Scaling Verification of Security Protocol Implementations

A Symbiotic Combination of Program Verification & Static Analyses

Linard Arquint

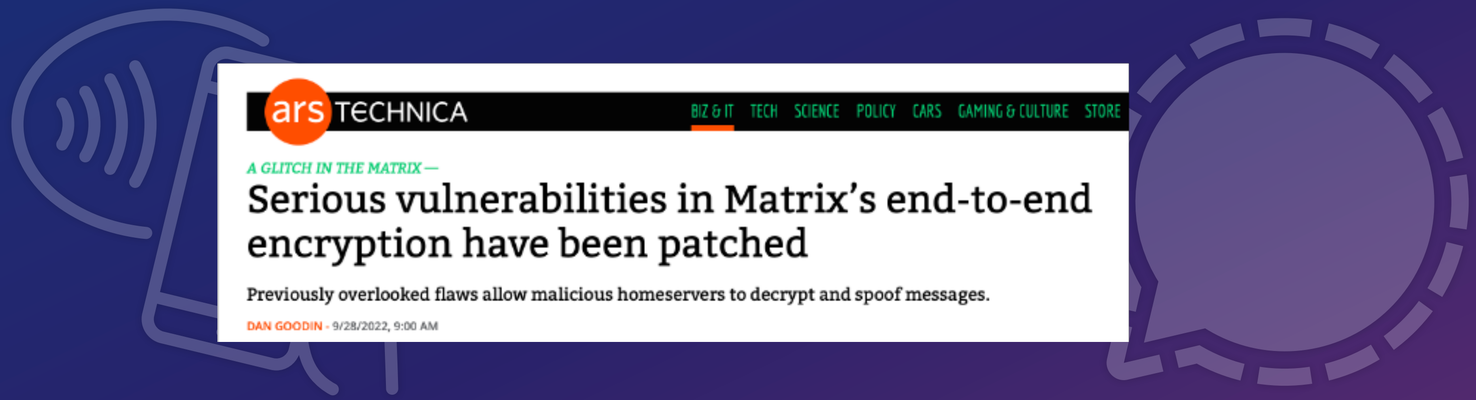
NUS PL Seminar — May 8, 2026

NUS
National University
of Singapore

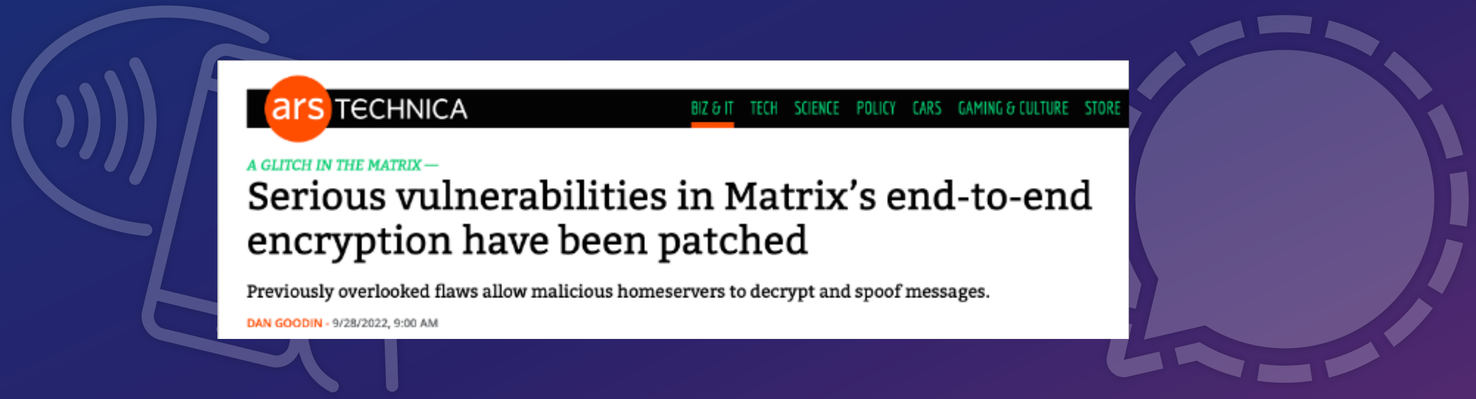
Department of
Computer Science
School of Computing

Motivation

Motivation

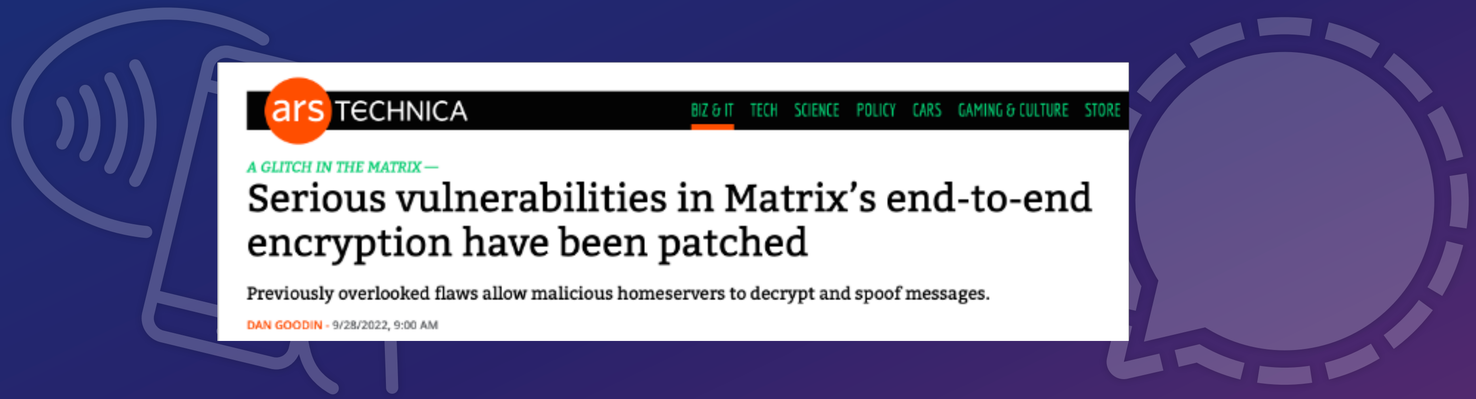

ars TECHNICA
BIZ & IT TECH SCIENCE POLICY CARS GAMING & CULTURE STORE
A GLITCH IN THE MATRIX —
Serious vulnerabilities in Matrix's end-to-end encryption have been patched
Previously overlooked flaws allow malicious homeservers to decrypt and spoof messages.
DAN GOODIN - 9/28/2022, 9:00 AM

Motivation



```
if ((err = SSLHashSHA1.update(…)) != 0)
  goto fail;
if ((err = SSLHashSHA1.update(…)) != 0)
  goto fail;
  goto fail;
if ((err = SSLHashSHA1.final(…)) != 0)
  goto fail;
```

Motivation



```
if ((err = SSLHashSHA1.update(…)) != 0)
  goto fail;
if ((err = SSLHashSHA1.update(…)) != 0)
  goto fail;
  goto fail;
if ((err = SSLHashSHA1.final(…)) != 0)
  goto fail;
```

State of the Art

Protocol model

- Formal protocol description
- Pen and paper proofs
- Automatic tools: Tamarin, ProVerif, …
- Successfully applied to many protocols

State of the Art

Protocol model

- Formal protocol description
- Pen and paper proofs
- Automatic tools: Tamarin, ProVerif, …
- Successfully applied to many protocols

Code generation

- Generated code cannot be optimized
- Integration into a codebase is a soundness threat

Gap

Existing implementation

Gap

Correct protocol implementation?

Existing implementation

Gap

Correct protocol implementation?

Free of buffer overflows?

Existing implementation

Gap

Existing implementation

Correct protocol implementation?

Free of buffer overflows?

Is it safe, i.e., no panics, divisions by zero, …?

