Complete, Safe Information Flow with Decentralized Labels

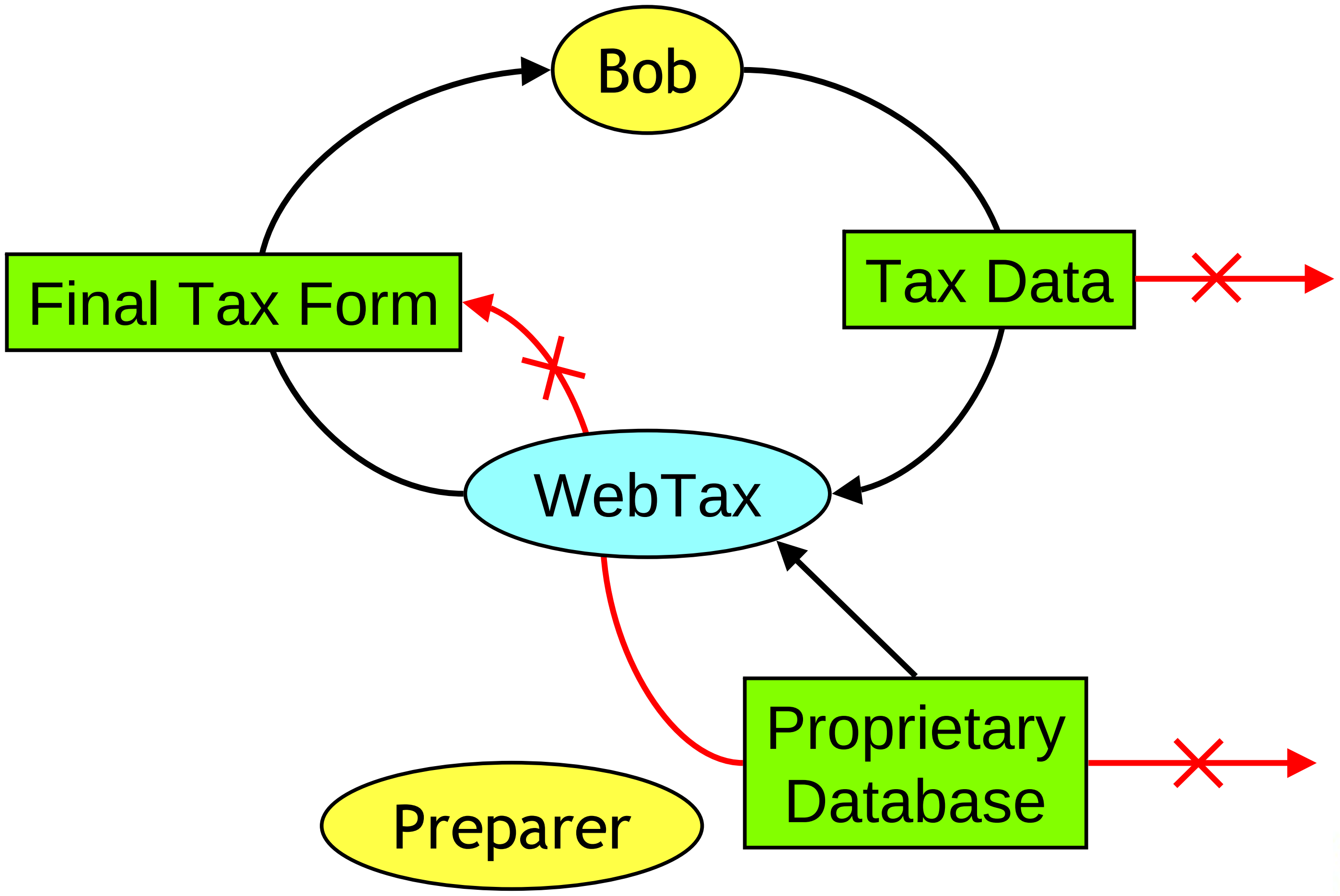
Andrew Myers
Barbara Liskov

MIT Laboratory for Computer Science

Protecting Private Data

- Goal:
  - prevent leaks of private data
  - allow cooperative data sharing
- Technique: statically analyze information flow in programs
  - correctly prevent storage channel leaks
  - good performance
- Implemented Java extension: **JFlow**

