Disjunction Category Labels

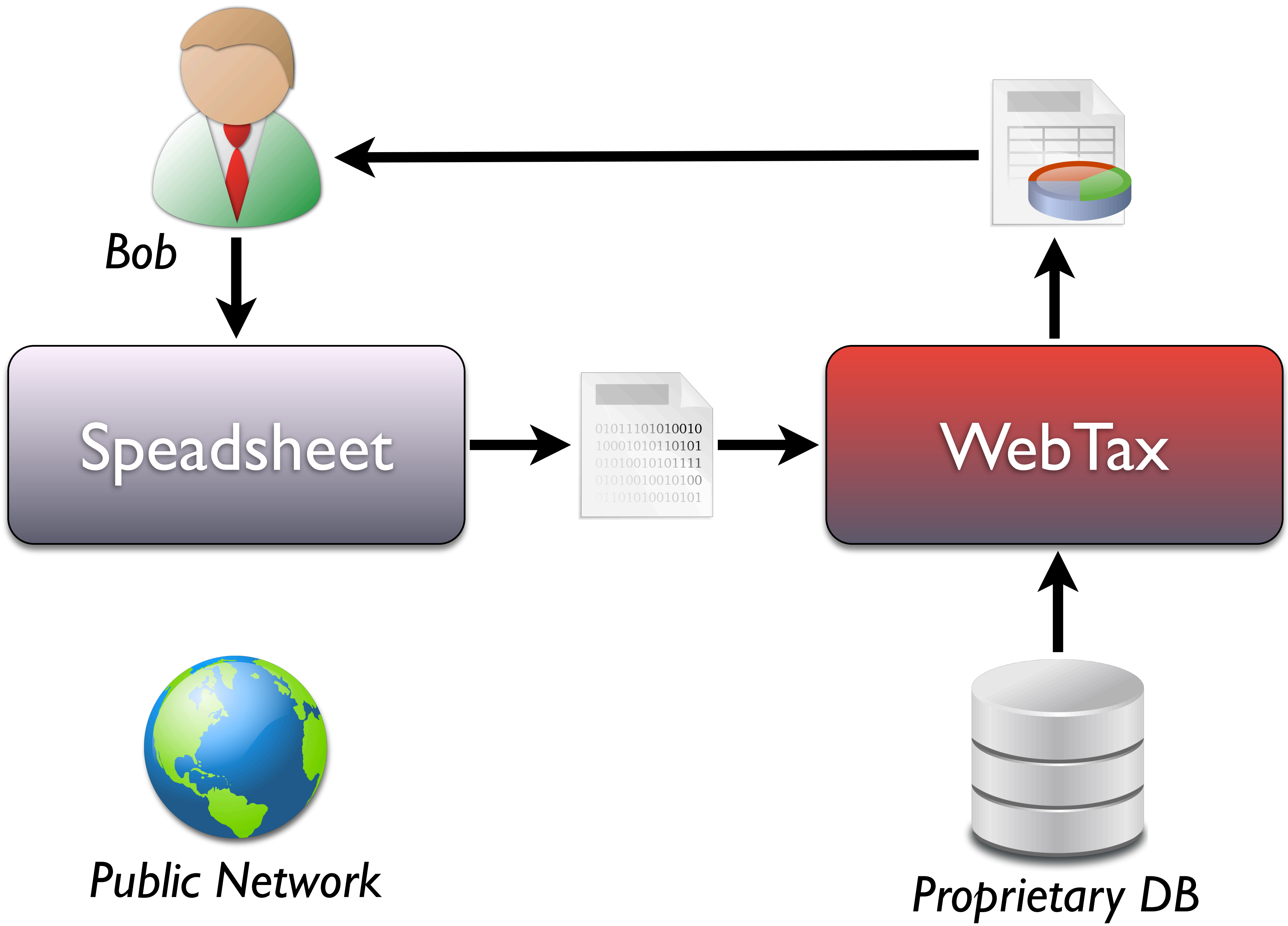
Deian Stefan, Alejandro Russo, David Mazières, John Mitchell

STANFORD UNIVERSITY

CHALMERS

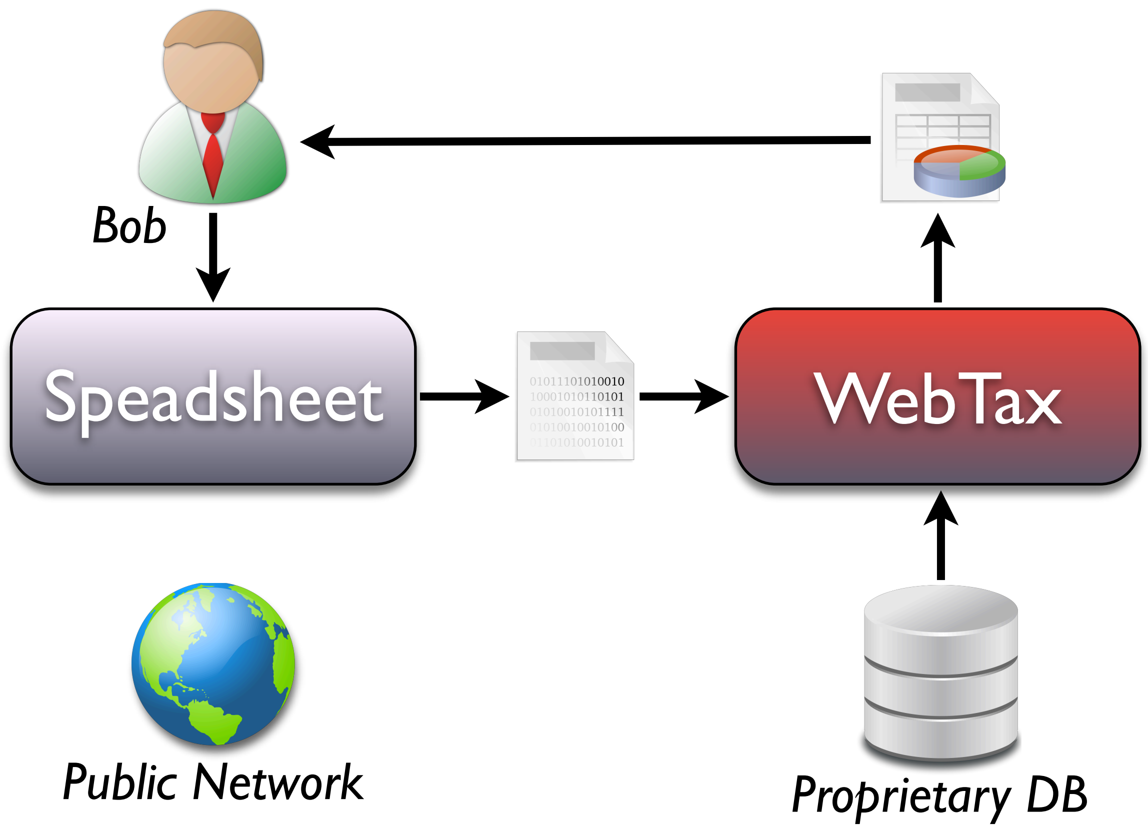
NordSec 2011

Motivating Example



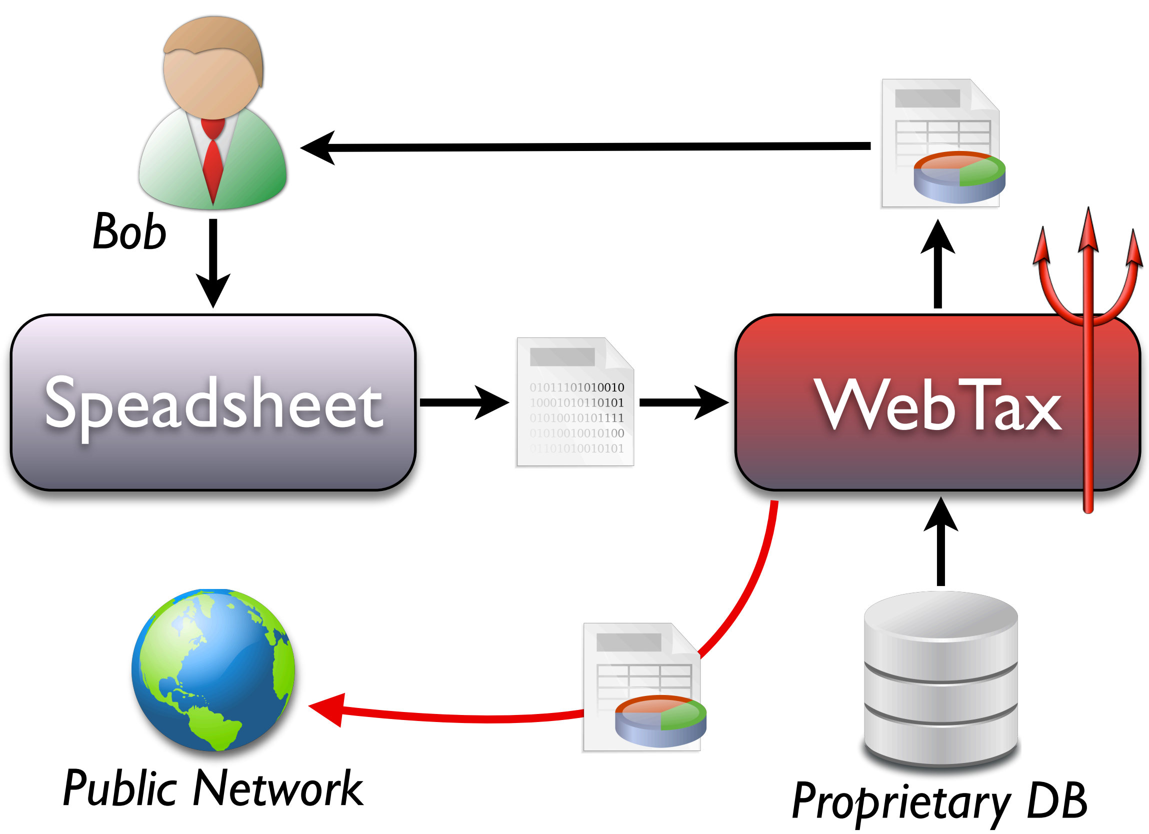
Motivating Example

- Bob does not trust WebTax
  - ➤ WebTax can exfiltrated his data
- WebTax author does not trust Bob
  - ➤ Bob can learn proprietary information by inspecting code
- WebTax author want to prevent leaks due to bugs



