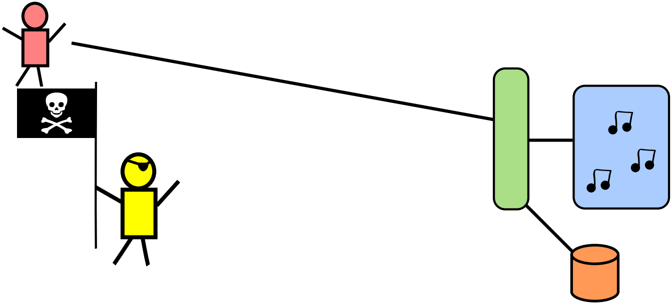
AuraConf: A Unified Approach to Authorization and Confidentiality

Jeff Vaughan

Department of Computer Science
University of California, Los Angeles

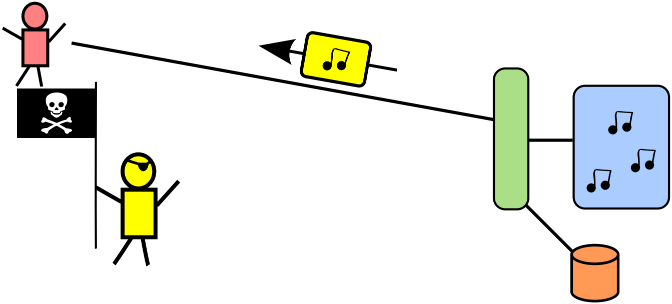
TLDI
January 25, 2011

Some attackers don't play fair.



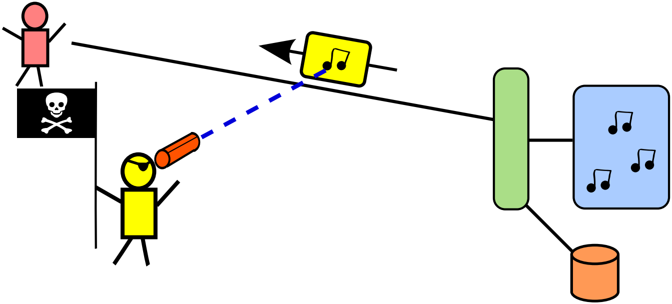
playFor: (s: Song) $\rightarrow$ (p: **prin**) $\rightarrow$
pf (RecCo **says** (MayPlay p s)) $\rightarrow$ Mp3Of s

Some attackers don't play fair.



playFor: (s: Song) → (p: **prin**) →
pf (RecCo **says** (MayPlay p s)) → Mp3Of s

Some attackers don't play fair.



playFor: (s: Song) → (p: **prin**) →
pf (RecCo **says** (MayPlay p s)) → Mp3Of s

AURA$_{conf}$ protects confidential data.

- Types provide a formal description of confidentiality policy.

AURA$_{\text{conf}}$ protects confidential data.

- Types provide a formal description of confidentiality policy.
- Encryption provides an enforcement mechanism.

AURA$_{conf}$ protects confidential data.

- Types provide a formal description of confidentiality policy.
- Encryption provides an enforcement mechanism.
- *Blame* mechanism allows audit of (some) failures.

First thought: borrow someone else's idea!

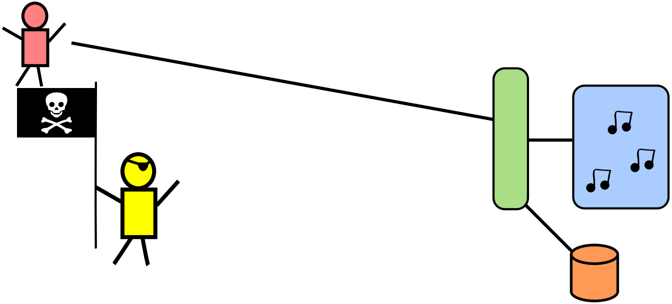
- Direct use of cryptography
  - Applied Crytpo. [Schneier '96]
- Language operations supporting cryptography
  - Spi Calculus [Abadi+ '98], λ_{seal} [Sumii+ '04]
- Type-based information flow
  - Aura [Jia & Zdancewic '09]
- Information flow + explicit cryptography
  - Key-Based DLM [Chothia+ '03], [Askarov+ '06]
- Declarative policy enforcement by automatic encryption
  - SImp [Vaughan & Zdancewic '06]

First thought: borrow someone else's idea!

