Proofcraft

seL4® verification roadmap

Rafal Kolanski, June Andronick, Gerwin Klein @Proofcraft

seL4 is a registered trademark of LF Projects, LLC

“The” seL4 Theorem(s)

different configs

different levels

seL4 keeps evolving

Verification makes seL4 unique

seL4’s formal proofs must evolve as well

Proofcraft is committed to keep this evolution alive

4

Success pushed evolution

seL4’s formal proofs evolve
with new architectures

seL4’s formal proofs evolve
with new features

Parallel verification creates challenges,
Convergence is planned

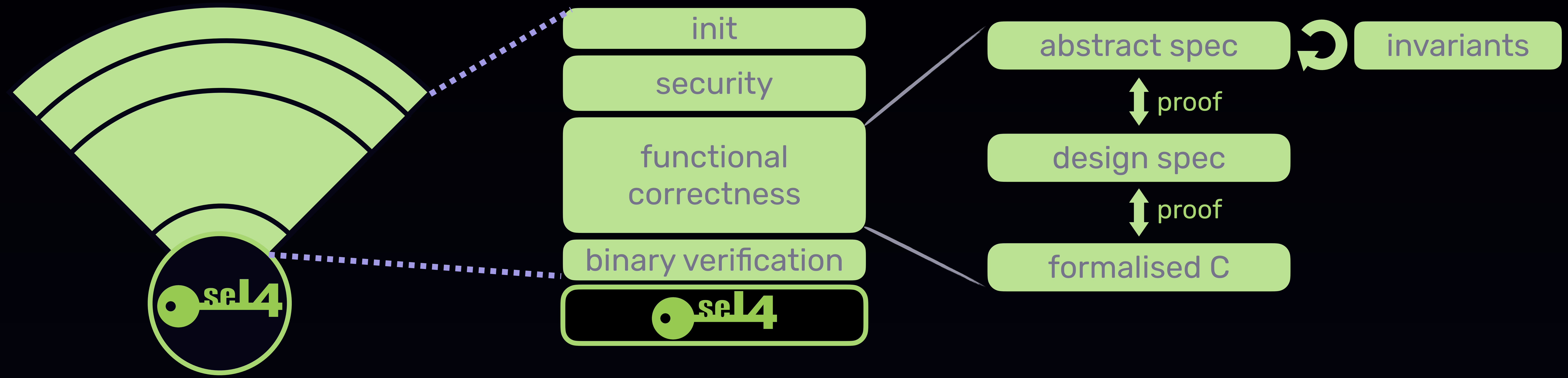
▶ Success pushed evolution

seL4’s formal proofs evolve
with new architectures

seL4’s formal proofs evolve
with new features

Parallel verification creates challenges,
Convergence is planned

Started as…



Then…

Success!

Interest!

Customers!

I want it all. And I want it now.

Photo by Nathan Dumlao on Unsplash

I want seL4 verified "with X on Y"

(It's usually what we don't have in stock :)

seL4-vanilla

seL4-MCS

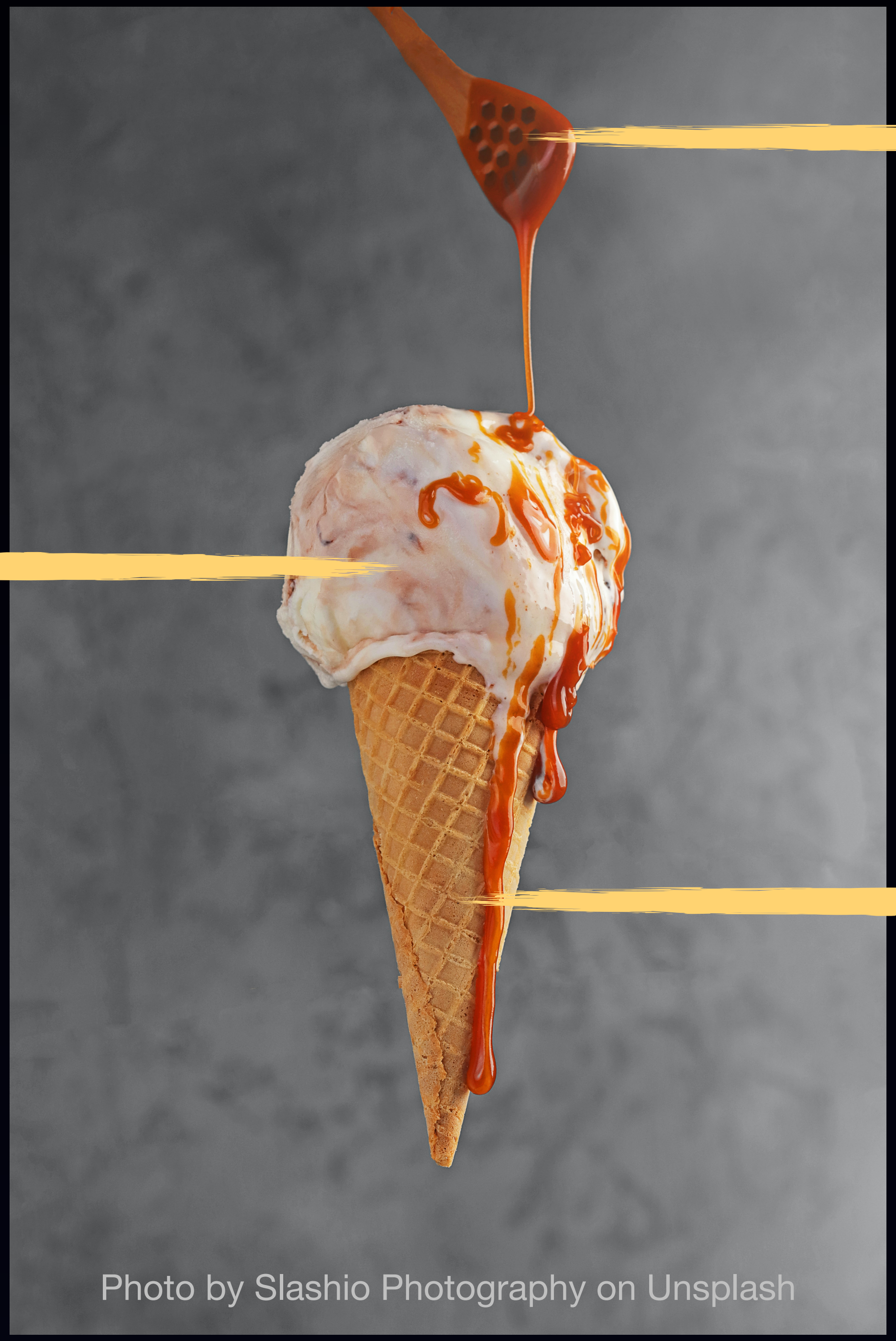


Photo by Slashio Photography on Unsplash

HYP

...

ARM	32b 64b
RISC-V	32b 64b
x86	32b 64b

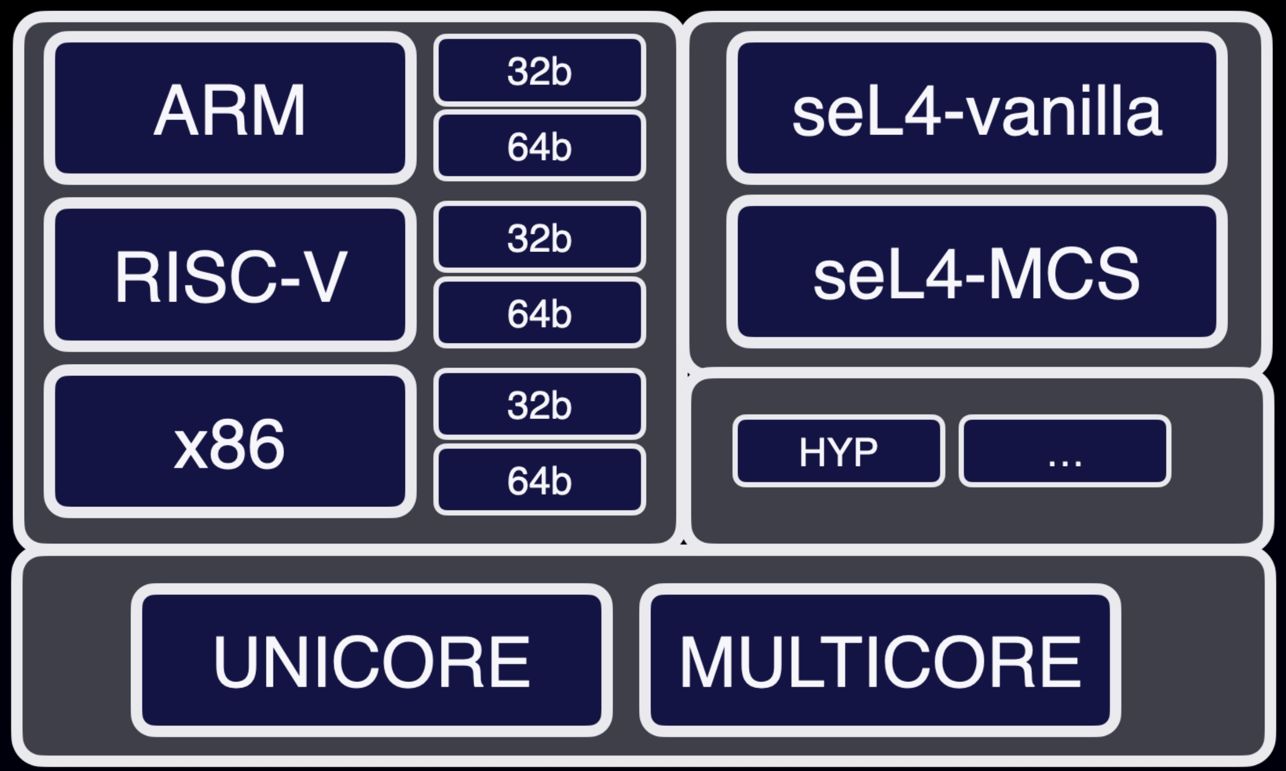
UNICORE

MULTICORE

MCS = Mixed-Criticality Systems

“The” seL4 Theorem(s)

different configs



ARM: 32b, 64b
RISC-V: 32b, 64b
x86: 32b, 64b

seL4-vanilla
seL4-MCS
HYP ...

