On Separation Logic, Computational Independence, and Pseudorandomness

Ugo Dal Lago[1,2] Davide Davoli[2,3] Bruce Kapron[4]

[1]Università di Bologna [2]INRIA

[3]Université Côte d'Azur [4]University of Victoria

PLAS Workshop

October 14th, 2024 – Salt Lake City, USA

UNIVERSITÉ CÔTE D'AZUR | ÉCOLE UNIVERSITAIRE DE RECHERCHE SYSTÈMES NUMÉRIQUES POUR L'HUMAIN | FRANCE 2030 Initiative d'Excellence

Outline

- Introduction on separation logic.
- Barthe et al.'s separation logic for *probabilistic programs*.
- **Our contribution: a separation logic for *computational cryptography*.**

Separation Logic [Reynolds and O'Hearn, 2002]

Introduced to reason about *heap* manipulating programs.

Separation Logic [Reynolds and O'Hearn, 2002]

Introduced to reason about *heap* manipulating programs.

***Separating* conjunction:**

$$h \models \phi * \psi \qquad :\Leftrightarrow \qquad \exists h_1, h_2 \text{ s.t. } h_1 \models \phi, h_2 \models \psi, \text{ and } h_1 \sqcup h_2 \sqsubseteq h,$$

Separation Logic [Reynolds and O'Hearn, 2002]

Introduced to reason about *heap* manipulating programs.

***Separating* conjunction:**

$$h \models \phi * \psi \qquad :\Leftrightarrow \qquad \exists h_1, h_2 \text{ s.t. } h_1 \models \phi, h_2 \models \psi, \text{ and } h_1 \sqcup h_2 \sqsubseteq h,$$

h_1	h_2	$h_1 \sqcup h_2$		h
`x: 3`		`x: 3`		`x: 3`
	`z: 1`	`z: 1`	$\sqsubseteq$	`z: 1`
				`y: 4`

$h_1 \models \mathtt{x} = 3$ $\qquad h_2 \models \mathtt{z} = 1$ $\qquad h \models \mathtt{x} = 3 * \mathtt{z} = 1$

Probabilistic Separation Logic (**PSL**) (1/2)

The interpretation of $\sqcup$ and $\sqsubseteq$ determines the semantics of $*$.

	Heap model (O'Hearn et. al)	**Probabilistic model (Barthe et al.'s PSL)**
$\sqcup$	Store union	
$\sqsubseteq$	Sub-store	
$*$	Locality	

Probabilistic Separation Logic (**PSL**) (1/2)

The interpretation of $\sqcup$ and $\sqsubseteq$ determines the semantics of $*$.

	Heap model (O'Hearn et. al)	**Probabilistic model (Barthe et al.'s PSL)**
$\sqcup$	Store union	Tensor Product
$\sqsubseteq$	Sub-store	Marginal Distribution
$*$	Locality	

Probabilistic Separation Logic (**PSL**) (1/2)

The interpretation of $\sqcup$ and $\sqsubseteq$ determines the semantics of $*$.

	Heap model (O'Hearn et. al)	**Probabilistic model (Barthe et al.'s PSL)**
$\sqcup$	Store union	Tensor Product
$\sqsubseteq$	Sub-store	Marginal Distribution
$*$	Locality	Statistical Independence

Probabilistic Separation Logic (**PSL**) (1/2)

The interpretation of $\sqcup$ and $\sqsubseteq$ determines the semantics of $*$.

	Heap model (O'Hearn et. al)	**Probabilistic model (Barthe et al.'s PSL)**
$\sqcup$	Store union	Tensor Product
$\sqsubseteq$	Sub-store	Marginal Distribution
$*$	Locality	Statistical Independence

Example

$$[\![x \leftarrow \mathrm{unif}(n); y \leftarrow \mathrm{unif}(n)]\!] \models \underbrace{\mathbf{U}(x) * \mathbf{U}(y)}_{x \text{ and } y \text{ are uniform and independent}}$$

Probabilistic Separation Logic (**PSL**) (1/2)

The interpretation of $\sqcup$ and $\sqsubseteq$ determines the semantics of $*$.

	Heap model (O'Hearn et. al)	**Probabilistic model (Barthe et al.'s PSL)**
$\sqcup$	Store union	Tensor Product
$\sqsubseteq$	Sub-store	Marginal Distribution
$*$	Locality	Statistical Independence

Example

$$[\![x \leftarrow \mathsf{unif}(n); y \leftarrow \mathsf{unif}(n)]\!] \models \underbrace{\mathbf{U}(x) * \mathbf{U}(y)}_{x \text{ and } y \text{ are uniform and independent}}$$

$$[\![x \leftarrow \mathsf{unif}(n); y \leftarrow x]\!] \not\models \mathbf{U}(x) * \mathbf{U}(y)$$

Probabilistic Separation Logic (**PSL**) (1/2)

The interpretation of $\sqcup$ and $\sqsubseteq$ determines the semantics of $*$.