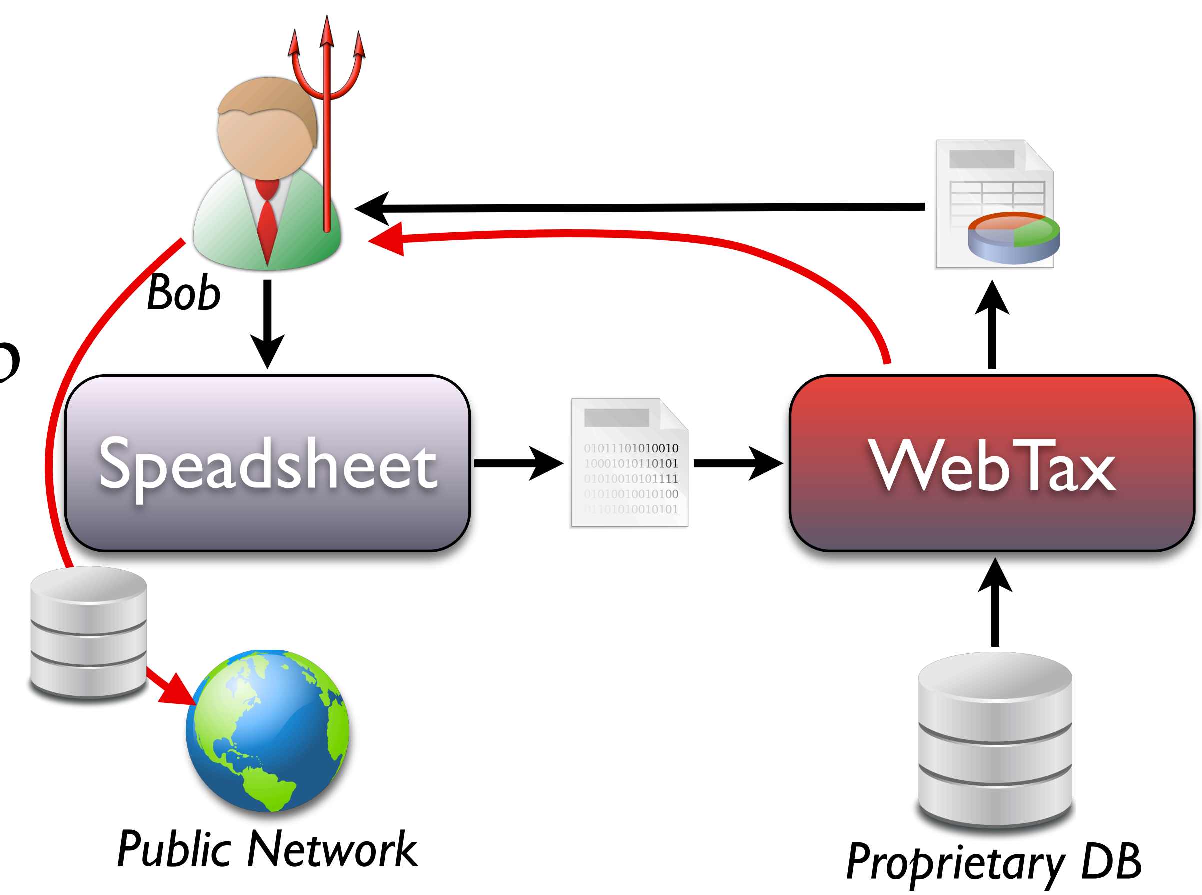
Motivating Example

- Bob does not trust WebTax
  - ➤ WebTax can exfiltrated his data
- WebTax author does not trust Bob
  - ➤ Bob can learn proprietary information by inspecting code
- WebTax author want to prevent leaks due to bugs



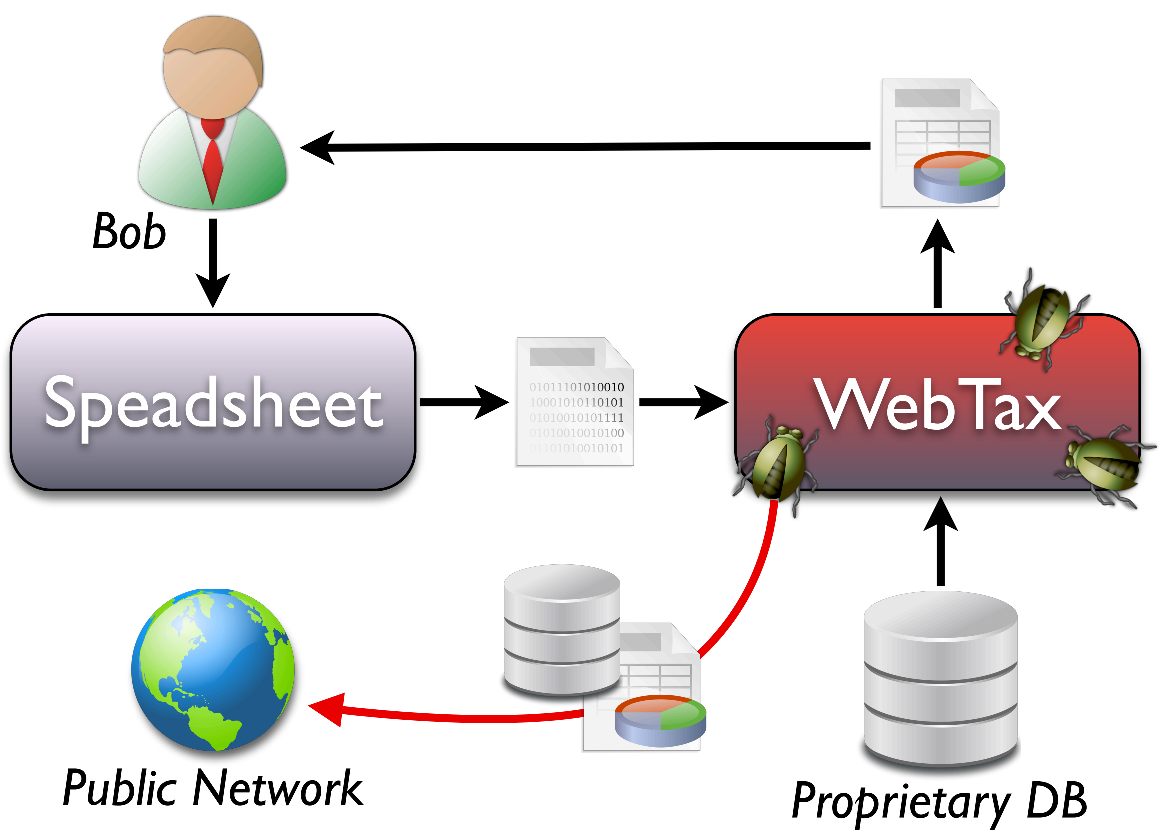
Motivating Example

- Bob does not trust WebTax
  - ➤ WebTax can exfiltrated his data
- WebTax author does not trust Bob
  - ➤ Bob can learn proprietary information by inspecting code
- WebTax author want to prevent leaks due to bugs



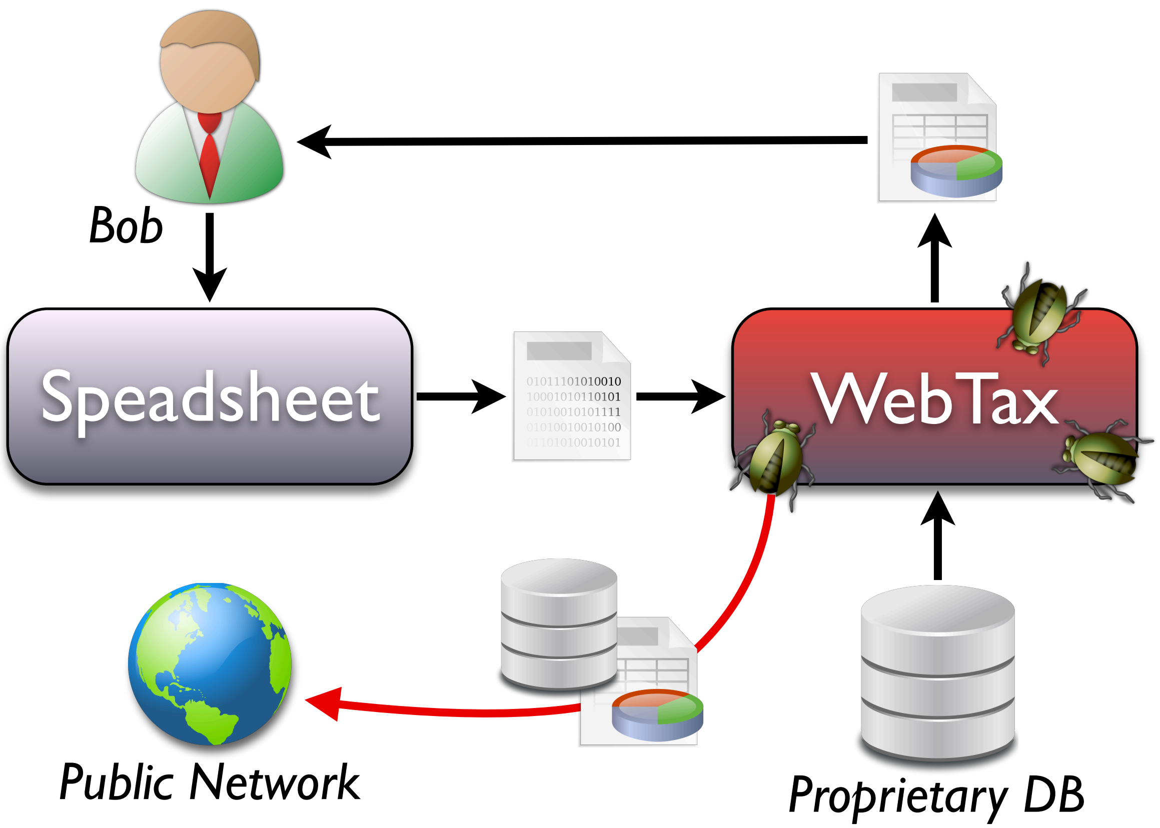
Motivating Example

- Bob does not trust WebTax
  - ➤ WebTax can exfiltrated his data
- WebTax author does not trust Bob
  - ➤ Bob can learn proprietary information by inspecting code
- WebTax author want to prevent leaks due to bugs



Motivating Example

- Bob does not trust WebTax
  - ➤ WebTax can exfiltrated his data
- WebTax author does not trust Bob
  - ➤ Bob can learn proprietary information by inspecting code
- WebTax author want to prevent leaks due to bugs

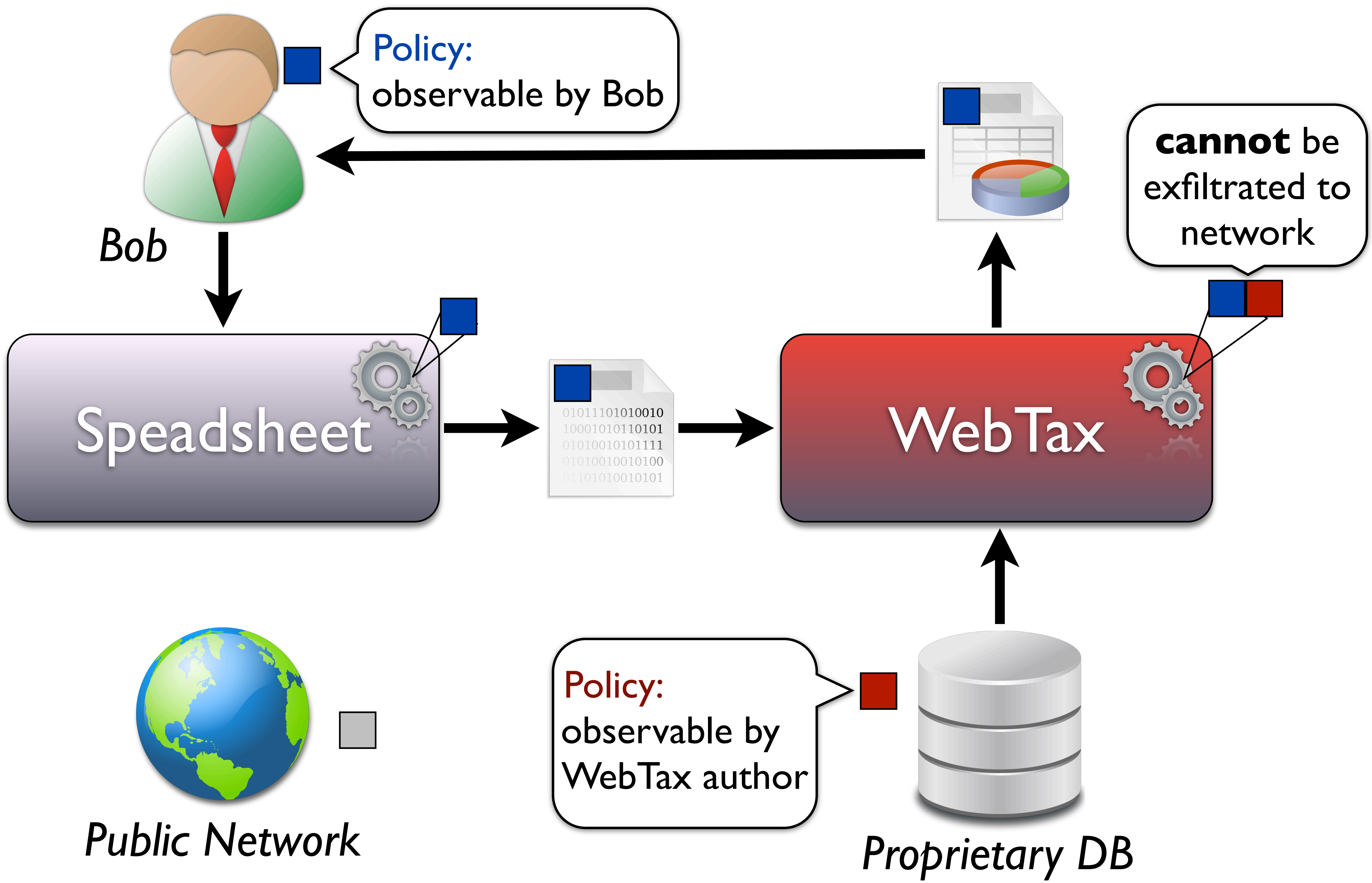


How do we address security in the presence of mutual-distrust?

