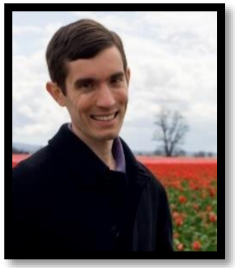
Ironclad Apps: End-to-End Security via Automated Full-System Verification

Chris Hawblitzel

Jon Howell

Jay Lorch

Arjun Narayan



Bryan Parno

Danfeng Zhang

Brian Zill

Microsoft Research

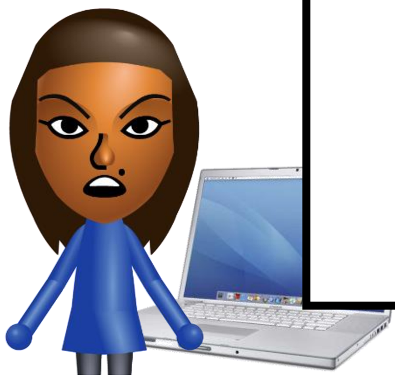
CHASE

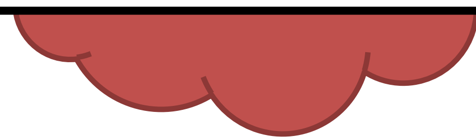
Online and Mobile Security

- Chase Online, the Chase Mobile app and the Chase Mobile website use Secure Socket Layer (SSL) technology

…

- We periodically review our operations and business practices to make sure they comply with the corporate policies and procedures we follow to protect confidential information

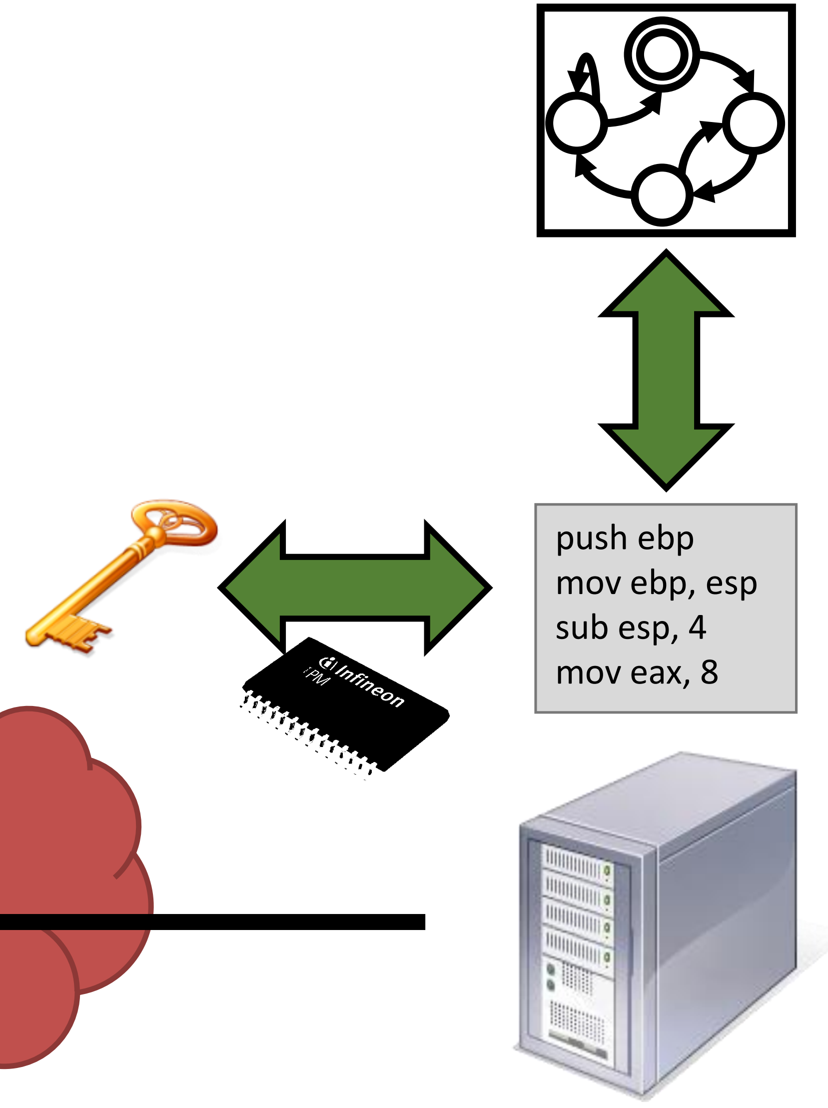


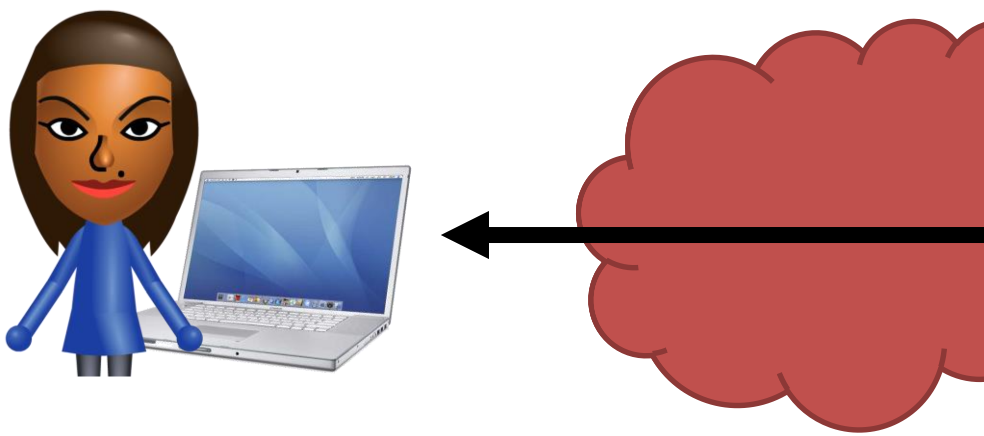


An *Ironclad app* guarantees to remote parties that every instruction it executes adheres to a high-level security spec.