This Talk

Model refinement

S&P ‘23

This Talk

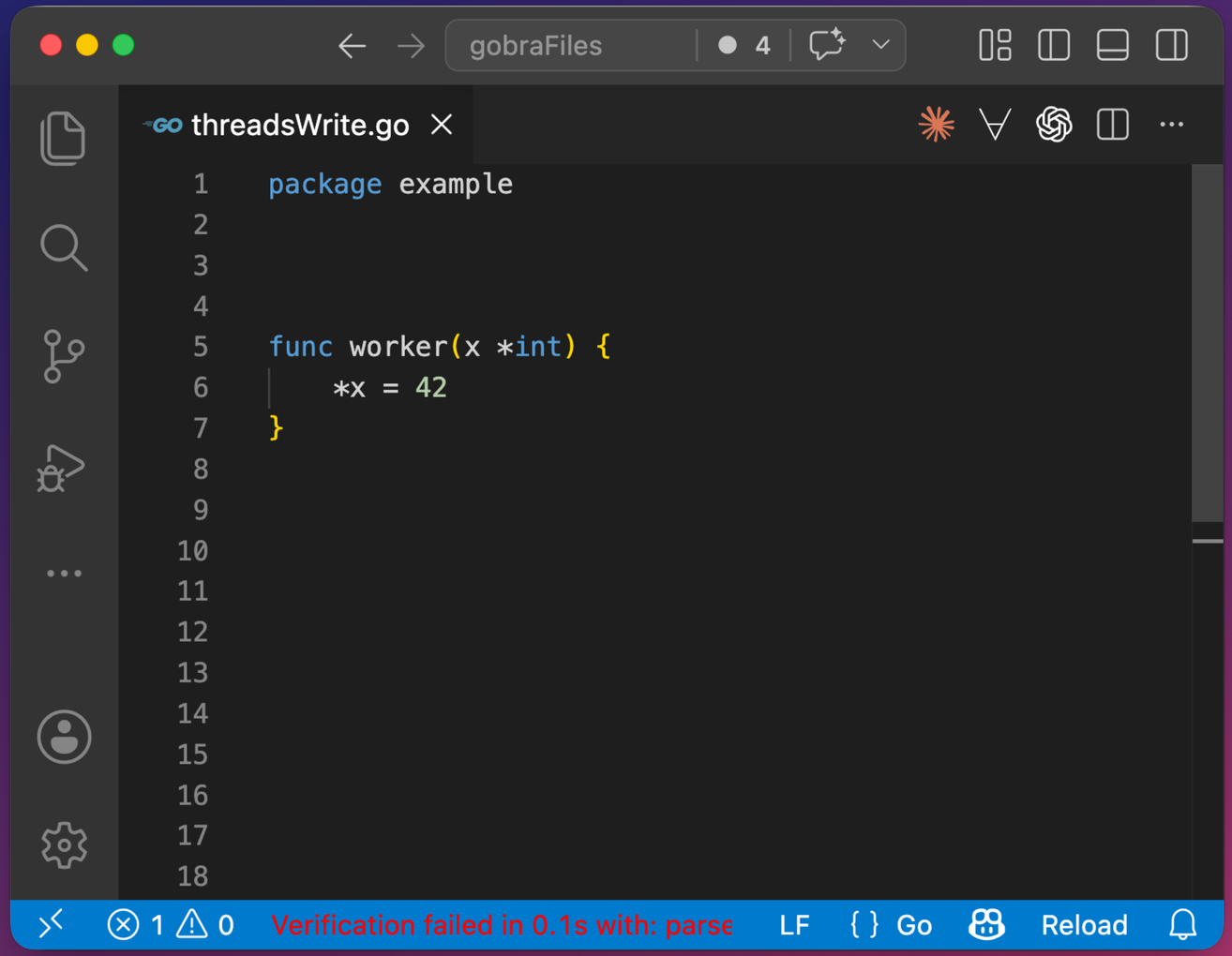
Model refinement

S&P ‘23

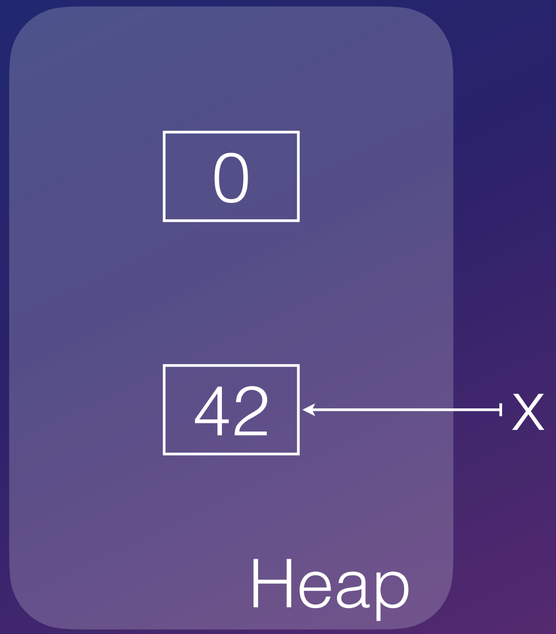
Scaling to large codebases

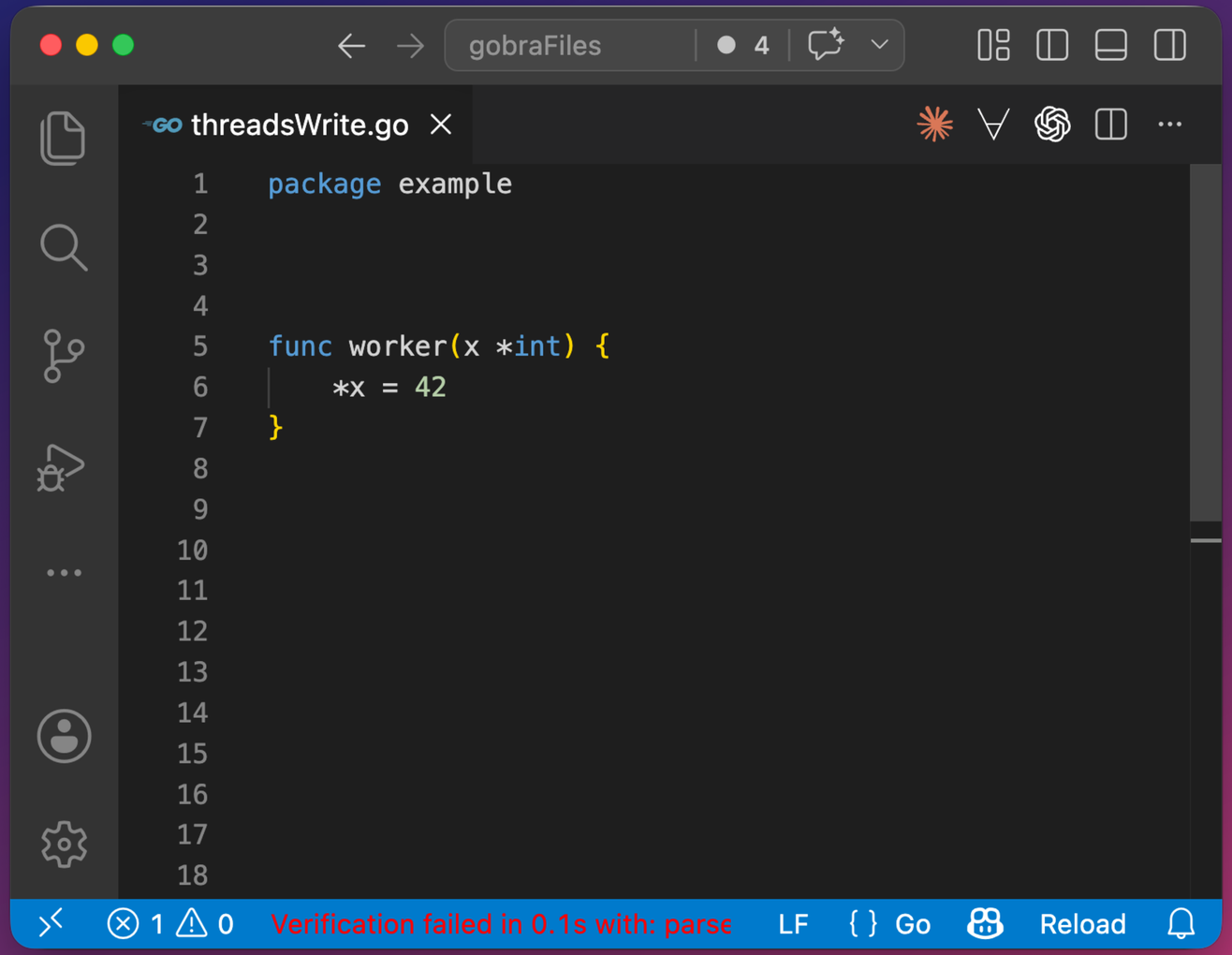
S&P ‘26

Separation Logic & gobra

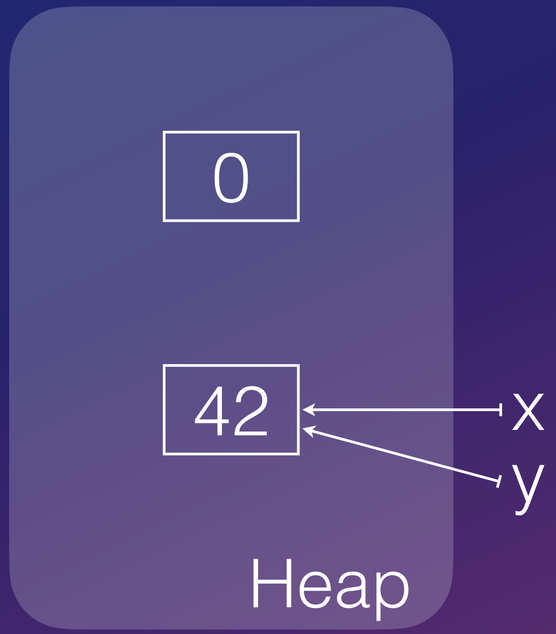


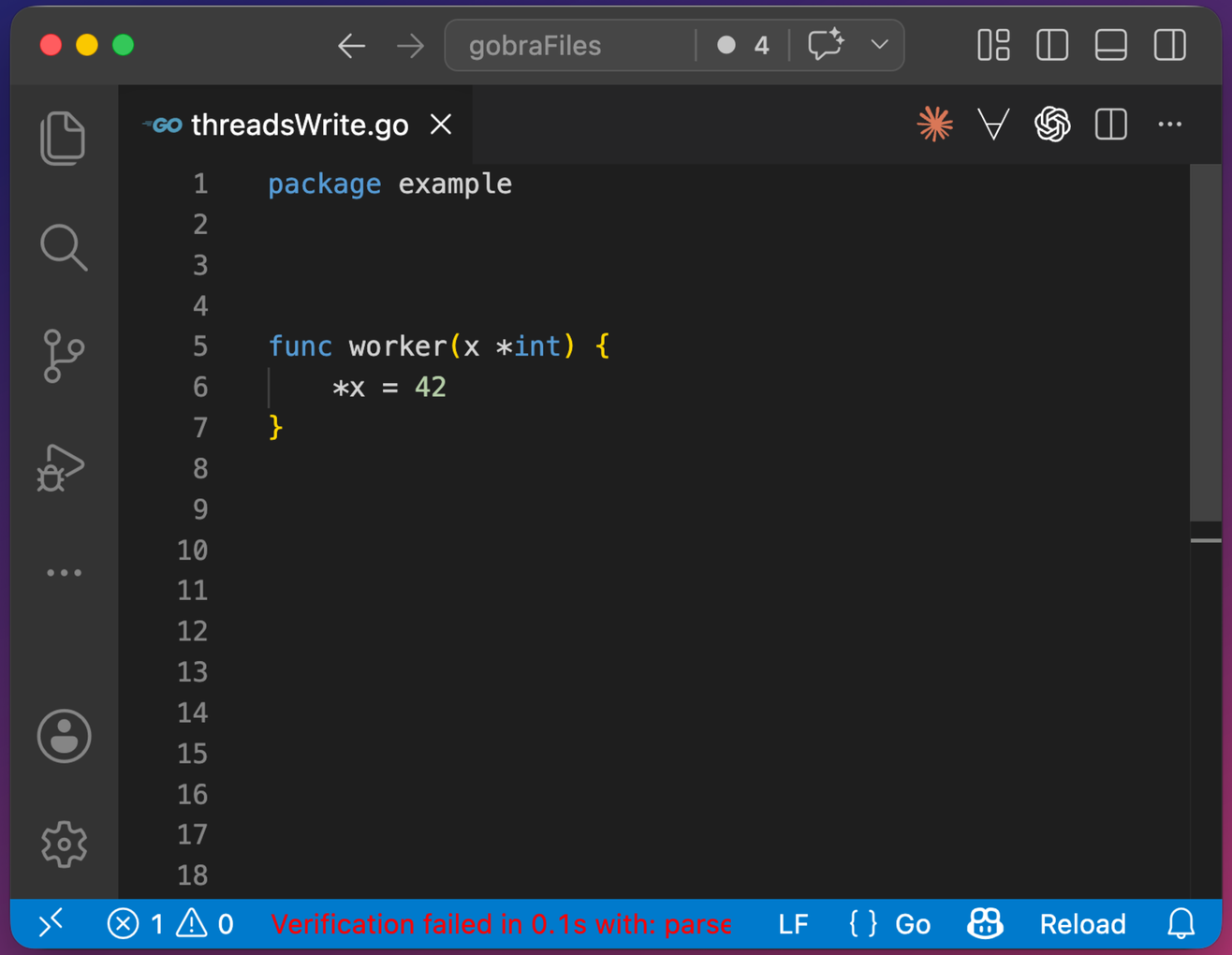
Separation Logic & gobra



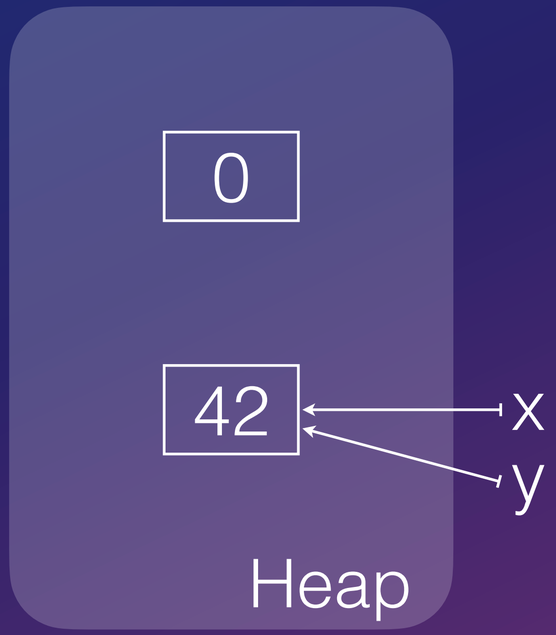


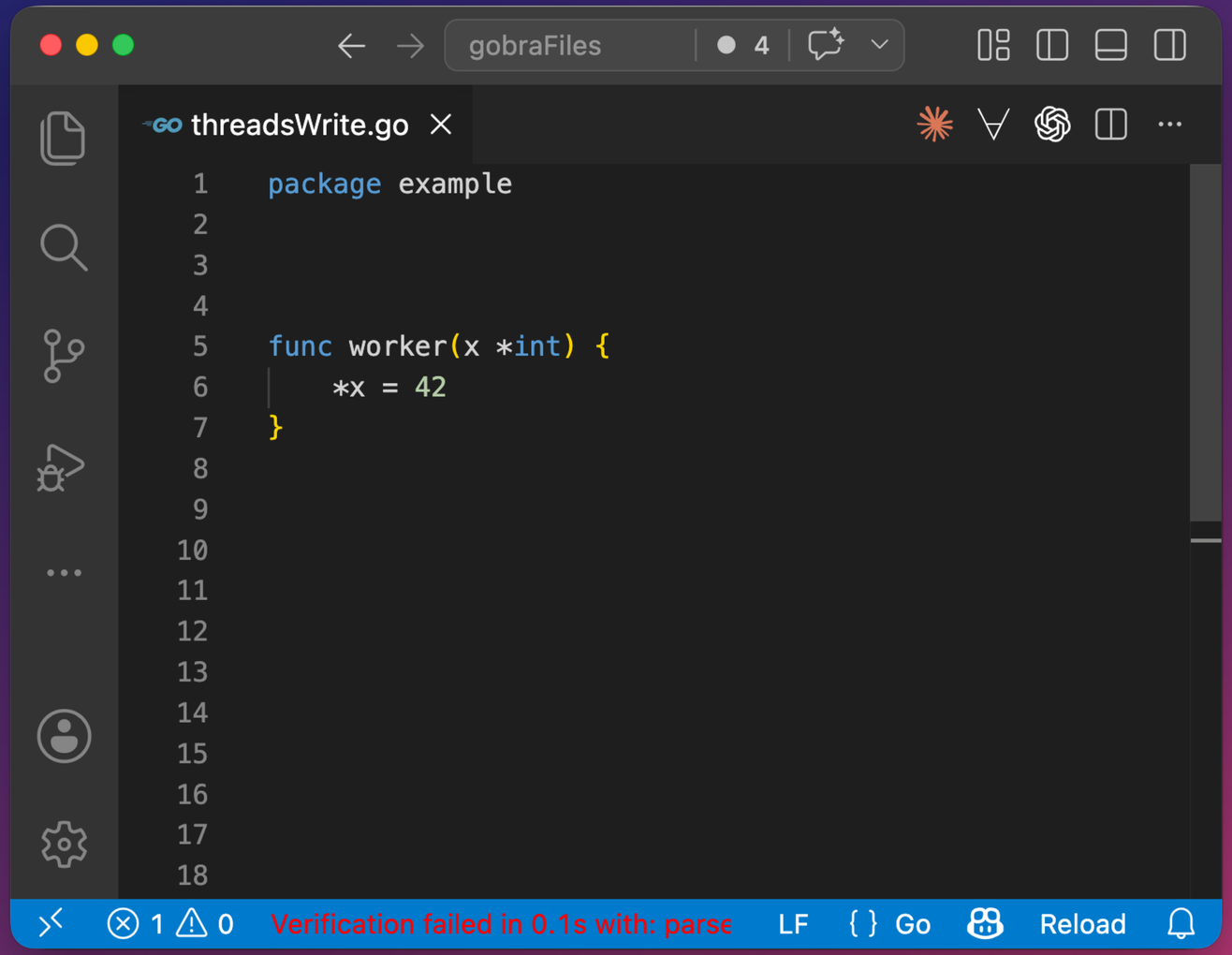
Separation Logic & gobra





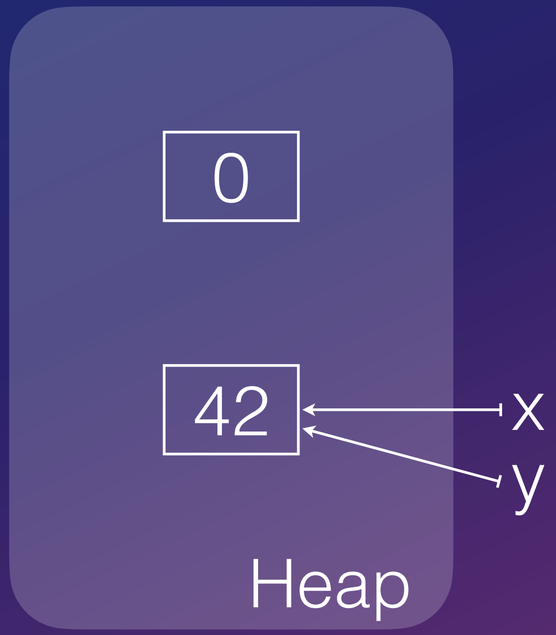
Separation Logic & gobra

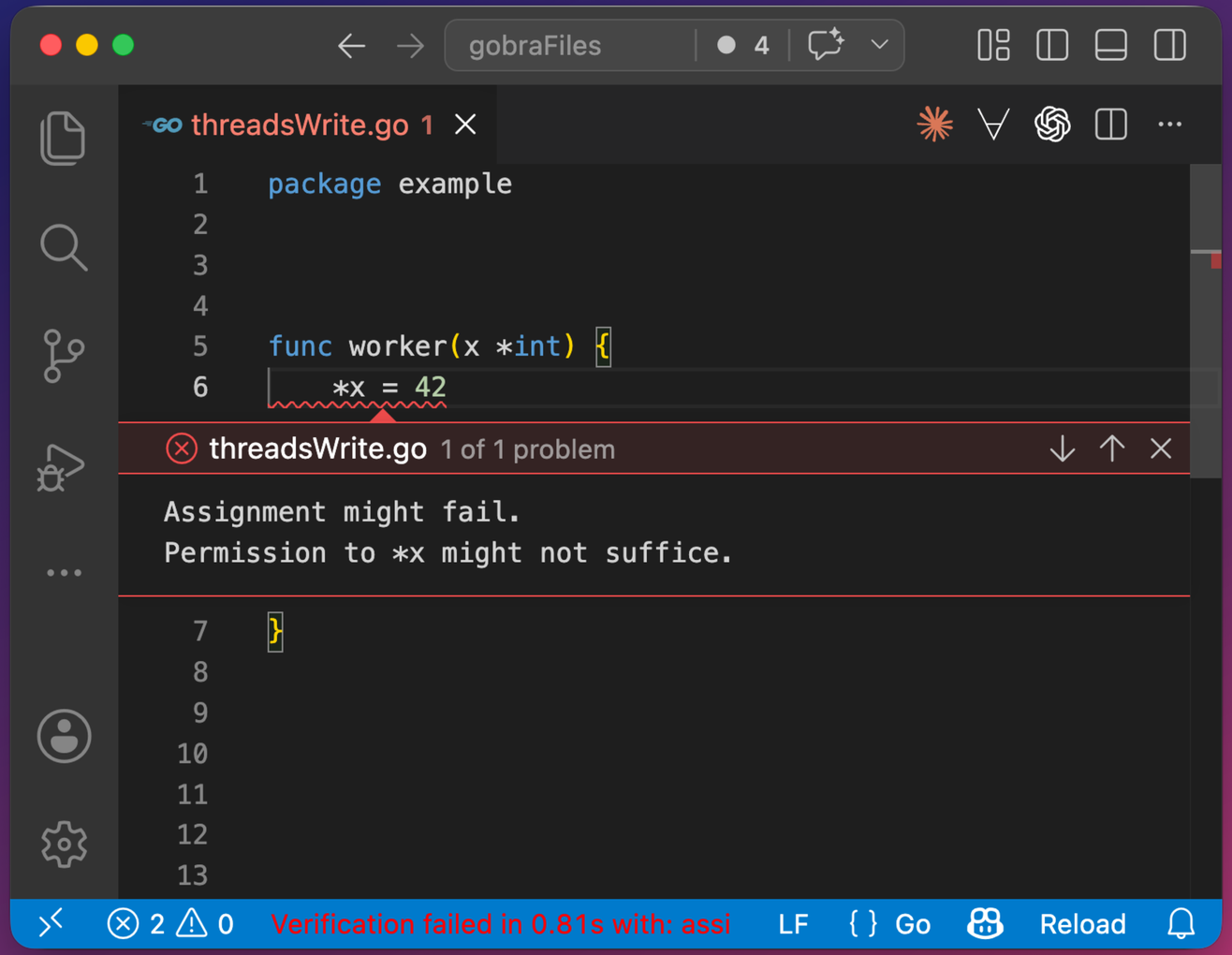




—> Permission per heap location

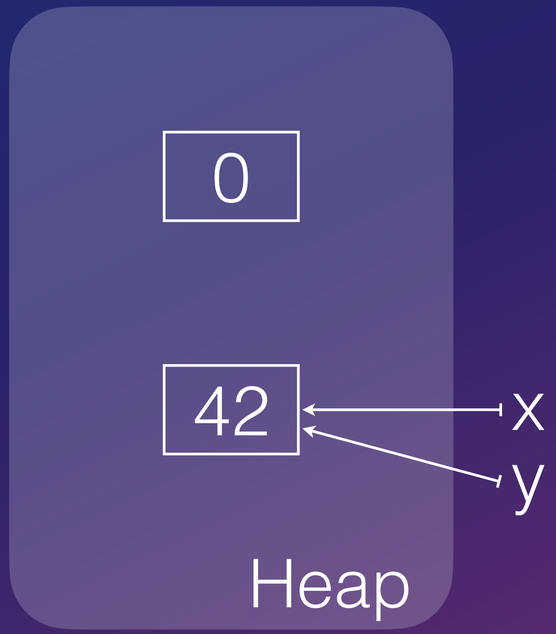
Separation Logic & gobra

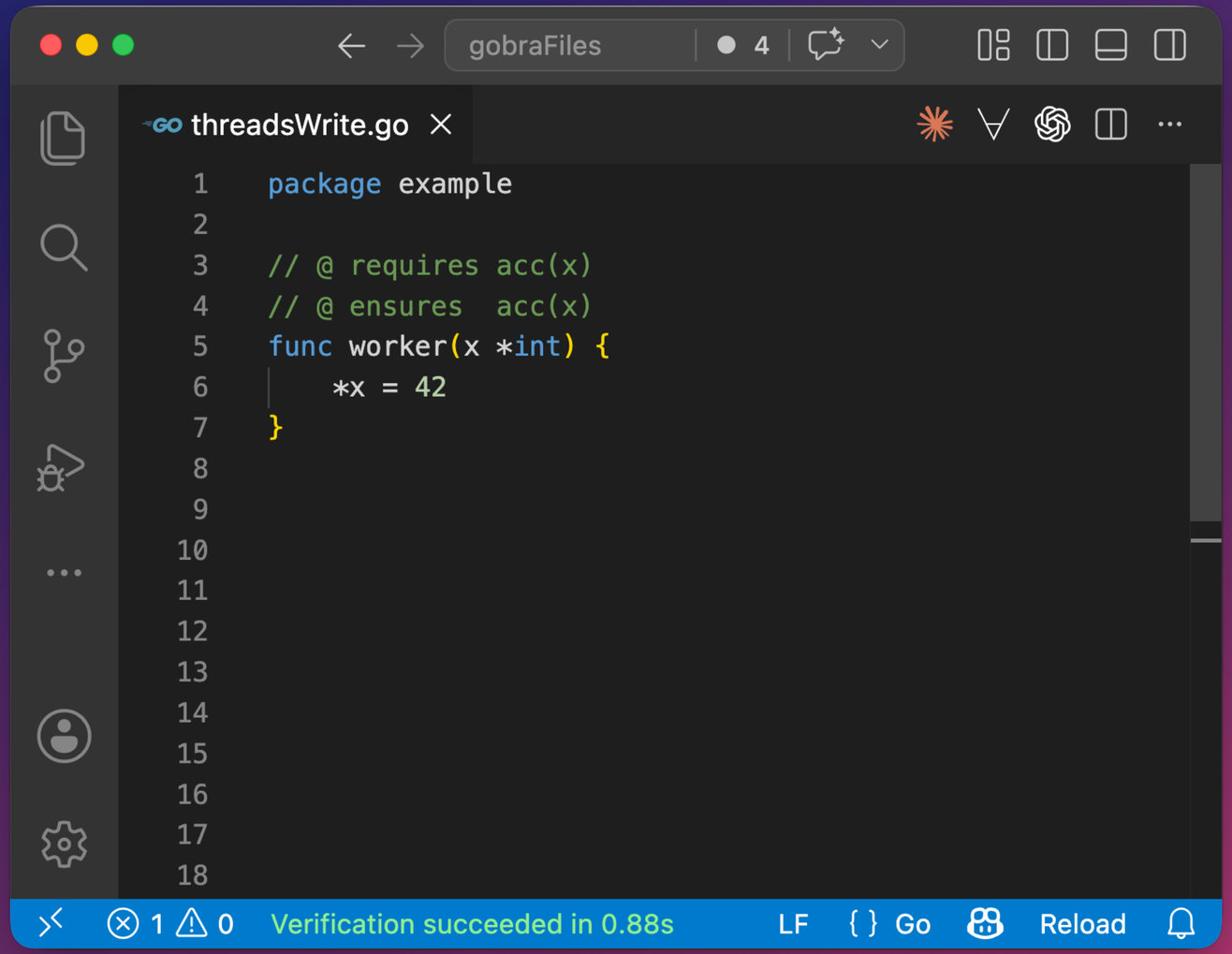




—> Permission per heap location

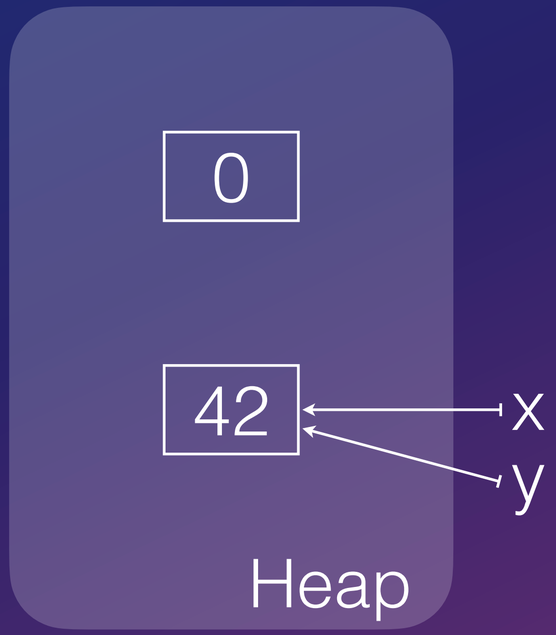
Separation Logic & gobra



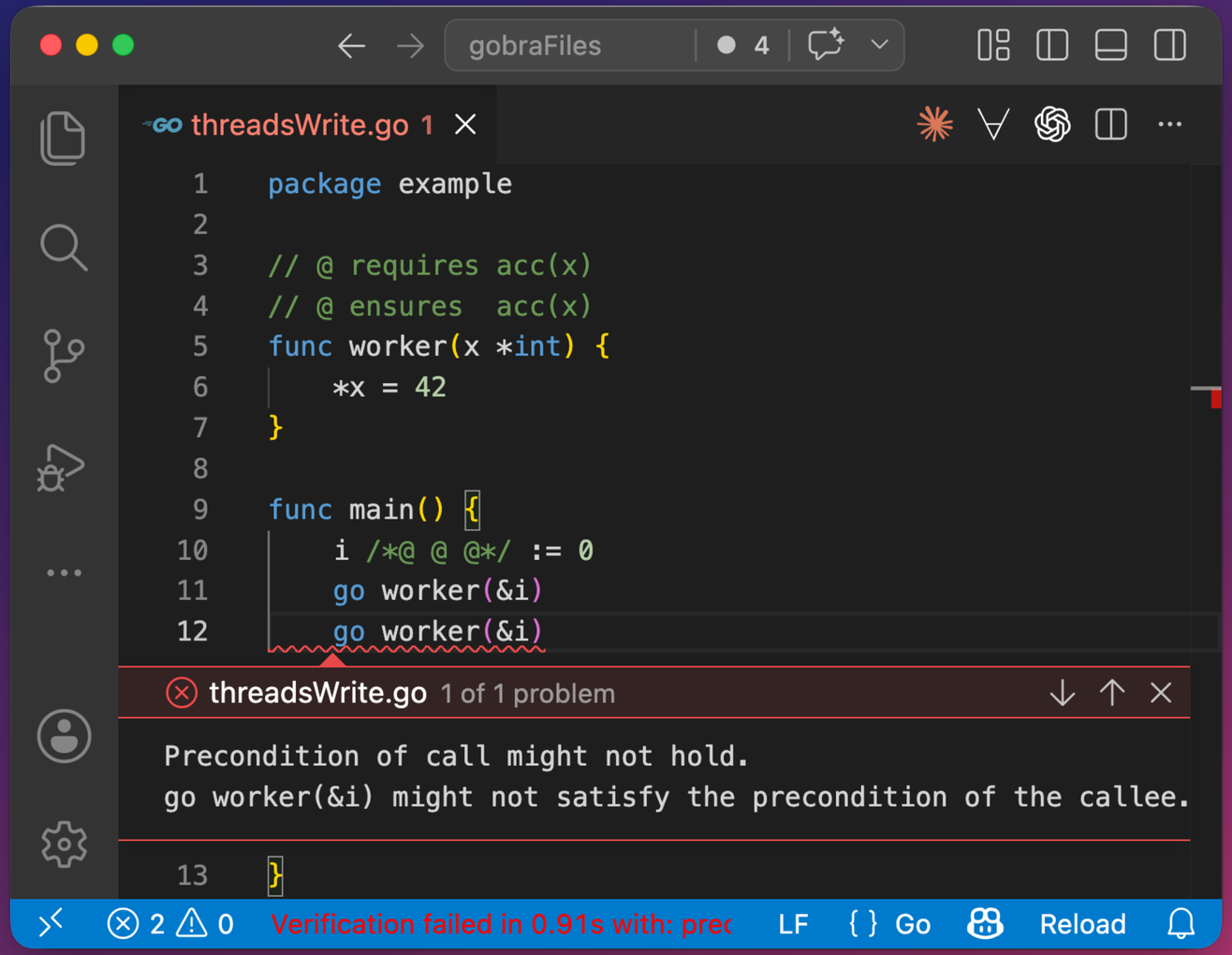