Information Flow Control

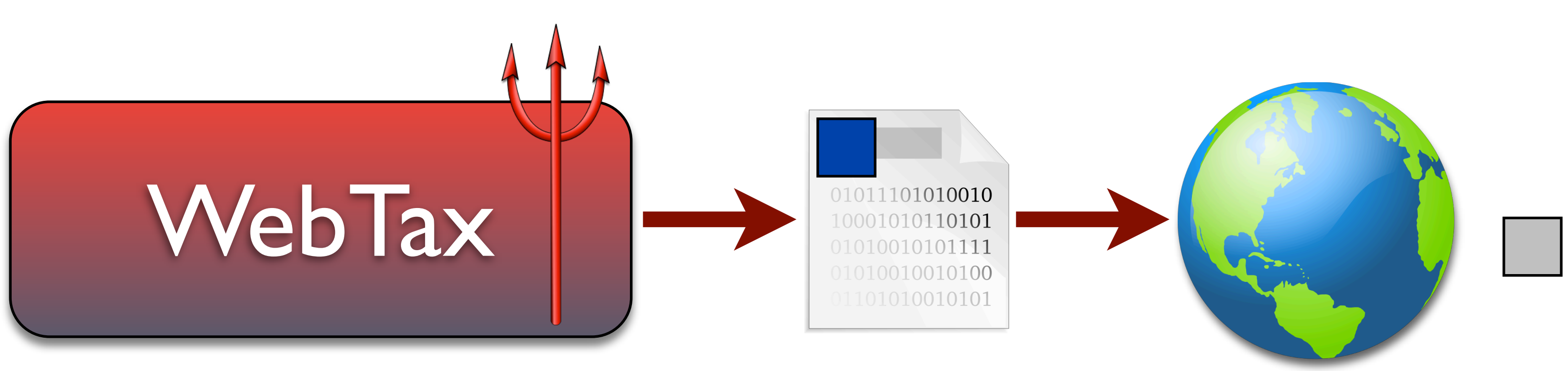
- Well-established approach to enforcing security
  - **Confidentiality:** prevent unwanted leaks
  - **Integrity:** prevent flows to critical operations
- Decentralized IFC addresses mutual distrust
- Suitable for executing *untrustworthy* code
  - Policies specify where data can flow

Example with IFC



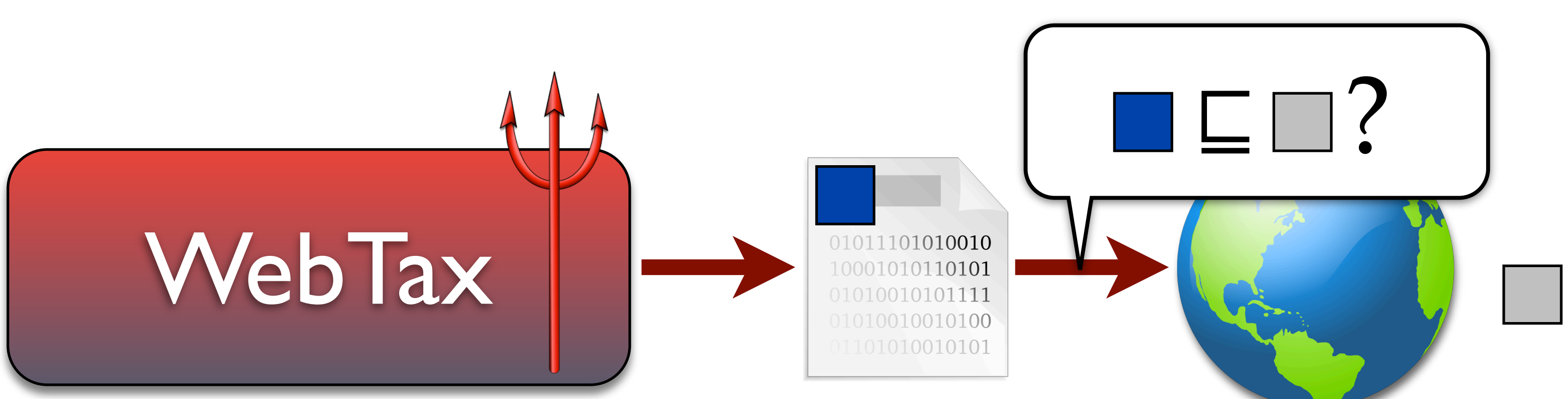
IFC Policies

- How are policies specified?
  - Associating a label ■ with every piece of data
- Labels form a lattice over can-flow-to relation $\sqsubseteq$
  - E.g., Bob's data cannot flow to network ■ $\not\sqsubseteq$ □
- Policies are enforced at every possible flow



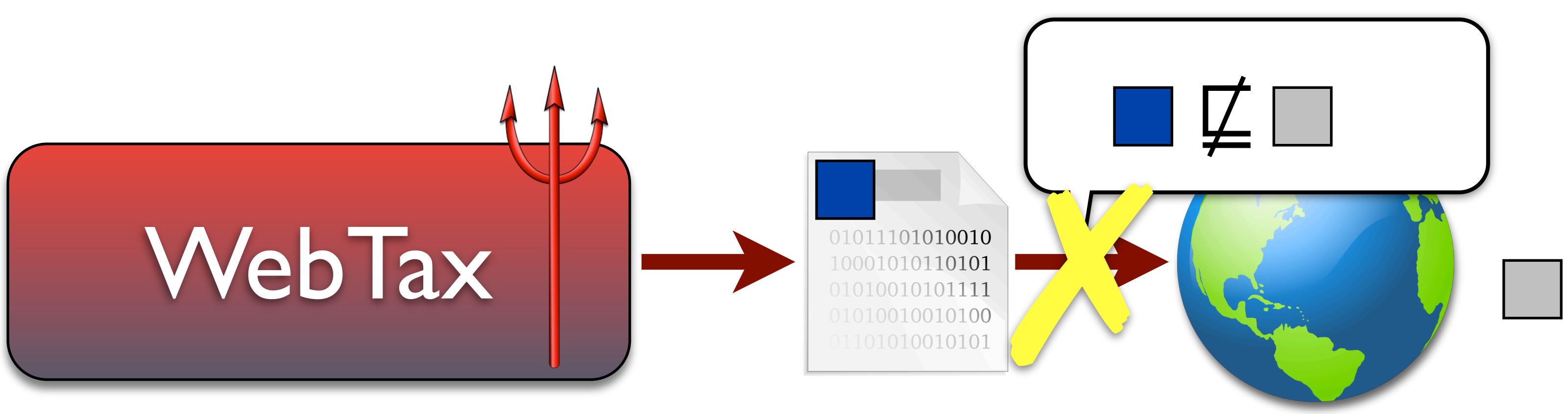
IFC Policies

- How are policies specified?
  - Associating a label ■ with every piece of data
- Labels form a lattice over can-flow-to relation $\sqsubseteq$
  - E.g., Bob's data cannot flow to network $\blacksquare \not\sqsubseteq \square$
- Policies are enforced at every possible flow



IFC Policies

- How are policies specified?
  - Associating a label ■ with every piece of data
- Labels form a lattice over can-flow-to relation $\sqsubseteq$
  - E.g., Bob's data cannot flow to network ■ $\not\sqsubseteq$ □
- Policies are enforced at every possible flow



Motivation for DC Labels

- Existing DIFC systems use ad-hoc label formats
  - DLM, Asbestos/HiStar, DStar, Flume, etc. all present their own label format
- Most labels have *not* been formalized
- Some rely on centralized components
- Need simple, sound, expressive & decentralized label format ➡ *DC Labels*

DC Labels

$$\langle S, I \rangle$$

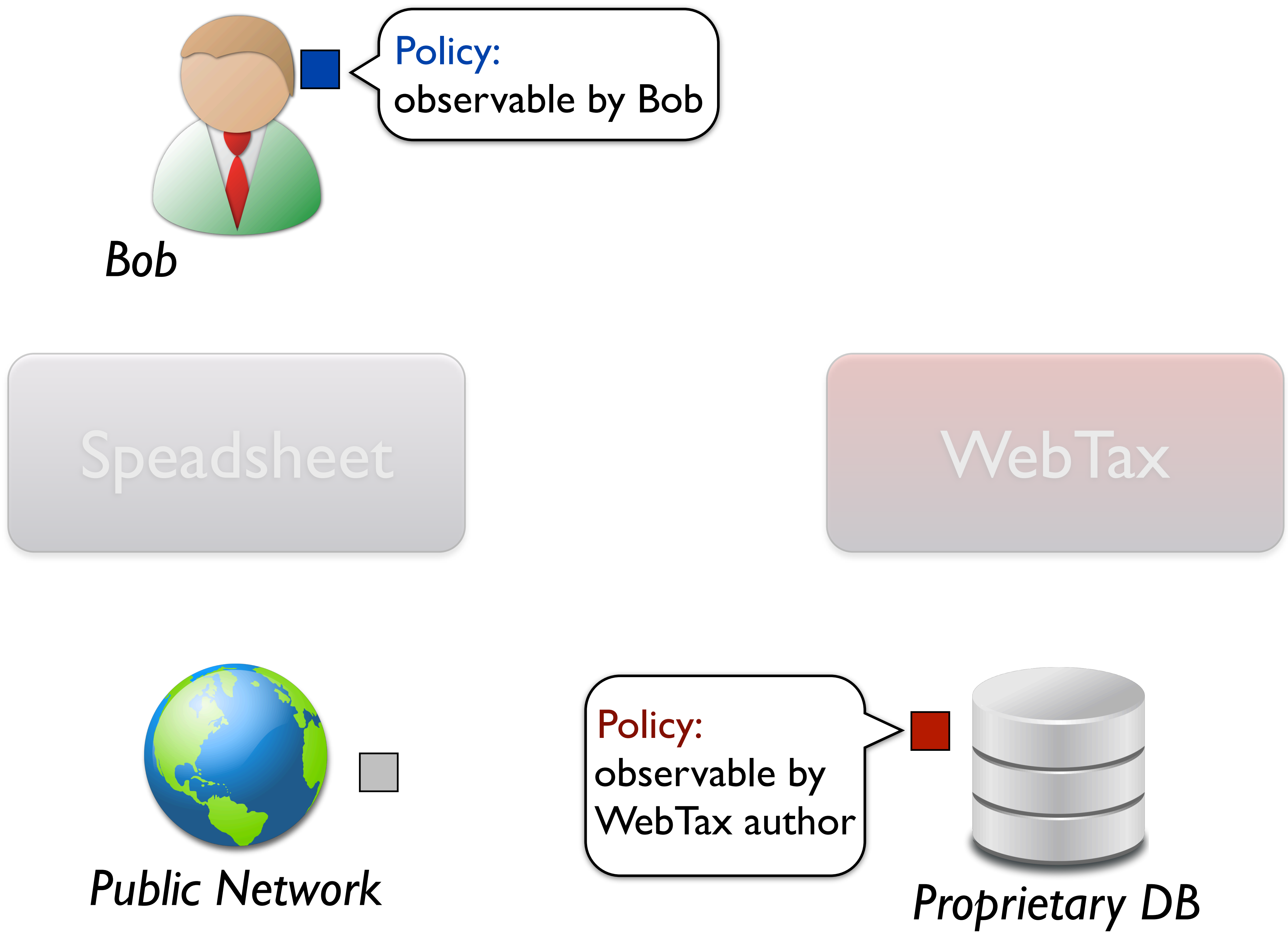
- Components S and I are formulas over *principals*
  - Components impose restrictions on data flow
- Principal is a source of authority (e.g., Bob)
- Restrictions:
  - S and I are minimal (sorted) formulas in CNF
  - Neither S nor I contain negated terms