	Heap model (O'Hearn et. al)	**Probabilistic model (Barthe et al.'s PSL)**
$\sqcup$	Store union	Tensor Product
$\sqsubseteq$	Sub-store	Marginal Distribution
$*$	Locality	Statistical Independence

Example

$$[\![x \leftarrow \mathrm{unif}(n); y \leftarrow \mathrm{unif}(n)]\!] \models \underbrace{\mathbf{U}(x) * \mathbf{U}(y)}_{x \text{ and } y \text{ are uniform and independent}}$$

$$[\![x \leftarrow \mathrm{unif}(n); y \leftarrow x]\!] \not\models \mathbf{U}(x) * \mathbf{U}(y)$$

$$[\![x \leftarrow \mathrm{unif}(n); y \leftarrow x]\!] \models \mathbf{U}(x) \wedge \mathbf{U}(y)$$

Probabilistic Separation Logic (**PSL**) (2/2)

Probabilistic Separtation Logic can be used to support Hoare style reasoning on cryptographic primitives.

Probabilistic Separation Logic (**PSL**) (2/2)

Probabilistic Separtation Logic can be used to support Hoare style reasoning on cryptographic primitives.

Example (One Time Pad)

PSL can prove perfect secrecy.

$$\begin{aligned}\mathtt{OTP} := {} & \mathit{key} \leftarrow \mathsf{unif}(n);\\ & \mathit{cyph} \leftarrow \mathit{msg} \oplus \mathit{key}.\end{aligned}$$

Probabilistic Separation Logic (**PSL**) (2/2)

Probabilistic Separtation Logic can be used to support Hoare style reasoning on cryptographic primitives.

Example (One Time Pad)

PSL can prove perfect secrecy.

$$\begin{aligned} \mathtt{OTP} := {} & \mathit{key} \leftarrow \mathsf{unif}(n); \\ & \mathit{cyph} \leftarrow \mathit{msg} \oplus \mathit{key}. \end{aligned}$$

In **PSL**, the following judgment is derivable:

$$\vdash_{\mathbf{PSL}} \{\underbrace{\mathbf{D}(\mathit{msg})}_{\mathit{msg}\text{ is defined}}\}\ \mathtt{OTP}\ \{\mathbf{D}(\mathit{msg}) * \mathbf{U}(\mathit{cyph})\}$$

Probabilistic Separation Logic (**PSL**) (2/2)

Probabilistic Separtation Logic can be used to support Hoare style reasoning on cryptographic primitives.

Example (One Time Pad)

PSL can prove perfect secrecy.

$$\begin{aligned}\texttt{OTP} := \; & \mathit{key} \leftarrow \mathsf{unif}(n); \\ & \mathit{cyph} \leftarrow \mathit{msg} \oplus \mathit{key}.\end{aligned}$$

In **PSL**, the following judgment is derivable:

$$\vdash_{\textbf{PSL}} \{\underbrace{\mathbf{D}(\mathit{msg})}_{\mathit{msg}\text{ is defined}}\} \; \texttt{OTP} \; \{\underbrace{\mathbf{D}(\mathit{msg}) * \mathbf{U}(\mathit{cyph})}_{\text{perfect secrecy}}\}$$

Towards a Computational Version of **PSL**

Property		**Characterization**	**Logic**
Perfect Secrecy	$\Leftrightarrow$	Statistical Independence	**PSL**

$$\underbrace{\text{Perfect Secrecy} \Leftrightarrow \text{Statistical Independence}}_{\text{[Shannon, 1940]}}$$

Towards a Computational Version of **PSL**

Property		**Characterization**	**Logic**
Perfect Secrecy	$\Leftrightarrow$	Statistical Independence	**PSL**

$$\underbrace{\text{Perfect Secrecy} \quad \Leftrightarrow \quad \text{Statistical Independence}}_{\text{[Shannon, 1940]}}$$

$$\underbrace{\textit{Computational}\ \text{Secrecy} \quad \Leftrightarrow \quad \textit{Computational}\ \text{independence}}_{\text{[Fay, 2015] (polysize circuits)}} \qquad ?$$

?