—> Permission per heap location

Separation Logic & gobra



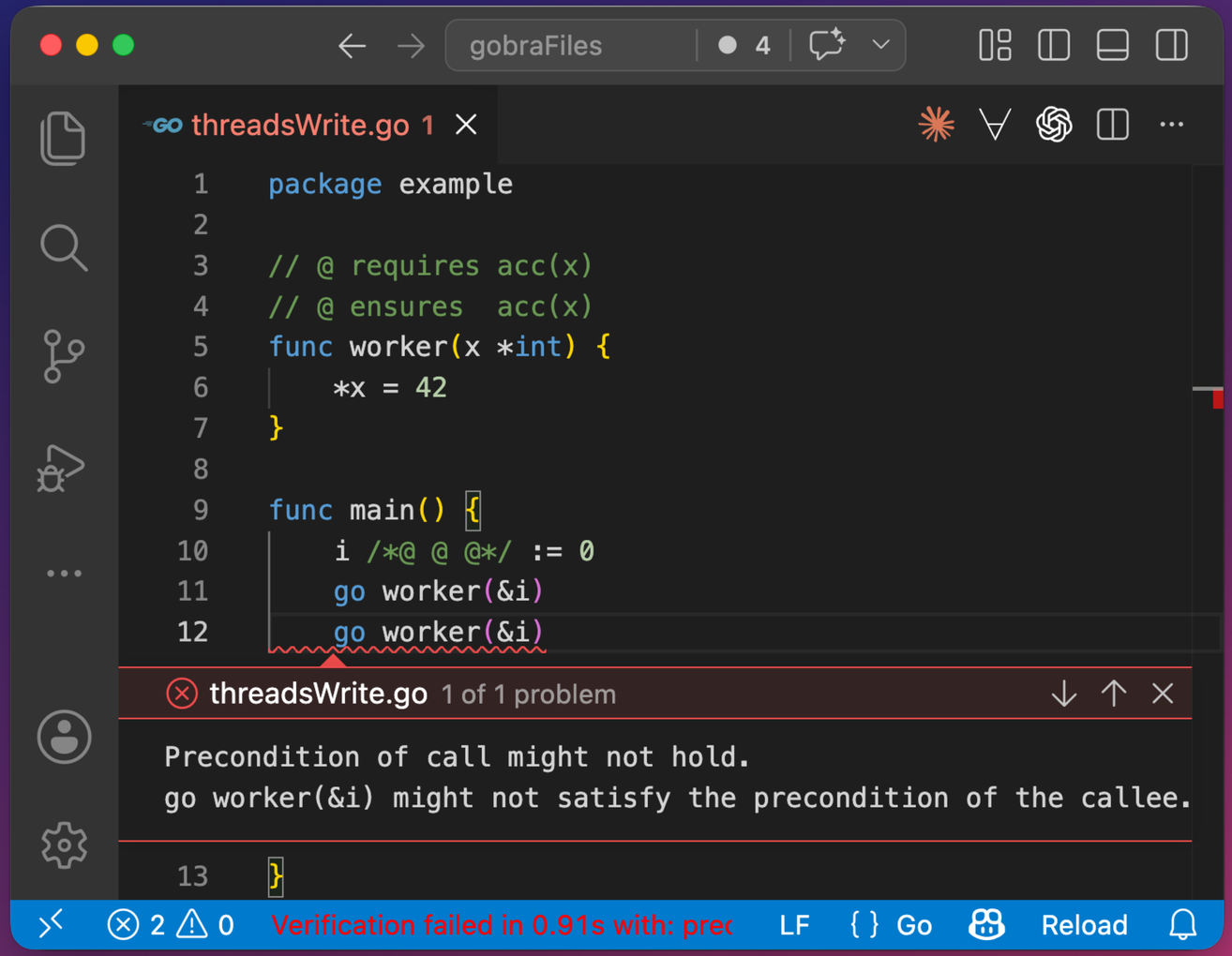
—> Permission per heap location



Separation Logic & gobra

- Automated program verifier for concurrent Go code
- Modular
- Memory safety & panic freedom
- Functional properties

https://gobra.ethz.ch



This Talk

Model refinement

S&P ‘23

Scaling to large codebases

S&P ‘26

Refinement-Based Verification

Verification of
protocol models
in Tamarin

Verification of
implementations
in program verifiers

Protocol model
in Tamarin

1

satisfies

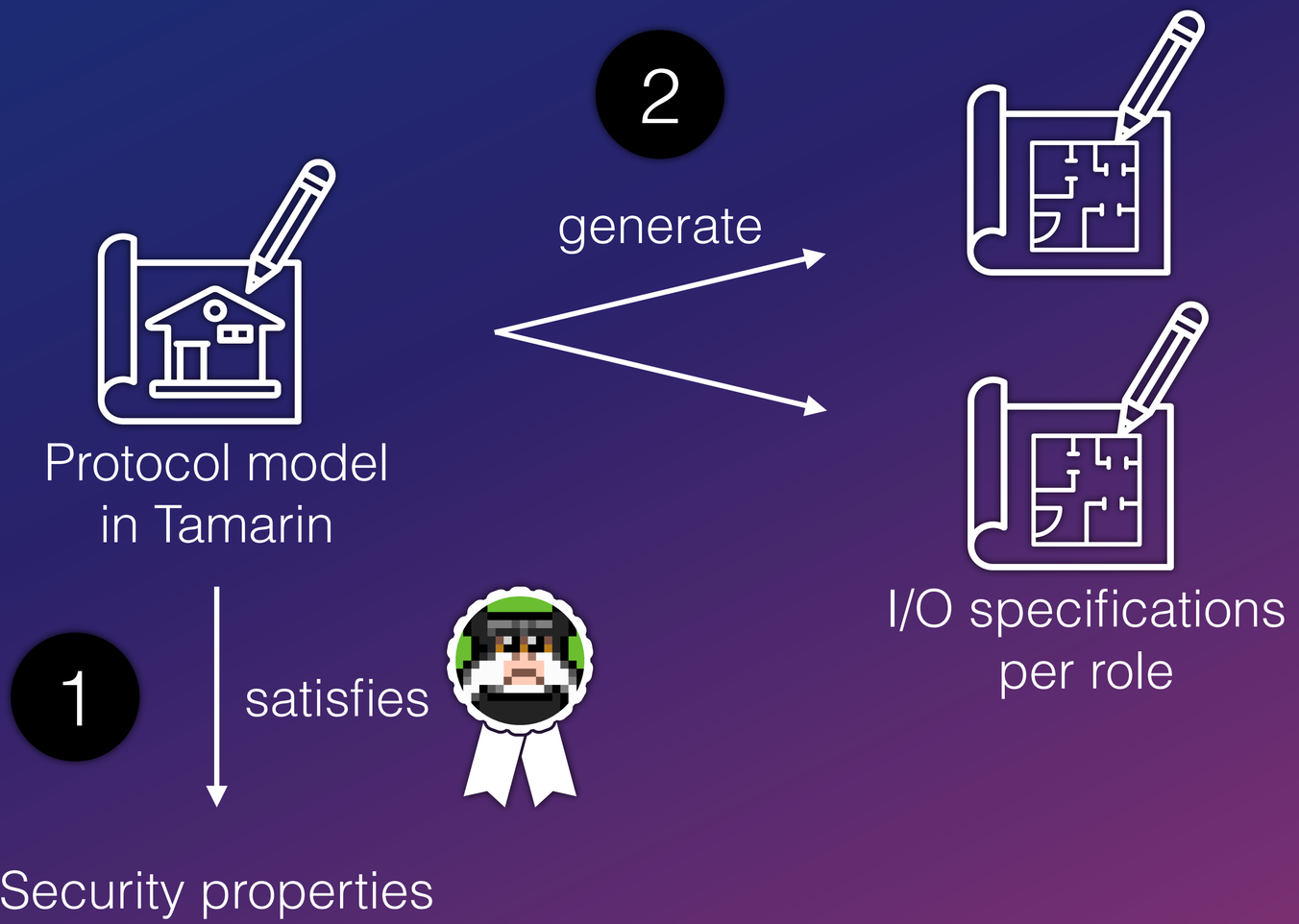
Security properties

Protocol model
in Tamarin

1

satisfies

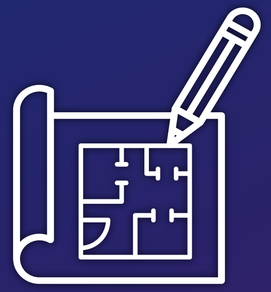
Security properties

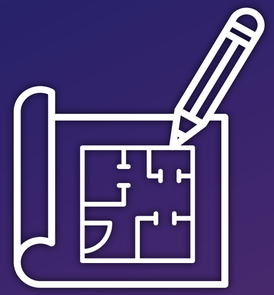

2
generate
Protocol model
in Tamarin
I/O specifications
per role
1
satisfies
Security properties

