FSLH: Flexible Mechanized Speculative Load Hardening

MAX PLANCK INSTITUTE
FOR SECURITY AND PRIVACY

Jonathan Baumann[1,2], Roberto Blanco[1,3], Léon Ducruet[1,4], Sebastian Harwig[1,5], Cătălin Hrițcu[1]

[1]MPI-SP, Germany [2]ENS Paris-Saclay, France [3]TU/e, Netherlands [4]ENS Lyon, France [5]Ruhr University Bochum, Germany

Why FSLH?

- Spectre Attacks remain a threat

Why FSLH?

- Spectre Attacks remain a threat
- Existing mitigations have complementary strengths and weaknesses:

Why FSLH?

- Spectre Attacks remain a threat
- Existing mitigations have complementary strengths and weaknesses:

Selective SLH (Shivakumar et al. 2023)

- sparse protection, low overhead

Why FSLH?

- Spectre Attacks remain a threat
- Existing mitigations have complementary strengths and weaknesses:

Selective SLH (Shivakumar et al. 2023)

- sparse protection, low overhead
- protects only cryptographic code

Why FSLH?

- Spectre Attacks remain a threat
- Existing mitigations have complementary strengths and weaknesses:

Selective SLH (Shivakumar et al. 2023)

- sparse protection, low overhead
- protects only cryptographic code

Ultimate SLH (Zhang et al. 2023)

- exhaustive, high overhead

Why FSLH?

- Spectre Attacks remain a threat
- Existing mitigations have complementary strengths and weaknesses:

Selective SLH (Shivakumar et al. 2023)

- sparse protection, low overhead
- protects only cryptographic code

Ultimate SLH (Zhang et al. 2023)

- exhaustive, high overhead
- protects all programs

Why FSLH?

- Spectre Attacks remain a threat
- Existing mitigations have complementary strengths and weaknesses:

Selective SLH (Shivakumar et al. 2023)

- sparse protection, low overhead
- protects only cryptographic code

Ultimate SLH (Zhang et al. 2023)

- exhaustive, high overhead
- protects all programs

Flexible SLH (published at CSF'25)

sparse protections for all programs

Why FSLH?

- Spectre Attacks remain a threat
- Existing mitigations have complementary strengths and weaknesses:

Selective SLH (Shivakumar et al. 2023)

- sparse protection, low overhead
- protects only cryptographic code

Ultimate SLH (Zhang et al. 2023)

- exhaustive, high overhead
- protects all programs

Flexible SLH (published at CSF'25)

sparse protections for all programs

- Existing mitigations rely on manual security proofs

Why FSLH?

- Spectre Attacks remain a threat
- Existing mitigations have complementary strengths and weaknesses:

Selective SLH (Shivakumar et al. 2023)

- sparse protection, low overhead
- protects only cryptographic code

Ultimate SLH (Zhang et al. 2023)

- exhaustive, high overhead
- protects all programs

Flexible SLH (published at CSF'25)

sparse protections for all programs

- Existing mitigations rely on manual security proofs
  - First machine-checked proofs for Selective, Ultimate, *and* Flexible SLH

Why FSLH?

- Spectre Attacks remain a threat
- Existing mitigations have complementary strengths and weaknesses:

Selective SLH (Shivakumar et al. 2023)

- sparse protection, low overhead
- protects only cryptographic code

Ultimate SLH (Zhang et al. 2023)

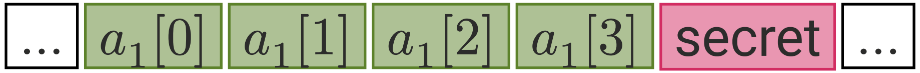
- exhaustive, high overhead
- protects all programs

Flexible SLH (published at CSF'25)

sparse protections for all programs

- Existing mitigations rely on manual security proofs
  - First machine-checked proofs for Selective, Ultimate, *and* Flexible SLH
    Rocq development: ~ 4300 lines

Spectre (v1)



…	$a_1[0]$	$a_1[1]$	$a_1[2]$	$a_1[3]$	secret	…

```
if i < $size(\mathtt{a}_1)$ then
    j ← $\mathtt{a}_1[\mathtt{i}]$;
    x ← $\mathtt{a}_2[\mathtt{j}]$

else
    …
```

Spectre (v1)

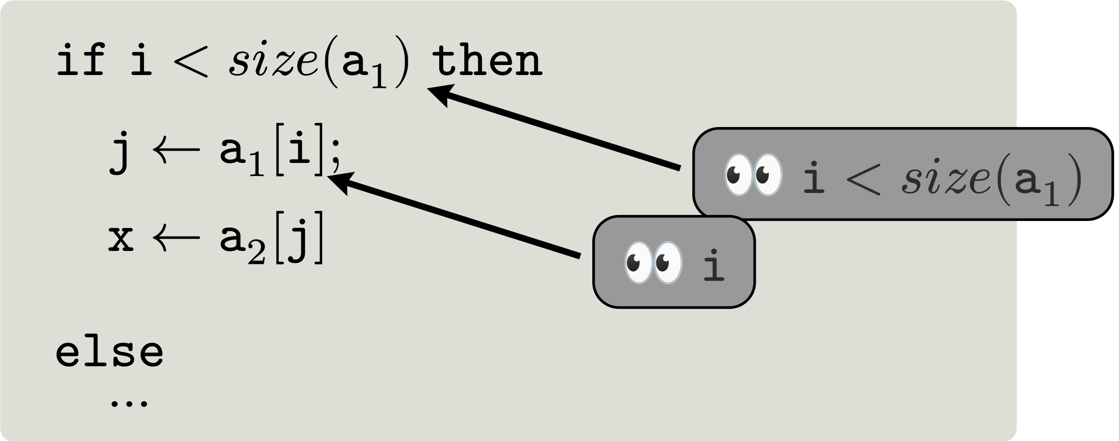
…	$a_1[0]$	$a_1[1]$	$a_1[2]$	$a_1[3]$	secret	…

```
if $\mathtt{i} < size(\mathtt{a}_1)$ then
    $\mathtt{j} \leftarrow \mathtt{a}_1[\mathtt{i}]$;
    $\mathtt{x} \leftarrow \mathtt{a}_2[\mathtt{j}]$

else
    …
```

👀 $\mathtt{i} < size(\mathtt{a}_1)$

Spectre (v1)

...	$a_1[0]$	$a_1[1]$	$a_1[2]$	$a_1[3]$	secret	...



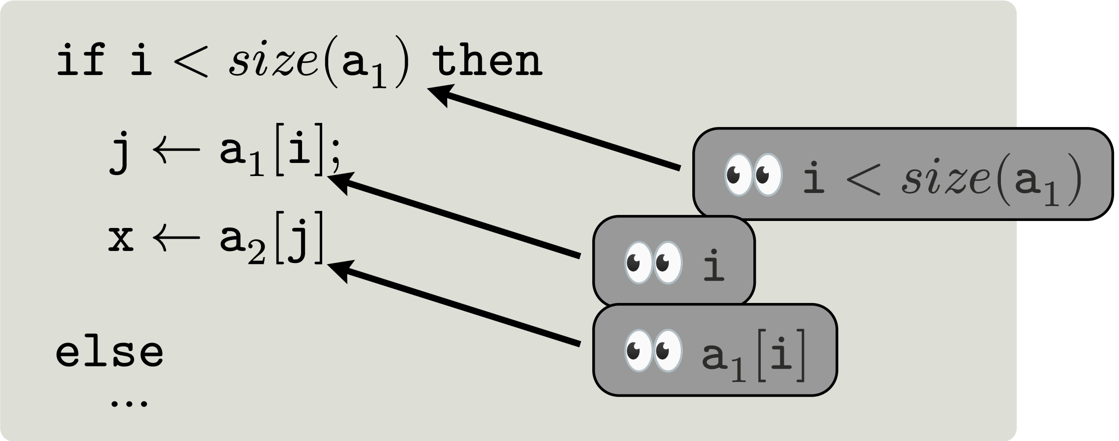
```
if i < $size$(a$_1$) then
   j ← a$_1$[i];
   x ← a$_2$[j]

else
   ...
```

👀 i < $size$(a$_1$)

👀 i

Spectre (v1)

…	$a_1[0]$	$a_1[1]$	$a_1[2]$	$a_1[3]$	🟥 secret	…



```
if i < size(a1) then
   j ← a1[i];
   x ← a2[j]

else
   …
```

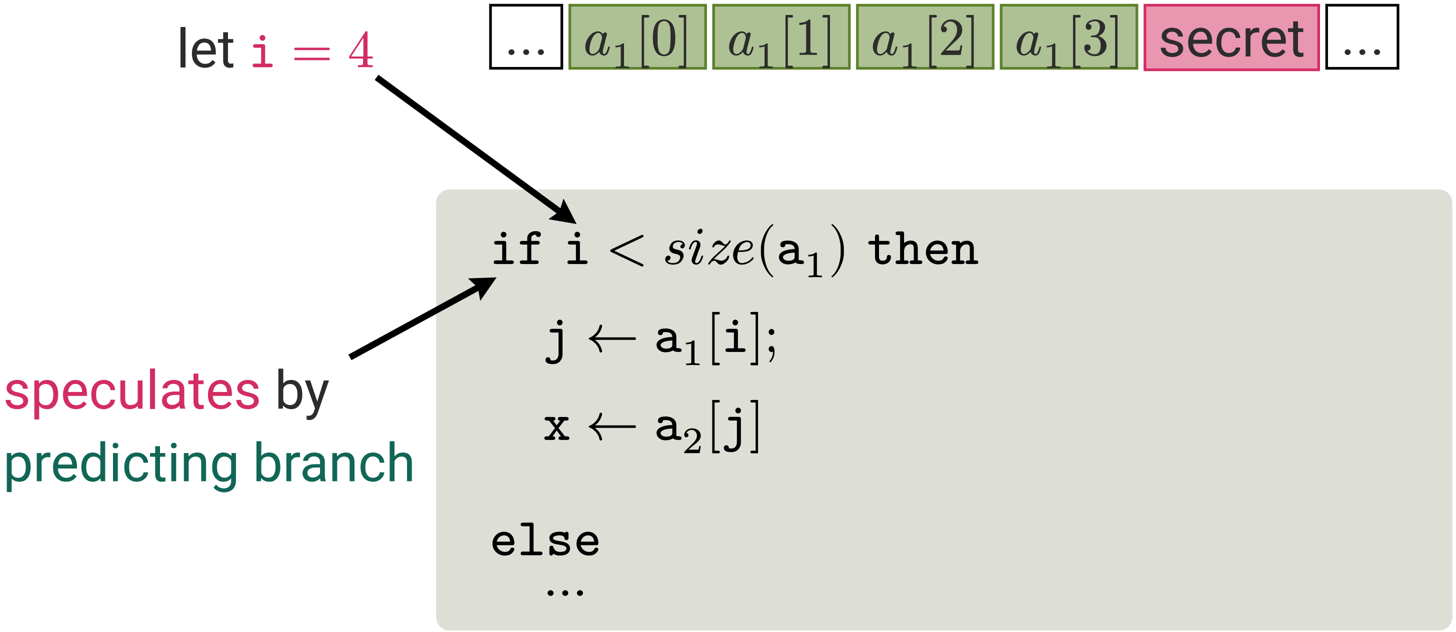
Spectre (v1)

let $\texttt{i} = 4$

…	$a_1[0]$	$a_1[1]$	$a_1[2]$	$a_1[3]$	secret	…

```
if $\texttt{i} < size(\texttt{a}_1)$ then
    $\texttt{j} \leftarrow \texttt{a}_1[\texttt{i}]$;
    $\texttt{x} \leftarrow \texttt{a}_2[\texttt{j}]$

else
    …
```

Spectre (v1)



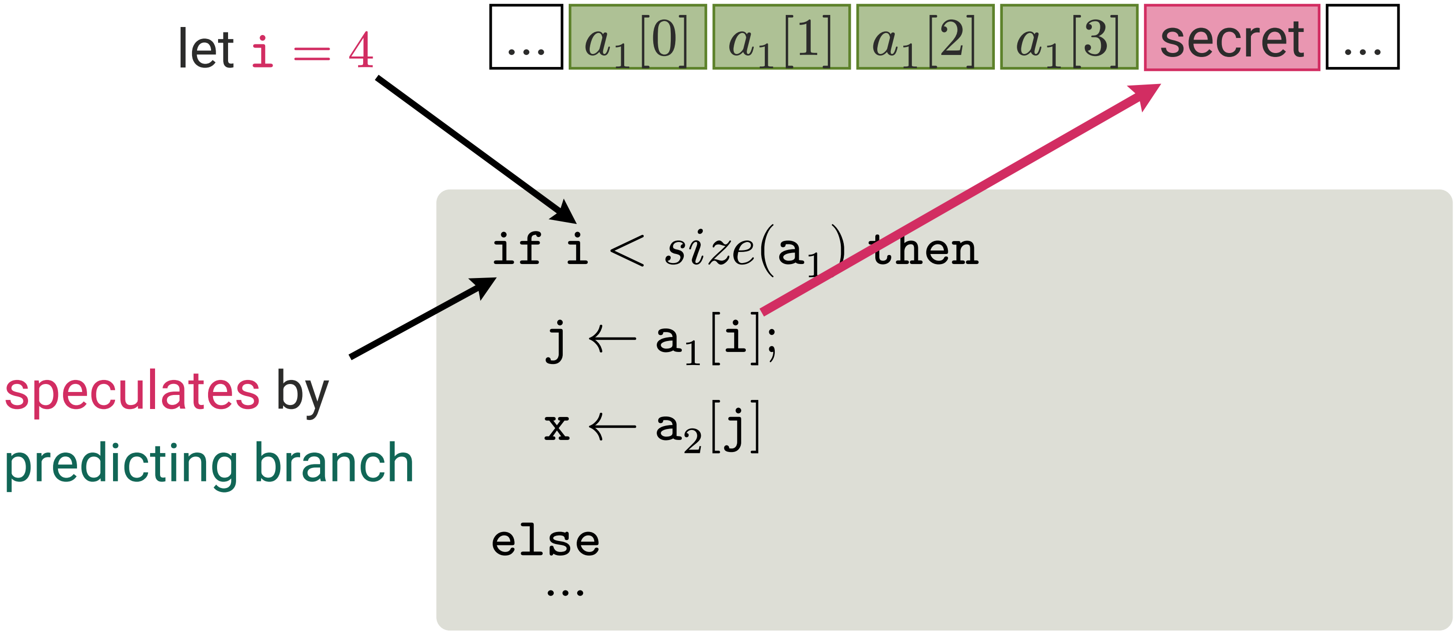
...	$a_1[0]$	$a_1[1]$	$a_1[2]$	$a_1[3]$	secret	...

let $\mathtt{i} = 4$

```
if i < $size(\mathtt{a}_1)$ then
   j ← $\mathtt{a}_1$[i];
   x ← $\mathtt{a}_2$[j]

else
   ...
```

speculates by predicting branch

Spectre (v1)



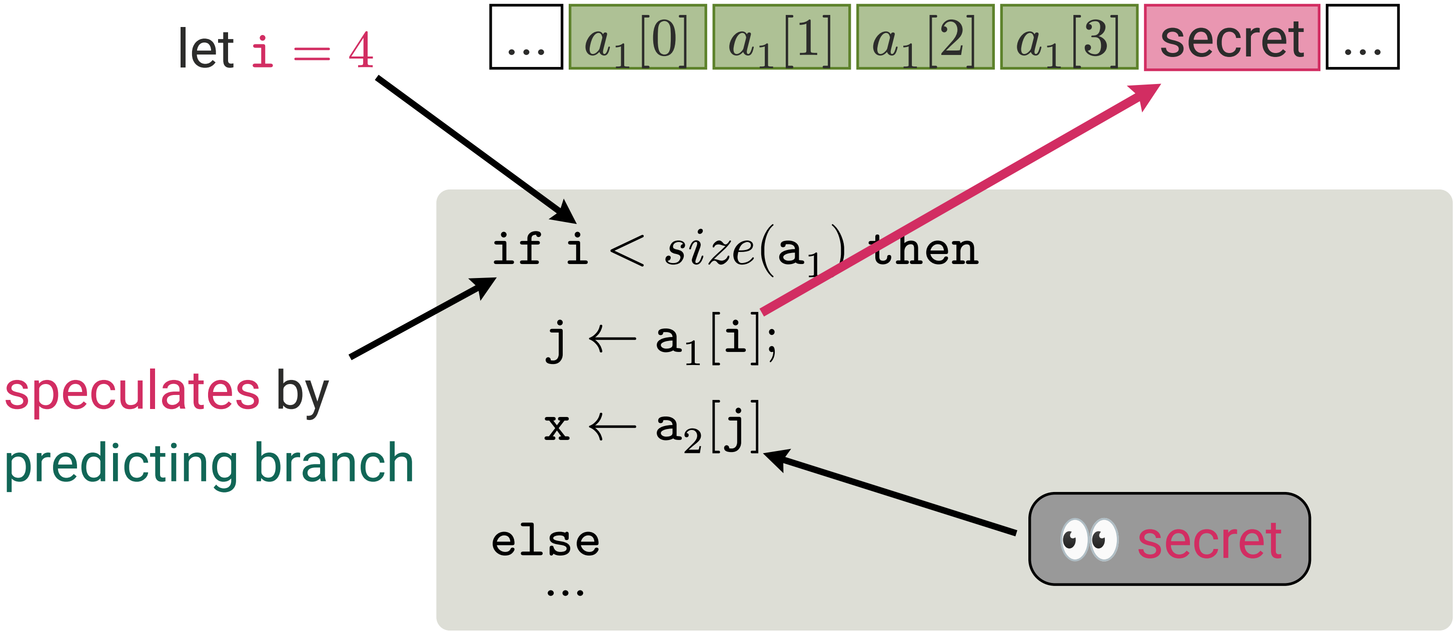
let $\mathtt{i} = 4$

…	$a_1[0]$	$a_1[1]$	$a_1[2]$	$a_1[3]$	secret	…

```
if $\mathtt{i} < size(\mathtt{a}_1)$ then
   $\mathtt{j} \leftarrow \mathtt{a}_1[\mathtt{i}]$;
   $\mathtt{x} \leftarrow \mathtt{a}_2[\mathtt{j}]$

else
   …
```

speculates by
predicting branch

Spectre (v1)



let $\mathtt{i} = 4$

…	$a_1[0]$	$a_1[1]$	$a_1[2]$	$a_1[3]$	secret	…

```
if i < $size(a_1)$ then
    j ← $a_1$[i];
    x ← $a_2$[j]

else
    …
```

speculates by predicting branch

👀 secret

Previous Work: Selective SLH (Shivakumar et al. 2023)

```
if i $< size(\mathtt{a}_1)$ then
    j $\leftarrow \mathtt{a}_1$ [i ];
    x $\leftarrow \mathtt{a}_2$ [j ]
else
```

Previous Work: Selective SLH (Shivakumar et al. 2023)

```
if $\texttt{i}_{\mathbb{T}} < size(\texttt{a}_1)_{\mathbb{T}}$ then
    $\texttt{j}_{\mathbb{T}} \leftarrow \texttt{a}_{1\mathbb{T}}[\texttt{i}_{\mathbb{T}}]$;
    $\texttt{x}_{\mathbb{F}} \leftarrow \texttt{a}_{2\mathbb{F}}[\texttt{j}_{\mathbb{T}}]$
else
```

- CCT type system:
  - variables and arrays public or secret

Previous Work: Selective SLH (Shivakumar et al. 2023)

MAX PLANCK INSTITUTE
FOR SECURITY AND PRIVACY

```
if $\mathtt{i}_{\mathbb{T}} < size(\mathtt{a}_1)_{\mathbb{T}}$ then
    $\mathtt{j}_{\mathbb{T}} \leftarrow \mathtt{a}_{1\mathbb{T}}[\mathtt{i}_{\mathbb{T}}]$;
    $\mathtt{x}_{\mathbb{F}} \leftarrow \mathtt{a}_{2\mathbb{F}}[\mathtt{j}_{\mathbb{T}}]$;
    ~~$\mathtt{y} \leftarrow \mathtt{a}_{3\mathbb{T}}[\mathtt{x}_{\mathbb{F}}]$;~~
    ~~if $\mathtt{y} < 10$ then … else …~~
else
    $\mathtt{b} := \mathtt{i}_{\mathbb{T}} < size(\mathtt{a}_1)_{\mathbb{T}}\ ?\ 1 : \mathtt{b}$
```