Towards a Computational Version of **PSL**

Property		**Characterization**	**Logic**
$\underbrace{\text{Perfect Secrecy} \quad \Leftrightarrow \quad \text{Statistical Independence}}_{\text{[Shannon, 1940]}}$			**PSL**
$\underbrace{\textit{Computational}\text{ Secrecy} \quad \Leftrightarrow \quad \textit{Computational}\text{ independence}}_{\text{[Fay, 2015] (polysize circuits)}}$			?

this work (polytime programs)

Towards a Computational Version of **PSL**

Property		**Characterization**	**Logic**
Perfect Secrecy	$\Leftrightarrow$	Statistical Independence	**PSL**

[Shannon, 1940]

Computational Secrecy	$\Leftrightarrow$	*Computational* independence	**CSL**

[Fay, 2015] (polysize circuits)

this work (polytime programs)

Cryptographic Separation Logic (CSL)

The interpretation of $\sqcup$ and $\sqsubseteq$ determines the semantics of $*$.

	PSL	**CSL**

Cryptographic Separation Logic (CSL)

The interpretation of $\sqcup$ and $\sqsubseteq$ determines the semantics of $*$.

	PSL	**CSL**
$\sqcup$	Tensor Product	Tensor Product

Cryptographic Separation Logic (CSL)

The interpretation of $\sqcup$ and $\sqsubseteq$ determines the semantics of $*$.

	PSL	**CSL**
$\sqcup$	Tensor Product	Tensor Product
$d_1 \sqsubseteq d_2$	d_1 is a marginal of d_2	

Cryptographic Separation Logic (CSL)

The interpretation of $\sqcup$ and $\sqsubseteq$ determines the semantics of $*$.

	PSL	**CSL**
$\sqcup$	Tensor Product	Tensor Product
$d_1 \sqsubseteq d_2$	$d_1 = \lambda x. \sum_y d_2(x, y)$	

Cryptographic Separation Logic (CSL)

The interpretation of $\sqcup$ and $\sqsubseteq$ determines the semantics of $*$.

	PSL	**CSL**
$\sqcup$	Tensor Product	Tensor Product
$d_1 \sqsubseteq d_2$	$d_1 = \lambda x. \sum_y d_2(x, y)$	$d_1 \approx \lambda x. \sum_y d_2(x, y)$

Cryptographic Separation Logic (CSL)

The interpretation of $\sqcup$ and $\sqsubseteq$ determines the semantics of $*$.

	PSL	CSL
$\sqcup$	Tensor Product	Tensor Product
$d_1 \sqsubseteq d_2$	$d_1 = \lambda x. \sum_y d_2(x, y)$	$d_1 \approx \lambda x. \sum_y d_2(x, y)$

Every *polytime* distingusher has *negligible* advantage on d_1 and $\lambda x. \sum_y d_2(x, y)$.

Cryptographic Separation Logic (CSL)

The interpretation of $\sqcup$ and $\sqsubseteq$ determines the semantics of $*$.

	PSL	**CSL**
$\sqcup$	Tensor Product	Tensor Product
$d_1 \sqsubseteq d_2$	$d_1 = \lambda x. \sum_y d_2(x, y)$	$d_1 \approx \lambda x. \sum_y d_2(x, y)$

Every *polytime* distingusher has *negligible* advantage on d_1 and $\lambda x. \sum_y d_2(x, y)$.

Theorem (Main Result)

The semantics of the separating conjunction ($*$) in **CSL** *is equivalent to Fay's computational independence.*

Polytime Programs

Syntax:

$$\mathtt{P}, \mathtt{R} ::= \mathtt{skip} \;\Big|\; r \leftarrow e \;\Big|\; \mathtt{P}; \mathtt{P} \;\Big|\; \mathtt{if}\ r\ \mathtt{then}\ \mathtt{P}\ \mathtt{else}\ \mathtt{P}$$

$$e, g ::= \ldots$$

Polytime Programs

Syntax:

$$\mathtt{P}, \mathtt{R} ::= \ \mathtt{skip} \ \big| \ r \leftarrow e \ \big| \ \mathtt{P}; \mathtt{P} \ \big| \ \mathtt{if}\ r\ \mathtt{then}\ \mathtt{P}\ \mathtt{else}\ \mathtt{P}$$

$$e, g ::= \ \ldots$$

polynomial in the security parameter

Type system:

- *Assumption:* $\Delta \vdash e : p(n)$ when e has size $p(n)$ and is polytime-computable.

