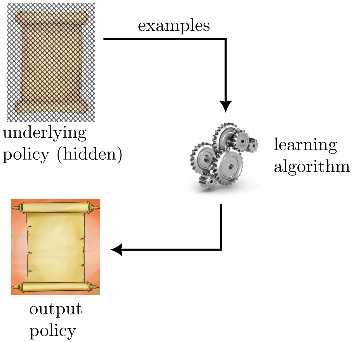
The Hardness of Learning Access Control Policies

Xiaomeng Lei   Mahesh Tripunitara
speaker

University of Waterloo, Canada
{xiaomeng.lei,tripunit}@uwaterloo.ca

Setting



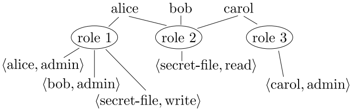
- Policy = Access control policy [12, 13, 16, 17]
- Parameters:
  - Policy model
  - Examples
  - Goodness properties of learning algorithm

Parameters I — Policy model

- We consider: Access matrix, RBAC, ReBAC
- A model specifies:
  - Policy syntax
  - Access request syntax + enforcement algorithm

Underlying policy model may = or ≠ output policy model.

	alice	bob	carol	secret-file
alice	admin	admin		read, write
bob				read
carol			admin	read



Parameters II — Examples

- Data from *access enforcement* [12, 13, 16, 17].
- Specifically, example $= \langle \textit{access-request}, \pi \rangle$
  - $\pi =$ some string generated during enforcement of *access-request*.
  - Special case: $\pi \in \{\text{allow}, \text{deny}\}$.
- Examples drawn under some distribution, $\mathcal{D}$.

Parameters III — Goodness of learning algorithm

1. Output policy suffers low error with high probability.
   - Adopt a notion of error $=$ $\Pr_{\mathcal{D}}\{\text{difference between underlying \& output policies}\}$
   - Then, for any δ, ϵ, where $\delta, \epsilon \in (0, 1/2)$, we seek: $\Pr\{\text{error} \leq \epsilon\} \geq 1 - \delta$.
   - "*Probably, approximately correct.*"
2. Learning algorithm runs in time polynomial in:
   Size of underlying policy, $1/\epsilon$, $1/\delta$

Models (syntax) and examples matter!

$$x_1 \wedge ((x_2 \wedge \neg x_1) \vee \neg x_4 \vee x_3)$$

$\langle\langle 1,1,1,0\rangle, 1\rangle$

$\longrightarrow$

$\langle\langle 1,0,1,0\rangle, 1\rangle$

$\langle\langle 0,0,1,1\rangle, 0\rangle$

as CNF $=$ easy

as DNF $=$ hard

	p_1	p_2	p_3	p_4
u_1		✓		✓
u_2			✓	
u_3	✓		✓	

$\langle\langle u_1, p_4\rangle, 1\rangle$

$\longrightarrow$

$\langle\langle u_2, p_3\rangle, 1\rangle$

$\langle\langle u_2, p_2\rangle, 0\rangle$

as RBAC $=$ easy

as min. # roles RBAC $=$ hard

Learning Problem I

- Underlying, output model = access matrix with positive rights
- Example = $\langle\langle u_i, p_j\rangle, \{\text{allow,deny}\}\rangle$
- Error = $\Pr_{\langle\langle u,p\rangle,b\rangle\leftarrow\mathcal{D}}\{\text{In output policy, } \langle u,p\rangle \mapsto \neg b\}$

	p_1	p_2	p_3	p_4
u_1		✓		✓
u_2			✓	
u_3	✓		✓	

$\emptyset$

$\langle\langle u_2, p_2\rangle, 0\rangle \longrightarrow$ $\emptyset$

$\langle\langle u_1, p_4\rangle, 1\rangle \longrightarrow$

	p_4
u_1	✓

$\langle\langle u_2, p_3\rangle, 1\rangle \longrightarrow$

	p_4	p_3
u_1	✓	
u_2		✓

$\langle\langle u_2, p_1\rangle, 0\rangle \longrightarrow$

	p_4	p_3
u_1	✓	
u_2		✓

Learning Problem I, contd.

This algorithm is efficient, and probably, approximately correct.

Proof.

