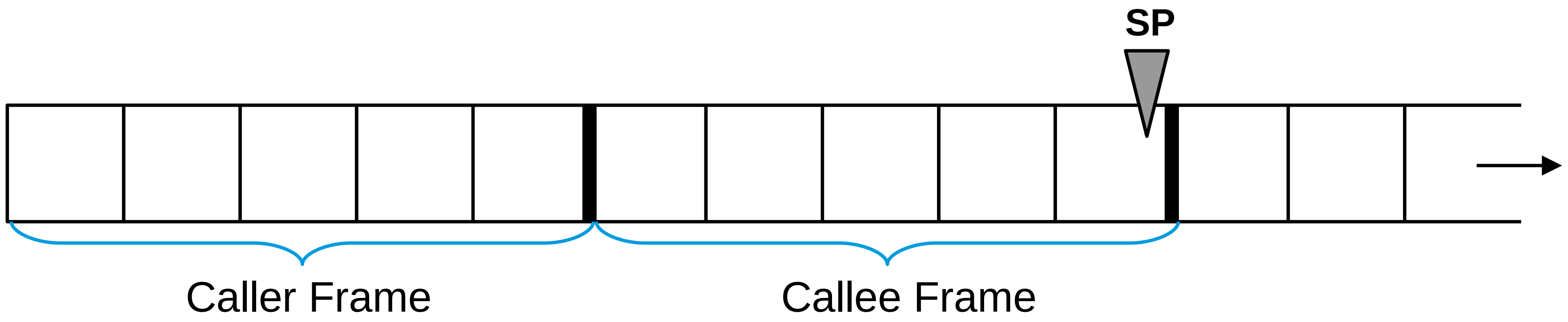
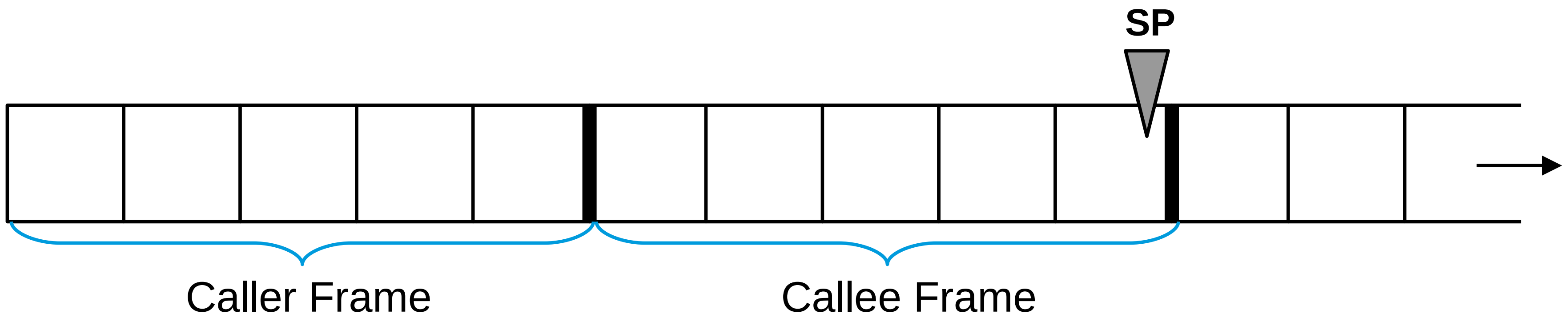
Formalizing Stack Safety as a Security Property

Sean Noble Anderson	Portland State University
Roberto Blanco	Max Planck Institute for Security and Privacy
Leonidas Lampropoulos	University of Maryland, College Park
Benjamin C. Pierce	University of Pennsylvania
Andrew Tolmach	Portland State University

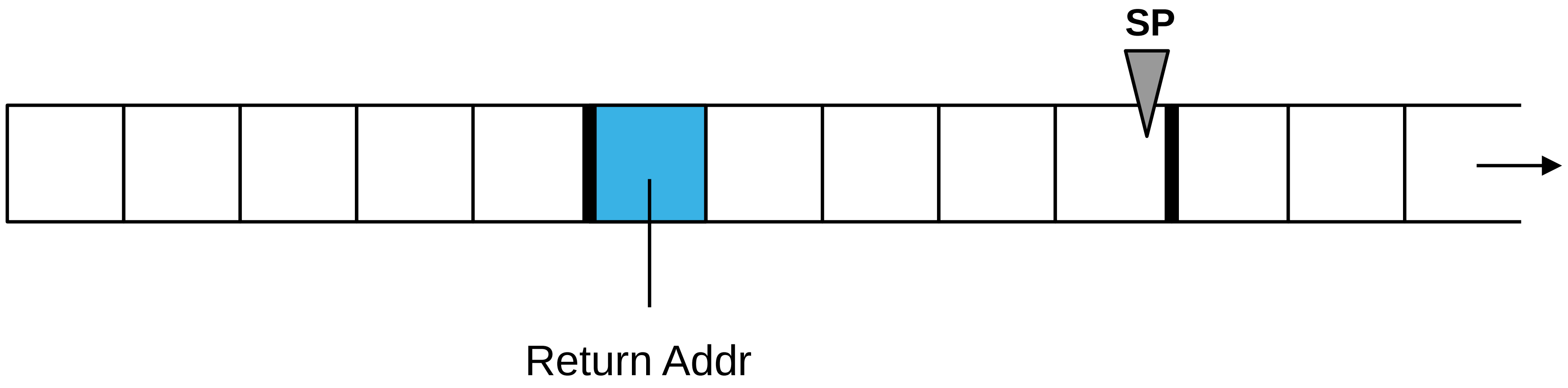
What Needs Protection on the Stack?



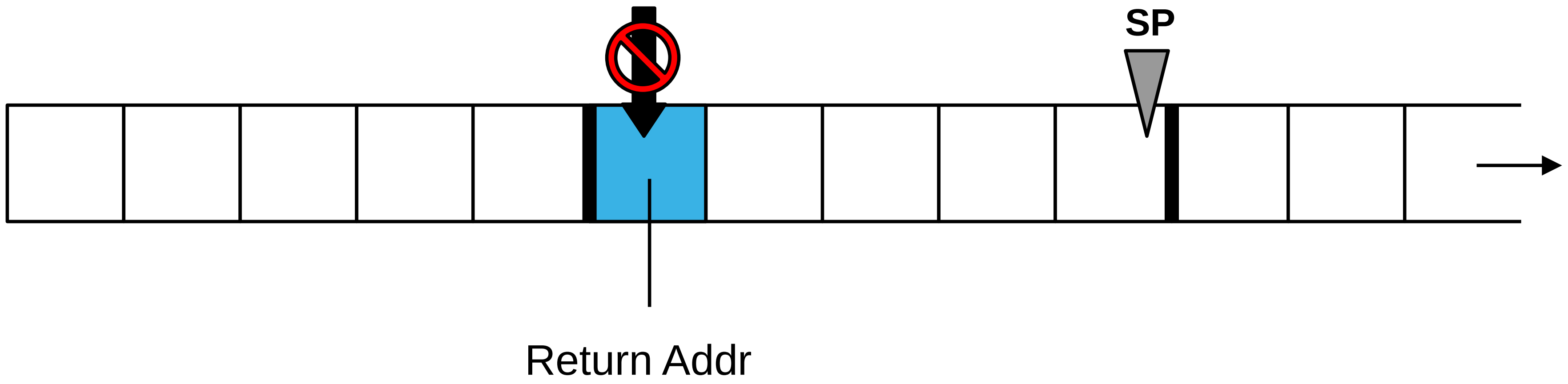
What Needs Protection on the Stack?



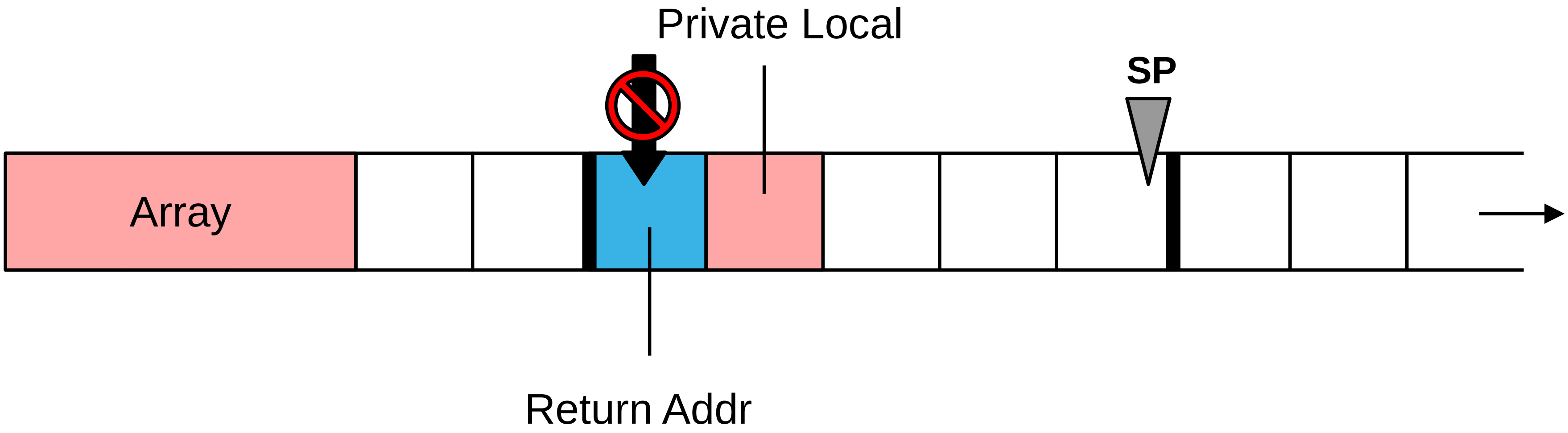
What Needs Protection on the Stack?



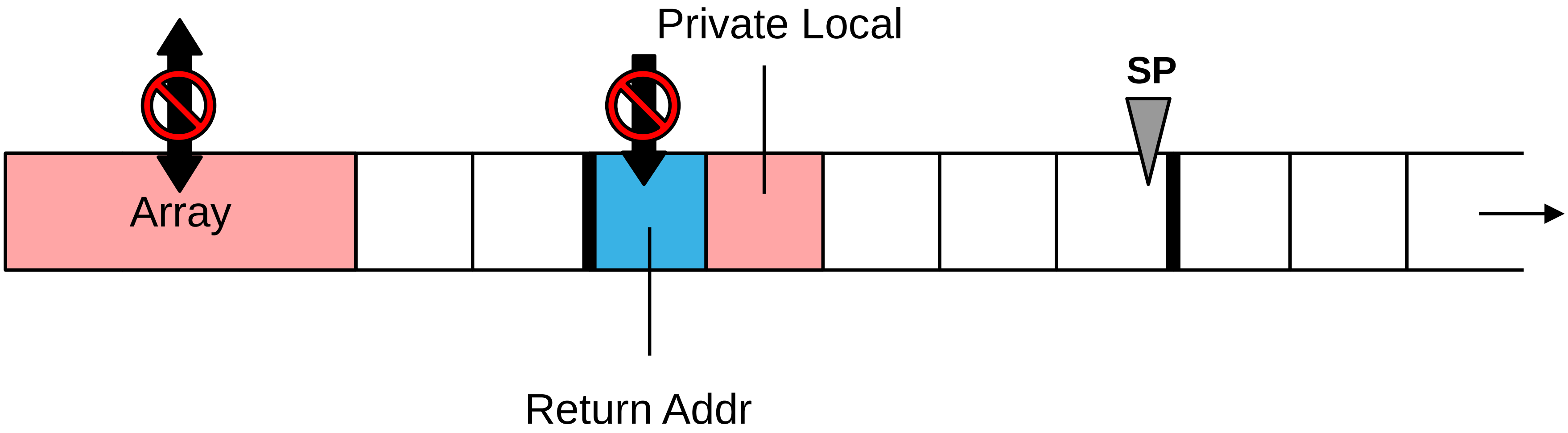
What Needs Protection on the Stack?



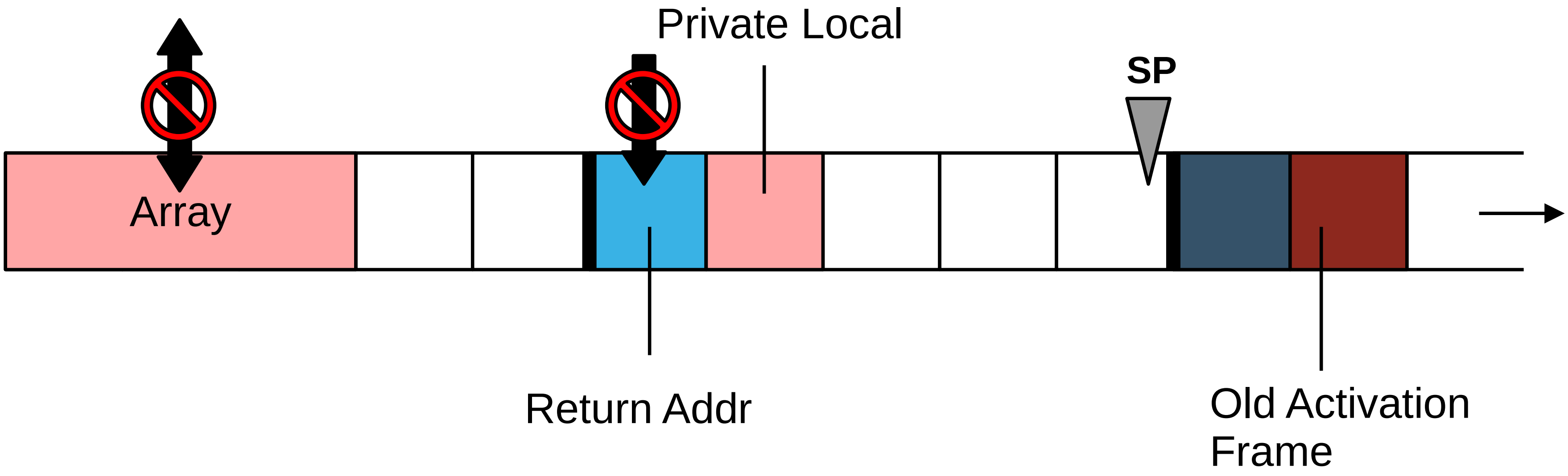
What Needs Protection on the Stack?



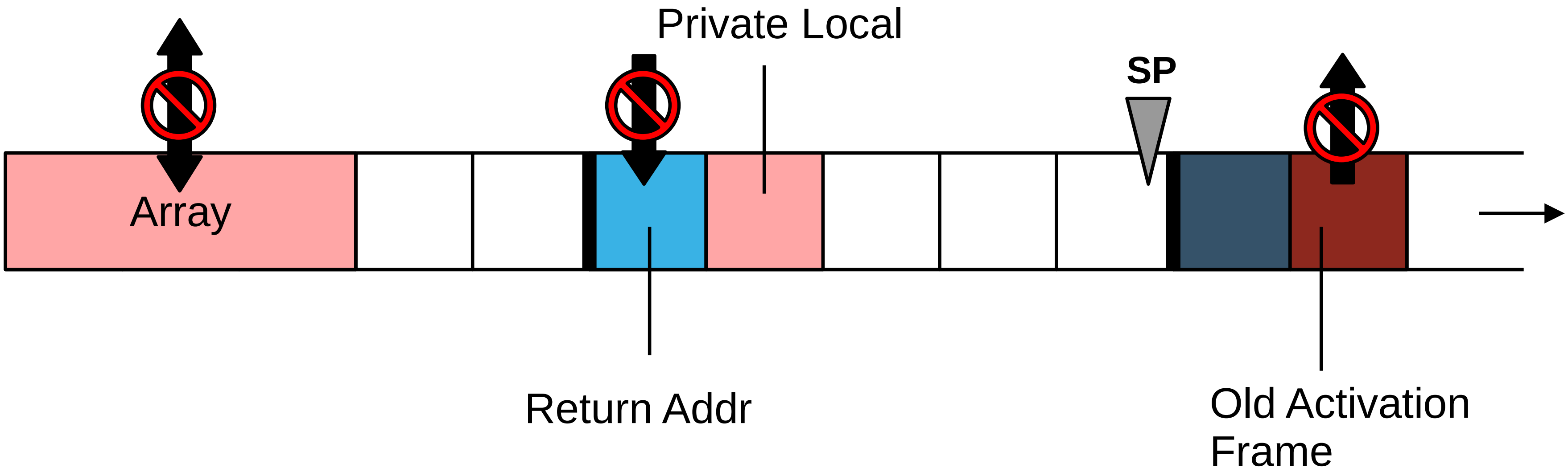
What Needs Protection on the Stack?