DC Labels

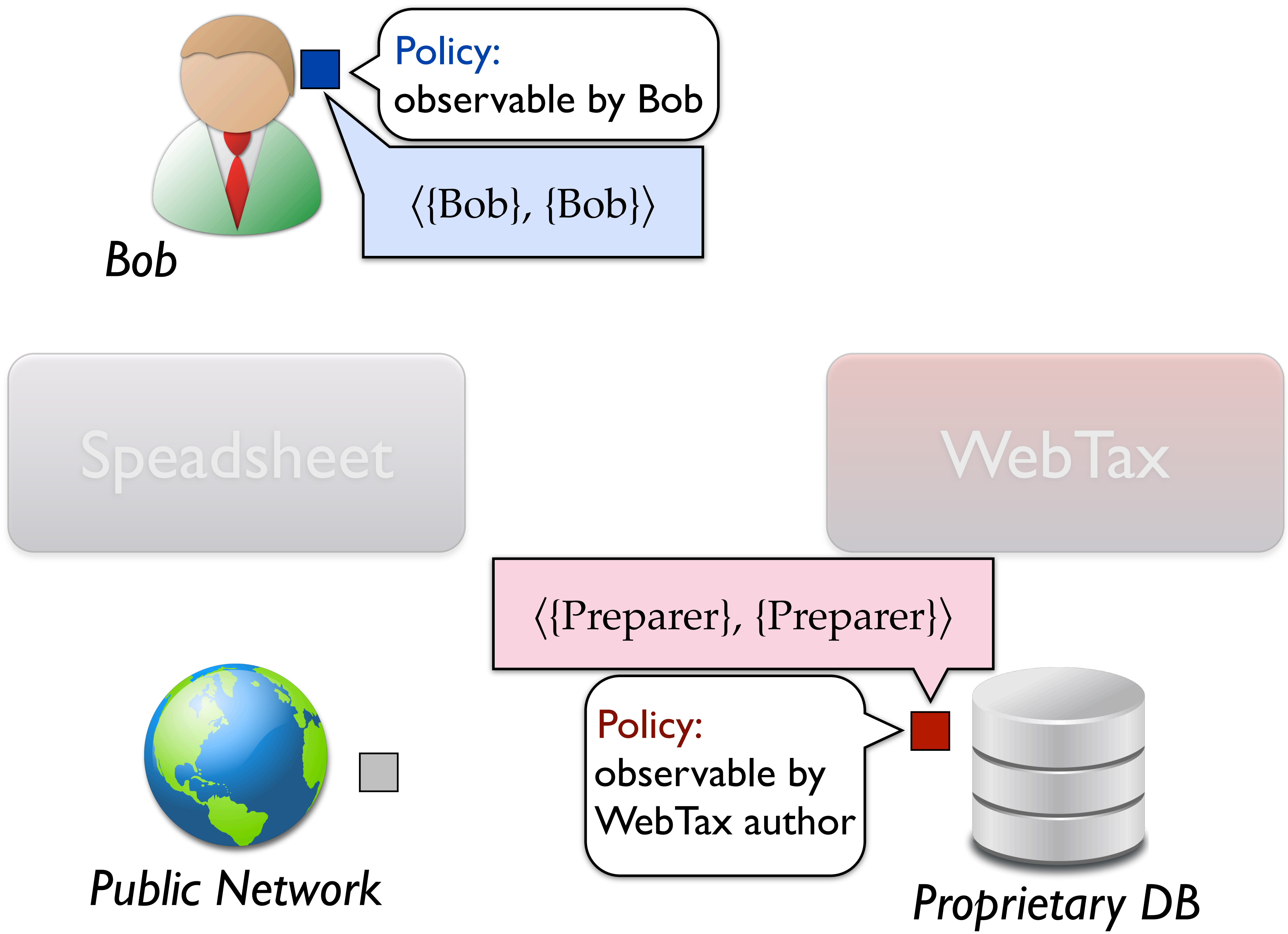
$$\langle S, I \rangle$$

- Secrecy component *S:*
  - ➤ Specifies principals allowed or whose consent is necessary to observe the data
- Integrity component *I*:
  - ➤ Specifies principals that created or are allowed to modify the data

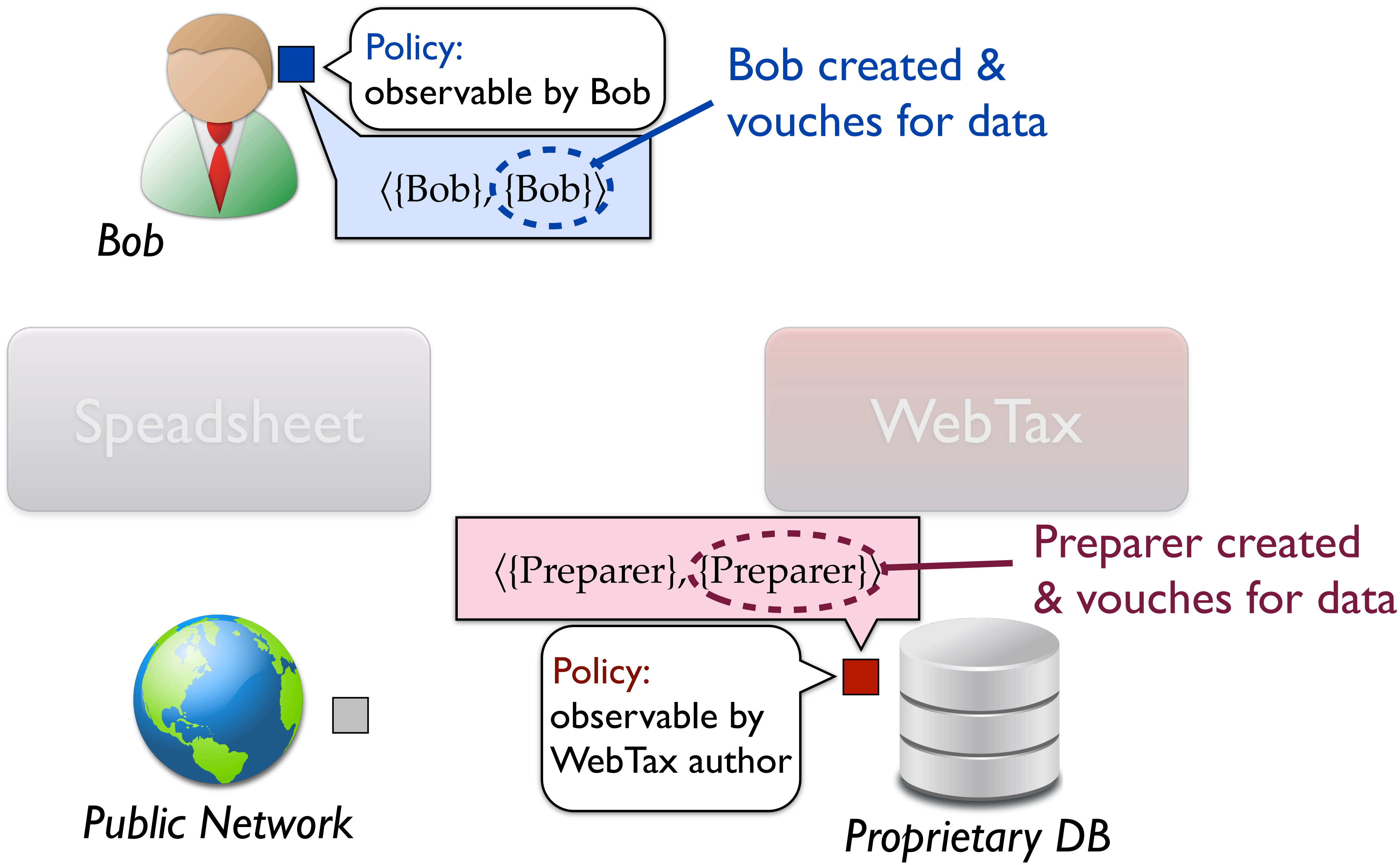
Example with DC Labels



Example with DC Labels



Example with DC Labels



A more interesting label

$$\langle \{(\mathrm{Bob} \vee \mathrm{Alice}) \wedge \mathrm{User}\},\ \{\mathrm{Bob} \vee \mathrm{Alice}\} \rangle$$

A more interesting label

Policy:
created/modified
by Bob or Alice

$$\langle\{(\mathrm{Bob}\vee\mathrm{Alice})\wedge\mathrm{User}\},\ \{\mathrm{Bob}\vee\mathrm{Alice}\}\rangle$$

A more interesting label

Policy:
created/modified by Bob or Alice

$$\langle \{(\mathrm{Bob} \vee \mathrm{Alice}) \wedge \mathrm{User}\},\ \{\mathrm{Bob} \vee \mathrm{Alice}\} \rangle$$

Policy I:
observable by Bob or Alice

Policy II:
observable by User (group)

Policy:
observable by Bob or Alice, given the consent the User group (or vice versa)

A more interesting label

“categories”

Policy:
created/modified by Bob or Alice

$$\langle \{(\mathrm{Bob} \vee \mathrm{Alice}) \wedge \mathrm{User}\},\ \{\mathrm{Bob} \vee \mathrm{Alice}\} \rangle$$

Policy I:
observable by Bob or Alice

Policy II:
observable by User (group)

Policy:
observable by Bob or Alice, given the consent the User group (or vice versa)

General observations

- **Secrecy:** $\{(A \vee B) \wedge C \wedge \cdots\}$
  - Disjunction ➠ allows more readers
  - Conjunction ➠ more restrictions ∴ more secret
- **Integrity:** $\{(A \vee B) \wedge C \wedge \cdots\}$
  - Disjunction ➠ allows more writers
  - Conjunction ➠ more restrictions ∴ trustworthy

Enforcing IFC

- Data may flow from one entity to another iff
  - it accumulates more secrecy restrictions
  - it losses integrity restrictions

$$\frac{S_2 \Longrightarrow S_1 \quad I_1 \Longrightarrow I_2}{\langle S_1, I_1 \rangle \sqsubseteq \langle S_2, I_2 \rangle}$$

Enforcing IFC

- Data may flow from one entity to another iff
  - it accumulates more secrecy restrictions
  - it losses integrity restrictions