- Only possible kind of error: output policy denies an access-request $\langle u, p \rangle$ that it should allow.
- Consider $\langle \langle u, p \rangle, 1 \rangle$ s.t. $\Pr_{\mathcal{D}} \{ \langle \langle u, p \rangle, 1 \rangle \text{ occurs} \} \geq \epsilon / n$, where $n =$ size of underlying policy.
- $\#$ examples $\geq (n/\epsilon)(\ln(n) + \ln(1/\delta)) \implies$ error $\leq \epsilon$

□

Learning Problem II

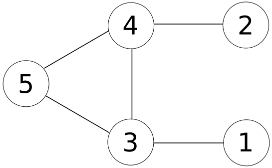
- Underlying, output policy = access matrix with positive and negative rights.
  - With "deny overrides" enforcement discipline.
- Example = $\langle \langle u, p \rangle, \{\text{allow,deny}\} \rangle$
- Error = $\Pr\{\text{difference in entries}\}$

	p_1	p_2	p_3	p_4
u_1		✓		✓
u_2			✓✗	✗
u_3	✓		✓	

$\langle u_1, p_2 \rangle \mapsto \text{allow}$ $\quad \langle u_3, p_2 \rangle \mapsto \text{deny}$

$\langle u_2, p_3 \rangle \mapsto \text{deny}$ $\quad \langle u_2, p_4 \rangle \mapsto \text{deny}$

To Prove Hardness



Decision problem

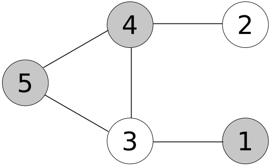
- Given $\langle G, k \rangle$, does G have a vertex cover of size k?
- If this problem $\in$ **RP**, then **RP** $=$ **NP**.

Certificate-construction problem

- Given $\langle G, k \rangle$ such that G has a vertex cover of size k, output one.
- If this problem has a randomized polynomial-time algorithm, then **RP** $=$ **NP**.

Encode $\langle G, k \rangle$ + certificate as an underlying access matrix.

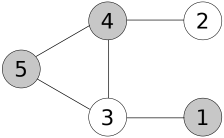
E.g., for $k = 3$



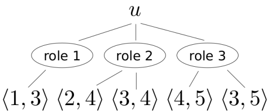
	p_1	p_2	p_3	p_4	p_5	p_6
u_1	✗		✓			✓
u_2				✓		✓
u_3	✓			✓	✓	✓
u_4		✓	✓	✗	✓	
u_5			✓	✓	✗	

Learning Problem III — RBAC

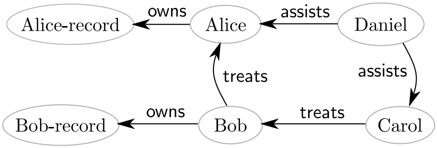
- Underlying, output policy = RBAC
- Example =
  - Role-activation, $\langle\langle u_i, r_j\rangle, \{\text{allow,deny}\}\rangle$
  - Access-request, $\langle\langle u_a, p_b\rangle, \{allow, deny\}\rangle$
- Error = $\Pr\{\text{Output policy} \not\equiv \text{underlying}\}$



Edge	$m(\cdot)$
$\langle 1, 3\rangle$	1
$\langle 2, 4\rangle$	4
$\langle 3, 4\rangle$	4
$\langle 3, 5\rangle$	5
$\langle 4, 5\rangle$	5



ReBAC — Model



System Graph, G

⟨owns⟩
⟨treats.owns⟩
⟨assists.treats.owns⟩

Relationship Patterns, P

$\langle \text{Bob}, \text{Bob-record} \rangle \mapsto \text{allow}$
$\langle \text{Daniel}, \text{Alice-record} \rangle \mapsto \text{deny}$

$\langle \text{Bob}, \text{Alice-record} \rangle \mapsto \text{allow}$
$\langle \text{Daniel}, \text{Carol} \rangle \mapsto \text{deny}$

Learning Problem IV — ReBAC

- Underlying = ReBAC policy
- Output = DFA
  - Min. # states
  - Accepts $S \subseteq P$ that occur in G; rejects others
- Example = $\langle \langle u, v \rangle, \pi \rangle$
  - u, v: vertices in G
  - $\langle u, v \rangle \mapsto$ allow $\implies \pi \in P$
  - $\langle u, v \rangle \mapsto$ deny $\implies \pi = \phi$
- Error = $\Pr\{\cdot\}$ either:
  - DFA accepts/rejects string incorrectly, or:
  - DFA is not min. sized

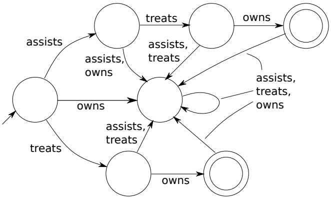
Learning Problem IV — Algorithm

- Start with DFA that rejects everything
- Ignore example in which $\pi = \phi$
- Otherwise, modify DFA to accept π
- Run minimization algorithm [10]

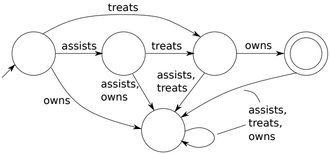
Only error in DFA is it rejects string it should accept.
DFA is not min. # sized $\implies$ it rejects a string it should accept.

After:
$\langle\langle$Daniel, Bob-record$\rangle$, $\langle$assists.treats.owns$\rangle\rangle$
$\langle\langle$Bob, Alice-record$\rangle$, $\langle$treats.owns$\rangle\rangle$

Without minimization



With minimization



fin