The
FORMOSA
CRYPTO approach to

Formally Verified Cryptography

Manuel Barbosa

University of Porto (FCUP) - INESC TEC - PQShield

mbb@fc.up.pt

Agenda

- **Introducing formosa-crypto**
- Formally verified cryptography landscape
- The formosa-crypto approach
- Where we are for the PQ FIPS standards
- Outreach and industrial adoption

mbb@fc.up.pt

Formosa Crypto

BOSTON UNIVERSITY

CASA
Cyber Security in the Age of Large-Scale Adversaries

CryptoExperts

TU/e Eindhoven University of Technology

h_da darmstadt university of applied sciences

member of EUT+ European University of Technology

institute IMDEA software

INESCTEC

Inria

Max Planck Institute for Security and Privacy

PQ SHIELD

Rosenpass

SandboxAQ

UCLouvain

Universidade do Minho

University of Bristol

The University of Melbourne

U.PORTO
FC FACULDADE DE CIÊNCIAS UNIVERSIDADE DO PORTO

Interactively in a Zulip server

formosa-crypto.org

Formosa Crypto

FORMOSA CRYPTO

News People Projects Publications Formosa Supporters

Projects

- **EasyCrypt** — Project Website — Git Repository
  EasyCrypt is a toolset for reasoning about relational properties of probabilistic computations with adversarial code. Its main application is the construction and verification of game-based cryptographic proofs.
- **Jasmin** — Project Website — Git Repository
  Jasmin is a workbench for high-assurance and high-speed cryptography. Jasmin implementations aim at being efficient, safe, correct, and secure.
- **Libjade** — Project Website — Git Repository
  Libjade is a cryptographic library written in jasmin, with computer-verified proof of correctness and security in EasyCrypt. The primary focus of libjade is to offer high-assurance software implementations of post-quantum crypto primitives.

Interactively in a Zulip server

formosa-crypto.org

libjade: formally verified assembly

- Open-source high-assurance cryptographic library (SUPERCOP-like C API)
- Current features:
  - High-speed implementations for AMD64 (aka x86_64 or x64), ARM v7, RISC V
  - Cryptographic hash functions and XOFs (SHA-2, SHA-3, SHAKE)
  - One-time authenticators and stream ciphers (poly1305, ChaCha, Salsa)
  - Authenticated encryption (XSalsa20Poly1305)
  - Curve 25519
  - Postquantum KEM and Signatures (ML-KEM, ML-DSA, SLH-DSA, ...)

https://github.com/formosa-crypto/libjade

What we formally verify

- Safety
  - For all valid inputs, program behavior is well defined
- Functional correctness
  - For all valid inputs, program is equivalent to specification
- Countermeasures against timing attacks:
  - no secret-dependent branching, memory access, problematic instructions
  - countermeasures against some speculative execution attacks

What we formally verify

- Safety
  - For all valid inputs, program behavior is well defined
- Functional correctness
- Countermeasures against timing attacks:
  - no secret-dependent branching, memory access, problematic instructions
  - countermeasures against some speculative execution attacks

In addition to standard validation techniques!

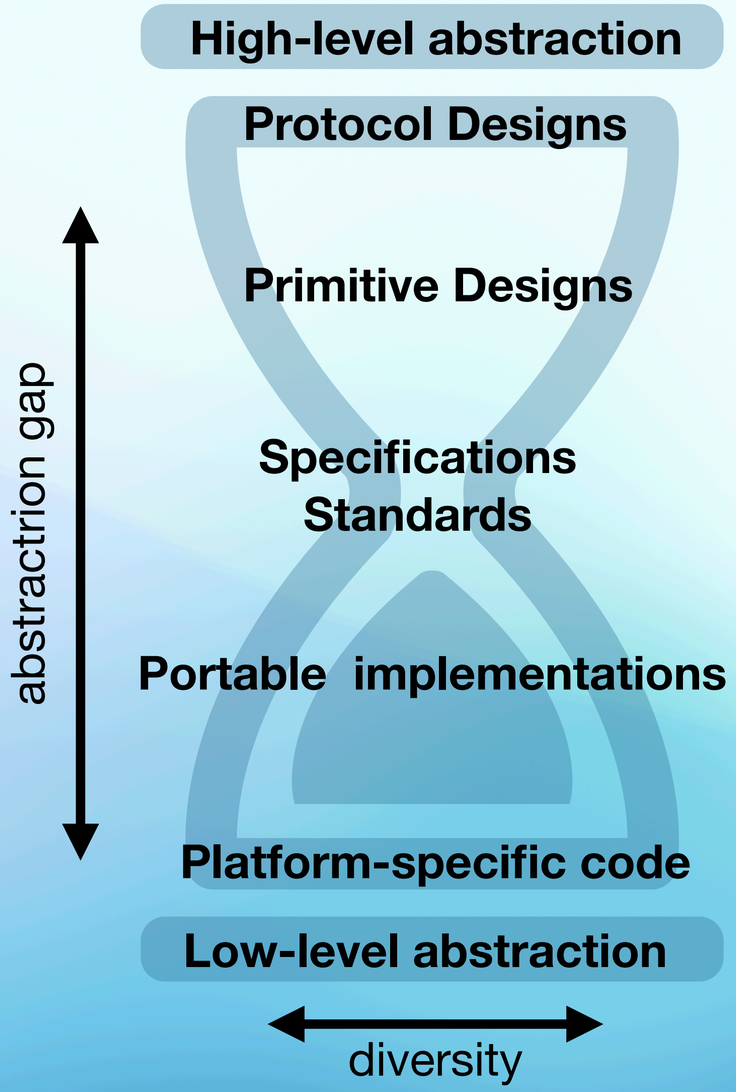
Agenda

- Introducing formosa-crypto
- **Formally verified cryptography landscape**
- The formosa-crypto approach
- Where we are for the PQ FIPS standards
- Outreach and industrial adoption