Polytime Programs

Syntax:

$$\mathtt{P}, \mathtt{R} ::= \mathtt{skip} \mid r \leftarrow e \mid \mathtt{P}; \mathtt{P} \mid \mathtt{if}\ r\ \mathtt{then}\ \mathtt{P}\ \mathtt{else}\ \mathtt{P}$$

$$e, g ::= \ldots$$

polynomial in the security parameter

Type system:

- *Assumption:* $\Delta \vdash e : p(n)$ when e has size $p(n)$ and is polytime-computable.
  Consequence: $\Delta \vdash \mathtt{P}$ when $\mathtt{P}$ is polytime in its input.

Polytime Programs

Syntax:

$$\mathtt{P},\mathtt{R} ::= \mathtt{skip} \mid r \leftarrow e \mid \mathtt{P};\mathtt{P} \mid \mathtt{if}\ r\ \mathtt{then}\ \mathtt{P}\ \mathtt{else}\ \mathtt{P}$$

$$e, g ::= \ \ldots$$

polynomial in the security parameter

Type system:

- *Assumption:* $\Delta \vdash e : p(n)$ when e has size $p(n)$ and is polytime-computable.
  Consequence: $\Delta \vdash \mathtt{P}$ when $\mathtt{P}$ is polytime in its input.
- We only want those inputs that are polysize in the security parameter, so:

$$[\![\Delta]\!] = \{\text{distributions of } \textit{polysize} \text{ stores}\}.$$

Polytime Programs

Syntax:

$$\mathtt{P}, \mathtt{R} ::= \ \mathtt{skip} \ \big| \ r \leftarrow e \ \big| \ \mathtt{P}; \mathtt{P} \ \big| \ \mathtt{if}\ r\ \mathtt{then}\ \mathtt{P}\ \mathtt{else}\ \mathtt{P}$$

$$e, g ::= \ \ldots$$

polynomial in the security parameter

Type system:

- *Assumption:* $\Delta \vdash e : p(n)$ when e has size $p(n)$ and is polytime-computable.
  Consequence: $\Delta \vdash \mathtt{P}$ when $\mathtt{P}$ is polytime in its input.
- We only want those inputs that are polysize in the security parameter, so:

$$[\![\Delta]\!] = \{\text{distributions of } \textit{polysize} \text{ stores}\}.$$

Semantics:

$$[\![\Delta \vdash \mathtt{P}]\!] : [\![\Delta]\!] \to [\![\Delta]\!]$$

Our semantics is polytime in the security parameter *by definition*.

CSL's Syntax and Semantics

$$
\begin{aligned}
A ::= {} & \mathbf{EQ}(e, g) && [\![e]\!] \text{ and } [\![g]\!] \text{ are the same distribution} \\
\mid {} & \mathbf{CI}(e, g) && [\![e]\!] \text{ and } [\![g]\!] \text{ are indistinguishable } ([\![e]\!] \approx [\![g]\!])
\end{aligned}
$$

CSL's Syntax and Semantics

$$
\begin{array}{ll}
A ::= \mathbf{EQ}(e, g) & [\![e]\!] \text{ and } [\![g]\!] \text{ are the same distribution} \\
\quad\mid \mathbf{CI}(e, g) & [\![e]\!] \text{ and } [\![g]\!] \text{ are indistinguishable } ([\![e]\!] \approx [\![g]\!])
\end{array}
$$

$$\phi ::= (A)^{\Delta} \mid (\phi \wedge \psi)^{\Delta} \mid (\phi * \psi)^{\Delta}$$

CSL's Syntax and Semantics

$$A ::= \mathbf{EQ}(e, g) \qquad [\![e]\!] \text{ and } [\![g]\!] \text{ are the same distribution}$$

$$\mid \mathbf{CI}(e, g) \qquad [\![e]\!] \text{ and } [\![g]\!] \text{ are indistinguishable } ([\![e]\!] \approx [\![g]\!])$$

$$\phi ::= (A)^{\Delta} \mid (\phi \land \psi)^{\Delta} \mid (\phi * \psi)^{\Delta}$$

Environments constraint the interpretation of formulas to some specific distributions...

$$d \models (\phi)^{\Delta} \text{ whenever } d \in [\![\Delta]\!] \text{ and } d \models \phi$$

CSL's Syntax and Semantics

$$
\begin{aligned}
A ::= {} & \mathbf{EQ}(e, g) && [\![e]\!] \text{ and } [\![g]\!] \text{ are the same distribution} \\
\mid {} & \mathbf{CI}(e, g) && [\![e]\!] \text{ and } [\![g]\!] \text{ are indistinguishable } ([\![e]\!] \approx [\![g]\!])
\end{aligned}
$$

$$\phi ::= (A)^{\Delta} \mid (\phi \wedge \psi)^{\Delta} \mid (\phi * \psi)^{\Delta}$$

Environments constraint the interpretation of formulas to some specific distributions...

$$d \models (\phi)^{\Delta} \text{ whenever } d \in [\![\Delta]\!] \text{ and } d \models \phi$$

...that define all the variables of the formula, thanks to the following conditions:

- For $(A(e, g))^{\Delta}$, we impose $\Delta \vdash e : \tau$, $\Delta \vdash g : \tau$.