- CCT type system:
  - variables and arrays public or secret
  - secret values may not be used as indices or branch conditions

Previous Work: Selective SLH (Shivakumar et al. 2023)

```
if $\mathtt{i}_{\mathbb{T}} < \mathit{size}(\mathtt{a}_1)_{\mathbb{T}}$ then
    $\mathtt{b} := \mathtt{i}_{\mathbb{T}} < \mathit{size}(\mathtt{a}_1)_{\mathbb{T}} \,?\, \mathtt{b} : 1$;
    $\mathtt{j}_{\mathbb{T}} \leftarrow \mathtt{a}_{1\mathbb{T}}[\mathtt{i}_{\mathbb{T}}]$;
    $\mathtt{x}_{\mathbb{F}} \leftarrow \mathtt{a}_{2\mathbb{F}}[\mathtt{j}_{\mathbb{T}}]$;
    ~~$\mathtt{y} \leftarrow \mathtt{a}_{3\mathbb{T}}[\mathtt{x}_{\mathbb{F}}]$;~~
    ~~if $\mathtt{y} < 10$ then … else …~~
else
    $\mathtt{b} := \mathtt{i}_{\mathbb{T}} < \mathit{size}(\mathtt{a}_1)_{\mathbb{T}} \,?\, 1 : \mathtt{b}$
```

- CCT type system:
  - variables and arrays public or secret
  - secret values may not be used as indices or branch conditions
- maintain a misspeculation flag

Previous Work: Selective SLH (Shivakumar et al. 2023)

```
if $\mathtt{i}_{\mathbb{T}} < \mathit{size}(\mathtt{a}_1)_{\mathbb{T}}$ then
   $\mathtt{b} := \mathtt{i}_{\mathbb{T}} < \mathit{size}(\mathtt{a}_1)_{\mathbb{T}} \, ? \, \mathtt{b} : 1$;
   $\mathtt{j}_{\mathbb{T}} \leftarrow \mathtt{a}_{1\mathbb{T}}[\mathtt{i}_{\mathbb{T}}]$;
   $\mathtt{x}_{\mathbb{F}} \leftarrow \mathtt{a}_{2\mathbb{F}}[\mathtt{j}_{\mathbb{T}}]$;
   ~~$\mathtt{y} \leftarrow \mathtt{a}_{3\mathbb{T}}[\mathtt{x}_{\mathbb{F}}]$;~~
   ~~if $\mathtt{y} < 10$ then … else …~~
else
   $\mathtt{b} := \mathtt{i}_{\mathbb{T}} < \mathit{size}(\mathtt{a}_1)_{\mathbb{T}} \, ? \, 1 : \mathtt{b}$
```

- CCT type system:
  - variables and arrays public or secret
  - secret values may not be used as indices or branch conditions
- maintain a misspeculation flag
  - updated with constant-time conditionals

Previous Work: Selective SLH (Shivakumar et al. 2023)

```
if $\mathtt{i}_{\mathbb{T}} < size(\mathtt{a}_1)_{\mathbb{T}}$ then
  $\mathtt{b} := \mathtt{i}_{\mathbb{T}} < size(\mathtt{a}_1)_{\mathbb{T}} \,?\, \mathtt{b} : 1;$
  $\mathtt{j}_{\mathbb{T}} \leftarrow \mathtt{a}_{1\mathbb{T}}[\mathtt{i}_{\mathbb{T}}]; \mathtt{j}_{\mathbb{T}} := \mathtt{b} \,?\, 0 : \mathtt{j}_{\mathbb{T}};$
  $\mathtt{x}_{\mathbb{F}} \leftarrow \mathtt{a}_{2\mathbb{F}}[\mathtt{j}_{\mathbb{T}}];$
  ~~$\mathtt{y} \leftarrow \mathtt{a}_{3\mathbb{T}}[\mathtt{x}_{\mathbb{F}}$~~$];$
  ~~if y < 10 then … else …~~
else
  $\mathtt{b} := \mathtt{i}_{\mathbb{T}} < size(\mathtt{a}_1)_{\mathbb{T}} \,?\, 1 : \mathtt{b}$
```

- CCT type system:
  - variables and arrays public or secret
  - secret values may not be used as indices or branch conditions
- maintain a misspeculation flag
  - updated with constant-time conditionals
- mask reads to public variables

Previous Work: Selective SLH (Shivakumar et al. 2023)

```
if $\mathtt{i}_{\mathbb{T}} < size(\mathtt{a}_1)_{\mathbb{T}}$ then
   $\mathtt{b} := \mathtt{i}_{\mathbb{T}} < size(\mathtt{a}_1)_{\mathbb{T}} \,?\, \mathtt{b} : 1;$
   $\mathtt{j}_{\mathbb{T}} \leftarrow \mathtt{a}_{1\mathbb{T}}[\mathtt{i}_{\mathbb{T}}]; \mathtt{j}_{\mathbb{T}} := \mathtt{b} \,?\, 0 : \mathtt{j}_{\mathbb{T}};$
   $\mathtt{x}_{\mathbb{F}} \leftarrow \mathtt{a}_{2\mathbb{F}}[\mathtt{j}_{\mathbb{T}}];$
   ~~$\mathtt{y} \leftarrow \mathtt{a}_{3\mathbb{T}}[\mathtt{x}_{\mathbb{F}}];$~~
   ~~if y < 10 then … else …~~
else
   $\mathtt{b} := \mathtt{i}_{\mathbb{T}} < size(\mathtt{a}_1)_{\mathbb{T}} \,?\, 1 : \mathtt{b}$
```

- CCT type system:
  - variables and arrays public or secret
  - secret values may not be used as indices or branch conditions
- maintain a misspeculation flag
  - updated with constant-time conditionals
- mask reads to public variables
  - secret variables can not leak anyway

Previous Work: Selective SLH (Shivakumar et al. 2023)

```
if $i_{\mathbb{T}} < size(a_1)_{\mathbb{T}}$ then
    $b := i_{\mathbb{T}} < size(a_1)_{\mathbb{T}} \mathrel{?} b : 1;$
    $j_{\mathbb{T}} \leftarrow a_{1\mathbb{T}}[i_{\mathbb{T}}]; j_{\mathbb{T}} := b \mathrel{?} 0 : j_{\mathbb{T}};$
    $x_{\mathbb{F}} \leftarrow a_{2\mathbb{F}}[j_{\mathbb{T}}];$
    ~~$y \leftarrow a_{3\mathbb{T}}[x_{\mathbb{F}}$~~$];$
    ~~if $y < 10$ then … else …~~
else
    $b := i_{\mathbb{T}} < size(a_1)_{\mathbb{T}} \mathrel{?} 1 : b$
```

- CCT type system:
  - variables and arrays public or secret
  - secret values may not be used as indices or branch conditions
- maintain a misspeculation flag
  - updated with constant-time conditionals
- mask reads to public variables
  - secret variables can not leak anyway
- efficient mitigation with only minimal masking

Previous Work: Selective SLH (Shivakumar et al. 2023)

```
if $\mathtt{i}_{\mathbb{T}} < size(\mathtt{a}_1)_{\mathbb{T}}$ then
  $\mathtt{b} := \mathtt{i}_{\mathbb{T}} < size(\mathtt{a}_1)_{\mathbb{T}} \mathbin{?} \mathtt{b} : 1$;
  $\mathtt{j}_{\mathbb{T}} \leftarrow \mathtt{a}_{1\mathbb{T}}[\mathtt{i}_{\mathbb{T}}]$; $\mathtt{j}_{\mathbb{T}} := \mathtt{b} \mathbin{?} 0 : \mathtt{j}_{\mathbb{T}}$;
  $\mathtt{x}_{\mathbb{F}} \leftarrow \mathtt{a}_{2\mathbb{F}}[\mathtt{j}_{\mathbb{T}}]$;
  ~~$\mathtt{y} \leftarrow \mathtt{a}_{3\mathbb{T}}[\mathtt{x}_{\mathbb{F}}]$;~~
  ~~if $\mathtt{y} < 10$ then … else …~~
else
  $\mathtt{b} := \mathtt{i}_{\mathbb{T}} < size(\mathtt{a}_1)_{\mathbb{T}} \mathbin{?} 1 : \mathtt{b}$
```

- CCT type system:
  - variables and arrays public or secret
  - secret values may not be used as indices or branch conditions
- maintain a misspeculation flag
  - updated with constant-time conditionals
- mask reads to public variables
  - secret variables can not leak anyway

- efficient mitigation with only minimal masking
  - for a very limited class of programs

Previous Work: Ultimate SLH (Zhang et al. 2023)

```
if i $< size(\mathtt{a}_1)$ then
  b $:=$ i $< size(\mathtt{a}_1)$ ? b : $1$;
  j $\leftarrow \mathtt{a}_1$ [i ];
  x $\leftarrow \mathtt{a}_2$ [j ];
  y $\leftarrow \mathtt{a}_3$ [x ];
  if y $< 10$ then … else …
else
  b $:=$ i $< size(\mathtt{a}_1)$ ? $1$ : b
```

- no type system, mask everything

Previous Work: Ultimate SLH (Zhang et al. 2023)

```
if i $< size(a_1)$ then
  b := i $< size(a_1)$ ? b : 1;
  j $\leftarrow a_1$ [b ? 0 : i];
  x $\leftarrow a_2$ [b ? 0 : j];
  y $\leftarrow a_3$ [b ? 0 : x];
  if y $< 10$ then … else …
else
  b := i $< size(a_1)$ ? 1 : b
```

- no type system, mask everything
- mask all indices

Previous Work: Ultimate SLH (Zhang et al. 2023)

```
if i $< size(\mathtt{a}_1)$ then
  b := i $< size(\mathtt{a}_1)$ ? b : 1;
  j $\leftarrow \mathtt{a}_1$ [b ? 0 : i];
  x $\leftarrow \mathtt{a}_2$ [b ? 0 : j];
  y $\leftarrow \mathtt{a}_3$ [b ? 0 : x];
  if b && y $< 10$ then … else …
else
  b := i $< size(\mathtt{a}_1)$ ? 1 : b
```

- no type system, mask everything
- mask all indices
- mask branch conditions as well

Previous Work: Ultimate SLH (Zhang et al. 2023)

```
if i $< size(a_1)$ then
  b := i $< size(a_1)$ ? b : 1;
  j $\leftarrow$ $a_1$ [b ? 0 : i];
  x $\leftarrow$ $a_2$ [b ? 0 : j];
  y $\leftarrow$ $a_3$ [b ? 0 : x];
  if b && y $< 10$ then ... else ...
else
  b := i $< size(a_1)$ ? 1 : b
```

- no type system, mask everything
- mask all indices
- mask branch conditions as well
- mask other leaking instructions

Previous Work: Ultimate SLH (Zhang et al. 2023)

```
if i $< size(\mathtt{a}_1)$ then
  b := i $< size(\mathtt{a}_1)$ ? b : 1;
  j $\leftarrow \mathtt{a}_1$ [b ? 0 : i];
  x $\leftarrow \mathtt{a}_2$ [b ? 0 : j];
  y $\leftarrow \mathtt{a}_3$ [b ? 0 : x];
  if b && y $< 10$ then ... else ...
else
  b := i $< size(\mathtt{a}_1)$ ? 1 : b
```

- no type system, mask everything
- mask all indices
- mask branch conditions as well
- mask other leaking instructions
- applies to all programs

Previous Work: Ultimate SLH (Zhang et al. 2023)

```
if i $< size(\mathtt{a}_1)$ then
  b $:=$ i $< size(\mathtt{a}_1)$ ? b : 1;
  j $\leftarrow \mathtt{a}_1$ [b ? 0 : i];
  x $\leftarrow \mathtt{a}_2$ [b ? 0 : j];
  y $\leftarrow \mathtt{a}_3$ [b ? 0 : x];
  if b && y $< 10$ then ... else ...
else
  b $:=$ i $< size(\mathtt{a}_1)$ ? 1 : b
```

- no type system, mask everything
- mask all indices
- mask branch conditions as well
- mask other leaking instructions
- applies to all programs
  - causes high overhead (150%)

Flexible SLH: The Best of Both Worlds

```
if $\mathtt{i}_{\mathbb{T}} < size(\mathtt{a}_1)_{\mathbb{T}}$ then
    $\mathtt{b} := \mathtt{i}_{\mathbb{T}} < size(\mathtt{a}_1)_{\mathbb{T}} \,?\, \mathtt{b} : 1;$
    $\mathtt{j}_{\mathbb{T}} \leftarrow \mathtt{a}_{1\mathbb{T}}[\mathtt{i}_{\mathbb{T}}]; \mathtt{j}_{\mathbb{T}} := \mathtt{b} \,?\, 0 : \mathtt{j}_{\mathbb{T}};$
    $\mathtt{x}_{\mathbb{F}} \leftarrow \mathtt{a}_{2\mathbb{F}}[\mathtt{j}_{\mathbb{T}}];$
    ~~$\mathtt{y} \leftarrow \mathtt{a}_{3\mathbb{T}}[\mathtt{x}_{\mathbb{F}}];$~~
    ~~if $\mathtt{y} < 10$ then … else …~~
else
    $\mathtt{b} := \mathtt{i}_{\mathbb{T}} < size(\mathtt{a}_1)_{\mathbb{T}} \,?\, 1 : \mathtt{b}$
```

- Obtain security levels with static information-flow analysis

Flexible SLH: The Best of Both Worlds

```
if $\mathtt{i}_{\mathbb{T}} < \mathit{size}(\mathtt{a}_1)_{\mathbb{T}}$ then
    $\mathtt{b} := \mathtt{i}_{\mathbb{T}} < \mathit{size}(\mathtt{a}_1)_{\mathbb{T}} \,?\, \mathtt{b} : 1;$
    $\mathtt{j}_{\mathbb{T}} \leftarrow \mathtt{a}_{1\mathbb{T}}[\mathtt{i}_{\mathbb{T}}]; \mathtt{j}_{\mathbb{T}} := \mathtt{b} \,?\, 0 : \mathtt{j}_{\mathbb{T}};$
    $\mathtt{x}_{\mathbb{F}} \leftarrow \mathtt{a}_{2\mathbb{F}}[\mathtt{j}_{\mathbb{T}}];$
    $\mathtt{y}_{\mathbb{F}} \leftarrow \mathtt{a}_{3\mathbb{T}}[\mathtt{x}_{\mathbb{F}}];$
    if $\mathtt{y}_{\mathbb{F}} < 10$ then ... else ...
else
    $\mathtt{b} := \mathtt{i}_{\mathbb{T}} < \mathit{size}(\mathtt{a}_1)_{\mathbb{T}} \,?\, 1 : \mathtt{b}$
```

- Obtain security levels with static information-flow analysis
  - without restricting the use of secrets

Flexible SLH: The Best of Both Worlds

```
if $\mathtt{i}_{\mathbb{T}} < size(\mathtt{a}_1)_{\mathbb{T}}$ then
  $\mathtt{b} := \mathtt{i}_{\mathbb{T}} < size(\mathtt{a}_1)_{\mathbb{T}} \,?\, \mathtt{b} : 1;$
  $\mathtt{j}_{\mathbb{T}} \leftarrow \mathtt{a}_{1\mathbb{T}}[\mathtt{i}_{\mathbb{T}}]; \mathtt{j}_{\mathbb{T}} := \mathtt{b} \,?\, 0 : \mathtt{j}_{\mathbb{T}};$
  $\mathtt{x}_{\mathbb{F}} \leftarrow \mathtt{a}_{2\mathbb{F}}[\mathtt{j}_{\mathbb{T}}];$
  $\mathtt{y}_{\mathbb{F}} \leftarrow \mathtt{a}_{3\mathbb{T}}[\mathtt{x}_{\mathbb{F}}];$
  if $\mathtt{y}_{\mathbb{F}} < 10$ then ... else ...
else
  $\mathtt{b} := \mathtt{i}_{\mathbb{T}} < size(\mathtt{a}_1)_{\mathbb{T}} \,?\, 1 : \mathtt{b}$
```

- Obtain security levels with static information-flow analysis
  - without restricting the use of secrets
  - accepts all programs

Flexible SLH: The Best of Both Worlds

```
if $\mathtt{i}_{\mathbb{T}} < size(\mathtt{a}_1)_{\mathbb{T}}$ then
    $\mathtt{b} := \mathtt{i}_{\mathbb{T}} < size(\mathtt{a}_1)_{\mathbb{T}} \,?\, \mathtt{b} : 1;$
    $\mathtt{j}_{\mathbb{T}} \leftarrow \mathtt{a}_{1\mathbb{T}}[\mathtt{i}_{\mathbb{T}}]; \mathtt{j}_{\mathbb{T}} := \mathtt{b} \,?\, 0 : \mathtt{j}_{\mathbb{T}};$
    $\mathtt{x}_{\mathbb{F}} \leftarrow \mathtt{a}_{2\mathbb{F}}[\mathtt{j}_{\mathbb{T}}];$
    $\mathtt{y}_{\mathbb{F}} \leftarrow \mathtt{a}_{3\mathbb{T}}[\mathtt{x}_{\mathbb{F}}];$
    if $\mathtt{y}_{\mathbb{F}} < 10$ then ... else ...
else
    $\mathtt{b} := \mathtt{i}_{\mathbb{T}} < size(\mathtt{a}_1)_{\mathbb{T}} \,?\, 1 : \mathtt{b}$
```

- Obtain security levels with static information-flow analysis
  - without restricting the use of secrets
  - accepts all programs
- Combine masking approaches:

Flexible SLH: The Best of Both Worlds

```
if $\mathtt{i}_{\mathbb{T}} < size(\mathtt{a}_1)_{\mathbb{T}}$ then
    $\mathtt{b} := \mathtt{i}_{\mathbb{T}} < size(\mathtt{a}_1)_{\mathbb{T}} \mathrel{?} \mathtt{b} : 1;$
    $\mathtt{j}_{\mathbb{T}} \leftarrow \mathtt{a}_{1\mathbb{T}}[\mathtt{i}_{\mathbb{T}}]; \mathtt{j}_{\mathbb{T}} := \mathtt{b} \mathrel{?} 0 : \mathtt{j}_{\mathbb{T}};$
    $\mathtt{x}_{\mathbb{F}} \leftarrow \mathtt{a}_{2\mathbb{F}}[\mathtt{j}_{\mathbb{T}}];$
    $\mathtt{y}_{\mathbb{F}} \leftarrow \mathtt{a}_{3\mathbb{T}}[\mathtt{b} \mathrel{?} 0 : \mathtt{x}];$
    if $\mathtt{y}_{\mathbb{F}} < 10$ then … else …
else
    $\mathtt{b} := \mathtt{i}_{\mathbb{T}} < size(\mathtt{a}_1)_{\mathbb{T}} \mathrel{?} 1 : \mathtt{b}$
```

- Obtain security levels with static information-flow analysis
  - without restricting the use of secrets
  - accepts all programs
- Combine masking approaches:
  - mask secret indices

Flexible SLH: The Best of Both Worlds

```
if $\mathtt{i}_{\mathbb{T}} < size(\mathtt{a}_1)_{\mathbb{T}}$ then
  $\mathtt{b} := \mathtt{i}_{\mathbb{T}} < size(\mathtt{a}_1)_{\mathbb{T}} \,?\, \mathtt{b} : 1;$
  $\mathtt{j}_{\mathbb{T}} \leftarrow \mathtt{a}_{1\mathbb{T}}[\mathtt{i}_{\mathbb{T}}];\ \mathtt{j}_{\mathbb{T}} := \mathtt{b} \,?\, 0 : \mathtt{j}_{\mathbb{T}};$
  $\mathtt{x}_{\mathbb{F}} \leftarrow \mathtt{a}_{2\mathbb{F}}[\mathtt{j}_{\mathbb{T}}];$
  $\mathtt{y}_{\mathbb{F}} \leftarrow \mathtt{a}_{3\mathbb{T}}[\mathtt{b} \,?\, 0 : \mathtt{x}];$
  if $\mathtt{y}_{\mathbb{F}} < 10$ then ... else ...
else
  $\mathtt{b} := \mathtt{i}_{\mathbb{T}} < size(\mathtt{a}_1)_{\mathbb{T}} \,?\, 1 : \mathtt{b}$
```

