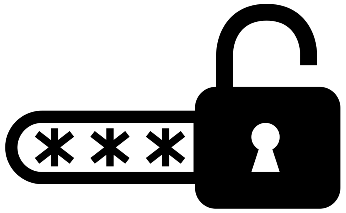
Facets of Information Flow Control

Marco Vassena

CISPA
HELMHOLTZ CENTER FOR INFORMATION SECURITY

Complex Software System

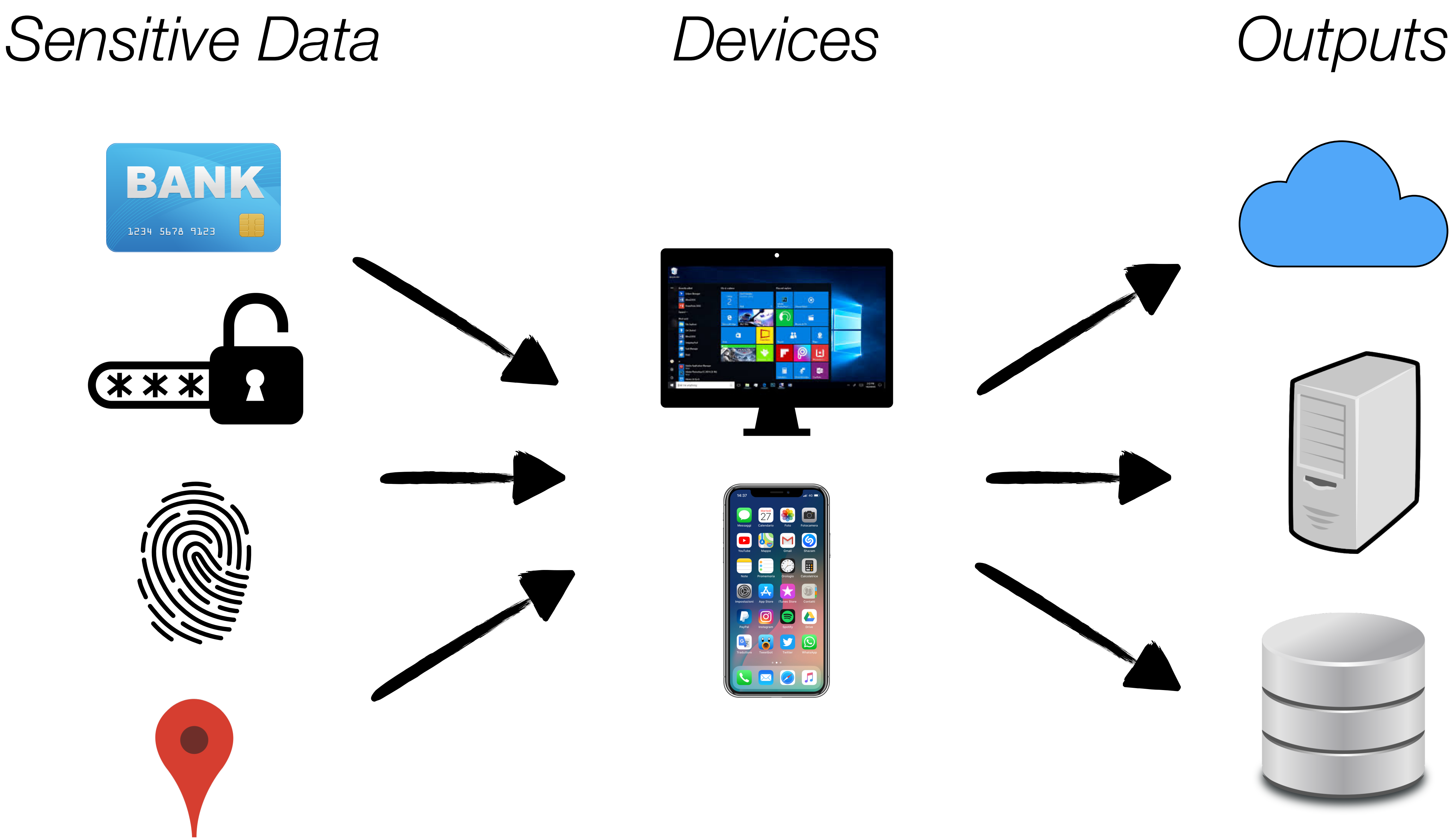
Sensitive Data



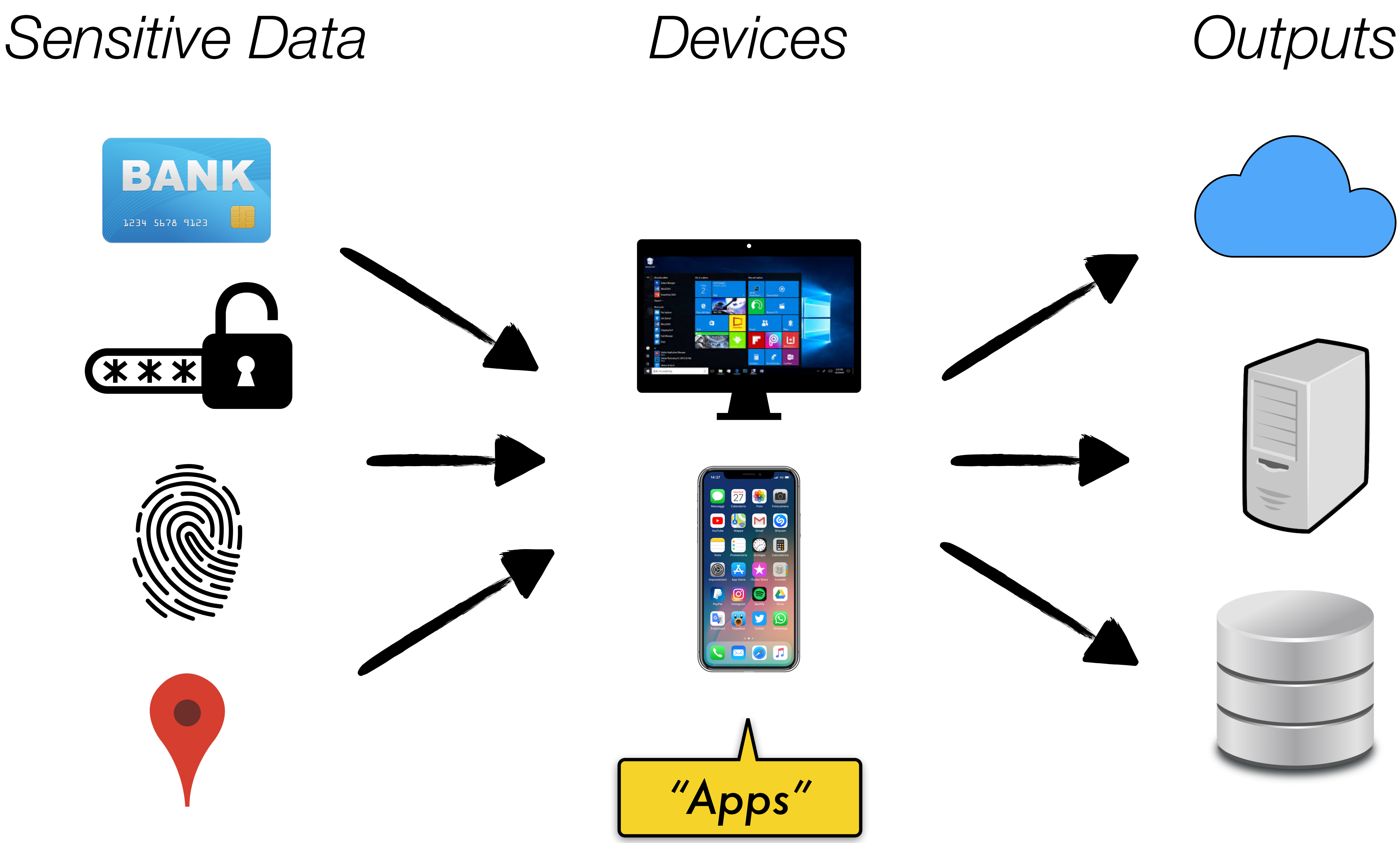
Complex Software System

Sensitive Data *Devices*

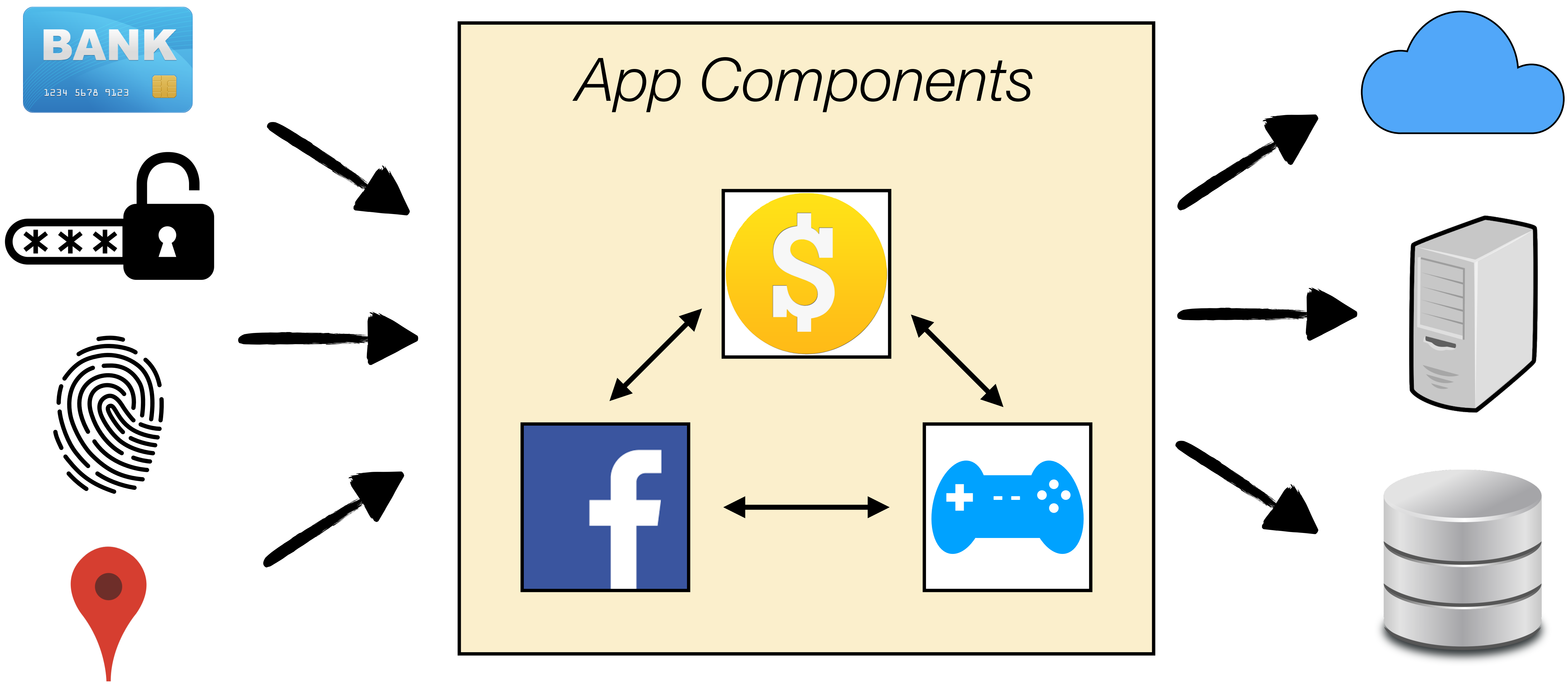
Complex Software System



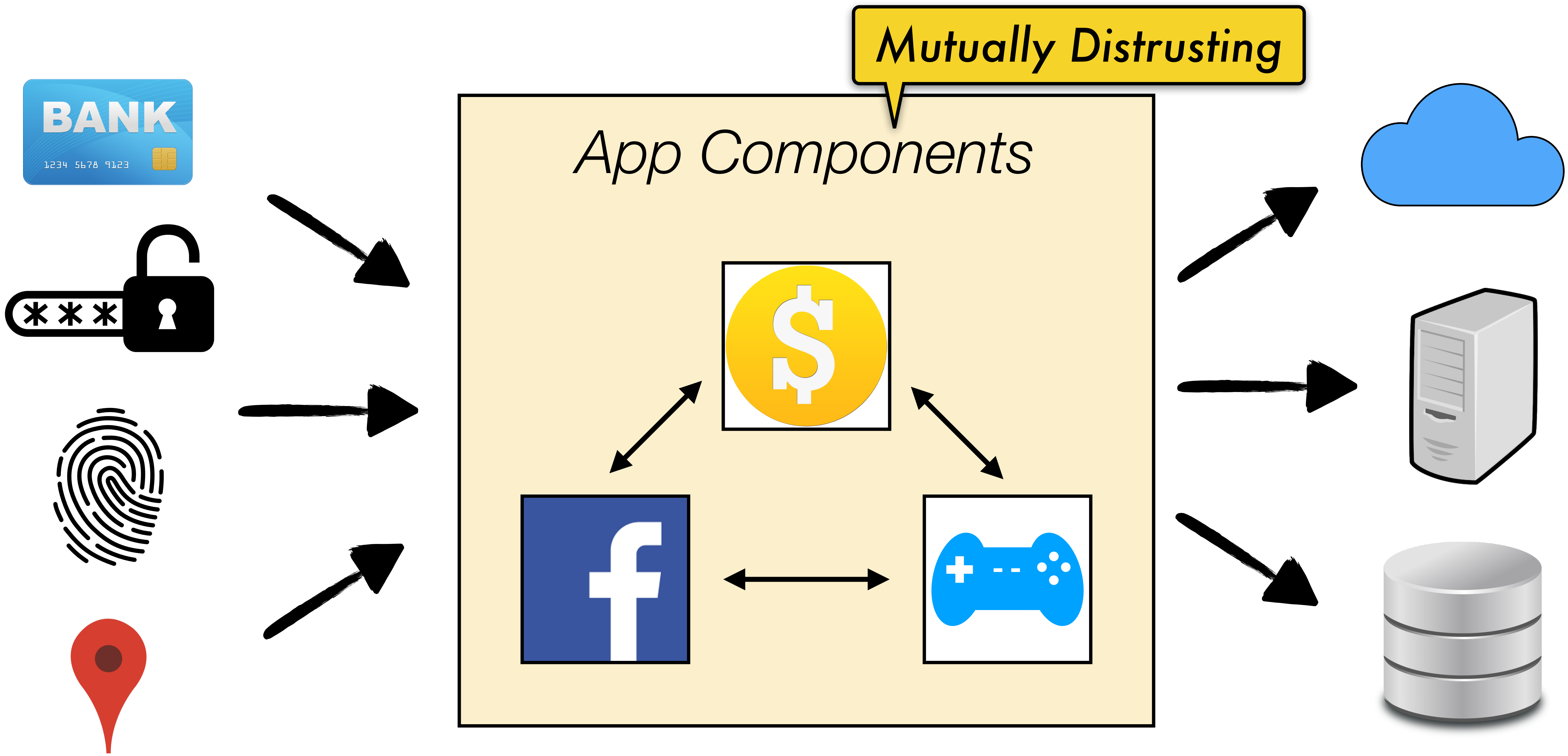
Complex Software System



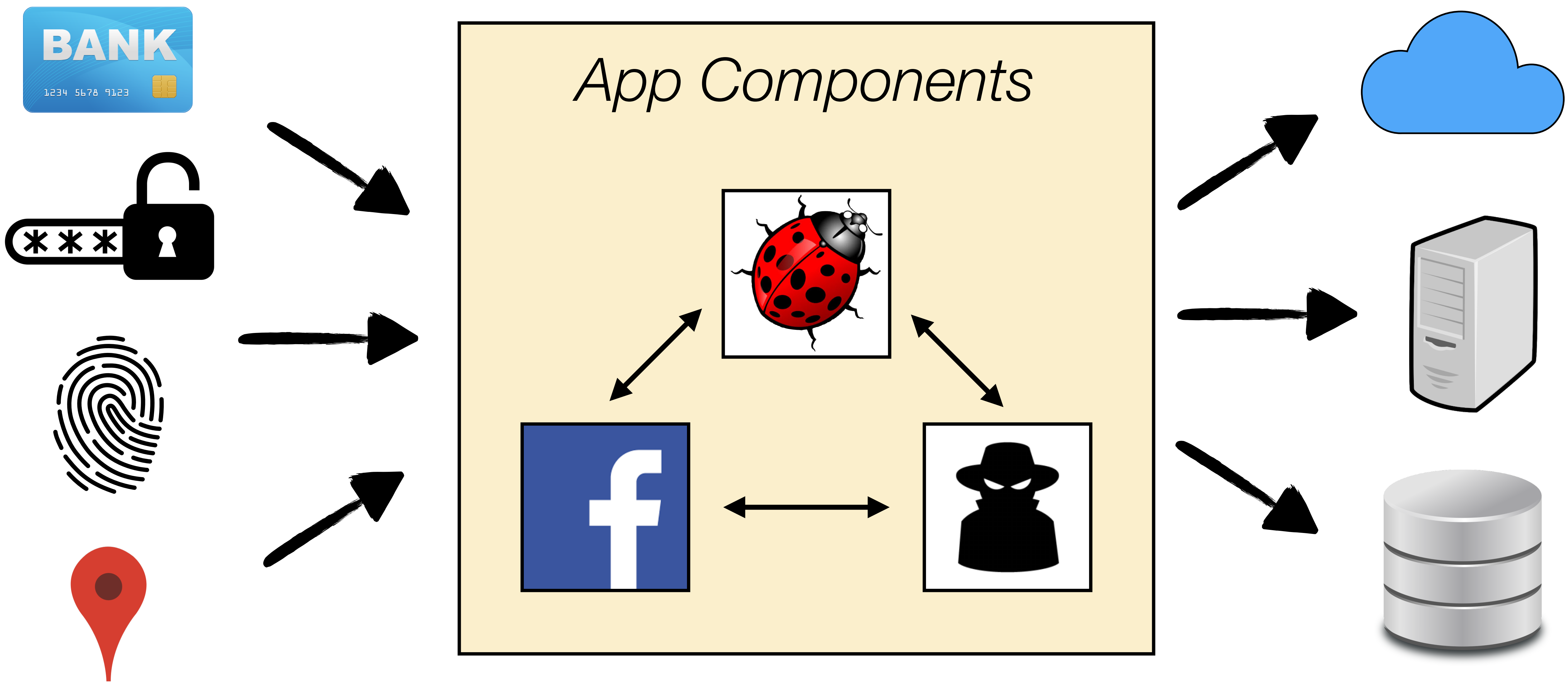
Modern software contains many 3rd party components!



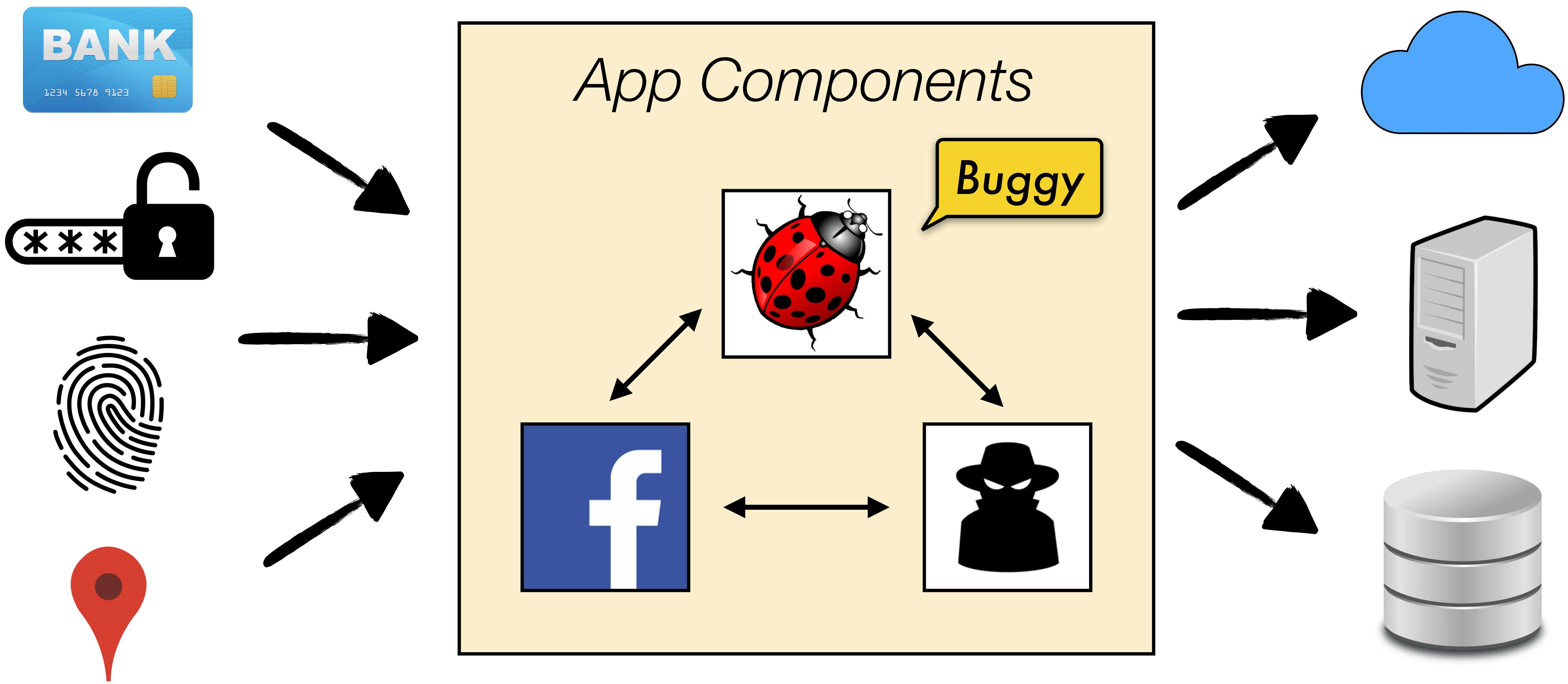
Modern software contains many 3rd party components!



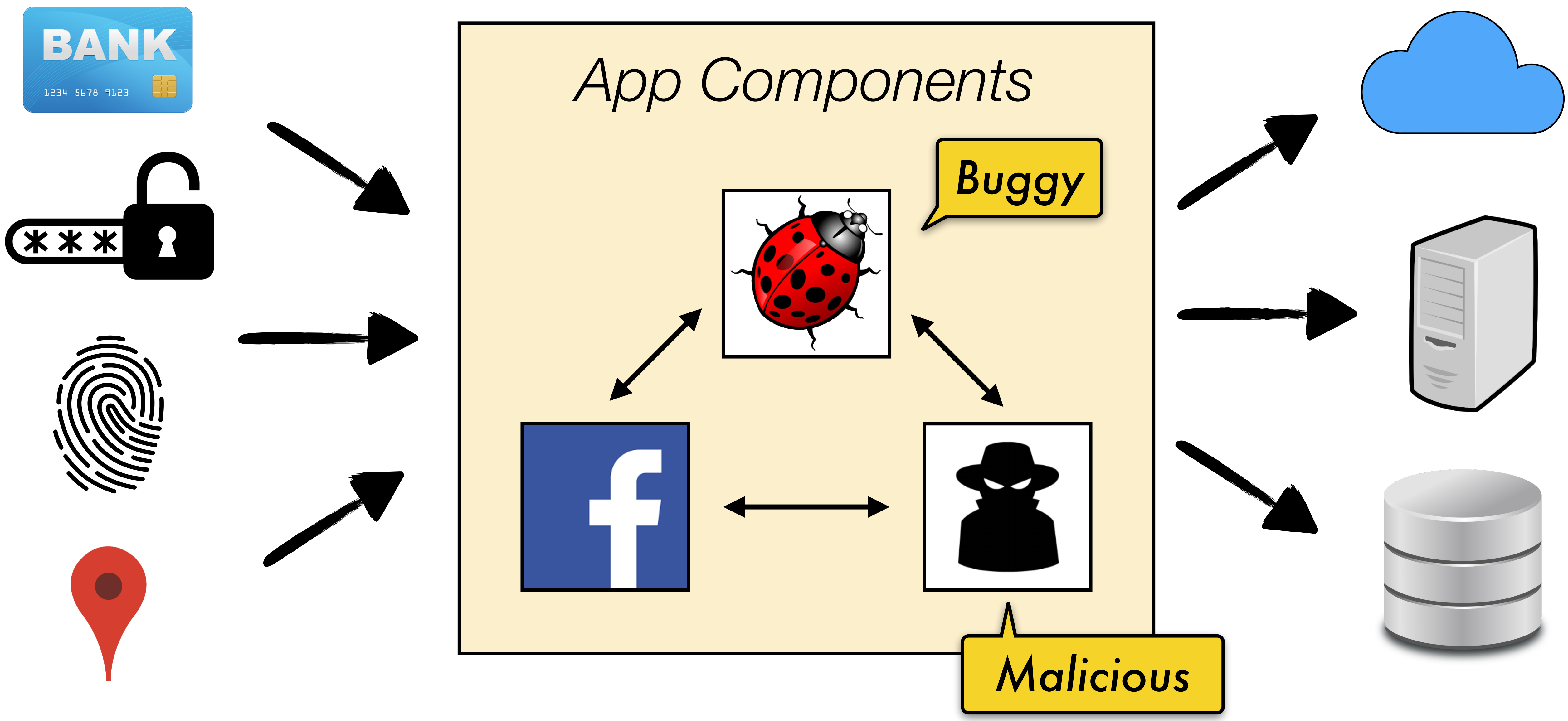
Modern software contains many 3rd party components!



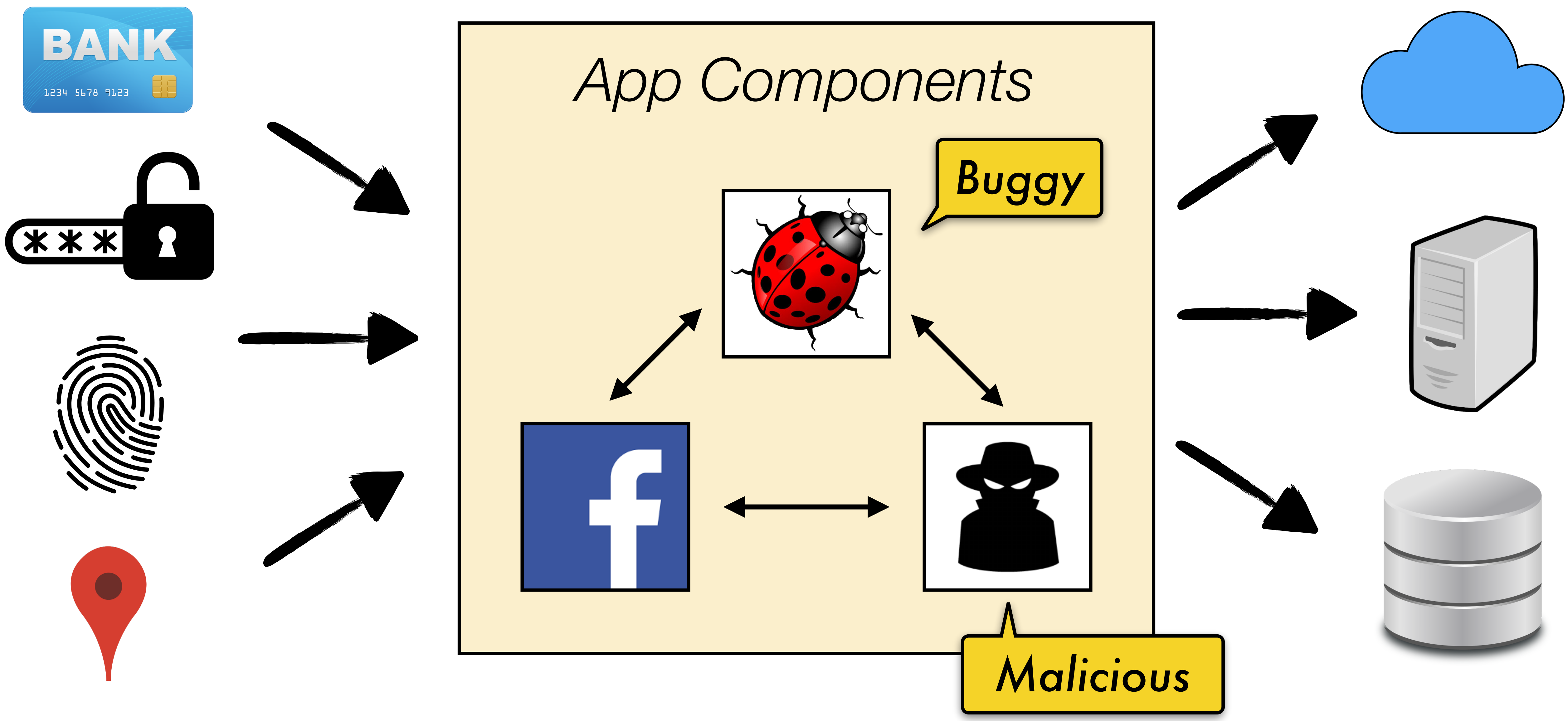
Modern software contains many 3rd party components!



Modern software contains many 3rd party components!

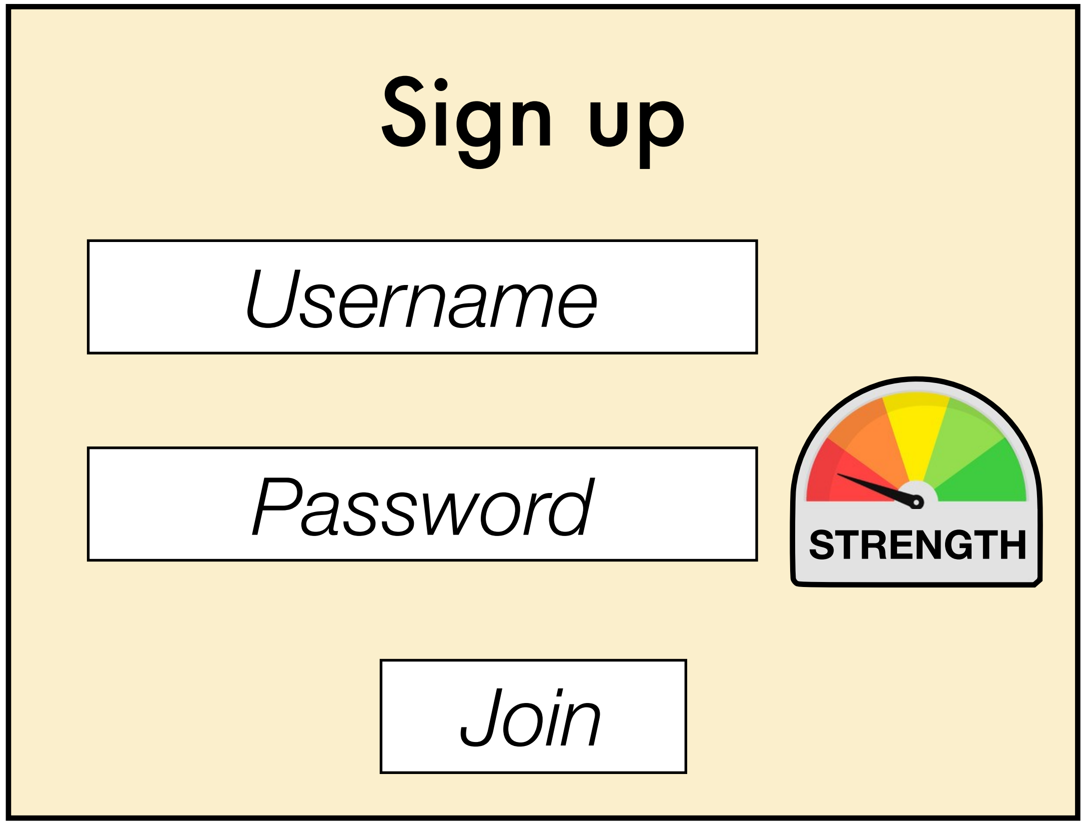


Modern software contains many 3rd party components!