- Direct use of cryptography
  - Applied Crytpo. [Schneier '96]
- Language operations supporting cryptography
  - Spi Calculus [Abadi+ '98], λ_{seal} [Sumii+ '04]
- Type-based information flow
  - Aura [Jia & Zdancewic '09]
- Information flow + explicit cryptography
  - Key-Based DLM [Chothia+ '03], [Askarov+ '06]
- Declarative policy enforcement by automatic encryption
  - SImp [Vaughan & Zdancewic '06]

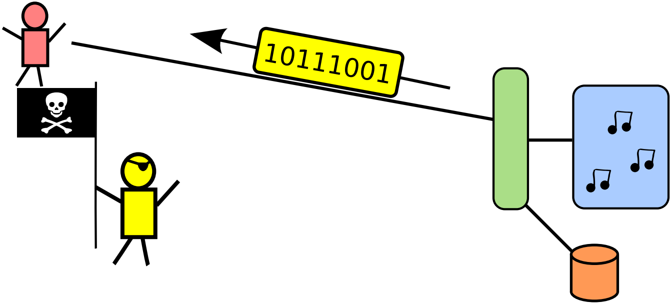
None of these are good fits with AURA.

New mechanism, **for** types describe encrypted data.



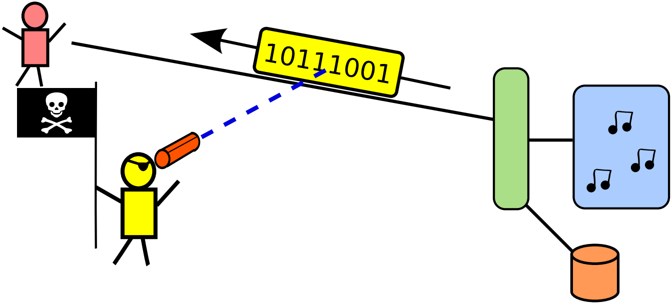
playForEnc: (s: Song) → (p: **prin**) →
pf (RecCo **says** MayPlay p s) →
(Mp3Of s) **for** p

New mechanism, **for** types describe encrypted data.



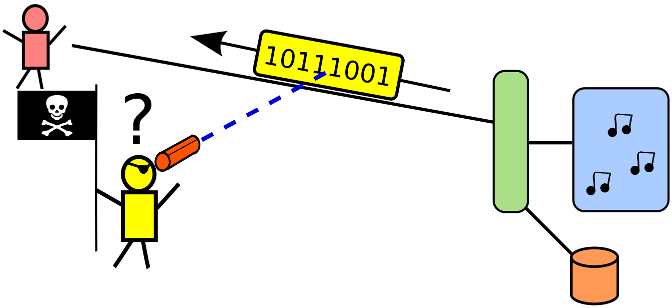
playForEnc: (s: Song) → (p: **prin**) →
pf (RecCo **says** MayPlay p s) →
(Mp3Of s) **for** p

New mechanism, **for** types describe encrypted data.



playForEnc: (s: Song) $\to$ (p: **prin**) $\to$
pf (RecCo **says** MayPlay p s) $\to$
(Mp3Of s) **for** p

New mechanism, **for** types describe encrypted data.



playForEnc: (s: Song) → (p: **prin**) →
pf (RecCo **says** MayPlay p s) →
(Mp3Of s) **for** p

Outline

Overview of for types

AURA$_{conf}$ represents confidentiality monadically: return.

return Alice 42: int **for** Alice

N.B.

Monads are a common Haskell design pattern:

- **return**: creates an object
- **run**: consumes an object
- **bind**: composes objects

AURA$_{conf}$ represents confidentiality monadically: return.

return Alice 42: int **for** Alice

$\rightsquigarrow$

$\mathscr{E}$(Alice, 42, 0x32A3)
and some metadata

N.B.

Monads are a common Haskell design pattern:

- **return**: creates an object
- **run**: consumes an object
- **bind**: composes objects

AURA$_{\text{conf}}$ represents confidentiality monadically: run.

run (**return** Alice 42): int

AURA$_{conf}$ represents confidentiality monadically: run.

run (**return** Alice 42): int

$\rightsquigarrow$

42

AURA$_{\text{conf}}$ represents confidentiality monadically: run.

$$\textbf{run}\ (\textbf{return}\ \text{Alice}\ \ 42){:}\ \ \text{int}$$

$$\rightsquigarrow$$

$$42$$

- **run** can fail on “bad” ciphertext.
  - wrong decryption key
  - ill-formed/ill-typed payload plaintext
  - corrupt ciphertext

- **run** $e \rightsquigarrow e'$ where e' blames p.

$\text{AURA}_{\text{conf}}$ represents confidentiality monadically: bind.

$$\textbf{bind}\ (\text{int}\ \ \textbf{for}\ \ \text{Alice})\ (\textbf{return}\ \text{Alice}\ \ 21)\ (\lambda\{\_\}\ x{:}\ \text{int}\ .\ \textbf{return}\ \text{Alice}\ \ (2*x))\ :\ \text{int}\ \ \textbf{for}\ \ \text{Alice}$$

AURA$_{conf}$ represents confidentiality monadically: bind.