Sharing with Mutual Distrust



Decentralized Label Model

- Privacy of multiple principals with mutual distrust: decentralized
- Safe declassification within model
- Static checking: good performance

Outline

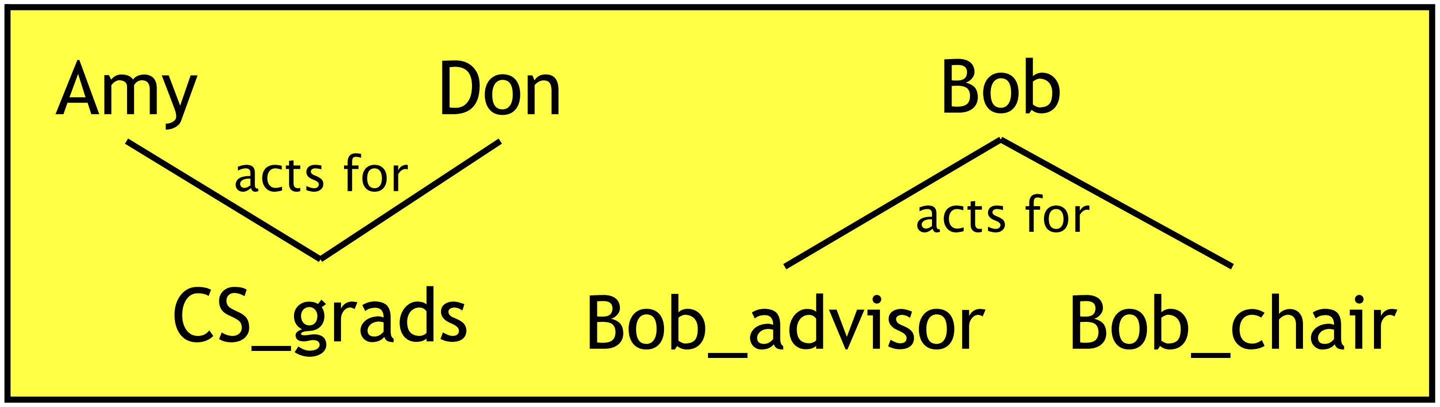
- Decentralized label model, rules
- Static checking & inference
- Soundness & completeness

Model

- Principals
- Labels
- Program rules
  - Assignment
  - Computation
  - Declassification

Principals

- Users, groups, and roles: principals
- Principal hierarchy generated by the acts-for relation ($\succcurlyeq$):



Labels

- Every data item has an attached label
- Label is a set of policies
- Each policy is owner: reader$_1$, reader$_2$,...
  - owner (principal)
  - set of readers (principals)

{ Bob: Bob, Preparer ; Preparer: Preparer }

- Every owner's policy is obeyed
- May have repeated owners

Assignment

- Assignment relabels a value

```
x = y;
```

- $\underline{y} \sqsubseteq \underline{x}$ means

  For every policy in $\underline{y}$, there is a policy in $\underline{x}$ that is at least as restrictive

- Binary label relation $\sqsubseteq$ defines the legal relabelings

Assignment Example

```
int {Bob: Bob, Preparer}  y;
int {Bob: Bob; Preparer: Preparer}  x;
x = y;
```

$\underline{y} \sqsubseteq \underline{x}$?

