Tipsy: A Source-to-Source Binary Obfuscation Tool

SSPREW 2018

Christian Collberg
University of Arizona

Tigress: A Source-to-Source-ish Code Obfuscation Tool

SSPREW 2018

Christian Collberg
University of Arizona

A Code Obfuscator transforms a program into one that is harder to analyze and reverse engineer

```
main(){
}
```

Code Obfuscator

Obfuscating Transformations

- Max complexity
- Min overhead

A Code Obfuscator transforms a program into one that is harder to analyze and reverse engineer

Code Obfuscator

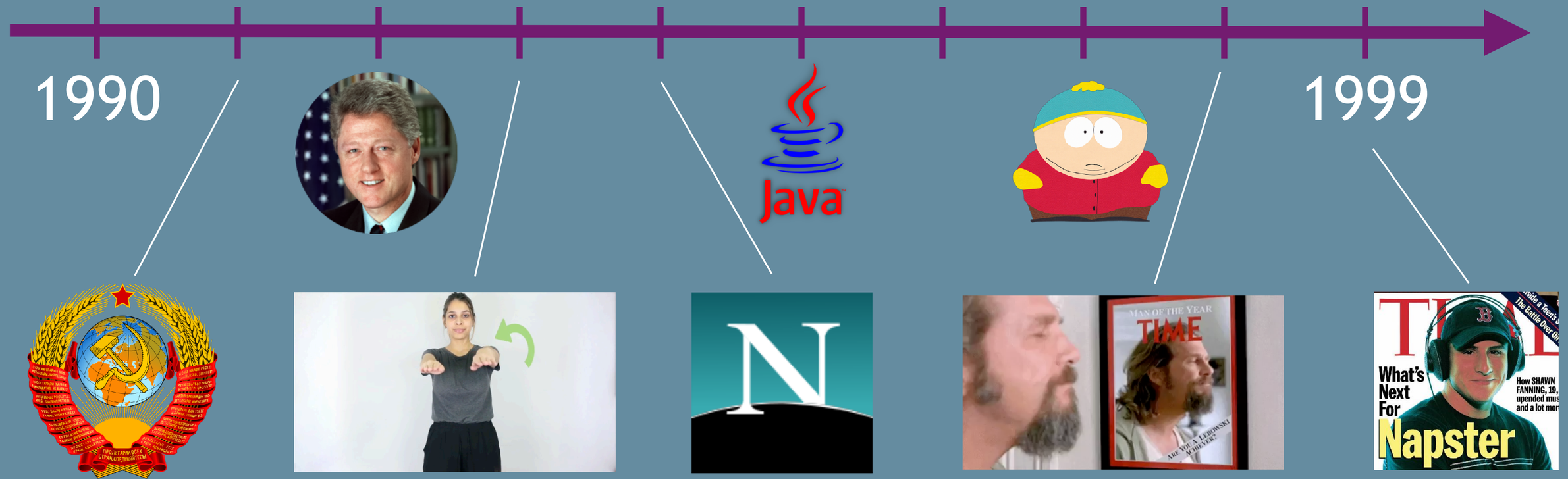
Obfuscating Transformations

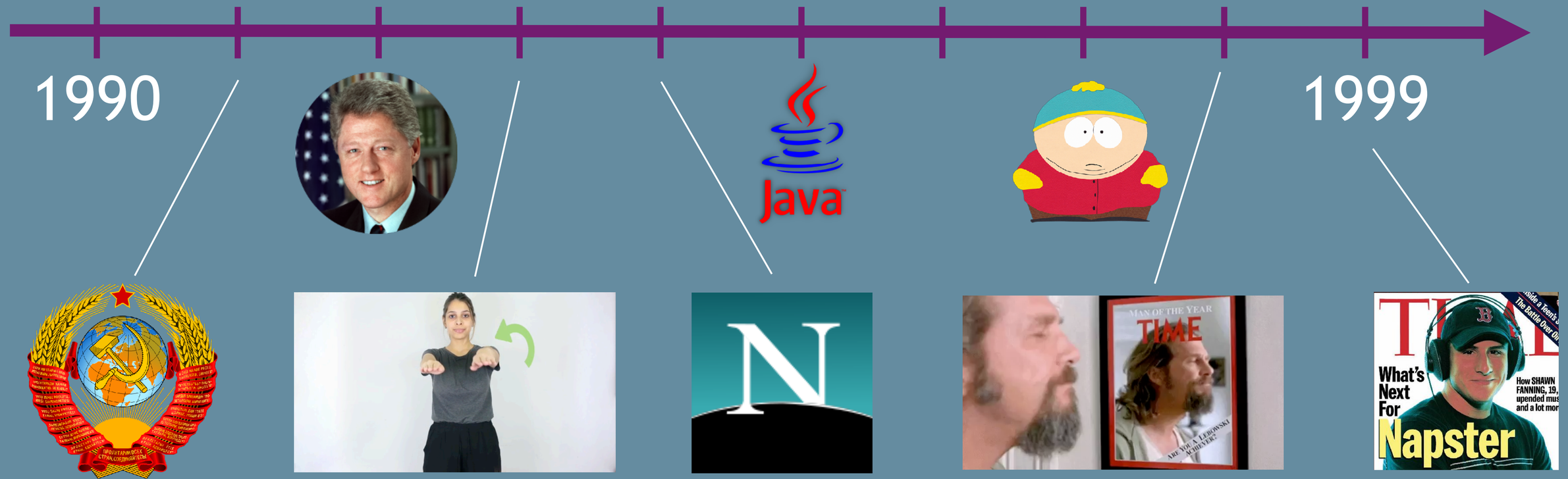
- Max complexity
- Min overhead

```
main(){
```

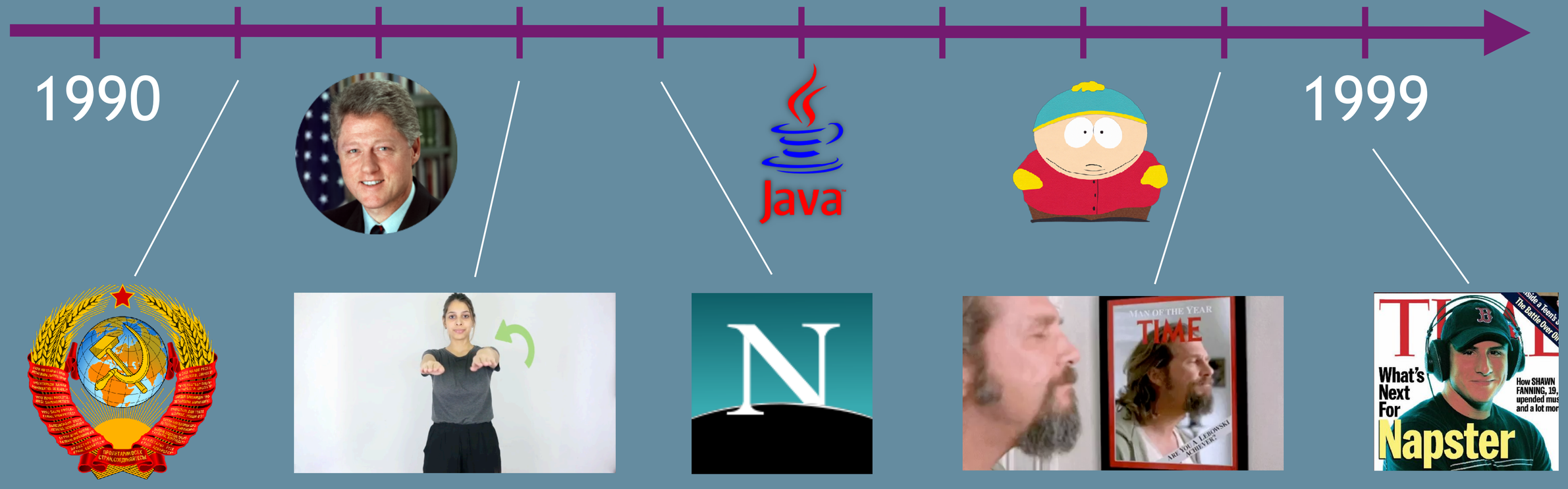
- Source
- Algorithms
- Keys
- Media
- Integrity Checks

```
}
```


1990
Java
1999
MAN OF THE YEAR
TIME
What's Next For Napster
How SHAWN FANNING, 19,
and a lot mor


1990
Java
1999
MAN OF THE YEAR
TIME
What's Next For Napster
How SHAWN FANNING, 19,
and a lot mor

Protect Against Media Theft (DRM)



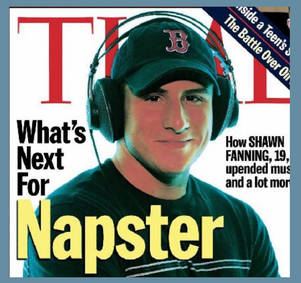
Protect Against Media Theft (DRM)

Protect Viruses Against Anti-Virus

1990

Java

1999



Protect Against Media Theft (DRM)

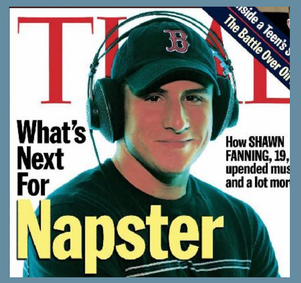
Protect Viruses Against Anti-Virus

Protect OS Against Malware

1990

Java

1999



Protect Against Media Theft (DRM)

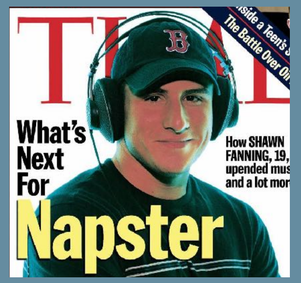
Protect Viruses Against Anti-Virus

Protect OS Against Malware

Protect Java Against IP Theft

1990

1999



Operating System Protection Through Program Evolution

By Dr. Frederick B. Cohen
October 1993

Seed →

Code Obfuscator

Diversifying Transformations

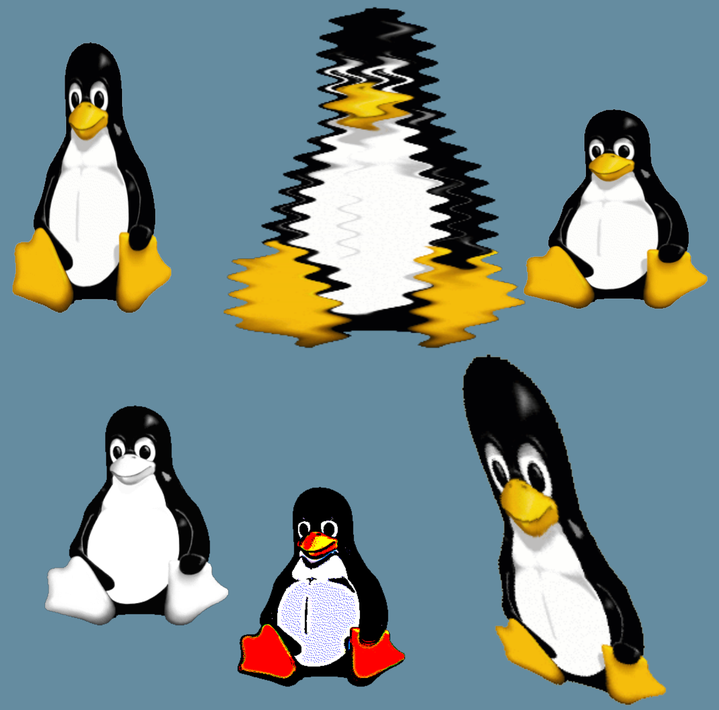
Operating System Protection Through Program Evolution

By Dr. Frederick B. Cohen
October 1993

Seed →

Code Obfuscator

Diversifying Transformations



United States Patent

Ginter et al.

Patent Number: 5,892,900

Date of Patent: Apr. 6, 1999

SYSTEM AND METHODS FOR SECURE TRANSACTION MANAGEMENT AND ELECTRONIC RIGHTS PROTECTION

Customer

Producer

intertrust®

Victor Shear

Banker

Publisher

Artist

United States Patent

Ginter et al.

Patent Number: 5,892,900

Date of Patent: Apr. 6, 1999

SYSTEM AND METHODS FOR SECURE TRANSACTION MANAGEMENT AND ELECTRONIC RIGHTS PROTECTION

Customer

Producer

intertrust®

Victor Shear

Banker

Publisher

Artist

United States Patent

Ginter et al.

Patent Number: 5,892,900

Date of Patent: Apr. 6, 1999

SYSTEM AND METHODS FOR SECURE TRANSACTION MANAGEMENT AND ELECTRONIC RIGHTS PROTECTION

Customer

Producer

intertrust®

Victor Shear

Banker

Publisher

Artist

United States Patent

Ginter et al.

Patent Number: 5,892,900

Date of Patent: Apr. 6, 1999

SYSTEM AND METHODS FOR SECURE TRANSACTION MANAGEMENT AND ELECTRONIC RIGHTS PROTECTION

Customer

Producer

intertrust®

Victor Shear

Banker

Publisher

Artist

United States Patent

Ginter et al.

Patent Number: 5,892,900

Date of Patent: Apr. 6, 1999

SYSTEM AND METHODS FOR SECURE TRANSACTION MANAGEMENT AND ELECTRONIC RIGHTS PROTECTION

intertrust®

Victor Shear

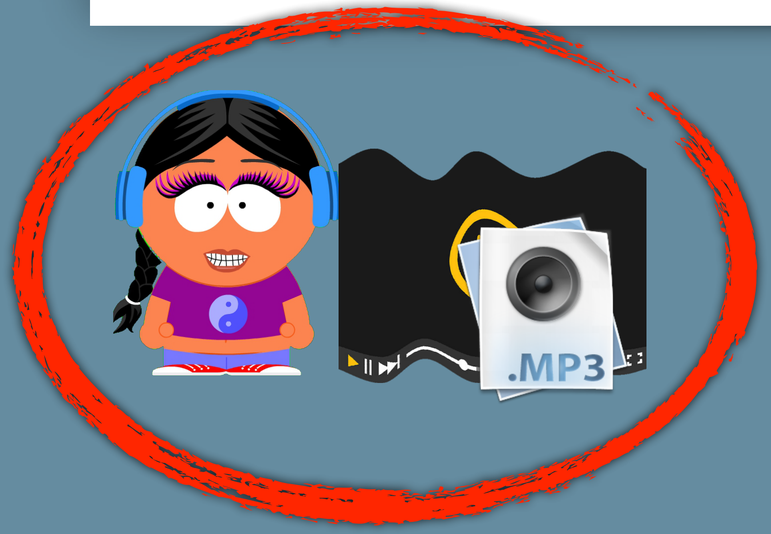
United States Patent

Ginter et al.

Patent Number: 5,892,900

Date of Patent: Apr. 6, 1999

SYSTEM AND METHODS FOR SECURE TRANSACTION MANAGEMENT AND ELECTRONIC RIGHTS PROTECTION



Producer

intertrust®

Victor Shear

Banker

Publisher

Artist

s Patent

Patent Number: 5,892,900
Date of Patent: Apr. 6, 1999

OR SECURE
MENT AND
OTECTION

napster

Producer

intertrust®

Banker

Publisher

Victo

A Taxonomy of Obfuscating Transformations

Christian Collberg Clark Thomborson. Douglas Low

July 1997

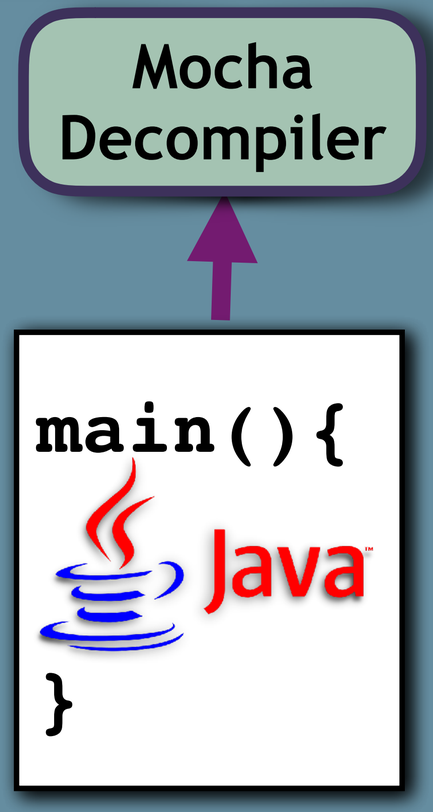
```
main(){
}
```

Java 1.0, 1995

A Taxonomy of Obfuscating Transformations

Christian Collberg Clark Thomborson. Douglas Low

July 1997

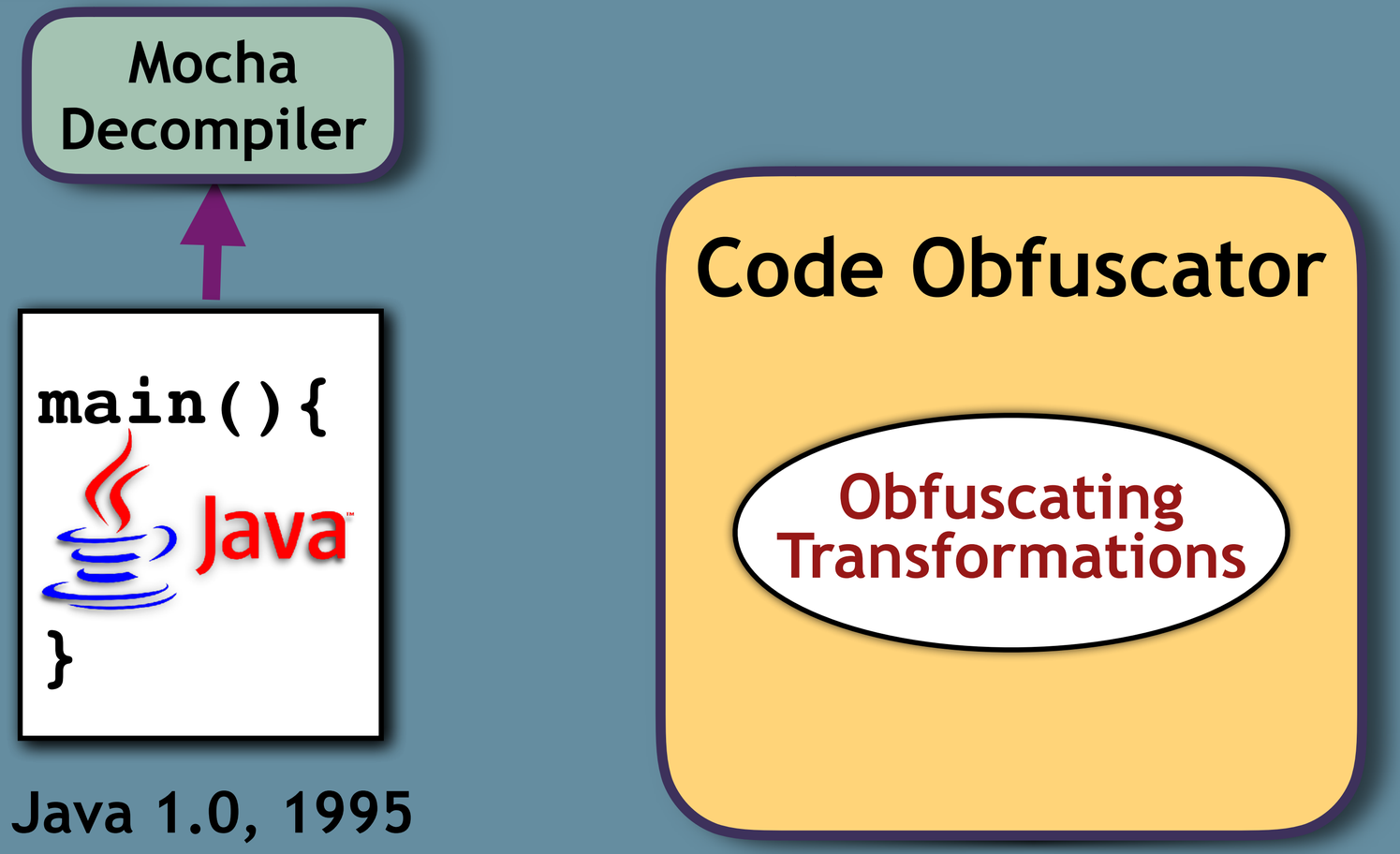


Java 1.0, 1995

A Taxonomy of Obfuscating Transformations

Christian Collberg Clark Thomborson. Douglas Low

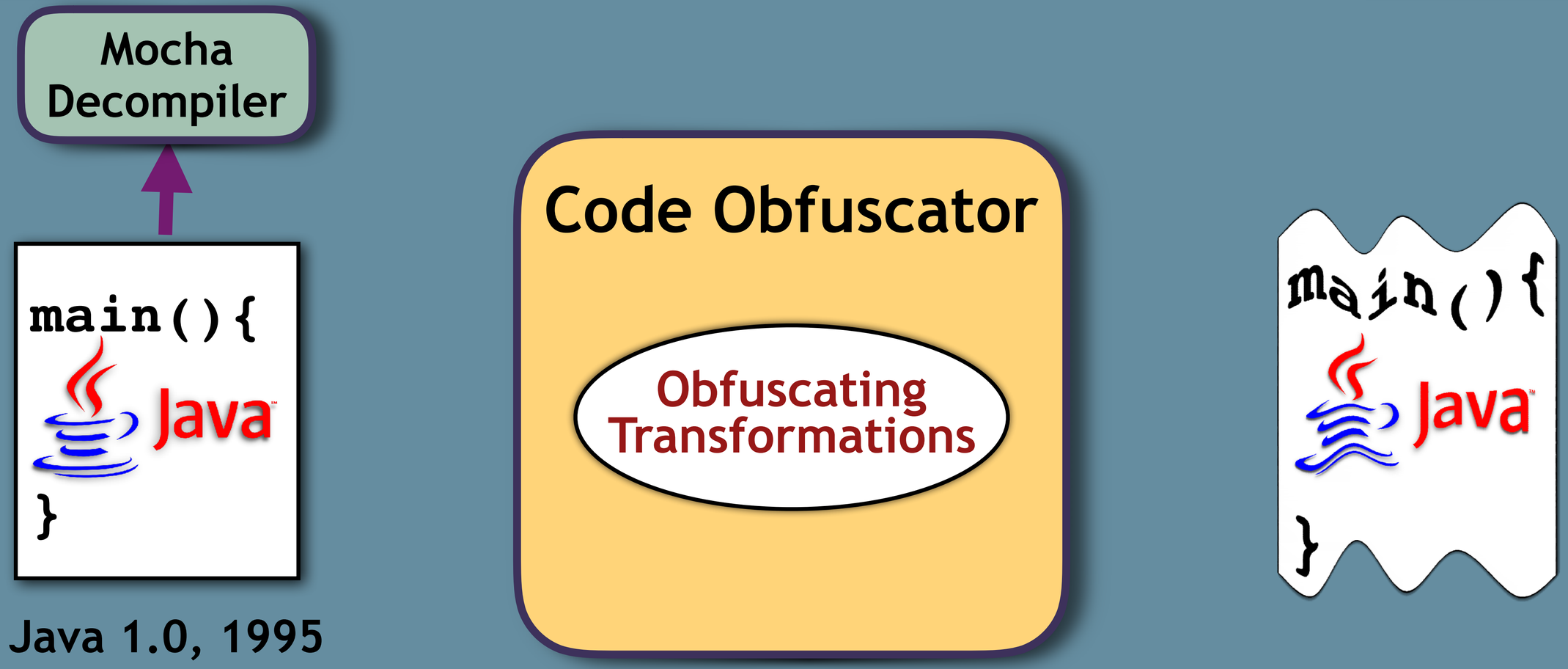
July 1997



A Taxonomy of Obfuscating Transformations

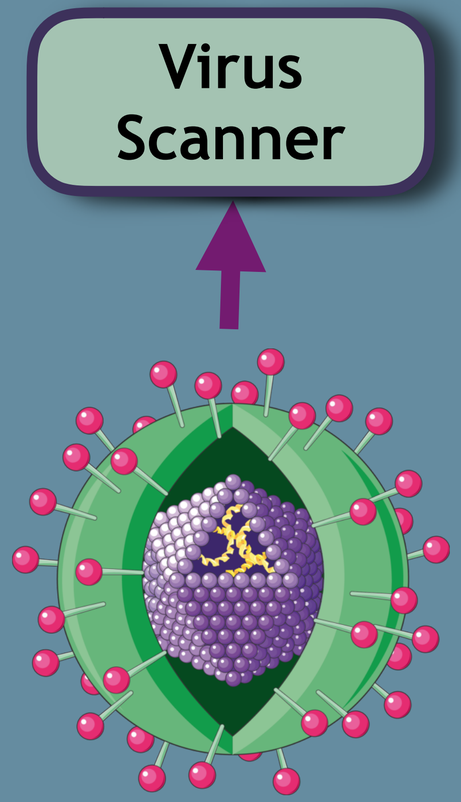
Christian Collberg Clark Thomborson. Douglas Low

July 1997



The Mutation Engine - The Final Nail?

April 1992

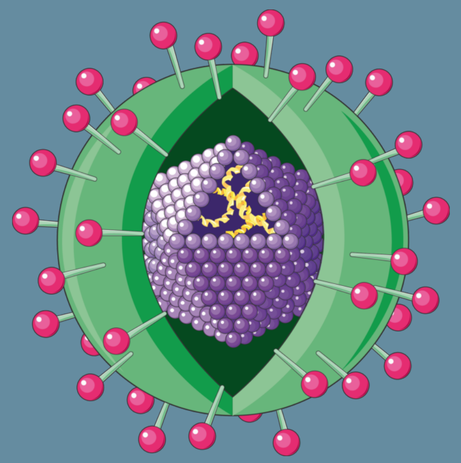


VIRUS BULLETIN

The Mutation Engine - The Final Nail?

April 1992

Virus Scanner



Dark Avenger Mutation Engine

Stealth Transformations

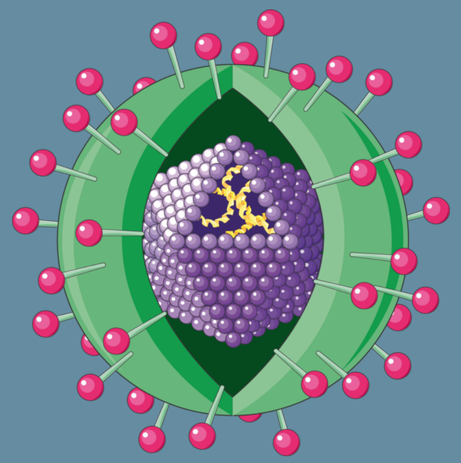
Diversifying Transformations

VIRUS BULLETIN

The Mutation Engine - The Final Nail?

April 1992

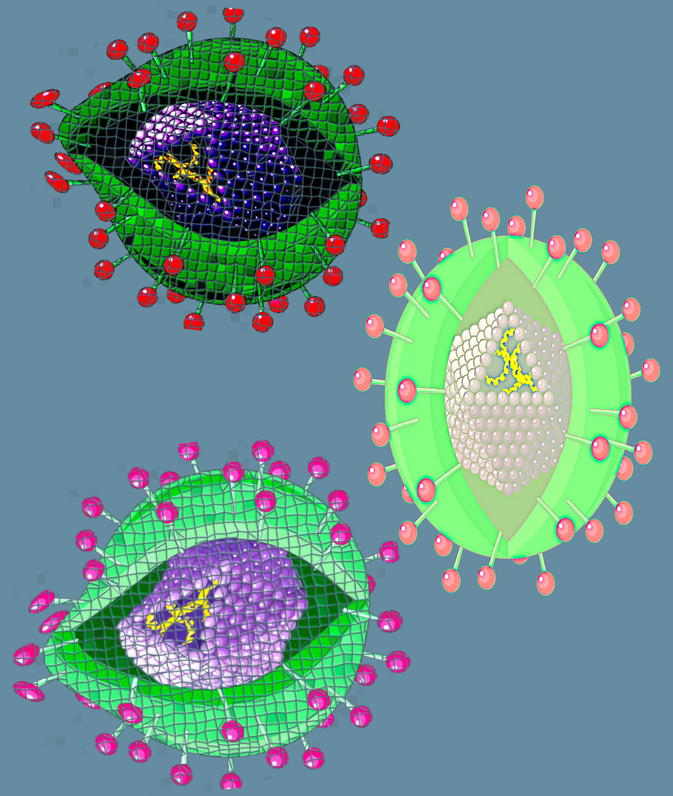
Virus Scanner

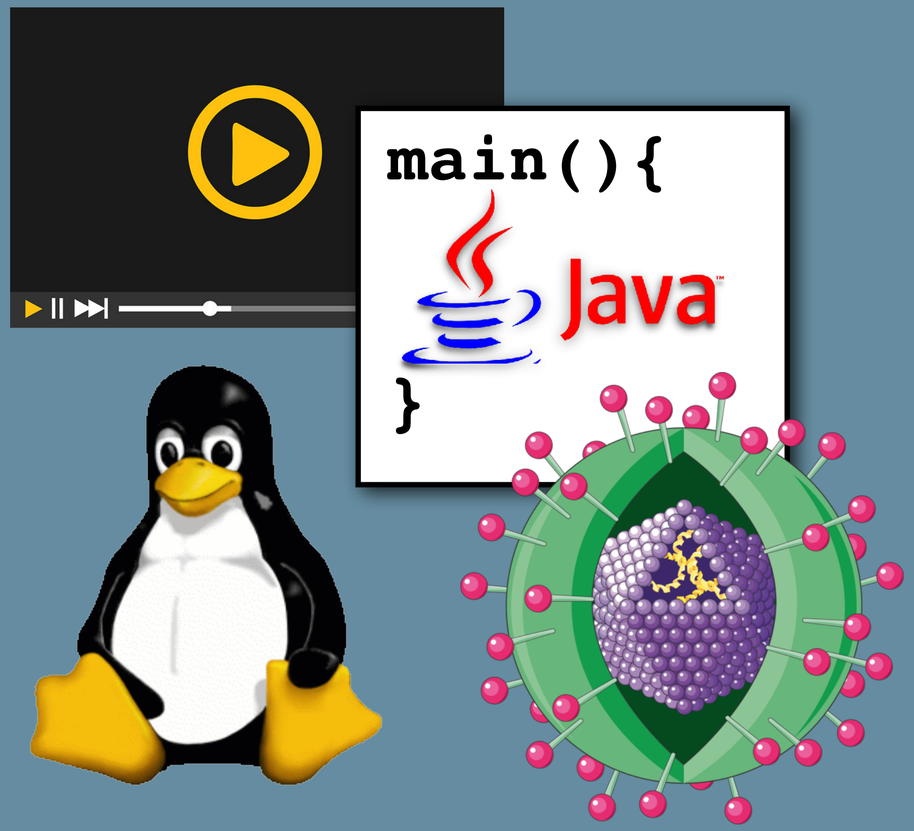


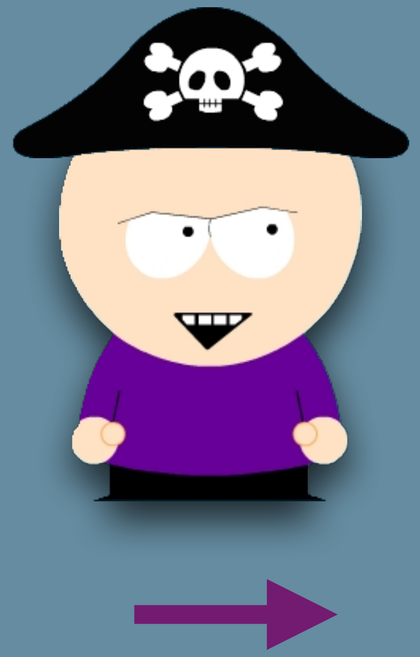
Dark Avenger Mutation Engine

Stealth Transformations

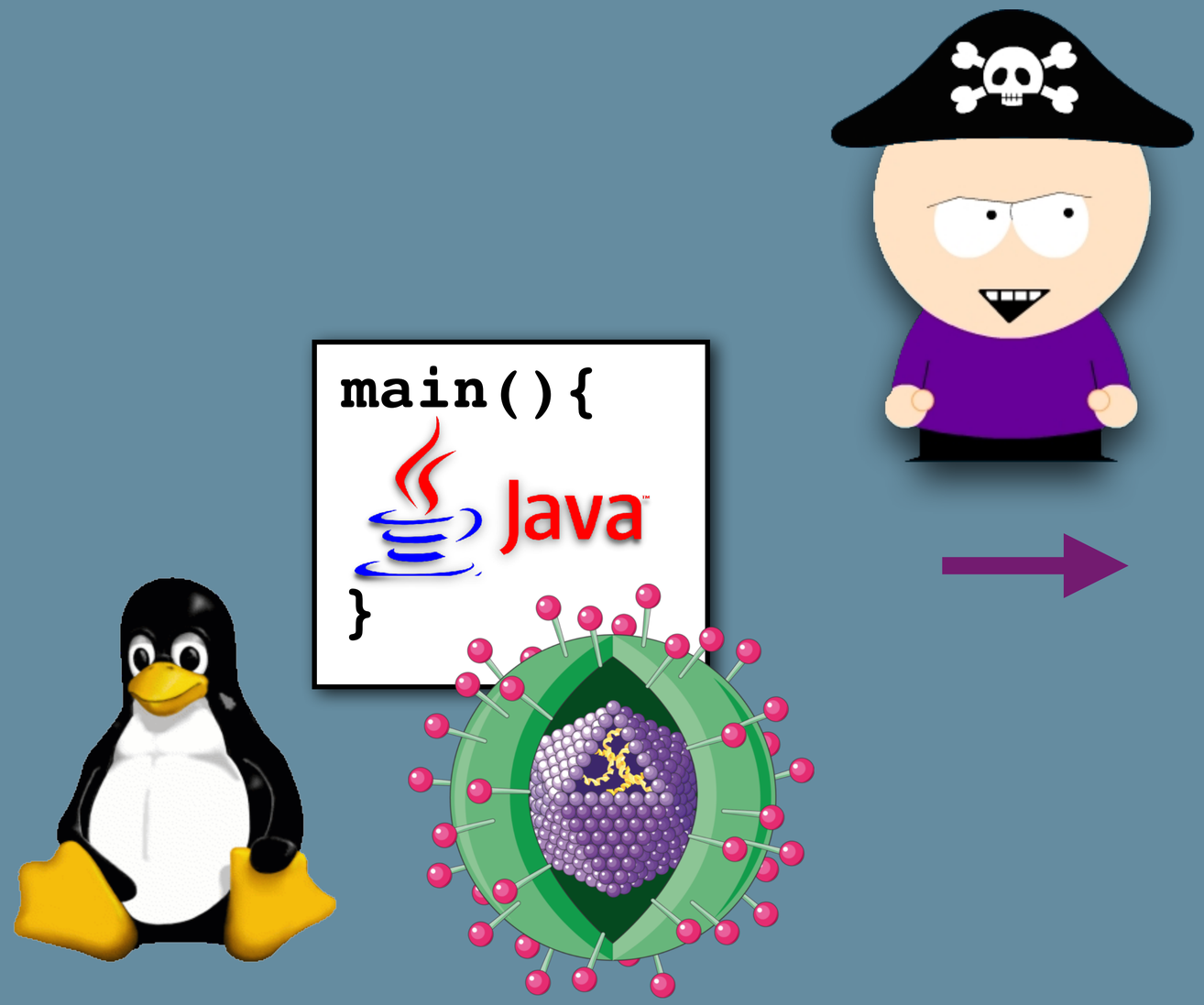
Diversifying Transformations



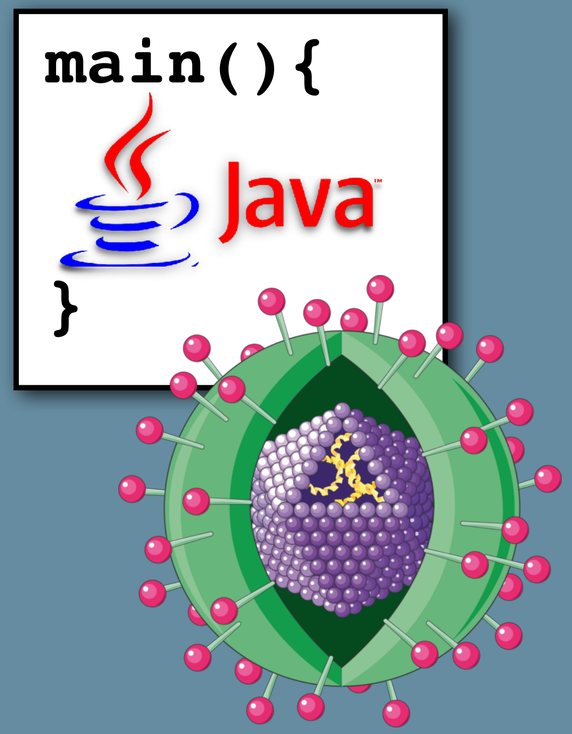


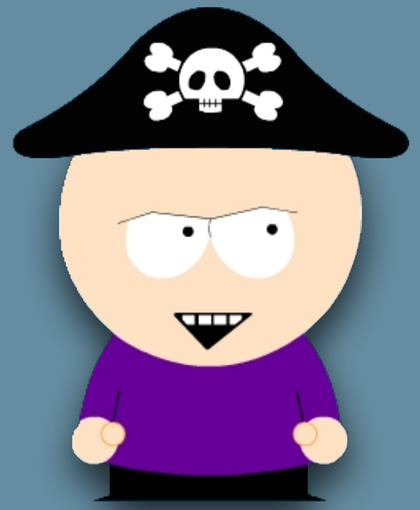


1. Extract keys
2. Recover source
3. Detect
4. Tamper

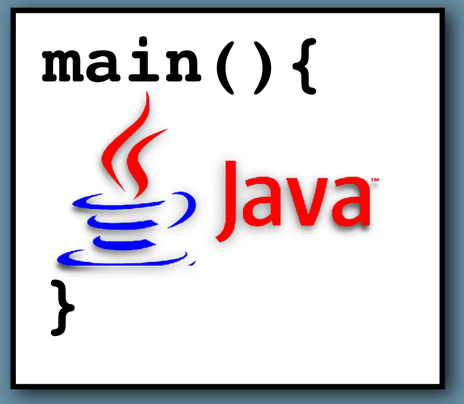


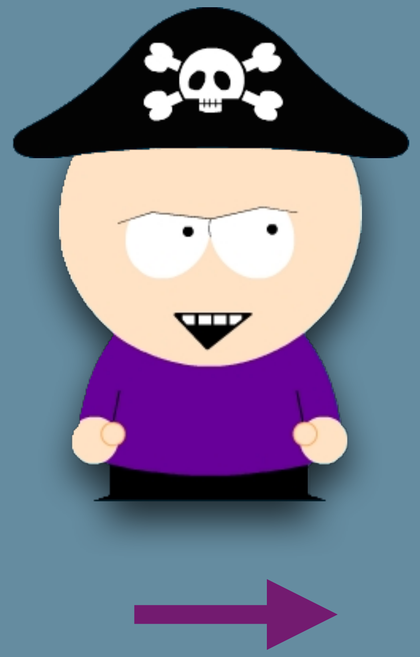
1. Extract keys
2. Recover source
3. Detect
4. Tamper





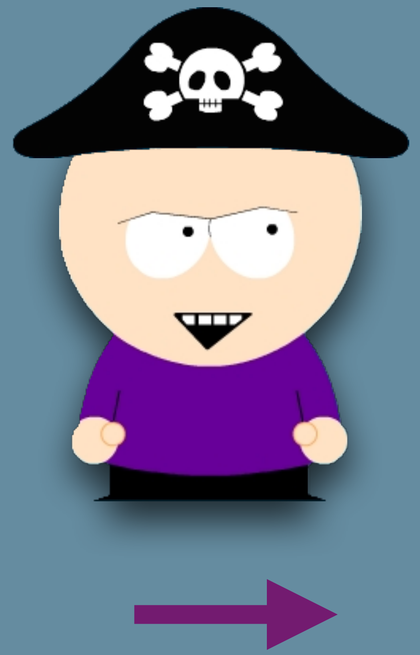
1. Extract keys
2. Recover source
3. Detect
4. Tamper





1. Extract keys
2. Recover source
3. Detect
4. Tamper

?



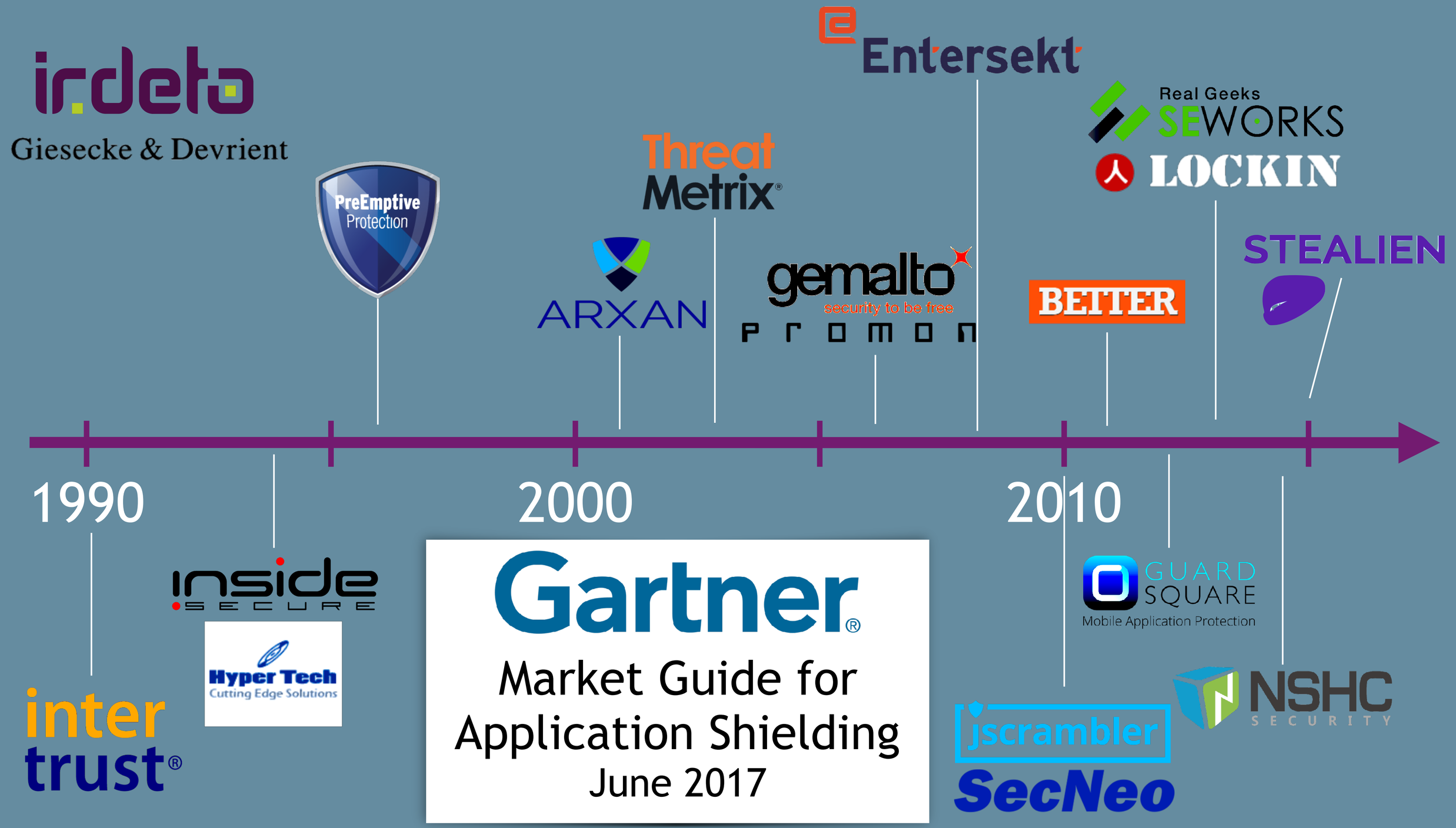
1. Extract keys
2. Recover source
3. Detect
4. Tamper

1. Extract keys
2. Recover source
3. Detect
4. Amber

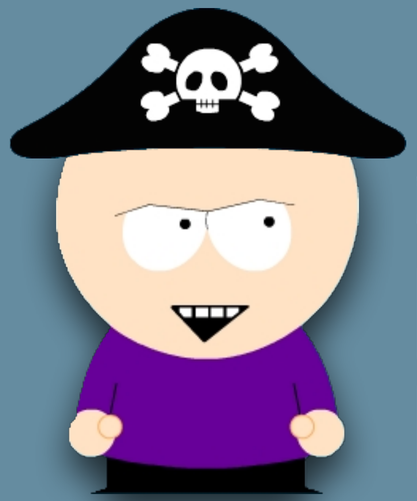
Gartner®

Market Guide for Application Shielding

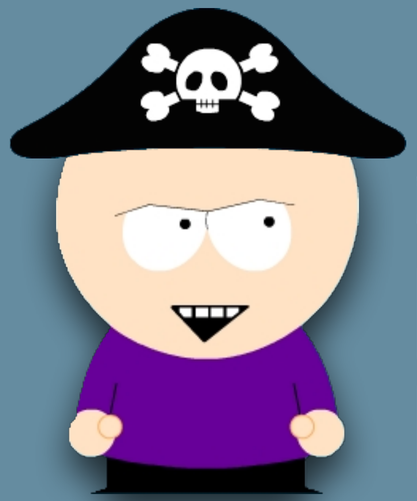
June 2017


irdeto
Giesecke & Devrient
PreEmptive Protection
ARXAN
Threat Metrix
gemalto
security to be free
PROMON
Entersekt
BETTER
Real Geeks
SEWORKS
LOCKIN
STEALIEN
1990
2000
2010
inside SECURE
Hyper Tech
Cutting Edge Solutions
inter trust
Gartner
Market Guide for Application Shielding
June 2017
jscrambler
SecNeo
GUARD SQUARE
Mobile Application Protection
NSHC SECURITY

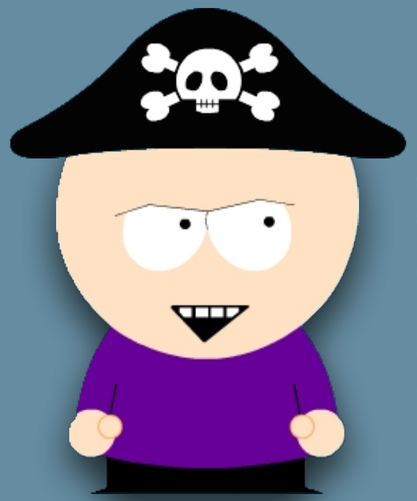
9:41
Tuesday, September 12

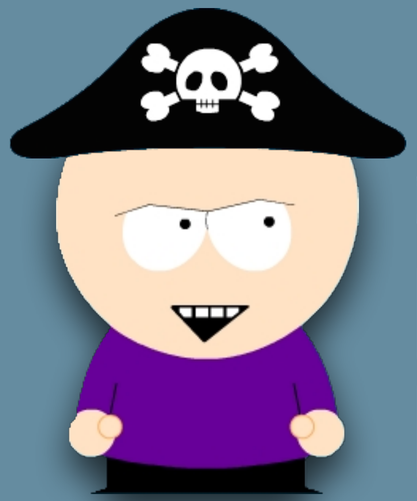


9:41
Tuesday, September 12

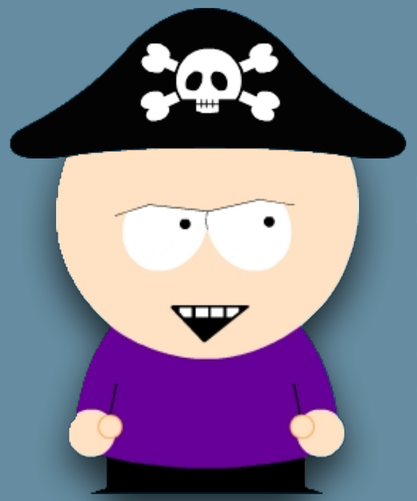


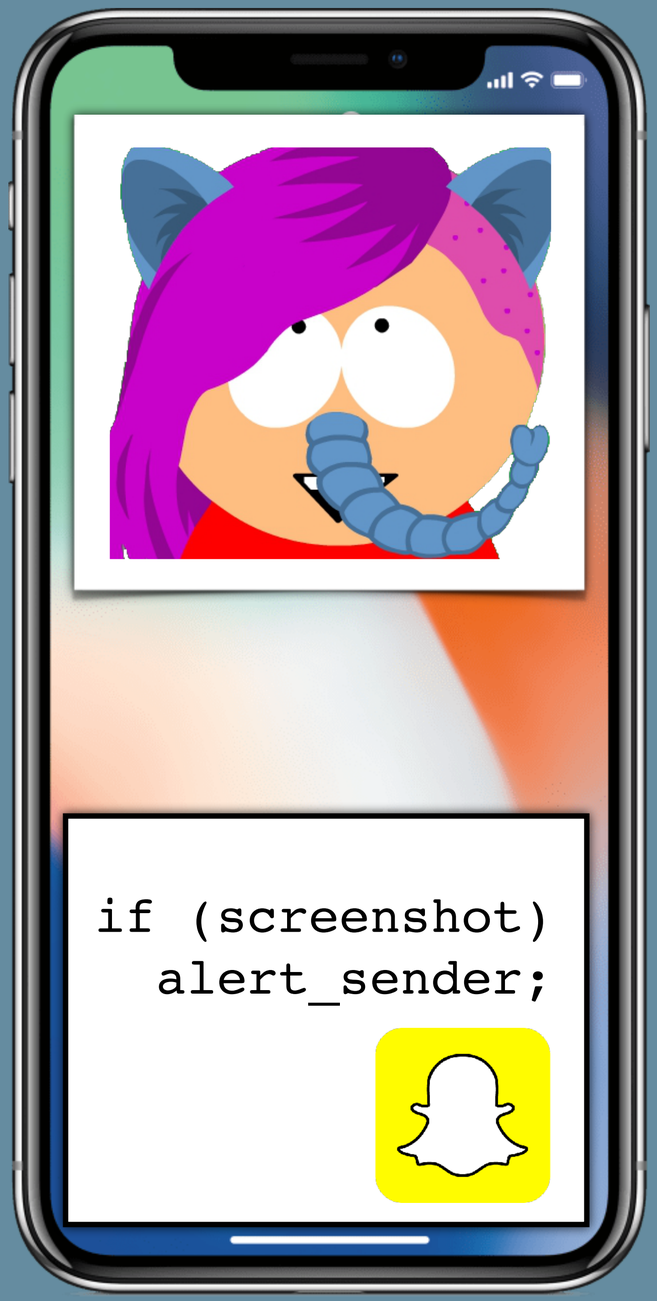
9:41
Tuesday, September 12



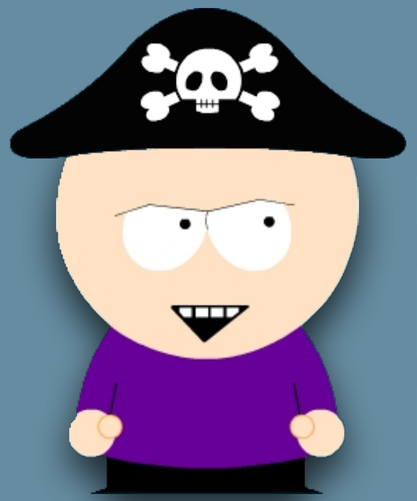


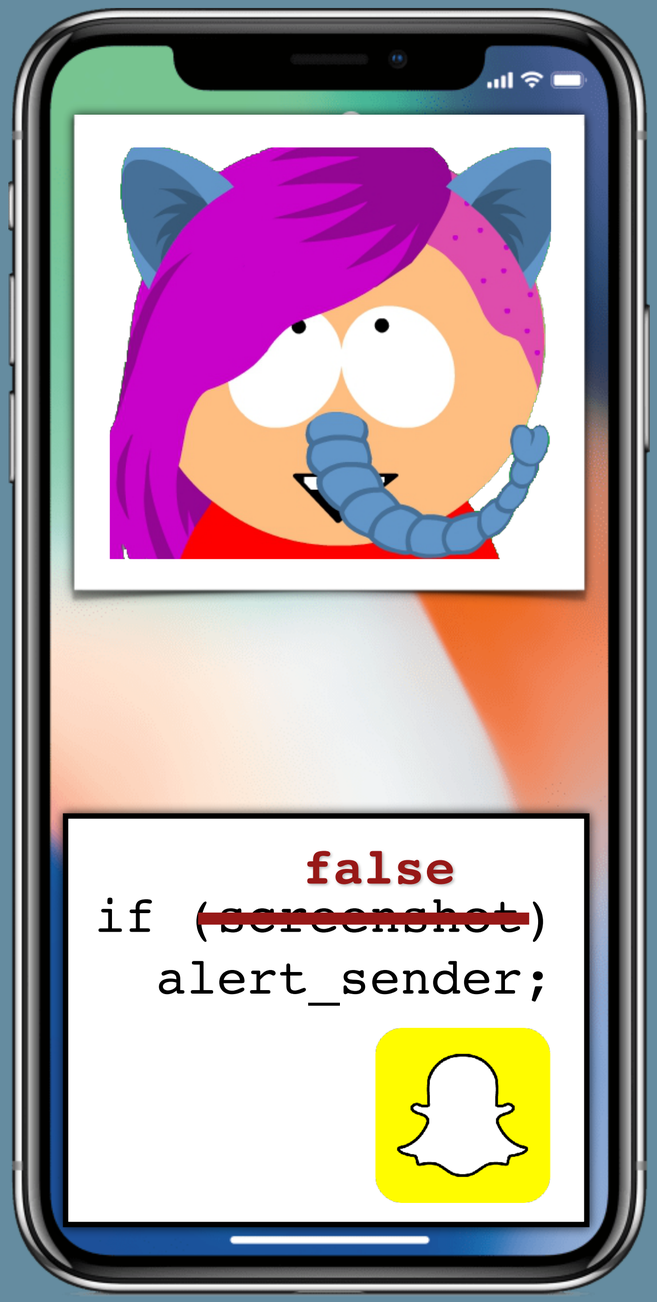
snapchat
Bob took a screenshot!



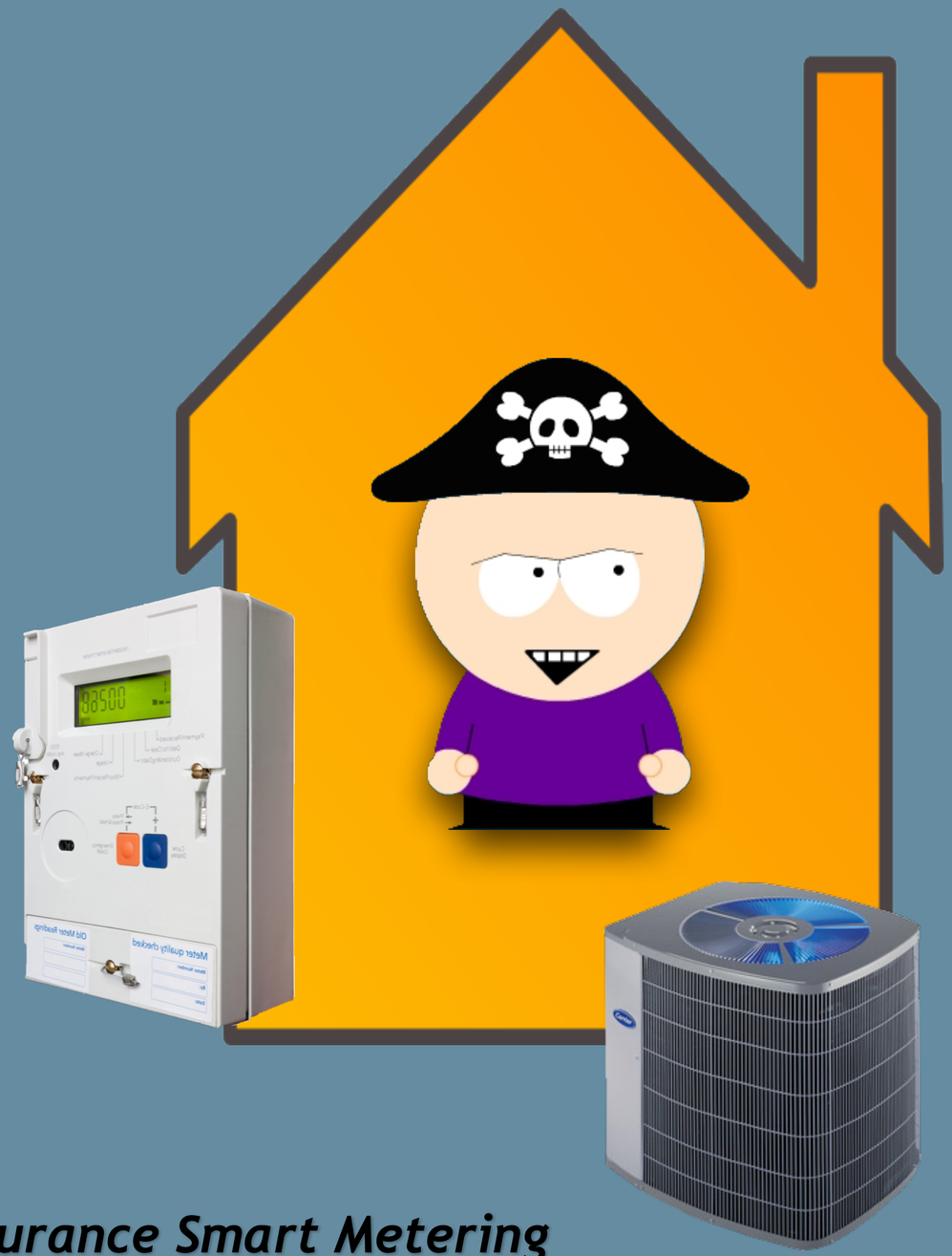

if (screenshot)
alert_sender;

snapchat
Bob took a screenshot!

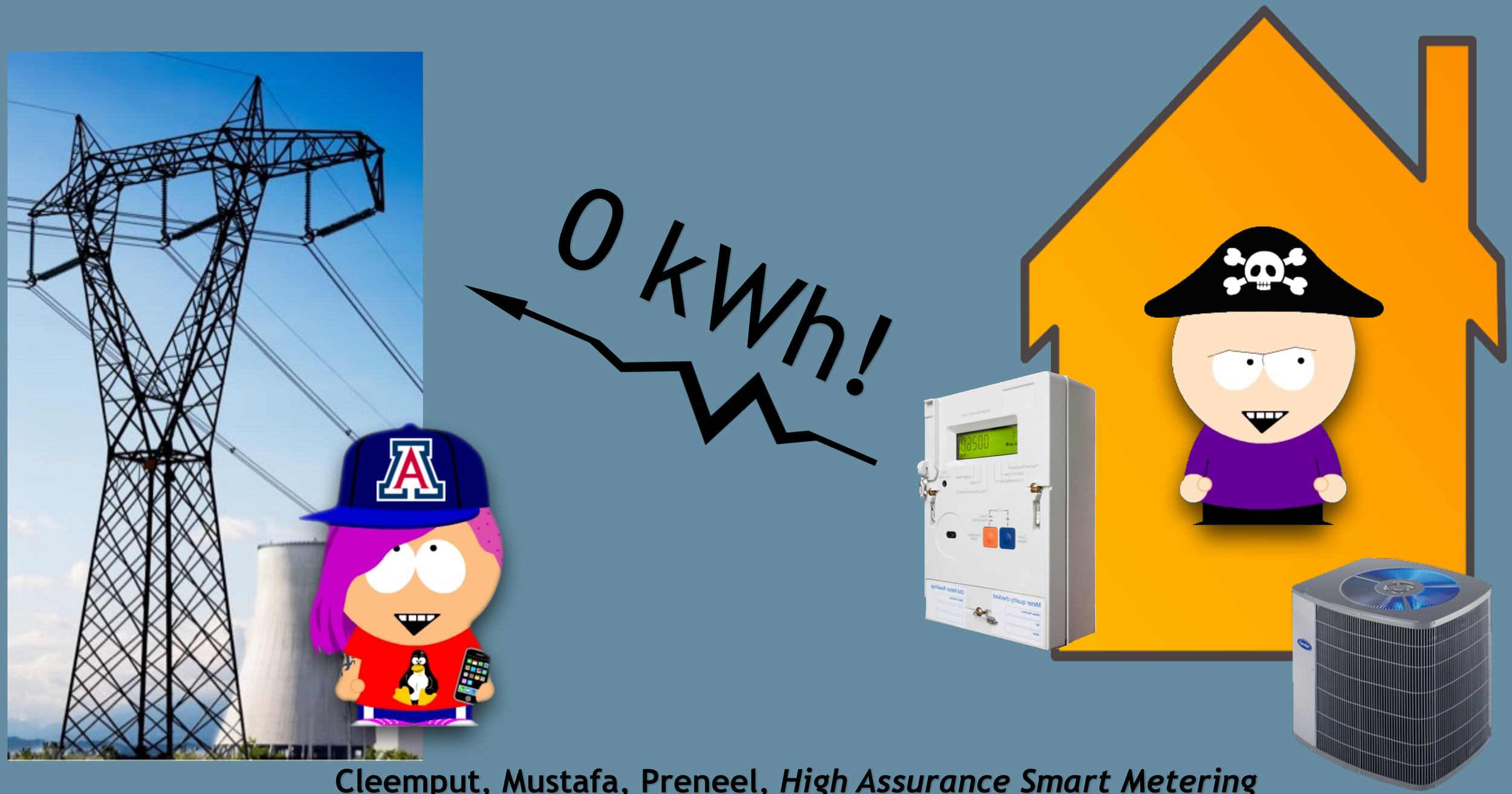



false
if (~~screenshot~~)
alert_sender;

snapchat
Bob took a screenshot!
if (screenshot) alert_sender;



Cleemput, Mustafa, Preneel, *High Assurance Smart Metering*

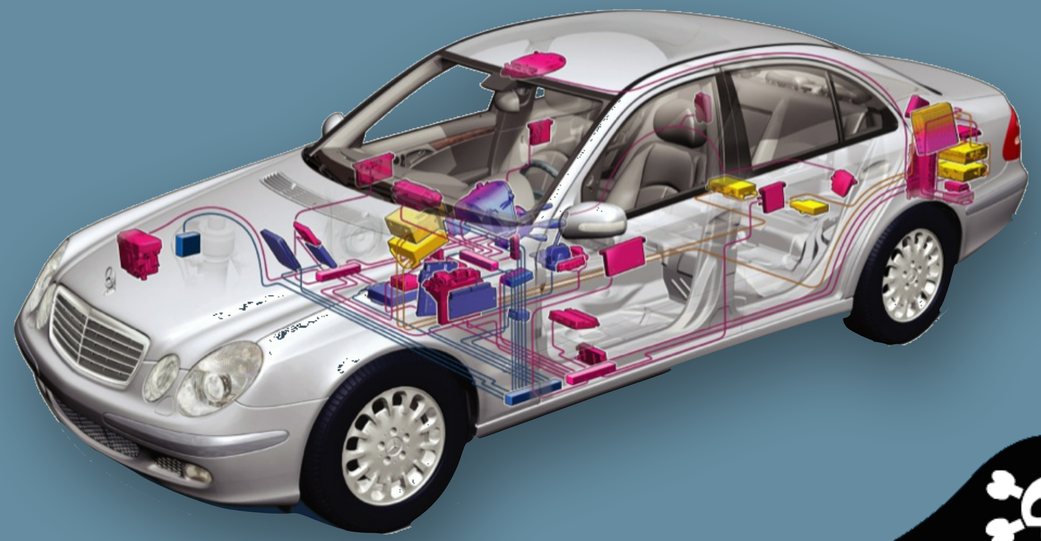


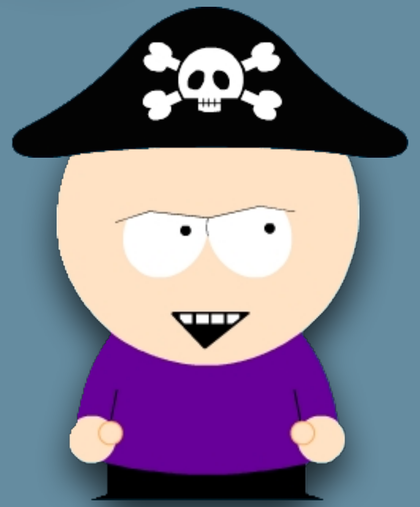
Cleemput, Mustafa, Preneel, *High Assurance Smart Metering*

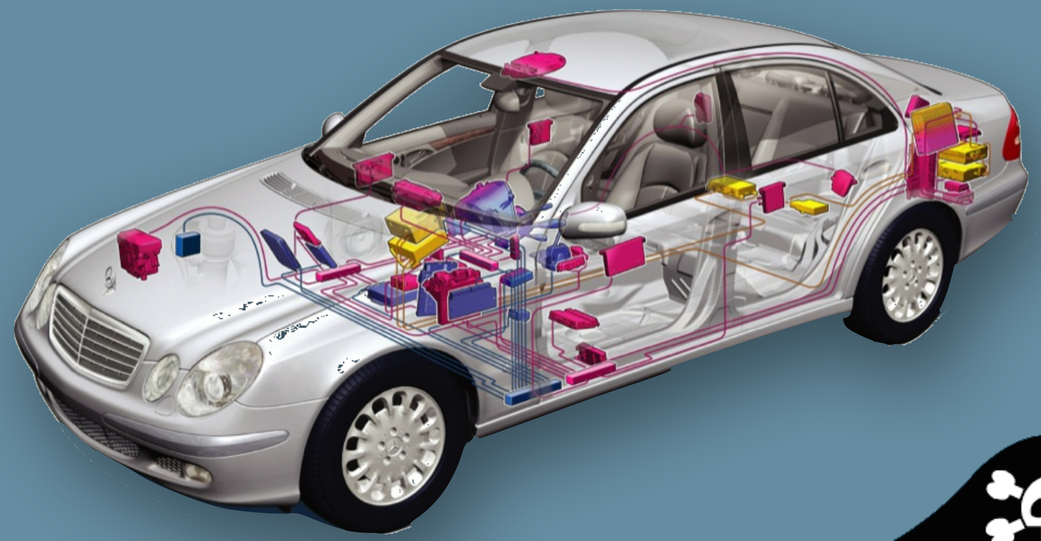
Cleemput, Mustafa, Preneel, *High Assurance Smart Metering*

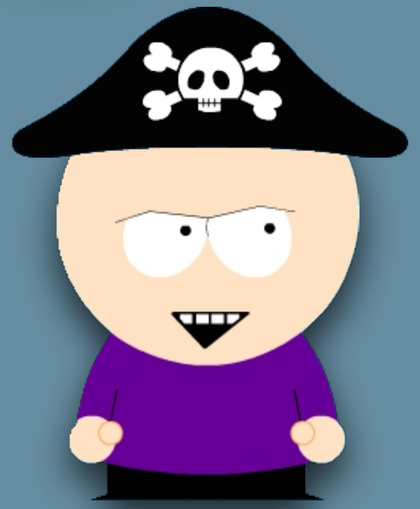
Cleemput, Mustafa, Preneel, *High Assurance Smart Metering*

Cleemput, Mustafa, Preneel, *High Assurance Smart Metering*

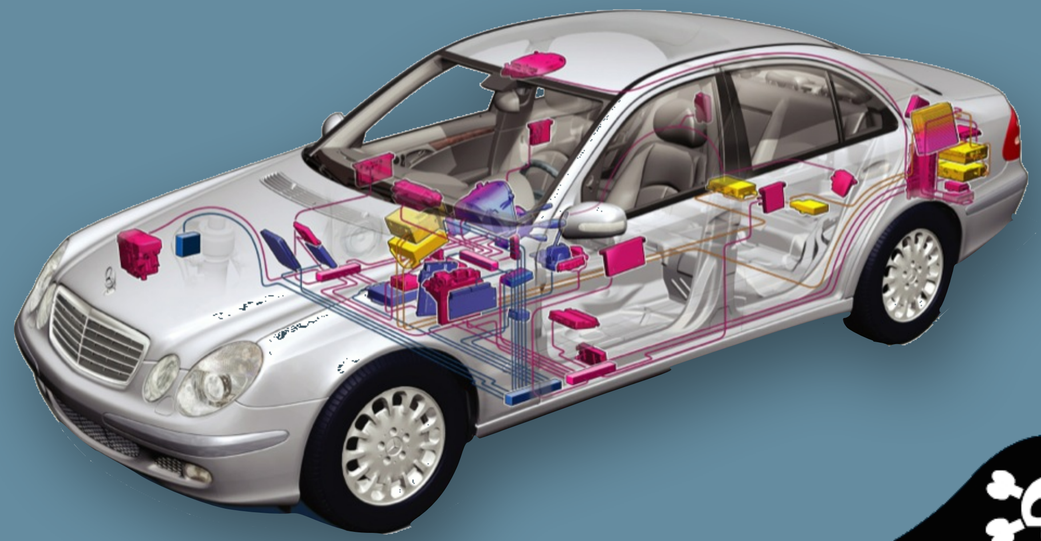


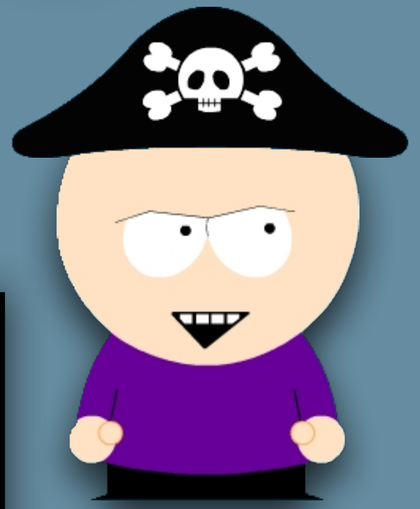






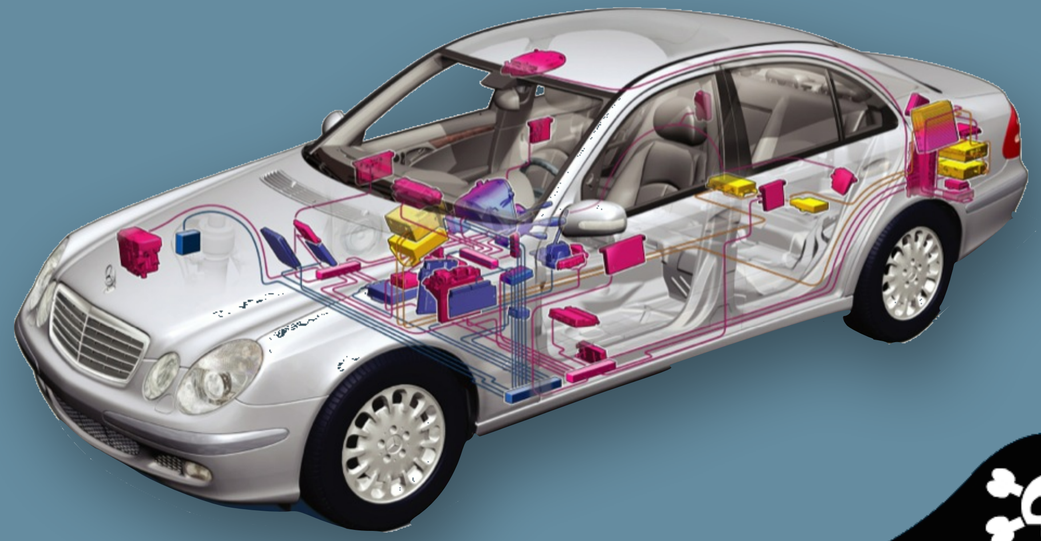
50000

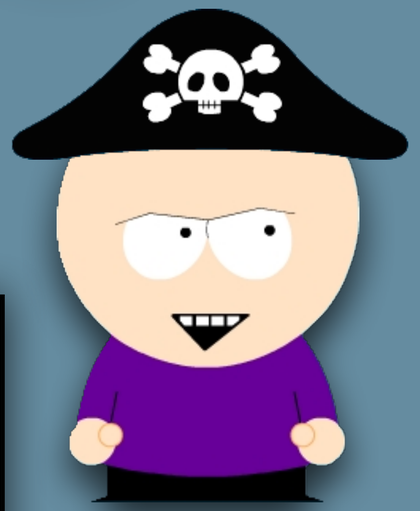




true

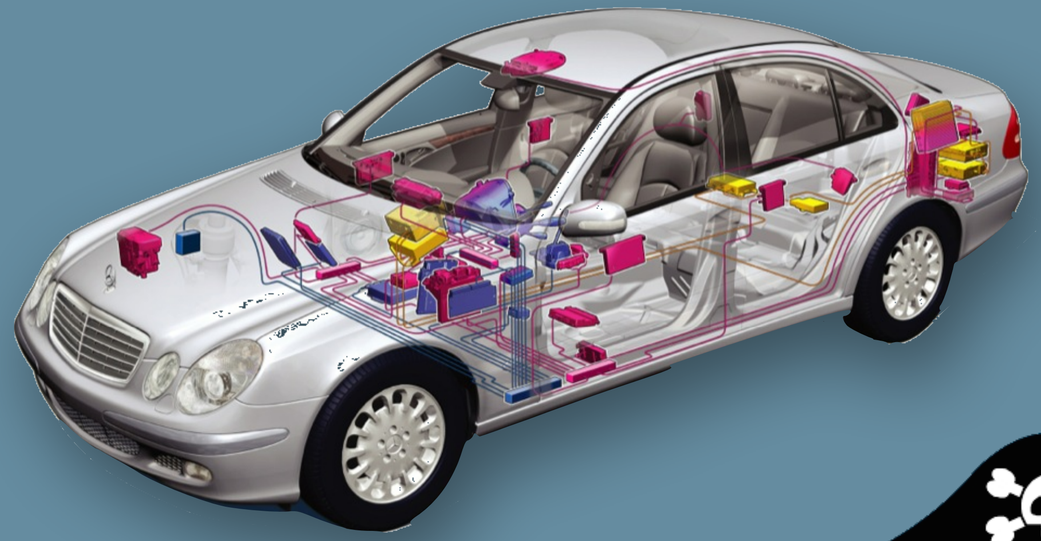
```
if (~~has_paid_for_upgrade()~~)
    allow_ridiculous_mode = 1;
```

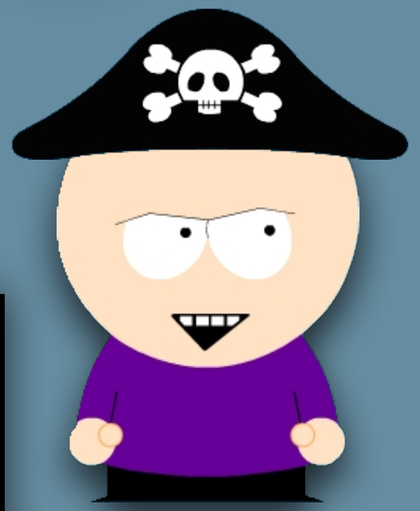




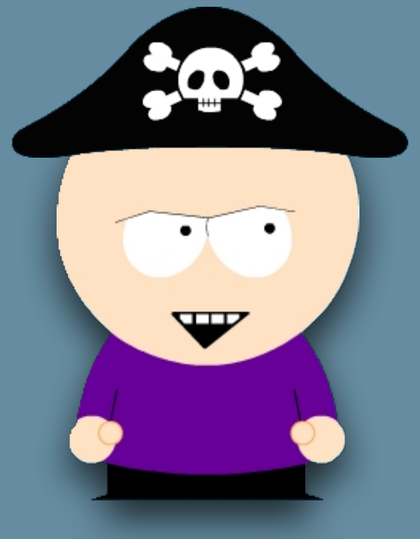
false

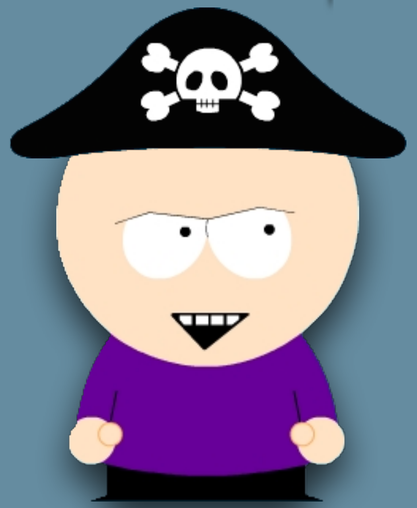
```
if (reported_stolen())
    send_GPS_coords();
```



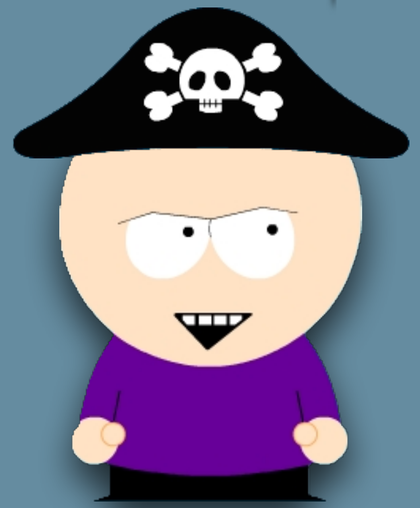


```
battery_management() {
    valuable trade secrets
}
```

