Risk Assessment in Distributed Authorization

Peter Chapin, Christian Skalka, X. Sean Wang
University of Vermont

November 11, 2005

Outline

- Trust Management and the RT Framework
- RT^R
- Credential Chain Discovery in RT^R
- Future Work

Trust Management

Authorization in a distributed system must be based on general certified attributes, not just identities.

- Authorizer writes policy describing characteristics of authorized users.

- Requester provides digitally signed credentials certifying requester's attributes.

- Authorizer checks if requester has the correct characteristics; that is, *complies with policy*.

Logically Well-Founded

Many informal trust management systems have been described.

- Their expressiveness and security characteristics are often not well understood until much later (if at all).

- Trust management systems with a formal, logical foundation have provable properties.

- When security is at stake, a system with a clear specification and assurances of correctness is essential.

RT_0*

Credential forms

$$A.r \longleftarrow B \qquad A.r \longleftarrow B.s \qquad A.r \longleftarrow A.s.t$$

$$A.r \longleftarrow B_1.r_1 \cap B_2.r_2 \cap \cdots \cap B_n.r_n$$

- Policies and credentials have the same form.
- Each principal has a local namespace for roles.
- Similar to SDSI extended with intersections.
- Meaning of a role, $\mathcal{S}(A.r)$, is the set of entities that are members of that role.

*Li, Mitchell, Winsborough. *Design of a Role Based Trust Management Framework*, 2002 IEEE Symposium on Security and Privacy

RT_0 Example

A hotel H wishes to offer discounts to its preferred customers and to members of certain organizations.

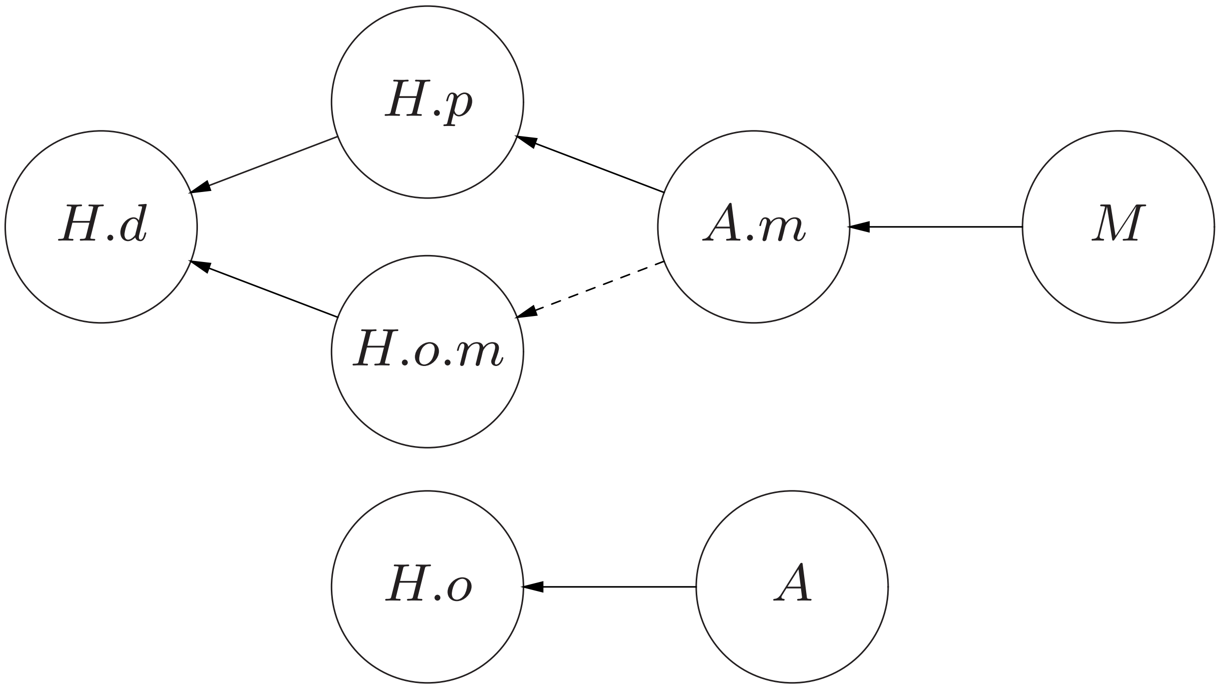
$$H.discount \longleftarrow H.preferred \qquad H.discount \longleftarrow H.orgs.members$$

$$H.orgs \longleftarrow AAA$$

A later marketing decision by H adds $H.preferred \longleftarrow AAA.members$.