UNICORE MULTICORE

different levels

Success pushed evolution

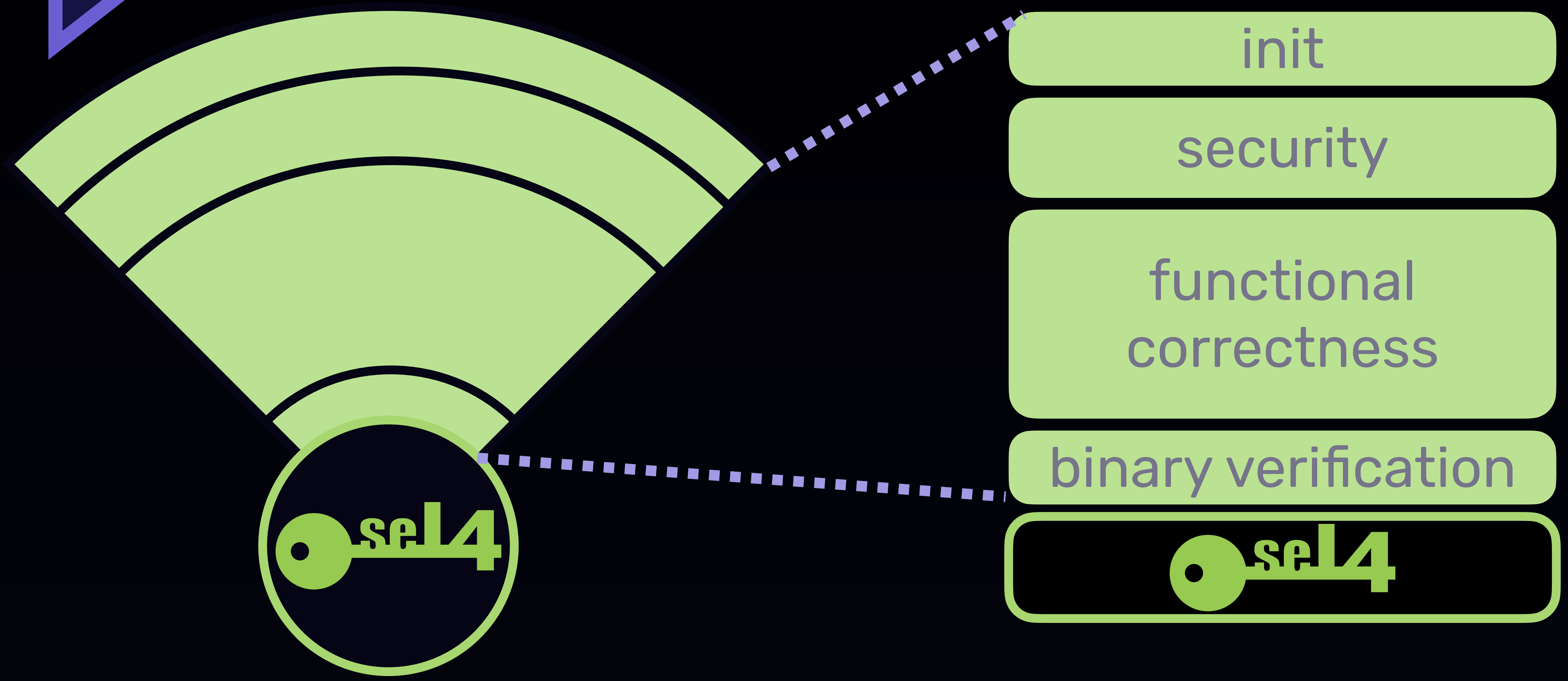
▶ seL4’s formal proofs evolve
with new architectures

seL4’s formal proofs evolve
with new features

Parallel verification creates challenges,
Convergence is planned

Started as…

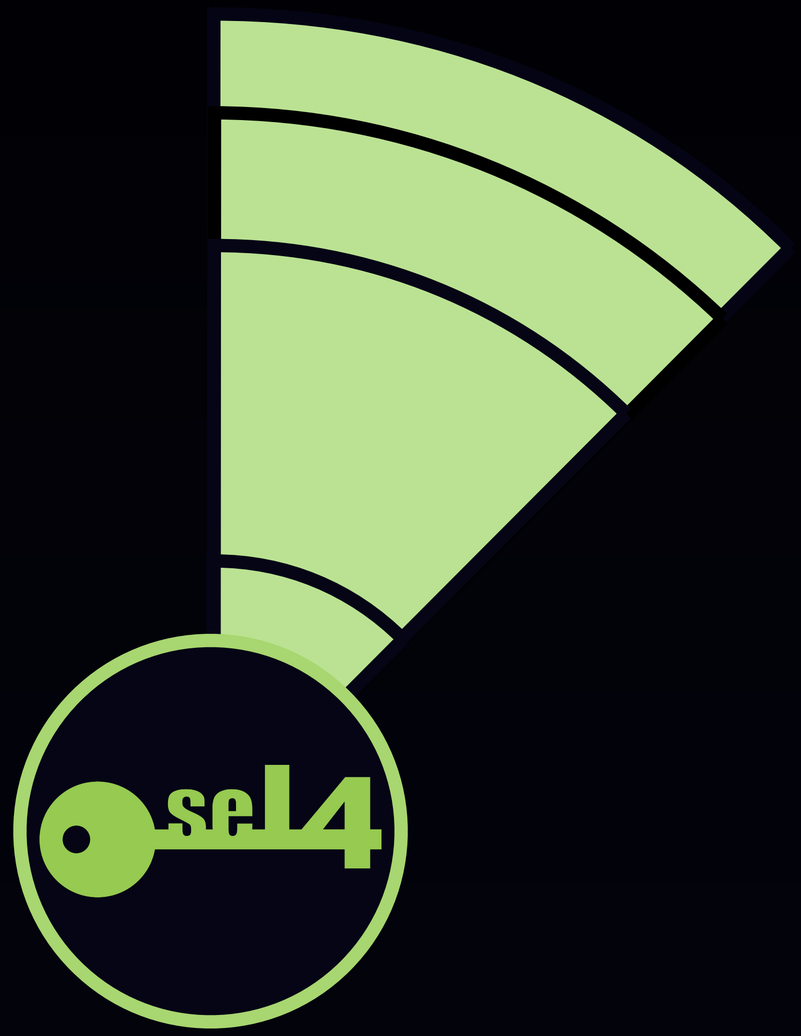
Arm 32-bit

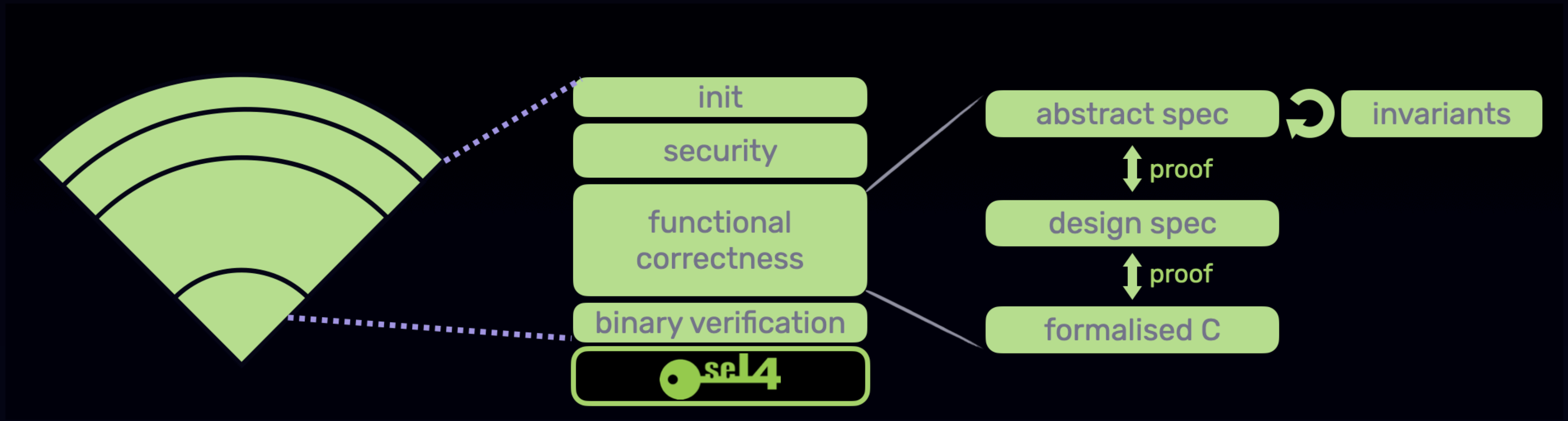


(non-MCS)
(unicore)