$$\begin{array}{l}\textbf{bind}\ (\mathsf{int}\ \ \textbf{for}\ \ \mathsf{Alice})\\ \quad\quad (\textbf{return}\ \mathsf{Alice}\ \ 21)\\ \quad\quad (\lambda\{\_\}\ \mathsf{x}{:}\ \ \mathsf{int}\,.\ \ \textbf{return}\ \mathsf{Alice}\ \ (2{*}\mathsf{x}))\\ \quad\quad\quad :\ \ \mathsf{int}\ \ \textbf{for}\ \ \mathsf{Alice}\end{array}$$

$$\rightsquigarrow$$

$$\begin{array}{l}\mathscr{E}(\mathsf{Alice},\\ \quad (\lambda\{\_\}\ \mathsf{x}{:}\ \ \mathsf{int}\,.\ \ \textbf{return}\ 2{*}\mathsf{x})\ \ (\textbf{run}\ \mathscr{E}(\mathsf{Alice},\ 21,\ \mathsf{0x32A4}))\\ \quad \mathsf{0x32A3})\end{array}$$

and some metadata

AURA$_{conf}$ represents confidentiality monadically: bind.

$$\textbf{bind}\ (\text{int}\ \ \textbf{for}\ \ \text{Alice})\ (\textbf{return}\ \text{Alice}\ \ 21)\ (\lambda\{\_\}\ x{:}\ \ \text{int}\,.\ \ \textbf{return}\ \text{Alice}\ \ (2*x))\ :\ \ \text{int}\ \ \textbf{for}\ \ \text{Alice}$$

$$\rightsquigarrow$$

$$\mathscr{E}(\text{Alice},\ (\lambda\{\_\}\ x{:}\ \ \text{int}\,.\ \ \textbf{return}\ 2*x)\ (\textbf{run}\ \mathscr{E}(\text{Alice},\ 21,\ \text{0x32A4}))\ \text{0x32A3})$$

and some metadata

$$\approx \mathscr{E}(\text{Alice},\ 42,\ \text{0x32A5})$$

and some metadata

$\text{A}\textsc{ura}_{\text{conf}}$ represents confidentiality monadically: bind.

$$
\begin{array}{l}
\textbf{bind}\ (\text{int}\ \ \textbf{for}\ \ \text{Alice}) \\
\quad (\textbf{return}\ \text{Alice}\ \ 21) \\
\quad (\lambda\{\_\}\ \text{x}:\ \ \text{int}\,.\ \ \textbf{return}\ \text{Alice}\ \ (2{*}\text{x})) \\
\qquad :\ \ \text{int}\ \ \textbf{for}\ \ \text{Alice}
\end{array}
$$

$$\rightsquigarrow$$

$$
\begin{array}{l}
\mathscr{E}(\text{Alice}, \\
\quad (\lambda\{\_\}\ \text{x}:\ \ \text{int}\,.\ \ \textbf{return}\ 2{*}\text{x})\ \ (\textbf{run}\ \mathscr{E}(\text{Alice},\ 21,\ \text{0x32A4})) \\
\quad \text{0x32A3})
\end{array}
$$

and some metadata

$$\approx \mathscr{E}(\text{Alice},\ 42,\ \text{0x32A5})$$

and some metadata

This is mobile code

Static and dynamic static coupled by **for** types

- Programs may dynamically load data or code with **run**
  - Dynamic type-checking needed to catch errors
  - Ciphertexts may be paired with digitally signed proofs describing their contents
  - In case of emergency, evaluation “blames” such proofs

- Well-typed clients create values that don’t cause blame
  - Typing of **bind** makes sure mobile expressions can be correctly decrypted by the receiver
  - Receiver’s dynamic resources are modeled by sender’s typechecker

Feature design

The tension in $\text{A}\textsc{ura}_{\text{conf}}$'s design.

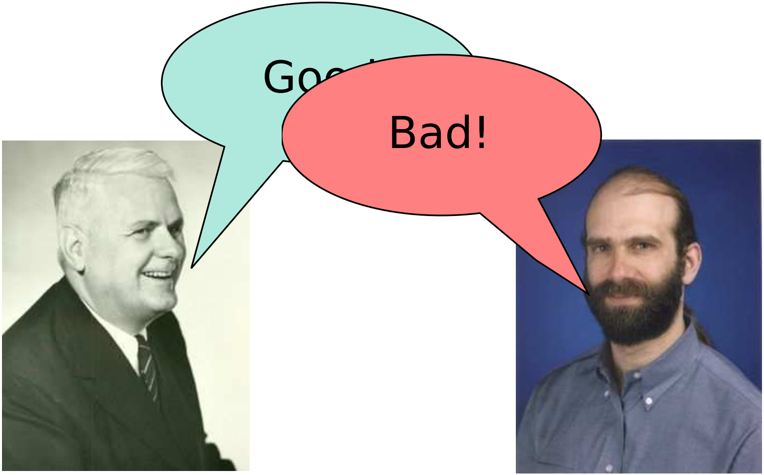
Suppose expression e contains secrets. A client analyzing e is:

Suppose expression *e* contains secrets. A client analyzing *e* is:

Type Theorist

The tension in AURA$_{conf}$'s design.