Ironclad combines:

- Late launch
- Trusted Computing
- Software verification

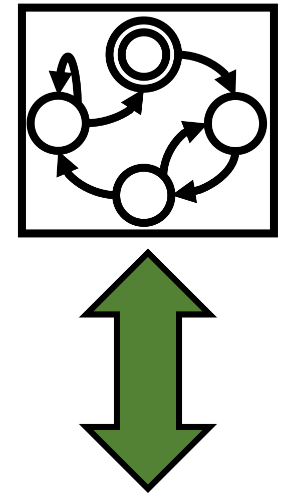


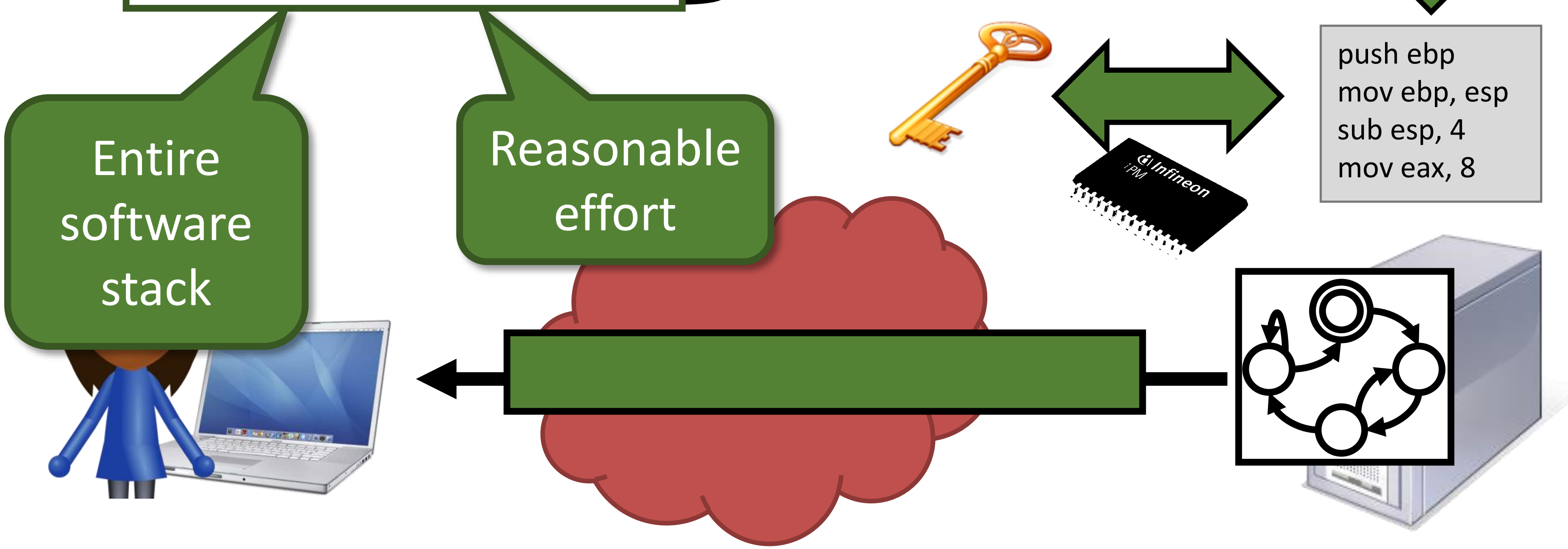


Ironclad combines:

- Late launch
- Trusted Computing
- Software verification

Secure Remote Equivalence



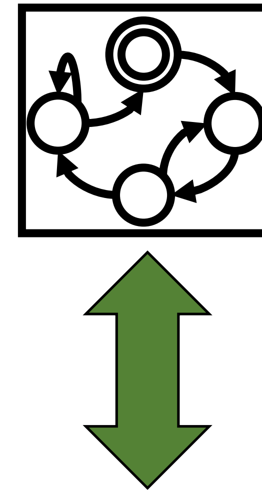


Verification implies:

- No buffer overflows
- No code in
- No type-sa
- No informa
- No crypto implementation flaws

…

We always know what the app will do with private data!



```
push ebp
mov ebp, esp
sub esp, 4
mov eax, 8
```

We don't prove:

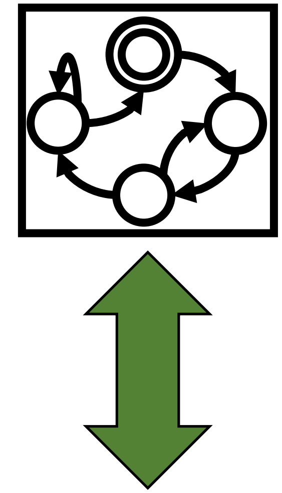
- Absence of side channels
- Liveness
- Physical security