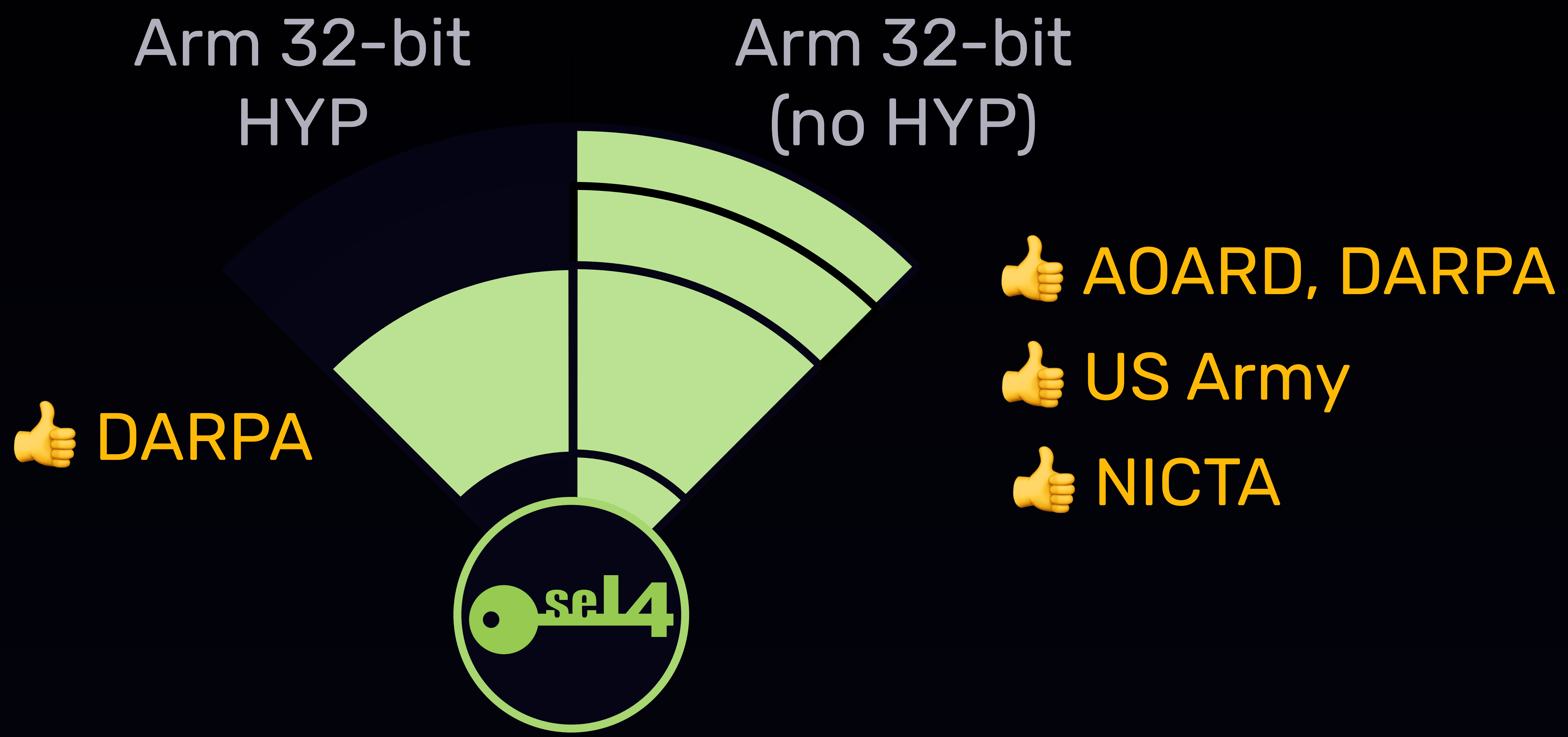
Then…

Arm 32-bit

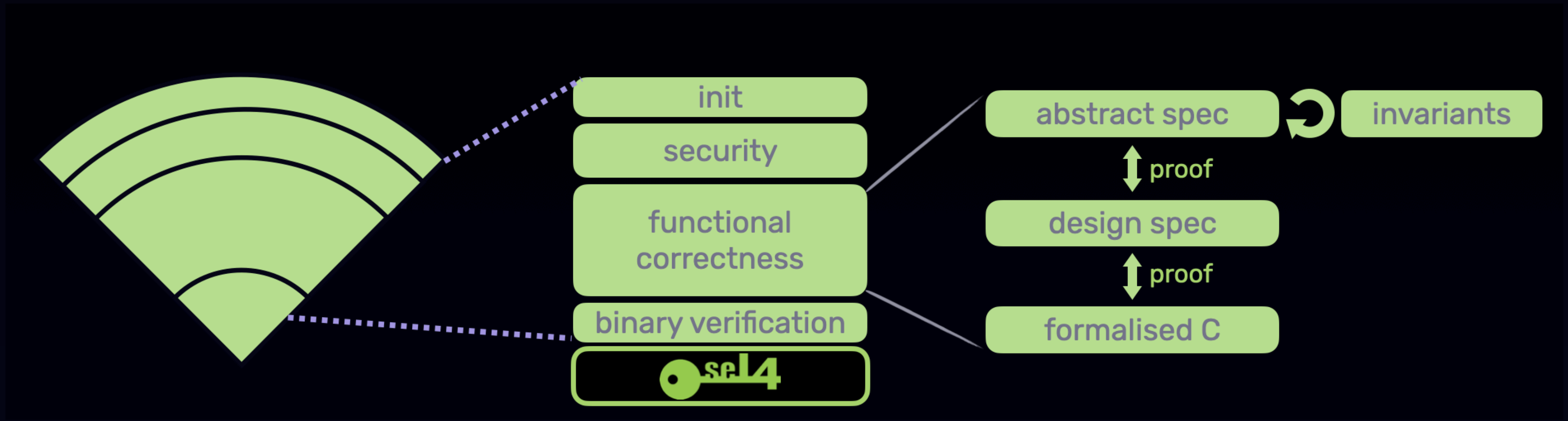




Then…

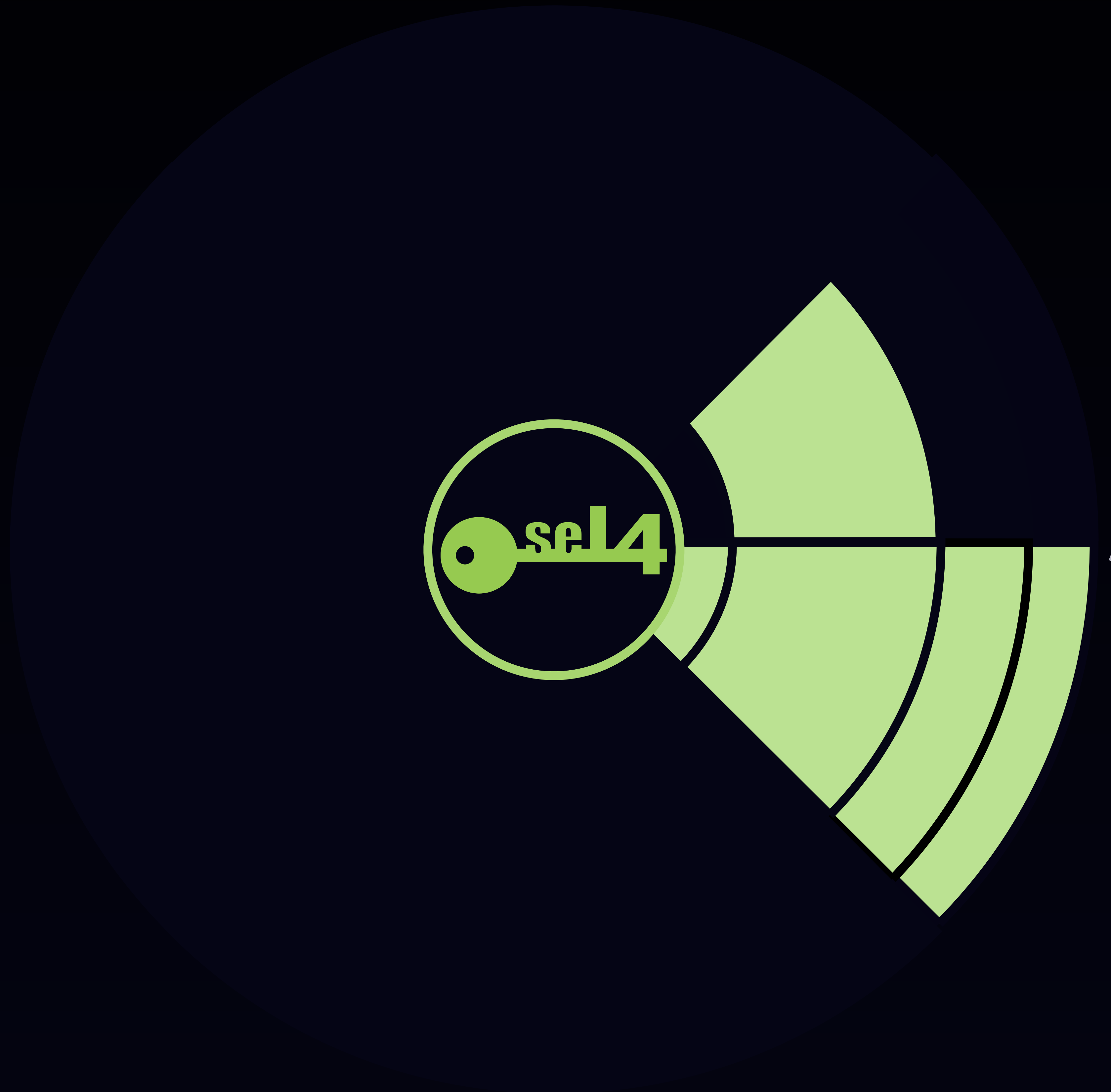


Arm 32-bit
HYP
Arm 32-bit
(no HYP)
👍 AOARD, DARPA
👍 US Army
👍 NICTA
👍 DARPA
seL4

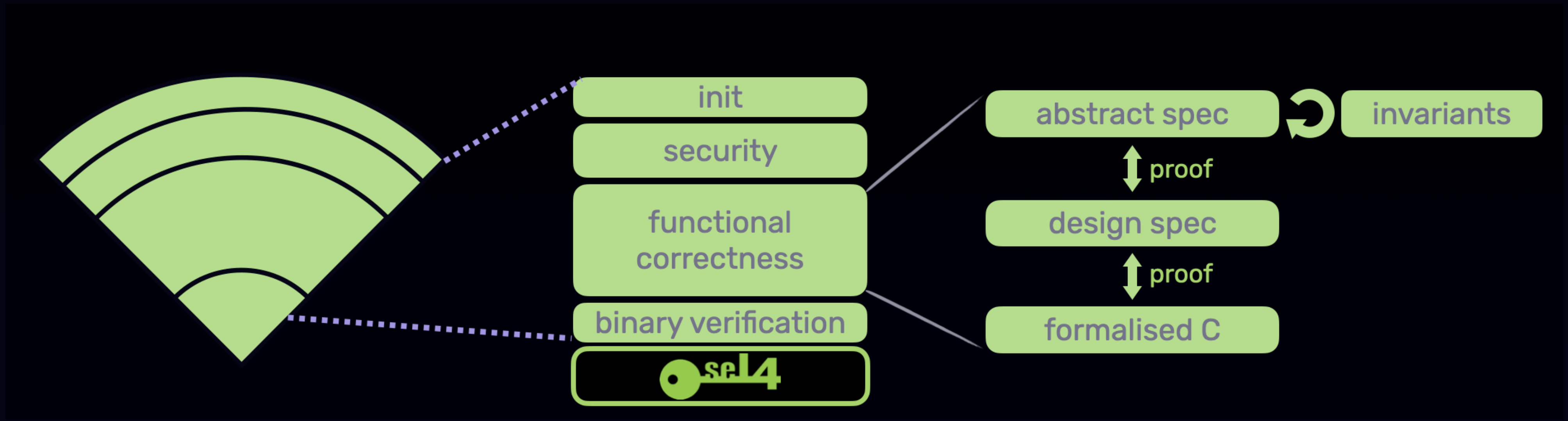


init
security
functional correctness
binary verification
seL4
abstract spec
invariants
proof
design spec
proof
formalised C