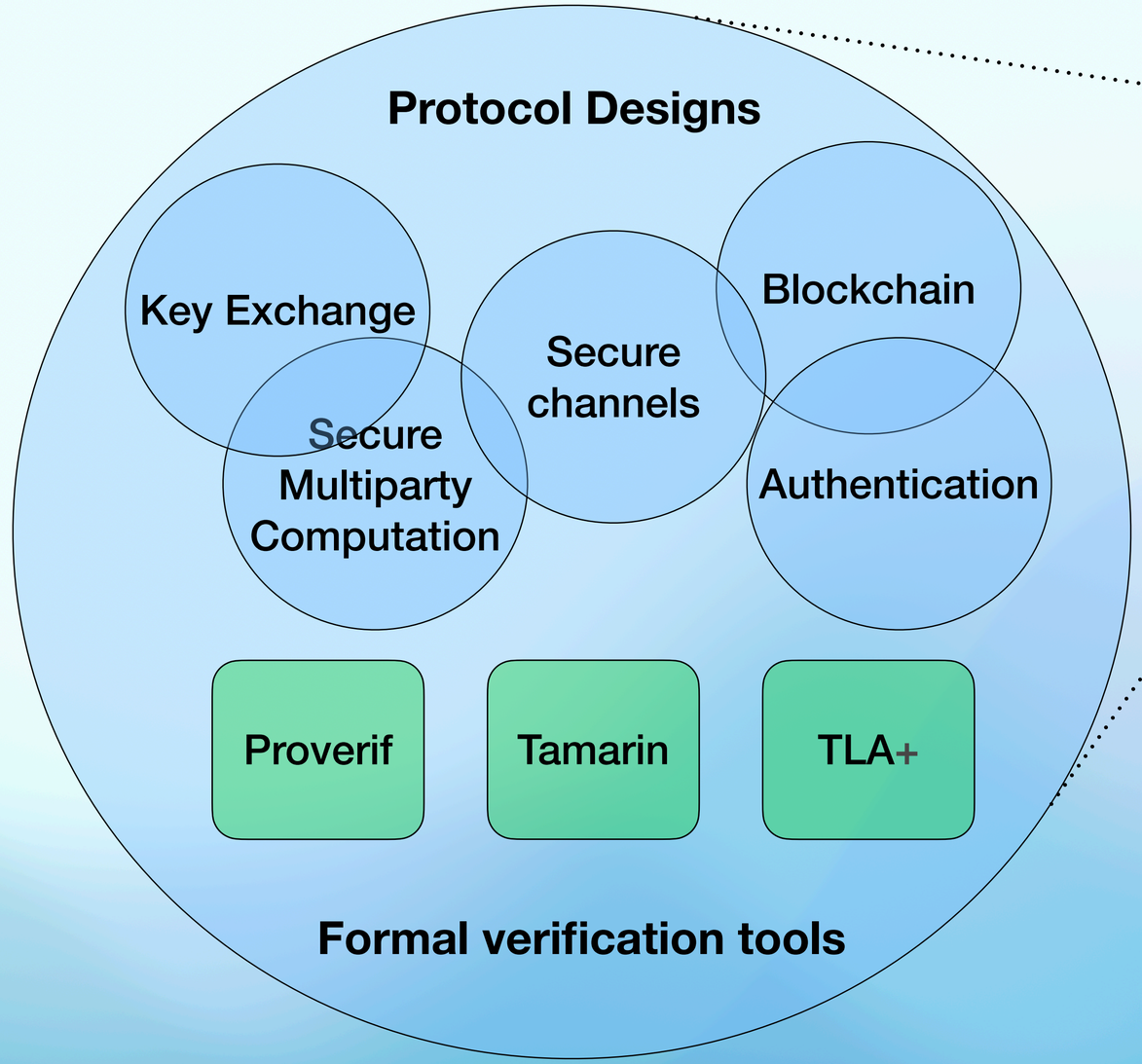
mbb@fc.up.pt

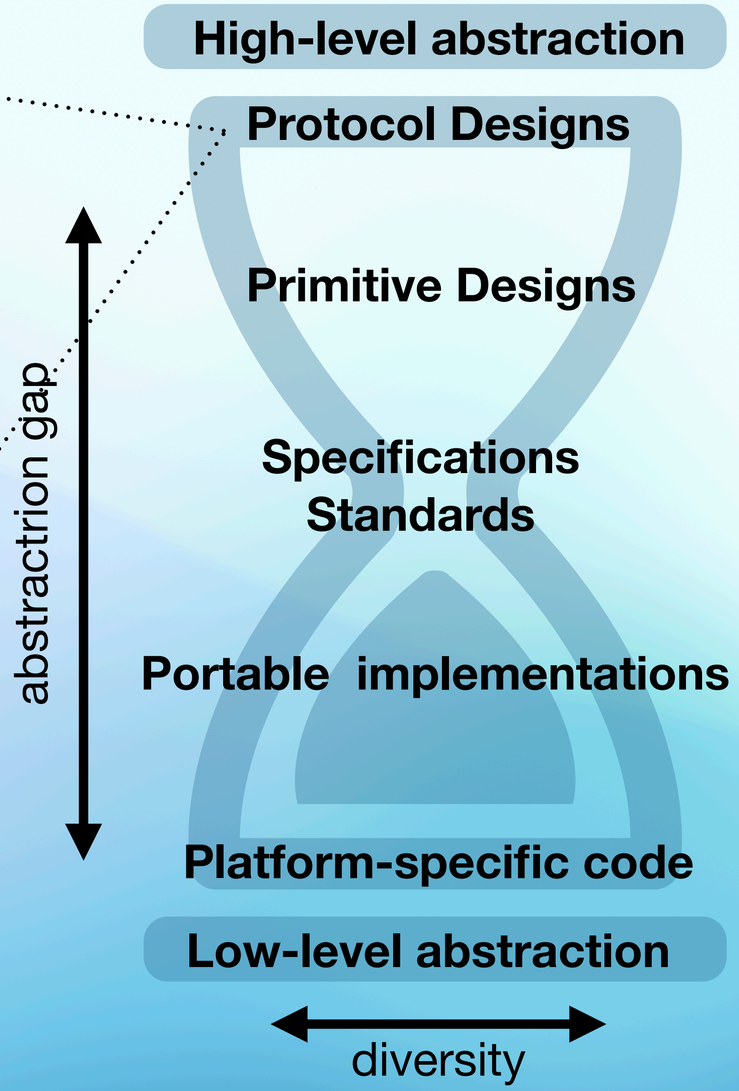
Formally Verified Crypto Landscape

Upcoming examples are not exhaustive.