Mary has credential $AAA.members \longleftarrow M$. This proves compliance with policy two different ways.

Example Credential Graph



$H.d \longleftarrow H.p \qquad H.d \longleftarrow H.o.m \qquad H.p \longleftarrow A.m \qquad H.o \longleftarrow A$

$A.m \longleftarrow M$

Problem

Not all credentials are created equal.

- Some might be signed by questionable keys.
- Some might be near expiration.
- Some might be assumed to exist, but not actually be in hand.

Existing trust management systems regard credentials as either completely valid or completely invalid. *This is not realistic.*

Introducing Risk

Assigning risks to credentials gives a way to express uncertainties about the credentials.

- Credentials signed by marginal authorities have high risk.

- Risk of a credential might increase as its expiration time approaches.

- Credentials that are presumed to exist have high risk.

- Credentials that are part of local policy have very low risk.

RT^R

RT^R extends RT_0 by assigned risk values to credentials.

- Let $(\mathcal{K}, \preccurlyeq)$ be a complete lattice over some set $\mathcal{K}$ of risk values with partial ordering $\preccurlyeq$.

- Credentials now $A.r \xleftarrow{\kappa} f$, $\kappa \in \mathcal{K}$

- Let $\oplus$ be an associative, commutative, monotonic *risk aggregration operator* over $\mathcal{K}$.

- Meaning of a role is now a set of risk associations called a *risk assessment*. $\mathcal{S}(A.r) = \{(B, \kappa_1), (B, \kappa_2), (C, \kappa_1)\}$

Canonical Risk Assessments

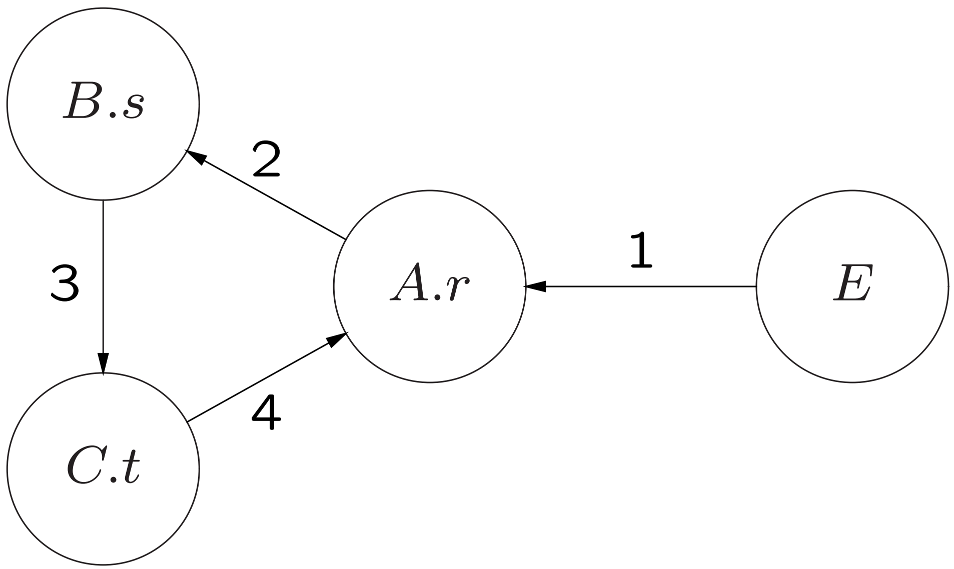
- Equivalence of risk assessements: $R \cup \{(A,\kappa_1),(A,\kappa_2)\} = R \cup \{(A,\kappa_1)\}$ where $\kappa_1 \preccurlyeq \kappa_2$.

- A risk assessment R is *canonical* if there is no $(A,\kappa_1),(A,\kappa_2) \in R$ such that $\kappa_1 \preccurlyeq \kappa_2$.