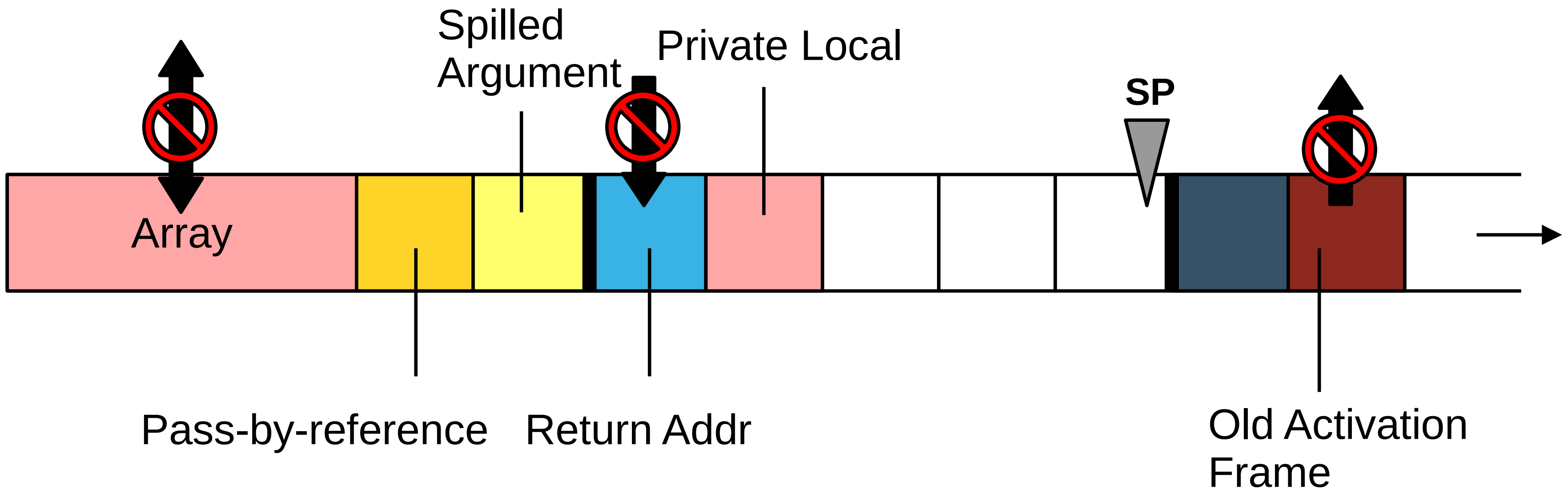
What Needs Protection on the Stack?



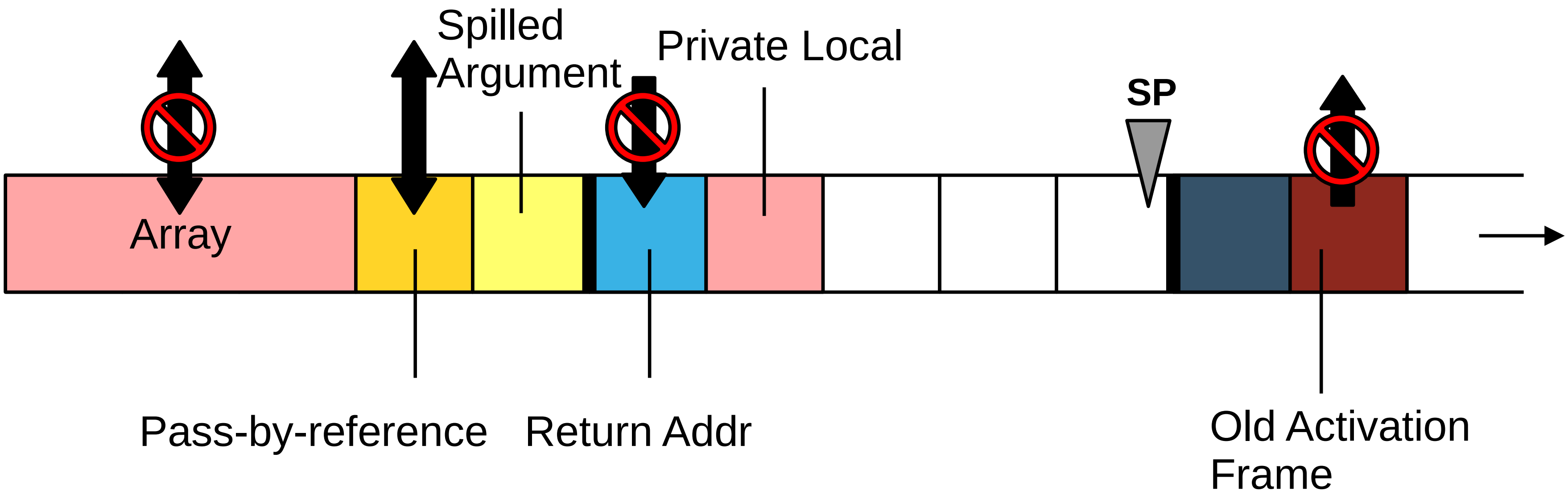
What Needs Protection on the Stack?



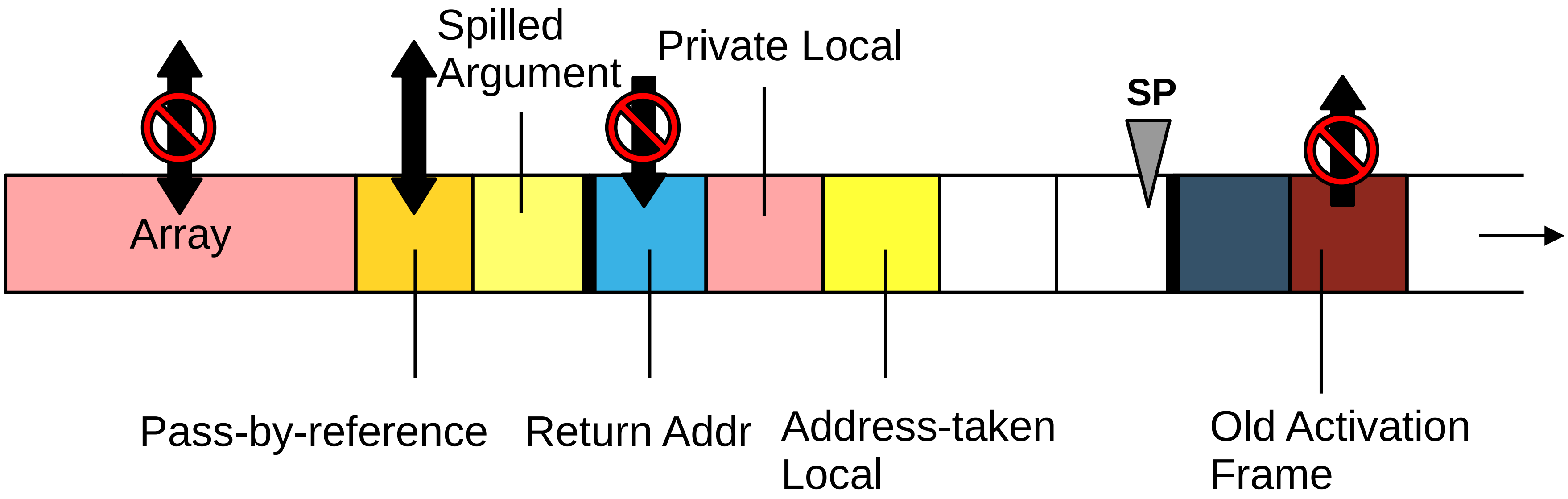
What Needs Protection on the Stack?



What Needs Protection on the Stack?

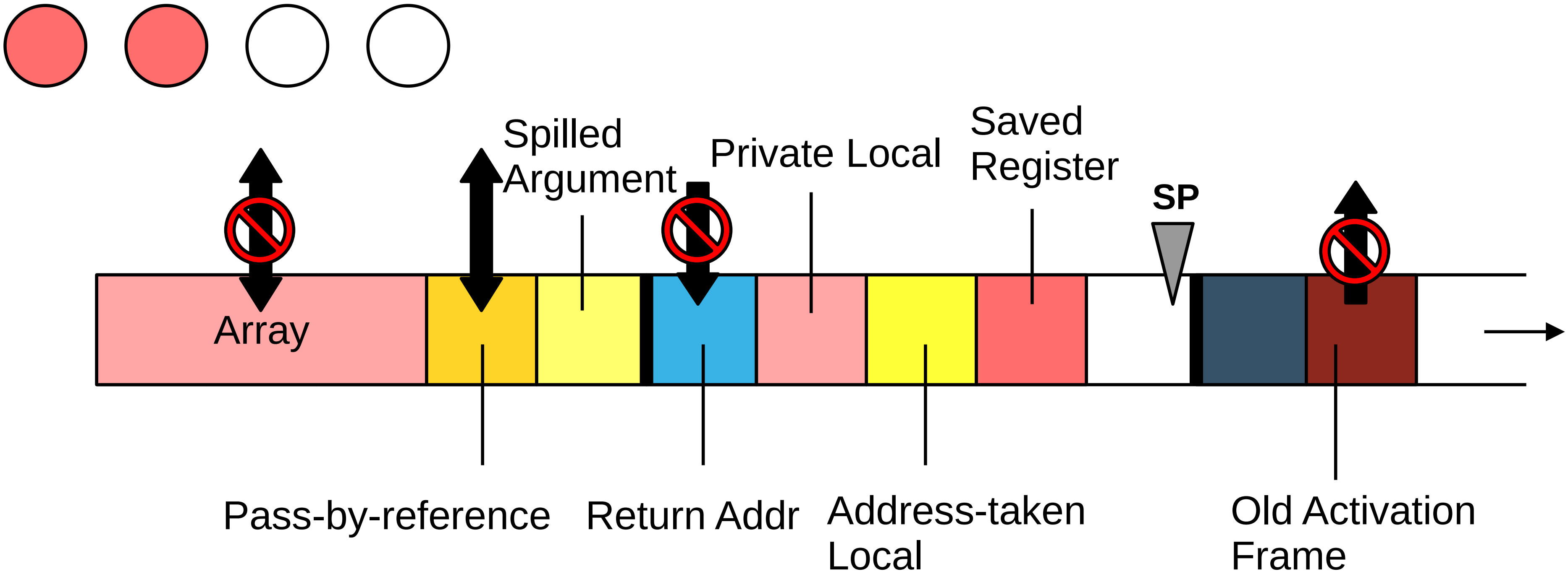


What Needs Protection on the Stack?



What Needs Protection on the Stack?

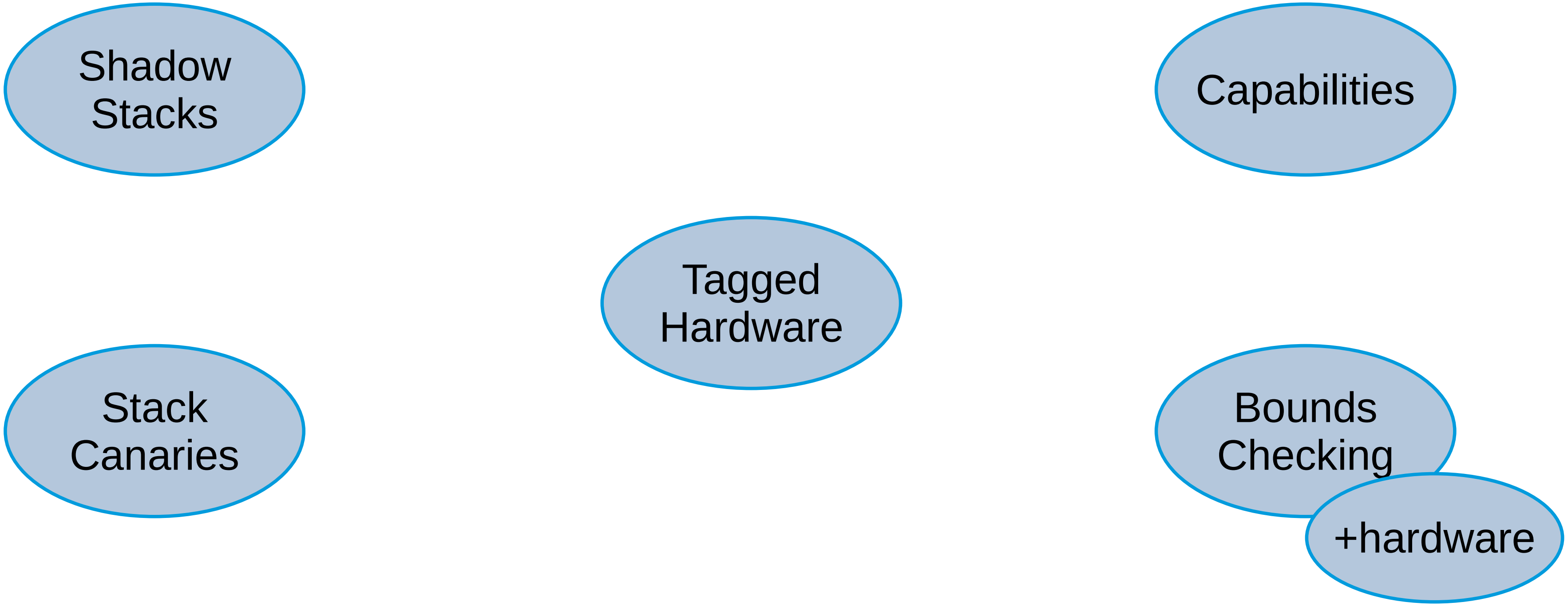
… And registers!