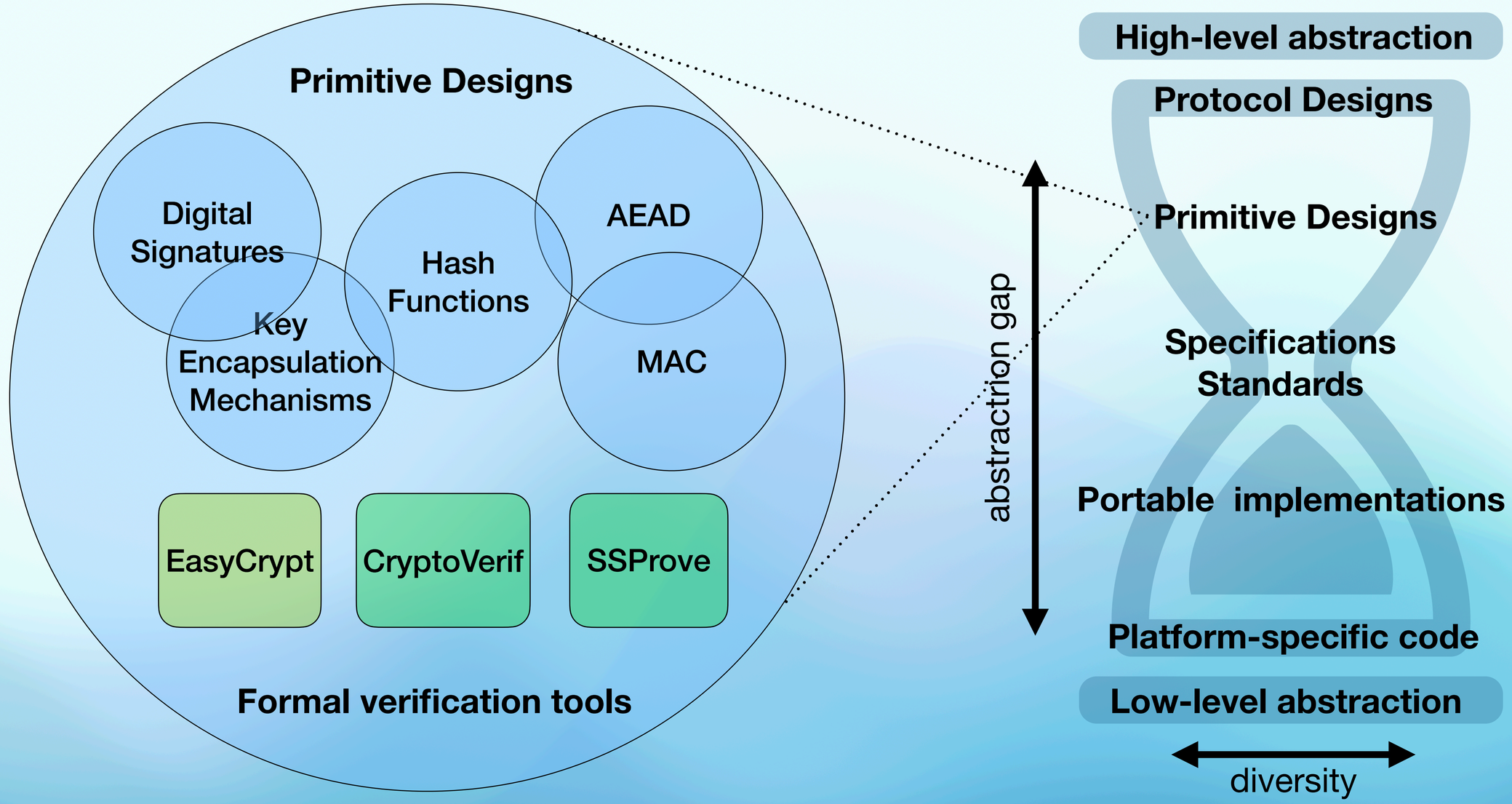


Formally Verified Crypto Landscape

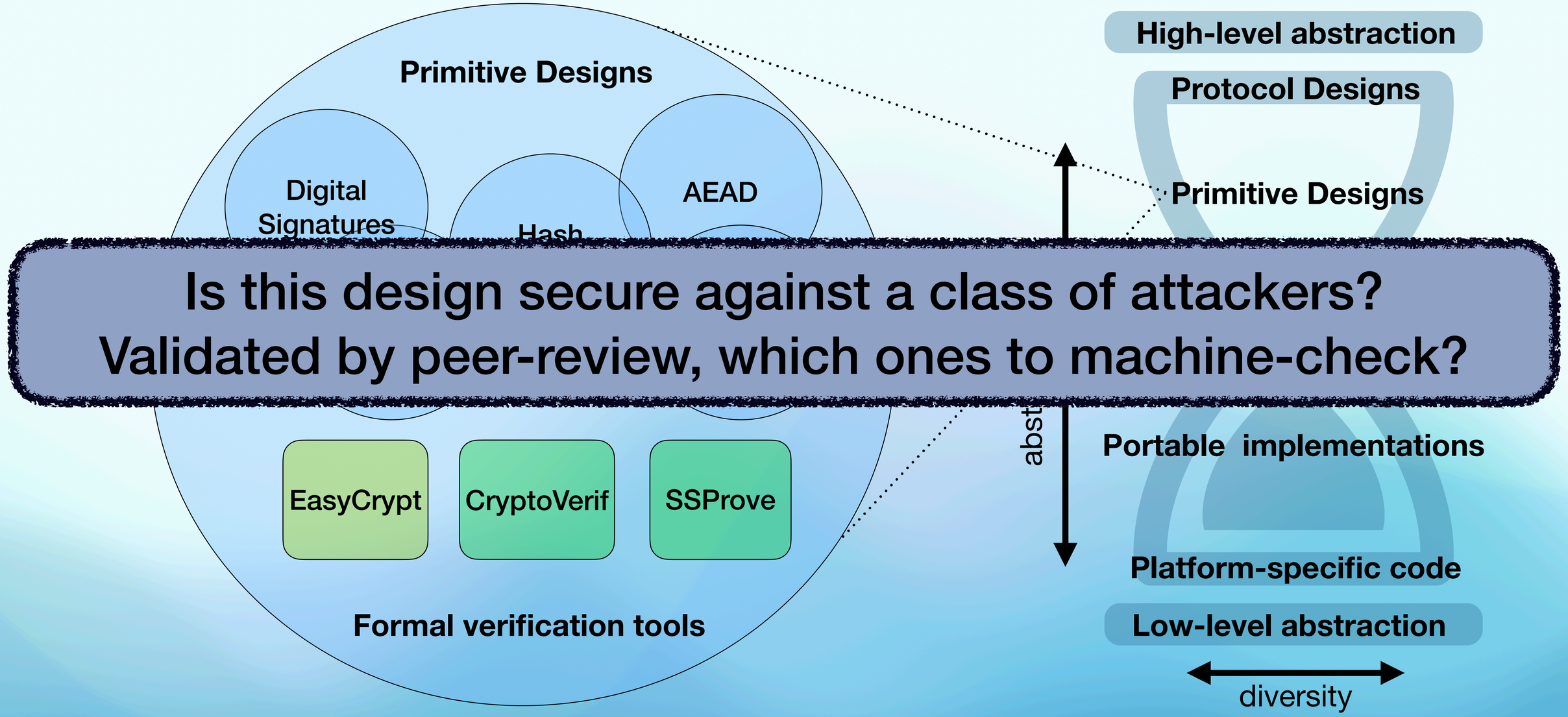




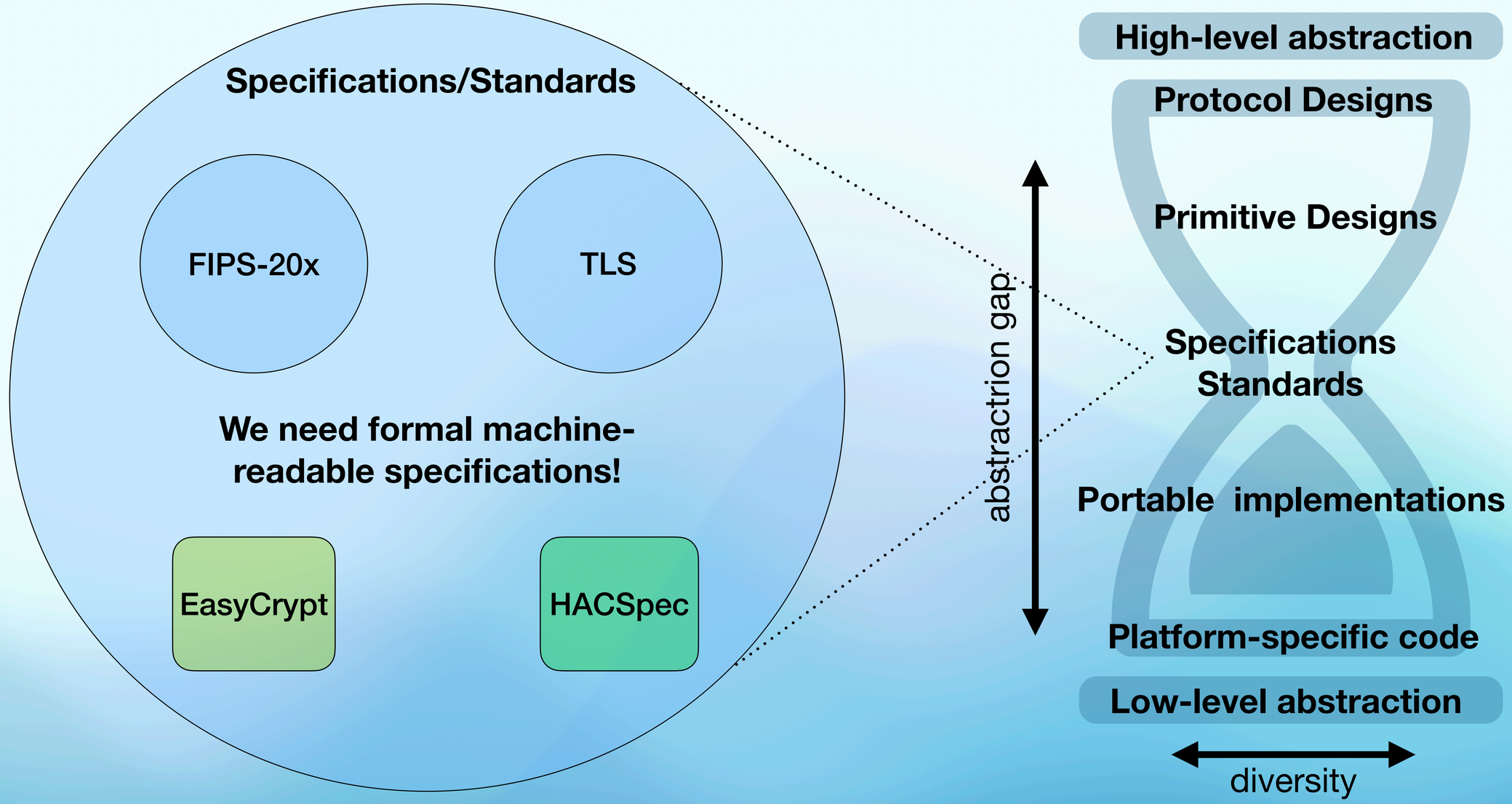
Formally Verified Crypto Landscape