Then…



Arm 32-bit



init

security

functional correctness

binary verification

abstract spec

invariants

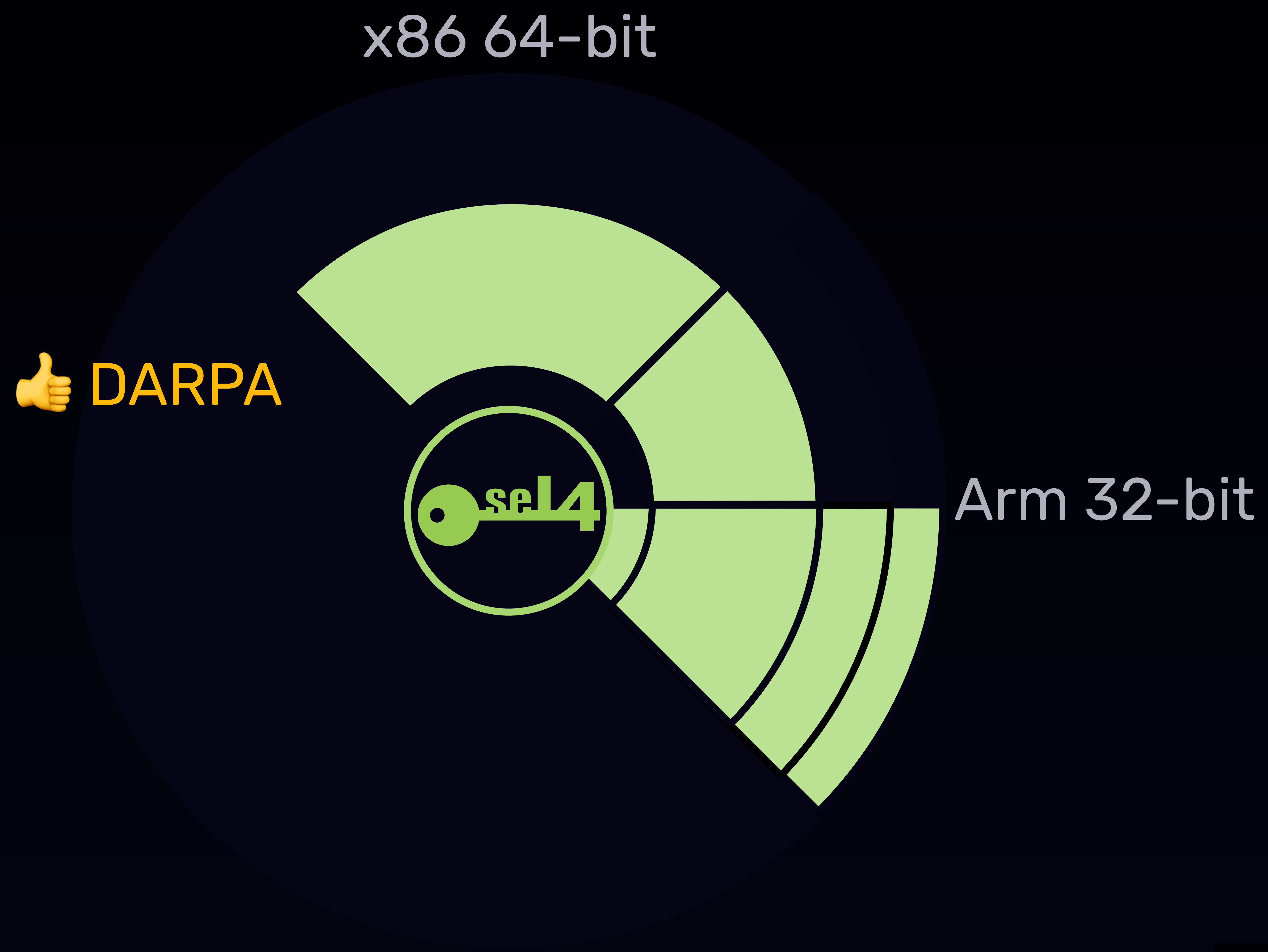
proof

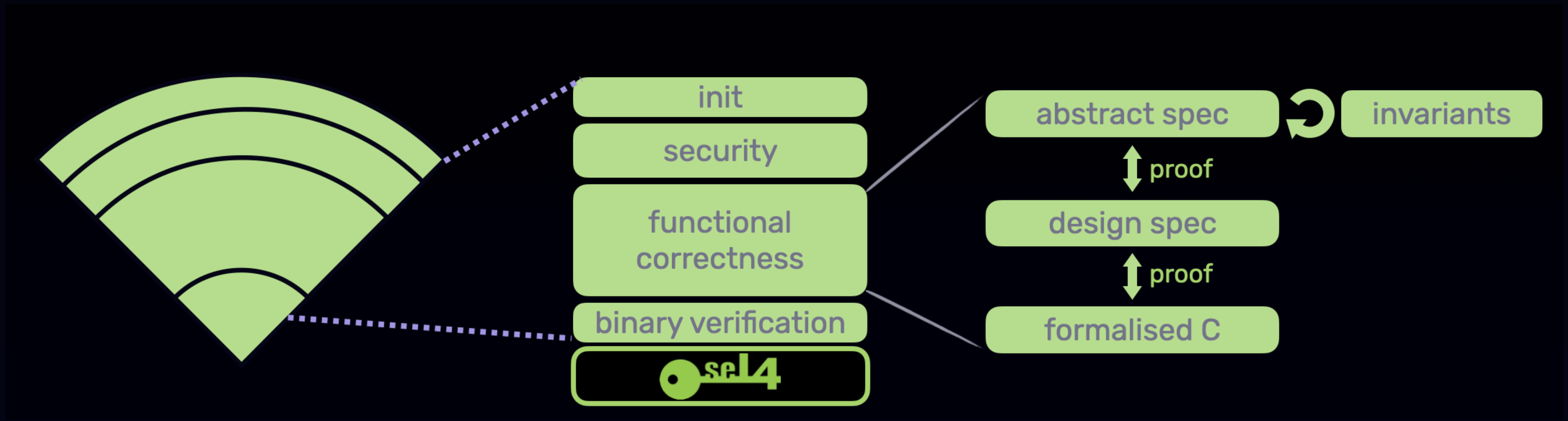
design spec

proof

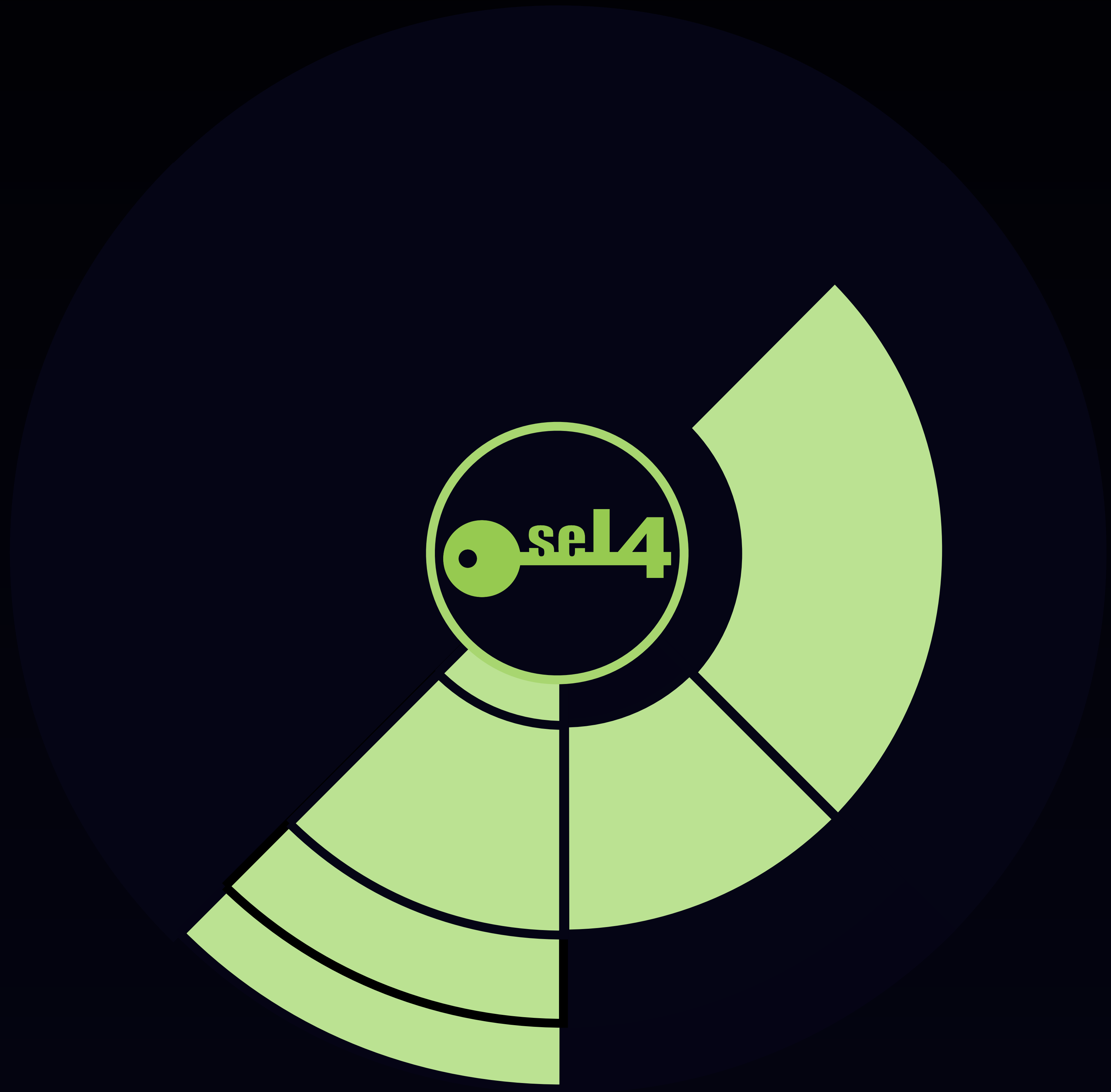
formalised C

Then…


x86 64-bit
👍 DARPA
Arm 32-bit
seL4

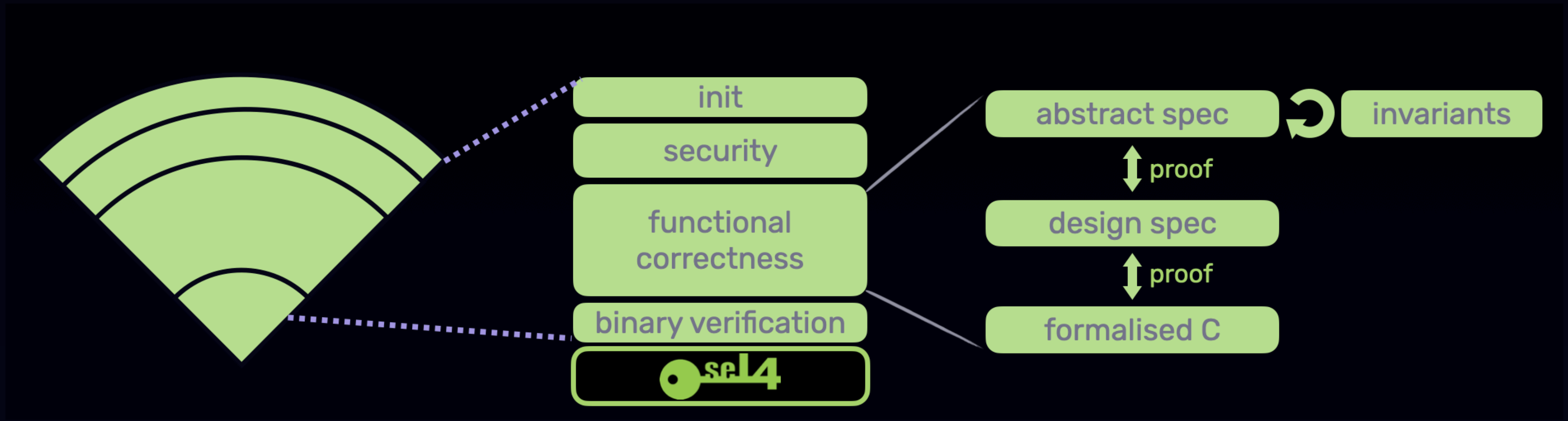

init
security
functional correctness
binary verification
seL4
abstract spec
invariants
proof
design spec
proof
formalised C