Principal's whose consent is needed to observe S_2 must include those of S_1

$$\frac{S_2 \Longrightarrow S_1 \quad I_1 \Longrightarrow I_2}{\langle S_1, I_1 \rangle \sqsubseteq \langle S_2, I_2 \rangle}$$

Enforcing IFC

- Data may flow from one entity to another iff
  - it accumulates more secrecy restrictions
  - it losses integrity restrictions

Principal's whose consent is needed to observe S_2 must include those of S_1

$$\frac{S_2 \Longrightarrow S_1 \quad I_1 \Longrightarrow I_2}{\langle S_1, I_1 \rangle \sqsubseteq \langle S_2, I_2 \rangle}$$

Dual of secrecy. I_2 must be less restricting than I_1

Example of label relations

Secrecy

$\langle\{\text{Alice}\vee\text{Bob}\}, \textbf{True}\rangle \longrightarrow \langle\{\text{Alice}\vee\text{Bob}\vee\text{Charlie}\}, \textbf{True}\rangle$

$\langle\{\text{Alice}\vee\text{Bob}\}, \textbf{True}\rangle \longrightarrow \langle\{\text{Alice}\wedge\text{Dan}\}, \textbf{True}\rangle$

$\langle\{\text{Alice}\wedge\text{Bob}\}, \textbf{True}\rangle \longrightarrow \langle\{\text{Alice}\}, \textbf{True}\rangle$

Example of label relations

Secrecy

$\langle\{\text{Alice}\vee\text{Bob}\}, \textbf{True}\rangle \not\longrightarrow \langle\{\text{Alice}\vee\text{Bob}\vee\text{Charlie}\}, \textbf{True}\rangle$

$\langle\{\text{Alice}\vee\text{Bob}\}, \textbf{True}\rangle \longrightarrow \langle\{\text{Alice}\wedge\text{Dan}\}, \textbf{True}\rangle$

$\langle\{\text{Alice}\wedge\text{Bob}\}, \textbf{True}\rangle \longrightarrow \langle\{\text{Alice}\}, \textbf{True}\rangle$

Example of label relations

Secrecy

$\langle\{\text{Alice}\vee\text{Bob}\}, \textbf{True}\rangle \not\longrightarrow \langle\{\text{Alice}\vee\text{Bob}\vee\text{Charlie}\}, \textbf{True}\rangle$

$\langle\{\text{Alice}\vee\text{Bob}\}, \textbf{True}\rangle \xrightarrow{\checkmark} \langle\{\text{Alice}\wedge\text{Dan}\}, \textbf{True}\rangle$

$\langle\{\text{Alice}\wedge\text{Bob}\}, \textbf{True}\rangle \longrightarrow \langle\{\text{Alice}\}, \textbf{True}\rangle$

Example of label relations

Secrecy

$\langle\{\text{Alice}\vee\text{Bob}\}, \textbf{True}\rangle \not\rightarrow \langle\{\text{Alice}\vee\text{Bob}\vee\text{Charlie}\}, \textbf{True}\rangle$

$\langle\{\text{Alice}\vee\text{Bob}\}, \textbf{True}\rangle \overset{\checkmark}{\rightarrow} \langle\{\text{Alice}\wedge\text{Dan}\}, \textbf{True}\rangle$

$\langle\{\text{Alice}\wedge\text{Bob}\}, \textbf{True}\rangle \not\rightarrow \langle\{\text{Alice}\}, \textbf{True}\rangle$

Example of label relations

Integrity

$$\langle \mathbf{True}, \{\text{Alice}\vee\text{Bob}\}\rangle \longrightarrow \langle \mathbf{True}, \{\text{Alice}\vee\text{Bob}\vee\text{Charlie}\}\rangle$$

$$\langle \mathbf{True}, \{\text{Alice}\}\rangle \longrightarrow \langle \mathbf{True}, \{\text{Alice}\vee\text{Bob}\}\rangle$$

$$\langle \mathbf{True}, \{\text{Alice}\}\rangle \longrightarrow \langle \mathbf{True}, \{\text{Alice}\wedge\text{Bob}\}\rangle$$

Example of label relations

Integrity

$$\langle \mathbf{True}, \{\text{Alice}\vee\text{Bob}\}\rangle \longrightarrow \langle \mathbf{True}, \{\text{Alice}\vee\text{Bob}\vee\text{Charlie}\}\rangle$$

$$\langle \mathbf{True}, \{\text{Alice}\}\rangle \longrightarrow \langle \mathbf{True}, \{\text{Alice}\vee\text{Bob}\}\rangle$$

$$\langle \mathbf{True}, \{\text{Alice}\}\rangle \longrightarrow \langle \mathbf{True}, \{\text{Alice}\wedge\text{Bob}\}\rangle$$

Example of label relations

Integrity

$$\langle \textbf{True}, \{\text{Alice} \vee \text{Bob}\}\rangle \longrightarrow \langle \textbf{True}, \{\text{Alice} \vee \text{Bob} \vee \text{Charlie}\}\rangle$$

$$\langle \textbf{True}, \{\text{Alice}\}\rangle \longrightarrow \langle \textbf{True}, \{\text{Alice} \vee \text{Bob}\}\rangle$$

$$\langle \textbf{True}, \{\text{Alice}\}\rangle \longrightarrow \langle \textbf{True}, \{\text{Alice} \wedge \text{Bob}\}\rangle$$

Example of label relations

Integrity

$\langle \textbf{True}, \{\text{Alice} \vee \text{Bob}\} \rangle \rightarrow \langle \textbf{True}, \{\text{Alice} \vee \text{Bob} \vee \text{Charlie}\} \rangle$

$\langle \textbf{True}, \{\text{Alice}\} \rangle \rightarrow \langle \textbf{True}, \{\text{Alice} \vee \text{Bob}\} \rangle$

$\langle \textbf{True}, \{\text{Alice}\} \rangle \not\rightarrow \langle \textbf{True}, \{\text{Alice} \wedge \text{Bob}\} \rangle$

DC Labels form a lattice

- Combining differently labeled data ➠ join $\sqcup$

$$\langle S_1, I_1\rangle \sqcup \langle S_2, I_2\rangle = \langle S_1 \wedge S_2, I_1 \vee I_2\rangle$$

- Writing to differently labeled entities ➠ meet $\sqcap$
  - ➤ Dual of join: $\langle S_1, I_1\rangle \sqcap \langle S_2, I_2\rangle = \langle S_1 \vee S_2, I_1 \wedge I_2\rangle$

DC Labels form a lattice

- Combining differently labeled data ➠ join $\sqcup$

Need consent of principals in
S_1 and S_2 to observe data

$$\langle S_1, I_1\rangle \sqcup \langle S_2, I_2\rangle = \langle S_1 \wedge S_2, I_1 \vee I_2\rangle$$

- Writing to differently labeled entities ➠ meet $\sqcap$
  - Dual of join: $\langle S_1, I_1\rangle \sqcap \langle S_2, I_2\rangle = \langle S_1 \vee S_2, I_1 \wedge I_2\rangle$

DC Labels form a lattice

- Combining differently labeled data ➠ join $\sqcup$

Need consent of principals in S_1 and S_2 to observe data

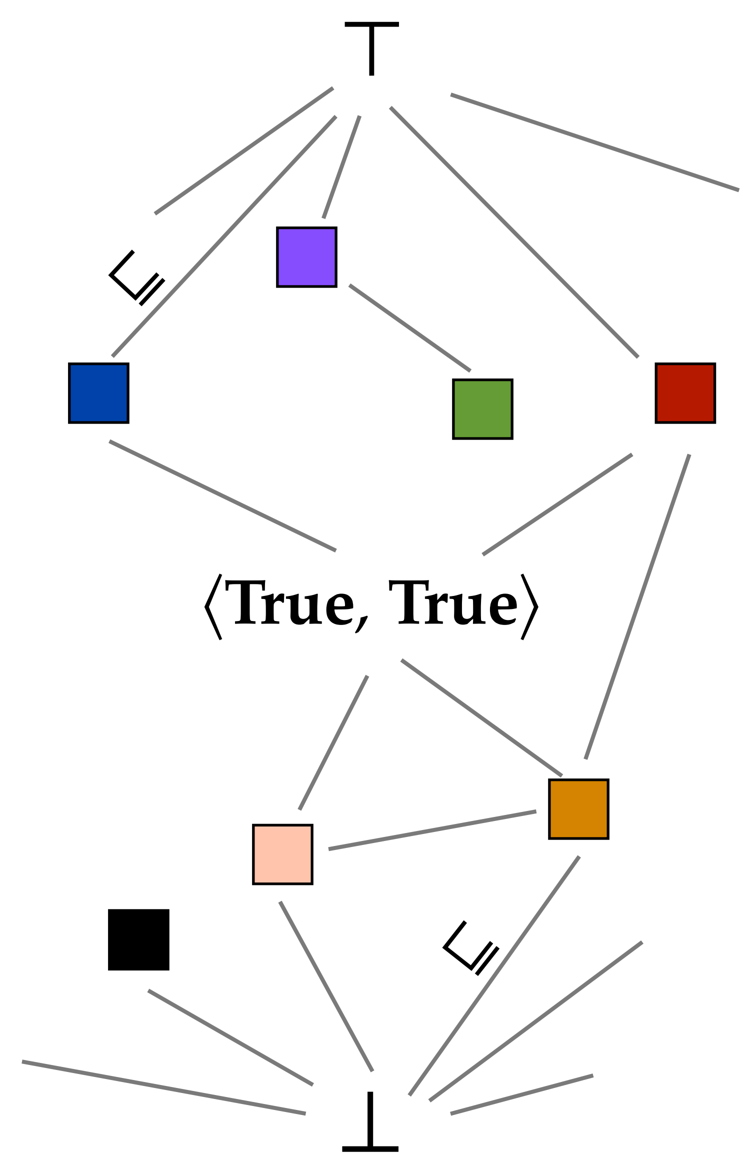
Principals of I_1 or I_2 could have created the data

$$\langle S_1, I_1\rangle \sqcup \langle S_2, I_2\rangle = \langle S_1 \wedge S_2, I_1 \vee I_2\rangle$$