Verification goals

- End-to-end security
  - Complete
  - Low-level
- Rapid development

- Non-goal: Verify existing code

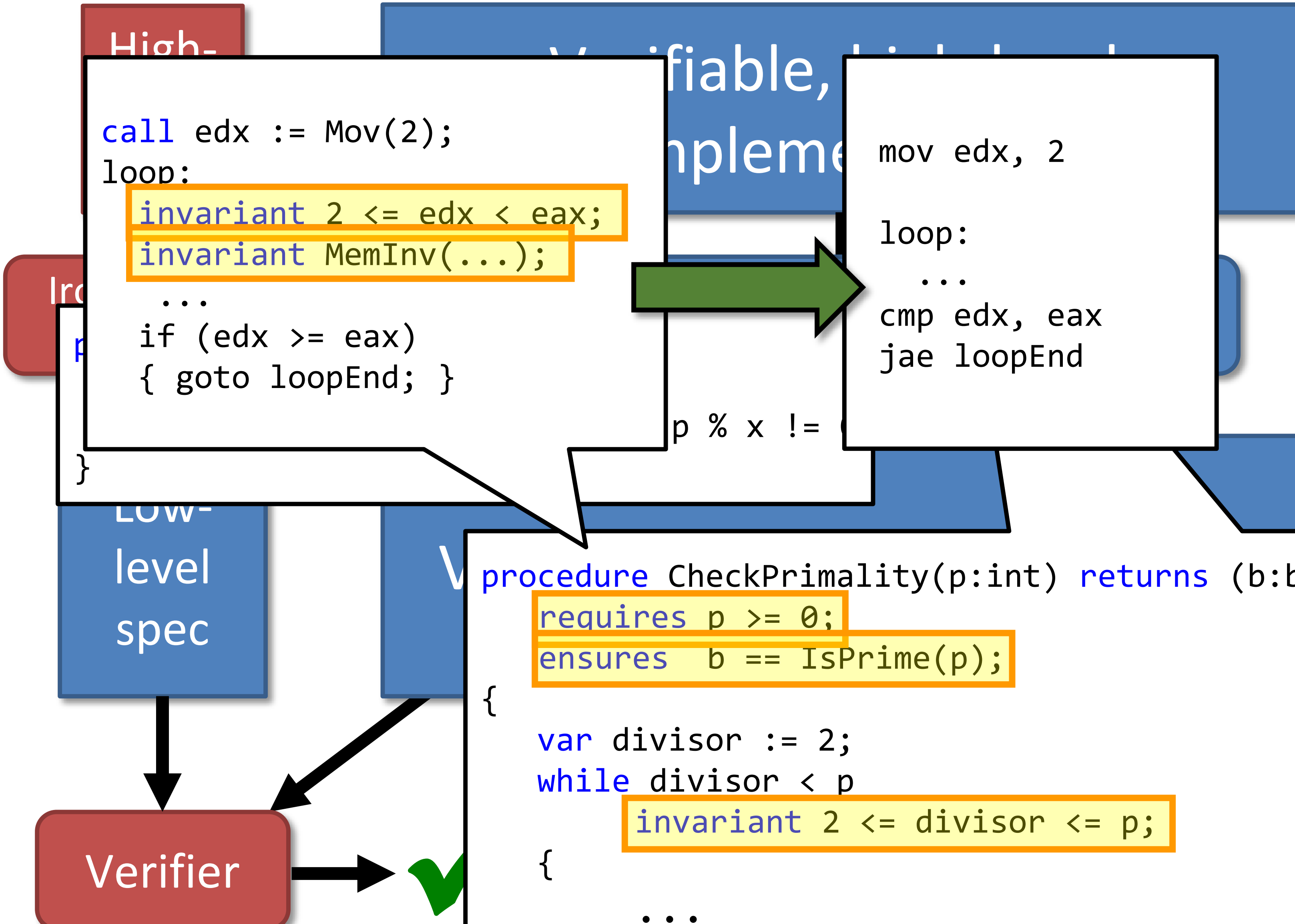
- Long-term: Performance matches unsafe code



```
push ebp
mov ebp, esp
sub esp, 4
mov eax, 8
```

Verification methodology

= Trusted
= Untrusted

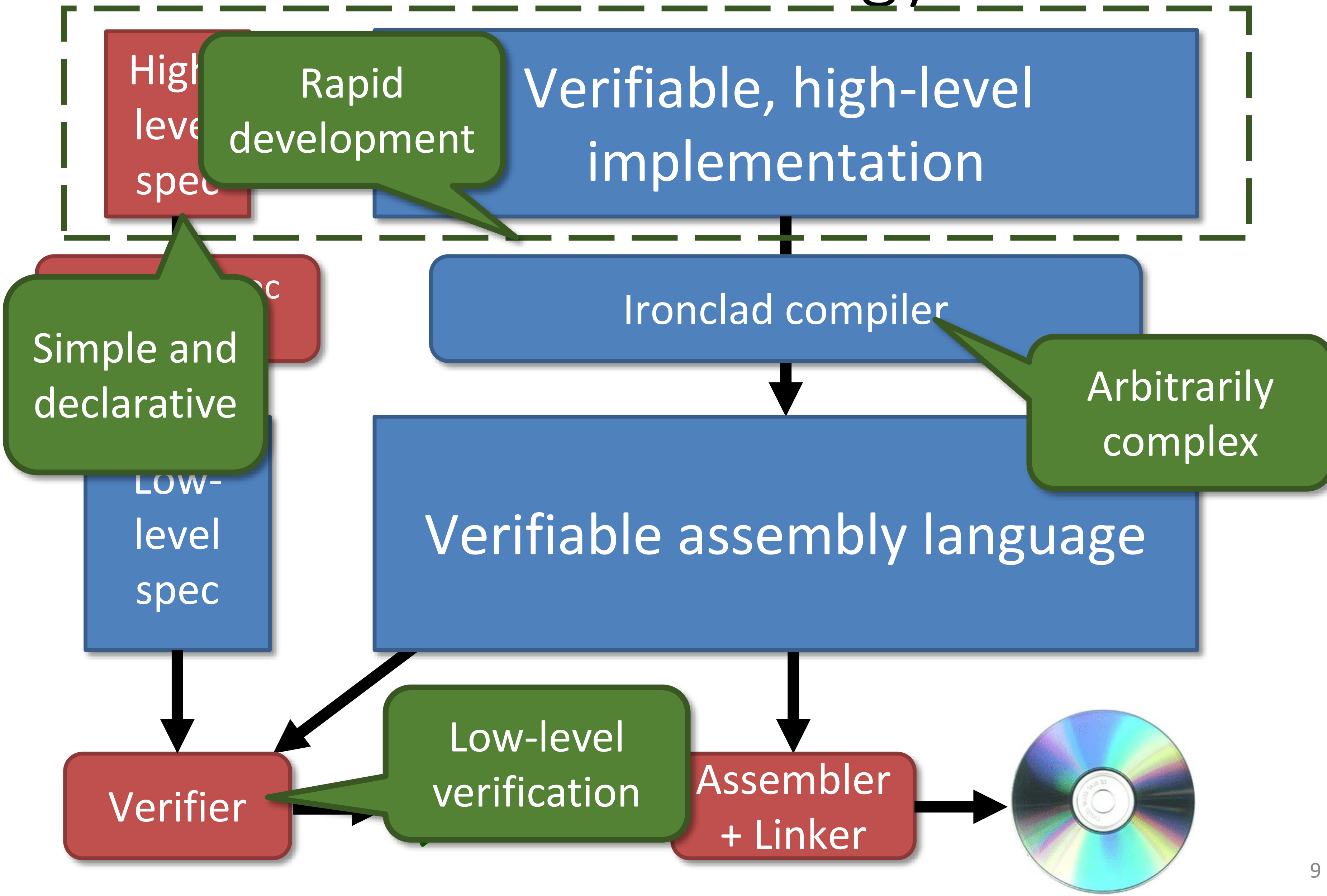



```
call edx := Mov(2);
loop:
  invariant 2 <= edx < eax;
  invariant MemInv(...);
   ...
  if (edx >= eax)
  { goto loopEnd; }
```

```
mov edx, 2

loop:
  ...
cmp edx, eax
jae loopEnd
```

```
procedure CheckPrimality(p:int) returns (b:bool)
   requires p >= 0;
   ensures  b == IsPrime(p);
{
   var divisor := 2;
   while divisor < p
        invariant 2 <= divisor <= p;
   {
        ...
```