Suppose expression *e* contains secrets. A client analyzing *e* is:

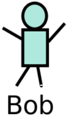


Type Theorist

Cryptographer

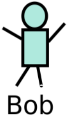
Challenge 1: Typing is relative.

return Alice "toaster"



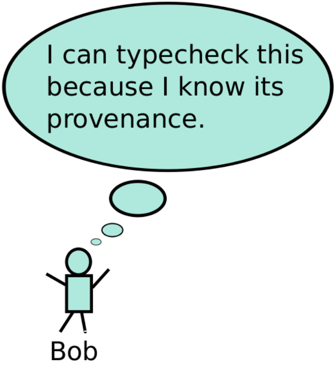
Challenge 1: Typing is relative.

$\mathscr{E}$(Alice, "toaster", 0x0312)



Challenge 1: Typing is relative.

$\mathscr{E}$(Alice, "toaster", 0x0312)



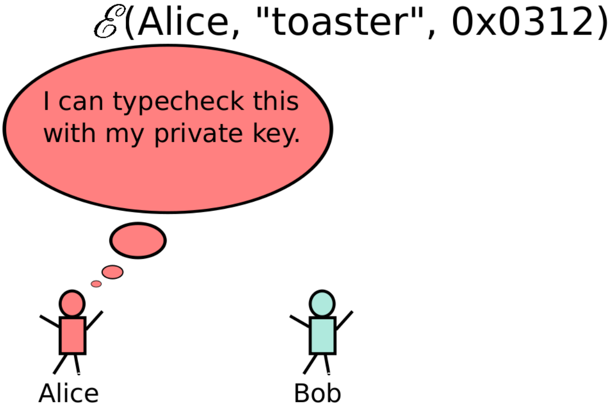
Challenge 1: Typing is relative.

$\mathscr{E}$(Alice, "toaster", 0x0312)

Challenge 1: Typing is relative.

$\mathscr{E}$(Alice, "toaster", 0x0312)

Challenge 1: Typing is relative.

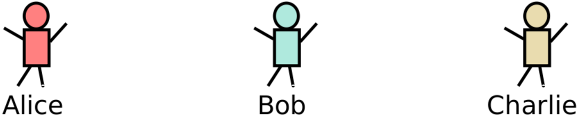


Challenge 1: Typing is relative.

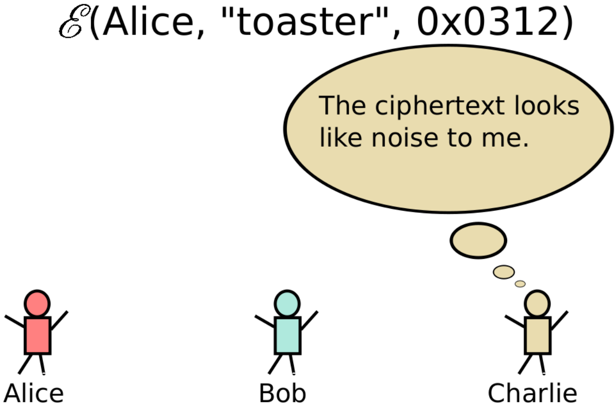
$\mathscr{E}$(Alice, "toaster", 0x0312)

Challenge 1: Typing is relative.

$\mathscr{E}$(Alice, "toaster", 0x0312)



Challenge 1: Typing is relative.



Challenge 1: Typing is relative.

$\mathscr{E}$(Alice, "toaster", 0x0312)

Metadata *casts* guide typing of ciphertexts.

True cast

cast $\mathscr{E}$(a, e, n) **to** (int **for** Alice) : int **for** Alice

- Possible if typechecker can statically decrypt $\mathscr{E}$(a,e,n).
- Also possible if the typechecker has a prerecorded *fact*, attesting to the form of $\mathscr{E}$(a,e,n).

Metadata *casts* guide typing of ciphertexts.

True cast

cast $\mathscr{E}$(a, e, n) **to** (int **for** Alice) : int **for** Alice

- Possible if typechecker can statically decrypt $\mathscr{E}$(a,e,n).
- Also possible if the typechecker has a prerecorded *fact*, attesting to the form of $\mathscr{E}$(a,e,n).

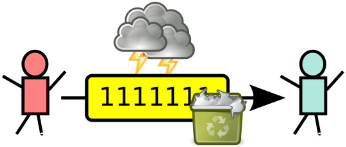
Justified cast

cast $\mathscr{E}$(a, e, n) **to** (int **for** Alice) **blaming** p: int **for** Alice

- Valid when p: c **says** ($\mathscr{E}$(a,e,n) **isa** (int **for** Alice)).
- Proof p can be blamed for decryption or typing failures.