- Obtain security levels with static information-flow analysis
  - without restricting the use of secrets
  - accepts all programs
- Combine masking approaches:
  - mask secret indices
  - mask only values for public indices

Flexible SLH: The Best of Both Worlds

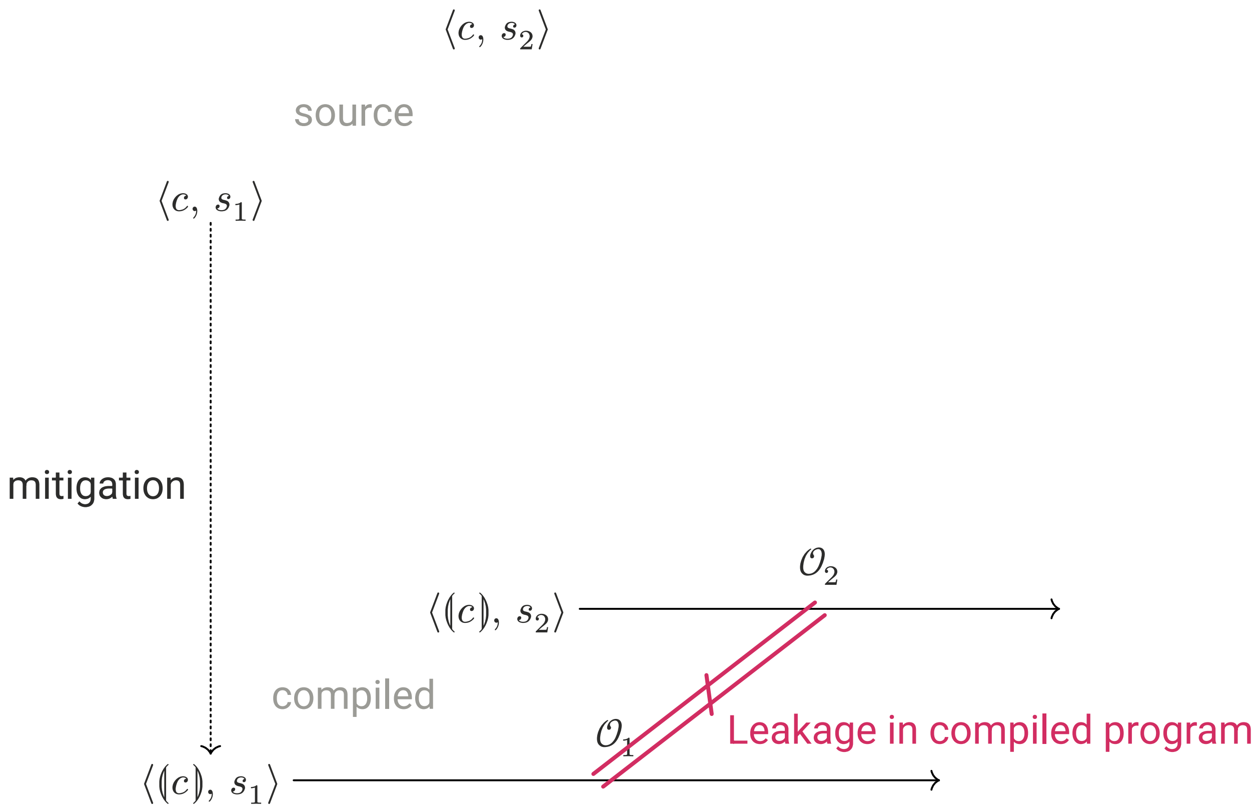
```
if $\mathtt{i}_{\mathbb{T}} < size(\mathtt{a}_1)_{\mathbb{T}}$ then
    $\mathtt{b} := \mathtt{i}_{\mathbb{T}} < size(\mathtt{a}_1)_{\mathbb{T}} \,?\, \mathtt{b} : 1;$
    $\mathtt{j}_{\mathbb{T}} \leftarrow \mathtt{a}_{1\mathbb{T}}[\mathtt{i}_{\mathbb{T}}]; \mathtt{j}_{\mathbb{T}} := \mathtt{b} \,?\, 0 : \mathtt{j}_{\mathbb{T}};$
    $\mathtt{x}_{\mathbb{F}} \leftarrow \mathtt{a}_{2\mathbb{F}}[\mathtt{j}_{\mathbb{T}}];$
    $\mathtt{y}_{\mathbb{F}} \leftarrow \mathtt{a}_{3\mathbb{T}}[\mathtt{b} \,?\, 0 : \mathtt{x}];$
    if $\mathtt{b} \,\&\&\, \mathtt{y}_{\mathbb{F}} < 10$ then … else …
else
    $\mathtt{b} := \mathtt{i}_{\mathbb{T}} < size(\mathtt{a}_1)_{\mathbb{T}} \,?\, 1 : \mathtt{b}$
```

- Obtain security levels with static information-flow analysis
  - without restricting the use of secrets
  - accepts all programs
- Combine masking approaches:
  - mask secret indices
  - mask only values for public indices
  - mask only secret branch conditions

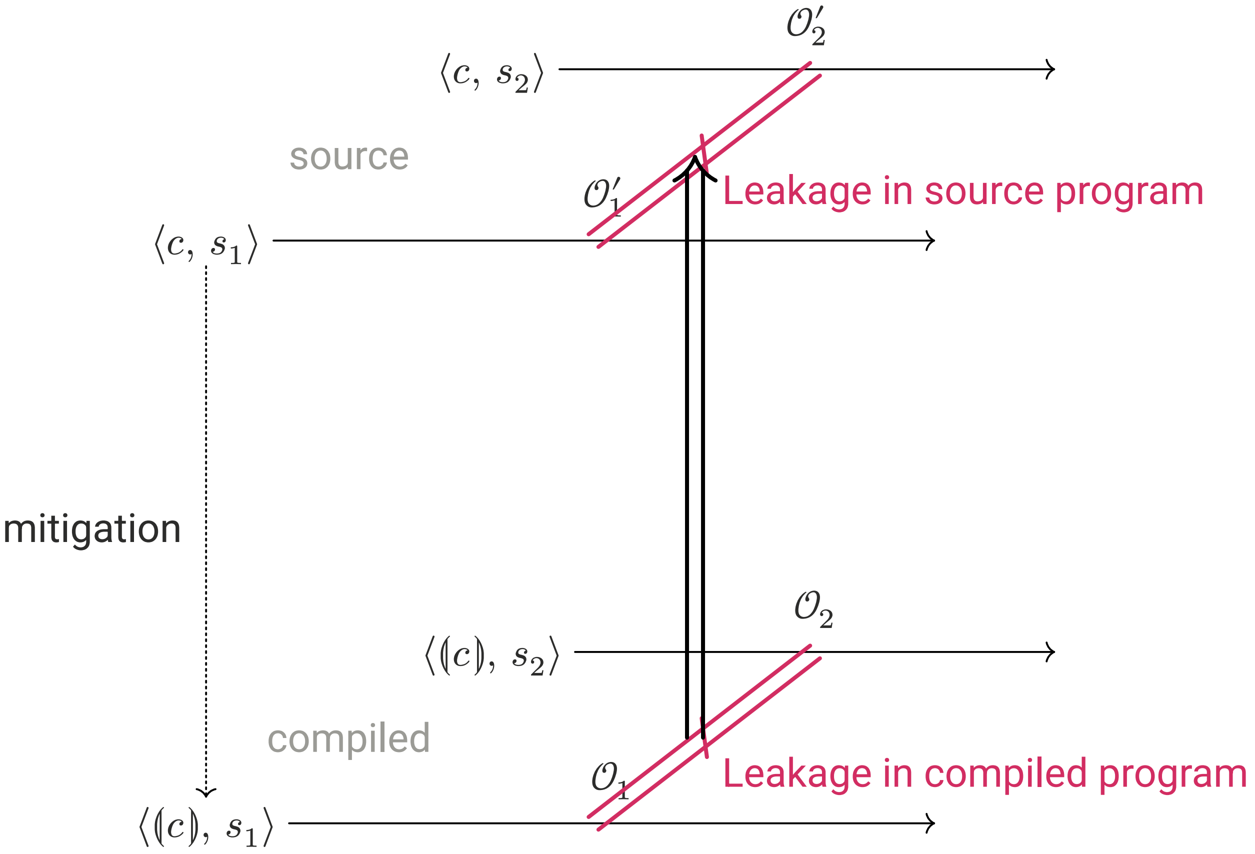
Security for Arbitrary Programs

$$\begin{tikzcd}
 & & \langle c,\ s_2\rangle & & & \\
 & \text{source} & & & & \\
\langle c,\ s_1\rangle \arrow[ddd, dotted, "\text{mitigation}"'] & & & & & \\
 & & & & & \\
 & & \langle (\!|c|\!),\ s_2\rangle \arrow[rrr, "\mathcal{O}_2"] & & & {} \\
 & \text{compiled} & & & & \\
\langle (\!|c|\!),\ s_1\rangle \arrow[rrrr, "\mathcal{O}_1"] & & & & {} &
\end{tikzcd}$$

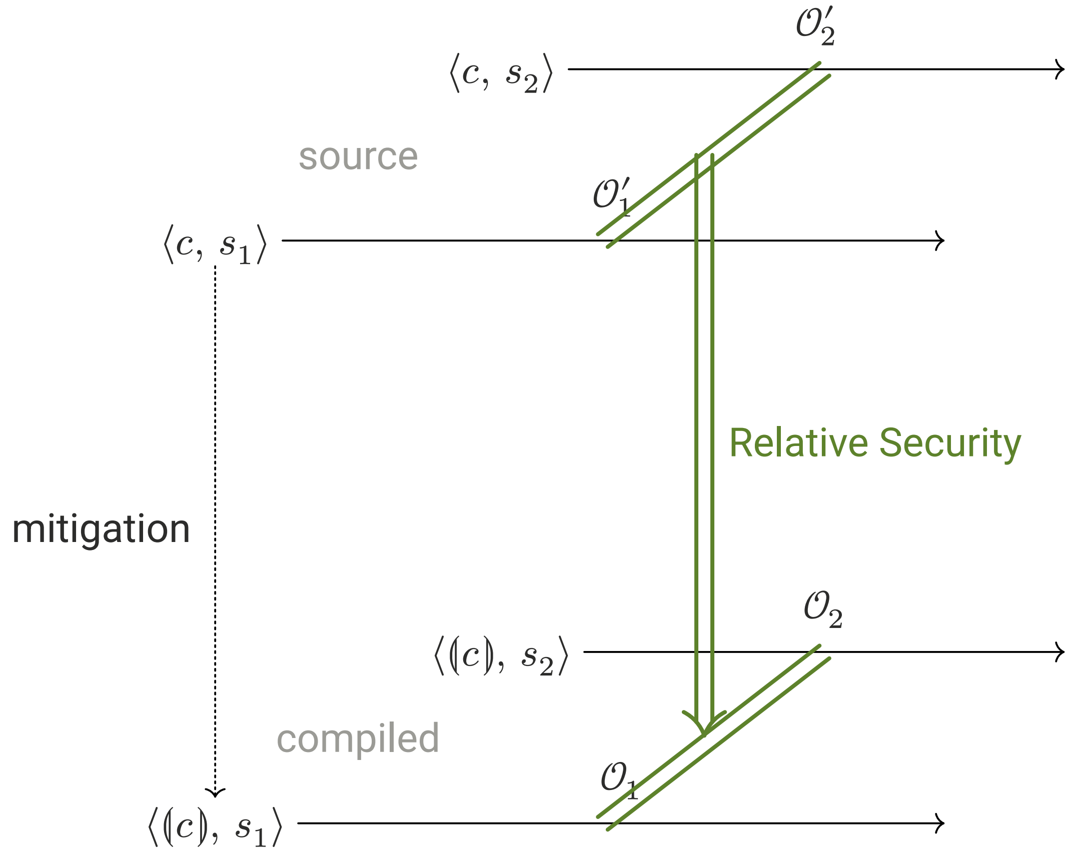
Security for Arbitrary Programs



Security for Arbitrary Programs


$\mathcal{O}_2'$
$\langle c,\ s_2 \rangle$
source
$\mathcal{O}_1'$
Leakage in source program
$\langle c,\ s_1 \rangle$
mitigation
$\mathcal{O}_2$
$\langle (\!| c |\!),\ s_2 \rangle$
compiled
$\mathcal{O}_1$
Leakage in compiled program
$\langle (\!| c |\!),\ s_1 \rangle$

Security for Arbitrary Programs


$\mathcal{O}_2'$
$\langle c,\ s_2 \rangle$
source
$\mathcal{O}_1'$
$\langle c,\ s_1 \rangle$
Relative Security
mitigation
$\mathcal{O}_2$
$\langle (\!|c|\!),\ s_2 \rangle$
compiled
$\mathcal{O}_1$
$\langle (\!|c|\!),\ s_1 \rangle$

Modeling Speculative Execution Attacks

- Attacker model:

Modeling Speculative Execution Attacks

- Attacker model:
  - observes control flow (branch conditions) and indices of memory accesses

Modeling Speculative Execution Attacks

- Attacker model:
  - observes control flow (branch conditions) and indices of memory accesses
  - directly controls speculation using directives

Modeling Speculative Execution Attacks

- Attacker model:
  - observes control flow (branch conditions) and indices of memory accesses
  - directly controls speculation using directives
  - chooses locations of out-of-bounds accesses using directives

Modeling Speculative Execution Attacks

- Attacker model:
  - observes control flow (branch conditions) and indices of memory accesses
  - directly controls speculation using directives
  - chooses locations of out-of-bounds accesses using directives
- Modeling speculative execution:

Modeling Speculative Execution Attacks

- Attacker model:
  - observes control flow (branch conditions) and indices of memory accesses
  - directly controls speculation using directives
  - chooses locations of out-of-bounds accesses using directives
- Modeling speculative execution:
  - Forward-only semantics (Barthe et al. 2021)

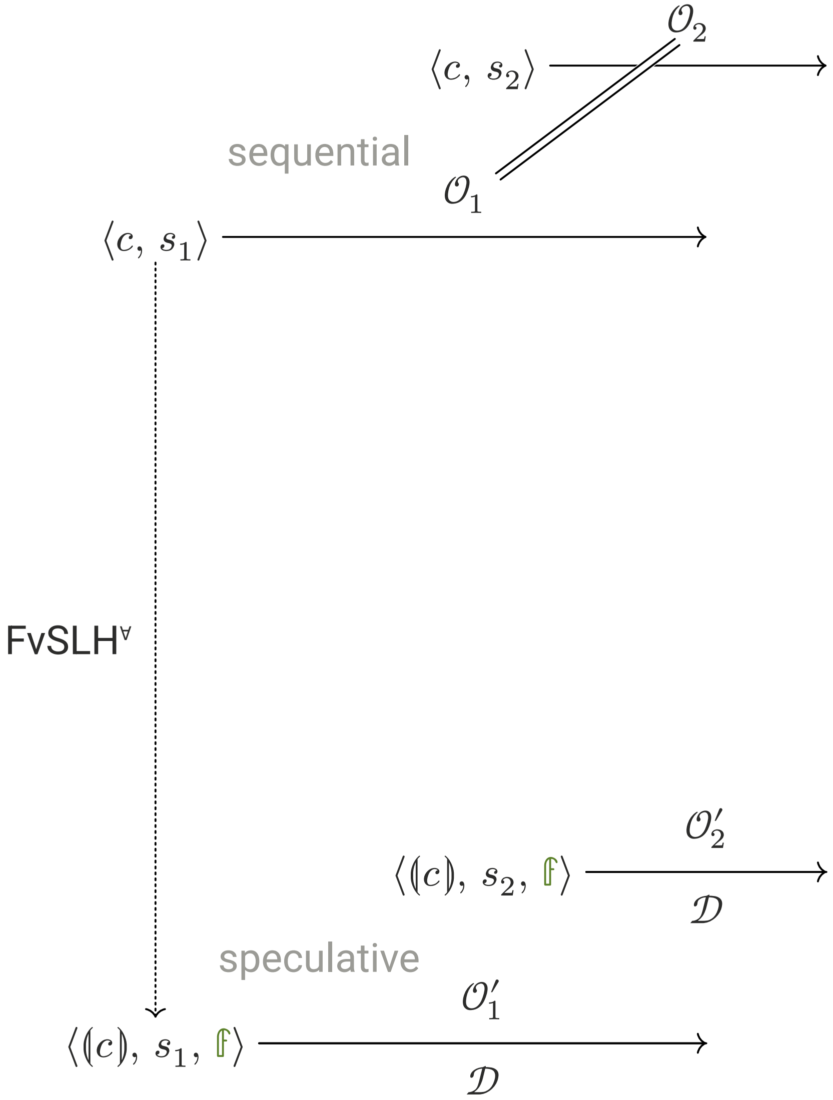
Modeling Speculative Execution Attacks

- Attacker model:
  - observes control flow (branch conditions) and indices of memory accesses
  - directly controls speculation using directives
  - chooses locations of out-of-bounds accesses using directives
- Modeling speculative execution:
  - Forward-only semantics (Barthe et al. 2021)
    - no rollbacks

Modeling Speculative Execution Attacks

- Attacker model:
  - observes control flow (branch conditions) and indices of memory accesses
  - directly controls speculation using directives
  - chooses locations of out-of-bounds accesses using directives
- Modeling speculative execution:
  - Forward-only semantics (Barthe et al. 2021)
    - no rollbacks
  - results carry over to semantics with rollbacks