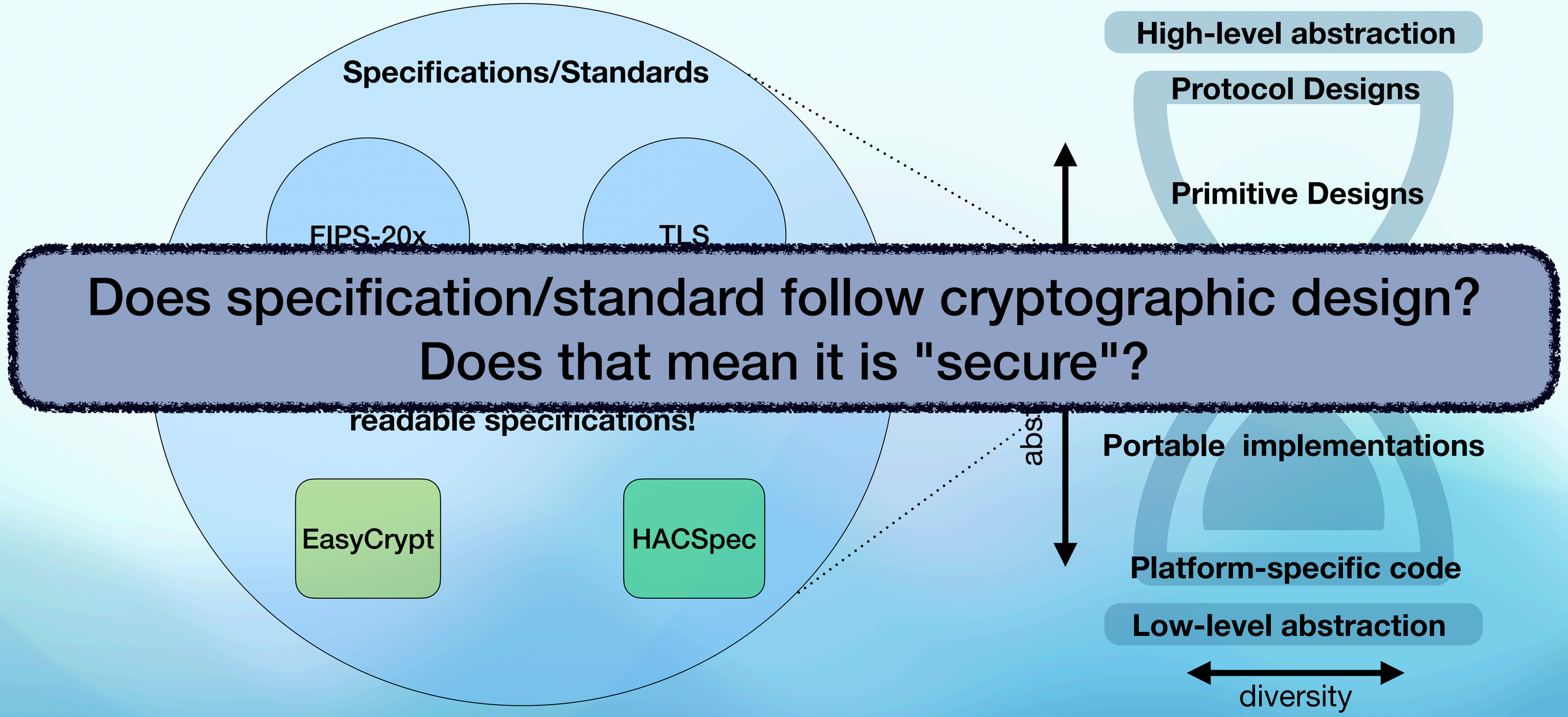
Formally Verified Crypto Landscape



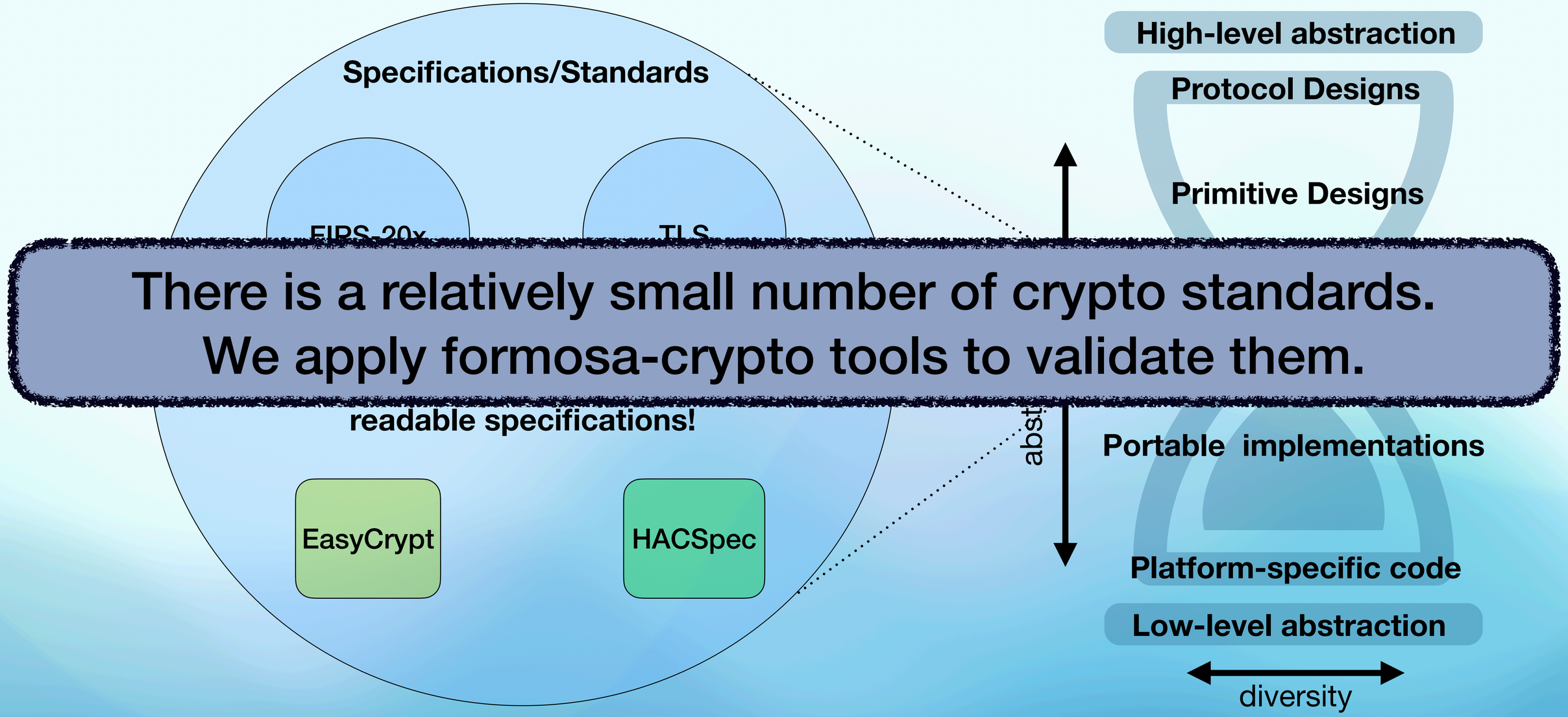
Formally Verified Crypto Landscape



Formally Verified Crypto Landscape