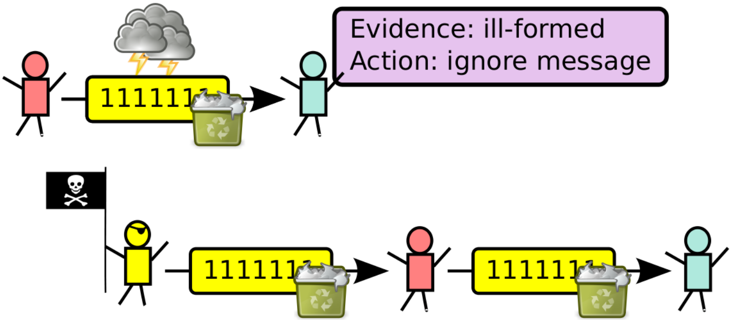
Decryption failures may be audited with justified casts.

Decryption failures may be audited with justified casts.

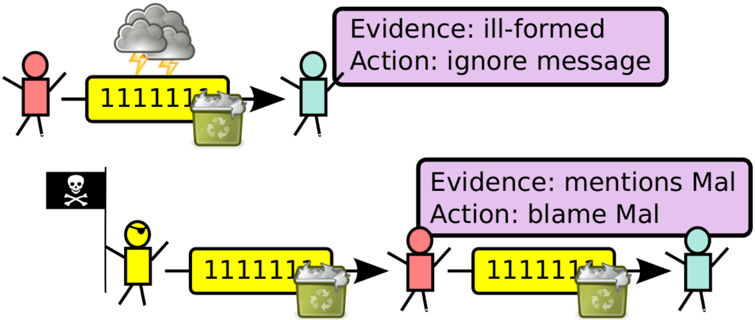


Decryption failures may be audited with justified casts.

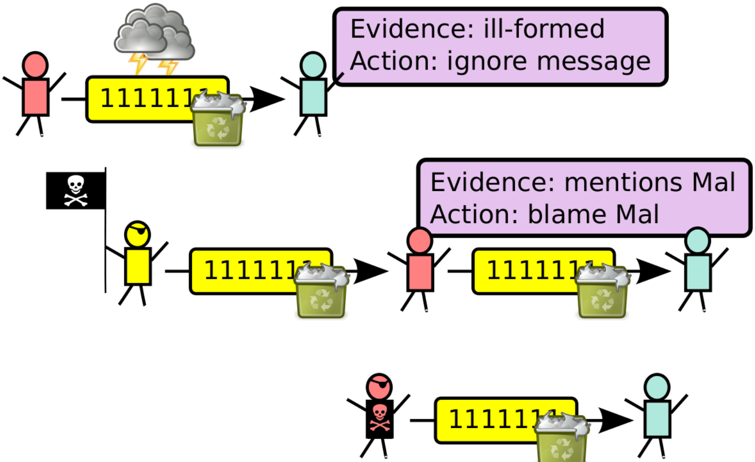
Decryption failures may be audited with justified casts.



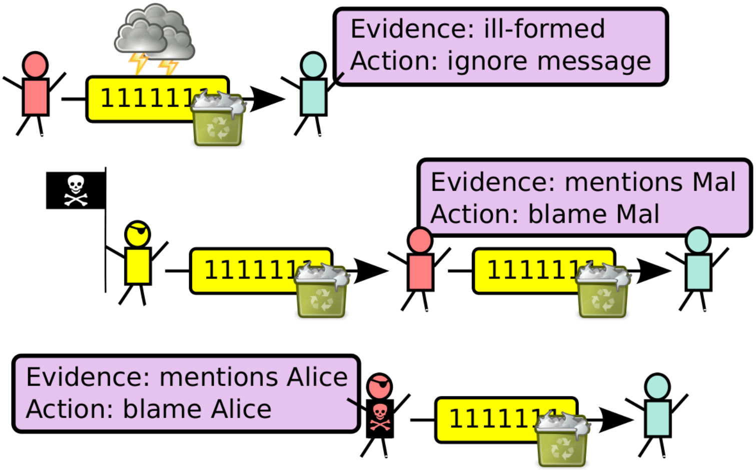
Decryption failures may be audited with justified casts.



Decryption failures may be audited with justified casts.



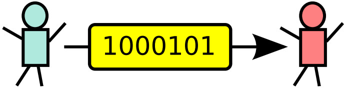
Decryption failures may be audited with justified casts.



Challenge 2: Keys affect static & dynamic semantics.

- Dynamic semantics
  - Keys are required at runtime to implement **run** (and **say**).
  - Type-and-effect analysis tracks these keys.
  - FX [Lucassen+ ’88], foundations [Talpin+ ’92]
- Static semantics
  - True casts need keys at *compile* time for typechecking.
  - Tracked using ideas from modal type systems.
  - Modal Proofs as Distributed Programs [Jia+ 04], ML5 [Murphy ’08]
- Combining these analyses is interesting!

Challenge 3: Typing exhibits history-dependence.



- Consider Bob preparing a confidential message for Alice

  return *Alice* 3 $\rightsquigarrow$ **cast** $\mathscr{E}(-)$ **to** *int* **for** *Alice*

- Naively: Bob lacks Alice's private key—he can't typecheck this.