CSL's Syntax and Semantics

$$A ::= \mathbf{EQ}(e, g) \qquad [\![e]\!] \text{ and } [\![g]\!] \text{ are the same distribution}$$
$$\mid \mathbf{CI}(e, g) \qquad [\![e]\!] \text{ and } [\![g]\!] \text{ are indistinguishable } ([\![e]\!] \approx [\![g]\!])$$

$$\phi ::= (A)^{\Delta} \mid (\phi \wedge \psi)^{\Delta} \mid (\phi * \psi)^{\Delta}$$

Environments constraint the interpretation of formulas to some specific distributions...

$$d \models (\phi)^{\Delta} \text{ whenever } d \in [\![\Delta]\!] \text{ and } d \models \phi$$

...that define all the variables of the formula, thanks to the following conditions:

- For $(A(e, g))^{\Delta}$, we impose $\Delta \vdash e : \tau$, $\Delta \vdash g : \tau$.
- For $((\phi)^{\Gamma} \wedge (\psi)^{\Theta})^{\Delta}$, Γ and Θ must be smaller than Δ.

CSL's Syntax and Semantics

$$
\begin{aligned}
A ::= {} & \mathbf{EQ}(e, g) && [\![e]\!] \text{ and } [\![g]\!] \text{ are the same distribution} \\
\mid {} & \mathbf{CI}(e, g) && [\![e]\!] \text{ and } [\![g]\!] \text{ are indistinguishable } ([\![e]\!] \approx [\![g]\!])
\end{aligned}
$$

$$\phi ::= (A)^{\Delta} \mid (\phi \wedge \psi)^{\Delta} \mid (\phi * \psi)^{\Delta}$$

Environments constraint the interpretation of formulas to some specific distributions...

$$d \models (\phi)^{\Delta} \text{ whenever } d \in [\![\Delta]\!] \text{ and } d \models \phi$$

...that define all the variables of the formula, thanks to the following conditions:

- For $(A(e, g))^{\Delta}$, we impose $\Delta \vdash e : \tau$, $\Delta \vdash g : \tau$.
- For $((\phi)^{\Gamma} \wedge (\psi)^{\Theta})^{\Delta}$, Γ and Θ must be smaller than Δ.
- For $((\phi)^{\Gamma} * (\psi)^{\Theta})^{\Delta}$, Γ and Θ must also be disjoint.

Typed Separating Conjunctions

The standard interpretation of $*$ is ambiguous:

$$h \models \phi * \psi \qquad :\Leftrightarrow \qquad \exists \underbrace{h_1, h_2}_{\text{what is their domain?}} \text{ s.t. } \ldots$$

Typed Separating Conjunctions

The standard interpretation of $*$ is ambiguous:

$$d \models \phi * \psi \qquad :\Leftrightarrow \qquad \exists \underbrace{d_1, d_2}_{\text{what is their domain?}} \text{s.t.} \ldots$$

- In **PSL**, we do not know which variables are independent.

Typed Separating Conjunctions

The standard interpretation of $*$ is ambiguous:

$$d \models \phi * \psi \qquad :\Leftrightarrow \qquad \exists \underbrace{d_1, d_2}_{\text{what is their domain?}} \text{ s.t. } \ldots$$

- In **PSL**, we do not know which variables are independent.
- In **CSL**, independent variables are *explicit in formulas*.

$$d \models ((\phi)^{\Gamma} * (\psi)^{\Theta})^{\Delta} \Rightarrow \text{ variables of } \Gamma \text{ and } \Theta \text{ are independent in } d.$$

Deduction rules

Judgments:

$$\{(\phi)^{\Delta}\}\ \Delta \vdash \mathtt{P}\ \{(\psi)^{\Delta}\}$$

Deduction rules

Judgments:

$$\{(\phi)^{\Delta}\}\ \Delta \vdash \mathtt{P}\ \{(\psi)^{\Delta}\}$$

For every $d \in [\![\Delta]\!]$, if $d \models \phi$, then $[\![\Delta \vdash \mathtt{P}]\!](d) \models \psi$.