{Bob: Bob, Preparer} $\sqsubseteq$ {Bob: Bob; Preparer: Preparer}

Computation

- Combining values: new label is *join* ($\sqcup$) of input labels

$$y + z \quad \rightarrow \quad \underline{y} \sqcup \underline{z} = \underline{y} \cup \underline{z}$$

- Label on data reflects all its sources

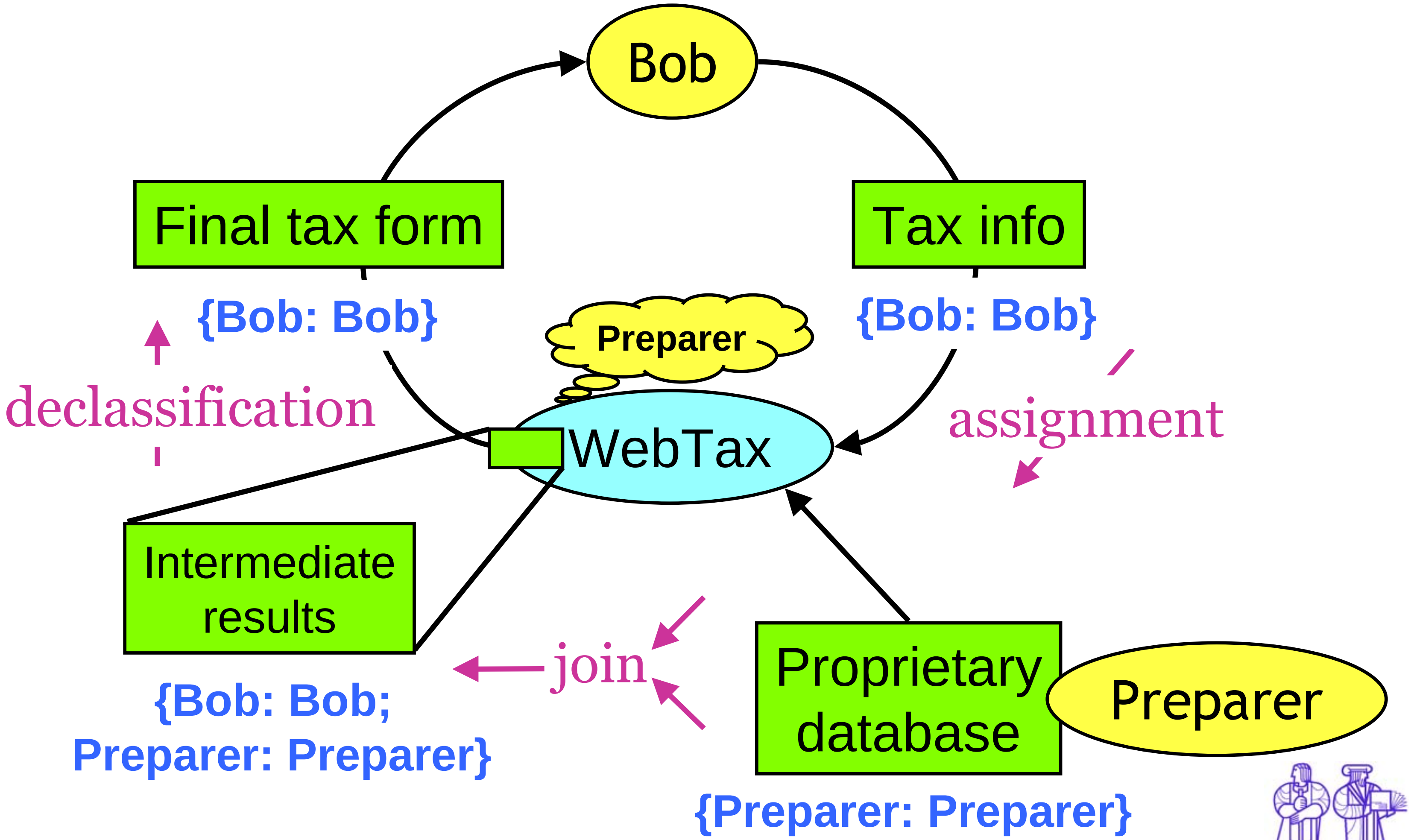
Declassification

- A principal can rewrite its part of the label

$$\begin{tikzcd} & \{A: R1, R2; B: R2\} \arrow[dl] \arrow[dr] & \\ \{A: R1, R2\} & B & \{A: R1, R2; B: R2, R3\} \end{tikzcd}$$