Proving Relative Security



$\langle c,\, s_2\rangle$

$\mathcal{O}_2$

sequential

$\mathcal{O}_1$

$\langle c,\, s_1\rangle$

FvSLH$^\forall$

$\langle (\!| c |\!),\, s_2,\, \mathbb{f}\rangle$

$\mathcal{O}_2'$

$\mathcal{D}$

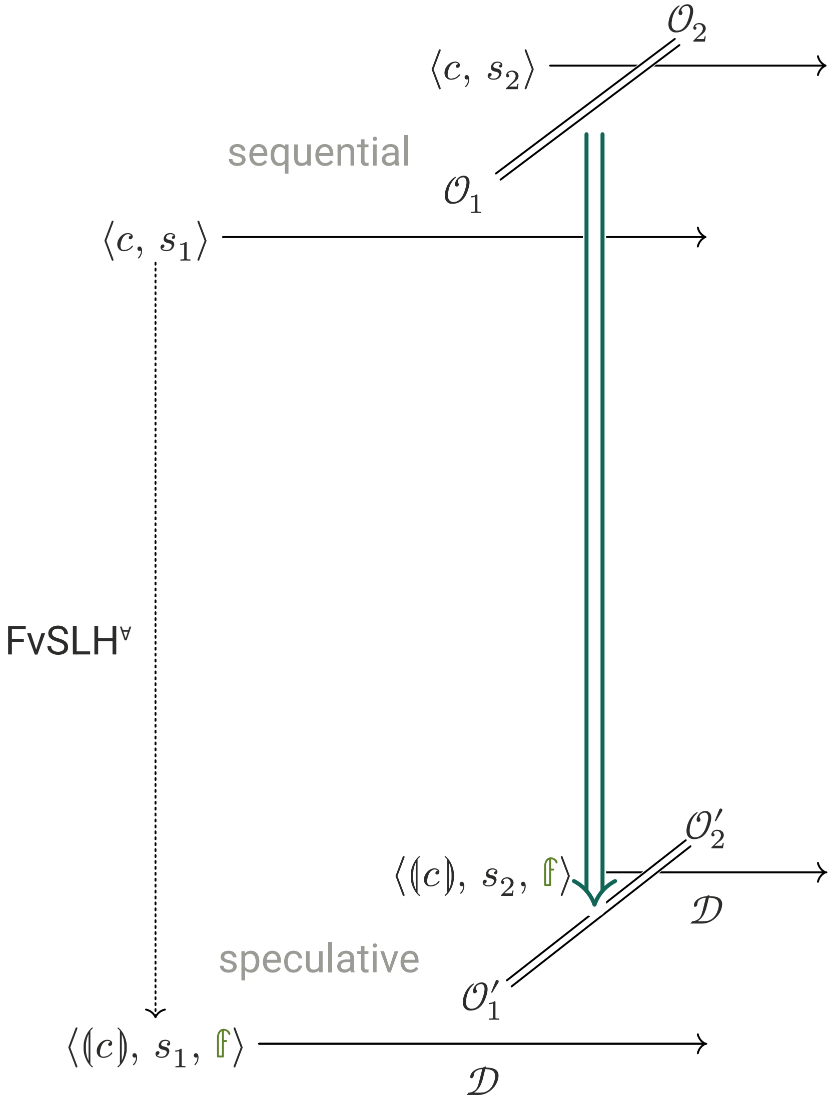
speculative

$\mathcal{O}_1'$

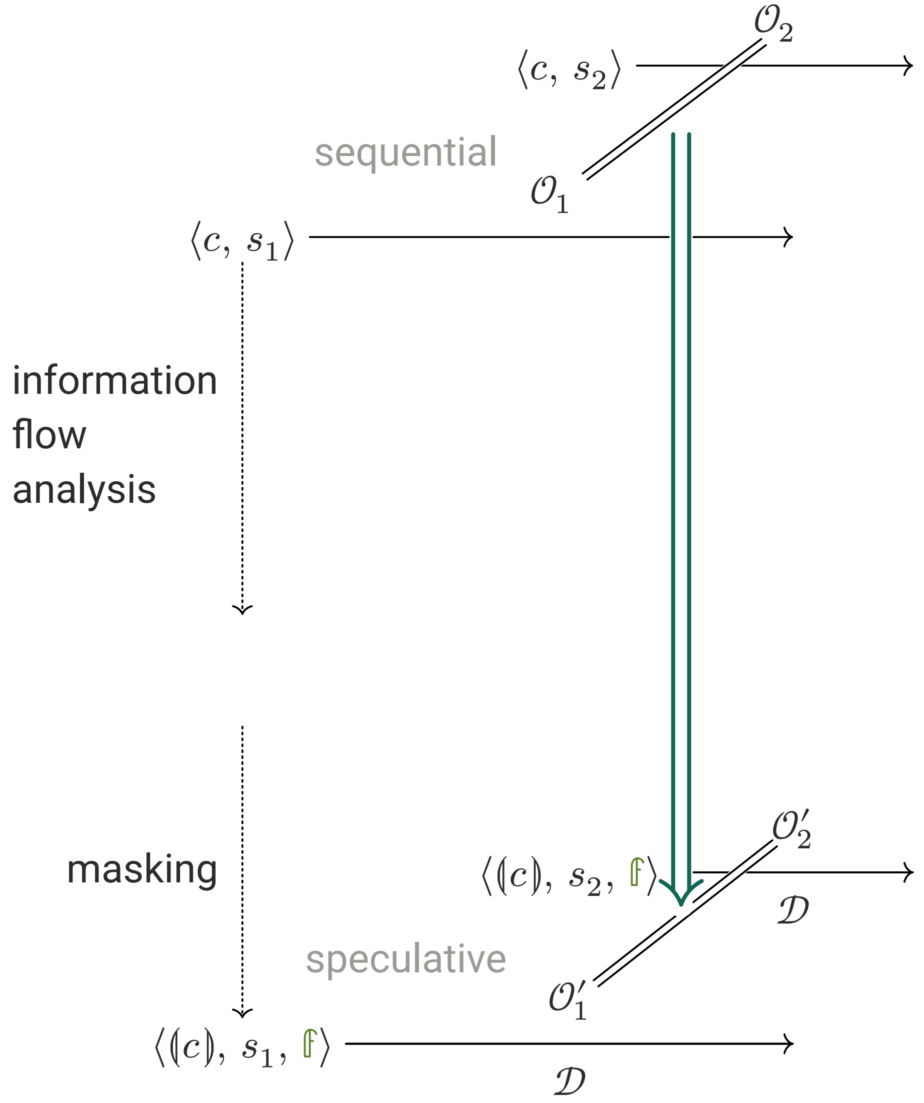
$\langle (\!| c |\!),\, s_1,\, \mathbb{f}\rangle$

$\mathcal{D}$

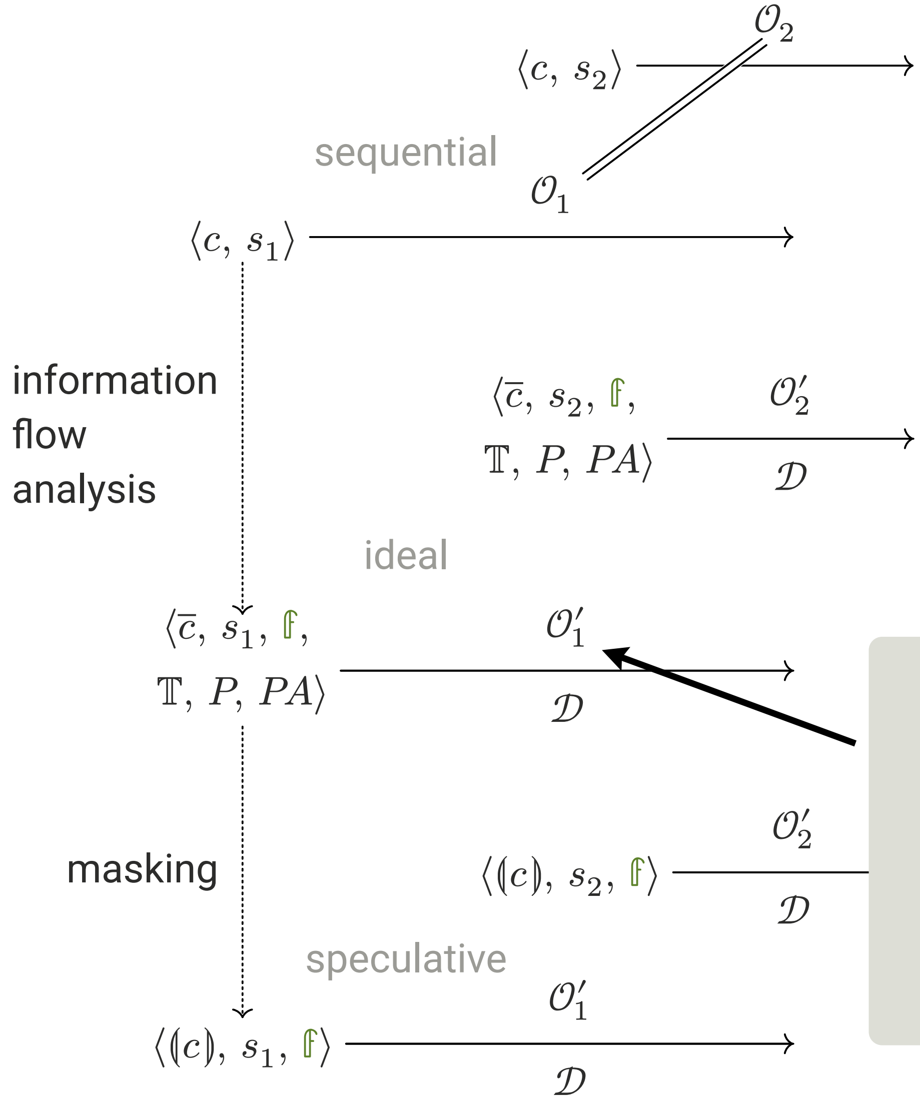
Proving Relative Security


$\mathcal{O}_2$
$\langle c,\ s_2 \rangle$
sequential
$\mathcal{O}_1$
$\langle c,\ s_1 \rangle$
FvSLH$^\forall$
$\mathcal{O}_2'$
$\langle (\!| c |\!),\ s_2,\ \mathbb{f} \rangle$
$\mathcal{D}$
speculative
$\mathcal{O}_1'$
$\langle (\!| c |\!),\ s_1,\ \mathbb{f} \rangle$
$\mathcal{D}$

Proving Relative Security



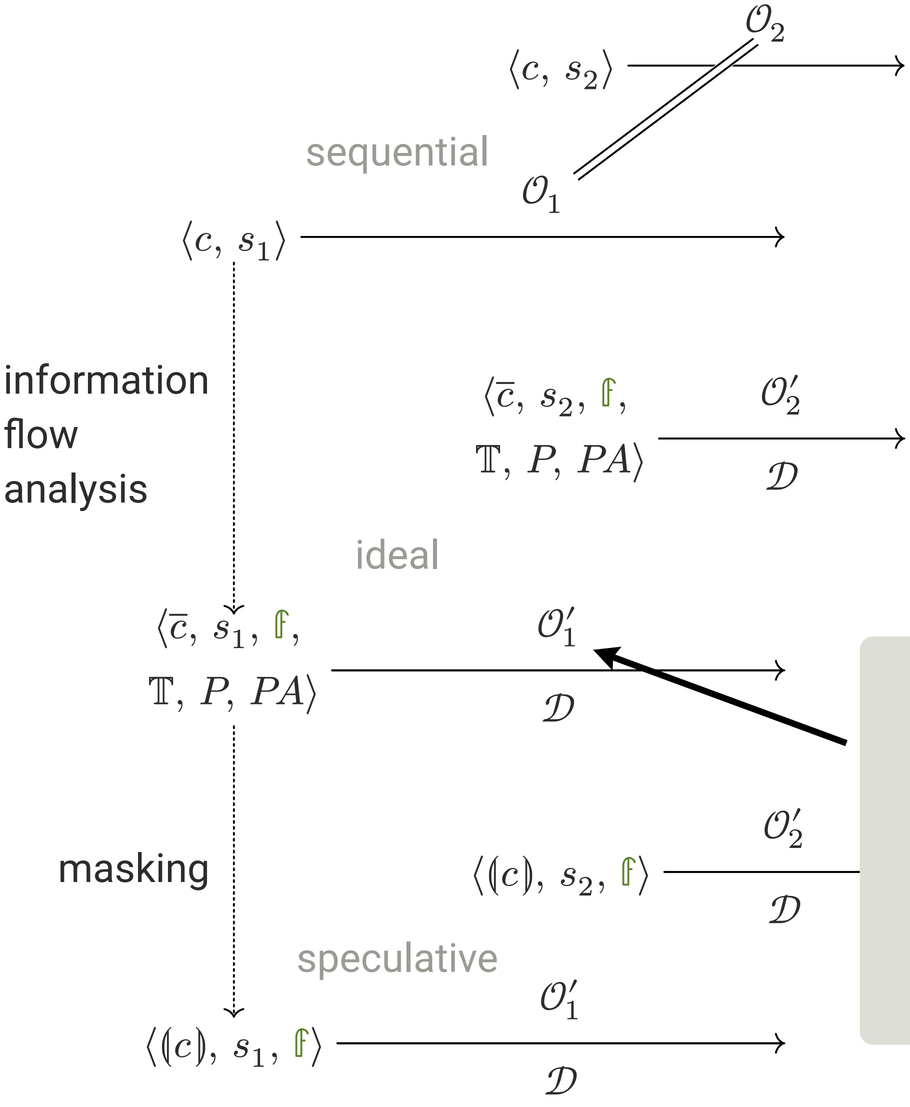
Proving Relative Security



$\langle c,\ s_2 \rangle \xrightarrow{\mathcal{O}_2}$
sequential
$\langle c,\ s_1 \rangle \xrightarrow{\mathcal{O}_1}$
information flow analysis
$\langle \overline{c},\ s_2,\ \mathbb{f},\ \mathbb{T},\ P,\ PA \rangle \xrightarrow[\mathcal{D}]{\mathcal{O}_2'}$
ideal
$\langle \overline{c},\ s_1,\ \mathbb{f},\ \mathbb{T},\ P,\ PA \rangle \xrightarrow[\mathcal{D}]{\mathcal{O}_1'}$
masking
$\langle (\!| c |\!),\ s_2,\ \mathbb{f} \rangle \xrightarrow[\mathcal{D}]{\mathcal{O}_2'}$
speculative
$\langle (\!| c |\!),\ s_1,\ \mathbb{f} \rangle \xrightarrow[\mathcal{D}]{\mathcal{O}_1'}$

Ideal semantics:

Proving Relative Security



$\mathcal{O}_2$

$\langle c,\ s_2\rangle$

sequential

$\mathcal{O}_1$

$\langle c,\ s_1\rangle$

information flow analysis

$\langle \overline{c},\ s_2,\ \mathbb{f},$ $\mathbb{T},\ P,\ PA\rangle$

$\mathcal{O}'_2$

$\mathcal{D}$

ideal

$\langle \overline{c},\ s_1,\ \mathbb{f},$ $\mathbb{T},\ P,\ PA\rangle$

$\mathcal{O}'_1$

$\mathcal{D}$

masking

$\langle (\!|c|\!),\ s_2,\ \mathbb{f}\rangle$

$\mathcal{O}'_2$

$\mathcal{D}$

speculative

$\langle (\!|c|\!),\ s_1,\ \mathbb{f}\rangle$

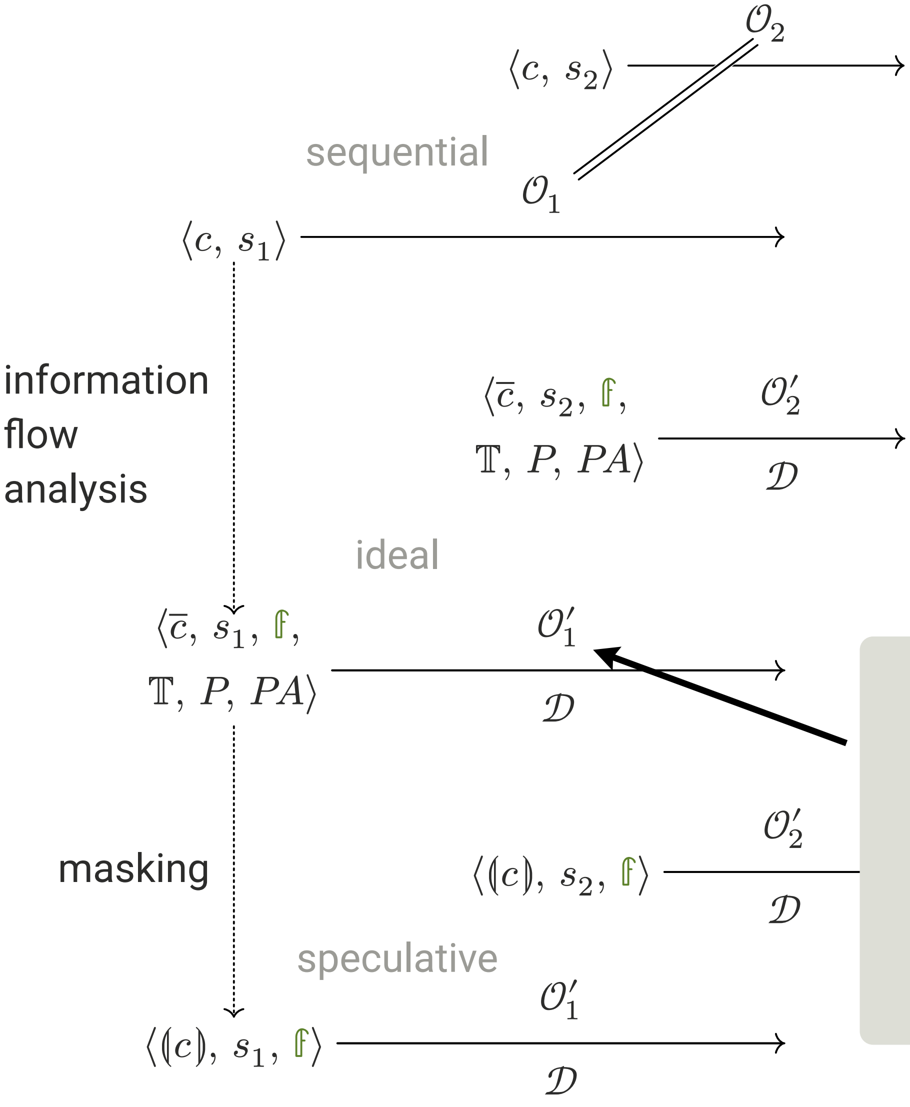
$\mathcal{O}'_1$

$\mathcal{D}$

Ideal semantics:

- speculative execution

Proving Relative Security



$\mathcal{O}_2$

$\langle c,\ s_2\rangle$

sequential

$\mathcal{O}_1$

$\langle c,\ s_1\rangle$

information flow analysis

$\langle \overline{c},\ s_2,\ \mathbb{f},\ \mathbb{T},\ P,\ PA\rangle$ $\xrightarrow[\mathcal{D}]{\mathcal{O}_2'}$

ideal

$\langle \overline{c},\ s_1,\ \mathbb{f},\ \mathbb{T},\ P,\ PA\rangle$ $\xrightarrow[\mathcal{D}]{\mathcal{O}_1'}$

masking

$\langle (\!|c|\!),\ s_2,\ \mathbb{f}\rangle$ $\xrightarrow[\mathcal{D}]{\mathcal{O}_2'}$

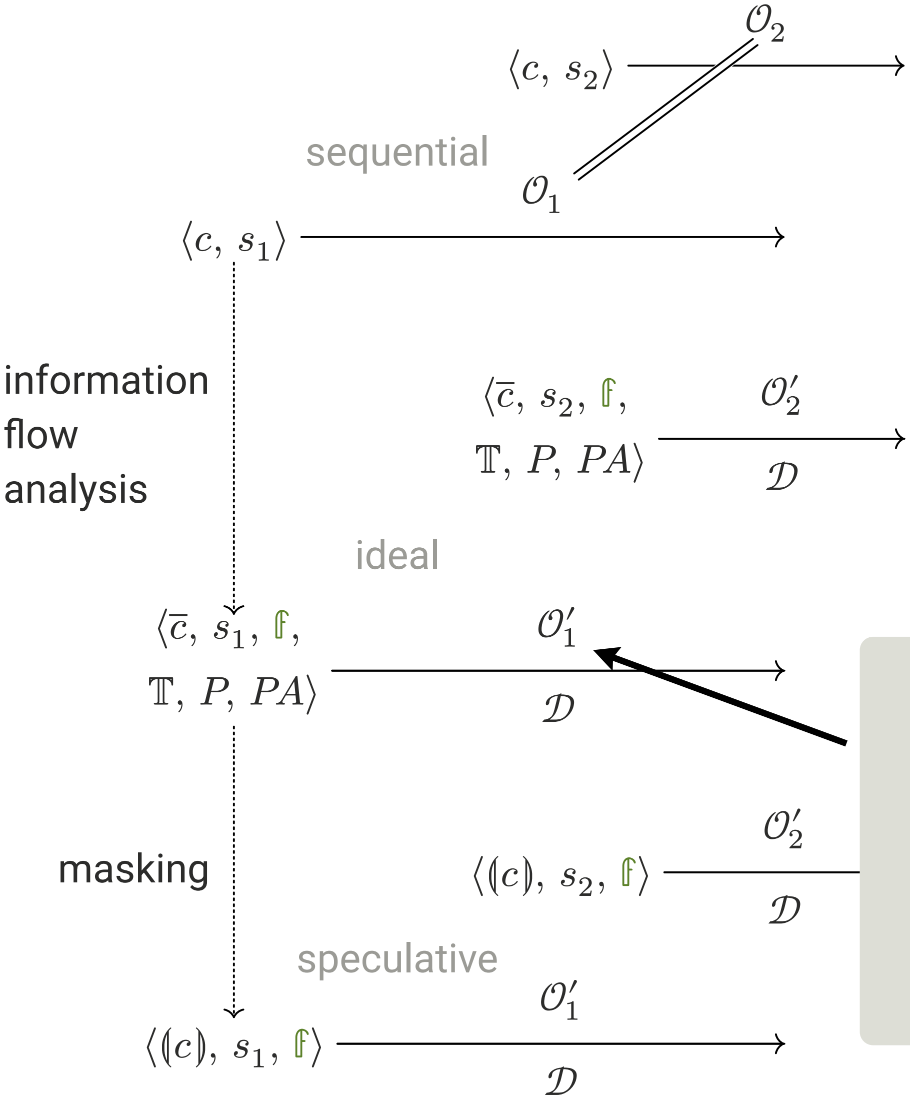
speculative

$\langle (\!|c|\!),\ s_1,\ \mathbb{f}\rangle$ $\xrightarrow[\mathcal{D}]{\mathcal{O}_1'}$

