18734: Foundations of Privacy

Policy Auditing over Incomplete Logs: The reduce algorithm

Anupam Datta
Carnegie Mellon University

Fall 2014

Example from HIPAA Privacy Rule

A covered entity may disclose an individual's protected health information (phi) to law-enforcement officials for the purpose of identifying an individual if the individual made a statement admitting participating in a violent crime that the covered entity believes may have caused serious physical harm to the victim

- Concepts in privacy policies
  - **Actions:** send(p1, p2, m)
  - **Roles:** inrole(p2, law-enforcement)
  - **Data attributes:** attr_in(prescription, phi)
  - **Temporal constraints:** in-the-past(state(q, m))

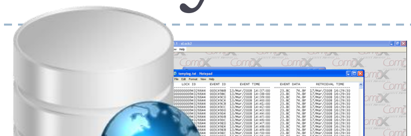
  - **Purposes:** purp_in(u, id-criminal))
  - **Beliefs:** believes-crime-caused-serious-harm(p, q, m)

Black-and-white concepts

Grey concepts

Detecting Privacy Violations

Priv



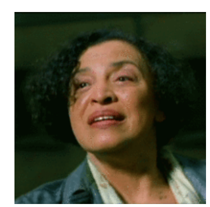
The Oracle	
The Matrix character	
Species	Computer Program
Title	A program designed to investigate the human psyche.

Automated audit for black-and-white policy concepts

Detect policy violations

$$\mathbf{G}\left(\forall p_1, p_2, m.\ \mathsf{send}(p_1,p_2,m) \supset (\forall d,u,q,t.\ (m = \mathrm{info}(d,u)) \wedge \mathsf{contains}(m,q,t) \supset (\bigvee_i \varphi_i^+) \wedge (\bigwedge_j \varphi_j^-)) \wedge (\forall t.\ (m = \mathrm{req\_for\_access}(p_1,t)) \supset \varphi^-_{\mathtt{164.524b2i'}} \wedge \varphi^-_{\mathtt{164.524b2ii'}})\right)$$

Computer-readable privacy policy



Audit

Oracles to audit for grey policy concepts

Auditing Black-and-White Policy Concepts

With D. Garg (CMU → MPI-SWS) and L. Jia (CMU)