Verification methodology: Benefits

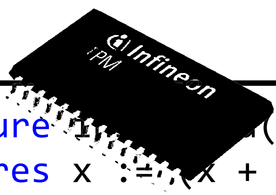


Writing trustworthy specifications

Idiomatic specification

= 3439 pages

1,364 lines of spec
(< 60 instructions)



= 795 pages

296 lines of spec
(secure randomness
+ attestation)

```
procedure i   (..., x:reg, y:reg)
  ensures x := (x + y) % 0x100000000;
  ...
```

```
type core = core(regs:[int]int, eip:int, ..., segments, paging, ...);
type machine = machine(cores:[int]core, mem:[int]int, io:IOState);
```

Hardware specs

Writing trustworthy specifications

Idiomatic

App spec

```
predicate ValidTransition(old_state:NotaryState,
                          new_state:NotaryState,
                          request:Request,
                          response:Response, ...)
{
  match request
```

request)

Lib specs

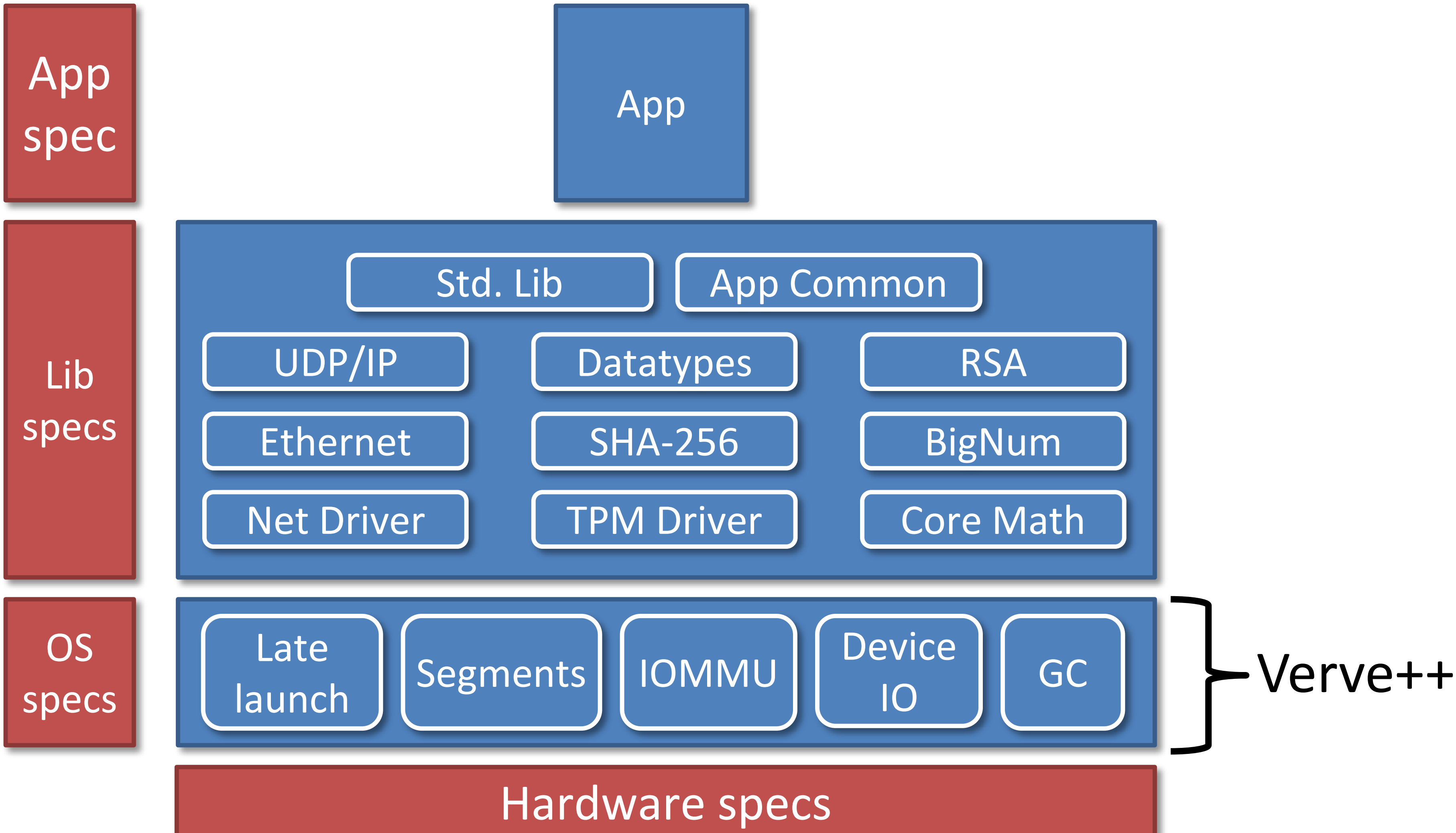
```
function SHA256(messageBits:seq<int>) : seq<int>
    requires |messageBits| < power2(64);
    requires IsBitSeq(messageBits);
{

    ...
}
```