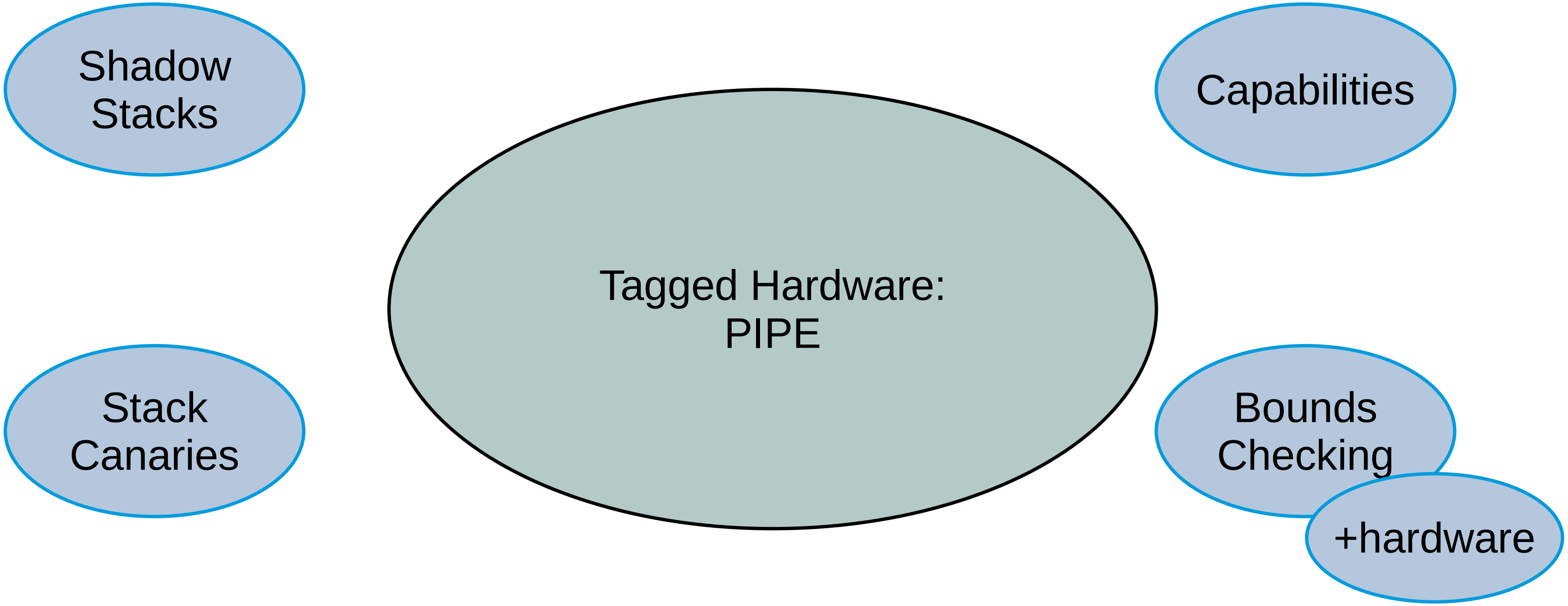
Protection Mechanisms

Protection Mechanisms

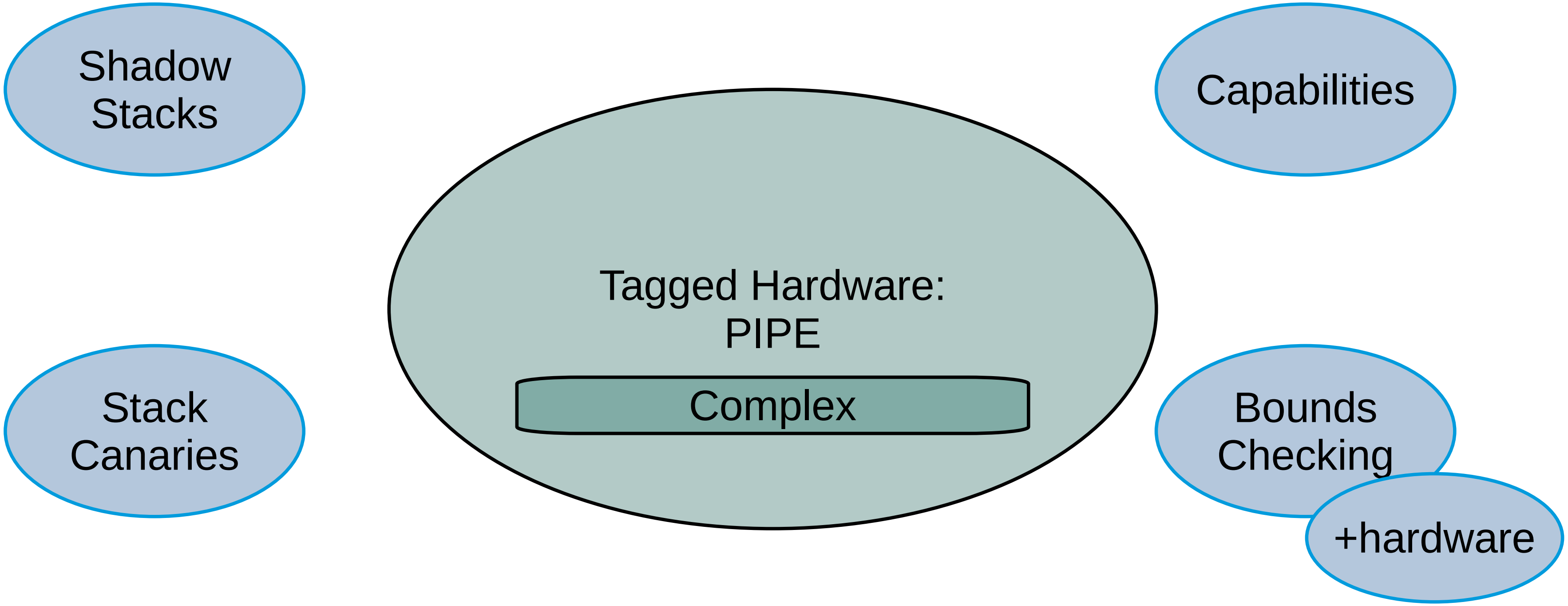
Protection Mechanisms



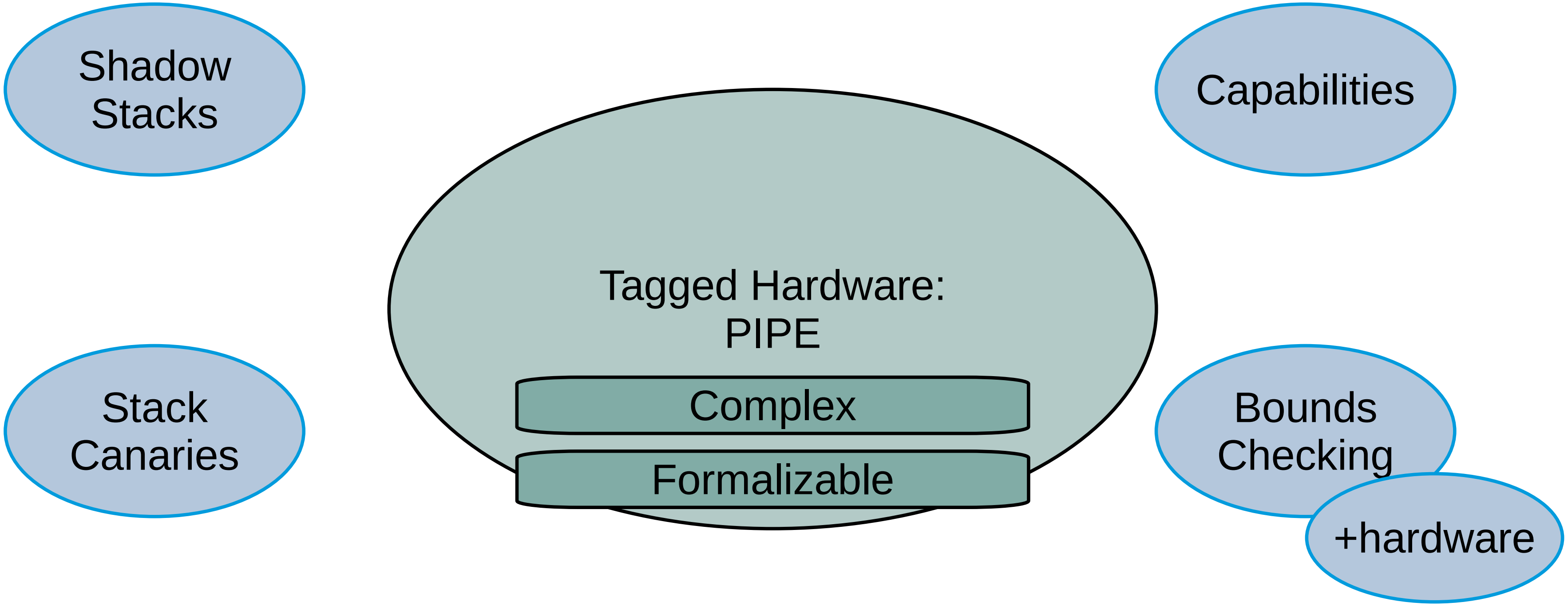
Protection Mechanisms



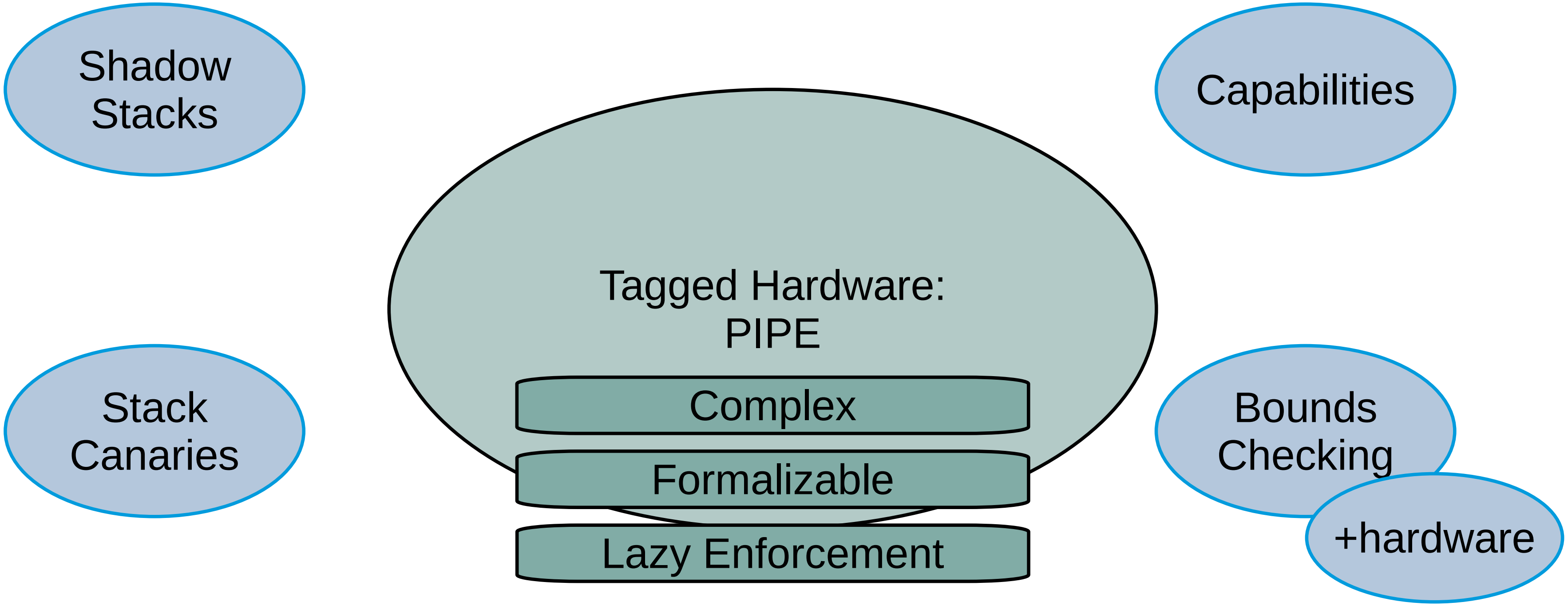
Protection Mechanisms



Protection Mechanisms



Protection Mechanisms



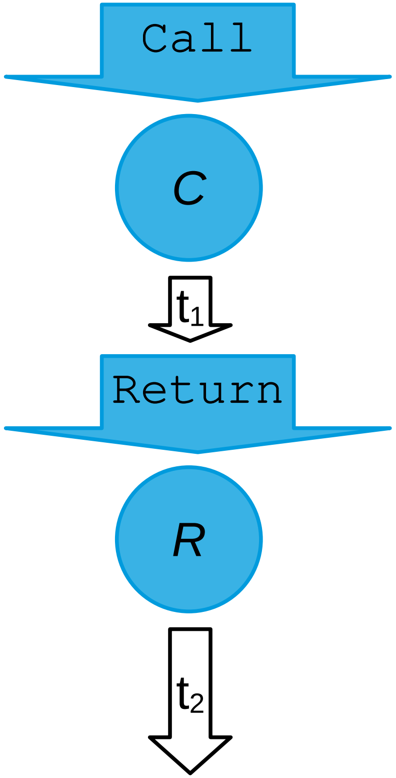
Seeking Specifications

Formal Properties Specifying Stack Safety,
Using Concepts from Theoretical Security

Seeking Specifications

Formal Properties Specifying Stack Safety,
Using Concepts from Theoretical Security

P (C,R,t_1,t_2) where

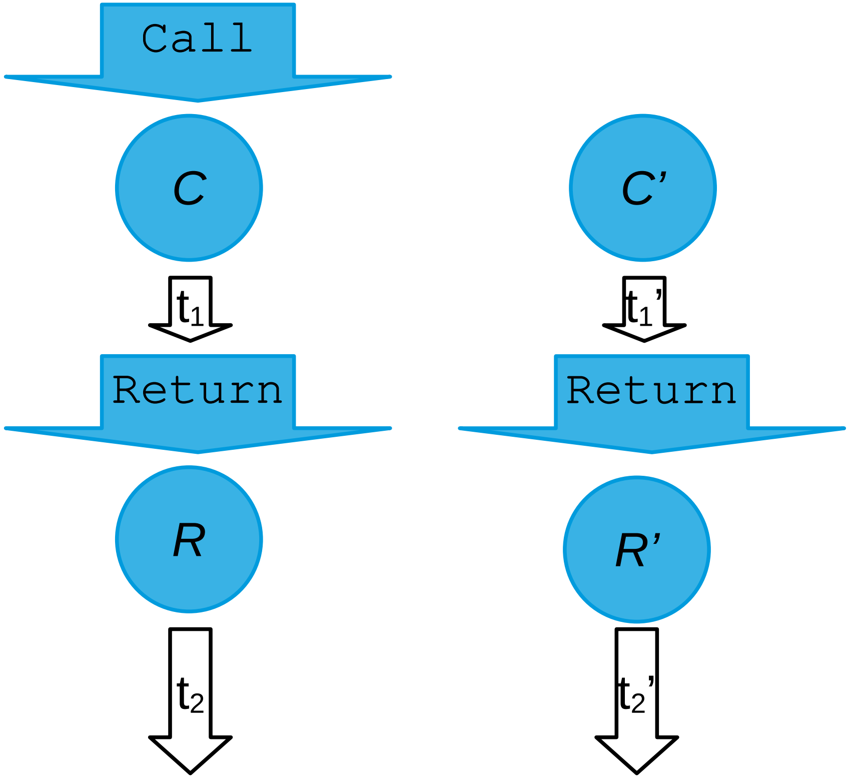


Seeking Specifications

Formal Properties Specifying Stack Safety,
Using Concepts from Theoretical Security

P (C,R,t_1,t_2) where

Or:
(C,R,t_1,t_2) R (C',R',t_1',t_2')



Contributions

Separate High-Level Security Notions

Observation-based Definitions for
Lazy Enforcement

Flexible Model of Language Features

Randomized Property-based Testing

Contributions

Separate High-Level Security Notions

- Well-bracketed Control Flow
- Caller Integrity
- Callee Confidentiality
- Caller Confidentiality
- Callee Integrity

Observation-based Definitions for Lazy Enforcement

Flexible Model of Language Features

Randomized Property-based Testing

Contributions

Separate High-Level Security Notions

Observation-based Definitions for
Lazy Enforcement

Flexible Model of Language Features

Randomized Property-based Testing

Contributions

Separate High-Level Security Notions

Observation-based Definitions for
Lazy Enforcement

Flexible Model of Language Features

Randomized Property-based Testing

Pass-by-reference

Callee-save Registers

Tailcalls

… etc.