Ideal semantics:

- speculative execution
- with masking in semantics

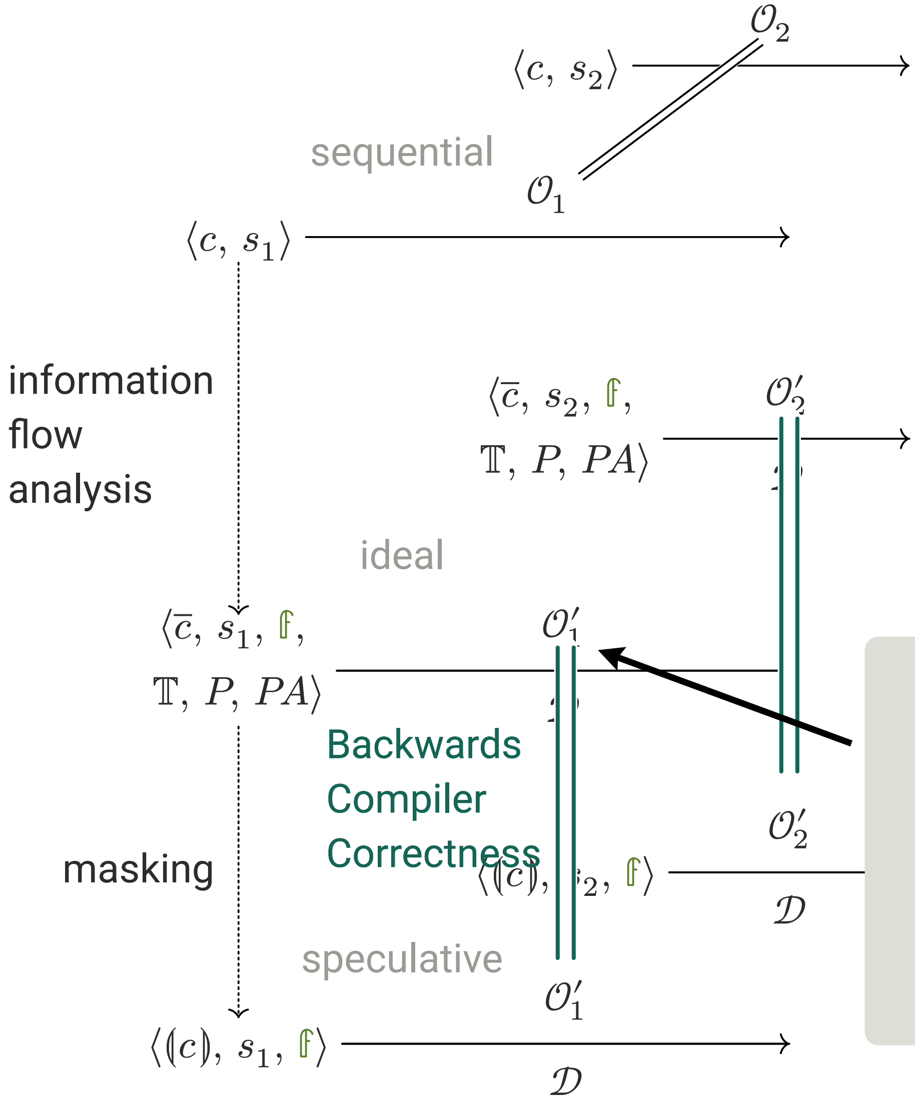
Proving Relative Security


$\mathcal{O}_2$
$\langle c,\ s_2\rangle$
sequential
$\mathcal{O}_1$
$\langle c,\ s_1\rangle$
information flow analysis
$\langle \overline{c},\ s_2,\ \mathbb{f},\ \mathbb{T},\ P,\ PA\rangle$
$\mathcal{O}'_2$
$\mathcal{D}$
ideal
$\langle \overline{c},\ s_1,\ \mathbb{f},\ \mathbb{T},\ P,\ PA\rangle$
$\mathcal{O}'_1$
$\mathcal{D}$
masking
$\langle (\!| c |\!),\ s_2,\ \mathbb{f}\rangle$
$\mathcal{O}'_2$
$\mathcal{D}$
speculative
$\mathcal{O}'_1$
$\langle (\!| c |\!),\ s_1,\ \mathbb{f}\rangle$
$\mathcal{D}$

Ideal semantics:

- speculative execution
- with masking in semantics
  - matches behaviour of compiled program

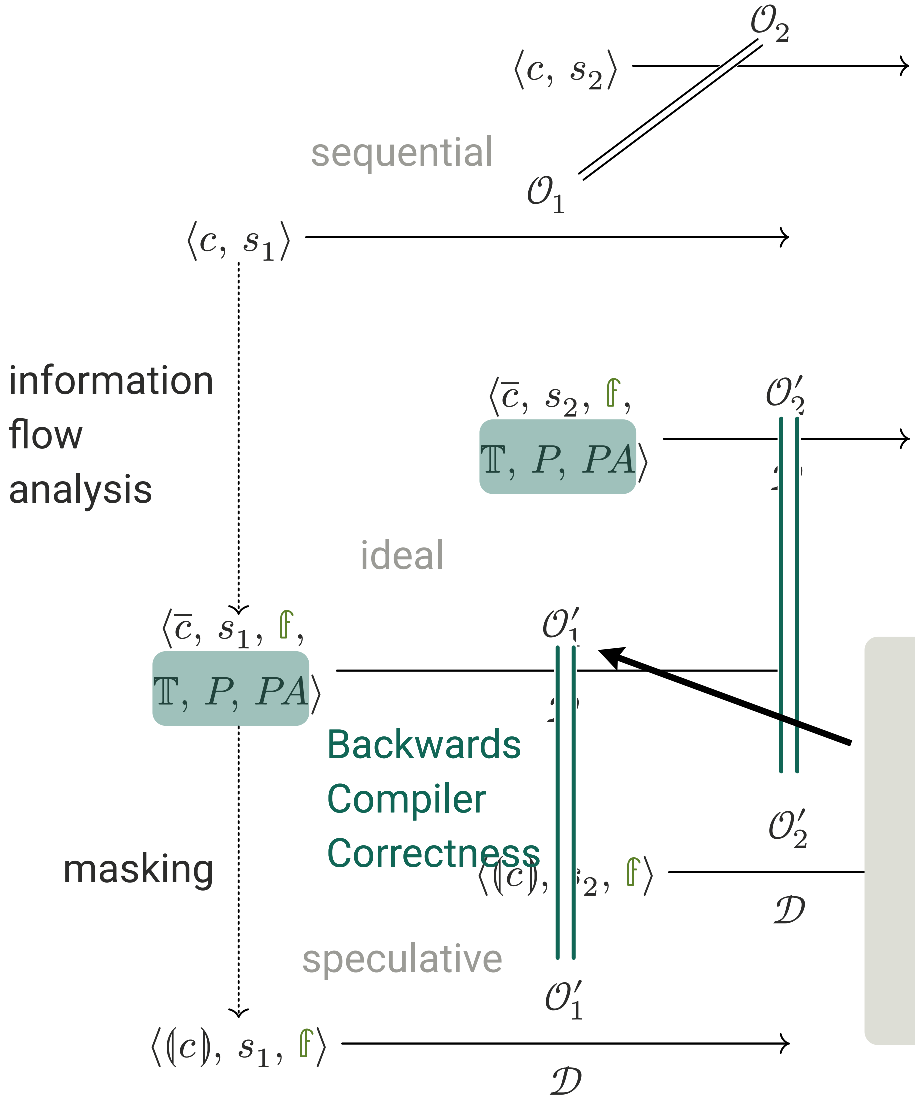
Proving Relative Security


$\mathcal{O}_2$
$\langle c,\ s_2\rangle$
sequential
$\mathcal{O}_1$
$\langle c,\ s_1\rangle$
information flow analysis
$\langle \overline{c},\ s_2,\ \mathbb{f},\ \mathbb{T},\ P,\ PA\rangle$
$\mathcal{O}_2'$
ideal
$\langle \overline{c},\ s_1,\ \mathbb{f},\ \mathbb{T},\ P,\ PA\rangle$
$\mathcal{O}_1'$
Backwards Compiler Correctness
$\mathcal{O}_2'$
$\langle (\!| c |\!),\ s_2,\ \mathbb{f}\rangle$
$\mathcal{D}$
masking
speculative
$\mathcal{O}_1'$
$\langle (\!| c |\!),\ s_1,\ \mathbb{f}\rangle$
$\mathcal{D}$

Ideal semantics:

- speculative execution
- with masking in semantics
  - matches behaviour of compiled program

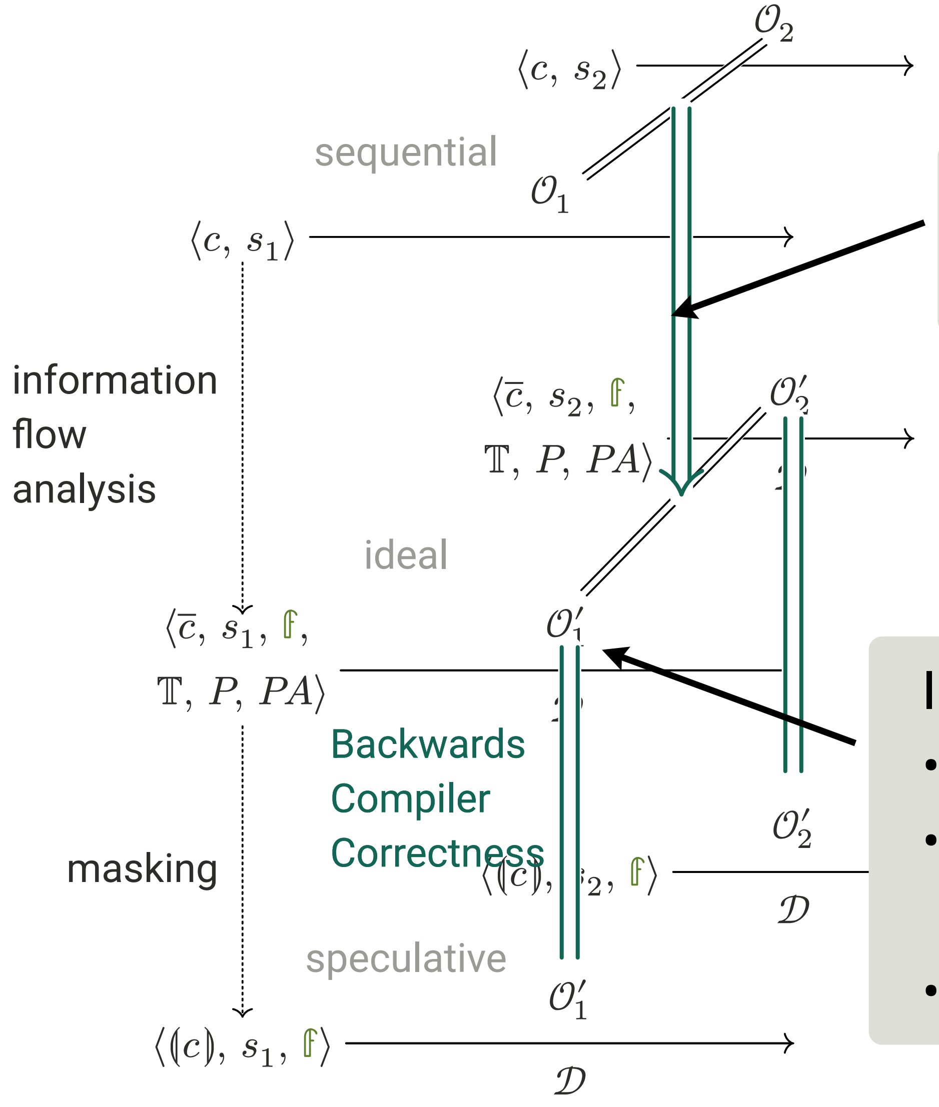
Proving Relative Security


$\mathcal{O}_2$
$\langle c,\ s_2\rangle$
sequential
$\mathcal{O}_1$
$\langle c,\ s_1\rangle$
information flow analysis
$\langle \overline{c},\ s_2,\ \mathbb{f},$
$\mathbb{T},\ P,\ PA\rangle$
$\mathcal{O}_2'$
ideal
$\langle \overline{c},\ s_1,\ \mathbb{f},$
$\mathbb{T},\ P,\ PA\rangle$
$\mathcal{O}_1'$
Backwards Compiler Correctness
masking
$\langle (\!|c|\!),\ s_2,\ \mathbb{f}\rangle$
$\mathcal{O}_2'$
$\mathcal{D}$
speculative
$\mathcal{O}_1'$
$\langle (\!|c|\!),\ s_1,\ \mathbb{f}\rangle$
$\mathcal{D}$

Ideal semantics:

- speculative execution
- with masking in semantics
  - matches behaviour of compiled program
- with dynamic information-flow tracking

Proving Relative Security

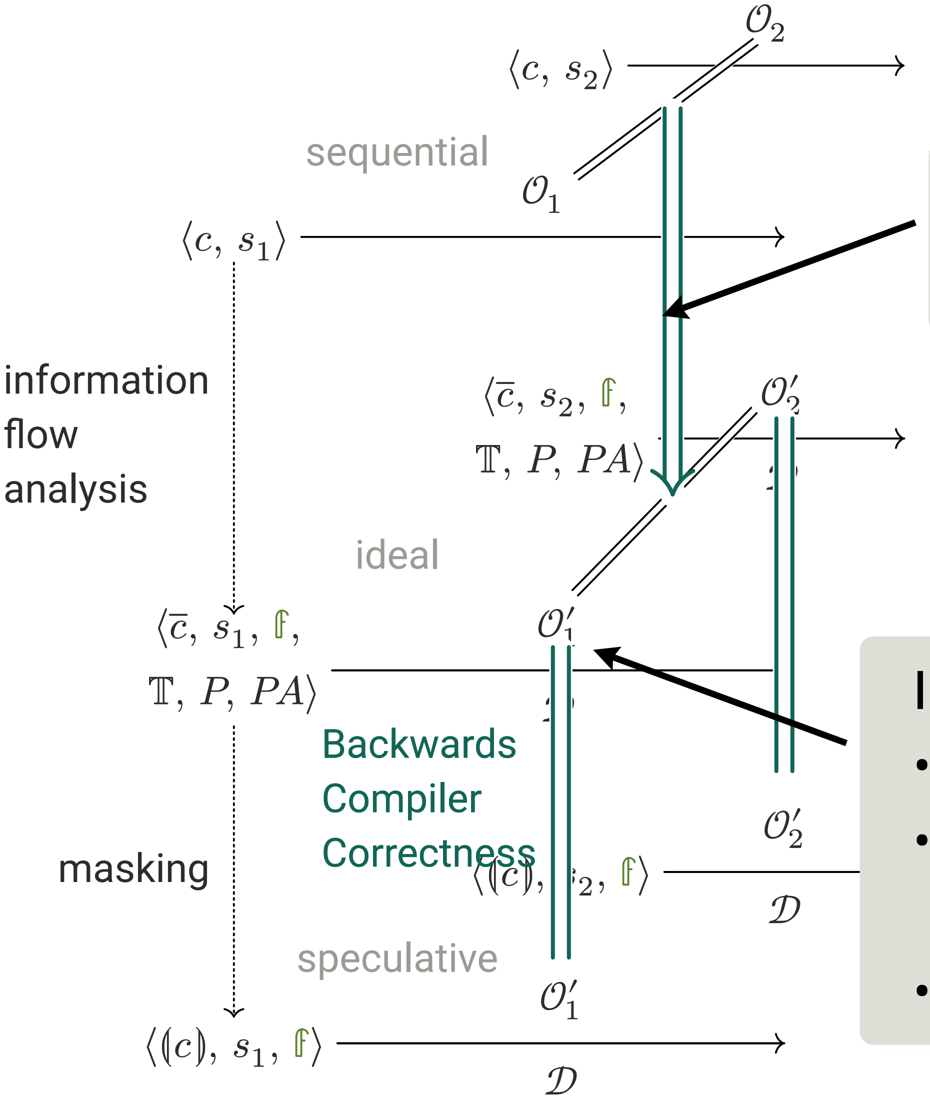

$\mathcal{O}_2$
$\langle c,\ s_2\rangle$
sequential
$\mathcal{O}_1$
$\langle c,\ s_1\rangle$
information flow analysis
$\langle \overline{c},\ s_2,\ \mathbb{f},\ \mathbb{T},\ P,\ PA\rangle$
$\mathcal{O}'_2$
ideal
$\langle \overline{c},\ s_1,\ \mathbb{f},\ \mathbb{T},\ P,\ PA\rangle$
$\mathcal{O}'_1$
Backwards Compiler Correctness
masking
$\langle (\!|c|\!),\ s_2,\ \mathbb{f}\rangle$
$\mathcal{O}'_2$
$\mathcal{D}$
speculative
$\mathcal{O}'_1$
$\langle (\!|c|\!),\ s_1,\ \mathbb{f}\rangle$
$\mathcal{D}$

Relative Security of ideal semantics:

Ideal semantics:

- speculative execution
- with masking in semantics
  - matches behaviour of compiled program
- with dynamic information-flow tracking

Proving Relative Security



Relative Security of ideal semantics:

⚠ depends on correctness of annotations

Ideal semantics:

- speculative execution
- with masking in semantics
  - matches behaviour of compiled program
- with dynamic information-flow tracking

Well-Labeledness

- Relative security requires correct annotations during execution

Well-Labeledness

- Relative security requires correct annotations during execution
- Annotations are produced by static analysis on the initial program

Well-Labeledness

- Relative security requires correct annotations during execution
- Annotations are produced by static analysis on the initial program
    - not suitable for preservation

Well-Labeledness

- Relative security requires correct annotations during execution
- Annotations are produced by static analysis on the initial program
  - not suitable for preservation
- Introduce a *typing-like* well-labeledness predicate:

Well-Labeledness

- Relative security requires correct annotations during execution
- Annotations are produced by static analysis on the initial program
  - not suitable for preservation
- Introduce a *typing-like* well-labeledness predicate:

$$P, PA \rightsquigarrow P'', PA'' \vdash_{pc} \overline{c}$$

Well-Labeledness

- Relative security requires correct annotations during execution
- Annotations are produced by static analysis on the initial program
  - not suitable for preservation
- Introduce a *typing-like* well-labeledness predicate:

$$P, PA \rightsquigarrow P'', PA'' \vdash_{pc} \overline{c}$$

annotated command

Well-Labeledness

- Relative security requires correct annotations during execution
- Annotations are produced by static analysis on the initial program
  - not suitable for preservation
- Introduce a *typing-like* well-labeledness predicate:

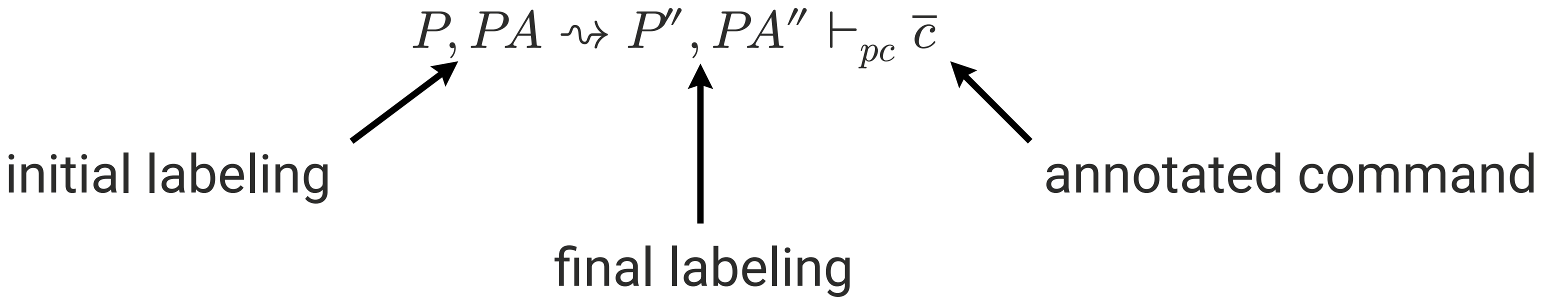
$$P, PA \rightsquigarrow P'', PA'' \vdash_{pc} \overline{c}$$

initial labeling

annotated command

Well-Labeledness

- Relative security requires correct annotations during execution
- Annotations are produced by static analysis on the initial program
  - not suitable for preservation
- Introduce a *typing-like* well-labeledness predicate:



$$P, PA \rightsquigarrow P'', PA'' \vdash_{pc} \overline{c}$$

Well-Labeledness

Lemma

The information-flow analysis produces well-labeled programs.