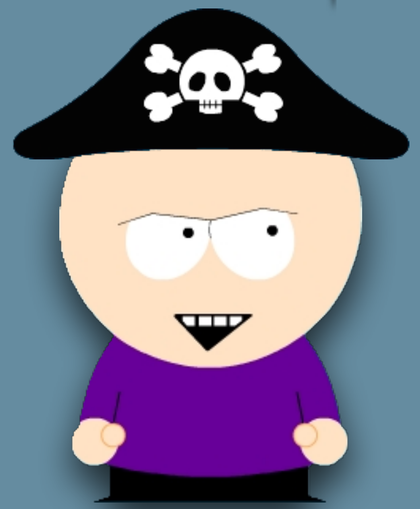




Extract Intellectual Property



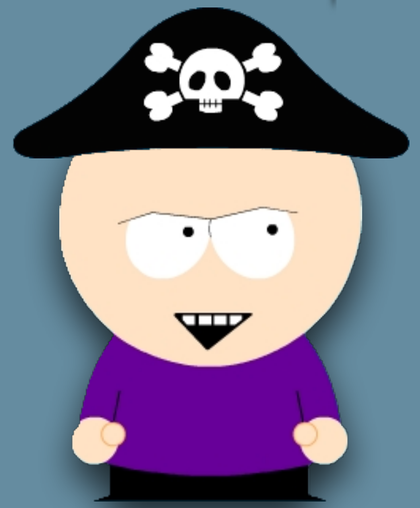
Extract Intellectual Property



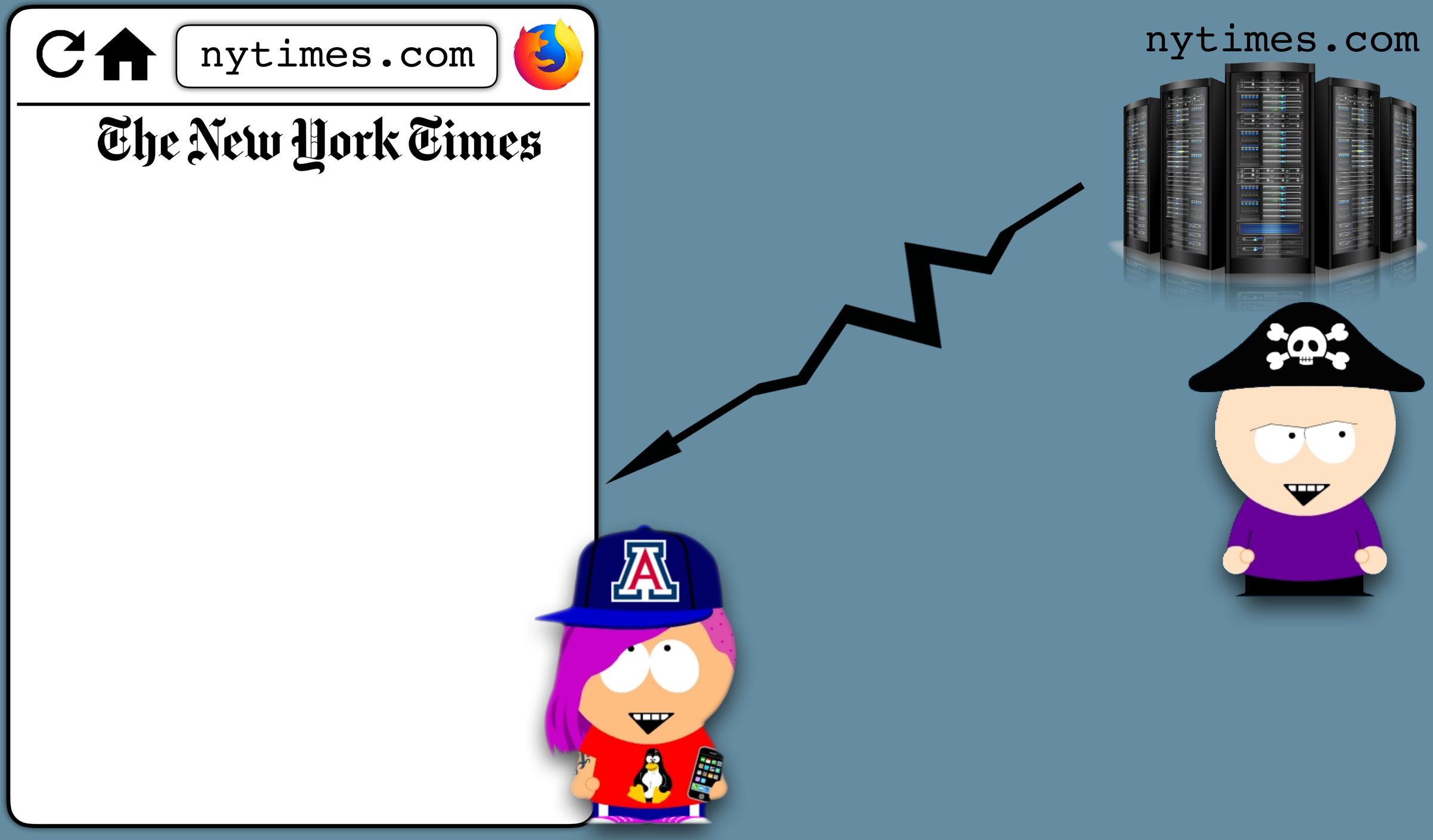
Tamper to Cause Harm

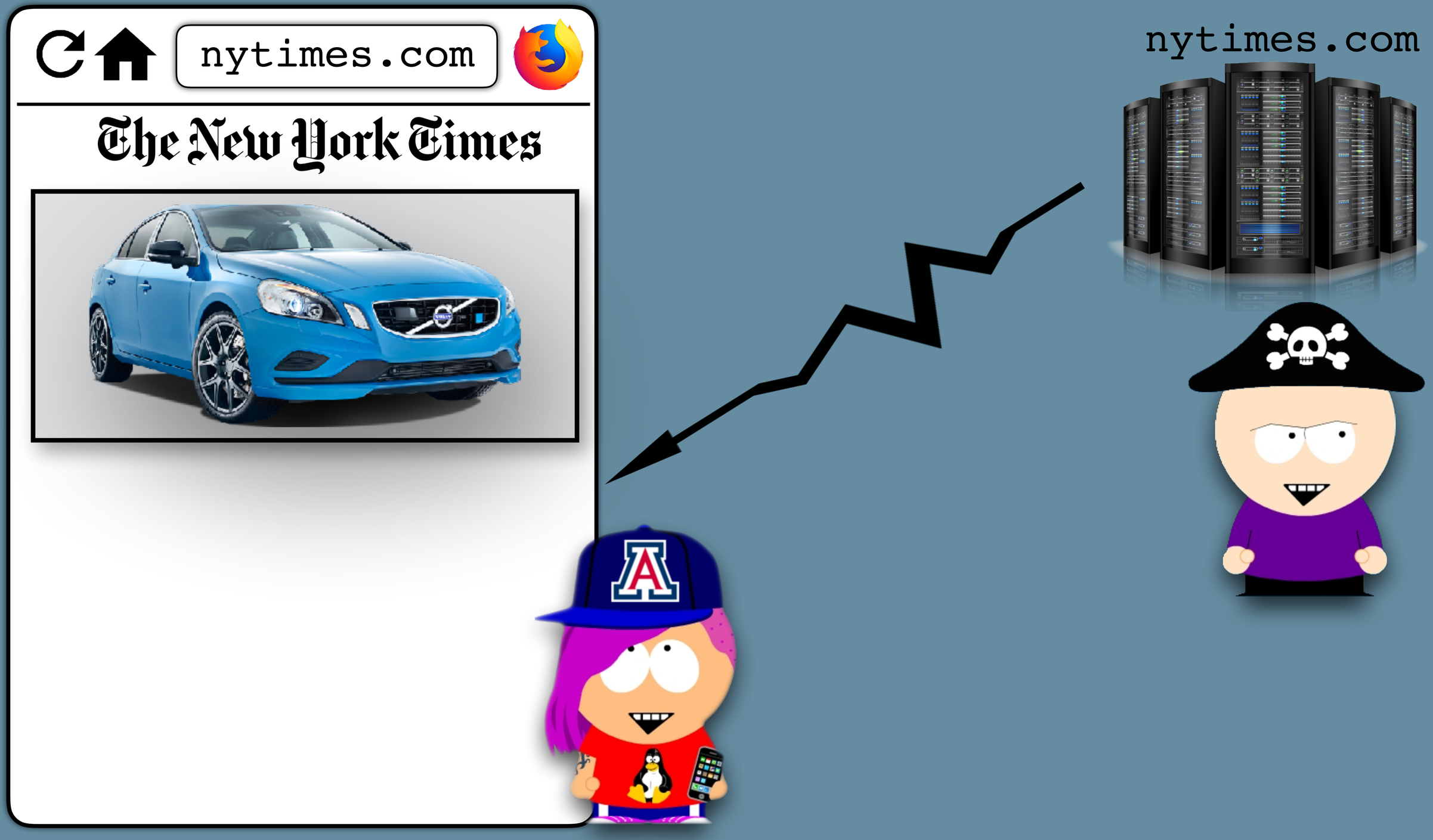
Extract Credentials

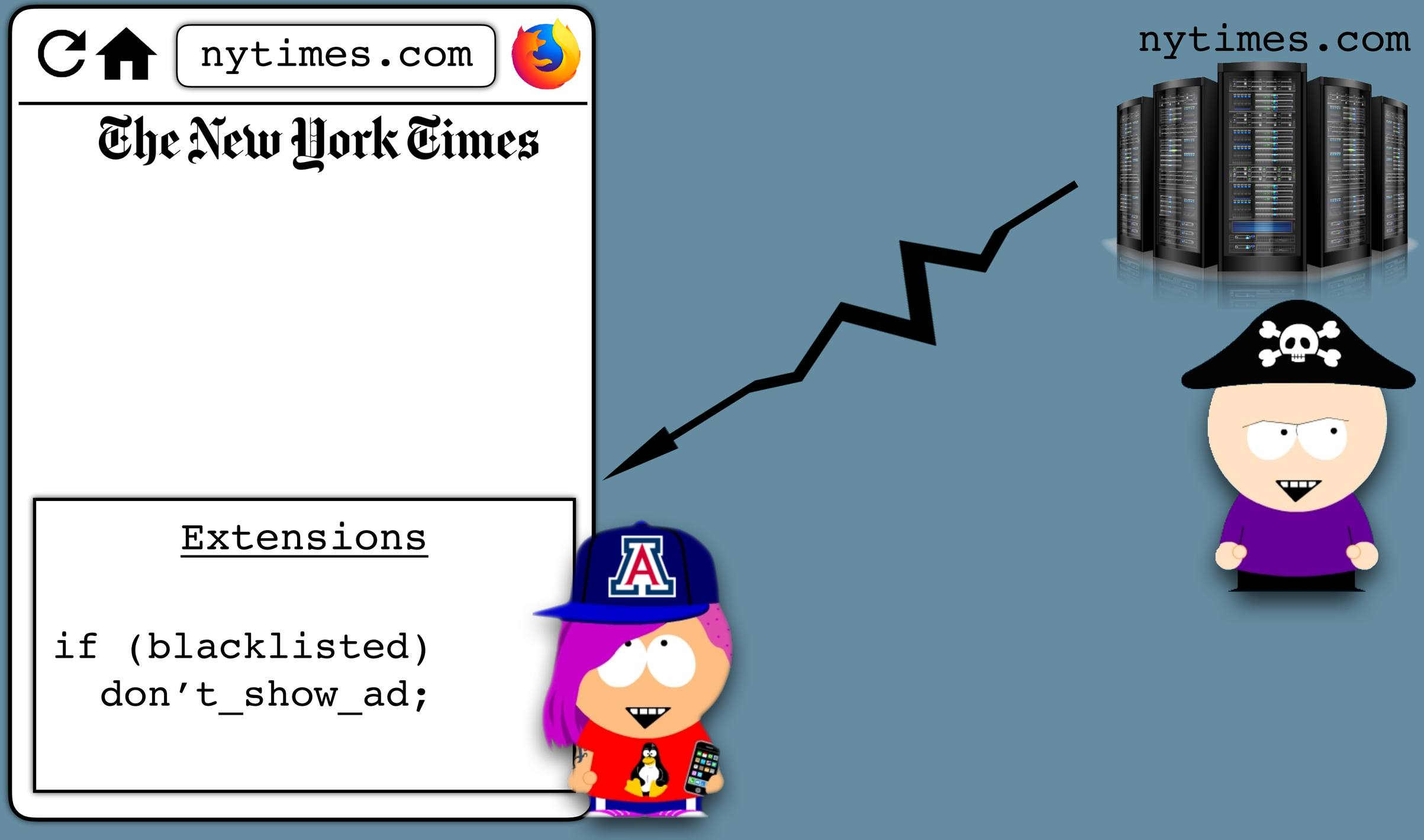
Extract Intellectual Property



Tamper to Cause Harm


nytimes.com
The New York Times
nytimes.com


nytimes.com
The New York Times
nytimes.com

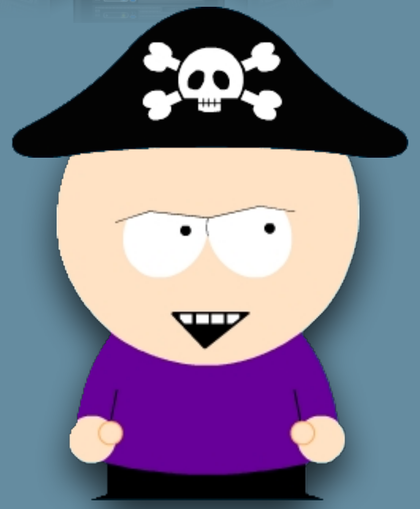

nytimes.com
The New York Times
nytimes.com
Extensions
if (blacklisted)
don't_show_ad;

nytimes.com

The New York Times

```
<script>
 if (ad_blocked)
   don't_show_content;
</script>
```

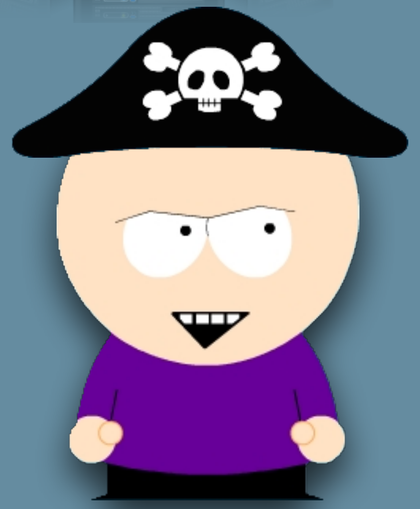
nytimes.com



nytimes.com

The New York Times

nytimes.com



Extensions

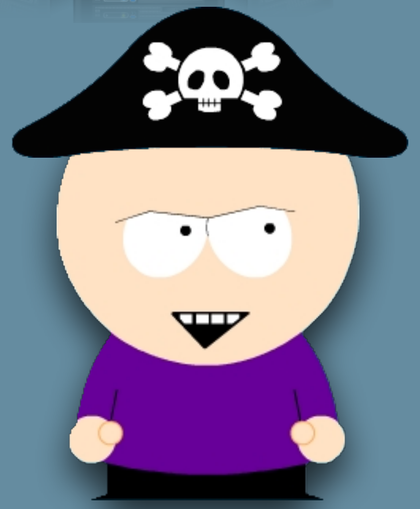
```
if (has_pattern(
   "if (ad_blocked)
      don't_show_content;"
))
   don't_show_ad;
```

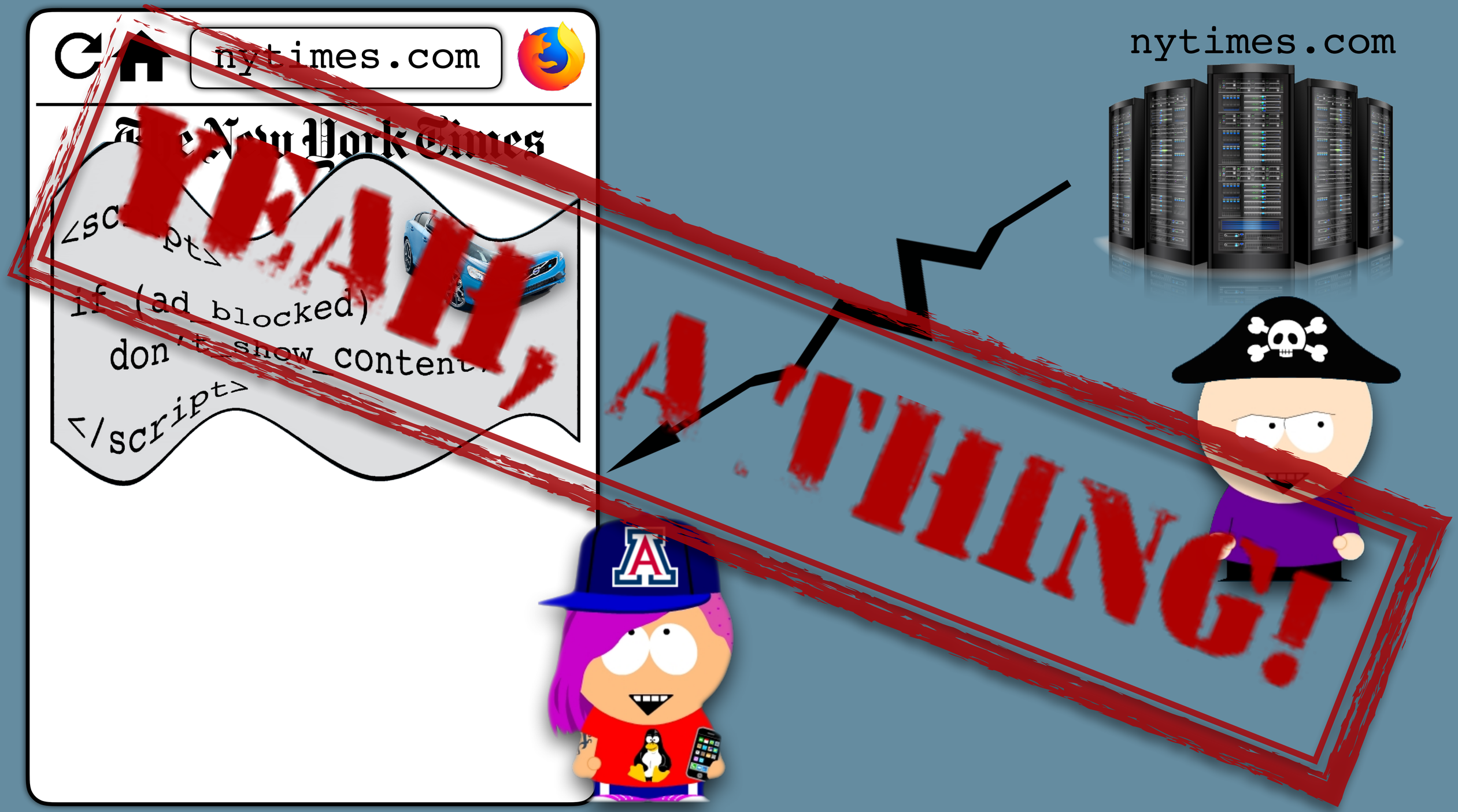
nytimes.com

The New York Times

```
<script>
if (ad_blocked)
   don't_show_content;
</script>
```

nytimes.com



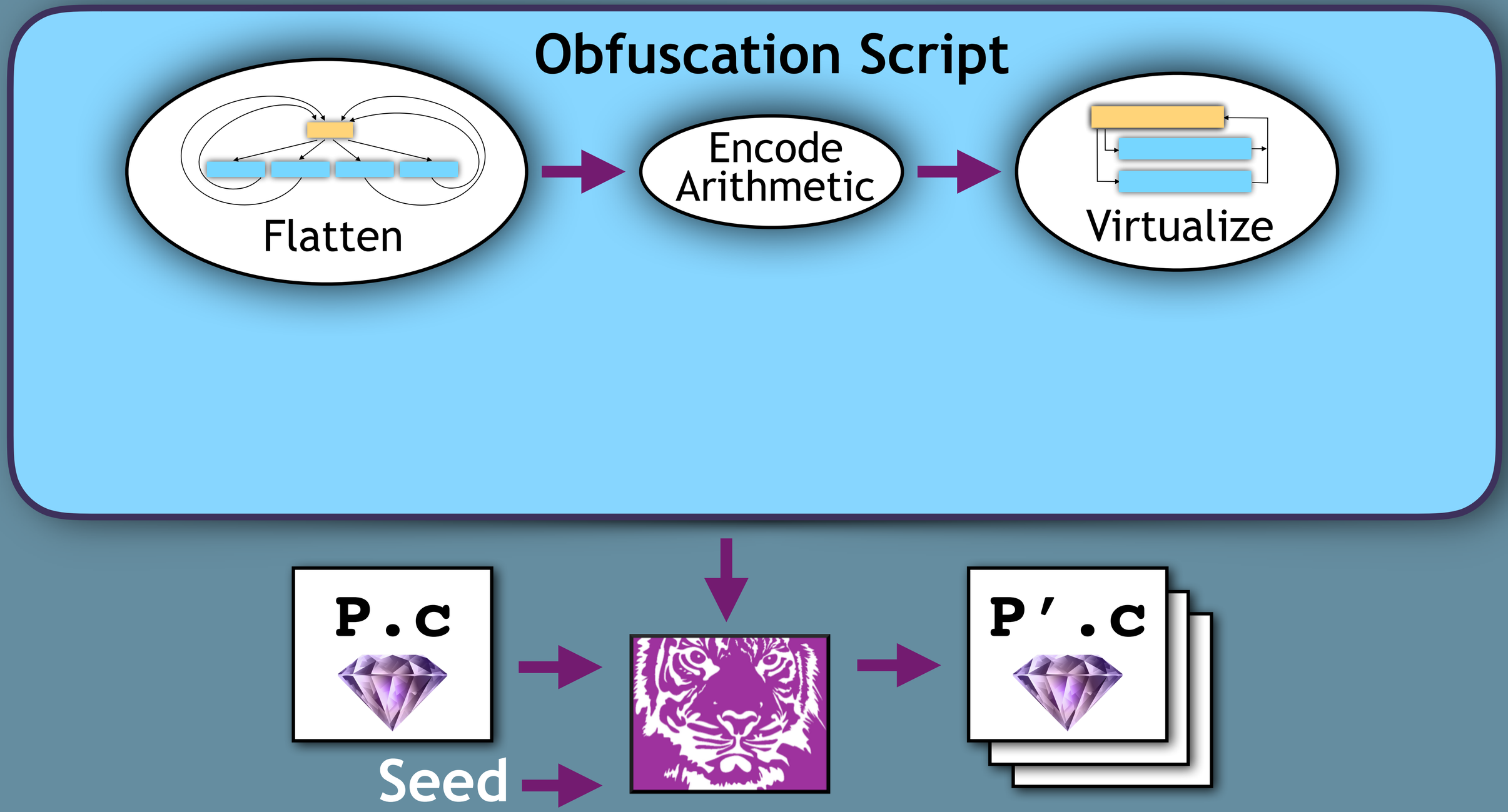

nytimes.com
The New York Times
<sc pt>
if (ad_blocked)
don't show_content
</script>
YEAH, ANYTHING!
nytimes.com

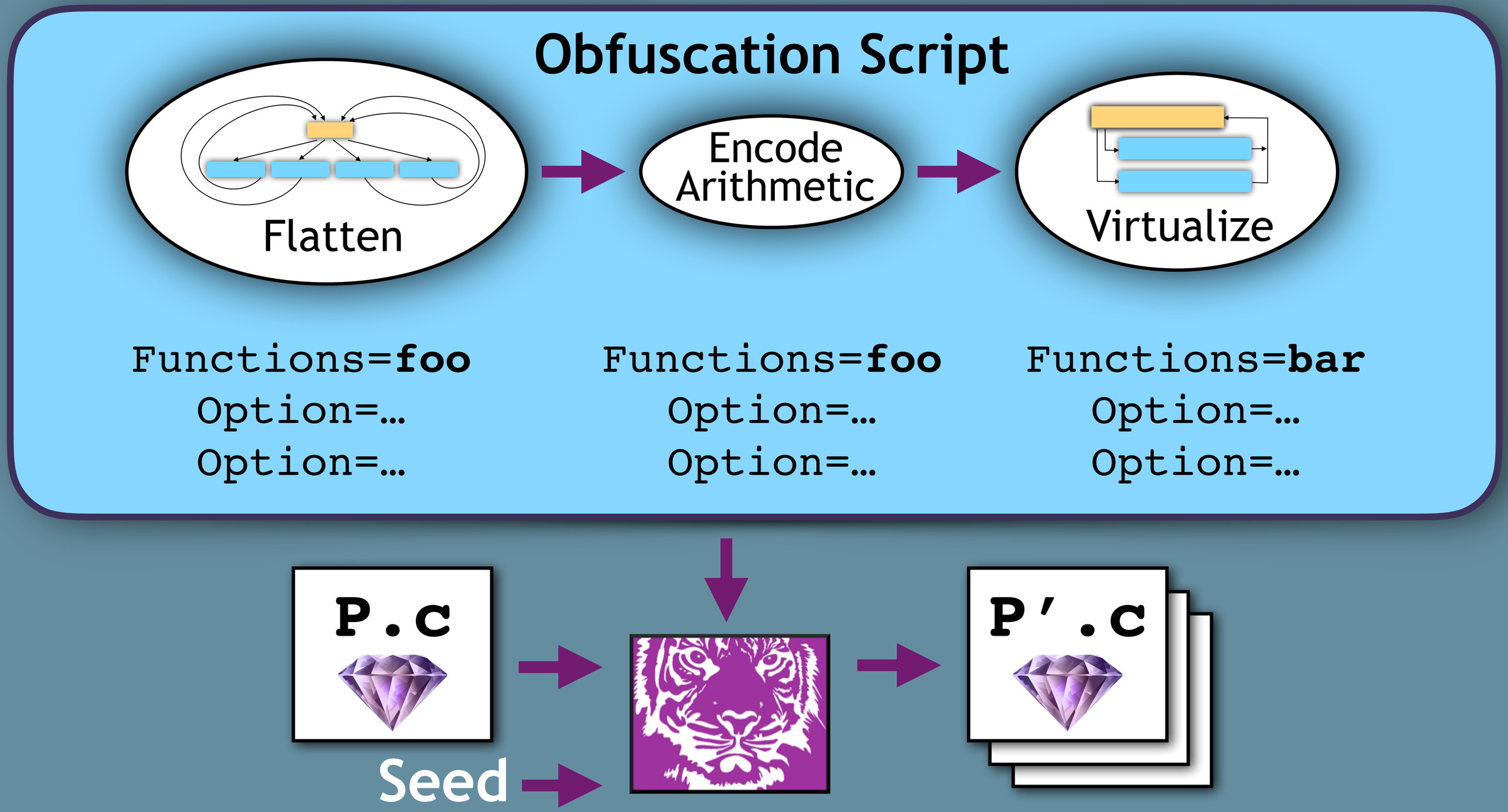
Tiger Squad

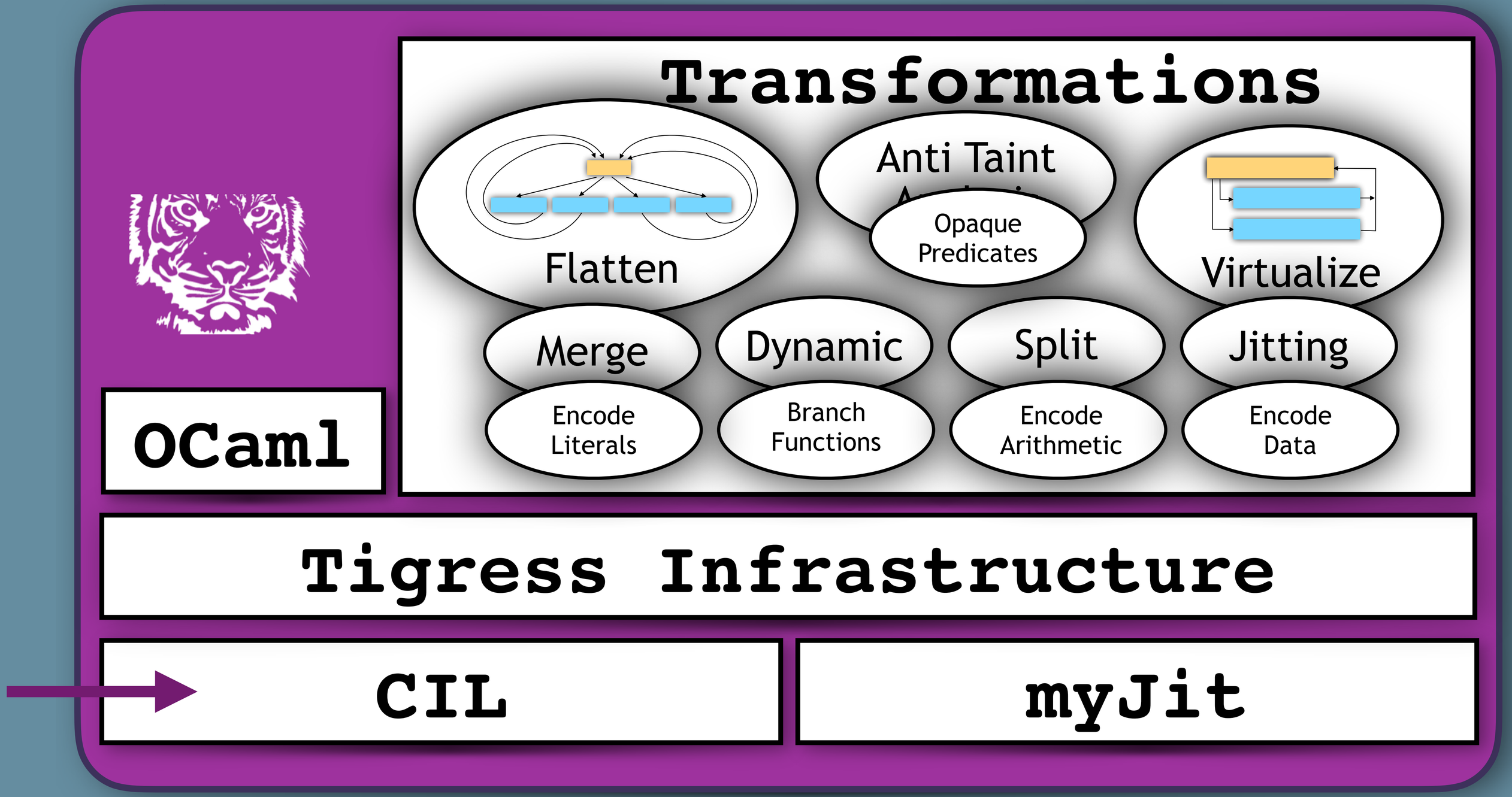
Tigress Design

P.c
P′.c

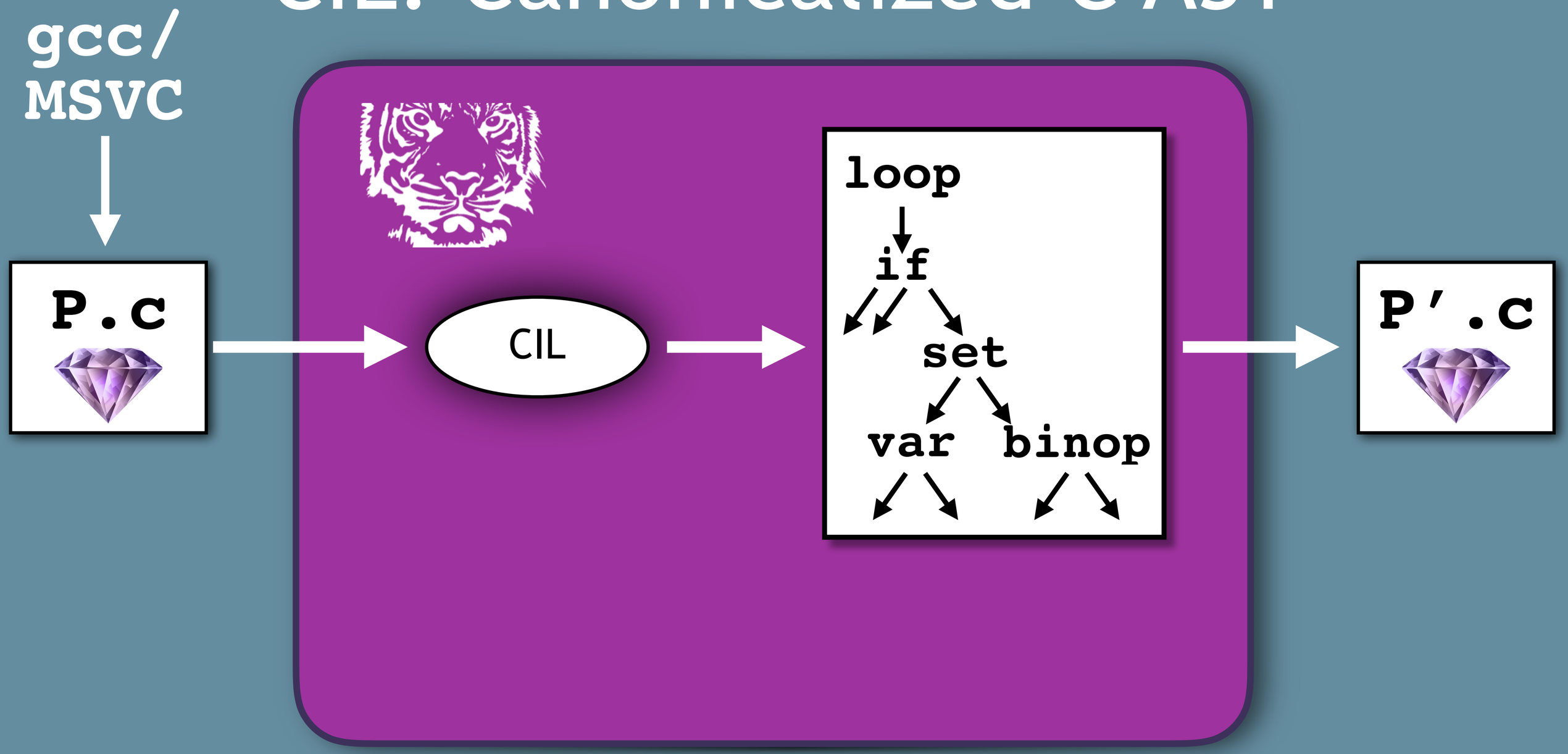
P.c
Seed
P′.c


Obfuscation Script
Flatten
Encode Arithmetic
Virtualize
P.c
Seed
P'.c

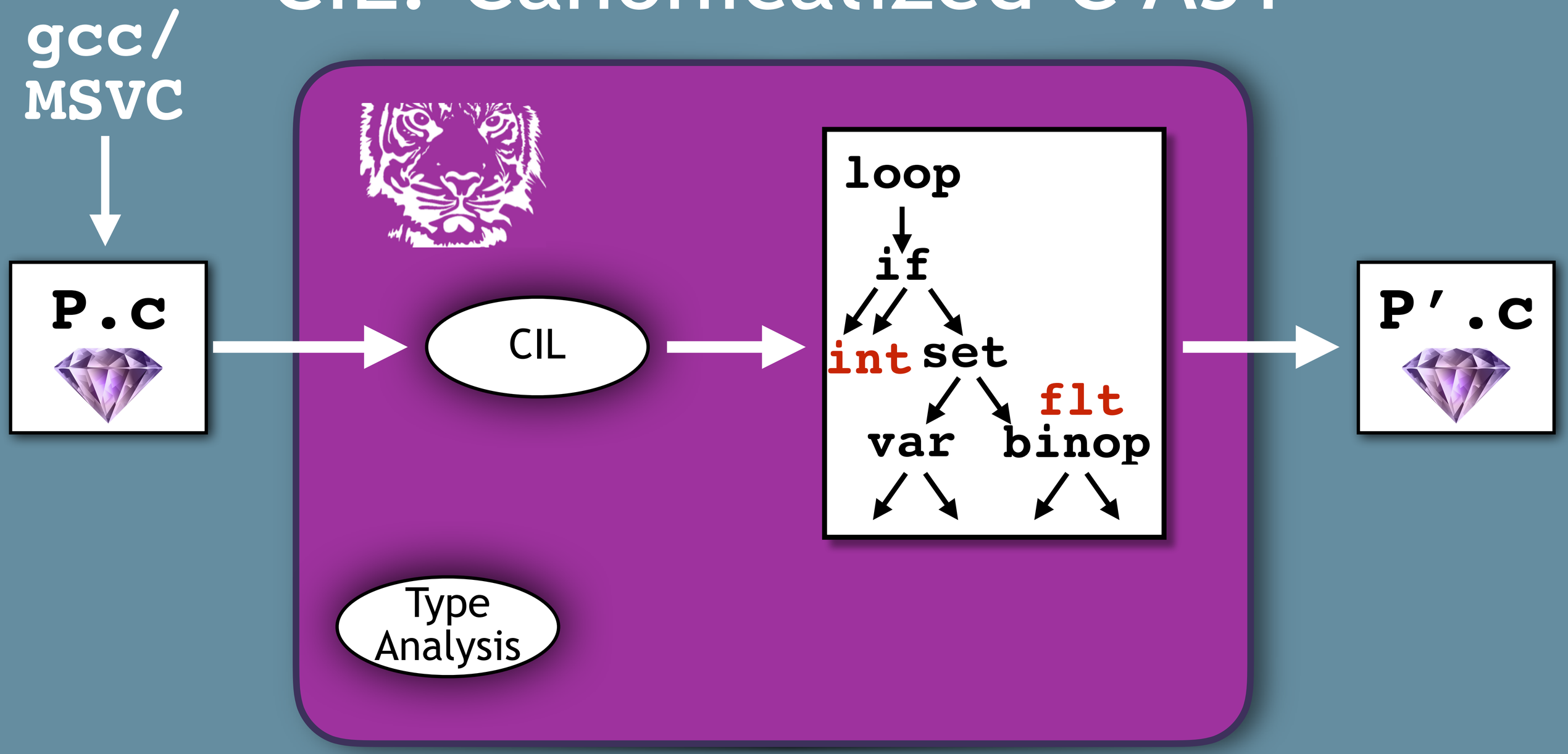

Obfuscation Script
Flatten
Encode Arithmetic
Virtualize
Functions=foo
Option=…
Option=…
Functions=foo
Option=…
Option=…
Functions=bar
Option=…
Option=…
P.c
Seed
P'.c


Transformations
Flatten
Anti Taint
Opaque Predicates
Virtualize
Merge
Dynamic
Split
Jitting
Encode Literals
Branch Functions
Encode Arithmetic
Encode Data
OCaml
Tigress Infrastructure
CIL
myJit

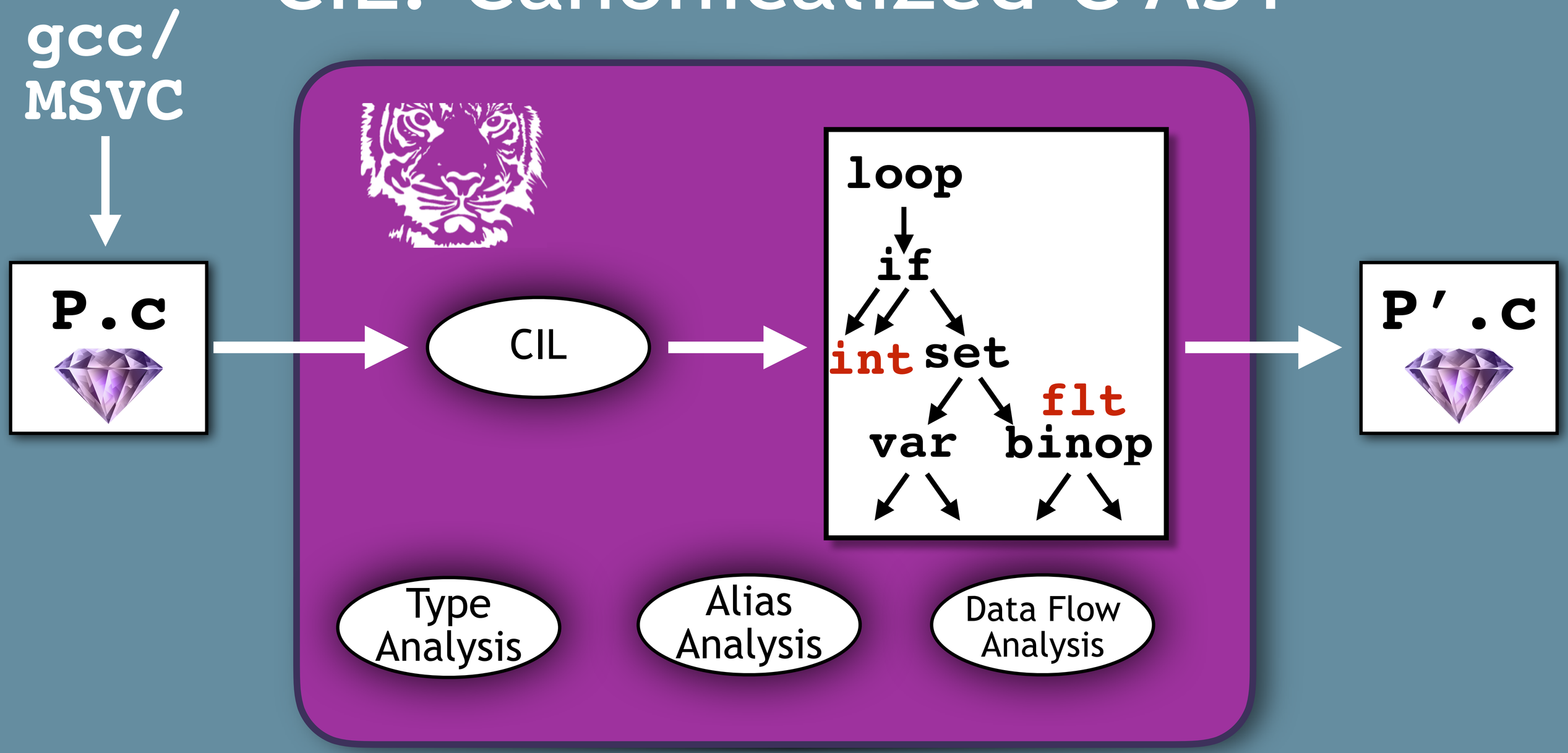
CIL: Canonicalized C AST



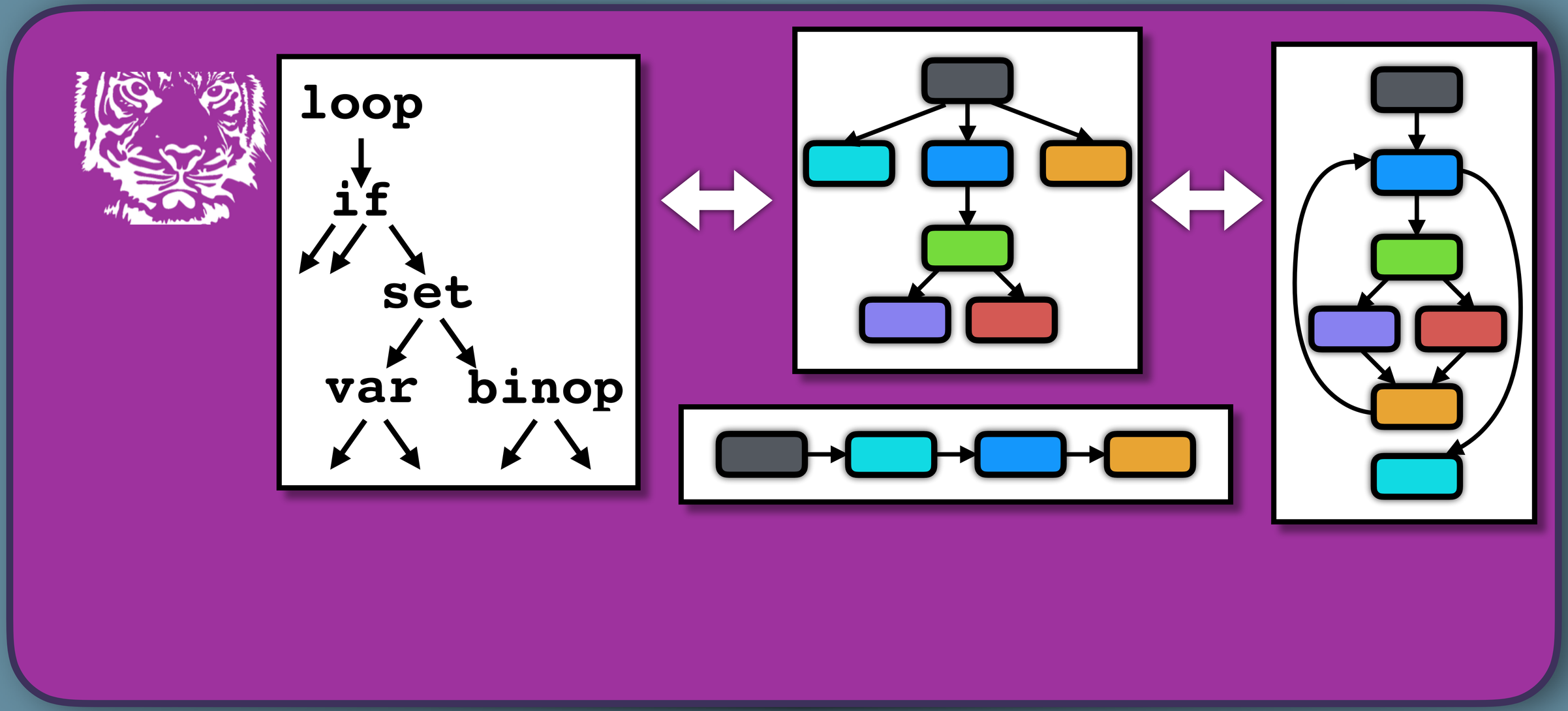
CIL: Canonicalized C AST



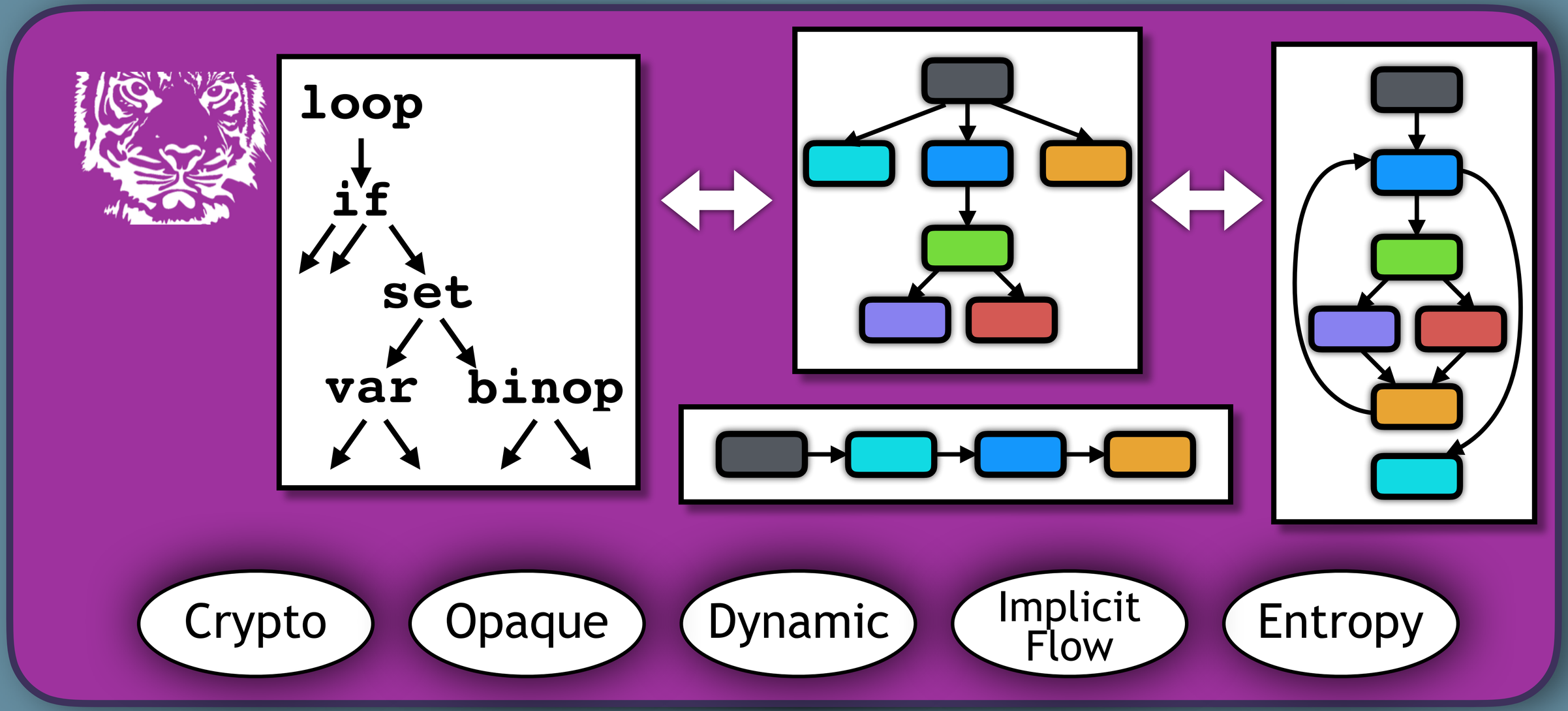
CIL: Canonicalized C AST



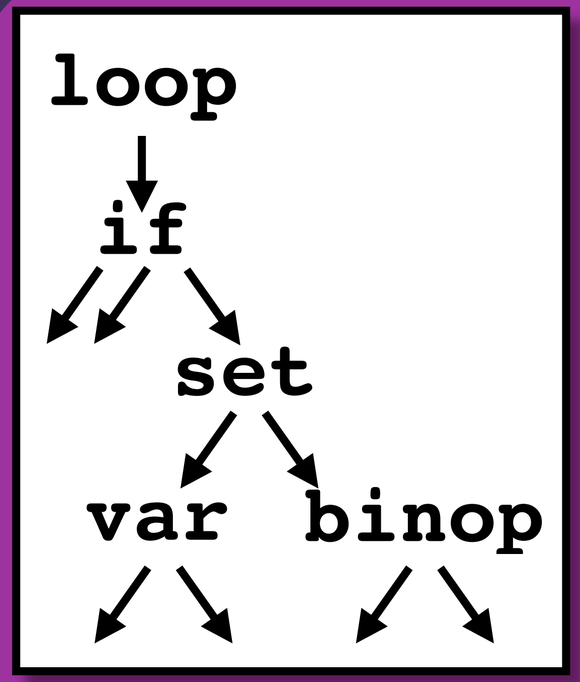
Tigress Infrastructure



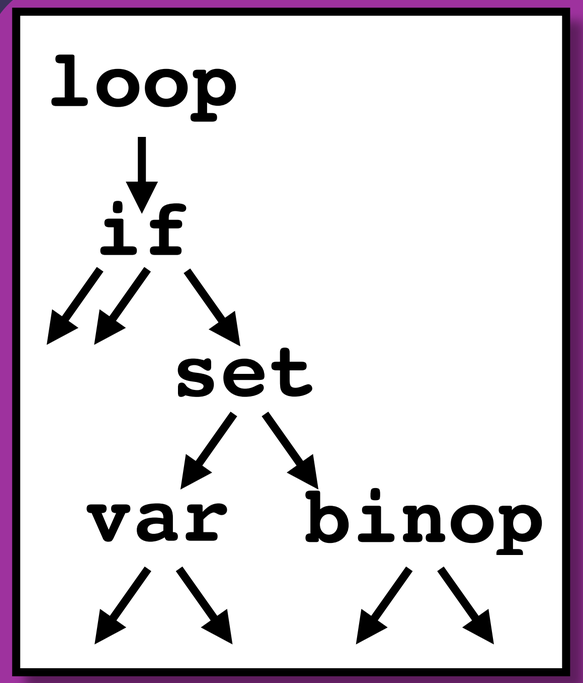
Tigress Infrastructure

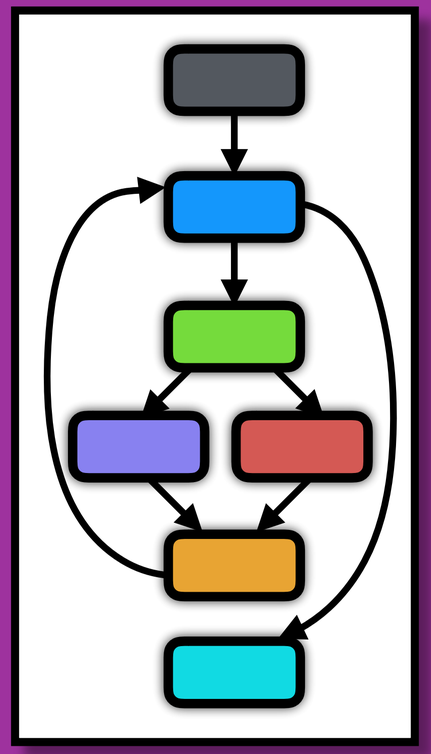


Tigress Transformations

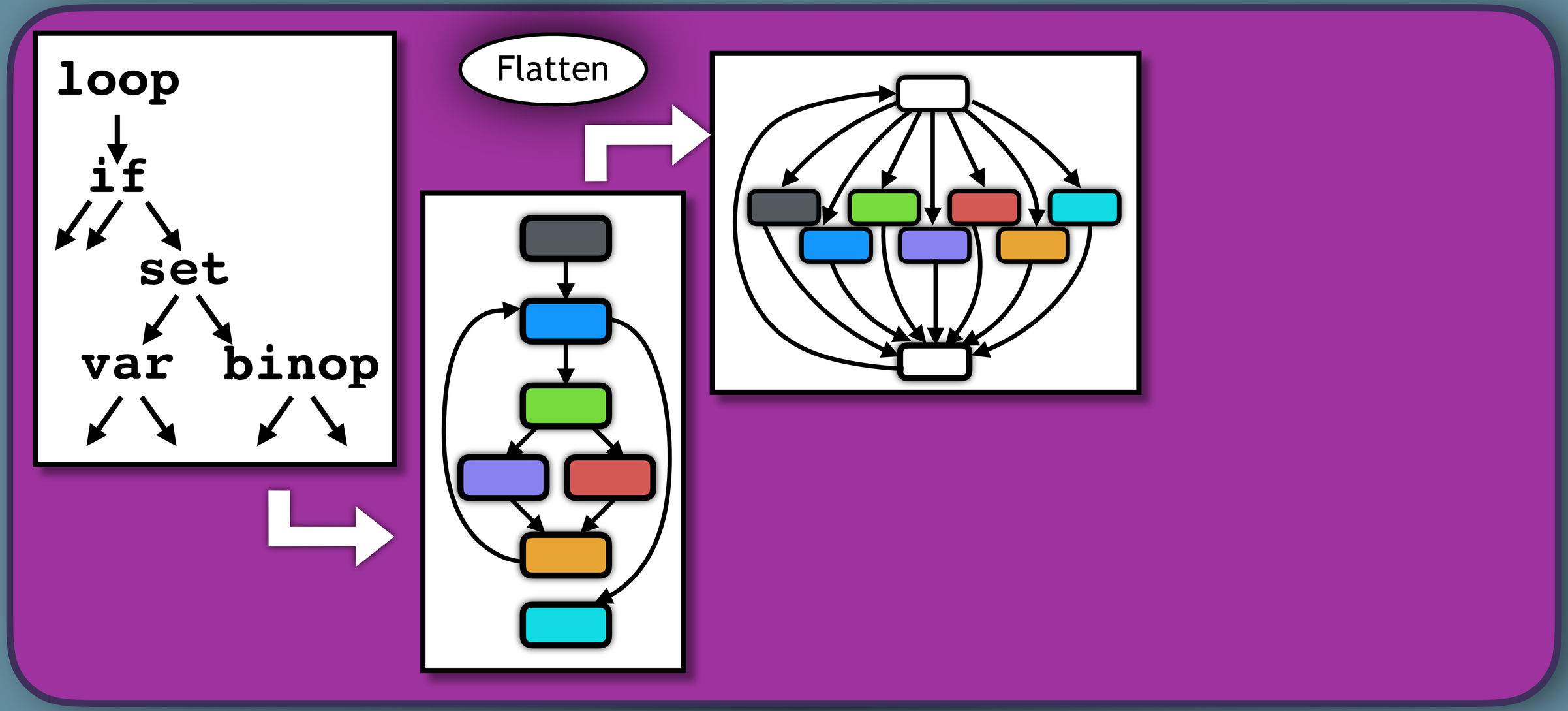


Tigress Transformations

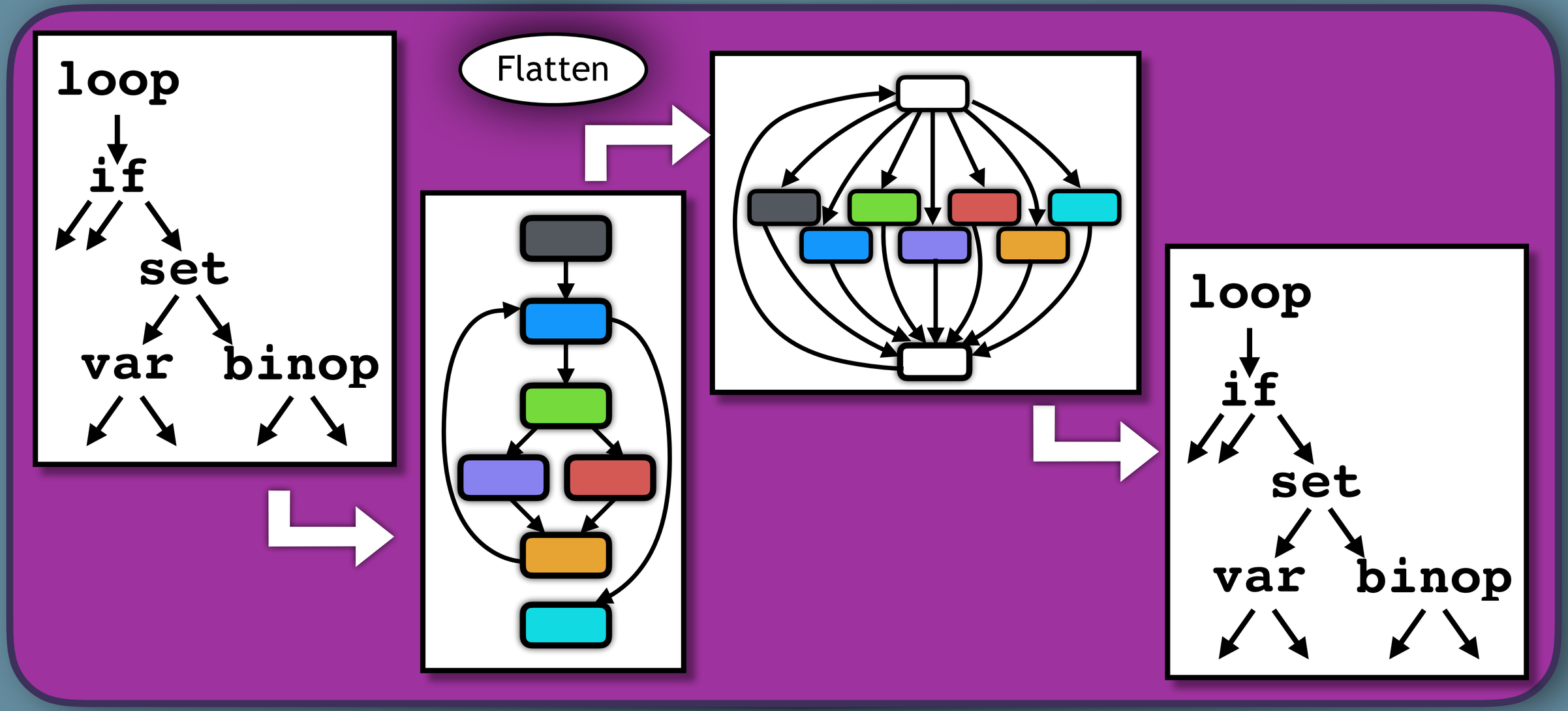




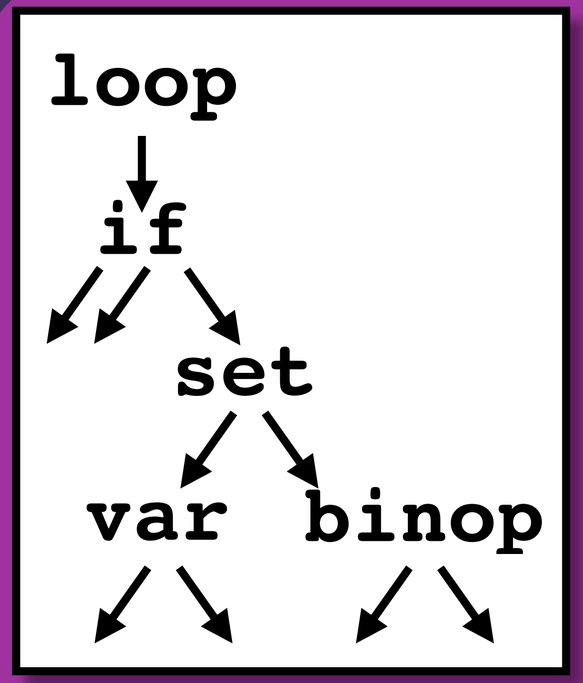
Tigress Transformations



Tigress Transformations



Binary Transformations

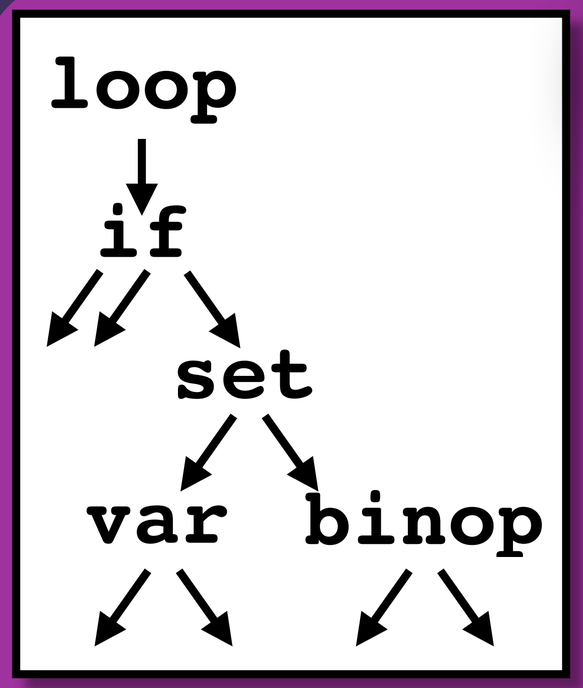


SPARC

arm

intel

Binary Transformations



JIT

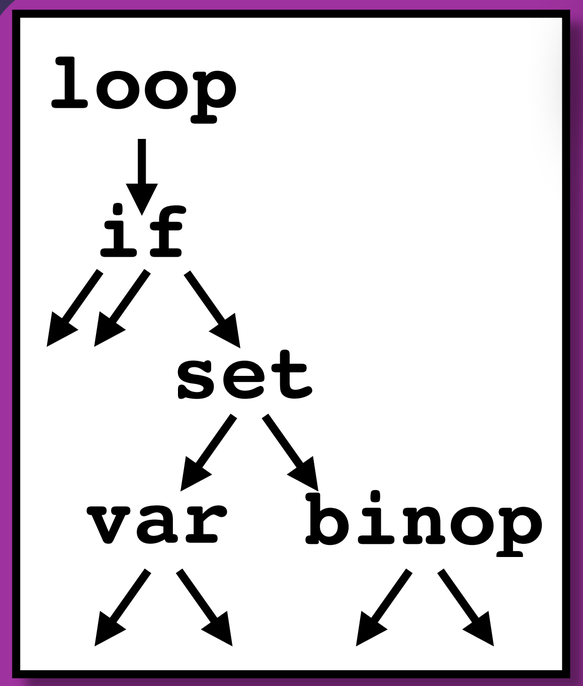
```
push   rbx
lea    rdi,[…]
call   puts
```