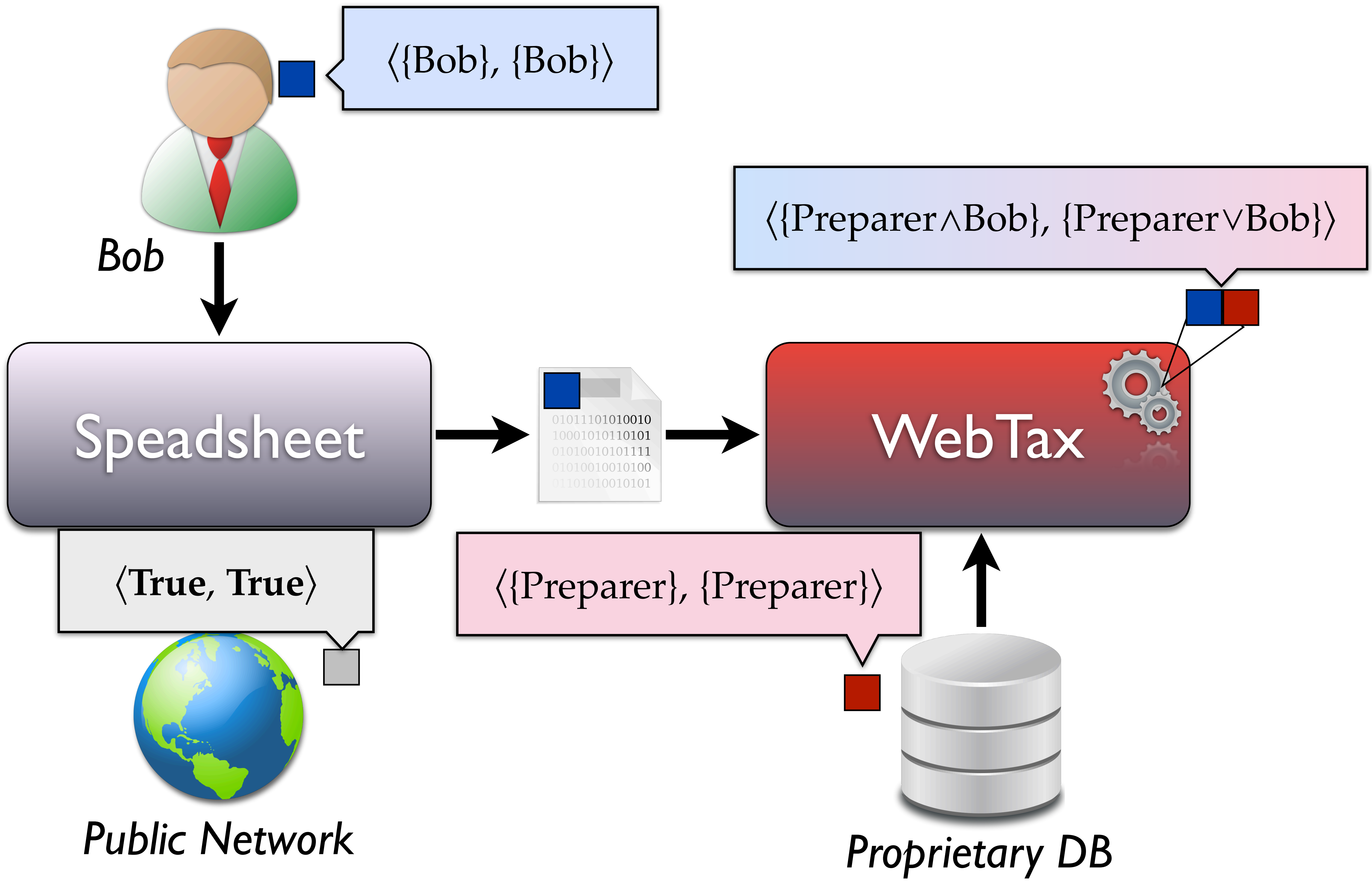
- Writing to differently labeled entities ➠ meet $\sqcap$
  - Dual of join: $\langle S_1, I_1\rangle \sqcap \langle S_2, I_2\rangle = \langle S_1 \vee S_2, I_1 \wedge I_2\rangle$

DC Labels form a lattice

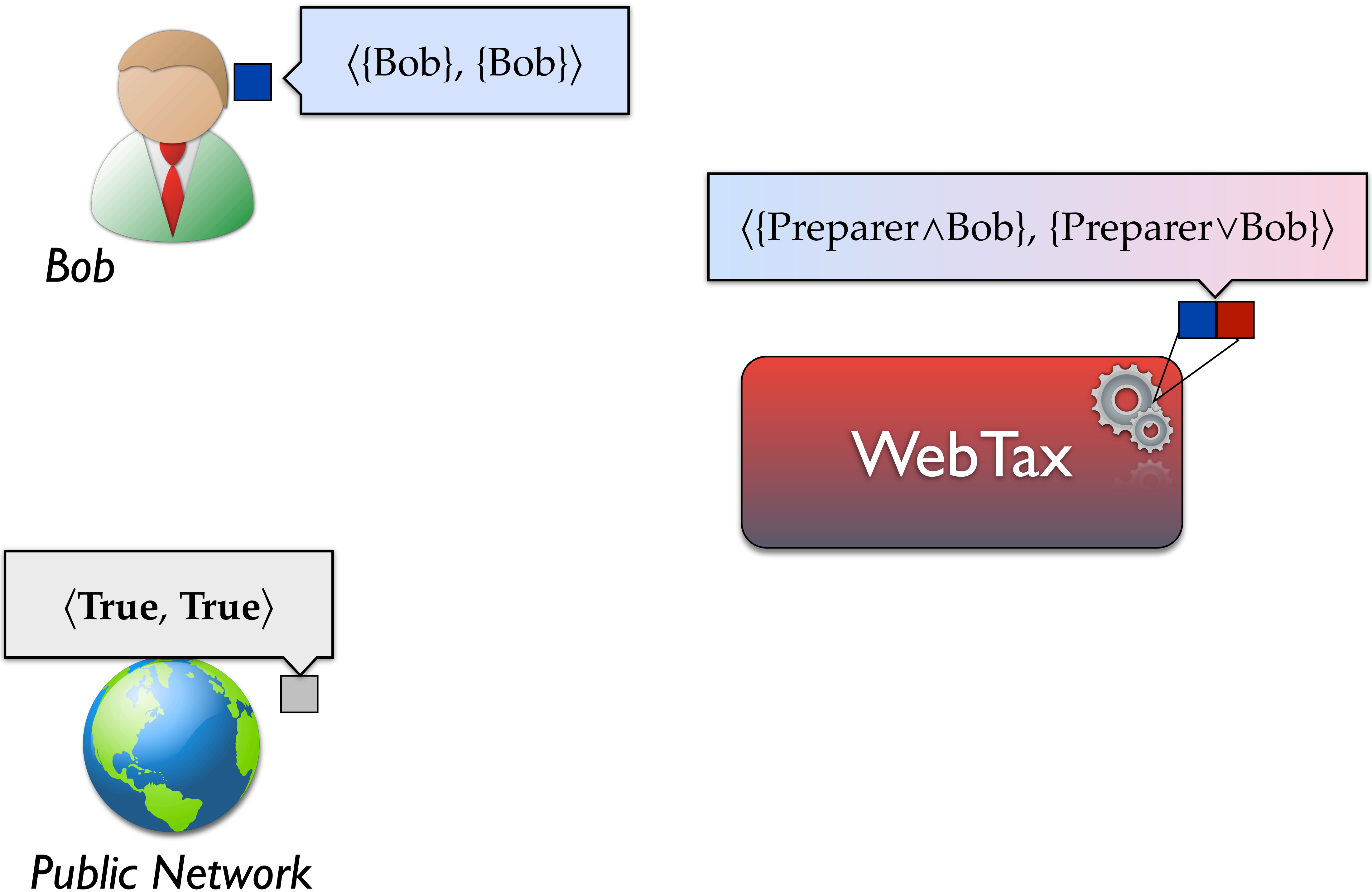
- DC Labels are partially ordered by $\sqsubseteq$ relation
- Have a well-defined join $\sqcup$
- Have a well-defined meet $\sqcap$
- We define top & bottom elements:
  - $\top = \langle$**False, True**$\rangle$
  - $\bot = \langle$**True, False**$\rangle$



