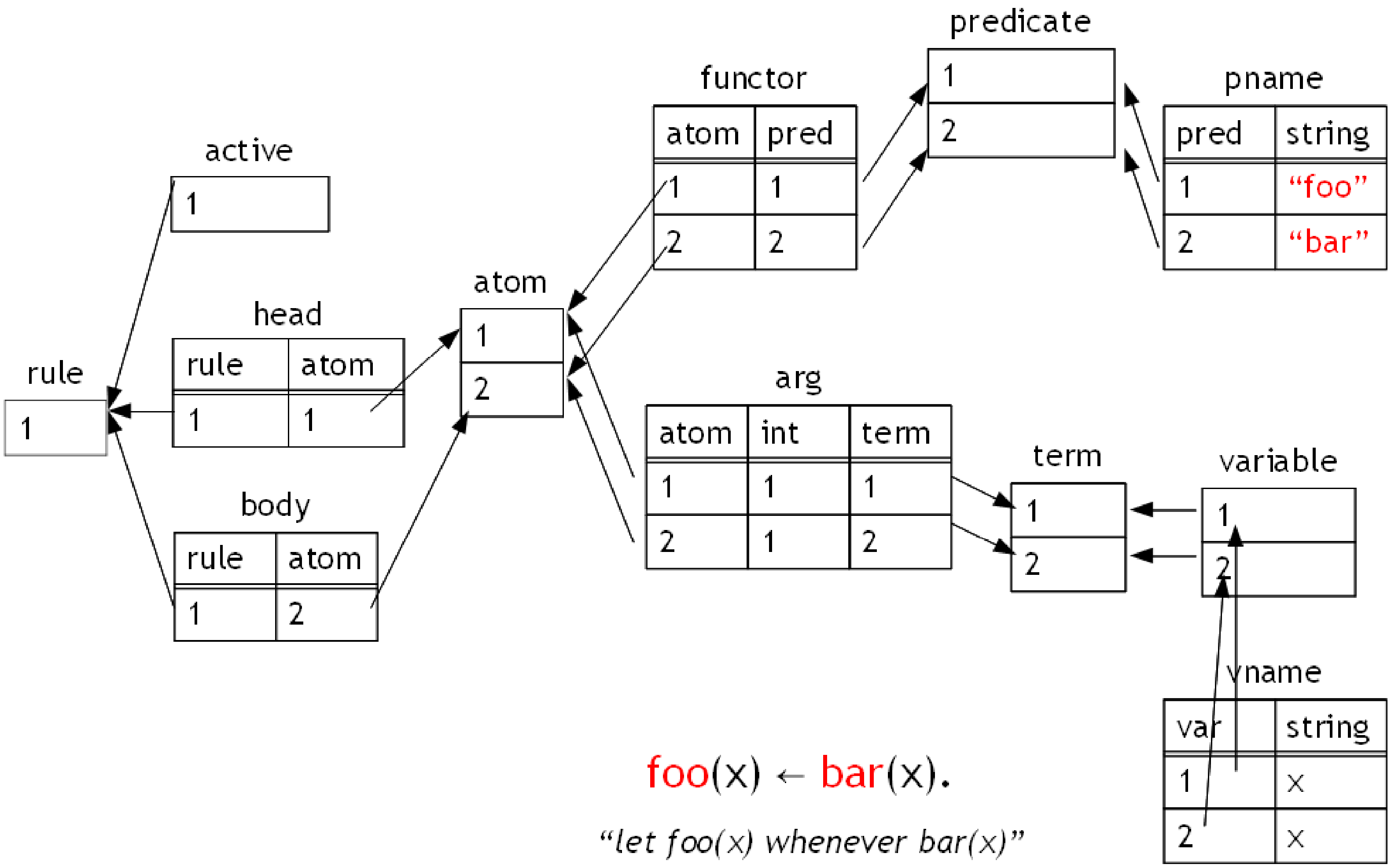
type constraint

Meta-Model Schema

```
rule(R) → .
active(R) → rule(R).
head(R,A) → rule(R), atom(A).
body(R,A) → rule(R), atom(A).

atom(A) → .
functor(A,P) → atom(A), predicate(P).
arg(A,I,T) → atom(A), int(I), term(T).
negated(A) → atom(A).


term(T) → .
variable(X) → term(X).
vname(X.N) → variable(X), string(N).
constant(C) → term(C).
value(C,V) → constant(C), string(V).

predicate(P) → .
pname(P,N) → predicate(P), string(N).
```

ensures rules are well-structured

Rules as Data



rule

1

active

1

head

rule	atom
1	1

body

rule	atom
1	2

atom

1
2

functor

atom	pred
1	1
2	2

predicate

1
2

pname

pred	string
1	“foo”
2	“bar”