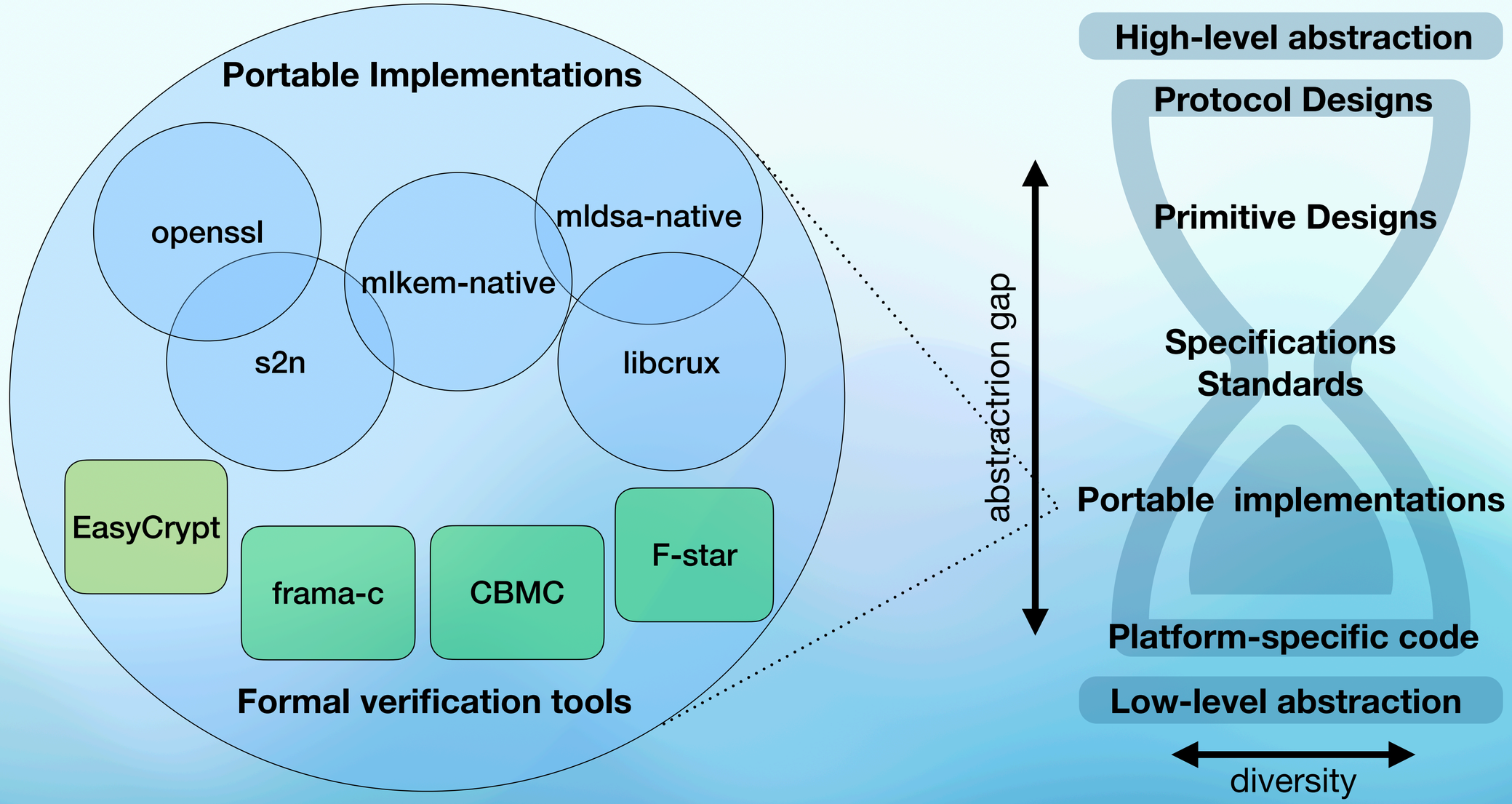


Formally Verified Crypto Landscape

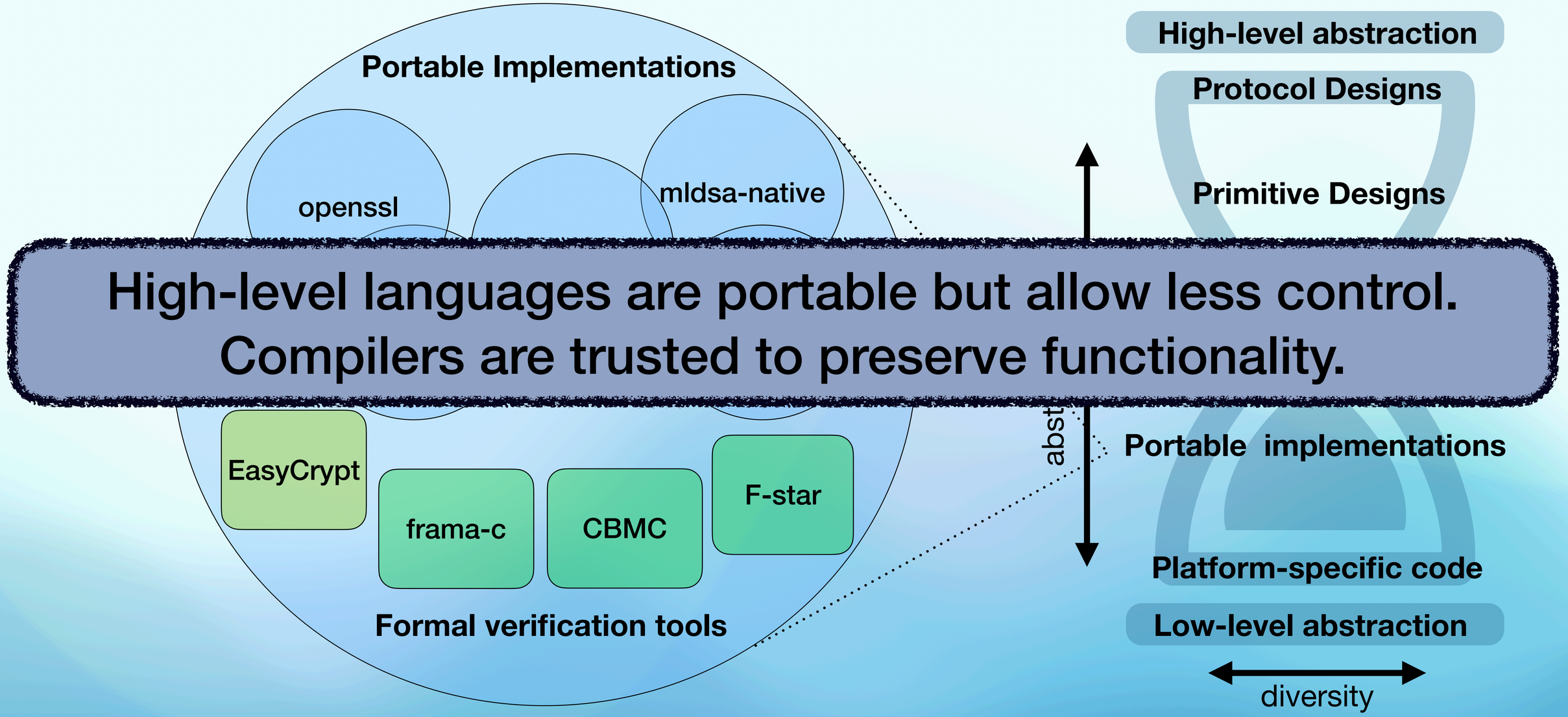


There is a relatively small number of crypto standards.
We apply formosa-crypto tools to validate them.