National Institute of Standards and Technology
Gaithersburg, MD 20899-8900

OS specs

Hardware specs

Architecture



Challenge: Whole-system verification

```
procedure CheckPrimality(p:int) returns (b:bool)
   requires p >= 0;
   ensures  b == IsPrime(p);
{
   var divisor := 2;
   while divisor < p
        invariant 2 <= divisor <= p;
   {
        ...
```

Functional verification (correctness)

F(all possible inputs) → all permitted output words

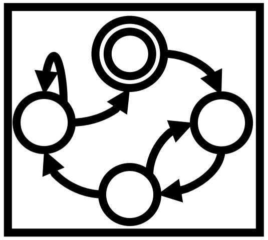
```
push ebp
mov ebp, esp
sub esp, 4
mov eax, 8
```

```
procedure Instr_outb(..., x:reg)
   requires ????
```

Solution: Relational verification

```
procedure instr_inb(..., x:reg)
    ensures public(x);
```

```
push ebp
mov ebp, esp
sub esp, 4
mov eax, 8
```



Declassifier

```
procedure instr_outb(..., x:reg)
    requires public(x);
```

Declassify X by proving the abstract app would have output X

Rapid verification

Automated tools

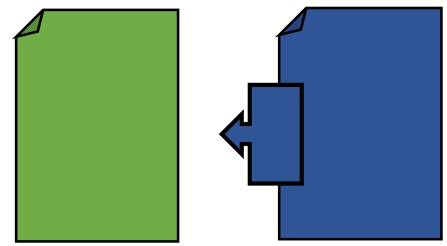
Dafny

SymDiff

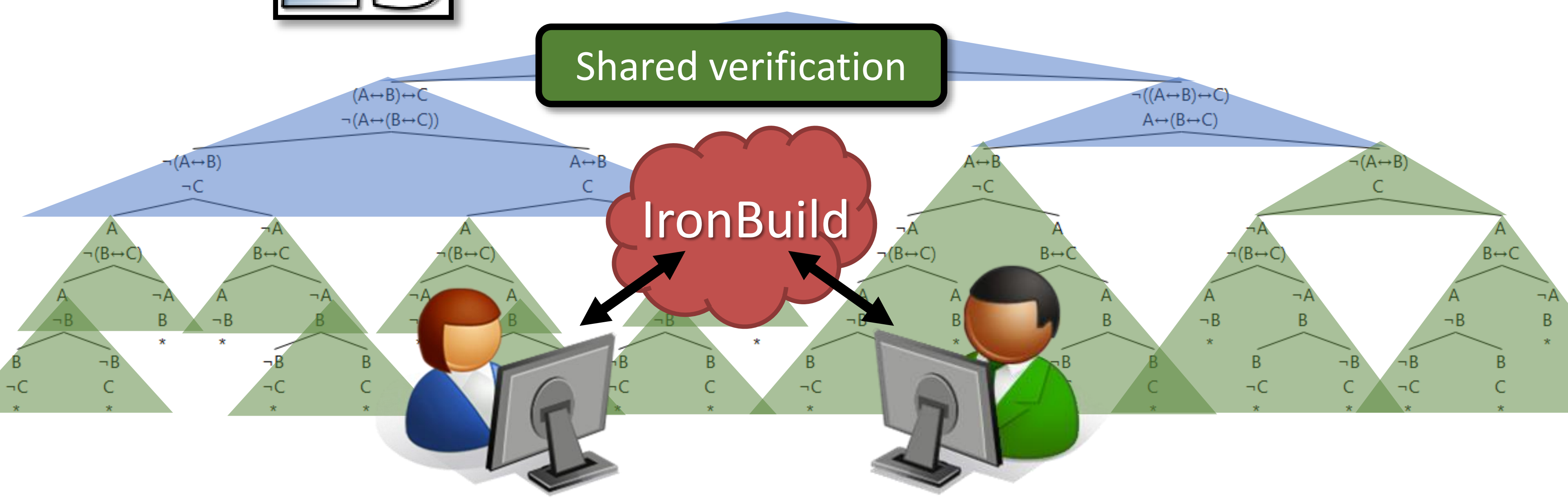
BOOGIE

Z3

Modular verification



Shared verification



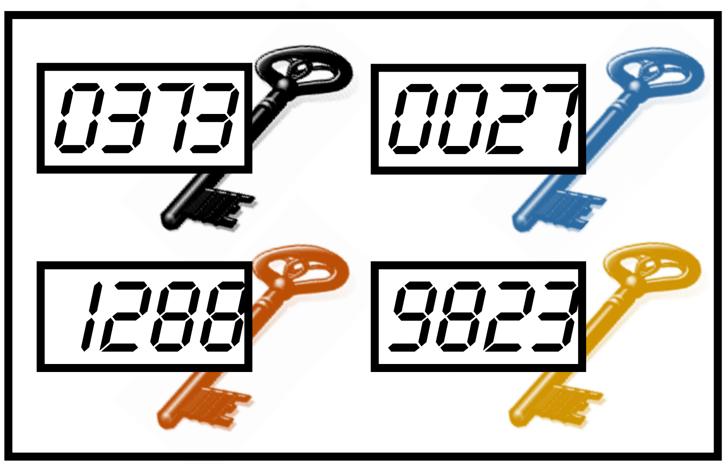
Ironclad Apps

Password Protector

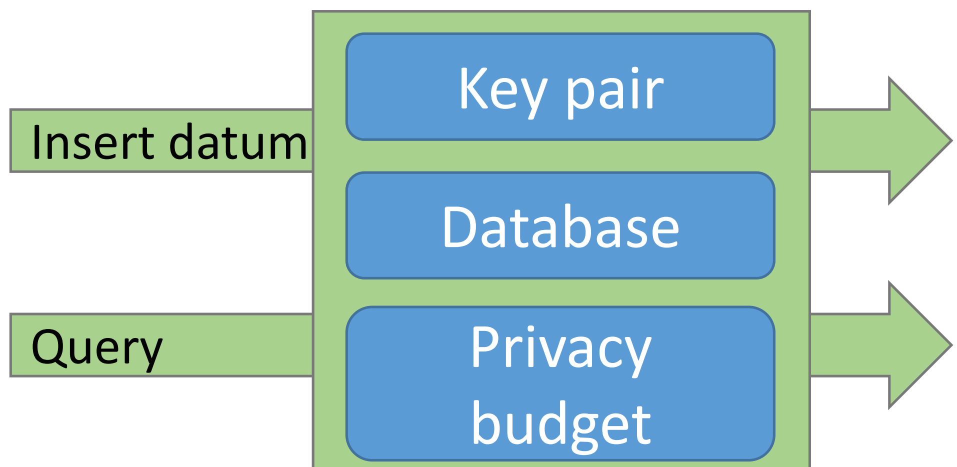
password letmein
12345 dragon
123 1111
abc seball
monk oveyou
qwerty trustno1

Notary

Trusted Incrementer



Differentially Private DB



Lessons learned

Automated ≠ Automatic

- ☺ Non-recursive functions
- ☺ Addition & subtraction
- ☺ Mul/div/mod by small constants

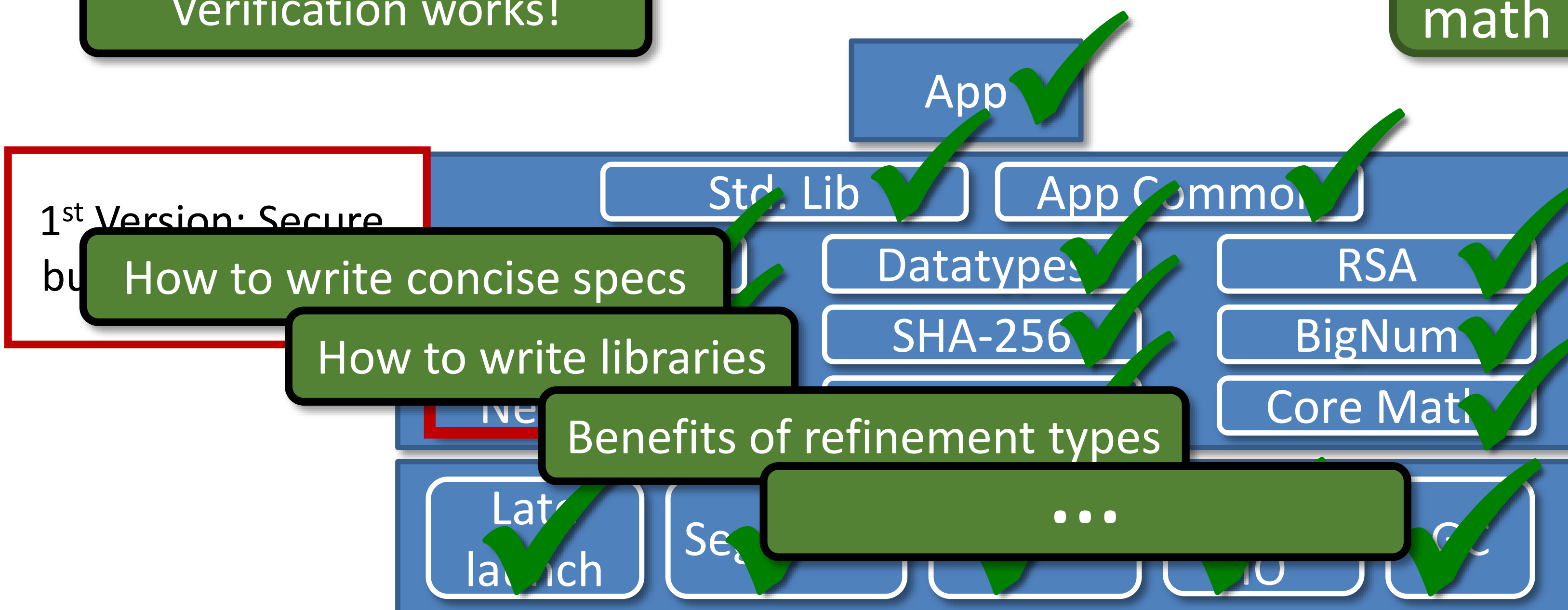
- 😐 Forall/exists
- 😐 Arrays/seqs

- ☹ Recursive functions
- ☹ General mul/div/mod

Opaque attributes

Custom math library

Verification works!