Protocol model
in Tamarin

2

generate





I/O specifications
per role

meets

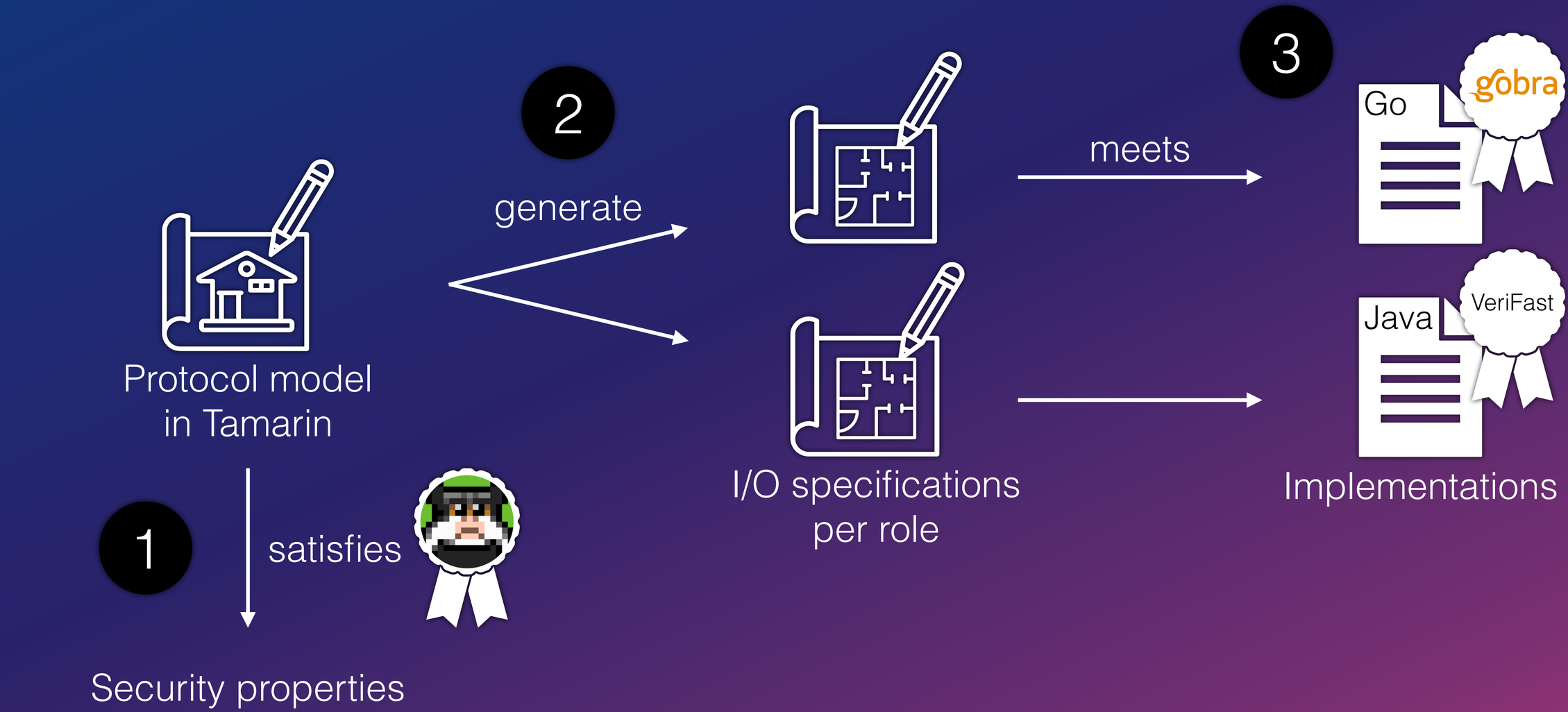
3

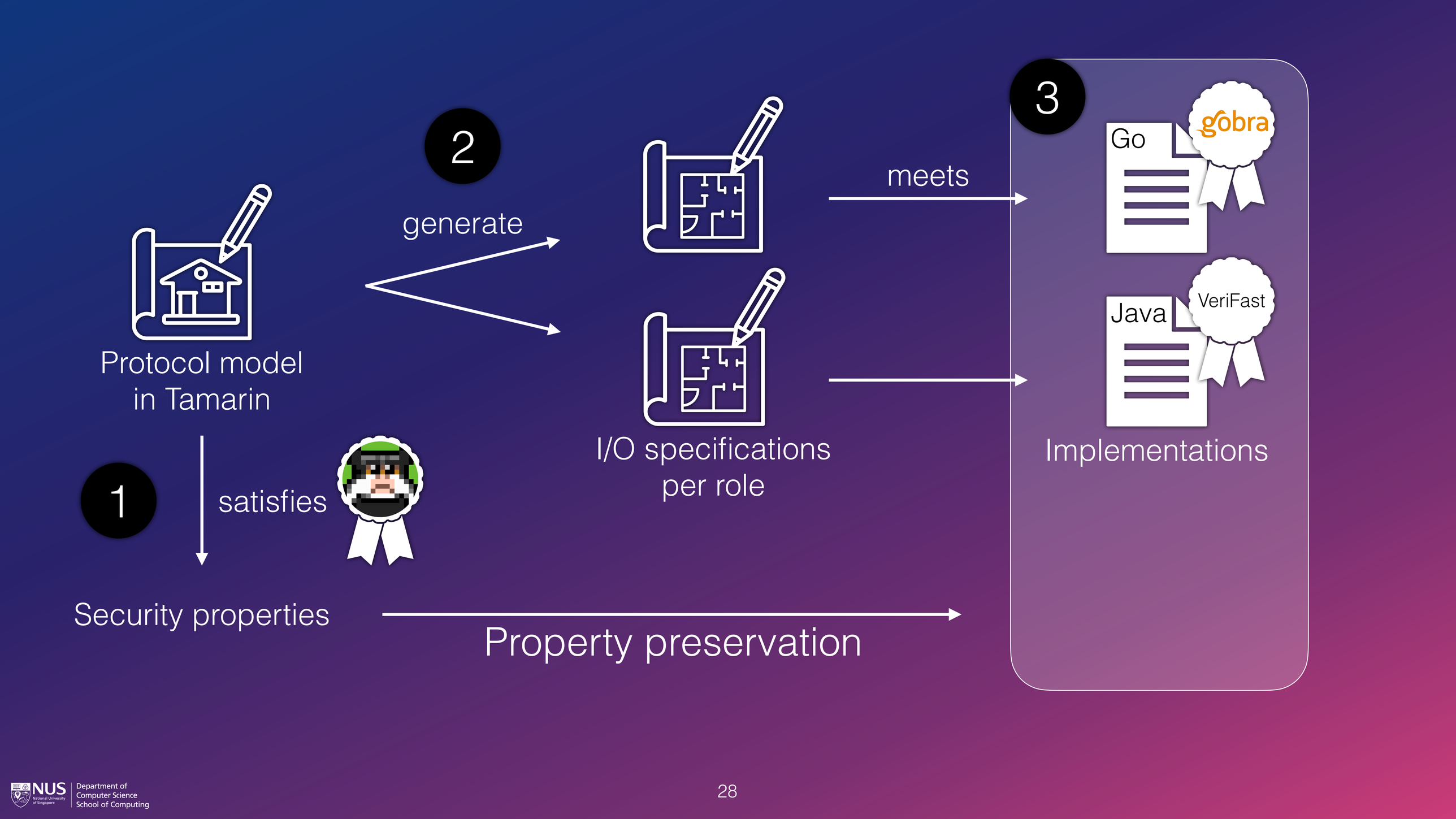
Implementations

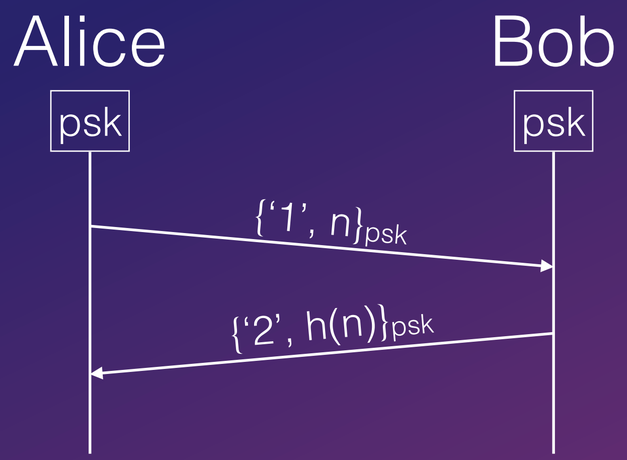
1

satisfies

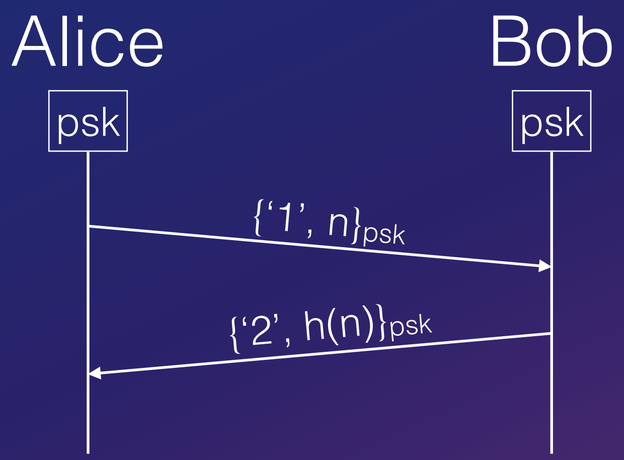
Security properties

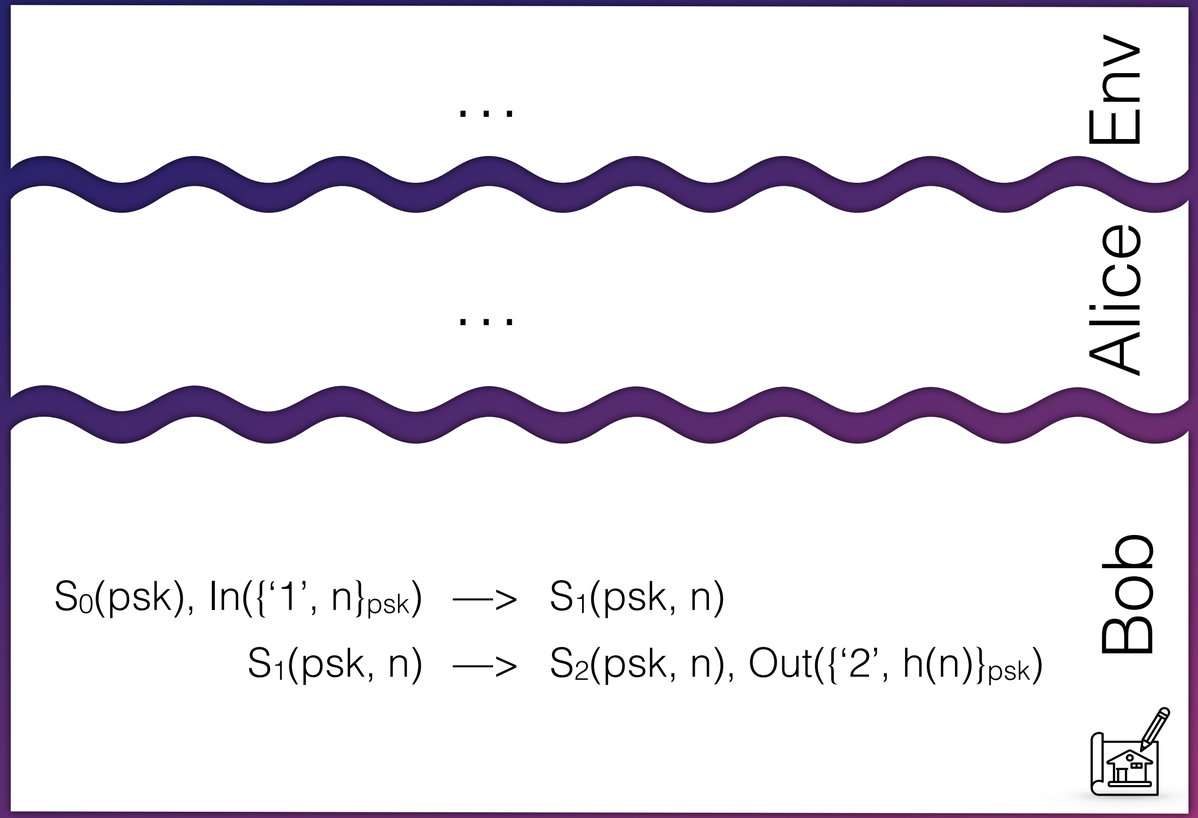

2
generate
3
meets
Protocol model
in Tamarin
I/O specifications
per role
Go
gobra
Java
VeriFast
Implementations
1
satisfies
Security properties


3
2
generate
meets
Go
gobra
Protocol model
in Tamarin
Java
VeriFast
I/O specifications
per role
Implementations
1
satisfies
Security properties
Property preservation


Alice
Bob
psk
psk
{‘1’, n}psk
{‘2’, h(n)}psk

1


Alice
Bob
psk
psk
{‘1’, n}psk
{‘2’, h(n)}psk


Env
Alice
Bob
...
...
S0(psk), In({‘1’, n}psk) —> S1(psk, n)
S1(psk, n) —> S2(psk, n), Out({‘2’, h(n)}psk)

3

$S_0(psk)$, $In(\{‘1’, n\}_{psk})$ —> $S_1(psk, n)$

$S_1(psk, n)$ —> $S_2(psk, n)$, $Out(\{‘2’, h(n)\}_{psk})$

Bob

```
func main(psk []byte) {


  m1 := recv()

  // parse m1
  p1, ok := sdecrypt(m1, psk)
  if !ok { return }

  tag1, n := destruct(p1)
  if tag1 != 1 { return }




  p2 := construct(2, hash(n))
  m2 := sencrypt(p2, psk)



  send(m2)
}
```

3

$S_0(psk)$, $In(\{‘1’, n\}_{psk})$ —> $S_1(psk, n)$

$S_1(psk, n)$ —> $S_2(psk, n)$, $Out(\{‘2’, h(n)\}_{psk})$

Bob

```
//@ requires Bob_IO_Spec(a0)
func main(psk []byte /*@, a0 S @*/) {


  m1 := recv()

  // parse m1
  p1, ok := sdecrypt(m1, psk)
  if !ok { return }

  tag1, n := destruct(p1)
  if tag1 != 1 { return }




  p2 := construct(2, hash(n))
  m2 := sencrypt(p2, psk)



  send(m2)
}
```