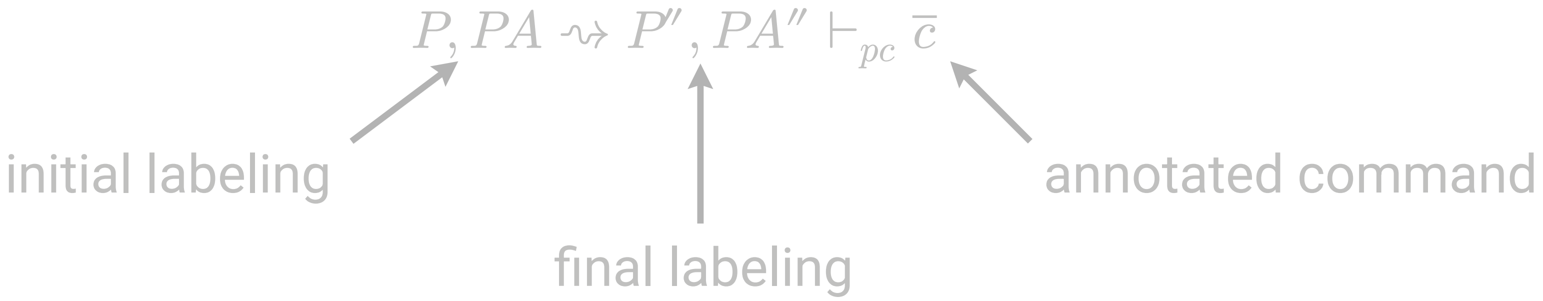
$$\langle\!\langle c \rangle\!\rangle_{pc}^{P,PA} = (\overline{c}, P', PA') \Rightarrow P, PA \rightsquigarrow P', PA' \vdash_{pc} \overline{c}$$

ng execution

e initial program

- not suitable for preservation

- Introduce a *typing-like* well-labeledness predicate:



$$P, PA \rightsquigarrow P'', PA'' \vdash_{pc} \overline{c}$$

initial labeling

final labeling

annotated command

Well-Labeledness

Lemma

The information-flow analysis produces well-labeled programs.

$$\langle\!\langle c \rangle\!\rangle_{pc}^{P,PA} = (\overline{c}, P', PA') \Rightarrow P, PA \rightsquigarrow P', PA' \vdash_{pc} \overline{c}$$

ng execution

e initial program

- not suitable for preser

- Introduce a *typing-like* we

initial labeli

Lemma

Ideal execution preserves well-labeledness.

$$P, PA \rightsquigarrow P'', PA'' \vdash_{pc} \overline{c} \Rightarrow$$

$$\langle \overline{c},\ \rho,\ \mu,\ b,\ pc,\ P,\ PA \rangle \xrightarrow[\mathcal{D}]{o}_{\mathrm{i}} \langle \overline{c'},\ \rho,\ \mu,\ b,\ pc',\ P',\ PA' \rangle \Rightarrow$$

$$P', PA' \rightsquigarrow P'', PA'' \vdash_{pc'} \overline{c'}$$

Well-Labeledness

Lemma

The information-flow analysis produces well-labeled programs.

$$\langle\!\langle c\rangle\!\rangle_{pc}^{P,PA} = (\overline{c}, P', PA') \Rightarrow P, PA \rightsquigarrow P', PA' \vdash_{pc} \overline{c}$$

ng execution

e initial program

- not suitable for preser

- Introduce a *typing-like* we

initial labeli

Lemma

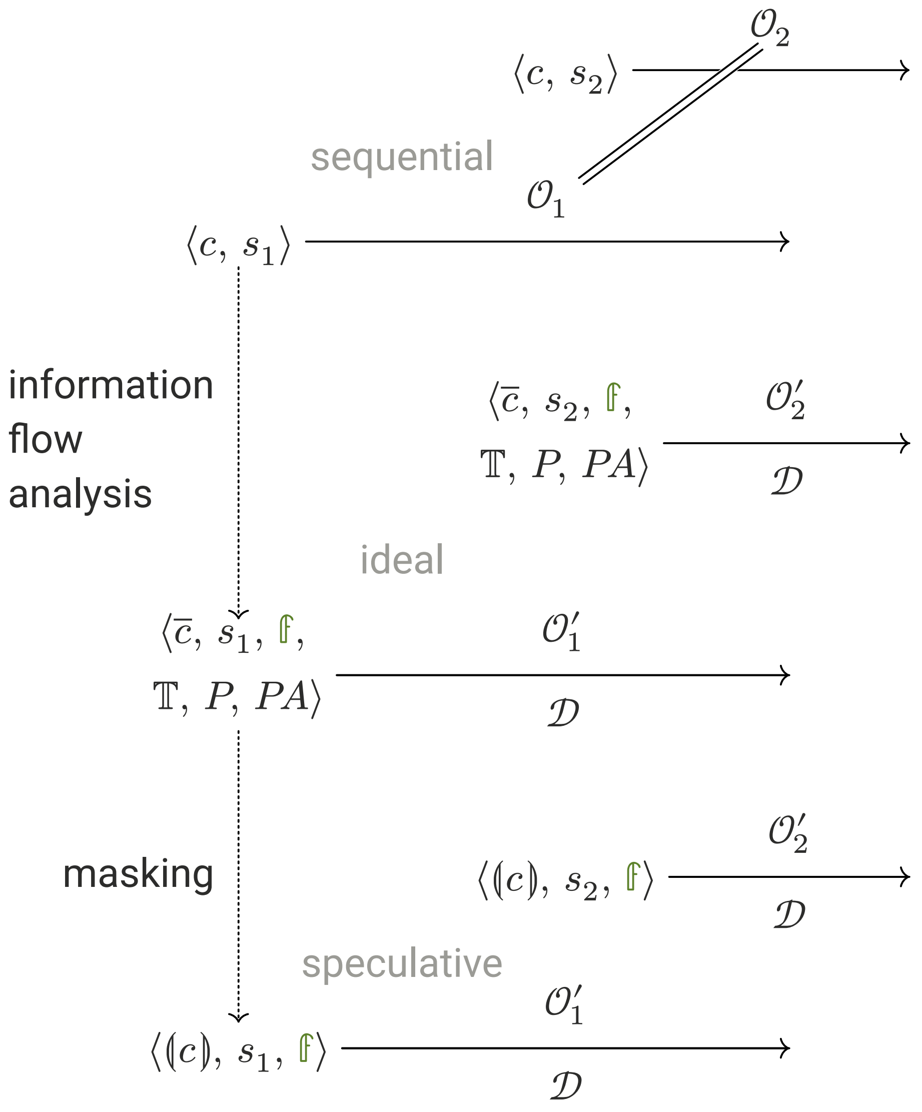
Ideal execution preserves well-labeledness.

$$P, PA \rightsquigarrow P'', PA'' \vdash_{pc} \overline{c} \quad \Rightarrow$$

$$\langle \overline{c}, \rho, \mu, b, pc, P, PA\rangle \xrightarrow[\mathcal{D}]{o}_{\mathrm{i}} \langle \overline{c'}, \rho, \mu, b, pc', P', PA'\rangle \Rightarrow$$

$$P', PA' \rightsquigarrow P'', PA'' \vdash_{pc'} \overline{c'}$$

Proving Relative Security



$\langle c,\ s_2\rangle \xrightarrow{\mathcal{O}_2}$

sequential

$\langle c,\ s_1\rangle \xrightarrow{\mathcal{O}_1}$

information flow analysis

$\langle \overline{c},\ s_2,\ \mathbb{f},\ \mathbb{T},\ P,\ PA\rangle \xrightarrow[\mathcal{D}]{\mathcal{O}'_2}$

ideal

$\langle \overline{c},\ s_1,\ \mathbb{f},\ \mathbb{T},\ P,\ PA\rangle \xrightarrow[\mathcal{D}]{\mathcal{O}'_1}$

masking

$\langle (\!|c|\!),\ s_2,\ \mathbb{f}\rangle \xrightarrow[\mathcal{D}]{\mathcal{O}'_2}$

speculative

$\langle (\!|c|\!),\ s_1,\ \mathbb{f}\rangle \xrightarrow[\mathcal{D}]{\mathcal{O}'_1}$

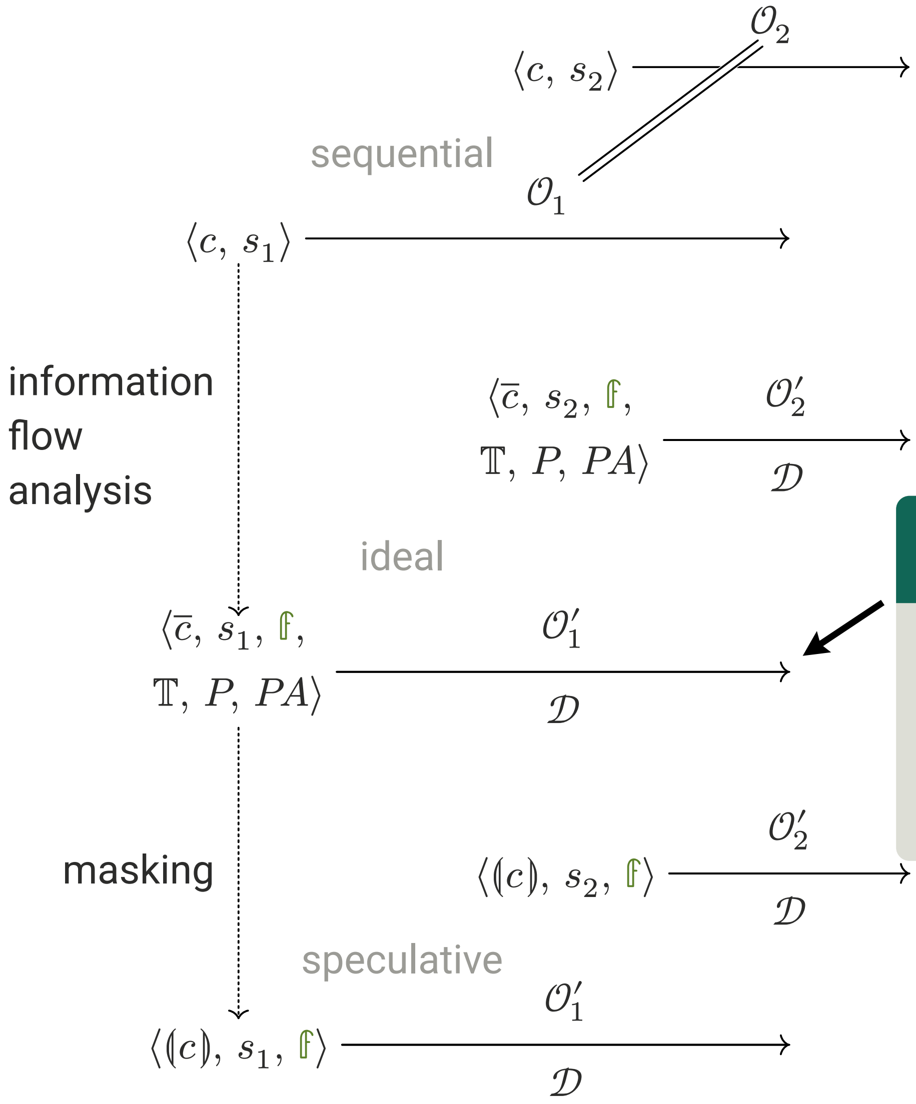
Proving Relative Security

$$\begin{tikzcd}
 & & & \mathcal{O}_2 & \\
 & & \langle c,\, s_2\rangle \arrow[rr] & & {} \\
 & \text{sequential} & \mathcal{O}_1 \arrow[uur, equal] & & \\
\langle c,\, s_1\rangle \arrow[rr] \arrow[ddd, dotted, "\text{information flow analysis}"'] & & {} & & \\
 & & \langle \overline{c},\, s_2,\, \mathbb{f},\, \mathbb{T},\, P,\, PA\rangle \arrow[rr, "\mathcal{O}_2'", "\mathcal{D}"'] & & {} \\
 & \text{ideal} & & & \\
\langle \overline{c},\, s_1,\, \mathbb{f},\, \mathbb{T},\, P,\, PA\rangle \arrow[rr, "\mathcal{O}_1'", "\mathcal{D}"'] \arrow[ddd, dotted, "\text{masking}"'] & & {} & & \\
 & & \langle (\!|c|\!),\, s_2,\, \mathbb{f}\rangle \arrow[rr, "\mathcal{O}_2'", "\mathcal{D}"'] & & {} \\
 & \text{speculative} & & & \\
\langle (\!|c|\!),\, s_1,\, \mathbb{f}\rangle \arrow[rr, "\mathcal{O}_1'", "\mathcal{D}"'] & & {} & &
\end{tikzcd}$$

Unwinding

During misspeculation:

Proving Relative Security



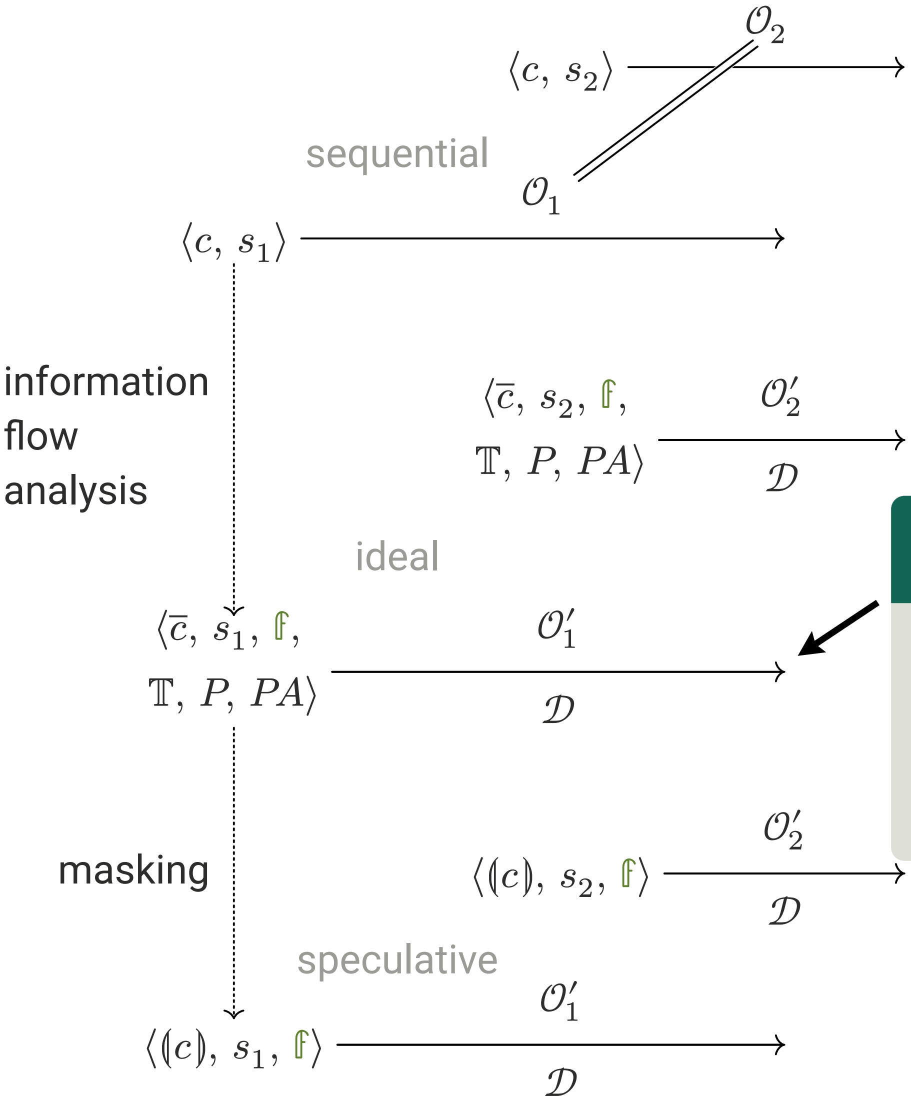
$\mathcal{O}_2$
$\langle c,\ s_2\rangle$
sequential
$\mathcal{O}_1$
$\langle c,\ s_1\rangle$
information flow analysis
$\langle \overline{c},\ s_2,\ \mathbb{f},$
$\mathbb{T},\ P,\ PA\rangle$
$\mathcal{O}'_2$
$\mathcal{D}$
ideal
$\langle \overline{c},\ s_1,\ \mathbb{f},$
$\mathbb{T},\ P,\ PA\rangle$
$\mathcal{O}'_1$
$\mathcal{D}$
masking
$\langle (\!| c |\!),\ s_2,\ \mathbb{f}\rangle$
$\mathcal{O}'_2$
$\mathcal{D}$
speculative
$\langle (\!| c |\!),\ s_1,\ \mathbb{f}\rangle$
$\mathcal{O}'_1$
$\mathcal{D}$

Unwinding

During misspeculation:

- all secret values are masked

Proving Relative Security

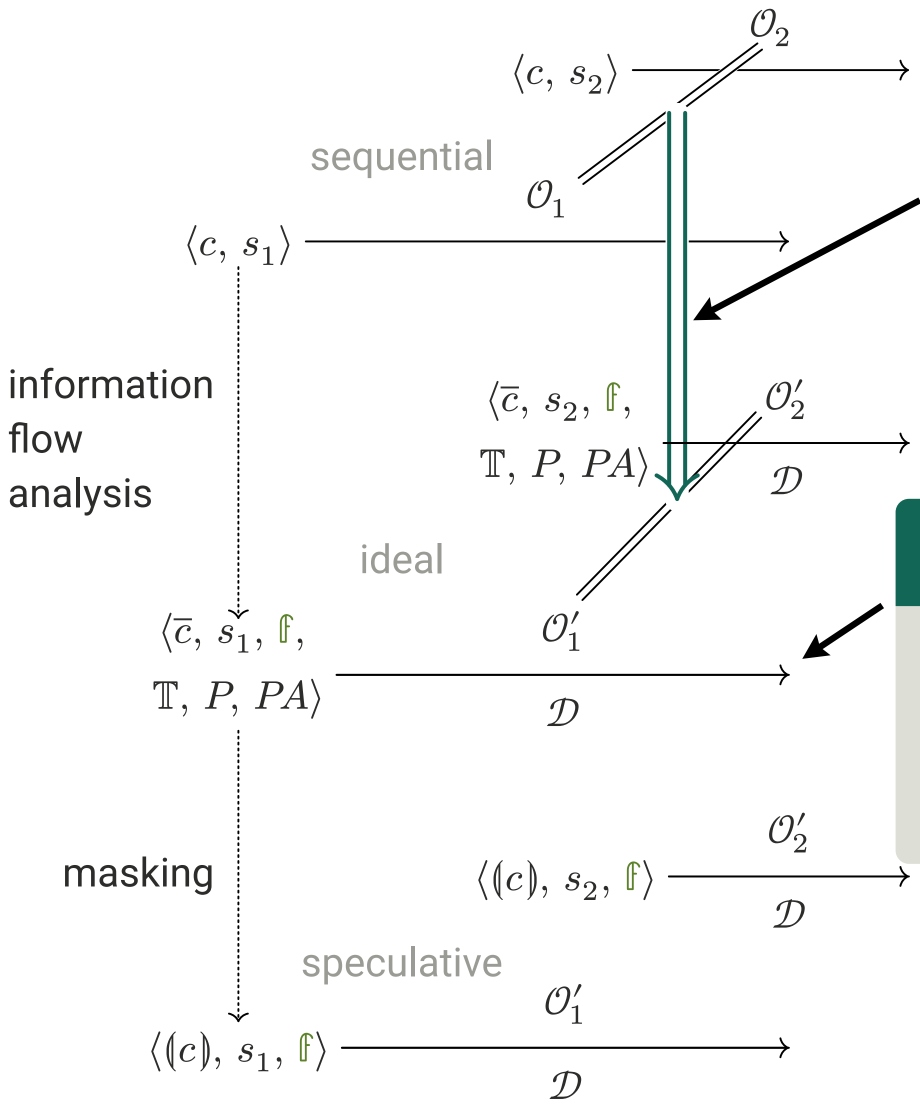

$\mathcal{O}_2$
$\langle c,\ s_2\rangle$
sequential
$\mathcal{O}_1$
$\langle c,\ s_1\rangle$
information flow analysis
$\langle \overline{c},\ s_2,\ \mathbb{f},\ \mathbb{T},\ P,\ PA\rangle$
$\mathcal{O}_2'$
$\mathcal{D}$
ideal
$\langle \overline{c},\ s_1,\ \mathbb{f},\ \mathbb{T},\ P,\ PA\rangle$
$\mathcal{O}_1'$
$\mathcal{D}$
masking
$\langle (\!|c|\!),\ s_2,\ \mathbb{f}\rangle$
$\mathcal{O}_2'$
$\mathcal{D}$
speculative
$\langle (\!|c|\!),\ s_1,\ \mathbb{f}\rangle$
$\mathcal{O}_1'$
$\mathcal{D}$