Contributions

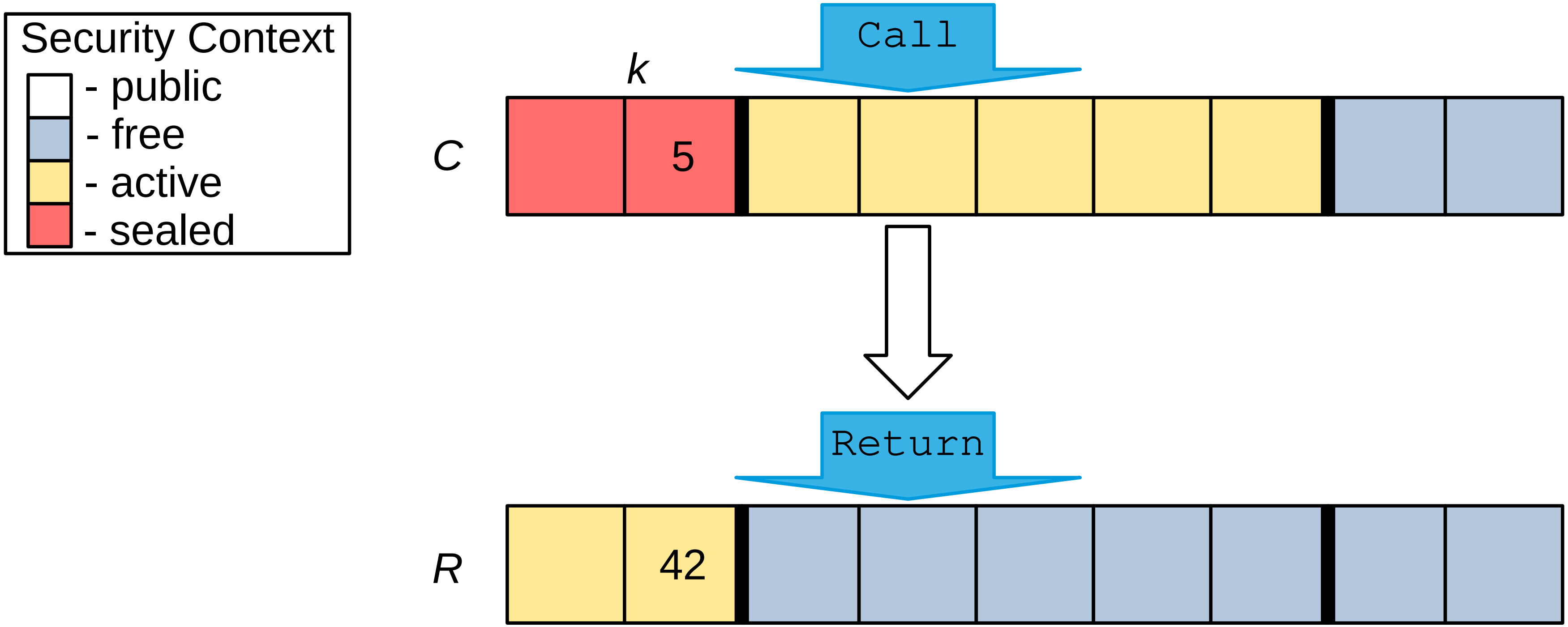
Separate High-Level Security Notions

Observation-based Definitions for
Lazy Enforcement

Flexible Model of Language Features

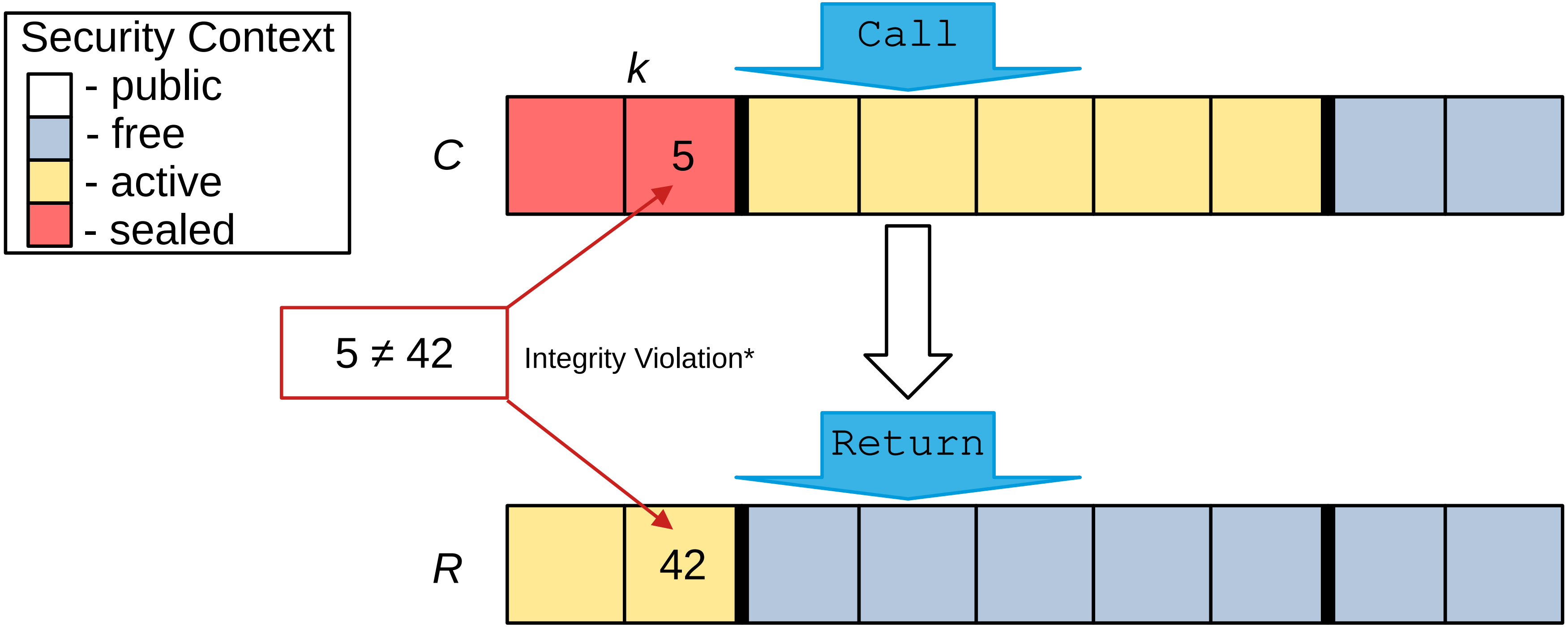
Randomized Property-based Testing

Caller Integrity



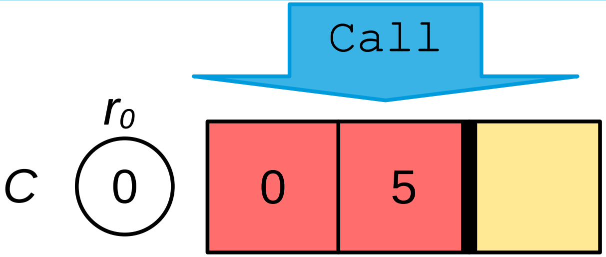
$$\forall\, C\; R\; k\,.\; k \in \mathit{sealed}(C) \rightarrow C[k] = R[k]$$

Caller Integrity



$$\forall\, C\ R\ k\ .\ k \in \mathit{sealed}(C) \rightarrow C[k] = R[k]$$

Caller Confidentiality = Non-interference



$\forall\, C$

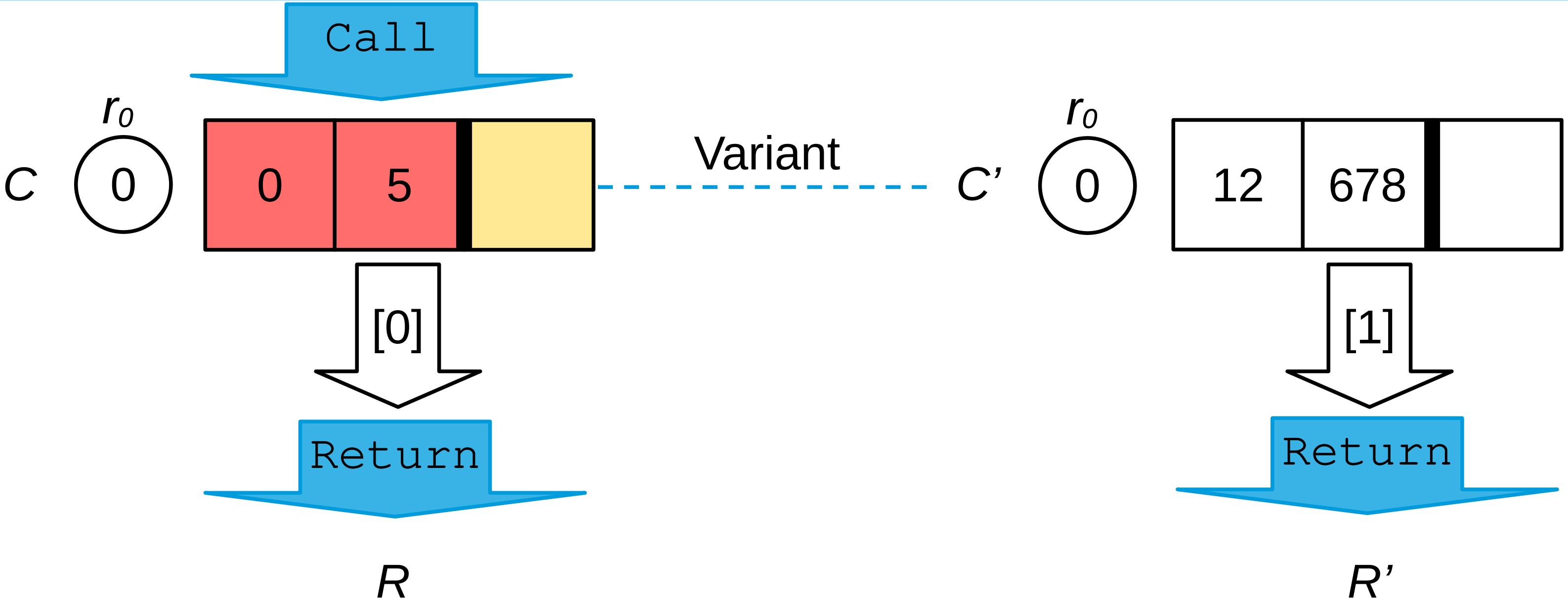
Caller Confidentiality = Non-interference

$\forall\ C\ C' \quad .\ (k \in sealed(C) \vee C[k] = C'[k]) \rightarrow$

Caller Confidentiality = Non-interference

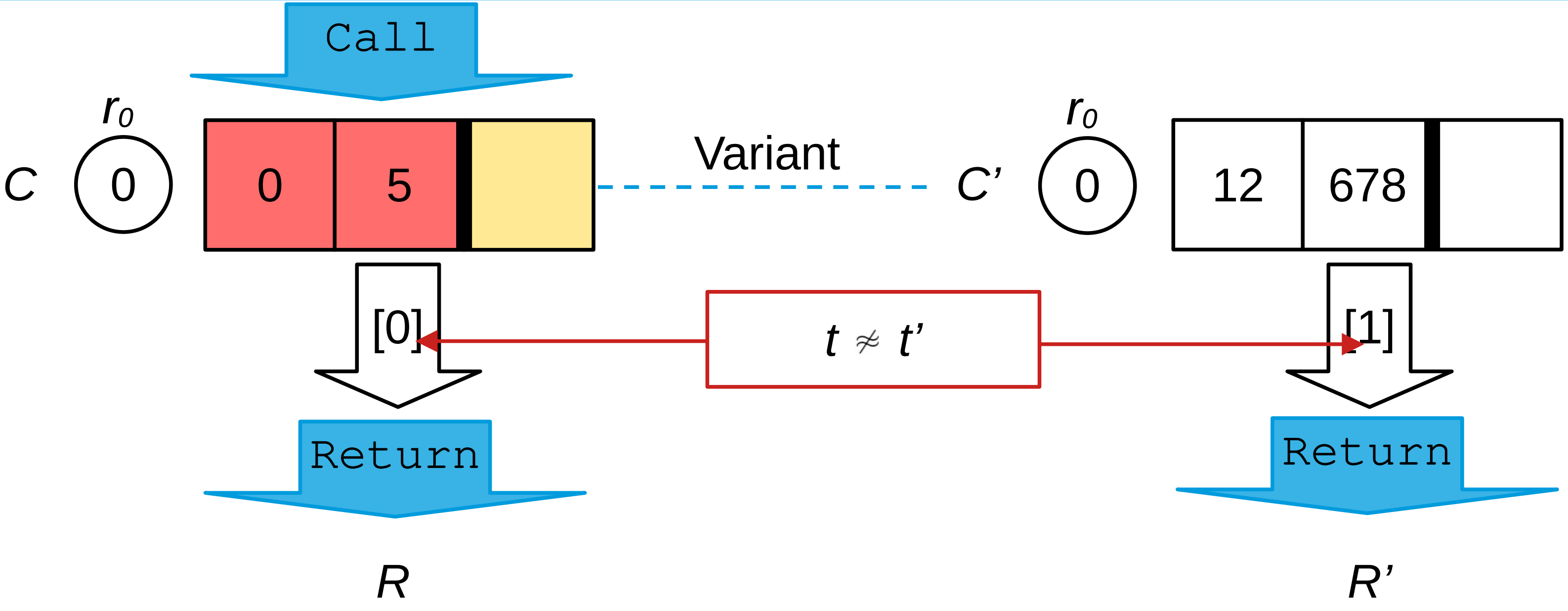
$\forall\, C\, C' \quad .\, C \sim C' \rightarrow$

Caller Confidentiality = Non-interference



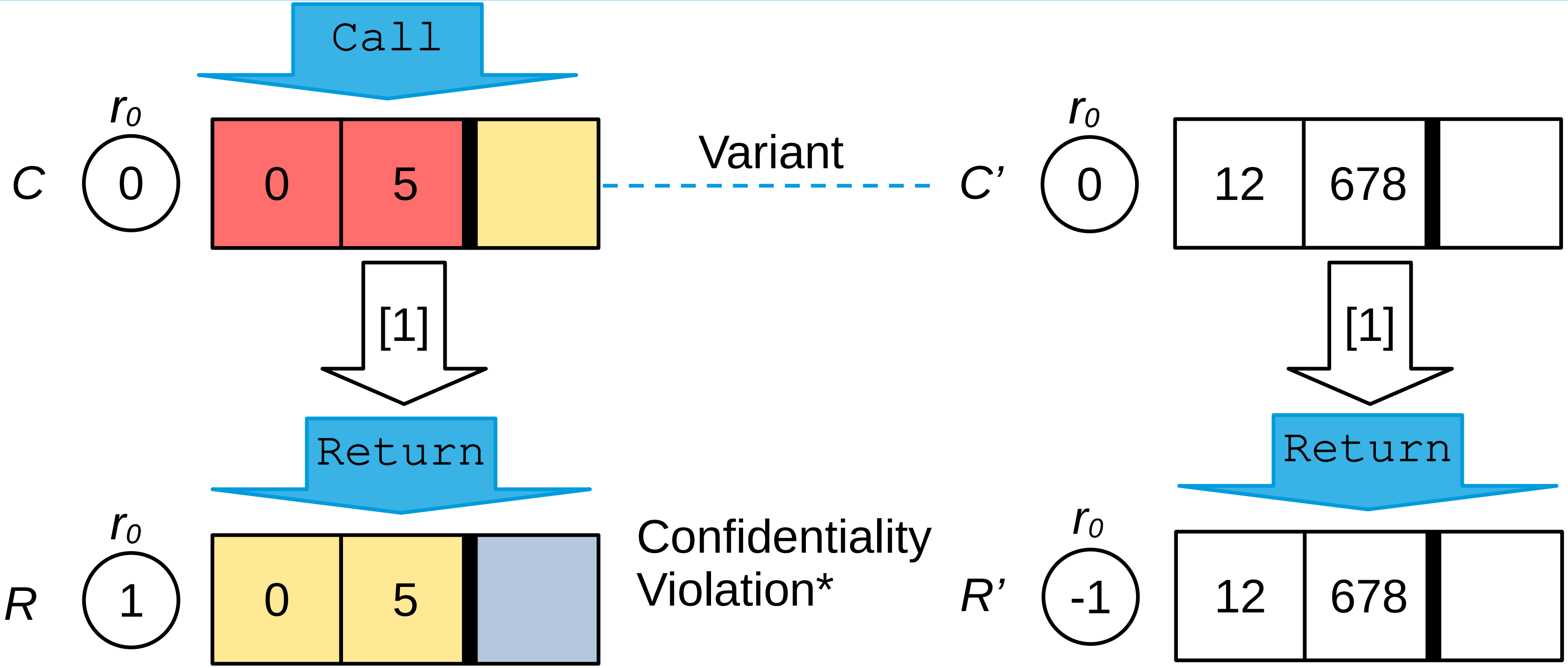
$$\forall\, C\, C'\, R\, R'.\; C \sim C' \to C \overset{t}{\Longrightarrow}{}^{*} R \to C' \overset{t'}{\Longrightarrow}{}^{*} R' \to$$