3

$S_0(psk)$, $In(\{‘1’, n\}_{psk})$ —> $S_1(psk, n)$

$S_1(psk, n)$ —> $S_2(psk, n)$, $Out(\{‘2’, h(n)\}_{psk})$

Bob

```
//@ requires Bob_IO_Spec(a0)
func main(psk []byte /*@, a0 S @*/) {
  // obtain permission to receive
  //@ a1 := apply_receive_transition(a0)
  m1 := recv()

  // parse m1
  p1, ok := sdecrypt(m1, psk)
  if !ok { return }

  tag1, n := destruct(p1)
  if tag1 != 1 { return }




  p2 := construct(2, hash(n))
  m2 := sencrypt(p2, psk)



  send(m2)
}
```

3

$S_0(psk)$, $In(\{‘1’, n\}_{psk})$ —> $S_1(psk, n)$

$S_1(psk, n)$ —> $S_2(psk, n)$, $Out(\{‘2’, h(n)\}_{psk})$

Bob

```
//@ requires Bob_IO_Spec(a0)
func main(psk []byte /*@, a0 S @*/) {
  // obtain permission to receive
  //@ a1 := apply_receive_transition(a0)
  m1 := recv()

  // parse m1
  p1, ok := sdecrypt(m1, psk)
  if !ok { return }

  tag1, n := destruct(p1)
  if tag1 != 1 { return }

  //@ a2 := apply_transition_1(a1)


  p2 := construct(2, hash(n))
  m2 := sencrypt(p2, psk)



  send(m2)
}
```

3

$S_0(psk)$, $In(\{‘1’, n\}_{psk})$ —> $S_1(psk, n)$

$S_1(psk, n)$ —> $S_2(psk, n)$, $Out(\{‘2’, h(n)\}_{psk})$

Bob

```
//@ requires Bob_IO_Spec(a0)
func main(psk []byte /*@, a0 S @*/) {
  // obtain permission to receive
  //@ a1 := apply_receive_transition(a0)
  m1 := recv()

  // parse m1
  p1, ok := sdecrypt(m1, psk)
  if !ok { return }

  tag1, n := destruct(p1)
  if tag1 != 1 { return }

  //@ a2 := apply_transition_1(a1)
  //@ a3 := apply_transition_2(a2)

  p2 := construct(2, hash(n))
  m2 := sencrypt(p2, psk)



  send(m2)
}
```

3

$S_0(psk)$, $In(\{‘1’, n\}_{psk})$ —> $S_1(psk, n)$

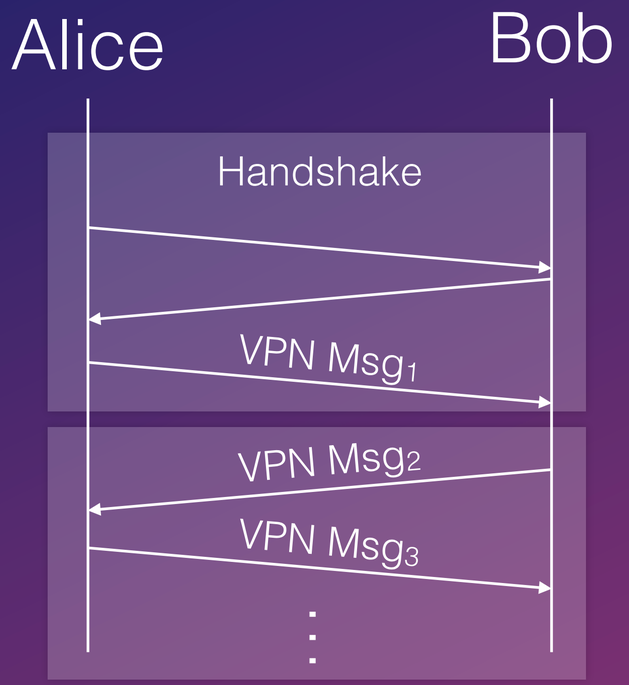
$S_1(psk, n)$ —> $S_2(psk, n)$, $Out(\{‘2’, h(n)\}_{psk})$

Bob

```
//@ requires Bob_IO_Spec(a0)
func main(psk []byte /*@, a0 S @*/) {
  // obtain permission to receive
  //@ a1 := apply_receive_transition(a0)
  m1 := recv()

  // parse m1
  p1, ok := sdecrypt(m1, psk)
  if !ok { return }

  tag1, n := destruct(p1)
  if tag1 != 1 { return }

  //@ a2 := apply_transition_1(a1)
  //@ a3 := apply_transition_2(a2)

  p2 := construct(2, hash(n))
  m2 := sencrypt(p2, psk)

  // obtain permission to send
  //@ a4 := apply_send_transition(a3)
  send(m2)
}
```

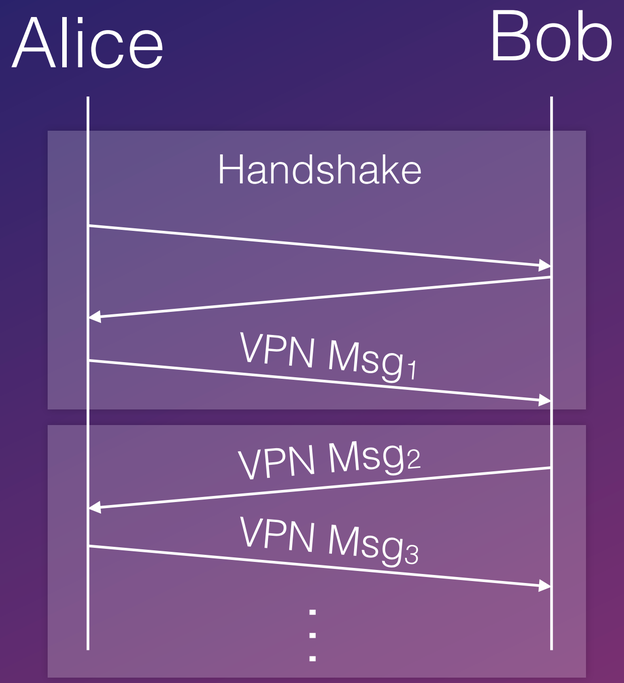
WIREGUARD®
FAST, MODERN, SECURE VPN TUNNEL

- VPN protocol consisting of handshake & transport phase
- ~350 LoC Tamarin model



WIREGUARD®
FAST, MODERN, SECURE VPN TUNNEL

- VPN protocol consisting of handshake & transport phase
- ~350 LoC Tamarin model
- ~550 LoC Go code
- ~1.3k lines of generated I/O spec
- ~2.8k lines of proof annotations