- Other owners' policies still respected

WebTax Example



Outline

- Decentralized label model, rules
- Static checking & inference
- Soundness & completeness

Checking Annotated Programs

- Annotate Java with labels
- Variables have type + label
- Label checking = type checking
- Handles implicit flows, exceptions, objects, dynamic type tests, etc.
- Label polymorphism, inference
- Implemented: **JFlow** compiler

Outline

- Decentralized label model, rules
- Static checking & inference
- Soundness & completeness

Sound Relabeling Rule

- Safe incremental relabelings:

– remove a reader:	{A:B,C} ➔ {A:B}
– add a policy:	{A:B} ➔ {A:B; C:D}
– replace an owner by a superior:	{A:B} ➔ {A’:B}
– add a superior reader:	{B:A} ➔ {B:A,A’}

- Every sequence of relabelings is safe
- What does “safe” mean?

Label Semantics

- Label L denotes a set of *flows* **X**(L)
- Flow is a (*owner*, *reader*) pair
- Omitted owners allow all flows:

$$\mathbf{X}(\{\ A\!:\ B\ ;\ C\!:\ A\ \}) =$$

$$\{\ (A, B), (C, A), (B, A), (B, B), (B, C)\ \}$$

- Constraints from principal hierarchy:

$$r' \succcurlyeq r\ \ \&\ \ (o, r) \in \mathbf{X}(L) \rightarrow (o, r') \in \mathbf{X}(L)$$

$$o' \succcurlyeq o\ \&\ (o', r) \notin \mathbf{X}(L) \rightarrow\ (o, r) \notin \mathbf{X}(L)$$

Safety

- A relabeling is safe if it does not create new flows:

$$L_1 \rightarrow L_2 \text{ is safe if } \mathbf{X}(L_1) \supseteq \mathbf{X}(L_2)$$

- Problem: $\mathbf{X}(L_1)$, $\mathbf{X}(L_2)$ evaluated statically using partial knowledge of principal hierarchy; safety condition must hold in run-time hierarchy!

Soundness/Completeness

- **Soundness: ($\Rightarrow$)**

  For every principal hierarchy consistent with a set of static observations, a relabeling does not create new flows

- **Completeness: ($\Leftarrow$)**

  The relabeling rule is the most permissive sound rule (& captures incremental rules)

$$P \vdash L_1 \sqsubseteq L_2 \Leftrightarrow \forall_{P' \supseteq P}\, \mathbf{X}(P', L_1) \supseteq \mathbf{X}(P', L_2)$$

Inference & Lattice Properties

- Derived values: at least as restrictive as the input values (least upper bound)

  x = y + z $\quad\quad\quad$ $\underline{y} \sqcup \underline{z} \sqsubseteq \underline{x}$

- Inferred variables: at most as restrictive as uses (greater lower bound)

  y = x; z = x; $\quad\quad\quad$ $\underline{x} \sqsubseteq \underline{y} \sqcap \underline{z}$

- Label model ($\sqsubseteq$) provides both LUB and GLB needed for inference

Related Work

- Bell, LaPadula, 1975
- Denning, 1976
- Denning & Denning, 1977
- McCollum, et al. IEEE S & P, 1990
- Ferrari, et al. IEEE S & P, 1997

Conclusions

- New decentralized label model
  - safe declassification
  - supports groups, roles
- Supports static checking
  - distributive LUB/GLB operators
  - label inference algorithm
- Formal semantics
  - relabeling proven sound and complete