Unwinding

During misspeculation:

- all secret values are masked
- all public values are equal in both executions

Proving Relative Security


$\mathcal{O}_2$
$\langle c,\ s_2\rangle$
sequential
$\mathcal{O}_1$
$\langle c,\ s_1\rangle$
information flow analysis
$\langle \overline{c},\ s_2,\ \mathbb{f},\ \mathbb{T},\ P,\ PA\rangle$
$\mathcal{O}_2'$
$\mathcal{D}$
ideal
$\mathcal{O}_1'$
$\langle \overline{c},\ s_1,\ \mathbb{f},\ \mathbb{T},\ P,\ PA\rangle$
$\mathcal{D}$
masking
$\mathcal{O}_2'$
$\langle (\!| c |\!),\ s_2,\ \mathbb{f}\rangle$
$\mathcal{D}$
speculative
$\mathcal{O}_1'$
$\langle (\!| c |\!),\ s_1,\ \mathbb{f}\rangle$
$\mathcal{D}$

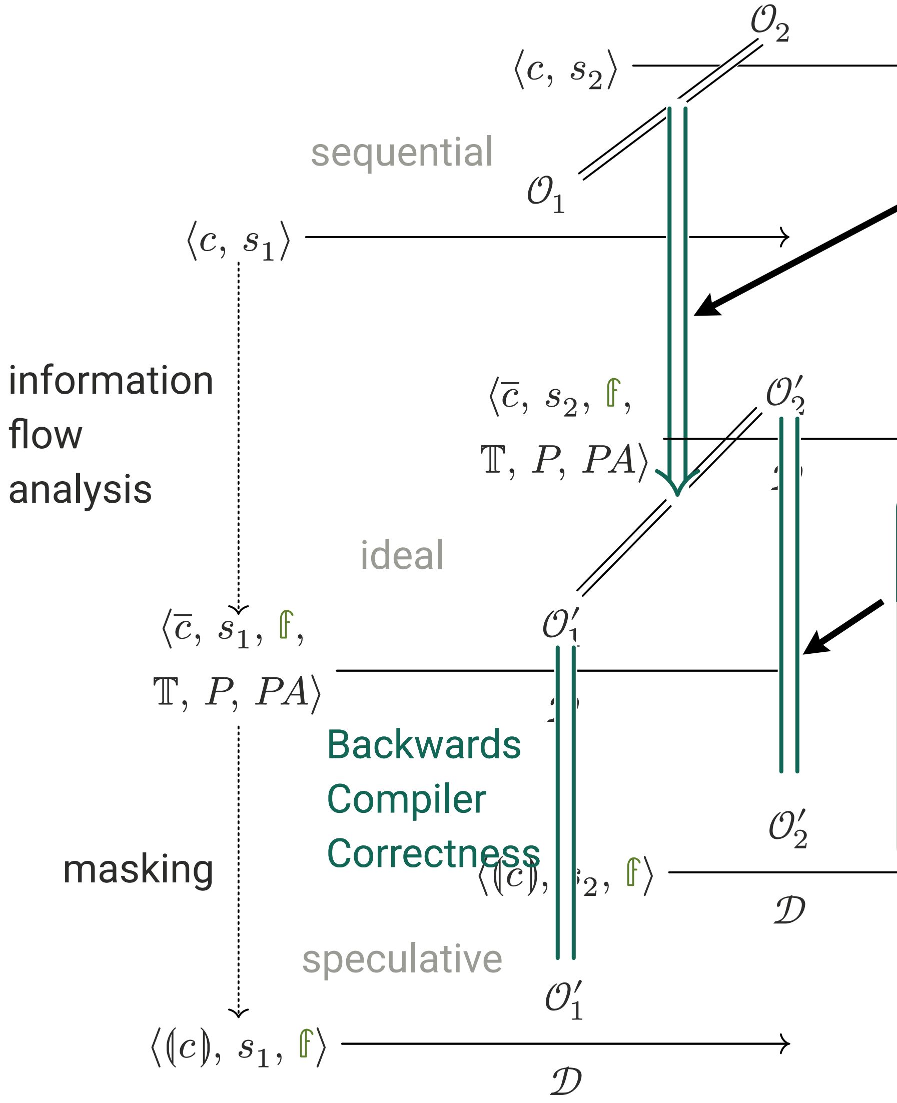
Same behaviour before misspeculation

Unwinding

During misspeculation:

- all secret values are masked
- all public values are equal in both executions

Proving Relative Security


$\mathcal{O}_2$
$\langle c,\ s_2 \rangle$
sequential
$\mathcal{O}_1$
$\langle c,\ s_1 \rangle$
information flow analysis
$\langle \overline{c},\ s_2,\ \mathbb{f},\ \mathbb{T},\ P,\ PA \rangle$
$\mathcal{O}_2'$
ideal
$\langle \overline{c},\ s_1,\ \mathbb{f},\ \mathbb{T},\ P,\ PA \rangle$
$\mathcal{O}_1'$
Backwards Compiler Correctness
$\mathcal{O}_2'$
masking
$\langle (\!| c |\!),\ s_2,\ \mathbb{f} \rangle$
$\mathcal{D}$
speculative
$\mathcal{O}_1'$
$\langle (\!| c |\!),\ s_1,\ \mathbb{f} \rangle$
$\mathcal{D}$

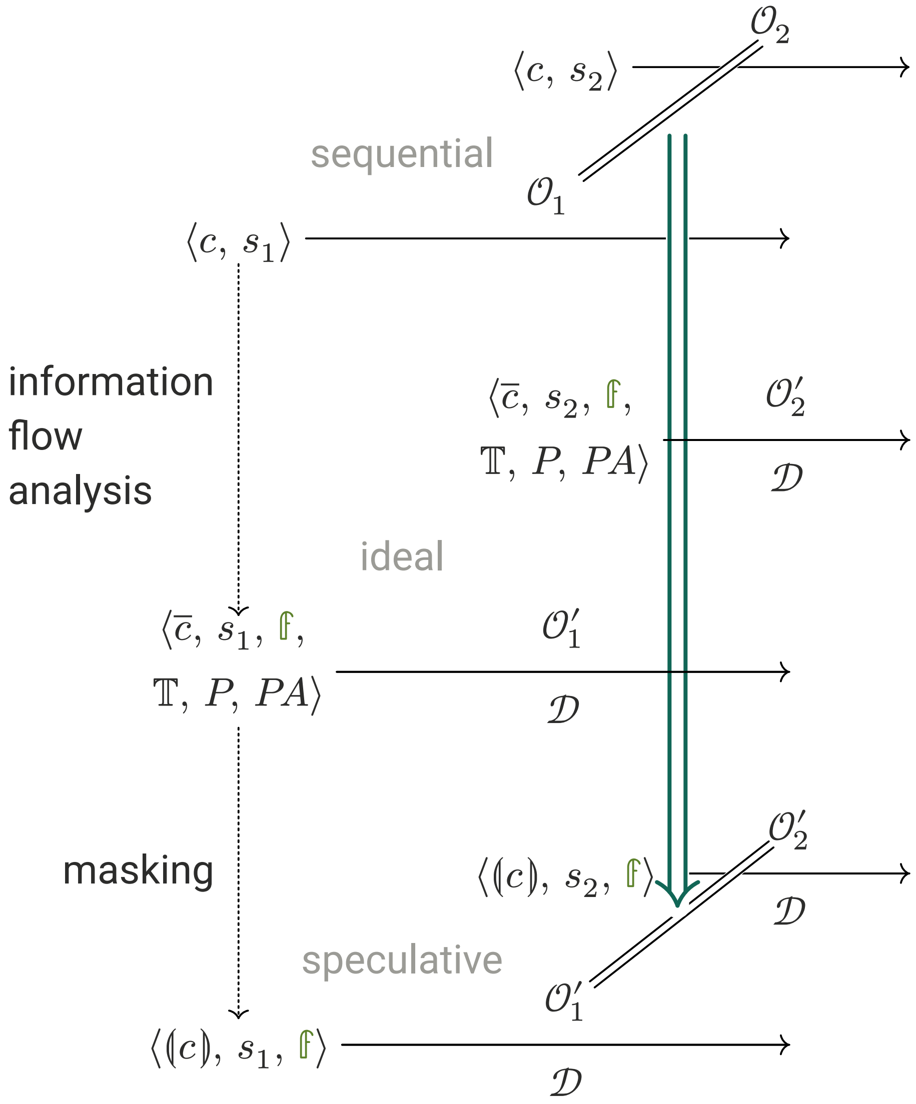
Same behaviour before misspeculation

Unwinding

During misspeculation:

- all secret values are masked
- all public values are equal in both executions

Proving Relative Security


$\mathcal{O}_2$
$\langle c,\ s_2\rangle$
sequential
$\mathcal{O}_1$
$\langle c,\ s_1\rangle$
information flow analysis
$\langle \overline{c},\ s_2,\ \mathbb{f},\ \mathbb{T},\ P,\ PA\rangle$
$\mathcal{O}_2'$
$\mathcal{D}$
ideal
$\langle \overline{c},\ s_1,\ \mathbb{f},\ \mathbb{T},\ P,\ PA\rangle$
$\mathcal{O}_1'$
$\mathcal{D}$
$\mathcal{O}_2'$
masking
$\langle (\!|c|\!),\ s_2,\ \mathbb{f}\rangle$
$\mathcal{D}$
speculative
$\mathcal{O}_1'$
$\langle (\!|c|\!),\ s_1,\ \mathbb{f}\rangle$
$\mathcal{D}$

Conclusions

- Secure: fully mechanized relative security proof in Rocq

Conclusions

- Secure: fully mechanized relative security proof in Rocq
- General: accepts all programs

Conclusions

- Secure: fully mechanized relative security proof in Rocq
- General: accepts all programs
- Efficient: only inserts protections where needed

Conclusions

- Secure: fully mechanized relative security proof in Rocq
- General: accepts all programs
- Efficient: only inserts protections where needed
- Generalization of both Selective SLH (Shivakumar et al. 2023) and Ultimate SLH (Zhang et al. 2023):

Conclusions

- Secure: fully mechanized relative security proof in Rocq
- General: accepts all programs
- Efficient: only inserts protections where needed
- Generalization of both Selective SLH (Shivakumar et al. 2023) and Ultimate SLH (Zhang et al. 2023):
  - coincides with Selective SLH on programs respecting the CCT discipline

Conclusions

- Secure: fully mechanized relative security proof in Rocq
- General: accepts all programs
- Efficient: only inserts protections where needed
- Generalization of both Selective SLH (Shivakumar et al. 2023) and Ultimate SLH (Zhang et al. 2023):
  - coincides with Selective SLH on programs respecting the CCT discipline
  - coincides with Ultimate SLH if everything is labeled as secret

Conclusions

- Secure: fully mechanized relative security proof in Rocq
- General: accepts all programs
- Efficient: only inserts protections where needed
- Generalization of both Selective SLH (Shivakumar et al. 2023) and Ultimate SLH (Zhang et al. 2023):
  - coincides with Selective SLH on programs respecting the CCT discipline
  - coincides with Ultimate SLH if everything is labeled as secret
- Mechanized security proofs for both mitigations obtained as corollaries

Future Work

- Practical implementation and evaluation of FSLH

Future Work

- Practical implementation and evaluation of FSLH
  - at what level should analysis be performed?

Future Work

- Practical implementation and evaluation of FSLH
  - at what level should analysis be performed?
  - preservation by other compilation passes?

Future Work

- Practical implementation and evaluation of FSLH
  - at what level should analysis be performed?
  - preservation by other compilation passes?
- Investigation of other LLVM SLH implementations

Future Work

- Practical implementation and evaluation of FSLH
  - at what level should analysis be performed?
  - preservation by other compilation passes?
- Investigation of other LLVM SLH implementations
  - too much complexity for fully mechanized proofs

Future Work

- Practical implementation and evaluation of FSLH
  - at what level should analysis be performed?
  - preservation by other compilation passes?
- Investigation of other LLVM SLH implementations
  - too much complexity for fully mechanized proofs
  - Property-based testing as a pragmatic compromise

Future Work

- Practical implementation and evaluation of FSLH
  - at what level should analysis be performed?
  - preservation by other compilation passes?
- Investigation of other LLVM SLH implementations
  - too much complexity for fully mechanized proofs
  - Property-based testing as a pragmatic compromise
- Mitigations for other SPECTRE variants

Future Work

- Practical implementation and evaluation of FSLH
  - at what level should analysis be performed?
  - preservation by other compilation passes?
- Investigation of other LLVM SLH implementations
  - too much complexity for fully mechanized proofs
  - Property-based testing as a pragmatic compromise
- Mitigations for other SPECTRE variants
  - e.g. prediction of indirect branch targets (ongoing work) and return addresses

Bibliography

Barthe, Gilles, Sunjay Cauligi, Benjamin Grégoire, et al. 2021. “High-Assurance Cryptography in the Spectre Era.” In “42nd IEEE Symposium on Security and Privacy, SP.” Special issue, *42nd IEEE Symposium on Security and Privacy, SP*, 1884–901. https://doi.org/10.1109/SP40001.2021.00046.

Shivakumar, Basavesh Ammanaghatta, Jack Barnes, Gilles Barthe, et al. 2023. “Spectre Declassified: Reading from the Right Place at the Wrong Time.” In “44th IEEE Symposium on Security and Privacy, SP.” Special issue, *44th IEEE Symposium on Security and Privacy, SP*, 1753–70. https://doi.org/10.1109/SP46215.2023.10179355.

Zhang, Zhiyuan, Gilles Barthe, Chitchanok Chuengsatiansup, Peter Schwabe, and Yuval Yarom. 2023. “Ultimate SLH: Taking Speculative Load Hardening to the Next Level.” In *32nd USENIX Security Symposium*, edited by Joseph A. Calandrino and Carmela Troncoso, *32nd USENIX Security Symposium*. USENIX Association. https://www.usenix.org/conference/usenixsecurity23/presentation/zhang-zhiyuan-slh.

Speculative Semantics

SPEC_ASGN

$$\frac{v = [\![ae]\!]_\rho}{\langle \texttt{X := } ae, \rho, \mu, b\rangle \xrightarrow[\bullet]{\bullet}_s \langle \texttt{skip}, [\texttt{X} \mapsto v]\rho, \mu, b\rangle}$$

SPEC_SEQ_STEP

$$\frac{\langle c_1, \rho, \mu, b\rangle \xrightarrow[d]{o}_s \langle c_1', \rho', \mu', b'\rangle}{\langle c_1 \texttt{;}\ c_2, \rho, \mu, b\rangle \xrightarrow[d]{o}_s \langle c_1' \texttt{;}\ c_2, \rho', \mu', b'\rangle}$$

SPEC_WHILE

$$\frac{c_{while} = \texttt{while } be \texttt{ do } c}{\langle c_{while}, \rho, \mu, b\rangle \xrightarrow[\bullet]{\bullet}_s \langle \texttt{if } be \texttt{ then } c\texttt{;}\ c_{while} \texttt{ else skip}, \rho, \mu, b\rangle}$$

SPEC_SEQ_SKIP

$$\frac{}{\langle \texttt{skip;}\ c, \rho, \mu, b\rangle \xrightarrow[\bullet]{\bullet}_s \langle c, \rho, \mu, b\rangle}$$

SPEC_IF

$$\frac{b' = [\![be]\!]_\rho}{\langle \texttt{if } be \texttt{ then } c_{\mathbb{T}} \texttt{ else } c_{\mathbb{F}}, \rho, \mu, b\rangle \xrightarrow[step]{branch\ b'}_s \langle c_{b'}, \rho, \mu, b\rangle}$$

SPEC_IF_FORCE

$$\frac{b' = [\![be]\!]_\rho}{\langle \texttt{if } be \texttt{ then } c_{\mathbb{T}} \texttt{ else } c_{\mathbb{F}}, \rho, \mu, b\rangle \xrightarrow[force]{branch\ b'}_s \langle c_{\neg b'}, \rho, \mu, \mathbb{T}\rangle}$$

SPEC_READ

$$\frac{i = [\![ie]\!]_\rho \quad v = [\![\texttt{a}[i]]\!]_\mu \quad i < |\texttt{a}|_\mu}{\langle \texttt{X} \leftarrow \texttt{a[}ie\texttt{]}, \rho, \mu, b\rangle \xrightarrow[step]{read\ \texttt{a}\ i}_s \langle \texttt{skip}, [\texttt{X} \mapsto v]\rho, \mu, b\rangle}$$

SPEC_READ_FORCE

$$\frac{i = [\![ie]\!]_\rho \quad v = [\![\texttt{b}[j]]\!]_\mu \quad i \geq |\texttt{a}|_\mu \quad j < |\texttt{b}|_\mu}{\langle \texttt{X} \leftarrow \texttt{a[}ie\texttt{]}, \rho, \mu, \mathbb{T}\rangle \xrightarrow[load\ \texttt{b}\ j]{read\ \texttt{a}\ i}_s \langle \texttt{skip}, [\texttt{X} \mapsto v]\rho, \mu, \mathbb{T}\rangle}$$

SPEC_WRITE

$$\frac{i = [\![ie]\!]_\rho \quad v = [\![ae]\!]_\rho \quad i < |\texttt{a}|_\mu}{\langle \texttt{a[}ie\texttt{]} \leftarrow ae, \rho, \mu, b\rangle \xrightarrow[step]{write\ \texttt{a}\ i}_s \langle \texttt{skip}, \rho, [\texttt{a}[i] \mapsto v]\mu, b\rangle}$$

SPEC_WRITE_FORCE

$$\frac{i = [\![ie]\!]_\rho \quad v = [\![ae]\!]_\rho \quad i \geq |\texttt{a}|_\mu \quad j < |\texttt{b}|_\mu}{\langle \texttt{a[}ie\texttt{]} \leftarrow ae, \rho, \mu, \mathbb{T}\rangle \xrightarrow[store\ \texttt{b}\ j]{write\ \texttt{a}\ i}_s \langle \texttt{skip}, \rho, [\texttt{b}[j] \mapsto v]\mu, \mathbb{T}\rangle}$$