gobra

This Talk

Model refinement

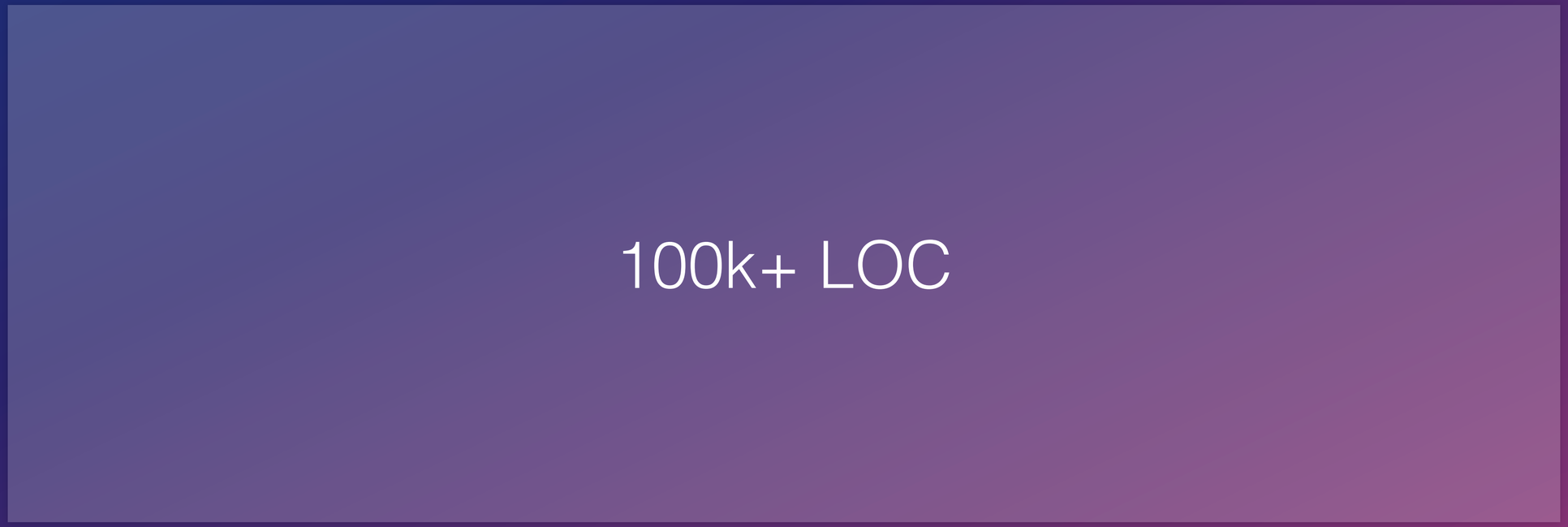
S&P ‘23

Scaling to large codebases

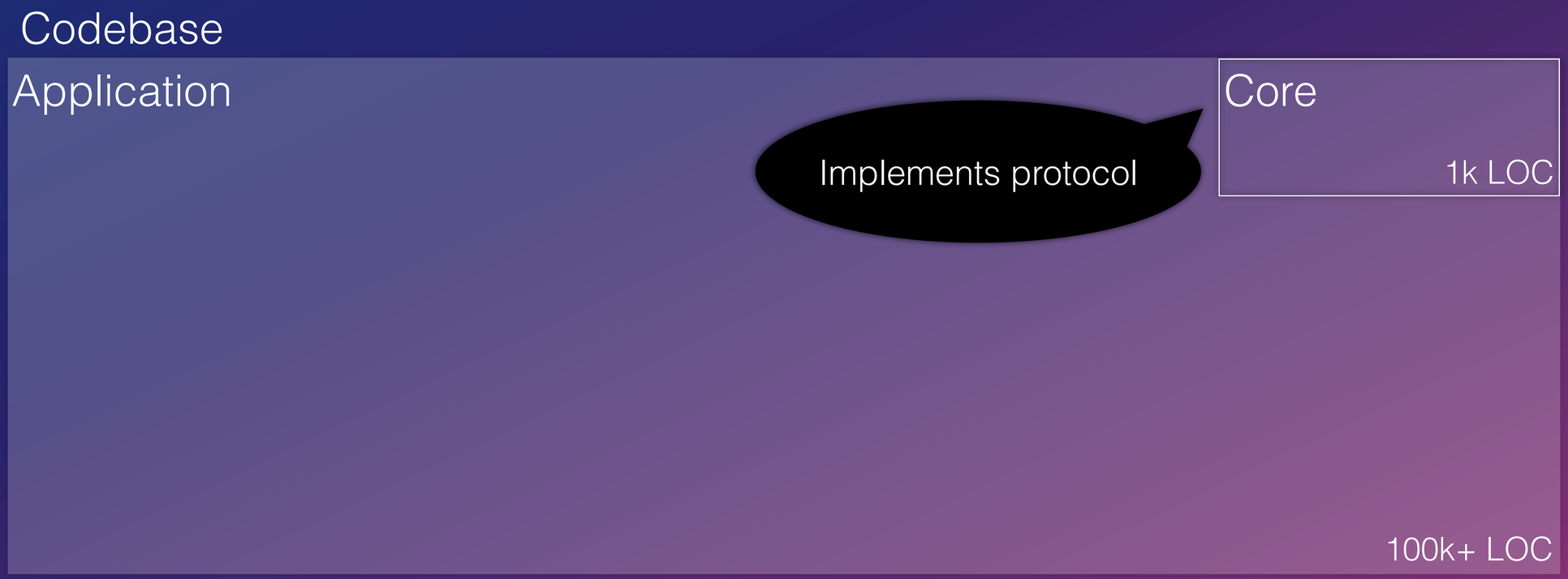
S&P ‘26

Problem

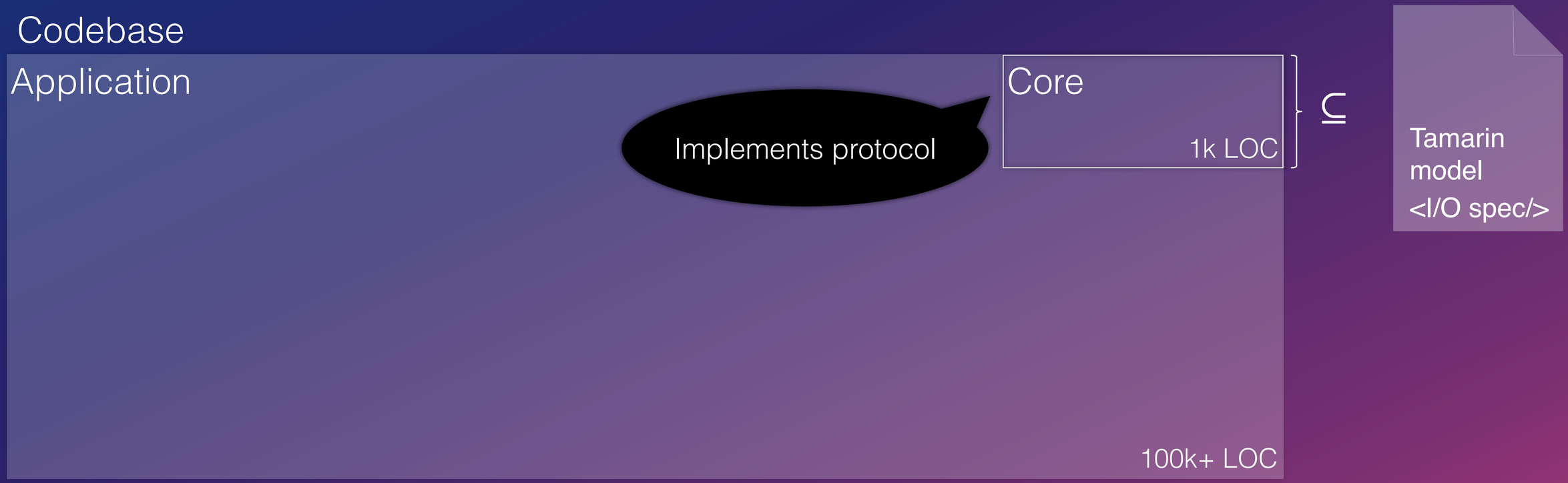
Codebase



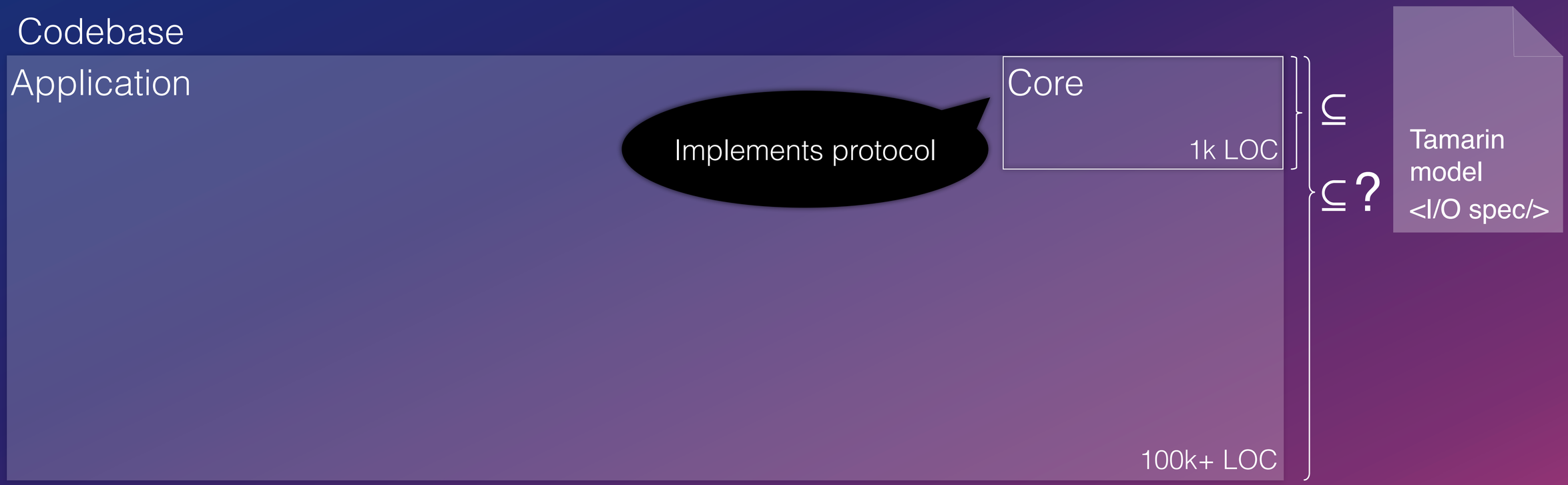
Problem


Codebase
Application
Implements protocol
Core
1k LOC
100k+ LOC

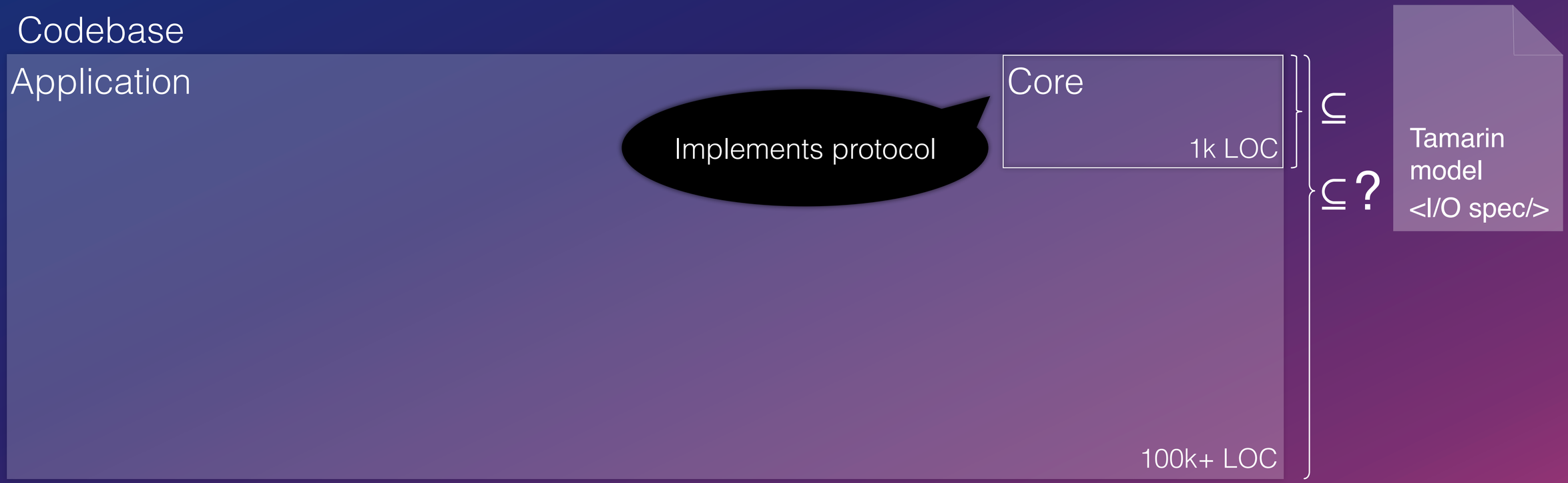
Problem


Codebase
Application
Implements protocol
Core
1k LOC
100k+ LOC
⊆
Tamarin model
<I/O spec/>

Problem

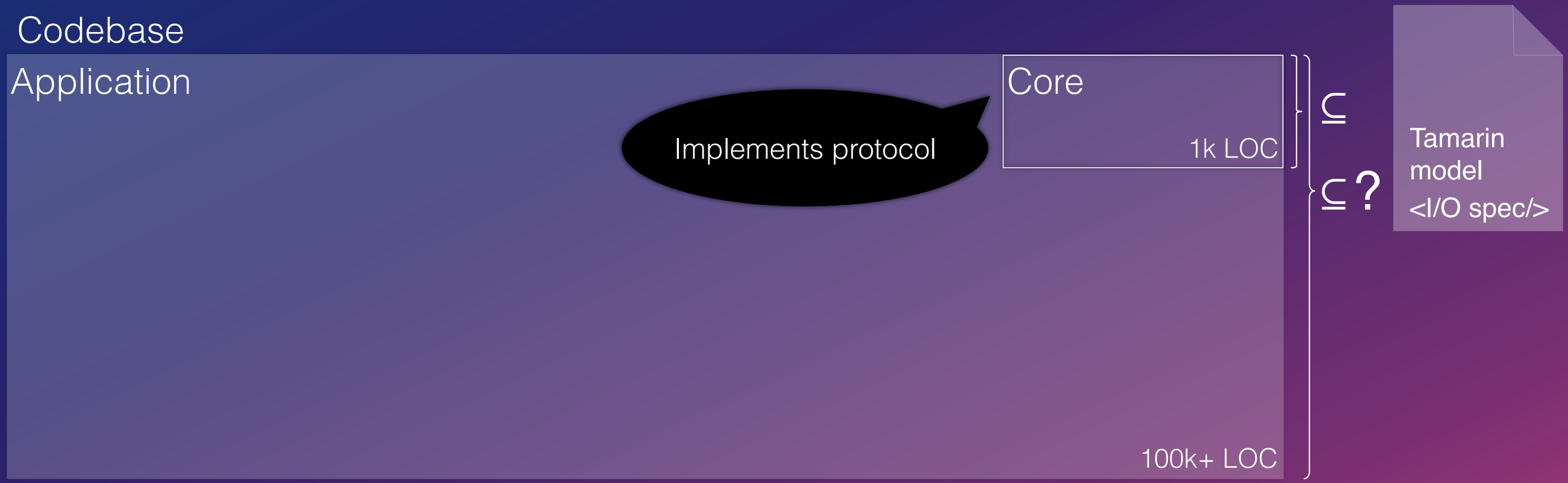

Codebase
Application
Implements protocol
Core
1k LOC
100k+ LOC
⊆
⊆ ?
Tamarin
model
<I/O spec/>

Problem



Sources of unsoundness

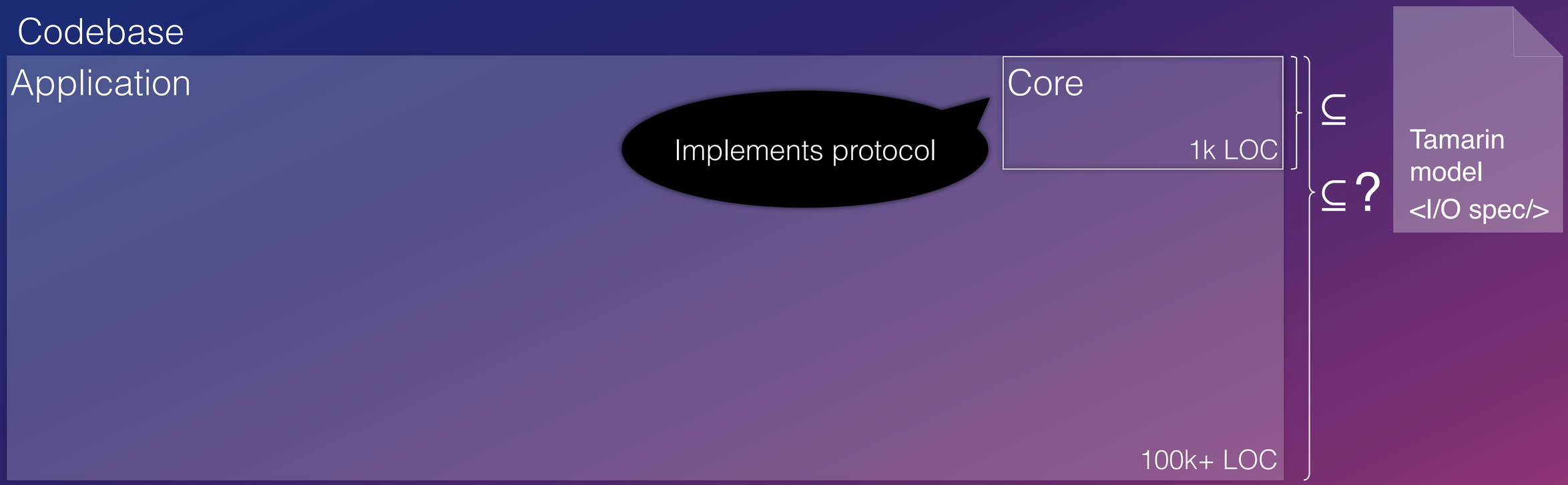
Problem



Sources of unsoundness

I/O within application

Problem



Sources of unsoundness

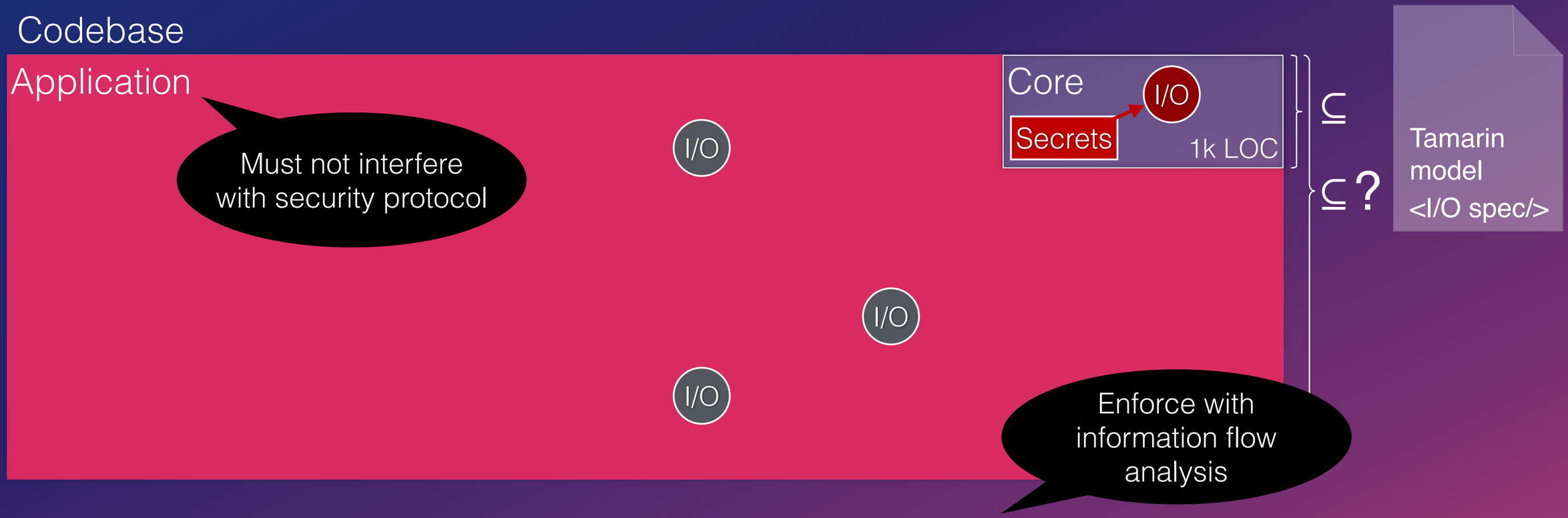
I/O within application

Application violating core's preconditions

1. I/O Independence

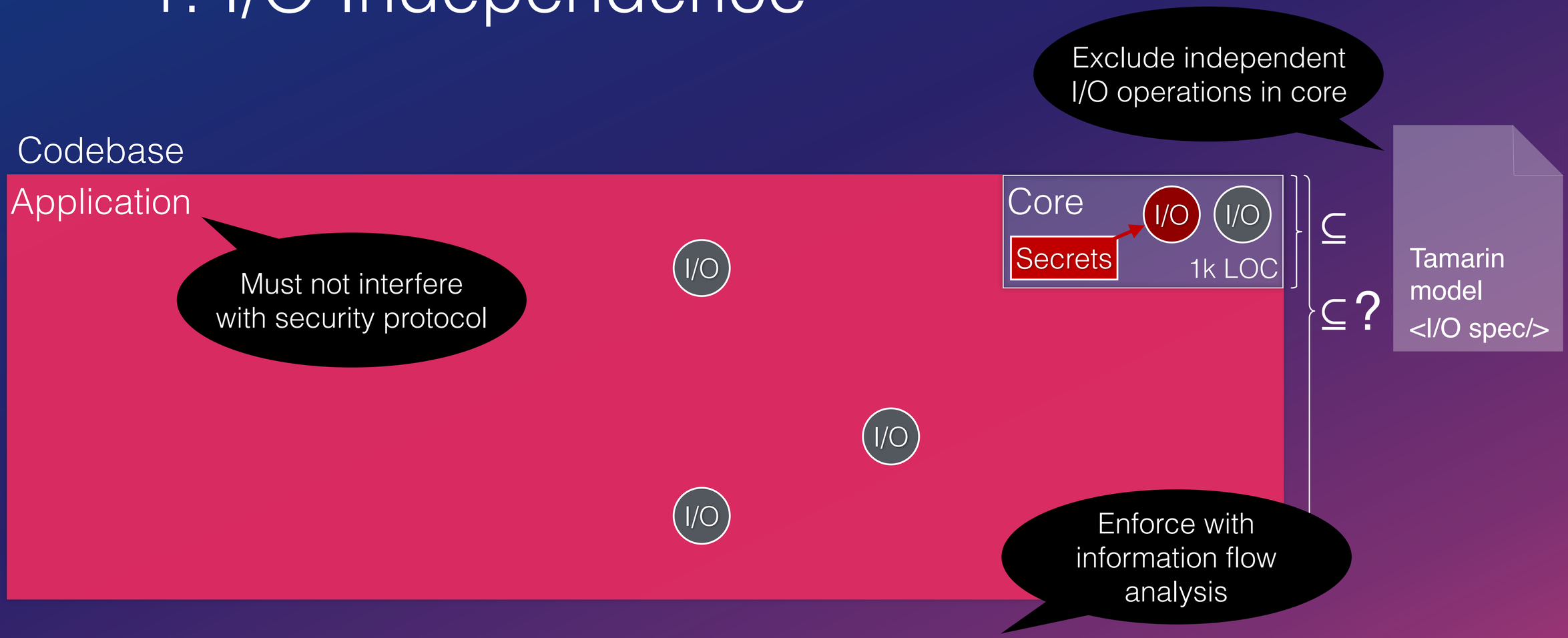
➔ I/O operations in application must be independent of core secrets