SPARC

arm

intel

Binary Transformations



JIT

```
push   rbx
lea    rdi,[…]
call   puts
```

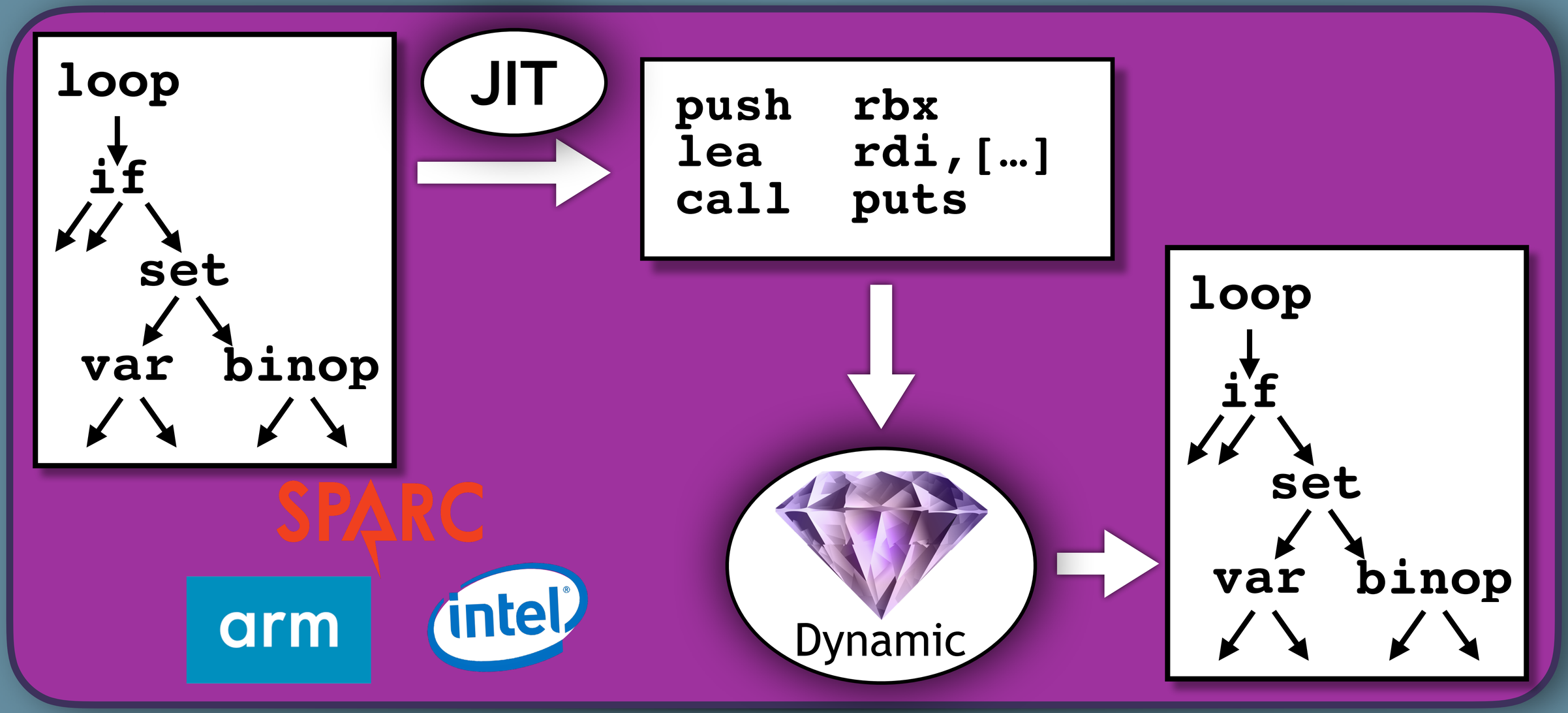
SPARC

arm

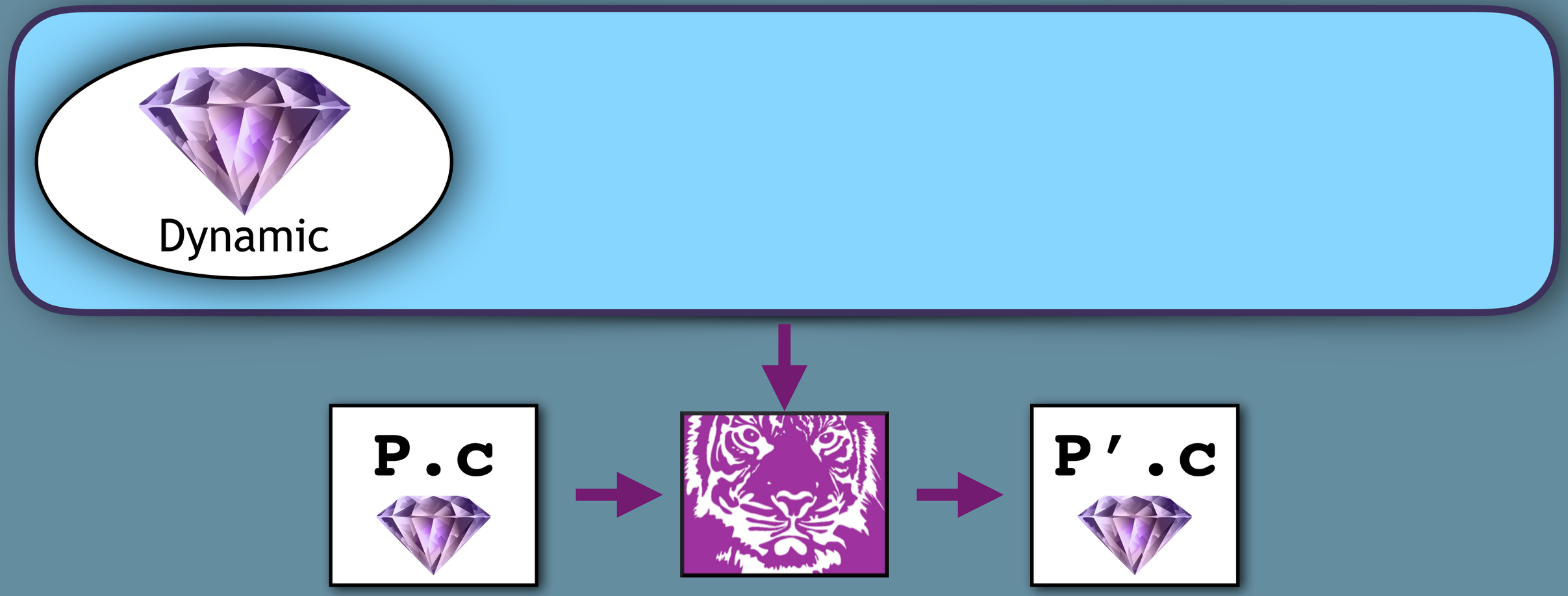
intel

Dynamic

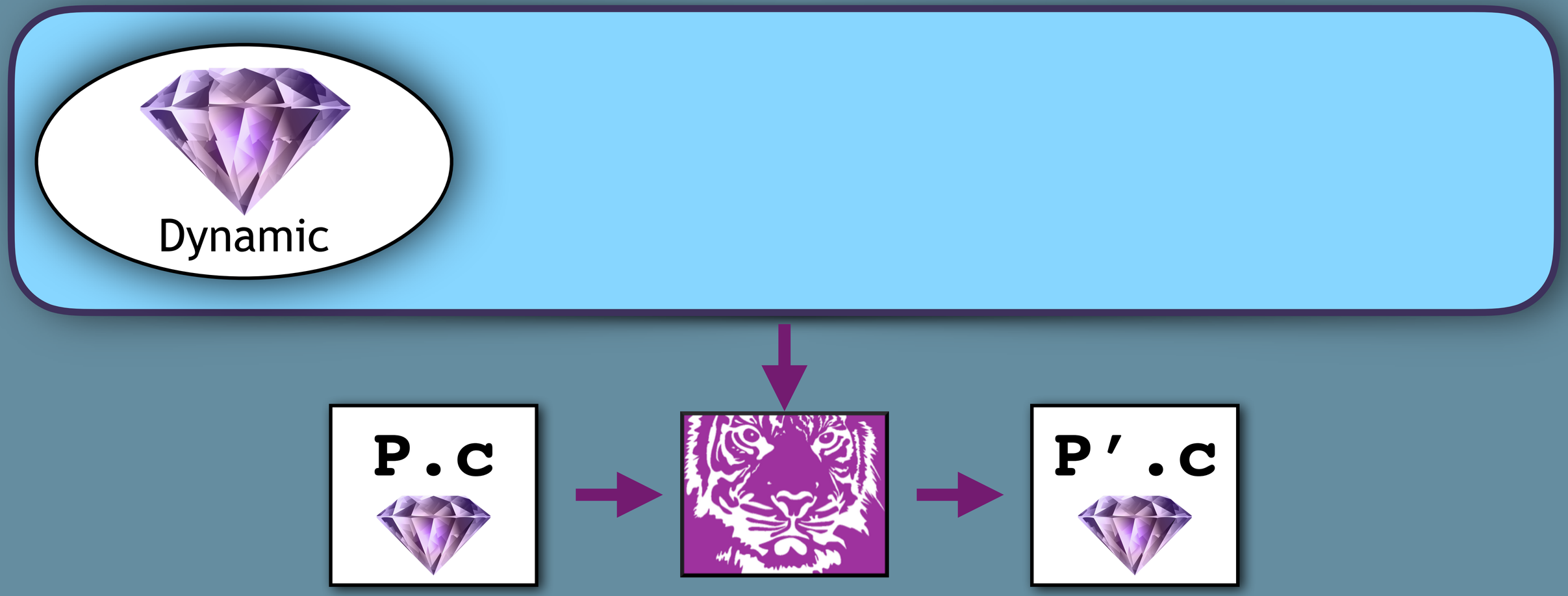
Binary Transformations



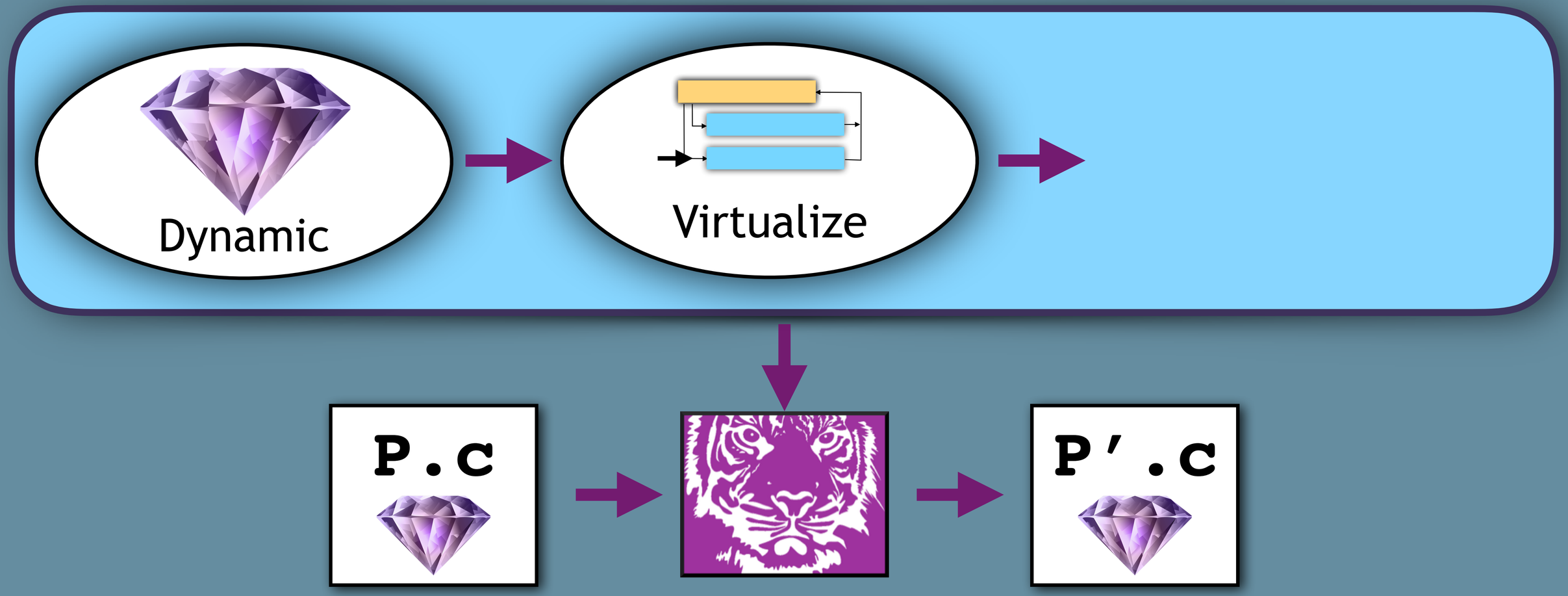
Remainder of This Talk



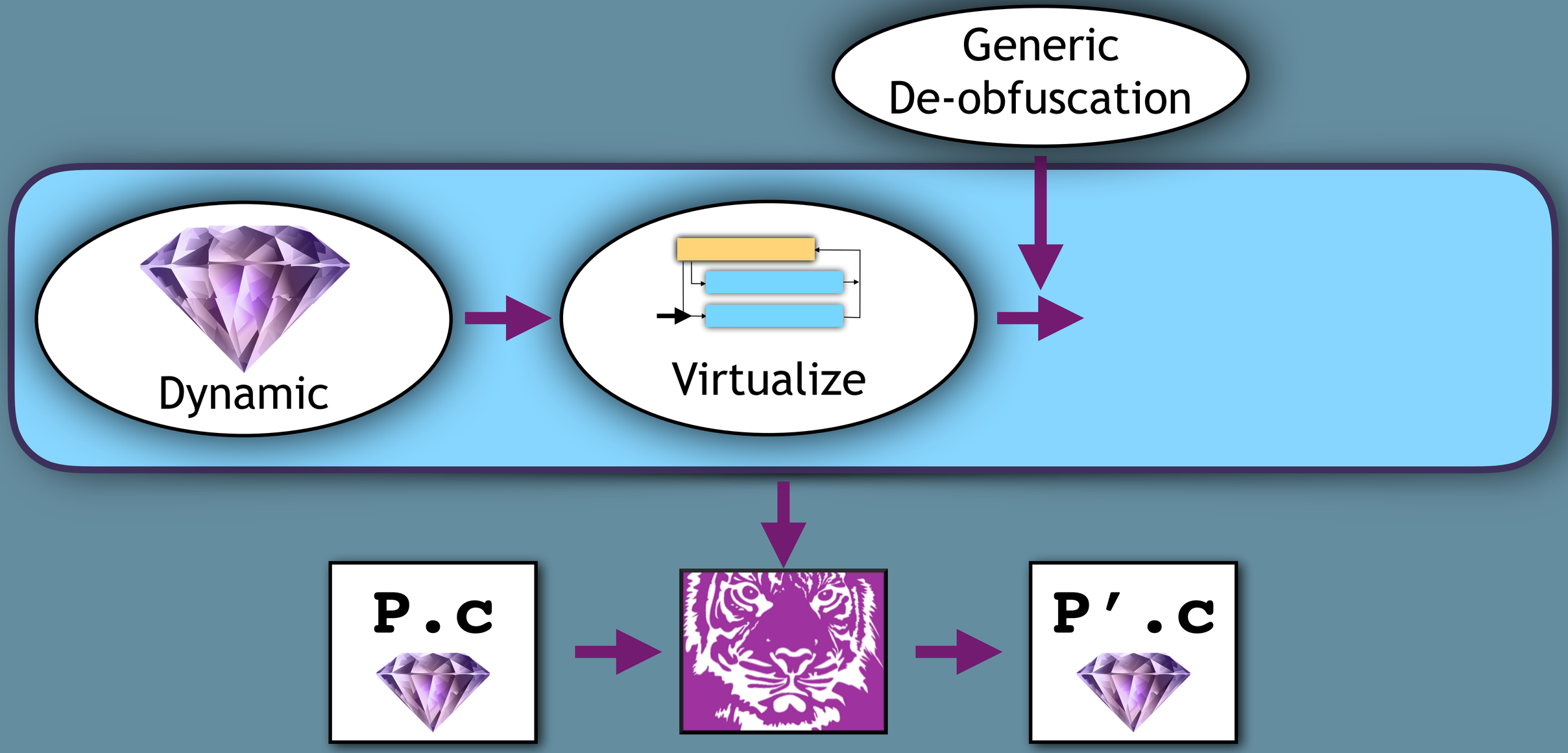
Remainder of This Talk



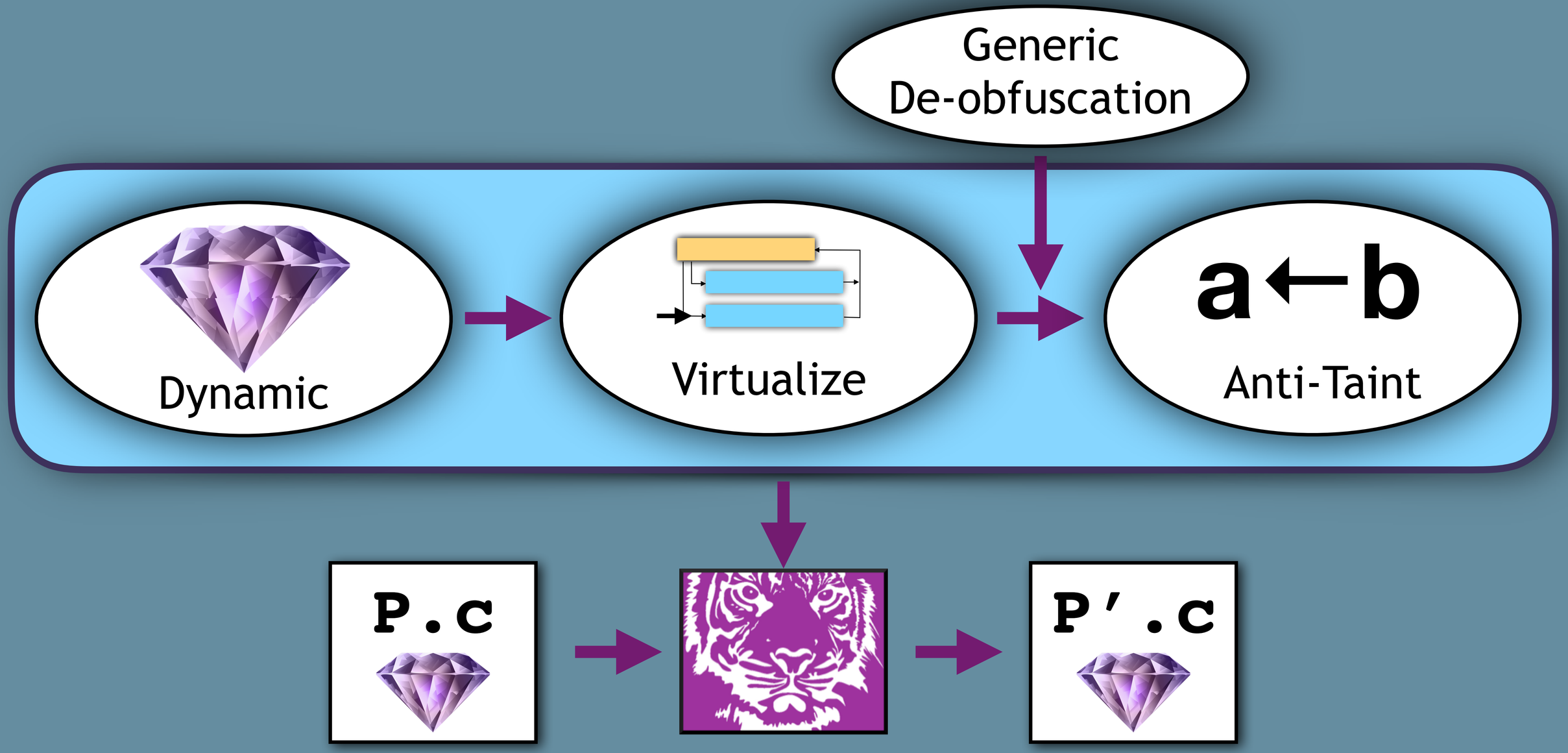
Remainder of This Talk



Remainder of This Talk

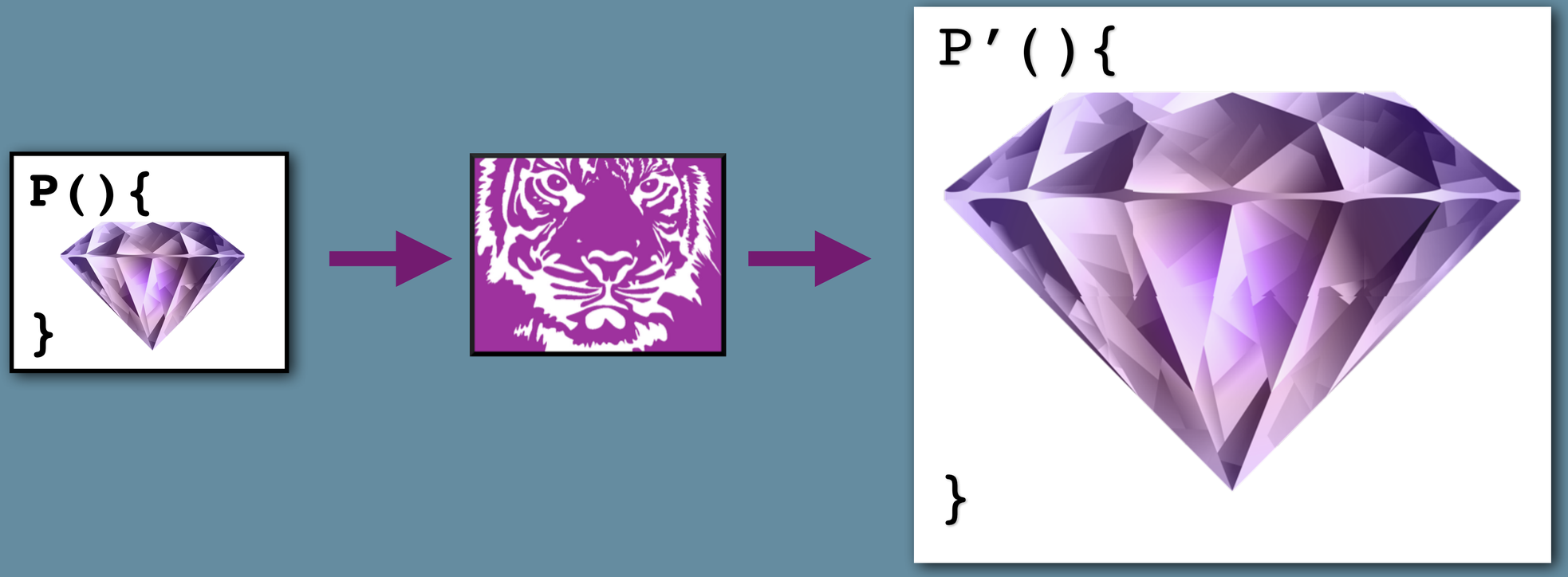


Remainder of This Talk

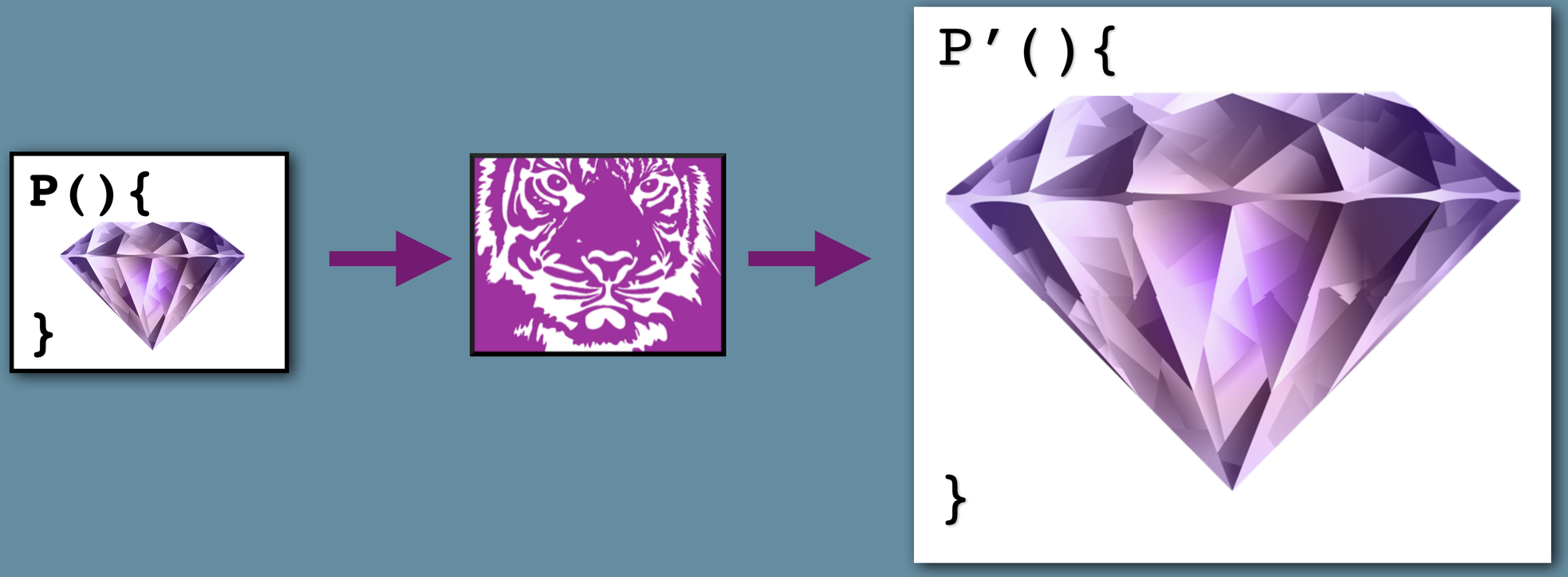


Generic Dynamic Motiaccuto

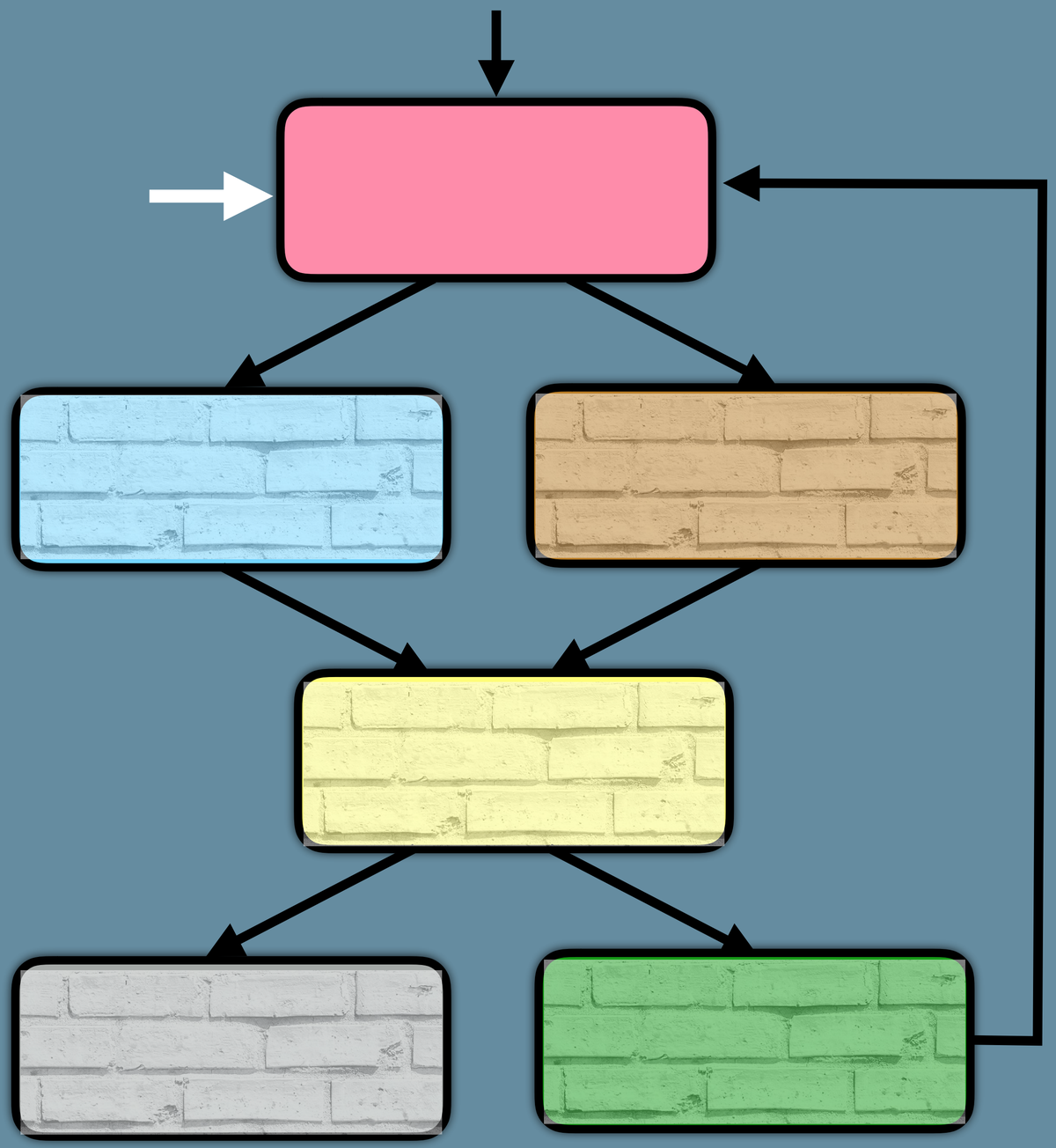
Generic Dynamic Obfuscation

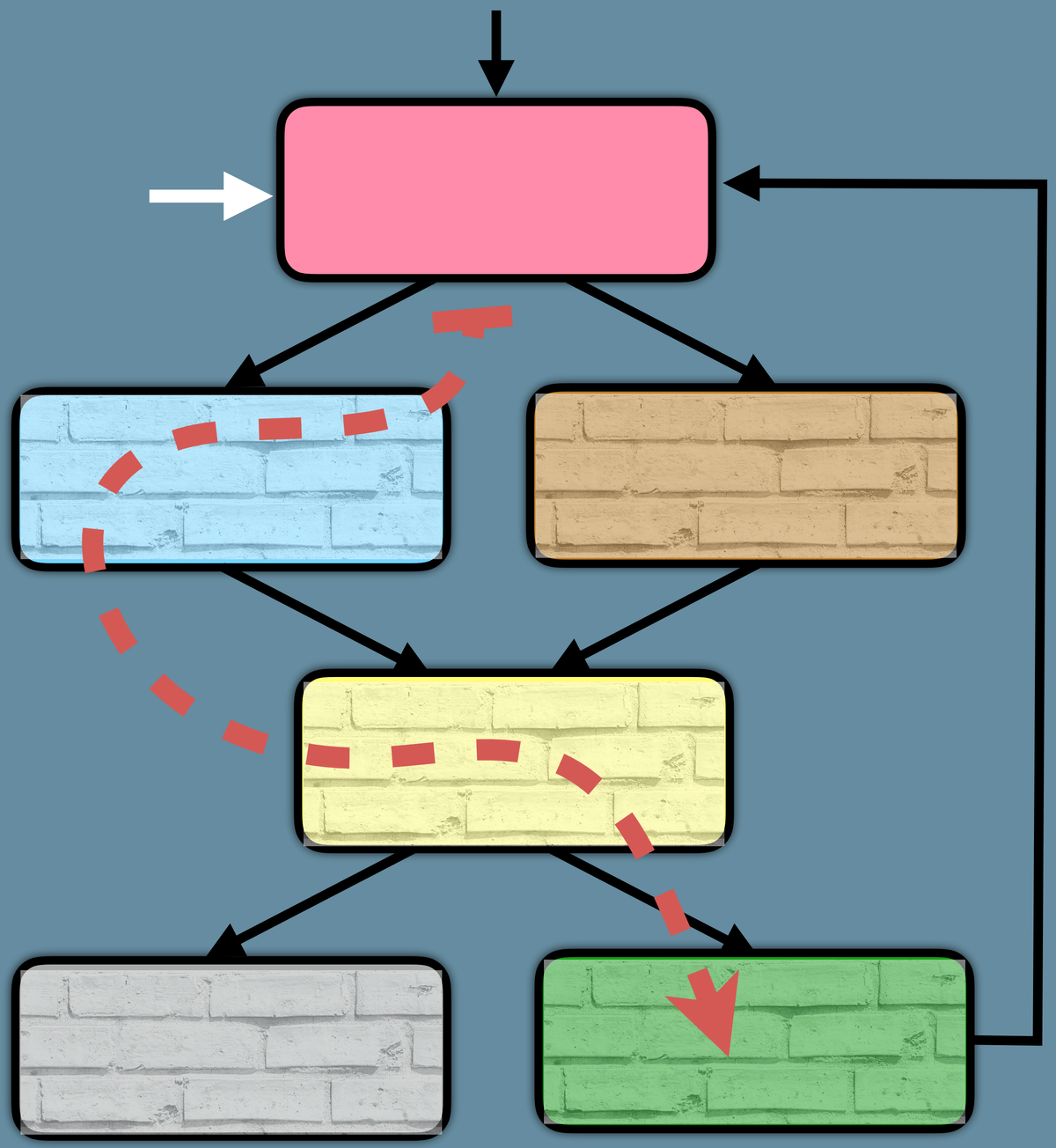


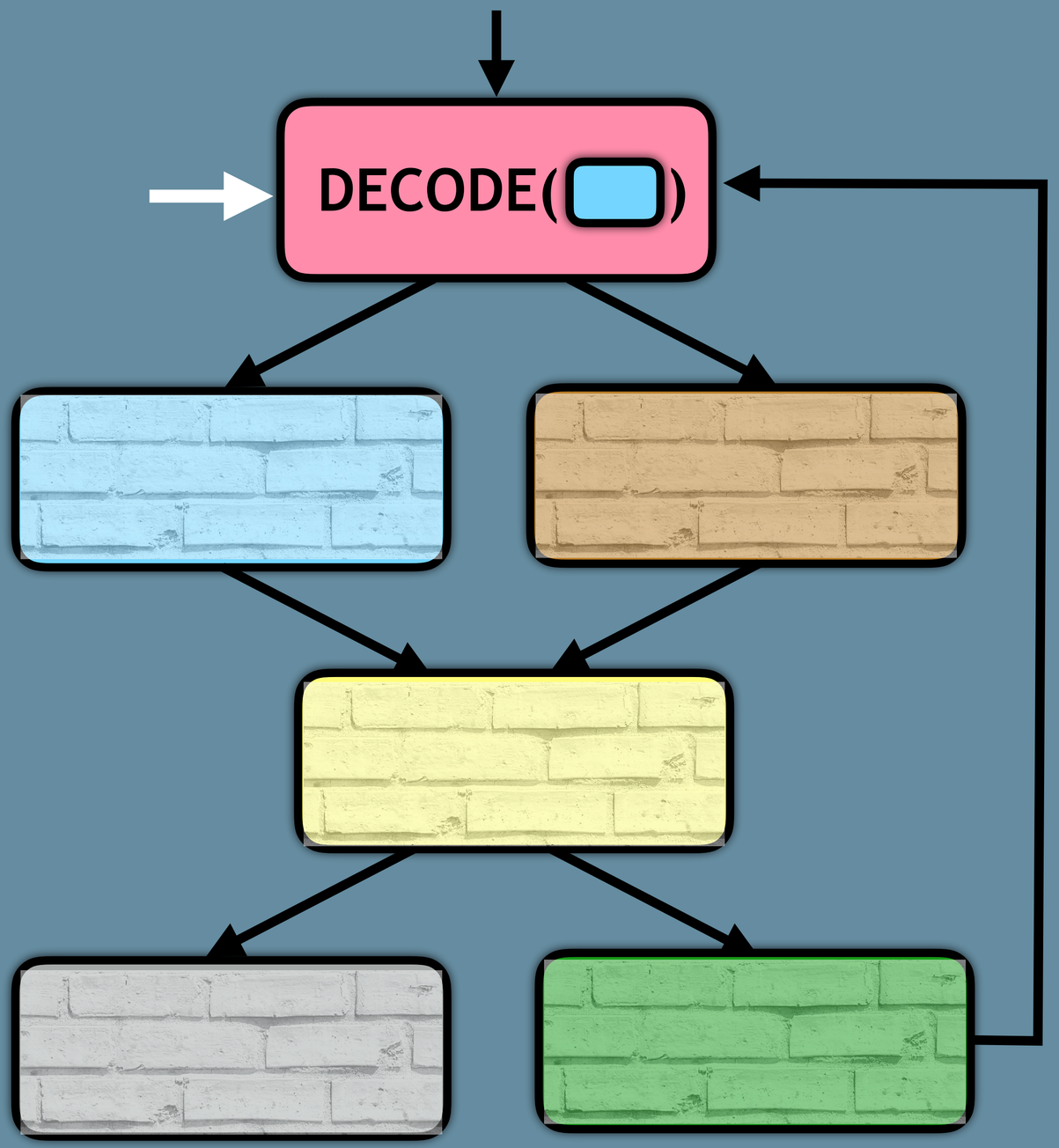
- Self-Modifying Code!
- Code in Constant Flux!
- Code Never Fully in the Clear!

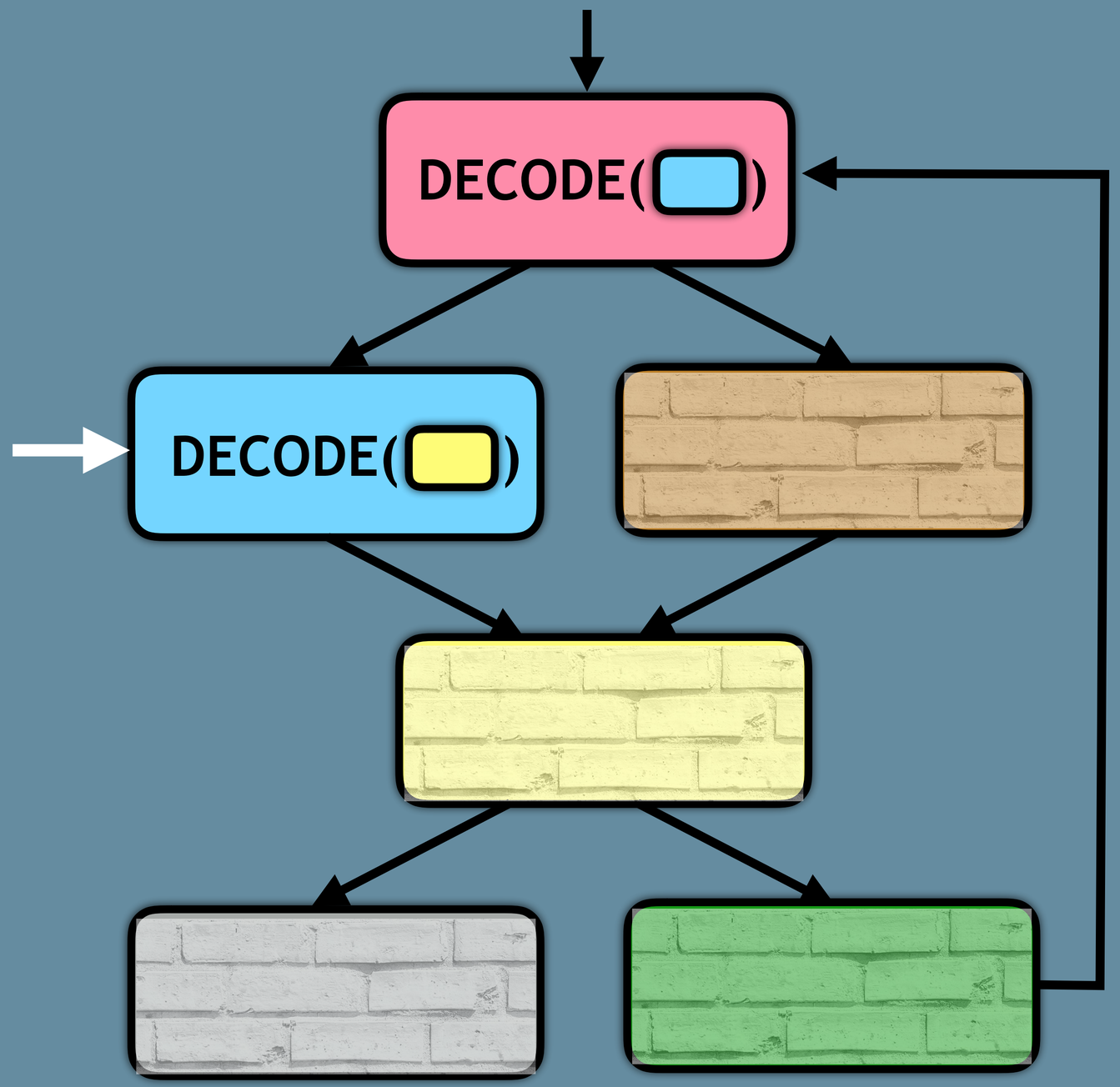


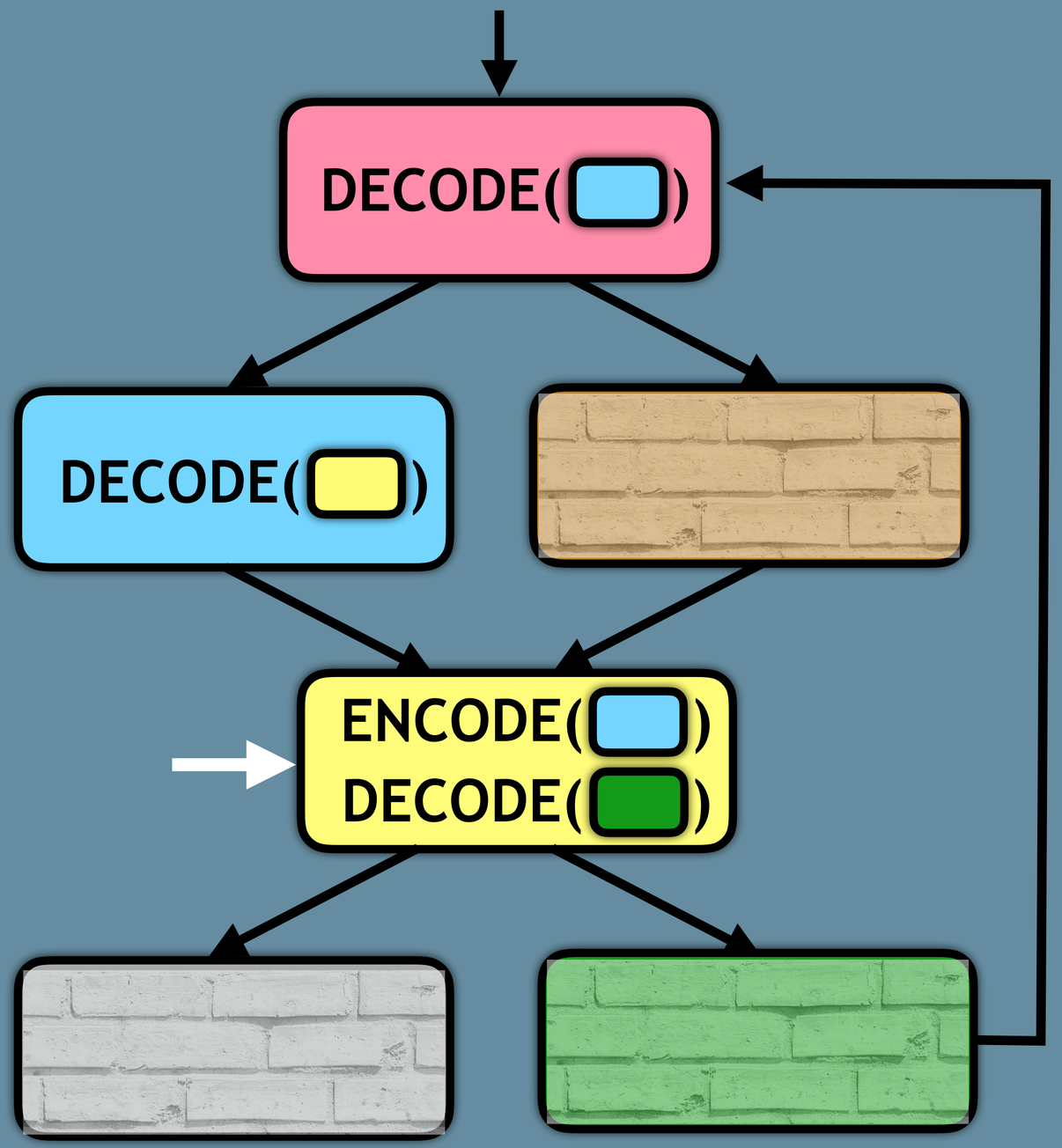
- Self-Modifying Code!
- Code in Constant Flux!
- Code Never Fully in the Clear!

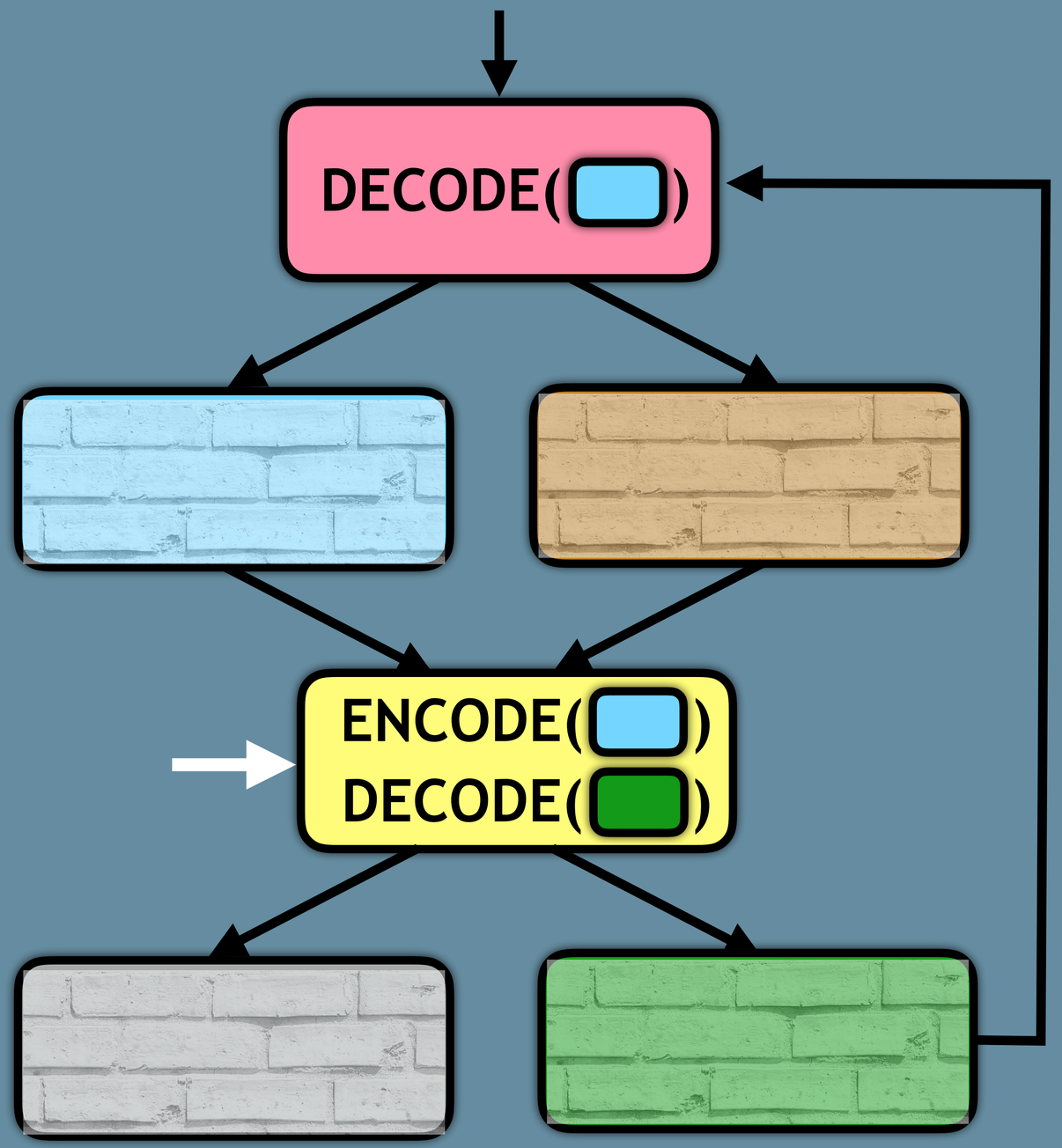


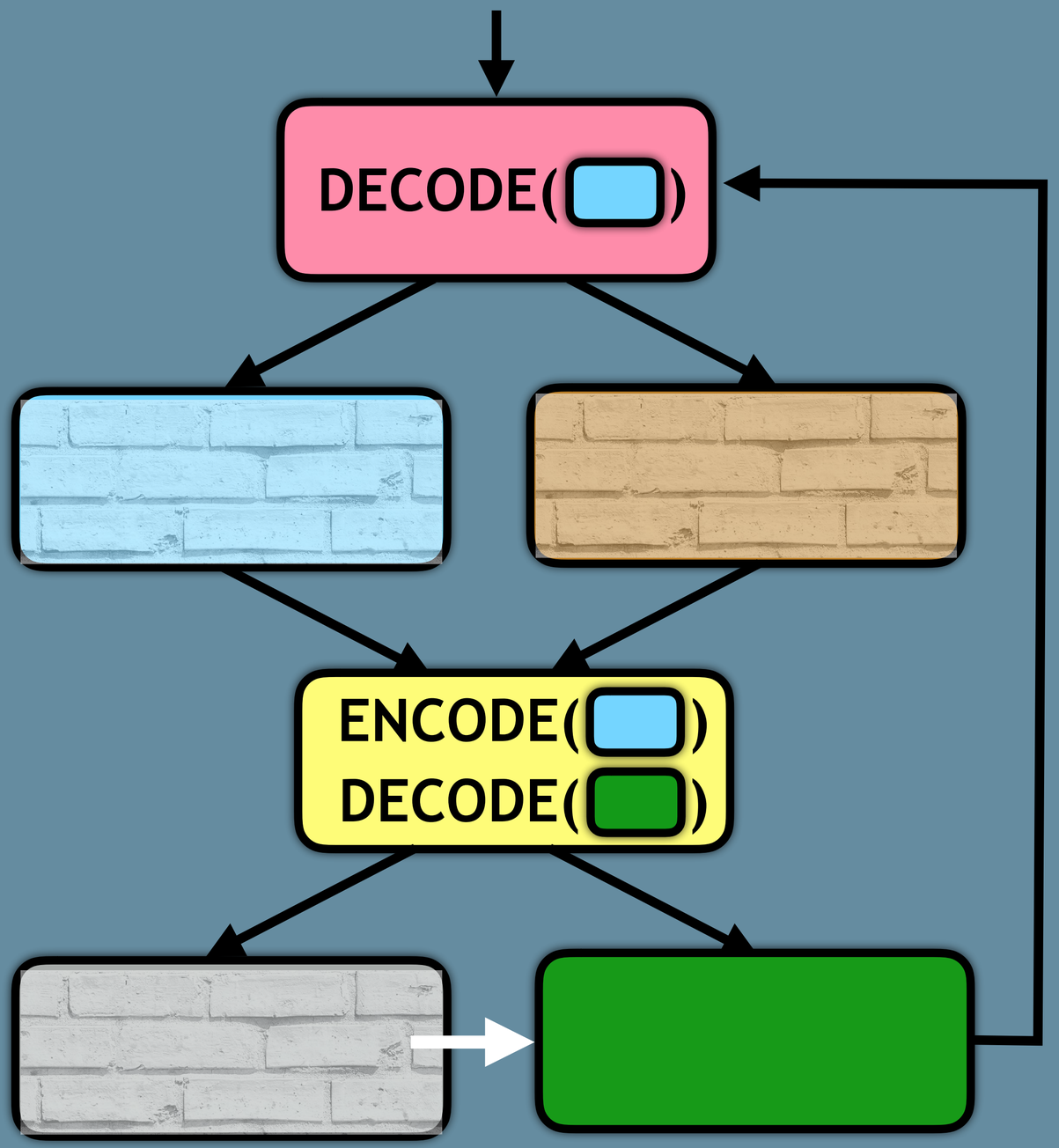


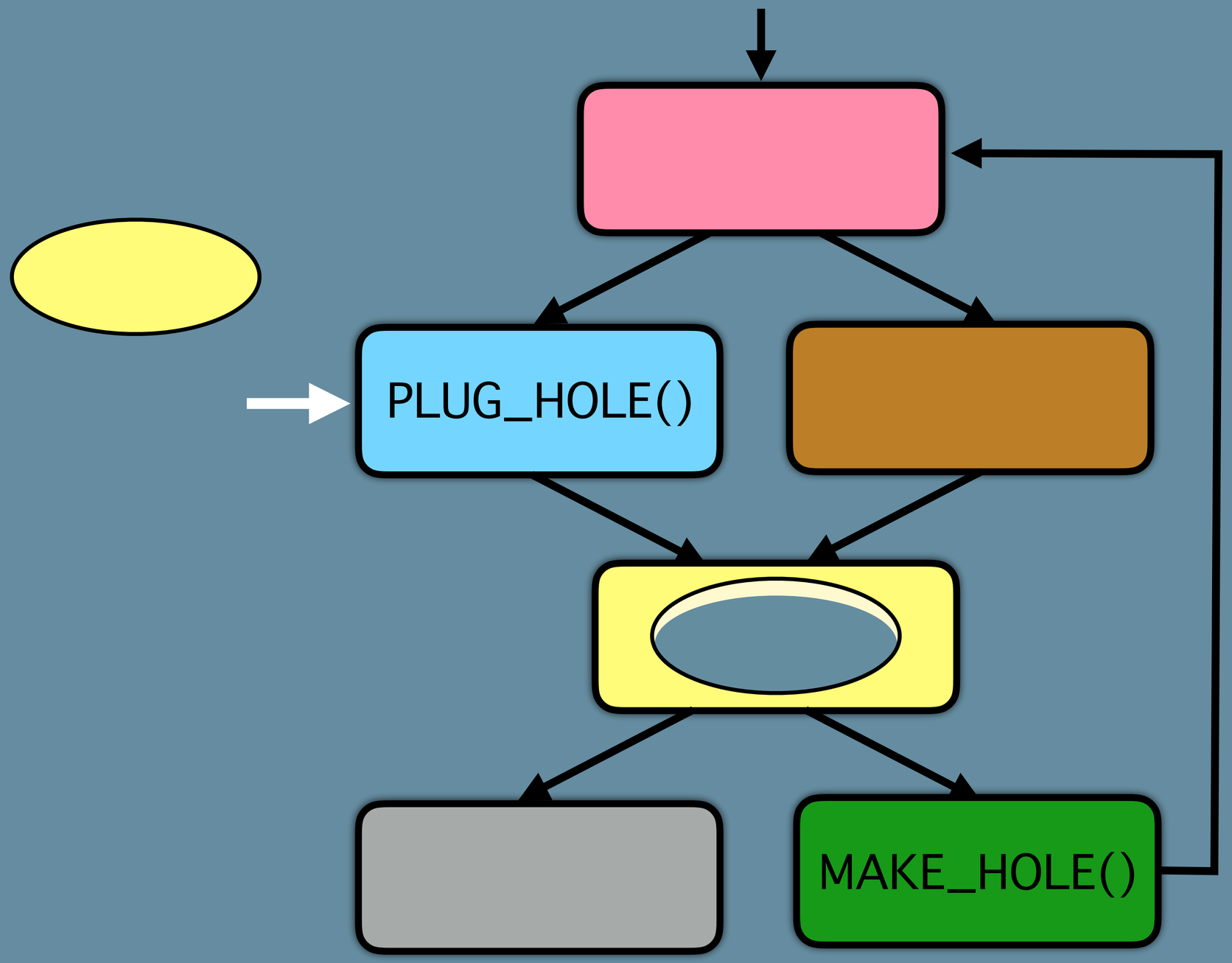

DECODE( )


DECODE( )
DECODE( )

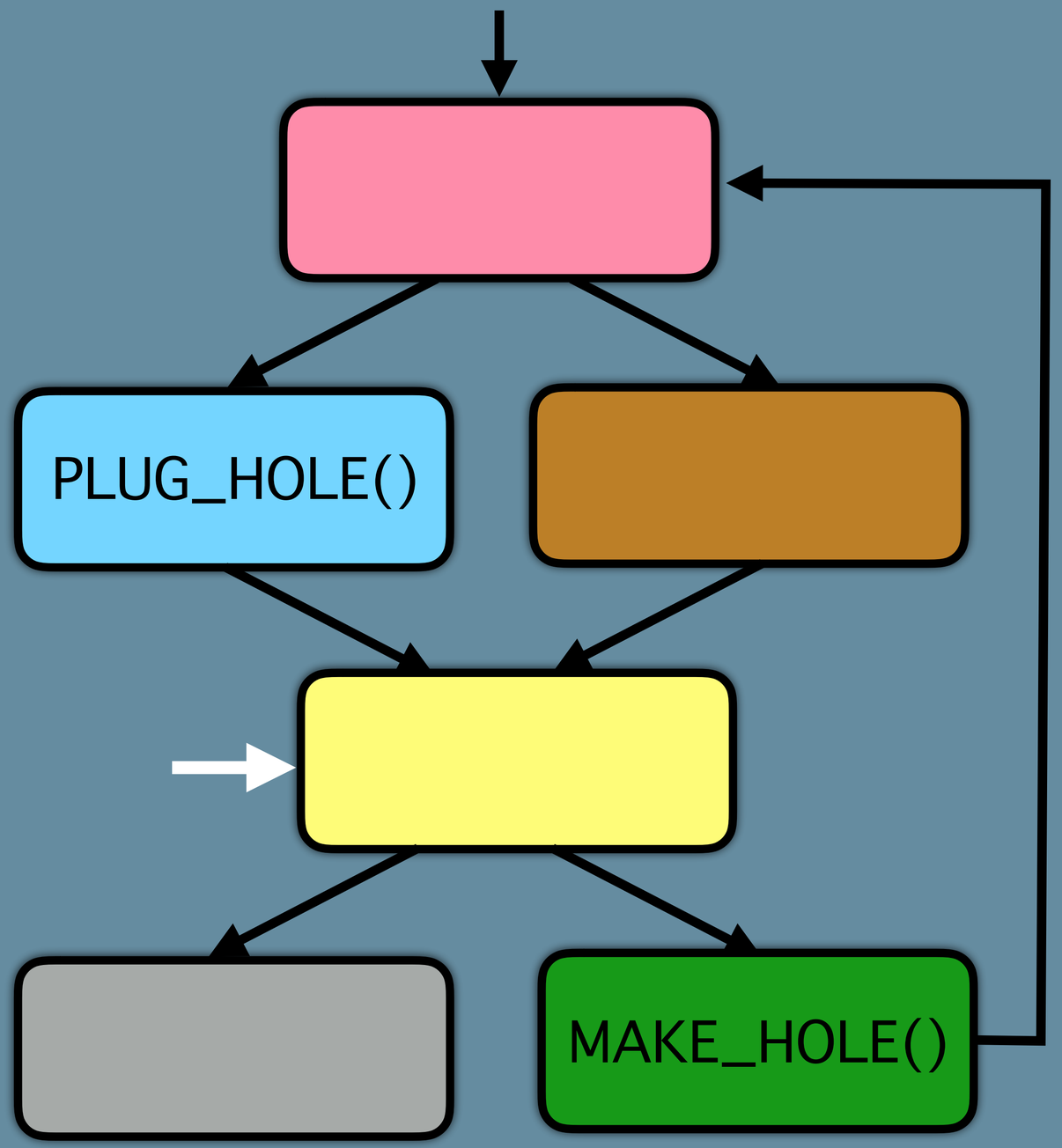

DECODE( )
DECODE( )
ENCODE( )
DECODE( )


DECODE( )
ENCODE( )
DECODE( )

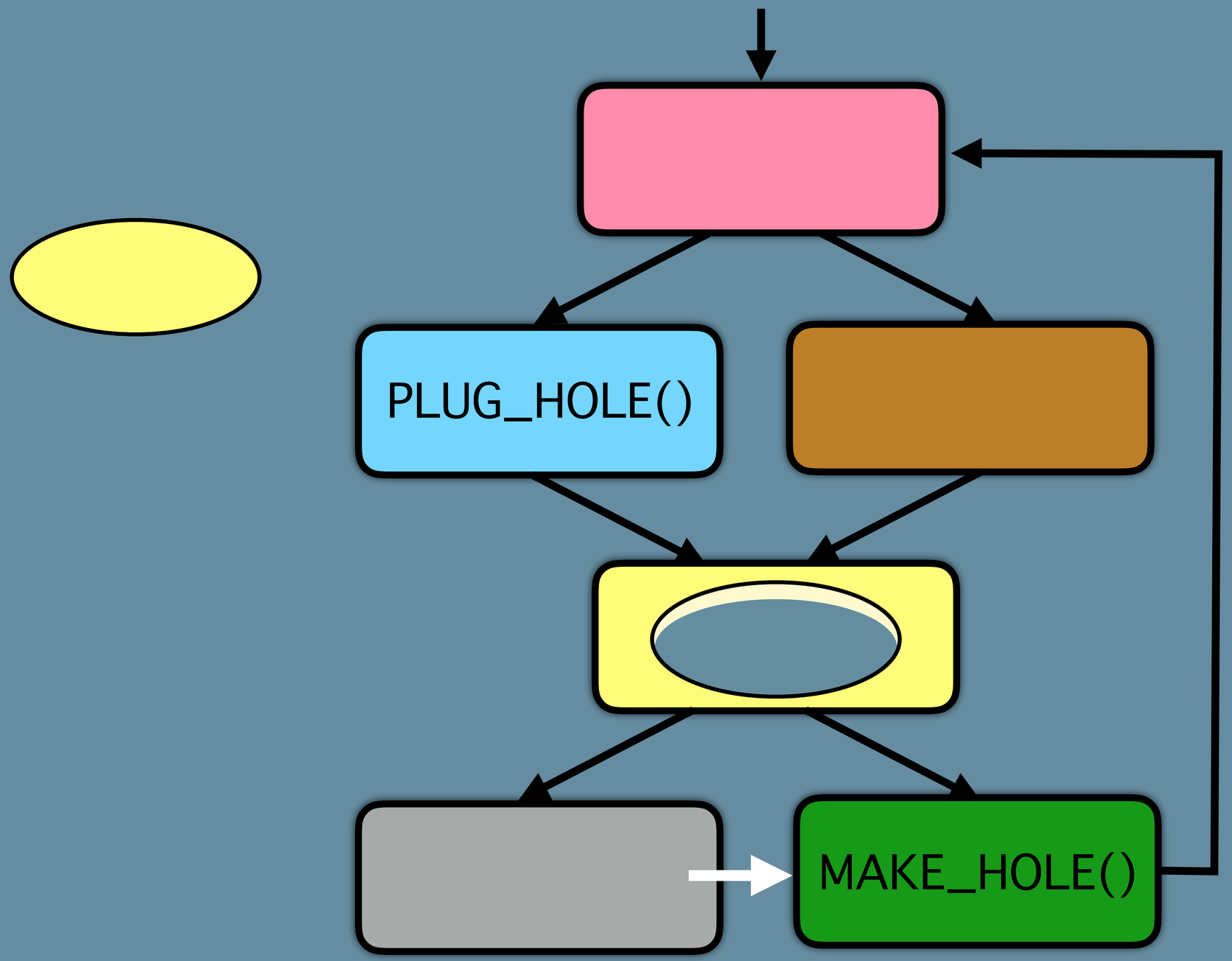

DECODE( )
ENCODE( )
DECODE( )



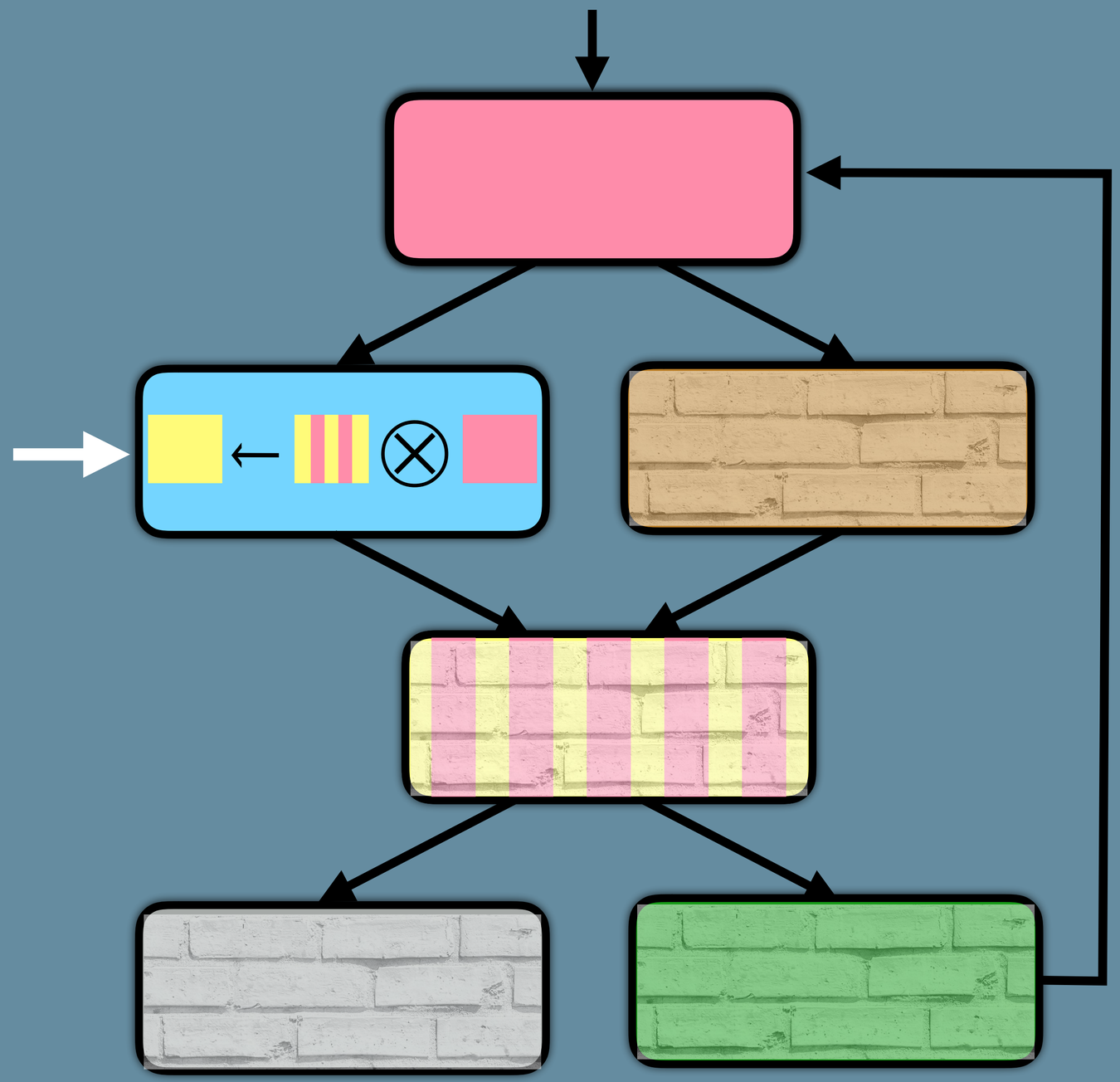
Kanzaki et al., Exploiting Self-Modification Mechanism for Program Protection



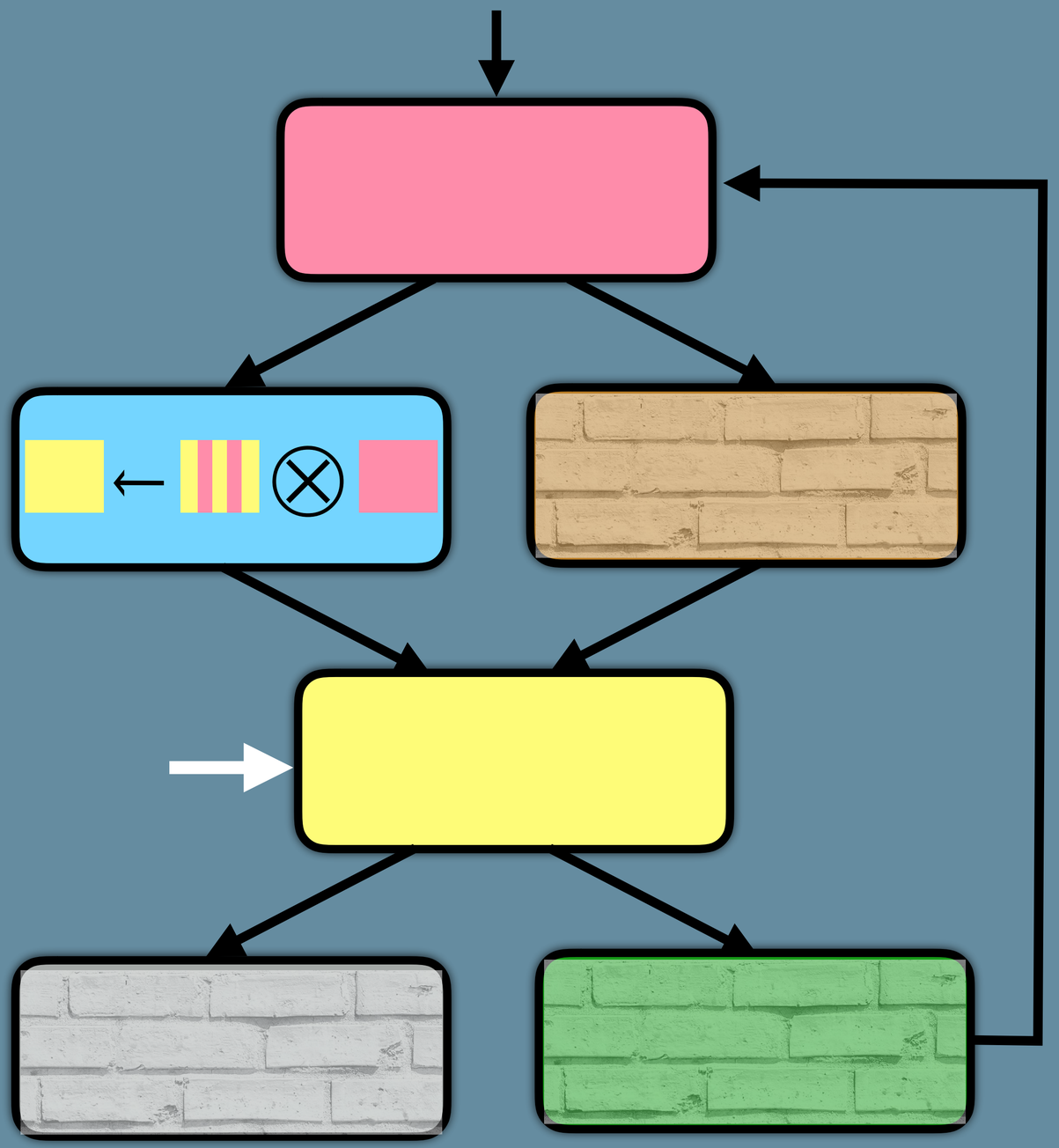
Kanzaki et al., Exploiting Self-Modification Mechanism for Program Protection



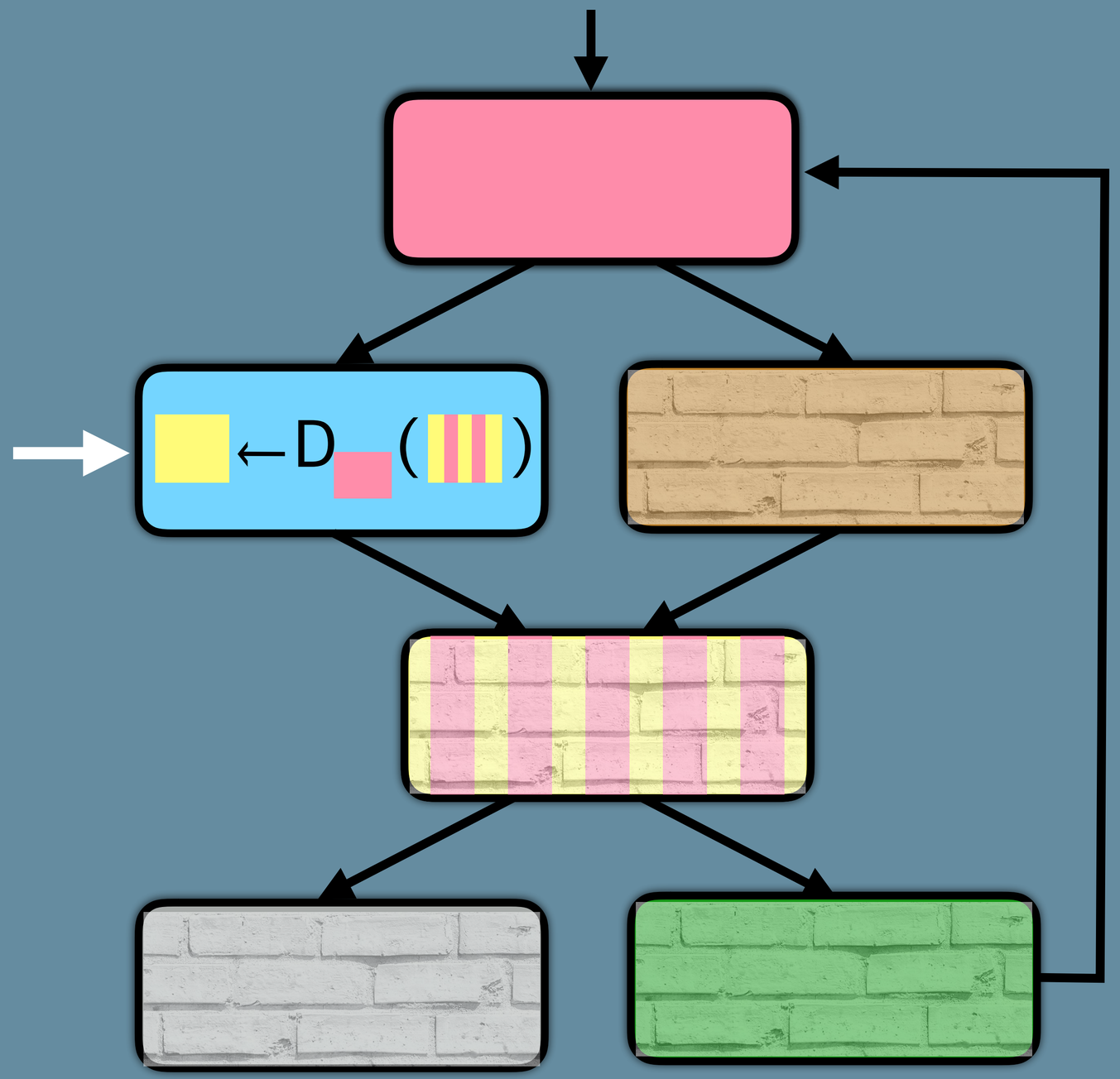
Kanzaki et al., Exploiting Self-Modification Mechanism for Program Protection



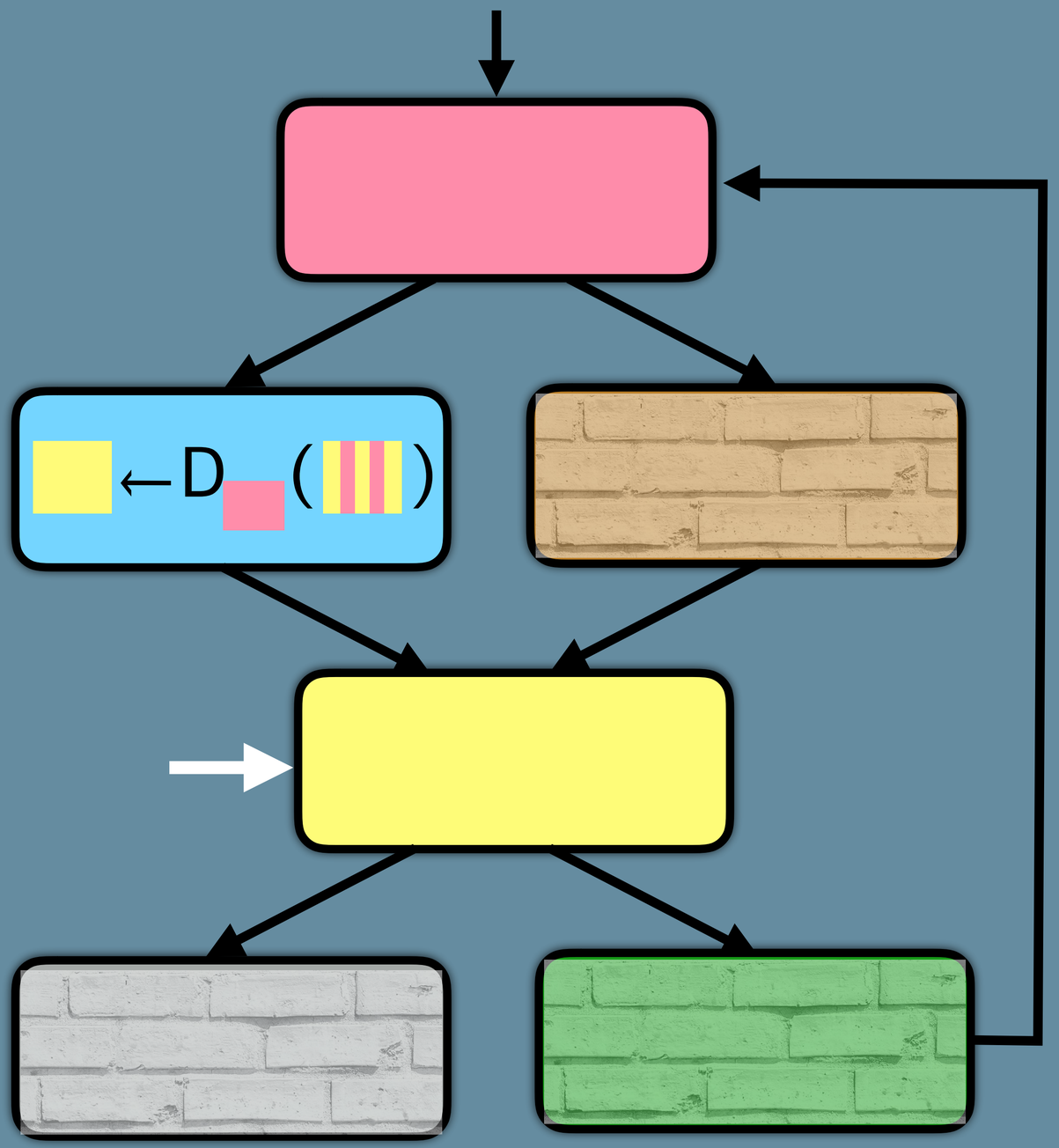
Aucsmith, Tamper Resistant Software: An Implementation, IH’96



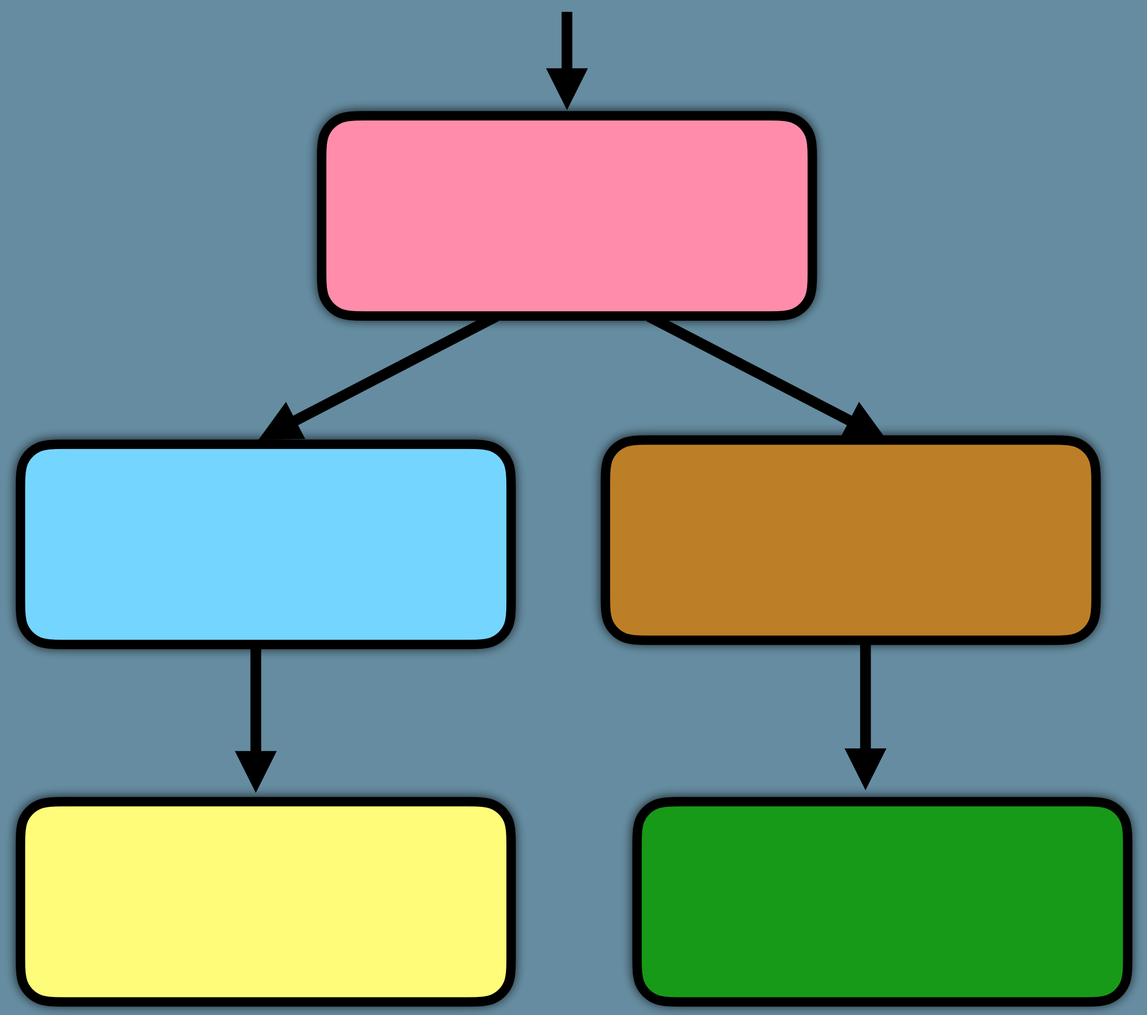
Aucsmith, Tamper Resistant Software: An Implementation, IH’96



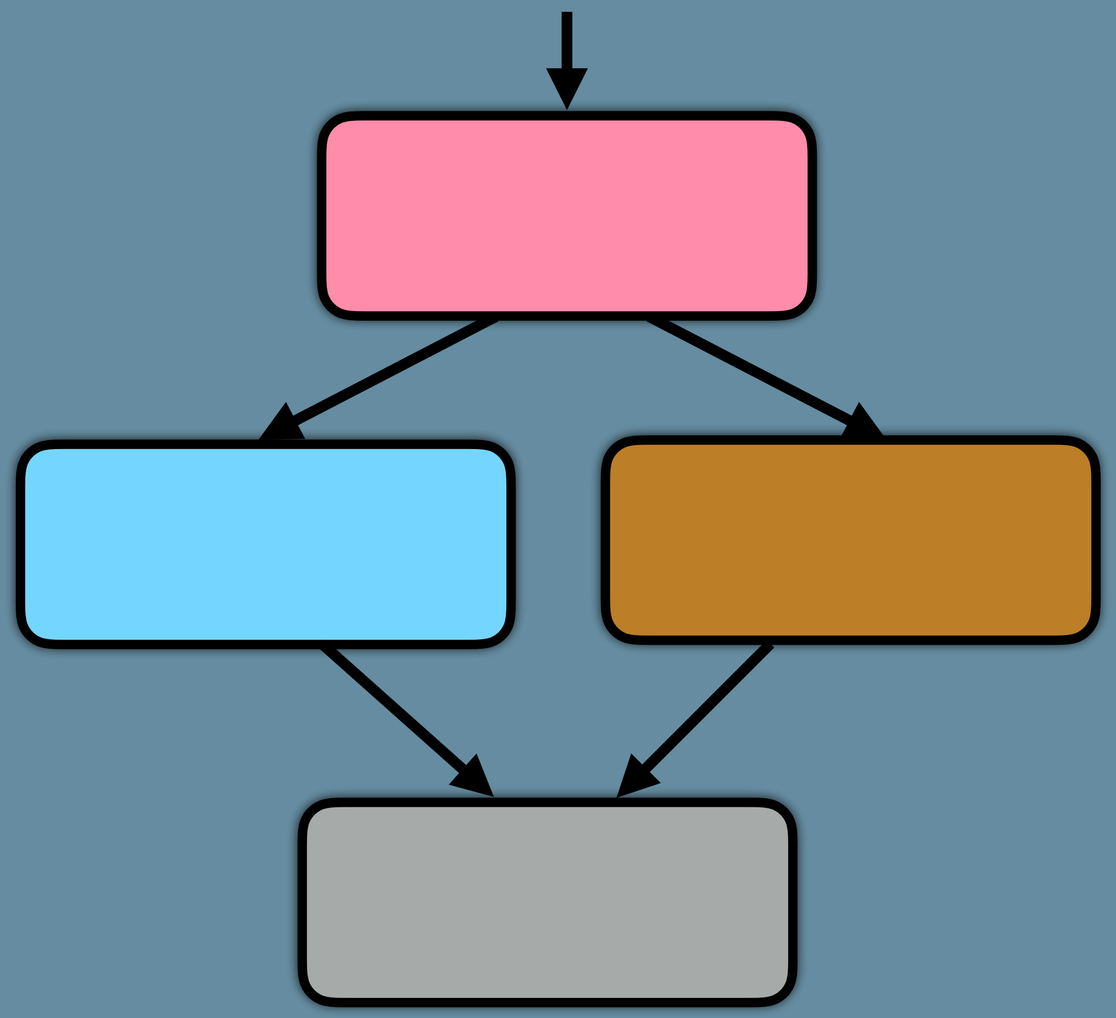
Cappaert, Preneel, et al. Towards Tamper Resistant Code Encryption P&E, ISPEC'08



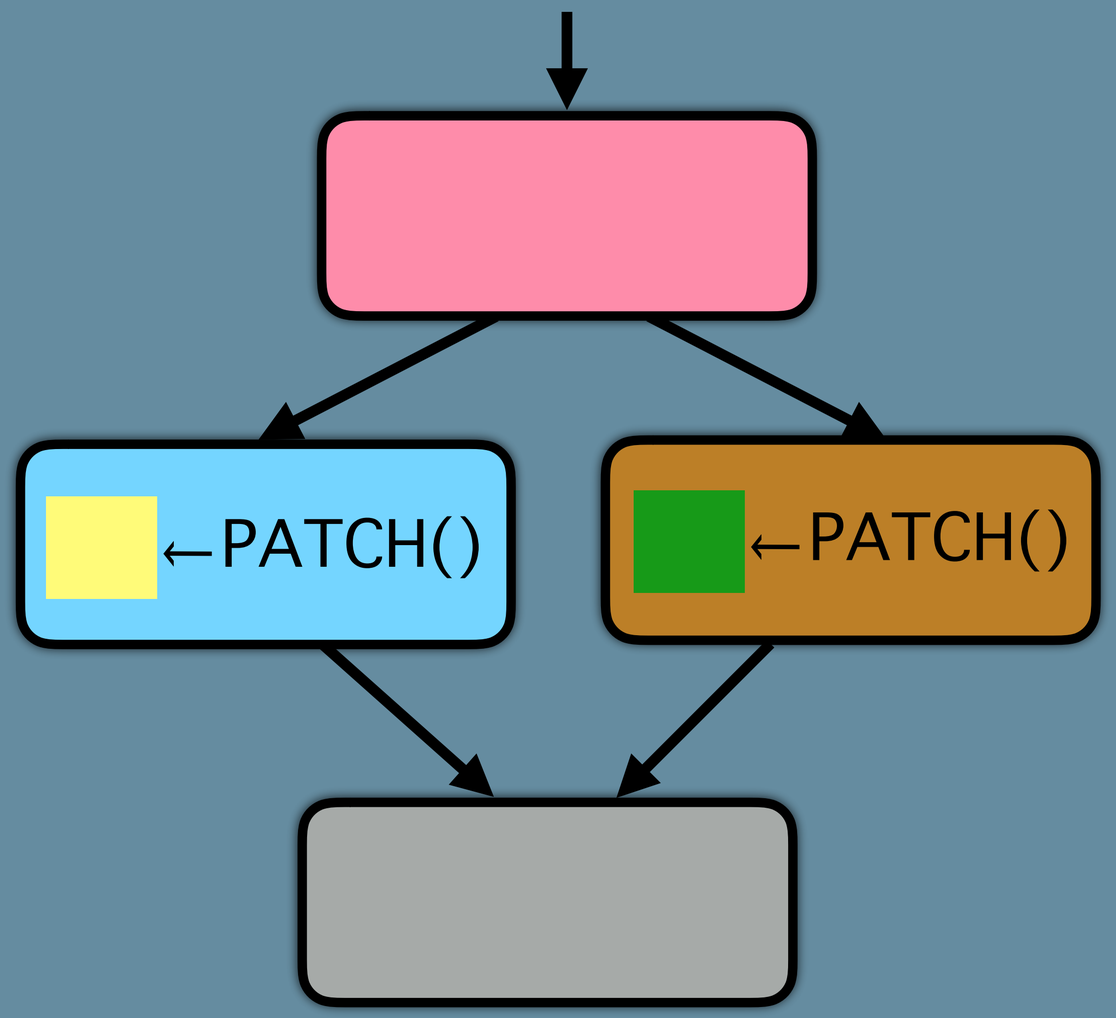
Cappaert, Preneel, et al. Towards Tamper Resistant Code Encryption P&E, ISPEC'08



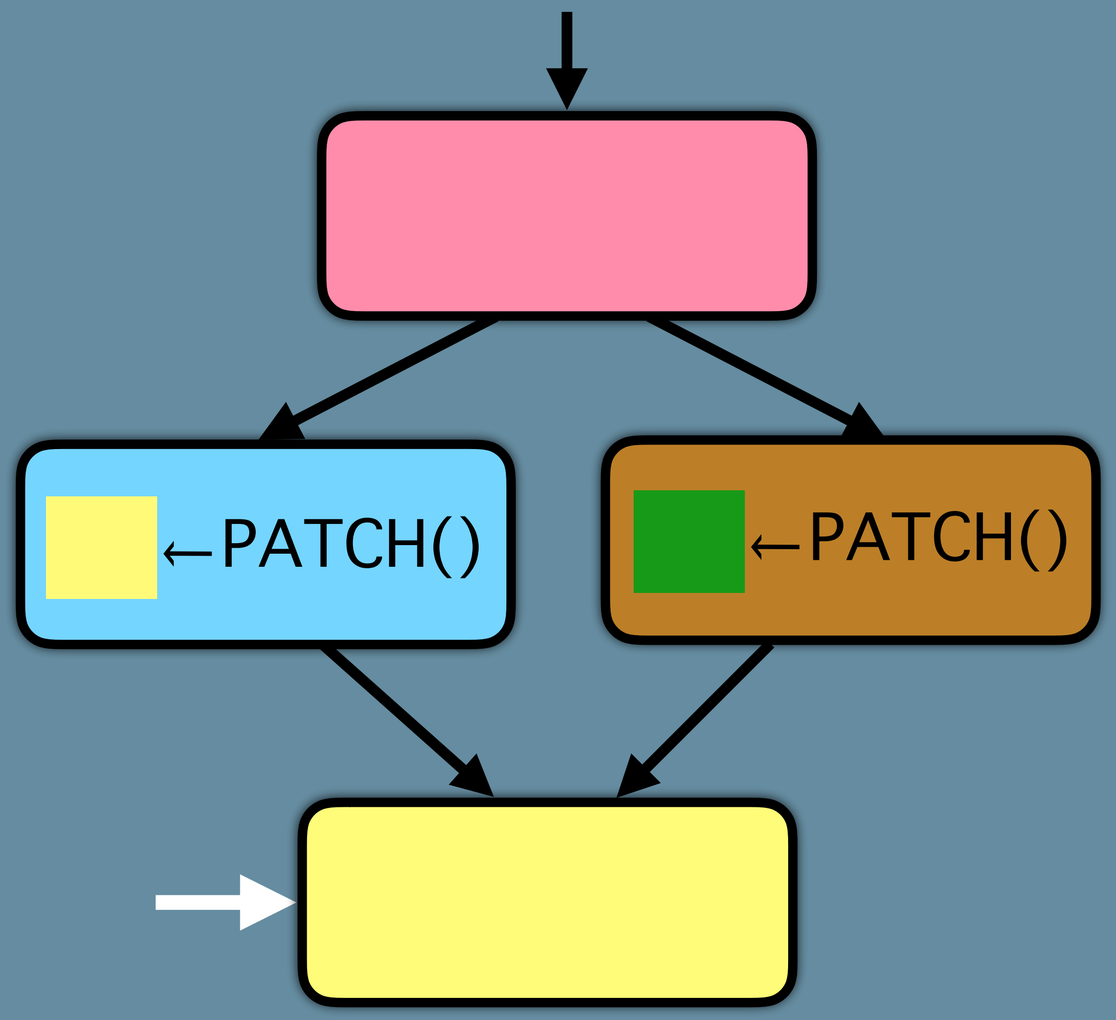
Madou, et al., Software protection through dynamic code mutation, WISA'05



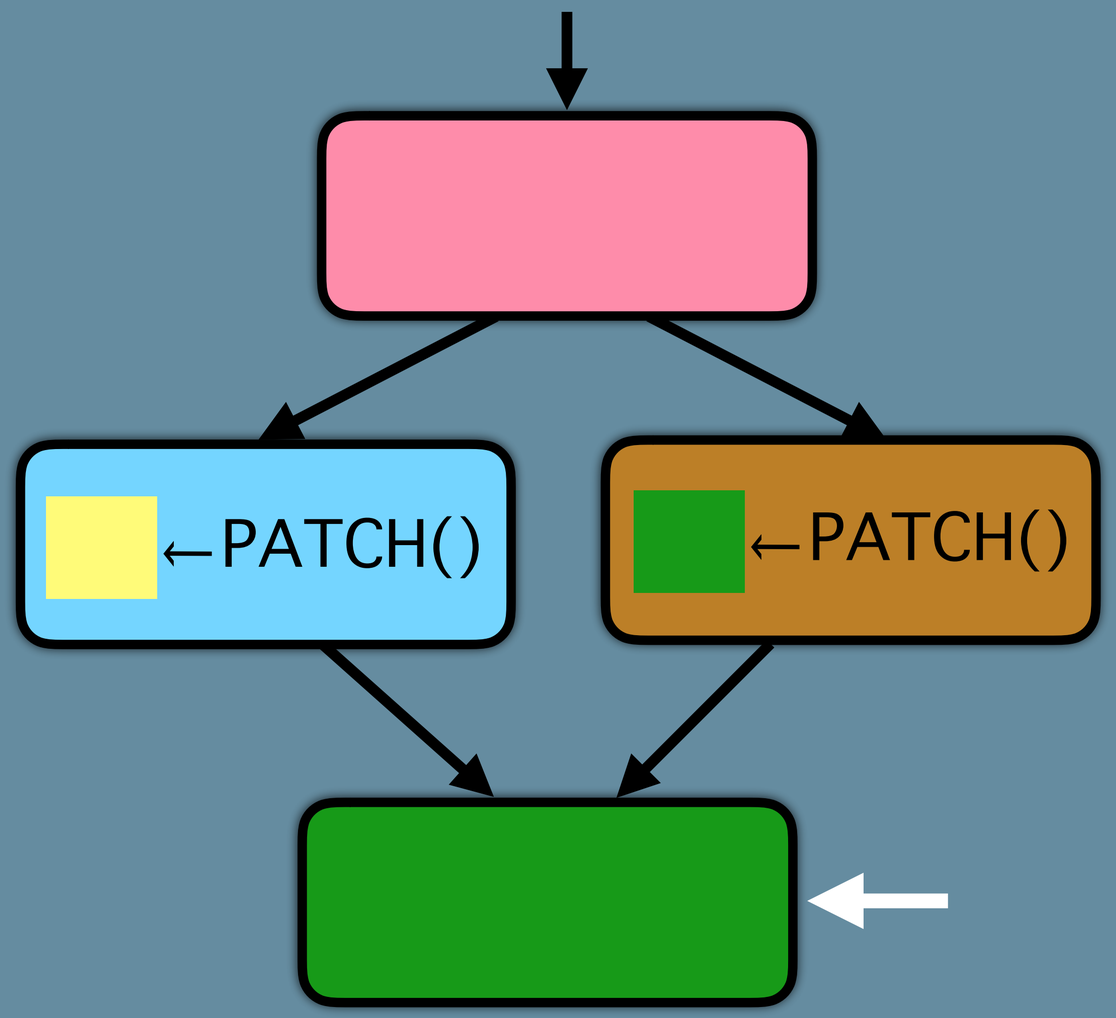
Madou, et al., Software protection through dynamic code mutation, WISA’05



Madou, et al., Software protection through dynamic code mutation, WISA'05



Madou, et al., Software protection through dynamic code mutation, WISA'05



Madou, et al., Software protection through dynamic code mutation, WISA'05

Three Little Languages

‹target,codec,location,transform›

Three Little Languages

Block to encode

‹target, codec, location, transform›

Three Little Languages

How to
ENCODE/
DECODE?

‹target,codec,location,transform›

Three Little Languages

Where to
ENCODE/
DECODE?