arg

atom	int	term
1	1	1
2	1	2

term

1
2

variable

1
2

vname

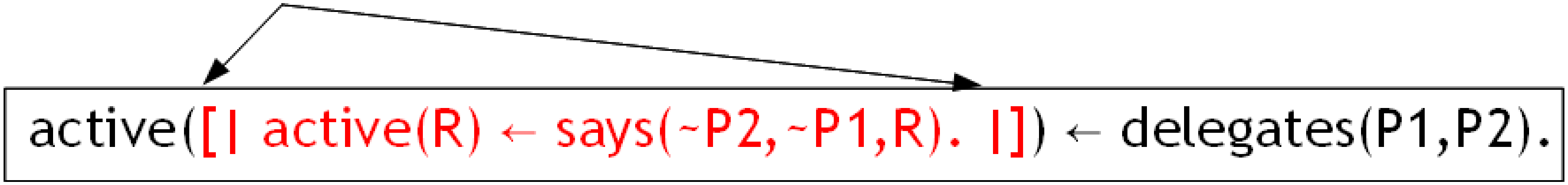
var	string
1	x
2	x

foo(x) ← bar(x).

“let foo(x) whenever bar(x)”

Meta Rules for Security

- Meta
  - □ Code generation (insert new rules that must be evaluated)
  - □ Reflection (query for program structure)
- Meta-Syntax
  - □ Embedded rule/bounded constants (~P2 and ~P1)



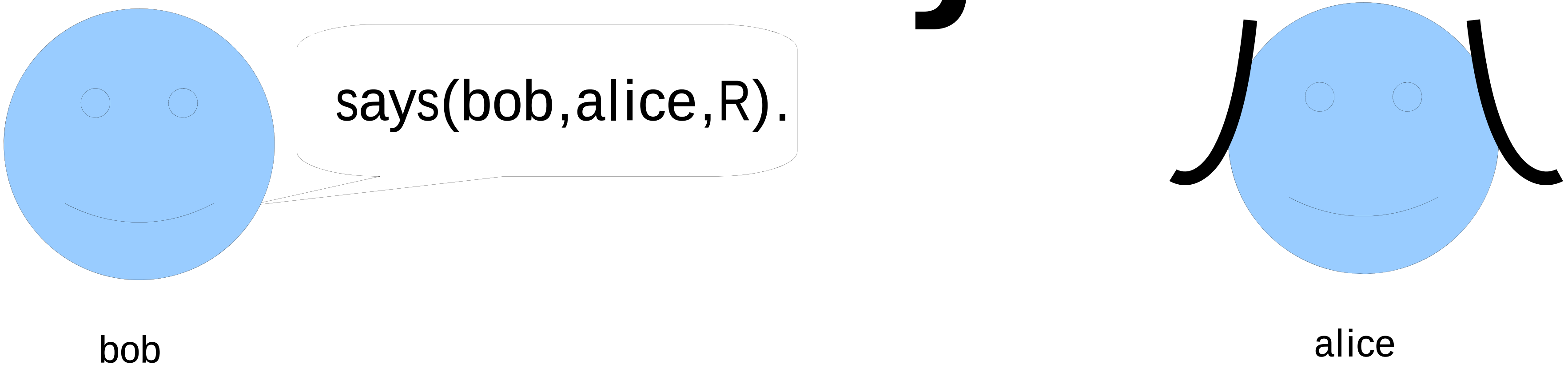
active([| active(R) ← says(~P2,~P1,R). |]) ← delegates(P1,P2).

“activate a rule active(R) ← says(P2,P1,R). for every delegates(P1,P2).”

A Concrete Example: The “Says” Authentication Construct

says(P1,P2,R) → prin(P1), prin(P2), rule(R).
rulesig(R,S) → rule(R), string(S).
rsapubkey(P,K) → prin(P), string(K).
rsaprivkey(P,K) → prin(P), string(K).

schema / type constraints



r1: rulesig(R,S) ←
says(P1,_,R),
rsaprivkey(P1,K),
rsasign(R,S,K).