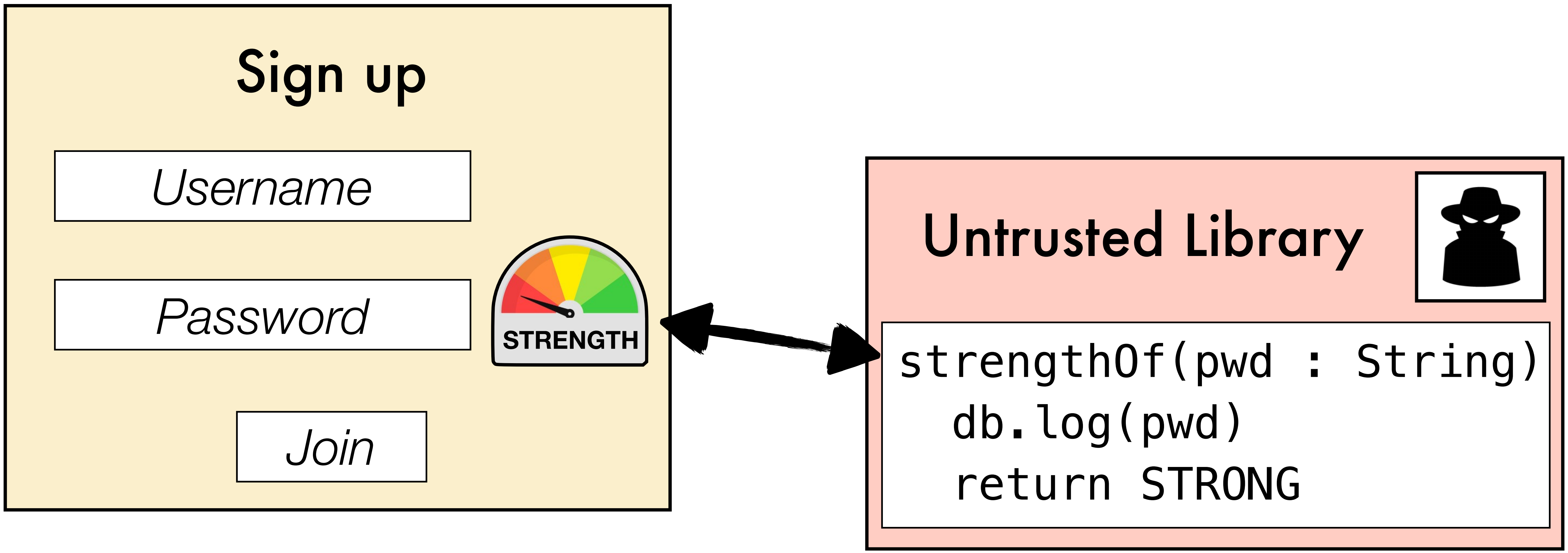


Data ***confidentiality*** *and* ***integrity*** *is at stake*

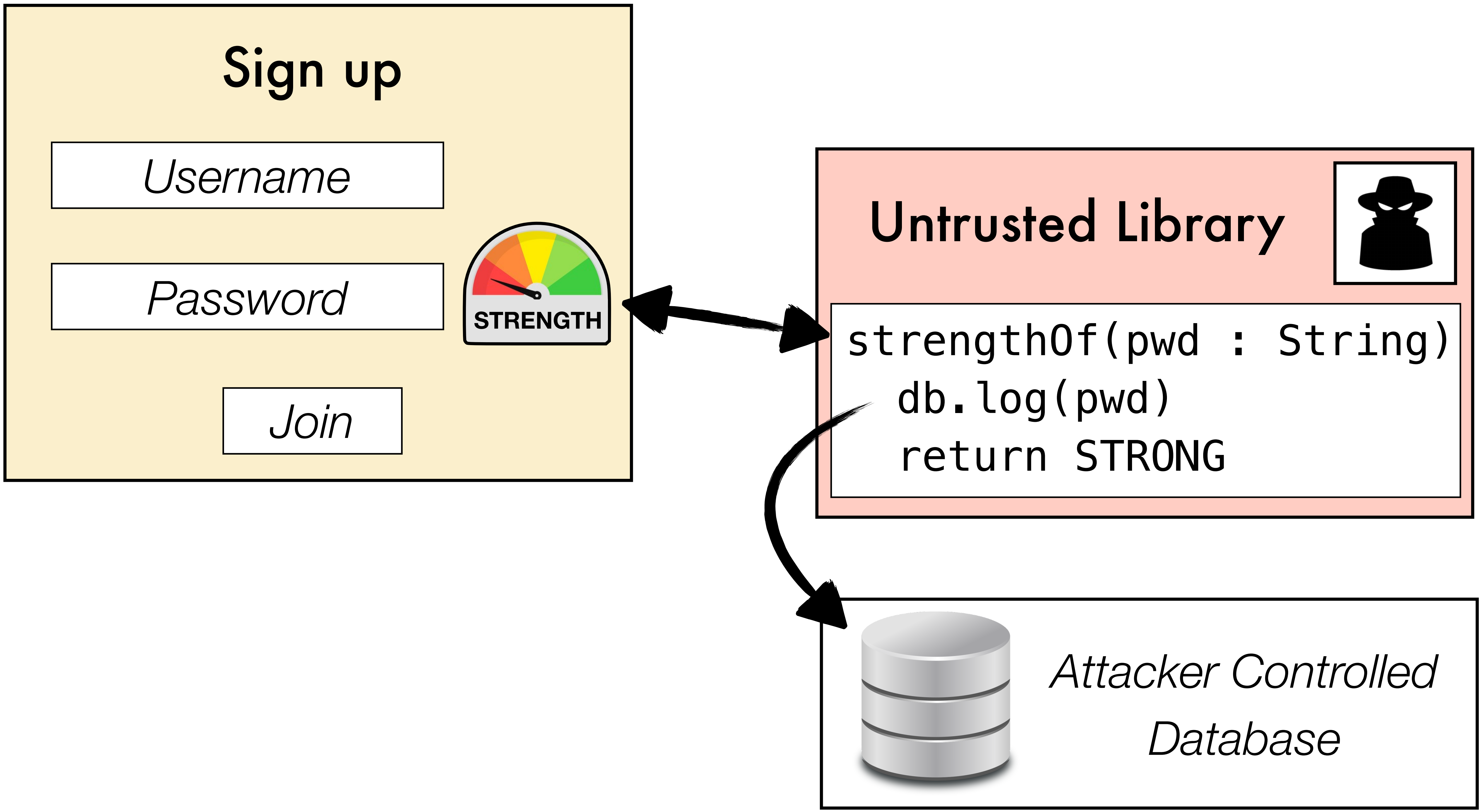
Example



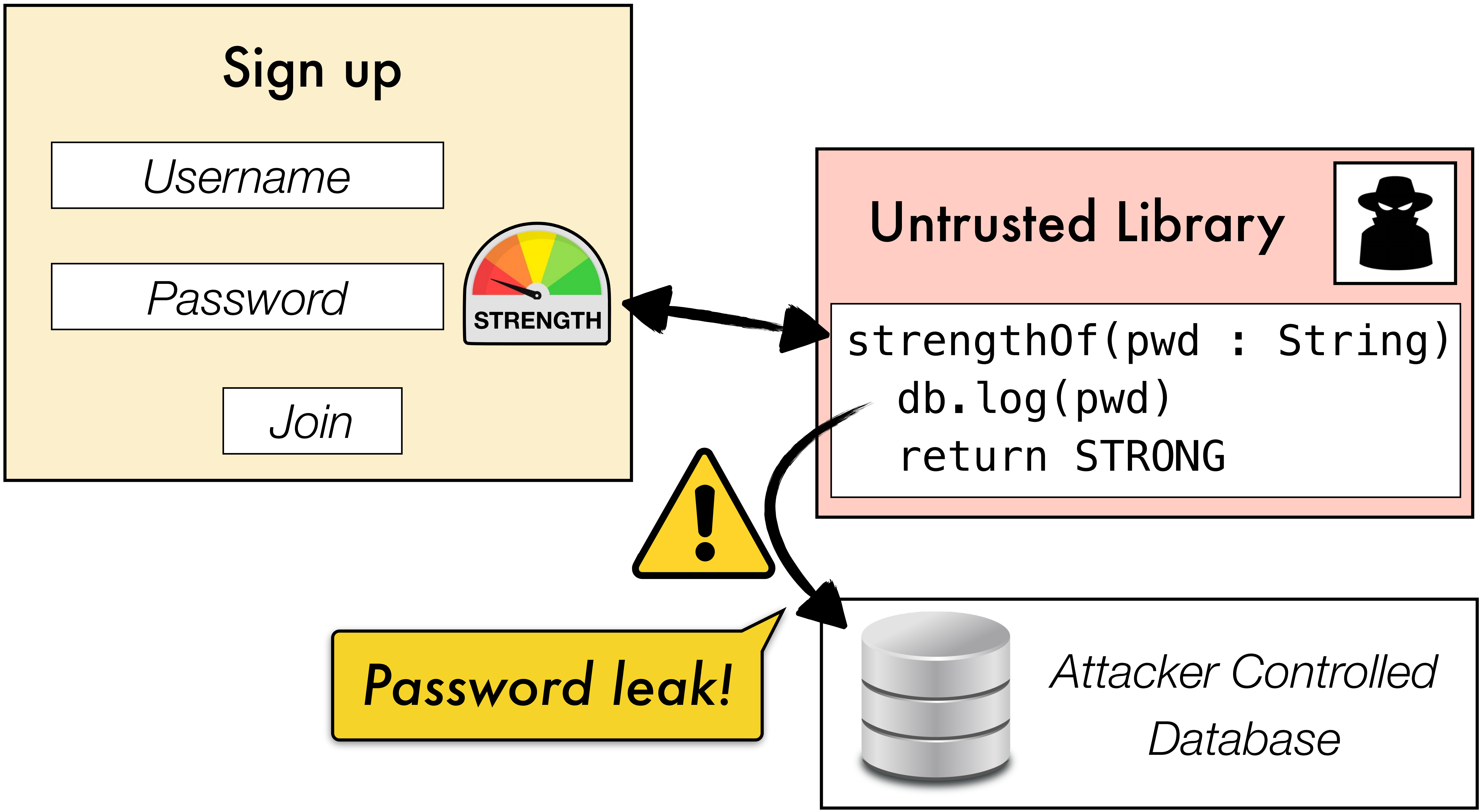
Example


Sign up
Username
Password
STRENGTH
Join
Untrusted Library
strengthOf(pwd : String)
db.log(pwd)
return STRONG

Example

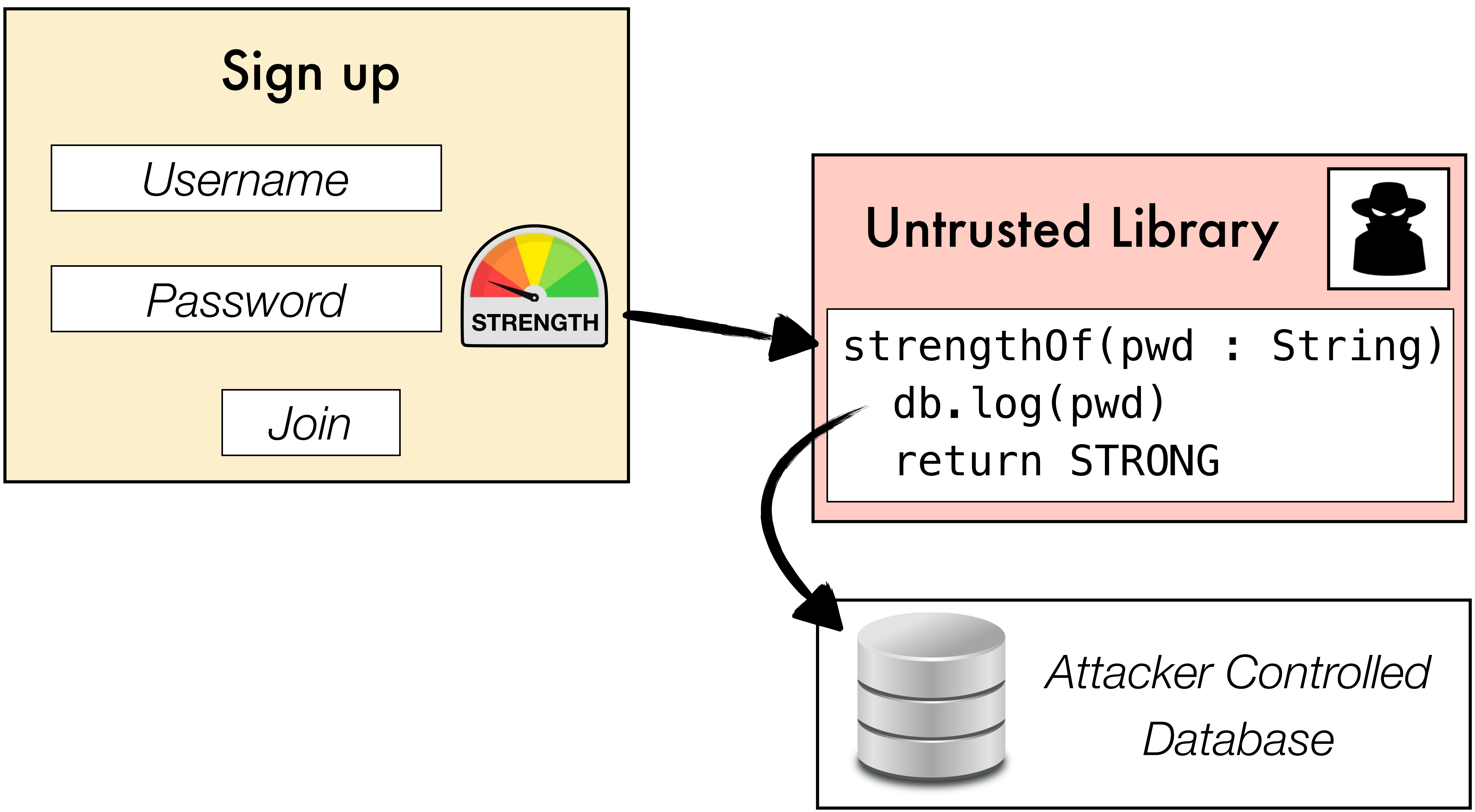

Sign up
Username
Password
STRENGTH
Join
Untrusted Library
strengthOf(pwd : String)
db.log(pwd)
return STRONG
Attacker Controlled Database

Example


Sign up
Username
Password
STRENGTH
Join
Untrusted Library
strengthOf(pwd : String)
db.log(pwd)
return STRONG
Password leak!
Attacker Controlled
Database

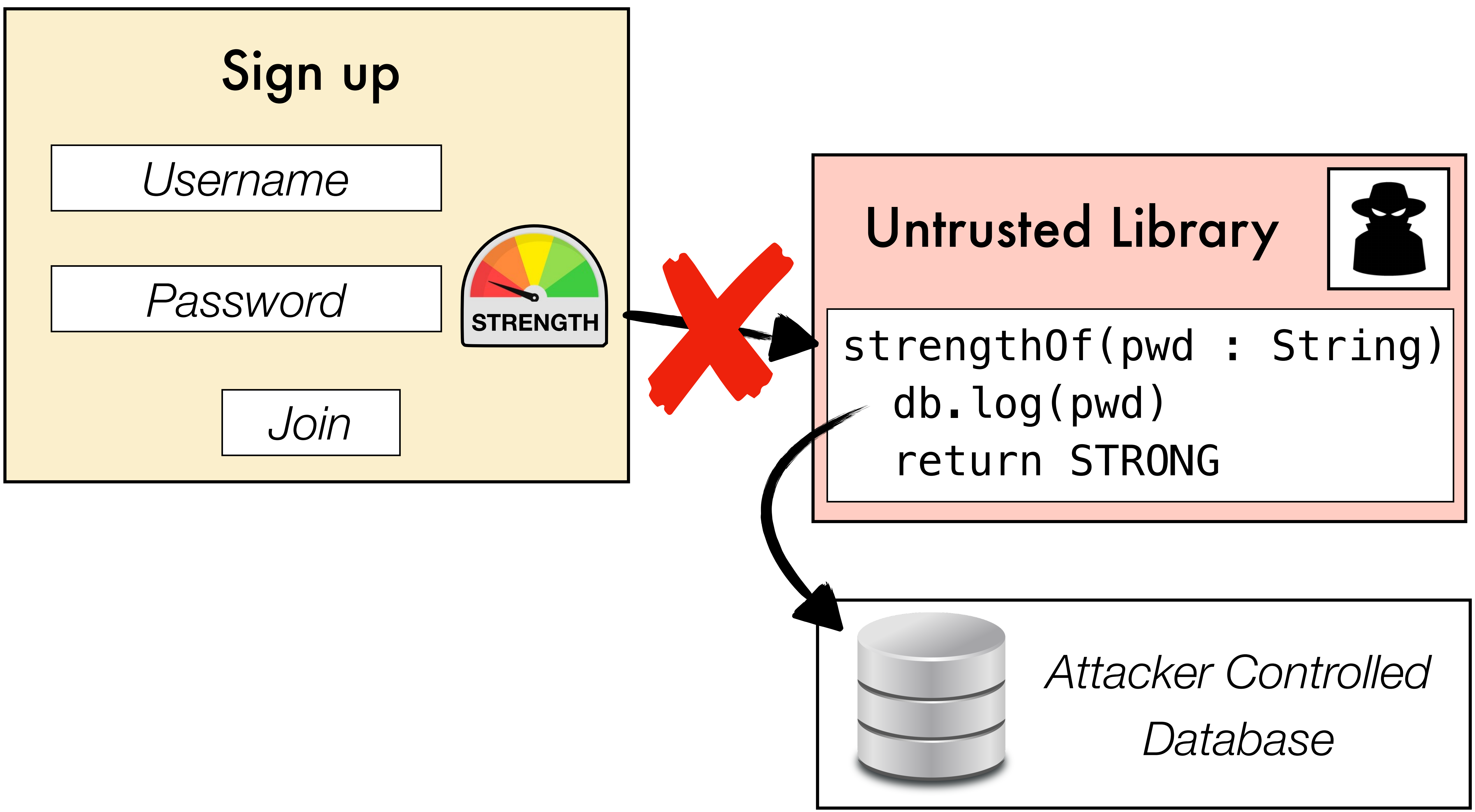
Access Control?

*Restrict **access** to sensitive data in untrusted components*



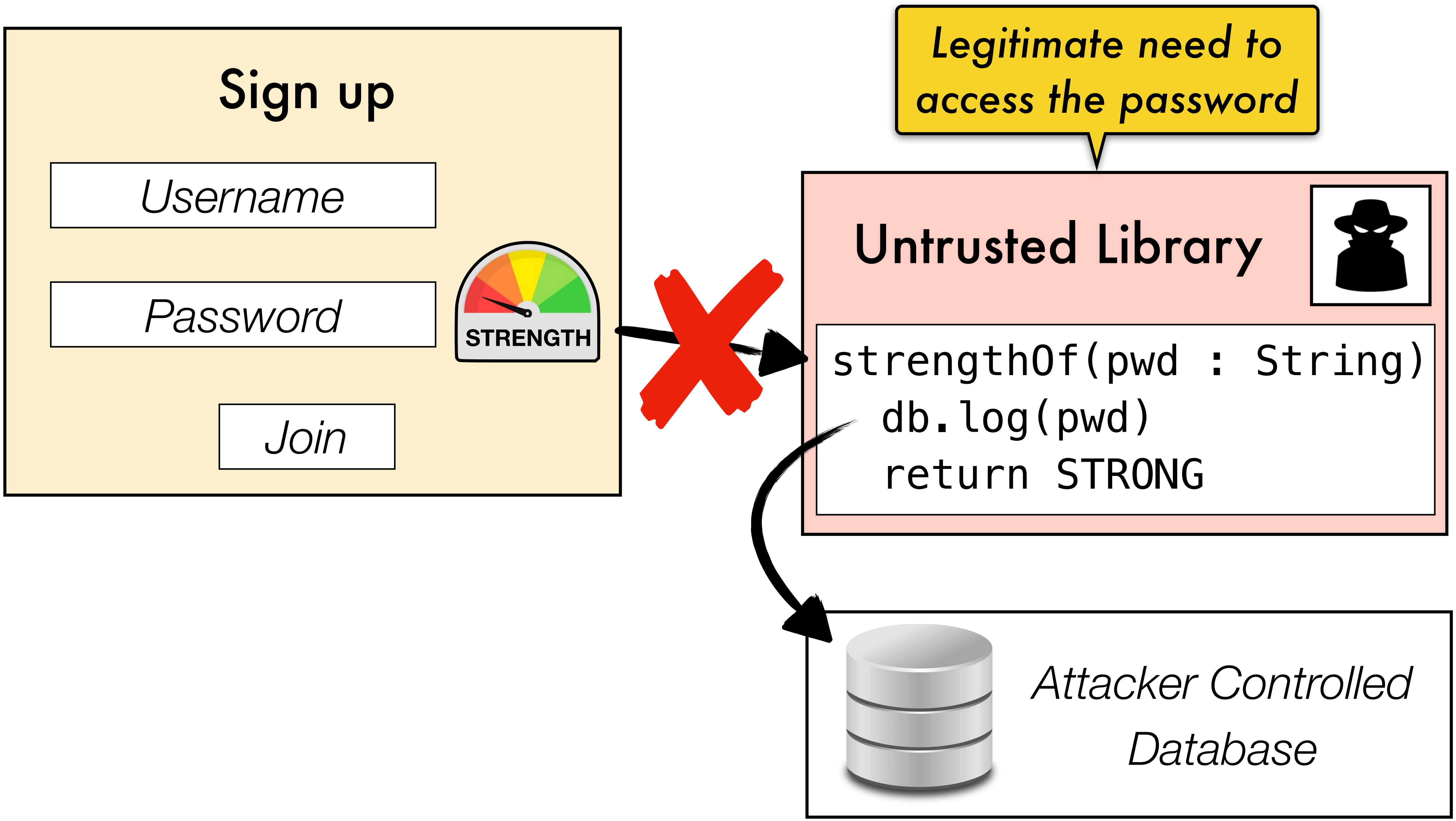
Access Control?

*Restrict **access** to sensitive data in untrusted components*



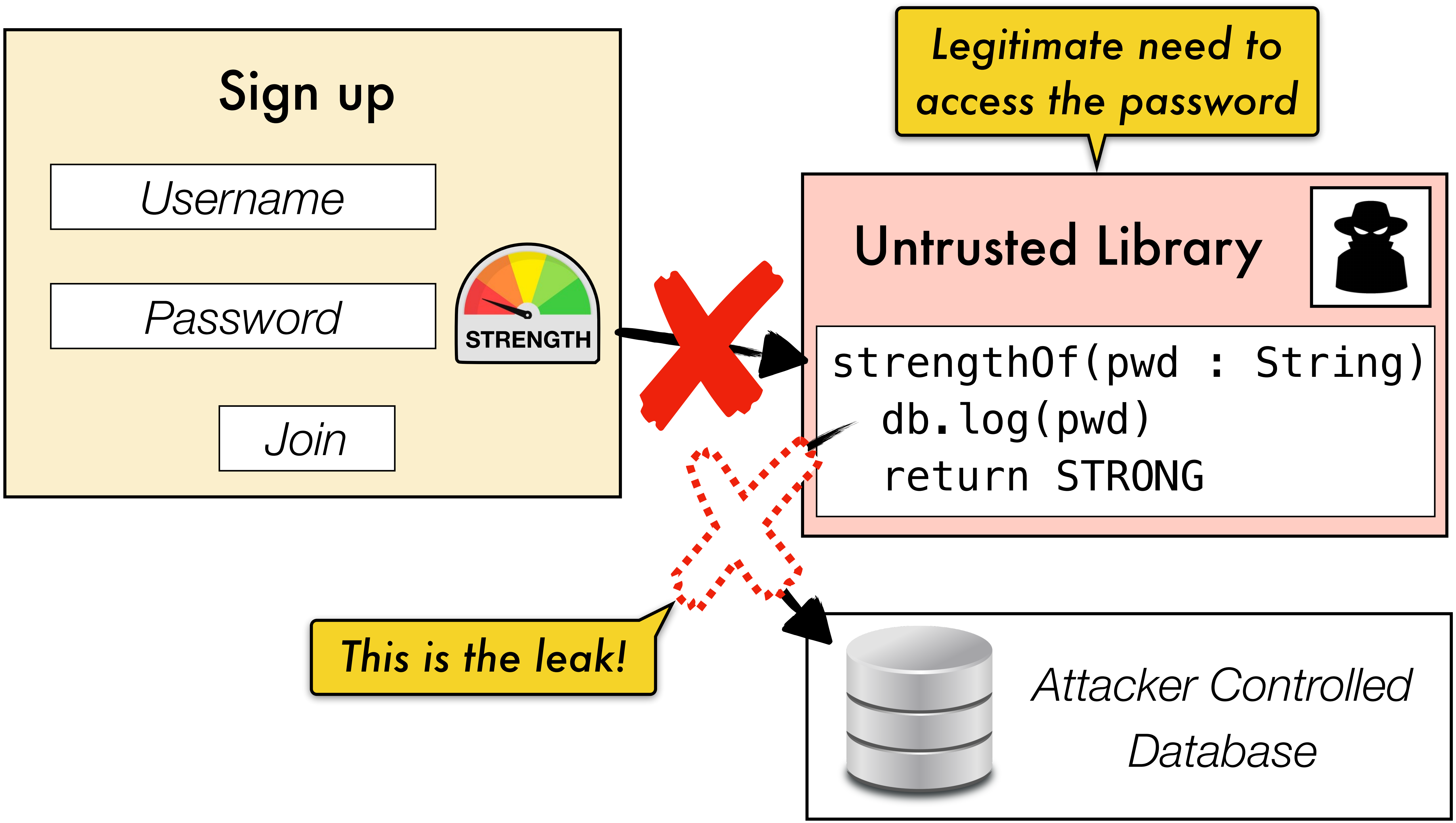
Access Control?

Restrict ***access*** *to sensitive data in untrusted components*



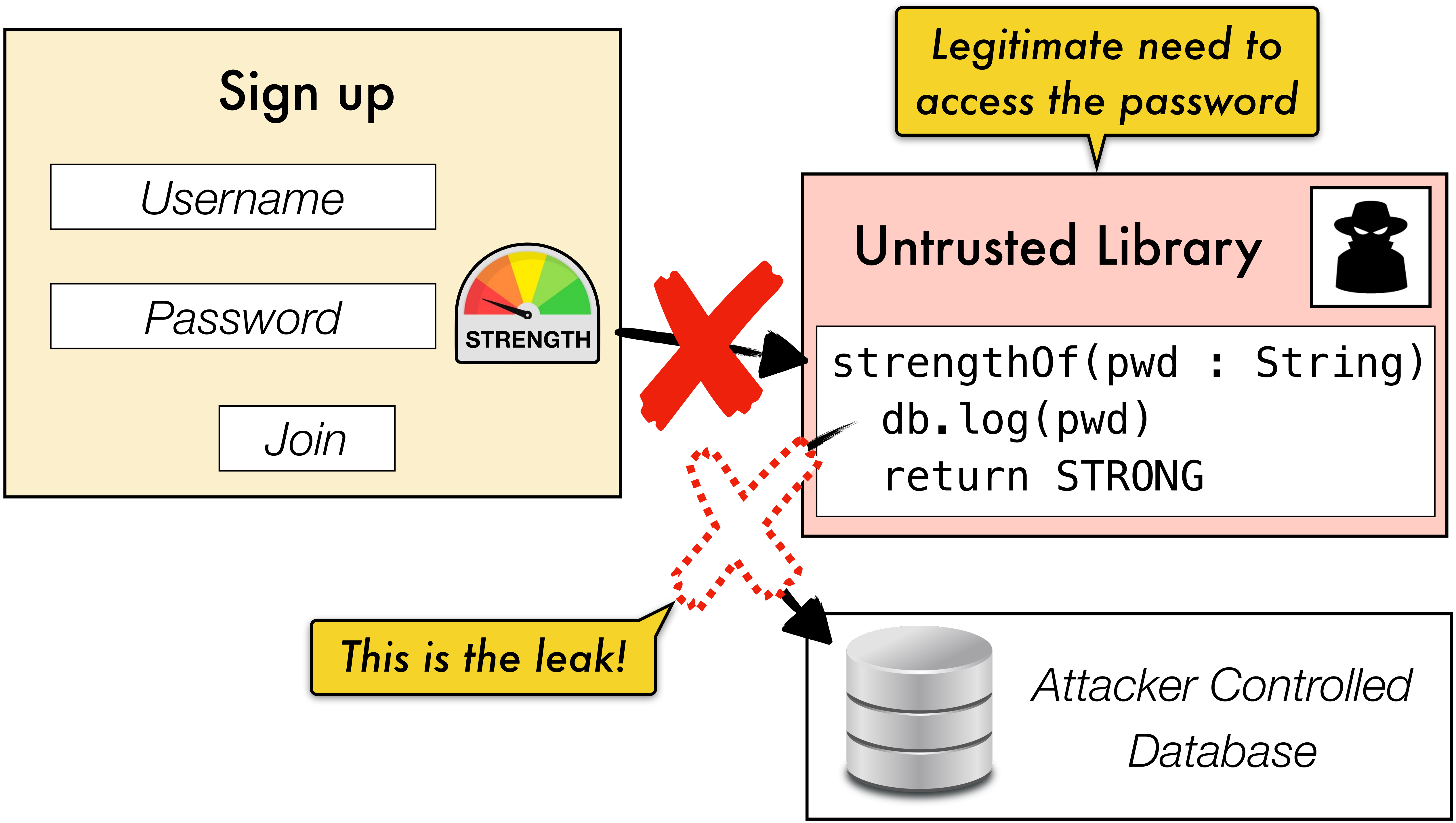
Access Control?

*Restrict **access** to sensitive data in untrusted components*



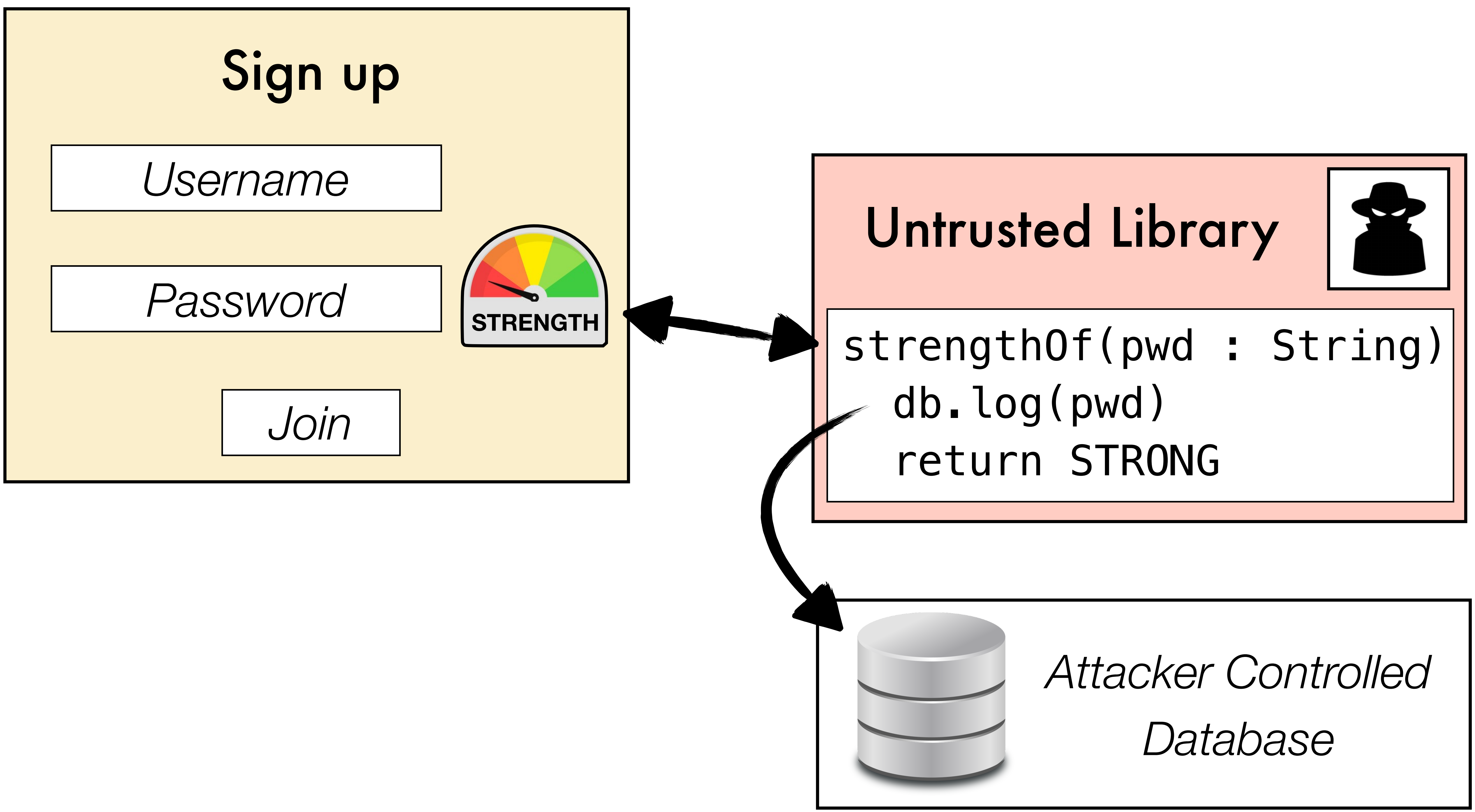
Access Control?