Caller Confidentiality = Non-interference



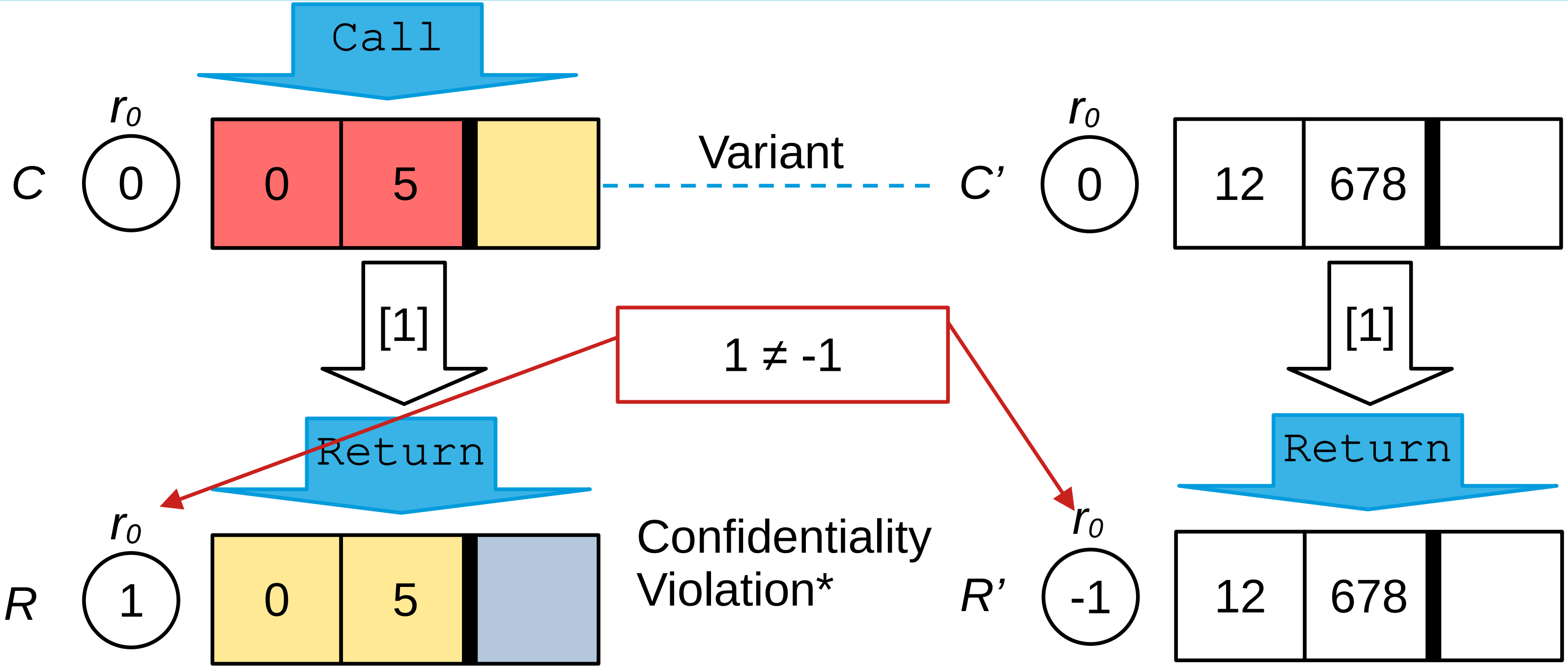
$$\forall\, C\, C'\, R\, R'.\; C \sim C' \to C \xRightarrow{t}{}^{*} R \to C' \xRightarrow{t'}{}^{*} R' \to t \approx t'$$

Caller Confidentiality = Non-interference



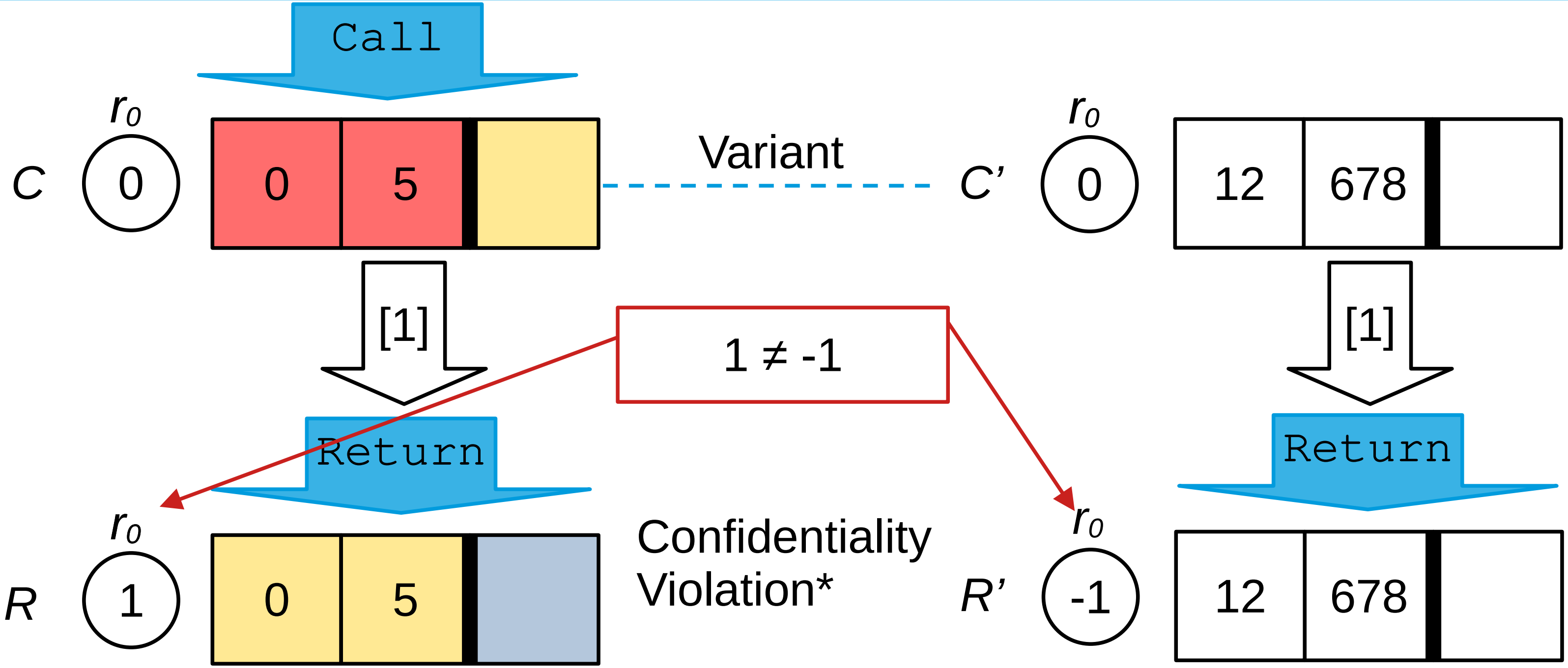
$$\forall\, C\; C'\; R\; R'.\; C \sim C' \rightarrow C \overset{t}{\Longrightarrow}{}^{*} R \rightarrow C' \overset{t'}{\Longrightarrow}{}^{*} R' \rightarrow t \approx t'$$

Caller Confidentiality = Non-interference



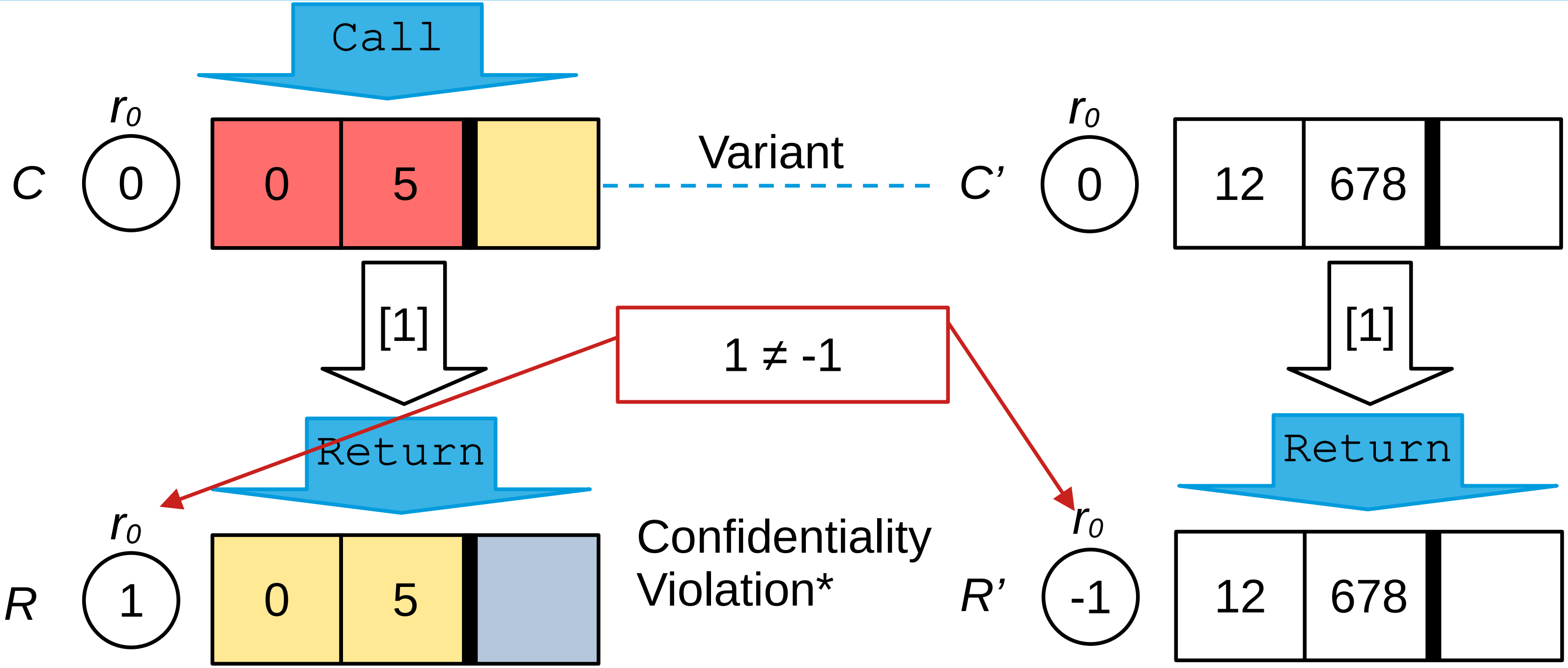
$$\forall\, C\, C'\, R\, R'.\; C \sim C' \rightarrow C \overset{t}{\Longrightarrow}{}^{*} R \rightarrow C' \overset{t'}{\Longrightarrow}{}^{*} R' \rightarrow t \approx t'$$

Caller Confidentiality = Non-interference



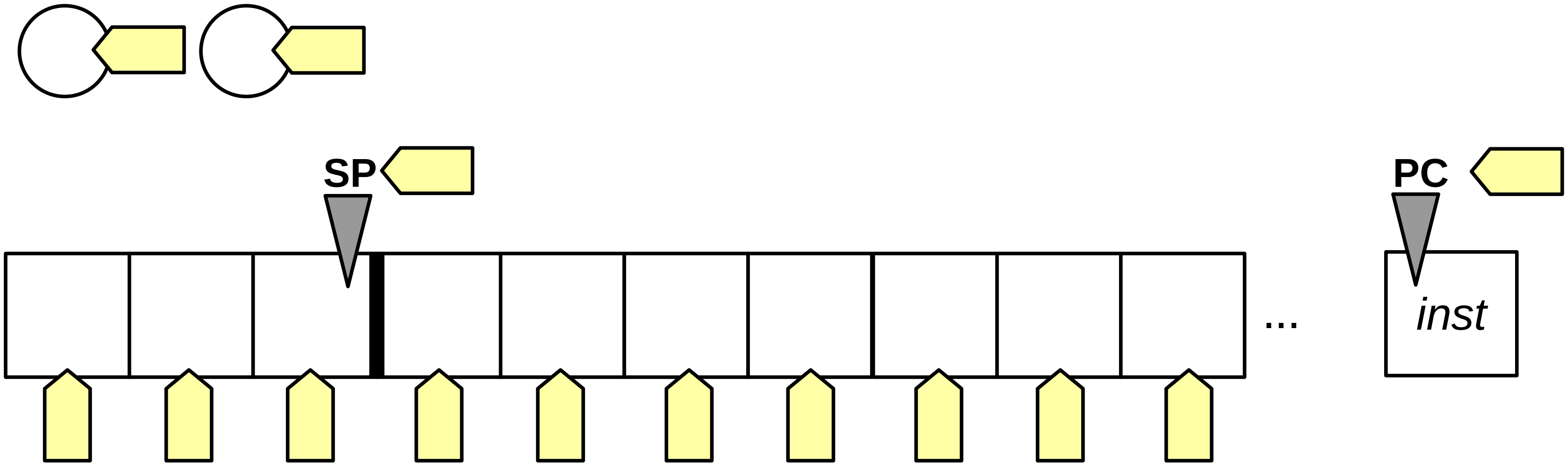
$$\forall\, C\; C'\; R\; R' .\; C \sim C' \rightarrow\; C \overset{t}{\Longrightarrow}{}^{*} R \rightarrow C' \overset{t'}{\Longrightarrow}{}^{*} R' \rightarrow$$
$$t \approx t' \wedge (\forall\, k .\; C[k] \neq R[k] \vee C'[k] \neq R'[k] \rightarrow R[k] = R'[k])$$