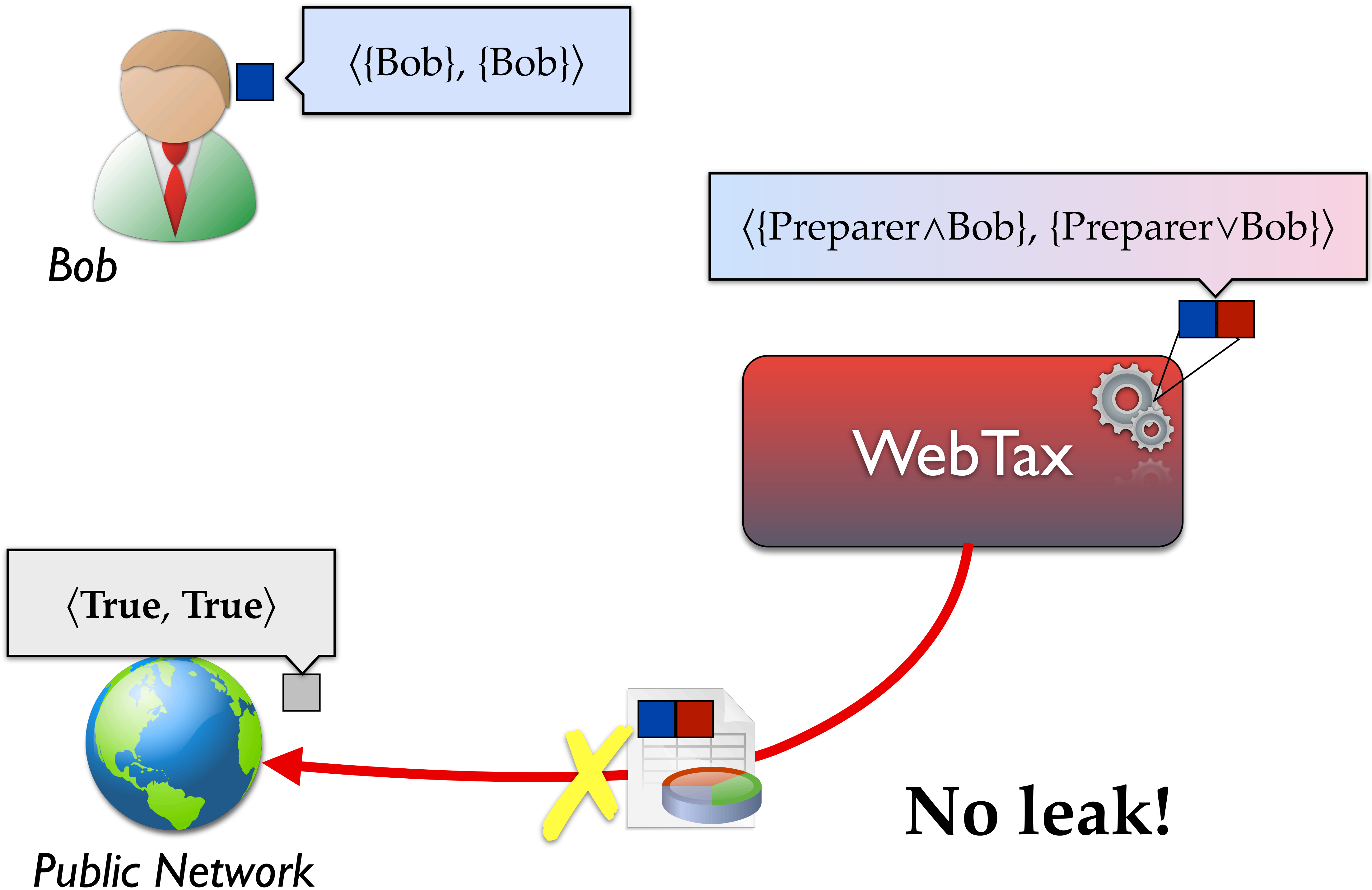
Example with DC Labels



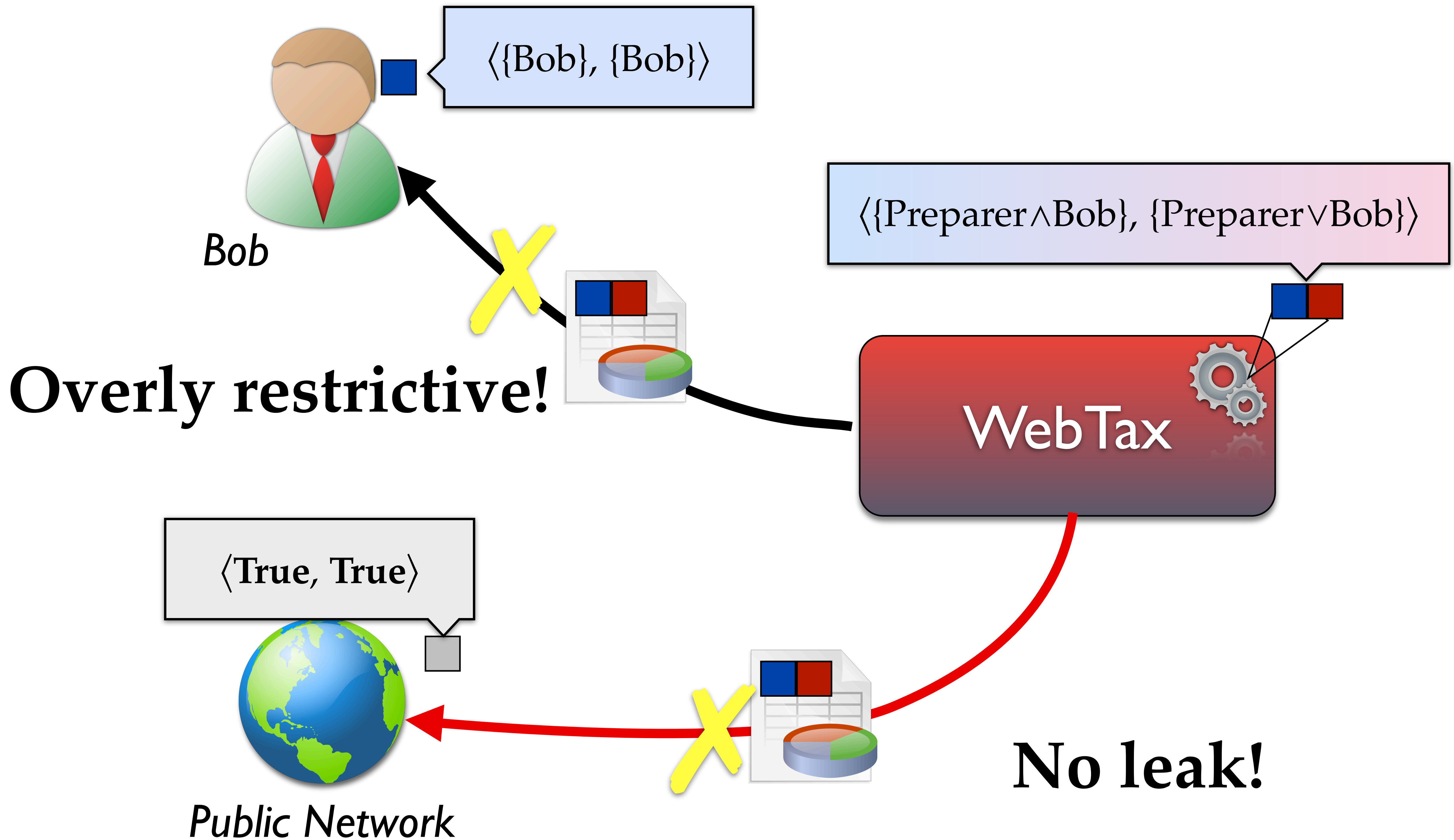
Example with DC Labels



Example with DC Labels



Example with DC Labels



Privileges

- In any practical system need to have method of releasing information
- Mutual-distrustful systems require *declassification*
  - E.g., WebTax needs to declassify data for Bob
- Code running on behalf of principals can exercise *privileges* corresponding to the principals
  - Can declassify & endorse data using $\sqsubseteq_P$ relation

"can-flow-to given privileges p"

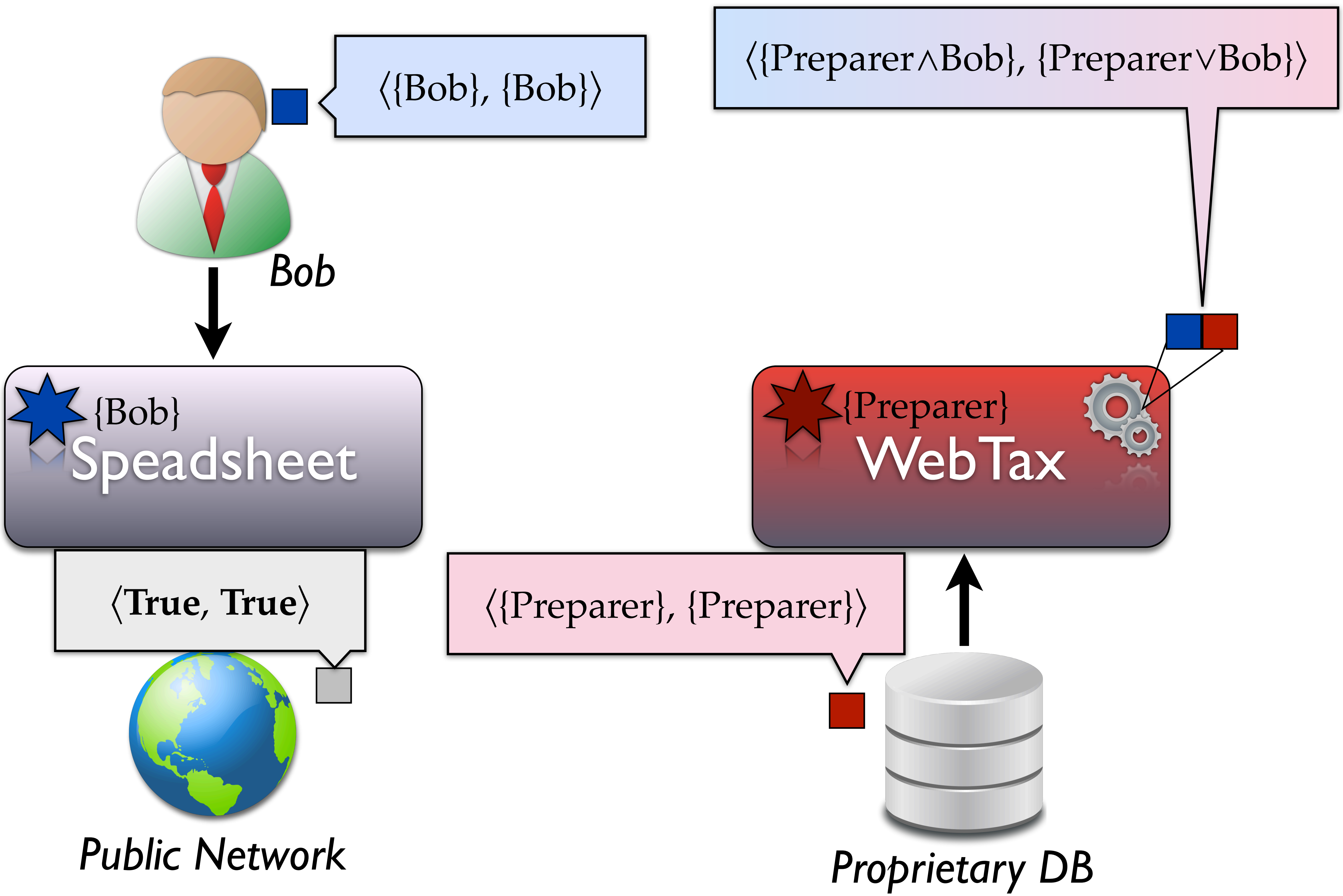
Privileges

- Privileges P are conjunctions of principals

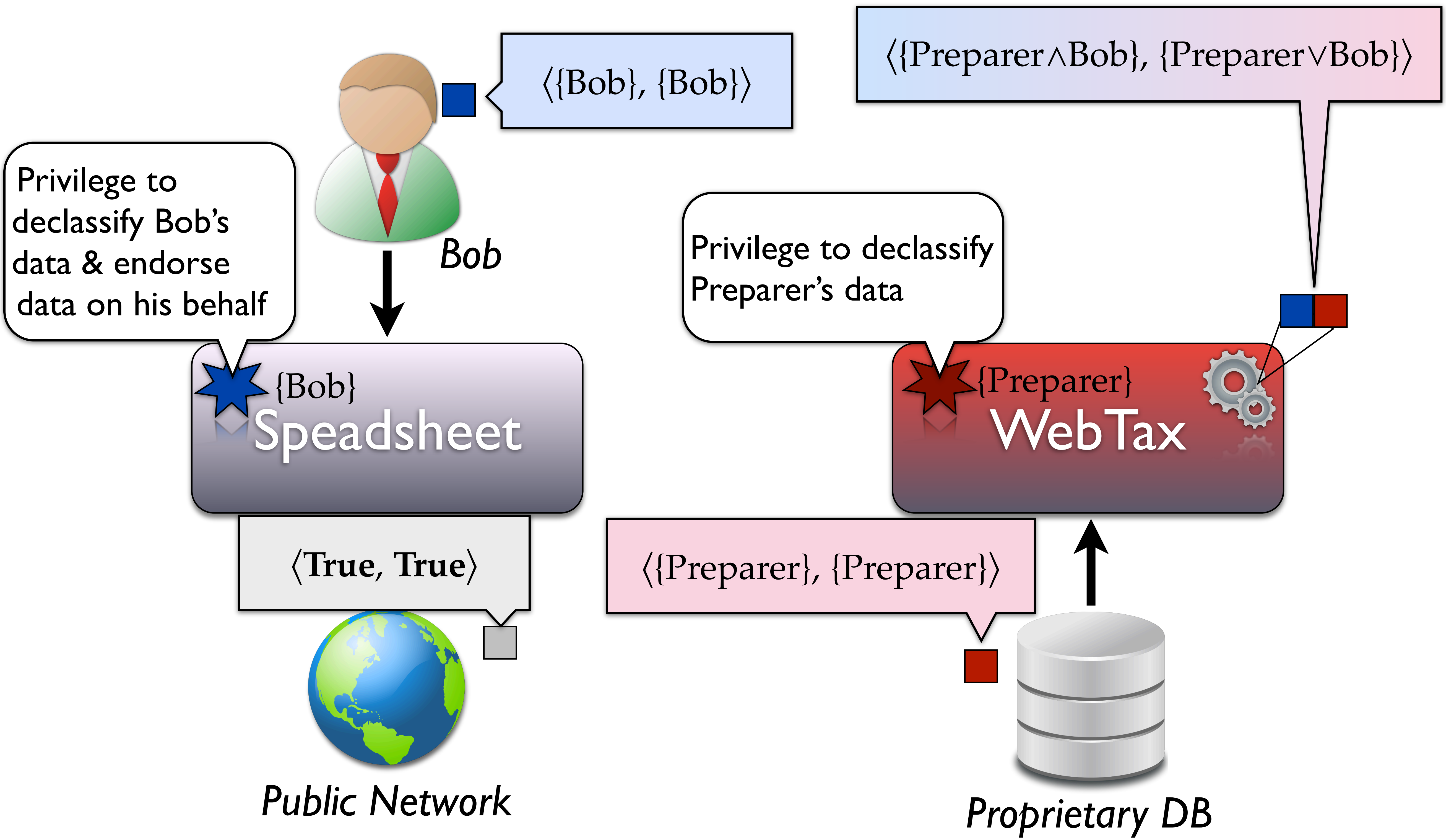
$$\frac{P \wedge S_2 \Longrightarrow S_1 \quad P \wedge I_1 \Longrightarrow I_2}{\langle S_1, I_1 \rangle \sqsubseteq_P \langle S_2, I_2 \rangle}$$

- Code can use privileges P to
  - remove a principal in P from the secrecy component of a label ➠ declassification
  - add a principal in P to an integrity component of a label ➠ endorsement