Restrict ***access*** *to sensitive data in untrusted components*



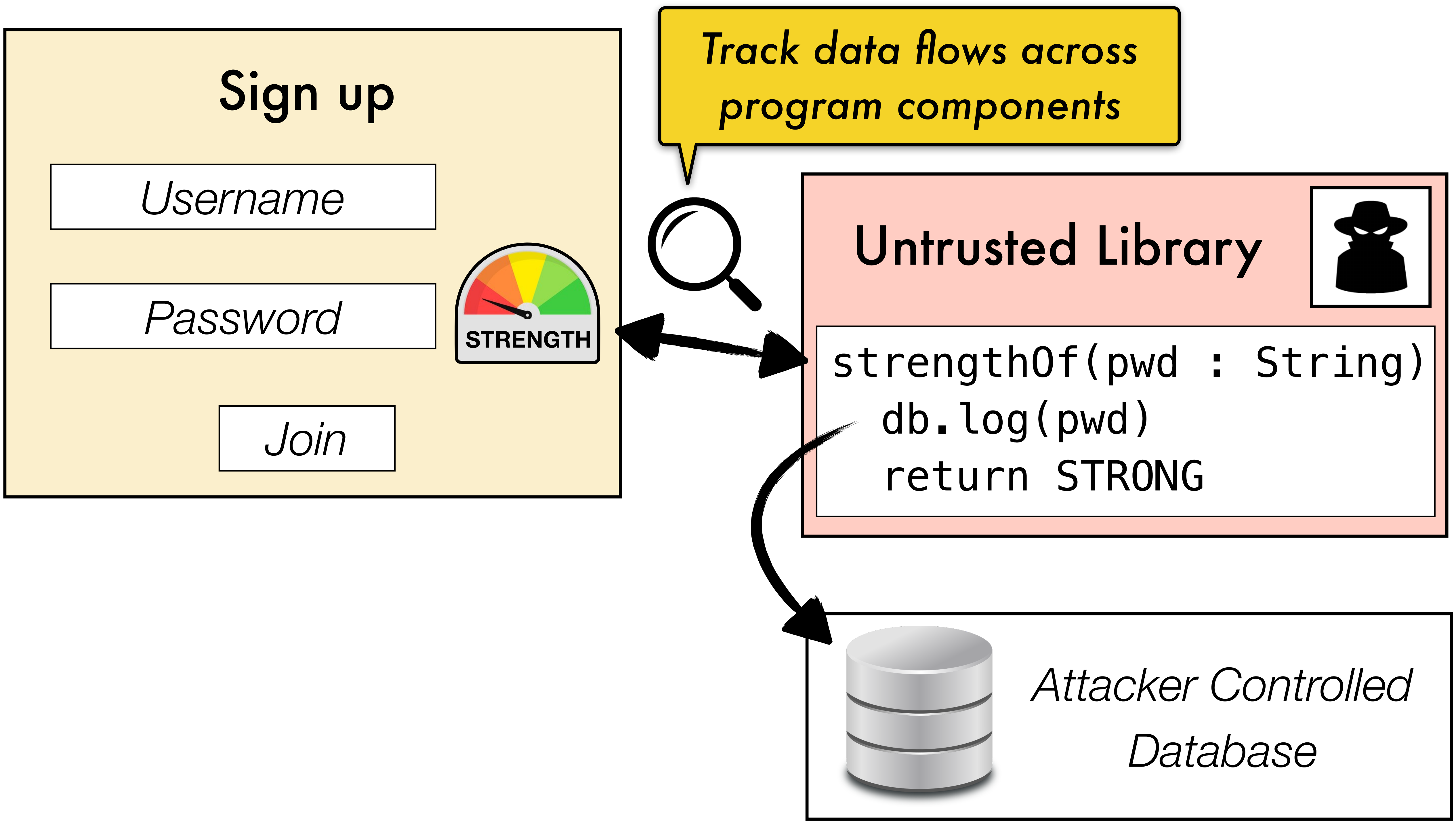
Information Flow Control

*Do not restrict data access, restrict **where** data can flow!*



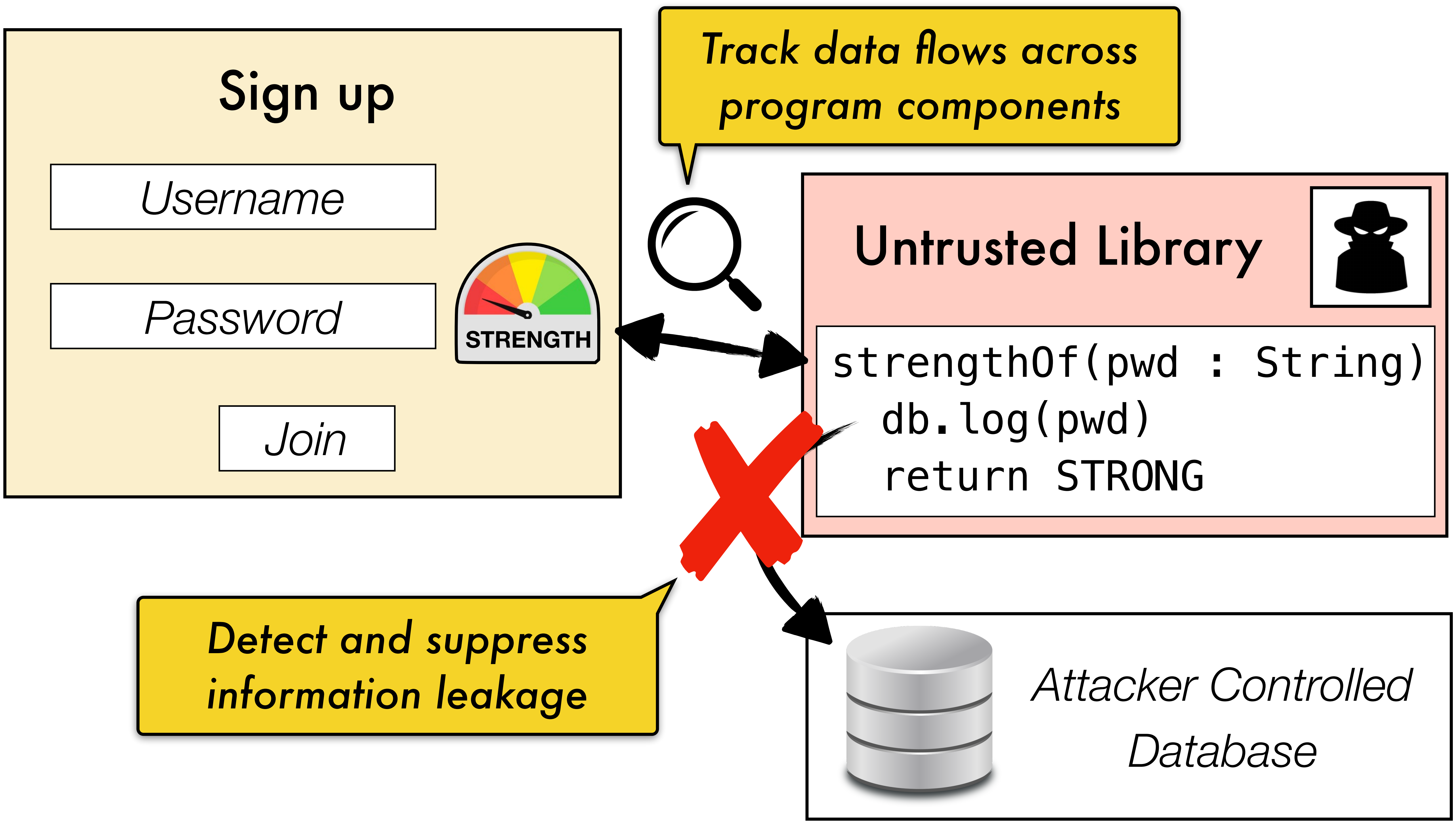
Information Flow Control

Do not restrict data access, restrict ***where*** *data can flow!*



Information Flow Control

*Do not restrict data access, restrict **where** data can flow!*



Facets of Language-based IFC

*Associate data with **security levels** to track data flows in programs*

Facets of Language-based IFC

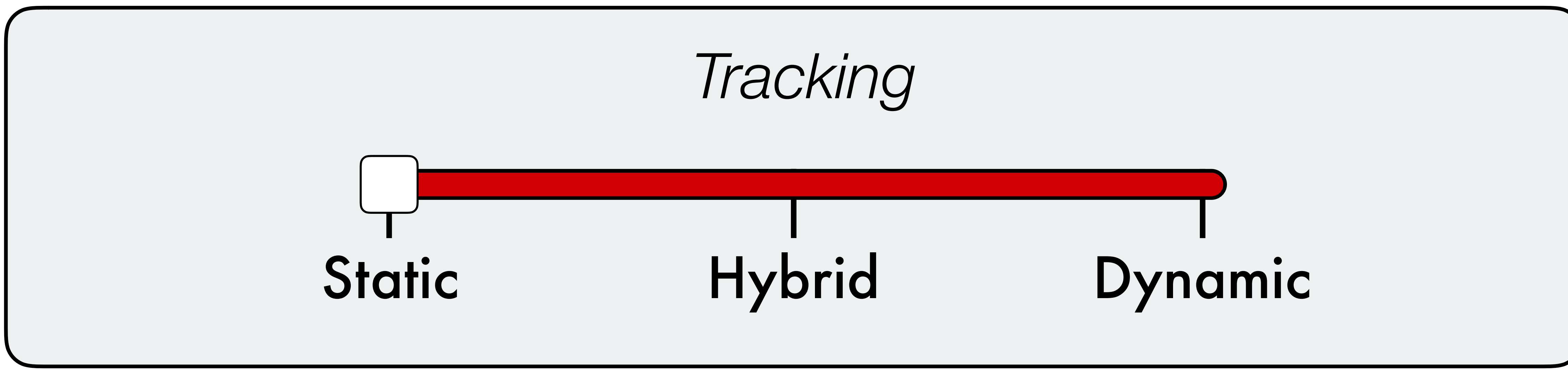
"Public" and "Secret"

Associate data with ***security levels*** *to track data flows in programs*

Facets of Language-based IFC

"Public" and "Secret"

Associate data with ***security levels*** *to track data flows in programs*

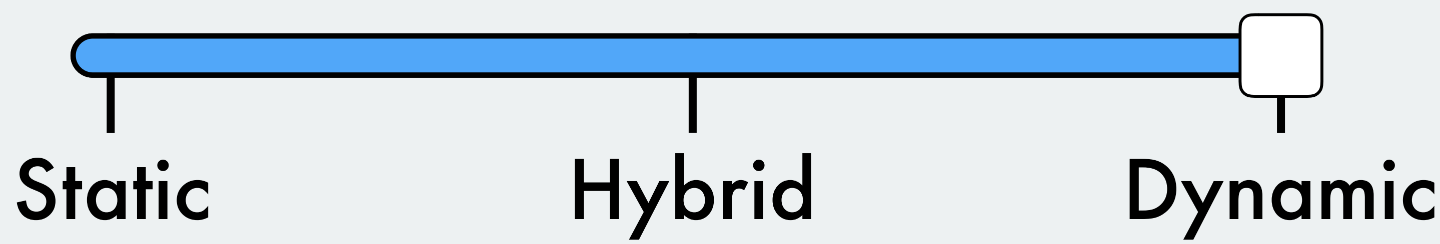


Facets of Language-based IFC

“Public” and “Secret”

Associate data with ***security levels*** *to track data flows in programs*

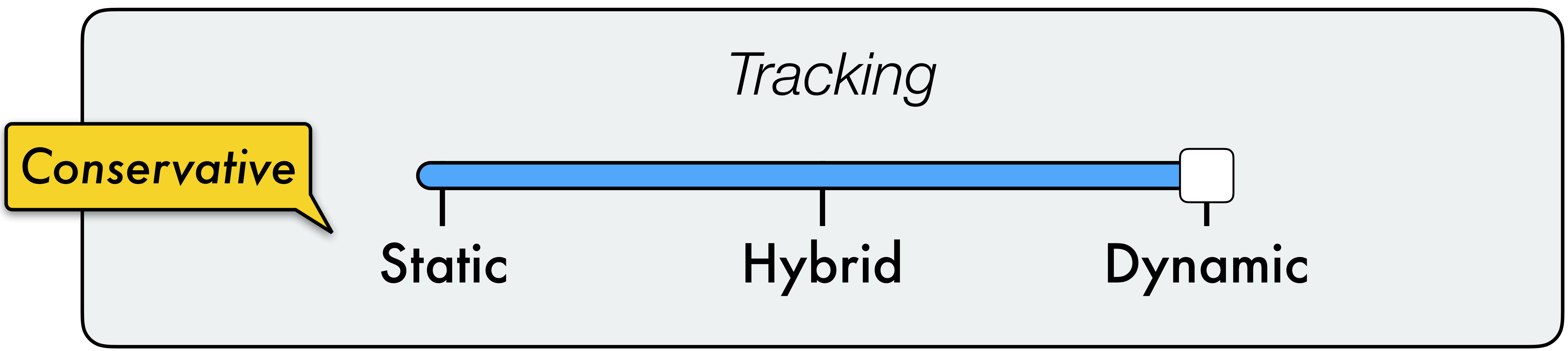
Tracking



Facets of Language-based IFC

"Public" and "Secret"

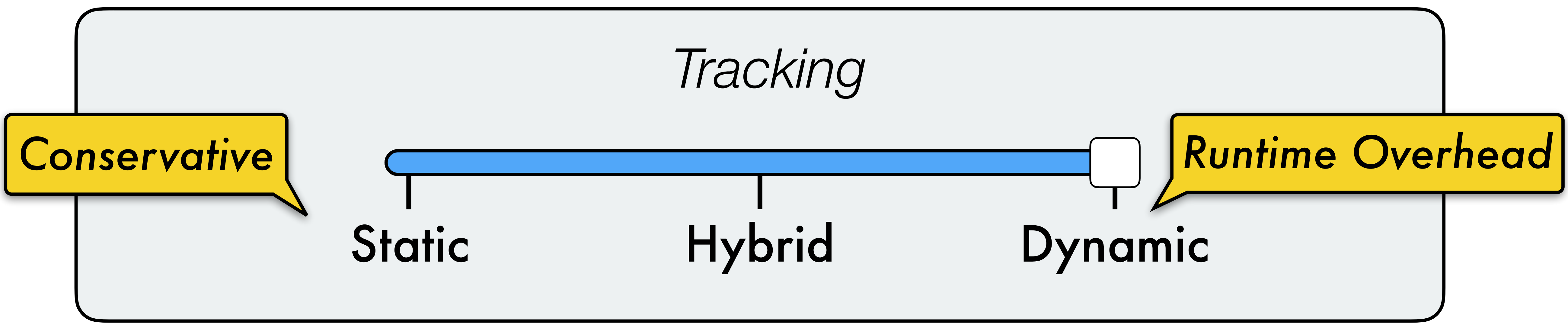
*Associate data with **security levels** to track data flows in programs*



Facets of Language-based IFC

“Public” and “Secret”

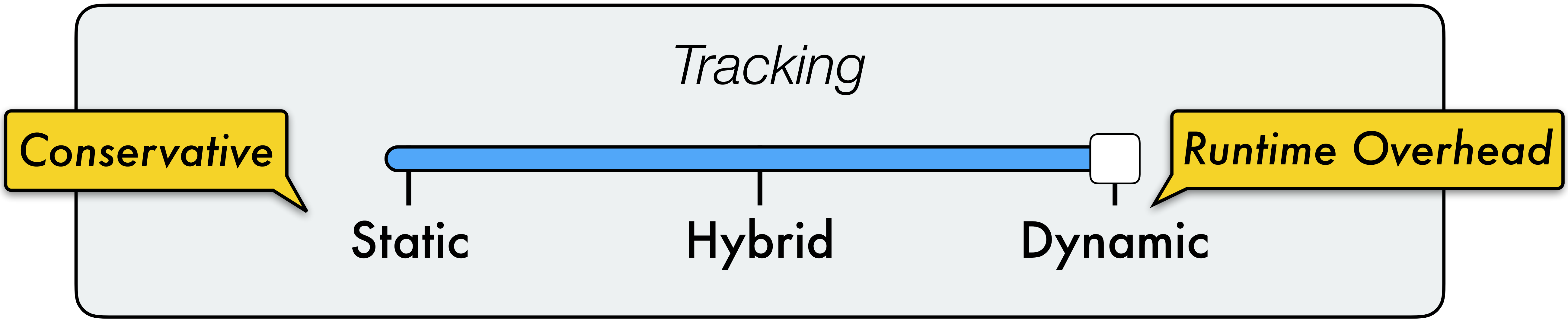
*Associate data with **security levels** to track data flows in programs*

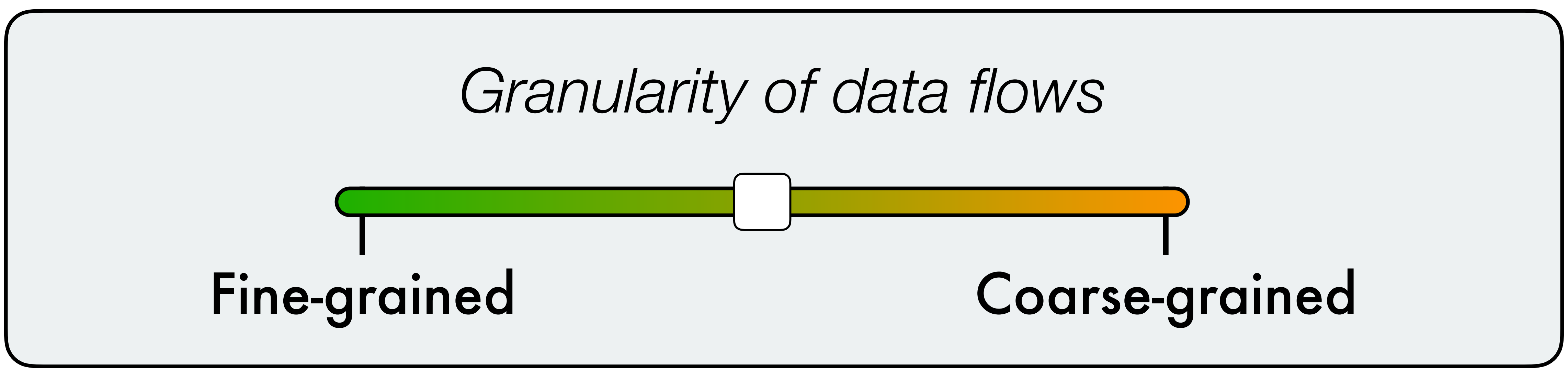


Facets of Language-based IFC

"Public" and "Secret"

Associate data with ***security levels*** *to track data flows in programs*

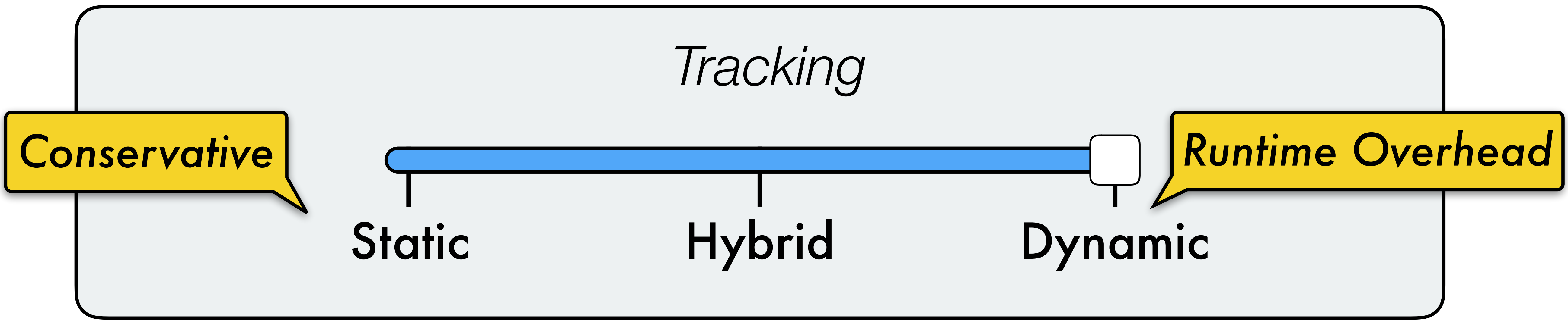


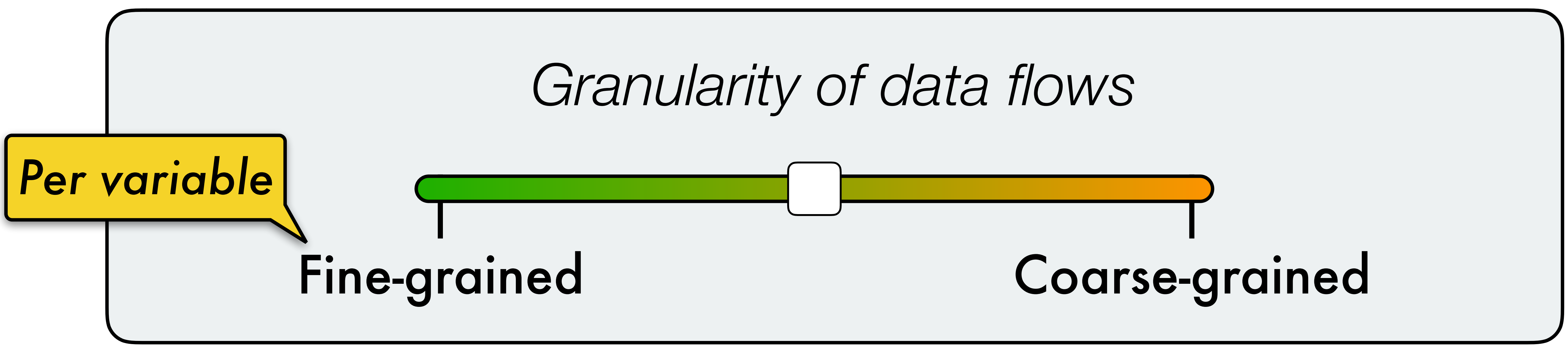


Facets of Language-based IFC

“Public” and “Secret”

Associate data with ***security levels*** *to track data flows in programs*

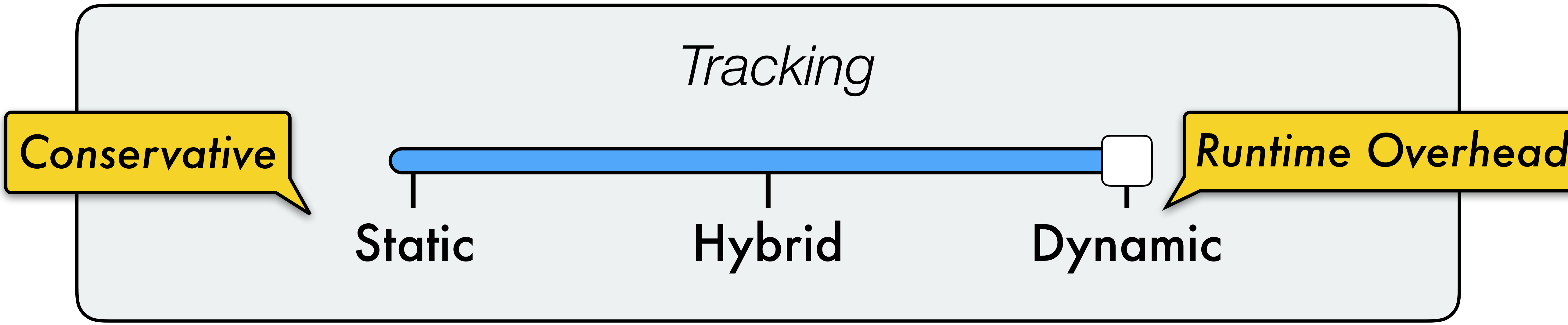


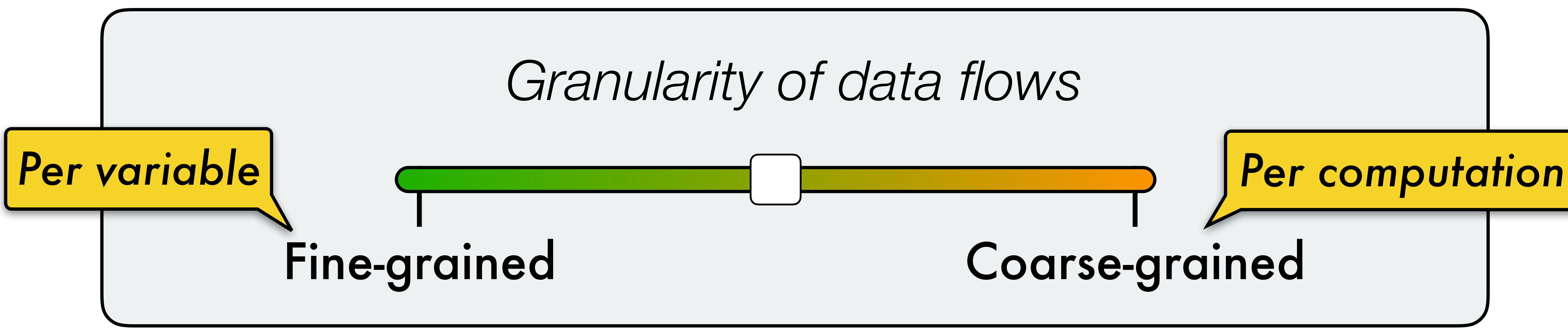


Facets of Language-based IFC

"Public" and "Secret"

*Associate data with **security levels** to track data flows in programs*





Plan

Overview of different language-based IFC approaches

- Non Interference

Plan

Overview of different language-based IFC approaches

- Non Interference

Confidentiality & Integrity

Plan

Overview of different language-based IFC approaches

- Non Interference

  Confidentiality & Integrity

- 4 IFC Languages

Plan

Overview of different language-based IFC approaches

- Non Interference

  Confidentiality & Integrity

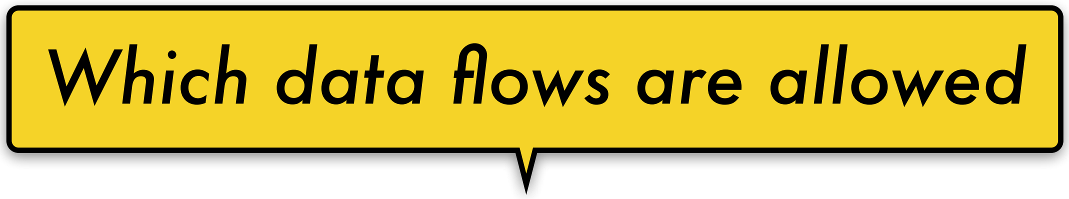
- 4 IFC Languages

	Static	*Dynamic*
Fine-grained	$\lambda^{\mathbf{SFG}}$	$\lambda^{\mathbf{DFG}}$
Coarse-grained	$\lambda^{\mathbf{SCG}}$	$\lambda^{\mathbf{DCG}}$

Security Policy

Information flow policies are specified by the security lattice

Security Policy



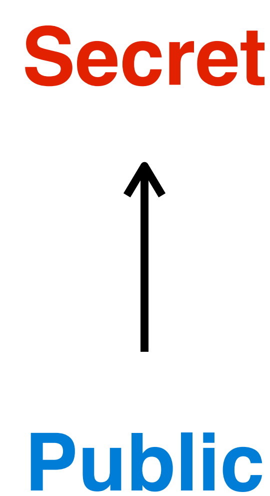
Information flow policies are specified by the security lattice

Security Policy

Which data flows are allowed

Information flow policies are specified by the security lattice

Simple lattice for ***confidentiality****:*



Security Policy

Which data flows are allowed

Information flow policies are specified by the security lattice

Simple lattice for ***confidentiality****:*

Public and Secret are security labels

Secret

↑

Public

Security Policy

Which data flows are allowed

Information flow policies are specified by the security lattice

Simple lattice for ***confidentiality****:*

Public *and* Secret *are security labels*

Secret

↑

Public

*"***Secret** *inputs* ***cannot*** *flow to* **Public** *outputs"*

Security Policy

Which data flows are allowed

Information flow policies are specified by the security lattice

Simple lattice for ***confidentiality****:*

Public and Secret are security labels

Secret

↑

Public

2-point lattice

*"****Secret*** *inputs* ***cannot*** *flow to* ***Public*** *outputs"*

Simple lattice for ***confidentiality****:*

Secret

↑

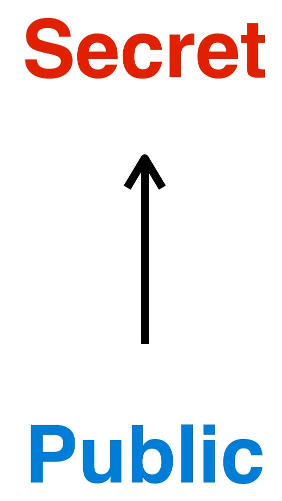
Public

*"***Secret** *inputs* ***cannot*** *flow to* **Public** *outputs"*

Formally:

$$\mathscr{L}^{\mathbf{C}} = (\{\mathbf{P},\mathbf{S}\} , \sqsubseteq^{\mathbf{C}} , \sqcup^{\mathbf{C}})$$

Simple lattice for ***confidentiality****:*



*"***Secret** *inputs* ***cannot*** *flow to* **Public** *outputs"*

Formally:

Partial order between labels

$$\mathscr{L}^{C} = (\{P,S\} , \sqsubseteq^{C} , \sqcup^{C})$$

*Simple lattice for **confidentiality**:*

Secret

↑ $\sqsubseteq^{\mathbf{C}}$

Public

*"**Secret** inputs **cannot** flow to **Public** outputs"*

Formally:

Partial order between labels

$$\mathscr{L}^{\mathbf{C}} = (\{\mathbf{P},\mathbf{S}\} , \sqsubseteq^{\mathbf{C}} , \sqcup^{\mathbf{C}})$$

Simple lattice for ***confidentiality****:*

Secret

$\uparrow \sqsubseteq^{C}$

Public

*"***Secret*** inputs* ***cannot*** *flow to* **Public** *outputs"*

Formally:

Partial order between labels

$$\mathscr{L}^{C} = (\{P, S\} , \sqsubseteq^{C} , \sqcup^{C})$$

where

$$P \sqsubseteq^{C} P \qquad S \sqsubseteq^{C} S$$

$$P \sqsubseteq^{C} S \qquad S \not\sqsubseteq^{C} P$$

*Simple lattice for **confidentiality**:*

Secret

$\uparrow \sqsubseteq^{C}$

Public

*"**Secret** inputs **cannot** flow to **Public** outputs"*

Formally:

Join Operator (least upper bound)

$$\mathscr{L}^{C} = (\{P, S\} , \sqsubseteq^{C} , \sqcup^{C})$$

Simple lattice for ***confidentiality****:*

Secret

$\uparrow \sqsubseteq^{\mathbf{C}}$

Public

*"***Secret** *inputs* ***cannot*** *flow to* **Public** *outputs"*

Formally:

Join Operator (least upper bound)

$$\mathscr{L}^{\mathbf{C}} = (\{\mathbf{P}, \mathbf{S}\} ,\ \sqsubseteq^{\mathbf{C}} ,\ \sqcup^{\mathbf{C}})$$

where

$$\mathbf{P} \sqcup^{\mathbf{C}} \mathbf{P} = \mathbf{P} \qquad \mathbf{S} \sqcup^{\mathbf{C}} \mathbf{S} = \mathbf{S}$$

$$\mathbf{P} \sqcup^{\mathbf{C}} \mathbf{S} = \mathbf{S} \qquad \mathbf{S} \sqcup^{\mathbf{C}} \mathbf{P} = \mathbf{S}$$

"Dual" lattice for ***integrity****:*

Untrusted

$\uparrow \sqsubseteq_I$

Trusted

*"***Untrusted** *inputs* ***cannot*** *flow to* **Trusted** *outputs"*

"Dual" lattice for ***integrity****:*

Untrusted

$\uparrow \sqsubseteq^{\mathbf{I}}$

Trusted

*"***Untrusted** *inputs* ***cannot*** *flow to* **Trusted** *outputs"*

Formally:

$$\mathscr{L}^{\mathbf{I}} = (\{\mathbf{T},\mathbf{U}\} , \sqsubseteq^{\mathbf{I}} , \sqcup^{\mathbf{I}})$$

"Dual" lattice for ***integrity****:*

Untrusted

$\uparrow \sqsubseteq^{I}$

Trusted

*"***Untrusted*** inputs* ***cannot*** *flow to* **Trusted** *outputs"*

Formally:

$$\mathscr{L}^{I} = (\{T, U\}, \sqsubseteq^{I}, \sqcup^{I})$$

where

$$T \sqsubseteq^{I} T \qquad U \sqsubseteq^{I} U$$

$$T \sqsubseteq^{I} U \qquad U \not\sqsubseteq^{I} T$$

"Dual" lattice for ***integrity****:*

Untrusted

$\uparrow \sqsubseteq^{I}$

Trusted

*"***Untrusted*** inputs* ***cannot*** *flow to* **Trusted** *outputs"*

Formally:

$$\mathscr{L}^{I} = (\{T, U\}, \sqsubseteq^{I}, \sqcup^{I})$$

where

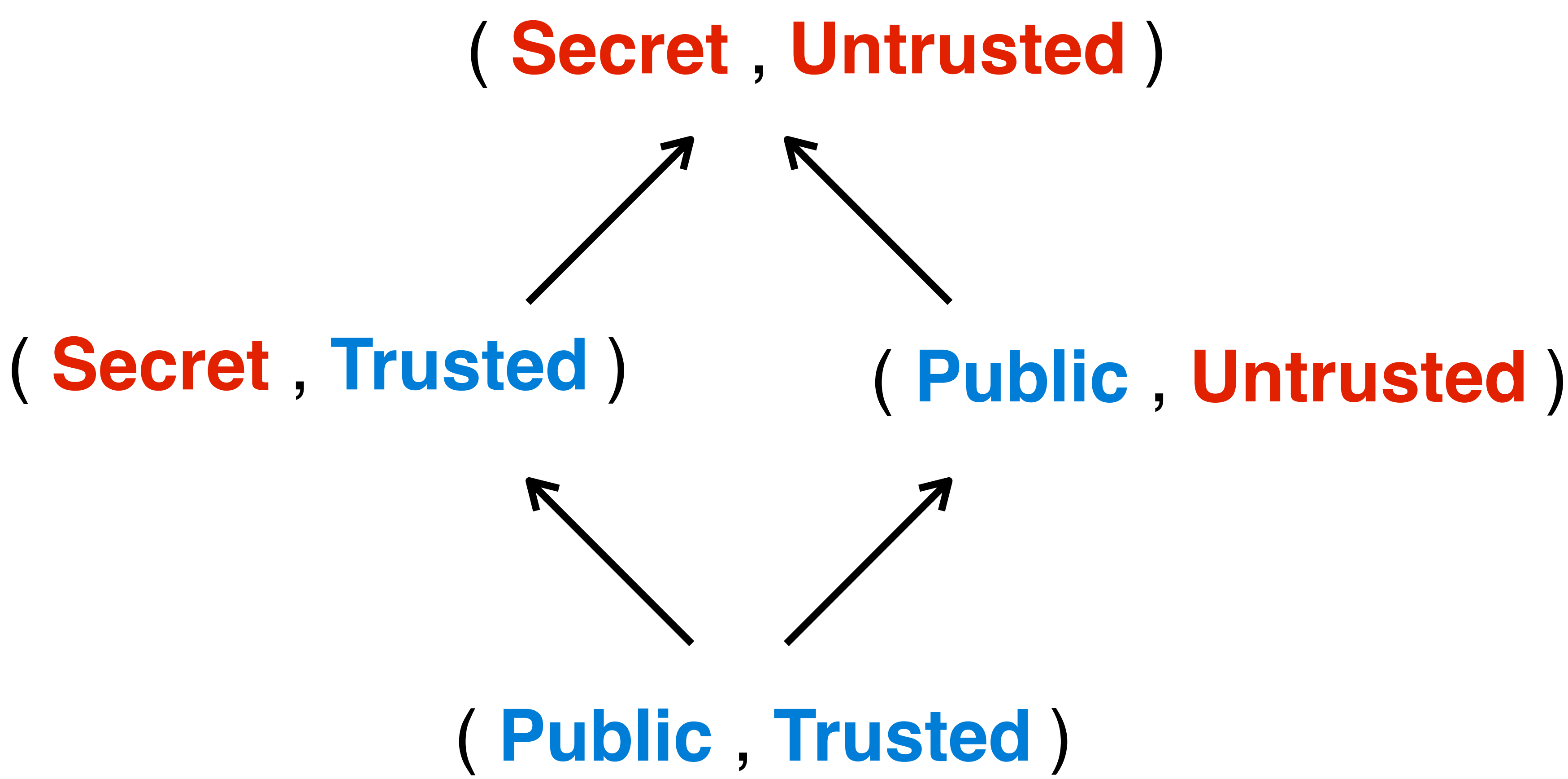
$$T \sqcup^{I} T = T \qquad U \sqcup^{I} U = U$$