Deduction rules

Judgments:

$$\{(\phi)^{\Delta}\}\ \Delta \vdash \mathtt{P}\ \{(\psi)^{\Delta}\}$$

For every $d \in [\![\Delta]\!]$, if $d \models \phi$, then $[\![\Delta \vdash \mathtt{P}]\!](d) \models \psi$.

Rules:

$$\frac{r \notin \mathsf{FV}(e)}{\vdash \{(\top)^{\Delta}\}\ \Delta \vdash r \leftarrow e\ \{(\mathbf{EQ}(r, e))^{\Delta}\}}\ \mathsf{Asgn}$$

Deduction rules

Judgments:

$$\{(\phi)^{\Delta}\}\ \Delta \vdash \mathtt{P}\ \{(\psi)^{\Delta}\}$$

For every $d \in [\![\Delta]\!]$, if $d \models \phi$, then $[\![\Delta \vdash \mathtt{P}]\!](d) \models \psi$.

Rules:

$$\frac{r \notin \mathsf{FV}(e)}{\vdash \{(\top)^{\Delta}\}\ \Delta \vdash r \leftarrow e\ \{(\mathbf{EQ}(r, e))^{\Delta}\}}\ \mathsf{Asgn}$$

In **PSL**: $\dfrac{r \notin \mathsf{FV}(e)}{\vdash \{\top\}\ r \leftarrow e\ \{\mathbf{EQ}(r, e)\}}\ \mathsf{Asgn}$

Deduction rules

Judgments:

$$\{(\phi)^{\Delta}\}\ \Delta \vdash \mathtt{P}\ \{(\psi)^{\Delta}\}$$

For every $d \in [\![\Delta]\!]$, if $d \models \phi$, then $[\![\Delta \vdash \mathtt{P}]\!](d) \models \psi$.

Rules:

$$\frac{r \notin \mathsf{FV}(e)}{\vdash \{(\top)^{\Delta}\}\ \Delta \vdash r \leftarrow e\ \{(\mathbf{EQ}(r, e))^{\Delta}\}}\ \text{Asgn}$$

In **PSL**: $$\frac{r \notin \mathsf{FV}(e)}{\vdash \{\top\}\ r \leftarrow e\ \{\mathbf{EQ}(r, e)\}}\ \text{Asgn}$$

$$\frac{\vdash \{(\phi)^{\Gamma}\}\ \Gamma \vdash \mathtt{P}\ \{(\psi)^{\Gamma}\}}{\vdash \{((\phi)^{\Gamma} * (\xi)^{\Theta})^{\Delta}\}\ \Delta \vdash \mathtt{P}\ \{((\psi)^{\Gamma} * (\xi)^{\Theta})^{\Delta}\}}\ \text{Frame}$$

Difficulty with loops

$$\frac{\forall i.\vdash \{\phi(i)\}\ \texttt{P}\ \{\phi(i+1)\}}{\vdash \{\phi(0)\}\ \texttt{for}\ i=0\ \texttt{to}\ n\ \texttt{do P}\ \{\phi(n)\}}$$

Soundness in classical Hoare logic:

Difficulty with loops

$$\frac{\forall i.\vdash \{\phi(i)\}\ \texttt{P}\ \{\phi(i+1)\}}{\vdash \{\phi(0)\}\ \texttt{for}\ i=0\ \texttt{to}\ n\ \texttt{do}\ \texttt{P}\ \{\phi(n)\}}$$

Soundness in classical Hoare logic:

State:	$\bullet$
Property:	$\phi(0)$

Difficulty with loops

$$\frac{\forall i.\ \vdash \{\phi(i)\}\ \texttt{P}\ \{\phi(i+1)\}}{\vdash \{\phi(0)\}\ \texttt{for}\ i = 0\ \texttt{to}\ n\ \texttt{do}\ \texttt{P}\ \{\phi(n)\}}$$

Soundness in classical Hoare logic:

State: $\bullet \xrightarrow{[\![\mathrm{P}]\!]} \bullet$

Property: $\phi(0) \underbrace{\Rightarrow}_{\text{assumption}} \phi(1)$

Difficulty with loops

$$\frac{\forall i.\ \vdash \{\phi(i)\}\ \mathtt{P}\ \{\phi(i+1)\}}{\vdash \{\phi(0)\}\ \mathtt{for}\ i = 0\ \mathtt{to}\ n\ \mathtt{do}\ \mathtt{P}\ \{\phi(n)\}}$$

Soundness in classical Hoare logic:

State: $\quad \bullet \xrightarrow{[\![\mathrm{P}]\!]} \bullet \xrightarrow{[\![\mathrm{P}]\!]} \bullet \xrightarrow{[\![\mathrm{P}]\!]} \dots \xrightarrow{[\![\mathrm{P}]\!]} \bullet$