How to write concise specs

How to write libraries

Benefits of refinement types

…

Eval: Proof burden

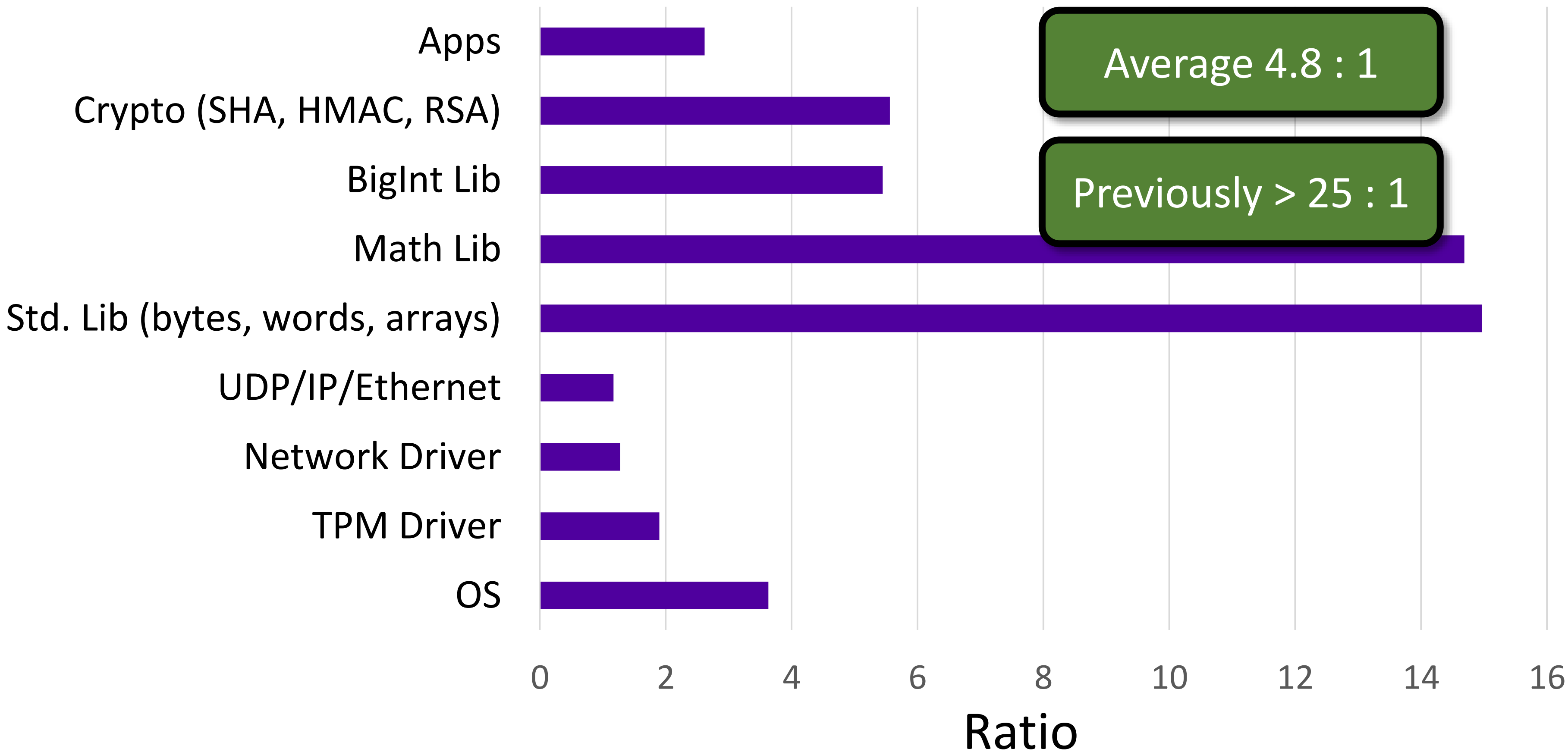
~3 person-years

Previously 22+ pys

Proof hints : Implementation LoC

Average 4.8 : 1

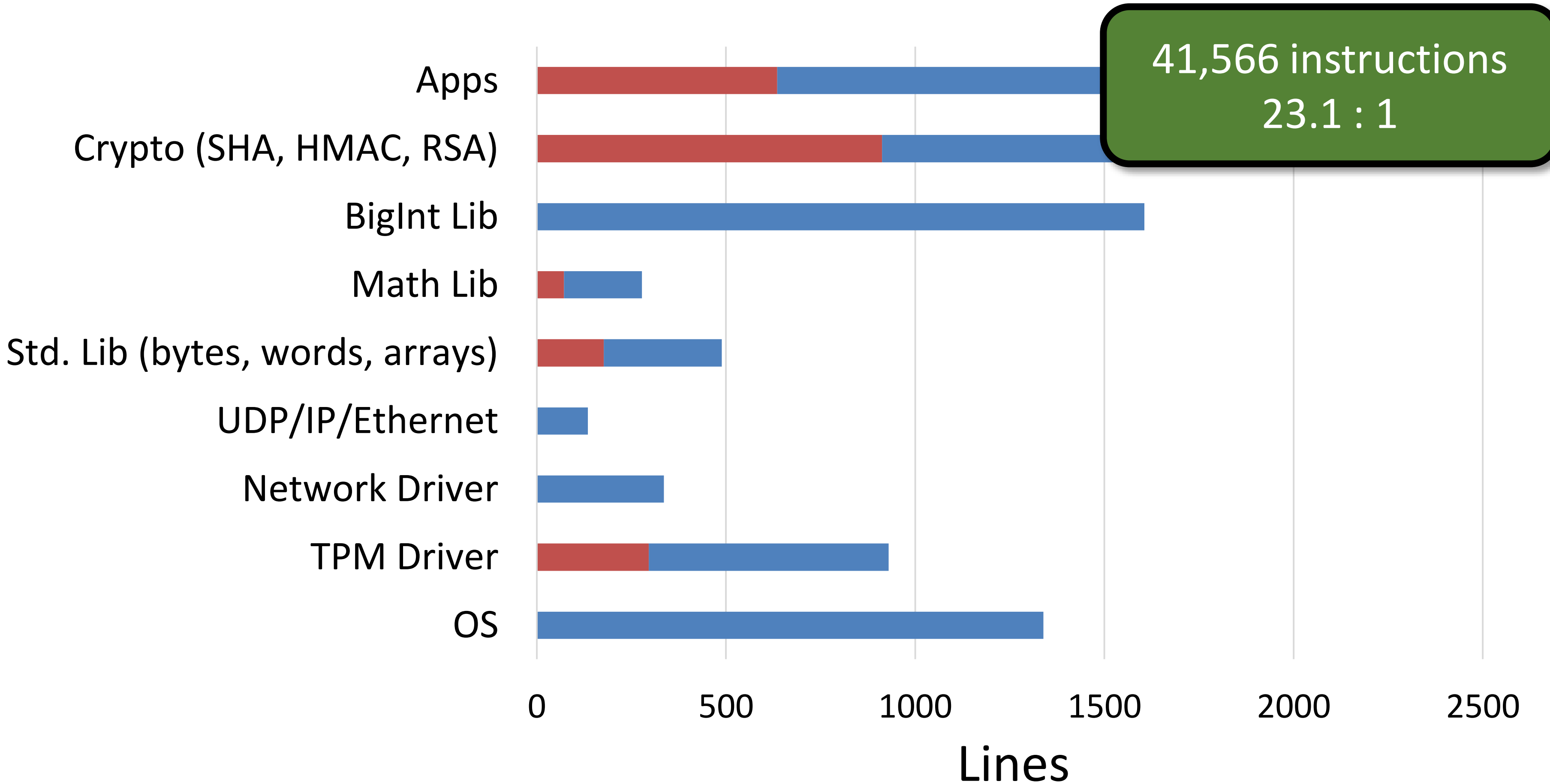
Previously > 25 : 1



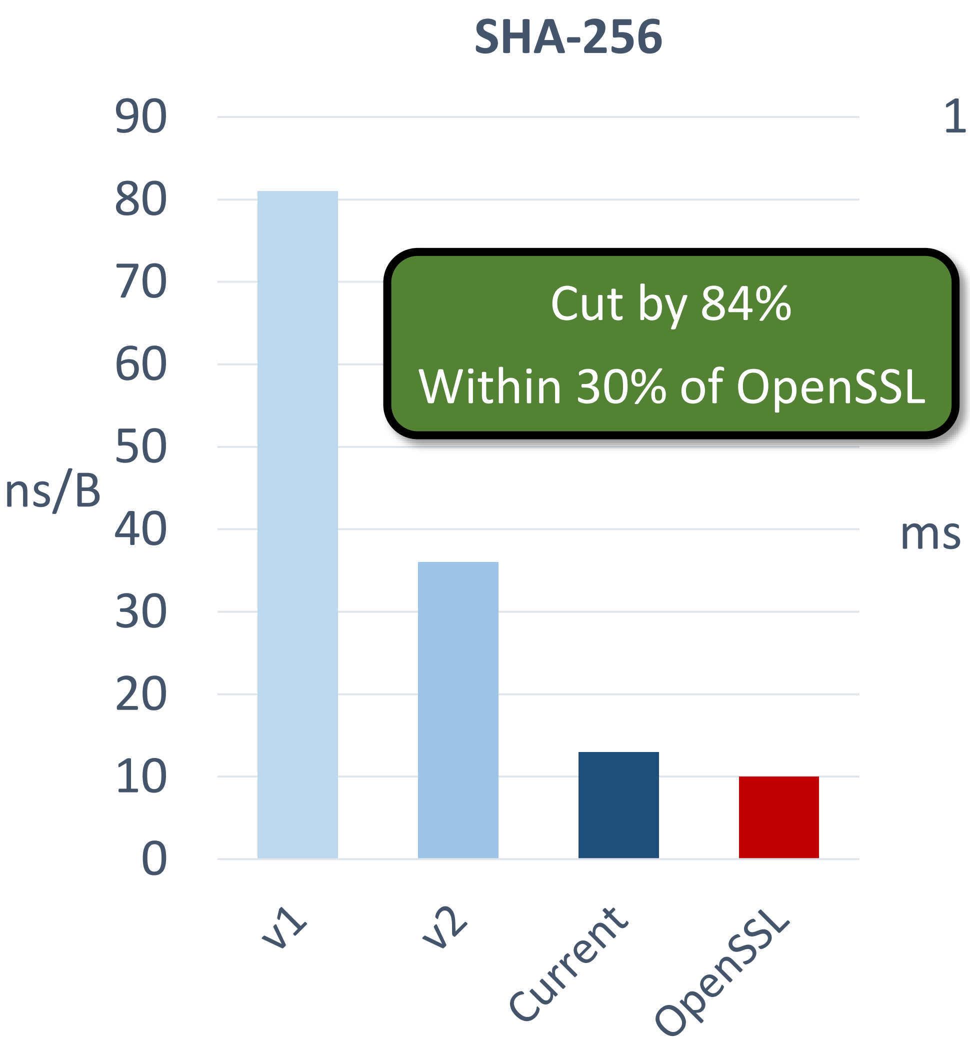
Eval: System size

	Software	Hardware
Trusted Spec	1796 LoC	1750 LoC
Implementation	6971 LoC	

SW Impl : Spec = 3.9 : 1



Eval: Performance



Related work

- Early security kernels
  - Examples: KVM/370, VAX VMM, SCOMP, GEMSOS
  - Formally specified, but no connection to implementation
- Recent verified systems
  - Examples: seL4, VCC, PROSPER, CompCert, Jitk
  - Focus on one layer
  - Many verify C code => Good performance
  - Typically less automation => More human proof burden

Conclusions

- Ironclad guarantees end-to-end security to remote parties: Every instruction meets the app's security spec
- Achieved via:
  - New and modified tools
  - A methodology for rapid verification of systems software
- Verification of systems code is quite feasible!

http://research.microsoft.com/ironclad

Thank you!

ironclad@microsoft.com