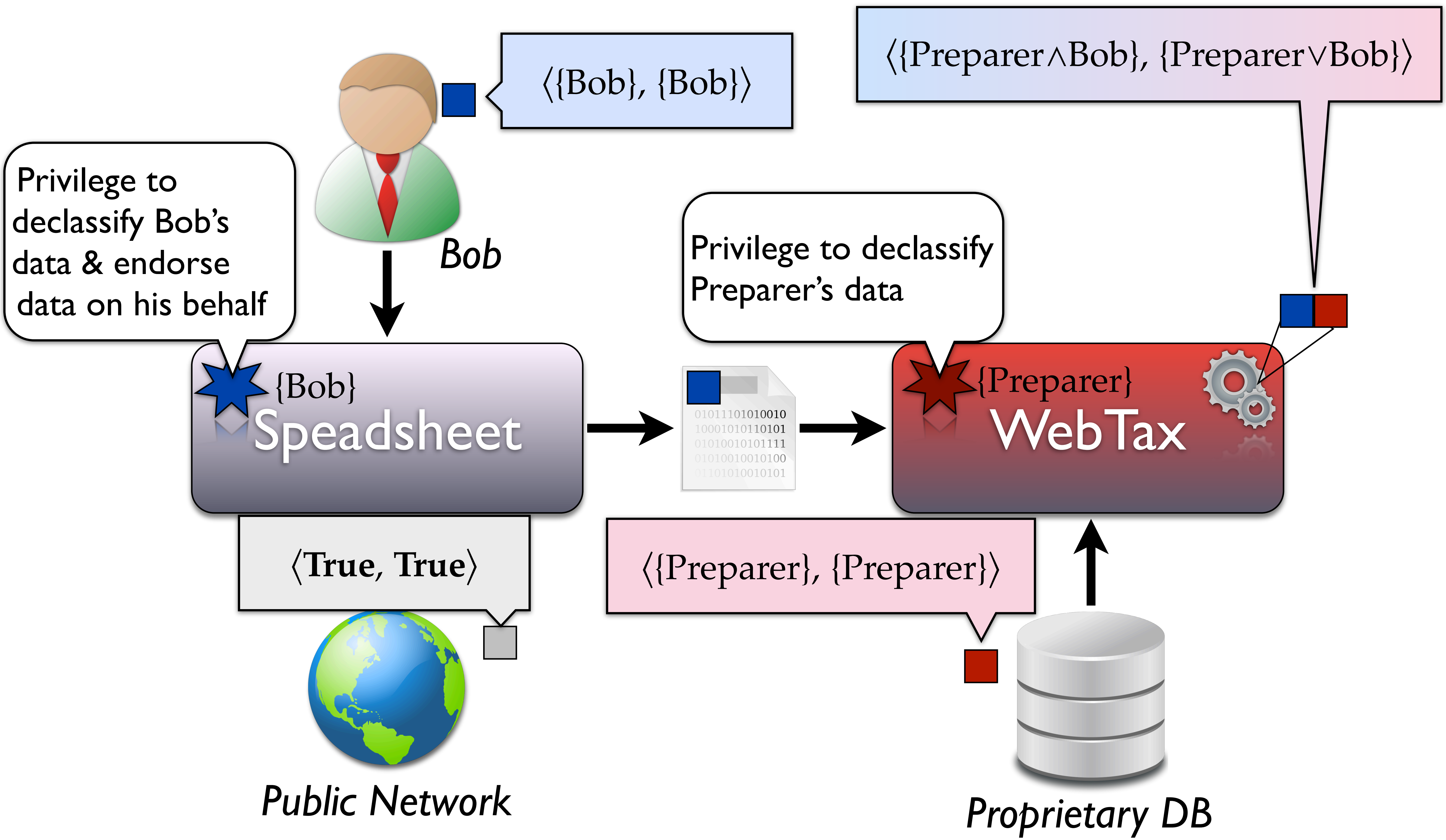
Example with Privileges



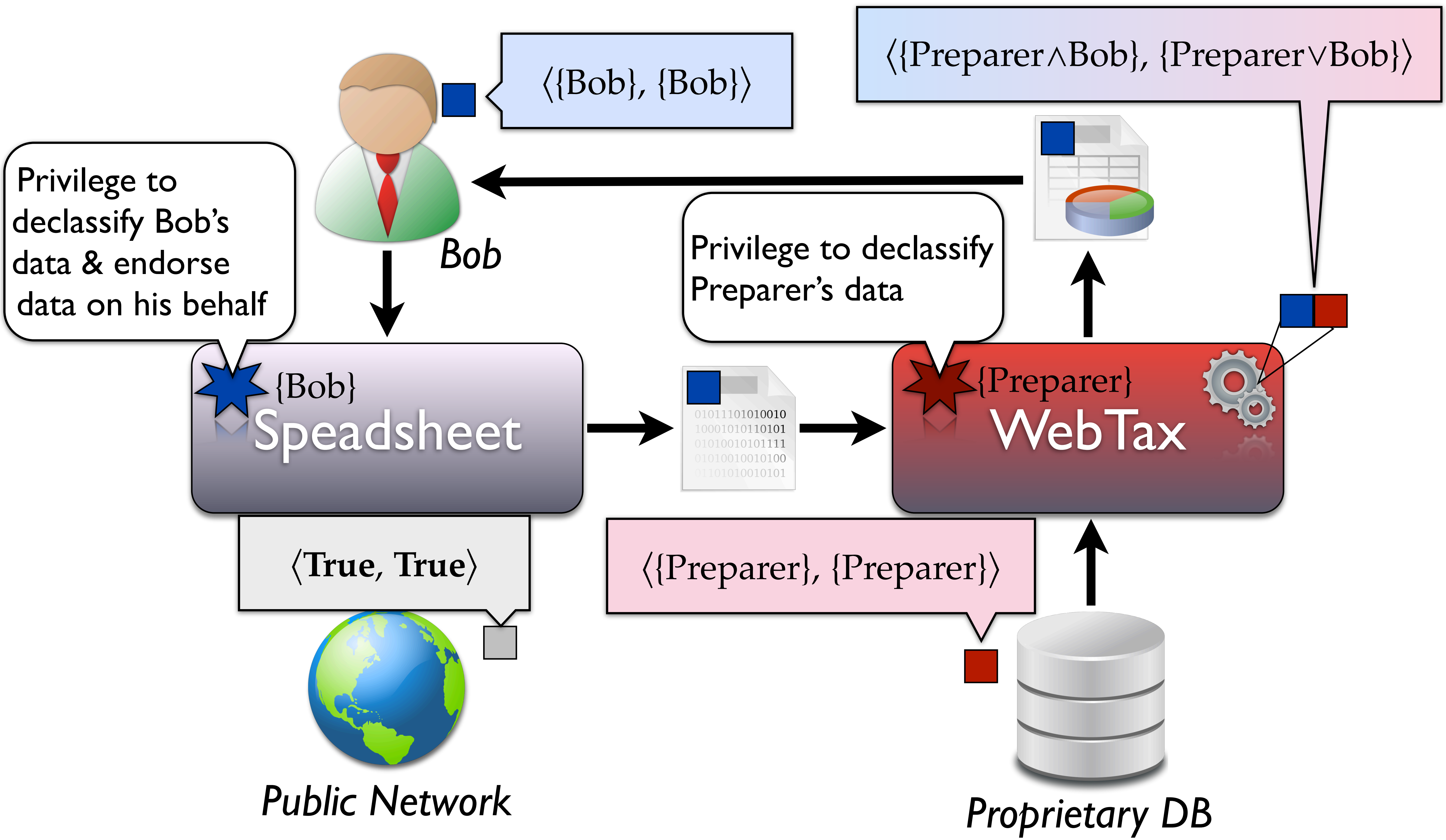
Example with Privileges



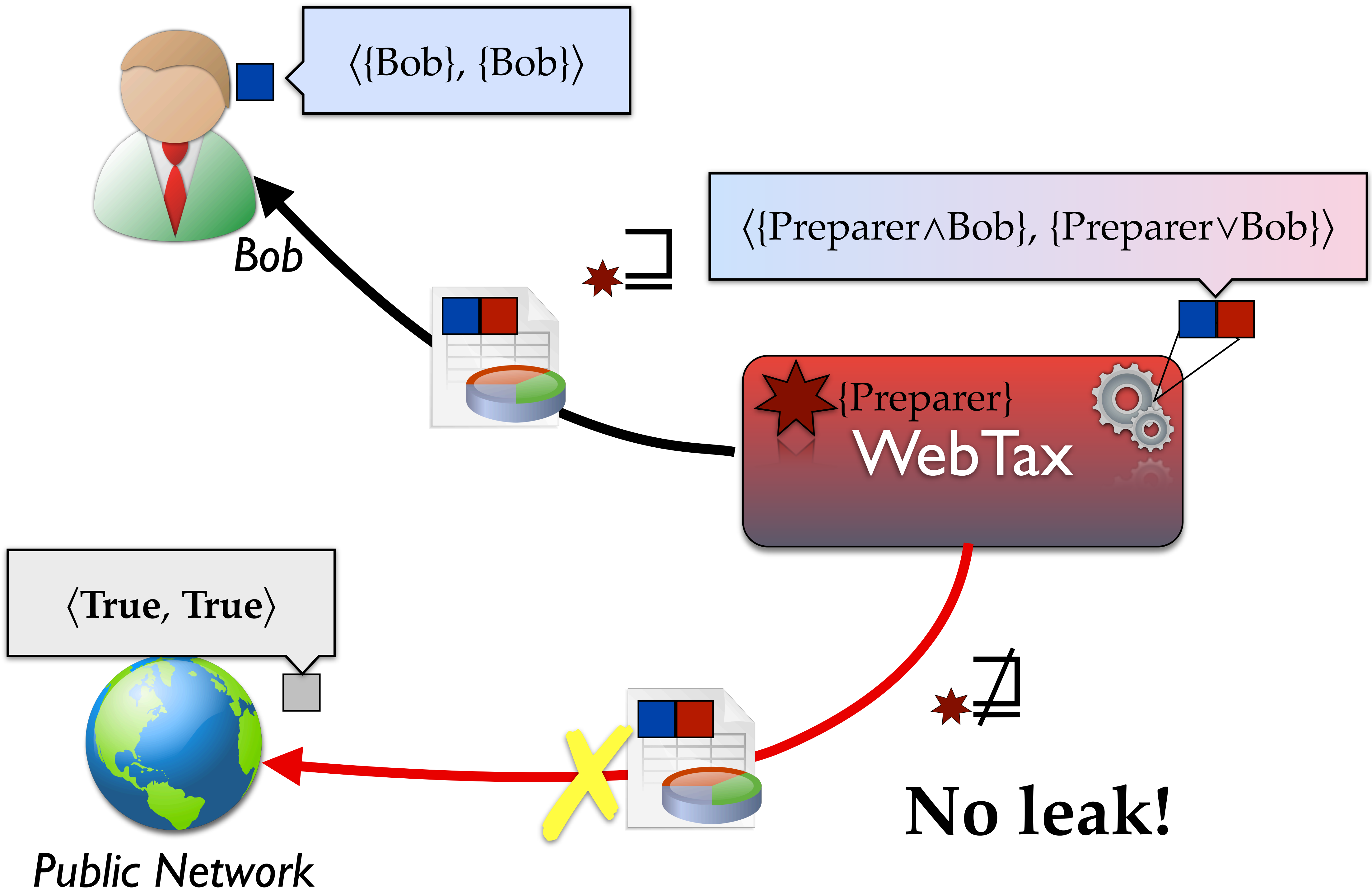
Example with Privileges



Example with Privileges



Example with Privileges



No leak!

Haskell Implementations

- Labels for dynamic IFC systems
  - Principals are strings
  - Categories are sets of principals
  - Components are sets categories
- Labels for static IFC systems
  - Prototype implementation that enforces IFC for secrecy-only DC Labels (a la Curry-Howard) with **no compiler modifications!**

Conclusions

- Presented new label format: DC Labels
  - Formalized using propositional logic
  - Proved several security properties
  - Showed their use in common design patterns
  - Presented two Haskell implementations
- Strength of DC Labels:
  - Model is simple & sound
  - Allows for specifying complex policies
  - Decentralized

Thank you!

```
$> cabal install dclabel
```

www.scs.stanford.edu/~deian/dclabels