Solution

Evaluation semantics creates new facts to guide the typechecker.

- This ensures types are preserved at runtime and programs don't "go wrong."

Language theory

Evaluation tracks fact generation and authority.

$$\Sigma; \mathscr{F}_0; W \vdash \{\!|e, n|\!\} \rightarrow \{\!|e', n'|\!\} \text{ learning } \mathscr{F}$$

Evaluation tracks fact generation and authority.

$$\Sigma; \mathscr{F}_0; W \vdash \{\!|e, n|\!\} \rightarrow \{\!|e', n'|\!\} \text{ learning } \mathscr{F}$$

- e steps to e'.

Evaluation tracks fact generation and authority.

$$\Sigma; \mathscr{F}_0; W \vdash \{\!| e, n |\!\} \rightarrow \{\!| e', n' |\!\} \text{ learning } \mathscr{F}$$

- e steps to e'.
- Randomization seed n is updated to n'.

Evaluation tracks fact generation and authority.

$$\Sigma; \mathscr{F}_0; W \vdash \{\!|e, n|\!\} \rightarrow \{\!|e', n'|\!\} \text{ learning } \mathscr{F}$$

- e steps to e'.
- Randomization seed n is updated to n'.
- Key W is available for signing and decrypting.
  "The program is running with W's authority."

Evaluation tracks fact generation and authority.

$$\Sigma; \mathscr{F}_0; W \vdash \{|e, n|\} \rightarrow \{|e', n'|\} \text{ learning } \mathscr{F}$$

- e steps to e'.
- Randomization seed n is updated to n'.
- Key W is available for signing and decrypting.
  "The program is running with W's authority."
- Signature Σ, facts context $\mathscr{F}_0$, and key W are available for dynamic type-checking.

Evaluation tracks fact generation and authority.

$$\Sigma; \mathscr{F}_0; W \vdash \{\!|e, n|\!\} \rightarrow \{\!|e', n'|\!\} \text{ learning } \mathscr{F}$$

- e steps to e'.
- Randomization seed n is updated to n'.
- Key W is available for signing and decrypting.
  "The program is running with W's authority."
- Signature Σ, facts context $\mathscr{F}_0$, and key W are available for dynamic type-checking.
- New facts $\mathscr{F}$ are produced during encryptions.

Evaluation tracks fact generation and authority.

$$\Sigma; \mathscr{F}_0; W \vdash \{\!| e, n |\!\} \rightarrow \{\!| e', n' |\!\} \text{ learning } \mathscr{F}$$

- e steps to e'.
- Randomization seed n is updated to n'.
- Key W is available for signing and decrypting.
  "The program is running with W's authority."
- Signature Σ, facts context $\mathscr{F}_0$, and key W are available for dynamic type-checking.
- New facts $\mathscr{F}$ are produced during encryptions.

Anatomy of the typing relation.

$$\Sigma; \mathscr{F}; W; \Gamma; U; V \vdash e : t$$

Anatomy of the typing relation.

$$\Sigma; \mathscr{F}; W; \Gamma; U; V \vdash e : t$$

- e has type t w.r.t. Γ's free variables and Σ's type definitions.

Anatomy of the typing relation.

$$\Sigma; \mathscr{F}; W; \Gamma; U; V \vdash e : t$$

- e has type t w.r.t. Γ's free variables and Σ's type definitions.
- Facts in $\mathscr{F}$ summarize knowledge about ciphertexts.

Anatomy of the typing relation.

$$\Sigma; \mathscr{F}; W; \Gamma; U; V \vdash e : t$$

- e has type t w.r.t. Γ's free variables and Σ's type definitions.
- Facts in $\mathscr{F}$ summarize knowledge about ciphertexts.
- *Statically available key* W indicates keys available for typechecking.

Anatomy of the typing relation.

$$\Sigma; \mathscr{F}; W; \Gamma; {\color{red}U}; V \vdash e : t$$

- e has type t w.r.t. Γ's free variables and Σ's type definitions.
- Facts in $\mathscr{F}$ summarize knowledge about ciphertexts.
- *Statically available key* W indicates keys available for typechecking.
- *Soft decryption limit* U specifies a subset of W safe to use currently.

Anatomy of the typing relation.

$$\Sigma; \mathscr{F}; W; \Gamma; U; \textcolor{red}{V} \vdash e : t$$

- e has type t w.r.t. Γ's free variables and Σ's type definitions.
- Facts in $\mathscr{F}$ summarize knowledge about ciphertexts.
- *Statically available key* W indicates keys available for typechecking.
- *Soft decryption limit* U specifies a subset of W safe to use currently.
- *Effects label* V summarizes the keys needed to run e.