2011 ACM Conference on Computer and Communications Security

Key Challenge for Auditing

Audit Logs are Incomplete

Future: store only past and current events

Example: Timely data breach notification refers to future event

Subjective: no "grey" information

Example: May not record evidence for purposes and beliefs

Spatial: remote logs may be inaccessible

Example: Logs distributed across different departments of a hospital

Abstract Model of Incomplete Logs

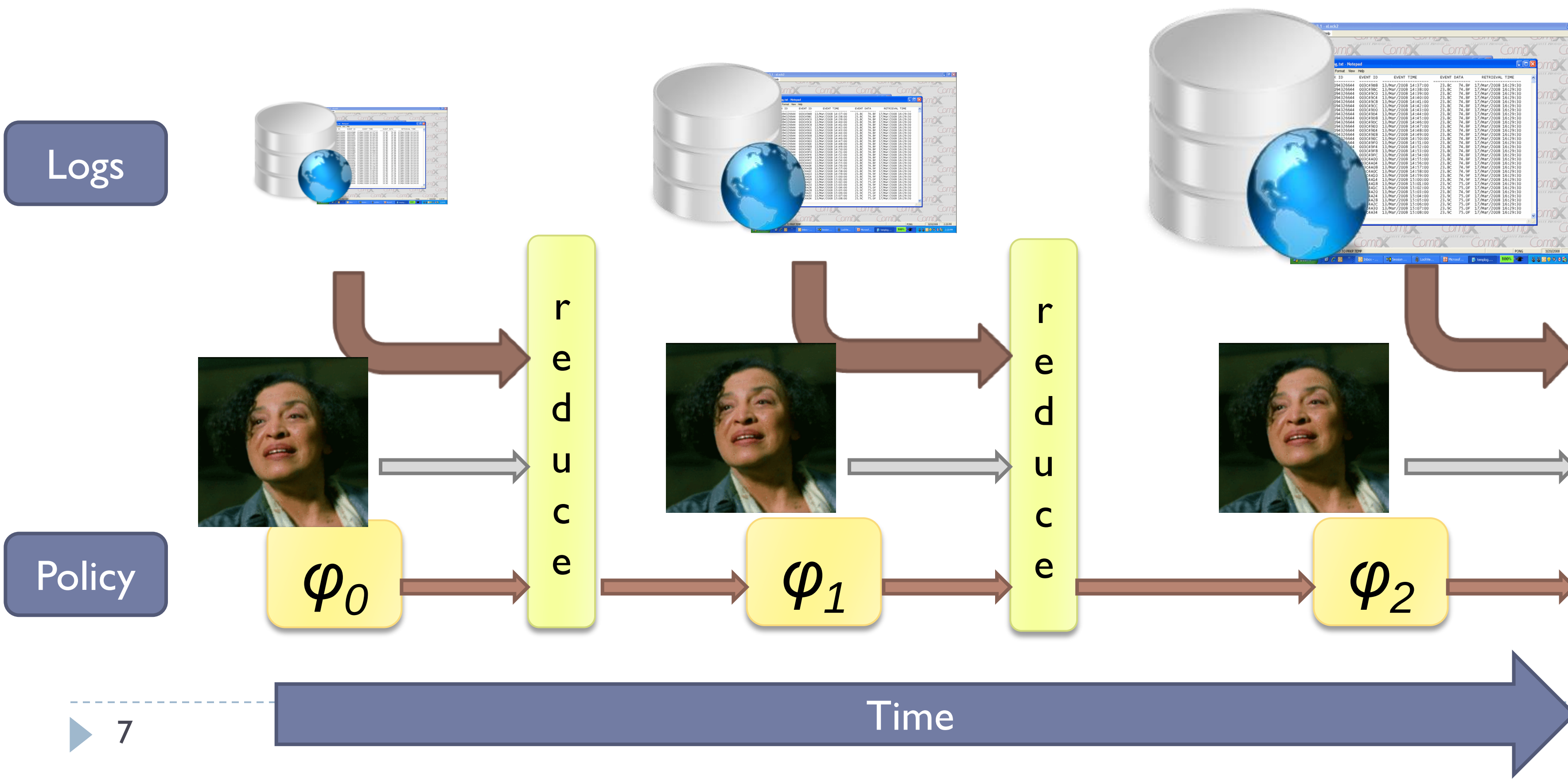
Model **all** incomplete logs uniformly as 3-**valued structures**

$$\mathcal{L}(P) \in \{\mathtt{tt}, \mathtt{ff}, \mathtt{uu}\}$$

Define **semantics** (meanings of formulas) over 3-valued structures

reduce: The Iterative Algorithm

$$\text{reduce}\ (\mathcal{L}, \varphi) = \varphi'$$



Syntax of Policy Logic

$$
\begin{array}{llll}
\text{Atoms} & P & ::= & p(t_1, \ldots, t_n) \\
\text{Formulas} & \varphi & ::= & P \mid \top \mid \bot \mid \\
 & & & \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid \\
 & & & \forall \vec{x}.(c \supset \varphi) \mid \exists \vec{x}.(c \wedge \varphi) \\
\text{Restrictions} & c & ::= & P \mid \top \mid \bot \mid c_1 \wedge c_2 \mid \\
 & & & c_1 \vee c_2 \mid \exists x.c
\end{array}
$$

- First-order logic with restricted quantification over *infinite domains* (challenge for reduce)
- Can express timed temporal properties, "grey" predicates

Example from HIPAA Privacy Rule

A covered entity may disclose an individual's protected health information (phi) to law-enforcement officials for the purpose of identifying an individual if the individual made a statement admitting participating in a violent crime that the covered entity believes may have caused serious physical harm to the victim

$$
\begin{aligned}
&\forall p1, p2, m, u, q, t.\\
&\quad (\text{send}(p1, p2, m) \wedge\\
&\quad \text{tagged}(m, q, t, u) \wedge\\
&\quad \text{attr\_in}(t, \text{phi}))\\
&\quad \supset \text{inrole}(p1, \text{covered-entity}) \wedge \ \text{inrole}(p2, \text{law-enforcement})\\
&\qquad (\text{purp\_in}(u, \text{id-criminal})) \wedge\\
&\qquad \wedge \exists\, m'. \Diamond\!\!-\, \text{state}(q, m') \wedge \text{is-admission-of-crime}(m')\\
&\qquad \wedge \text{believes-crime-caused-serious-harm}(p1, q, m')
\end{aligned}
$$

reduce: Formal Definition

$\{ \top \quad \text{if } \rho(P) = \text{tt}$

General Theorem: If initial policy passes a syntactic **mode check**, then finite substitutions can be computed

$\texttt{reduce}(L, \; \forall x.\varphi \quad)$

c is a fo
fini
substitu
c

Applications: The entire HIPAA and GLBA Privacy Rules pass this check

Example

φ =

∀p1, p2, m, u, q, t.