$$T \sqcup^{I} U = U \qquad U \sqcup^{I} P = U$$

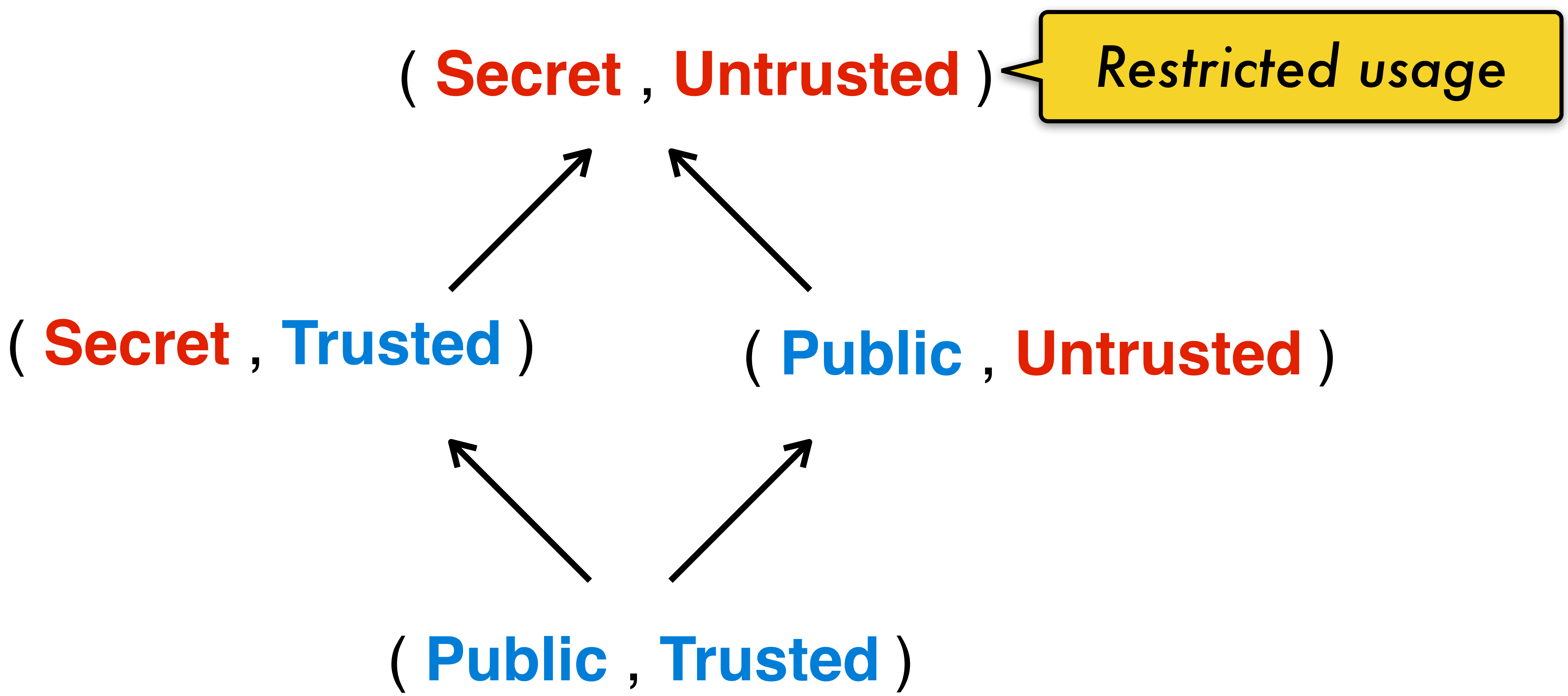

Secret
⊑C
Public
Untrusted
⊑I
Trusted

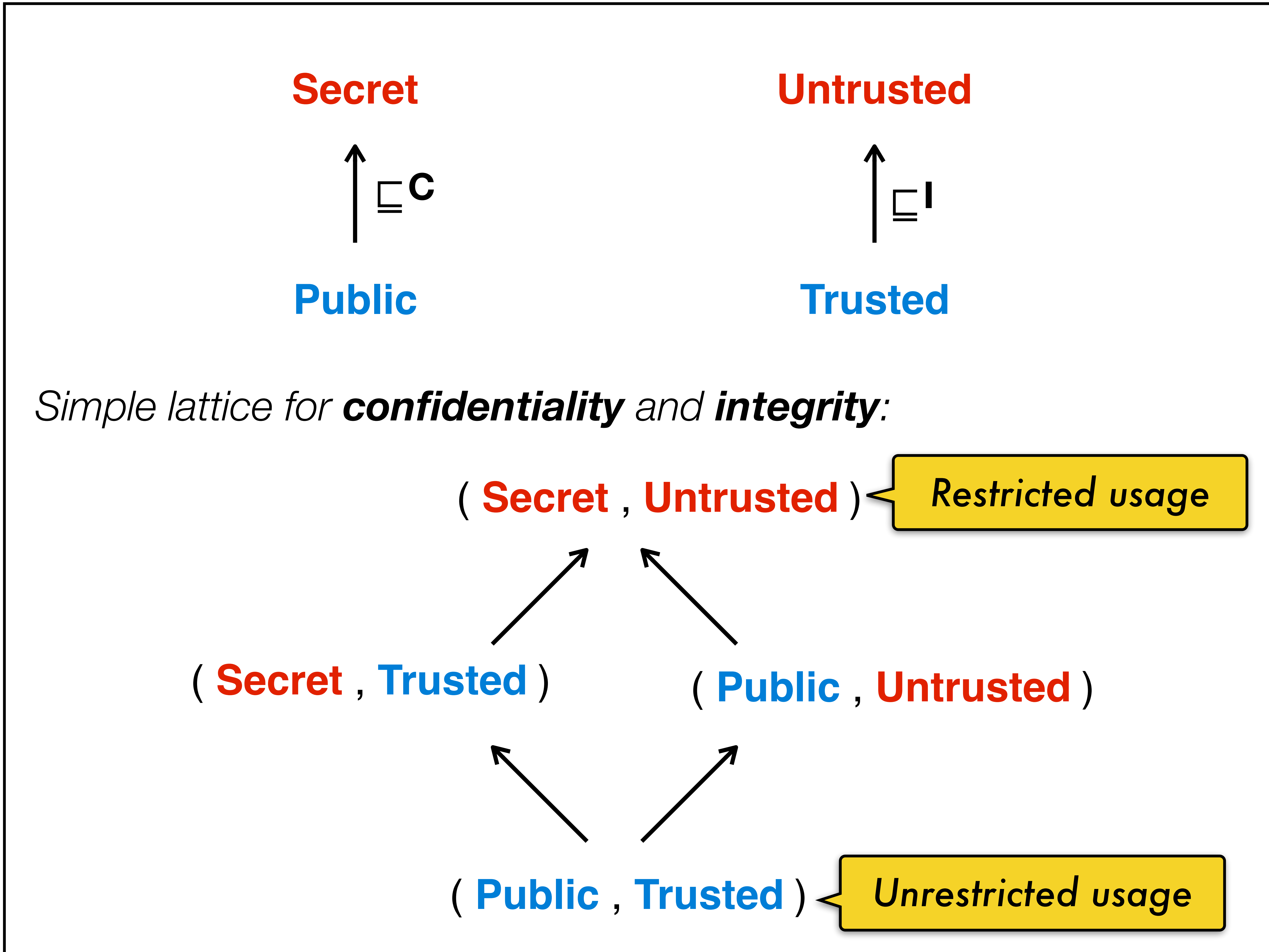
*Simple lattice for **confidentiality** and **integrity**:*

Simple lattice for ***confidentiality*** *and* ***integrity****:*

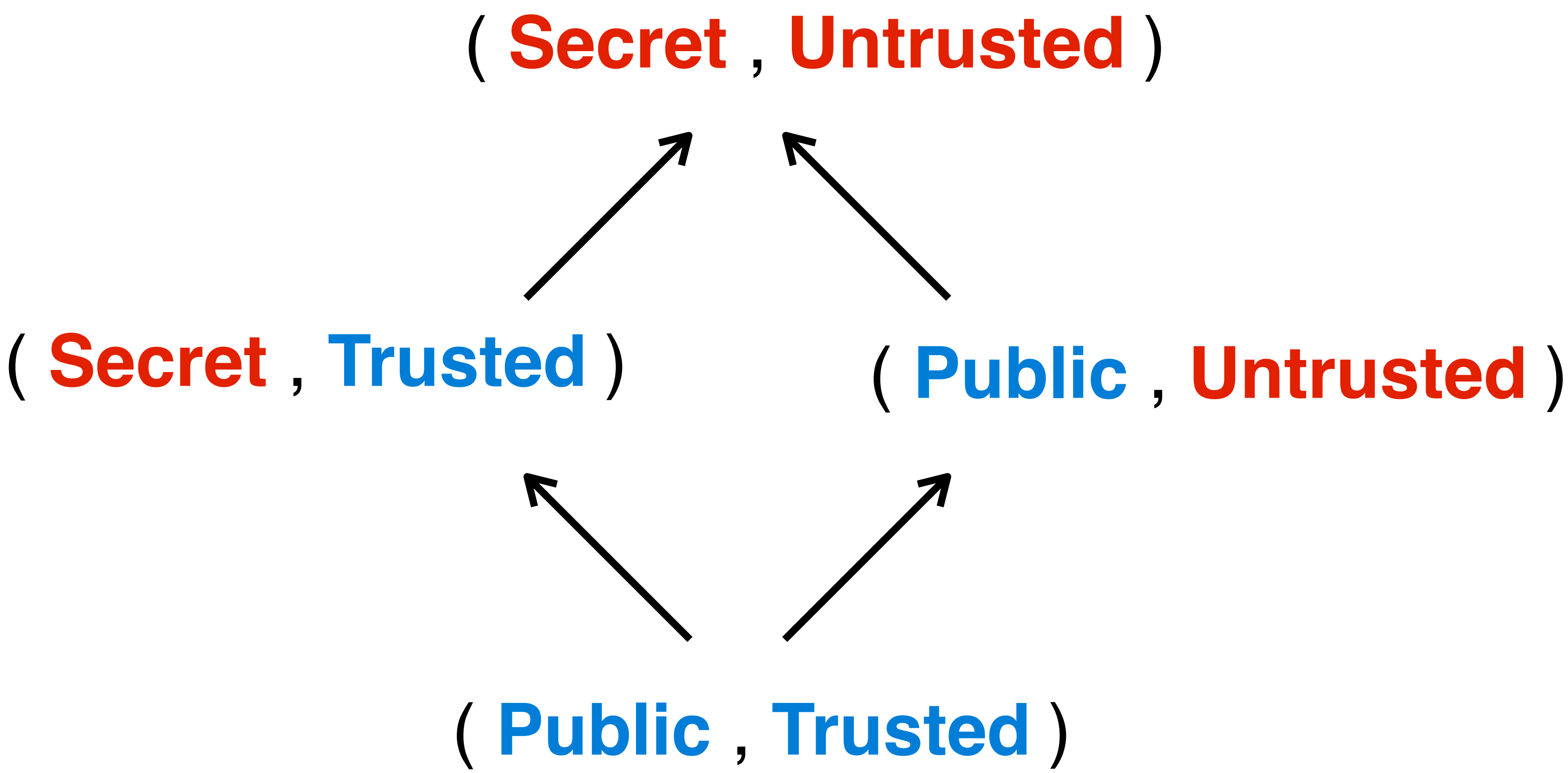


Simple lattice for ***confidentiality*** *and* ***integrity****:*

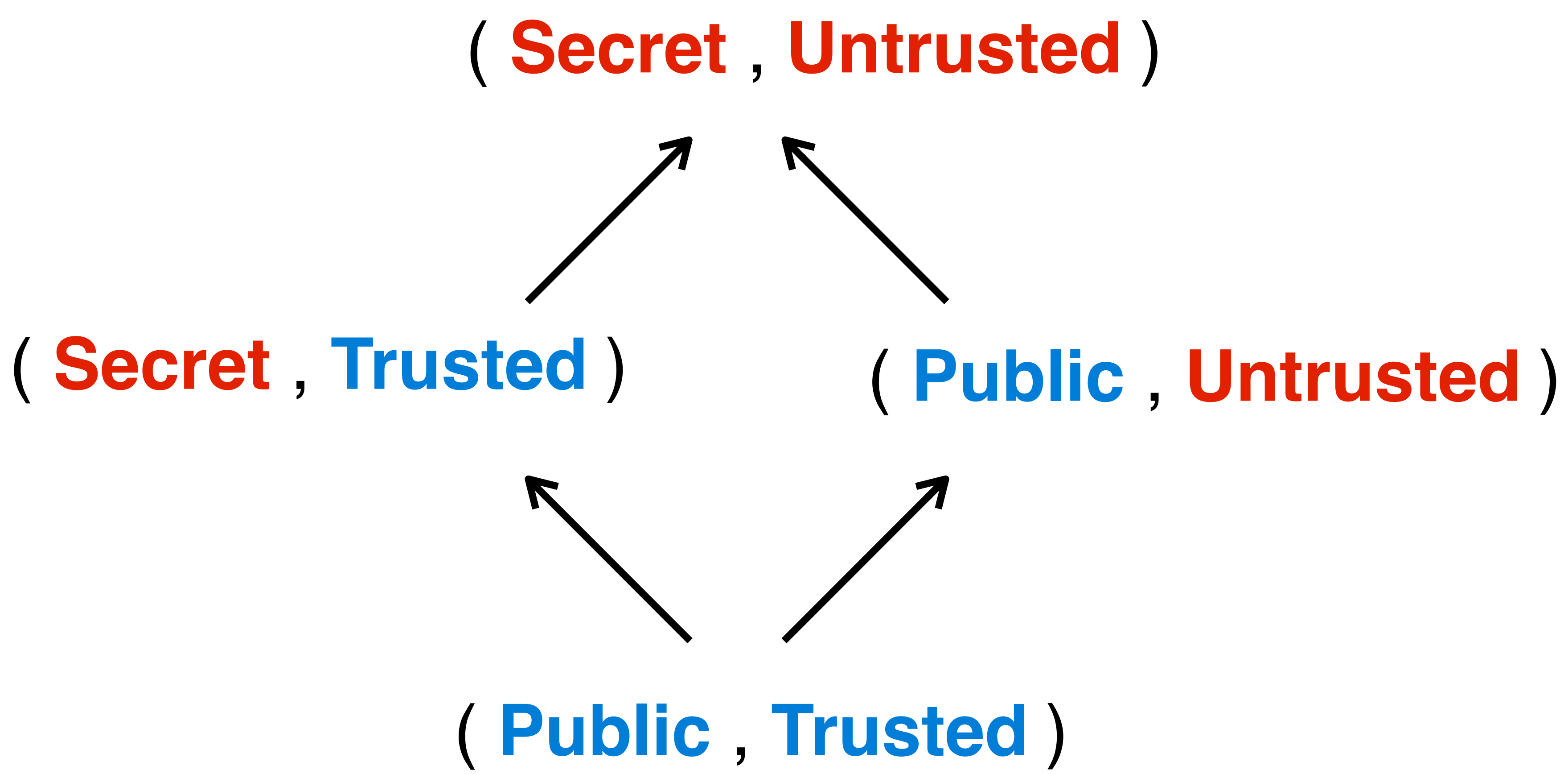



Secret
⊑C
Public
Untrusted
⊑I
Trusted
Simple lattice for confidentiality and integrity:
(Secret , Untrusted)
Restricted usage
(Secret , Trusted)
(Public , Untrusted)
(Public , Trusted)
Unrestricted usage

*Simple lattice for **confidentiality** and **integrity**:*



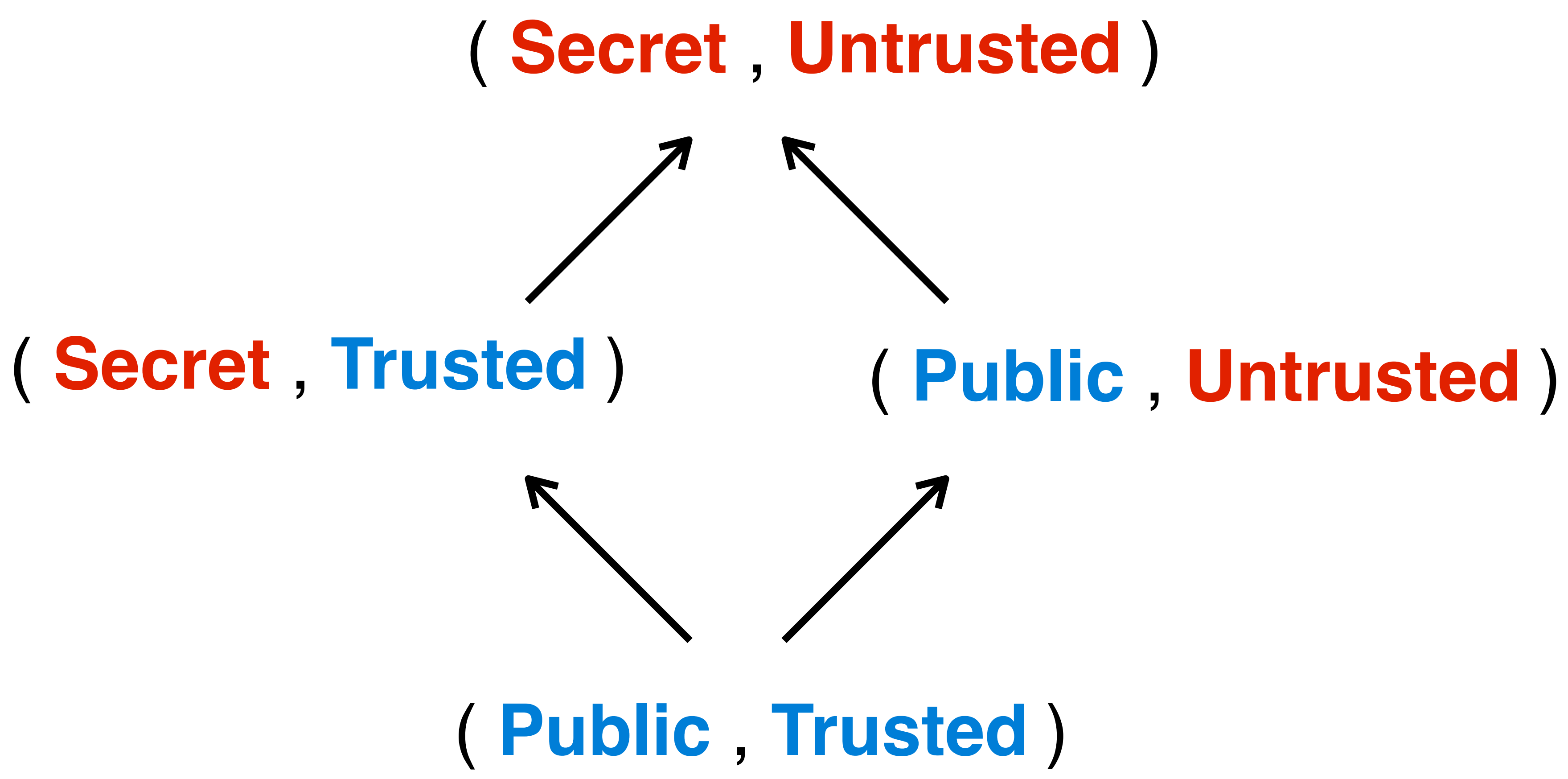
Simple lattice for ***confidentiality*** *and* ***integrity****:*



Formally:

$$\mathscr{L}^{CI} = (\{P,S\} \times \{T,U\} , \sqsubseteq^{C} \times \sqsubseteq^{I} , \sqcup^{C} \times \sqcup^{I})$$

Simple lattice for ***confidentiality*** *and* ***integrity****:*



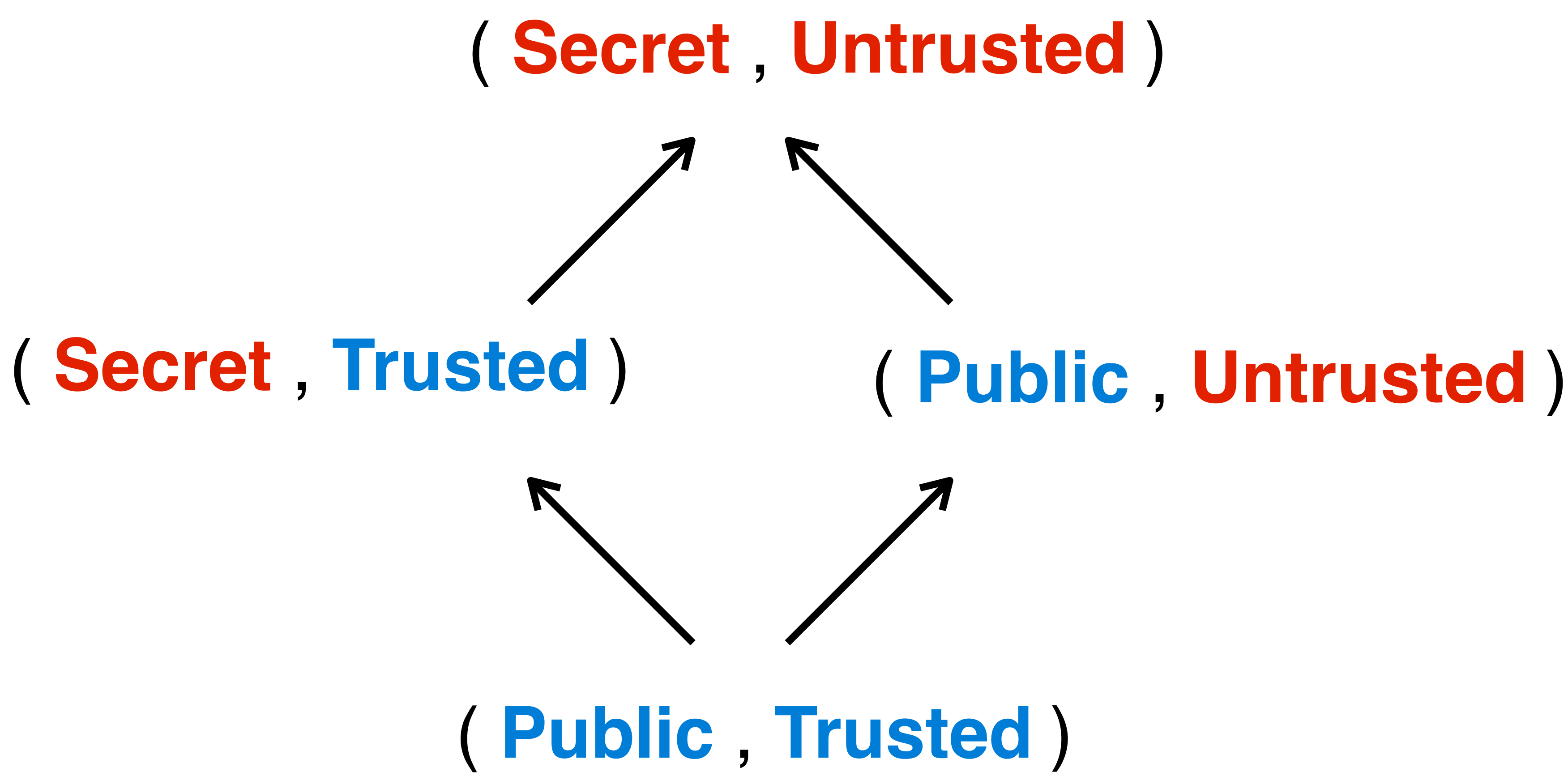
Formally:

$$\mathscr{L}^{CI} = (\{P,S\} \times \{T,U\} , \sqsubseteq^{C} \times \sqsubseteq^{I} , \sqcup^{C} \times \sqcup^{I})$$

Notice

$$(S , T) \not\sqsubseteq^{CI} (P , U) \qquad (P , U) \not\sqsubseteq^{CI} (S , T)$$

Simple lattice for ***confidentiality*** *and* ***integrity****:*



Formally:

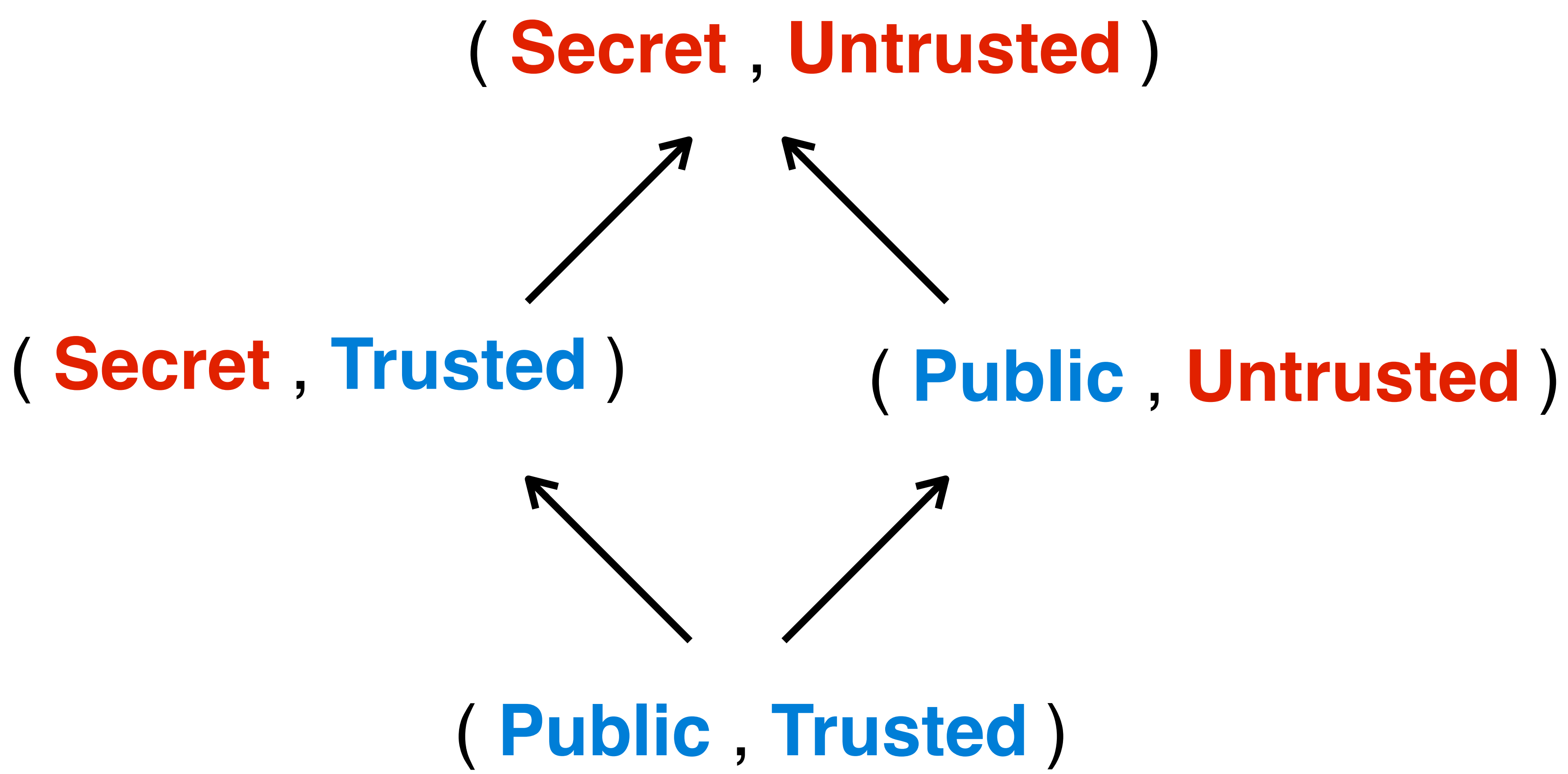
$$\mathscr{L}^{CI} = (\{P,S\} \times \{T,U\} , \sqsubseteq^{C} \times \sqsubseteq^{I} , \sqcup^{C} \times \sqcup^{I})$$

Notice

Mutually Incomparable

$$(S , T) \not\sqsubseteq^{CI} (P , U) \qquad (P , U) \not\sqsubseteq^{CI} (S , T)$$

Simple lattice for ***confidentiality*** *and* ***integrity****:*



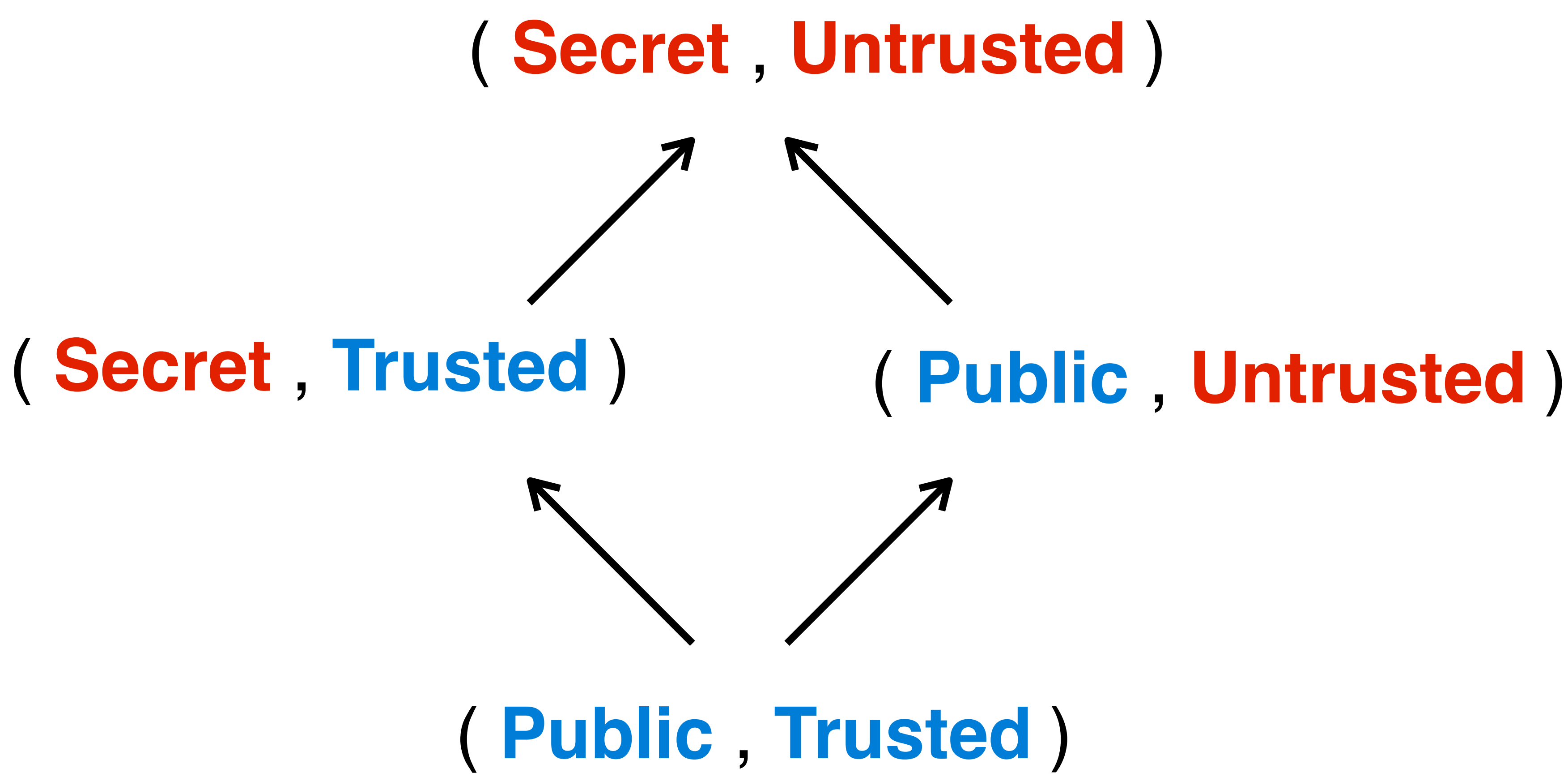
Formally:

$$\mathscr{L}^{CI} = (\{P,S\} \times \{T,U\} , \sqsubseteq^{C} \times \sqsubseteq^{I} , \sqcup^{C} \times \sqcup^{I})$$

Notice

$$(S , T) \sqcup^{CI} (P , U)$$

Simple lattice for ***confidentiality*** *and* ***integrity****:*



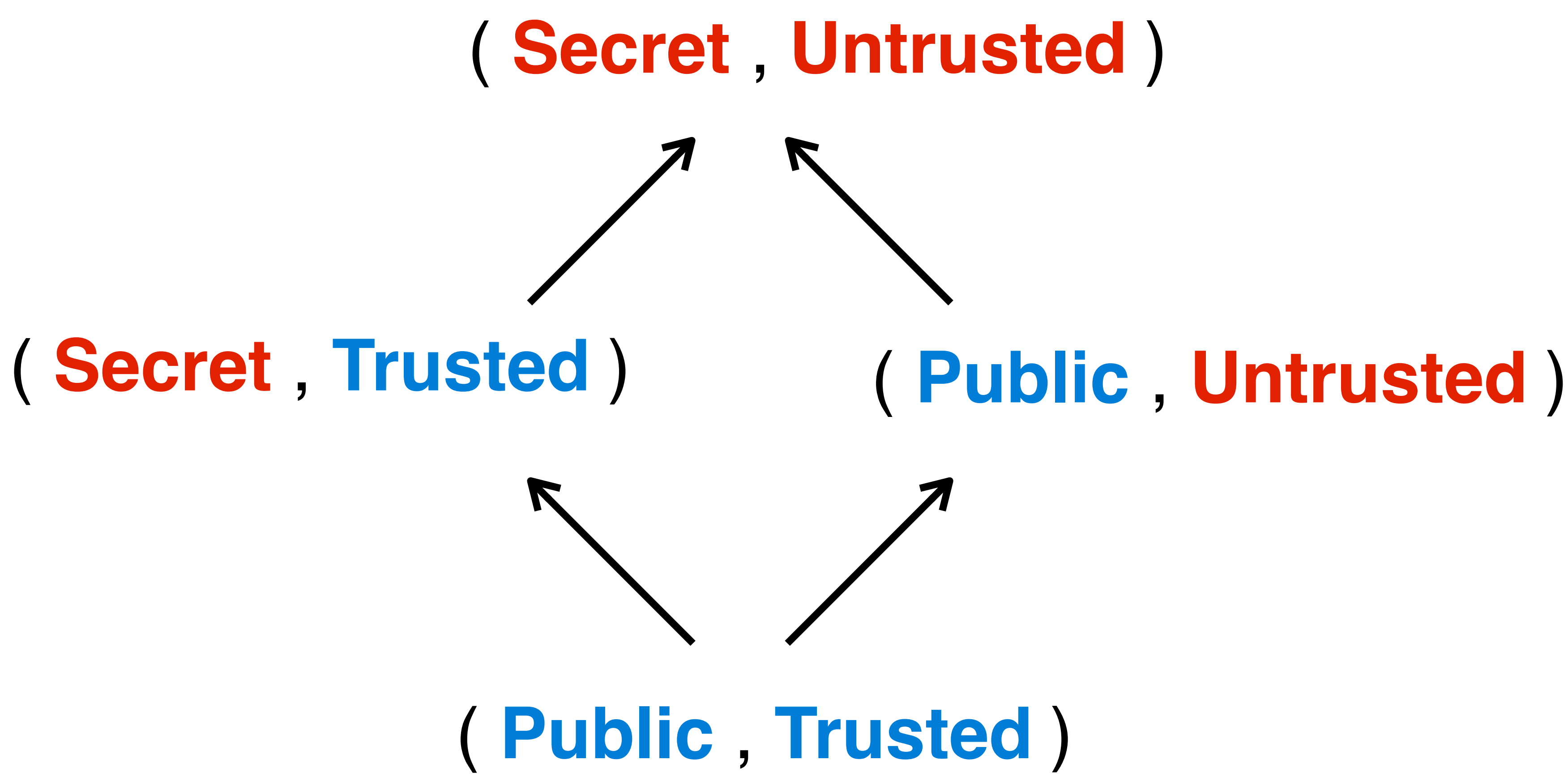
Formally:

$$\mathscr{L}^{CI} = (\{P,S\} \times \{T,U\} , \sqsubseteq^{C} \times \sqsubseteq^{I} , \sqcup^{C} \times \sqcup^{I})$$