‹target,codec,location,transform›

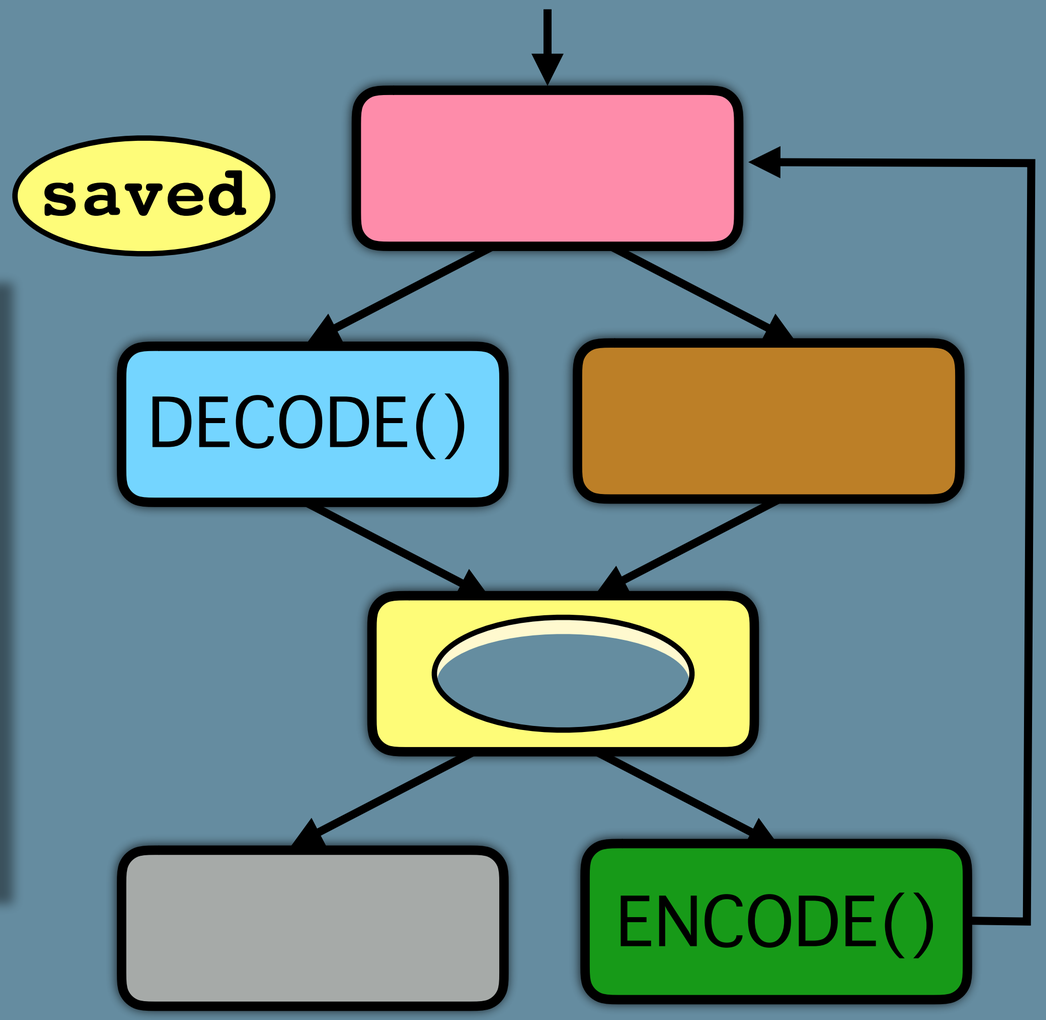
Three Little Languages

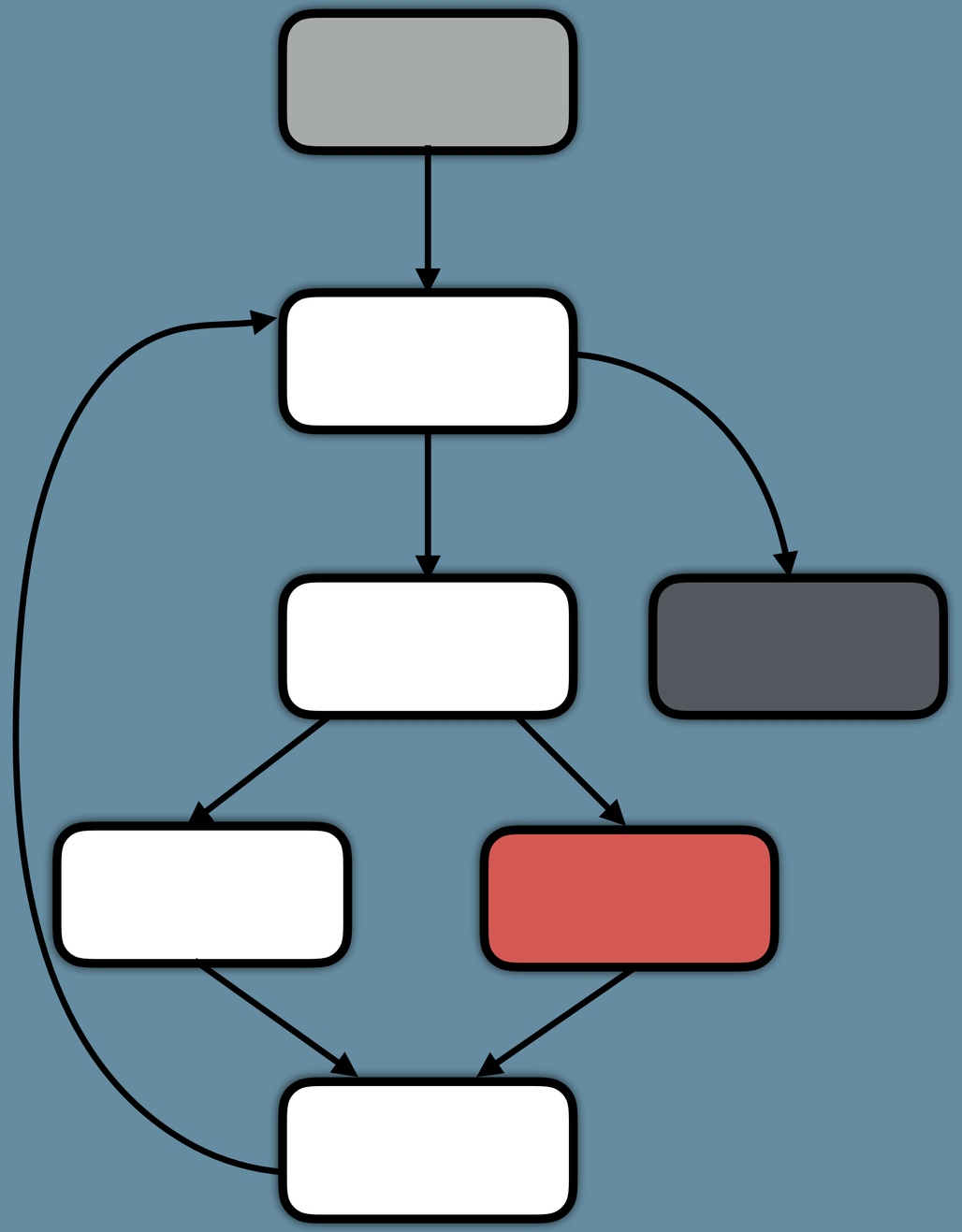
How to modify CFG?

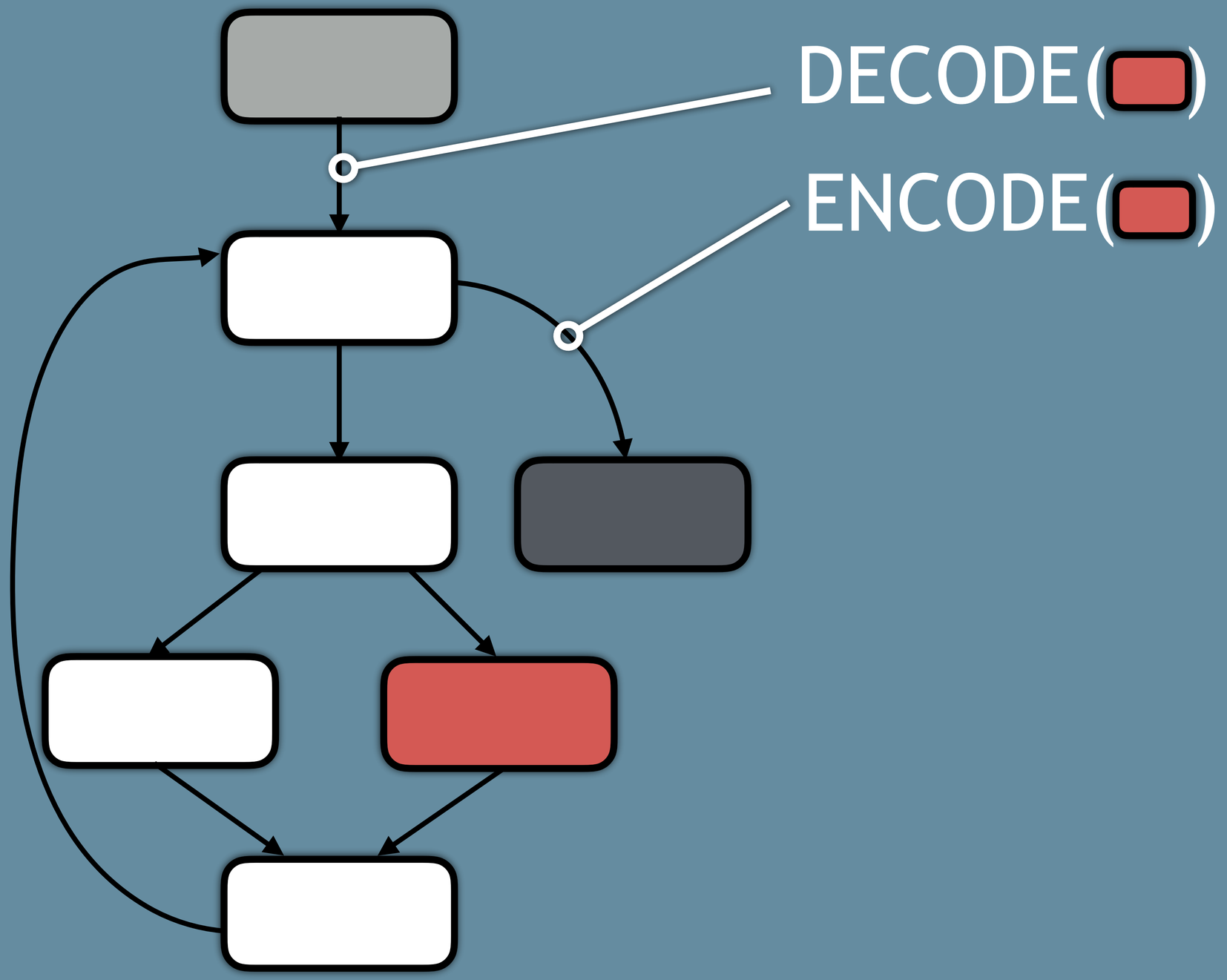
‹target,codec,location,transform›

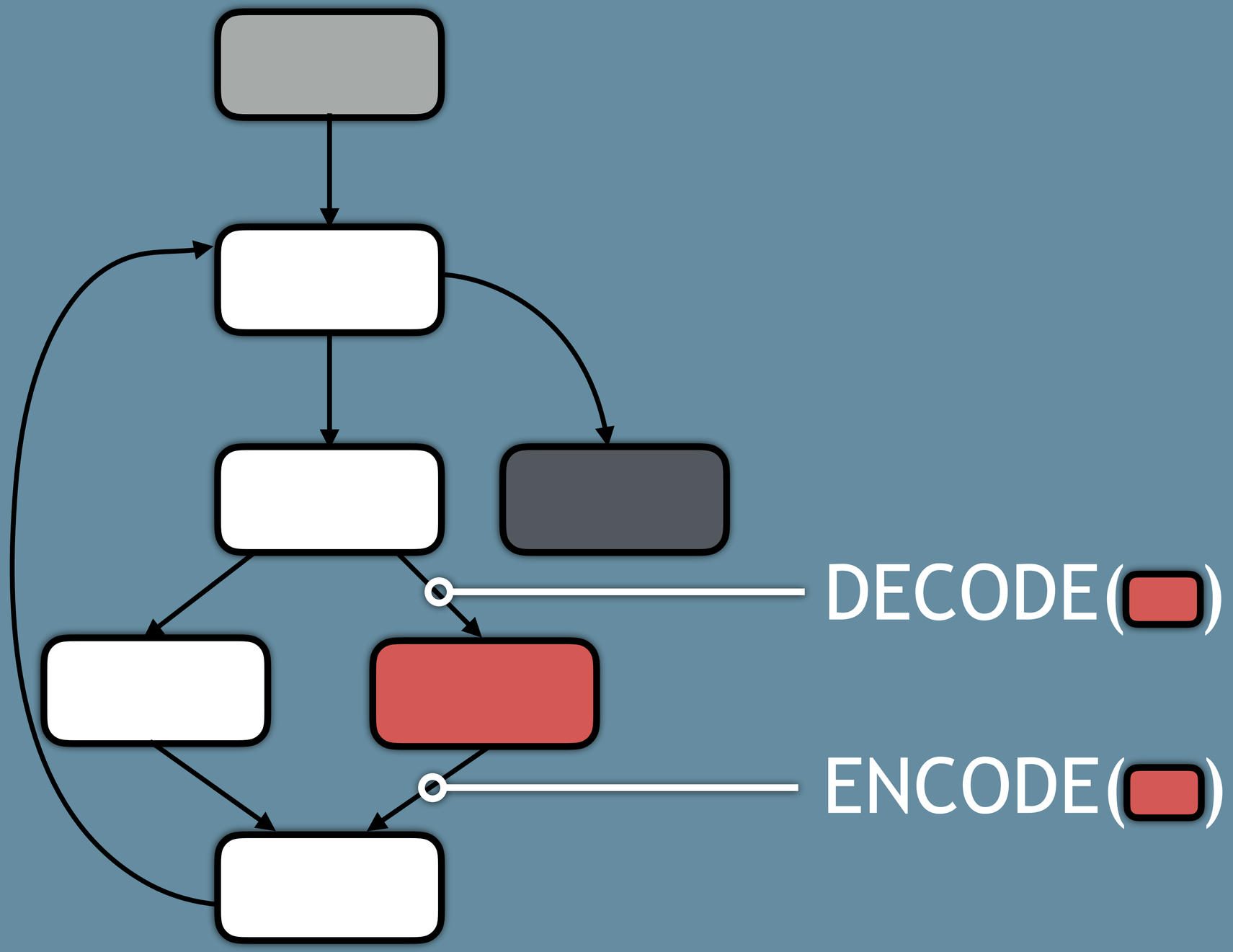
‹target, codec, location, transform›

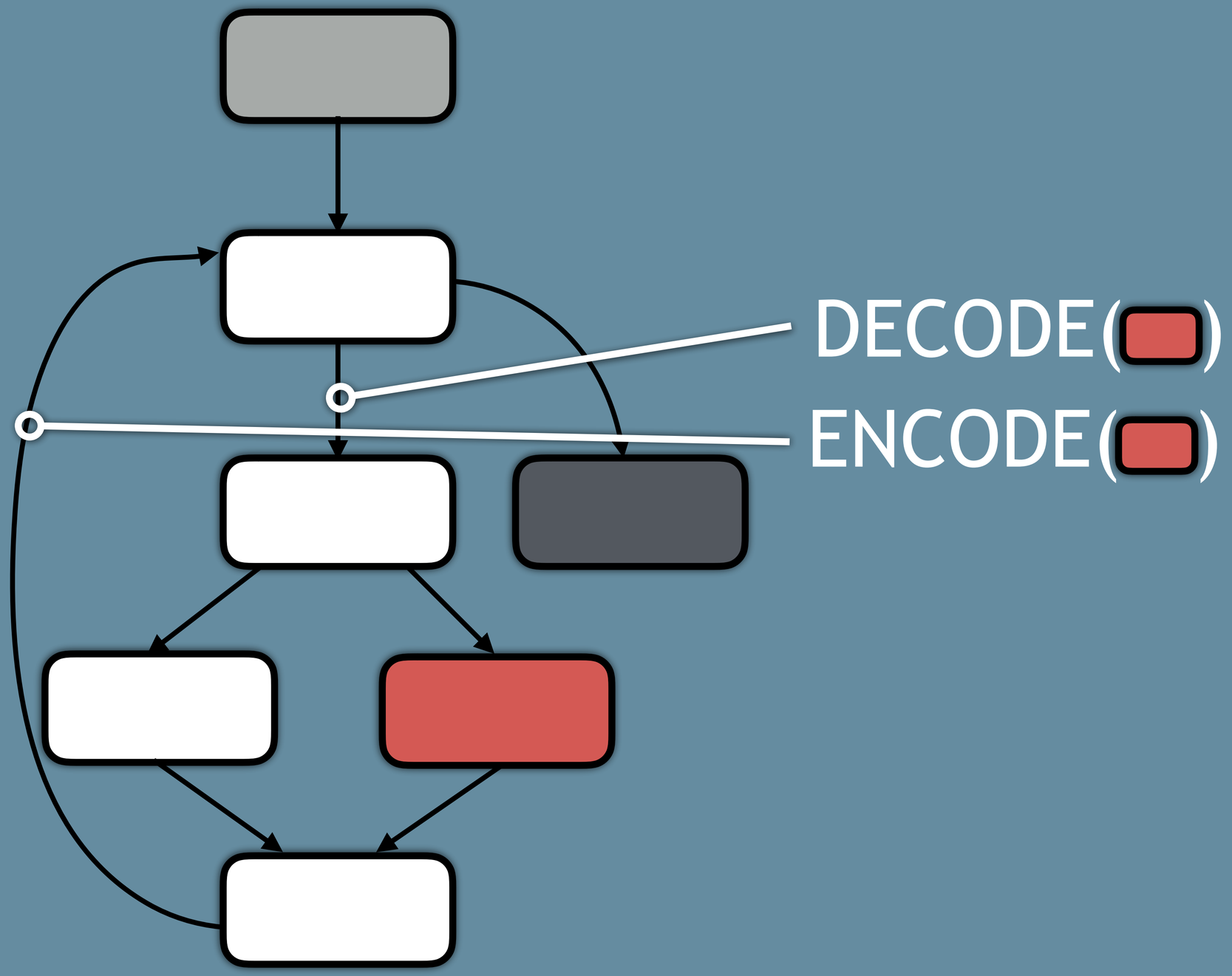
```
encode(offset) : [
  saved ← offset];
  [offset] ← rnd();
]
```

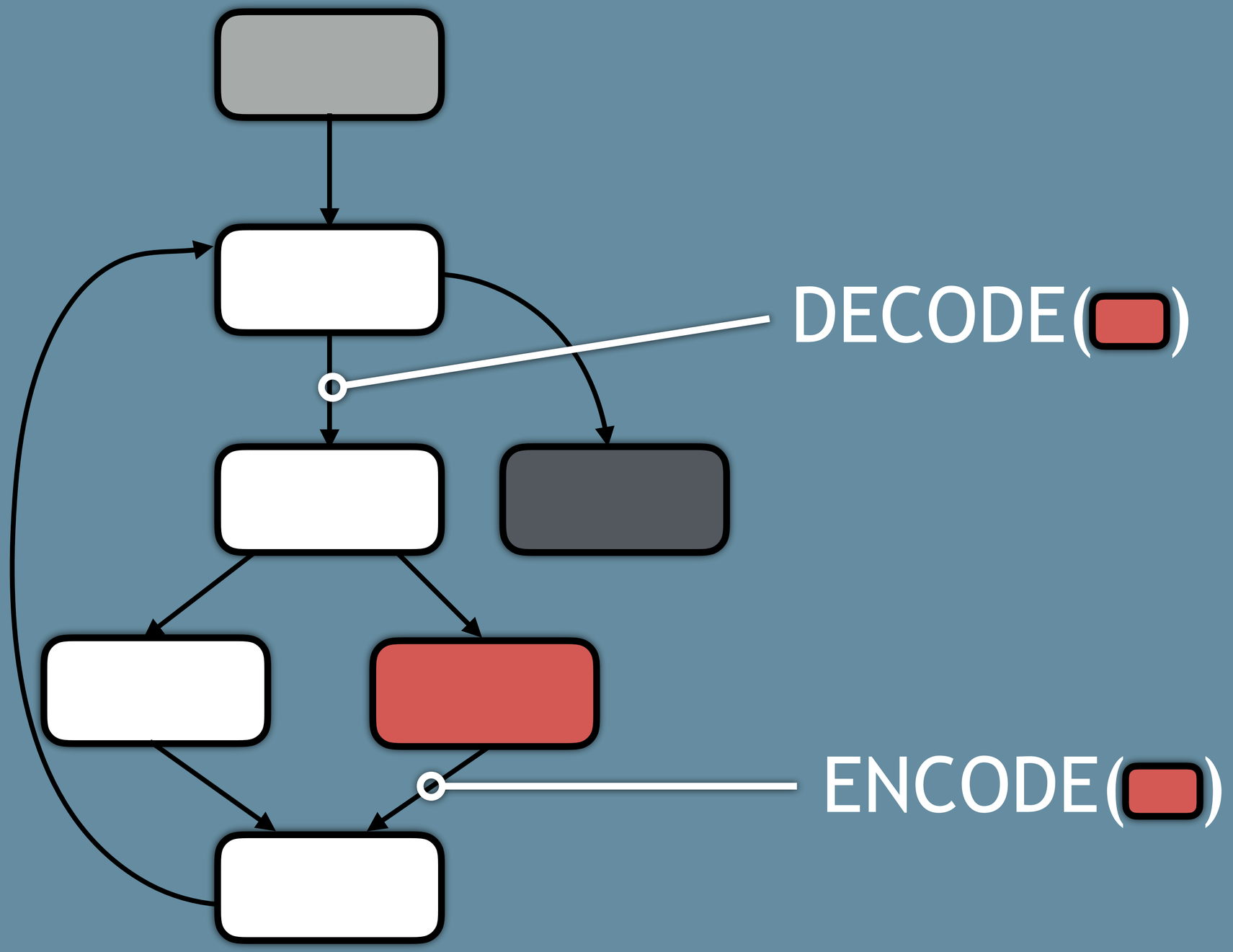


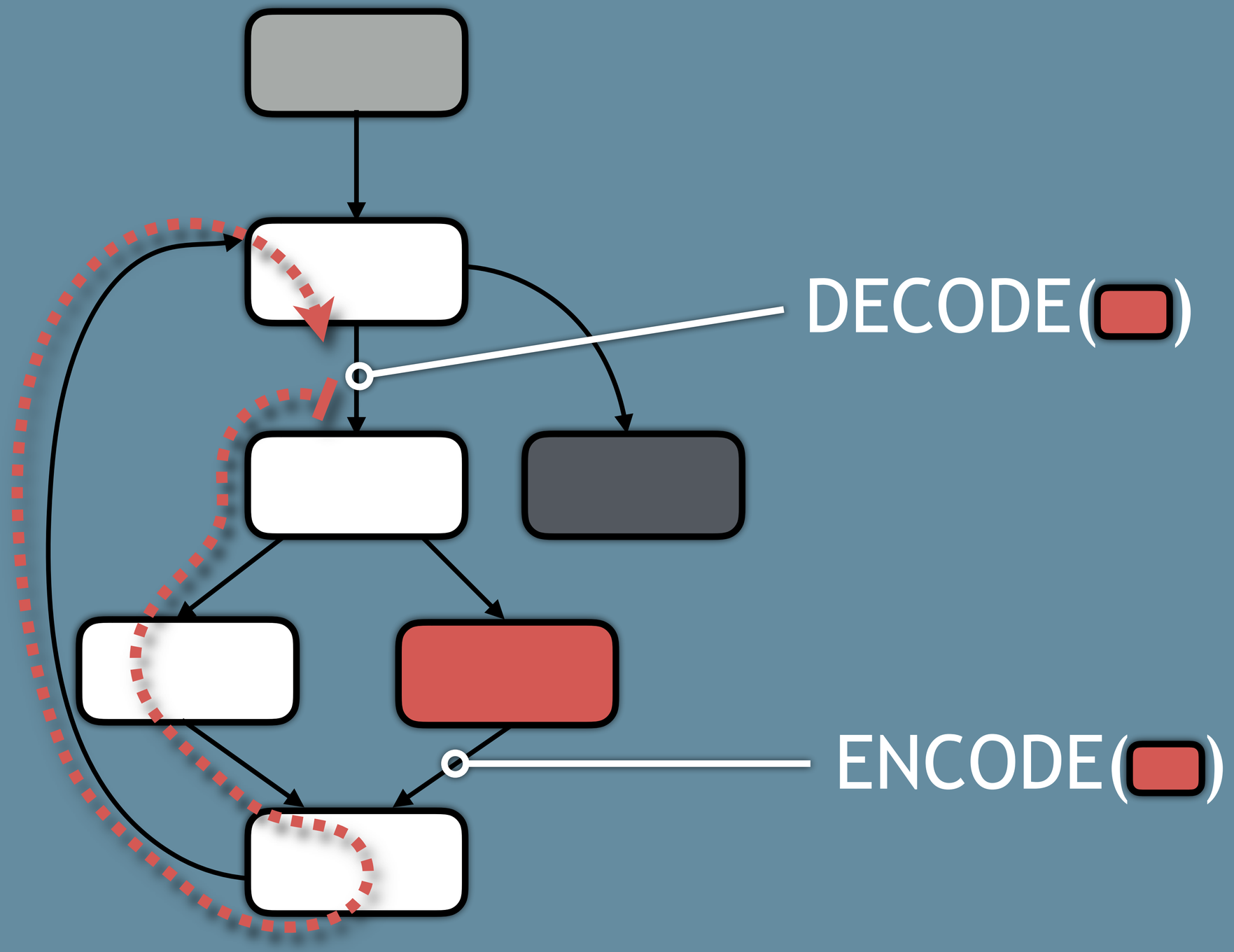



DECODE( )
ENCODE( )


DECODE( )
ENCODE( )

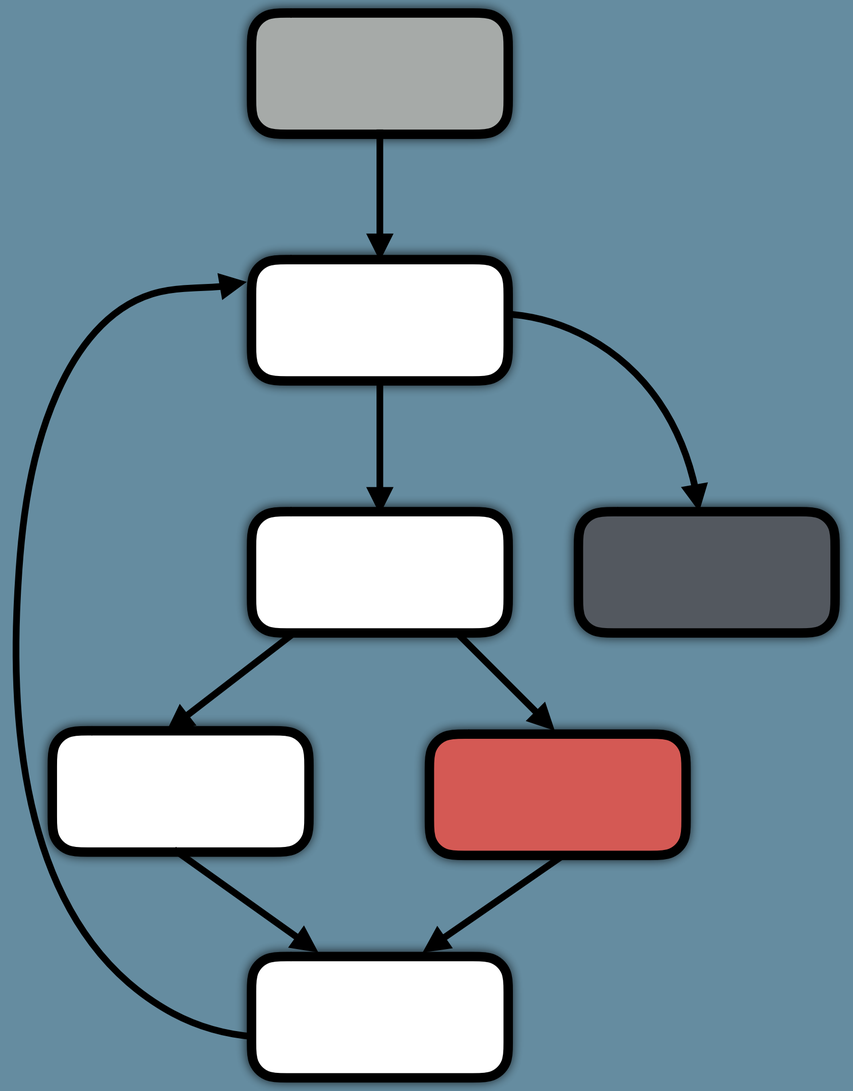

DECODE( )
ENCODE( )


DECODE( )
ENCODE( )

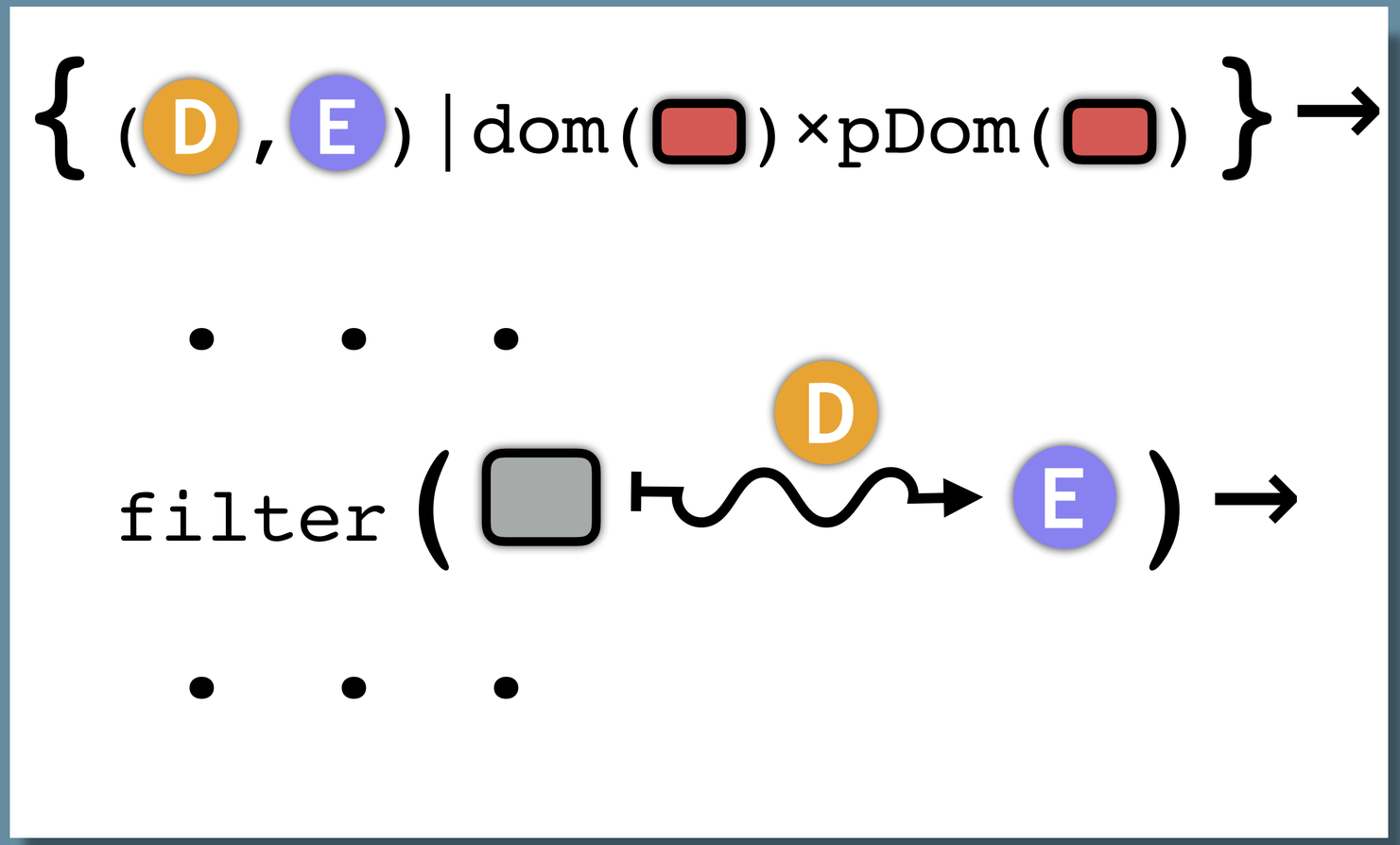

DECODE( )
ENCODE( )

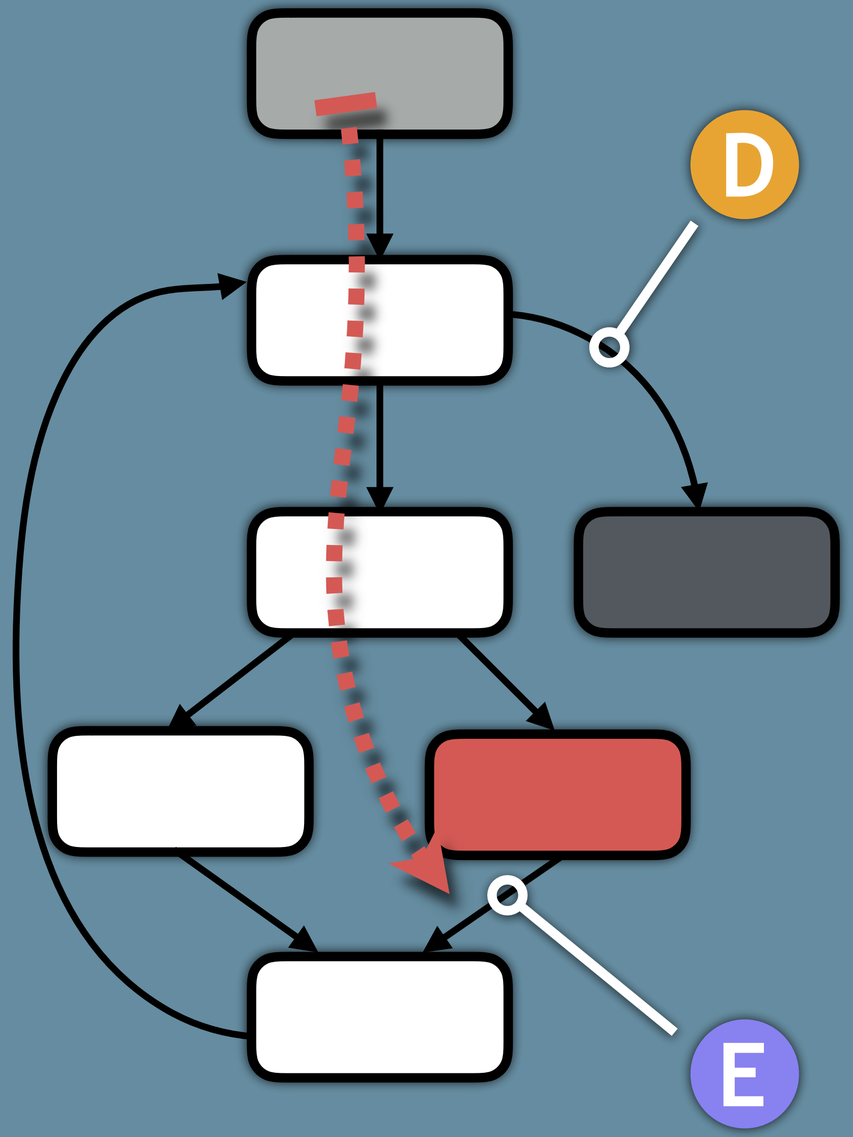
‹target, codec, <u>location</u>, transform›

{ (D, E) | dom( ) × pDom( ) } →

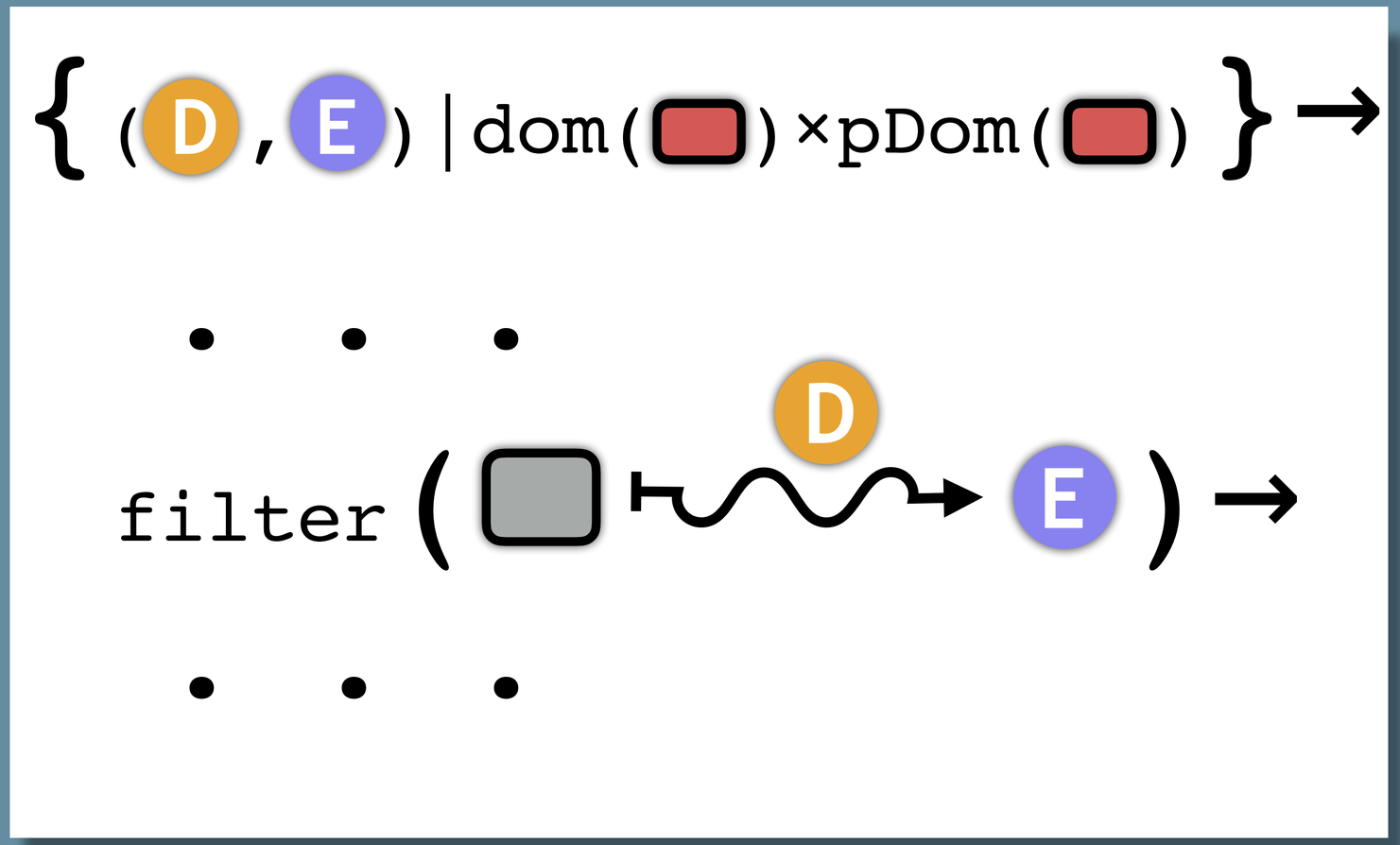


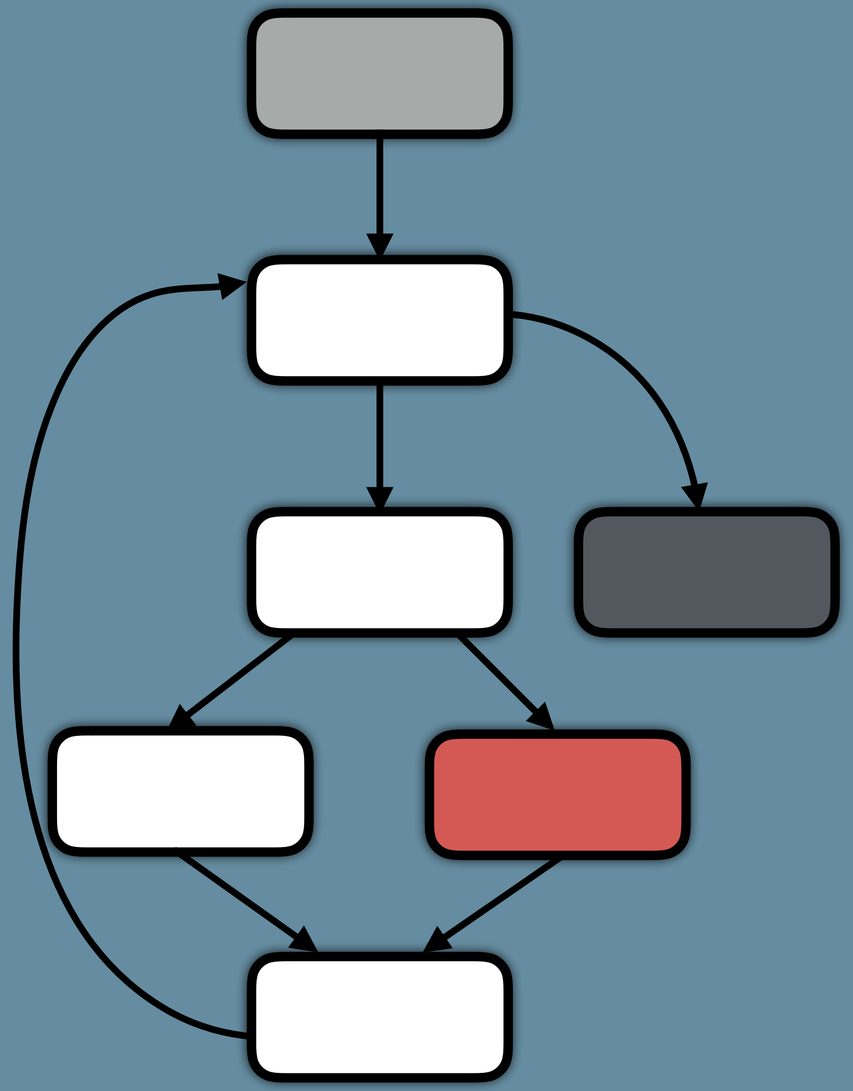
‹target, codec, location, transform›



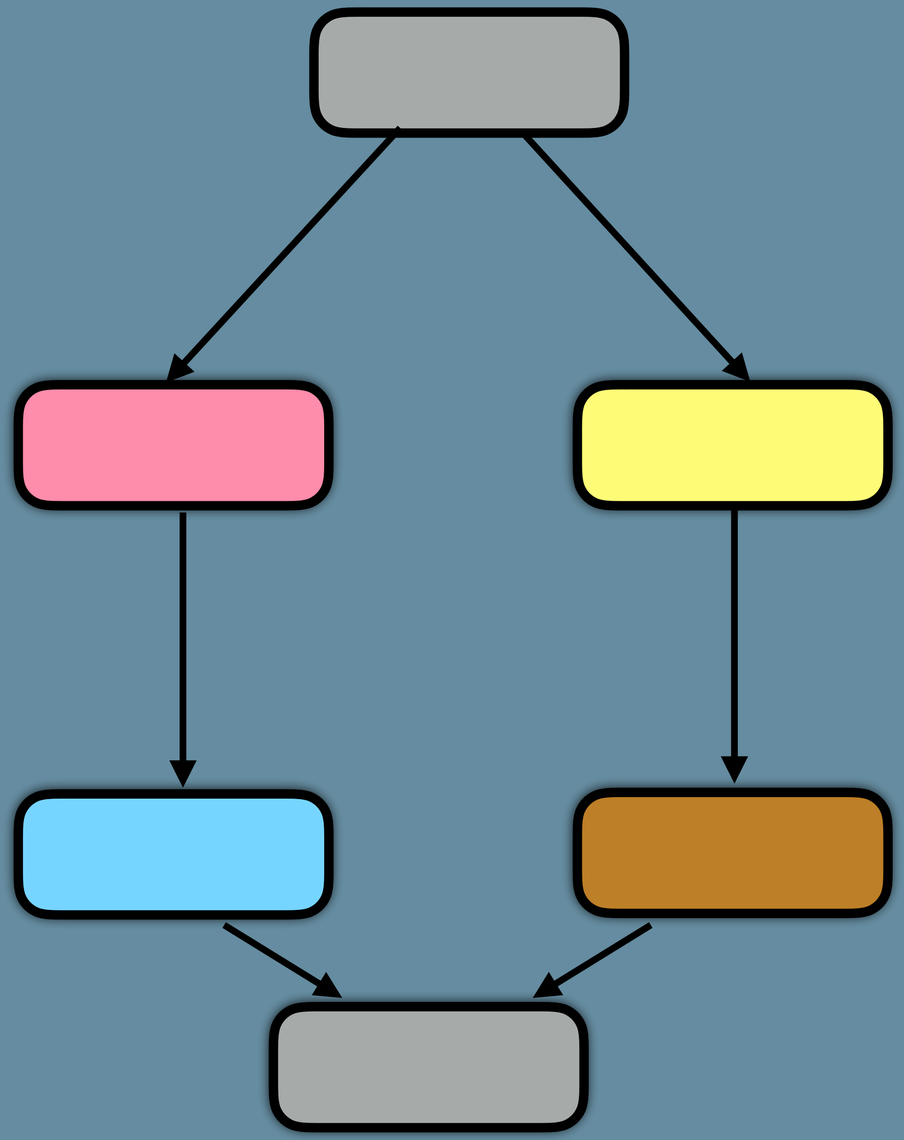


‹target, codec, location, transform›

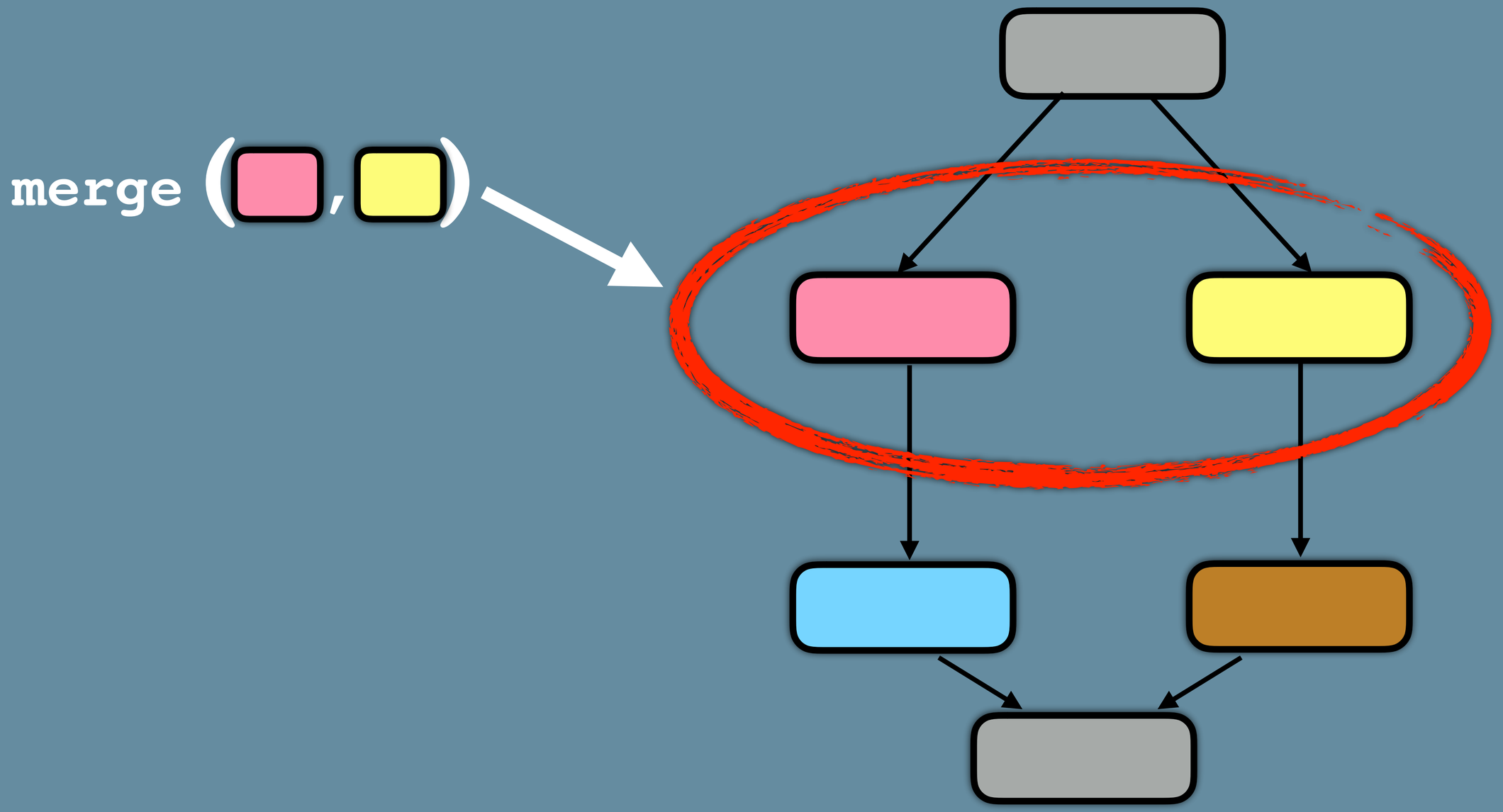




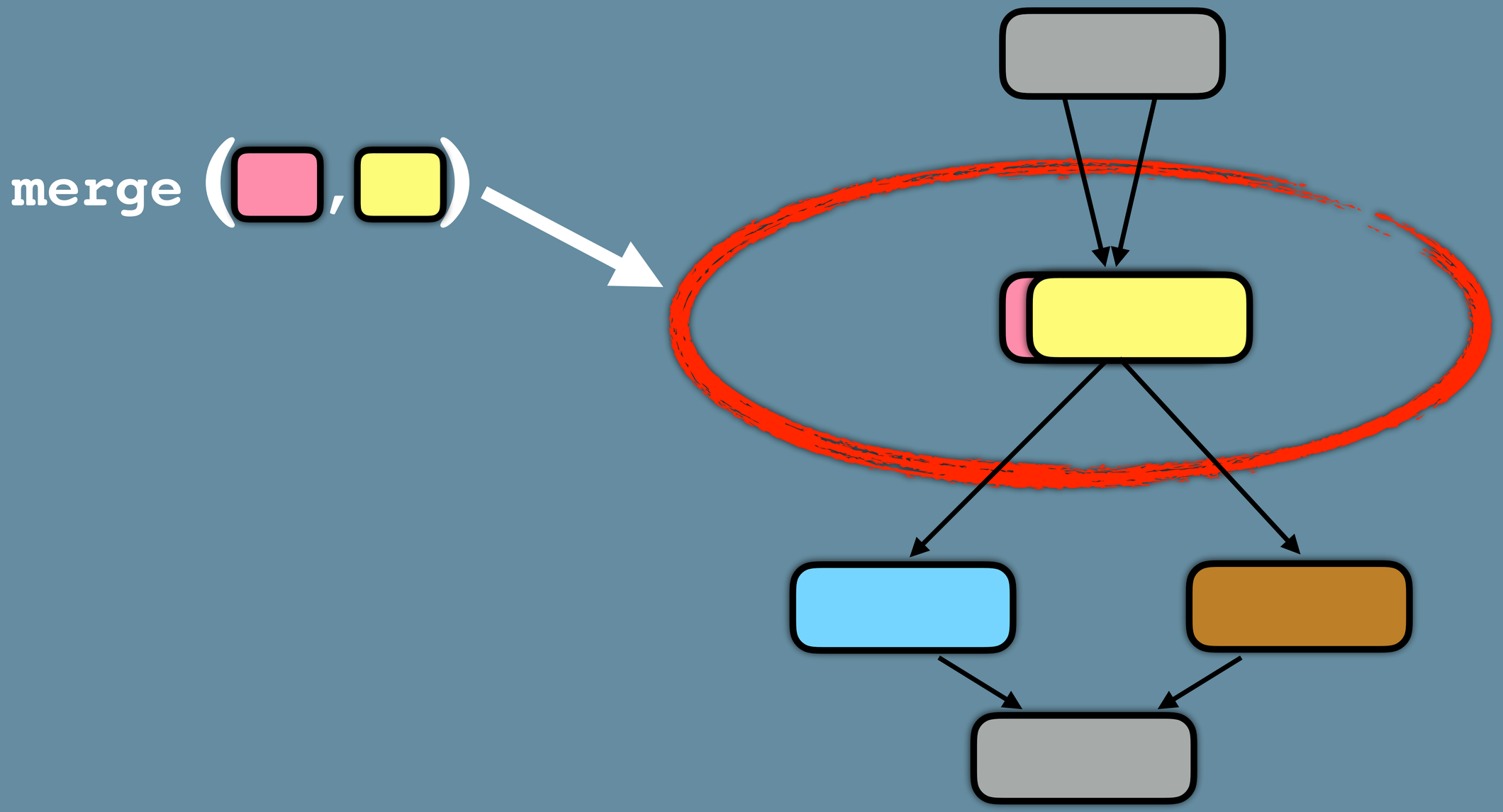
‹target,codec,location,transform›



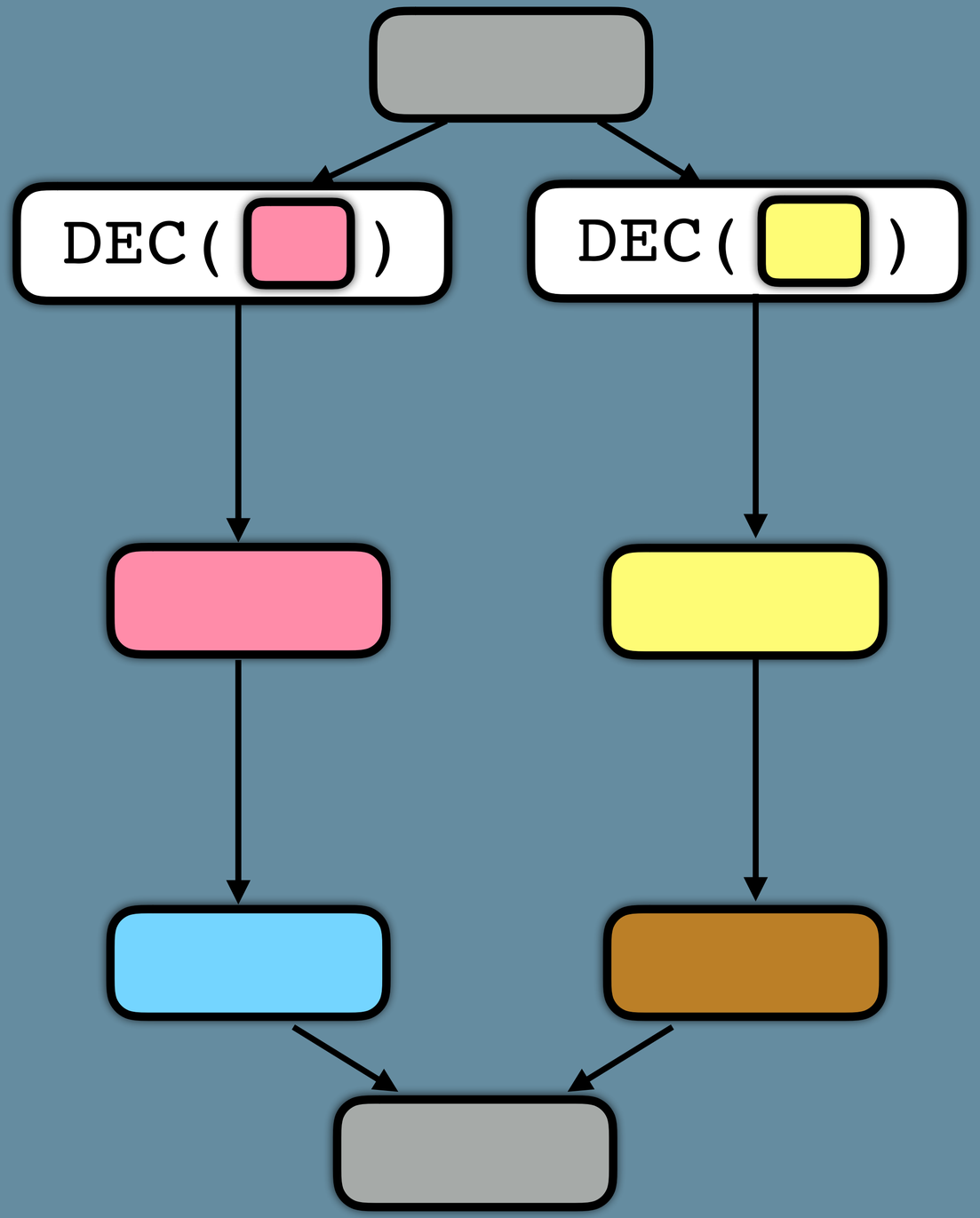
‹target,codec,location,transform›



‹target,codec,location,transform›



```
insert ( DEC( ■ ) )
insert ( DEC( ■ ) )
```

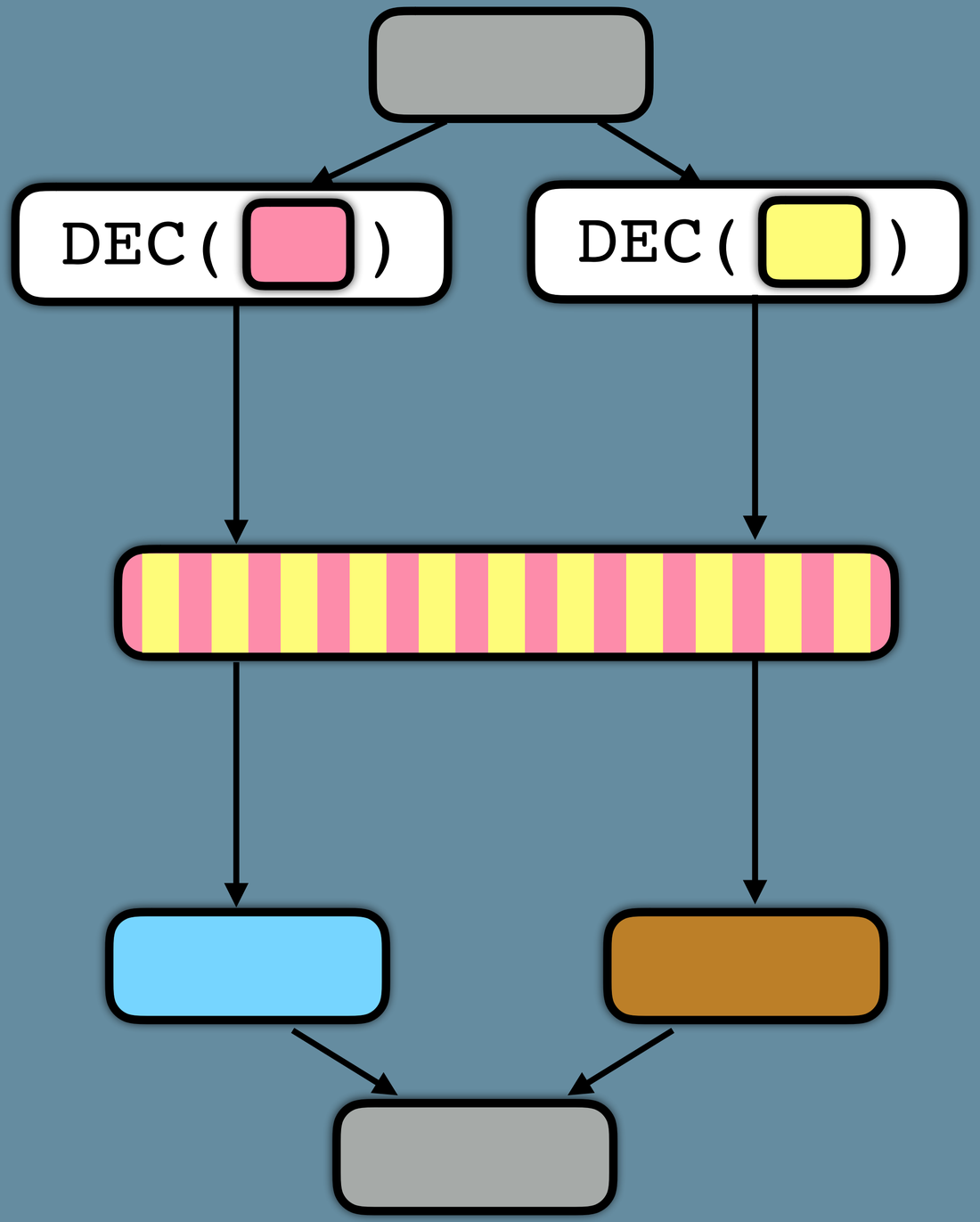


insert (DEC( ))

insert (DEC( ))

pad (,)

merge (,)



insert (DEC( ))

insert (DEC( ))

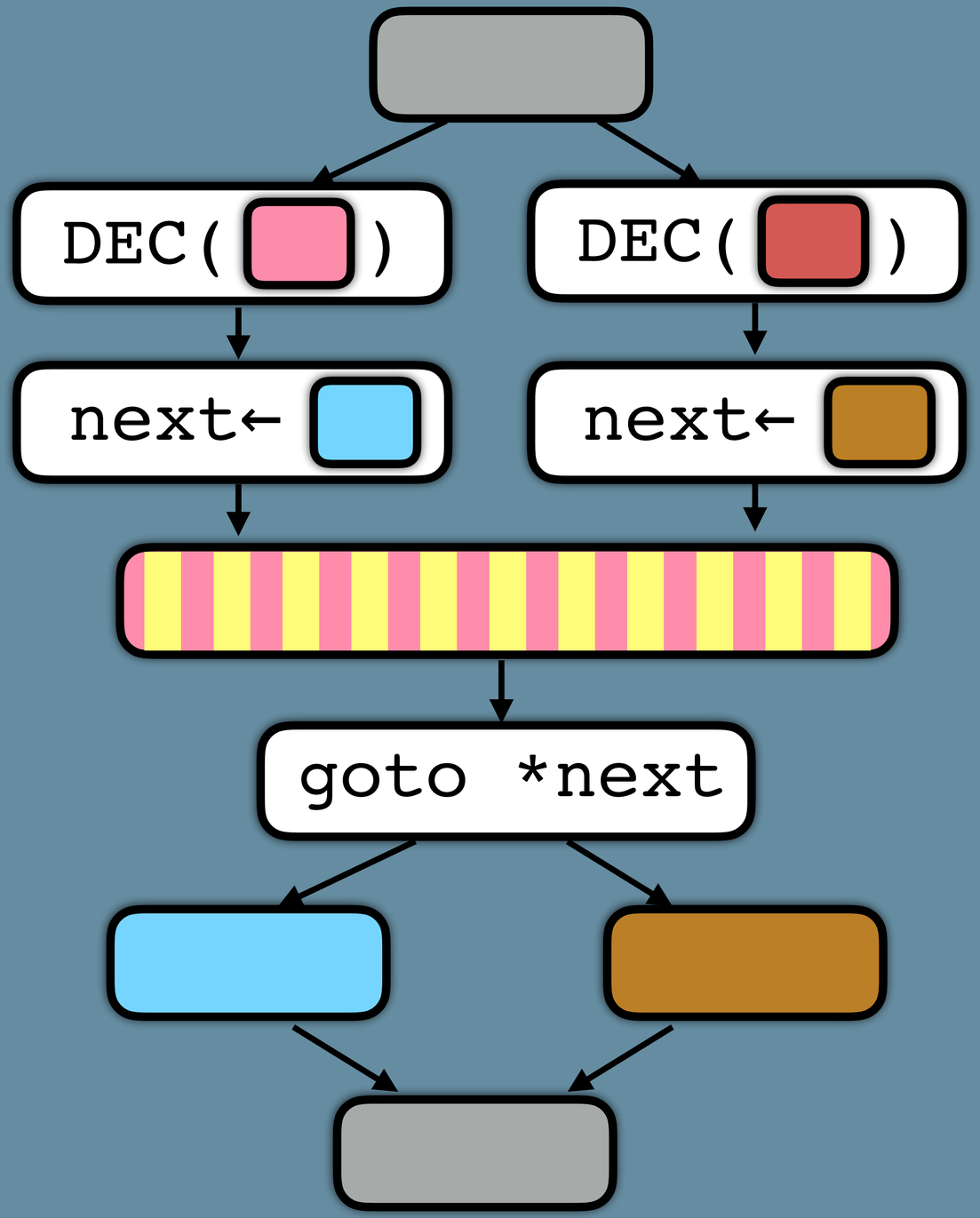
pad (,)

merge (,)

insert (next←)

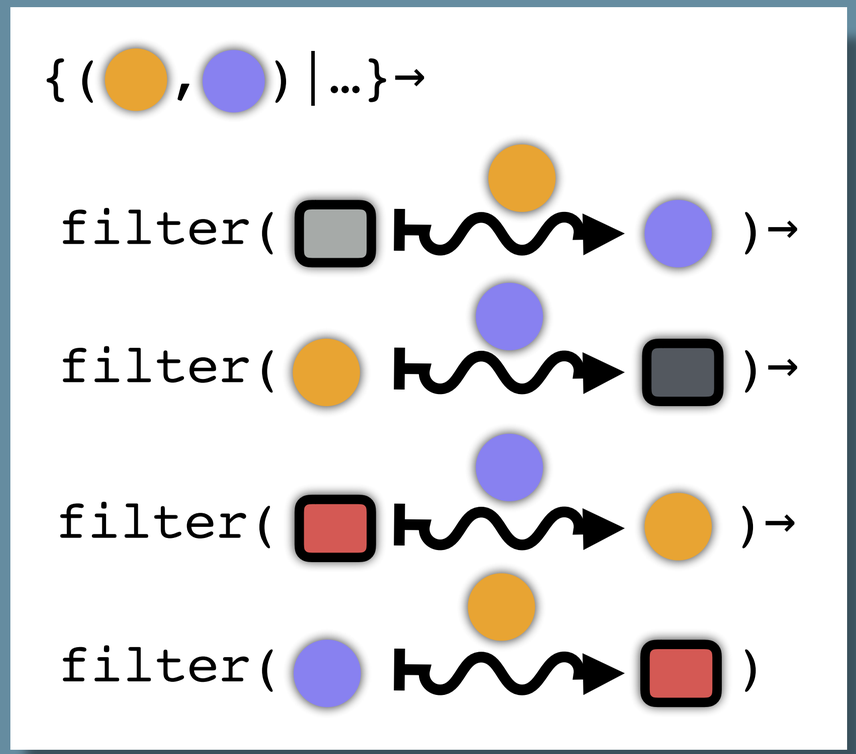
insert (next←)

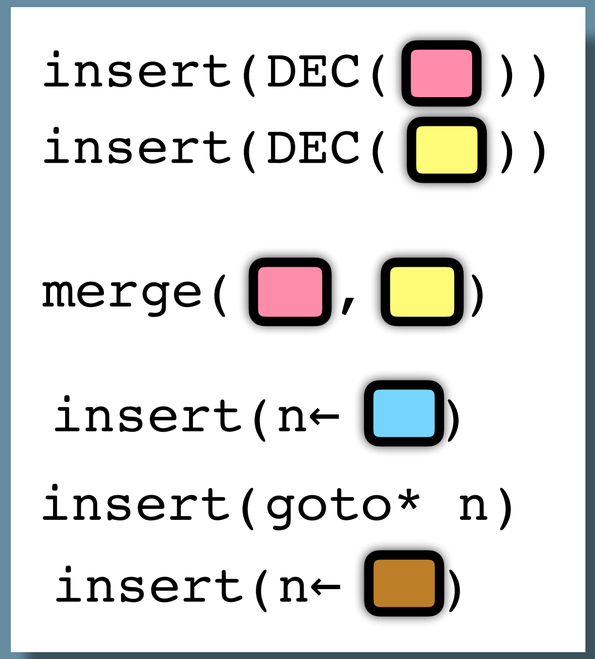
insert (goto *next)



‹target,codec,location,transform›

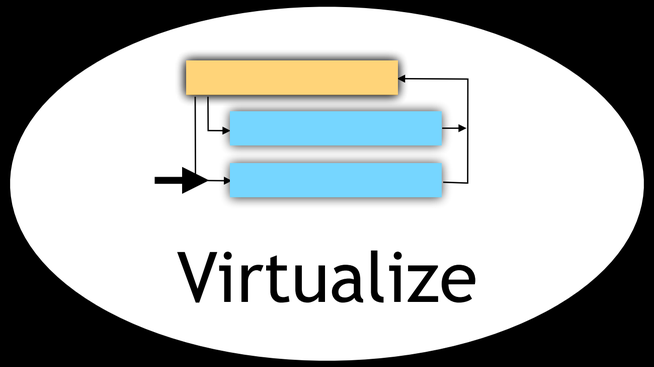
```
encode: [
 block[t][o]←rnd();
]

decode:[
 block[t][o]←0x42;
]
```



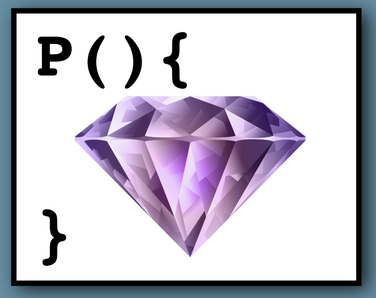


Virtualisation

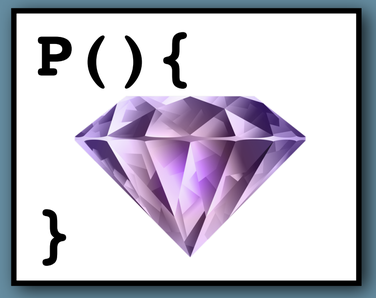
Cirenec



Generic Virtualization



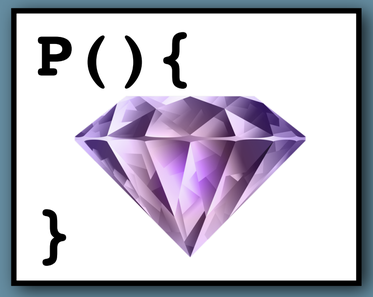
Tigress



Tigress

Virtual Instruction Set

Opcode	Mnemonic	Semantics
0	add	push(pop()+pop())
1	store L	Mem[L]=pop()
2	breq L	if pop()=pop() goto L



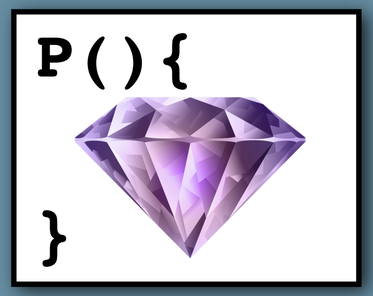
Tigress

Virtual Instruction Set

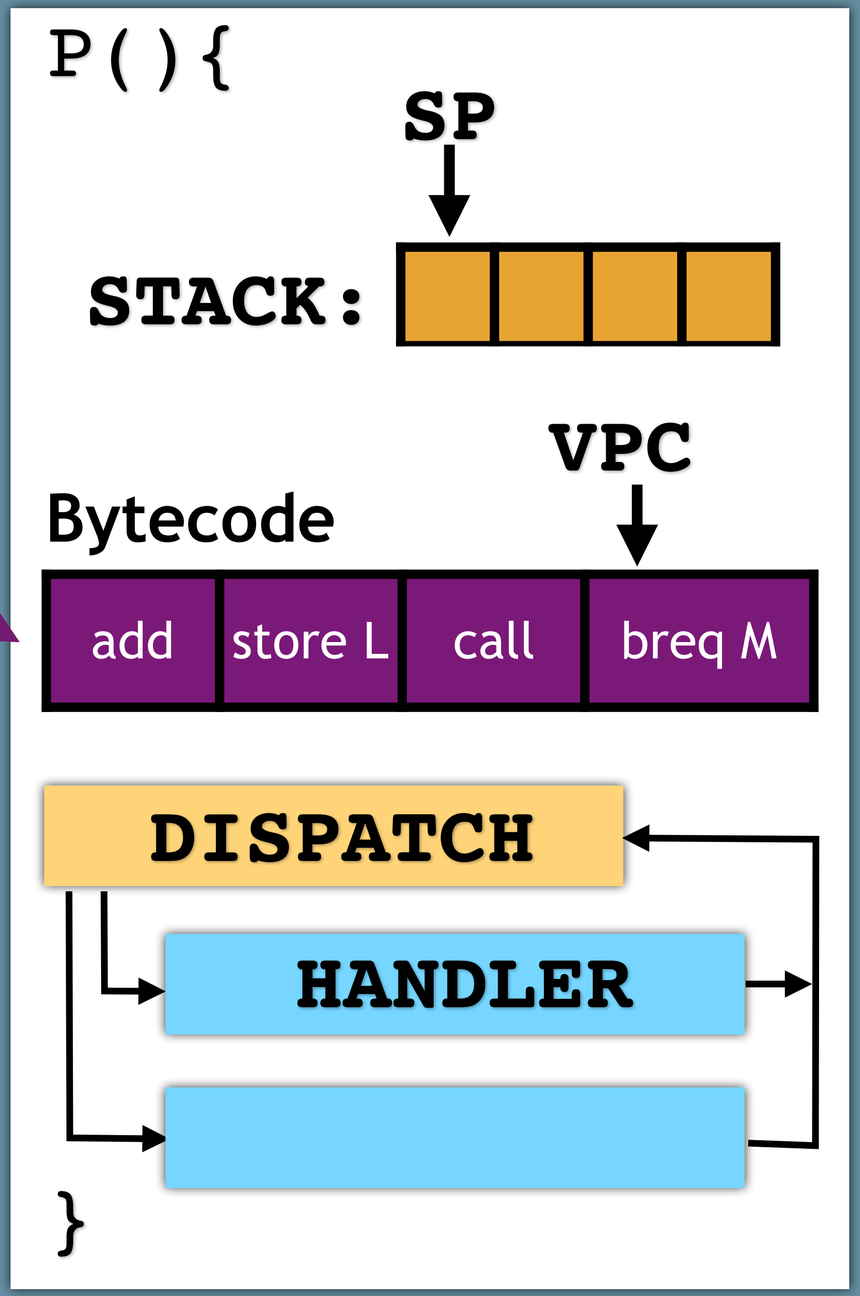
Opcode	Mnemonic	Semantics
0	add	push(pop()+pop())
1	store L	Mem[L]=pop()
2	breq L	if pop()=pop() goto L

Bytecode

add	store L	call	breq M

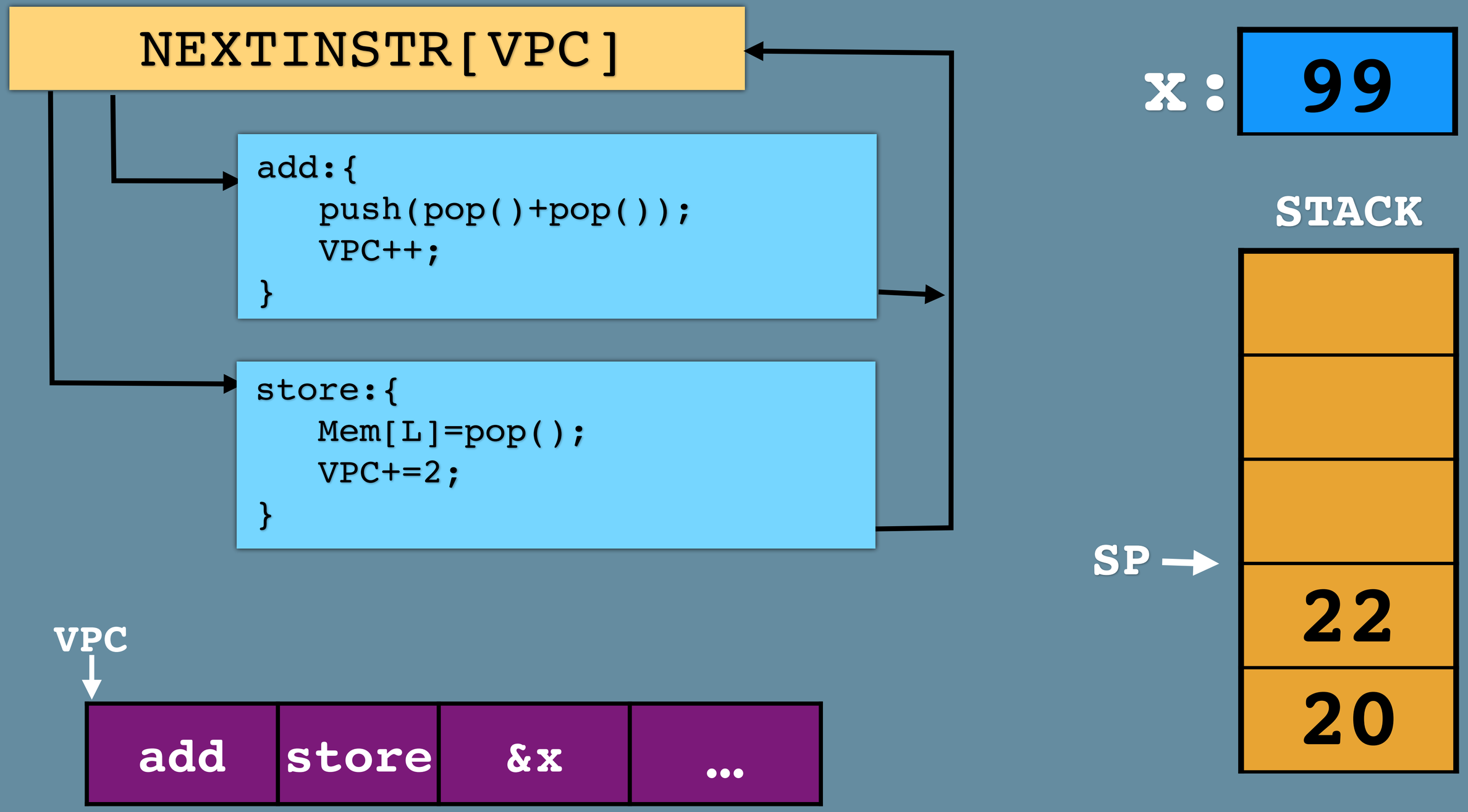


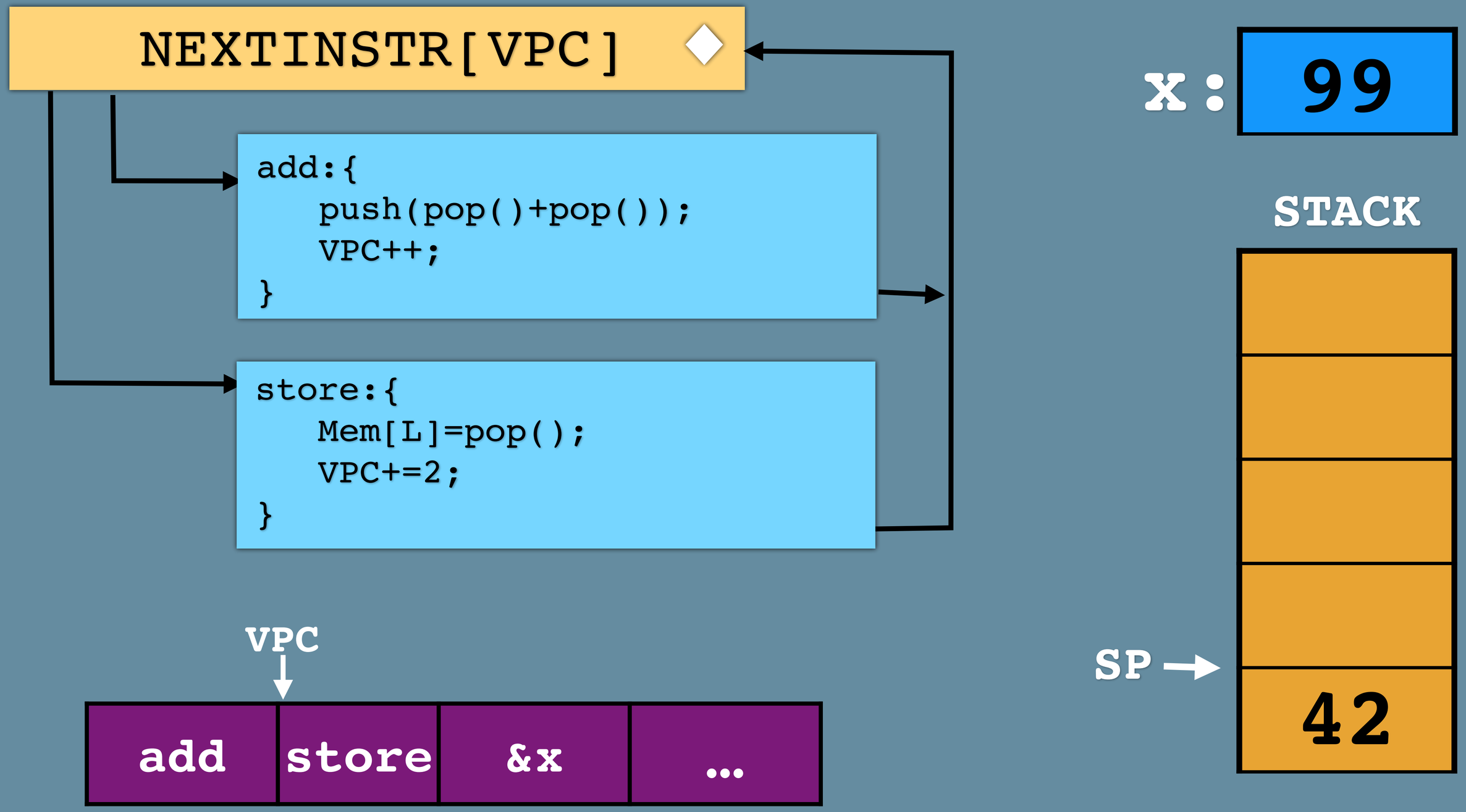
Tigress



Virtual Instruction Set

Opcode	Mnemonic	Semantics
0	add	push(pop()+pop())
1	store L	Mem[L]=pop()
2	breq L	if pop()=pop() goto L


NEXTINSTR[VPC]
add:{
push(pop()+pop());
VPC++;
}
store:{
Mem[L]=pop();
VPC+=2;
}
x:
99
STACK
SP
22
20
VPC
add
store
&x
...