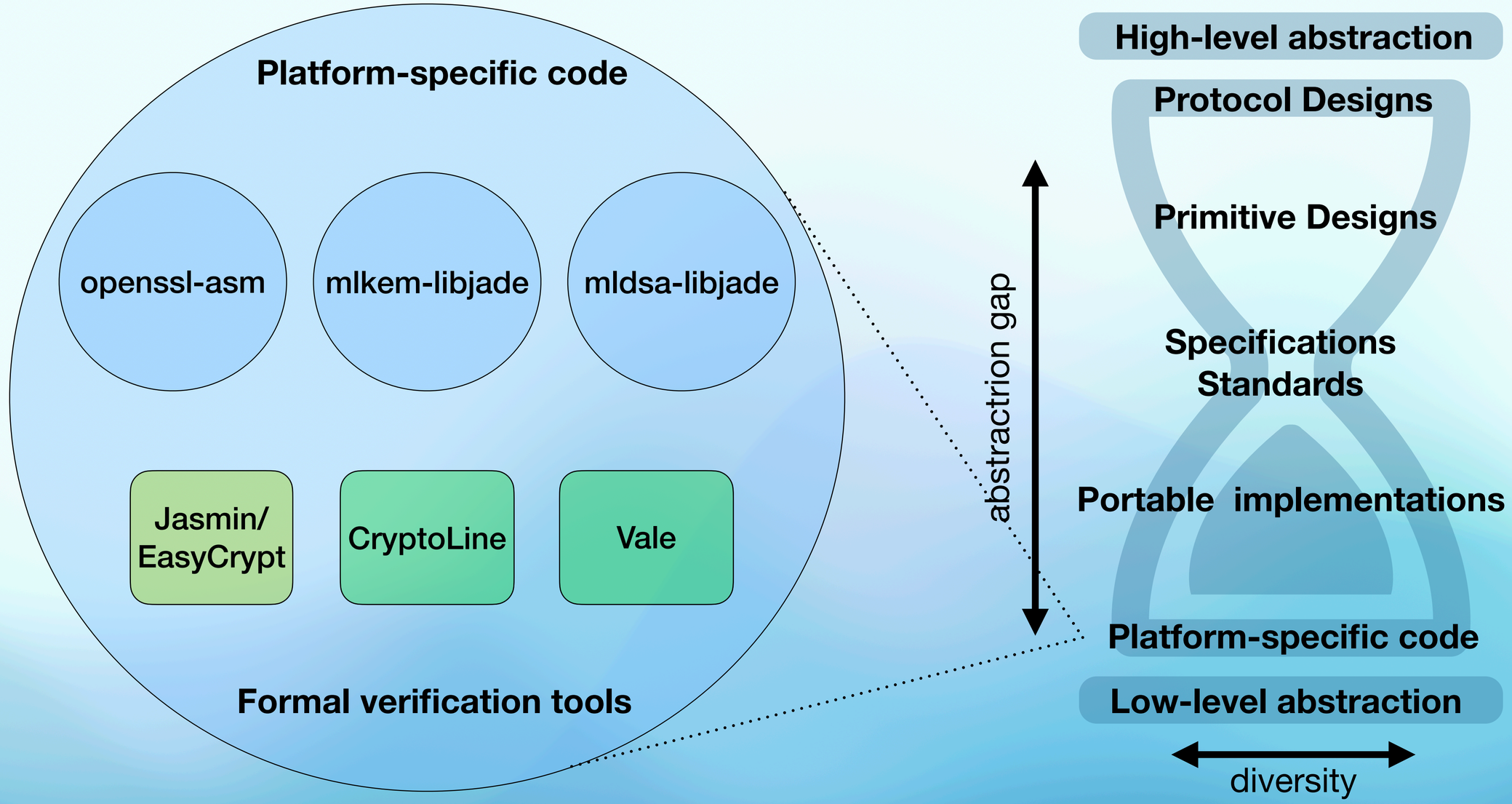
Formally Verified Crypto Landscape



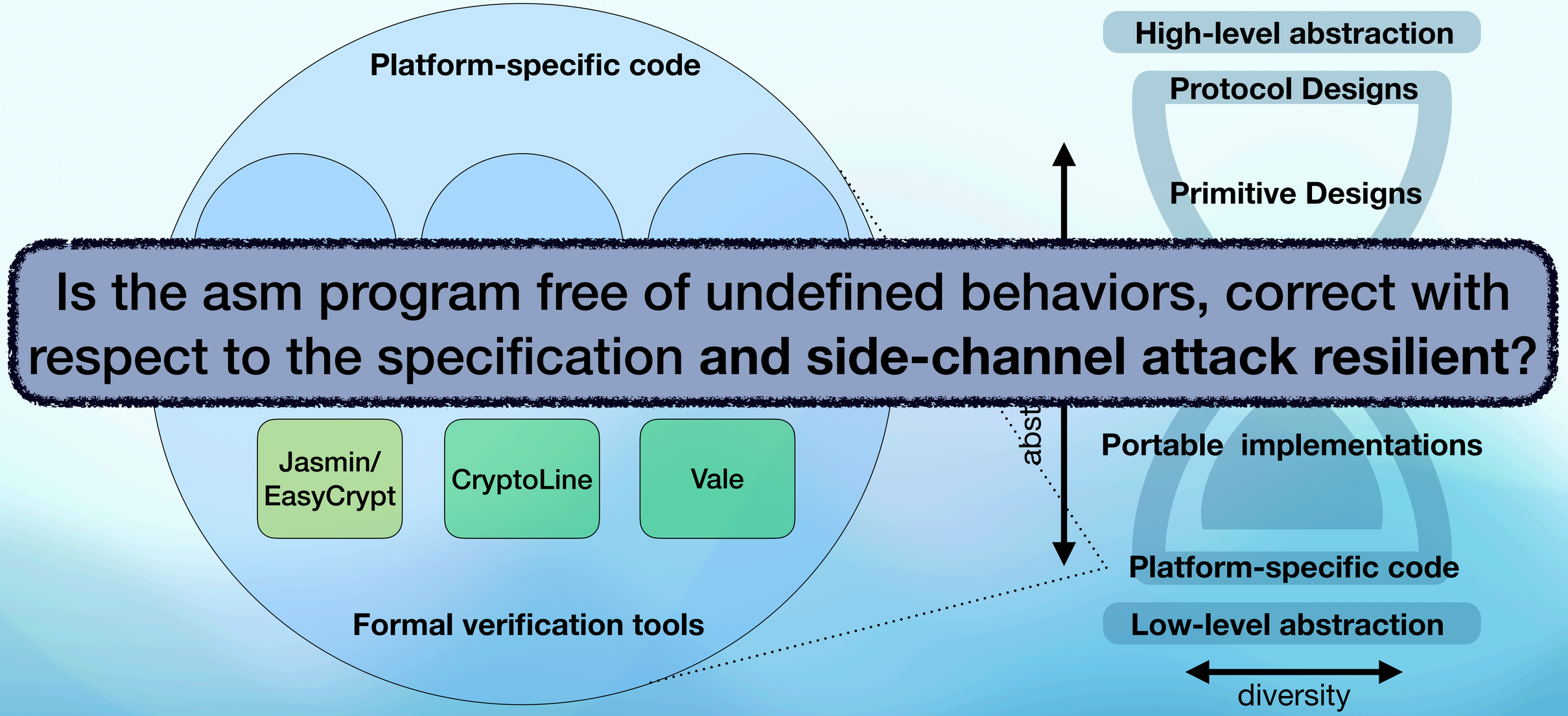
Formally Verified Crypto Landscape



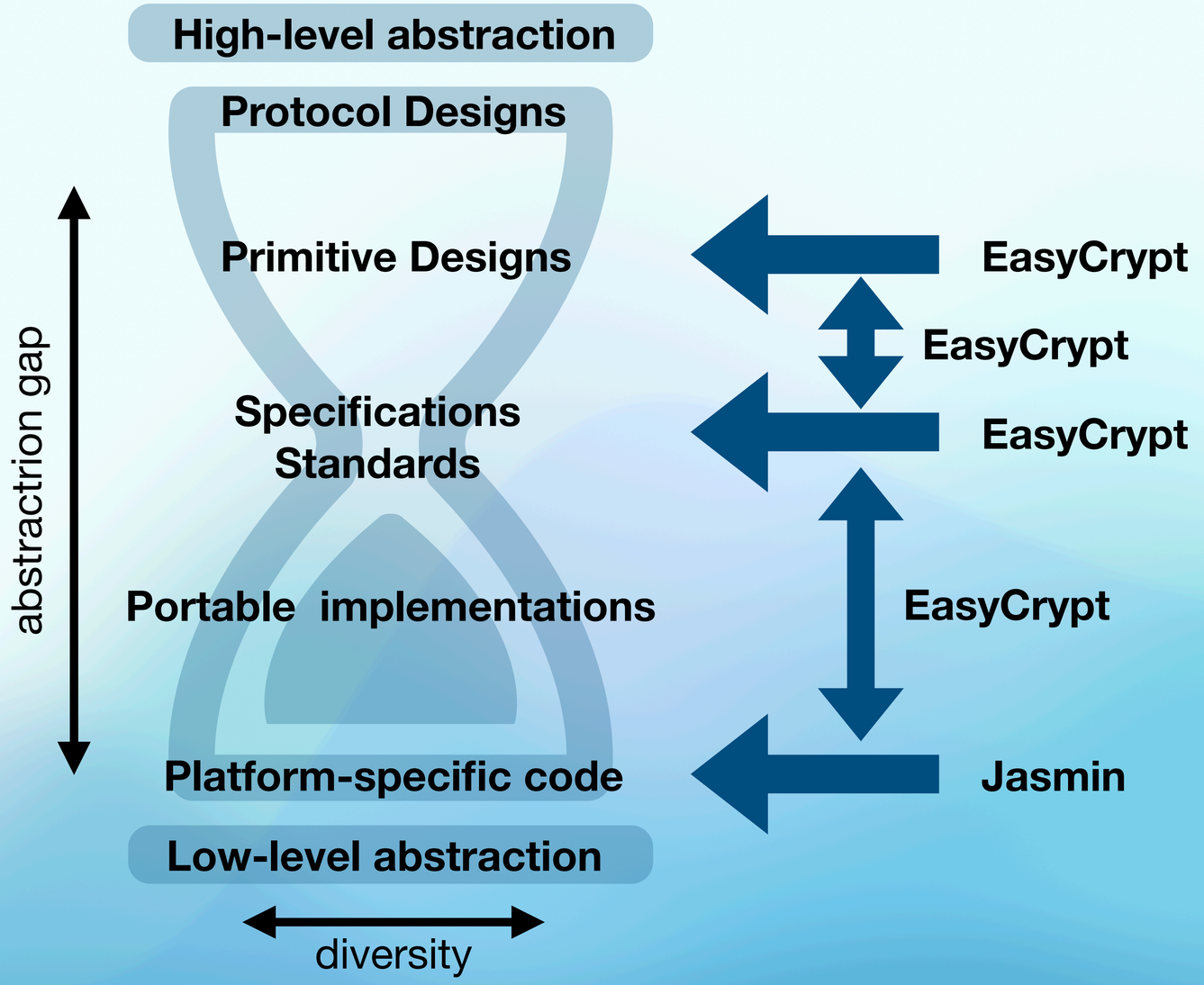
Formally Verified Crypto Landscape



Formally Verified Crypto Landscape



Formally Verified Crypto Landscape



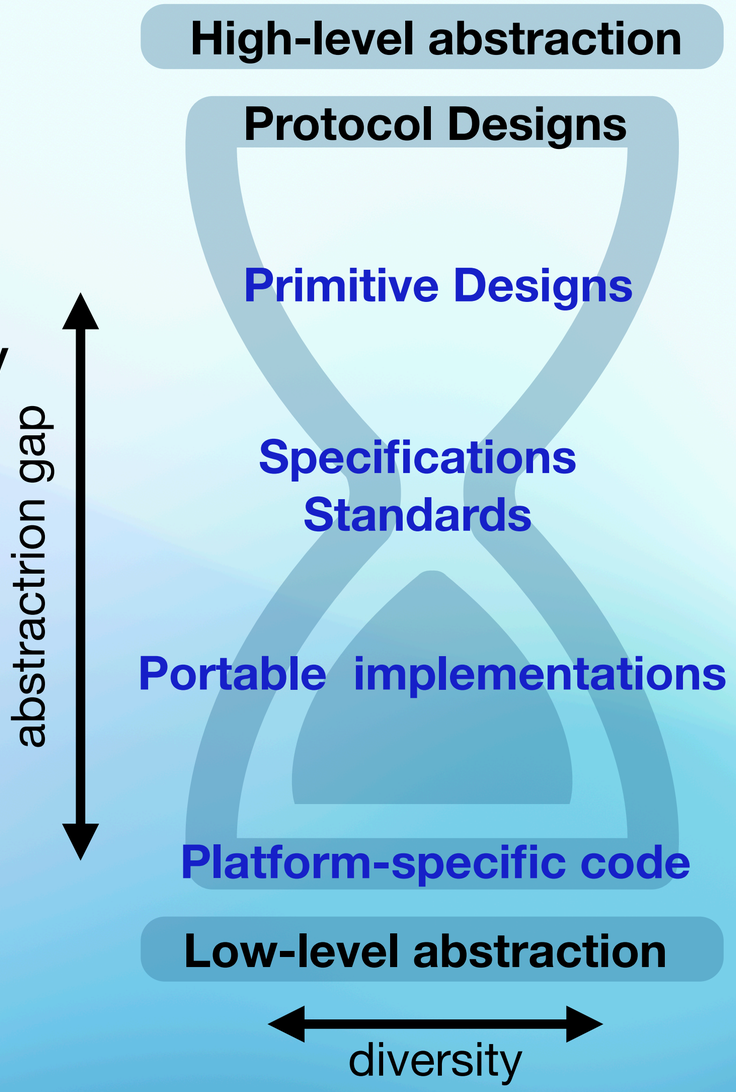
Agenda

- Introducing formosa-crypto
- Formally verified cryptography landscape
- **The formosa-crypto approach**
- Where we are for the PQ FIPS standards
- Outreach and industrial adoption