- Thus any equivalence class of risk assessments has a unique canonical form. *Use this canonical form to represent the meaning of a role.*

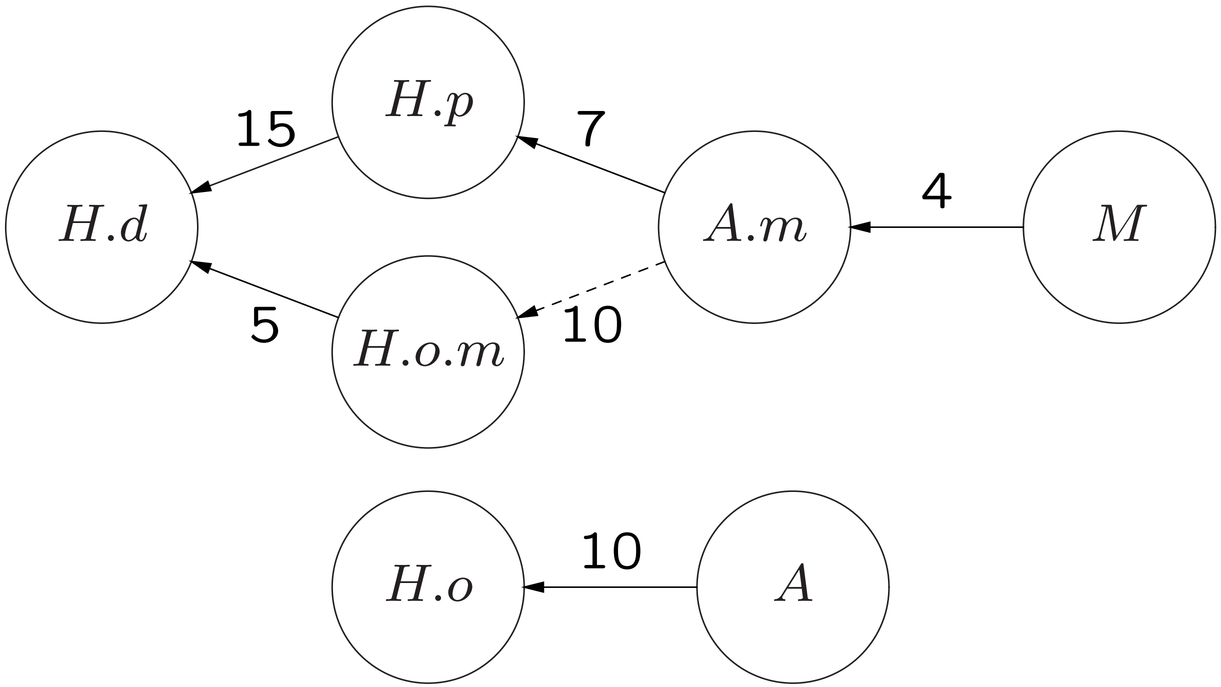
Credential Graph Cycles

Canonical risk assessments are finite even with cycles in the credential graph.



$$\mathcal{S}(A.r) = \{(E, 1), (E, 10)\} = \{(E, 1)\}$$

Example Revisited



$\mathcal{S}(H.d) = \{(M, 19)\}$

Bounded Proof Search

Given a collection of credentials find a *credential chain* that proves some entity E is in a particular role $A.r$ with a bounded risk.

Abort search in directions where risk is too high.

- Reduces searching and speeds up the authorization decision.
- In a distributed search, one may be able to avoid fetching credentials that are not useful.
- If risks represent wait times, the search finds a credential chain where no certificate takes longer than a given bound to verify.