(send(p1, p2, m) ∧
tagged(m, q, t, u) ∧
attr_in(t, phi))

inrole(p1, covered-entity) ∧ inrole(p2, law-enforcement)

∧ purp_in(u, id-criminal)

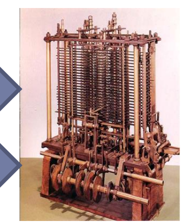
∧ ∃ m'. ($\Diamond^{-}$ state(q, m')

∧ is-admission-of-crime(m')

∧ believes-crime-caused-serious-harm(p1, m'))

{ p1→ UPMC,
p2→ allegeny-police,
m → M2,
q → Bob,
u → id-bank-robber,
t → date-of-treatment
}

{ m' → M1 }



Log

Aug 15, 2014
state(Bob, M1)

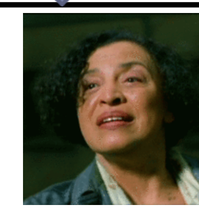
Sept 17, 2014
send(UPMC, allegeny-police, M2)
tagged(M2, Bob, date-of-treatment, id-bank-robber)

φ' = T

∧ purp_in(id-bank-robber, id-criminal)

∧ is-admission-of-crime(M1)

∧ believes-crime-caused-serious-harm(UPMC, M1)



Correctness of Reduce

Theorem 3.2 (Partial correctness of reduce). *If* $\mathsf{reduce}(\mathcal{L}, \varphi) = \psi$ *and* $\mathcal{L} \leq \mathcal{L}'$*, then (1)* $\mathcal{L}' \models \varphi$ *iff* $\mathcal{L}' \models \psi$ *and (2)* $\mathcal{L}' \models \overline{\varphi}$ *iff* $\mathcal{L}' \models \overline{\psi}$.

Implementation and Case Study

- Implementation and evaluation over simulated audit logs for compliance with *all* 84 disclosure-related clauses of HIPAA Privacy Rule

- Performance:
  - Average time for checking compliance of each disclosure of protected health information is 0.12s for a 15MB log

- Mechanical enforcement:
  - reduce can automatically check 80% of all the atomic predicates

Ongoing Transition Efforts

- Integration of reduce algorithm into Illinois Health Information Exchange prototype
  - Joint work with UIUC and Illinois HLN
- Auditing logs for policy compliance
  - Ongoing conversations with Symantec Research

Applications of Reduce

- Audit to detect violations of policy or demonstrate compliance
- Provide explanations for violations (e.g., which clause of HIPAA was violated)
- Help train employees about privacy laws (e.g., check whether a certain type of disclosure is permitted by HIPAA)

Learning Outcomes for You

- Translate privacy laws into first-order logic for use by reduce
- Use reduce tool to check logs for compliance with laws
- Use reduce to check whether certain types of disclosures are permitted by a privacy law

Homework 1 will make you work through these problems

Possible project around other privacy laws such as FERPA, COPPA

Related Work

Privacy Specification Languages

- P3P[Cranor et al.], XACML[OASIS], EPAL[Backes et al.]:
  Less expressive (no temporal ops,..)
- *Logic of Privacy and Utility* [Barth et al]: Related specification logic; enforcement only for propositional fragment

Related Work

Logical Specification of Privacy Laws

Smaller fragments of laws

- *Logic of Privacy and Utility* [Barth et al.]: Example clauses from HIPAA and GLBA
- PrivacyAPIs [Gunter et al.]: HIPAA164.506
- Datalog HIPAA [Lam et al.]: HIPAA 164.502, 164.506, 164.510

Related Work

Runtime monitoring in MFOTL

[Basin et al '10]

- Pre-emptive enforcement
- Efficient implementation
- Assumes past-completeness of logs
- Less expressive mode checking ("safe-range check")
- Cannot express HIPAA or GLBA

Related Work

Industry practice

Fairwarning Audit Tool

- Customized SQL queries over access logs
- Queries not tied to policy clauses

Detecting Policy Violations

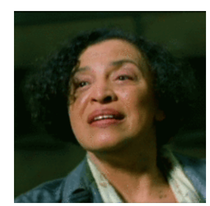
Pri

The Oracle	
The Matrix character	
Species	Computer Program
Title	A program designed to investigate the human psyche.