NEXTINSTR[VPC]
add:{
push(pop()+pop());
VPC++;
}
store:{
Mem[L]=pop();
VPC+=2;
}
x:
99
STACK
SP
42
VPC
add
store
&x
...

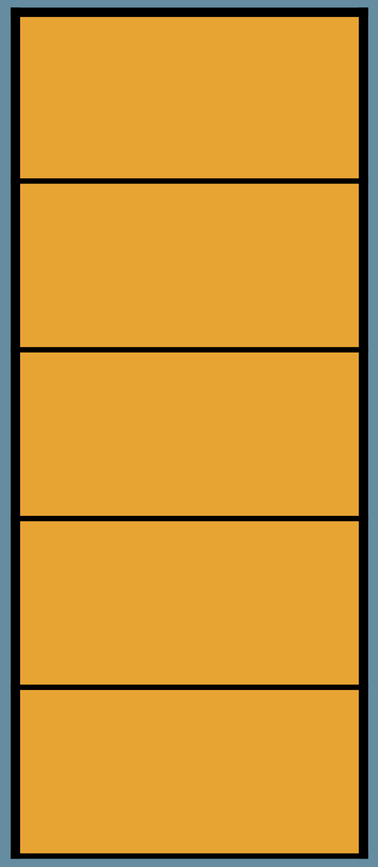
NEXTINSTR[VPC] ◆

```
add:{
    push(pop()+pop());
    VPC++;
}
```

```
store:{
    Mem[L]=pop();
    VPC+=2;
}
```

x : 42

STACK



SP

VPC

add	store	&x	...

Generic Virtualization

‹target,ISA,dispatch,dynamic,stealth›

Generic Virtualization

Arbitrarily complex
instruction set

‹target,ISA,dispatch,dynamic,stealth›

Generic Virtualization

Multiple dispatch techniques

‹target,ISA,dispatch dynamic,stealth›

Generic Virtualization

‹target,ISA,dispatch,dynamic,stealth›

- Self-Modifying VM Code!
- VM Code in Constant Flux!
- VM Code Never Fully in the Clear!

Generic Virtualization

Mix VM and
real code

‹target,ISA,dispatch,dynamic,stealth›

‹ISA, dispatch, dynamic, stealth›

```
c ← add a b
c ← sub a b
breq a b
```

⟨ISA, dispatch, dynamic, stealth⟩

```
c ← add a b
c ← sub a b
breq a b
```

Superoperators →

- `d ← add_add a b c`
- `d ← add_sub a b c`
- `breq_add_add a b c d`

Proebsting, Optimizing an ANSI C interpreter with superoperators, POPL'95

⟨ISA, dispatch, dynamic, stealth⟩

```
c ← add a b
c ← sub a b
breq a b
```

Superoperators →

- `d ← add_add a b c`
- `d ← add_sub a b c`
- `breq_add_add a b c d`

Random Operands ↓

- `d/stack ← add_sub_1 a/stack b/stack c/stack`
- `d/reg ← add_sub_2 a/reg b/reg c/reg`
- `d/reg ← add_sub_3 a/stack b/reg c/stack`

‹ISA, dispatch, dynamic, stealth›

```
c ← add a b
c ← sub a b
breq a b
```

Superoperators →

- `d ← add_add a b c`
- `d ← add_sub a b c`
- `breq_add_add a b c d`

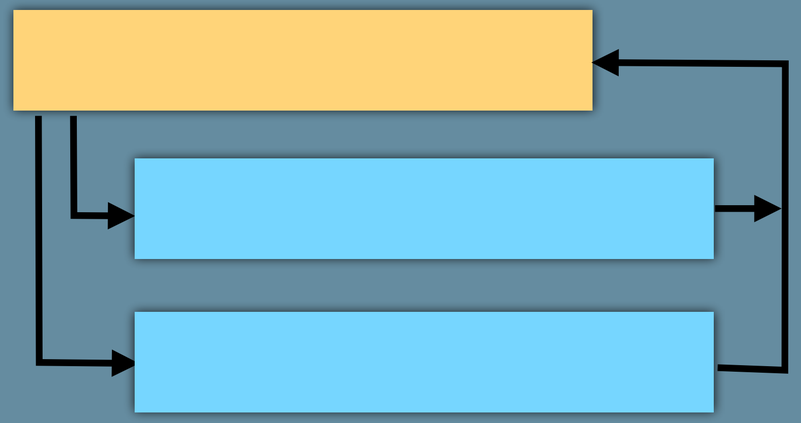
↓ Random Operands

- `d/stack ← add_sub_1 a/stack b/stack c/stack`
- `d/reg ← add_sub_2 a/reg b/reg c/reg`
- `d/reg ← add_sub_3 a/stack b/reg c/stack`

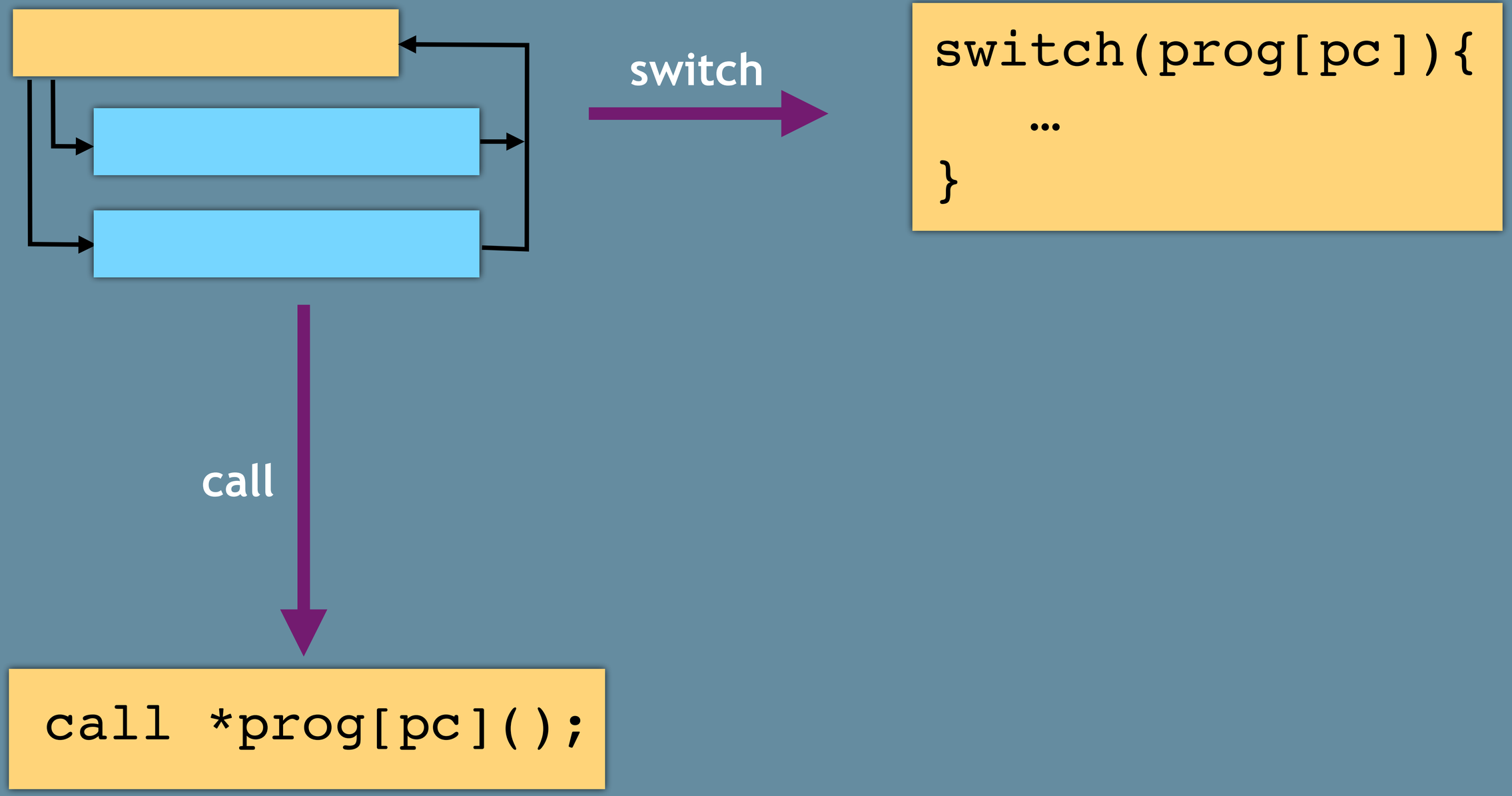
→ Random Opcodes

Opcode	Mnemonic	Semantics
42	add_sub_1	(a-b)+c
96	add_sub_2	(a-b)+c
252	breq_add_add	breq (a+b+c) L

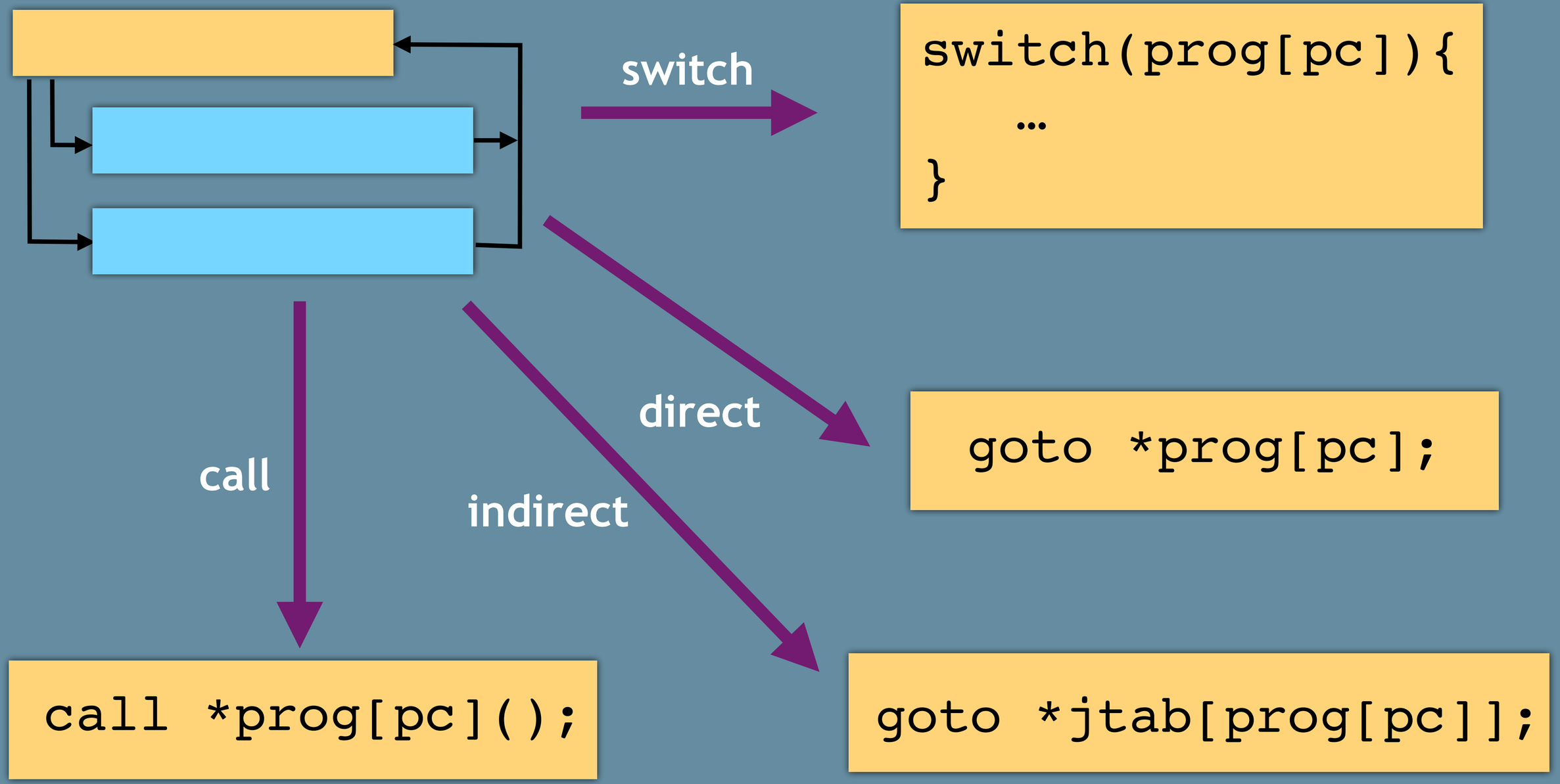
‹ISA, dispatch, dynamic, stealth›



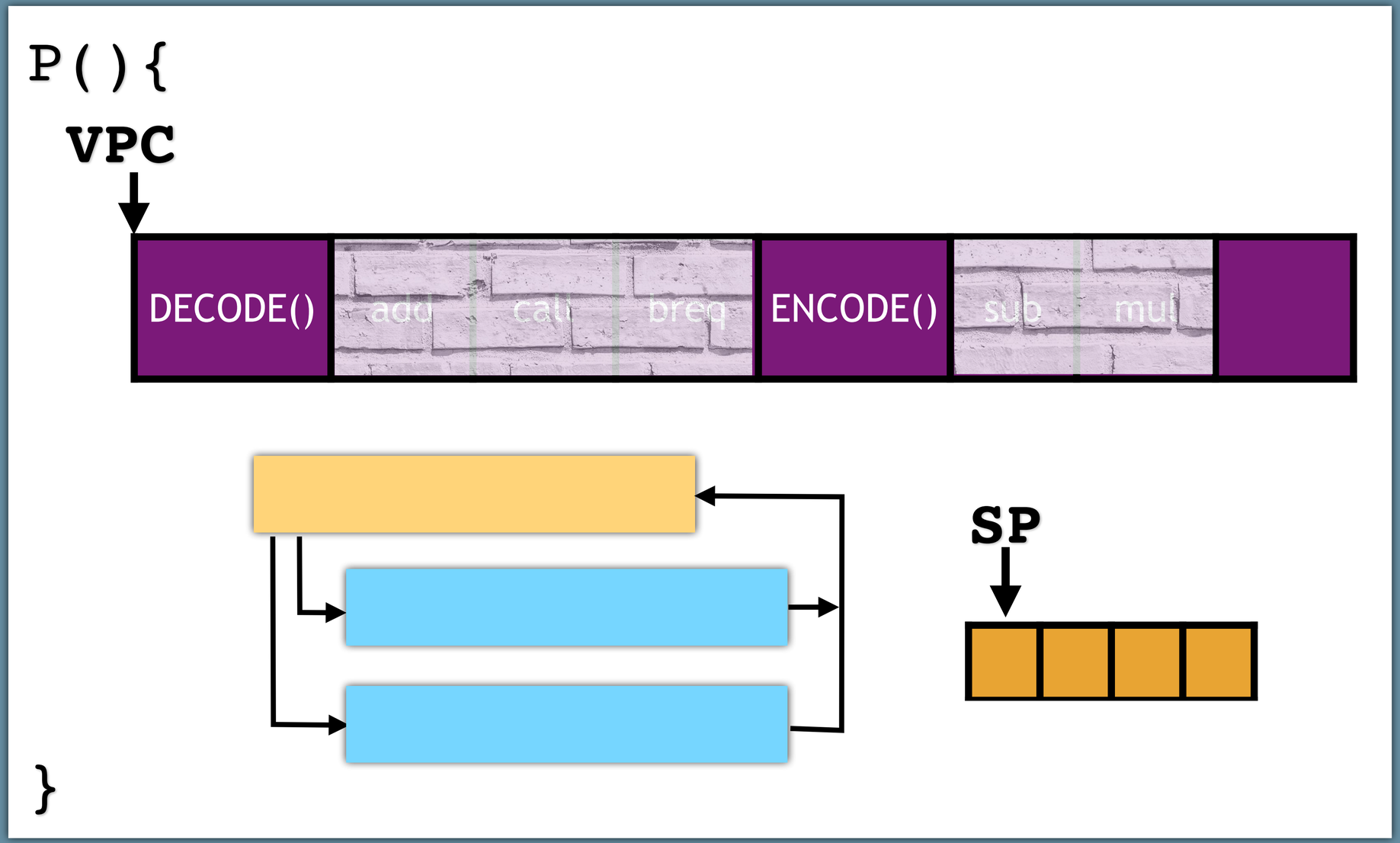
‹ISA, dispatch, dynamic, stealth›



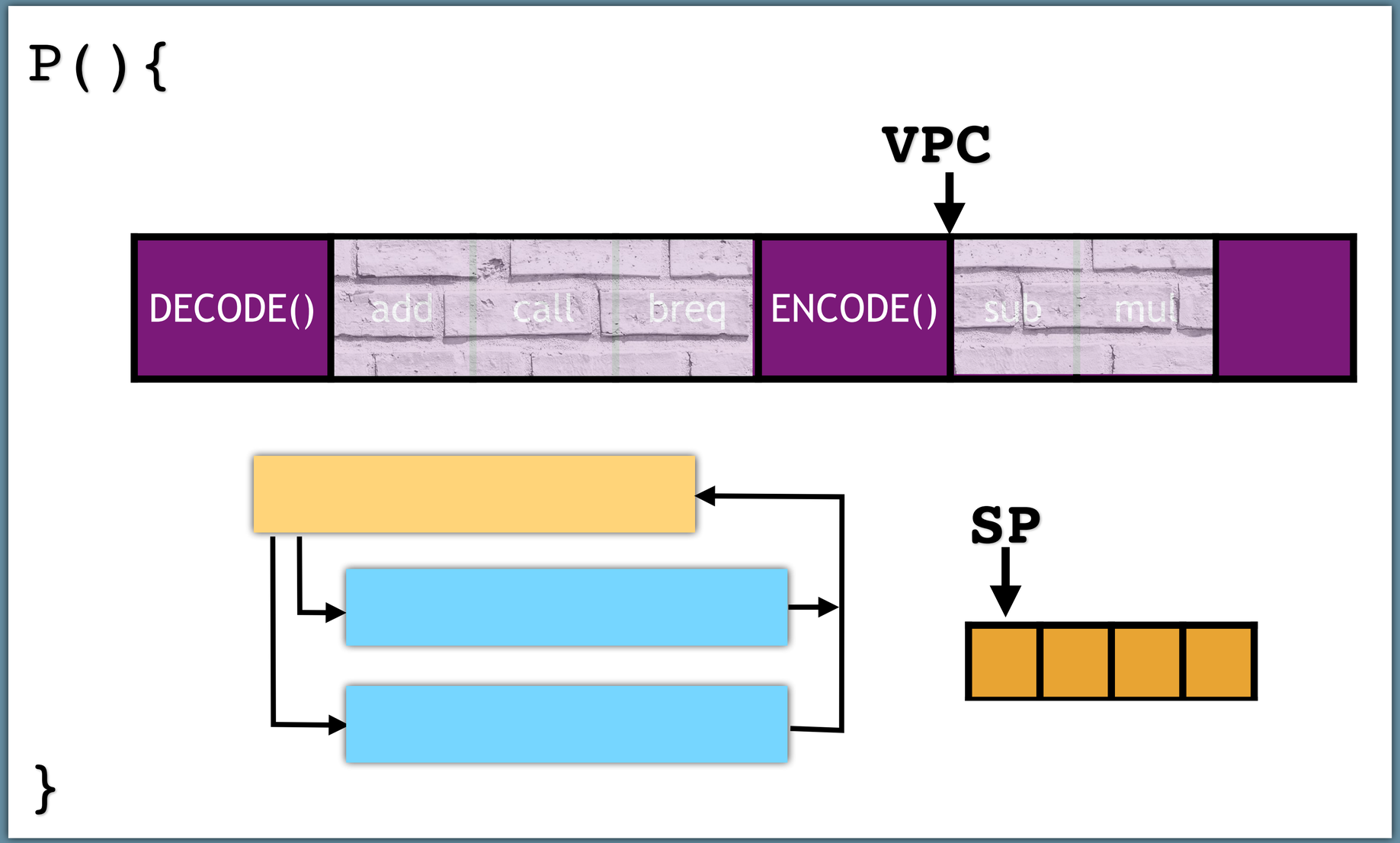
‹ISA, dispatch, dynamic, stealth›

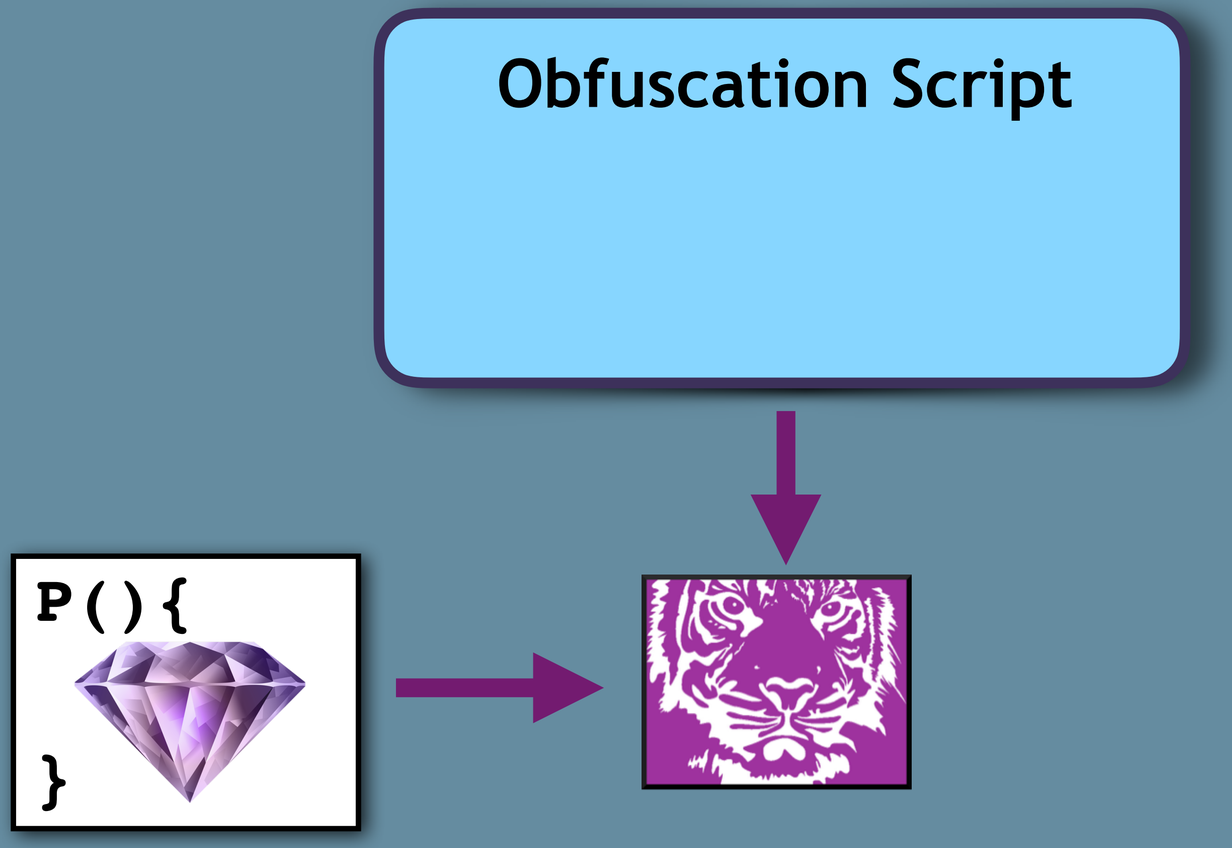


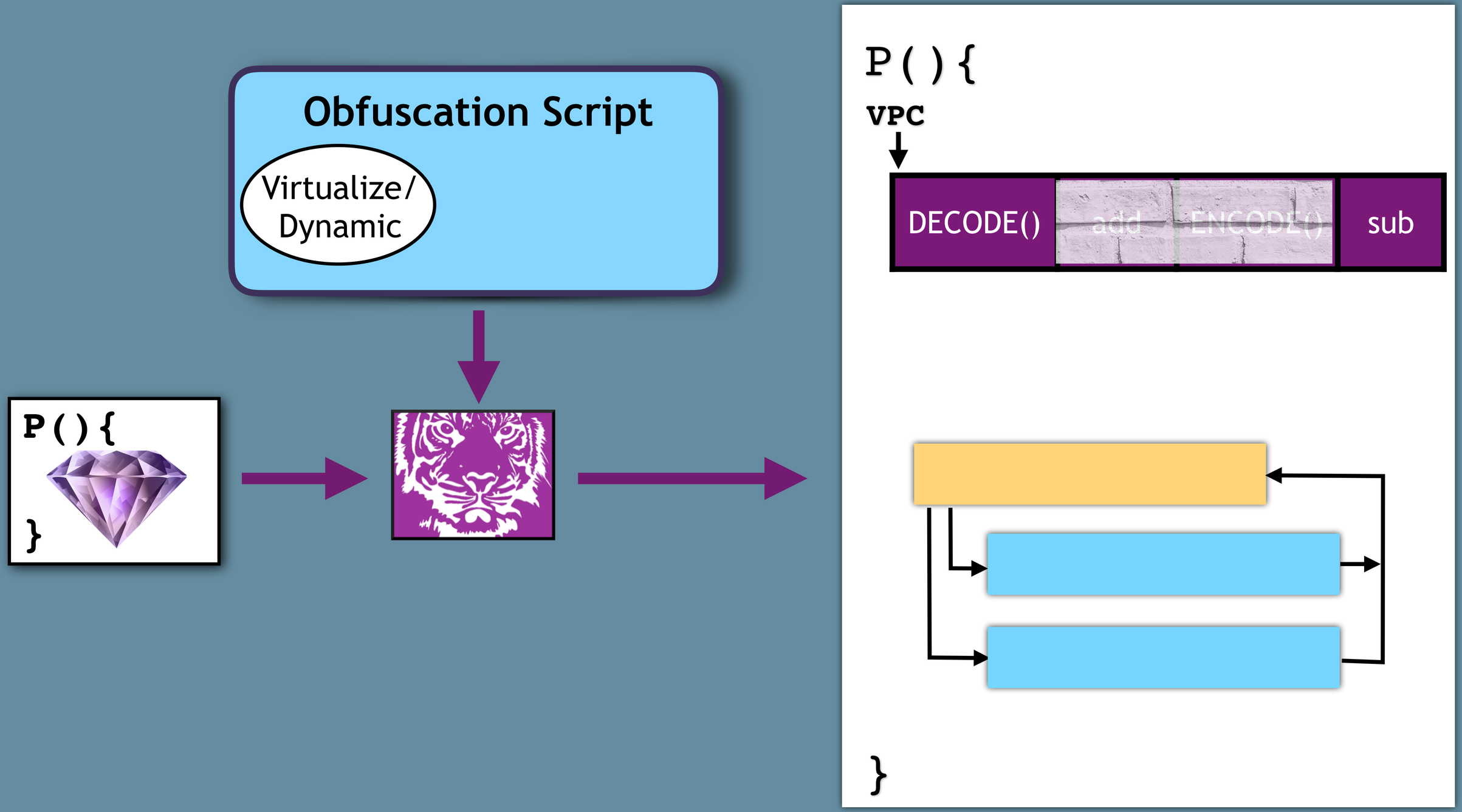
‹ISA, dispatch, dynamic, stealth›

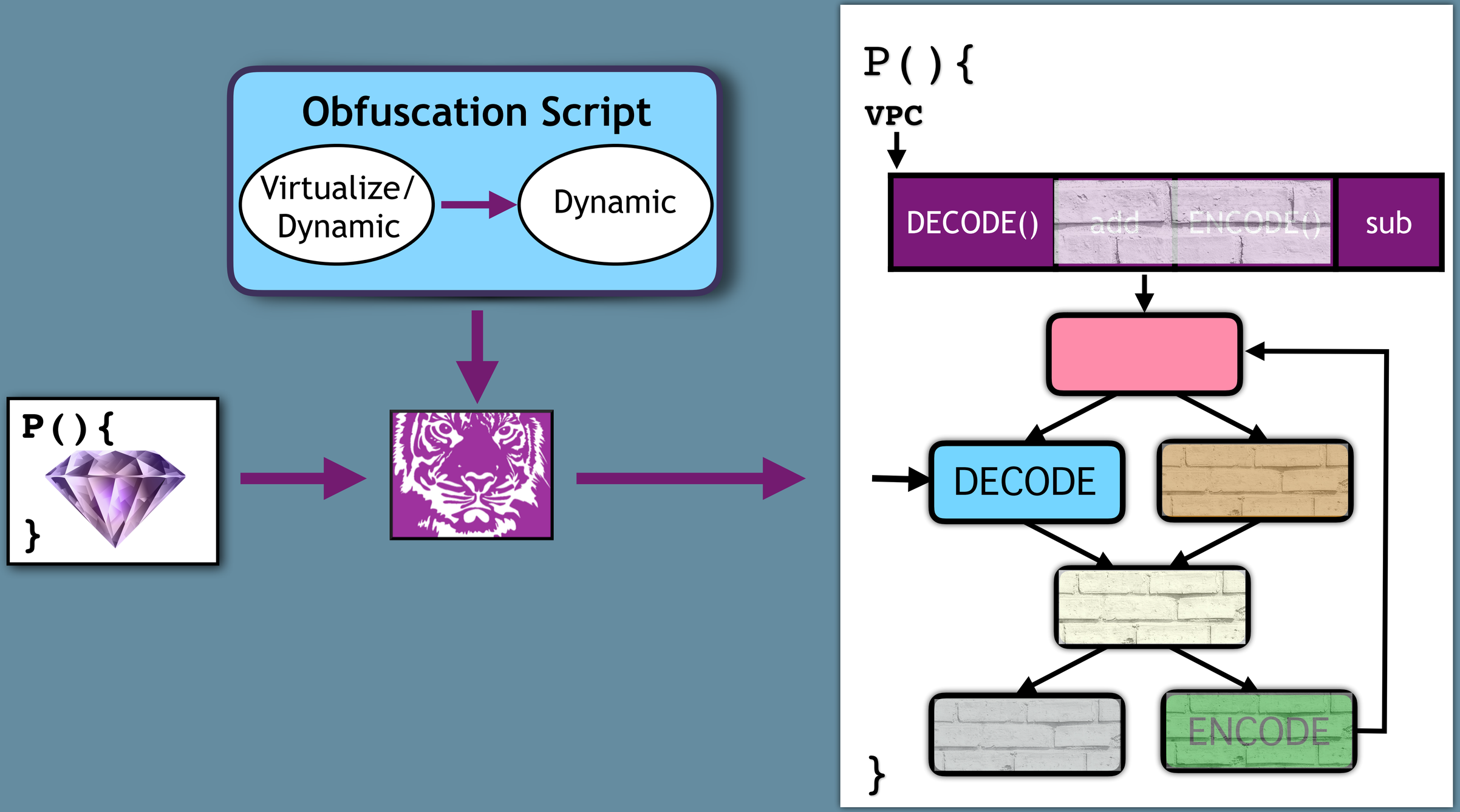


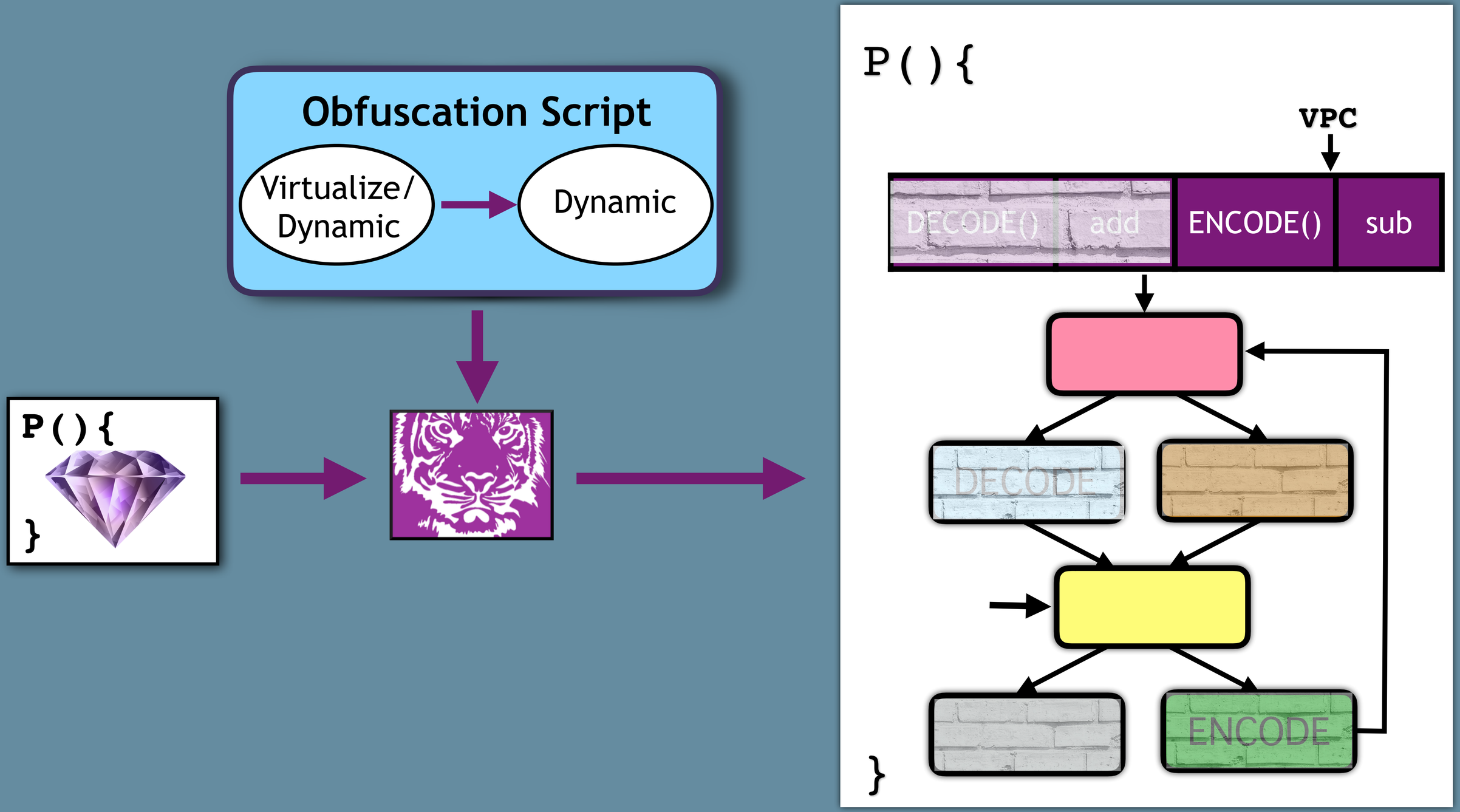
‹ISA, dispatch, dynamic, stealth›




Obfuscation Script
P(){
}

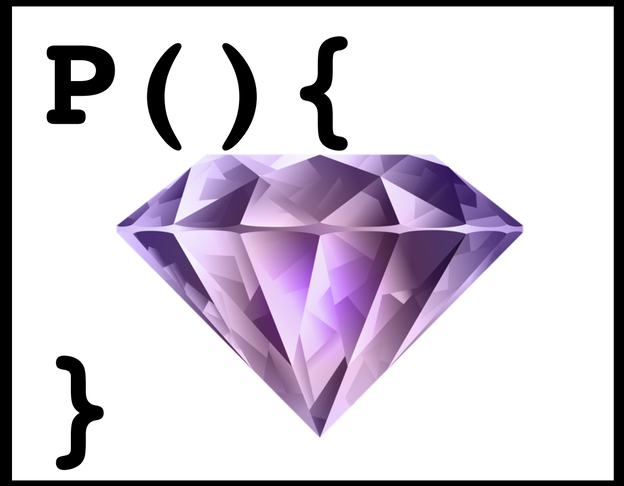

Obfuscation Script
Virtualize/
Dynamic
P ( ) {
}
P ( ) {
VPC
DECODE()
add
ENCODE()
sub
}


Obfuscation Script
Virtualize/ Dynamic
Dynamic
P ( ) {
}
P ( ) {
VPC
DECODE()
add
ENCODE()
sub
DECODE
ENCODE
}


Obfuscation Script
Virtualize/ Dynamic
Dynamic
P ( ) {
}
P ( ) {
VPC
DECODE()
add
ENCODE()
sub
DECODE
ENCODE
}

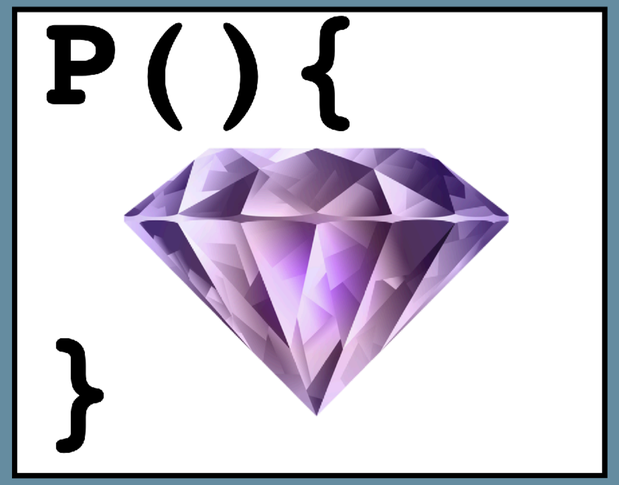
cirenec

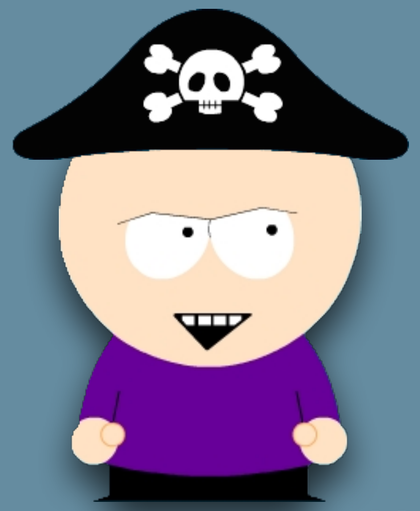
noitacsufbed



Generic Deobfuscation

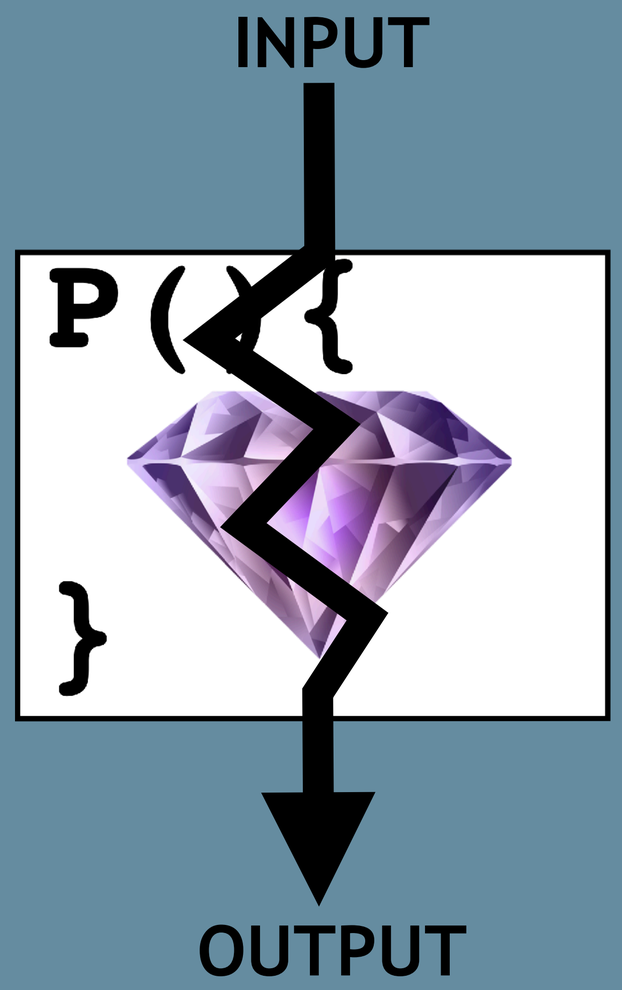
Dynamic Analysis

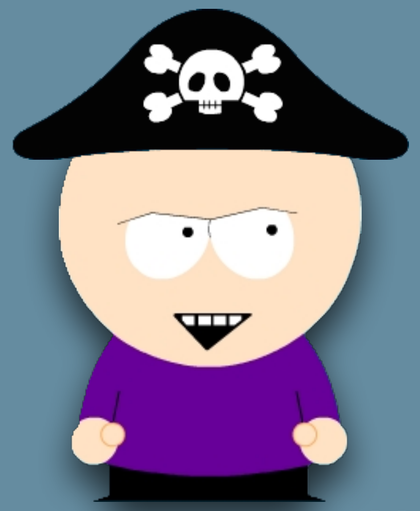




Yadegari, et al., A Generic Approach to Deobfuscation. IEEE S&P'15

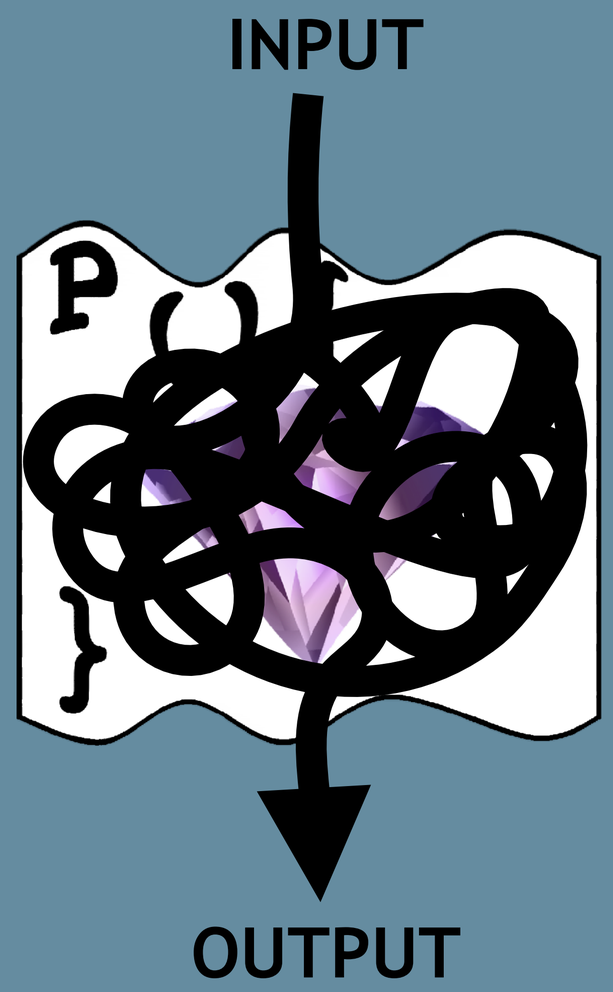
Dynamic Analysis

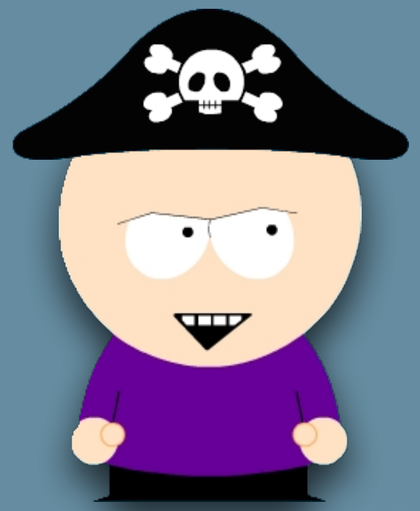




Yadegari, et al., A Generic Approach to Deobfuscation. IEEE S&P'15

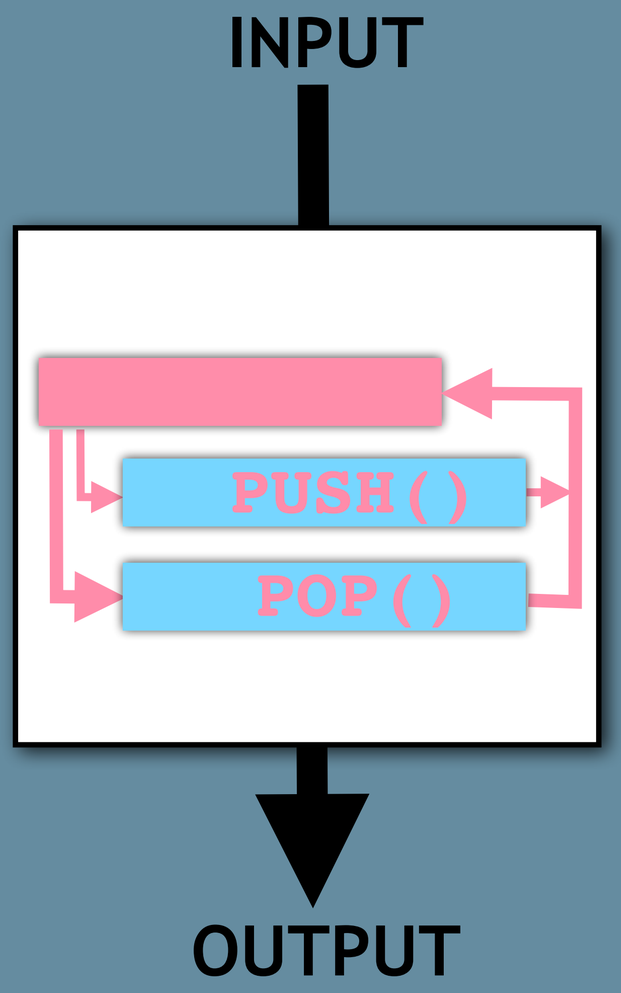
Dynamic Analysis





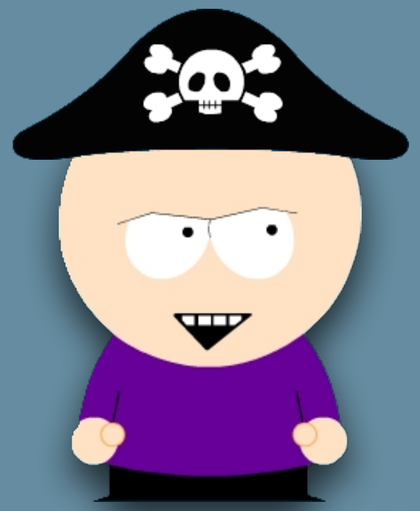
Yadegari, et al., A Generic Approach to Deobfuscation. IEEE S&P'15

Dynamic Analysis



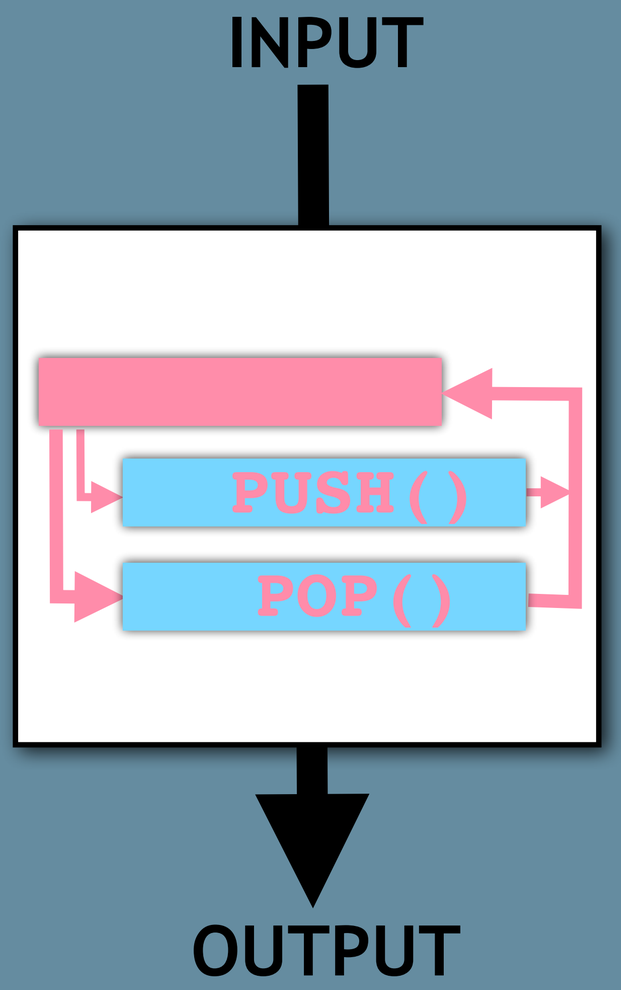
TRACE

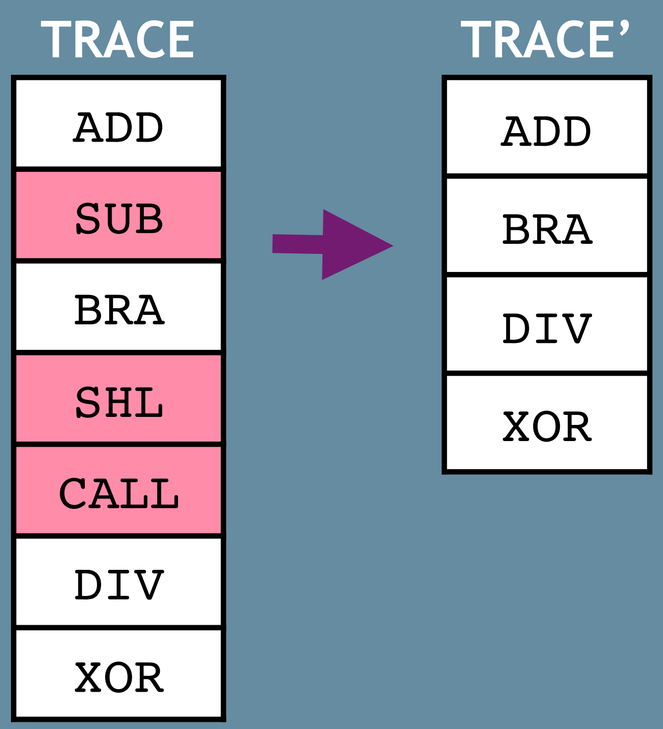
ADD
SUB
BRA
SHL
CALL
DIV
XOR

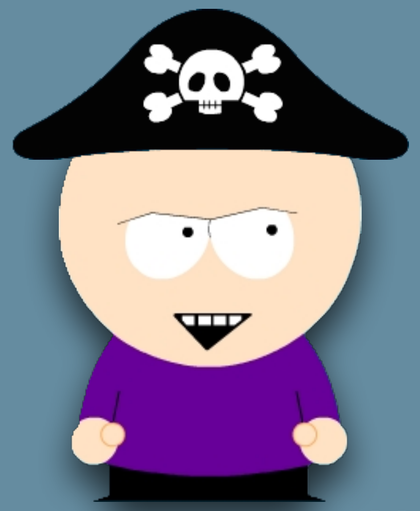


Yadegari, et al., A Generic Approach to Deobfuscation. IEEE S&P'15

Dynamic Analysis

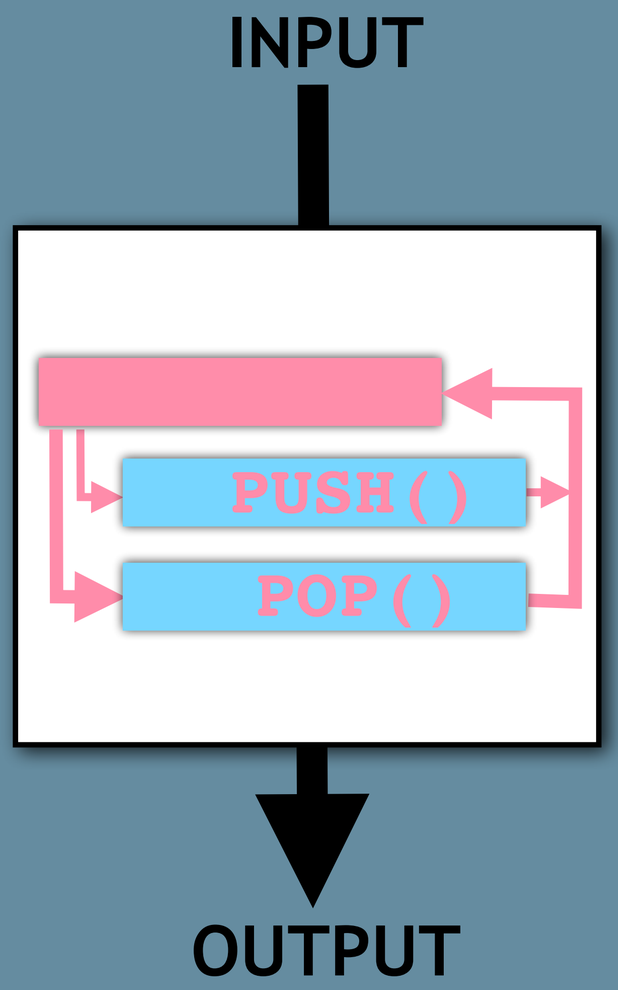


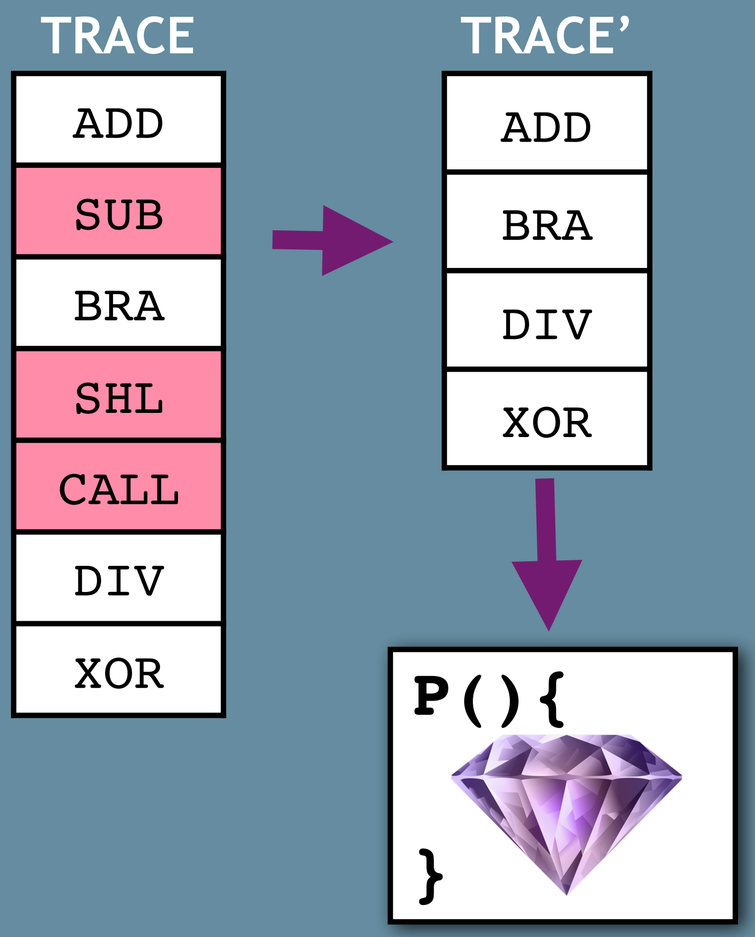


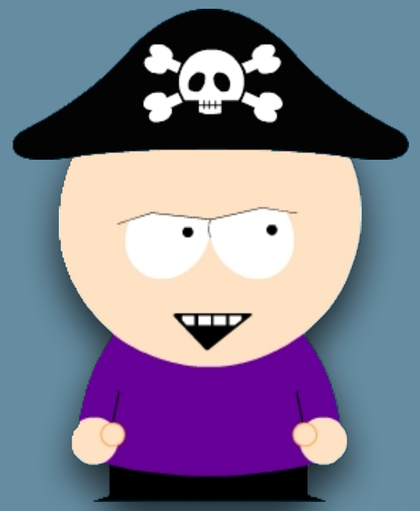


Yadegari, et al., A Generic Approach to Deobfuscation. IEEE S&P'15

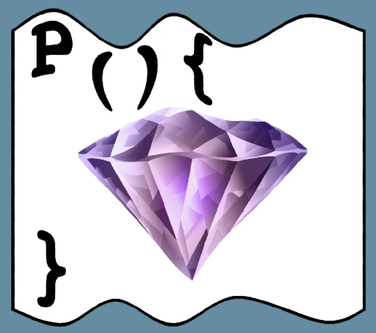
Dynamic Analysis



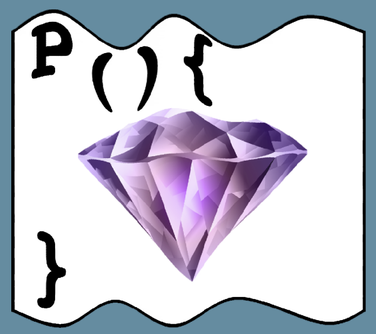




Yadegari, et al., A Generic Approach to Deobfuscation. IEEE S&P'15



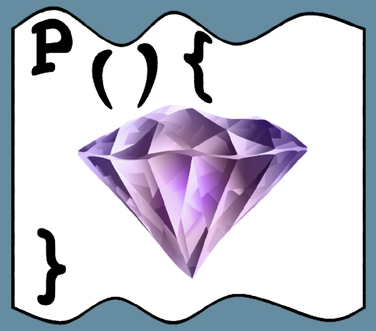
ADD
SUB
BRA
SHL
CALL
DIV
XOR



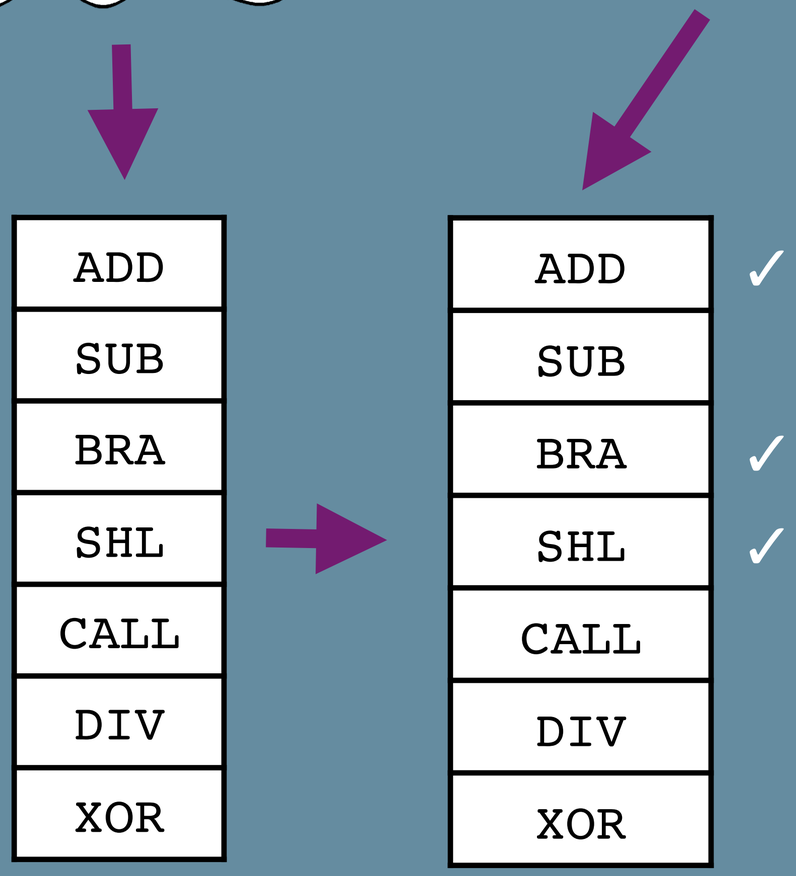
Forward Taint Analysis (depend on input)

ADD
SUB
BRA
SHL
CALL
DIV
XOR

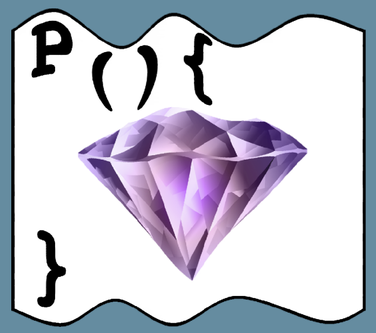
ADD	✓
SUB	
BRA	✓
SHL	✓
CALL	
DIV	
XOR	



Forward
Taint Analysis
(depend on input)




```
main (a) {
    b ← a;
    c ← b;
}
```

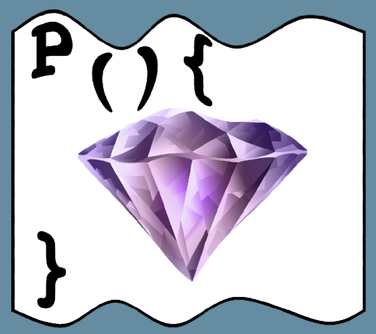


Forward
Taint Analysis
(depend on input)

ADD
SUB
BRA
SHL
CALL
DIV
XOR

ADD	✓
SUB	
BRA	✓
SHL	✓
CALL	
DIV	
XOR	

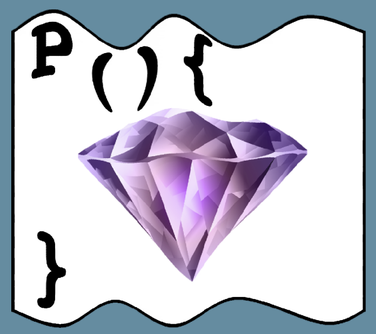
```
main (a) {
  b ← a;
  c ← b;
}
```



Forward Taint Analysis (depend on input)

Backward Taint Analysis (contribute to output)

ADD	ADD ✓	ADD ✓
SUB	SUB	SUB
BRA	BRA ✓	BRA ✓
SHL	SHL ✓	SHL ✓
CALL	CALL	CALL
DIV	DIV	DIV ✓
XOR	XOR	XOR ✓

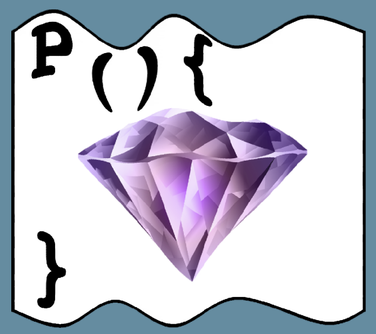


Forward
Taint Analysis
(depend on input)

ADD
SUB
BRA
SHL
CALL
DIV
XOR

ADD	✓
SUB	
BRA	✓
SHL	✓
CALL	
DIV	
XOR	

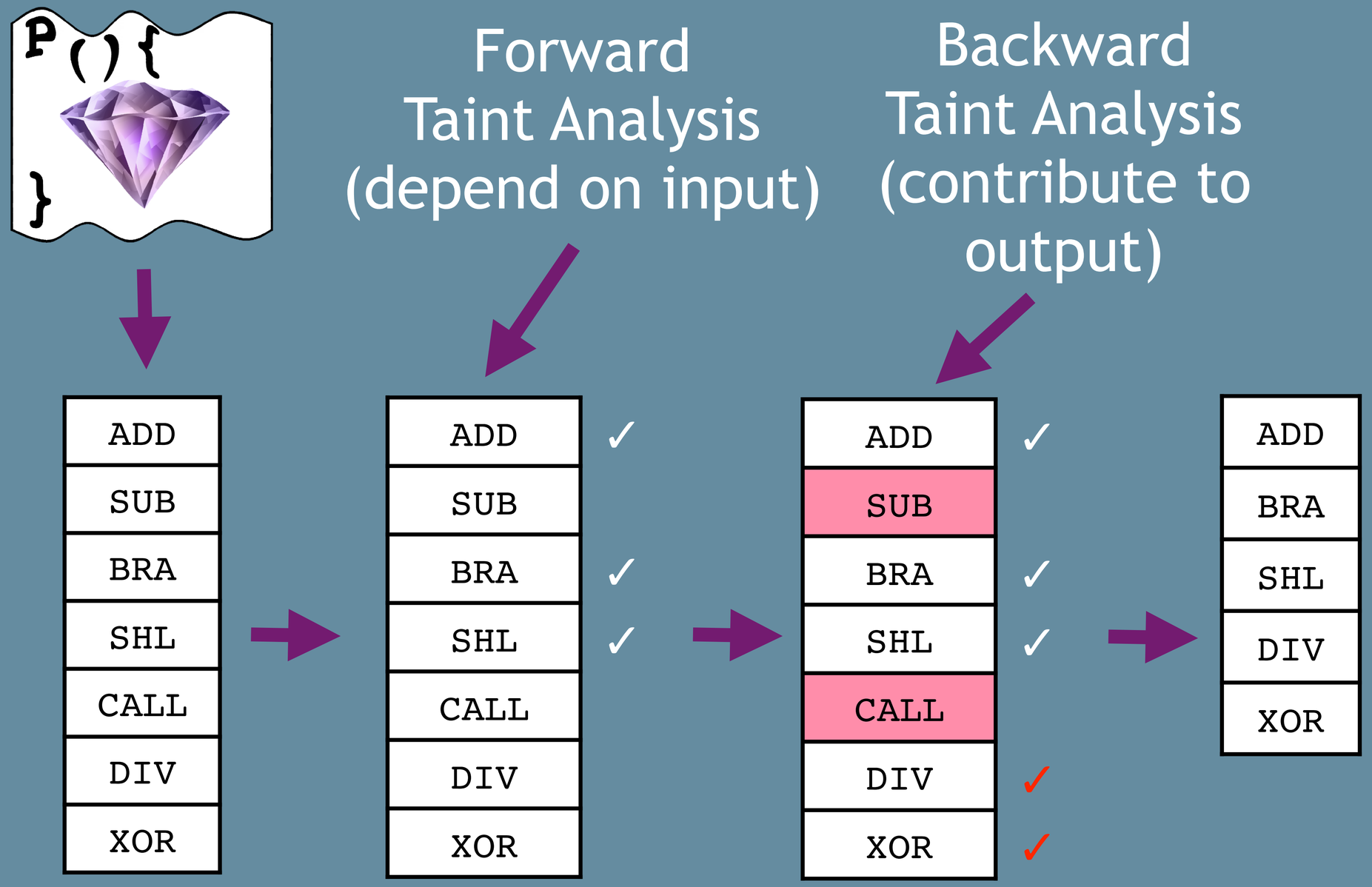
```
main () {
  b ← a;
  printf(b);
}
```

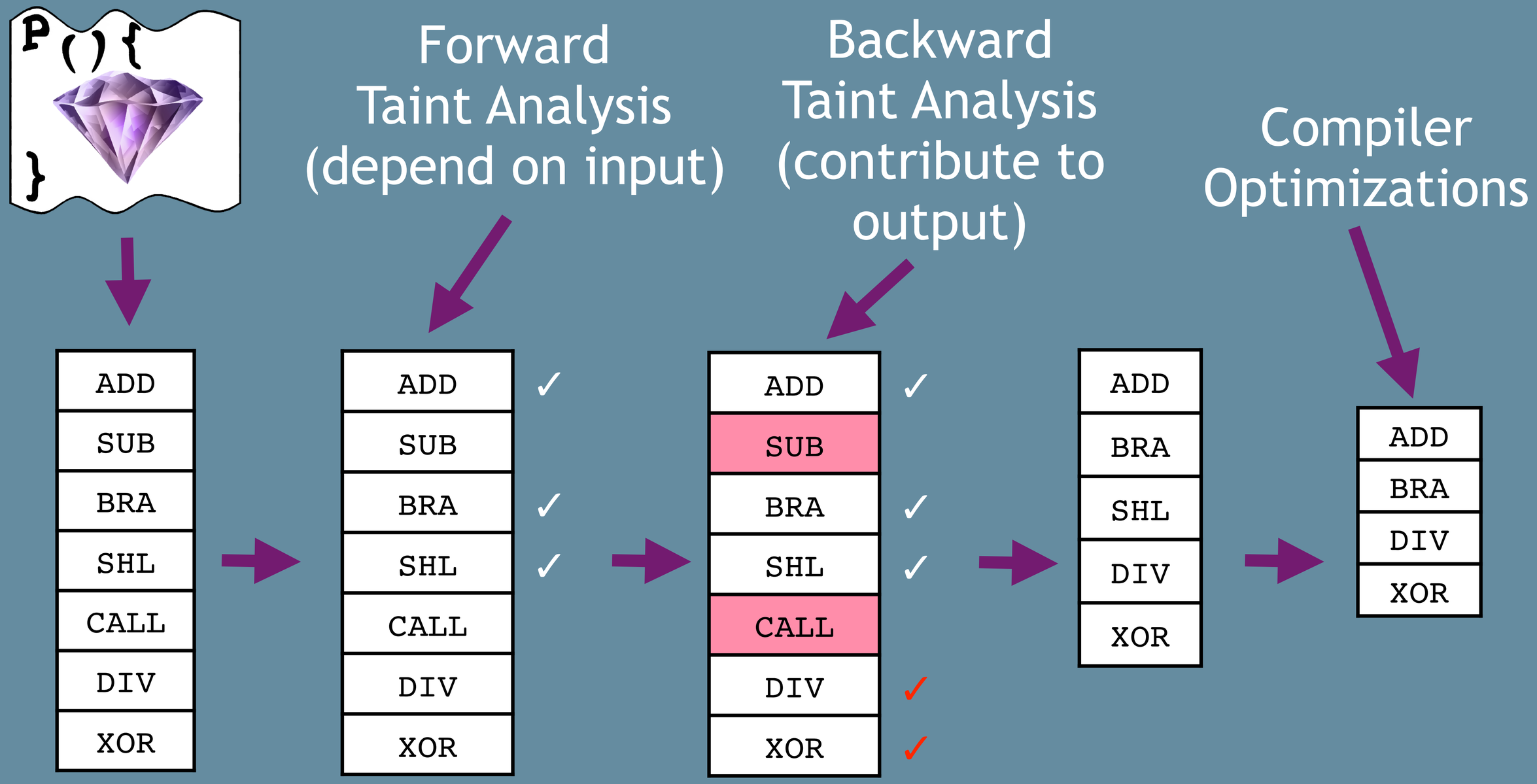


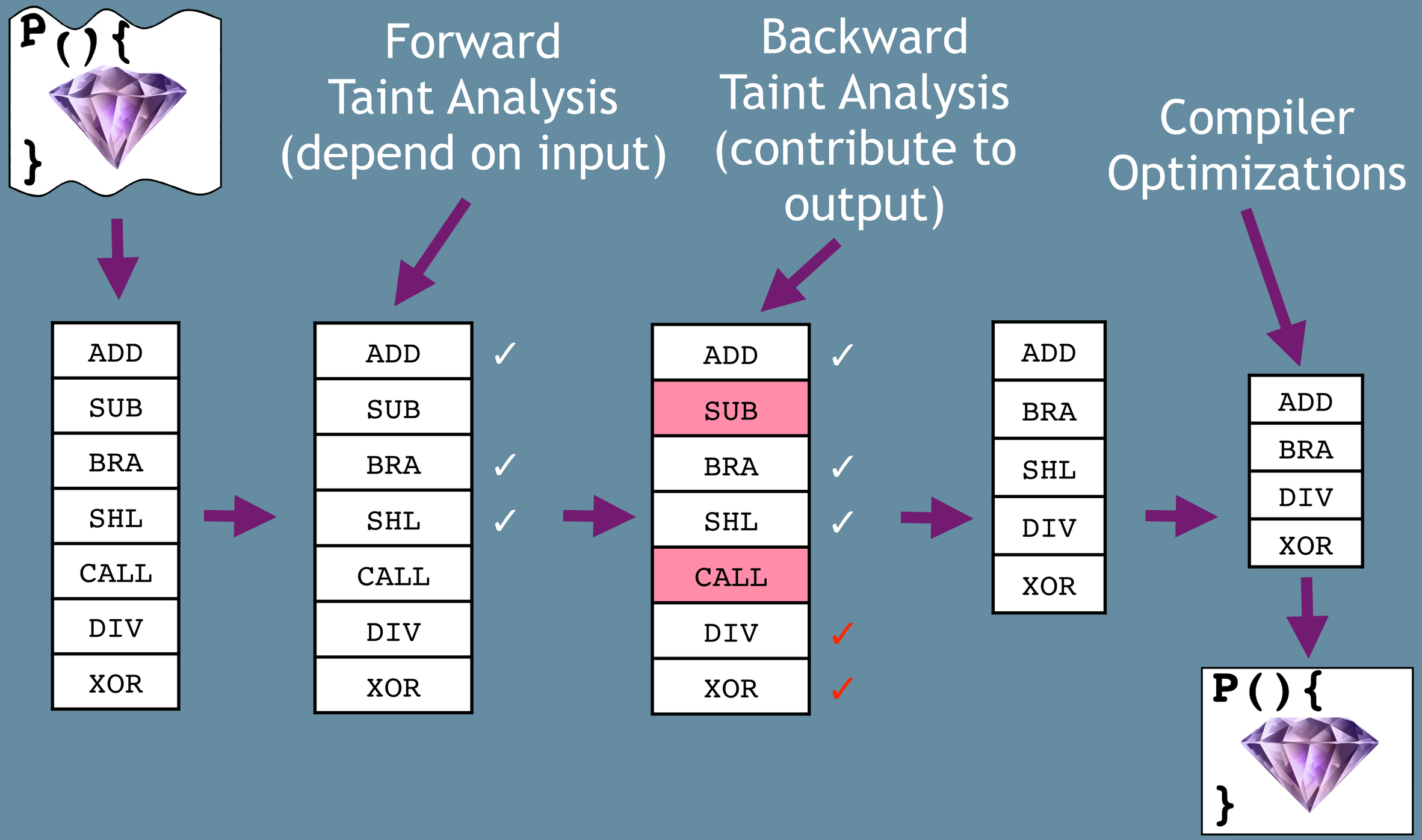
Forward Taint Analysis (depend on input)

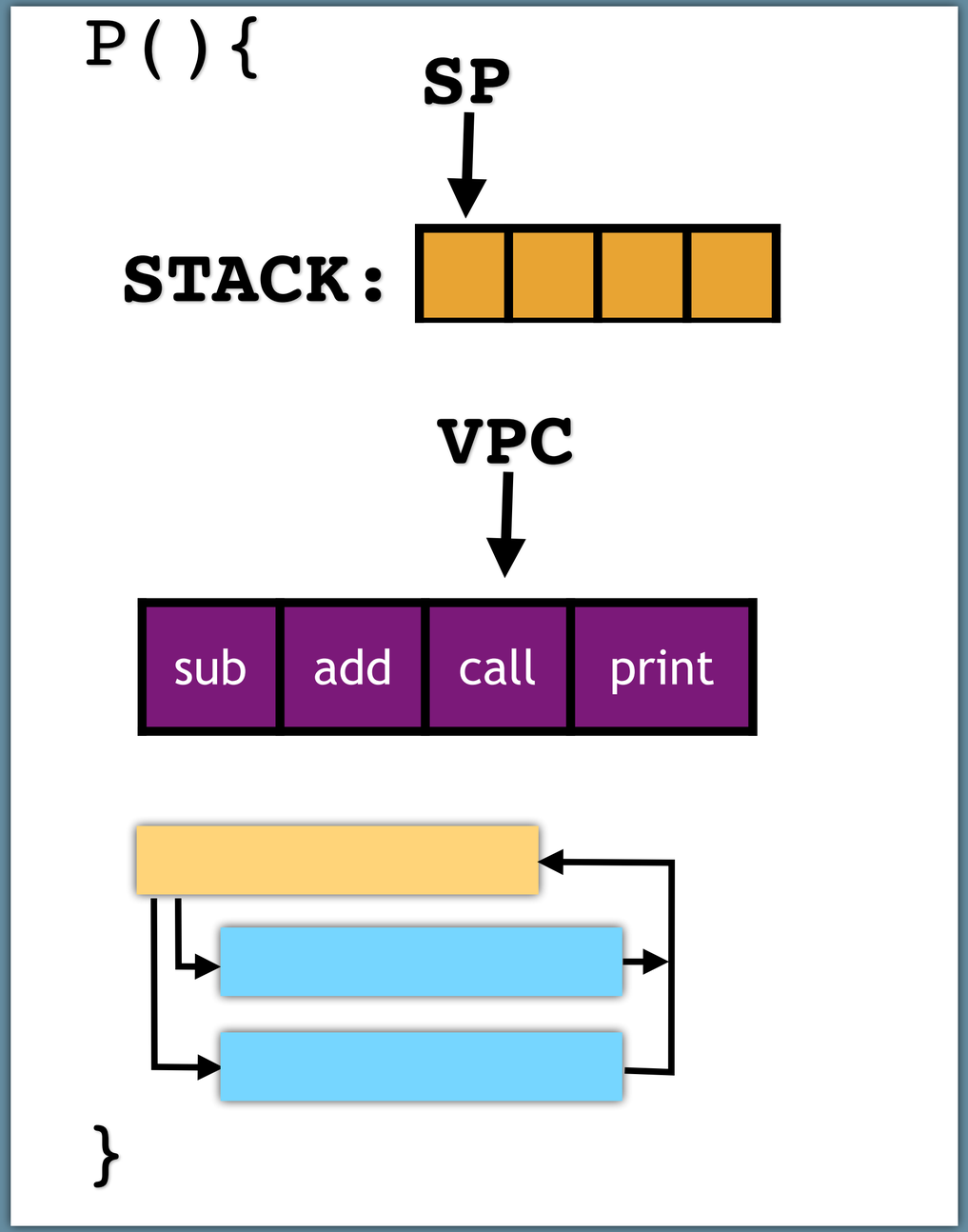
Backward Taint Analysis (contribute to output)