Notice

$$(S , T) \sqcup^{CI} (P , U) = (S \sqcup^{C} P , T \sqcup^{I} U)$$

Simple lattice for ***confidentiality*** *and* ***integrity****:*



Formally:

$$\mathscr{L}^{CI} = (\{P,S\} \times \{T,U\} , \sqsubseteq^{C} \times \sqsubseteq^{I} , \sqcup^{C} \times \sqcup^{I})$$

Notice

$$(S , T) \sqcup^{CI} (P , U) = (S \sqcup^{C} P , T \sqcup^{I} U) = (S , U)$$

*General lattice for principals **P**:*

*General lattice for principals **P**:*

P = {Alice, Bob, Charlie}

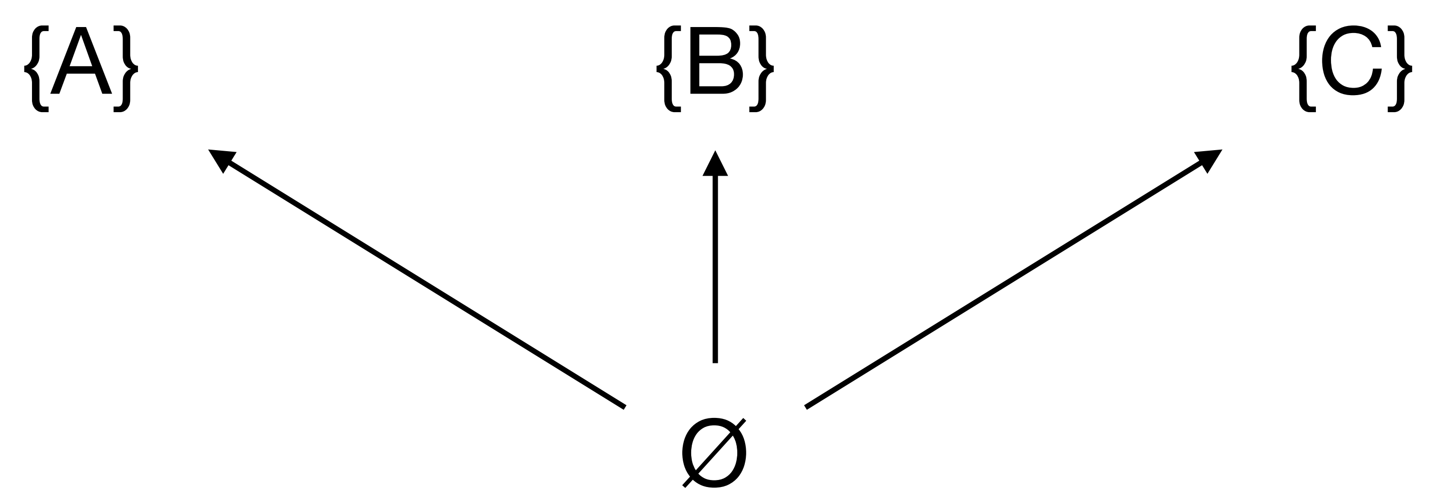
*General lattice for principals **P**:*

P = {Alice, Bob, Charlie}

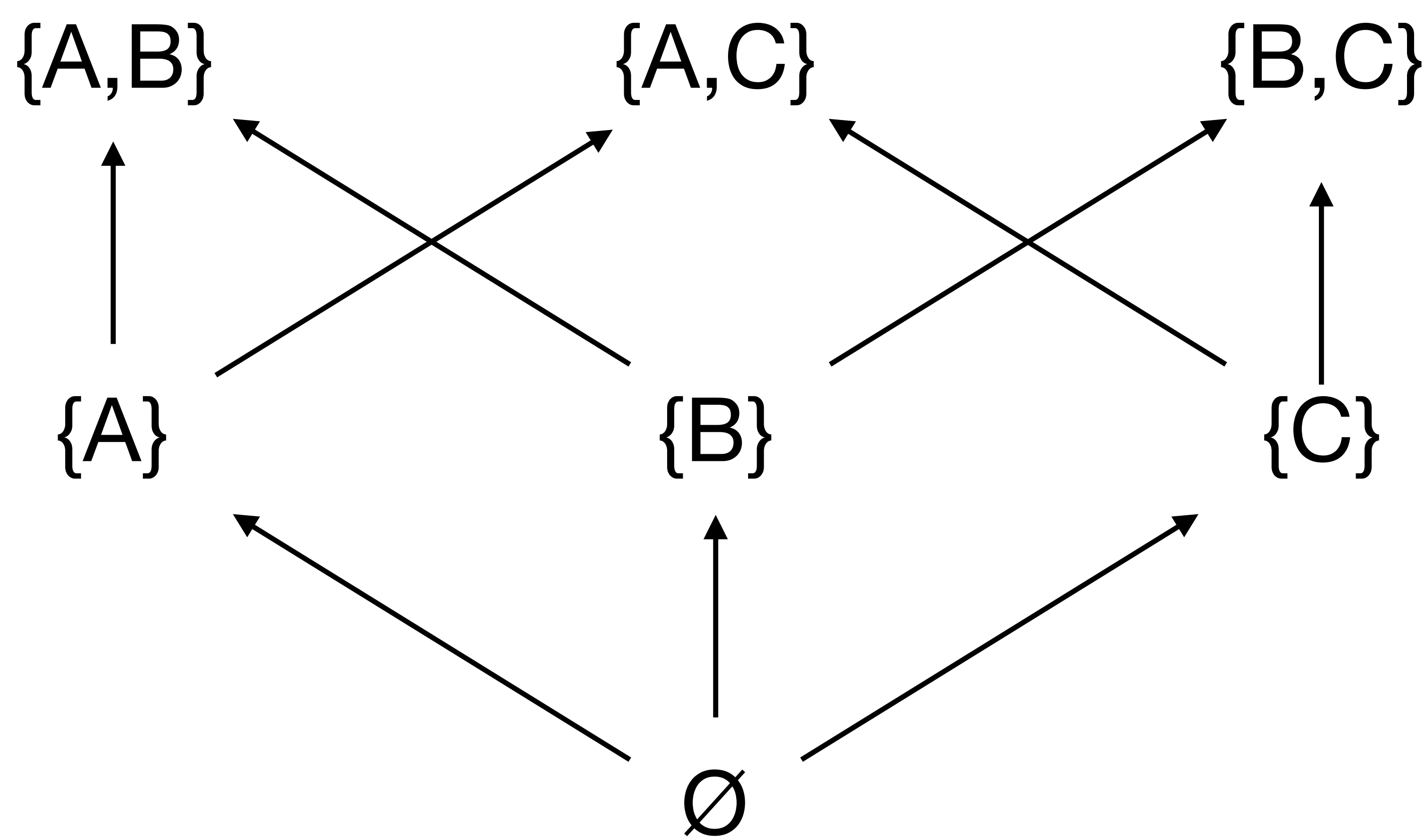
Ø

*General lattice for principals **P**:*

P = {Alice, Bob, Charlie}

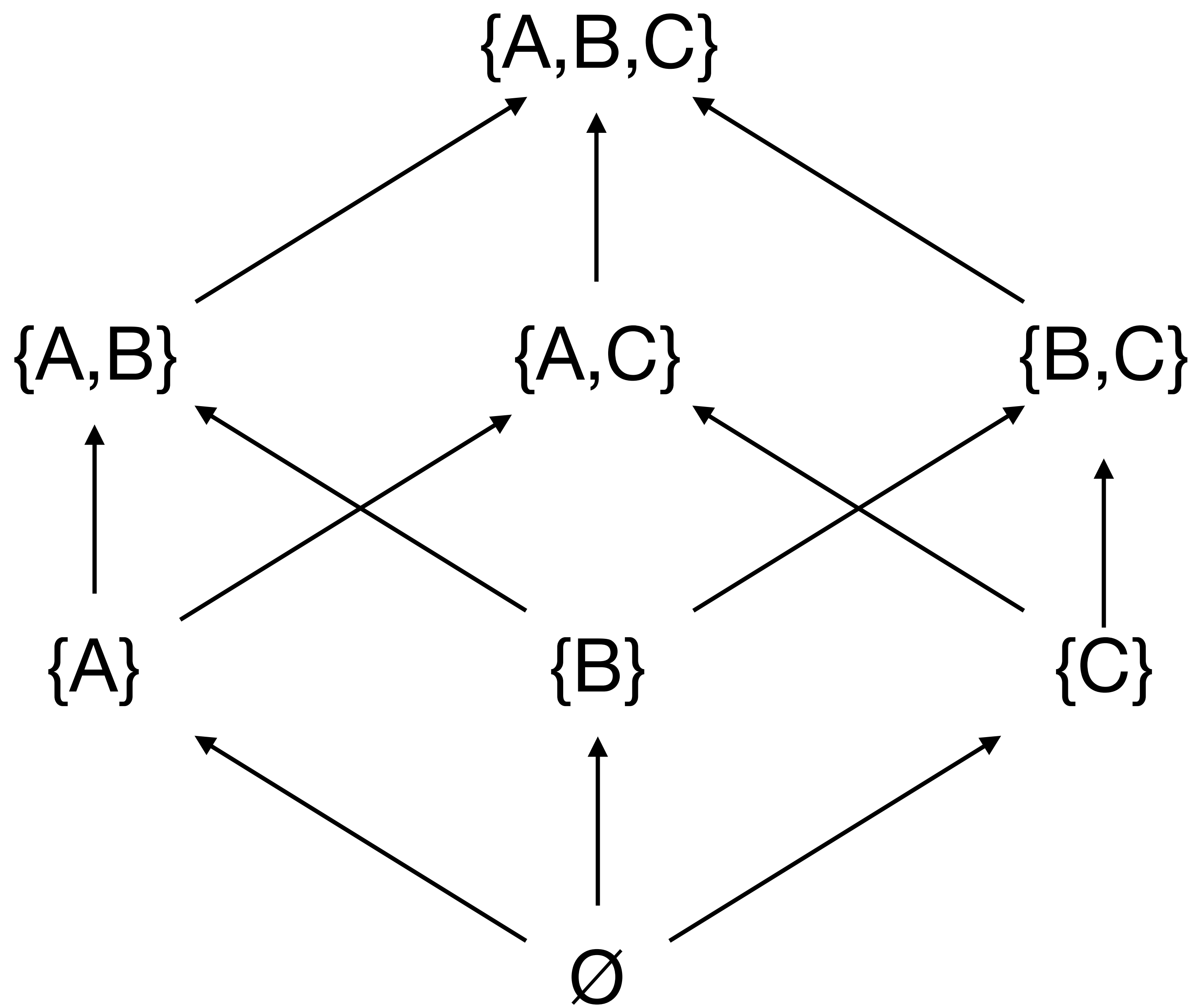


*General lattice for principals **P**:* **P** = {Alice, Bob, Charlie}

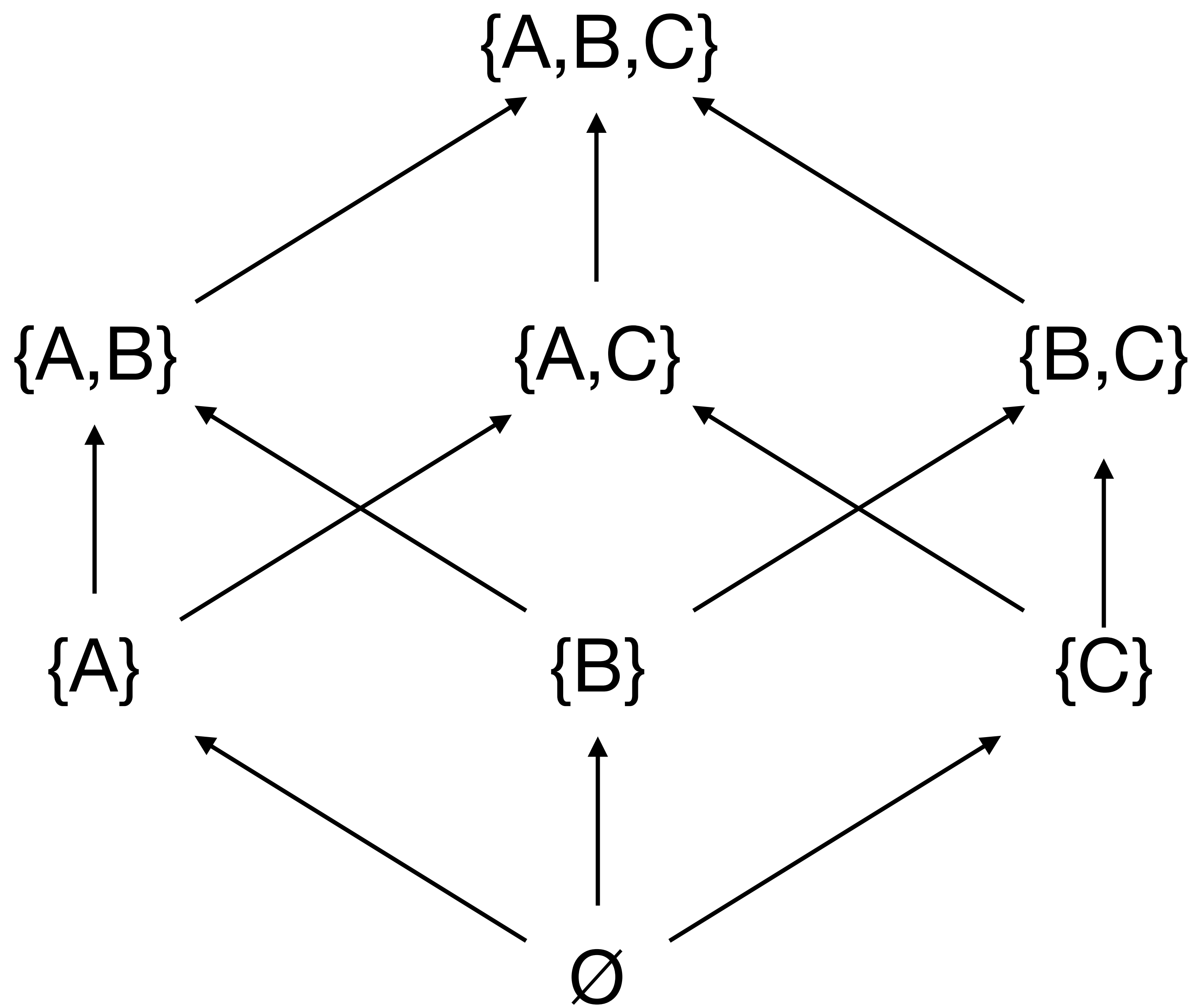


*General lattice for principals **P**:*

P = {Alice, Bob, Charlie}

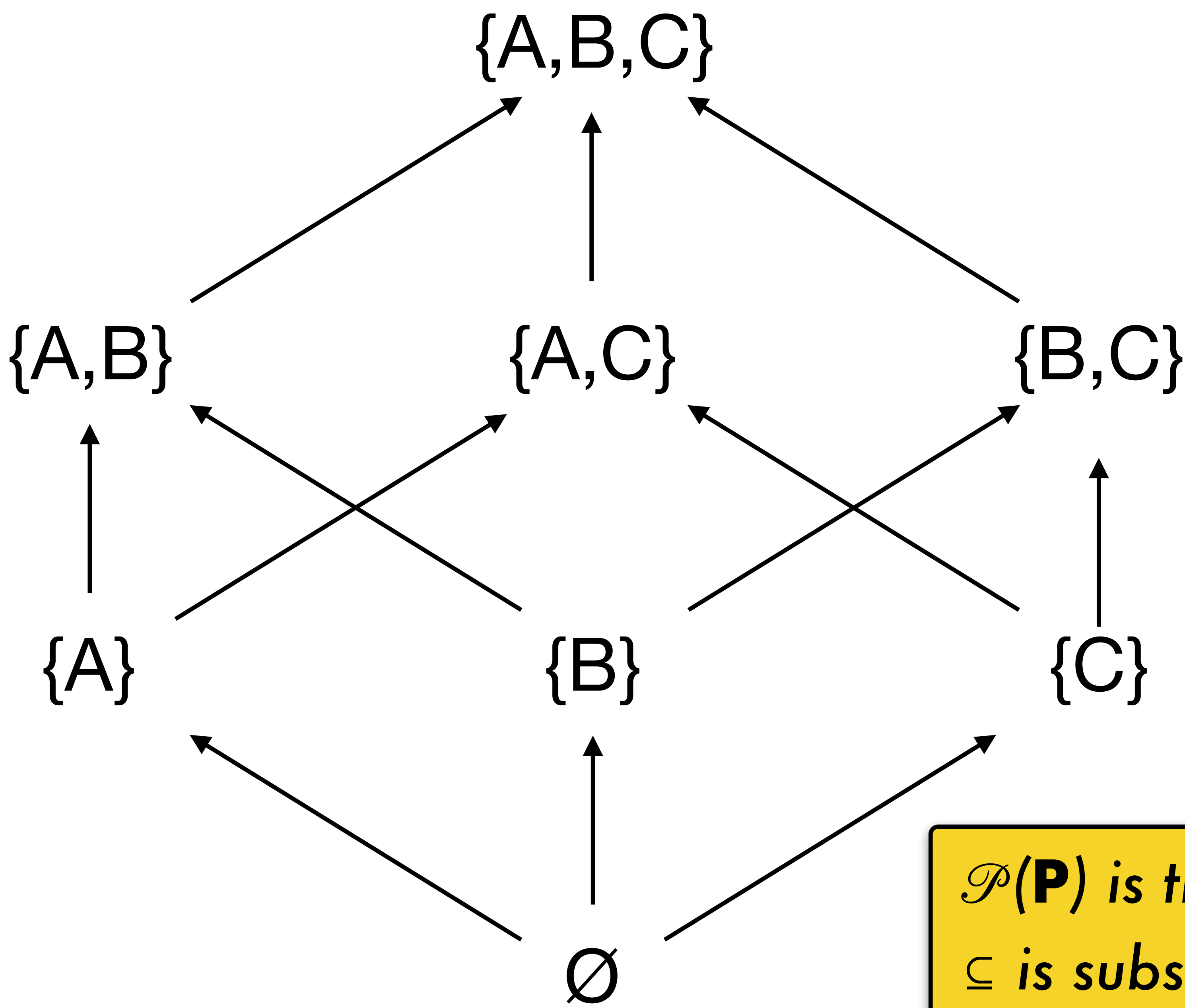


*General lattice for principals **P**:* **P** = {Alice, Bob, Charlie}



Formally: $\mathscr{L}^{\mathbf{P}} = (\mathscr{P}(\mathbf{P}) , \subseteq , \cup)$

*General lattice for principals **P**:* **P** = {Alice, Bob, Charlie}



$\mathscr{P}(\mathbf{P})$ *is the power set of* **P**

$\subseteq$ *is subset inclusion*

$\cup$ *is set union*

Formally: $\mathscr{L}^{\mathbf{P}} = (\ \mathscr{P}(\mathbf{P})\ ,\ \subseteq\ ,\ \cup\ )$

*In general we work with an **abstract lattice** with standard properties*

$$\mathscr{L} = (\boldsymbol{L} , \sqsubseteq , \sqcup)$$

$\sqsubseteq$ *is reflexive, transitive, and antisymmetric.*

$\sqcup$ *is idempotent, commutative, and associative.*

*In general we work with an **abstract lattice** with standard properties*

$$\mathscr{L} = (\boldsymbol{L} , \sqsubseteq , \sqcup)$$

$\sqsubseteq$ *is reflexive, transitive, and antisymmetric.*

$\sqcup$ *is idempotent, commutative, and associative.*

$\bot$ *element:*

In general we work with an ***abstract lattice*** *with standard properties*

$$\mathscr{L} = (\boldsymbol{L} , \sqsubseteq , \sqcup)$$

$\sqsubseteq$ *is reflexive, transitive, and antisymmetric.*

$\sqcup$ *is idempotent, commutative, and associative.*

Bottom of the lattice

$\bot$ *element:*

In general we work with an ***abstract lattice*** *with standard properties*

$$\mathscr{L} = (\boldsymbol{L} , \sqsubseteq , \sqcup)$$

$\sqsubseteq$ *is reflexive, transitive, and antisymmetric.*

Bottom of the lattice

$\sqcup$ *is idempotent, commutative, and associative.*

$\bot$ *element:* $\forall \ell .\ \bot \sqsubseteq \ell \ \wedge\ \bot \sqcup \ell = \ell$

*In general we work with an **abstract lattice** with standard properties*

$$\mathscr{L} = (\boldsymbol{L} \, , \sqsubseteq \, , \sqcup)$$

$\sqsubseteq$ *is reflexive, transitive, and antisymmetric.*

$\sqcup$ *is idempotent, commutative, and associative.*

Bottom of the lattice

$\bot$ *element:* $\forall \, \ell \, . \;\; \bot \sqsubseteq \ell \;\; \wedge \;\; \bot \sqcup \ell = \ell$

$$\forall \; \ell_1 \;\; \ell_2 \;\; \ell_3 \;\; . \;\; \ell_1 \sqsubseteq \ell_1 \sqcup \ell_2 \;\; \wedge \;\; \ell_2 \sqsubseteq \ell_1 \sqcup \ell_2$$

In general we work with an ***abstract lattice*** *with standard properties*

$$\mathscr{L} = (\boldsymbol{L} , \sqsubseteq , \sqcup)$$

$\sqsubseteq$ *is reflexive, transitive, and antisymmetric.*

$\sqcup$ *is idempotent, commutative, and associative.*

Bottom of the lattice

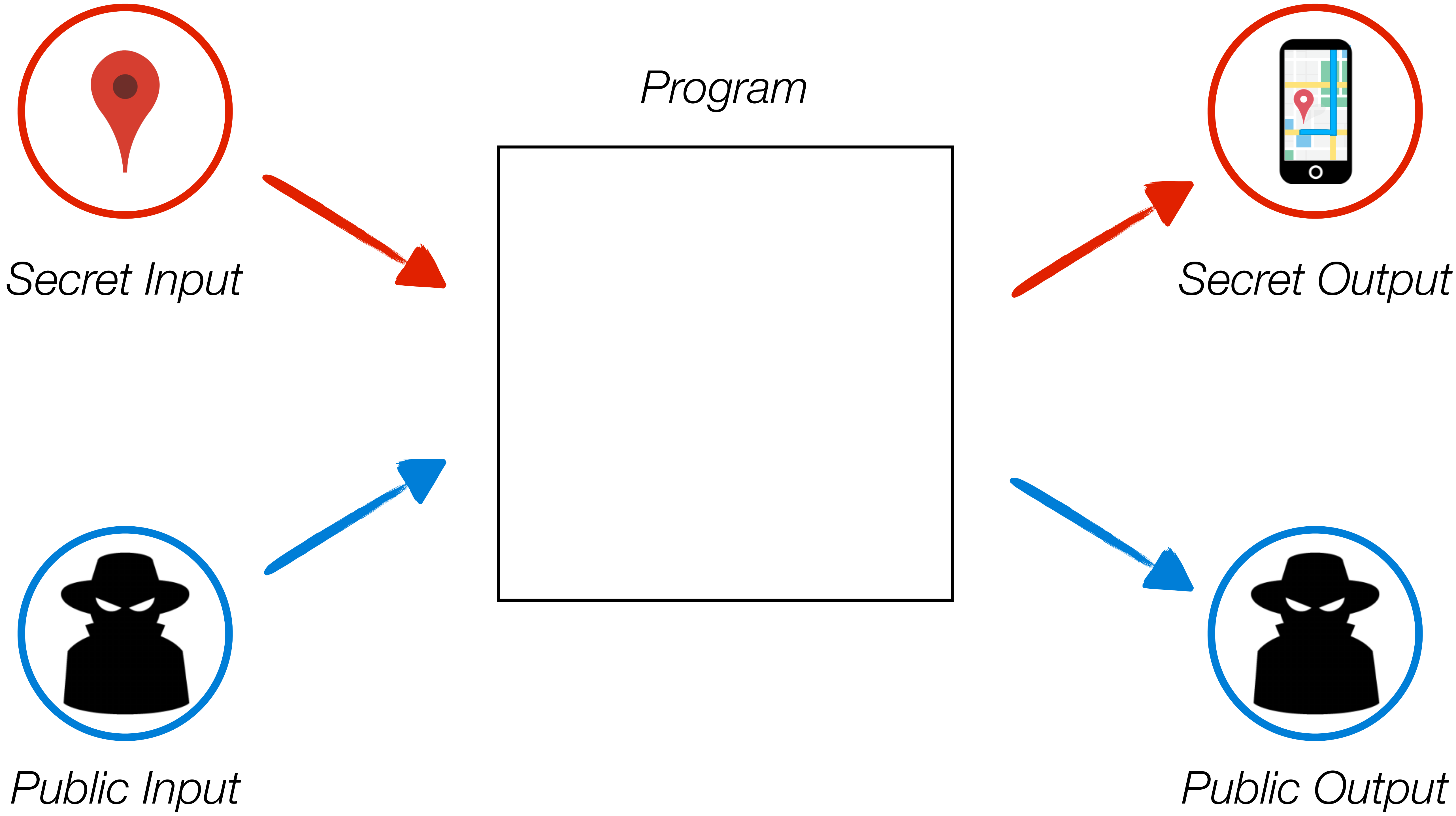
$\bot$ *element:* $\forall \ell . \ \bot \sqsubseteq \ell \ \wedge \ \bot \sqcup \ell = \ell$

Join and partial order "agree"

$$\forall \ \ell_1 \ \ell_2 \ \ell_3 \ . \ \ell_1 \sqsubseteq \ell_1 \sqcup \ell_2 \ \wedge \ \ell_2 \sqsubseteq \ell_1 \sqcup \ell_2$$

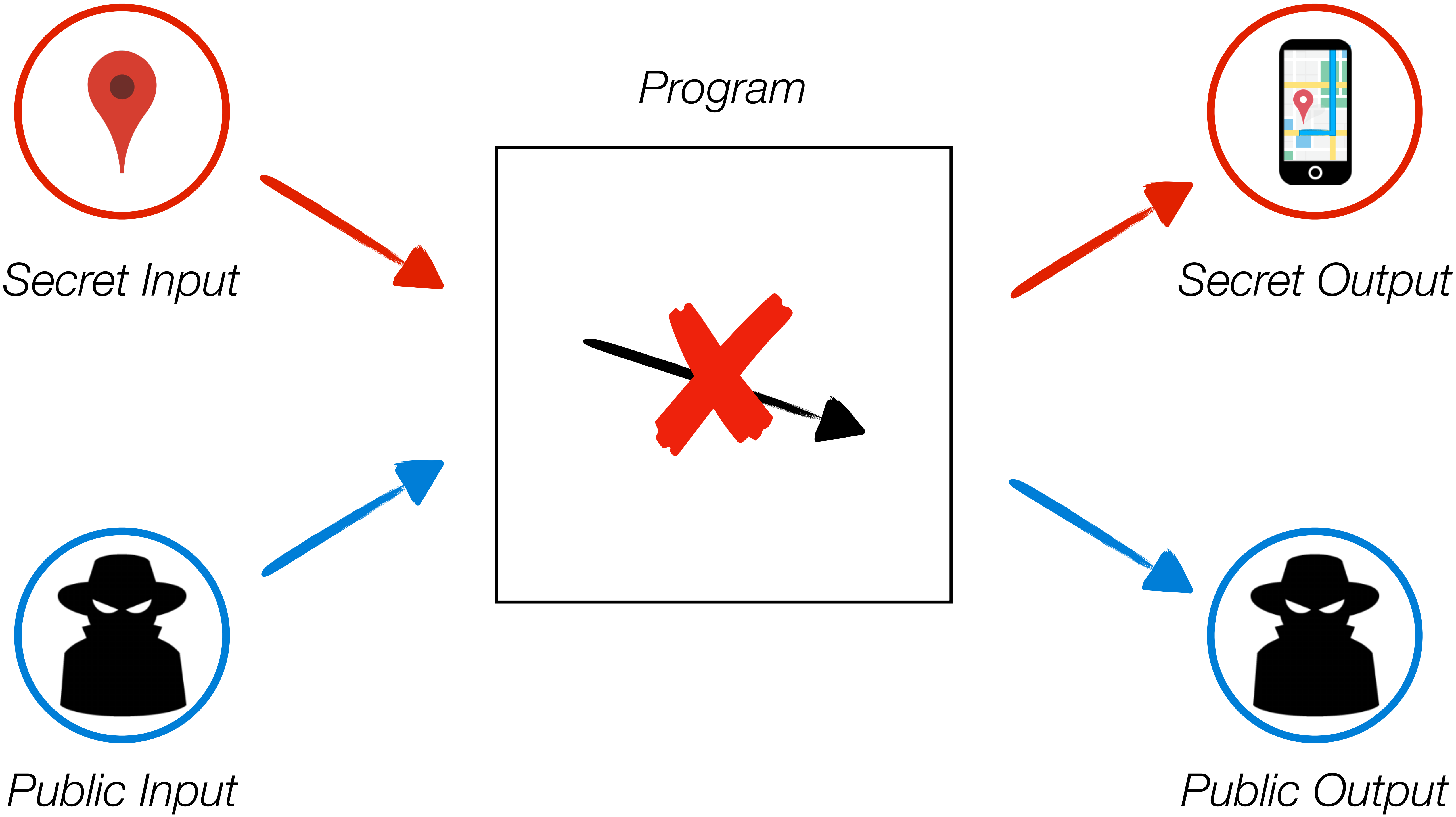
Non-Interference

Public *outputs must not depend on* ***secret*** *inputs.*



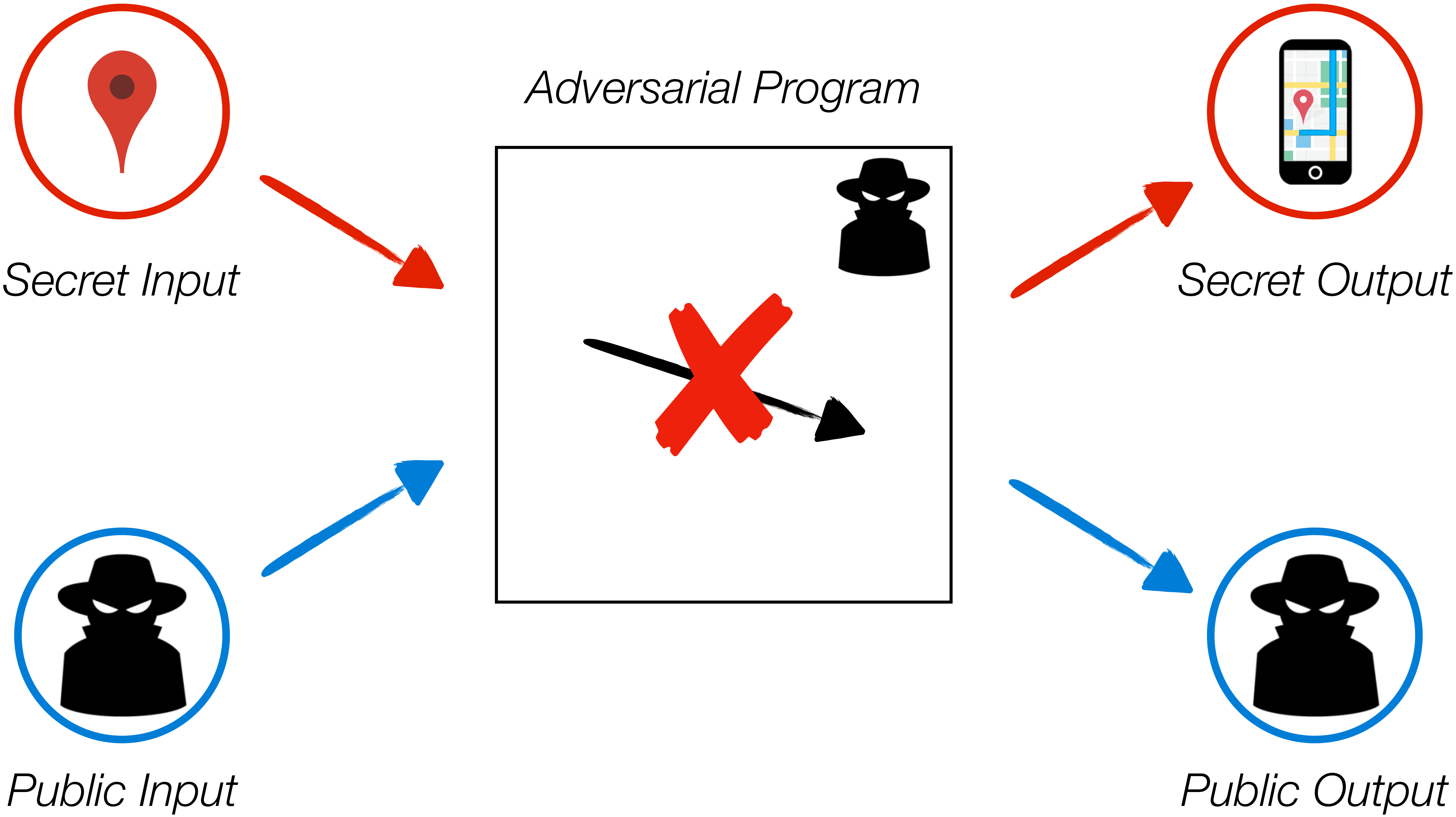
Non-Interference

Public *outputs must not depend on* ***secret*** *inputs.*



Non-Interference

Public *outputs must not depend on* ***secret*** *inputs.*



Quiz

Do the following programs satisfy non-interference?

```
h := input^H()
l := input^L()
output^H(l + h)
```