1. I/O Independence



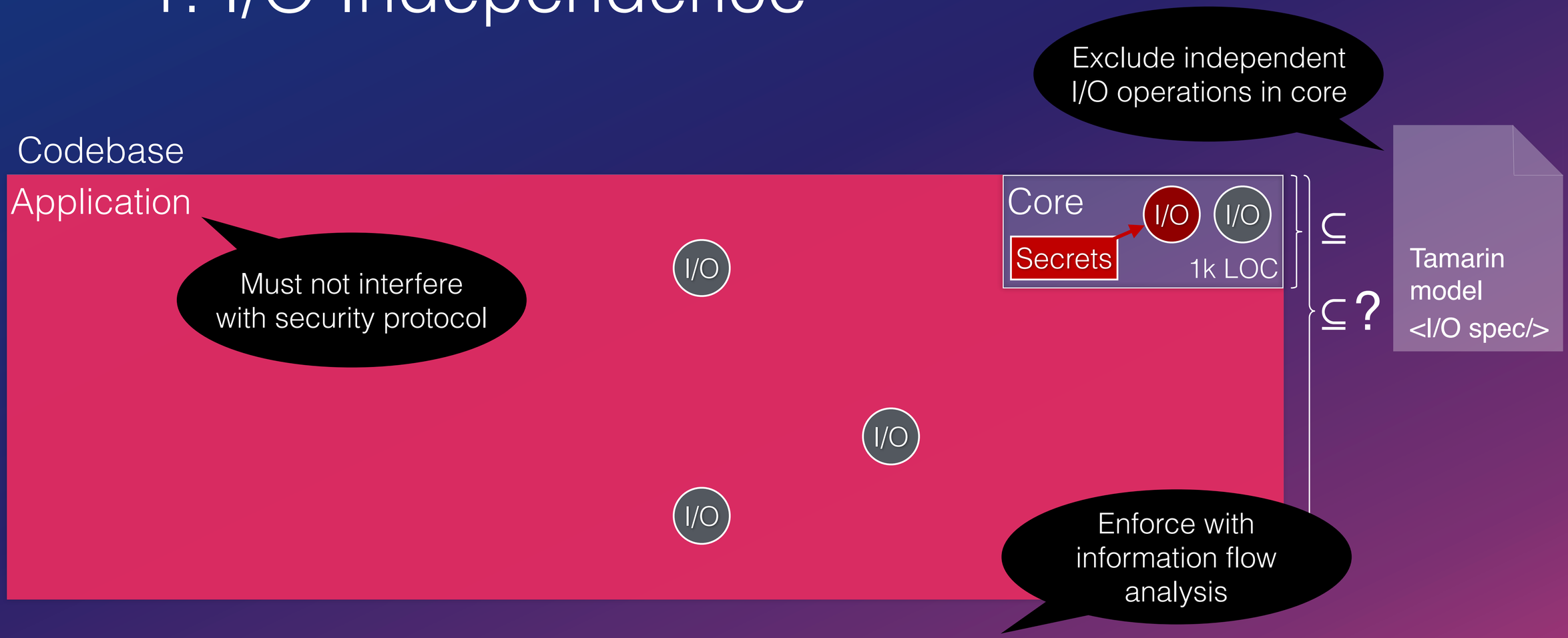
➔ I/O operations in application must be independent of core secrets

1. I/O Independence



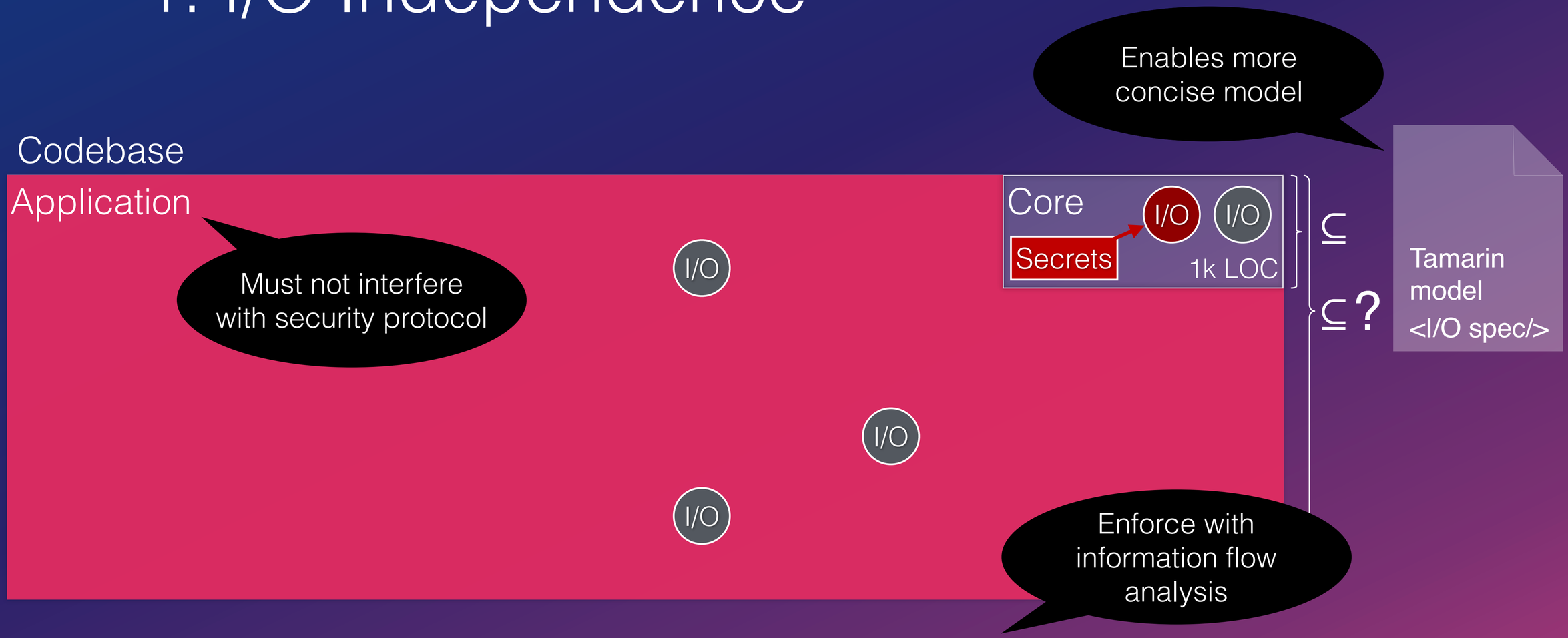
➔ I/O operations in application must be independent of core secrets

1. I/O Independence



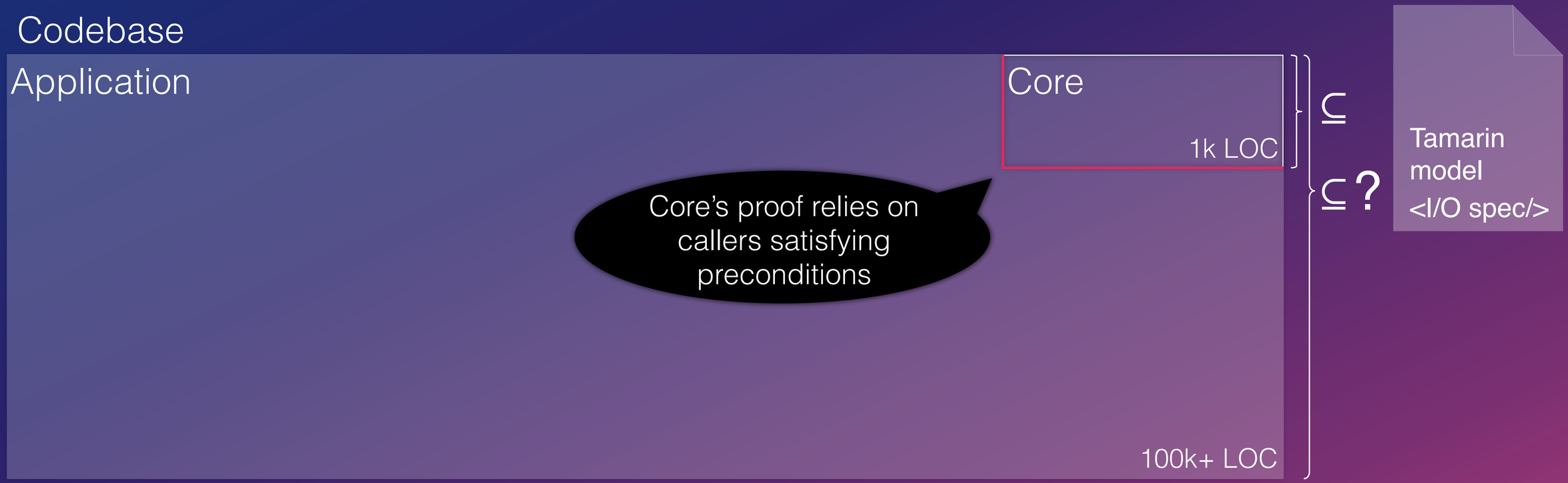
➔ I/O operations in application must be independent of core secrets

1. I/O Independence



➔ I/O operations in application must be independent of core secrets

2. Discharging Core's Assumptions



➔ Syntactically restrict specifications to make them amenable to static analyses

2. Discharging Core's Assumptions

Constructors

Members

➔ Syntactically restrict specifications to make them amenable to static analyses

2. Discharging Core's Assumptions

Constructors

```
func Constructor(psk []byte) (c *Core) {
  ...
}
```

Members

```
func Send(c *Core, msg []byte) {
  ...
}
```

➔ Syntactically restrict specifications to make them amenable to static analyses

2. Discharging Core's Assumptions

Constructors

```
//@ requires psk != nil ==> acc(psk)
//@ ensures  psk != nil ==> acc(psk)
//@ ensures  Inv(c)
func Constructor(psk []byte) (c *Core) {
  ...
}
```

Members

```
func Send(c *Core, msg []byte) {
  ...
}
```

➔ Syntactically restrict specifications to make them amenable to static analyses

2. Discharging Core's Assumptions

Constructors

```
//@ requires psk != nil ==> acc(psk)
//@ ensures  psk != nil ==> acc(psk)
//@ ensures  Inv(c)
func Constructor(psk []byte) (c *Core) {
  ...
}
```

Members

```
//@ requires c != nil ==> Inv(c)
//@ requires msg != nil ==> acc(msg)
//@ ensures  c != nil ==> Inv(c)
//@ ensures  msg != nil ==> acc(msg)
func Send(c *Core, msg []byte) {
  ...
}
```

➔ Syntactically restrict specifications to make them amenable to static analyses

Ensuring Locality

Constructors

```
//@ requires psk != nil ==> acc(psk)
//@ ensures  psk != nil ==> acc(psk)
//@ ensures  Inv(c)
func Constructor(psk []byte) (c *Core) {
  ...
}
```

Members

```
//@ requires c != nil ==> Inv(c)
//@ requires msg != nil ==> acc(msg)
//@ ensures  c != nil ==> Inv(c)
//@ ensures  msg != nil ==> acc(msg)
func Send(c *Core, msg []byte) {
  ...
}
```

Ensuring Locality

Constructors

```
//@ requires psk != nil ==> acc(psk)
//@ ensures  psk != nil ==> acc(psk)
//@ ensures  Inv(c)
func Constructor(psk []byte) (c *Core) {
  ...
}
```

Members

```
//@ requires c != nil ==> Inv(c)
//@ requires msg != nil ==> acc(msg)
//@ ensures  c != nil ==> Inv(c)
//@ ensures  msg != nil ==> acc(msg)
func Send(c *Core, msg []byte) {
  ...
}
```

```
func BuggyApp1(psk []byte) {
  c := Constructor(psk)
  go Send(c, []byte{})
  go Send(c, []byte{})
}
```

Ensuring Locality

Constructors

```
//@ requires psk != nil ==> acc(psk)
//@ ensures  psk != nil ==> acc(psk)
//@ ensures  Inv(c)
func Constructor(psk []byte) (c *Core) {
  ...
}
```

Members

```
//@ requires c != nil ==> Inv(c)
//@ requires msg != nil ==> acc(msg)
//@ ensures  c != nil ==> Inv(c)
//@ ensures  msg != nil ==> acc(msg)
func Send(c *Core, msg []byte) {
  ...
}
```

```
func BuggyApp1(psk []byte) {
  c := Constructor(psk)
  go Send(c, []byte{})
  go Send(c, []byte{})
}
```

Ensuring Locality

Constructors

```
//@ requires psk != nil ==> acc(psk)
//@ ensures  psk != nil ==> acc(psk)
//@ ensures  Inv(c)
func Constructor(psk []byte) (c *Core) {
  ...
}
```

```
func BuggyApp1(psk []byte) {
  c := Constructor(psk)
  go Send(c, []byte{})
  go Send(c, []byte{})
}
```

Members

```
//@ requires c != nil ==> Inv(c)
//@ requires msg != nil ==> acc(msg)
//@ ensures  c != nil ==> Inv(c)
//@ ensures  msg != nil ==> acc(msg)
func Send(c *Core, msg []byte) {
  ...
}
```

```
func BuggyApp2(psk []byte) {
  go func() {
    c0 := Constructor([]byte{})
    Send(c0, psk)
  }()
  go func() {
    c1 := Constructor([]byte{})
    Send(c0, psk)
  }()
}
```

Ensuring Locality

Constructors

```
//@ requires psk != nil ==> acc(psk)
//@ ensures  psk != nil ==> acc(psk)
//@ ensures  Inv(c)
func Constructor(psk []byte) (c *Core) {
  ...
}
```

```
func BuggyApp1(psk []byte) {
  c := Constructor(psk)
  go Send(c, []byte{})
  go Send(c, []byte{})
}
```

Members

```
//@ requires c != nil ==> Inv(c)
//@ requires msg != nil ==> acc(msg)
//@ ensures  c != nil ==> Inv(c)
//@ ensures  msg != nil ==> acc(msg)
func Send(c *Core, msg []byte) {
  ...
}
```

```
func BuggyApp2(psk []byte) {
  go func() {
    c0 := Constructor([]byte{})
    Send(c0, psk)
  }()
  go func() {
    c1 := Constructor([]byte{})
    Send(c0, psk)
  }()
}
```

Ensuring Locality

Constructors

```
//@ requires psk != nil ==> acc(psk)
//@ ensures  psk != nil ==> acc(psk)
//@ ensures  Inv(c)
func Constructor(psk []byte) (c *Core) {
  ...
}
```

```
func BuggyApp1(psk []byte) {
  c := Constructor(psk)
  go Send(c, []byte{})
  go Send(c, []byte{})
}
```

Members

```
//@ requires c != nil ==> Inv(c)
//@ requires msg != nil ==> acc(msg)
//@ ensures  c != nil ==> Inv(c)
//@ ensures  msg != nil ==> acc(msg)
func Send(c *Core, msg []byte) {
  ...
}
```

```
func BuggyApp2(psk []byte) {
  go func() {
    c0 := Constructor([]byte{})
    Send(c0, psk)
  }()
  go func() {
    c1 := Constructor([]byte{})
    Send(c0, psk)
  }()
}
```

➔ Escape analysis ensures locality of Core function arguments

Ensuring Disjointness

Constructors

```
//@ requires psk != nil ==> acc(psk)
//@ ensures  psk != nil ==> acc(psk)
//@ ensures  Inv(c)
func Constructor(psk []byte) (c *Core) {
  ...
}
```

Members

```
//@ requires c != nil ==> Inv(c)
//@ requires msg != nil ==> acc(msg)
//@ ensures  c != nil ==> Inv(c)
//@ ensures  msg != nil ==> acc(msg)
func Send(c *Core, msg []byte) {
  ...
}
```

Ensuring Disjointness

Constructors

```
//@ requires psk != nil ==> acc(psk)
//@ ensures  psk != nil ==> acc(psk)
//@ ensures  Inv(c)
func Constructor(psk []byte) (c *Core) {
  ...
}
```

```
func BuggyApp3(psk []byte) {
  c := Constructor(psk)
  ApiFunc(c, psk, psk)
}
```

Members

```
//@ requires c != nil ==> Inv(c)
//@ requires msg != nil ==> acc(msg)
//@ ensures  c != nil ==> Inv(c)
//@ ensures  msg != nil ==> acc(msg)
func Send(c *Core, msg []byte) {
  ...
}
```

Ensuring Disjointness

Constructors

```
//@ requires psk != nil ==> acc(psk)
//@ ensures  psk != nil ==> acc(psk)
//@ ensures  Inv(c)
func Constructor(psk []byte) (c *Core) {
  ...
}
```

```
func BuggyApp3(psk []byte) {
  c := Constructor(psk)
  ApiFunc(c, psk, psk)
}
```

Members

```
//@ requires c != nil ==> Inv(c)
//@ requires msg != nil ==> acc(msg)
//@ ensures  c != nil ==> Inv(c)
//@ ensures  msg != nil ==> acc(msg)
func Send(c *Core, msg []byte) {
  ...
}
```

Ensuring Disjointness

Constructors

```
//@ requires psk != nil ==> acc(psk)
//@ ensures  psk != nil ==> acc(psk)
//@ ensures  Inv(c)
func Constructor(psk []byte) (c *Core) {
  ...
}
```