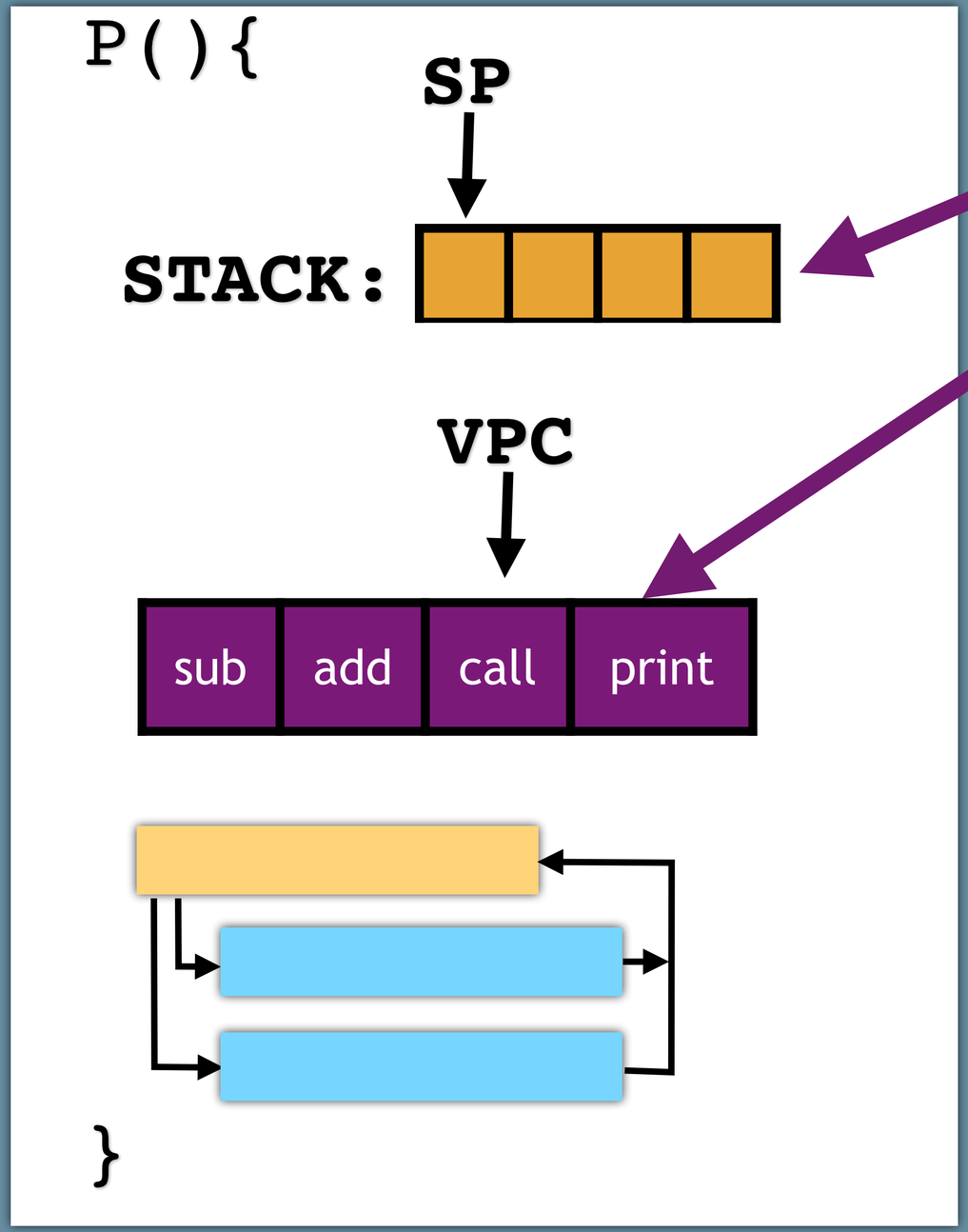
ADD	ADD ✓	ADD ✓
SUB	SUB	SUB
BRA	BRA ✓	BRA ✓
SHL	SHL ✓	SHL ✓
CALL	CALL	CALL
DIV	DIV	DIV ✓
XOR	XOR	XOR ✓


P ( ) {
}
Forward
Taint Analysis
(depend on input)
Backward
Taint Analysis
(contribute to
output)
ADD
SUB
BRA
SHL
CALL
DIV
XOR
ADD ✓
SUB
BRA ✓
SHL ✓
CALL
DIV
XOR
ADD ✓
SUB
BRA ✓
SHL ✓
CALL
DIV ✓
XOR ✓
ADD
BRA
SHL
DIV
XOR

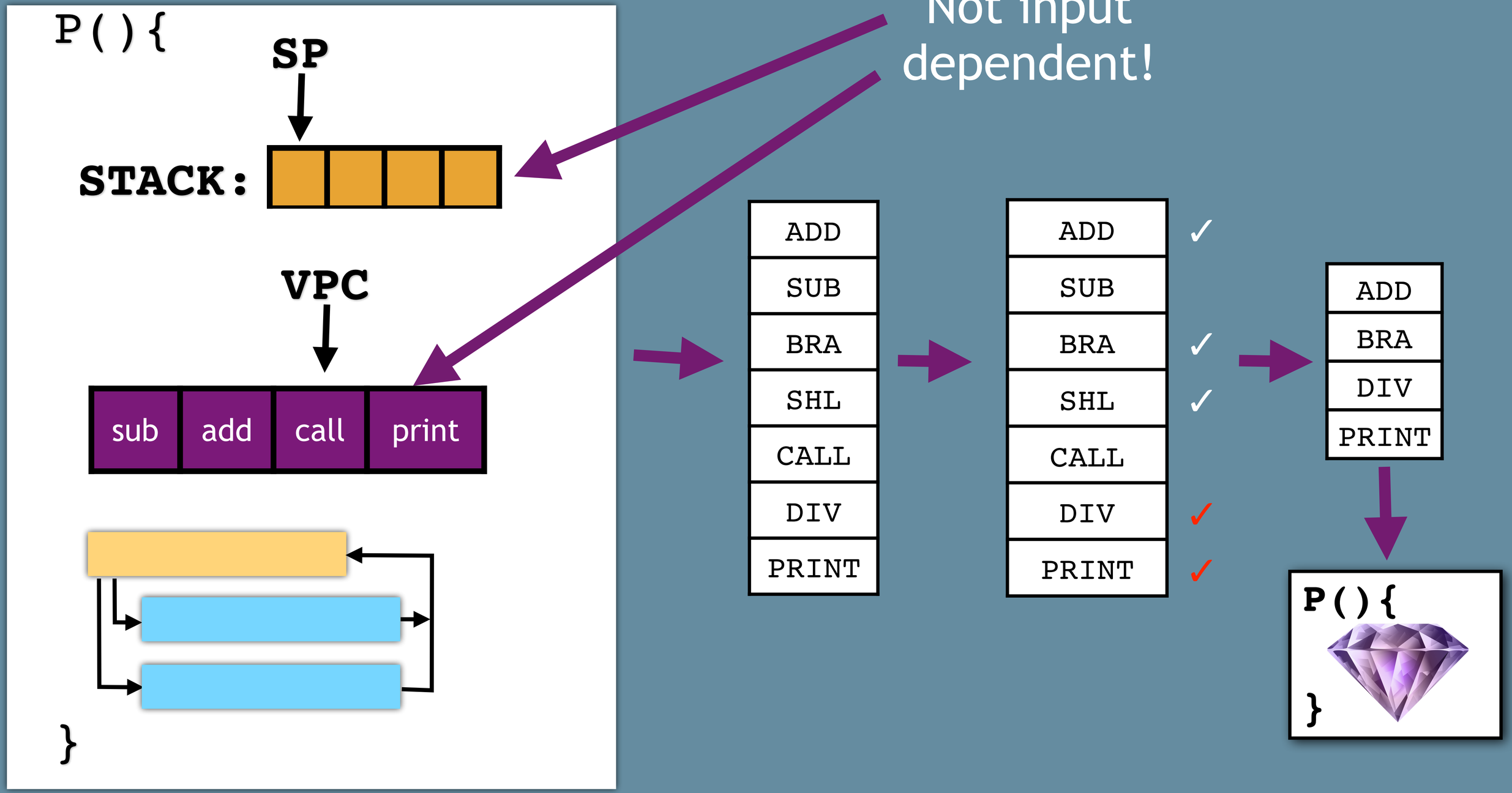

P ( ) {
}
Forward
Taint Analysis
(depend on input)
Backward
Taint Analysis
(contribute to
output)
Compiler
Optimizations
ADD
SUB
BRA
SHL
CALL
DIV
XOR
ADD ✓
SUB
BRA ✓
SHL ✓
CALL
DIV
XOR
ADD ✓
SUB
BRA ✓
SHL ✓
CALL
DIV ✓
XOR ✓
ADD
BRA
SHL
DIV
XOR
ADD
BRA
DIV
XOR

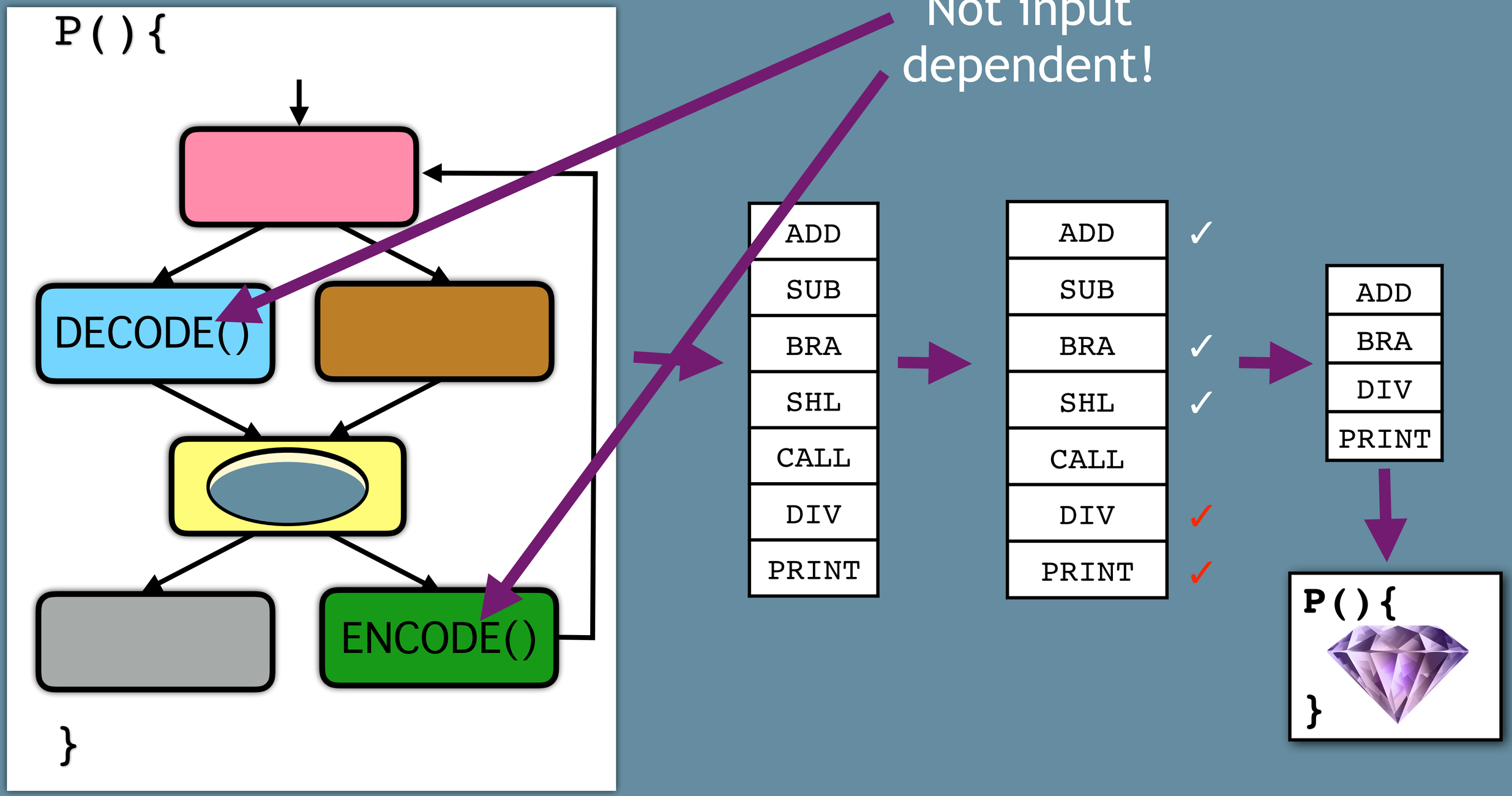

P ( ) {
}
Forward
Taint Analysis
(depend on input)
Backward
Taint Analysis
(contribute to
output)
Compiler
Optimizations
ADD
SUB
BRA
SHL
CALL
DIV
XOR
ADD ✓
SUB
BRA ✓
SHL ✓
CALL
DIV
XOR
ADD ✓
SUB
BRA ✓
SHL ✓
CALL
DIV ✓
XOR ✓
ADD
BRA
SHL
DIV
XOR
ADD
BRA
DIV
XOR
P ( ) {
}


P(){
SP
STACK:
VPC
sub
add
call
print
}



Not input dependent!

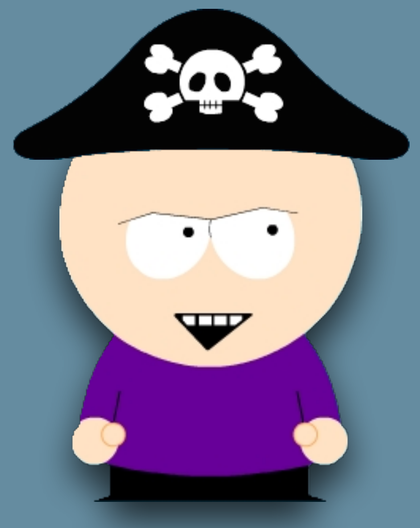

P(){
SP
STACK:
VPC
sub
add
call
print
}
Not input dependent!
ADD
SUB
BRA
SHL
CALL
DIV
PRINT
ADD
SUB
BRA
SHL
CALL
DIV
PRINT
ADD
BRA
DIV
PRINT
P(){
}


P(){
DECODE()
ENCODE()
}
Not input dependent!
ADD
SUB
BRA
SHL
CALL
DIV
PRINT
ADD
SUB
BRA
SHL
CALL
DIV
PRINT
ADD
BRA
DIV
PRINT
P(){
}

Anti-Dynamic Analysis

```
main(){


}
```

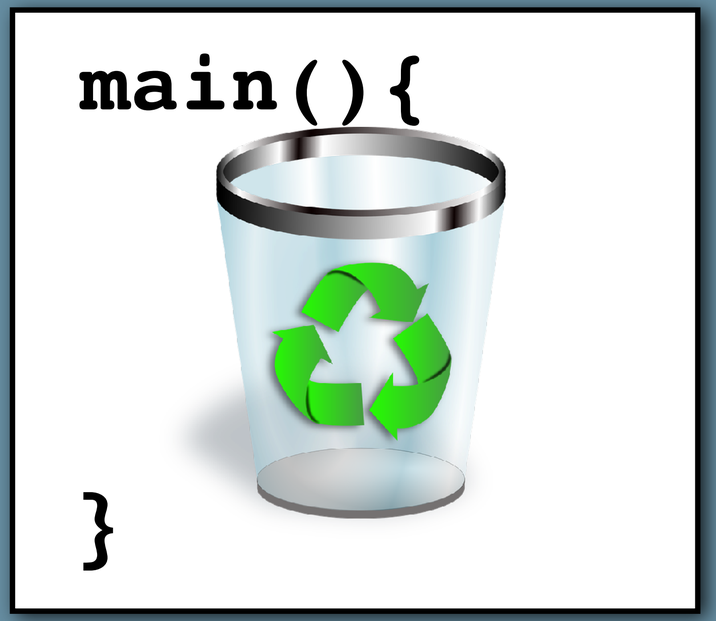
TEST
LEA
JREG
XOR
JMP

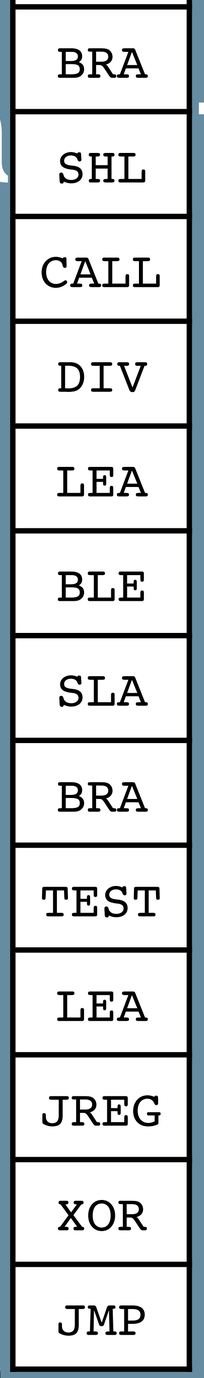


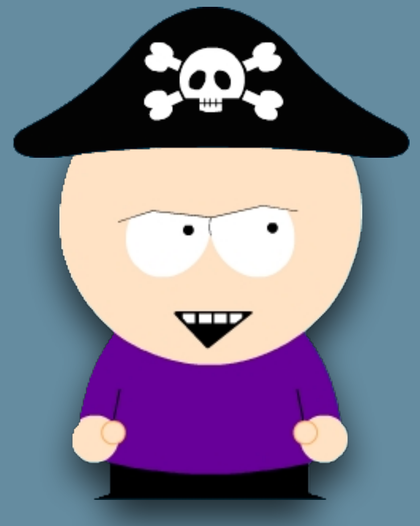
Pawlovski, et al., Probfuscation: An Obfuscation Approach …

Anti-Dynamic Analysis

- Artificially inflate trace size



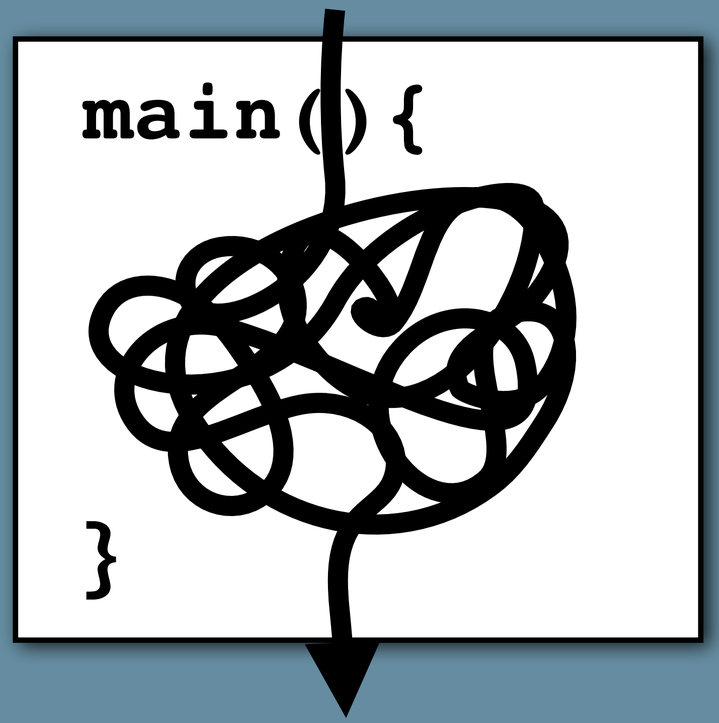




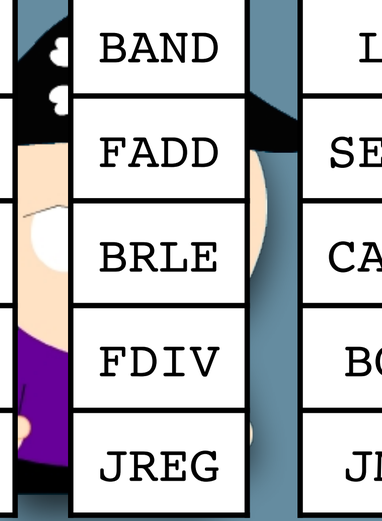
Pawlovski, et al., Probfuscation: An Obfuscation Approach …

Anti-Dynamic Analysis

- Artificially inflate trace size
- Force the collection of multiple traces



BRA	FDIV	FADD	BR
SHL	CALL	CALL	SI
CALL	LEA	JMP	FA
DIV	MUL	TEST	L
LEA	JMP	LEA	ST
BLE	BLE	BLE	FS
SLA	SRL	ADD	CA
BRA	CALL	SUB	BI
TEST	BOR	BAND	L
LEA	STO	FADD	SE
JREG	BRA	BRLE	CA
XOR	AND	FDIV	BO
JMP	CALL	JREG	JI

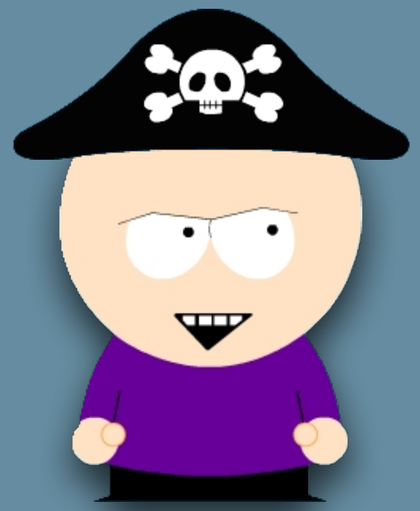


Pawlovski, et al., Probfuscation: An Obfuscation Approach …

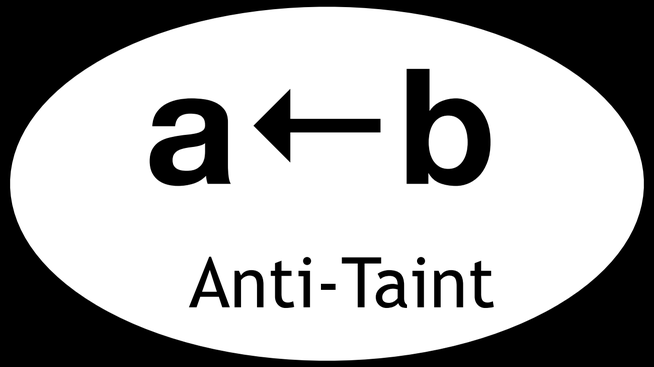
Anti-Dynamic Analysis

- Artificially inflate trace size
- Force the collection of multiple traces
- Prevent traces from being collected

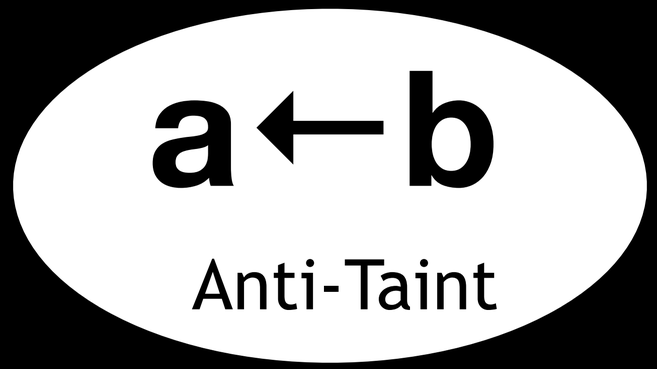
```
main(){

 if (traced)
     abort();


}
```



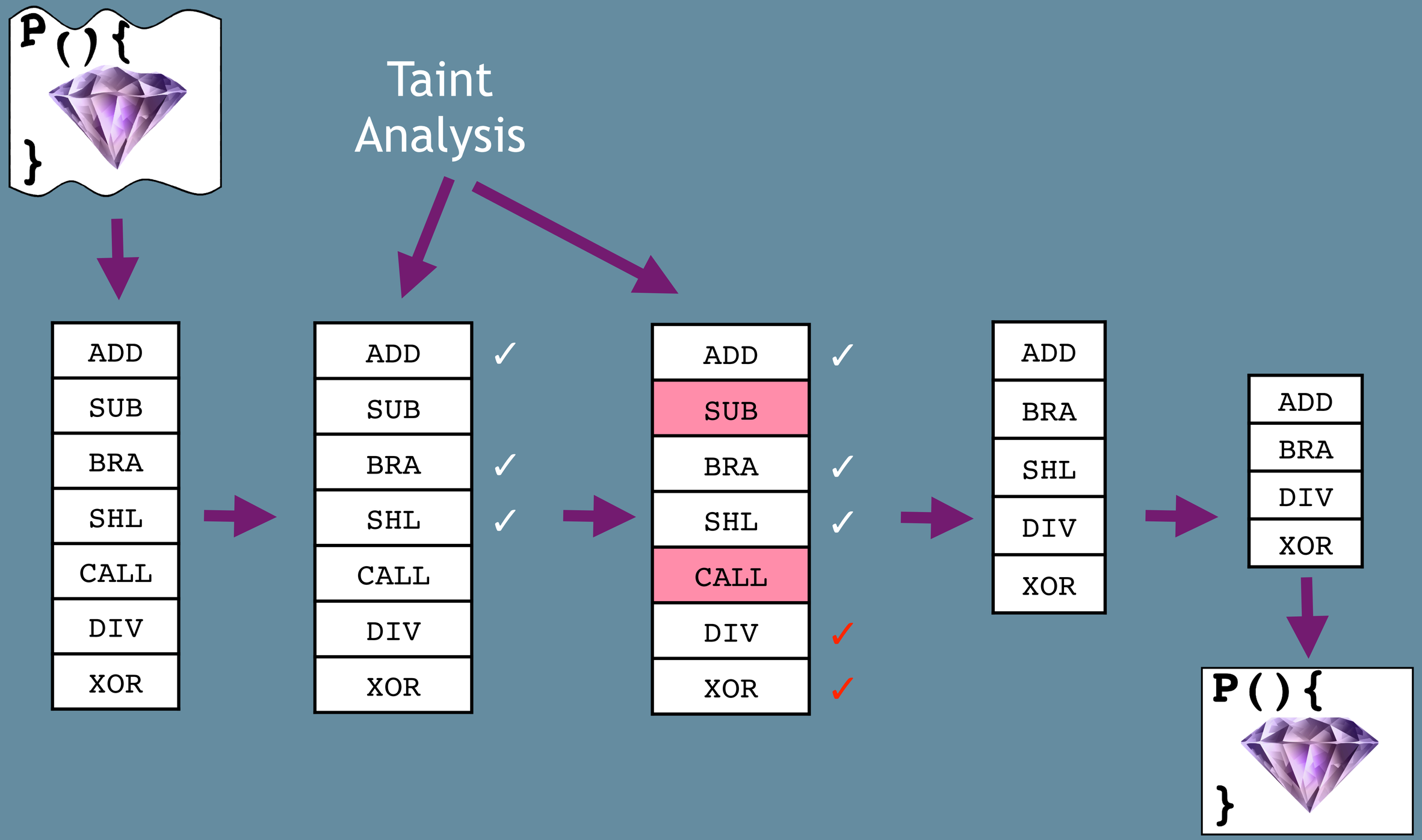
Pawlovski, et al., Probfuscation: An Obfuscation Approach …

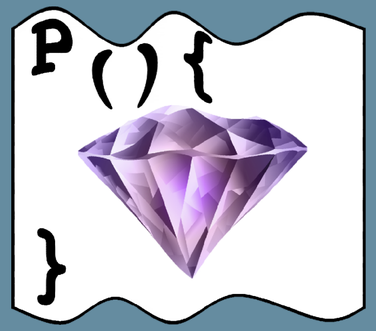


Anti Taint-Analysis



Anti Taint-Analysis


P ( ) {
}
Taint Analysis
ADD
SUB
BRA
SHL
CALL
DIV
XOR
ADD ✓
SUB
BRA ✓
SHL ✓
CALL
DIV
XOR
ADD ✓
SUB
BRA ✓
SHL ✓
CALL
DIV ✓
XOR ✓
ADD
BRA
SHL
DIV
XOR
ADD
BRA
DIV
XOR
P ( ) {
}

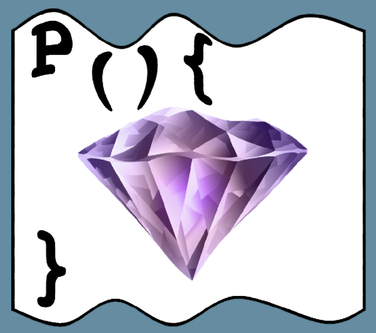


Taint Analysis

ADD	
SUB	
BRA	
SHL	
CALL	
DIV	
XOR	

ADD	✓
SUB	
BRA	✓
SHL	✓
CALL	
DIV	
XOR	

```
main () {
  bit a,b;

  a ← b;

}
```

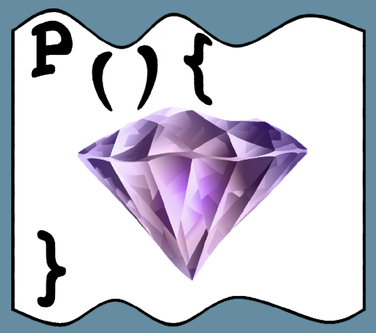


Taint Analysis

ADD	
SUB	
BRA	
SHL	
CALL	
DIV	
XOR	

ADD	✓
SUB	
BRA	✓
SHL	✓
CALL	
DIV	
XOR	

```
main () {
  bit a,b;

  a ← b;

}
```



Taint Analysis

ADD	
SUB	
BRA	
SHL	
CALL	
DIV	
XOR	

ADD	✓
SUB	
BRA	✓
SHL	✓
CALL	
DIV	
XOR	

```
main () {
  bit a,b;

  a ? b;

}
```

Implicit Flow

```
main(){
   int a ← b
}
```

Implicit Flow

```
main(){
  int a ← b
}
```

```
main(){
  int a ← 0

}
```

Implicit Flow

```
main(){
  int a←b
}
```

```
main(){
  int a ← 0
  for(i←0; i<b; i++)
    a++
}
```

Implicit Flow

```
main(){
  int a ← b
}
```

```
main(){
  int a ← 0
  for(i ← 0; i<b; i++)
    a++
}
```

😄 no data dependence!

Implicit Flow

```
main(){
  int a←b
}
```

```
main(){
  int a ← 0
  for(i←0; i<b; i++)
    a++
}
```

but control dependence

```
main(){
   bit a←b
}
```

```
main(){
   bit a←b
}
```

```
bit value;
void handler(...) {
   value ← 1
}

main(){
  signal(31,handler)



}
```

```
main(){
   bit a←b
}
```

```
bit value;
void handler(...) {
   value ← 1
}

main(){
  signal(31,handler)
  value ← 0
  if (b == 1)
      raise(31)
   a ← value
}
```

```
main(){
   bit a←b
}
```

```
bit value;
void handler(...) {
   value ← 1
}

main(){
  signal(31,handler)
  value ← 0
  if (b == 1)
      raise(31)
   a ← value
}
```

```
main(){
   bit a←b
}
```

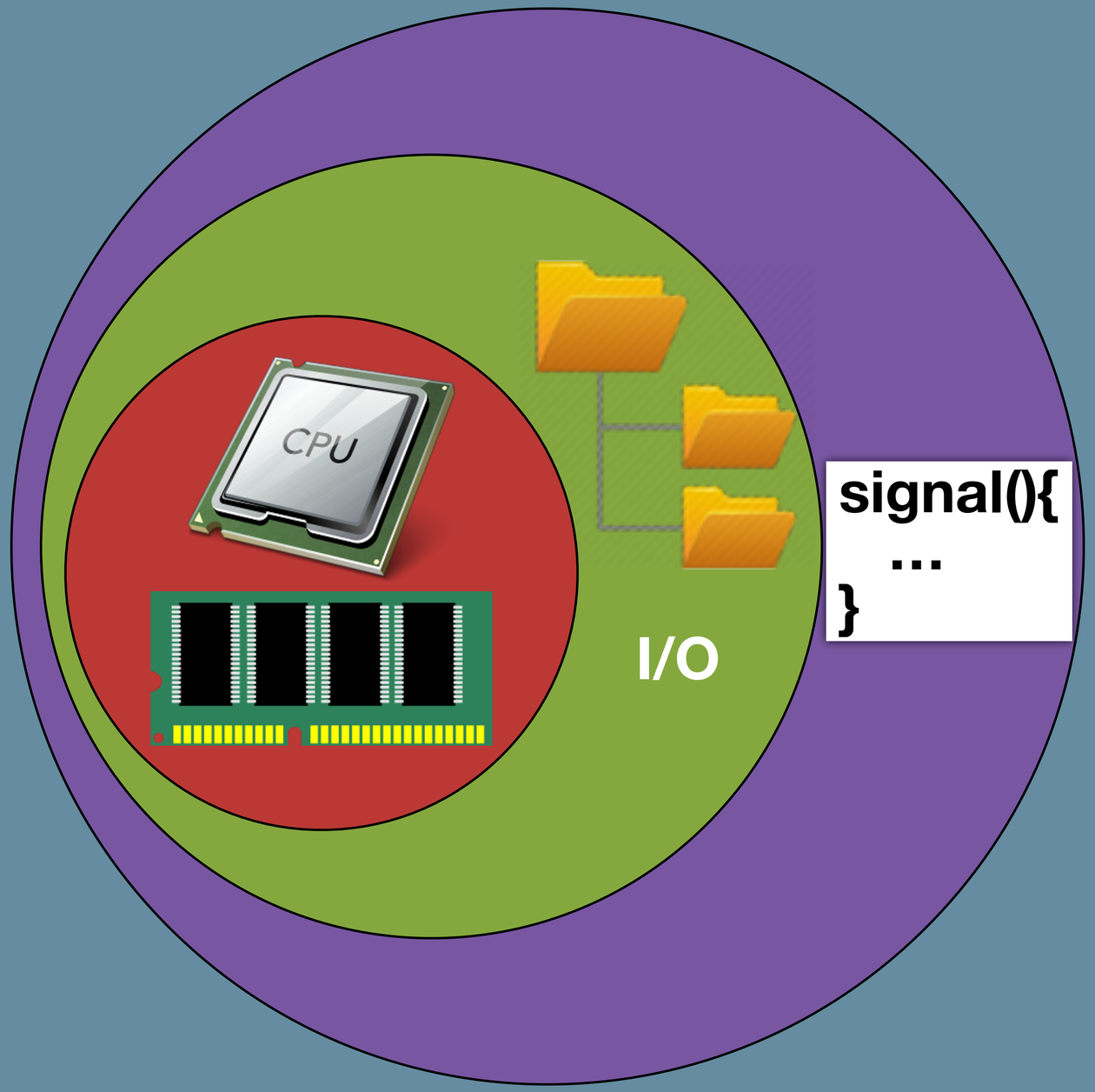
```
bit value;
void handler(...) {
   value ← 1
}

main(){
  signal(31,handler)
  value ← 0
  if (b == 1)
     raise(31)
   a ← value
}
```

no data 😄 dependence!

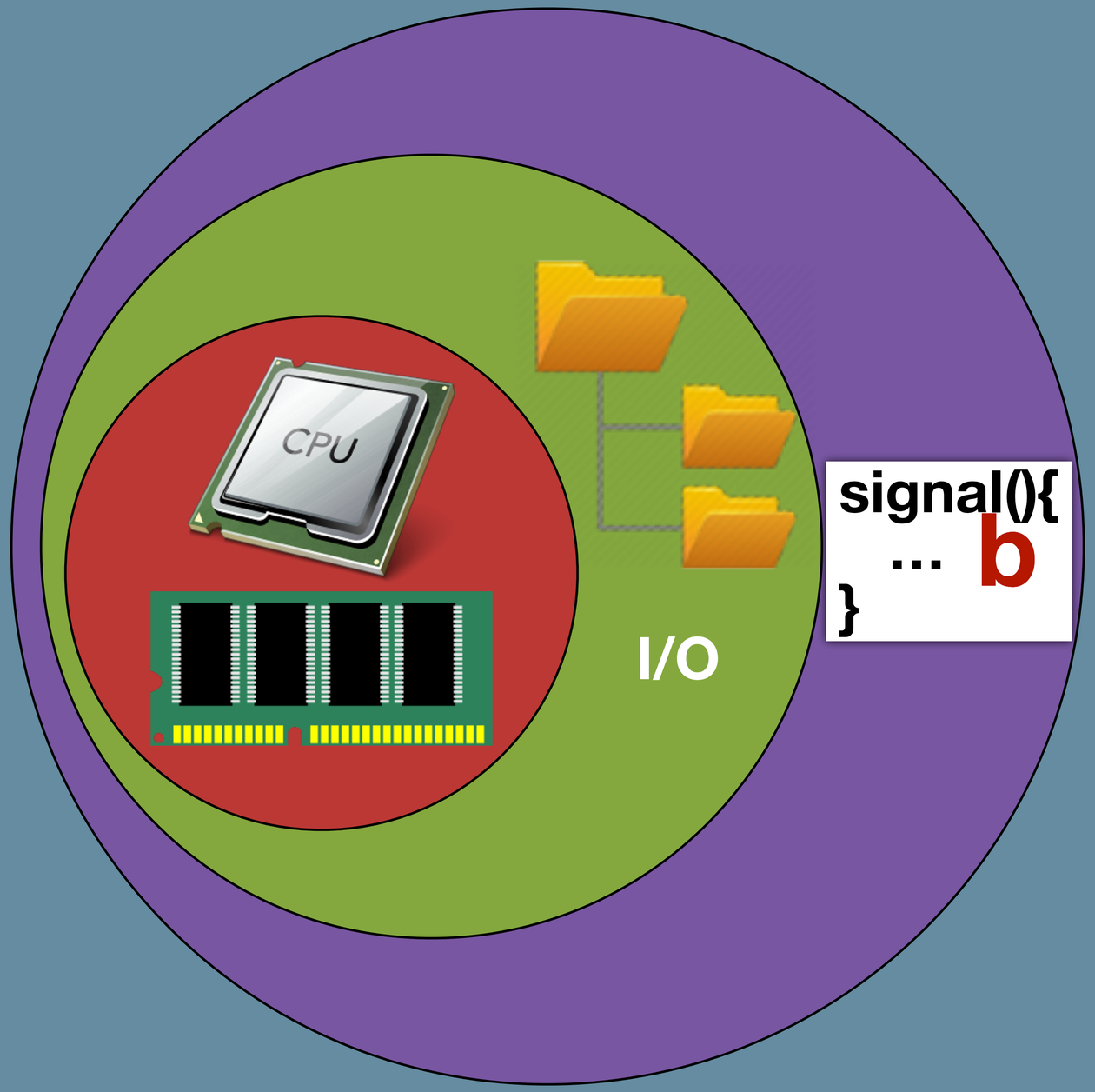
```
main(){
   bit a←b
}
```

```
bit value;
void handler(...) {
   value ← 1
}

main(){
  signal(31,handler)
   value ← 0
   if (b == 1)
      raise(31)
   a ← value
}
```

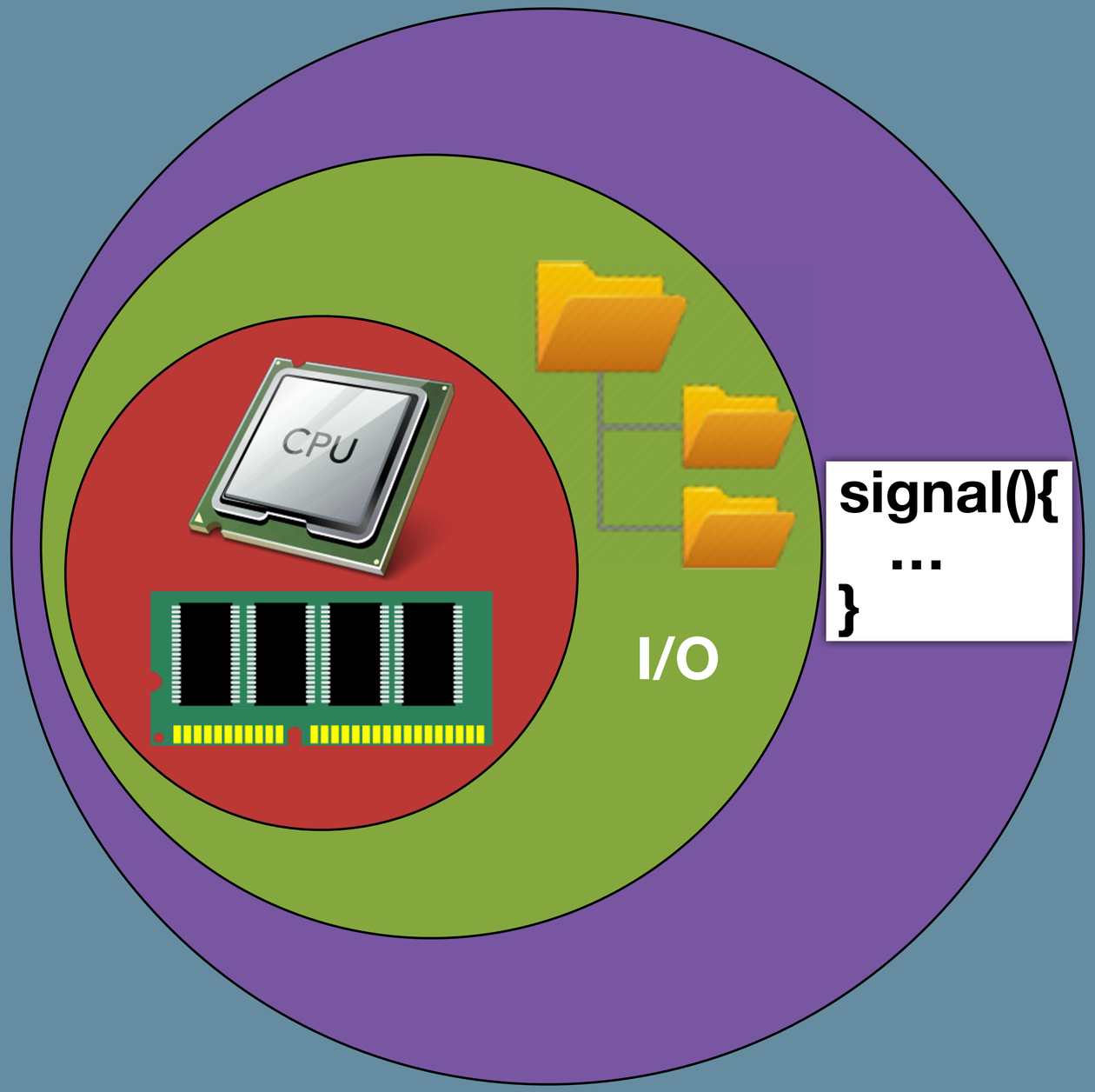
☹️ still control dependence



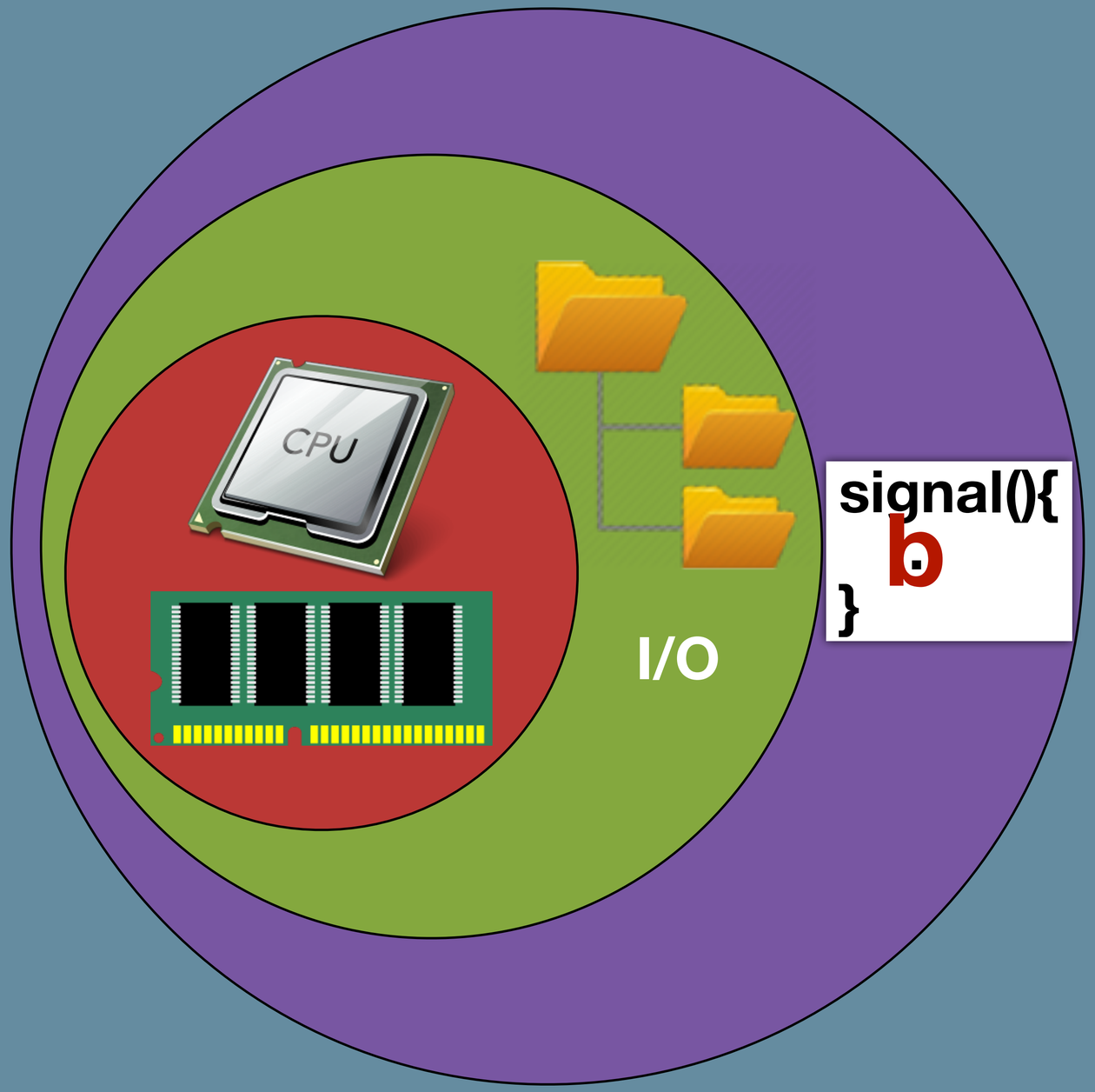

```
bit value;
void handler(...) {
   value ← 1
}

main(){
   value ← 0
   signal(31,handler)
   if (b == 1)
      raise(31)
   a ← value
}
```



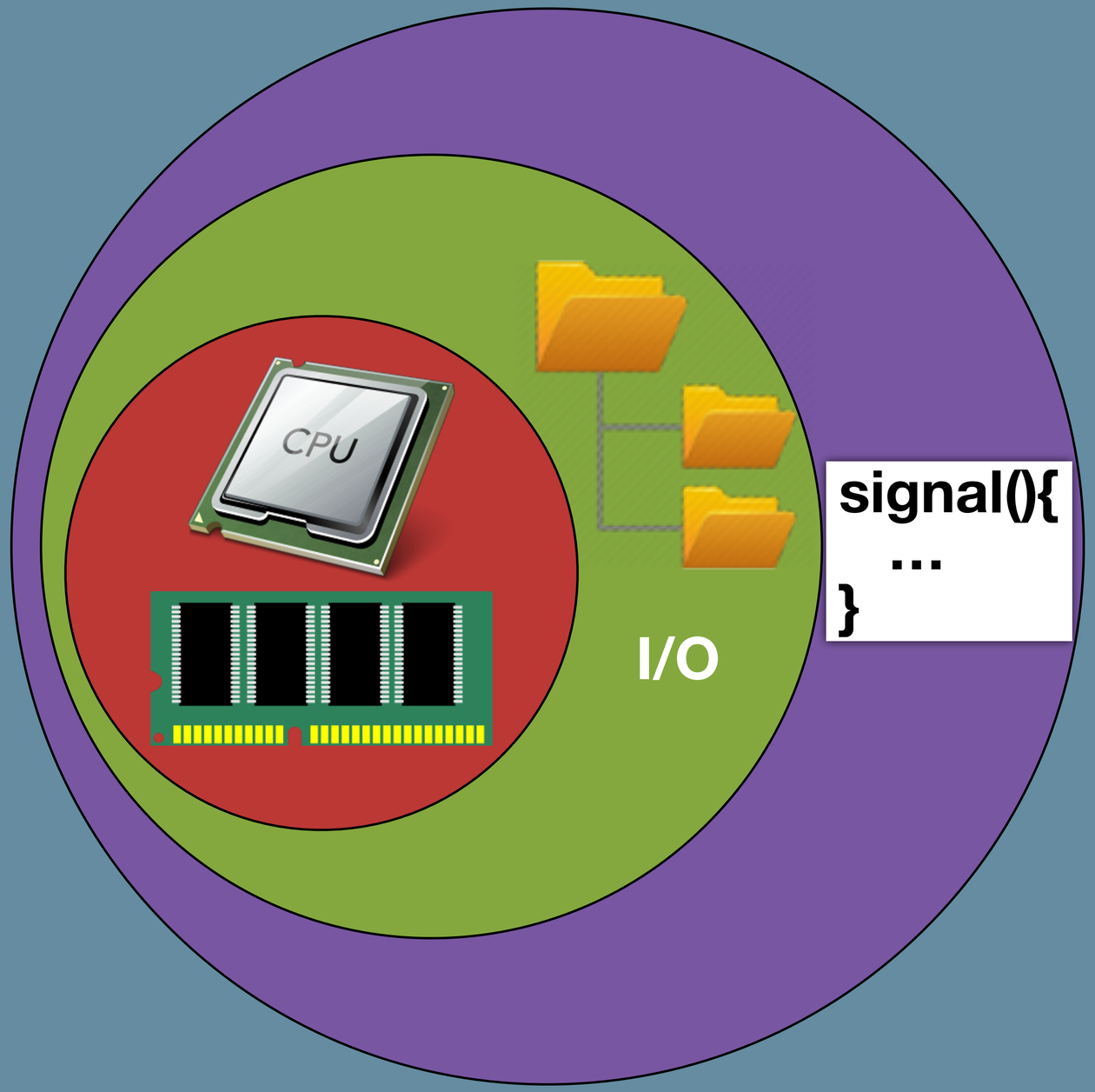

```
bit value;
void handler(...) {
   value ← 1
}

main(){
   value ← 0
   signal(31,handler)
   if (b == 1)
      raise(31)
   a ← value
}
```




```
bit value;
void handler(...) {
   value ← 1 b
}

main(){
   value ← 0
   signal(31,handler)
   if (b == 1)
      raise(31)
   a ← value
}
```




```
bit value;
void handler(...) {
   value ← 1
}

main(){
   value ← 0
   signal(31,handler)
   if (b == 1)
      raise(31)
   a ← value
}
```

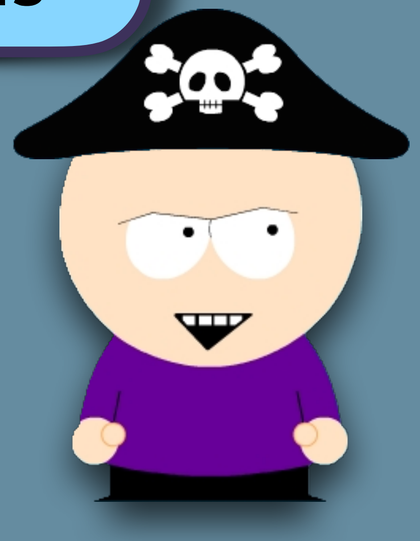



```
bit value;
void handler(...) {
   value ← 1
}

main(){
   value ← 0
   signal(31,handler)
   if (b == 1)
      raise(31)
   a ← value
}
```

```
main(){
   a ← b
}
```

Visible State

Program Analysis



```
main(){
  a ← b
}
```

Visible State

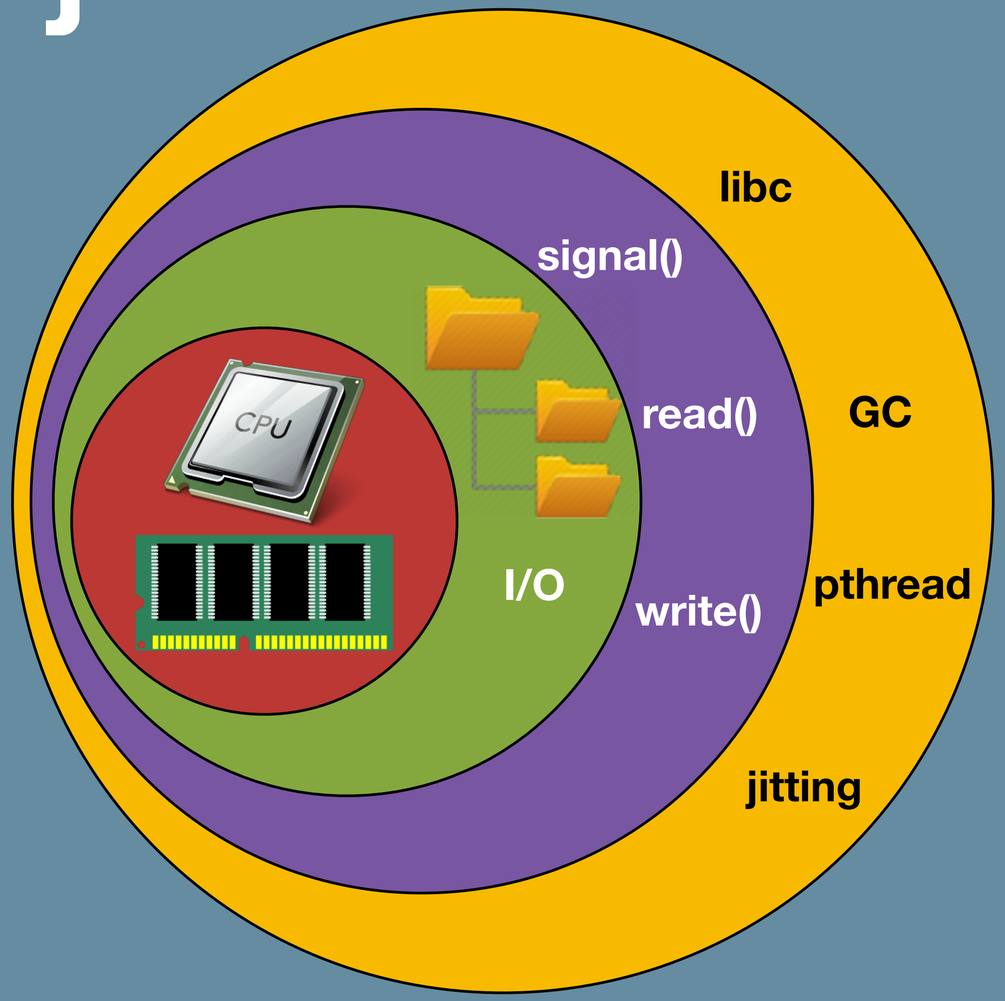
Invisible State

- Runtime clock
- State of CPU caches
- State of file cache
- State of GC
- State of JITter

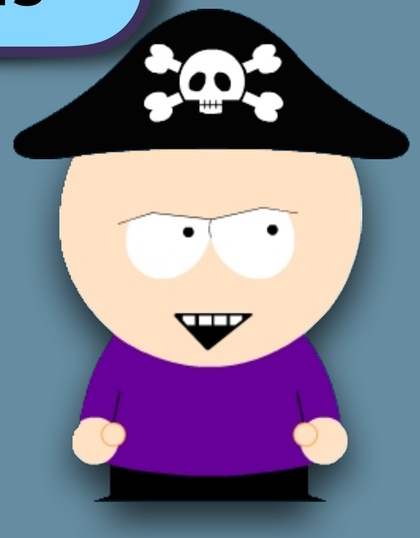
Program Analysis

```
main(){
  a ← b
}
```

Visible State

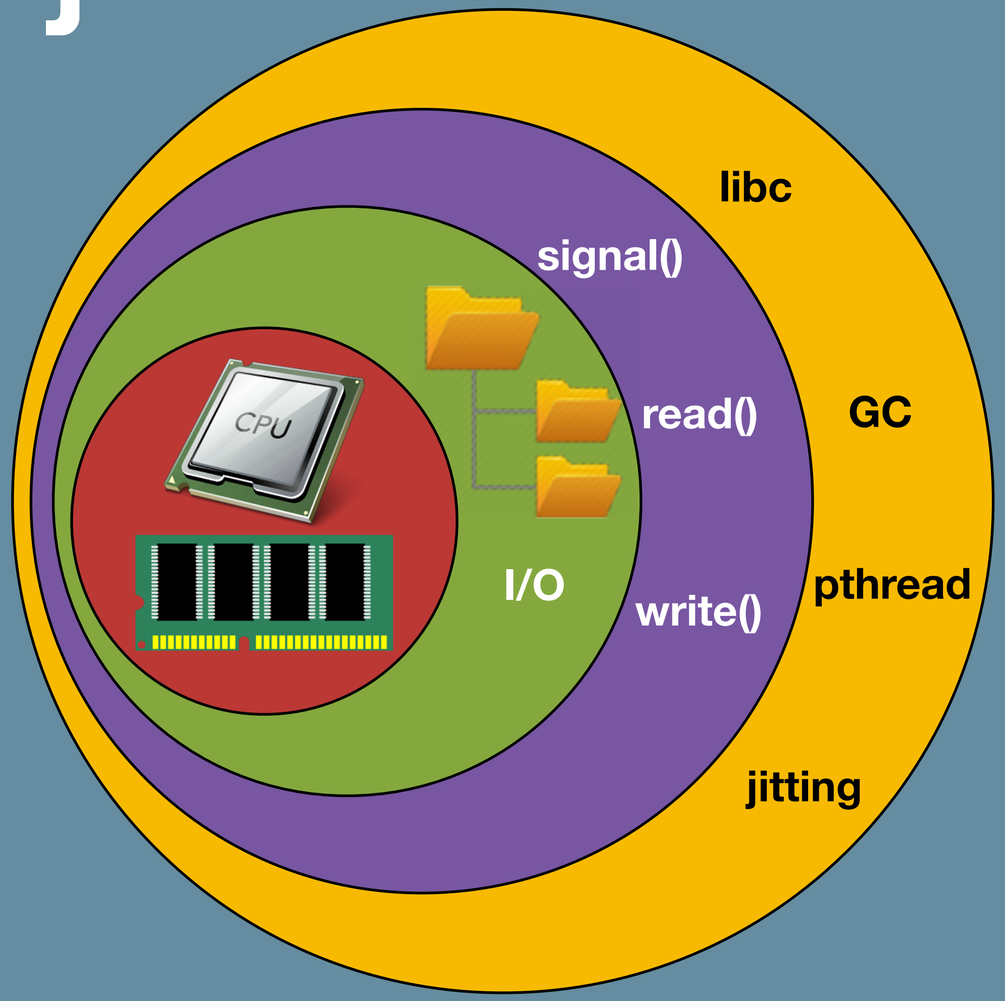


Program Analysis

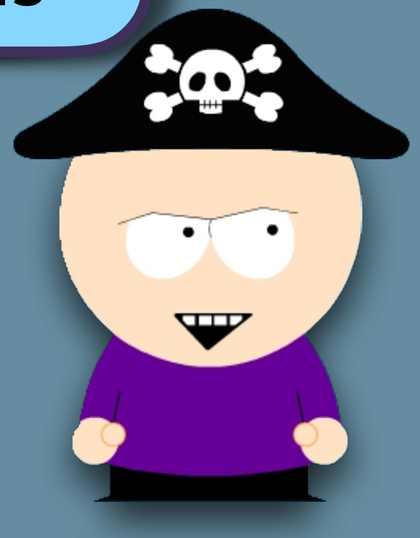


```
main(){
  a ← b
}
```

Visible State

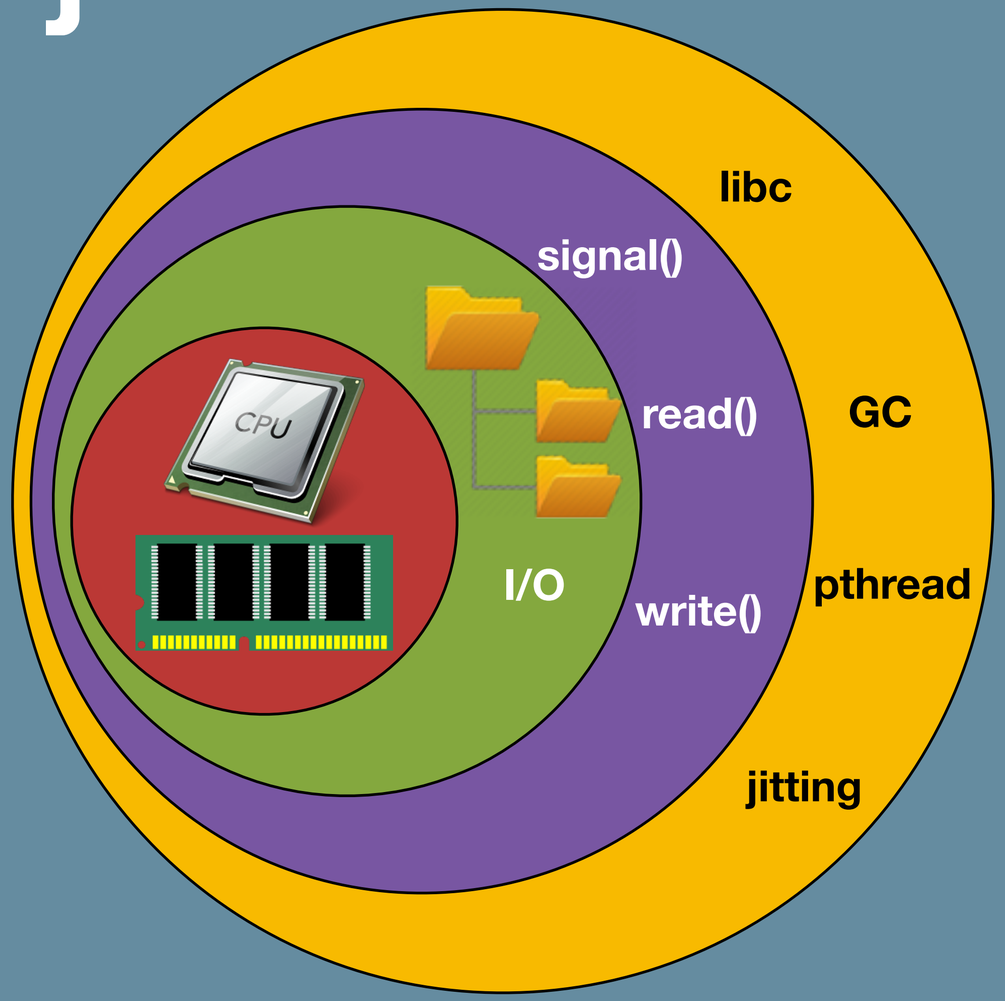


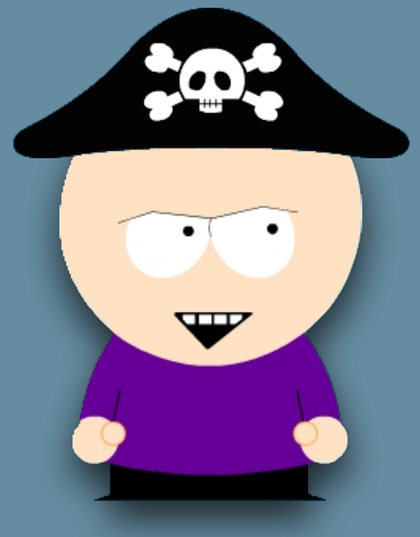
Program Analysis



```
main(){
  a ← b
}
```

Visible State

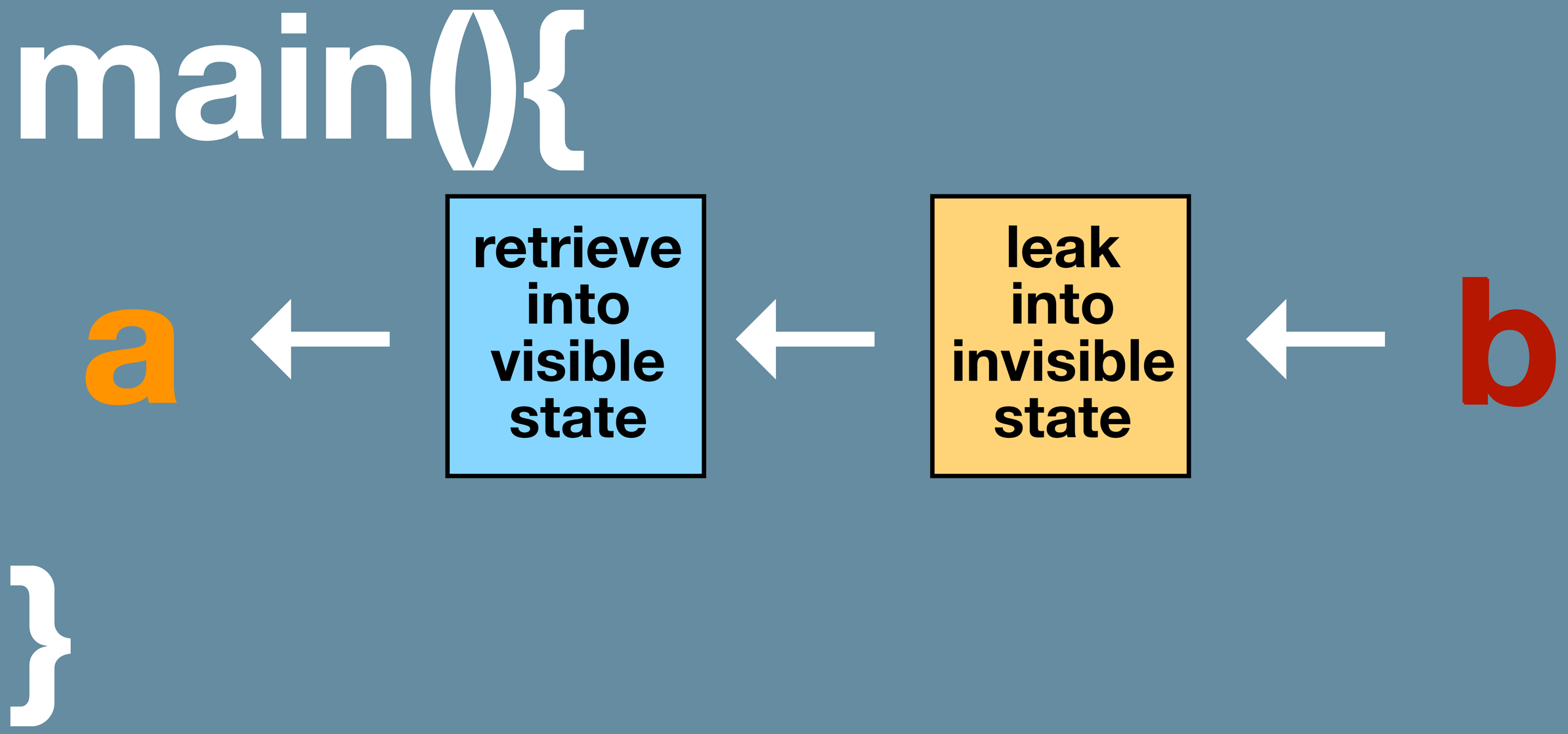




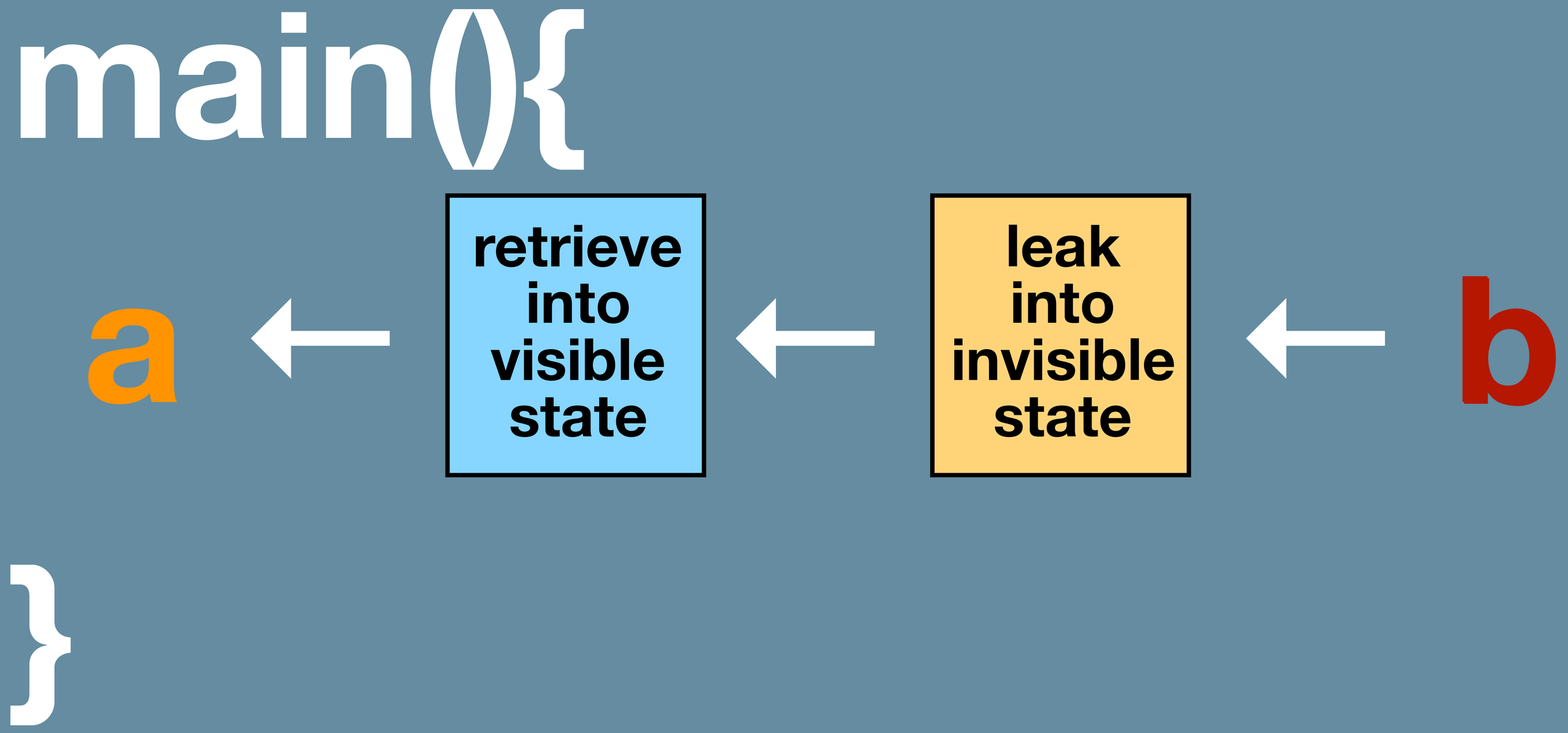
```
main(){
  a ← b
}
```

Stephens, et al., Probabilistic Obfuscation through Covert Channels, Euro S&P 2018

Stephens, et al., Probabilistic Obfuscation through Covert Channels, Euro S&P 2018



Stephens, et al., Probabilistic Obfuscation through Covert Channels, Euro S&P 2018



Stephens, et al., Probabilistic Obfuscation through Covert Channels, Euro S&P 2018

if (b == 1)

else

start ← clock()
if (b == 1)

else

a ← (clock()-start) > δ

start ← clock()

if (b == 1)

else

a ← (clock()-start) > δ

no data or control 😄 dependence!

start ← clock()
if (b == 1)

else

a ← (clock()-start) > δ

no data or control dependence! 😄

Program Analysis

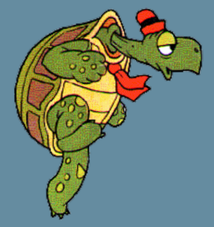
"a depends on clock()"

start ← clock()
if (b == 1)

else

a ← (clock()-start) > δ

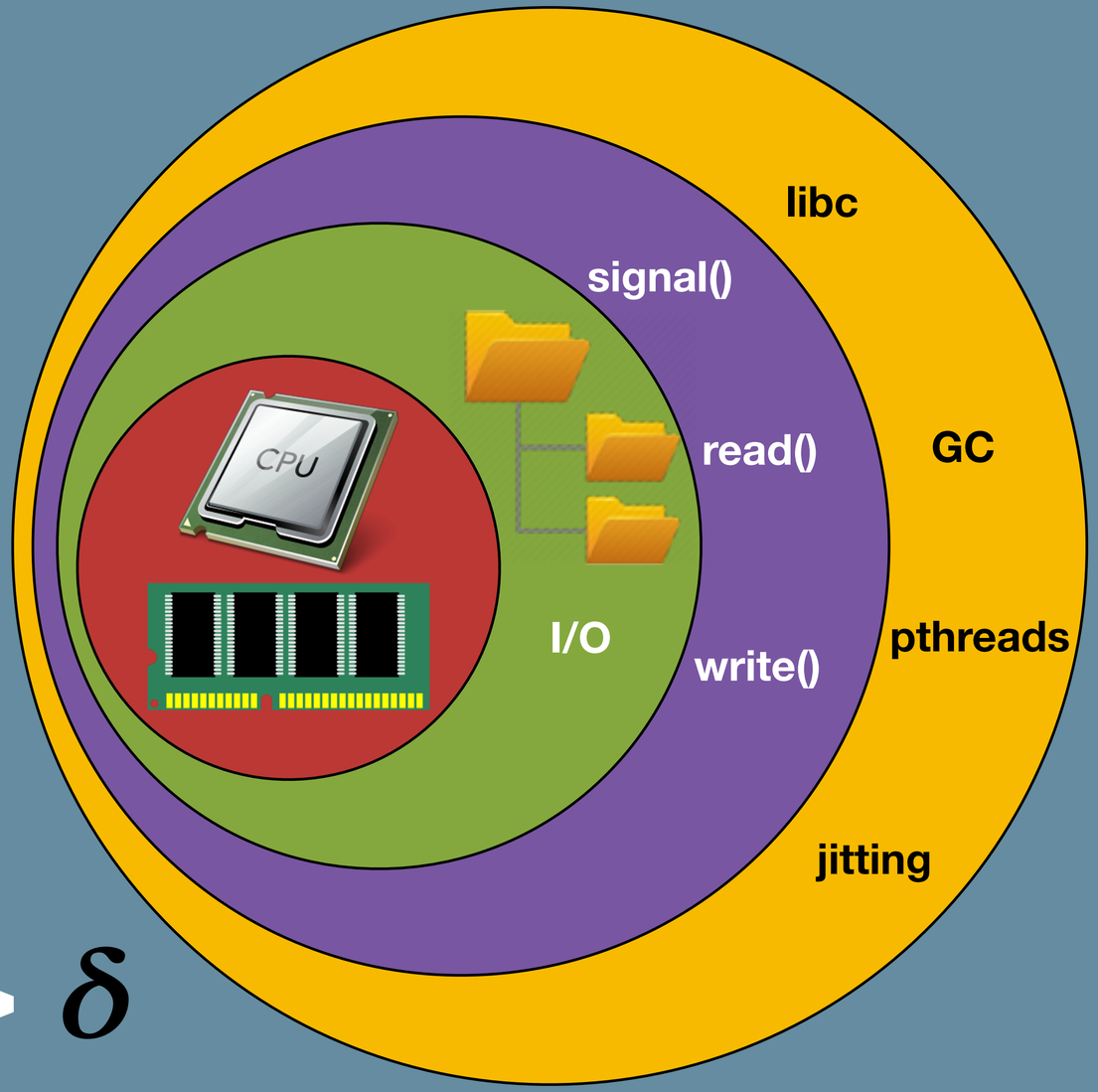
```
start ← clock()
if (b == 1)
    for(i=0; i<n; i++);
else
    for(i=0; i<m; i++);
a ← (clock()-start) > δ
```



start ← clock()
if (b == 1)

else

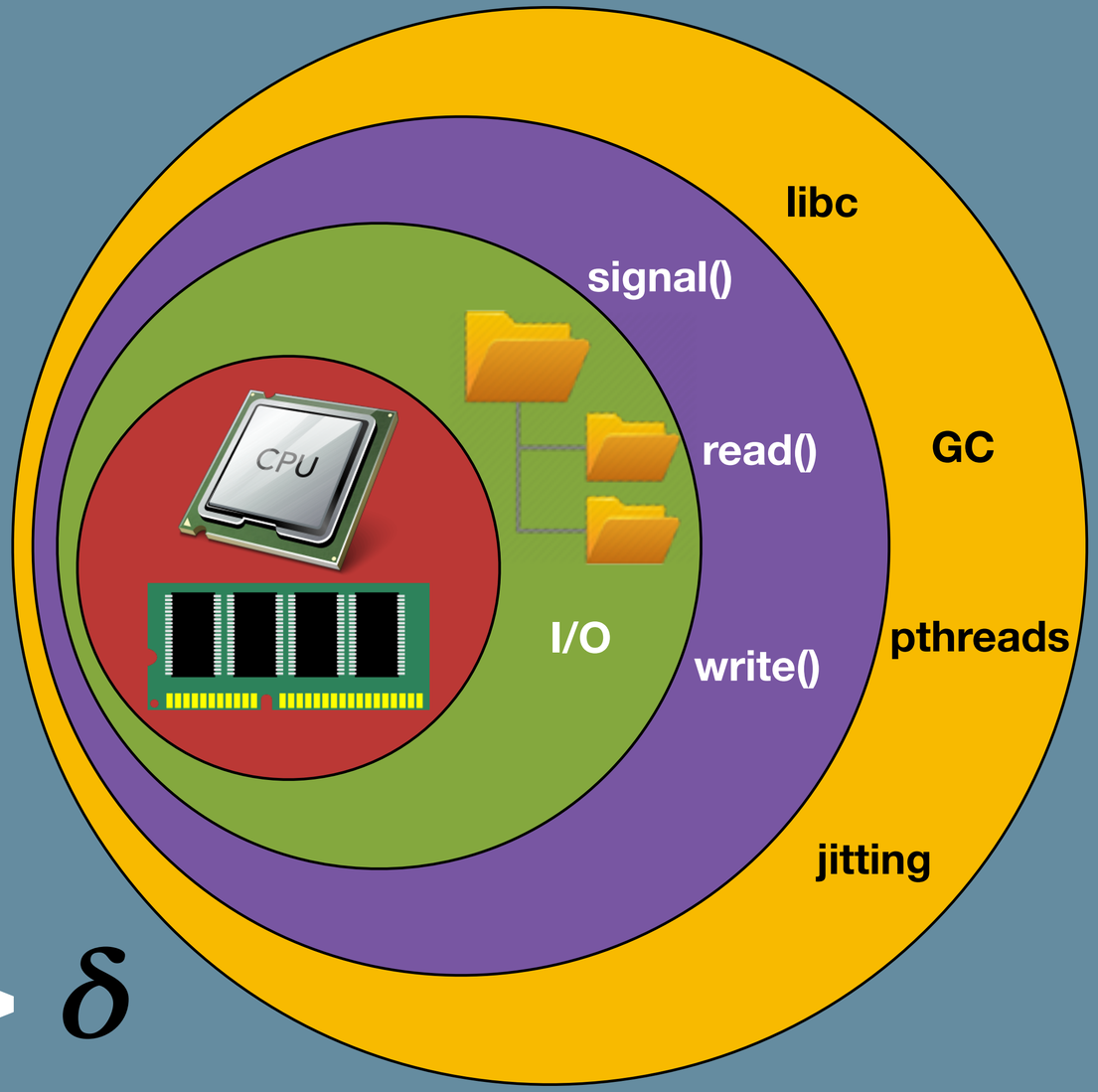
a ← (clock()-start) > δ



start ← clock()
if (b == 1)

else

a ← (clock()-start) > δ



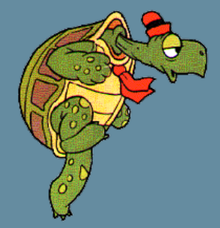
```
for (i=0; i<n; i++)
  if (b == 1)
    flush buf1 ; read buf1 ; write buf1
  else
    flush buf1 ; read buf2 ; write buf2
```

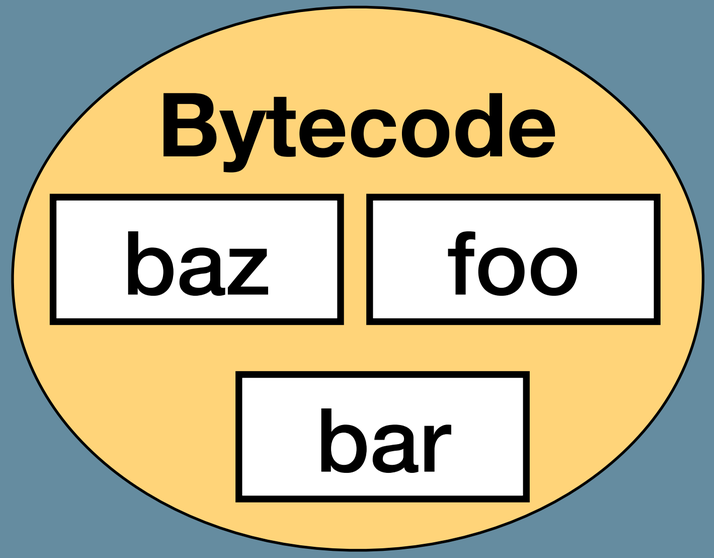
```
start ← clock()

for (i=0; i<n; i++)
   if (b == 1)
      flush buf1 ; read buf1 ; write buf1
   else
      flush buf1 ; read buf2 ; write buf2

a ← (clock()-start) > δ
```