mbb@fc.up.pt

EasyCrypt

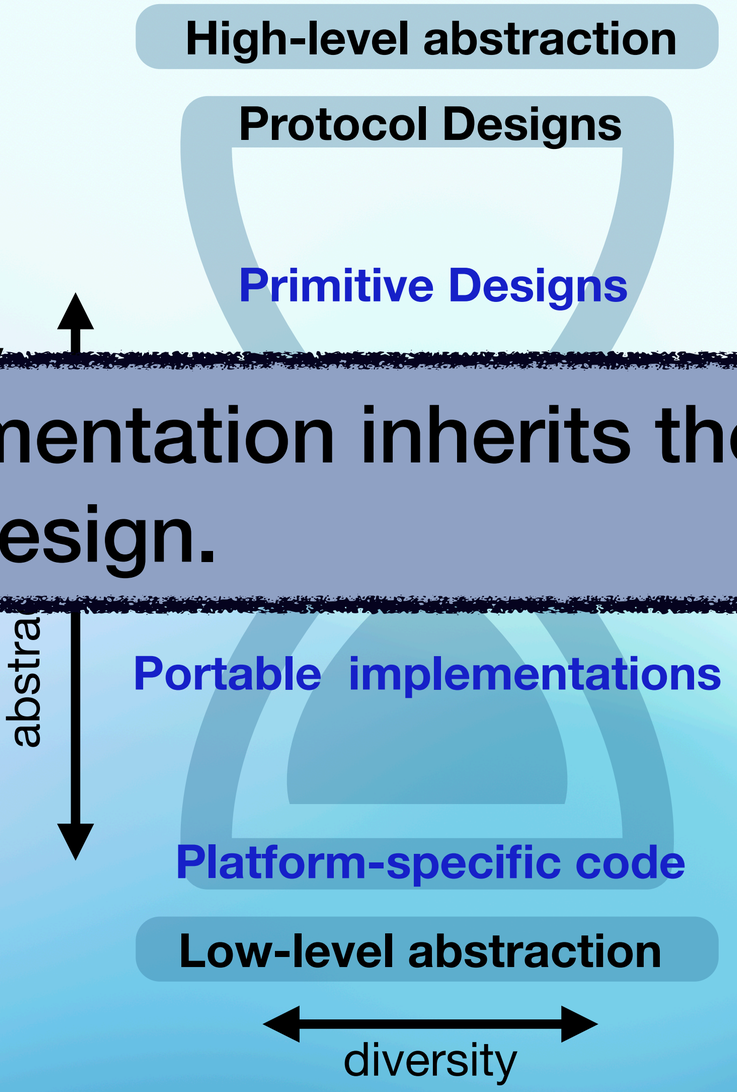
- Flexible interactive proof environment
- Reasoning about probabilistic programs:
  - write your own model, prove your own property
- Designed to machine-check provable security
  - security models are probabilistic programs
- Improved to machine-check implementations
  - implementations are probabilistic programs



EasyCrypt

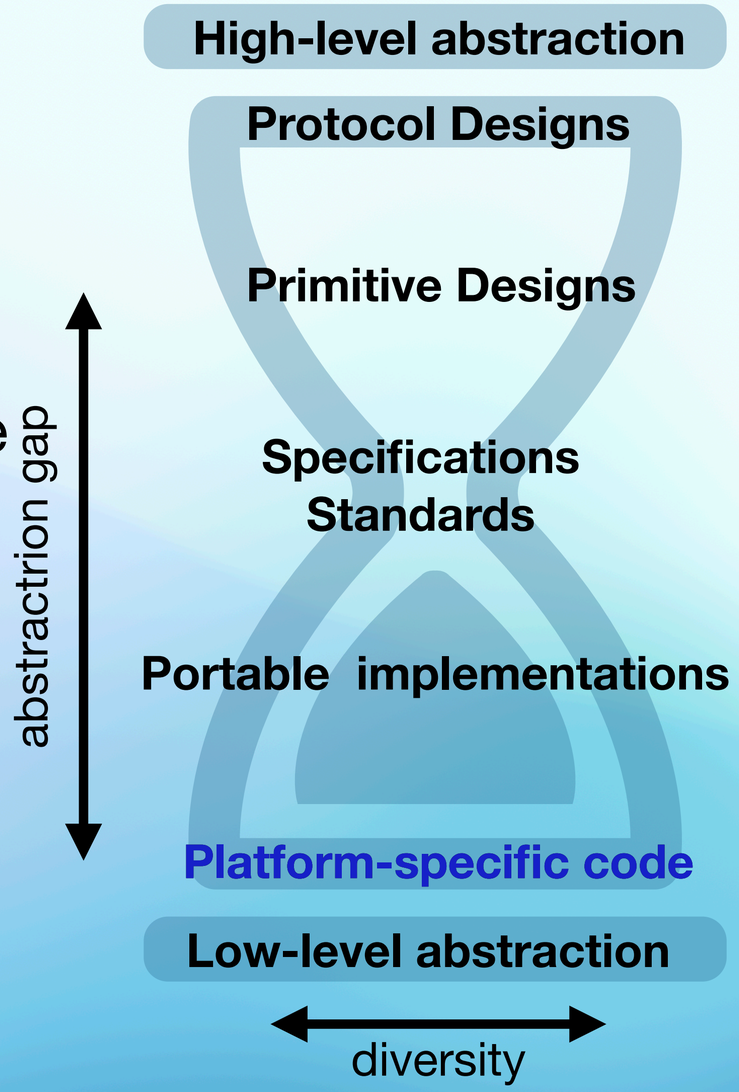
- Flexible interactive proof environment
- Reasoning about probabilistic programs:
  - write your own model, prove your own property
  - security models are probabilistic programs
- Improved to machine-check implementations
  - implementations are probabilistic programs

Results can be chained together: implementation inherits the security of the abstract design.



Jasmin

- Programming language with nice (rust-like) syntax
- (Almost) full control "asm-in-the-head" paradigm:
  - Jasmin programmer has specific asm code in mind
  - Non-optimizing compiler: output asm is predictable
- Compiler is certified:
  - assembly behaves as Jasmin source
- Compiler is thin:
  - does not introduce new timing leaks

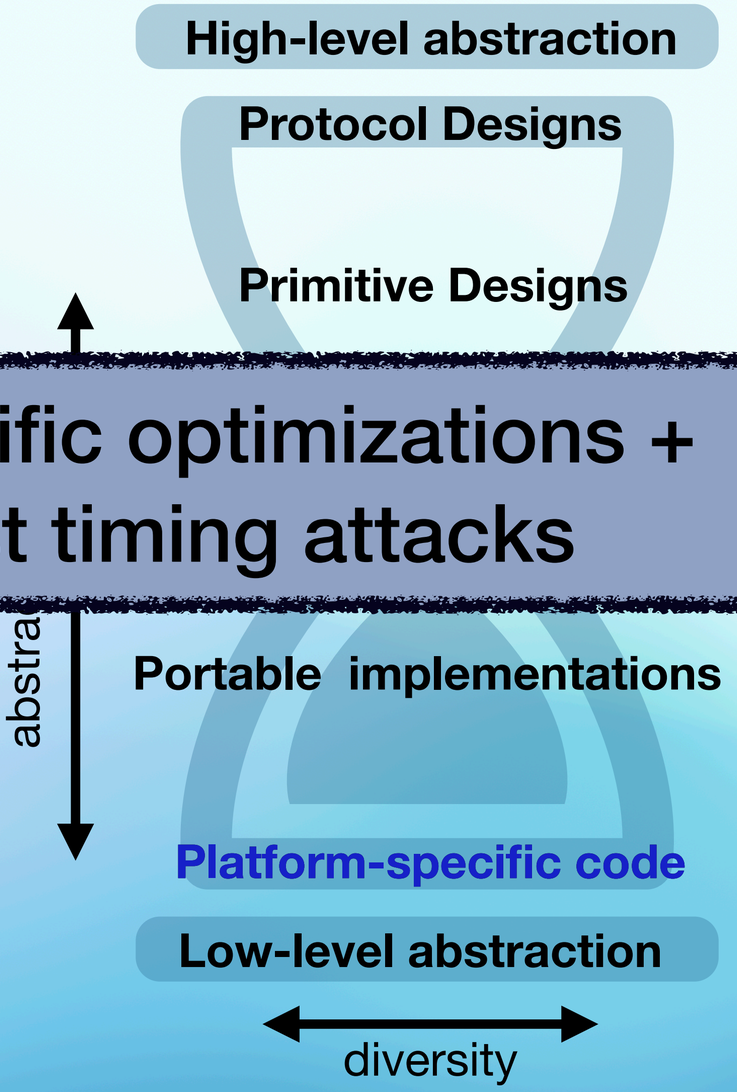


Jasmin

- Programming language with nice (rust-like) syntax
- (Almost) full control "asm-in-the-head" paradigm:
  - Jasmin programmer has specific asm code in mind