Property: $\quad \phi(0) \underbrace{\Rightarrow}_{\text{assumption}} \phi(1) \underbrace{\Rightarrow}_{\text{assumption}} \phi(2) \underbrace{\Rightarrow}_{\text{assumption}} \dots \underbrace{\Rightarrow}_{\text{assumption}} \phi(n)$

Difficulty with loops

n is the security parameter

$$\frac{\forall i.\vdash \{\phi(i)\}\ \mathtt{P}\ \{\phi(i+1)\}}{\vdash \{\phi(0)\}\ \mathtt{for}\ i=0\ \mathtt{to}\ n\ \mathtt{do}\ \mathtt{P}\ \{\phi(n)\}}$$

In CSL:

$\phi(i)$:= a negligible function bounds the probability of distinguishing d_i from *unif*

Difficulty with loops

n is the security parameter

$$\frac{\forall i. \vdash \{\phi(i)\}\ \mathtt{P}\ \{\phi(i+1)\}}{\vdash \{\phi(0)\}\ \mathtt{for}\ i = 0\ \mathtt{to}\ n\ \mathtt{do}\ \mathtt{P}\ \{\phi(n)\}}$$

In CSL:

$\phi(i)$:= a negligible function bounds the probability of distinguishing d_i from *unif*

Pre-condition:

$unif \xleftarrow{}\nu_0\xrightarrow{}\ \ d_0$

Difficulty with loops

n is the security parameter

$$\frac{\forall i. \vdash \{\phi(i)\}\ \mathtt{P}\ \{\phi(i+1)\}}{\vdash \{\phi(0)\}\ \mathtt{for}\ i = 0\ \mathtt{to}\ n\ \mathtt{do}\ \mathtt{P}\ \{\phi(n)\}}$$

In CSL:

$\phi(i) :=$ a negligible function bounds the probability of distinguishing d_i from *unif*

Pre-condition:

$$\left.\begin{array}{l} \mathit{unif} \leftarrow \nu_0 \rightarrow \quad d_0 \\ \mathit{unif} \longleftarrow \nu_1 \longrightarrow d_1 = [\![\mathrm{P}]\!](d_0) \end{array}\right\} \text{assumption}$$

Difficulty with loops

n is the security parameter

$$\frac{\forall i.\vdash \{\phi(i)\}\ \mathtt{P}\ \{\phi(i+1)\}}{\vdash \{\phi(0)\}\ \mathtt{for}\ i = 0\ \mathtt{to}\ n\ \mathtt{do}\ \mathtt{P}\ \{\phi(n)\}}$$

In CSL:

$\phi(i) :=$ a negligible function bounds the probability of distinguishing d_i from *unif*

Pre-condition:

$$\left.\begin{array}{l} \mathit{unif} \leftarrow \nu_0 \rightarrow\ d_0 \\ \mathit{unif} \longleftarrow \nu_1 \longrightarrow d_1 = [\![\mathrm{P}]\!](d_0) \end{array}\right\}\ \text{assumption}$$

$$\begin{array}{l} \vdots \qquad\qquad \ddots \\ \mathit{unif} \longleftarrow \nu_k \longrightarrow d_k = [\![\mathrm{P}]\!](d_{k-1}) \\ \vdots \qquad\qquad\qquad \ddots \end{array}$$

Difficulty with loops

n is the security parameter

$$\frac{\forall i.\vdash \{\phi(i)\}\ \texttt{P}\ \{\phi(i+1)\}}{\vdash \{\phi(0)\}\ \texttt{for}\ i = 0\ \texttt{to}\ n\ \texttt{do P}\ \{\phi(n)\}}$$

In CSL:

$\phi(i) :=$ a negligible function bounds the probability of distinguishing d_i from *unif*

Pre-condition:

$$\left.\begin{array}{l} unif \leftarrow \nu_0 \rightarrow \quad d_0 \\ unif \longleftarrow \nu_1 \longrightarrow d_1 = [\![\mathrm{P}]\!](d_0) \end{array}\right\} \text{assumption}$$

$$\begin{array}{l} \vdots \qquad\qquad \ddots \\ unif \longleftarrow \nu_k \longrightarrow d_k = [\![\mathrm{P}]\!](d_{k-1}) \\ \vdots \qquad\qquad\qquad \ddots \end{array}$$

$$\nu_i = \frac{n^i}{2^n},$$

Difficulty with loops

n is the security parameter

$$\frac{\forall i.\vdash \{\phi(i)\}\ \mathtt{P}\ \{\phi(i+1)\}}{\vdash \{\phi(0)\}\ \mathtt{for}\ i=0\ \mathtt{to}\ n\ \mathtt{do}\ \mathtt{P}\ \{\phi(n)\}}$$