Then…



x86 64-bit

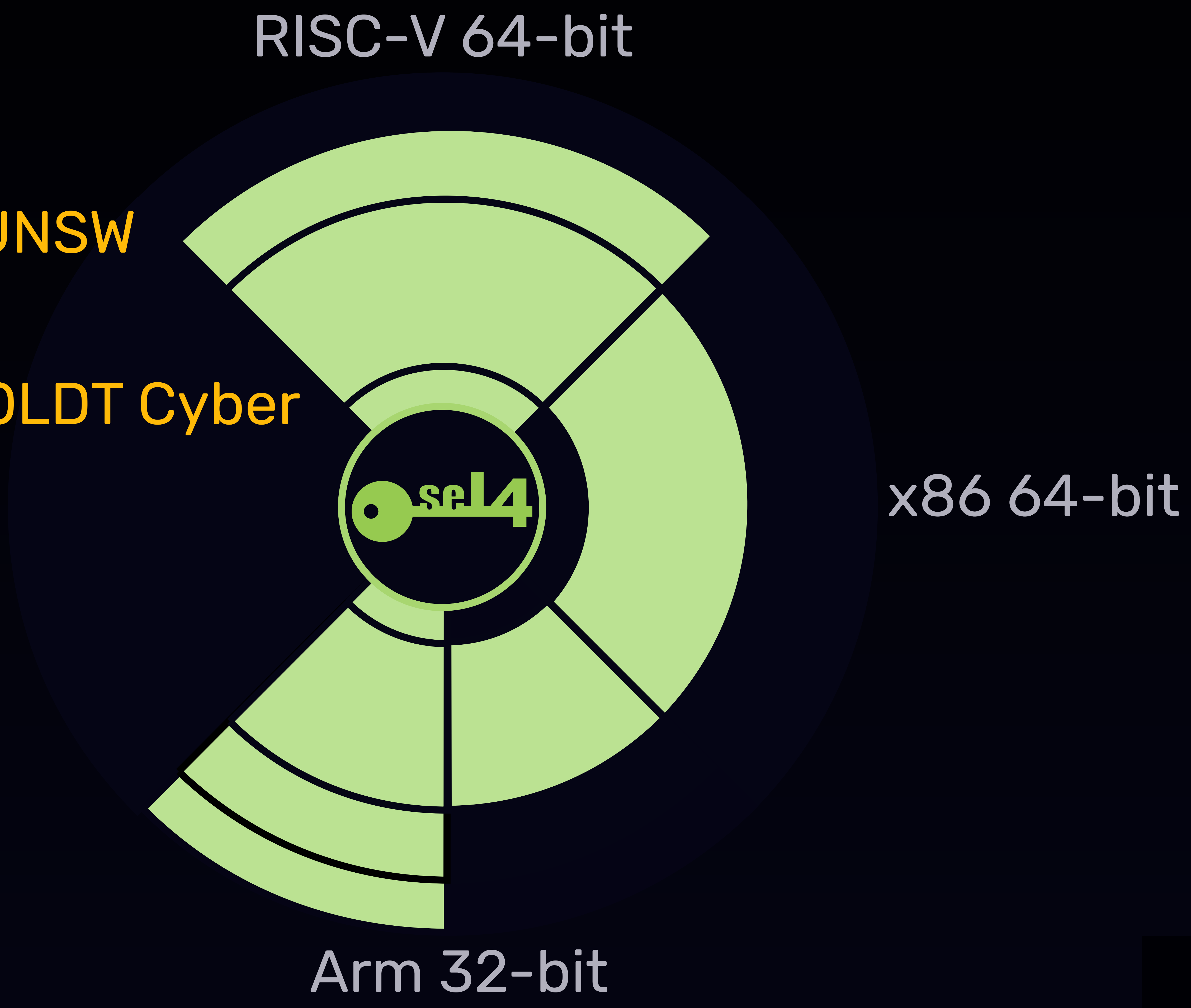
Arm 32-bit

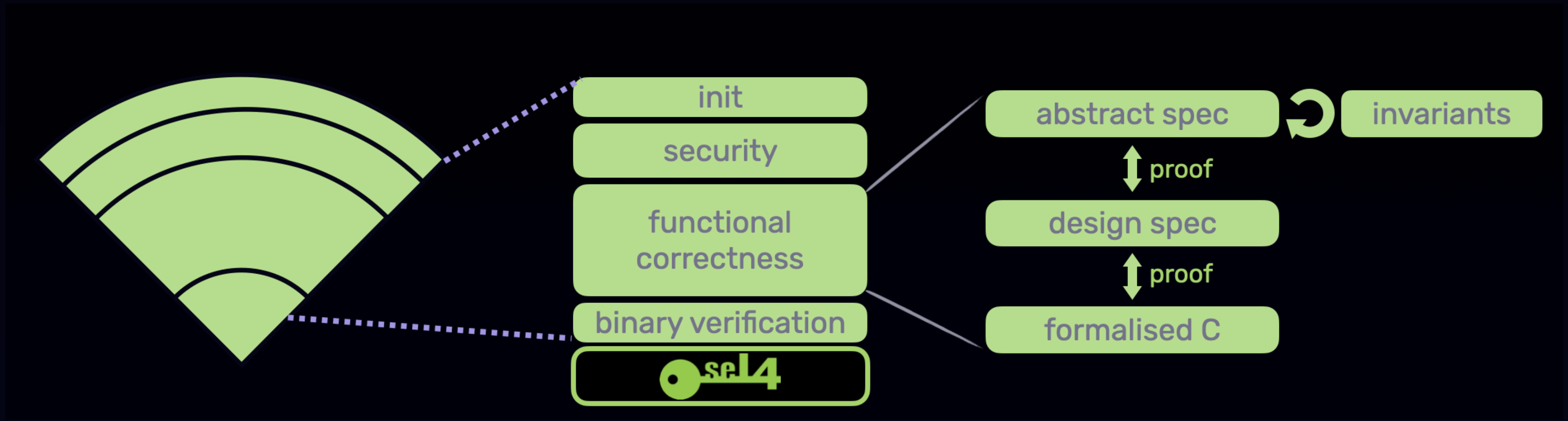


Then…

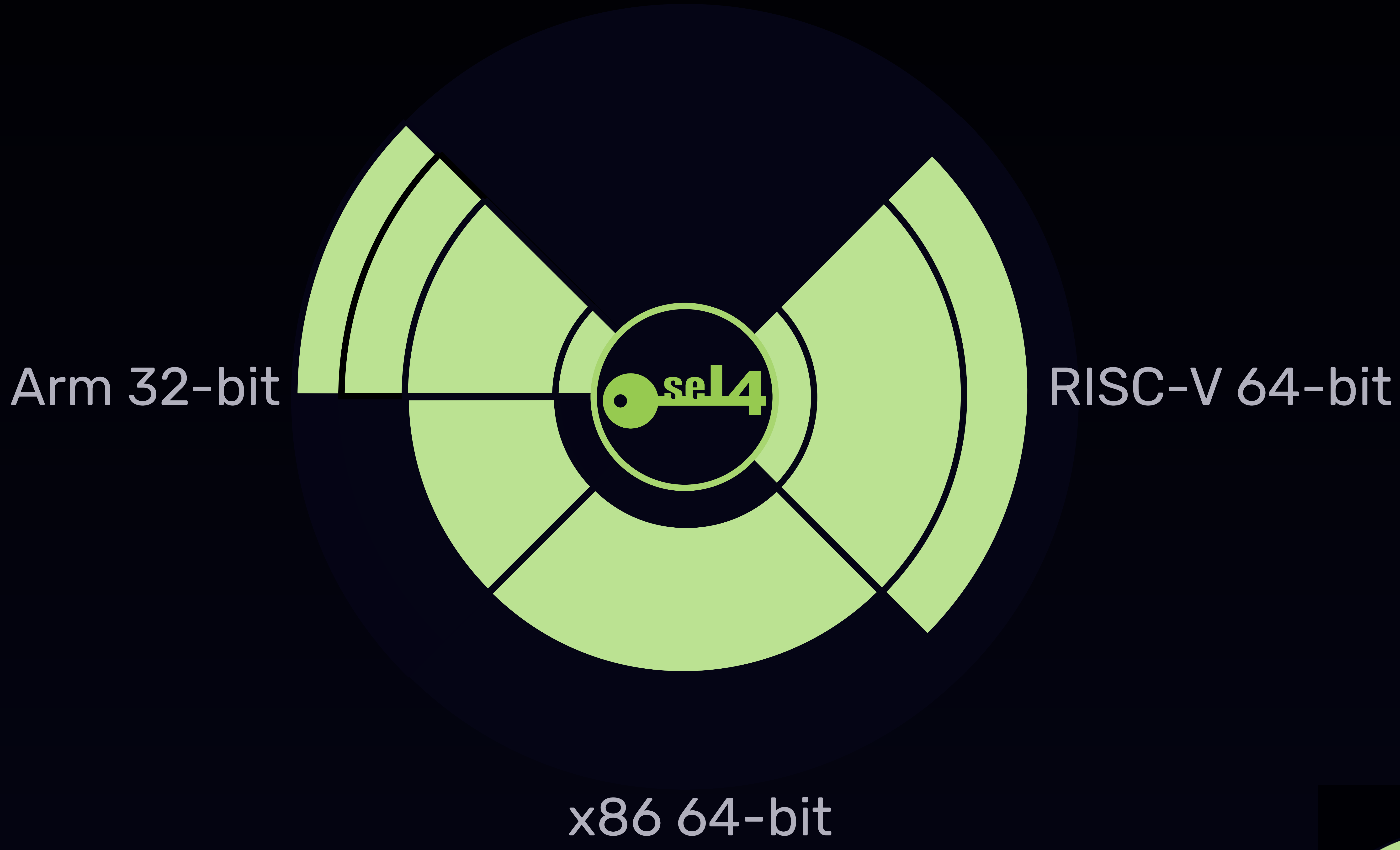
👍 TS @ UNSW

👍 HENSOLDT Cyber





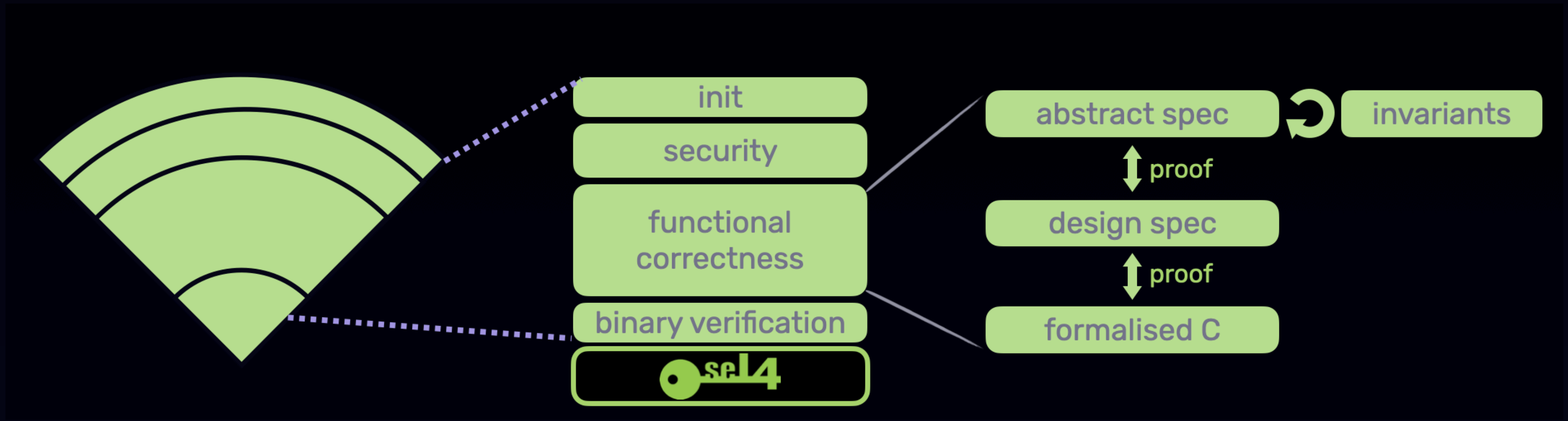
Then…



Arm 32-bit

RISC-V 64-bit

x86 64-bit



init

security

functional correctness

binary verification

abstract spec

invariants

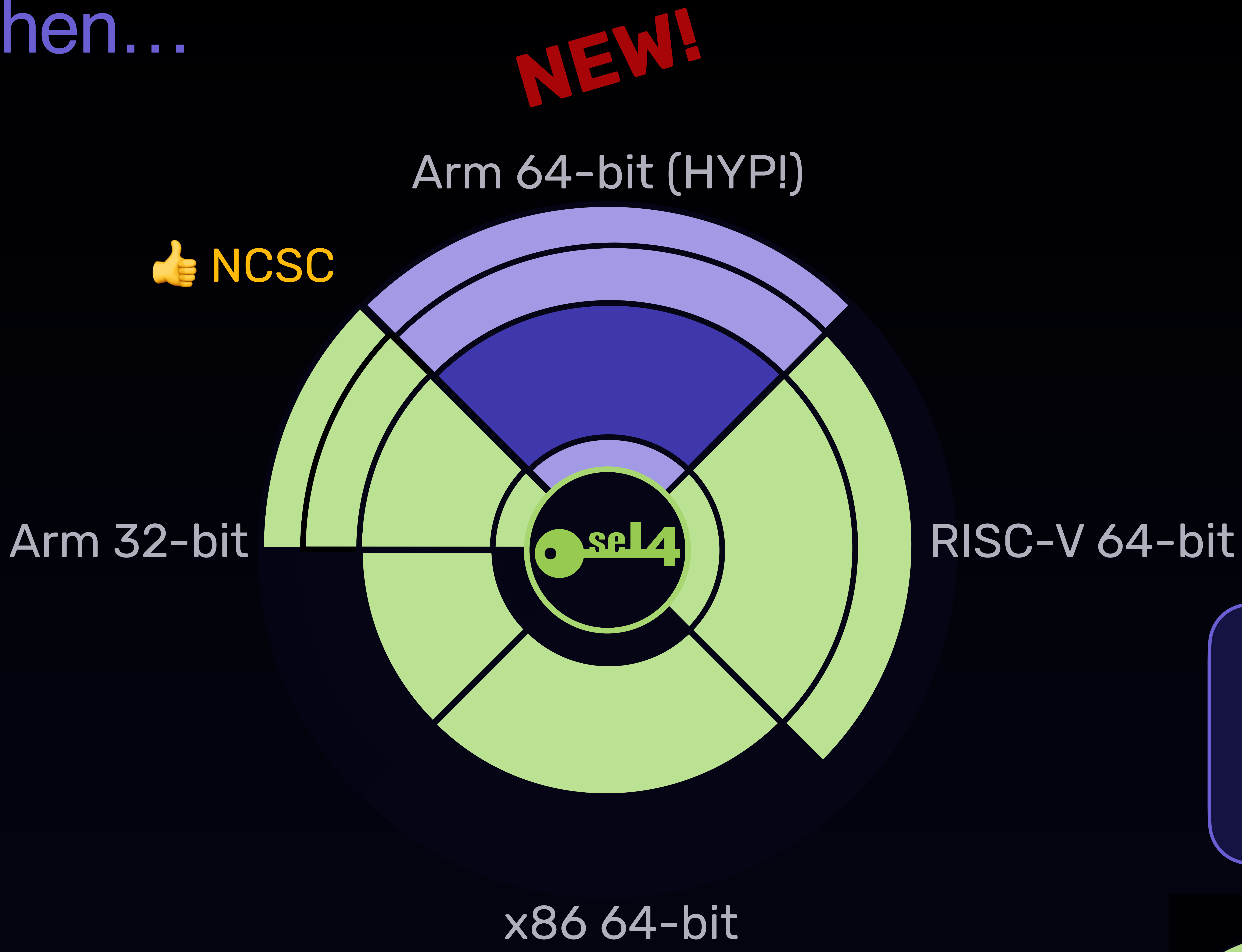
proof