Caller Confidentiality = Non-interference

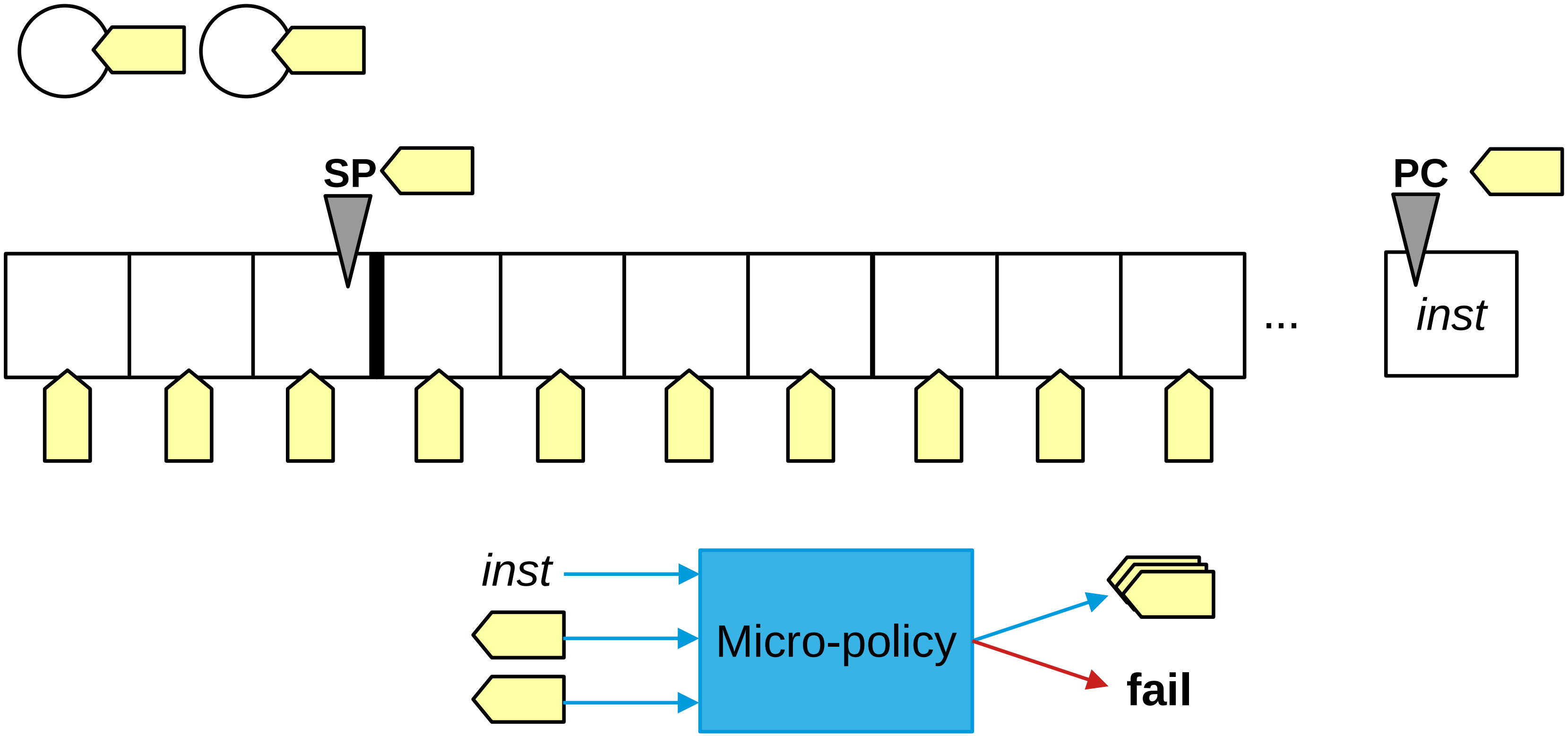


$$\forall\, C\ C'\ R\ R' .\ C \sim C' \rightarrow C \overset{t}{\Longrightarrow}{}^{*} R \rightarrow C' \overset{t'}{\Longrightarrow}{}^{*} R' \rightarrow$$
$$t \approx t' \wedge (\forall\, k .\ consistent(C,C',R,R',k))$$

Programmable Interlocks for Policy Enforcement: **PIPE**

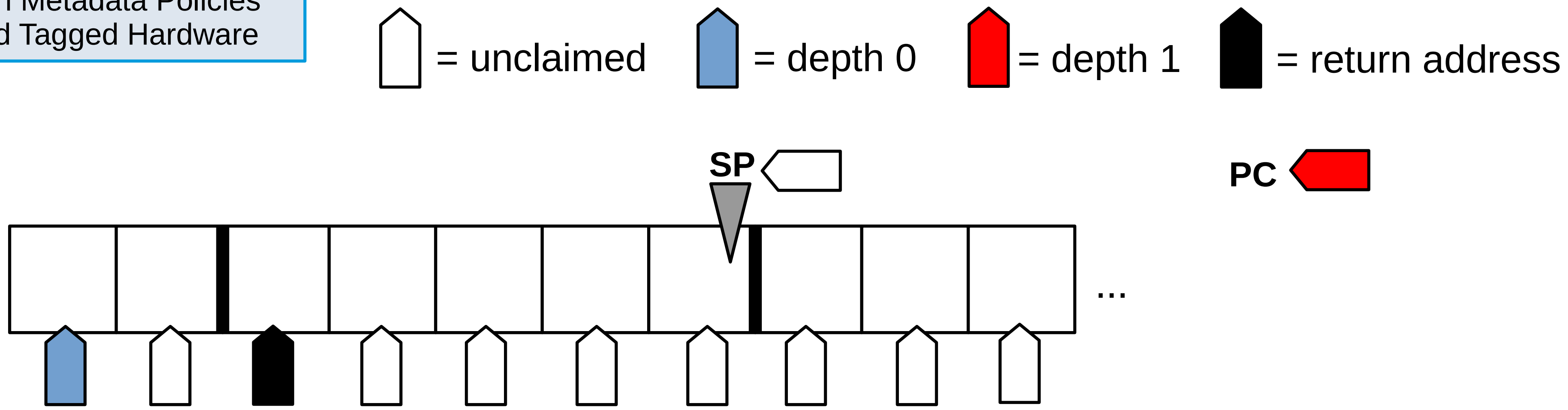


Programmable Interlocks for Policy Enforcement: **PIPE**



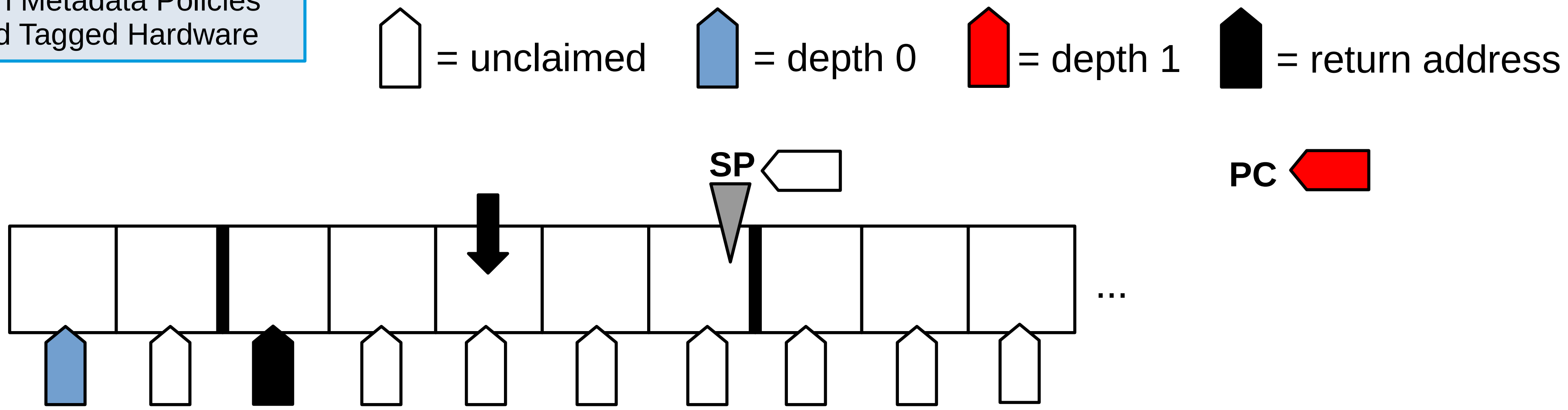
Lazy Depth Isolation

Roessler and DeHon, 2018
Protecting the Stack
with Metadata Policies
and Tagged Hardware



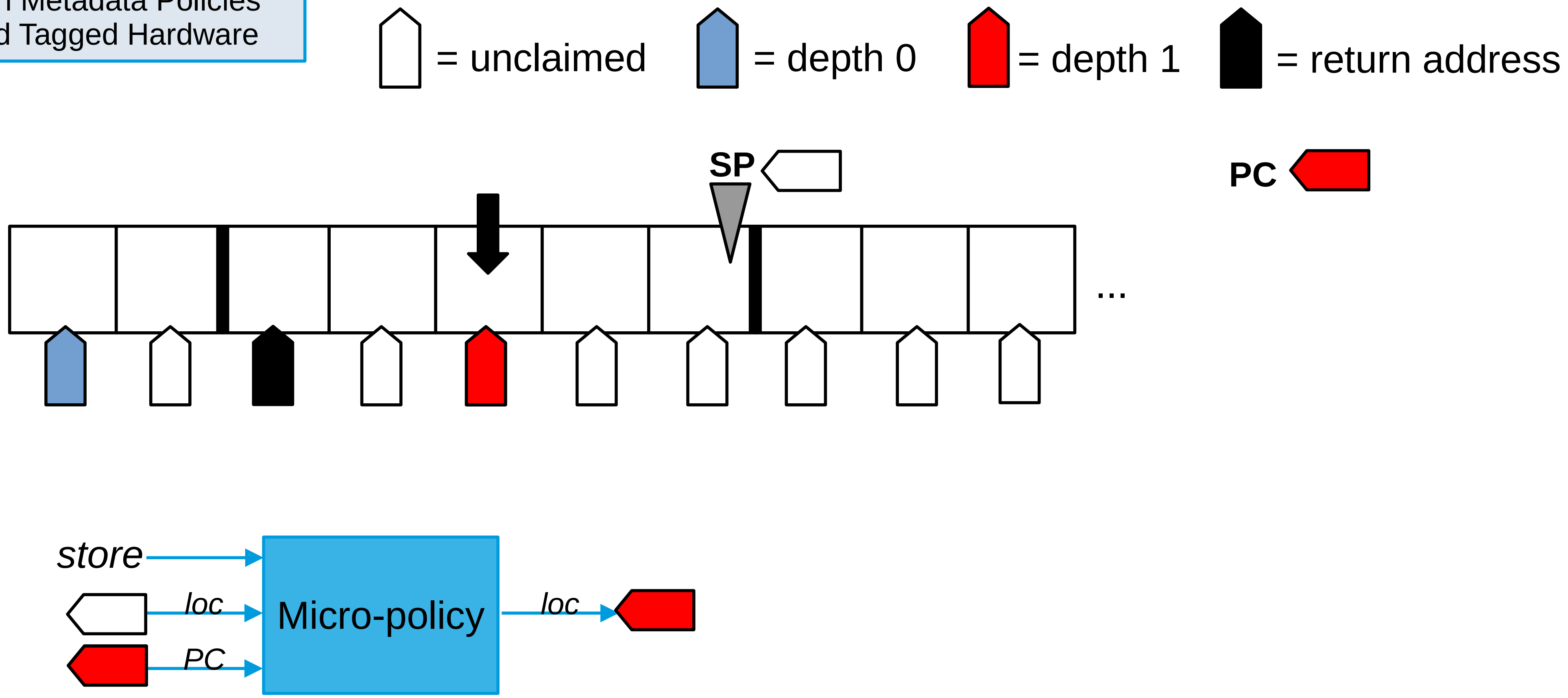
Roessler and DeHon, 2018
Protecting the Stack
with Metadata Policies
and Tagged Hardware

Lazy Depth Isolation



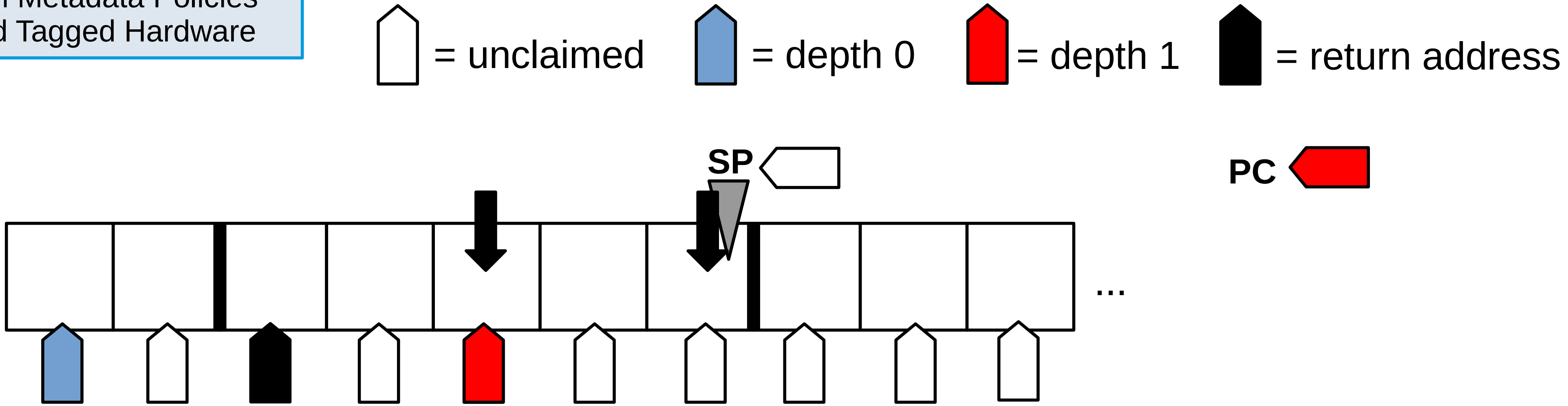
Lazy Depth Isolation

Roessler and DeHon, 2018
Protecting the Stack
with Metadata Policies
and Tagged Hardware



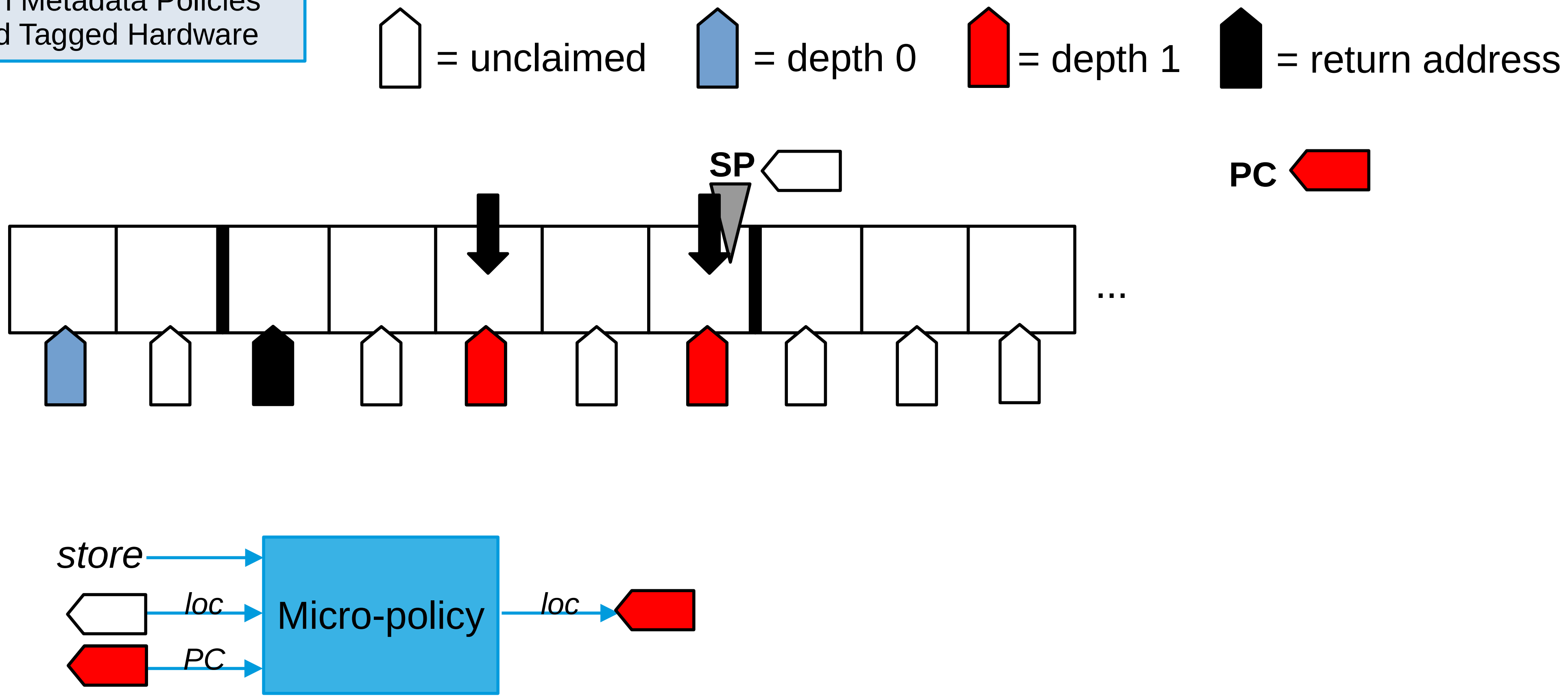
Lazy Depth Isolation

Roessler and DeHon, 2018
Protecting the Stack
with Metadata Policies
and Tagged Hardware