In CSL:

$\phi(i) :=$ a negligible function bounds the probability of distinguishing d_i from *unif*

Pre-condition:

$$\left.\begin{array}{l} unif \xleftarrow{} \nu_0 \xrightarrow{} d_0 \\ unif \longleftarrow \nu_1 \longrightarrow d_1 = [\![\mathrm{P}]\!](d_0) \end{array}\right\}\ \text{assumption}$$

$$\nu_i = \frac{n^i}{2^n},$$

$$\vdots \qquad \ddots$$

$$unif \longleftarrow \nu_k \longrightarrow d_k = [\![\mathrm{P}]\!](d_{k-1})$$

$$\vdots \qquad \ddots$$

necessary for negligibility: $\lim_{n\to\infty} \nu(n) = 0$

Difficulty with loops

n is the security parameter

$$\frac{\forall i.\vdash \{\phi(i)\}\ \texttt{P}\ \{\phi(i+1)\}}{\vdash \{\phi(0)\}\ \texttt{for}\ i=0\ \texttt{to}\ n\ \texttt{do}\ \texttt{P}\ \{\phi(n)\}}$$

In CSL:

$\phi(i) :=$ a negligible function bounds the probability of distinguishing d_i from *unif*

Pre-condition:

$$\left.\begin{array}{l} unif \xleftarrow{}\ \nu_0 \xrightarrow{}\ d_0 \\ unif \longleftarrow \nu_1 \longrightarrow d_1 = [\![\mathrm{P}]\!](d_0) \end{array}\right\}\ \text{assumption}$$

$$\begin{array}{l} \vdots \qquad \ddots \\ unif \longleftarrow \nu_k \longrightarrow d_k = [\![\mathrm{P}]\!](d_{k-1}) \\ \vdots \qquad \ddots \end{array}$$

$$\nu_i = \frac{n^i}{2^n}, \quad \nu_n = \frac{n^n}{2^n}$$

necessary for negligibility: $\lim_{n\to\infty} \nu(n) = 0$

Pseudo One Time Pad

The Pseudo One Time Pad is the *computationally secret* variant of the One Time Pad.

Pseudo One Time Pad

The Pseudo One Time Pad is the *computationally secret* variant of the One Time Pad.

$$\begin{aligned}\texttt{OTP} := \; & \mathit{key} \leftarrow \mathsf{unif}(n); \\ & \mathit{cyph} \leftarrow \mathit{msg} \oplus \mathit{key}.\end{aligned}$$

Pseudo One Time Pad

The Pseudo One Time Pad is the *computationally secret* variant of the One Time Pad.

$$\begin{aligned}\texttt{POTP} := {} & key \leftarrow \mathrm{unif}(n);\\ & r \leftarrow g(key);\\ & cyph \leftarrow msg \oplus r.\end{aligned}$$

g is a *pseudorandom* generator.

Pseudo One Time Pad

The Pseudo One Time Pad is the *computationally secret* variant of the One Time Pad.

$$\begin{aligned} \mathtt{POTP} := {} & key \leftarrow \mathrm{unif}(n); \\ & r \leftarrow g(key); \\ & cyph \leftarrow msg \oplus r. \end{aligned}$$

g is a *pseudorandom* generator.

$$\vdash_{\mathbf{CSL}} \{(\top)^{\Delta}\}\ \Delta \vdash \mathtt{POTP}\ \{\underbrace{((\top)^{\{msg:\ldots\}} * (\mathbf{CU}(cyph))^{\{cyph:\ldots\}})^{\Delta}}_{\text{computational secrecy}}\}$$

Conclusion and Future Work

Conclusion

- CSL is a separation logic for *computational independence*.

Conclusion and Future Work

Conclusion

- CSL is a separation logic for *computational independence*.
- The inference rules of **CSL** are similar to those of **PSL**.

Conclusion and Future Work

Conclusion

- CSL is a separation logic for *computational independence*.
- The inference rules of **CSL** are similar to those of **PSL**.
- CSL can be used to prove *computational secrecy*, thanks to our Fay-style characterization.

Conclusion and Future Work

Conclusion

- CSL is a separation logic for *computational independence*.
- The inference rules of **CSL** are similar to those of **PSL**.
- CSL can be used to prove *computational secrecy*, thanks to our Fay-style characterization.

Future work

- Extend the language supported by **CSL** with for-loops.