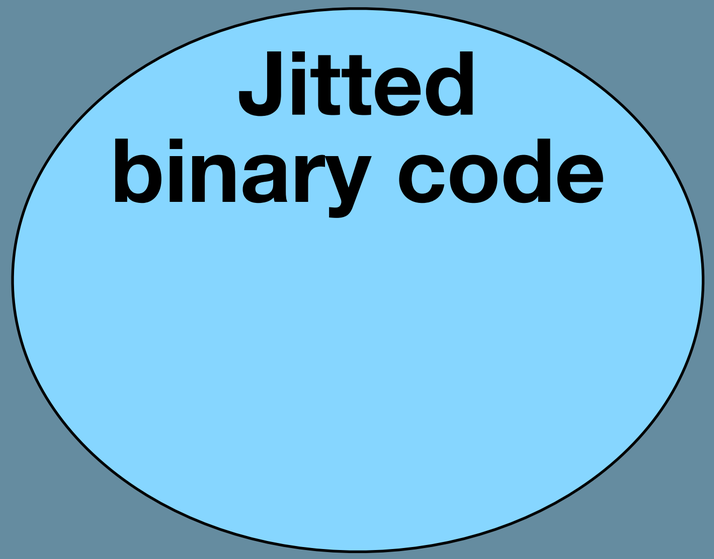
```
if (b == 1)
   open(“foo”,NOCACHE)
else
   open(“foo”,CACHE)
```

```
if (b == 1)
   open("foo",NOCACHE)
else
   open("foo",CACHE)
start ← clock()
read("foo")
a ← (clock()-start) > δ
```

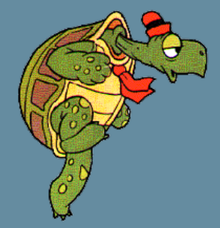
```
if (b == 1)
    ;
else
    foo()
```

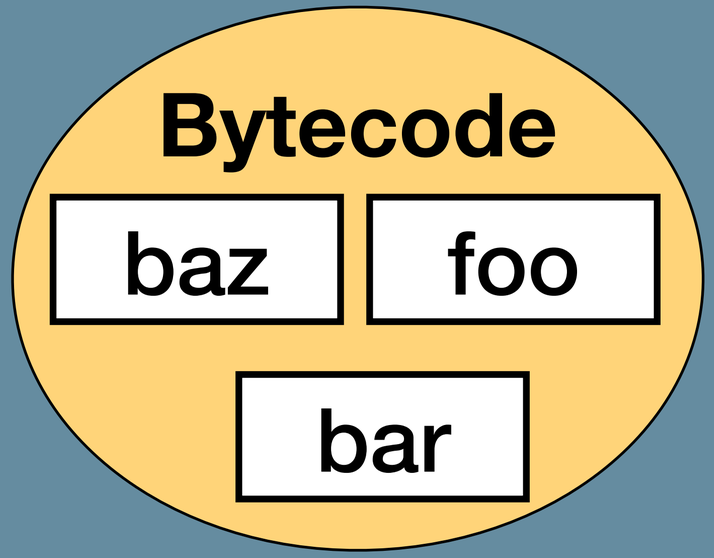


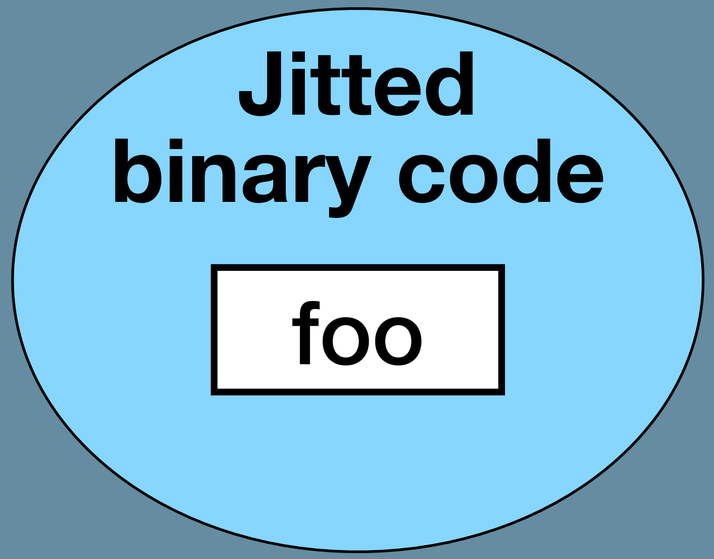




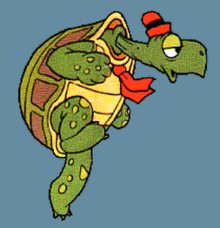
```
if (b == 1)
    ;
else
    foo()
```

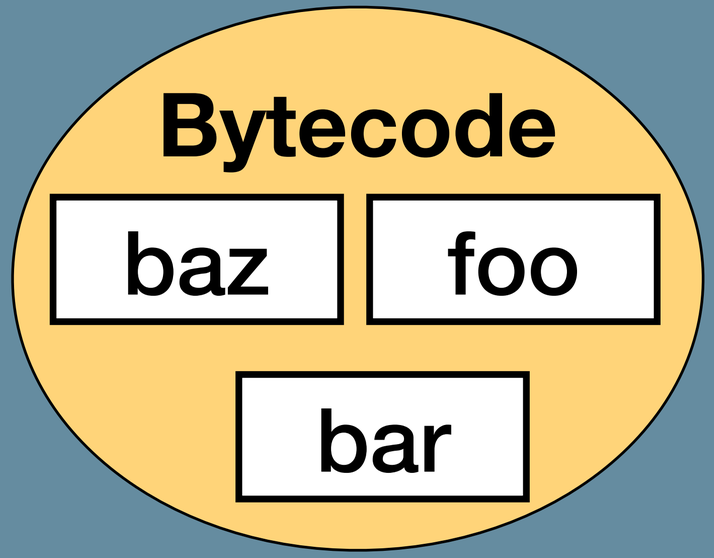


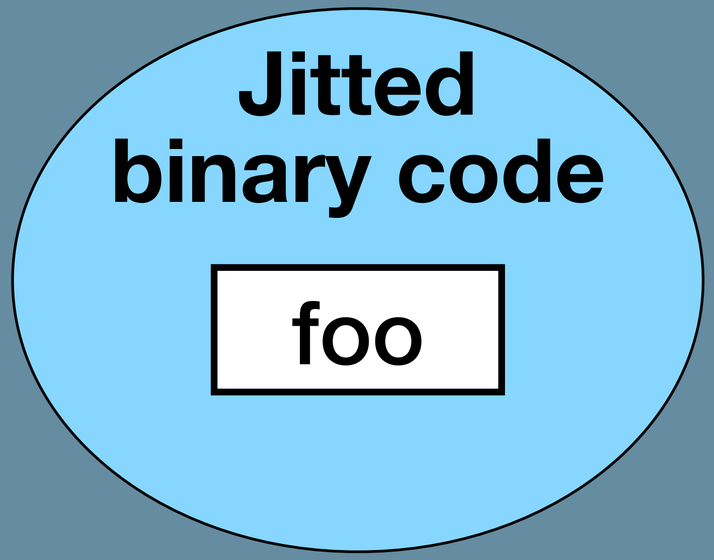




```
if (b == 1)
   ;
else
   foo()
start ← clock()
foo()
a ← (clock()-start) > δ
```

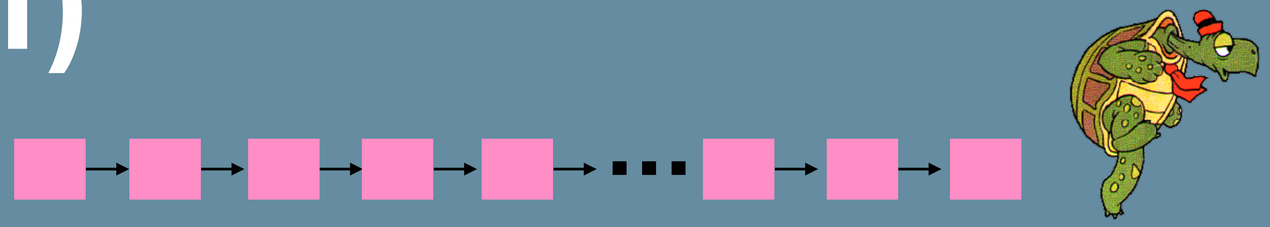




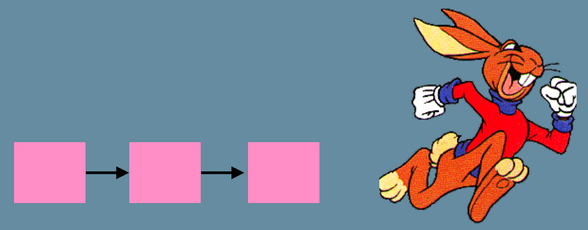


GC()

if (b == 1)

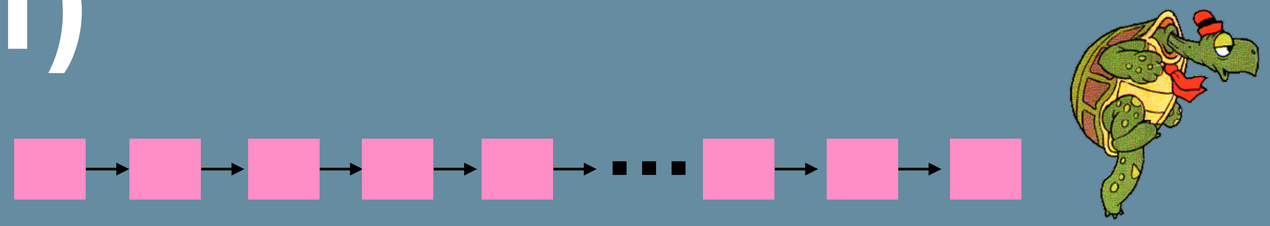
list ← 

else

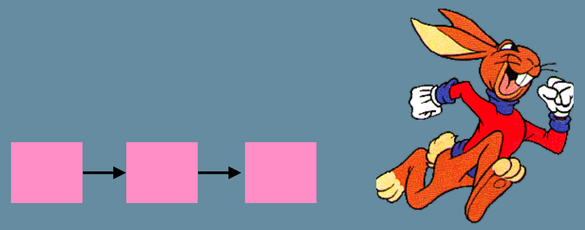
list ← 

GC()

if (b == 1)

list ← 

else

list ← 

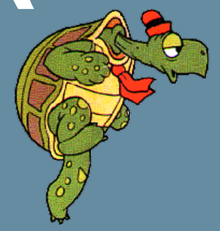
start ← clock()

GC()

a ← (clock()-start) > δ

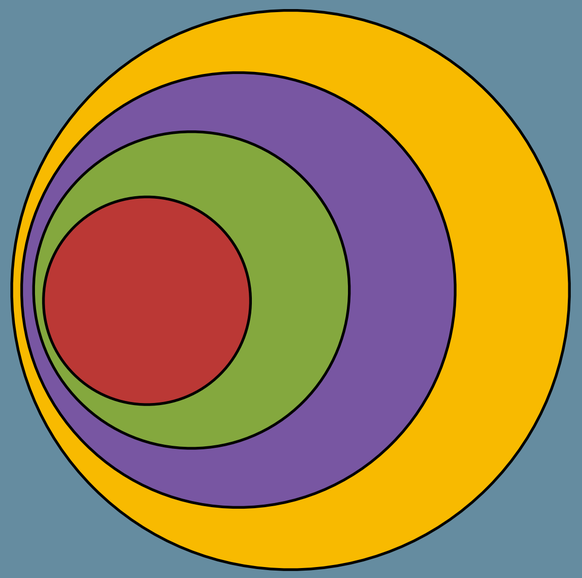
Issue #1: Pattern-Matching Attacks

start ← clock()
if (b == 1)



else

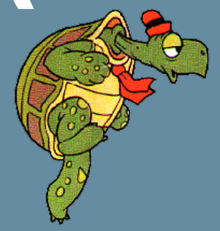
a ← (clock()-start)> δ



Pattern Analysis

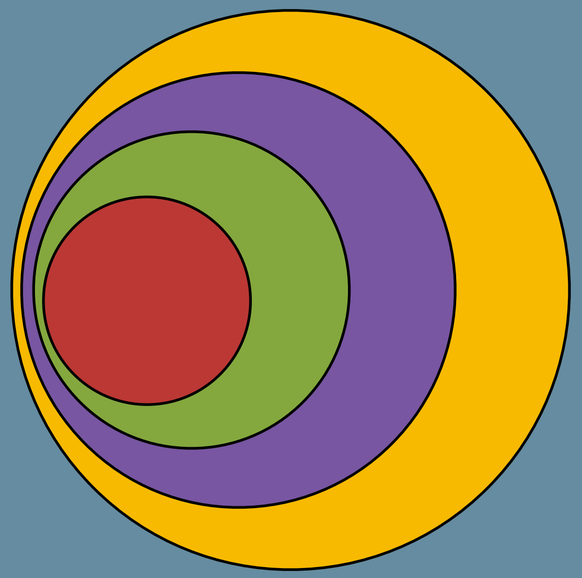
Issue #1: Pattern-Matching Attacks

start ← clock()
if (b == 1)



else

a ← (clock()-start)> δ

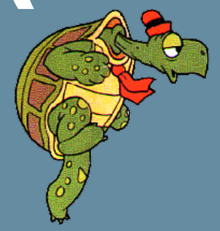


Pattern Analysis

□ ← *time*()
if (□)
□
else
□
□ ← (*time*()-□)>□

Issue #1: Pattern-Matching Attacks

start ← clock()
if (b == 1)



else

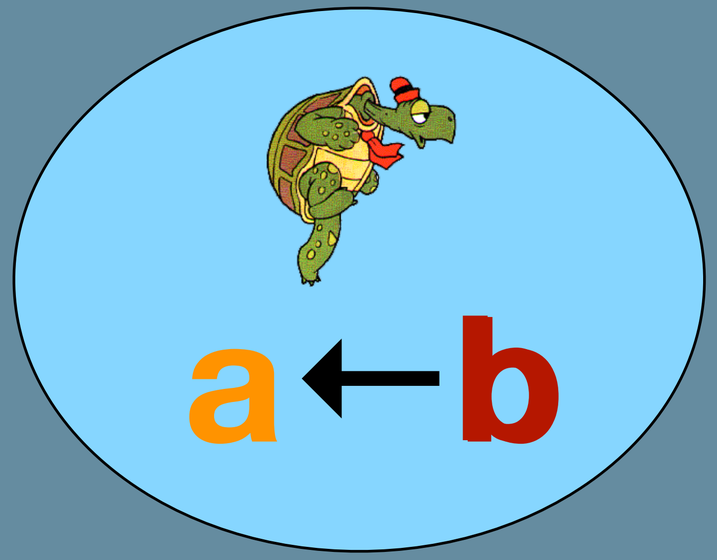
a ← (clock()-start)> δ

Pattern Analysis

□ ← *time*()
if (□)
□
else
□
□ ← (*time*()-□)>□

```
main(){

  bit a← b

}
```

```
main(){
  create( )

  bit a ← b

}
```



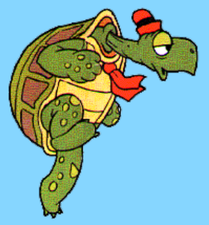
main(){

create( )

create( )

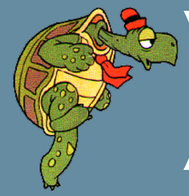
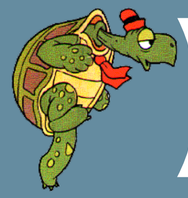
bit a ← b

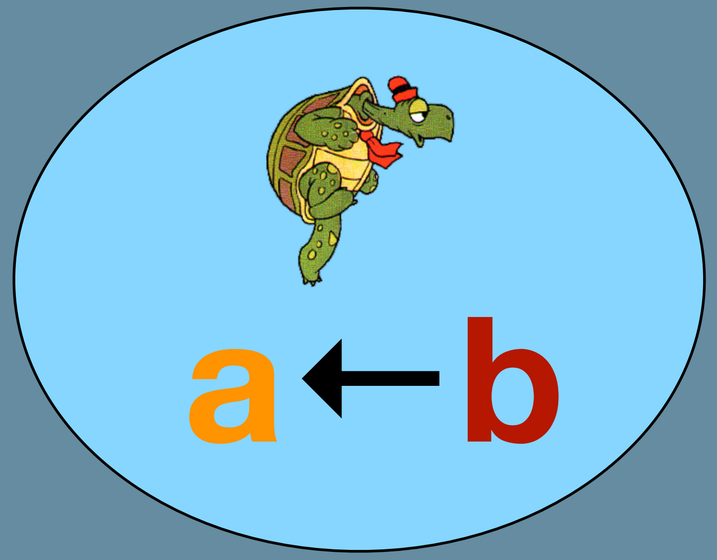
}

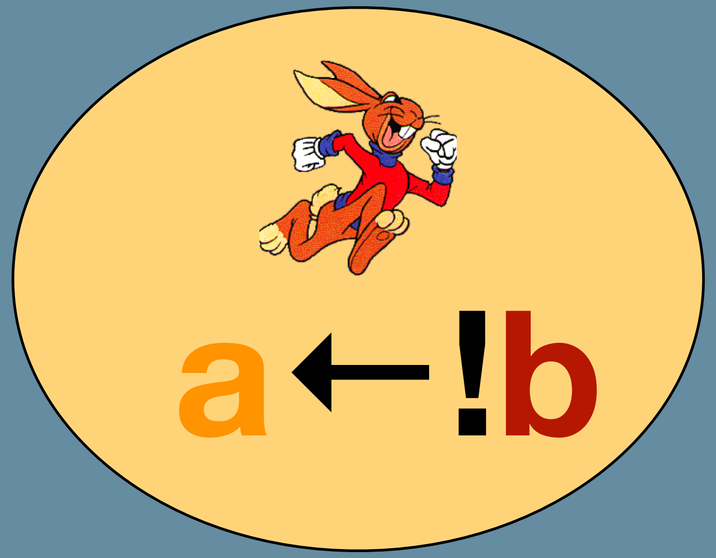


a ← b

a ← !b

```
main(){
  create(  )
  create( )
  bit a ← b
  join( ,  )
}
```





Issue #2: Correctness

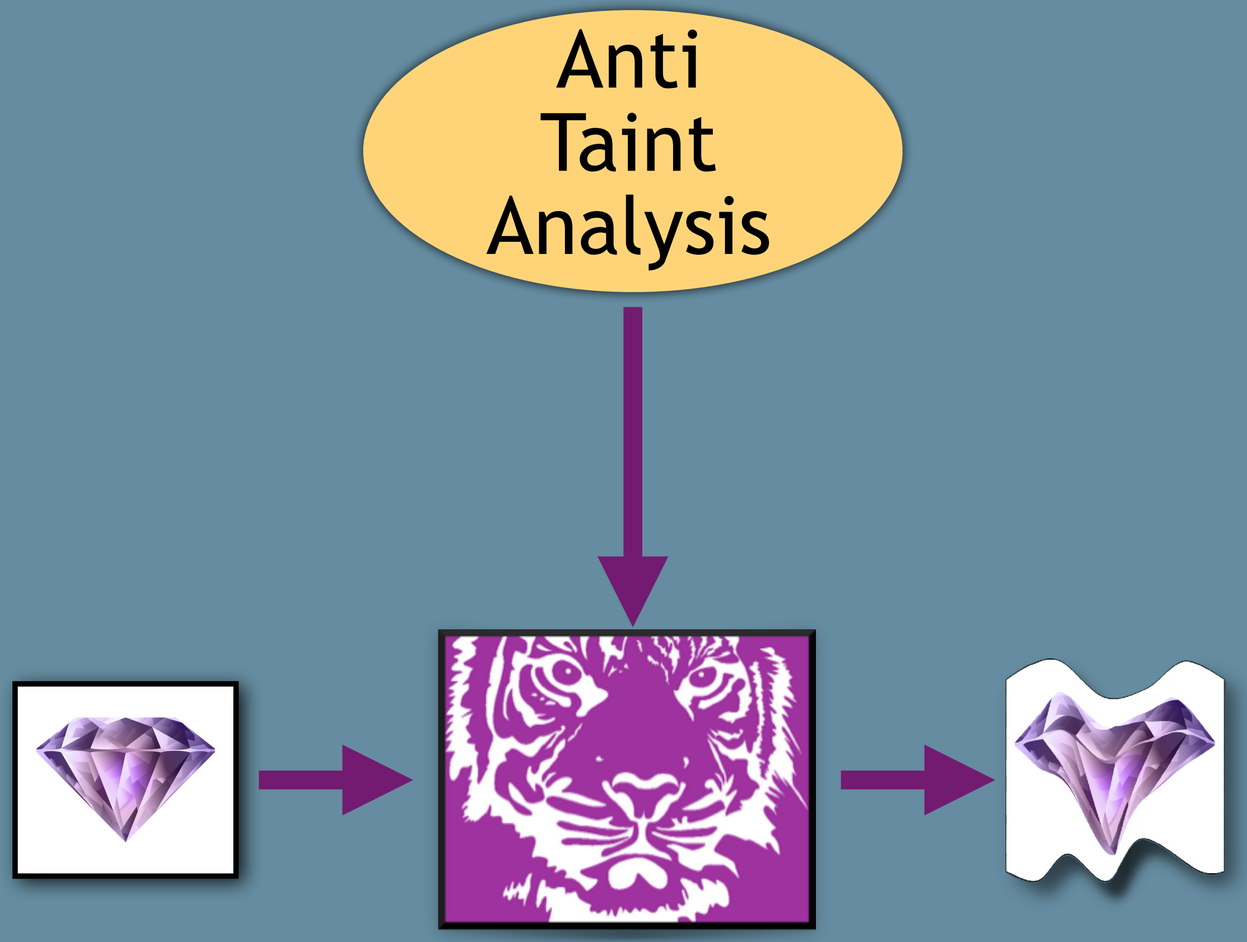
```
main(){
   bit a←b
}
```

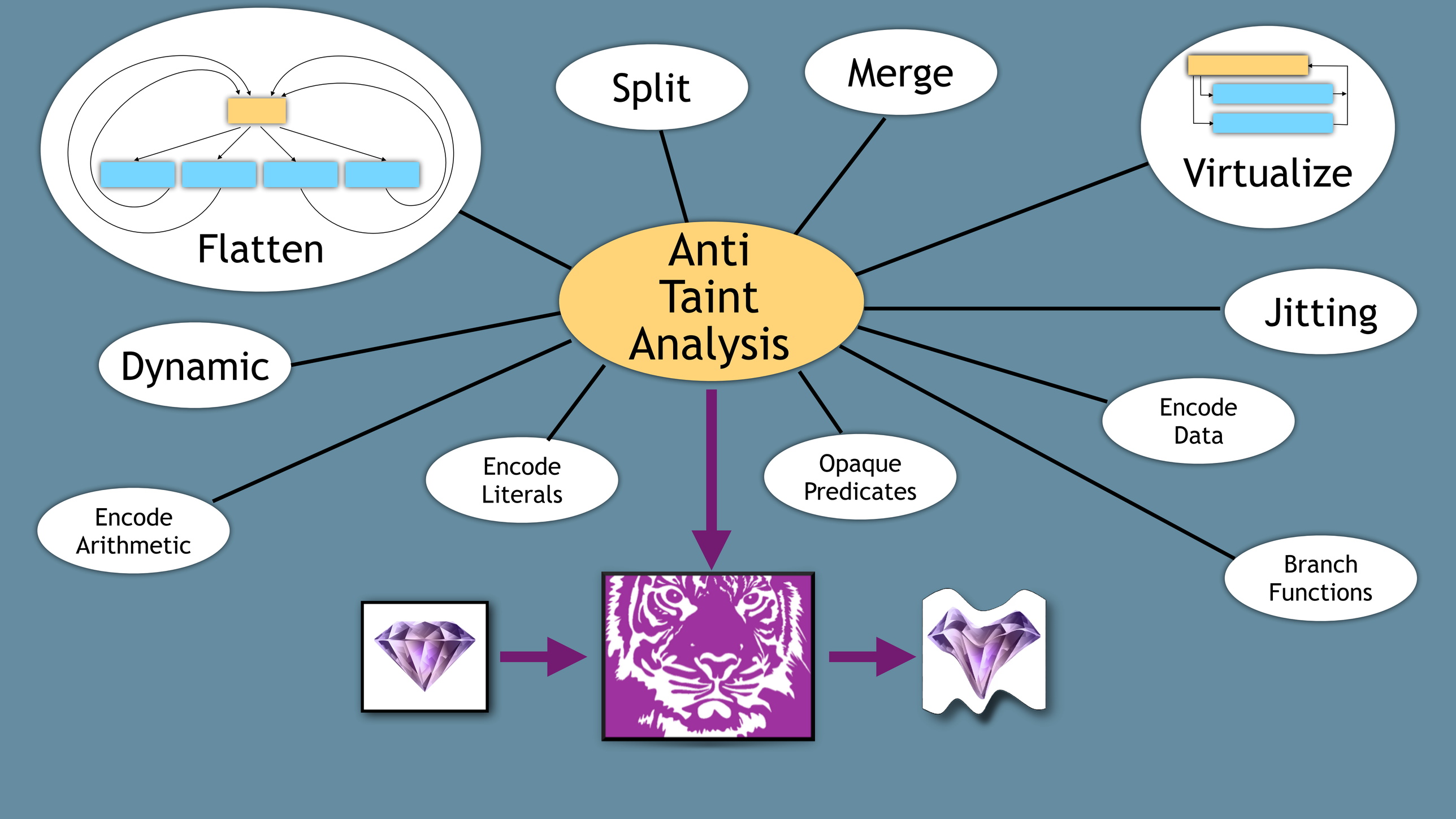
```
start ← clock()
flush
if (b == 1)
    read
    write
else
    read
    write
a ← (clock()-start) > δ
```

Issue #2: Correctness

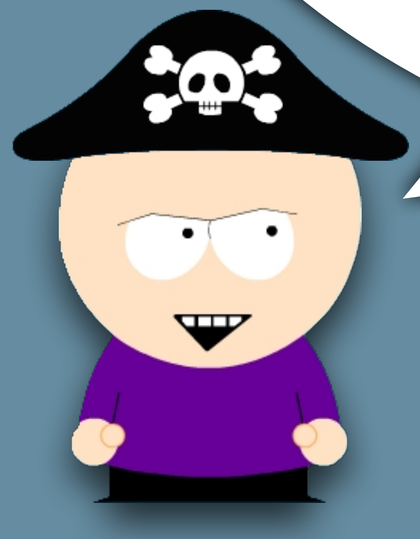
```
main(){
   bit a←b
}
```

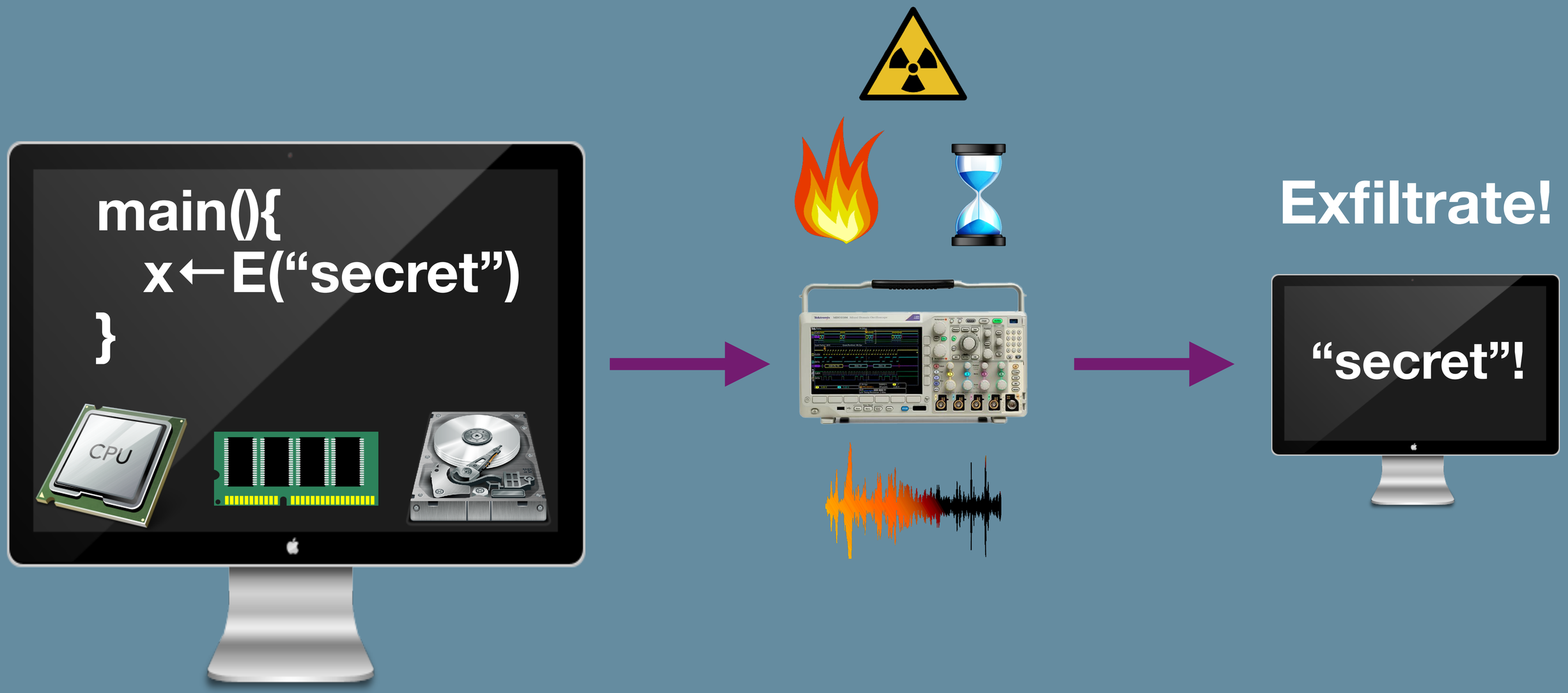
```
start ← clock()
flush
if (b == 1)
    read
    write
else
    read
    write
a ← (clock()-start) > δ
```

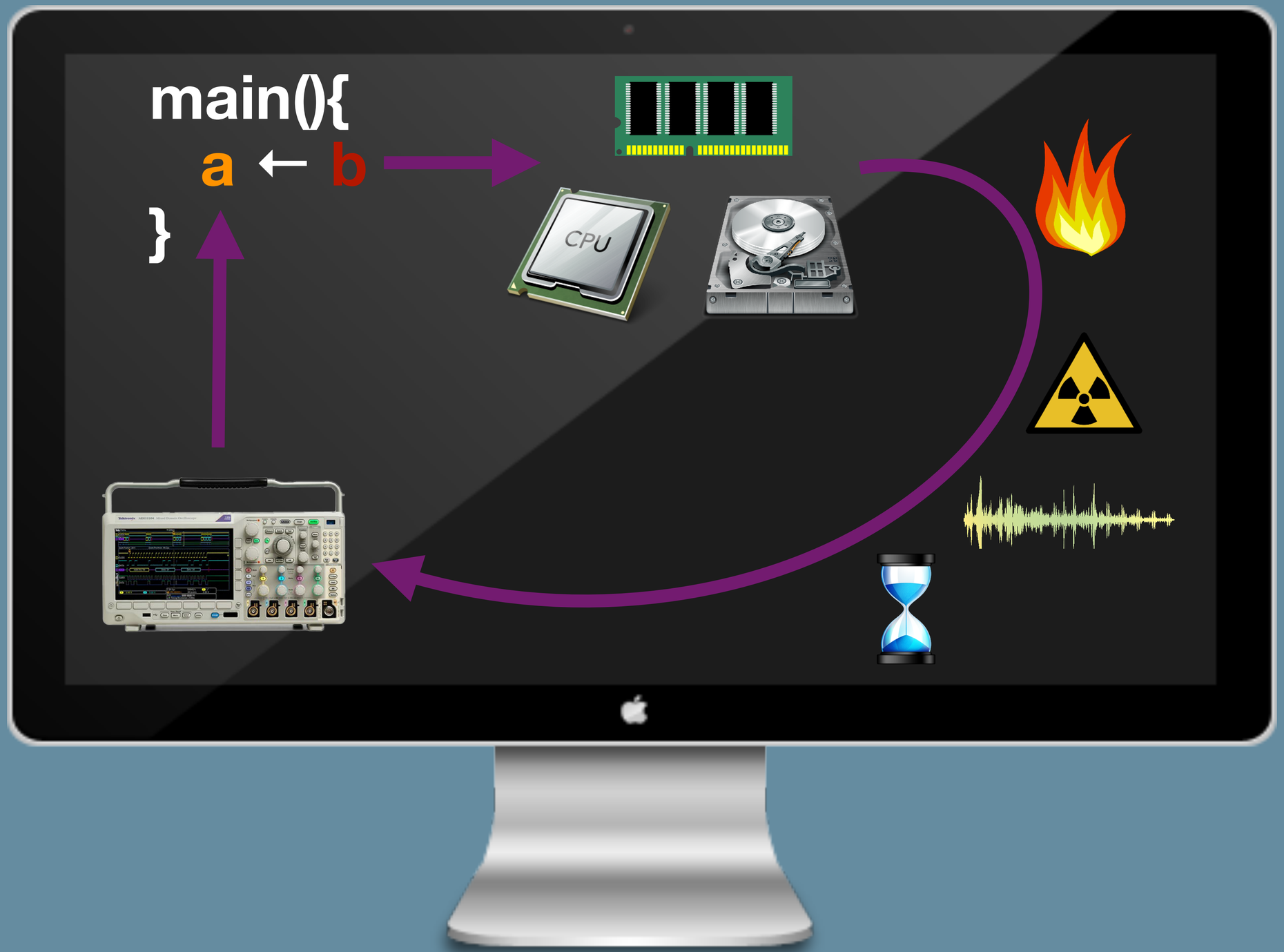

Anti
Taint
Analysis

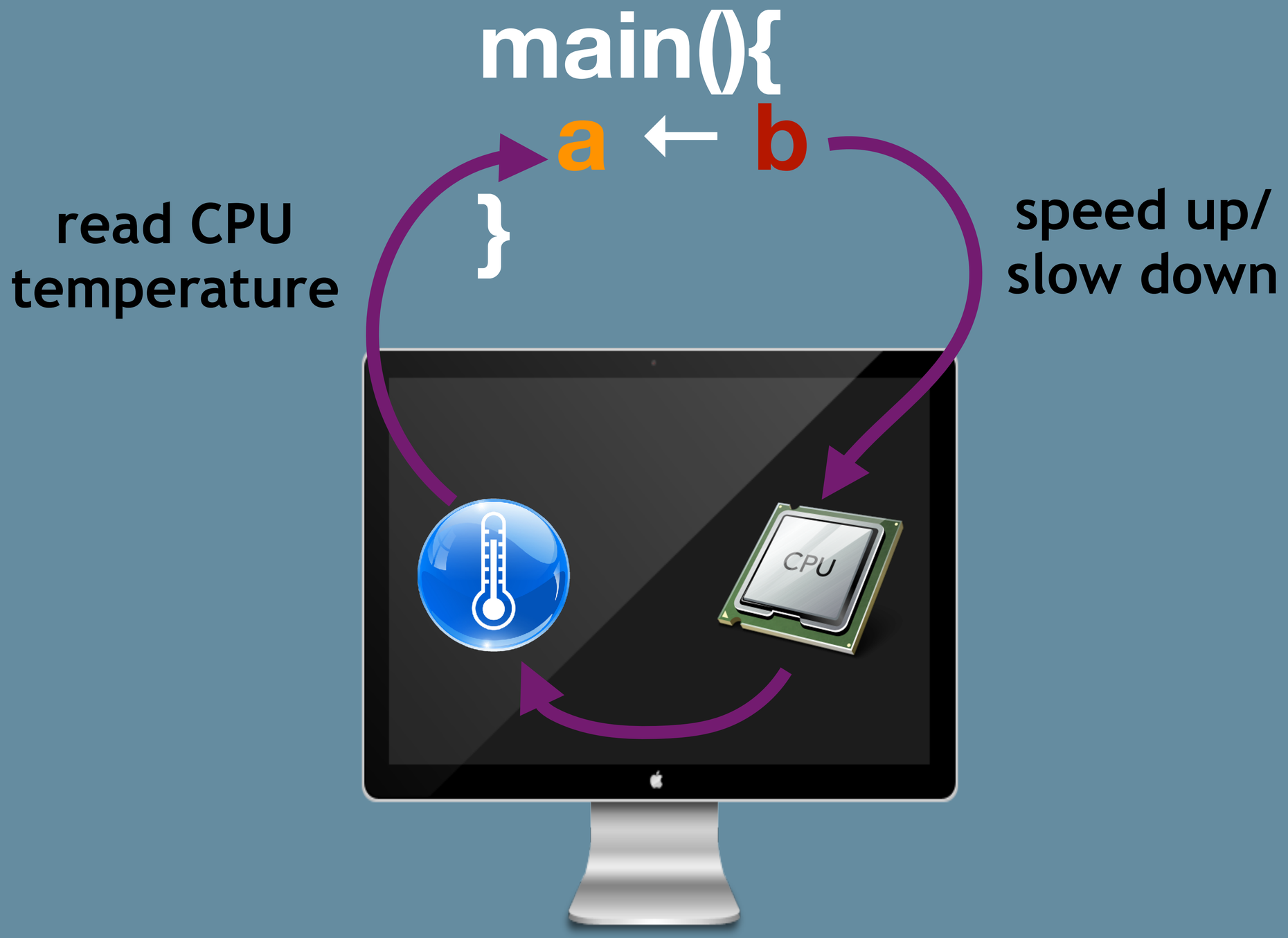

Flatten
Split
Merge
Virtualize
Anti
Taint
Analysis
Jitting
Dynamic
Encode
Data
Encode
Literals
Opaque
Predicates
Encode
Arithmetic
Branch
Functions

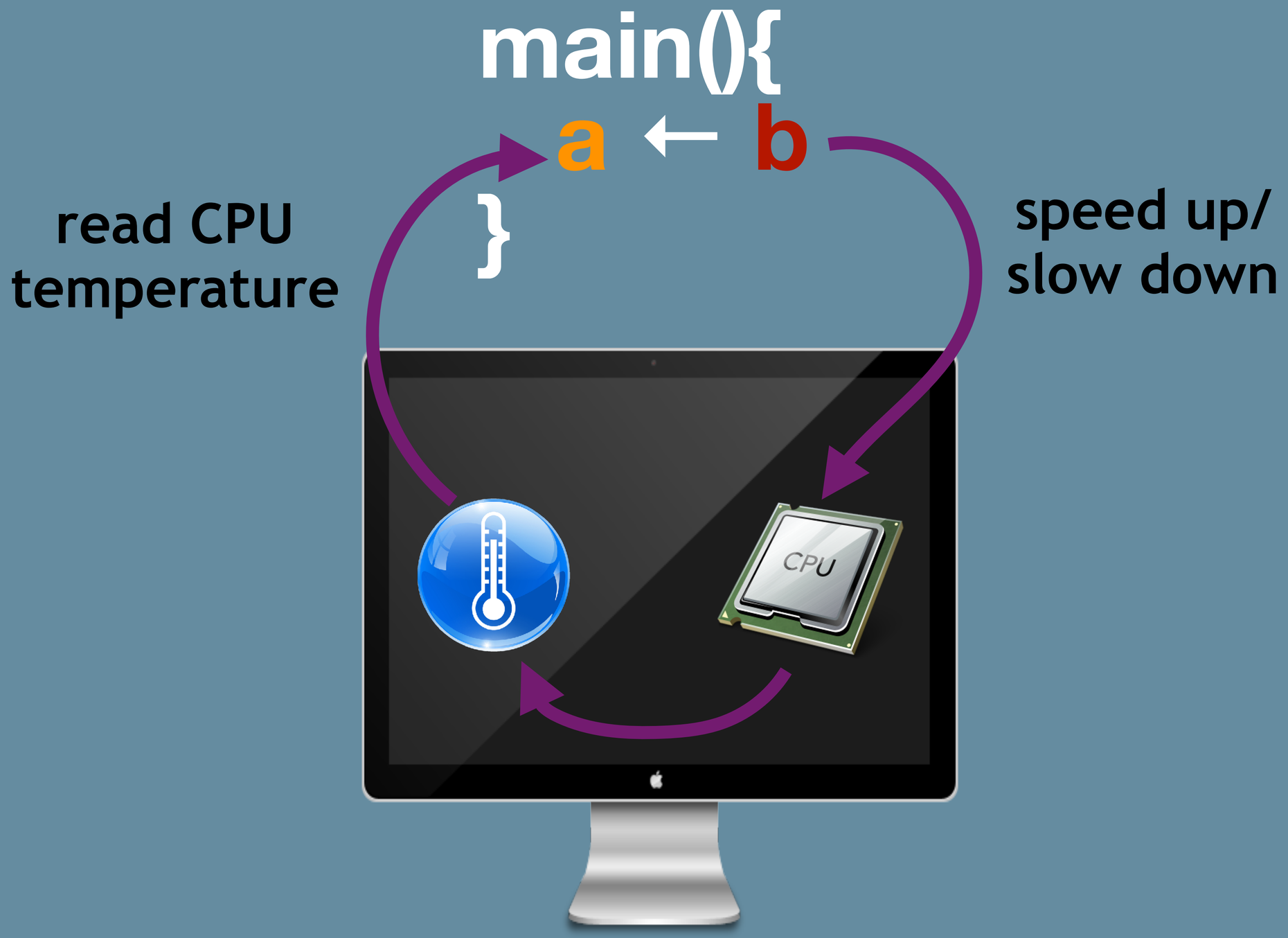
Hey, wait a minute, aren’t these just side channels?

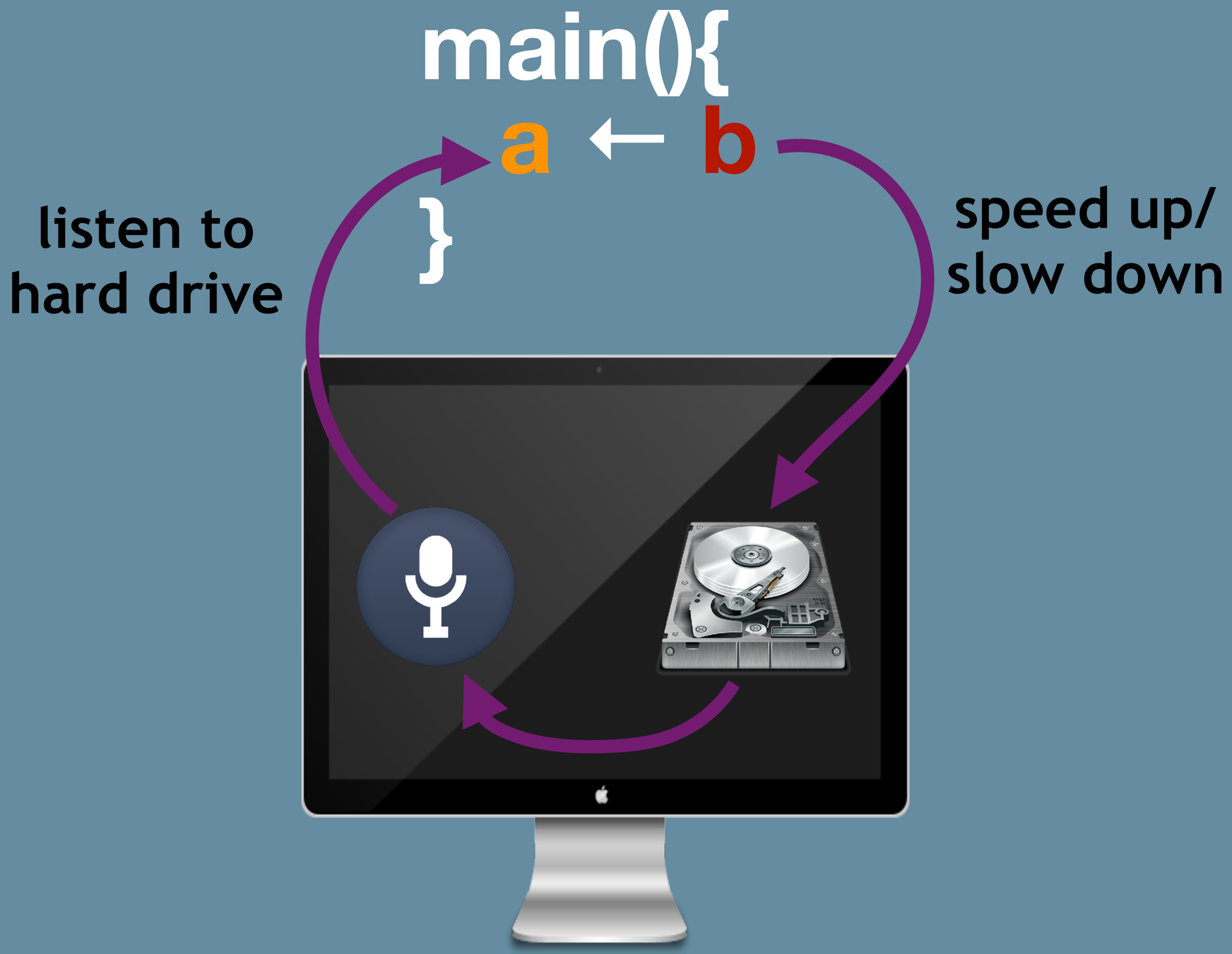


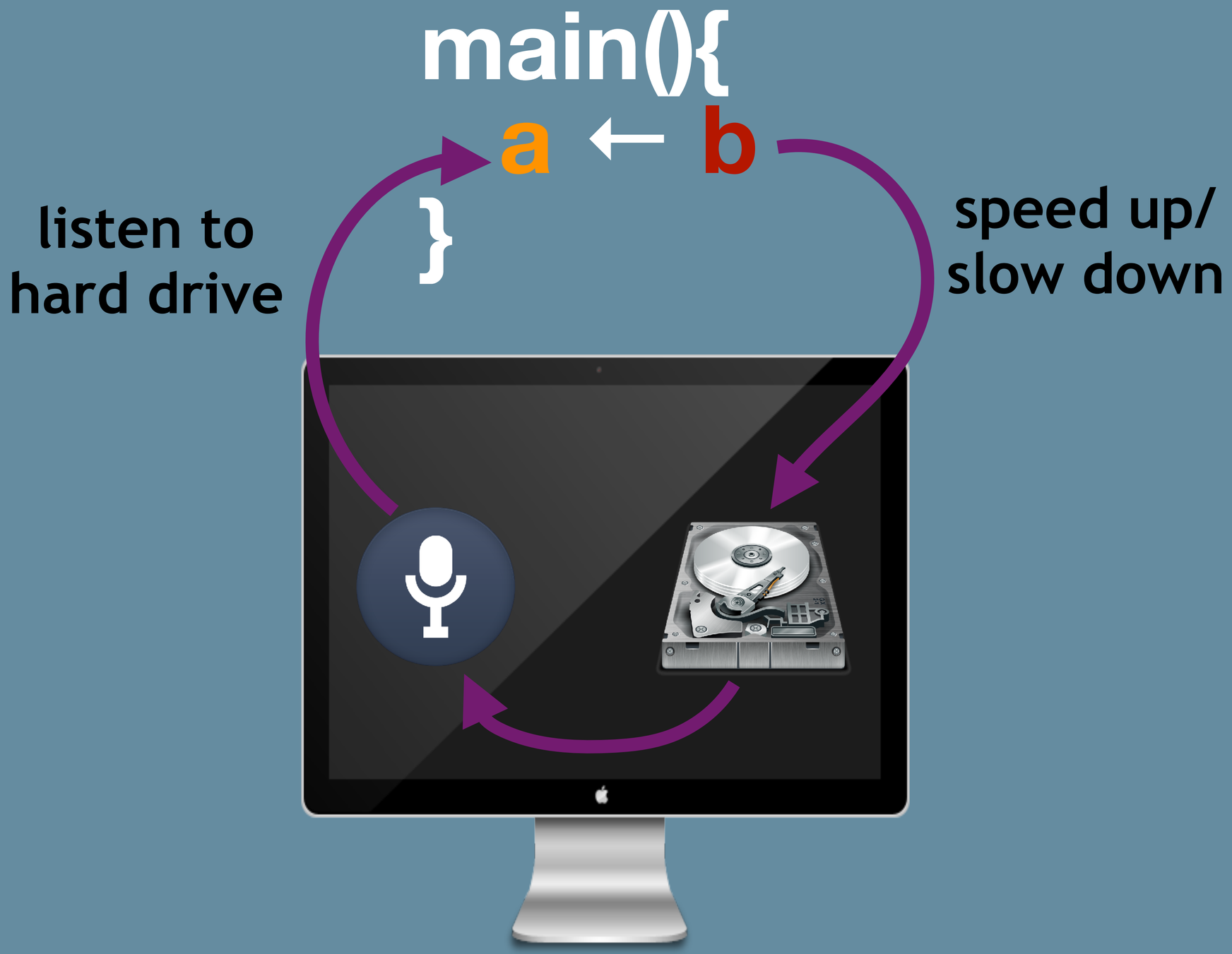

main(){
x←E("secret")
}
CPU
Exfiltrate!
"secret"!

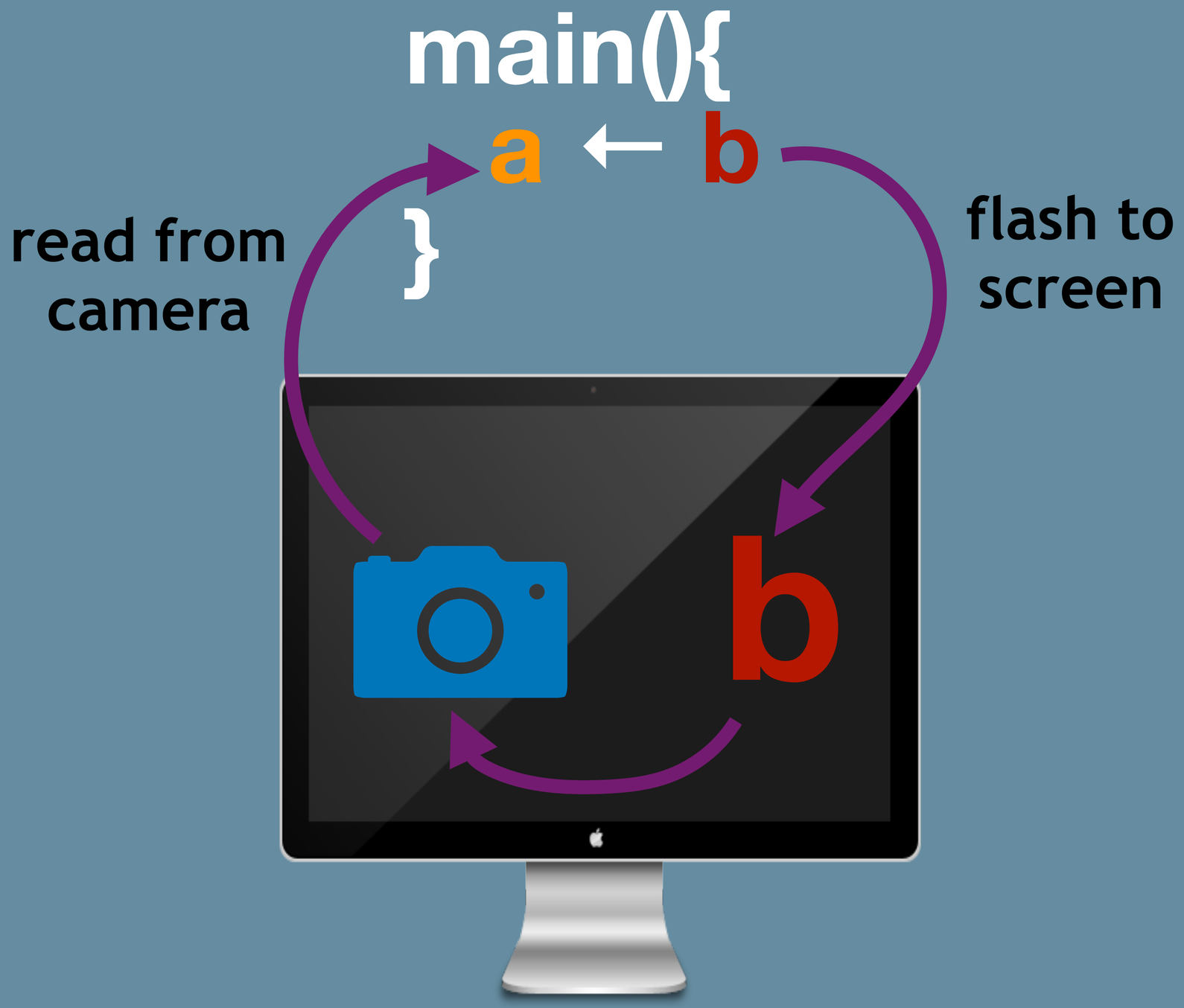

main(){
a ← b
}
CPU

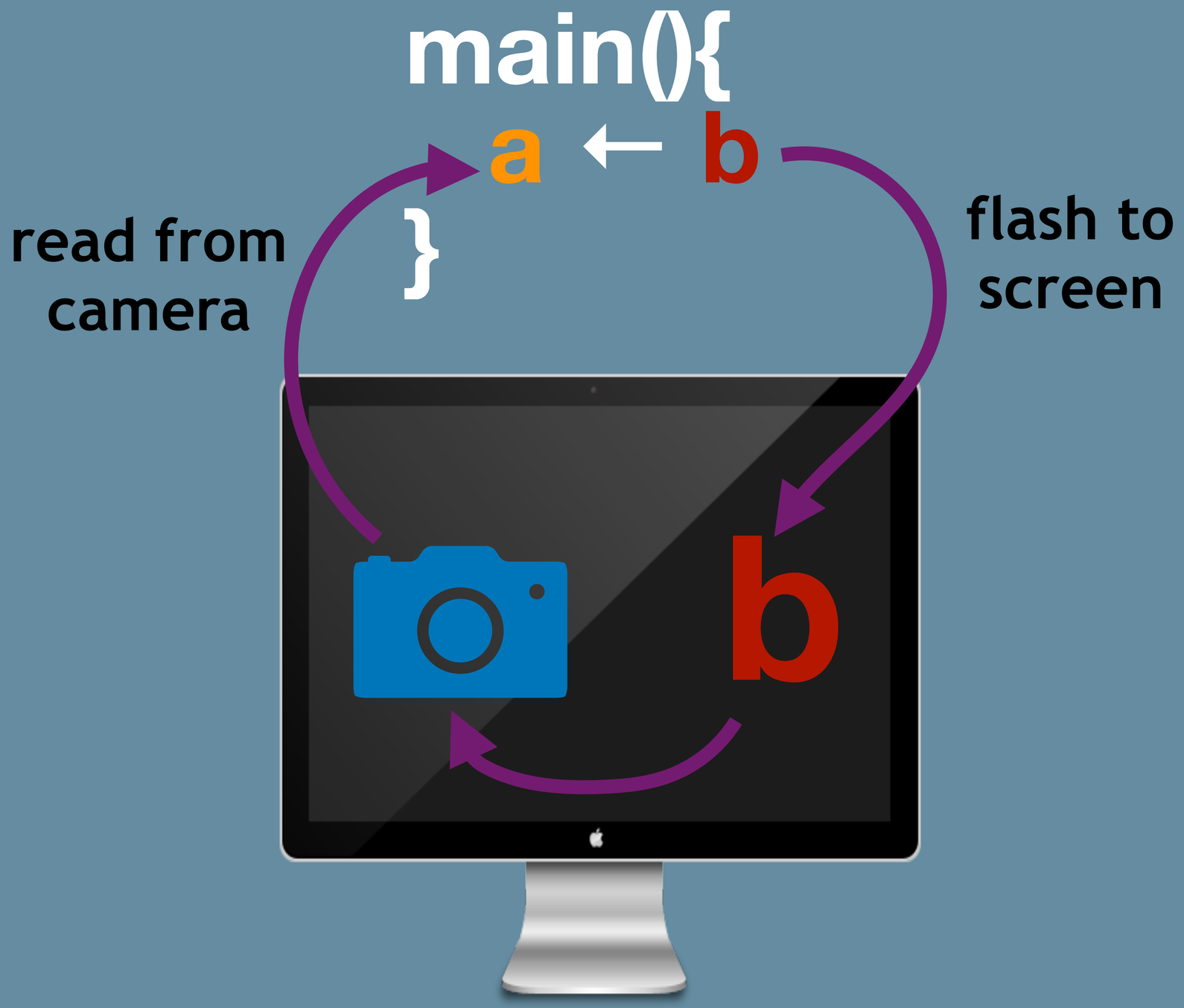

main(){
a ← b
}
read CPU temperature
speed up/ slow down
CPU


main(){
a ← b
}
read CPU temperature
speed up/ slow down
CPU


main(){
a ← b
}
listen to
hard drive
speed up/
slow down

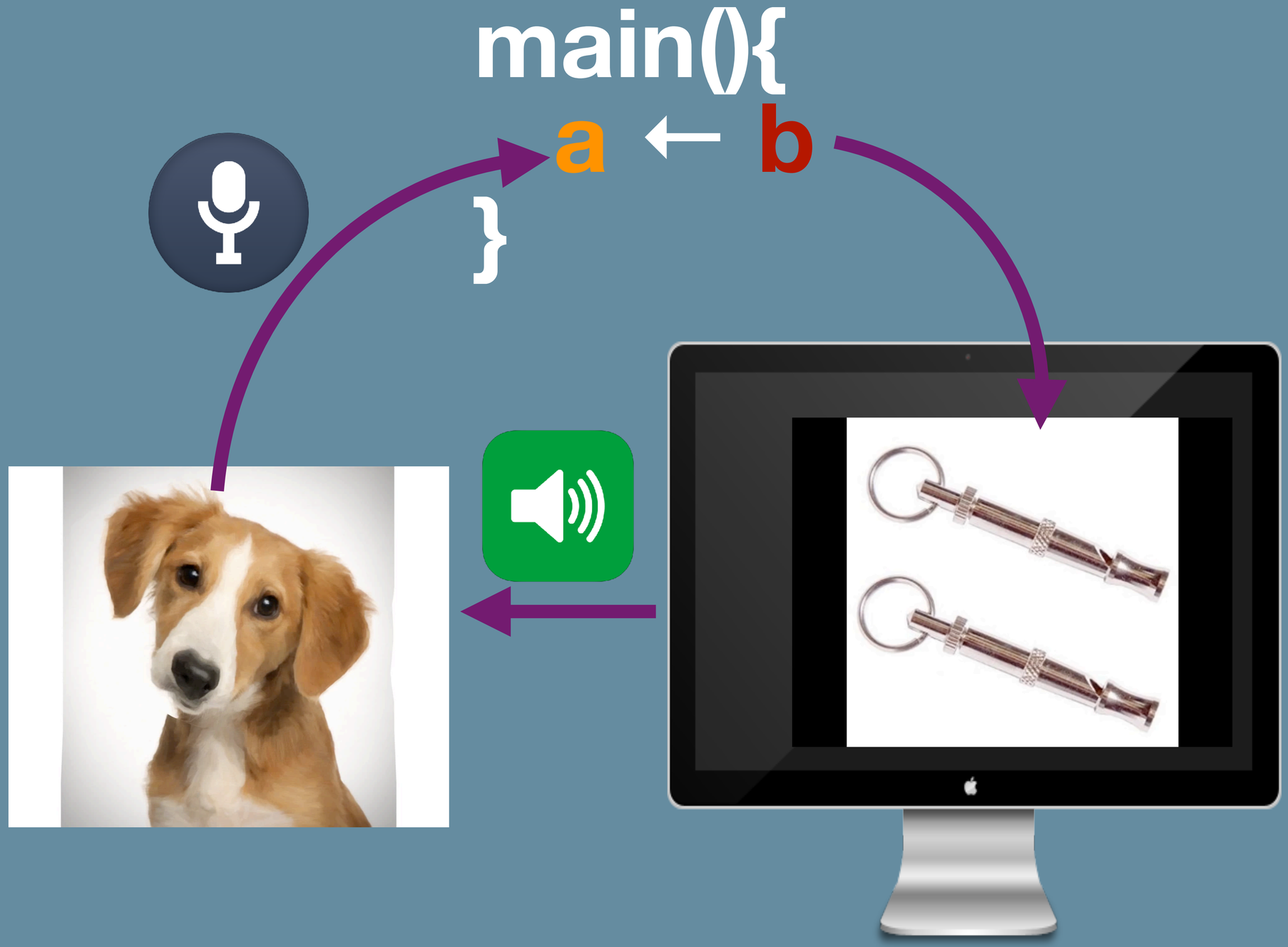

main(){
a ← b
}
listen to
hard drive
speed up/
slow down

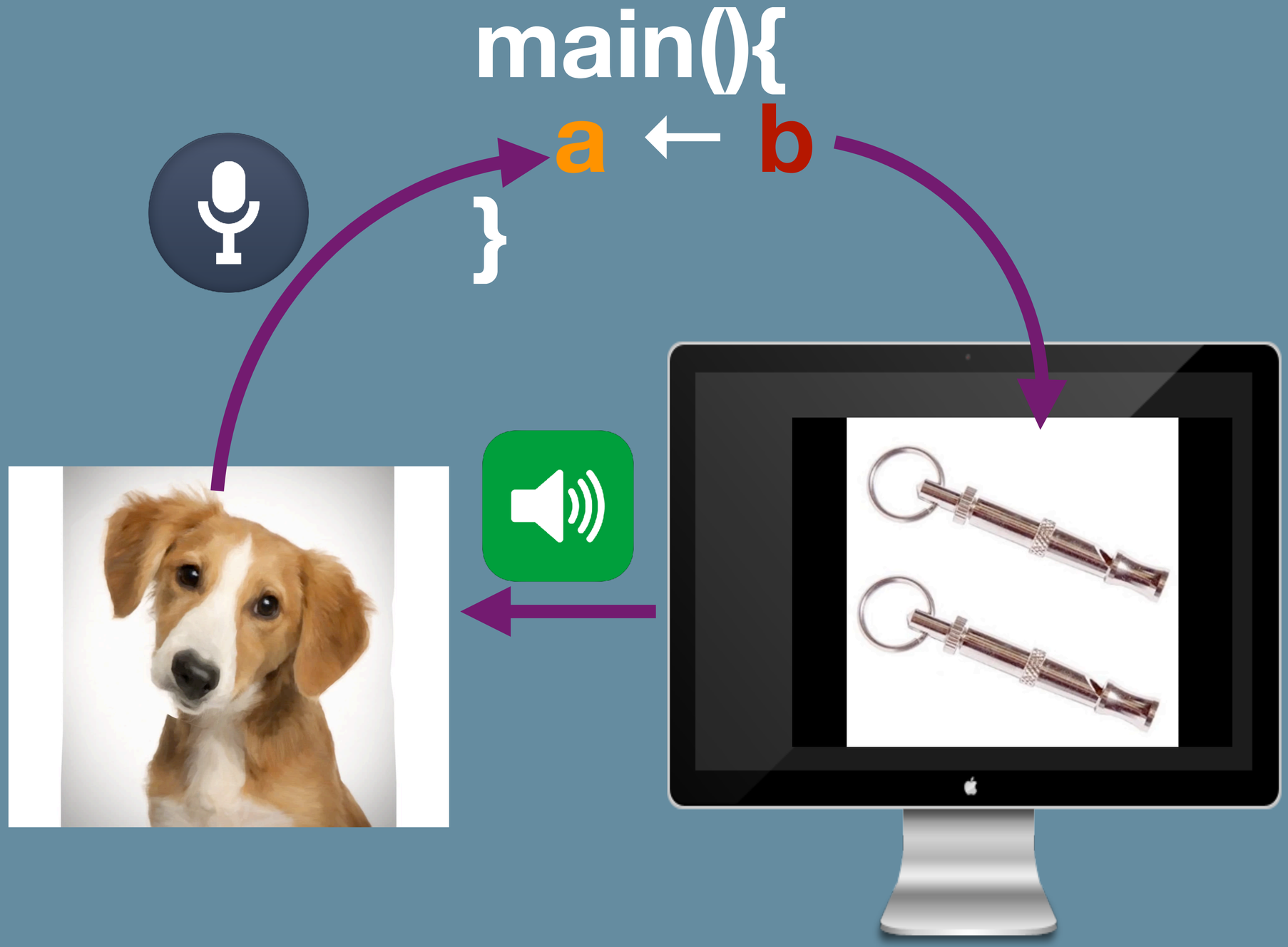

main(){
a ← b
}
read from camera
flash to screen
b

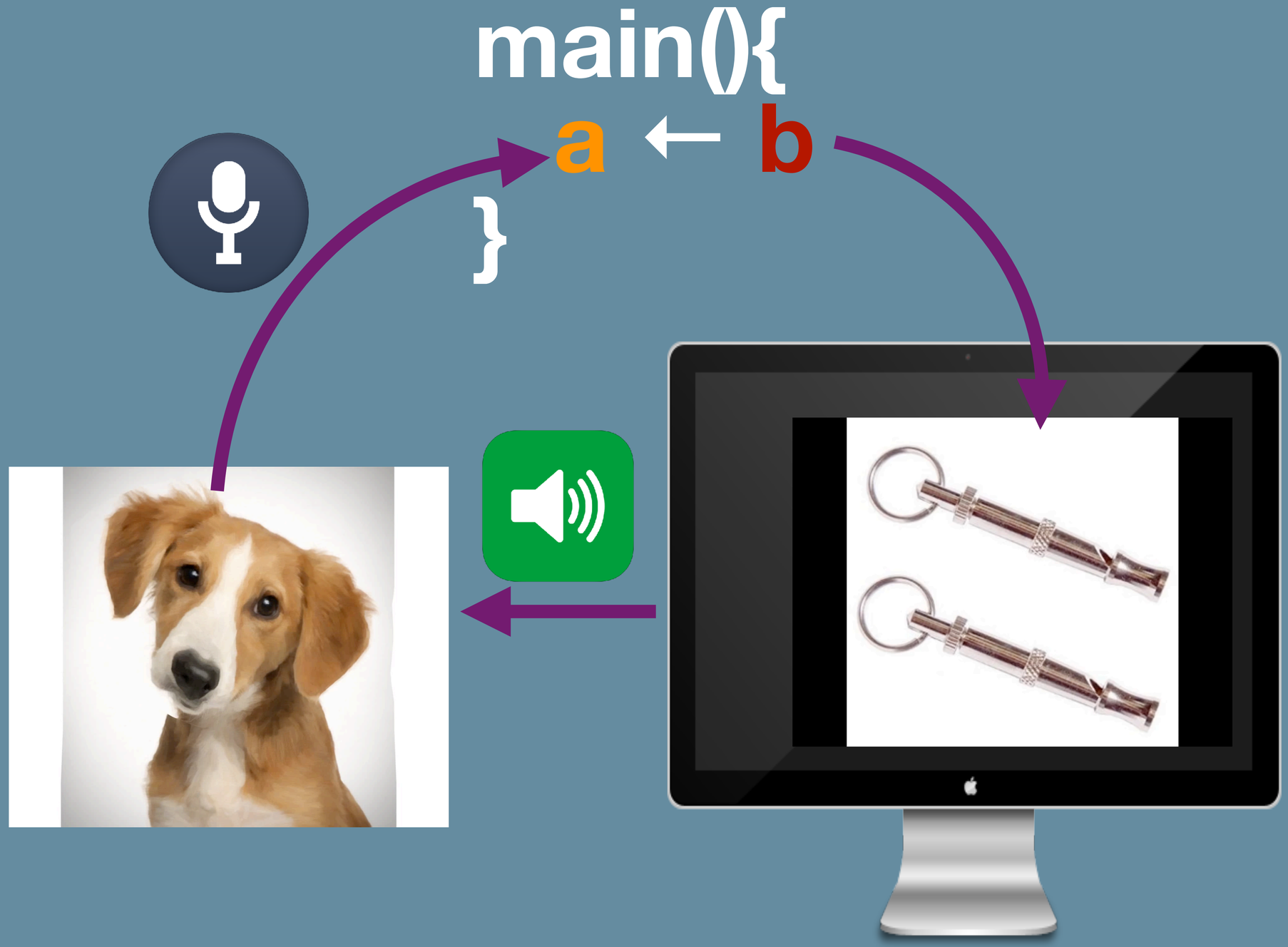

main(){
a ← b
}
read from camera
flash to screen
b

main(){
a ← b
}
read from microphone
play sound

main(){
a ← b
}
read from microphone
play sound


main(){
a ← b
}

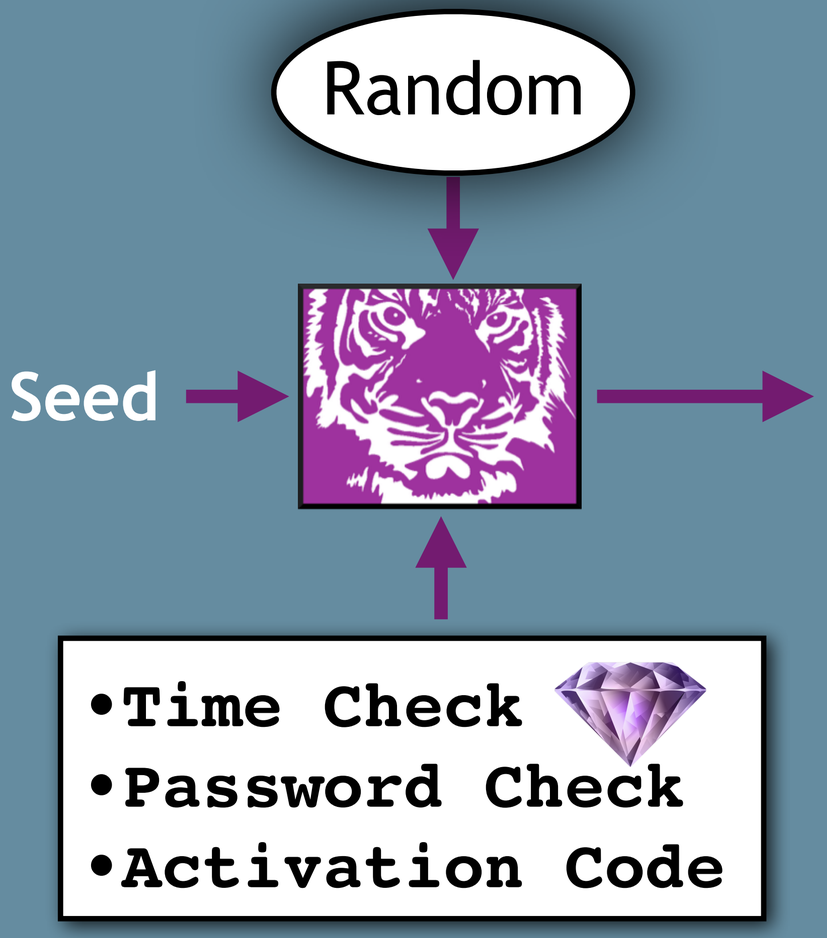

main(){
a ← b
}

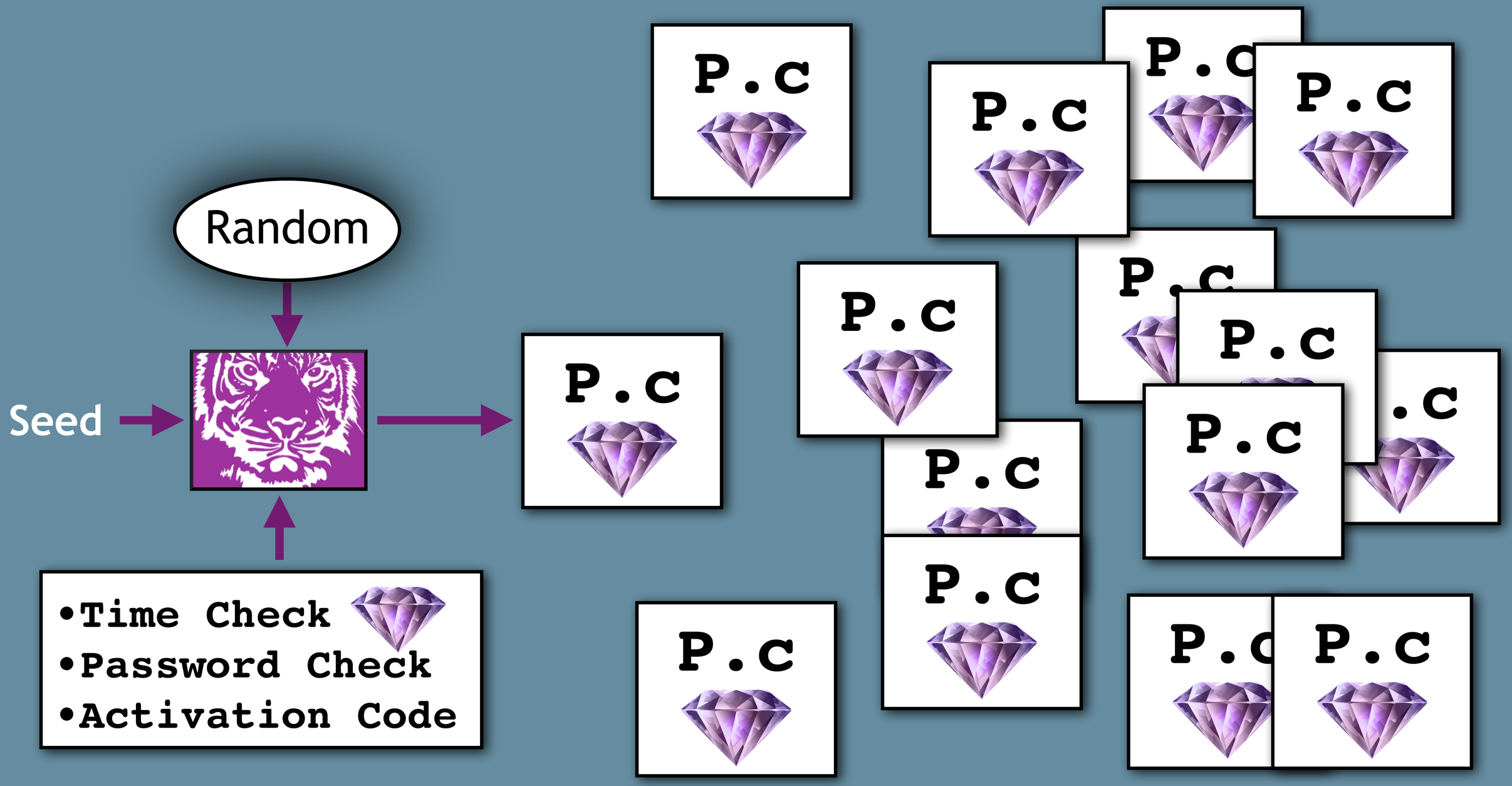

main(){
a ← b
}

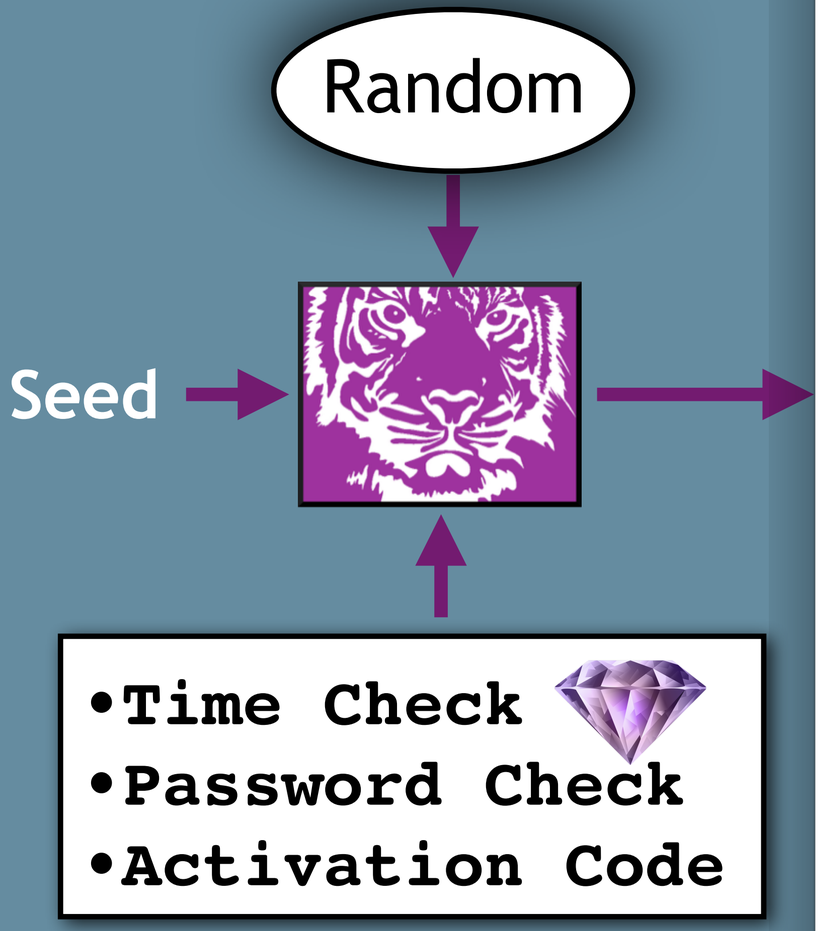
OK, maybe not

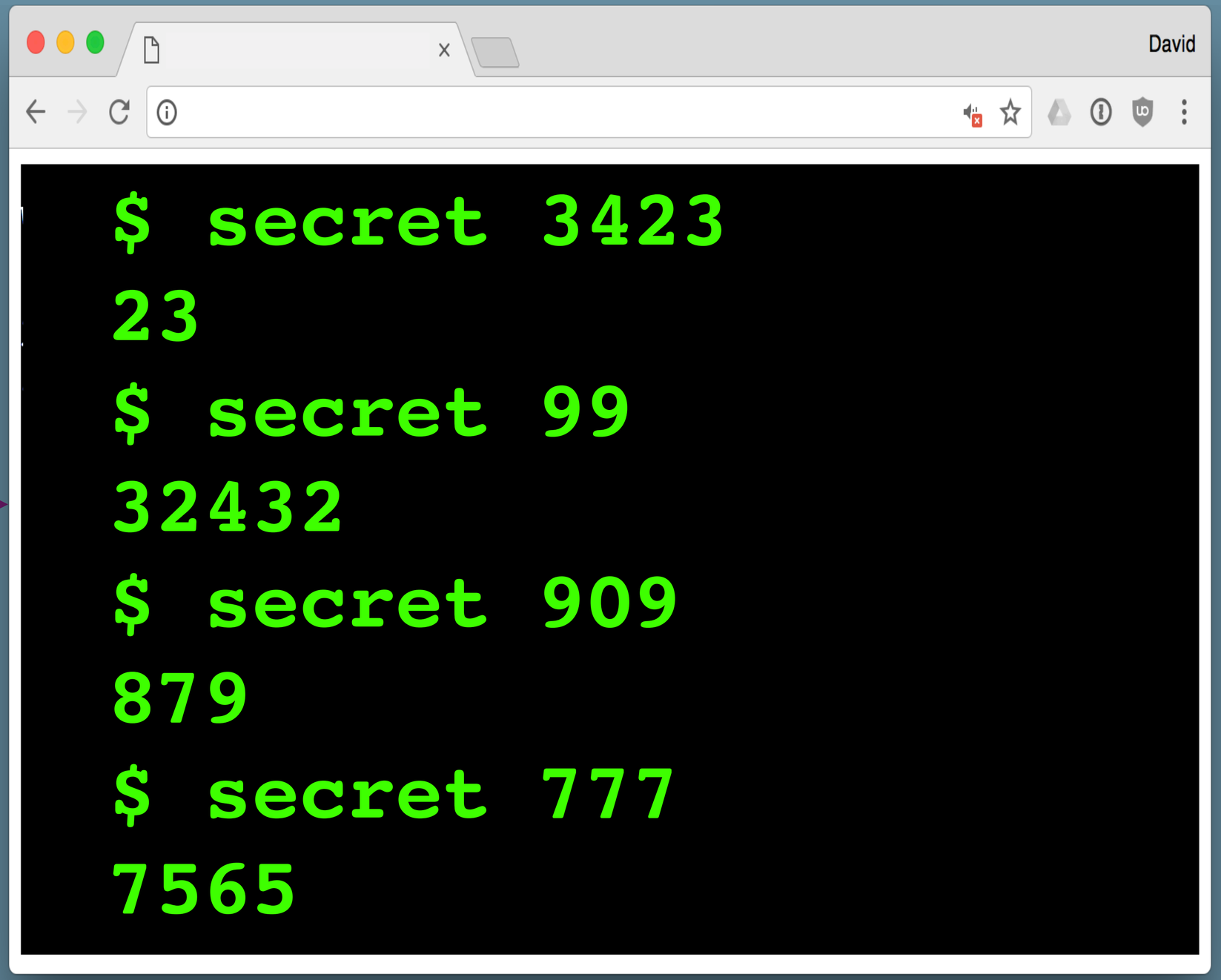
Tipress
Applications

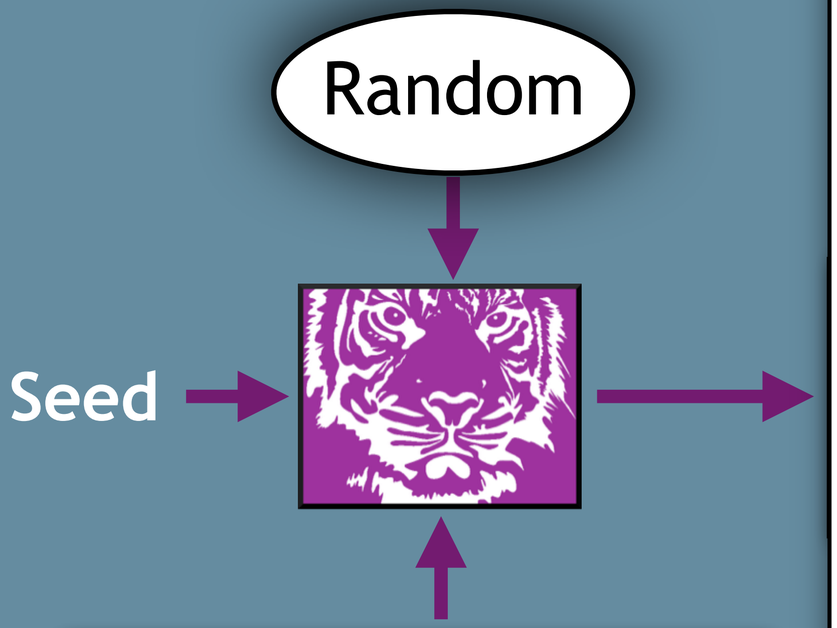
Tigress Applications


Random
Seed
•Time Check
•Password Check
•Activation Code


Random
Seed
P.c
•Time Check
•Password Check
•Activation Code


Random
Seed
•Time Check
•Password Check
•Activation Code


David
$ secret 3423
23
$ secret 99
32432
$ secret 909
879
$ secret 777
7565



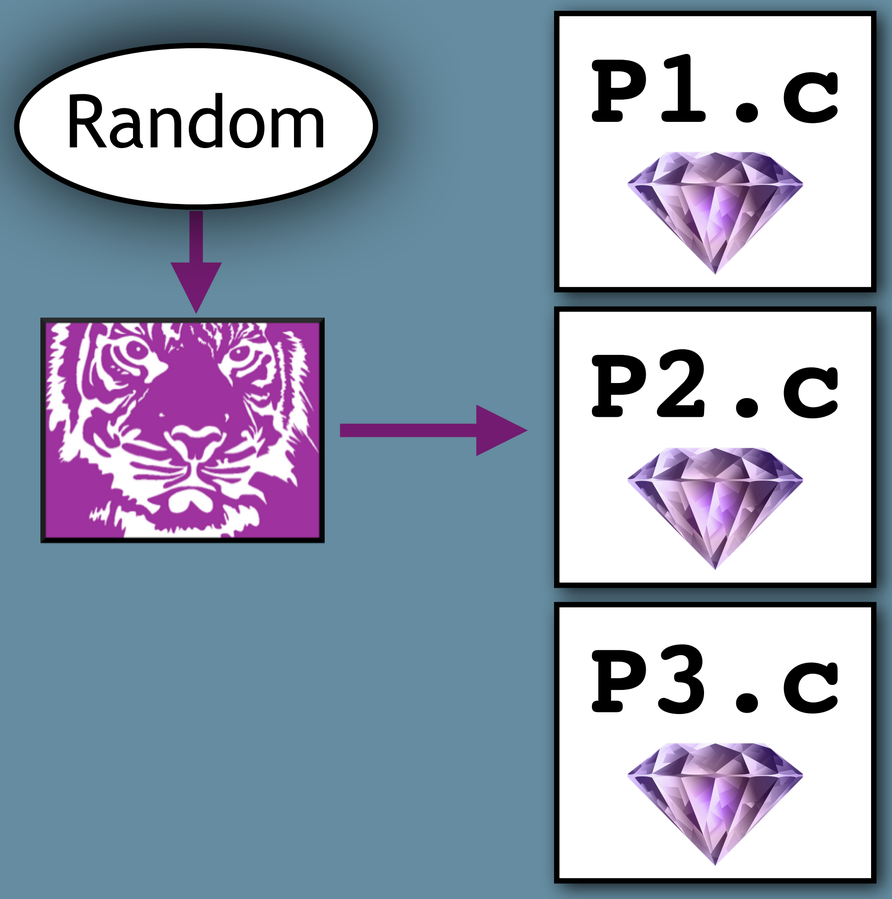
- **Time Check**
- **Password Check**
- **Activation Code**

```
void SECRET(unsigned long i[1] , unsigned long o[1] ) {
  unsigned long s[4] ;

  s[0UL] = i[0UL] + 762537946UL;
  s[1UL] = i[0UL] | ((16601096UL << (s[0UL] % 16UL | 1UL)) |
           (16601096UL >> (64 - (s[0UL] % 16UL | 1UL))));
  s[2UL] = (i[0UL] ^ 643136481UL) ^ (s[0UL] + 292656718UL);
  s[3UL] = (i[0UL] << (((s[1UL] >> 4UL) & 15UL) | 1UL)) |
           (i[0UL] >> (64 - (((s[1UL] >> 4UL) & 15UL) | 1UL)));

  unsigned long l = 0UL;
  while (l < 3UL) {
    s[1UL] |= (s[2UL] & 15UL) << 3UL;
    s[l + 1UL] = s[l]; l += 2UL;
  }
  if ((s[0UL] | s[1UL]) > (s[2UL] | s[3UL])) {
    s[3UL] |= (s[1UL] & 31UL) << 3UL;
  } else {
    s[2UL] = s[0UL]; s[3UL] |= (s[2UL] & 15UL) << 3UL;
  }
  s[0UL] = s[2UL];

  o[0UL] = (s[0UL] << (s[1UL] % 8UL | 1UL)) <<
           ((((s[2UL] << (s[3UL] % 8UL | 1UL))
           >> 1UL) & 7UL) | 1UL);
}
```