Quiz

Do the following programs satisfy non-interference?

```
h := input^H()
l := input^L()
output^H(l + h)
```

✓

Quiz

Do the following programs satisfy non-interference?

h := **input**^{H}()
l := **input**^{L}()
output^{H}(l + h)

Public and secret data can flow to secret outputs

Quiz

Do the following programs satisfy non-interference?

```
h := input^H()
l := input^L()
output^H(l + h)
```

Public and secret data can flow to secret outputs

```
h := input^H()
output^L(h + 1)
```

Quiz

Do the following programs satisfy non-interference?

```
h := input^H()
l := input^L()
output^H(l + h)
```

Public and secret data can flow to secret outputs

✓

```
h := input^H()
output^L(h + 1)
```

✗

Quiz

Do the following programs satisfy non-interference?

```
h := input^H()
l := input^L()
output^H(l + h)
```

Public and secret data can flow to secret outputs

✓

```
h := input^H()
output^L(h + 1)
```

Secret data must not flow to public outputs

✗

Quiz

Do the following programs satisfy non-interference?

```
h := input^H()
l := input^L()
output^H(l + h)
```

Public and secret data can flow to secret outputs

✓

```
h := input^H()
output^L(h + 1)
```

Secret data must not flow to public outputs

✗

This is an example of an ***explicit flow***

Quiz

Do the following programs satisfy non-interference?

```
h := input^H()
if h
  output^L(0)
```

Quiz

Do the following programs satisfy non-interference?

```
h := input^H()
if h
  output^L(0)
```

✗

Quiz

Do the following programs satisfy non-interference?

```
h := input^H()
if h
  output^L(0)
```

The presence of a public output leaks information about the secret

✗

Quiz

Do the following programs satisfy non-interference?

```
h := input^H()
if h
  output^L(0)
```

The presence of a public output leaks information about the secret

✗

This is an example of an ***implicit flow***

Quiz

Do the following programs satisfy non-interference?

```
h := input^H()
if h
  output^L(0)
```

The presence of a public output leaks information about the secret

✗

This is an example of an ***implicit flow***

```
h := input^H()
output^L(h - h)
```

Quiz

Do the following programs satisfy non-interference?

```
h := input^H()
if h
  output^L(0)
```

The presence of a public output leaks information about the secret

✗

This is an example of an ***implicit flow***

```
h := input^H()
output^L(h - h)
```

✓

Quiz

Do the following programs satisfy non-interference?

```
h := input^H()
if h
  output^L(0)
```

The presence of a public output leaks information about the secret

✗

This is an example of an ***implicit flow***

```
h := input^H()
output^L(h - h)
```

equivalent to

```
h := input^H()
output^L(0)
```

✓

Quiz

Do the following programs satisfy non-interference?

```
h := input^H()
if h
  output^L(0)
```

The presence of a public output leaks information about the secret

This is an example of an ***implicit flow***

```
h := input^H()
output^L(h - h)
```

equivalent to

```
h := input^H()
output^L(0)
```

Most IFC languages reject this program

Quiz

Do the following programs satisfy non-interference?

```
h := input^H()
if h
  output^L(0)
```

The presence of a public output leaks information about the secret

This is an example of an ***implicit flow***

```
h := input^H()
output^L(h - h)
```

equivalent to

```
h := input^H()
output^L(0)
```

False positive

Most IFC languages reject this program

Outline

Overview of different language-based IFC approaches

- Non Interference
- 4 IFC Languages

	Static	*Dynamic*
Fine-grained	$\lambda^{\mathbf{SFG}}$	$\lambda^{\mathbf{DFG}}$
Coarse-grained	$\lambda^{\mathbf{SCG}}$	$\lambda^{\mathbf{DCG}}$

Outline

Overview of different language-based IFC approaches

- Non Interference
- 4 IFC Languages

	Static	*Dynamic*
Fine-grained	$\lambda^{\textbf{SFG}}$	$\lambda^{\textbf{DFG}}$
Coarse-grained	$\lambda^{\textbf{SCG}}$	$\lambda^{\textbf{DCG}}$

Static Fine-grained IFC

$\lambda^{\textbf{SFG}}$

Syntax

Static Fine-grained IFC

$\lambda^{\textbf{SFG}}$

Syntax

Labeled Types $\tau ::= s^{\ell}$

Simple Types $s ::= \textbf{unit} \mid \tau \rightarrow \tau \mid \tau + \tau \mid \tau \times \tau$

Static Fine-grained IFC

λ^{SFG}

Syntax

Label annotation used in IFC type-system

Labeled Types $\tau ::= s^{\ell}$

Simple Types $s ::= \textbf{unit} \mid \tau \rightarrow \tau \mid \tau + \tau \mid \tau \times \tau$

Static Fine-grained IFC

$\lambda^{\textsf{SFG}}$

Syntax

Label annotation used in IFC type-system

Labeled Types	τ ::=	s^{ℓ}
Simple Types	s ::=	**unit** | $\tau \to \tau$ | $\tau + \tau$ | $\tau \times \tau$
Expressions	e ::=	() | x | λx.e | e e
		| ⟨e , e⟩ | **fst**(e) | **snd**(e)
		| **inl**(e) | **inr**(e) | **case**(e, x.e, x.e)

Static Fine-grained IFC

λ^{SFG}

Syntax

Label annotation used in IFC type-system

Labeled Types	τ ::= s^{ℓ}
Simple Types	s ::= **unit** | $\tau \to \tau$ | $\tau + \tau$ | $\tau \times \tau$
Expressions	e ::= () | x | λx.e | e e
	| ⟨e , e⟩ | **fst**(e) | **snd**(e)
	| **inl**(e) | **inr**(e) | **case**(e, x.e, x.e)
Values	v ::= () | (x.e , θ) | ⟨v , v⟩ | **inl**(v) | **inr**(v)
Environments	θ ∈ Var ⇀ Value

Static Fine-grained IFC

$\lambda^{\textsf{SFG}}$

Syntax

Label annotation used in IFC type-system

Labeled Types $\tau ::= s^{\ell}$

Simple Types $s ::= \textbf{unit} \mid \tau \rightarrow \tau \mid \tau + \tau \mid \tau \times \tau$

Expressions $e ::= () \mid x \mid \lambda x.e \mid e\ e$
$\mid \langle e\ ,\ e \rangle \mid \textbf{fst}(e) \mid \textbf{snd}(e)$
$\mid \textbf{inl}(e) \mid \textbf{inr}(e) \mid \textbf{case}(e,\ x.e,\ x.e)$

Values $v ::= () \mid (x.e\ ,\ \theta) \mid \langle v\ ,\ v \rangle \mid \textbf{inl}(v) \mid \textbf{inr}(v)$

Function Closure

Environments $\theta \in \mathrm{Var} \rightharpoonup \mathrm{Value}$

Dynamic Semantics $e \Downarrow^{\theta} v$

$\lambda^{\textbf{SFG}}$

Dynamic Semantics

$e \Downarrow^{\theta} v$

Standard: no security checks!

λ^{SFG}

Standard: no security checks!

Dynamic Semantics $e \Downarrow^{\theta} v$

Static Semantics

$\Gamma \vdash e : \tau$ *where* $\Gamma \in Var \rightarrow LTypes$

$\lambda^{\mathbf{SFG}}$

Dynamic Semantics $e \Downarrow^{\theta} v$

Standard: no security checks!

Static Semantics

Well-typed program are secure

$\Gamma \vdash e : \tau$ *where* $\Gamma \in \text{Var} \rightharpoonup \text{LTypes}$

λSFG

Dynamic Semantics $e \Downarrow^{\theta} v$

Standard: no security checks!

Static Semantics

Well-typed program are secure

$\Gamma \vdash e : \tau$ *where* $\Gamma \in Var \rightarrow LTypes$

Exercise. *Prove that the following program is **ill-typed**:*

$$\Gamma \nvdash \textbf{if } h \textbf{ then } l_1 \textbf{ else } l_2 : Bool^{L}$$

with typing environment

$$\Gamma = [\ h \mapsto Bool^{H}\ ,\ l_1 \mapsto Bool^{L}\ ,\ l_2 \mapsto Bool^{L}\ ]$$

λ^{SFG}

Dynamic Semantics $e \Downarrow^{\theta} v$

Standard: no security checks!

Static Semantics

Well-typed program are secure

$\Gamma \vdash e : \tau$ *where* $\Gamma \in \text{Var} \rightharpoonup \text{LTypes}$

Exercise. *Prove that the following program is **ill-typed**:*

$$\Gamma \nvdash \textbf{if}\ h\ \textbf{then}\ l_1\ \textbf{else}\ l_2 : \text{Bool}^{L}$$

with typing environment

$$\Gamma = [\, h \mapsto \text{Bool}^{H},\ l_1 \mapsto \text{Bool}^{L},\ l_2 \mapsto \text{Bool}^{L} \,]$$

where $\text{Bool}^{\ell} \triangleq (\textbf{unit}^{L} + \textbf{unit}^{L})^{\ell}$

$$\textbf{if}\ e\ \textbf{then}\ e_1\ \textbf{else}\ e_2 \triangleq \textbf{case}(e,\ \_.e_1,\ \_.e_2)$$

λSFG

Standard: no security checks!

Dynamic Semantics

$$e \Downarrow^{\theta} v$$

Static Semantics

Well-typed program are secure

$$\Gamma \vdash e : \tau \quad \textit{where} \quad \Gamma \in Var \rightarrow LTypes$$

Exercise. *Prove that the following program is* ***ill-typed****:*

$$\Gamma \nvdash \textbf{if}\ h\ \textbf{then}\ l_1\ \textbf{else}\ l_2 : Bool^{L}$$

with typing environment

$$\Gamma = [\ h \mapsto Bool^{H}\ ,\ l_1 \mapsto Bool^{L}\ ,\ l_2 \mapsto Bool^{L}\ ]$$

where $Bool^{\ell} \triangleq (\textbf{unit}^{L} + \textbf{unit}^{L})^{\ell}$

$$\textbf{if}\ e\ \textbf{then}\ e_1\ \textbf{else}\ e_2 \triangleq \textbf{case}(e,\ \_.e_1,\ \_.e_2)$$

Syntactic Sugar

λ^{SFG}

Static Semantics

$\Gamma \vdash e : \tau$ *where* $\Gamma \in \text{Var} \rightharpoonup \text{LTypes}$

λ^SFG

Static Semantics

$\Gamma \vdash e : \tau$ *where* $\Gamma \in \texttt{Var} \rightarrow \texttt{LTypes}$

Observations & Remarks

Elimination rules include security checks

$\lambda^{\mathbf{SFG}}$

Static Semantics

$\Gamma \vdash e : \tau$ *where* $\Gamma \in Var \rightarrow LTypes$

Observations & Remarks

Elimination rules include security checks

Avoid implicit leaks through the result

λ^{SFG}

Static Semantics

$\Gamma \vdash e : \tau$ *where* $\Gamma \in \mathtt{Var} \rightarrow \mathtt{LTypes}$

Observations & Remarks

Elimination rules include security checks

Avoid implicit leaks through the result

Introduction rules only generate label ⊥

λSFG

Static Semantics

$$\Gamma \vdash e : \tau \qquad \textit{where} \quad \Gamma \in \mathtt{Var} \rightarrow \mathtt{LTypes}$$

Observations & Remarks

Elimination rules include security checks

Avoid implicit leaks through the result

Introduction rules only generate label ⊥

Can be increased via subtyping

λSFG

Static Semantics

$$\Gamma \vdash e : \tau \qquad \textit{where} \quad \Gamma \in \texttt{Var} \rightarrow \texttt{LTypes}$$

Observations & Remarks

Elimination rules include security checks — Avoid implicit leaks through the result

Introduction rules only generate label ⊥ — Can be increased via subtyping

To state and prove non-interference we also need:

λ^{SFG}

Static Semantics

$\Gamma \vdash e : \tau$ *where* $\Gamma \in Var \rightharpoonup LTypes$

Observations & Remarks

Elimination rules include security checks

Avoid implicit leaks through the result

Introduction rules only generate label $\bot$

Can be increased via subtyping

To state and prove non-interference we also need:

$\vdash v : \tau$

λSFG

Static Semantics

$$\Gamma \vdash e : \tau \qquad \textit{where} \qquad \Gamma \in \mathrm{Var} \rightarrow \mathrm{LTypes}$$

Observations & Remarks

Elimination rules include security checks

Avoid implicit leaks through the result

Introduction rules only generate label ⊥

Can be increased via subtyping

To state and prove non-interference we also need:

$$\vdash v : \tau$$

Similar to the intro rules for expressions

λSFG

Static Semantics

$\Gamma \vdash e : \tau$ *where* $\Gamma \in \mathrm{Var} \rightharpoonup \mathrm{LTypes}$

Observations & Remarks

Elimination rules include security checks

Avoid implicit leaks through the result

Introduction rules only generate label ⊥

Can be increased via subtyping

To state and prove non-interference we also need:

$\vdash v : \tau$

Similar to the intro rules for expressions

$\vdash \theta : \Gamma$

Environment and typing contexts "agree"

Subtyping Relation

τ <: τ

$$\frac{\ell_1 \sqsubseteq \ell_2 \qquad s_1 <: s_2}{{s_1}^{\ell_1} <: {s_2}^{\ell_2}} \quad \text{[Sub-LType]}$$

Subtyping Relation

$\tau <: \tau$

$$\frac{\ell_1 \sqsubseteq \ell_2 \qquad s_1 <: s_2}{{s_1}^{\ell_1} <: {s_2}^{\ell_2}} \quad \text{[Sub-LType]}$$

$s <: s$

unit <: unit [Sub-Unit]

Subtyping Relation

τ <: τ

$$\frac{\ell_1 \sqsubseteq \ell_2 \qquad s_1 <: s_2}{{s_1}^{\ell_1} <: {s_2}^{\ell_2}} \quad \text{[Sub-LType]}$$

s <: s

$$\textbf{unit} <: \textbf{unit} \quad \text{[Sub-Unit]}$$

$$\oplus \in \{+, \times\} \; \frac{i \in \{1,2\} \qquad \tau_i <: \tau_i'}{\tau_1 \oplus \tau_2 <: \tau_1' \oplus \tau_2'}$$

Subtyping Relation

$\tau <: \tau$

$$\frac{\ell_1 \sqsubseteq \ell_2 \qquad s_1 <: s_2}{s_1^{\ell_1} <: s_2^{\ell_2}}$$ [Sub-LType]

$s <: s$

$$\mathbf{unit} <: \mathbf{unit}$$ [Sub-Unit]

$$\oplus \in \{+, \times\} \quad \frac{i \in \{1,2\} \qquad \tau_i <: \tau_i'}{\tau_1 \oplus \tau_2 <: \tau_1' \oplus \tau_2'}$$ [Sub-Sum] [Sub-Pair]

Structural for sums and pairs

Subtyping Relation

$\tau <: \tau$

$$\frac{\ell_1 \sqsubseteq \ell_2 \qquad s_1 <: s_2}{{s_1}^{\ell_1} <: {s_2}^{\ell_2}}$$ [Sub-LType]

$s <: s$

$$\textbf{unit} <: \textbf{unit}$$ [Sub-Unit]

$$\oplus \in \{+,\times\} \quad \frac{i \in \{1,2\} \qquad \tau_i <: \tau_i'}{\tau_1 \oplus \tau_2 <: \tau_1' \oplus \tau_2'}$$ [Sub-Sum]

[Sub-Pair]

$$\frac{\tau_1' <: \tau_1 \qquad \tau_2 <: \tau_2'}{\tau_1 \rightarrow \tau_2 <: \tau_1' \rightarrow \tau_2'}$$

Subtyping Relation

$\tau <: \tau$

$$\frac{\ell_1 \sqsubseteq \ell_2 \qquad s_1 <: s_2}{s_1{}^{\ell_1} <: s_2{}^{\ell_2}}$$ [Sub-LType]

$s <: s$

$$\textbf{unit} <: \textbf{unit}$$ [Sub-Unit]

$$\oplus \in \{+,\times\} \quad \frac{i \in \{1,2\} \qquad \tau_i <: \tau_i'}{\tau_1 \oplus \tau_2 <: \tau_1' \oplus \tau_2'}$$ [Sub-Sum] [Sub-Pair]

$$\frac{\tau_1' <: \tau_1 \qquad \tau_2 <: \tau_2'}{\tau_1 \to \tau_2 <: \tau_1' \to \tau_2'}$$

Covariant in the result

Subtyping Relation

$\tau <: \tau$

$$\frac{\ell_1 \sqsubseteq \ell_2 \qquad s_1 <: s_2}{s_1{}^{\ell_1} <: s_2{}^{\ell_2}} \quad \text{[Sub-LType]}$$

$s <: s$

$$\mathbf{unit} <: \mathbf{unit} \quad \text{[Sub-Unit]}$$

$$\oplus \in \{+,\times\} \quad \frac{i \in \{1,2\} \qquad \tau_i <: \tau_i'}{\tau_1 \oplus \tau_2 <: \tau_1' \oplus \tau_2'} \quad \text{[Sub-Sum] [Sub-Pair]}$$

Contravariant in the argument

$$\frac{\tau_1' <: \tau_1 \qquad \tau_2 <: \tau_2'}{\tau_1 \to \tau_2 <: \tau_1' \to \tau_2'}$$

Covariant in the result

$\boxed{\tau <: \tau}$

$$\frac{\ell_1 \sqsubseteq \ell_2 \qquad s_1 <: s_2}{{s_1}^{\ell_1} <: {s_2}^{\ell_2}} \quad \text{[Sub-LType]}$$

$\boxed{s <: s}$

$$\textbf{unit} <: \textbf{unit} \quad \text{[Sub-Unit]}$$

$$\oplus \in \{+,\times\} \; \frac{i \in \{1,2\} \qquad \tau_i <: \tau_i'}{\tau_1 \oplus \tau_2 <: \tau_1' \oplus \tau_2'} \quad \text{[Sub-Sum] [Sub-Pair]}$$

$$\frac{\tau_1' <: \tau_1 \qquad \tau_2 <: \tau_2'}{\tau_1 \to \tau_2 <: \tau_1' \to \tau_2'} \quad \text{[Sub-Fun]}$$

Exercise. *Prove that* $\texttt{Bool}^{\texttt{H}} \to \texttt{Bool}^{\texttt{L}} <: \texttt{Bool}^{\texttt{L}} \to \texttt{Bool}^{\texttt{H}}$

$\tau <: \tau$

$$\frac{\ell_1 \sqsubseteq \ell_2 \qquad s_1 <: s_2}{s_1{}^{\ell_1} <: s_2{}^{\ell_2}} \quad \text{[Sub-LType]}$$

$s <: s$

$$\texttt{unit} <: \texttt{unit} \quad \text{[Sub-Unit]}$$

$$\oplus \in \{+,\times\} \; \frac{i \in \{1,2\} \qquad \tau_i <: \tau_i'}{\tau_1 \oplus \tau_2 <: \tau_1' \oplus \tau_2'} \quad \text{[Sub-Sum] [Sub-Pair]}$$

$$\frac{\tau_1' <: \tau_1 \qquad \tau_2 <: \tau_2'}{\tau_1 \to \tau_2 <: \tau_1' \to \tau_2'} \quad \text{[Sub-Fun]}$$

Non-Interference for $\lambda^{\mathbf{SFG}}$

For all $\lambda^{\mathbf{SFG}}$ types, expressions, and values such that:

$$\mathtt{x} : \tau \vdash \mathtt{e} : \mathtt{Bool}^{\mathbf{L}}$$

Non-Interference for λ^{SFG}

For all λ^{SFG} *types, expressions, and values such that:*

Secret *input*

$$x : \tau \vdash e : \text{Bool}^{L}$$

Non-Interference for λ^{SFG}

For all λ^{SFG} types, expressions, and values such that:

Secret *input*

Public *output*

$$x : \tau \vdash e : \mathtt{Bool}^{L}$$

Non-Interference for λ^{SFG}

For all λ^{SFG} types, expressions, and values such that:

***Secret** input*

***Public** output*

$$\texttt{x} : \tau \vdash \texttt{e} : \texttt{Bool}^{L}$$

where

Non-Interference for λ^{SFG}

For all λ^{SFG} types, expressions, and values such that:

***Secret** input*

***Public** output*

$$x : \tau \vdash e : \texttt{Bool}^{L}$$

where

L *is the attacker security level*

Non-Interference for λ^{SFG}

For all λ^{SFG} types, expressions, and values such that:

***Secret** input*

***Public** output*

$$x : \tau \vdash e : \mathtt{Bool}^{L}$$

where

L *is the attacker security level*

τ *is **not** observable by the attacker:*

Non-Interference for $\lambda^{\mathbf{SFG}}$

For all $\lambda^{\mathbf{SFG}}$ types, expressions, and values such that:

***Secret** input*

***Public** output*

$$x : \tau \vdash e : \mathtt{Bool}^{\mathbf{L}}$$

where

L *is the attacker security level*

τ *is **not** observable by the attacker:*

$\tau = s^{\ell}$ *such that* $\ell \not\sqsubseteq \mathbf{L}$

Non-Interference for $\lambda^{\mathbf{SFG}}$

For all $\lambda^{\mathbf{SFG}}$ *types, expressions, and values such that:*

$$x : \tau \vdash e : \mathtt{Bool}^{\mathbf{L}}$$

Non-Interference for $\lambda^{\mathbf{SFG}}$

For all $\lambda^{\mathbf{SFG}}$ *types, expressions, and values such that:*

$$x : \tau \vdash e : \mathtt{Bool}^{\mathbf{L}}$$

$$v_1 : \tau$$

$$v_2 : \tau$$

Non-Interference for λ^{SFG}

For all λ^{SFG} types, expressions, and values such that:

$$x : \tau \vdash e : \text{Bool}^{L}$$

Any 2 ***secret*** *input values*

$$v_1 : \tau$$

$$v_2 : \tau$$

Non-Interference for λ^{SFG}

For all λ^{SFG} types, expressions, and values such that:

$$x : \tau \vdash e : Bool^{L}$$

Any 2 ***secret*** *input values*

$$v_1 : \tau$$

$$v_2 : \tau$$

If

$$\left. \begin{array}{l} e \Downarrow_{[x \mapsto v_1]} v \\ e \Downarrow_{[x \mapsto v_2]} v' \end{array} \right\}$$

Non-Interference for λ^{SFG}

For all λ^{SFG} types, expressions, and values such that:

$$x : \tau \vdash e : Bool^{L}$$

Any 2 ***secret*** *input values*

$$v_1 : \tau$$

$$v_2 : \tau$$

If $\left.\begin{array}{l} e \Downarrow^{[x \mapsto v_1]} v \\ e \Downarrow^{[x \mapsto v_2]} v' \end{array}\right\}$ ***then*** $v = v'$

Non-Interference for λ^{SFG}

For all λ^{SFG} types, expressions, and values such that:

$$x : \tau \vdash e : \mathtt{Bool}^{L}$$

*Any 2 **secret** input values*

$$v_1 : \tau$$

$$v_2 : \tau$$

*Same **public** output*

If $\left.\begin{array}{l} e \Downarrow^{[x \mapsto v_1]} v \\ e \Downarrow^{[x \mapsto v_2]} v' \end{array}\right\}$ ***then*** $v = v'$

Non-Interference for λ^{SFG}

For all λ^{SFG} types, expressions, and values such that:

$$x : \tau \vdash e : Bool^{L}$$

Any 2 ***secret*** *input values*

$$v_1 : \tau$$

$$v_2 : \tau$$

If

$$e \Downarrow^{[x \mapsto v_1]} v$$

$$e \Downarrow^{[x \mapsto v_2]} v'$$

then $v = v'$

Same ***public*** *output*

*"**Public** outputs do not depend on **secret** inputs"*

Proof Technique

1 *Define a* ***logical relation*** *for programs giving* ***equal public outputs***

Proof Technique

1 *Define a **logical relation** for programs giving **equal public outputs***

$$\mathbf{E}[\![\ \tau\ ]\!]^{L} = \{ ((e_1, \theta_1), (e_2, \theta_2)) \mid$$

Proof Technique

1 *Define a* ***logical relation*** *for programs giving* ***equal public outputs***

$$\mathbf{E}[\![\, \tau \,]\!]^{L} = \{ ((e_1, \theta_1), (e_2, \theta_2)) \mid$$

$$e_1 \Downarrow^{\theta_1} v_1 \;\wedge\; e_2 \Downarrow^{\theta_2} v_2 \implies (v_1, v_2) \in \mathbf{V}[\![\, \tau \,]\!]^{L} \}$$

Proof Technique

1 *Define a **logical relation** for programs giving **equal public outputs***

$$\mathbf{E}[\![\ \tau\ ]\!]^{L} = \{ ((e_1, \theta_1), (e_2, \theta_2)) \mid e_1 \Downarrow^{\theta_1} v_1 \ \wedge\ e_2 \Downarrow^{\theta_2} v_2 \implies (v_1, v_2) \in \mathbf{V}[\![\ \tau\ ]\!]^{L} \}$$

Equivalent values at level L

Proof Technique

1 *Define a **logical relation** for programs giving **equal public outputs***

$$\mathbf{E}[\![\tau]\!]^{L} = \{ ((e_1, \theta_1), (e_2, \theta_2)) \mid e_1 \Downarrow^{\theta_1} v_1 \ \wedge \ e_2 \Downarrow^{\theta_2} v_2 \implies (v_1, v_2) \in \mathbf{V}[\![\tau]\!]^{L} \}$$

Equivalent values at level L

2 *Prove the **fundamental theorem** of logical relations*

Proof Technique

1 *Define a **logical relation** for programs giving **equal public outputs***

$$\mathbf{E}[\![\ \tau\ ]\!]^{L} = \{ ((e_1, \theta_1), (e_2, \theta_2)) \mid e_1 \Downarrow^{\theta_1} v_1 \ \wedge\ e_2 \Downarrow^{\theta_2} v_2 \implies (v_1, v_2) \in \mathbf{V}[\![\ \tau\ ]\!]^{L} \}$$

Equivalent values at level **L**

2 *Prove the **fundamental theorem** of logical relations*

If $\Gamma \vdash e : \tau$ *then*

Proof Technique

1 *Define a **logical relation** for programs giving **equal public outputs***

$$\mathbf{E}[\![\, \tau \,]\!]^{L} = \{ ((e_1, \theta_1), (e_2, \theta_2)) \mid e_1 \Downarrow^{\theta_1} v_1 \;\wedge\; e_2 \Downarrow^{\theta_2} v_2 \implies (v_1, v_2) \in \mathbf{V}[\![\, \tau \,]\!]^{L} \}$$

Equivalent values at level **L**

2 *Prove the **fundamental theorem** of logical relations*

If $\Gamma \vdash e : \tau$ *then*

$$\forall\ (\theta_1, \theta_2) \in \mathbf{I}[\![\, \Gamma \,]\!]^{L} \implies ((e, \theta_1), (e, \theta_2)) \in \mathbf{E}[\![\, \tau \,]\!]^{L}$$

Proof Technique

1 *Define a **logical relation** for programs giving **equal public outputs***

$$\mathbf{E}[\![\tau]\!]^{L} = \{ ((e_1, \theta_1), (e_2, \theta_2)) \mid e_1 \Downarrow^{\theta_1} v_1 \ \wedge \ e_2 \Downarrow^{\theta_2} v_2 \implies (v_1, v_2) \in \mathbf{V}[\![\tau]\!]^{L} \}$$

Equivalent values at level L

2 *Prove the **fundamental theorem** of logical relations*

If $\Gamma \vdash e : \tau$ *then*

$$\forall\ (\theta_1, \theta_2) \in \mathbf{I}[\![\Gamma]\!]^{L} \implies ((e, \theta_1), (e, \theta_2)) \in \mathbf{E}[\![\tau]\!]^{L}$$

Equivalent input envs at L

Proof Technique

1 *Define a **logical relation** for programs giving **equal public outputs***

$$\mathbf{E}[\![\tau]\!]^{L} = \{ ((e_1, \theta_1), (e_2, \theta_2)) \mid e_1 \Downarrow^{\theta_1} v_1 \ \wedge \ e_2 \Downarrow^{\theta_2} v_2 \implies (v_1, v_2) \in \mathbf{V}[\![\tau]\!]^{L} \}$$

Equivalent values at level L

2 *Prove the **fundamental theorem** of logical relations*

If $\Gamma \vdash e : \tau$ *then*

$$\forall \ (\theta_1, \theta_2) \in \mathbf{I}[\![\Gamma]\!]^{L} \implies ((e, \theta_1), (e, \theta_2)) \in \mathbf{E}[\![\tau]\!]^{L}$$

Equivalent input envs at L

3 *Derive non-interference as a **corollary***

λ^{SFG} with References

λ^{SFG}

Syntax with references

Simple Types $\quad s ::= \cdots \mid \textbf{Ref}\ \tau \mid \tau \xrightarrow{\ell} \tau$

$\lambda^{\textsf{SFG}}$ with References

$\lambda^{\textsf{SFG}}$

Syntax with references

Keep tracks of side-effects

Simple Types $\quad s ::= \cdots \mid \mathbf{Ref}\ \tau \mid \tau \xrightarrow{\ell} \tau$

λ^{SFG} with References

λ^{SFG}

Syntax with references

Keep tracks of side-effects

Simple Types $\texttt{s} ::= \cdots \mid \textbf{Ref}\ \tau \mid \tau \xrightarrow{\ell} \tau$

Expressions $\texttt{e} ::= \cdots \mid \textbf{new}\ \texttt{e} \mid \texttt{!e} \mid \texttt{e} \mathrel{\textbf{:=}} \texttt{e}$

λ^{SFG} with References

λ^{SFG}

Syntax with references

Keep tracks of side-effects

Simple Types	`s ::= ···	Ref τ	τ` $\xrightarrow{\ell}$ `τ`
Expressions	`e ::= ···	new e	!e	e := e`
Values	`v ::= ···	n`

λ^{SFG} with References

λ^{SFG}

Syntax with references

Keep tracks of side-effects

Simple Types	`s ::= ···	Ref τ	τ` $\xrightarrow{\ell}$ `τ`
Expressions	`e ::= ···	new e	!e	e := e`
Values	`v ::= ···	n`

Address in store

λ^{SFG} with References

λ^{SFG}

Syntax with references

Keep tracks of side-effects

Simple Types	$s ::= \cdots \mid \mathbf{Ref}\ \tau \mid \tau \xrightarrow{\ell} \tau$
Expressions	$e ::= \cdots \mid \mathbf{new}\ e \mid !e \mid e := e$
Values	$v ::= \cdots \mid n$
Store	Σ

Address in store

λ^{SFG} with References

λ^{SFG}

Syntax with references

Keep tracks of side-effects

Simple Types	$s ::= \cdots \mid \textbf{Ref}\ \tau \mid \tau \xrightarrow{\ell} \tau$
Expressions	$e ::= \cdots \mid \textbf{new}\ e \mid !e \mid e := e$
Values	$v ::= \cdots \mid n$
Store	Σ

Address in store

Dynamic Semantics

$$\langle \Sigma, e \rangle \Downarrow^{\theta} \langle \Sigma', v \rangle$$

λ^{SFG} with References

λ^{SFG}

Syntax with references

Keep tracks of side-effects

Simple Types	$s ::= \cdots \mid \mathbf{Ref}\ \tau \mid \tau \xrightarrow{\ell} \tau$
Expressions	$e ::= \cdots \mid \mathbf{new}\ e \mid\ !e \mid e\ \mathbf{:=}\ e$
Values	$v ::= \cdots \mid n$
Store	Σ

Address in store

Dynamic Semantics

$$\langle \Sigma, e \rangle \Downarrow^{\theta} \langle \Sigma', v \rangle$$

Standard

Static Semantics

$$\Gamma \vdash_{\mathbf{pc}} e : \tau$$

Static Semantics

$\Gamma \vdash_{\mathbf{pc}} e : \tau$

"Program Counter" label

λSFG

Static Semantics

$$\Gamma \vdash_{\mathbf{pc}} e : \tau$$

"Program Counter" label

*The **pc** label is a **lower bound** on the **write effects** of the program* e

$\lambda^{\mathbf{SFG}}$

Static Semantics

$$\Gamma \vdash_{\mathbf{pc}} e : \tau$$

"Program Counter" label

Program e ***cannot*** *create and write references labeled below the* **pc**

The ***pc*** *label is a* ***lower bound*** *on the* ***write effects*** *of the program* e

λSFG

Static Semantics

$\Gamma \vdash_{\mathbf{pc}} e : \tau$

Program e ***cannot*** *create and write references labeled below the* **pc**

"Program Counter" label

Eliminate implicit leaks through the store

The ***pc*** *label is a* ***lower bound*** *on the* ***write effects*** *of the program* e

Static Semantics

$\Gamma \vdash_{\mathbf{pc}} e : \tau$

Program e ***cannot*** *create and write references labeled below the* **pc**

"Program Counter" label

Eliminate implicit leaks through the store

The ***pc*** *label is a* ***lower bound*** *on the* ***write effects*** *of the program* e

Exercise. *Prove that the following program is* ***ill-typed****:*

$\Gamma \nvdash_{\mathsf{L}}$ **if** h **then** l := true **else** () : **unit**$^{\mathsf{H}}$

Static Semantics

λ^{SFG}

$$\Gamma \vdash_{\mathbf{pc}} e : \tau$$

Program e ***cannot*** *create and write references labeled below the* **pc**

"Program Counter" label

Eliminate implicit leaks through the store

The ***pc*** *label is a* ***lower bound*** *on the* ***write effects*** *of the program* e

Exercise. *Prove that the following program is* ***ill-typed****:*

$$\Gamma \nvdash_{L} \textbf{if}\ h\ \textbf{then}\ l := \mathtt{true}\ \textbf{else}\ ()\ :\ \textbf{unit}^{H}$$

with typing environment

$$\Gamma = [\ h \mapsto \mathtt{Bool}^{H}\ ,\ l \mapsto (\textbf{Ref}\ \mathtt{Bool}^{L})^{L}\ ]$$

Subtyping Relation

s <: s

$$\frac{\tau_1' <: \tau_1 \qquad \tau_2 <: \tau_2' \qquad \ell' \sqsubseteq \ell}{\tau_1 \xrightarrow{\ell} \tau_2 <: \tau_1' \xrightarrow{\ell'} \tau_2'} \quad \text{[Sub-Fun]}$$

Subtyping Relation

s <: s

Contravariant

$$\frac{\tau_1' <: \tau_1 \qquad \tau_2 <: \tau_2' \qquad \ell' \sqsubseteq \ell}{\tau_1 \xrightarrow{\ell} \tau_2 <: \tau_1' \xrightarrow{\ell'} \tau_2'} \text{[Sub-Fun]}$$

Subtyping Relation

$s <: s$

Contravariant

$$\frac{\tau_1' <: \tau_1 \qquad \tau_2 <: \tau_2' \qquad \ell' \sqsubseteq \ell}{\tau_1 \xrightarrow{\ell} \tau_2 <: \tau_1' \xrightarrow{\ell'} \tau_2'} \quad \text{[Sub-Fun]}$$

References ?