Anatomy of the typing relation.

$$\Sigma; \mathscr{F}; W; \Gamma; U; V \vdash e : t$$

- *e* has type *t* w.r.t. Γ's free variables and Σ's type definitions.
- Facts in $\mathscr{F}$ summarize knowledge about ciphertexts.
- *Statically available key* W indicates keys available for typechecking.
- *Soft decryption limit* U specifies a subset of *W* safe to use currently.
- *Effects label* V summarizes the keys needed to run *e*.

Anatomy of the typing relation.

$$\Sigma; \mathscr{F}; W; \Gamma; U; V \vdash e : t$$

- e has type t w.r.t. Γ's free variables and Σ's type definitions.
- Facts in $\mathscr{F}$ summarize knowledge about ciphertexts.
- *Statically available key* W indicates keys available for typechecking.
- *Soft decryption limit* U specifies a subset of W safe to use currently.
- *Effects label* V summarizes the keys needed to run e.

soft decryption limit $\sim$ modal-logic world

effects label $\sim$ standard type-and-effects label

Soundness requires handling fact contexts explicitly.

Definition ($\mathrm{valid}_\Sigma\,\mathscr{F}$)

$\mathrm{valid}_\Sigma\,\mathscr{F}$ holds when

1. Σ is well formed: $\Sigma \vdash \diamond$.
2. Facts are true: $\mathscr{E}(a, e, n) : t$ **for** $b \in \mathscr{F}$ implies $a = b$ and $\Sigma; \cdot; b; \cdot; b; b \vdash e : t$.

Soundness requires handling fact contexts explicitly.

Definition ($\mathrm{valid}_\Sigma\, \mathscr{F}$)

$\mathrm{valid}_\Sigma\, \mathscr{F}$ holds when

1. Σ is well formed: $\Sigma \vdash \diamond$.
2. Facts are true: $\mathscr{E}(a, e, n) : t$ **for** $b \in \mathscr{F}$ implies $a = b$ and $\Sigma; \cdot; b; \cdot; b; b \vdash e : t$.

Lemma (New Fact Validity)

Assume $\mathrm{valid}_\Sigma\, \mathscr{F}_0$ *and* $\Sigma; \mathscr{F}_0; W; \Gamma; U; V \vdash e : t$. *Then* $\Sigma; \mathscr{F}_0; W \vdash \{|e, n|\} \rightarrow \{|e', n'|\}$ *learning* $\mathscr{F}$ *implies* $\mathrm{valid}_\Sigma\, \mathscr{F}$.

Soundness requires handling fact contexts explicitly.

Definition ($\mathrm{valid}_\Sigma\,\mathscr{F}$)

$\mathrm{valid}_\Sigma\,\mathscr{F}$ holds when

1. Σ is well formed: $\Sigma \vdash \diamond$.
2. Facts are true: $\mathscr{E}(a,e,n) : t$ **for** $b \in \mathscr{F}$ implies $a = b$ and $\Sigma; \cdot; b; \cdot; b; b \vdash e : t$.

Lemma (New Fact Validity)

Assume $\mathrm{valid}_\Sigma\,\mathscr{F}_0$ *and* $\Sigma; \mathscr{F}_0; W; \Gamma; U; V \vdash e : t$. *Then* $\Sigma; \mathscr{F}_0; W \vdash \{|e, n|\} \rightarrow \{|e', n'|\}$ *learning* $\mathscr{F}$ *implies* $\mathrm{valid}_\Sigma\,\mathscr{F}$.

Slogan

Preservation + Progress + New Fact Validity = Soundness

Soundness requires handling fact contexts explicitly.

Soundness results mechanized in Coq

Definition ($\mathrm{valid}_\Sigma\, \mathscr{F}$)

$\mathrm{valid}_\Sigma\, \mathscr{F}$ holds when

1. Σ is well formed: $\Sigma \vdash \diamond$.
2. Facts are true: $\mathscr{E}(a,e,n) : t$ **for** $b \in \mathscr{F}$ implies $a = b$ and $\Sigma; \cdot; b; \cdot; b; b \vdash e : t$.

Lemma (New Fact Validity)

Assume $\mathrm{valid}_\Sigma\, \mathscr{F}_0$ *and* $\Sigma; \mathscr{F}_0; W; \Gamma; U; V \vdash e : t$. *Then* $\Sigma; \mathscr{F}_0; W \vdash \{\!|e,n|\!\} \to \{\!|e',n'|\!\}$ *learning* $\mathscr{F}$ *implies* $\mathrm{valid}_\Sigma\, \mathscr{F}$.

Slogan

Preservation + Progress + New Fact Validity = Soundness

Noninterference: Secrets don't affect public outputs.

$b \vdash$ Aura Program

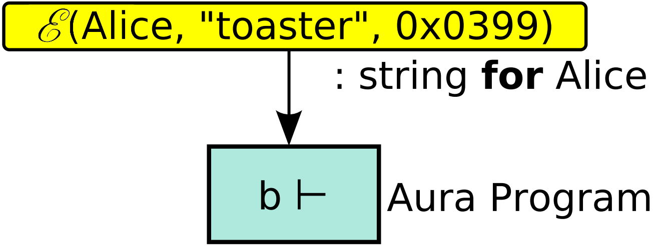
Noninterference: Secrets don't affect public outputs.