design spec

proof

formalised C

Then…

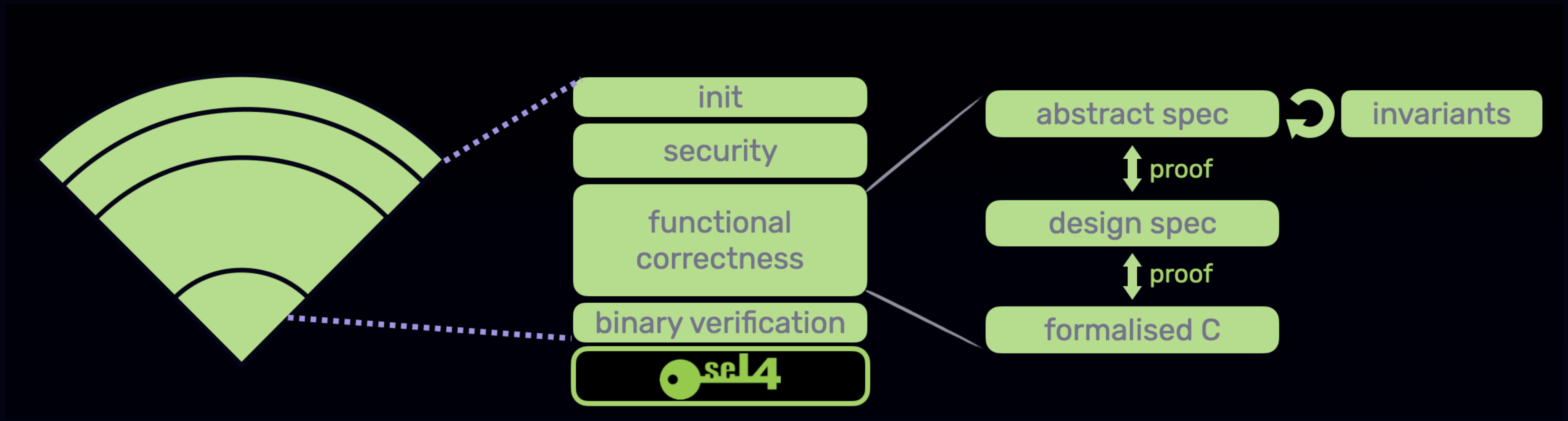


seL4 proofs

Done
Ongoing
Future

(non-MCS, unicore)

seL4's formal proofs evolve with new architectures



Success pushed evolution

seL4’s formal proofs evolve
with new architectures

seL4’s formal proofs evolve
with new features

Parallel verification creates challenges,
Convergence is planned

The proofs have evolved with new features over the years