Subtyping Relation

$s <: s$

Contravariant

$$\frac{\tau_1' <: \tau_1 \qquad \tau_2 <: \tau_2' \qquad \ell' \sqsubseteq \ell}{\tau_1 \xrightarrow{\ell} \tau_2 <: \tau_1' \xrightarrow{\ell'} \tau_2'} \text{[Sub-Fun]}$$

References ?

Covariant

$$\frac{\tau <: \tau'}{\textbf{Ref}\ \tau <: \textbf{Ref}\ \tau'}$$

Subtyping Relation

s <: s

Contravariant

$$\frac{\tau_1' <: \tau_1 \qquad \tau_2 <: \tau_2' \qquad \ell' \sqsubseteq \ell}{\tau_1 \xrightarrow{\ell} \tau_2 <: \tau_1' \xrightarrow{\ell'} \tau_2'} \text{[Sub-Fun]}$$

References ?

Covariant

$$\frac{\tau <: \tau'}{\mathbf{Ref}\ \tau <: \mathbf{Ref}\ \tau'}$$

Contravariant

$$\frac{\tau' <: \tau}{\mathbf{Ref}\ \tau <: \mathbf{Ref}\ \tau'}$$

Subtyping Relation

s <: s

Contravariant

$$\frac{\tau_1' <: \tau_1 \qquad \tau_2 <: \tau_2' \qquad \ell' \sqsubseteq \ell}{\tau_1 \xrightarrow{\ell} \tau_2 <: \tau_1' \xrightarrow{\ell'} \tau_2'} \text{[Sub-Fun]}$$

References ?

Covariant

$$\frac{\tau <: \tau'}{\textbf{Ref } \tau <: \textbf{Ref } \tau'}$$

Invariant

$$\frac{}{\textbf{Ref } \tau <: \textbf{Ref } \tau}$$

Contravariant

$$\frac{\tau' <: \tau}{\textbf{Ref } \tau <: \textbf{Ref } \tau'}$$

Subtyping Relation

$s <: s$

Contravariant

$$\frac{\tau_1' <: \tau_1 \qquad \tau_2 <: \tau_2' \qquad \ell' \sqsubseteq \ell}{\tau_1 \xrightarrow{\ell} \tau_2 <: \tau_1' \xrightarrow{\ell'} \tau_2'} \text{[Sub-Fun]}$$

References ?

Covariant	Invariant	Contravariant
$\dfrac{\tau <: \tau'}{\mathbf{Ref}\ \tau <: \mathbf{Ref}\ \tau'}$	$\dfrac{}{\mathbf{Ref}\ \tau <: \mathbf{Ref}\ \tau}$	$\dfrac{\tau' <: \tau}{\mathbf{Ref}\ \tau <: \mathbf{Ref}\ \tau'}$
✗	✓	✗

Exercise

Find a well-typed program that leaks if we consider references ***covariant****:*

Covariant ✗

$$\frac{\tau <: \tau'}{\textbf{Ref}\ \tau <: \textbf{Ref}\ \tau'}$$

Find a well-typed program that leaks if we consider references ***contravariant****:*

Contravariant ✗

$$\frac{\tau' <: \tau}{\textbf{Ref}\ \tau <: \textbf{Ref}\ \tau'}$$

Soundness issues!

Covariant ✗

$$\frac{\tau <: \tau'}{\mathbf{Ref}\ \tau <: \mathbf{Ref}\ \tau'}$$

Contravariant ✗

$$\frac{\tau' <: \tau}{\mathbf{Ref}\ \tau <: \mathbf{Ref}\ \tau'}$$

Soundness issues!

Covariant ✗

$$\frac{\tau <: \tau'}{\textbf{Ref}\ \tau <: \textbf{Ref}\ \tau'}$$

`Ref Bool`^{L} *can be* ***written*** *as* `Ref Bool`^{H}

Contravariant ✗

$$\frac{\tau' <: \tau}{\textbf{Ref}\ \tau <: \textbf{Ref}\ \tau'}$$

Soundness issues!

Covariant ✗

$$\frac{\tau <: \tau'}{\textbf{Ref}\ \tau <: \textbf{Ref}\ \tau'}$$

`Ref` $\texttt{Bool}^{L}$ *can be* ***written*** *as* `Ref` $\texttt{Bool}^{H}$

```
let h_ref = l_ref in
  h_ref := h
  !l_ref
```

Contravariant ✗

$$\frac{\tau' <: \tau}{\textbf{Ref}\ \tau <: \textbf{Ref}\ \tau'}$$

Soundness issues!

Covariant ✗

$$\frac{\tau <: \tau'}{\mathbf{Ref}\ \tau <: \mathbf{Ref}\ \tau'}$$

`Ref Bool`^{L} *can be* ***written*** *as* `Ref Bool`^{H}

```
let h_ref = l_ref in
  h_ref := h
  !l_ref
```

Contravariant ✗

$$\frac{\tau' <: \tau}{\mathbf{Ref}\ \tau <: \mathbf{Ref}\ \tau'}$$

`Ref Bool`^{H} *can be* ***read*** *as* `Ref Bool`^{L}

Soundness issues!

Covariant

$$\frac{\tau <: \tau'}{\textbf{Ref}\ \tau <: \textbf{Ref}\ \tau'}$$

Ref Bool^{L} *can be* ***written*** *as* Ref Bool^{H}

```
let h_ref = l_ref in
  h_ref := h
  !l_ref
```

Contravariant

$$\frac{\tau' <: \tau}{\textbf{Ref}\ \tau <: \textbf{Ref}\ \tau'}$$

Ref Bool^{H} *can be* ***read*** *as* Ref Bool^{L}

```
let l_ref = h_ref in
  !l_ref
```

Soundness issues!

Covariant ✗

$$\frac{\tau <: \tau'}{\textbf{Ref}\ \tau <: \textbf{Ref}\ \tau'}$$

`Ref Bool`^{L} *can be* ***written*** *as* `Ref Bool`^{H}

```
let h_ref = l_ref in
  h_ref := h
  !l_ref
```

Contravariant ✗

$$\frac{\tau' <: \tau}{\textbf{Ref}\ \tau <: \textbf{Ref}\ \tau'}$$

`Ref Bool`^{H} *can be* ***read*** *as* `Ref Bool`^{L}

```
let l_ref = h_ref in
  !l_ref
```

Well-typed but leak!

Covariant ✗

`Ref Bool`^{L} *can be* ***written*** *as* `Ref Bool`^{H}

Contravariant ✗

`Ref Bool`^{H} *can be* ***read*** *as* `Ref Bool`^{L}

*References are input (**read**) and output (**write**) channels!*

Invariant ✓

$$\frac{}{\texttt{Ref}\ \tau <: \texttt{Ref}\ \tau}\quad \text{[Sub-Ref]}$$

Soundness Proof

Non-Interference for λ$^{\textbf{SFG}}$ with ***higher-order state***

Soundness Proof

The store can contain references

Non-Interference for λ^{SFG} with ***higher-order state***

Soundness Proof

The store can contain references

Non-Interference for λ^{SFG} with ***higher-order state***

↑

Step-indexed Kripke *logical relation*

Soundness Proof

The store can contain references

Non-Interference for λ^{SFG} with ***higher-order state***

↑

Avoid circular reasoning

Step-indexed Kripke *logical relation*

Soundness Proof

The store can contain references

Non-Interference for λ^{SFG} with ***higher-order state***

↑

Avoid circular reasoning

Step-indexed Kripke *logical relation*

See "On the Expressiveness and Semantics of Information Flow Types" by Rajani and Garg

Outline

Overview of different language-based IFC approaches

- Non Interference
- **4 IFC Languages**

	Static	*Dynamic*
Fine-grained	$\lambda^{\textbf{SFG}}$	$\lambda^{\textbf{DFG}}$
Coarse-grained	$\lambda^{\textbf{SCG}}$	$\lambda^{\textbf{DCG}}$

Outline

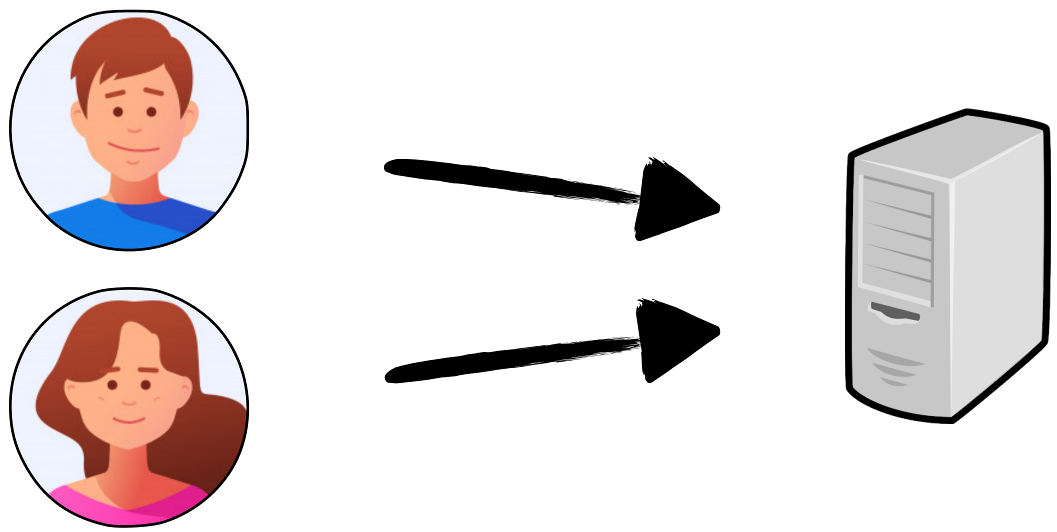
Overview of different language-based IFC approaches

- Non Interference
- **4 IFC Languages**

	Static	*Dynamic*
Fine-grained	$\lambda^{\textbf{SFG}}$	$\lambda^{\textbf{DFG}}$
Coarse-grained	$\lambda^{\textbf{SCG}}$	$\lambda^{\textbf{DCG}}$

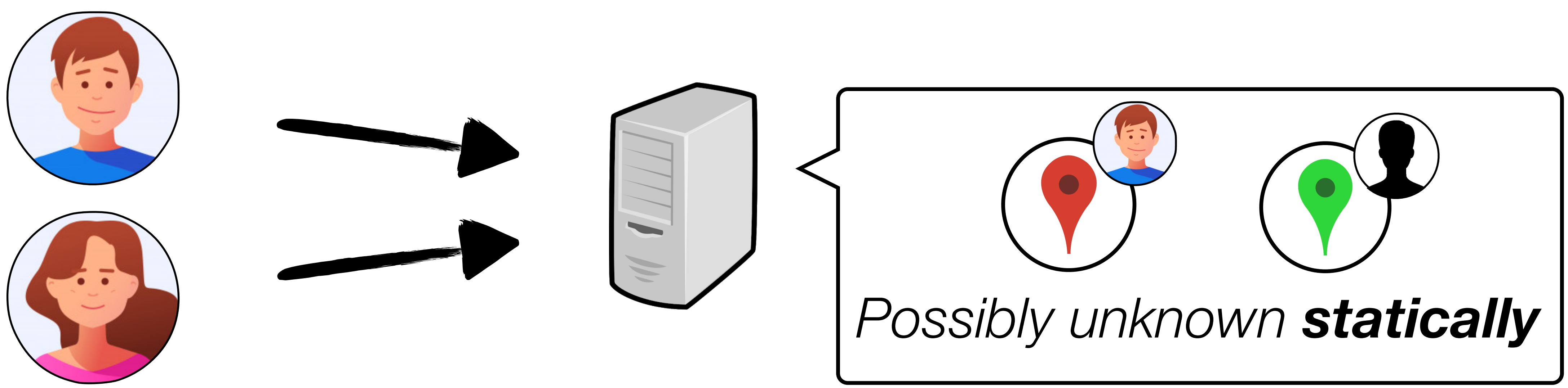
Dynamic Fine-Grained IFC

Enforce dynamic security policies

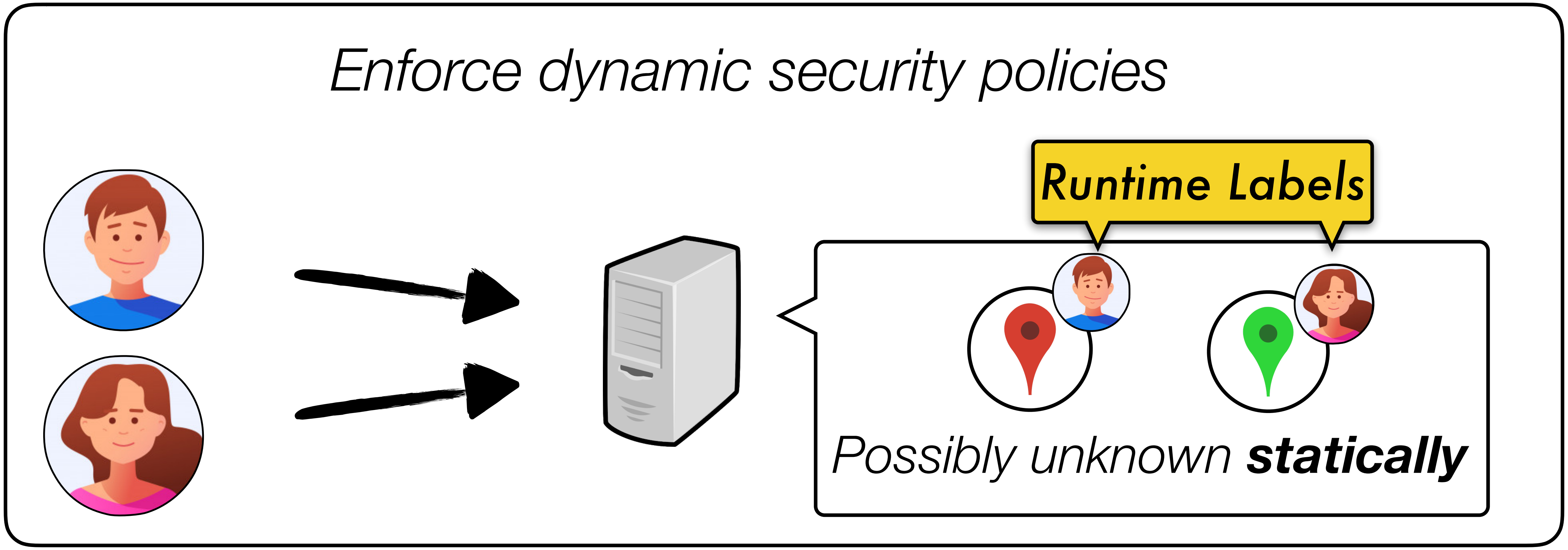


Dynamic Fine-Grained IFC

Enforce dynamic security policies

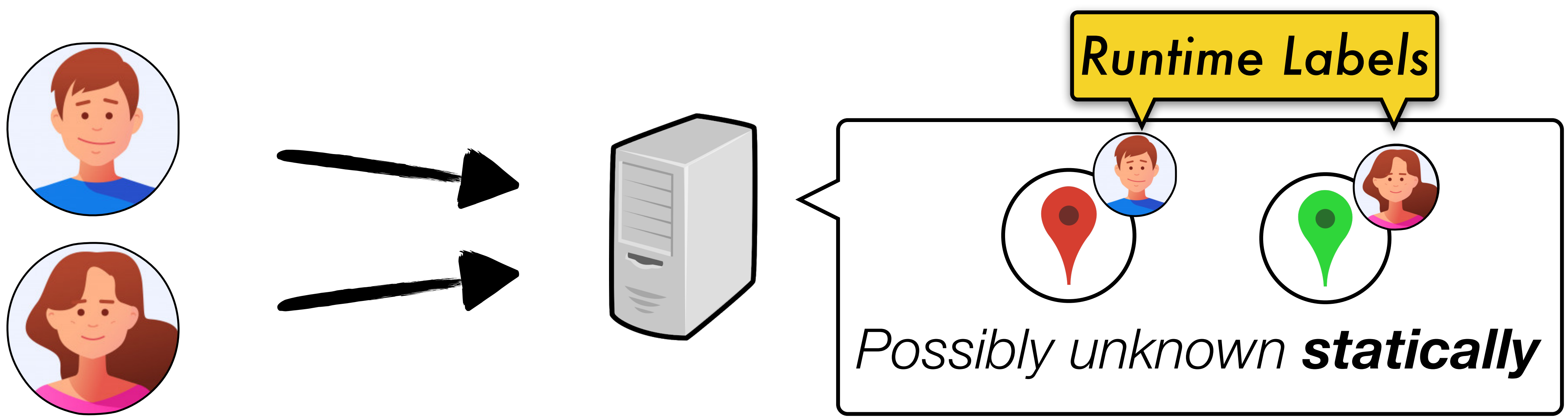


Dynamic Fine-Grained IFC

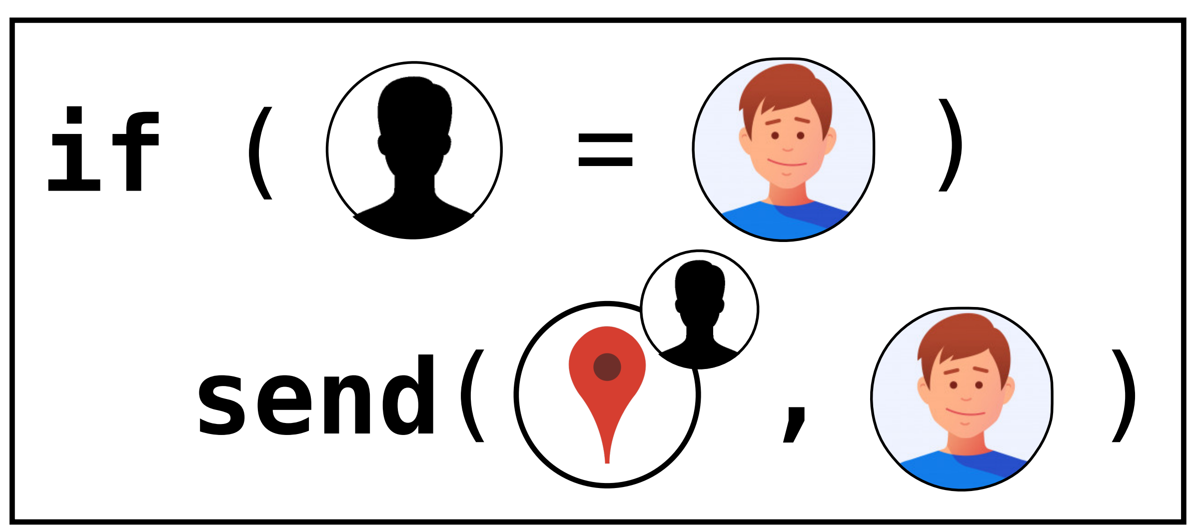


Dynamic Fine-Grained IFC

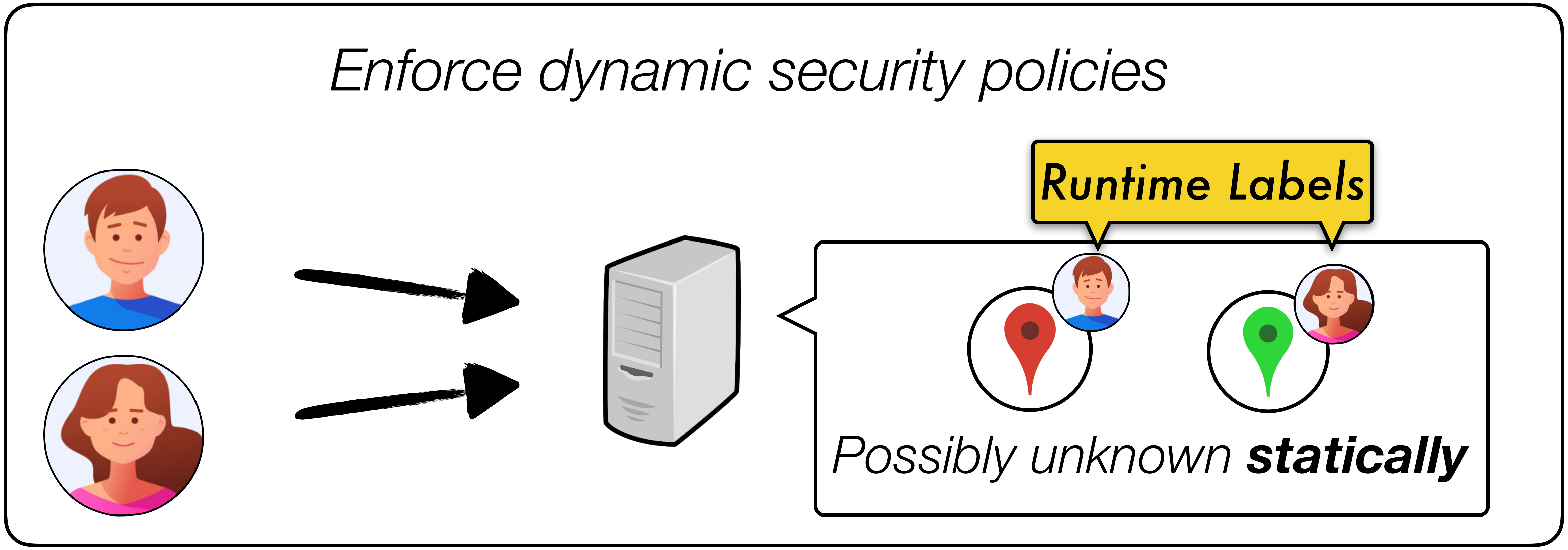
Enforce dynamic security policies

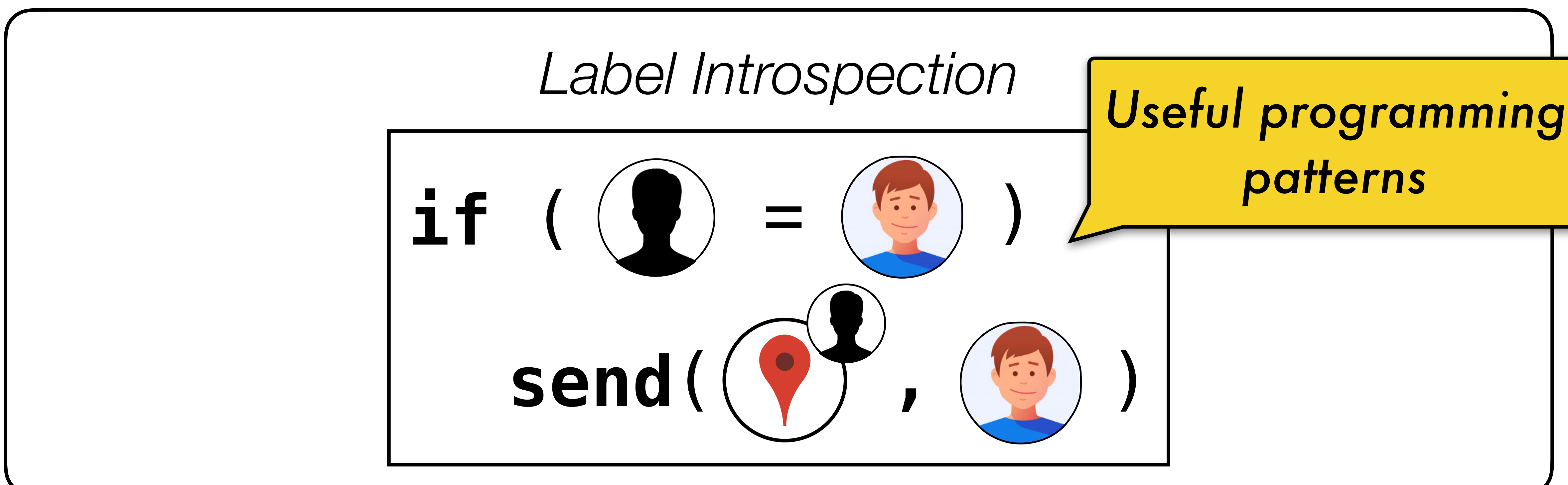


Label Introspection



Dynamic Fine-Grained IFC





Dynamic Fine-grained IFC

$\lambda^{\mathbf{DFG}}$

Syntax

Types τ ::= **unit** | $\tau \rightarrow \tau$ | $\tau + \tau$ | $\tau \times \tau$ | **Label**

Dynamic Fine-grained IFC

$\lambda^{\mathbf{DFG}}$

Syntax

New!

Types τ ::= **unit** | $\tau \rightarrow \tau$ | $\tau + \tau$ | $\tau \times \tau$ | **Label**

Dynamic Fine-grained IFC

$\lambda^{\textbf{DFG}}$

Syntax

New!

Types τ ::= **unit** | τ → τ | τ + τ | τ × τ | **Label**

Labeled Values v ::= r^{ℓ}

Dynamic Fine-grained IFC

λ^{DFG}

Syntax

New!

Types τ ::= **unit** | $\tau \to \tau$ | $\tau + \tau$ | $\tau \times \tau$ | **Label**

Labeled Values v ::= r^{ℓ}

Raw value at security level ℓ

Dynamic Fine-grained IFC

λ^{DFG}

Syntax

New!

Types	τ ::= **unit** | $\tau \to \tau$ | $\tau + \tau$ | $\tau \times \tau$ | **Label**
Labeled Values	v ::= r^{ℓ}
Raw Values	r ::= () | (x.e , θ) | ⟨v , v⟩
	| **inl**(v) | **inr**(v) | ℓ
Environments	θ ∈ Var ⇀ LValue

Raw value at security level ℓ

Dynamic Fine-grained IFC

$\lambda^{\textbf{DFG}}$

Syntax

New!

Types τ ::= **unit** | τ → τ | τ + τ | τ × τ | **Label**

Labeled Values v ::= r^{ℓ}

Raw value at security level ℓ

Raw Values r ::= () | (x.e , θ) | ⟨v , v⟩

| **inl**(v) | **inr**(v) | ℓ

Runtime labels

Environments θ ∈ Var ⇀ LValue

Dynamic Fine-grained IFC

λ^{DFG}

Syntax

New!

Types τ ::= **unit** | $\tau \rightarrow \tau$ | $\tau + \tau$ | $\tau \times \tau$ | **Label**

Labeled Values v ::= r^{ℓ}

Raw value at security level ℓ

Raw Values r ::= () | (x.e , θ) | $\langle$v , v$\rangle$

| **inl**(v) | **inr**(v) | ℓ

Runtime labels

Environments $\theta \in$ Var $\rightharpoonup$ LValue

Expressions e ::= ··· | **labelOf**(e) | **getPC** | e $\sqsubseteq$? e

Dynamic Fine-grained IFC

$\lambda^{\textbf{DFG}}$

Syntax

Types $\tau ::= \textbf{unit} \mid \tau \rightarrow \tau \mid \tau + \tau \mid \tau \times \tau \mid \textbf{Label}$ — *New!*

Labeled Values $v ::= r^{\ell}$ — *Raw value at security level* ℓ

Raw Values $r ::= () \mid (x.e\ ,\ \theta) \mid \langle v\ ,\ v \rangle$

$\mid \textbf{inl}(v) \mid \textbf{inr}(v) \mid \ell$ — *Runtime labels*

Environments $\theta \in \mathrm{Var} \rightharpoonup \mathrm{LValue}$

Label Introspection

Expressions $e ::= \cdots \mid \textbf{labelOf}(e) \mid \textbf{getPC} \mid e \sqsubseteq ?\ e$

Semantics

Static $\Gamma \vdash e : \tau$

Semantics

Standard: no security checks!

Static $\Gamma \vdash e : \tau$

Semantics

Standard: no security checks!

Static $\Gamma \vdash e : \tau$

Dynamic $e \Downarrow^{\theta}_{\mathbf{pc}} v$

λDFG

Semantics

Standard: no security checks!

Static $\Gamma \vdash e : \tau$

Security Monitor

Dynamic $e \Downarrow^{\theta}_{\mathbf{pc}} v$

Semantics

Standard: no security checks!

Static $\Gamma \vdash e : \tau$

Security Monitor

Dynamic $e \Downarrow^{\theta}_{\mathbf{pc}} v$

Program Counter

λDFG

Semantics

Standard: no security checks!

Static $\Gamma \vdash e : \tau$

Security Monitor

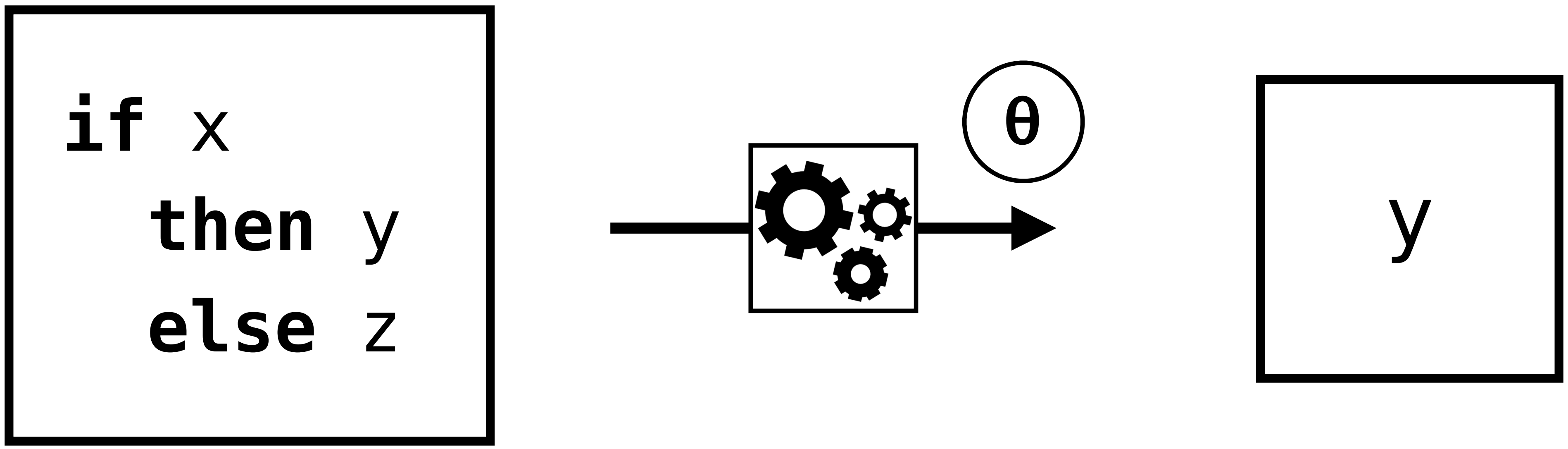
Dynamic $e \Downarrow^{\theta}_{\mathbf{pc}} v$

Program Counter

*The monitor **propagates labels** from inputs to outputs*

Label Propagation

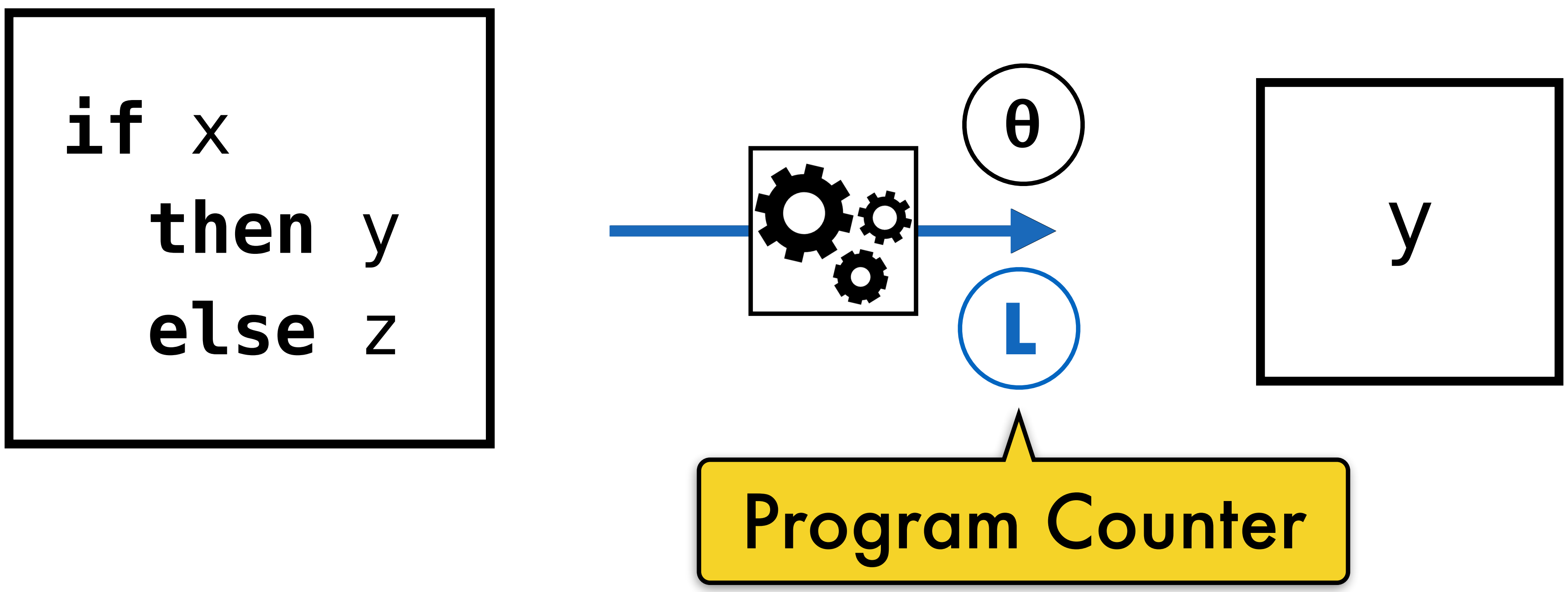
The semantics tracks control-flow dependencies with the ***program counter*** *label.*



$$\theta = [\ x \mapsto \mathbf{true}^{H},\ y \mapsto \mathbf{true}^{L},\ z \mapsto \mathbf{false}^{L}\ ]$$