Members

```
//@ requires c != nil ==> Inv(c)
//@ requires msg != nil ==> acc(msg)
//@ ensures  c != nil ==> Inv(c)
//@ ensures  msg != nil ==> acc(msg)
func Send(c *Core, msg []byte) {
  ...
}
```

```
func BuggyApp3(psk []byte) {
  c := Constructor(psk)
  ApiFunc(c, psk, psk)
}

func BuggyApp4(psk []byte) {
  c := Constructor(psk)
  Send(c, c.psk)
}
```

Ensuring Disjointness

Constructors

```
//@ requires psk != nil ==> acc(psk)
//@ ensures  psk != nil ==> acc(psk)
//@ ensures  Inv(c)
func Constructor(psk []byte) (c *Core) {
  ...
}
```

```
func BuggyApp3(psk []byte) {
  c := Constructor(psk)
  ApiFunc(c, psk, psk)
}

func BuggyApp4(psk []byte) {
  c := Constructor(psk)
  Send(c, c.psk)
}
```

Members

```
//@ requires c != nil ==> Inv(c)
//@ requires msg != nil ==> acc(msg)
//@ ensures  c != nil ==> Inv(c)
//@ ensures  msg != nil ==> acc(msg)
func Send(c *Core, msg []byte) {
  ...
}
```

Ensuring Disjointness

Constructors

```
//@ requires psk != nil ==> acc(psk)
//@ ensures  psk != nil ==> acc(psk)
//@ ensures  Inv(c)
func Constructor(psk []byte) (c *Core) {
  ...
}
```

Members

```
//@ requires c != nil ==> Inv(c)
//@ requires msg != nil ==> acc(msg)
//@ ensures  c != nil ==> Inv(c)
//@ ensures  msg != nil ==> acc(msg)
func Send(c *Core, msg []byte) {
  ...
}
```

```
func BuggyApp3(psk []byte) {
  c := Constructor(psk)
  ApiFunc(c, psk, psk)
}

func BuggyApp4(psk []byte) {
  c := Constructor(psk)
  Send(c, c.psk)
}
```

```
func BuggyApp5(psk []byte) {
  c := Constructor(psk)
  c.psk = []byte{}
  Send(c, []byte{})
}
```

Ensuring Disjointness

Constructors

```
//@ requires psk != nil ==> acc(psk)
//@ ensures  psk != nil ==> acc(psk)
//@ ensures  Inv(c)
func Constructor(psk []byte) (c *Core) {
  ...
}
```

```
func BuggyApp3(psk []byte) {
  c := Constructor(psk)
  ApiFunc(c, psk, psk)
}

func BuggyApp4(psk []byte) {
  c := Constructor(psk)
  Send(c, c.psk)
}
```

Members

```
//@ requires c != nil ==> Inv(c)
//@ requires msg != nil ==> acc(msg)
//@ ensures  c != nil ==> Inv(c)
//@ ensures  msg != nil ==> acc(msg)
func Send(c *Core, msg []byte) {
  ...
}
```

```
func BuggyApp5(psk []byte) {
  c := Constructor(psk)
  c.psk = []byte{}
  Send(c, []byte{})
}
```

Ensuring Disjointness

Constructors

```
//@ requires psk != nil ==> acc(psk)
//@ ensures  psk != nil ==> acc(psk)
//@ ensures  Inv(c)
func Constructor(psk []byte) (c *Core) {
  ...
}
```

Members

```
//@ requires c != nil ==> Inv(c)
//@ requires msg != nil ==> acc(msg)
//@ ensures  c != nil ==> Inv(c)
//@ ensures  msg != nil ==> acc(msg)
func Send(c *Core, msg []byte) {
  ...
}
```

```
func BuggyApp3(psk []byte) {
  c := Constructor(psk)
  ApiFunc(c, psk, psk)
}

func BuggyApp4(psk []byte) {
  c := Constructor(psk)
  Send(c, c.psk)
}
```

```
func BuggyApp5(psk []byte) {
  c := Constructor(psk)
  c.psk = []byte{}
  Send(c, []byte{})
}
```

➔ Pointer analysis ensures disjointness of Core function arguments and maintenance of Core invariant

Soundness

```
func main() {

  c := Constructor()

  a := []byte{}



  Send(c, a)

}
```

```
//@ ensures  Inv(c)
func Constructor() (c *Core) {
  ...
}
```

```
//@ requires c != nil ==> Inv(c)
//@ requires msg != nil ==> acc(msg)
//@ ensures  c != nil ==> Inv(c)
//@ ensures  msg != nil ==> acc(msg)
func Send(c *Core, msg []byte) {
  ...
}
```

Soundness

```
func main() {
  //@ L := {}; I := {}

  c := Constructor()


  a := []byte{}




  Send(c, a)


}
```

```
//@ ensures  Inv(c)
func Constructor() (c *Core) {
  ...
}
```

```
//@ requires c != nil ==> Inv(c)
//@ requires msg != nil ==> acc(msg)
//@ ensures  c != nil ==> Inv(c)
//@ ensures  msg != nil ==> acc(msg)
func Send(c *Core, msg []byte) {
  ...
}
```

Soundness

```
func main() {
  //@ L := {}; I := {}

  c := Constructor()
  //@ I = I ∪ {c}

  a := []byte{}




  Send(c, a)


}
```

```
//@ ensures  Inv(c)
func Constructor() (c *Core) {
  ...
}
```

```
//@ requires c != nil ==> Inv(c)
//@ requires msg != nil ==> acc(msg)
//@ ensures  c != nil ==> Inv(c)
//@ ensures  msg != nil ==> acc(msg)
func Send(c *Core, msg []byte) {
  ...
}
```

Soundness

```
func main() {
  //@ L := {}; I := {}

  c := Constructor()
  //@ I = I ∪ {c}

  a := []byte{}
  //@ L = L ∪ {a}



  Send(c, a)


}
```

```
//@ ensures  Inv(c)
func Constructor() (c *Core) {
  ...
}
```

```
//@ requires c != nil ==> Inv(c)
//@ requires msg != nil ==> acc(msg)
//@ ensures  c != nil ==> Inv(c)
//@ ensures  msg != nil ==> acc(msg)
func Send(c *Core, msg []byte) {
  ...
}
```

Soundness

```
func main() {
  //@ L := {}; I := {}

  c := Constructor()
  //@ I = I ∪ {c}

  a := []byte{}
  //@ L = L ∪ {a}

  //@ I = I \ {c}

  Send(c, a)


}
```

```
//@ ensures  Inv(c)
func Constructor() (c *Core) {
  ...
}
```

```
//@ requires c != nil ==> Inv(c)
//@ requires msg != nil ==> acc(msg)
//@ ensures  c != nil ==> Inv(c)
//@ ensures  msg != nil ==> acc(msg)
func Send(c *Core, msg []byte) {
  ...
}
```

Soundness

```
func main() {
  //@ L := {}; I := {}

  c := Constructor()
  //@ I = I ∪ {c}

  a := []byte{}
  //@ L = L ∪ {a}

  //@ I = I \ {c}
  //@ L = L \ {a}
  Send(c, a)


}
```

```
//@ ensures  Inv(c)
func Constructor() (c *Core) {
  ...
}
```

```
//@ requires c != nil ==> Inv(c)
//@ requires msg != nil ==> acc(msg)
//@ ensures  c != nil ==> Inv(c)
//@ ensures  msg != nil ==> acc(msg)
func Send(c *Core, msg []byte) {
  ...
}
```

Soundness

```
func main() {
  //@ L := {}; I := {}

  c := Constructor()
  //@ I = I ∪ {c}

  a := []byte{}
  //@ L = L ∪ {a}

  //@ I = I \ {c}
  //@ L = L \ {a}
  Send(c, a)
  //@ I = I ∪ {c}

}
```

```
//@ ensures  Inv(c)
func Constructor() (c *Core) {
  ...
}
```

```
//@ requires c != nil ==> Inv(c)
//@ requires msg != nil ==> acc(msg)
//@ ensures  c != nil ==> Inv(c)
//@ ensures  msg != nil ==> acc(msg)
func Send(c *Core, msg []byte) {
  ...
}
```

Soundness

```
func main() {
  //@ L := {}; I := {}

  c := Constructor()
  //@ I = I ∪ {c}

  a := []byte{}
  //@ L = L ∪ {a}

  //@ I = I \ {c}
  //@ L = L \ {a}
  Send(c, a)
  //@ I = I ∪ {c}
  //@ L = L ∪ {a}
}
```

```
//@ ensures  Inv(c)
func Constructor() (c *Core) {
  ...
}
```

```
//@ requires c != nil ==> Inv(c)
//@ requires msg != nil ==> acc(msg)
//@ ensures  c != nil ==> Inv(c)
//@ ensures  msg != nil ==> acc(msg)
func Send(c *Core, msg []byte) {
  ...
}
```

Soundness

```
func main() {
  //@ L := {}; I := {}

  c := Constructor()
  //@ I = I ∪ {c}

  a := []byte{}
  //@ L = L ∪ {a}

  //@ I = I \ {c}
  //@ L = L \ {a}
  Send(c, a)
  //@ I = I ∪ {c}
  //@ L = L ∪ {a}
}
```

```
//@ ensures  Inv(c)
func Constructor() (c *Core) {
  ...
}
```

```
//@ requires c != nil ==> Inv(c)
//@ requires msg != nil ==> acc(msg)
//@ ensures  c != nil ==> Inv(c)
//@ ensures  msg != nil ==> acc(msg)
func Send(c *Core, msg []byte) {
  ...
}
```

Program invariant $\texttt{pinv} \stackrel{\text{def}}{=} (\forall a \in L.\ \texttt{acc}(a)) \bigstar (\forall c \in I.\ \texttt{Inv}(c))$

Soundness

Program invariant $\texttt{pinv} \stackrel{\text{def}}{=} (\forall \mathsf{a} \in \mathtt{L}.\ \mathtt{acc}(\mathsf{a})) \bigstar (\forall \mathsf{c} \in \mathtt{I}.\ \mathsf{Inv}(\mathsf{c}))$

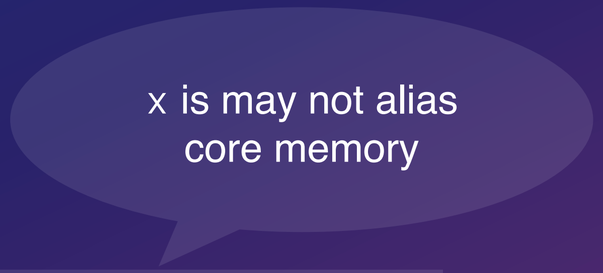
Soundness

Program invariant $\texttt{pinv} \stackrel{\text{def}}{=} (\forall a \in L.\ \texttt{acc}(a)) \bigstar (\forall c \in I.\ \texttt{Inv}(c))$

$$\frac{x \in L \cup G}{\vdash \{\texttt{pinv}\}\, \mathbb{G}(*x := e)\, \{\texttt{pinv}\}}$$

Soundness

Program invariant $\texttt{pinv} \triangleq (\forall a \in L.\ \texttt{acc}(a)) \bigstar (\forall c \in I.\ \texttt{Inv}(c))$



$$\frac{x \in L \cup G}{\vdash \{\texttt{pinv}\}\, \mathbb{G}(*x := e)\, \{\texttt{pinv}\}}$$

Soundness

Program invariant $\texttt{pinv} \stackrel{\text{def}}{=} (\forall \texttt{a} \in \texttt{L}.\ \texttt{acc(a)}) \bigstar (\forall \texttt{c} \in \texttt{I}.\ \texttt{Inv(c)})$

x is may not alias core memory

$$\frac{\texttt{x} \in \texttt{L} \cup \texttt{G}}{\vdash \{\texttt{pinv}\}\,\mathbb{G}(\texttt{*x := e})\,\{\texttt{pinv}\}}$$

$$\frac{\texttt{a} \in \texttt{L} \qquad \texttt{c} \in \texttt{I}}{\vdash \{\texttt{pinv}\}\,\mathbb{G}(\texttt{Send(c, a)})\,\{\texttt{pinv}\}}$$

Soundness

Program invariant $\texttt{pinv} \triangleq (\forall a \in L.\ \texttt{acc}(a)) \bigstar (\forall c \in I.\ \texttt{Inv}(c))$

x is may not alias core memory

$$\frac{x \in L \cup G}{\vdash \{\texttt{pinv}\}\,\mathbb{G}(\texttt{*x := e})\,\{\texttt{pinv}\}}$$

a is may not alias core memory & must not escape current thread

$$\frac{a \in L \qquad c \in I}{\vdash \{\texttt{pinv}\}\,\mathbb{G}(\texttt{Send(c, a)})\,\{\texttt{pinv}\}}$$

Soundness

Program invariant $\texttt{pinv} \triangleq (\forall \texttt{a} \in \texttt{L}.\ \texttt{acc(a)}) \bigstar (\forall \texttt{c} \in \texttt{I}.\ \texttt{Inv(c)})$

x is may not alias core memory

a is may not alias core memory & must not escape current thread

c is must not escape current thread

$$\frac{\texttt{x} \in \texttt{L} \cup \texttt{G}}{\vdash \{\texttt{pinv}\}\mathbb{G}(\texttt{*x := e})\{\texttt{pinv}\}}$$

$$\frac{\texttt{a} \in \texttt{L} \qquad \texttt{c} \in \texttt{I}}{\vdash \{\texttt{pinv}\}\mathbb{G}(\texttt{Send(c, a)})\{\texttt{pinv}\}}$$

Soundness

Whole codebase proof

Soundness

Whole codebase proof

Separation logic proof for the whole codebase with side-conditions dischargeable by static analyses

Soundness

Whole codebase proof

Separation logic proof for the whole codebase with side-conditions dischargeable by static analyses

$$\frac{\text{side-conditions}}{\vdash \{\,\texttt{pinv}\,\}\,\mathbb{G}(\texttt{prog})\,\{\,\texttt{pinv}\,\}}$$

Soundness

Whole codebase proof

Separation logic proof for the whole codebase with side-conditions dischargeable by static analyses

$$\frac{\text{side-conditions}}{\vdash \{\texttt{pinv}\}\,\mathbb{G}(\texttt{prog})\,\{\texttt{pinv}\}}$$

ghost code tracking heap locations, core invariant, and I/O specification

Soundness

Whole codebase proof

Separation logic proof for the whole codebase with side-conditions dischargeable by static analyses

$$\frac{\text{side-conditions}}{\vdash \{\,\texttt{pinv}\,\}\,\mathbb{G}(\texttt{prog})\,\{\,\texttt{pinv}\,\}}$$

ghost code tracking heap locations, core invariant, and I/O specification

I/O independence

Soundness

Whole codebase proof

Separation logic proof for the whole codebase with side-conditions dischargeable by static analyses

$$\frac{\text{side-conditions}}{\vdash \{\,\texttt{pinv}\,\}\,\mathbb{G}(\texttt{prog})\,\{\,\texttt{pinv}\,\}}$$

ghost code tracking heap locations, core invariant, and I/O specification

I/O independence

Secret-independent I/O operations refine the symbolic attacker

Evaluation — AWS SSM Agent

Tamarin

gobra

Argot

Model	
Core Refinement	
I/O Independence	Argot
Core Assumptions	Argot

Evaluation — AWS SSM Agent

		LoC	Proof annotations	Verification time [min]	Effort
 Tamarin	Model	319	75	3.3	< 2 pms
gobra	Core Refinement	749	2761 + 1064 (generated)	1.2	< 3 pms
Argot	I/O Independence	~105k		0.5	< 0.5 pms
	Core Assumptions	~105k		2.2	< 1.5 pms

Conclusions

Implements protocol

Codebase

Application

Core

1k LOC

100k+ LOC

⊆

⊆

Tamarin model

<I/O spec/>

Tamarin

gobra

Argot

Conclusions

Implements protocol

Codebase

Application

Core

1k LOC

⊆

⊆

Tamarin model

<I/O spec/>

Codebase may perform additional *independent* I/O operations

100k+ LOC

Tamarin

gobra

Argot

Conclusions

Implements protocol

Codebase

Application

Core

1k LOC

⊆

⊆

Tamarin model

<I/O spec/>

Static analyses enforce that Application uses Core correctly

Codebase may perform additional *independent* I/O operations

100k+ LOC

Tamarin

gobra

Argot