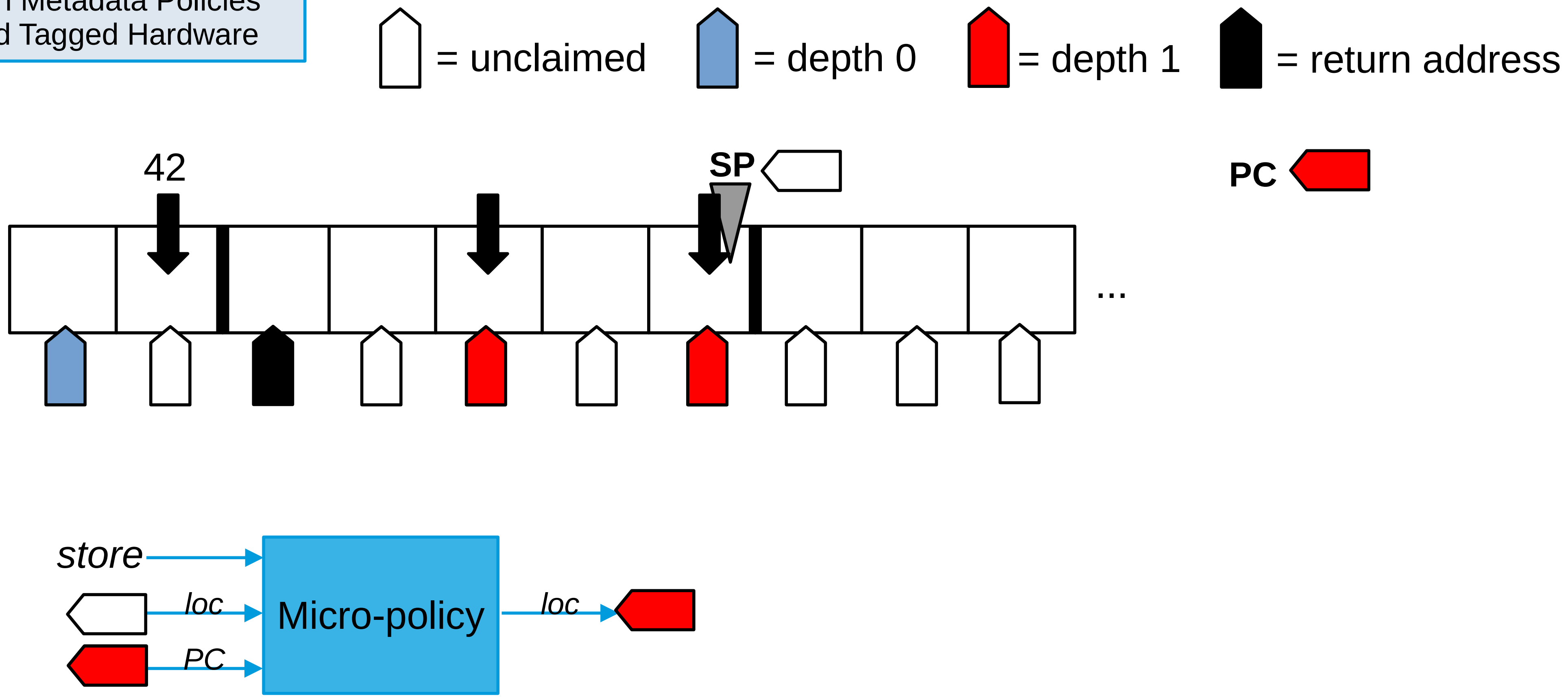
Lazy Depth Isolation

Roessler and DeHon, 2018
Protecting the Stack
with Metadata Policies
and Tagged Hardware



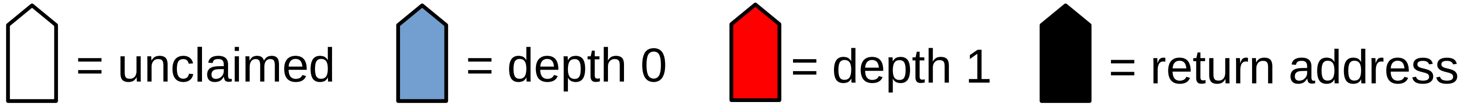
Lazy Depth Isolation

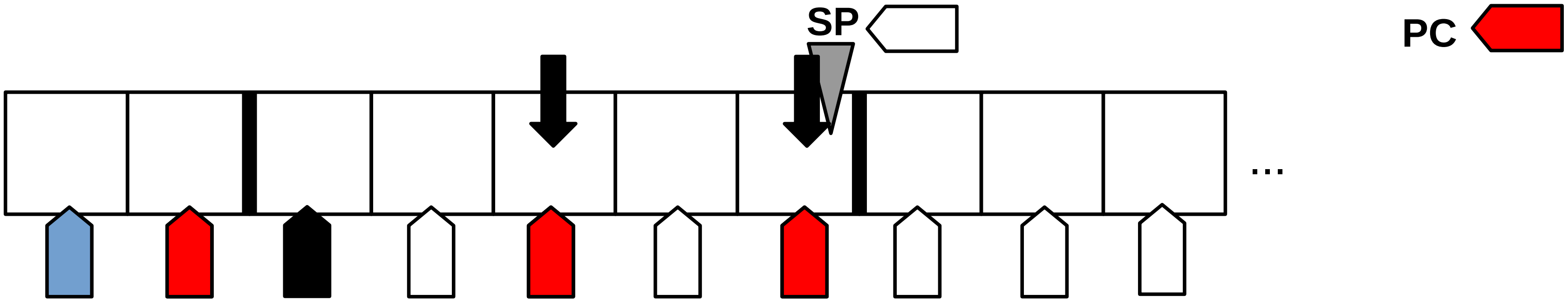
Roessler and DeHon, 2018
Protecting the Stack
with Metadata Policies
and Tagged Hardware



Roessler and DeHon, 2018
Protecting the Stack
with Metadata Policies
and Tagged Hardware

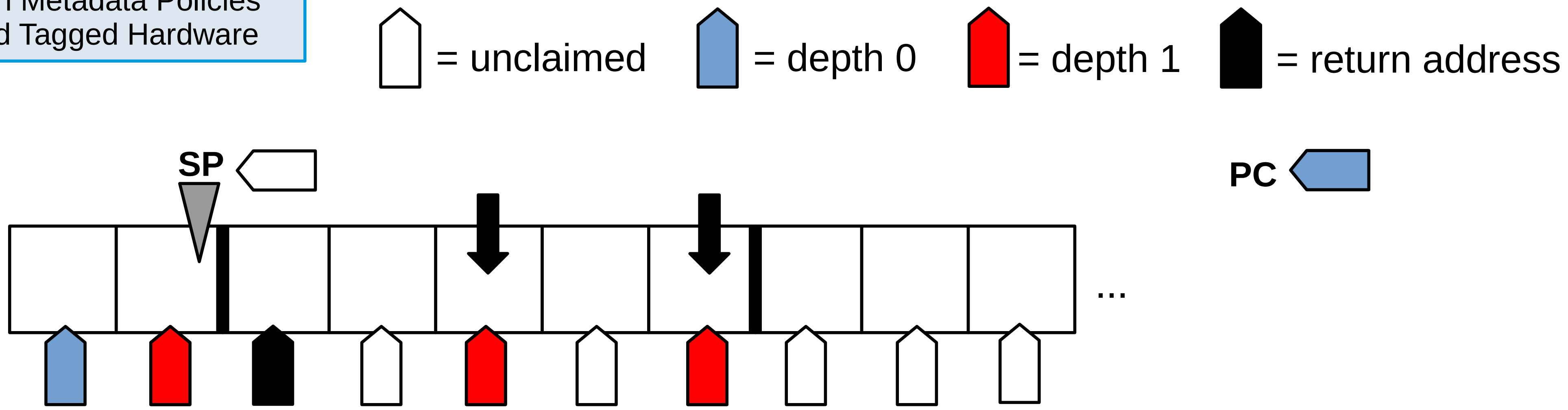
Lazy Depth Isolation





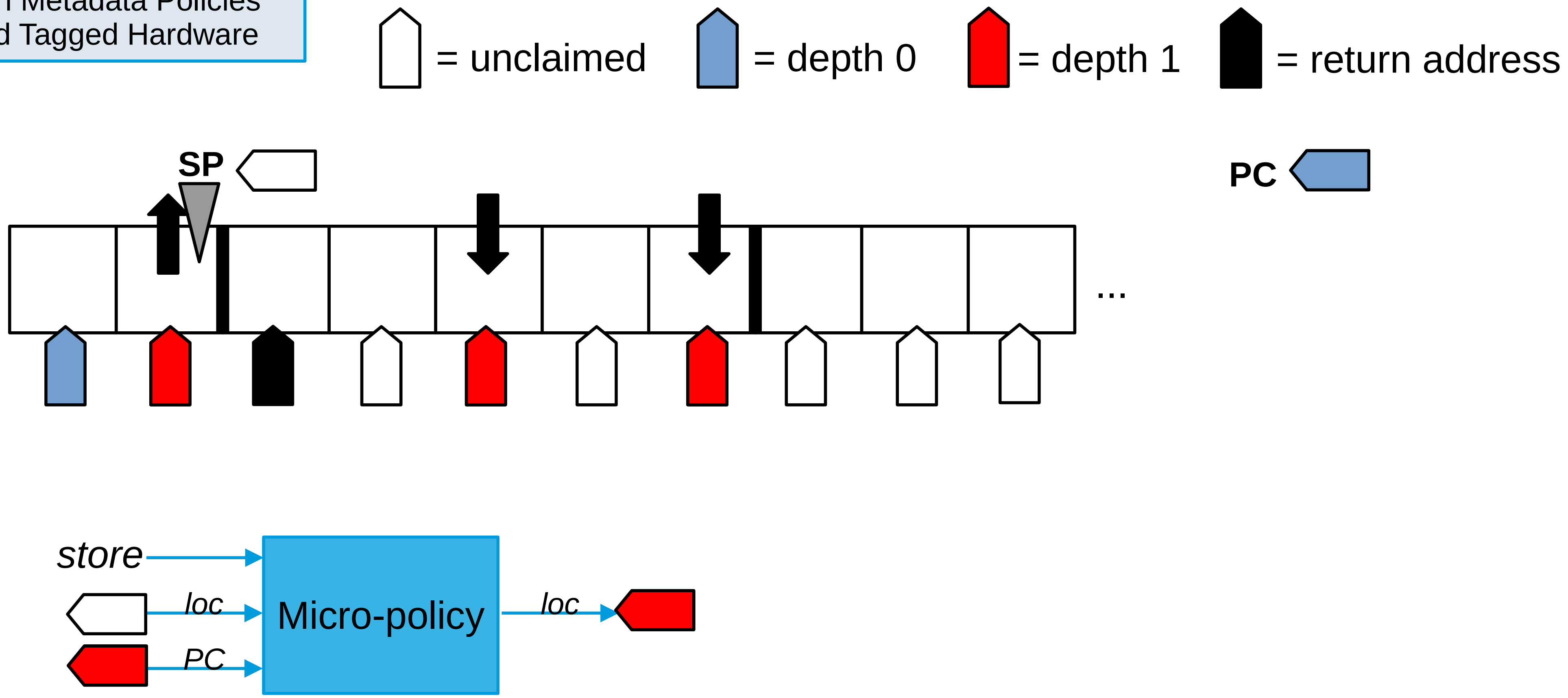
Roessler and DeHon, 2018
Protecting the Stack
with Metadata Policies
and Tagged Hardware

Lazy Depth Isolation



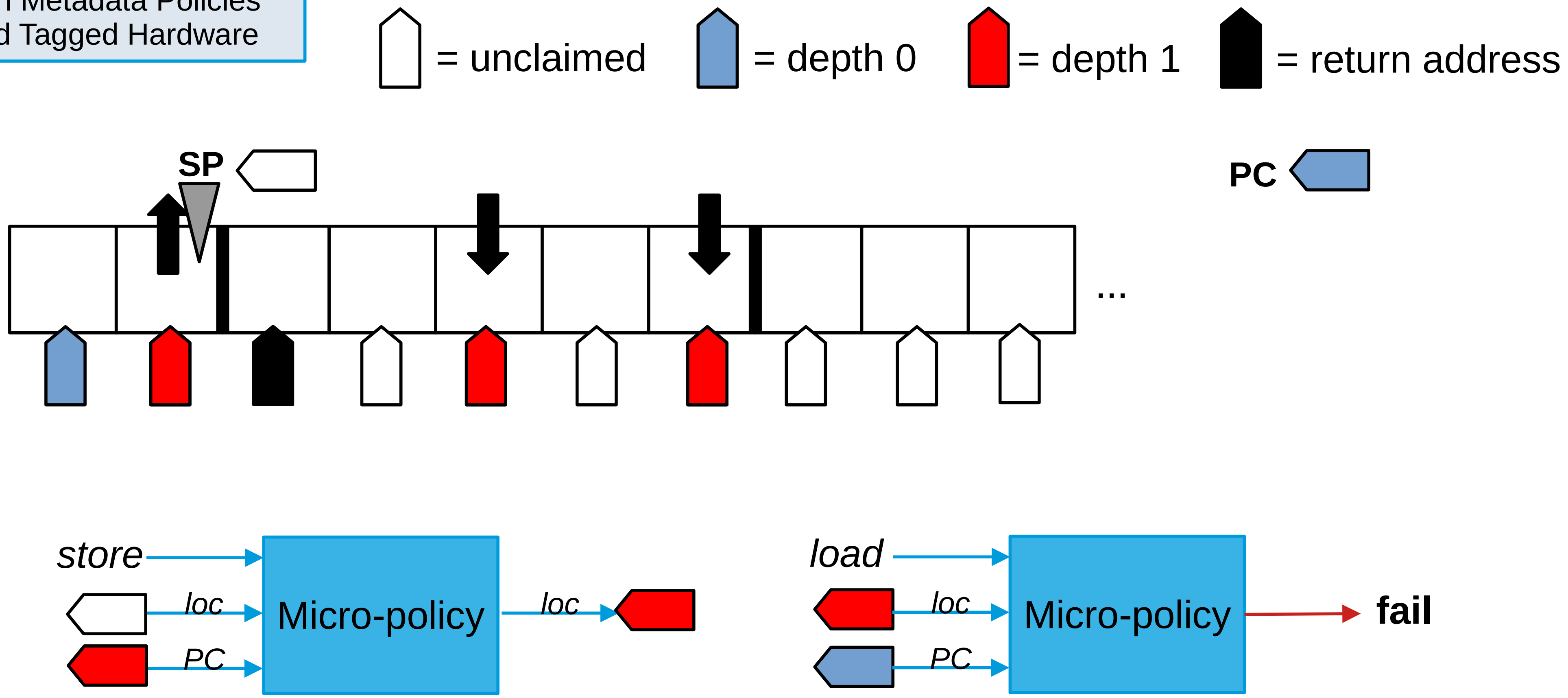
Lazy Depth Isolation

Roessler and DeHon, 2018
Protecting the Stack
with Metadata Policies
and Tagged Hardware