Teaching Reverse Engineering

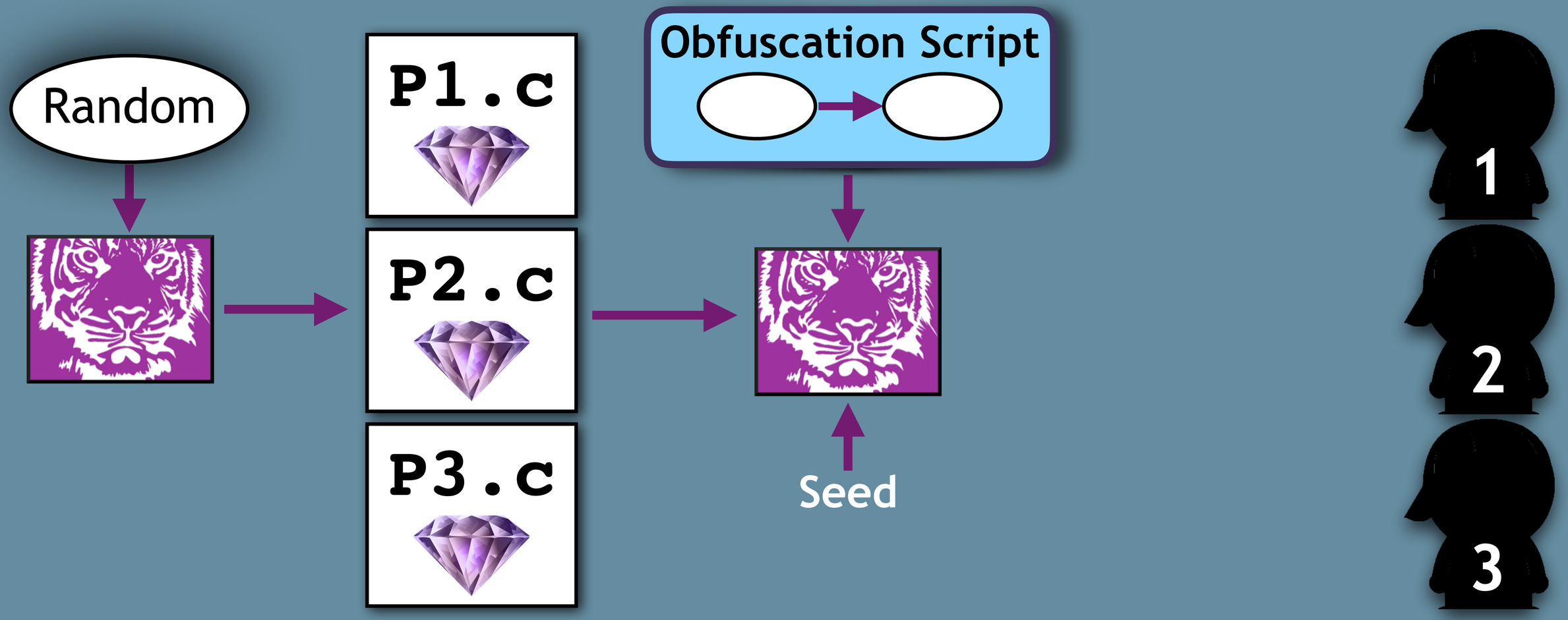
Claire Taylor, http://RevEngE.cs.arizona.edu

Teaching Reverse Engineering



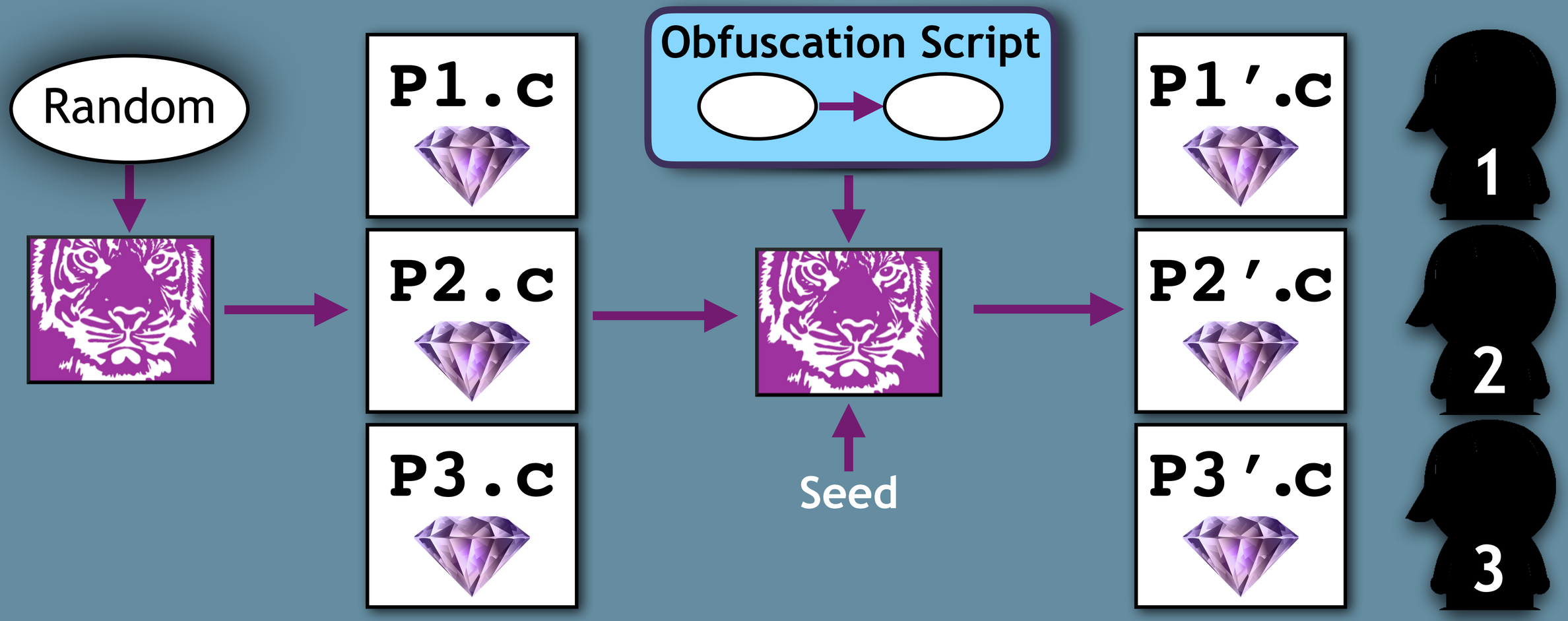
Claire Taylor, http://RevEngE.cs.arizona.edu

Teaching Reverse Engineering



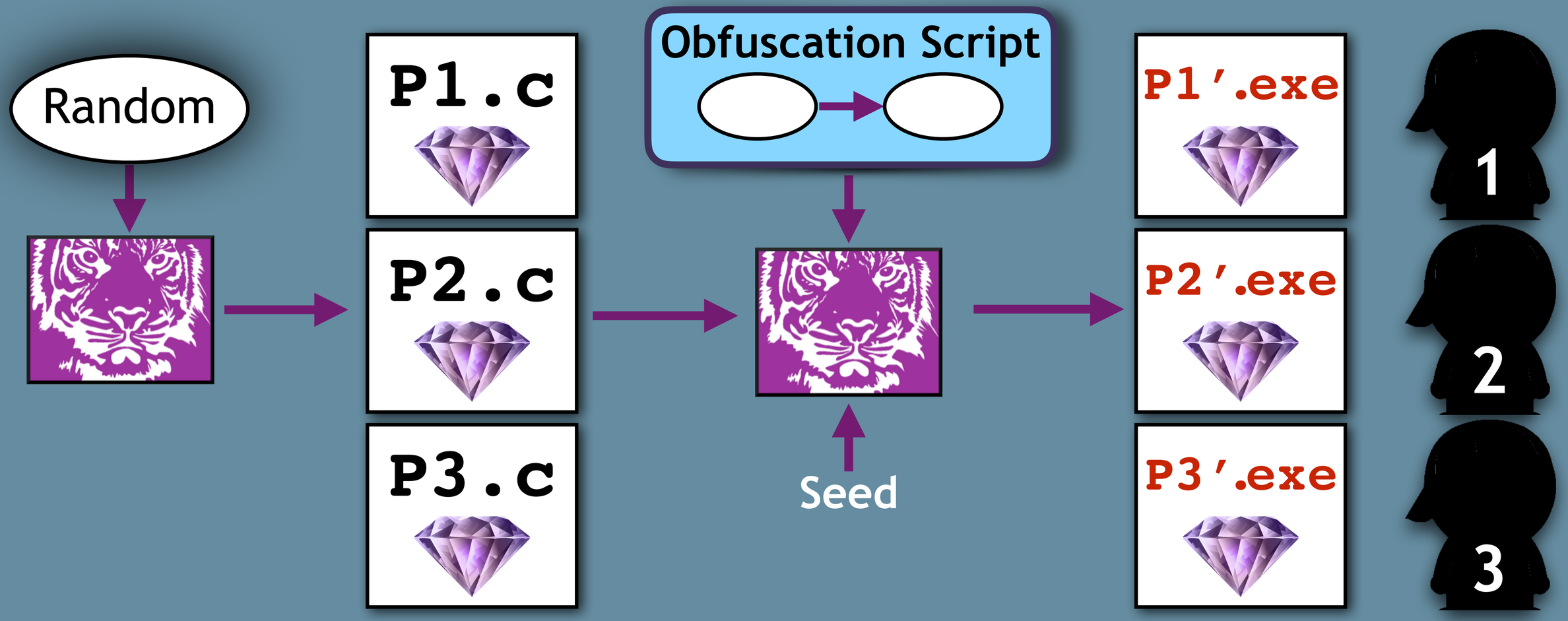
Claire Taylor, http://RevEngE.cs.arizona.edu

Teaching Reverse Engineering



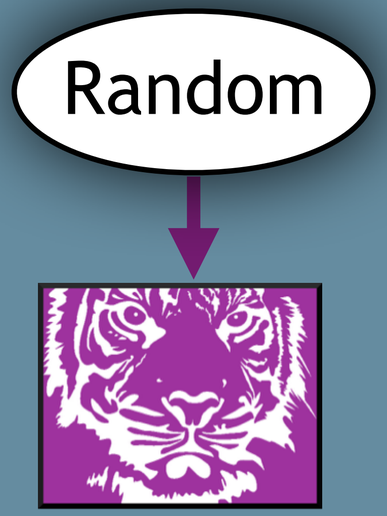
Claire Taylor, http://RevEngE.cs.arizona.edu

Teaching Reverse Engineering



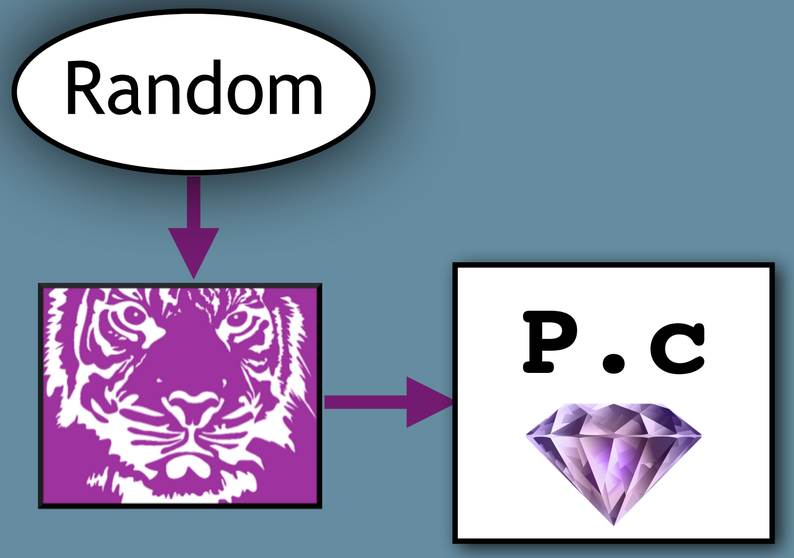
Claire Taylor, http://RevEngE.cs.arizona.edu

Monitoring Reverse Engineers



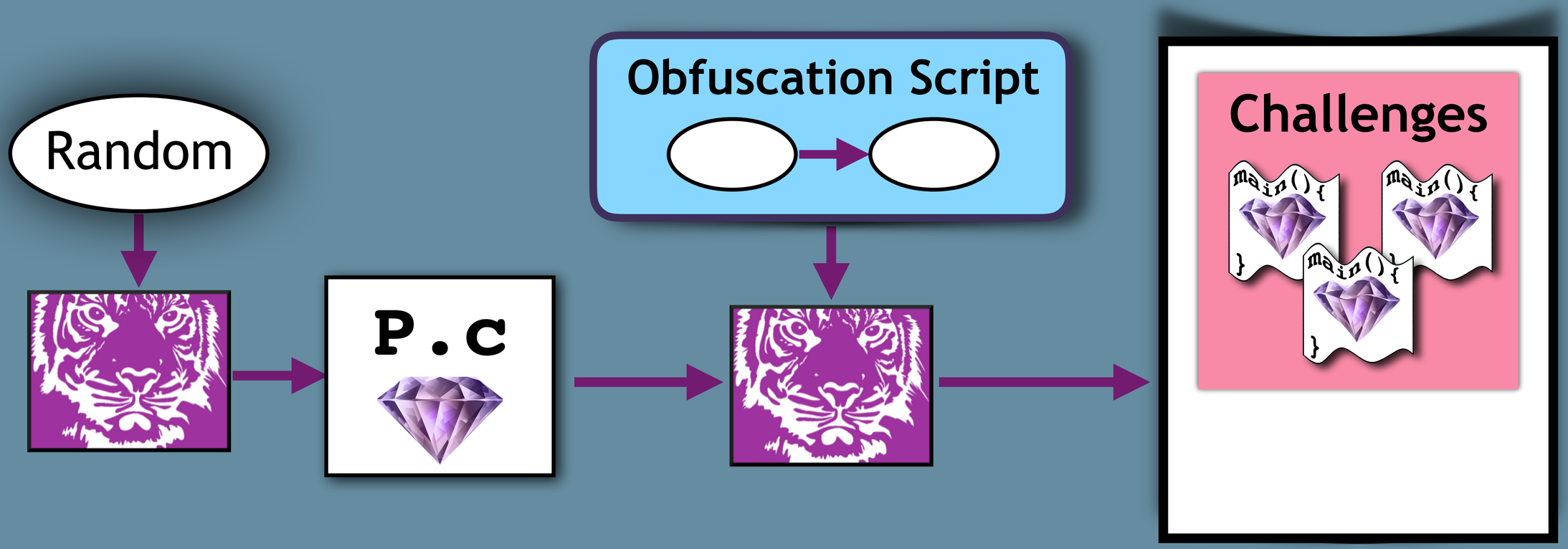
Claire Taylor, http://RevEngE.cs.arizona.edu

Monitoring Reverse Engineers



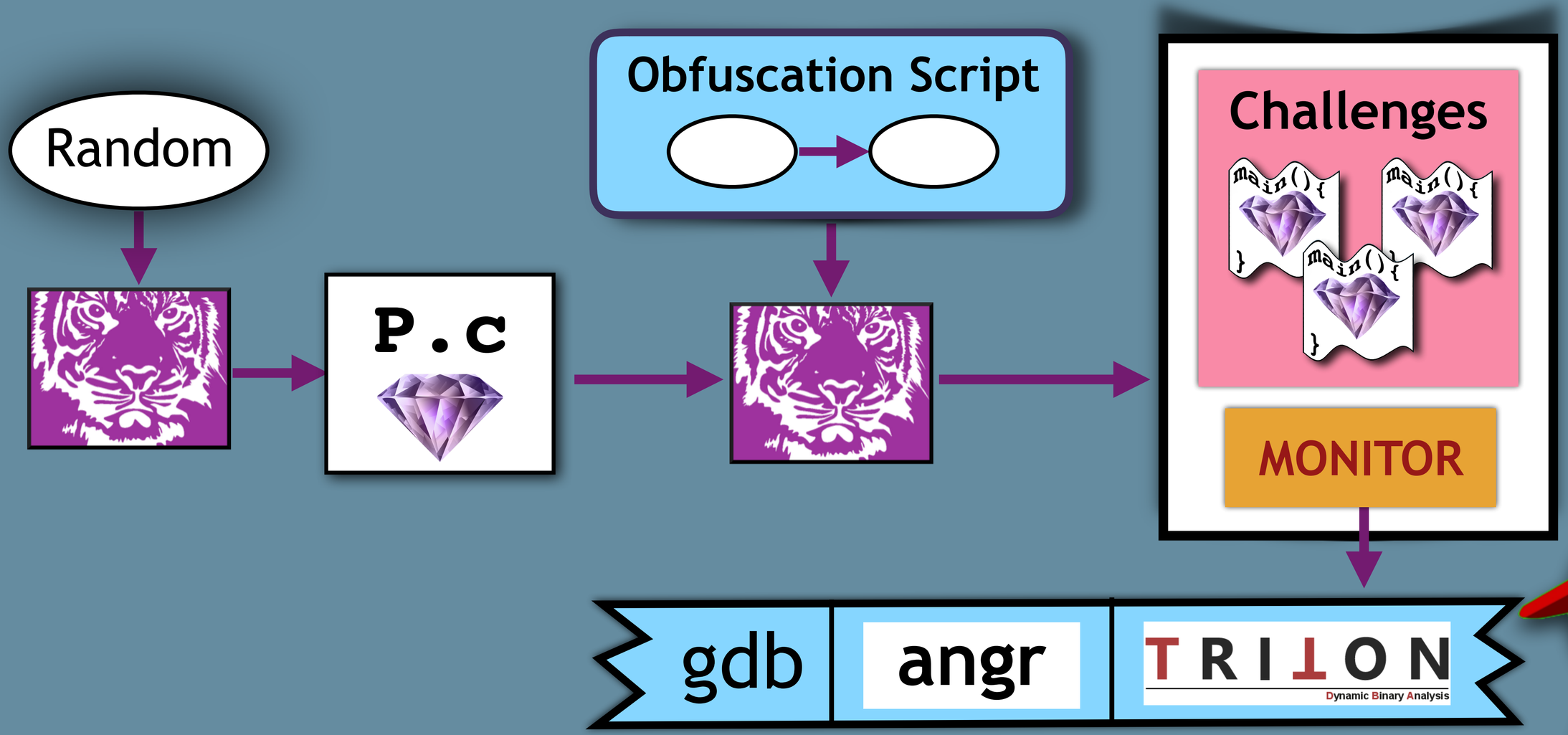
Claire Taylor, http://RevEngE.cs.arizona.edu

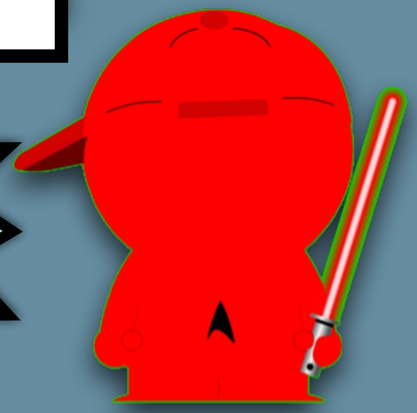
Monitoring Reverse Engineers



Claire Taylor, http://RevEngE.cs.arizona.edu

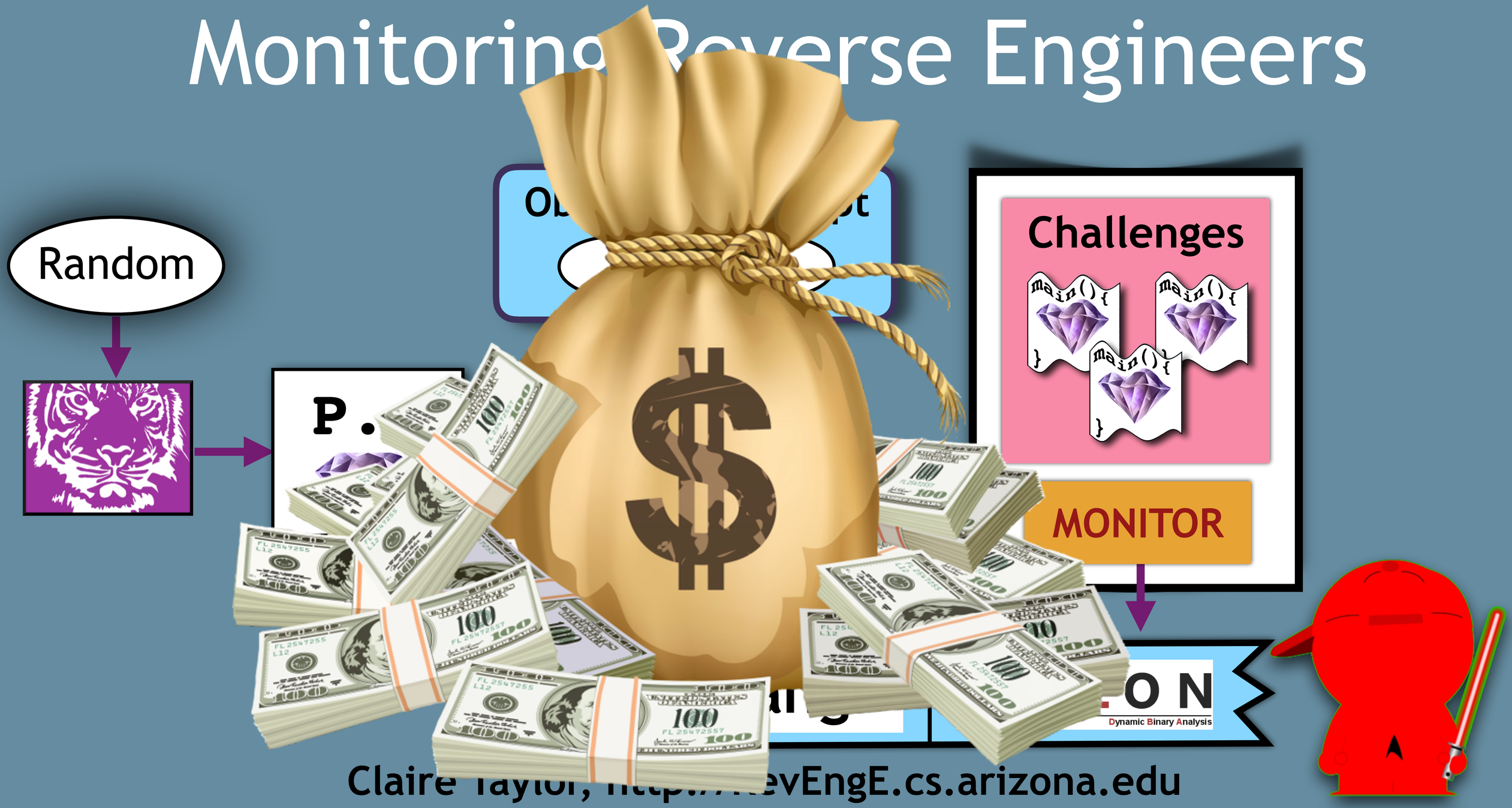
Monitoring Reverse Engineers





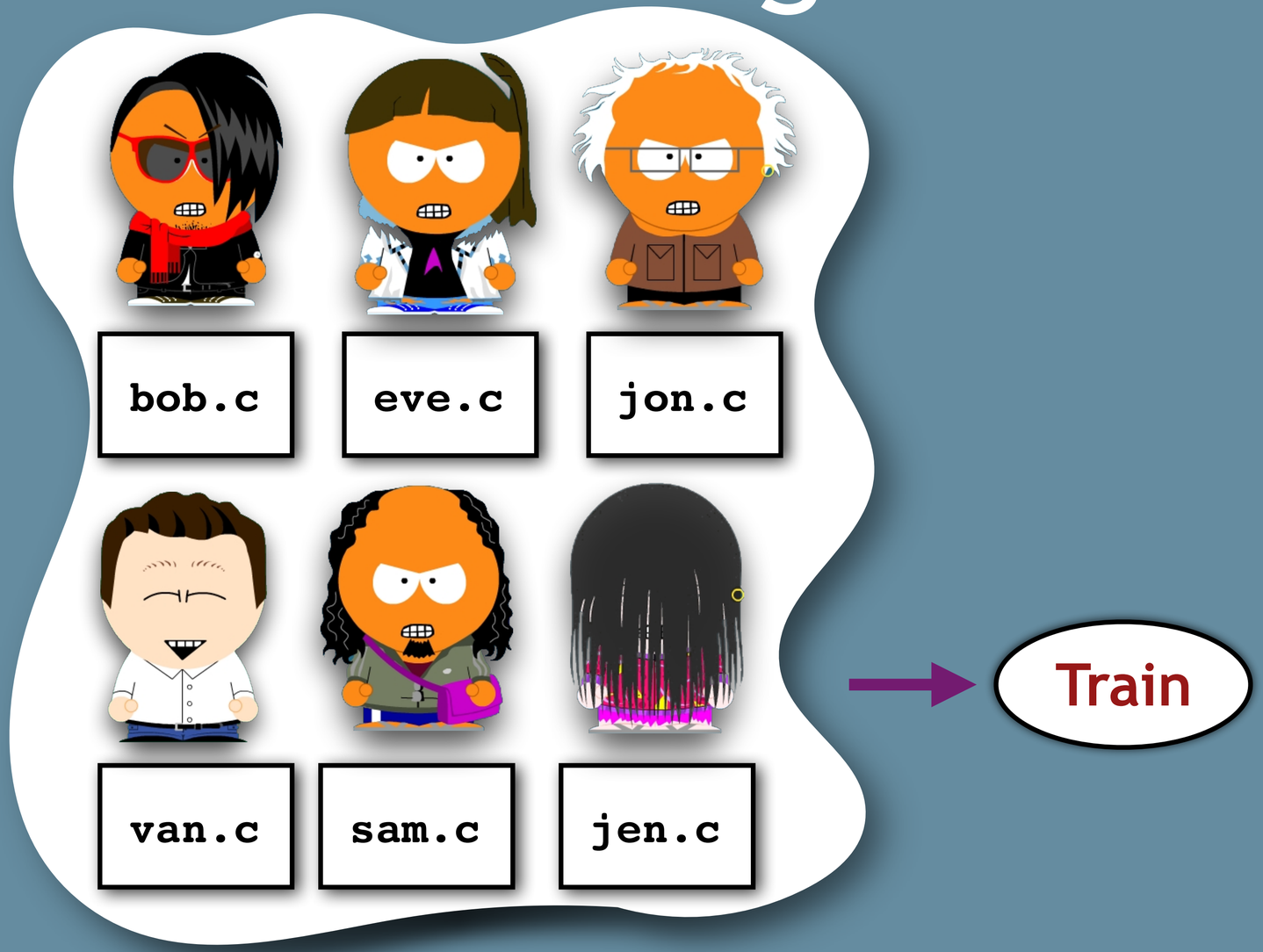
Claire Taylor, http://RevEngE.cs.arizona.edu

Monitoring Reverse Engineers



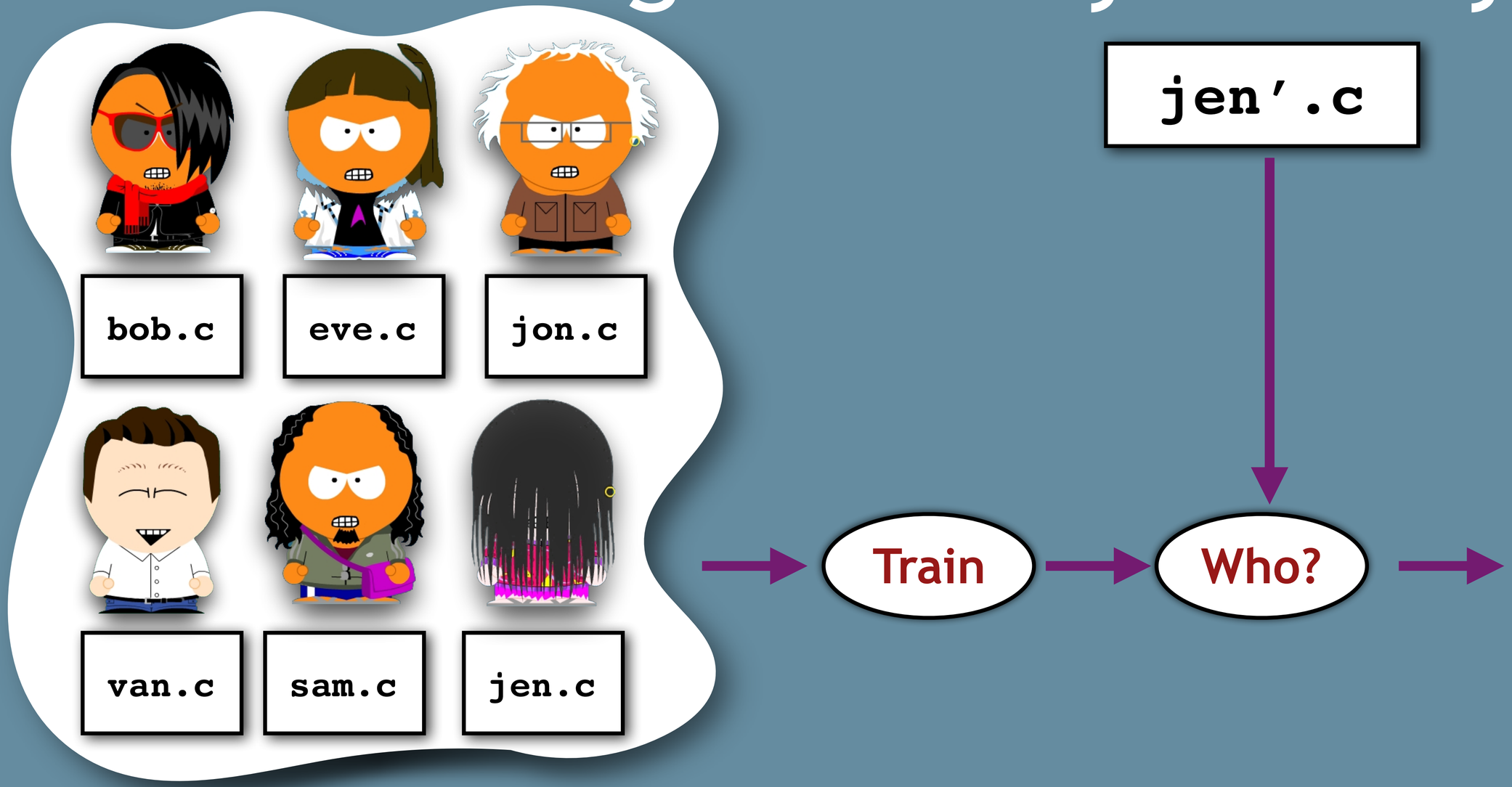
Claire Taylor, http://RevEngE.cs.arizona.edu

Testing Code Stylometry



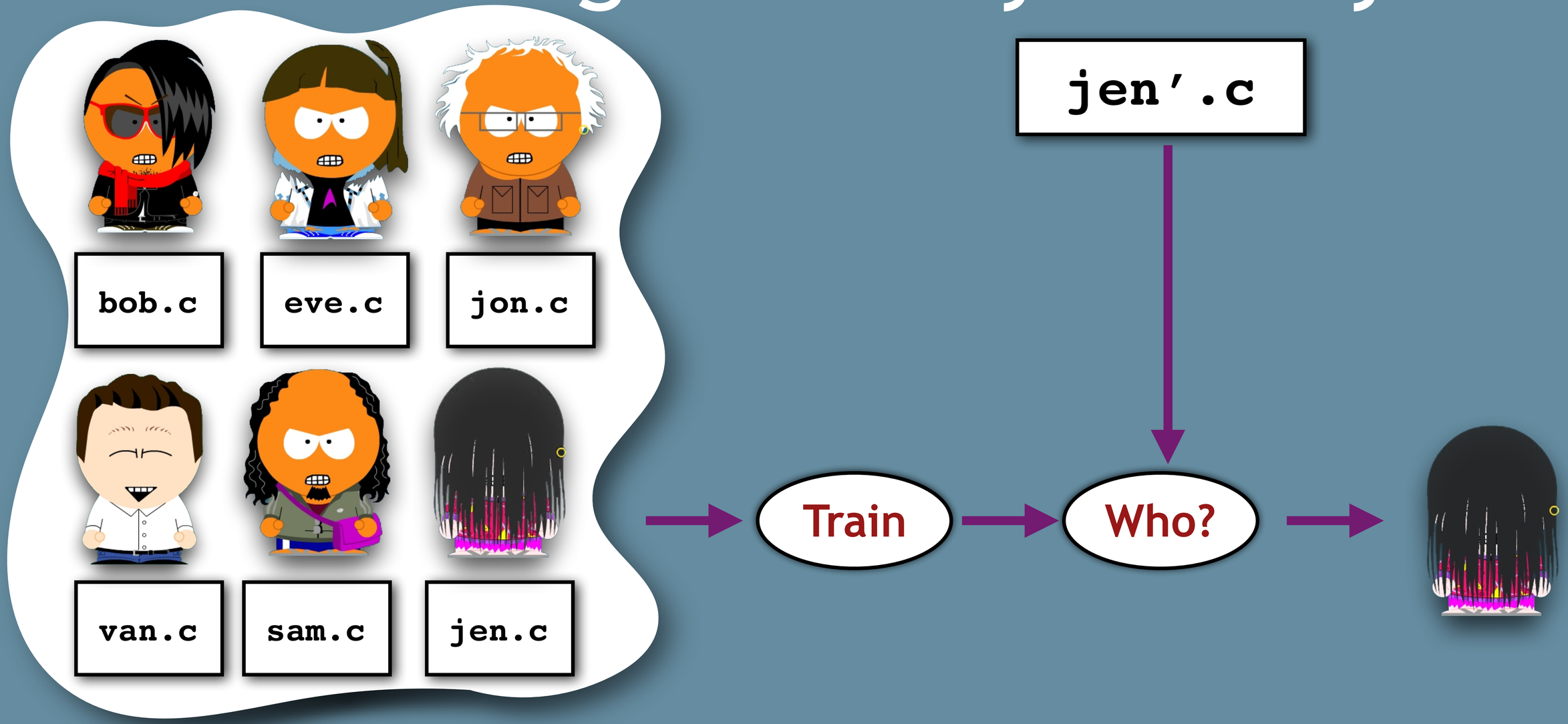
Caliskan-Islam, et al., De-anonymizing Programmers via Code Stylometry

Testing Code Stylometry



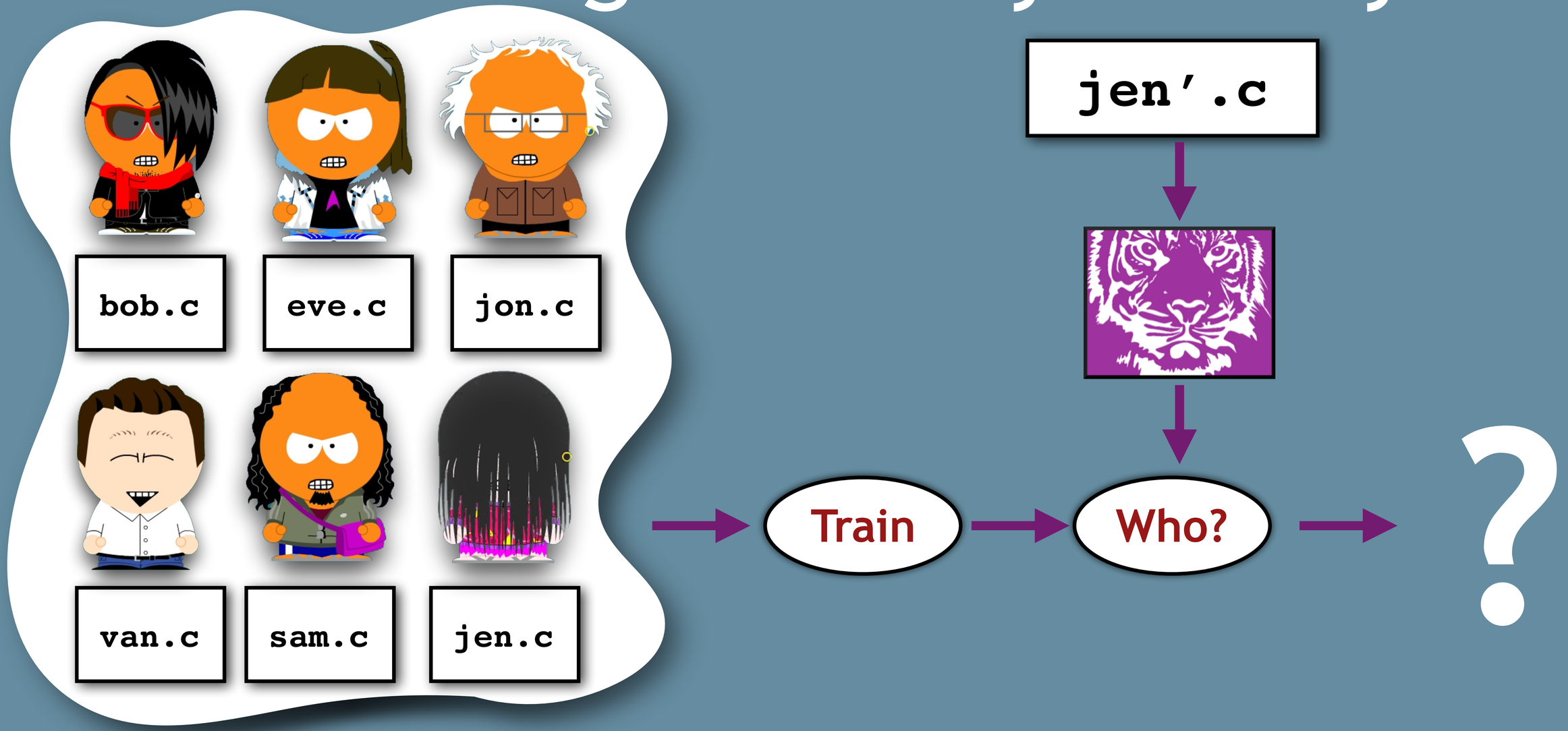
Caliskan-Islam, et al., De-anonymizing Programmers via Code Stylometry

Testing Code Stylometry



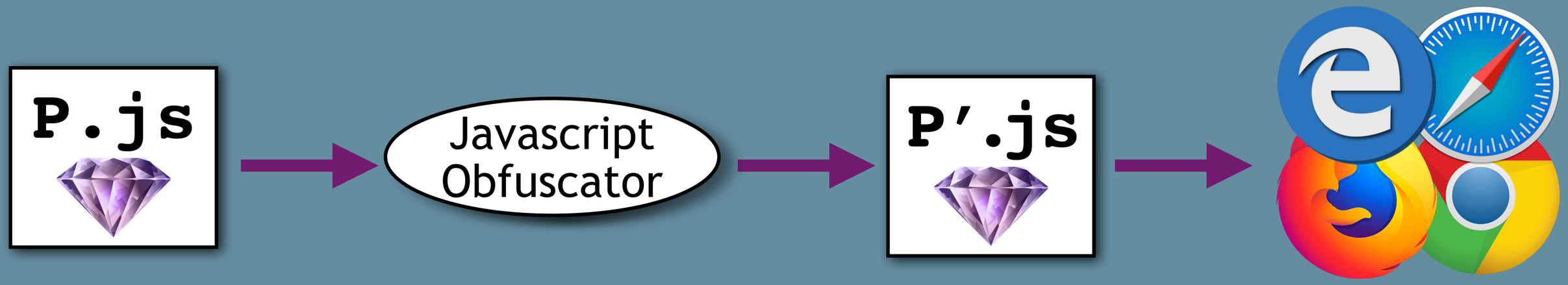
Caliskan-Islam, et al., De-anonymizing Programmers via Code Stylometry

Testing Code Stylometry

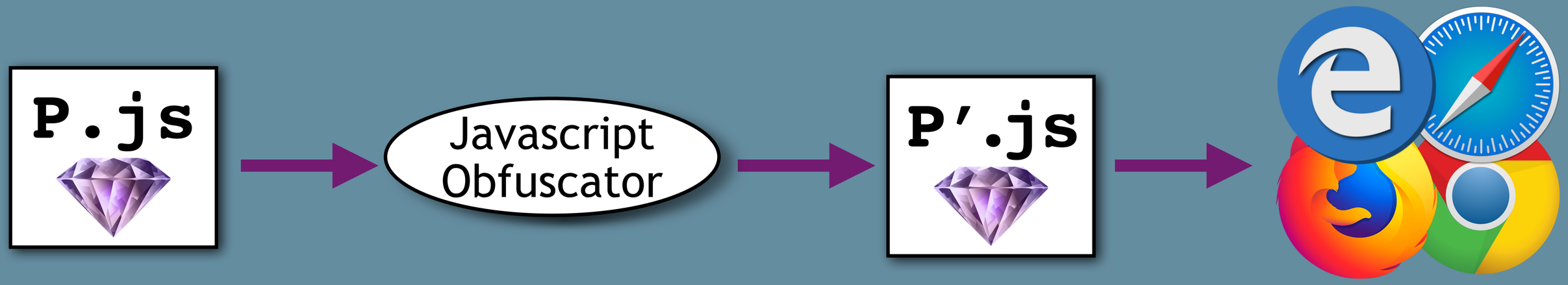


Caliskan-Islam, et al., De-anonymizing Programmers via Code Stylometry

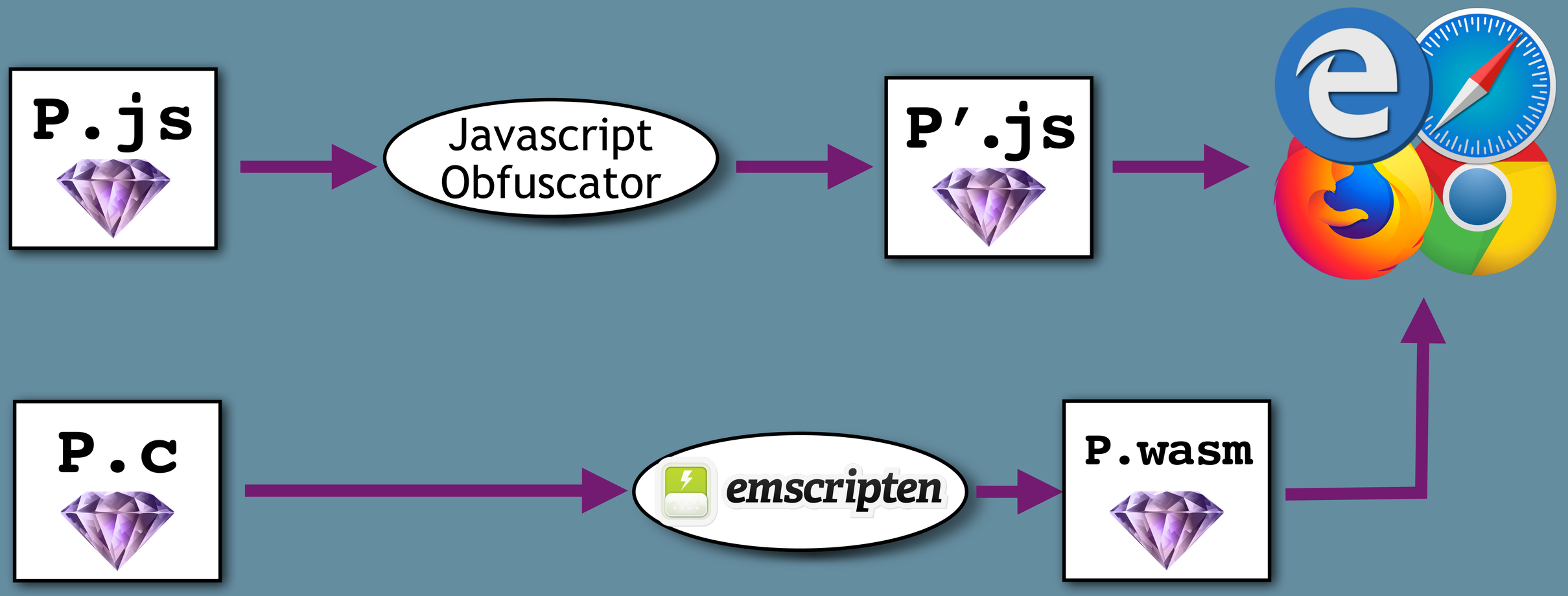
WebAssembly Obfuscation



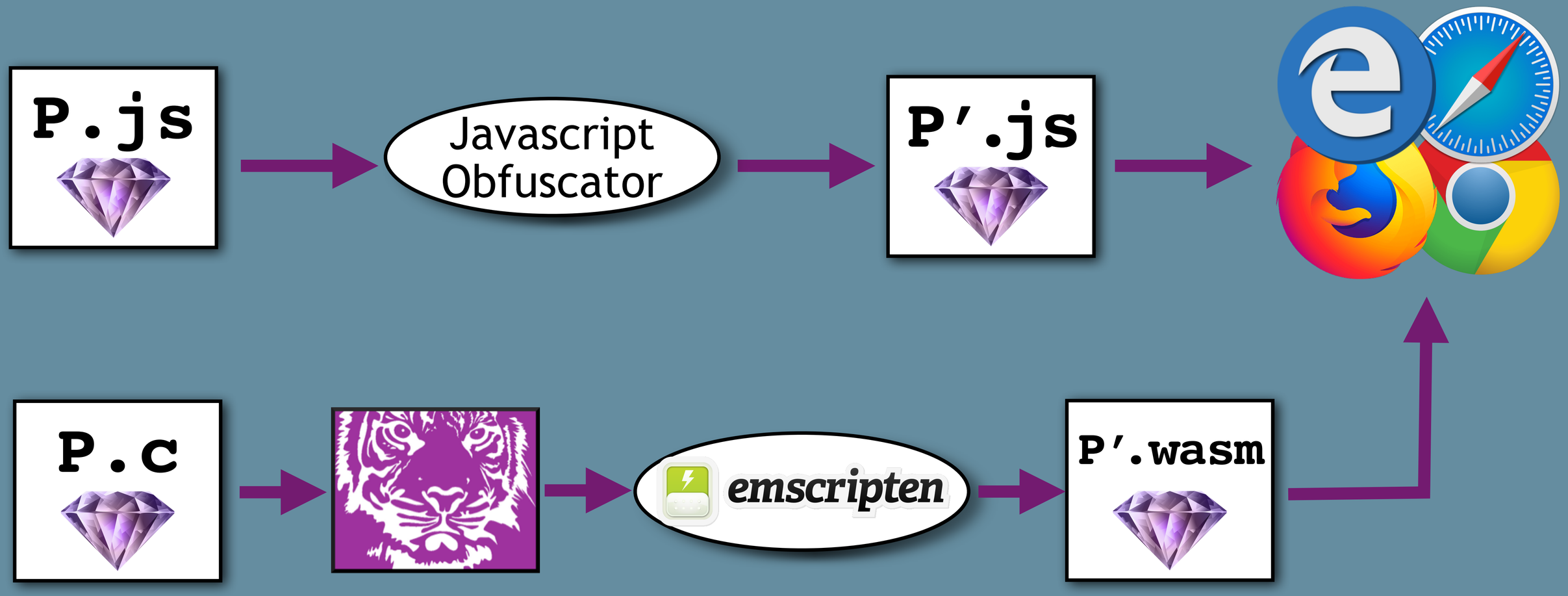
WebAssembly Obfuscation



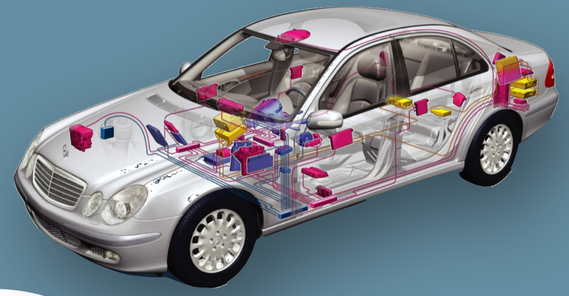
WebAssembly Obfuscation



WebAssembly Obfuscation

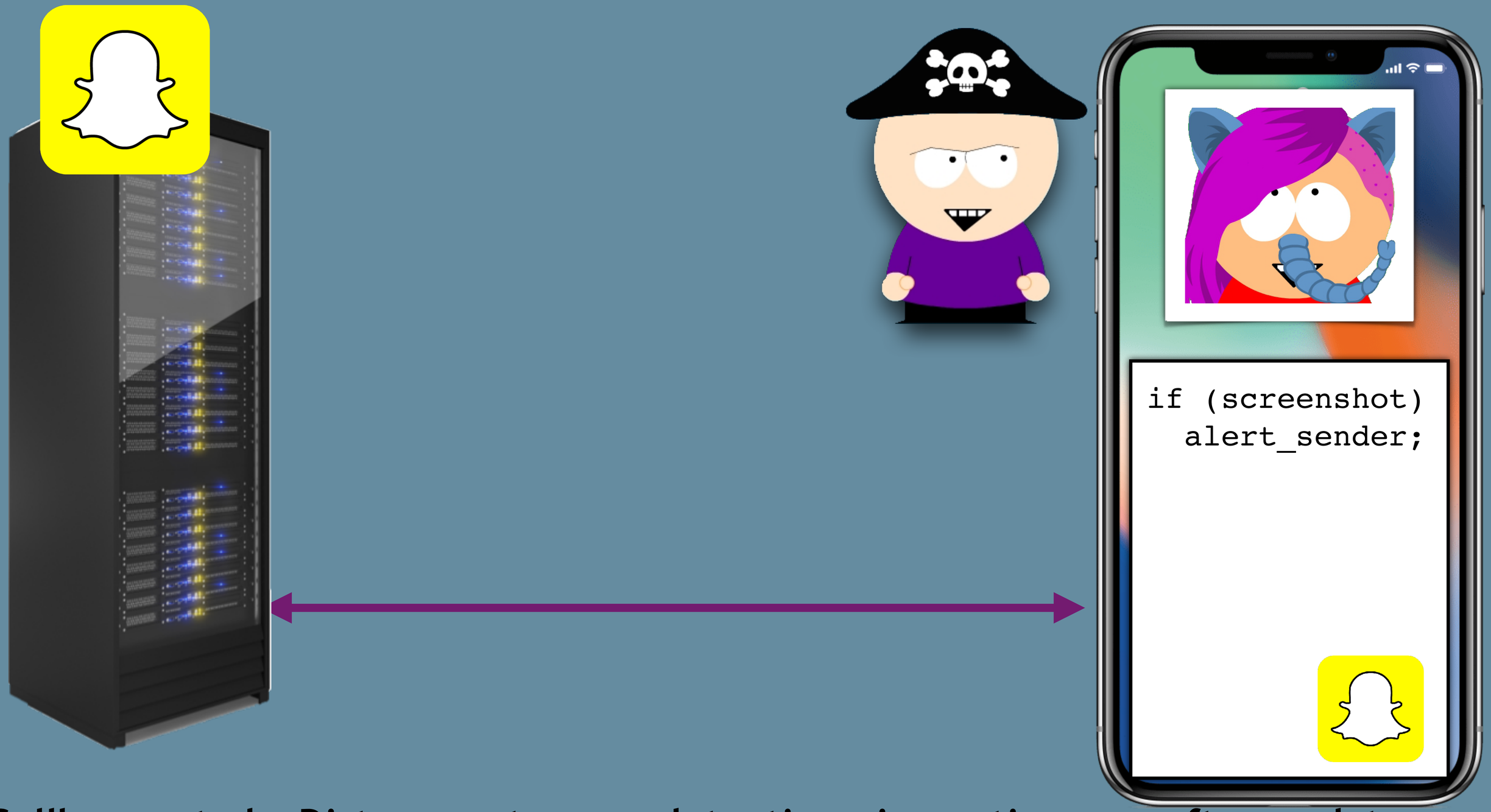


Commercial Efforts

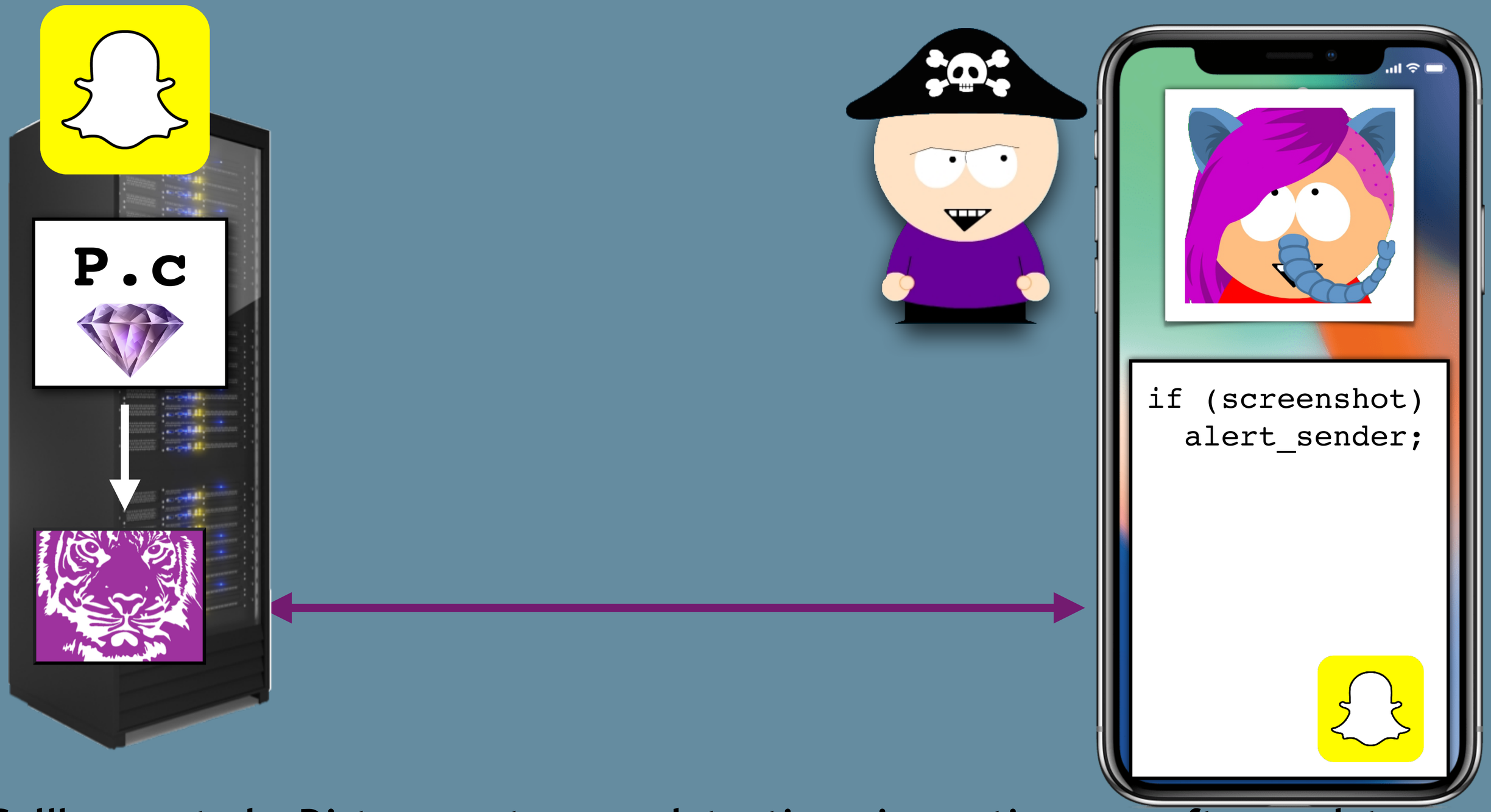


?

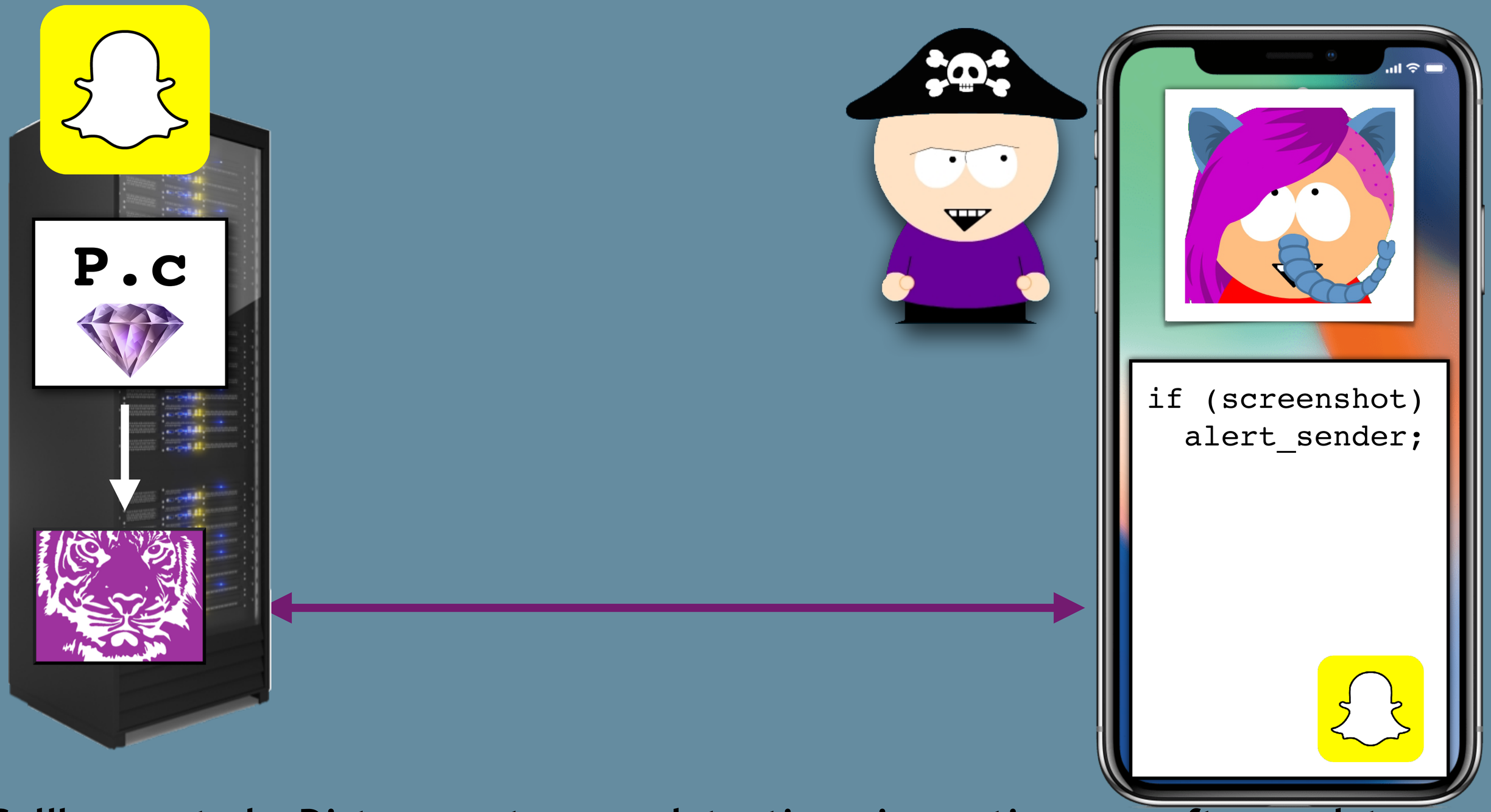
?



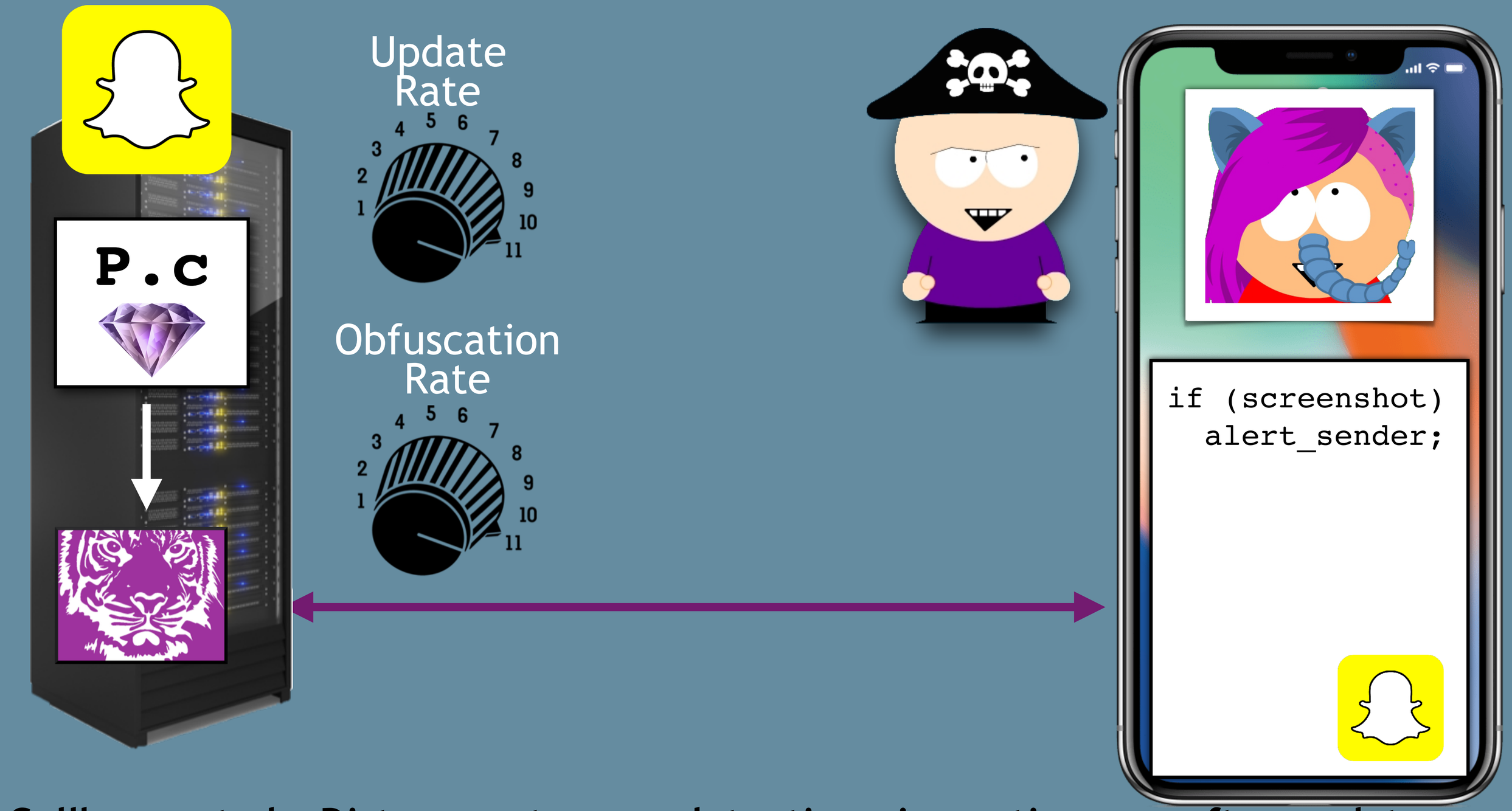
Collberg, et al., Distr. app. tamper detection via continuous softw. updates



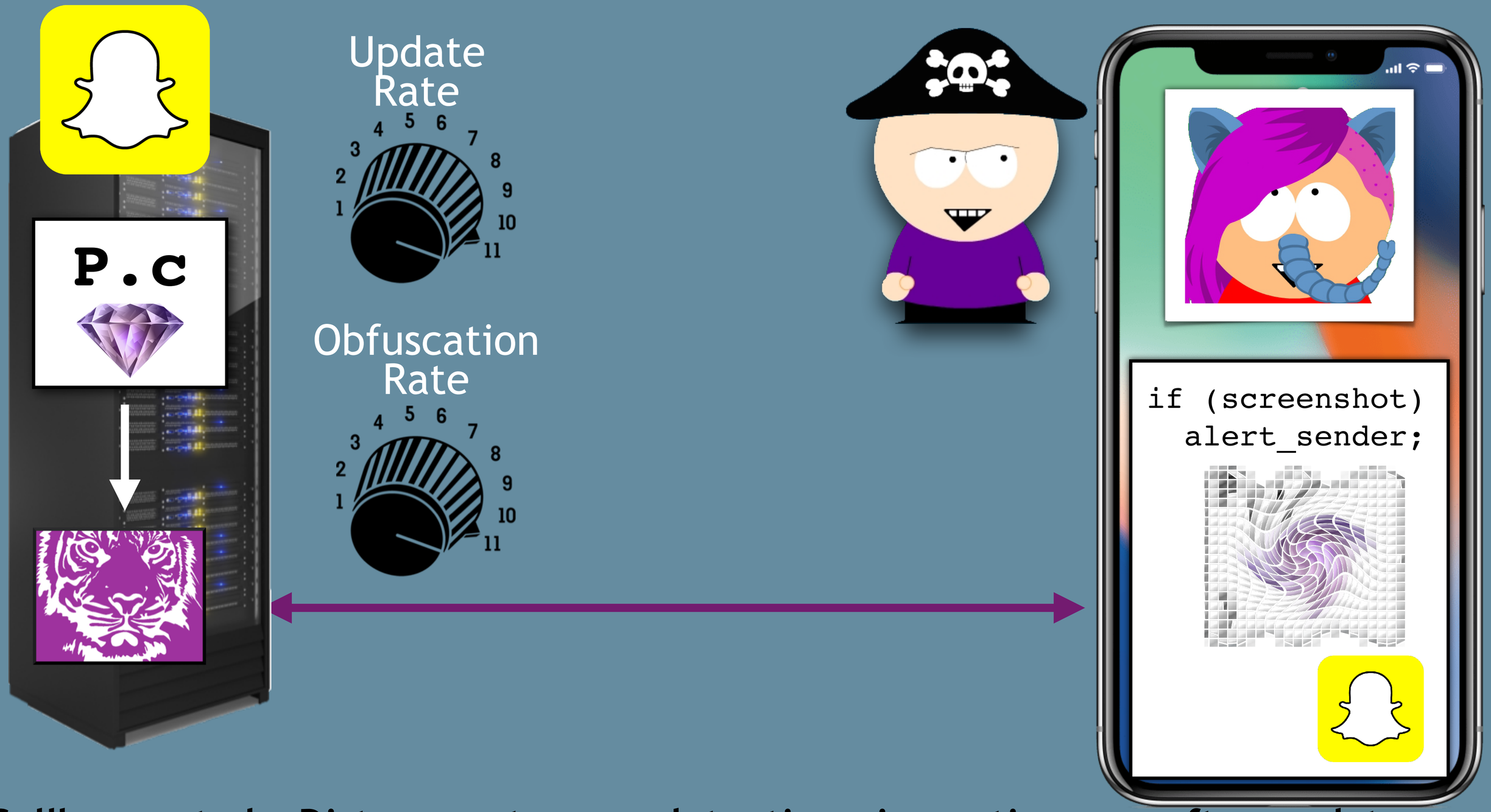
Collberg, et al., Distr. app. tamper detection via continuous softw. updates



Collberg, et al., Distr. app. tamper detection via continuous softw. updates



Collberg, et al., Distr. app. tamper detection via continuous softw. updates



Collberg, et al., Distr. app. tamper detection via continuous softw. updates

Evaluating (De-)Obfuscation

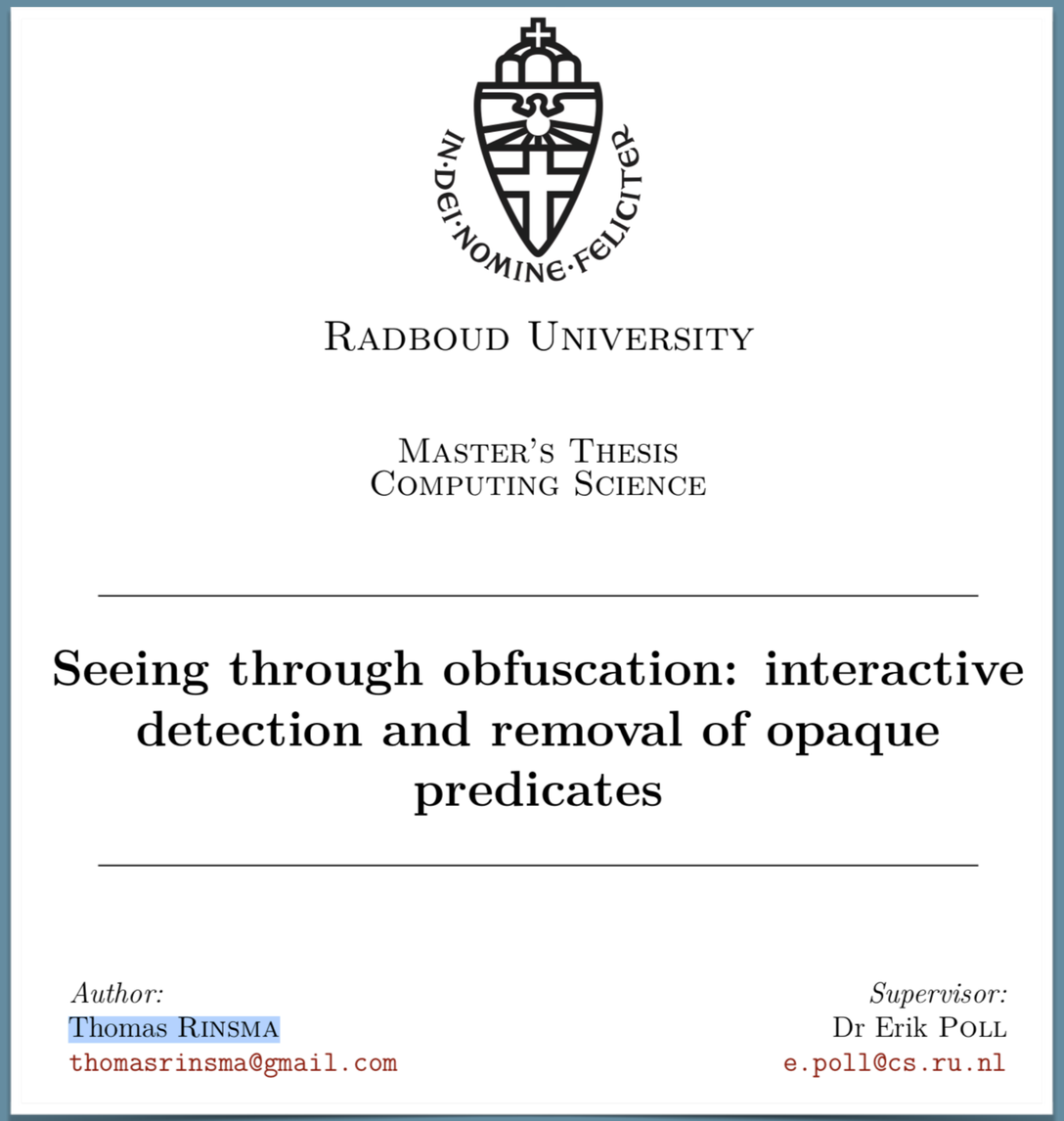
Symbolic deobfuscation:
from virtualized code back to the original*

Jonathan Salwan[1], Sébastien Bardin[2], and Marie-Laure Potet[3]

[1] Quarkslab, Paris, France
[2] CEA, LIST, Univ. Paris-Saclay, France
[3] Univ. Grenoble Alpes, F-38000 Grenoble, France
jsalwan@quarkslab.com, sebastien.bardin@cea.fr,
marie-laure.potet@univ-grenoble-alpes.fr

Program	Performance
Tigress Challenge 3.4	2.2 hours, 170GB RAM

Evaluating (De-)Obfuscation



Radboud University

Master's Thesis
Computing Science

Seeing through obfuscation: interactive detection and removal of opaque predicates

Author:
Thomas Rinsma
thomasrinsma@gmail.com

Supervisor:
Dr Erik Poll
e.poll@cs.ru.nl

Thank You!

collberg.cs.arizona.edu

tigress.cs.arizona.edu

RevEngE.cs.arizona.edu

Supported by NSF CNS-1525820

Thanks to my collaborators:
Taylor, Debray, Stephens, Yadegari, Scheidegger

Thank You!

collberg.cs.arizona.edu

tigress.cs.arizona.edu

RevEngE.cs.arizona.edu

Supported by NSF CNS-1525820

Thanks to my collaborators:
Taylor, Debray, Stephens, Yadegari, Scheidegger