$\mathscr{E}$(Alice, "toaster", 0x0399)

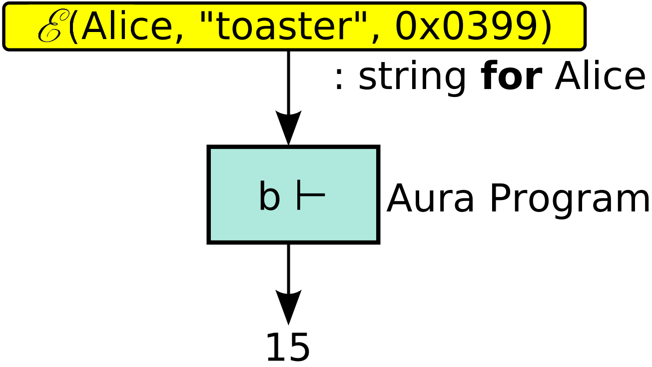
: string **for** Alice

b $\vdash$ Aura Program

Noninterference: Secrets don't affect public outputs.



Noninterference: Secrets don’t affect public outputs.



Noninterference: Secrets don't affect public outputs.

$\mathscr{E}$(Alice, "toaster", 0x0399)

: string **for** Alice

b ⊢ Aura Program

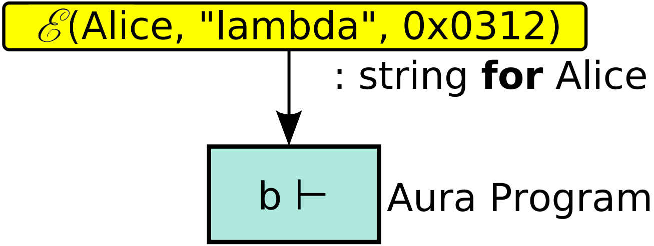
Noninterference: Secrets don't affect public outputs.

$\mathscr{E}$(Alice, "lambda", 0x0312)

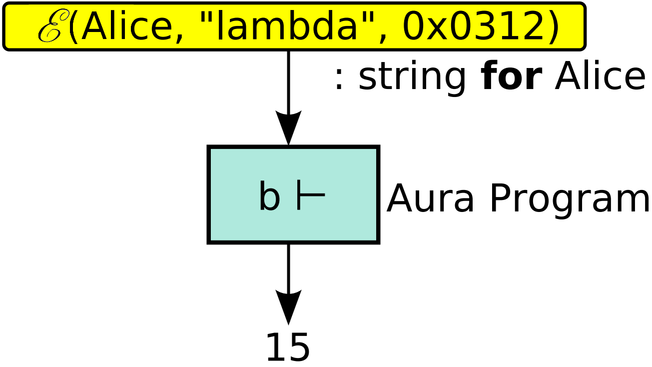
: string **for** Alice

b ⊢ Aura Program

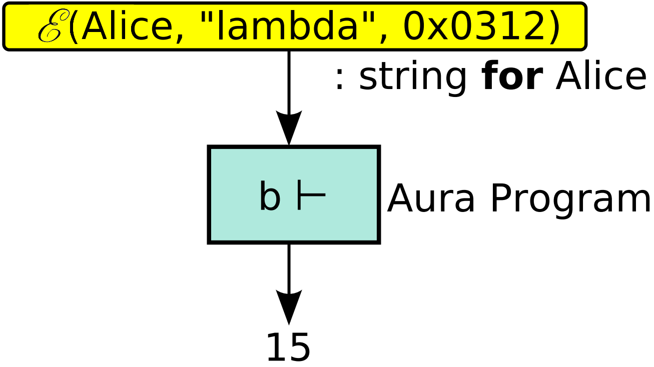
Noninterference: Secrets don't affect public outputs.



Noninterference: Secrets don't affect public outputs.



Noninterference: Secrets don't affect public outputs.



Noninterference [Denning+ '77],
Termination Insensitive Noninterference [Askarov+ '08]

Conclusion

Summary

- Type specification + cryptographic enforcement
  $\rightsquigarrow$ confidentiality

- Type-and-effects analysis + modal-type theory
  $\rightsquigarrow$ precise resource tracking

- AURA$_{\text{conf}}$ unifies mechanisms for confidentiality, audit and access control.

Acknowledgments

Thank you to all my collaborators on Aura project!

- Limin Jia
- Karl Mazurak
- Joseph Schorr
- Luke Zarko
- Steve Zdancewic
- Jianzhou Zhao

Acknowledgments

Thank you to all my collaborators on Aura project!

- Limin Jia
- Karl Mazurak
- Joseph Schorr
- Luke Zarko
- Steve Zdancewic
- Jianzhou Zhao

Questions?