IFC type system

$$\textsc{WT\_Skip}\quad \frac{}{P; PA \vdash_{pc} \texttt{skip}}
\qquad
\textsc{WT\_Asgn}\quad \frac{P(a) = \ell \quad pc \sqcup \ell \sqsubseteq P(\mathsf{X})}{P; PA \vdash_{pc} \mathsf{X}\ \texttt{:=}\ \mathsf{a}}$$

$$\textsc{WT\_Seq}\quad \frac{P; PA \vdash_{pc} c_1 \quad P; PA \vdash_{pc} c_2}{P; PA \vdash_{pc} c_1\texttt{;}\ c_2}
\qquad
\textsc{WT\_If}\quad \frac{P(be) = \ell \quad P; PA \vdash_{pc \sqcup \ell} c_1 \quad P; PA \vdash_{pc \sqcup \ell} c_2}{P; PA \vdash_{pc} \texttt{if}\ be\ \texttt{then}\ c_1\ \texttt{else}\ c_2}$$

$$\textsc{WT\_While}\quad \frac{P(be) = \ell \quad P; PA \vdash_{pc \sqcup \ell} c}{P; PA \vdash_{pc} \texttt{while}\ be\ \texttt{do}\ c}
\qquad
\textsc{WT\_ARead}\quad \frac{P(i) = \ell_i \quad pc \sqcup \ell_i \sqcup PA(\mathsf{a}) \sqsubseteq P(\mathsf{X})}{P; PA \vdash_{pc} \mathsf{X} \leftarrow \mathsf{a}[i]}$$

$$\textsc{WT\_AWrite}\quad \frac{P(i) = \ell_i \quad P(e) = \ell \quad pc \sqcup \ell_i \sqcup \ell \sqsubseteq PA(\mathsf{a})}{P; PA \vdash_{pc} \mathsf{a}[i] \leftarrow e}$$

Ideal Semantics (IFC)

IDEAL_IF

$$\frac{P(be) = \ell \quad b' = (\ell \vee \neg b) \wedge [\![be]\!]_\rho}{\langle \texttt{if}\ be\ \texttt{then}\ c_\mathbb{T}\ \texttt{else}\ c_\mathbb{F}, \rho, \mu, b\rangle \xrightarrow[\mathit{step}]{\mathit{branch}\ b'}_i \langle c_{b'}, \rho, \mu, b\rangle}$$

IDEAL_IF_FORCE

$$\frac{P(be) = \ell \quad b' = (\ell \vee \neg b) \wedge [\![be]\!]_\rho}{\langle \texttt{if}\ be\ \texttt{then}\ c_\mathbb{T}\ \texttt{else}\ c_\mathbb{F}, \rho, \mu, b\rangle \xrightarrow[\mathit{force}]{\mathit{branch}\ b'}_i \langle c_{\neg b'}, \rho, \mu, \mathbb{T}\rangle}$$

IDEAL_READ

$$\frac{\begin{array}{c} P(ie) = \ell_i \quad i = \begin{cases} 0 & \mathit{if}\ (\neg \ell_i \vee P(\mathsf{X})) \wedge b \\ [\![ie]\!]_\rho & \mathit{otherwise} \end{cases} \\ v = [\![\mathsf{a}[i]]\!]_\mu \quad i < |\mathsf{a}|_\mu \end{array}}{\langle \mathsf{X} \leftarrow \mathtt{a}[ie], \rho, \mu, b\rangle \xrightarrow[\mathit{step}]{\mathit{read}\ \mathsf{a}\ i}_i \langle \texttt{skip}, [\mathsf{X} \mapsto v]\rho, \mu, b\rangle}$$

IDEAL_READ_FORCE

$$\frac{\begin{array}{c} P(ie) \quad \neg P(\mathsf{X}) \quad i = [\![ie]\!]_\rho \\ v = [\![\mathsf{b}[j]]\!]_\mu \quad i \geq |\mathsf{a}|_\mu \quad j < |\mathsf{b}|_\mu \end{array}}{\langle \mathsf{X} \leftarrow \mathtt{a}[ie], \rho, \mu, \mathbb{T}\rangle \xrightarrow[\mathit{load}\ \mathsf{b}\ j]{\mathit{read}\ \mathsf{a}\ i}_i \langle \texttt{skip}, [\mathsf{X} \mapsto v]\rho, \mu, \mathbb{T}\rangle}$$

IDEAL_WRITE

$$\frac{\begin{array}{c} i = \begin{cases} 0 & \mathit{if}\ (\neg \ell_i \vee \neg \ell) \wedge b \\ [\![ie]\!]_\rho & \mathit{otherwise} \end{cases} \\ P(ie) = \ell_i \quad P(ae) = \ell \quad v = [\![ae]\!]_\rho \quad i < |\mathsf{a}|_\mu \end{array}}{\langle \mathtt{a}[ie] \leftarrow ae, \rho, \mu, b\rangle \xrightarrow[\mathit{step}]{\mathit{write}\ \mathsf{a}\ i}_i \langle \texttt{skip}, \rho, [\mathsf{a}[i] \mapsto v]\mu, b\rangle}$$

IDEAL_WRITE_FORCE

$$\frac{\begin{array}{c} P(ie) \quad P(ae) \\ i = [\![ie]\!]_\rho \quad v = [\![ae]\!]_\rho \quad i \geq |\mathsf{a}|_\mu \quad j < |\mathsf{b}|_\mu \end{array}}{\langle \mathtt{a}[ie] \leftarrow ae, \rho, \mu, \mathbb{T}\rangle \xrightarrow[\mathit{store}\ \mathsf{b}\ j]{\mathit{write}\ \mathsf{a}\ i}_i \langle \texttt{skip}, \rho, [\mathsf{b}[j] \mapsto v]\mu, \mathbb{T}\rangle}$$

Ideal Semantics (annotated programs)

$$\textsc{Ideal\_If}\ \frac{\cancel{P(be)=\ell} \quad b' = (\ell \vee \neg b) \wedge [\![be]\!]_\rho}{\langle \texttt{if}\ be_{@\ell}\ \texttt{then}\ \overline{c_{\mathbb{T}}}\ \texttt{else}\ \overline{c_{\mathbb{F}}}, \rho, \mu, b, pc, P, PA\rangle \xrightarrow[\text{step}]{\text{branch } b'}_{i} \langle \texttt{branch}\ pc\ \overline{c_{b'}}, \rho, \mu, b, pc \sqcup \ell, P, PA\rangle}$$

$$\textsc{Ideal\_If\_Force}\ \frac{\cancel{P(be)=\ell} \quad b' = (\ell \vee \neg b) \wedge [\![be]\!]_\rho}{\langle \texttt{if}\ be_{@\ell}\ \texttt{then}\ \overline{c_{\mathbb{T}}}\ \texttt{else}\ \overline{c_{\mathbb{F}}}, \rho, \mu, b, pc, P, PA\rangle \xrightarrow[\text{force}]{\text{branch } b'}_{i} \langle \texttt{branch}\ pc\ \overline{c_{\neg b'}}, \rho, \mu, \mathbb{T}, pc \sqcup \ell, P, PA\rangle}$$

$$\textsc{Ideal\_Read}\ \frac{\cancel{P(ie)=\ell_i} \quad i = \begin{cases} 0 & \text{if } \neg\ell_i \wedge b \\ [\![ie]\!]_\rho & \text{otherwise} \end{cases} \quad v = \begin{cases} 0 & \text{if } \ell_{\mathsf{X}} \wedge \ell_i \wedge b \\ [\![\mathsf{a}[i]]\!]_\mu & \text{otherwise} \end{cases} \quad i < |\mathsf{a}|_\mu}{\langle \mathsf{X}_{@\ell_{\mathsf{X}}} \leftarrow \texttt{a}[ie_{@\ell_i}], \rho, \mu, b, pc, P, PA\rangle \xrightarrow[\text{step}]{\text{read } \mathsf{a}\ i}_{i} \langle \texttt{skip}, [\mathsf{X} \mapsto v]\rho, \mu, b, pc, [\mathsf{X} \mapsto \ell_{\mathsf{X}}]P, PA\rangle}$$

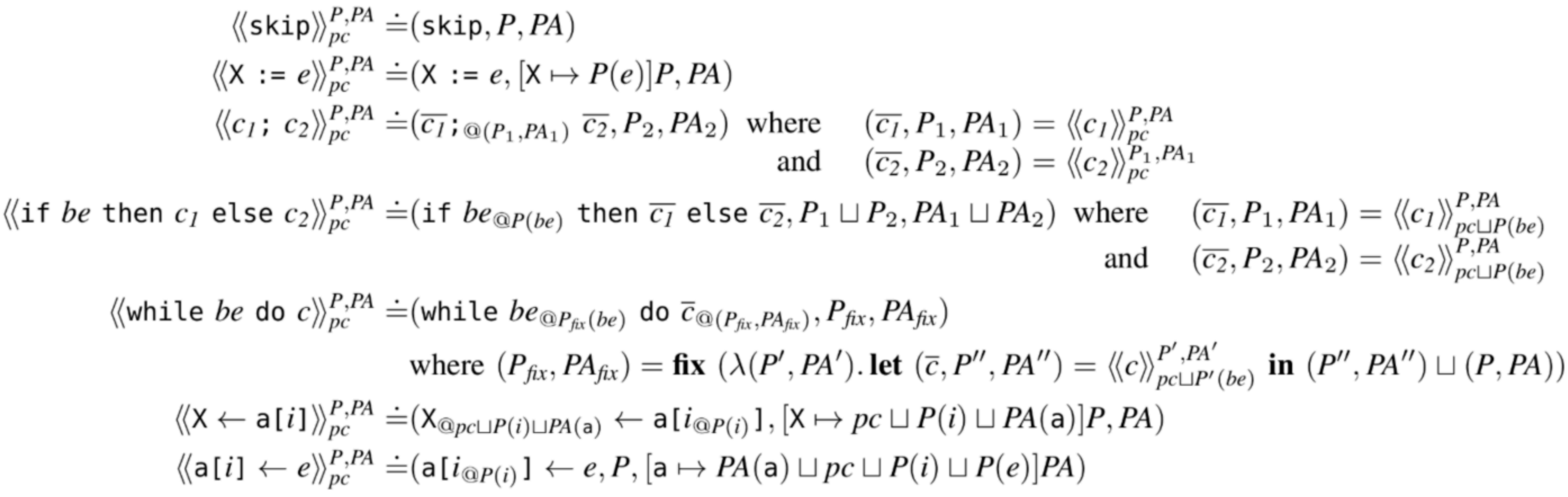
$$\textsc{Ideal\_Read\_Force}\ \frac{\cancel{P(ie)} \quad i = [\![ie]\!]_\rho \quad v = \begin{cases} 0 & \text{if } \ell_{\mathsf{X}} \\ [\![\mathsf{b}[j]]\!]_\mu & \text{otherwise} \end{cases} \quad i \geq |\mathsf{a}|_\mu \quad j < |\mathsf{b}|_\mu}{\langle \mathsf{X}_{@\ell_{\mathsf{X}}} \leftarrow \texttt{a}[ie_{@\mathbb{T}}], \rho, \mu, \mathbb{T}, pc, P, PA\rangle \xrightarrow[\text{step}]{\text{read } \mathsf{a}\ i}_{i} \langle \texttt{skip}, [\mathsf{X} \mapsto v]\rho, \mu, \mathbb{T}, pc, [\mathsf{X} \mapsto \ell_{\mathsf{X}}]P, PA\rangle}$$

$$\textsc{Ideal\_Seq\_Skip}\ \frac{\textit{terminal}\ \overline{c_1}}{\langle \overline{c_1}\,\texttt{;}_{@(P',PA')}\ \overline{c_2}, \rho, \mu, b, pc, P, PA\rangle \xrightarrow[\bullet]{\bullet}_{i} \langle \overline{c_2}, \rho, \mu, b, pc\text{-}after\ \overline{c_1}\ pc, P, PA\rangle}$$

$$\textsc{Ideal\_Branch}\ \frac{\langle \overline{c}, \rho, \mu, b, pc, P, PA\rangle \xrightarrow[d]{o}_{i} \langle \overline{c'}, \rho', \mu', b', pc', P', PA'\rangle}{\langle \texttt{branch}\ \ell\ \overline{c}, \ldots\rangle \xrightarrow[d]{o}_{i} \langle \texttt{branch}\ \ell\ \overline{c'}, \ldots\rangle}$$

Fig. 12: Ideal semantics for FvSLH$^\forall$ (selected rules)

Flow-sensitive static analysis



$$
\begin{aligned}
\langle\!\langle \texttt{skip} \rangle\!\rangle_{pc}^{P,PA} &\doteq (\texttt{skip}, P, PA) \\
\langle\!\langle \texttt{X := } e \rangle\!\rangle_{pc}^{P,PA} &\doteq (\texttt{X := } e, [\texttt{X} \mapsto P(e)]P, PA) \\
\langle\!\langle c_1\texttt{;}\ c_2 \rangle\!\rangle_{pc}^{P,PA} &\doteq (\overline{c_1}\,\texttt{;}_{@(P_1,PA_1)}\ \overline{c_2}, P_2, PA_2) \ \text{ where } \ (\overline{c_1}, P_1, PA_1) = \langle\!\langle c_1 \rangle\!\rangle_{pc}^{P,PA} \\
&\qquad\qquad\qquad\qquad\qquad\ \ \text{and } \ (\overline{c_2}, P_2, PA_2) = \langle\!\langle c_2 \rangle\!\rangle_{pc}^{P_1,PA_1} \\
\langle\!\langle \texttt{if } be \texttt{ then } c_1 \texttt{ else } c_2 \rangle\!\rangle_{pc}^{P,PA} &\doteq (\texttt{if } be_{@P(be)} \texttt{ then } \overline{c_1} \texttt{ else } \overline{c_2}, P_1 \sqcup P_2, PA_1 \sqcup PA_2) \ \text{ where } \ (\overline{c_1}, P_1, PA_1) = \langle\!\langle c_1 \rangle\!\rangle_{pc \sqcup P(be)}^{P,PA} \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{and } \ (\overline{c_2}, P_2, PA_2) = \langle\!\langle c_2 \rangle\!\rangle_{pc \sqcup P(be)}^{P,PA} \\
\langle\!\langle \texttt{while } be \texttt{ do } c \rangle\!\rangle_{pc}^{P,PA} &\doteq (\texttt{while } be_{@P_{\mathit{fix}}(be)} \texttt{ do } \overline{c}_{@(P_{\mathit{fix}},PA_{\mathit{fix}})}, P_{\mathit{fix}}, PA_{\mathit{fix}}) \\
&\quad \text{where } (P_{\mathit{fix}}, PA_{\mathit{fix}}) = \mathbf{fix}\ (\lambda(P', PA').\, \mathbf{let}\ (\overline{c}, P'', PA'') = \langle\!\langle c \rangle\!\rangle_{pc \sqcup P'(be)}^{P',PA'}\ \mathbf{in}\ (P'', PA'') \sqcup (P, PA)) \\
\langle\!\langle \texttt{X} \leftarrow \texttt{a[}i\texttt{]} \rangle\!\rangle_{pc}^{P,PA} &\doteq (\texttt{X}_{@pc \sqcup P(i) \sqcup PA(\texttt{a})} \leftarrow \texttt{a[}i_{@P(i)}\texttt{]}, [\texttt{X} \mapsto pc \sqcup P(i) \sqcup PA(\texttt{a})]P, PA) \\
\langle\!\langle \texttt{a[}i\texttt{]} \leftarrow e \rangle\!\rangle_{pc}^{P,PA} &\doteq (\texttt{a[}i_{@P(i)}\texttt{]} \leftarrow e, P, [\texttt{a} \mapsto PA(\texttt{a}) \sqcup pc \sqcup P(i) \sqcup P(e)]PA)
\end{aligned}
$$

Fig. 11: Flow-sensitive IFC analysis generating annotated commands

Well-Labeledness

$$\text{WL\_SKIP}\ \frac{(P_1, PA_1) \sqsubseteq (P_2, PA_2)}{P_1, PA_1 \rightsquigarrow P_2, PA_2 \vdash_{pc} \texttt{skip}}$$

$$\text{WL\_ASGN}\ \frac{([\mathsf{X} \mapsto P_1(e)]P_1, PA_1) \sqsubseteq P_2, PA_2}{P_1, PA_1 \rightsquigarrow P_2, PA_2 \vdash_{pc} (\mathsf{X}\ \texttt{:=}\ e)}$$

$$\text{WL\_SEQ}\ \frac{\begin{array}{c} \textit{branch-free}\,\overline{c_2} \qquad P_1, PA_1 \rightsquigarrow P', PA' \vdash_{pc} \overline{c_1} \\ P', PA' \rightsquigarrow P_2, PA_2 \vdash_{(\textit{pc-after}\,\overline{c_1}\,pc)} \overline{c_2} \end{array}}{P_1, PA_1 \rightsquigarrow P_2, PA_2 \vdash_{pc} (\overline{c_1}\,\texttt{;}_{@(P',PA')}\ \overline{c_2})}$$

$$\text{WL\_IF}\ \frac{\begin{array}{c} P_1(be) \sqsubseteq \ell_{be} \qquad \textit{branch-free}\,\overline{c_1} \qquad \textit{branch-free}\,\overline{c_2} \\ P_1, PA_1 \rightsquigarrow P_2, PA_2 \vdash_{pc \sqcup \ell_{be}} \overline{c_1} \quad P_1, PA_1 \rightsquigarrow P_2, PA_2 \vdash_{pc \sqcup \ell_{be}} \overline{c_2} \end{array}}{P_1, PA_1 \rightsquigarrow P_2, PA_2 \vdash_{pc} (\texttt{if}\ be_{@\ell_{be}}\ \texttt{then}\ \overline{c_1}\ \texttt{else}\ \overline{c_2})}$$

$$\text{WL\_WHILE}\ \frac{\begin{array}{c} P_1(be) \sqsubseteq \ell_{be} \qquad \textit{branch-free}\,\overline{c} \\ (P_1, PA_1) \sqsubseteq (P', PA') \qquad (P', PA') \sqsubseteq (P_2, PA_2) \\ P', PA' \rightsquigarrow P', PA' \vdash_{pc \sqcup \ell_{be}} \overline{c_1} \end{array}}{P_1, PA_1 \rightsquigarrow P_2, PA_2 \vdash_{pc} (\texttt{while}\ be_{@\ell_{be}}\ \texttt{do}\ \overline{c}_{@(P',PA')})}$$

$$\text{WL\_AREAD}\ \frac{\begin{array}{c} P_1(e) \sqsubseteq \ell_i \quad pc \sqsubseteq \ell_{\mathsf{X}} \\ \ell_i \sqsubseteq \ell_{\mathsf{X}} \quad PA_1(\mathsf{a}) \sqsubseteq \ell_{\mathsf{X}} \quad ([\mathsf{X} \mapsto \ell_{\mathsf{X}}]P_1, PA_1) \sqsubseteq (P_2, PA_2) \end{array}}{P_1, PA_1 \rightsquigarrow P_2, PA_2 \vdash_{pc} (\mathsf{X}_{@\ell_{\mathsf{X}}} \leftarrow \mathtt{a[e}_{@\ell_i}\mathtt{]})}$$

$$\text{WL\_AWRITE}\ \frac{\begin{array}{c} P_1(i) \sqsubseteq \ell_i \\ (P_1, [\mathsf{a} \mapsto PA_1(\mathsf{a}) \sqcup pc \sqcup \ell_i \sqcup P_1(e)]PA_1) \sqsubseteq (P_2, PA_2) \end{array}}{P_1, PA_1 \rightsquigarrow P_2, PA_2 \vdash_{pc} (\mathtt{a[}i_{@\ell_i}\mathtt{]} \leftarrow e)}$$

$$\text{WL\_BRANCH}\ \frac{P_1, PA_1 \rightsquigarrow P_2, PA_2 \vdash_{pc} \overline{c}}{P_1, PA_1 \rightsquigarrow P_2, PA_2 \vdash_{pc} (\texttt{branch}\ \ell\ \overline{c})}$$

iSLH must protect stores

```
if i < secrets_size then
  secrets[i] <- key;
  x <- a[0];
  if x then...
```

- out-of-bounds `i` could write to `a[0]`
- read from public array `a` is unprotected
  - reads speculatively stored secret