signature derivation

signature check constraint

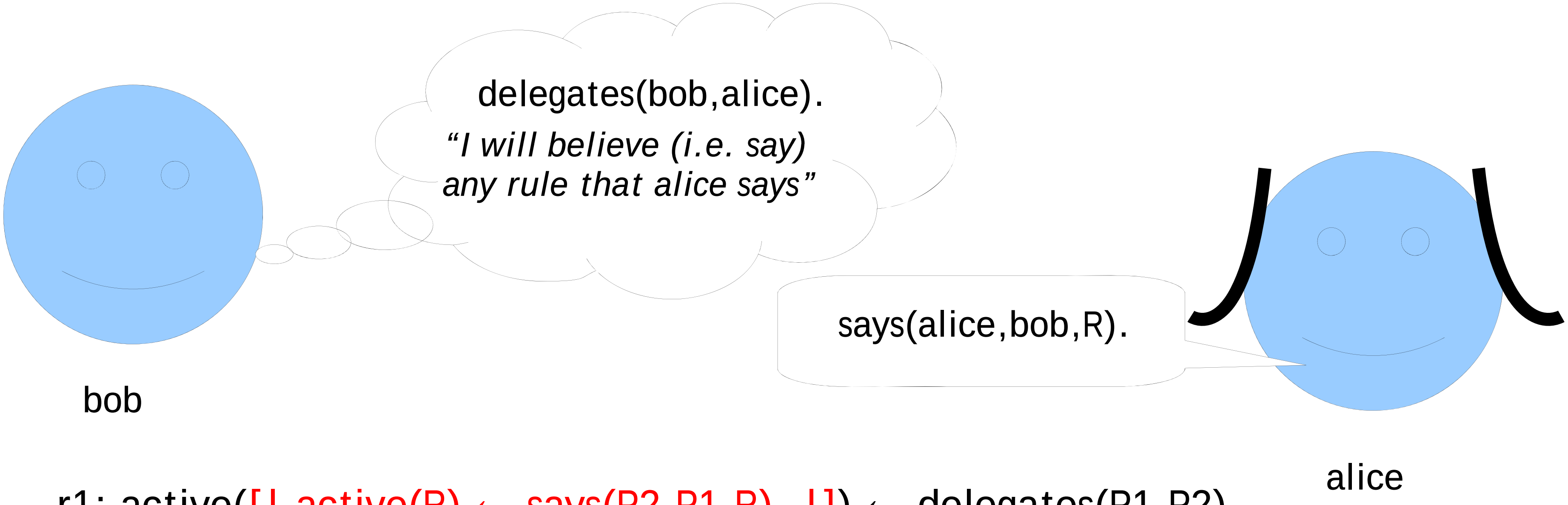
r2: says(P1,_,R),
rsapubkey(P1,K),
rulesig(R,S) →
rsaverify(R,S,K).

Delegation (Basic)

alice *"speaks-for"* bob == "if alice says something, bob says it too."

speaks-for is a special form of delegation:

delegates(P1,P2) → prin(P1), prin(p2).



r1: active([| active(R) ← says(P2,P1,R). |]) ← delegates(P1,P2).

r2: active(R) ← says(alice,bob,R).

Other cool features

Declarative Reconfigurable Trust Management. William R. Marczak, et. al. CIDR 2009.

- Conditional Delegations:
  - Constraint by width, depth, or predicates
  - Detecting delegation violations (use of provenance)
- Customizable distribution/partitioning policies
  - Partition data and rules by principals
  - Distribute principals across machines
- Same security policy rules can run in local/distributed environment
  - Use meta-rules to rewrite top-down access control to execute in a bottom-up evaluation engine
- Example languages:
  - Binder, Delegation logic, D1LP,
  - Secure Network Datalog [ICDE 09],
- Usage: Authenticated routing protocols, access control in distributed databases, distributed file systems

Summary of Contributions

- **Key ideas:**
  - Declarative framework for networks and security specifications
  - Authenticated query processing techniques for distributed settings
  - Network provenance: usage, maintenance and optimizations
  - LBTrust: Distributed reconfigurable trust management

- **Future Work**
  - Optimizing network provenance maintenance and querying
    - Performance / security tradeoff, distributed provenance
  - Applications:
    - Extensible secure routing (http://a3.cis.upenn.edu)
    - Securing cloud data (multi-user across network administrative domains)
  - Verification

Relevant Publications

http://www.cis.upenn.edu/~boonloo/pubs.html

- **Recursive Computation of Regions and Connectivity in Networks.**
  Mengmeng Liu, Nicholas E. Taylor, Wenchao Zhou, Zachary Ives, and Boon Thau Loo.
  25th International Conference on Data Engineering (ICDE), Apr 2009.
- **Unified Declarative Platform for Secure Networked Information Systems.**
  Wenchao Zhou, Yun Mao, Boon Thau Loo, and Martín Abadi.
  25th International Conference on Data Engineering (ICDE), Apr 2009.
- **Declarative Reconfigurable Trust Management.**
  William R. Marczak, David Zook, Wenchao Zhou, Molham Aref, and Boon Thau Loo.
  4th Biennial Conference on Innovative Data Systems Research (CIDR), Jan 2009.
- **Provenance-aware Secure Networks.**
  Wenchao Zhou, Eric Cronin and Boon Thau Loo.
  4th International Workshop on Networking meets Databases (NetDB), Apr 2008.
- **Scalable Link-Based Relay Selection for Anonymous Routing.**
  Micah Sherr, Matt Blaze, and Boon Thau Loo.
  9th Privacy Enhancing Technologies Symposium (PETS), Aug 2009.

Other Research Highlights

- DAWN: Declarative Adaptive Wireless Networks
  - □ In collaboration with BBN Technologies under the DARPA Wireless Networks After Next (WNaN) program
  - □ Deployment on Orbit wireless testbed
  - □ SIGCOMM ‘09 demonstration (Declarative toolkit integrated with ns-3)

- Verifiable networking
  - □ Combining theorem proving, model checking and declarative network verification/synthesis [PADL’09, TPHOL’09, AFM’09]

- Visit http://netdb.cis.upenn.edu for more details! ☺

Thank You …

http://netdb.cis.upenn.edu