Two examples:

- bound notification endpoints
- bitfield scheduler optimisation

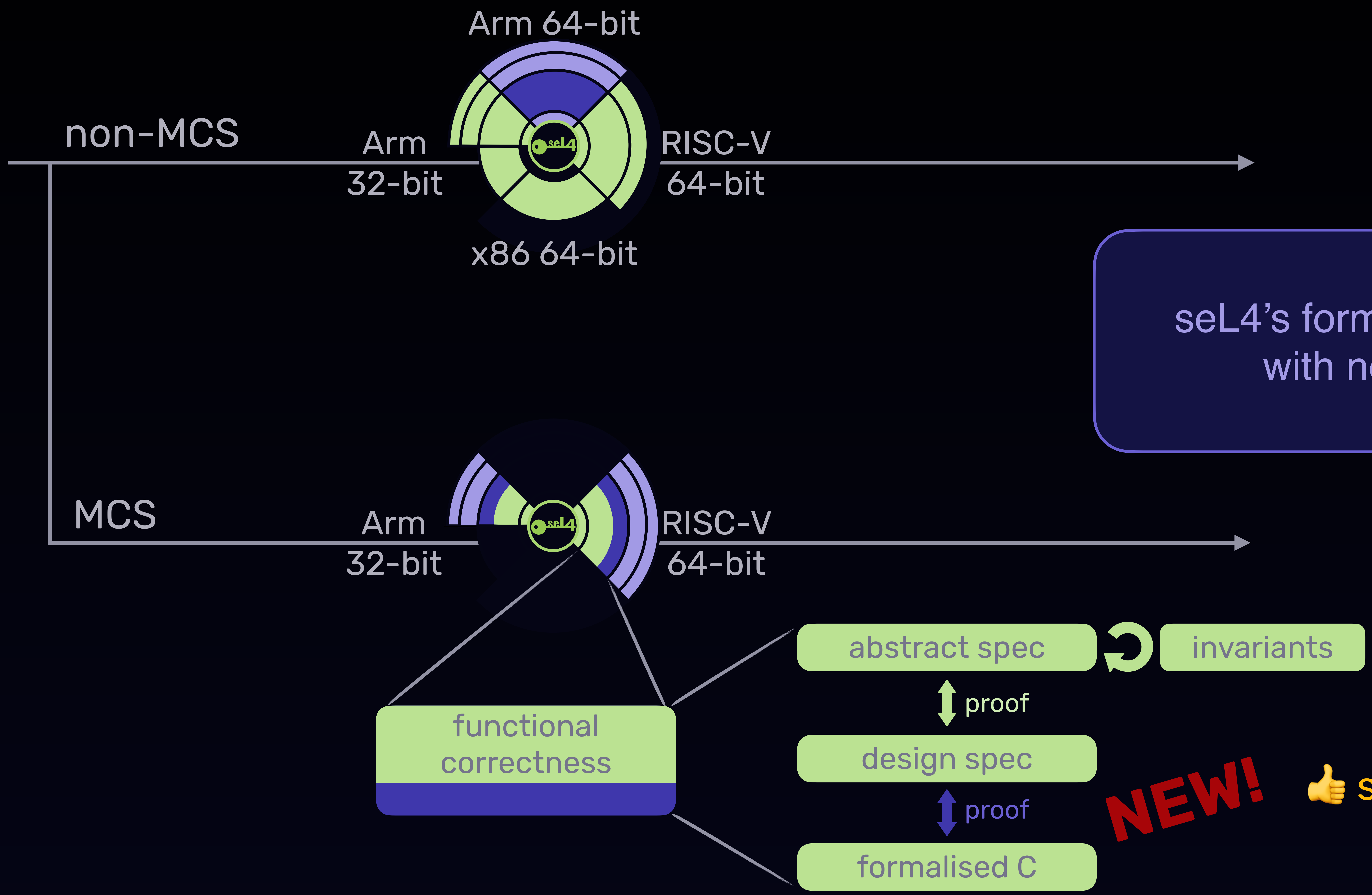
MCS is different:

- Mixed-Criticality Systems
- time as a resource
- large, invasive change

Big Feature: Mixed-Criticality Systems

non-MCS
seL4
MCS
seL4

Verification of multiple configs in parallel



seL4's formal proofs evolve with new features

👍 HENSOLDT Cyber
👍 DHS

NEW!