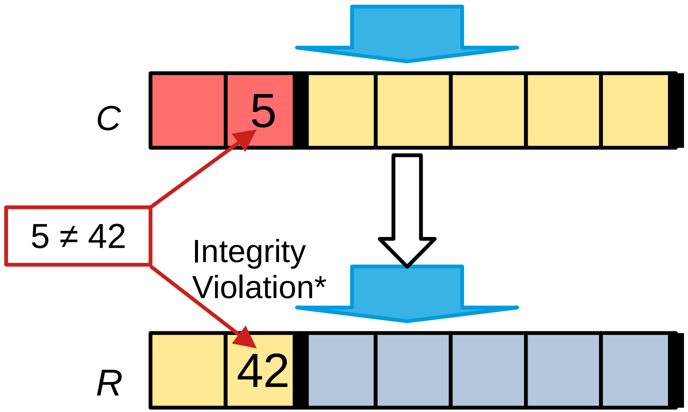


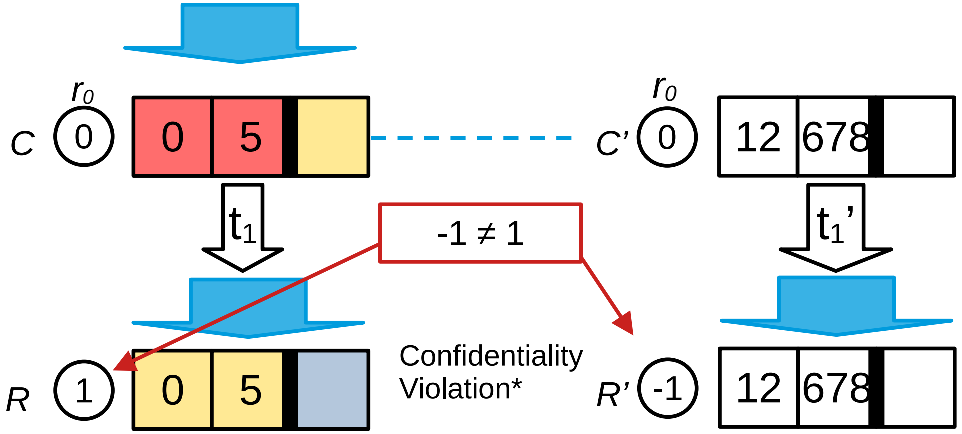
Roessler and DeHon, 2018
Protecting the Stack
with Metadata Policies
and Tagged Hardware

Lazy Depth Isolation

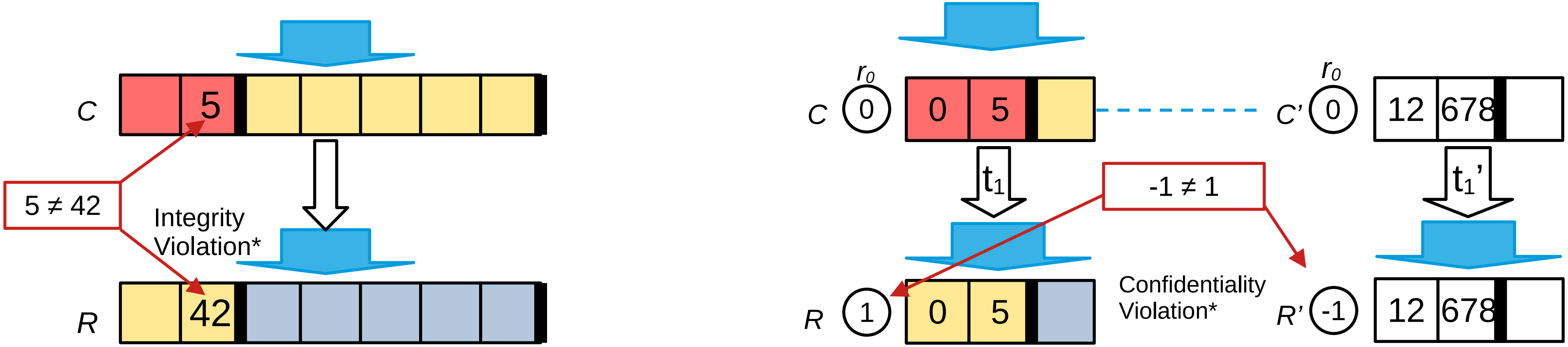


Properties, Observationally



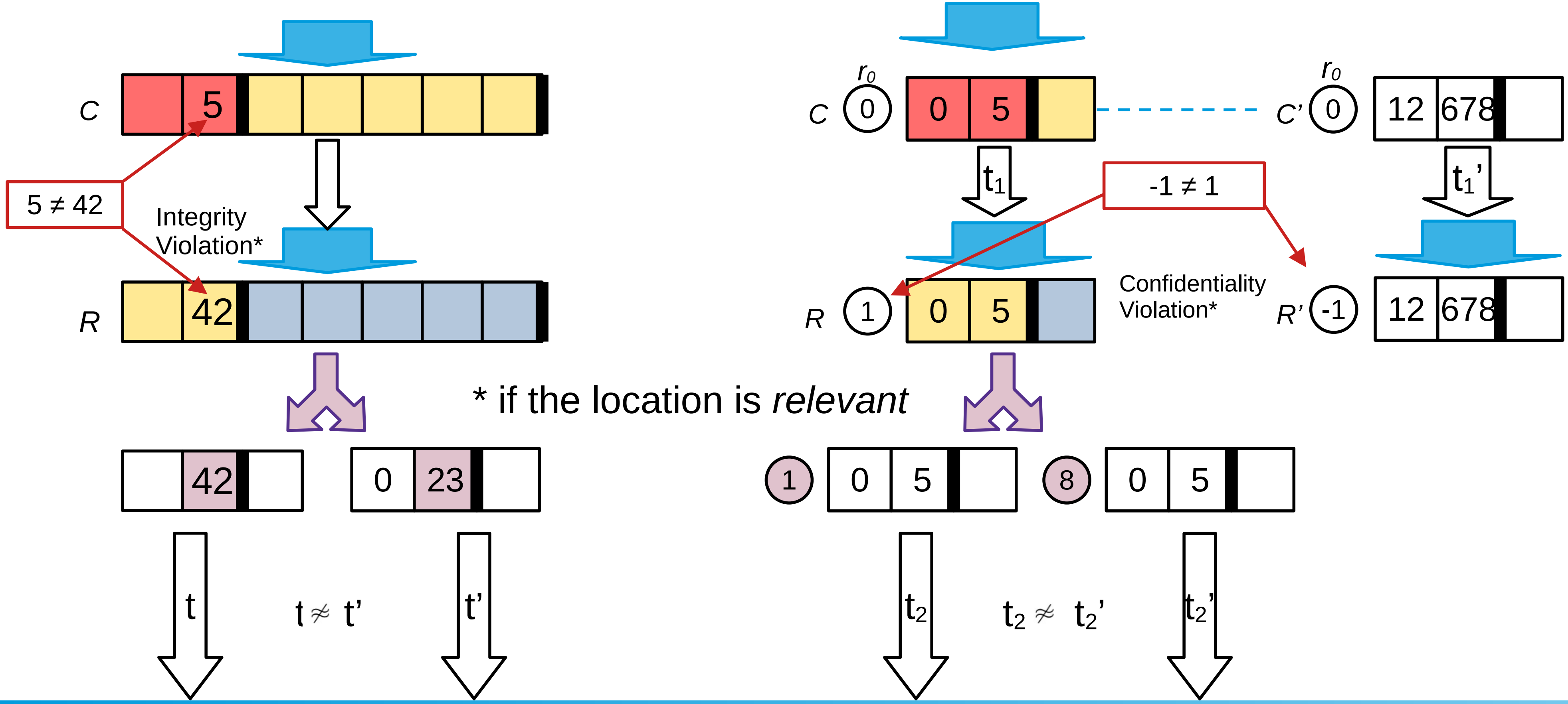


Properties, Observationally



* if the location is *relevant*

Properties, Observationally



* if the location is *relevant*

Properties, Observationally

$$\textit{irrelevant}(R, K) \triangleq \forall R' . (\forall k \notin K . C[k] = C'[k]) \rightarrow$$

$$R \xRightarrow{t}{}^{*} \perp \rightarrow R' \xRightarrow{t'}{}^{*} \perp \rightarrow$$

$$t \approx t'$$

Caller Integrity $\triangleq$

$$\forall C\ R\ k . k : \textit{sealed}(C) \rightarrow \textit{irrelevant}(R, \{ k \mid C[k] \neq R[k] \})$$

Caller Confidentiality $\triangleq$

$$\forall C\ C'\ R\ R' . C \sim C' \rightarrow C \xRightarrow{t}{}^{*} R \rightarrow C' \xRightarrow{t'}{}^{*} R' \rightarrow$$

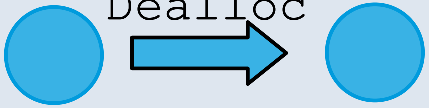
$$t \approx t' \wedge \textit{irrelevant}(R, \{ k \mid \neg\, \textit{consistent}(C, C', R, R', k) \})$$


Security-relevant Operations
Update
Security Context
Inform
Hyperproperties

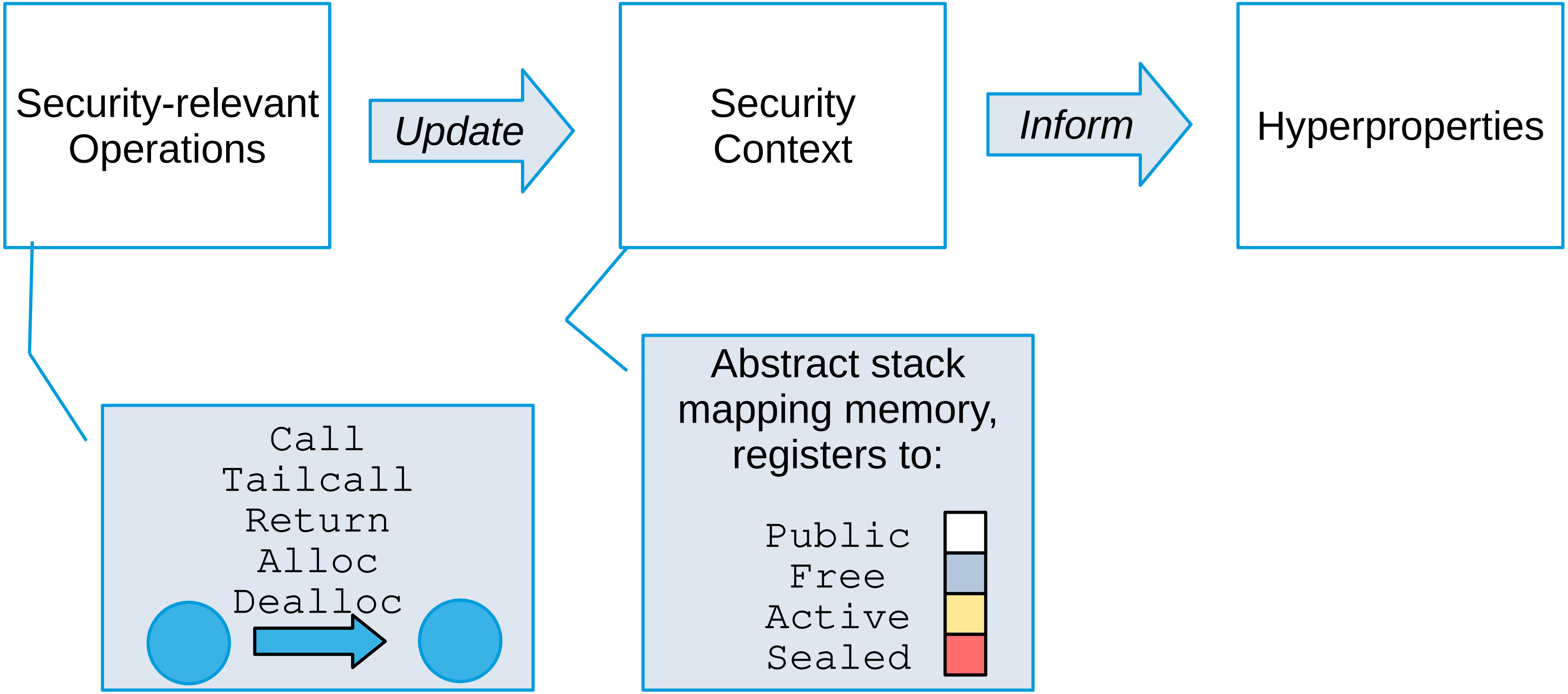
Language Features via Security Semantics

Security-relevant Operations → *Update* → Security Context → *Inform* → Hyperproperties

```
Call
Tailcall
Return
Alloc
Dealloc
```



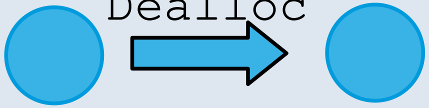
Language Features via Security Semantics



Language Features via Security Semantics

Security-relevant Operations → *Update* → Security Context → *Inform* → Hyperproperties

Call
Tailcall
Return
Alloc
Dealloc



Abstract stack mapping memory, registers to:

Public
Free
Active
Sealed

Given security context at call:

WBCF
Integrity
Confidentiality

Language Features

Callee-save Registers

Pass-by-reference

“Public” (address-taken) variables

Tailcalls

Arguments Spilled to Memory

Memory Shared by Capability

Ongoing Work:

Exceptions

Concurrency

Property-based Testing w/ QuickChick

Random Program

Integrity Test

Conf. Test

\+

(Eager) Depth Isolation

Depth Isolation w/ Lazy Tagging

“Activation Isolation” w/ Lazy Tagging

Property-based Testing w/ QuickChick

Random Program

Integrity Test

Conf. Test

✗/? ✗/?

+

(Eager) Depth Isolation ✓

Depth Isolation w/ Lazy Tagging ✗

“Activation Isolation” w/ Lazy Tagging ✓

Property-based Testing w/ QuickChick

Random Program +

Integrity Test → ✗ / ?

Conf. Test → ✗ / ?

- (Eager) Depth Isolation ✓
- Depth Isolation w/ Lazy Tagging ✗
- "Activation Isolation" w/ Lazy Tagging ✓

Injected Bugs ✗

Future Work

- Testing Other Mechanisms
  - Cerise, CHERI-based calling convention
  - Software bounds checking
- Expanding model
  - Exceptions
  - Concurrency
- Mechanized Proofs

Summary

- Theory of Stack Safety: Integrity, Confidentiality, and/or WBCF
- Security Semantics: useful factorization over language features
- Observation-based properties to describe sophisticated enforcement mechanism
- Testing identifies flawed micro-policies