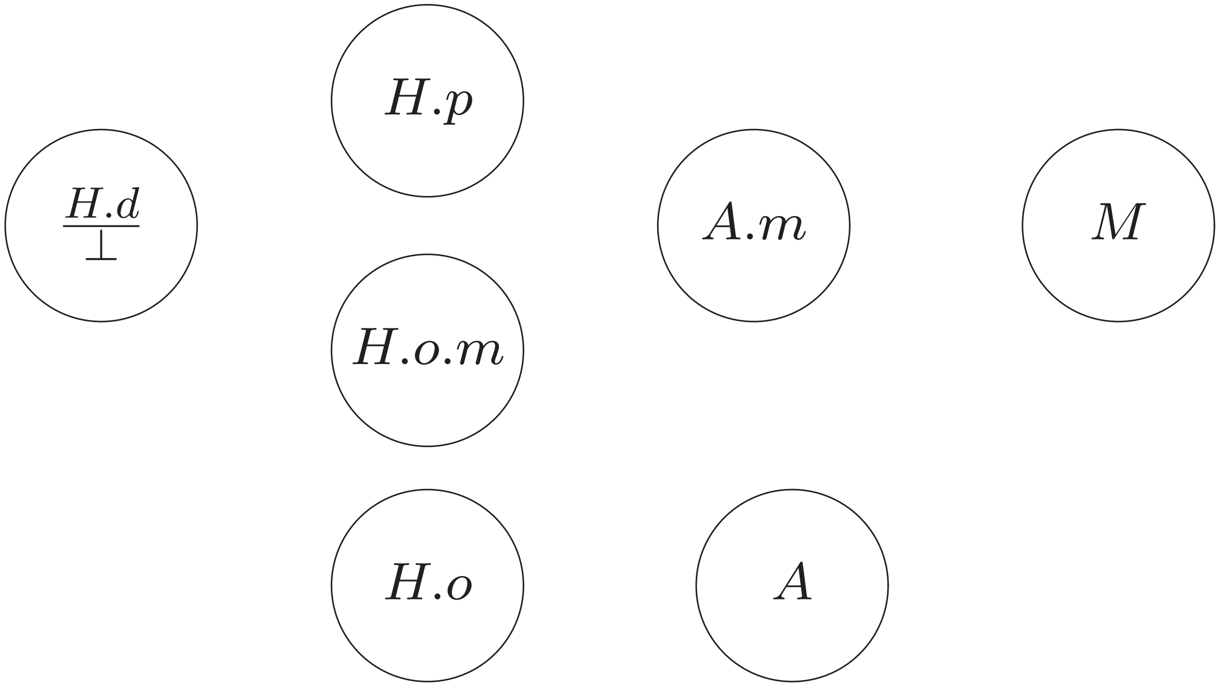
Search Algorithm

Algorithm is a modification of that in [Li et. al.]*

- Modified breadth-first-search of credential graph.
- Starts at role $A.r$ and works toward the entities.
- Graph mutates as search progresses (derived edges added).
- Accumulated risks tracked; search abandoned where risks excessive.

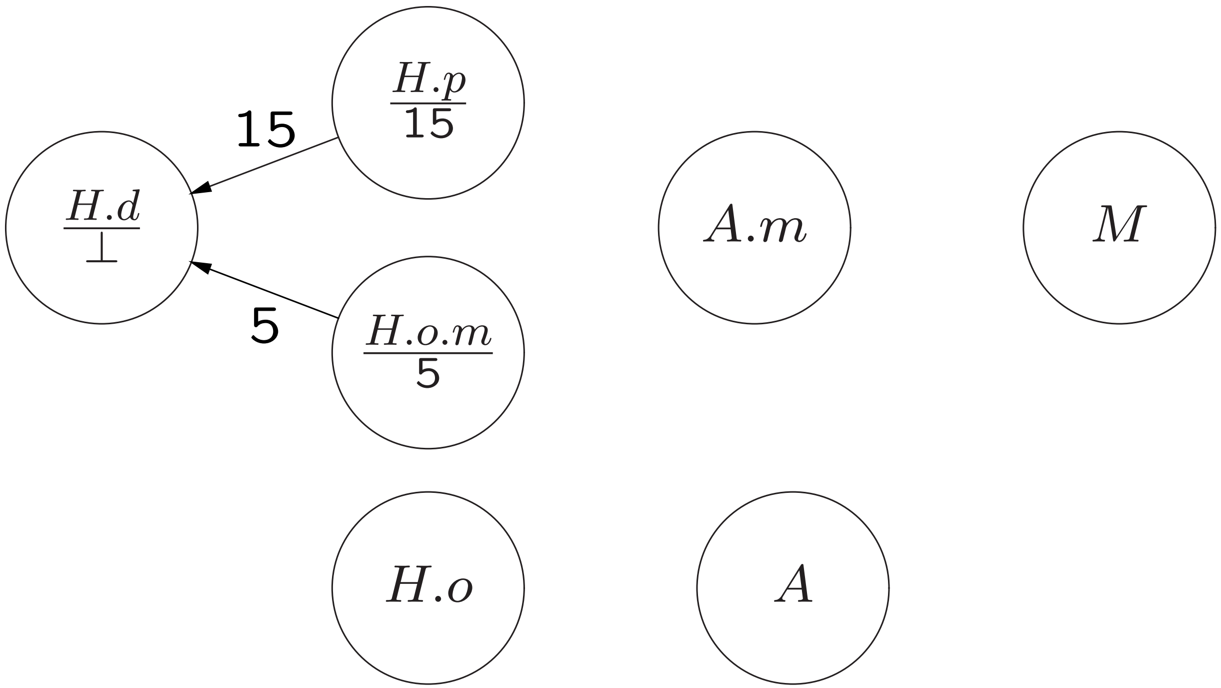
*Li, Winsborough, Mitchell, *Distributed Chain Discovery in Trust Management*, Journal of Computer Security, February 2003