👍 seL4 Foundation

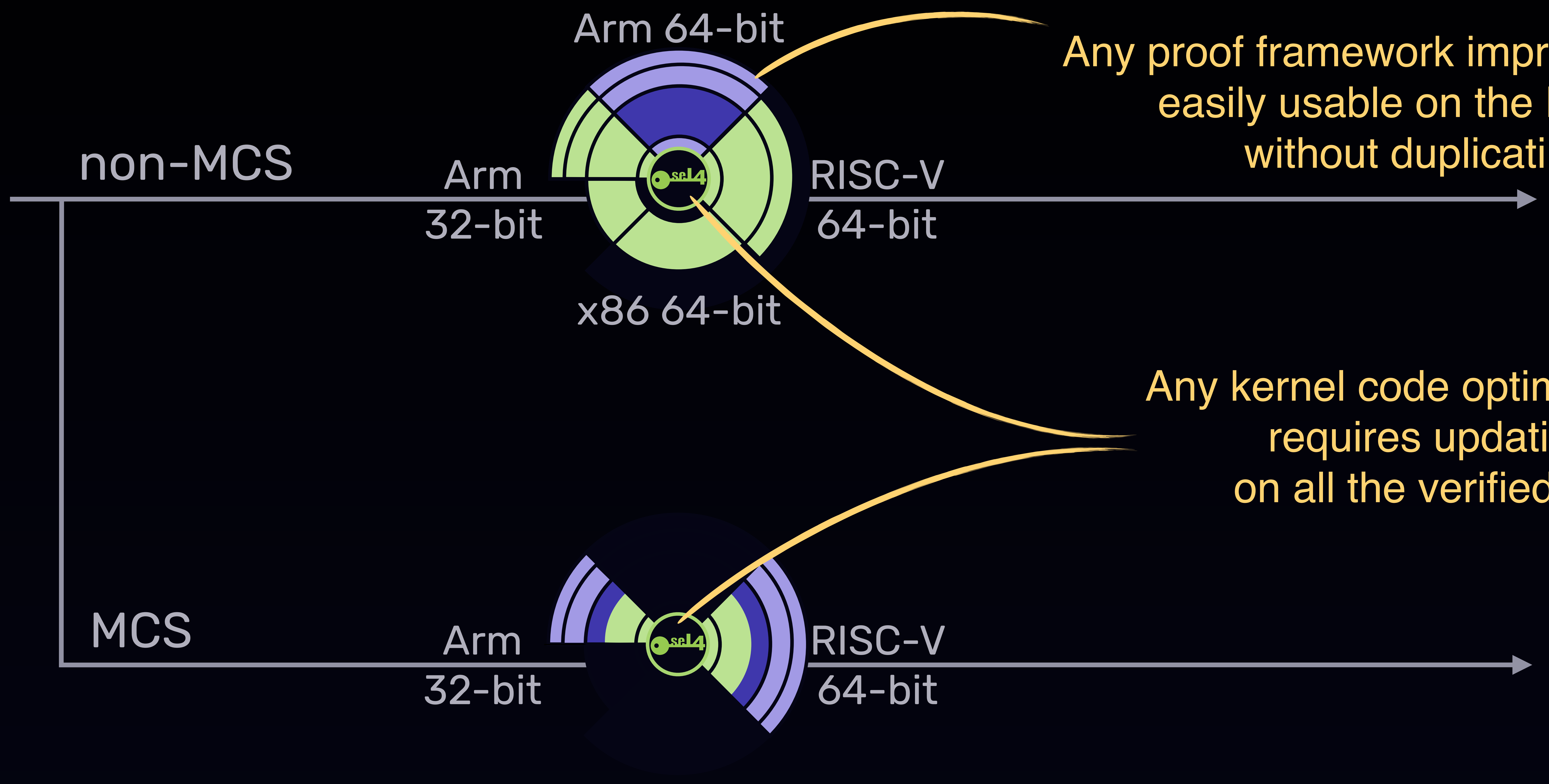
Success pushed evolution

seL4’s formal proofs evolve
with new architectures

seL4’s formal proofs evolve
with new features

▶ Parallel verification creates challenges,
Convergence is planned

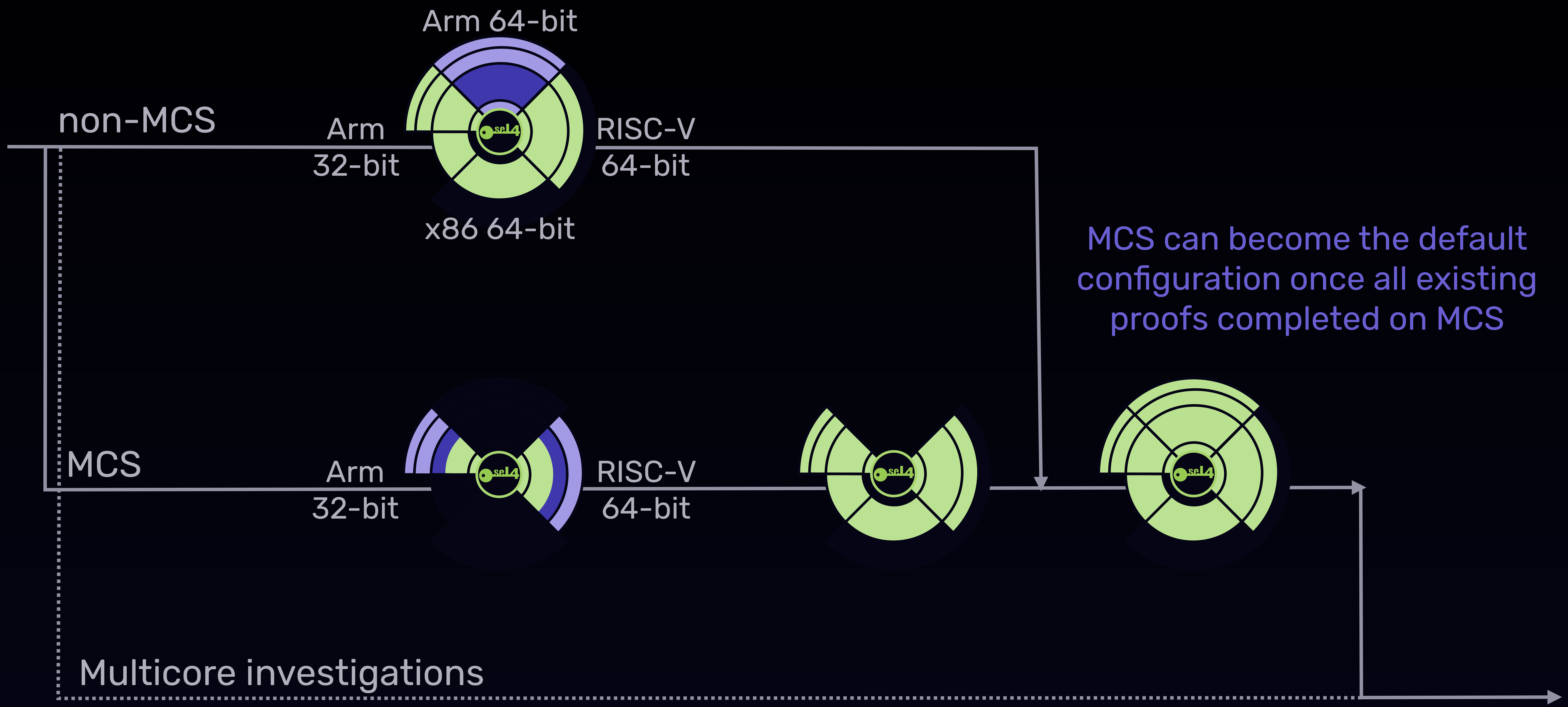
Challenges



Any proof framework improvements are not easily usable on the MCS branch without duplicating work

Any kernel code optimisation or evolution requires updating the proofs on all the verified configurations

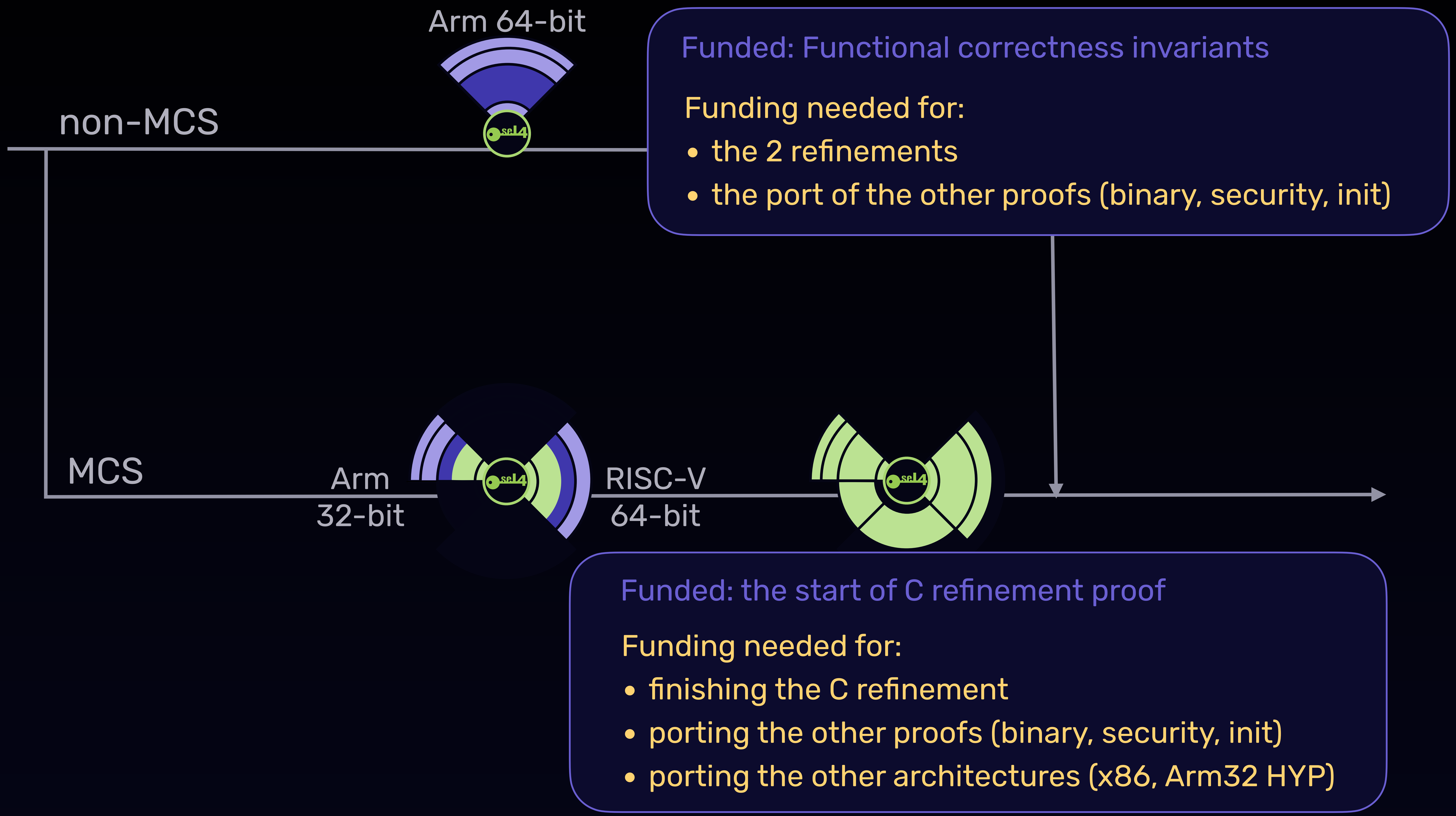
Roadmap



MCS can become the default configuration once all existing proofs completed on MCS

Eventually, seL4 verified on multicore, with unicore as an instance

Status and funding situation



Funded: Functional correctness invariants

Funding needed for:

- the 2 refinements
- the port of the other proofs (binary, security, init)

Funded: the start of C refinement proof

Funding needed for:

- finishing the C refinement
- porting the other proofs (binary, security, init)
- porting the other architectures (x86, Arm32 HYP)

Conclusion

seL4's formal proofs evolve with new architectures:
verified AArch64 seL4 is coming!

seL4's formal proofs evolve with new features:
verified MCS seL4 is coming!

Convergence and funding drive the roadmap:
Contact us if you're interested!

Proofcraft

https://proofcraft.systems