Automated audit for black-and-white policy concepts

Detect policy violations

$$\mathbf{G}\ \Big(\forall p_1, p_2, m.\ \mathsf{send}(p_1,p_2,m) \supset \\ (\forall d,u,q,t. \\ (m = \mathrm{info}(d,u)) \wedge \mathsf{contains}(m,q,t) \supset \\ (\bigvee_i \varphi_i^+) \wedge (\bigwedge_j \varphi_j^-)) \wedge \\ (\forall t.\ (m = \mathrm{req\_for\_access}(p_1,t)) \supset \\ \varphi^-_{164.524b2i'} \wedge \varphi^-_{164.524b2ii'}\Big)$$



Oracles to audit for grey policy concepts

Computer-readable privacy policy

Audit

Thanks!

More Technical Details

Definition of $\widehat{\texttt{sat}}$

Assume: The function $\texttt{sat}(L, P)$ computes all substitutions σ for variables in P such that $L \models P\sigma$, if certain argument positions in P are ground.

$$
\begin{array}{lcl}
\widehat{\texttt{sat}}(L, p_O(t_1, \ldots, t_n)) & = & \texttt{sat}(L, p_O(t_1, \ldots, t_n)) \\
\widehat{\texttt{sat}}(L, \top) & = & \{\bullet\} \\
\widehat{\texttt{sat}}(L, \bot) & = & \{\} \\
\widehat{\texttt{sat}}(L, c_1 \wedge c_2) & = & \bigcup_{\sigma \in \widehat{\texttt{sat}}(L, c_1)} \sigma + \widehat{\texttt{sat}}(L, c_2\sigma) \\
\widehat{\texttt{sat}}(L, c_1 \vee c_2) & = & \widehat{\texttt{sat}}(L, c_1) \cup \widehat{\texttt{sat}}(L, c_2) \\
\widehat{\texttt{sat}}(L, \exists x.c) & = & \widehat{\texttt{sat}}(L, c) \backslash \{x\} \qquad (x \text{ fresh})
\end{array}
$$

Mode Analysis: Idea

- Example 1: addless(x, y, a) = $x + y < a$

- Key idea: If input positions are grounded, then only finite number of satisfying substitutions for output positions.

- Example 1 moding: addless(+, -, +)

- Example 2: θ = send(p1, p2, m) $\wedge$ tagged(m, q, t, u)

- send(-,-,-): all positions are output mode
- tagged(+,-,-,-): message position is input mode

Mode Analysis: Predicates

$$\boxed{\chi_I \vdash c : \chi_O}$$

1. {} |- send(p1, p2, m): {p1, p2, m}
2. {p1, p2, m} |- tagged(m, q, t, u): {p1, p2, m, q, t, u}

$$\frac{\forall k \in I(p_O).\ \mathtt{fv}(t_k) \subseteq \chi_I \qquad \chi_O = \chi_I \cup \left(\bigcup_{j \in O(p_O)} \mathtt{fv}(t_j) \right)}{\chi_I \vdash p_O(t_1, \ldots, t_n) : \chi_O}$$

Mode Analysis: Conjunction

1. {} |- send(p1, p2, m): {p1, p2, m}
2. {p1, p2, m} |- tagged(m, q, t, u): {p1, p2, m, q, t, u}

3. {} |- send(p1, p2, m) ∧ tagged(m, q, t, u): {p1, p2, m, q, t, u}

$$\frac{\chi_I \vdash c_1 : \chi \qquad \chi \vdash c_2 : \chi_O}{\chi_I \vdash c_1 \wedge c_2 : \chi_O}$$

Mode Analysis and $\widehat{\texttt{sat}}$

Example: θ = send(p1, p2, m) $\wedge$ tagged(m, q, t, u)

- send(-,-,-): all positions are output mode
- tagged(+,-,-,-): message position is input mode
- $\widehat{\texttt{sat}}$ (θ) = sat(send(p1,p2,m)) + sat(tagged(m,q,t,u) σ)

σ

{ p1 $\rightarrow$ UPMC,
p2 $\rightarrow$ allegeny-police,
m $\rightarrow$ M2,
q $\rightarrow$ Bob,
u $\rightarrow$ id-bank-robber,
t $\rightarrow$ date-of-treatment
}

Log

Jan 1, 2011
state(Bob, M1)

Jan 5, 2011
send(UPMC, allegeny-police, M2)
tagged(M2, Bob, date-of-treatment, id-bank-robber)

Mode Analysis: Termination of $\widehat{\texttt{sat}}$

General Theorem: If initial policy passes a syntactic **mode check**, then finite substitutions can be computed

Applications: The entire HIPAA and GLBA Privacy Rules pass this check