Search Algorithm Example: 1



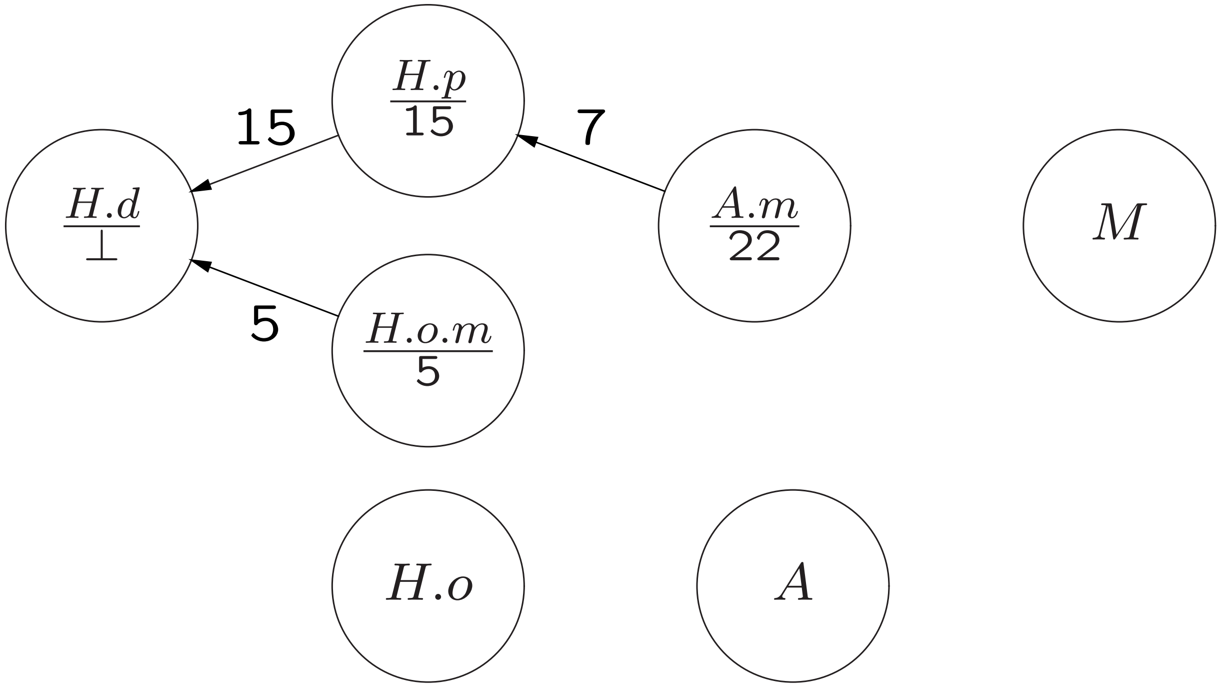
$\kappa_M = 20$

Search Algorithm Example: 2



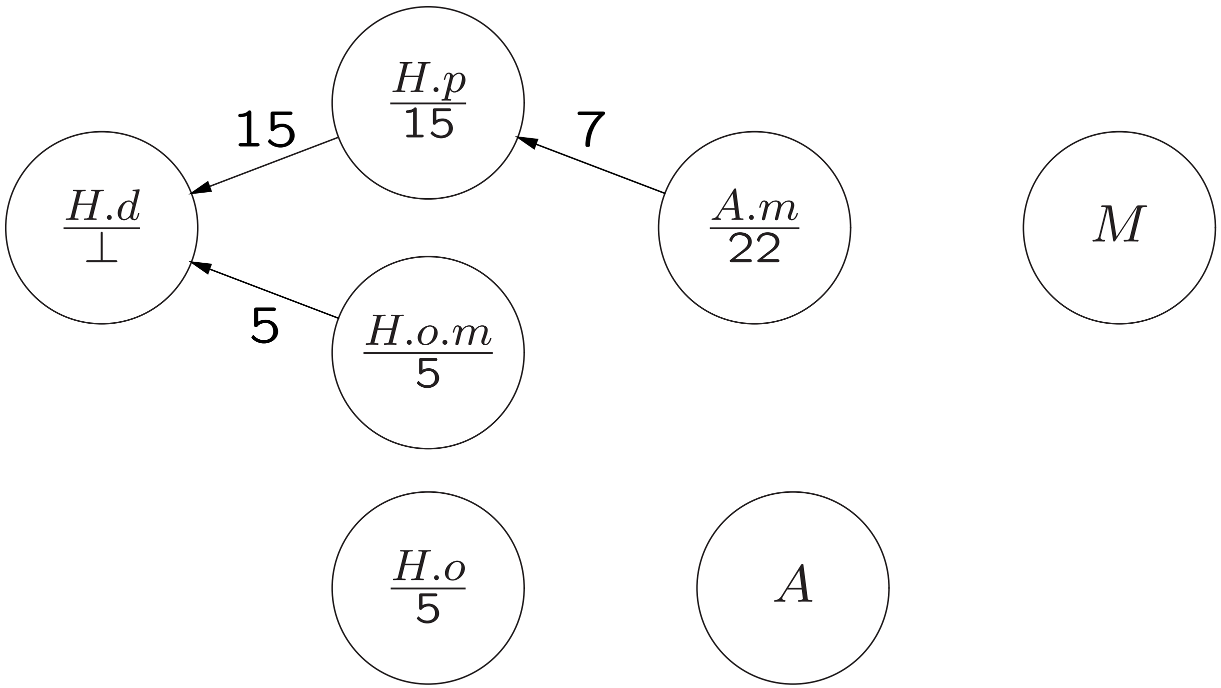
$\kappa_M = 20$

Search Algorithm Example: 3



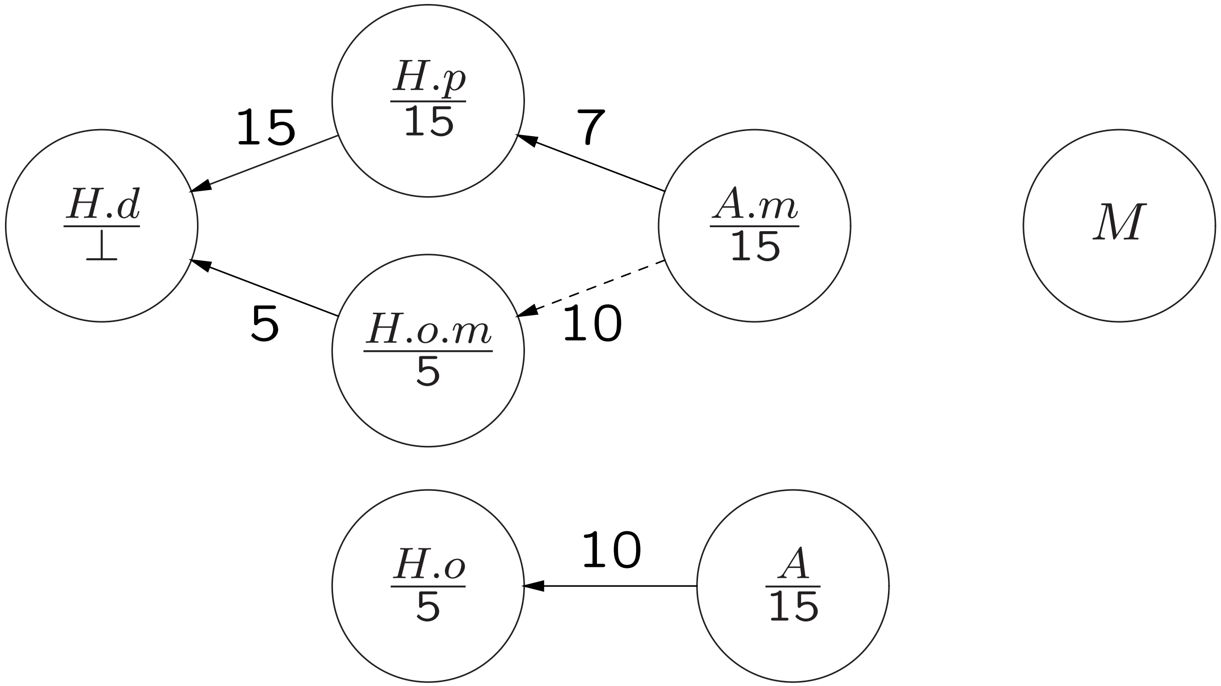
$\kappa_M = 20$

Search Algorithm Example: 4



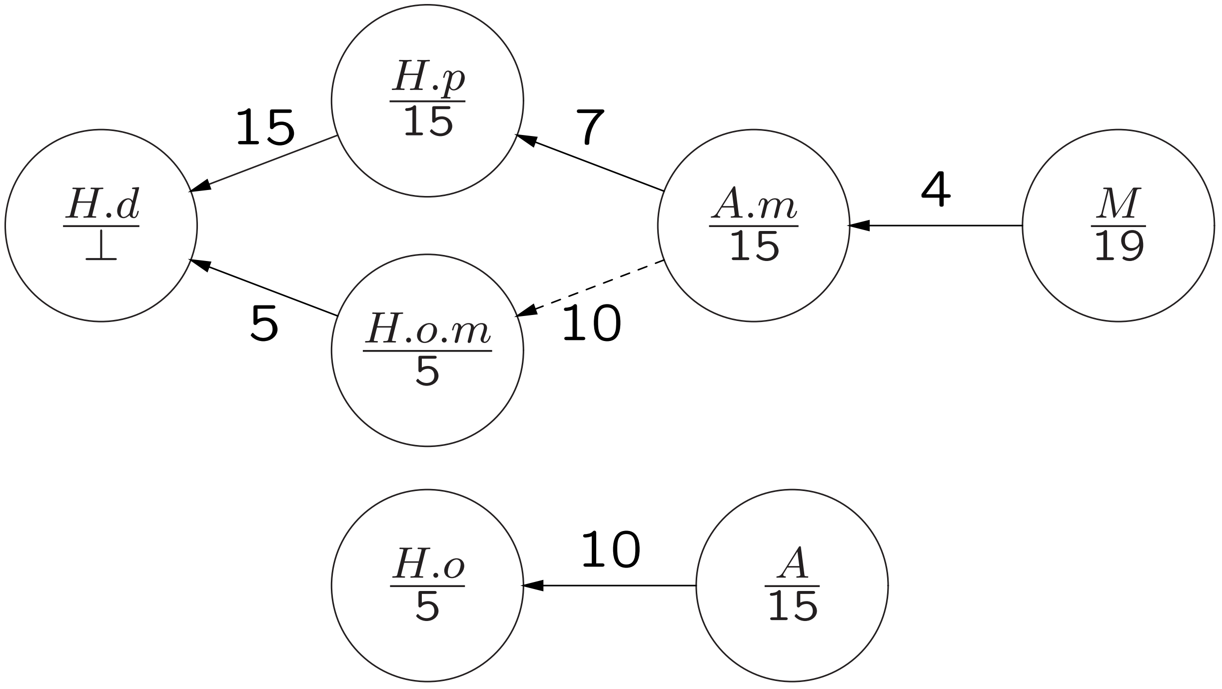
$\kappa_M = 20$

Search Algorithm Example: 5



$\kappa_M = 20$

Search Algorithm Example: 6



$\kappa_M = 20$

Future Work: Trust-but-Verify

- Context of authorization is formally transformed to include trusted elements to speed up the on-line decision.

- Off-line verification checks the on-line result*

- In RT^R the trust transformation could inject new, high risk credentials and raise the search risk threshold.

- Verification could search without the injected credentials or prove that the injected credentials do not produce spurious results.

*Skalka and Wang, *Trust But Verify: Authorization for Web Services* ACM Workshop on Secure Web Services; Fairfax, Virgina; October 29, 2004.

Future Work: Cost/Benefit Analysis

- Let risk values have the form (κ, t)
- Let $(\kappa_1, t_1) \preccurlyeq (\kappa_2, t_2) \Leftrightarrow (\kappa_1 \preccurlyeq \kappa_2) \wedge (t_1 \preccurlyeq t_2)$
- If a search fails, one can try again raising either κ or t in the threshold.
- Can trade off inherently risky credentials against those that are hard to verify.

Questions?

http://www.cs.uvm.edu/∼skalka/skalka-pubs/skalka-projects.html