Label Propagation

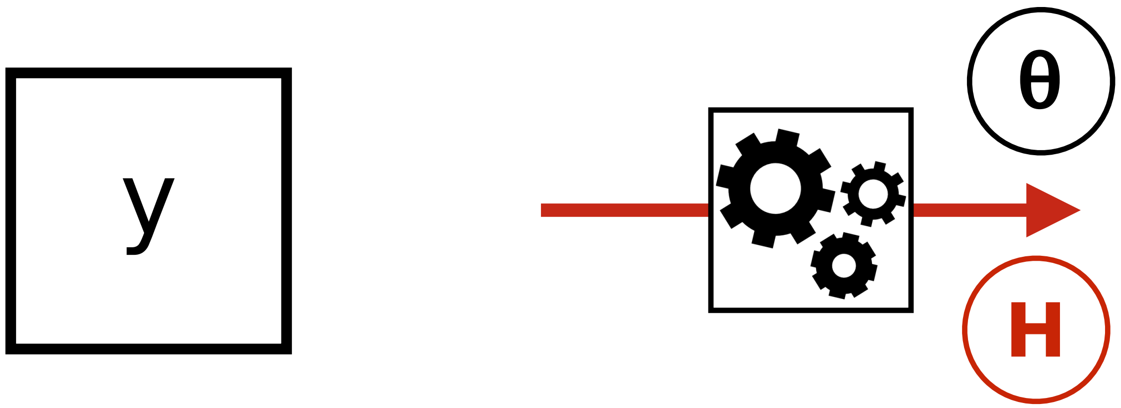
The semantics tracks control-flow dependencies with the ***program counter*** *label.*



$$\theta = [\ x \mapsto \mathbf{true}^{H},\ y \mapsto \mathbf{true}^{L},\ z \mapsto \mathbf{false}^{L}\ ]$$

Label Propagation

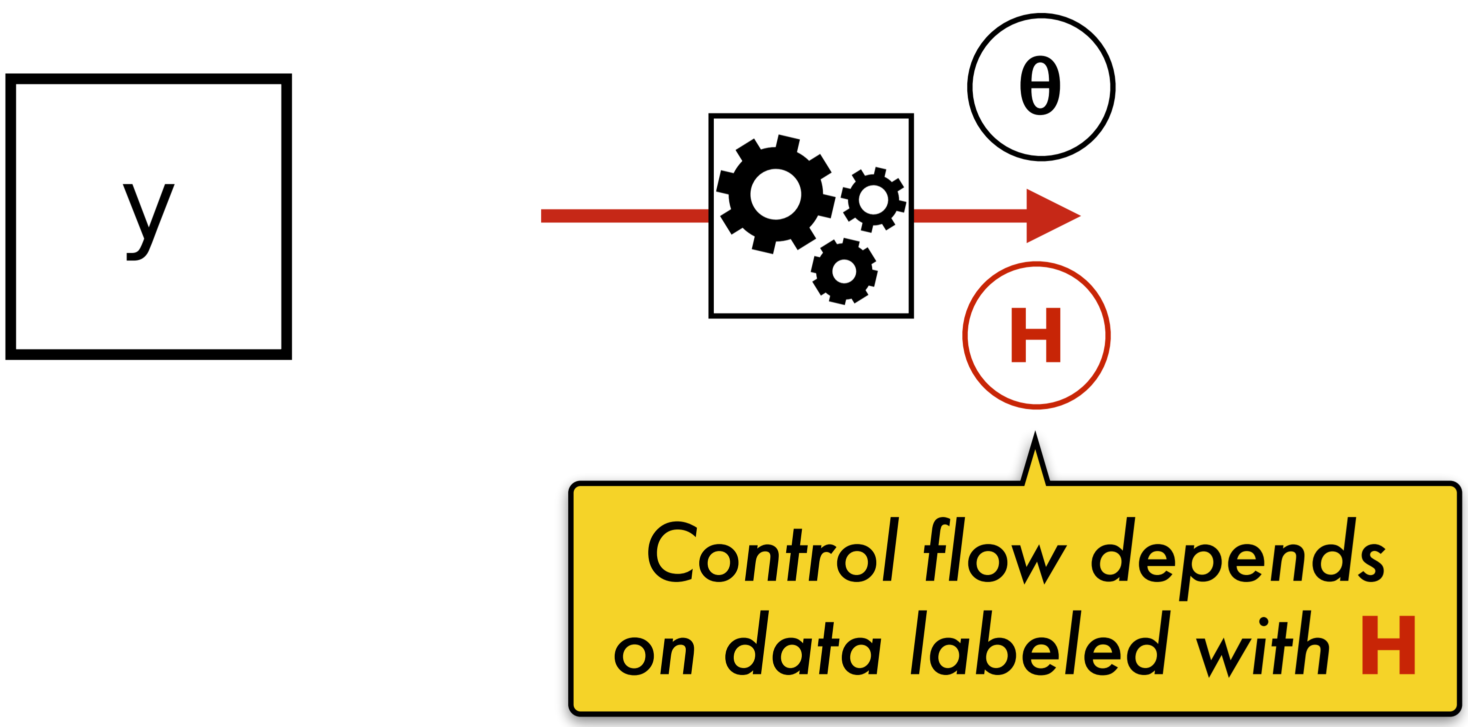
The semantics tracks control-flow dependencies with the ***program counter*** *label.*



θ = [x ↦ **true**^{H}, y ↦ **true**^{L}, z ↦ **false**^{L}]

Label Propagation

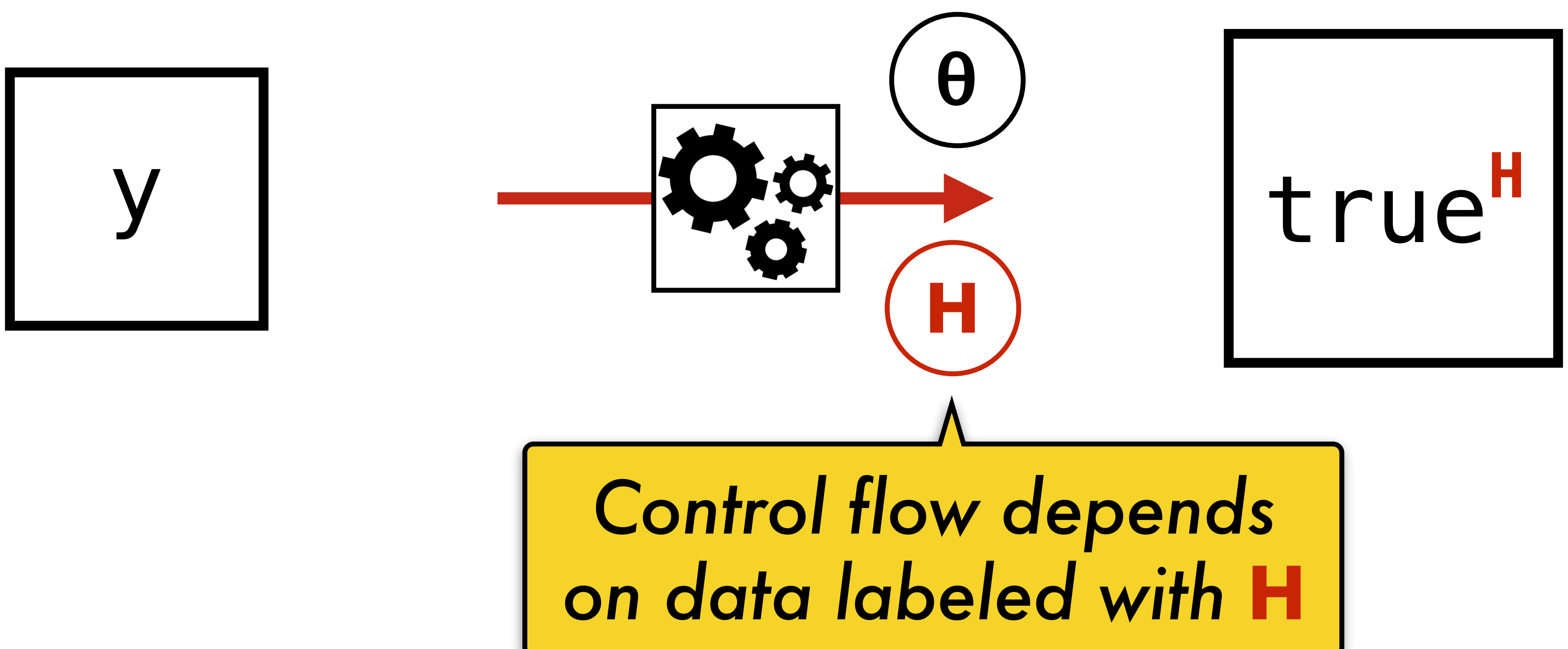
The semantics tracks control-flow dependencies
*with the **program counter** label.*



θ = [x ↦ **true**$^{\mathbf{H}}$, y ↦ **true**$^{\mathbf{L}}$, z ↦ **false**$^{\mathbf{L}}$]

Label Propagation

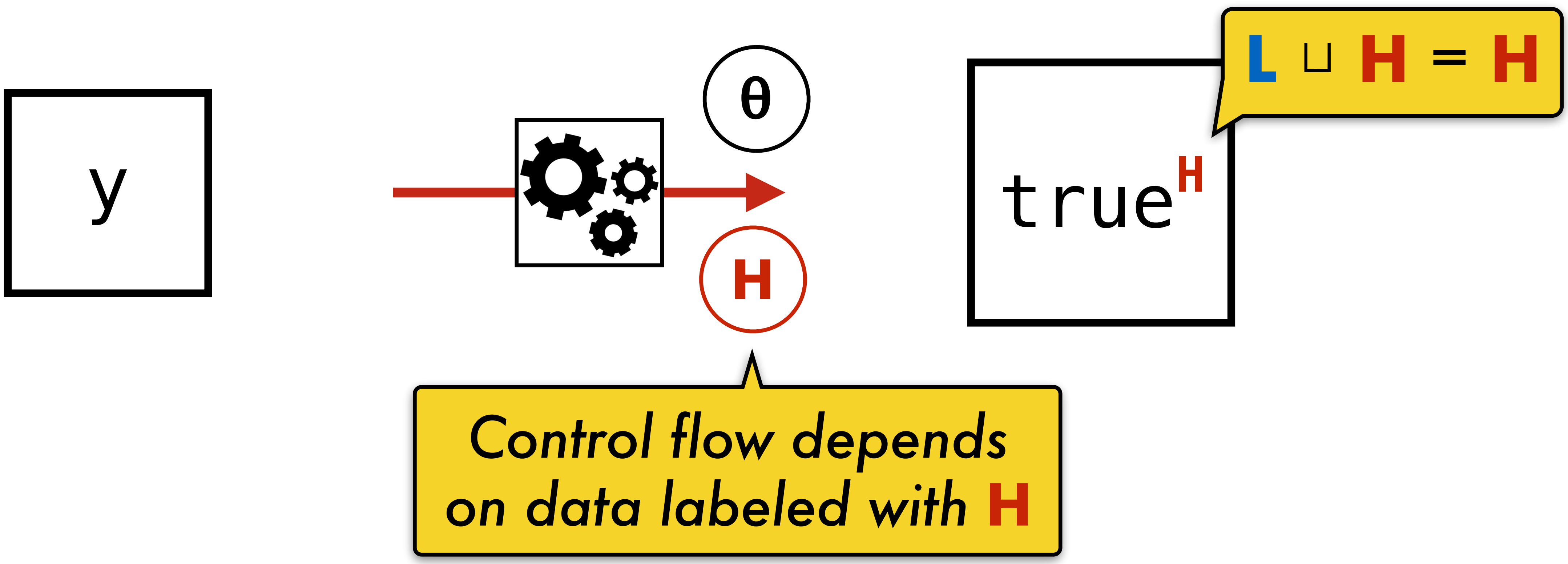
*The semantics tracks control-flow dependencies with the **program counter** label.*



θ = [x ↦ **true**^{H}, y ↦ **true**^{L}, z ↦ **false**^{L}]

Label Propagation

*The semantics tracks control-flow dependencies with the **program counter** label.*



$$\theta = [\ x \mapsto \mathbf{true}^{H},\ y \mapsto \mathbf{true}^{L},\ z \mapsto \mathbf{false}^{L}\ ]$$

Dynamic Semantics $e \Downarrow^{\theta}_{\mathbf{pc}} v$

$\lambda^{\mathbf{SFG}}$

Dynamic Semantics $e \Downarrow^{\theta}_{\mathbf{pc}} v$

Observations

*Introduction rules label the result with the **program counter***

*Elimination rules **taint** the result with the intermediate value*

λSFG

Dynamic Semantics $e \Downarrow^{\theta}_{\mathbf{pc}} v$

Observations

Introduction rules label the result with the ***program counter***

Elimination rules ***taint*** *the result with the intermediate value*

Invariant

If $e \Downarrow^{\theta}_{pc} r^{\ell}$ *then* $pc \sqsubseteq \ell$

Label Introspection

$$\textbf{labelOf}(e) \Downarrow^{\theta}_{pc}$$

Label Introspection

$$\frac{e \Downarrow^{\theta}_{pc} r^{\ell}}{\mathbf{labelOf}(e) \Downarrow^{\theta}_{pc}}$$

Label Introspection

$$\frac{\texttt{e} \;\Downarrow^{\theta}_{\texttt{pc}}\; \texttt{r}^{\ell}}{\textbf{\texttt{labelOf}}\texttt{(e)} \;\Downarrow^{\theta}_{\texttt{pc}}\; \ell}$$

Label Introspection

$$\frac{e \;\Downarrow^{\theta}_{pc}\; r^{\ell}}{\mathbf{labelOf}(e) \;\Downarrow^{\theta}_{pc}\; \ell^{\,?}}$$

What is the label of the label itself?

Label Introspection

$$\frac{e \ \Downarrow^{\theta}_{pc} \ r^{\ell}}{\mathbf{labelOf}(e) \ \Downarrow^{\theta}_{pc} \ \ell^{\ell}}$$

Label Introspection

The label has the ***same sensitivity*** *of the result!*

$$\frac{e \Downarrow^{\theta}_{pc} r^{\ell}}{\mathbf{labelOf}(e) \Downarrow^{\theta}_{pc} \ell^{\ell}}$$

Label Introspection

The label has the ***same sensitivity*** *of the result!*

$$\frac{e \;\Downarrow^{\theta}_{pc}\; r^{\ell}}{\mathtt{labelOf}(e) \;\Downarrow^{\theta}_{pc}\; \ell^{\ell}}$$

$$\mathtt{getPC} \;\Downarrow^{\theta}_{pc}\; pc^{pc}$$

λ^DFG with References

λ^DFG

Syntax with references

Simple Types τ **::=** ··· | **Ref** τ

$\lambda^{\textbf{DFG}}$ with References

$\lambda^{\textbf{DFG}}$

Syntax with references

Simple Types τ ::= ··· | **Ref** τ

Values v ::= ··· | n_ℓ

λ^{DFG} with References

λ^{DFG}

Syntax with references

Simple Types	τ ::= ··· | **Ref** τ
Values	v ::= ··· | n_{ℓ}

Reference to data labeled ℓ

$\lambda^{\textsf{DFG}}$ with References

$\lambda^{\textsf{DFG}}$

Syntax with references

Simple Types	τ ::= ··· | **Ref** τ
Values	v ::= ··· | n_ℓ
Expressions	e ::= ··· | **new** e | !e | e := e
	| **labelOfRef**(e)

Reference to data labeled ℓ

$\lambda^{\mathbf{DFG}}$ with References

$\lambda^{\mathbf{DFG}}$

Syntax with references

Simple Types	`τ ::= ···	**Ref** τ`
Values	`v ::= ··· |` n_ℓ
Expressions	`e ::= ···	**new** e	!e	e := e`
	`| **labelOfRef**(e)`

Reference to data labeled ℓ

Label introspection on refs

λ^DFG with References

λ^DFG

Syntax with references

Simple Types τ ::= ··· | **Ref** τ

Values v ::= ··· | n_ℓ

Reference to data labeled ℓ

Expressions e ::= ··· | **new** e | !e | e **:=** e
| **labelOfRef**(e)

Label introspection on refs

Store Σ ∈ (ℓ : Label) → Memory ℓ

Memory ℓ M ::= [] | r : M

λ^{DFG} with References

λ^{DFG}

Syntax with references

Simple Types τ ::= ··· | **Ref** τ

Values v ::= ··· | n_ℓ

Reference to data labeled ℓ

Expressions e ::= ··· | **new** e | !e | e := e
| **labelOfRef**(e)

Label introspection on refs

Store Σ ∈ (ℓ : Label) → Memory ℓ

Memory ℓ M ::= [] | r : M

The store is partitioned by label

Dynamic Semantics

$\langle \Sigma , e \rangle \Downarrow^{\theta}_{pc} \langle \Sigma' , v \rangle$

$$\frac{}{\langle \Sigma , \mathbf{new}\ e \rangle \Downarrow^{\theta}_{pc} \langle \Sigma'' , (n_{\ell})^{pc} \rangle}\ [\text{New}]$$

λ^{DFG}

Dynamic Semantics

$$\langle \Sigma, e \rangle \Downarrow^{\theta}_{pc} \langle \Sigma', v \rangle$$

$$\frac{\langle \Sigma, e \rangle \Downarrow^{\theta}_{pc} \langle \Sigma', r^{\ell} \rangle}{\langle \Sigma, \mathbf{new}\ e \rangle \Downarrow^{\theta}_{pc} \langle \Sigma'', (n_{\ell})^{pc} \rangle} \text{ [New]}$$

Dynamic Semantics

$\langle \Sigma , e \rangle \Downarrow^{\theta}_{pc} \langle \Sigma' , v \rangle$

Allocate in memory ℓ

$$\frac{\langle \Sigma , e \rangle \Downarrow^{\theta}_{pc} \langle \Sigma' , r^{\ell} \rangle}{\langle \Sigma , \mathbf{new}\ e \rangle \Downarrow^{\theta}_{pc} \langle \Sigma'' , (n_{\ell})^{pc} \rangle}$$

[New]

$\lambda^{\mathbf{DFG}}$

Dynamic Semantics

$\langle \Sigma , e \rangle \Downarrow^{\theta}_{pc} \langle \Sigma' , v \rangle$

Allocate in memory ℓ

$$\frac{\langle \Sigma , e \rangle \Downarrow^{\theta}_{pc} \langle \Sigma' , r^{\ell} \rangle \qquad n = |\Sigma'(\ell)|}{\langle \Sigma , \mathbf{new}\ e \rangle \Downarrow^{\theta}_{pc} \langle \Sigma'' , (n_{\ell})^{pc} \rangle} \text{[New]}$$

λ^{DFG}

Dynamic Semantics

$$\langle \Sigma, e \rangle \Downarrow^{\theta}_{pc} \langle \Sigma', v \rangle$$

Allocate in memory ℓ

Fresh Address

$$\frac{\langle \Sigma, e \rangle \Downarrow^{\theta}_{pc} \langle \Sigma', r^{\ell} \rangle \qquad n = |\Sigma'(\ell)|}{\langle \Sigma, \textbf{new}\ e \rangle \Downarrow^{\theta}_{pc} \langle \Sigma'', (n_{\ell})^{pc} \rangle}\ [\text{New}]$$

Dynamic Semantics

$$\langle \Sigma, e \rangle \Downarrow^{\theta}_{pc} \langle \Sigma', v \rangle$$

Allocate in memory ℓ

Fresh Address

$$\frac{\langle \Sigma, e \rangle \Downarrow^{\theta}_{pc} \langle \Sigma', r^{\ell} \rangle \qquad n = |\Sigma'(\ell)| \qquad \Sigma'' = \Sigma'[\ell \mapsto \Sigma'(\ell)[n \mapsto r]]}{\langle \Sigma, \mathbf{new}\ e \rangle \Downarrow^{\theta}_{pc} \langle \Sigma'', (n_{\ell})^{pc} \rangle} \text{[New]}$$

Dynamic Semantics

$\langle \Sigma , e \rangle \Downarrow^{\theta}_{pc} \langle \Sigma' , v \rangle$

Allocate in memory ℓ

Fresh Address

Update the store

$$\frac{\langle \Sigma , e \rangle \Downarrow^{\theta}_{pc} \langle \Sigma' , r^{\ell} \rangle \qquad n = |\Sigma'(\ell)| \qquad \Sigma'' = \Sigma'[\ell \mapsto \Sigma'(\ell)[n \mapsto r]]}{\langle \Sigma , \mathbf{new}\ e \rangle \Downarrow^{\theta}_{pc} \langle \Sigma'' , (n_{\ell})^{pc} \rangle}\ [\text{New}]$$

Dynamic Semantics

$\langle \Sigma , e \rangle \Downarrow^{\theta}_{pc} \langle \Sigma' , v \rangle$

$$\frac{}{\langle \Sigma , !e \rangle \Downarrow^{\theta}_{pc}}\ [\text{ Read }]$$

Dynamic Semantics

$\langle \Sigma, e \rangle \Downarrow^{\theta}_{pc} \langle \Sigma', v \rangle$

$$\frac{\langle \Sigma, e \rangle \Downarrow^{\theta}_{pc} \langle \Sigma', (n_{\ell})^{\ell'} \rangle}{\langle \Sigma, !e \rangle \Downarrow^{\theta}_{pc}} \quad [\text{ Read }]$$

λDFG

Dynamic Semantics

$$\langle \Sigma, e \rangle \Downarrow^{\theta}_{pc} \langle \Sigma', v \rangle$$

Protects the "identity" of the ref

$$\frac{\langle \Sigma, e \rangle \Downarrow^{\theta}_{pc} \langle \Sigma', (n_{\ell})^{\ell'} \rangle}{\langle \Sigma, !e \rangle \Downarrow^{\theta}_{pc}} \text{[Read]}$$

Dynamic Semantics

λ^{DFG}

$$\langle \Sigma, e\rangle \Downarrow^{\theta}_{pc} \langle \Sigma', v\rangle$$

Protects the "identity" of the ref

$$\frac{\langle \Sigma, e\rangle \Downarrow^{\theta}_{pc} \langle \Sigma', (n_{\ell})^{\ell'}\rangle \quad \Sigma'(\ell)[n] = r}{\langle \Sigma, !e\rangle \Downarrow^{\theta}_{pc}} \text{[Read]}$$

Dynamic Semantics

$\langle \Sigma, e \rangle \Downarrow^{\theta}_{pc} \langle \Sigma', v \rangle$

Protects the "identity" of the ref

$$\frac{\langle \Sigma, e \rangle \Downarrow^{\theta}_{pc} \langle \Sigma', (n_{\ell})^{\ell'} \rangle \quad \Sigma'(\ell)[n] = r}{\langle \Sigma, !e \rangle \Downarrow^{\theta}_{pc} \langle \Sigma', r^{\ell \sqcup \ell'} \rangle} \text{[Read]}$$

λDFG

Dynamic Semantics

$$\langle \Sigma, e \rangle \Downarrow^{\theta}_{pc} \langle \Sigma', v \rangle$$

Protects the "identity" of the ref

$$\frac{\langle \Sigma, e \rangle \Downarrow^{\theta}_{pc} \langle \Sigma', (n_{\ell})^{\ell'} \rangle \quad \Sigma'(\ell)[n] = r}{\langle \Sigma, !e \rangle \Downarrow^{\theta}_{pc} \langle \Sigma', r^{\ell \sqcup \ell'} \rangle} \text{[Read]}$$

Tainted with original label + identity of the ref

Dynamic Semantics

$$\langle \Sigma , e \rangle \Downarrow^{\theta}_{pc} \langle \Sigma' , v \rangle$$

$$\frac{}{\langle \Sigma , e_1 := e_2 \rangle \Downarrow^{\theta}_{pc}} \quad [\text{ Write }]$$

Dynamic Semantics

$\langle \Sigma , e \rangle \Downarrow_{pc}^{\theta} \langle \Sigma' , v \rangle$

$$\frac{\langle \Sigma , e_1 \rangle \Downarrow_{pc}^{\theta} \langle \Sigma' , (n_\ell)^{\ell_1} \rangle}{\langle \Sigma , e_1 := e_2 \rangle \Downarrow_{pc}^{\theta}}$$

[Write]

Dynamic Semantics

$\langle \Sigma , e \rangle \Downarrow^{\theta}_{pc} \langle \Sigma' , v \rangle$

$$\frac{\langle \Sigma , e_1 \rangle \Downarrow^{\theta}_{pc} \langle \Sigma' , (n_\ell)^{\ell_1} \rangle \qquad \langle \Sigma' , e_2 \rangle \Downarrow^{\theta}_{pc} \langle \Sigma'' , r^{\ell_2} \rangle}{\langle \Sigma , e_1 := e_2 \rangle \Downarrow^{\theta}_{pc}}$$

[Write]

λ^{DFG}

Dynamic Semantics

$\langle \Sigma , e \rangle \Downarrow_{pc}^{\theta} \langle \Sigma' , v \rangle$

$$\frac{\langle \Sigma , e_1 \rangle \Downarrow_{pc}^{\theta} \langle \Sigma' , (n_\ell)^{\ell_1} \rangle \qquad \ell_1 \sqsubseteq \ell \qquad \langle \Sigma' , e_2 \rangle \Downarrow_{pc}^{\theta} \langle \Sigma'' , r^{\ell_2} \rangle \qquad \ell_2 \sqsubseteq \ell}{\langle \Sigma , e_1 := e_2 \rangle \Downarrow_{pc}^{\theta}} \text{[Write]}$$

Security Checks

λDFG

Dynamic Semantics

$$\langle \Sigma, e \rangle \Downarrow^{\theta}_{pc} \langle \Sigma', v \rangle$$

$$\frac{\langle \Sigma, e_1 \rangle \Downarrow^{\theta}_{pc} \langle \Sigma', (n_\ell)^{\ell_1} \rangle \quad \ell_1 \sqsubseteq \ell \qquad \langle \Sigma', e_2 \rangle \Downarrow^{\theta}_{pc} \langle \Sigma'', r^{\ell_2} \rangle \quad \ell_2 \sqsubseteq \ell}{\langle \Sigma, e_1 := e_2 \rangle \Downarrow^{\theta}_{pc}} \text{[Write]}$$

Security Checks

$\ell_1 \sqsubseteq \ell$ *The decision of writing **this** reference must not depend on data above the label of the reference*

Dynamic Semantics

$\langle \Sigma , e \rangle \Downarrow^{\theta}_{pc} \langle \Sigma' , v \rangle$

$$\langle \Sigma , e_1 \rangle \Downarrow^{\theta}_{pc} \langle \Sigma' , (n_\ell)^{\ell_1} \rangle \qquad \ell_1 \sqsubseteq \ell$$

$$\langle \Sigma' , e_2 \rangle \Downarrow^{\theta}_{pc} \langle \Sigma'' , r^{\ell_2} \rangle \qquad \ell_2 \sqsubseteq \ell$$

Security Checks

[Write]

$$\langle \Sigma , e_1 := e_2 \rangle \Downarrow^{\theta}_{pc}$$

$\ell_1 \sqsubseteq \ell$ *The decision of writing* ***this*** *reference must not depend on data above the label of the reference*

$\ell_2 \sqsubseteq \ell$ *Must not write data above the label of the reference*

Dynamic Semantics

λ^{DFG}

$$\langle \Sigma , e \rangle \Downarrow^{\theta}_{pc} \langle \Sigma' , v \rangle$$

$$\dfrac{\langle \Sigma , e_1 \rangle \Downarrow^{\theta}_{pc} \langle \Sigma' , (n_{\ell})^{\ell_1} \rangle \quad \ell_1 \sqsubseteq \ell \quad \langle \Sigma' , e_2 \rangle \Downarrow^{\theta}_{pc} \langle \Sigma'' , r^{\ell_2} \rangle \quad \ell_2 \sqsubseteq \ell \quad \Sigma''' = \Sigma''[\ell \mapsto \Sigma''(\ell)[n \mapsto r]]}{\langle \Sigma , e_1 := e_2 \rangle \Downarrow^{\theta}_{pc} \langle \Sigma''' , ()^{pc} \rangle} \text{[Write]}$$

Security Checks

Update store

$\ell_1 \sqsubseteq \ell$ *The decision of writing **this** reference must not depend on data above the label of the reference*

$\ell_2 \sqsubseteq \ell$ *Must not write data above the label of the reference*

Proof Technique

1 *Define the **low-equivalence** relation* $v_1 \approx^{\tau}_{L} v_2$

Proof Technique

v_1 and v_2 are indistinguishable at security level L

1 *Define the **low-equivalence** relation* $v_1 \approx^{\tau}_{L} v_2$

Proof Technique

v_1 *and* v_2 *are indistinguishable at security level* **L**

1. *Define the* ***low-equivalence*** *relation* $v_1 \approx^{\tau}_{L} v_2$

2. *Prove that the semantics* ***preserves*** *the relation:*

$$\left.\begin{array}{l} \theta_1 \approx \theta_2 \\ c_1 \approx c_2 \end{array}\right\}$$

Proof Technique

v_1 *and* v_2 *are indistinguishable at security level* **L**

① *Define the* ***low-equivalence*** *relation* $v_1 \approx^{\tau}_{L} v_2$

② *Prove that the semantics* ***preserves*** *the relation:*

$$\left.\begin{array}{l} \theta_1 \approx \theta_2 \\ c_1 \approx c_2 \end{array}\right\} \textit{ if } \left\{\begin{array}{l} c_1 \Downarrow^{\theta_1}_{pc} c_1' \\ c_2 \Downarrow^{\theta_2}_{pc} c_2' \end{array}\right\}$$

Proof Technique

v_1 *and* v_2 *are indistinguishable at security level* **L**

① *Define the* ***low-equivalence*** *relation* $v_1 \approx^{\tau}_{L} v_2$

② *Prove that the semantics* ***preserves*** *the relation:*

$$\left.\begin{array}{l} \theta_1 \approx \theta_2 \\ c_1 \approx c_2 \end{array}\right\} \textit{ if } \left.\begin{array}{l} c_1 \Downarrow^{\theta_1}_{pc} c_1' \\ c_2 \Downarrow^{\theta_2}_{pc} c_2' \end{array}\right\} \textit{ then } c_1' \approx c_2'$$

Proof Technique

*v_1 and v_2 are indistinguishable at security level **L***

1. *Define the **low-equivalence** relation* $v_1 \approx^{\tau}_{L} v_2$

2. *Prove that the semantics **preserves** the relation:*

$$\left.\begin{array}{l} \theta_1 \approx \theta_2 \\ c_1 \approx c_2 \end{array}\right\} \textit{if} \quad \left.\begin{array}{l} c_1 \Downarrow^{\theta_1}_{pc} c_1' \\ c_2 \Downarrow^{\theta_2}_{pc} c_2' \end{array}\right\} \quad \textit{then} \quad c_1' \approx c_2'$$

3. *Derive non-interference as a **corollary***

Outline

Overview of different language-based IFC approaches

- Non Interference
- 4 IFC Languages

	Static	*Dynamic*
Fine-grained	$\lambda^{\textbf{SFG}}$	$\lambda^{\textbf{DFG}}$
Coarse-grained	$\lambda^{\textbf{SCG}}$	$\lambda^{\textbf{DCG}}$

Outline

Overview of different language-based IFC approaches

- **Non Interference** ✓
- **4 IFC Languages**

	Static	*Dynamic*
Fine-grained	$\lambda^{\textbf{SFG}}$	$\lambda^{\textbf{DFG}}$
Coarse-grained	$\lambda^{\textbf{SCG}}$	$\lambda^{\textbf{DCG}}$

Outline

Overview of different language-based IFC approaches

- **Non Interference** ✓
- **4 IFC Languages**

	Static	*Dynamic*	
Fine-grained	$\lambda^{\textbf{SFG}}$	$\lambda^{\textbf{DFG}}$	✓
Coarse-grained	$\lambda^{\textbf{SCG}}$	$\lambda^{\textbf{DCG}}$	

References

Introduction and Surveys

Different Variants of Non-Interference

Language-based information-flow security
Andrei Sabelfeld and Andrew C. Myers

A Perspective on Information-Flow Control
Daniel Hedin and Andrei Sabelfeld

Dynamic vs Static IFC

From dynamic to static and back:
Riding the roller coaster of information-flow control research
Andrei Sabelfeld and Alejandro Russo

Fine-Grained IFC

Static

On the Expressiveness and Semantics of Information Flow Types
Vineet Rajani and Deepak Garg

Dynamic

Efficient purely dynamic information flow analysis
Thomas H. Austin and Cormac Flanagan

Hybrid

Type-Driven Gradual Security with References
Matías Toro, Ronald Garcia, Éric Tanter

Coarse-Grained IFC

Static

MAC, A Verified Static Information-Flow Control Library
Marco Vassena, Alejandro Russo, Pablo Buiras, Lucas Waye

Dynamic

Flexible Dynamic Information Flow Control in Presence of Exceptions
Deian Stefan, Alejandro Russo, John Mitchell, and David Mazières

Hybrid

HLIO: Mixing Static and Dynamic Typing for Information-Flow Control in Haskell
Pablo Buiras, Dimitrios Vytiniotis, and Alejandro Russo

Covert Channels

Addressing Covert Termination and Timing Channels in Concurrent Information Flow Systems

Deian Stefan, Alejandro Russo, Pablo Buiras, Amit Levy, John C. Mitchell, and David Mazières

Securing Concurrent Lazy Programs Against Information Leakage

Marco Vassena, Joachim Breitner and Alejandro Russo

Foundations for Parallel Information Flow Control Runtime Systems

Marco Vassena, Gary Soeller, Peter Amidon, Matthew Chan, and Deian Stefan

From trash to treasure: timing-sensitive garbage collection

Mathias V. Pedersen and Aslan Askarov

A Library For Removing Cache-based Attacks in Concurrent Information Flow Systems

Pablo Buiras, Deian Stefan, Amit Levy, Alejandro Russo, and David Mazières

Declassification and Endorsement

Declassification: Dimensions and principles

Andrei Sabelfeld and David Sands

A Semantic Framework for Declassification and Endorsement

Aslan Askarov and Andrew C. Myers

Nonmalleable Information Flow Control

Ethan Cecchetti, Andrew C. Myers, Owen Arden