Target applications: architecture-specific optimizations + asm-level countermeasures against timing attacks

  - assembly behaves as Jasmin source
- Compiler is thin:
  - does not introduce new timing leaks



Formal Verification Approach

e.g., ML-KEM asm behaves like ML-KEM jasmin

Jasmin Code

Functional correctness (automatic)

Safety check

CT check

No timing leakage (*)

Spectre v1 Check

certified compilation

No speculative leakage (*)

asm Code

Algorithm spec

Machine-checked in EasyCrypt

crypto proof

Security model

e.g., ML-KEM spec is a correct IND-CCA secure KEM

(*) in a formally defined (abstract) leakage model

Formal Verification Approach

e.g., ML-KEM asm behaves like ML-KEM jasmin

Functional correctness (automatic)

No timing leakage (*)

No speculative leakage (*)

Jasmin Code

Safety check

CT check

Spectre v1 Check

certified compilation

asm Code

implementation security ?

compliance/ Interoperability ?

Algorithm spec

crypto proof

Security model

Machine-checked in EasyCrypt

e.g., ML-KEM spec is a correct IND-CCA secure KEM

(*) in a formally defined (abstract) leakage model

Formal Verification Approach

e.g., ML-KEM asm behaves like ML-KEM jasmin

Functional correctness (automatic)

No timing leakage (*)

No speculative leakage (*)

Jasmin Code

Safety check

CT check

Spectre v1 Check

certified compilation

asm Code

Functional Correctness (interactive)

implementation security

compliance/ Interoperability ?

depends on Spec

Algorithm spec

crypto proof

Security model

Machine-checked in EasyCrypt

e.g., ML-KEM spec is a correct IND-CCA secure KEM

(*) in a formally defined (abstract) leakage model

Formal Verification Approach

e.g., ML-KEM asm behaves like ML-KEM jasmin

Functional correctness (automatic)

No timing leakage (*)

No speculative leakage (*)

Jasmin Code

Safety check

CT check

Spectre v1 Check

certified compilation

asm Code

Functional Correctness (interactive)

implementation security

compliance/ Interoperability

Algorithm spec

Other specs? (e.g. HACSpec)

Standard?

crypto proof

Machine-checked in EasyCrypt

Security model

e.g., ML-KEM spec is a correct IND-CCA secure KEM

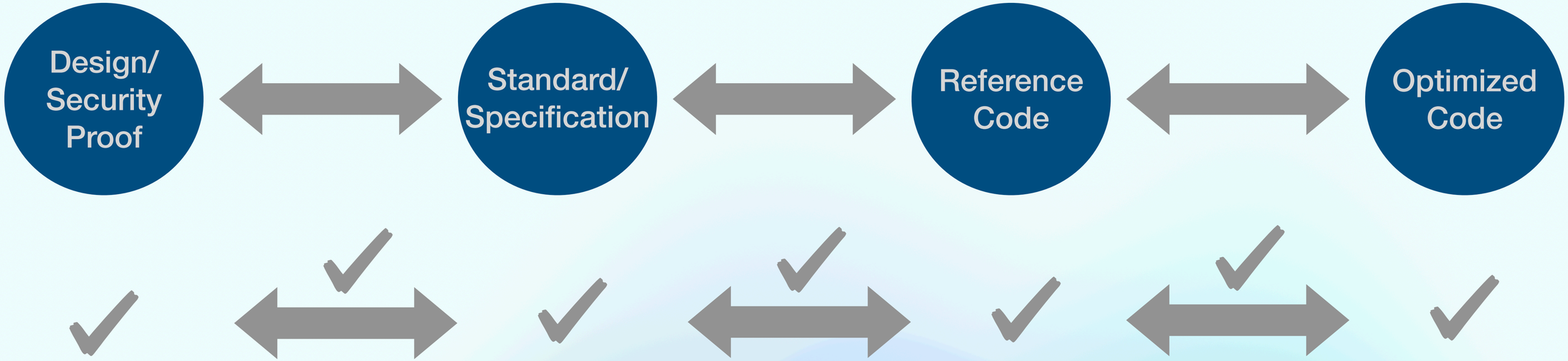
(*) in a formally defined (abstract) leakage model

Agenda

- Introducing formosa-crypto
- Formally verified cryptography landscape
- The formosa-crypto approach
- **Where we are for the PQ FIPS standards**
- Outreach and industrial adoption

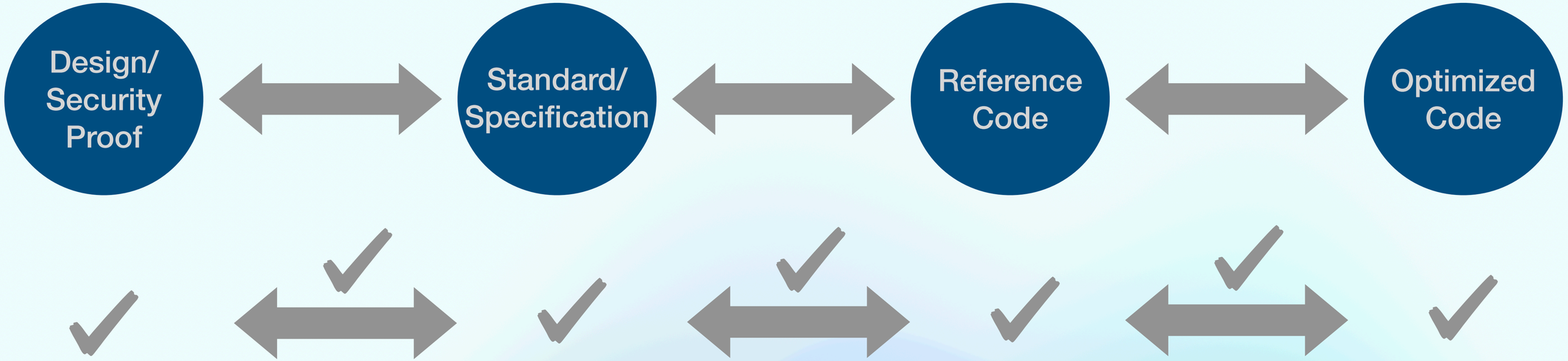
mbb@fc.up.pt

Status Of Proof Chains



ML-KEM

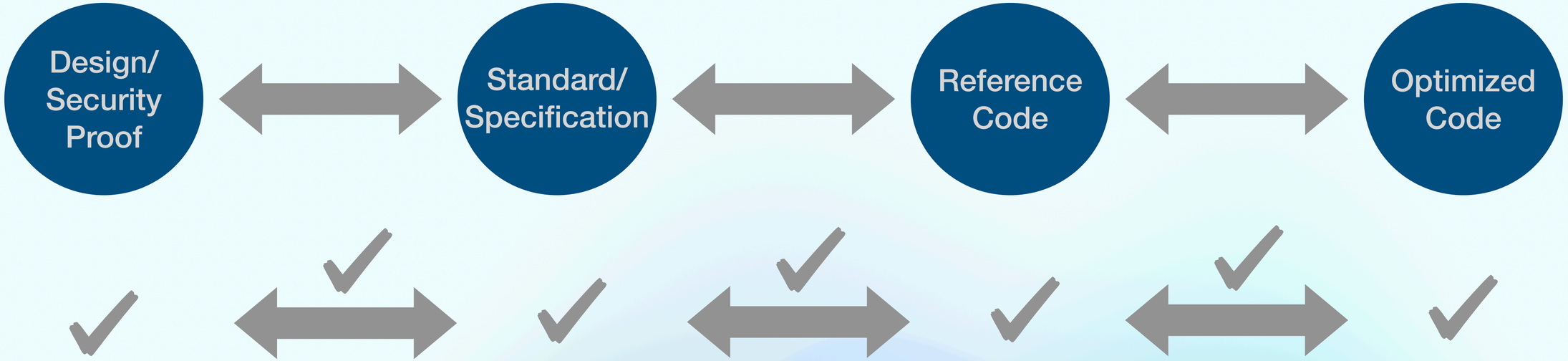
Status Of Proof Chains



ML-KEM

Security proofs are formally verified wrt classical adversaries.

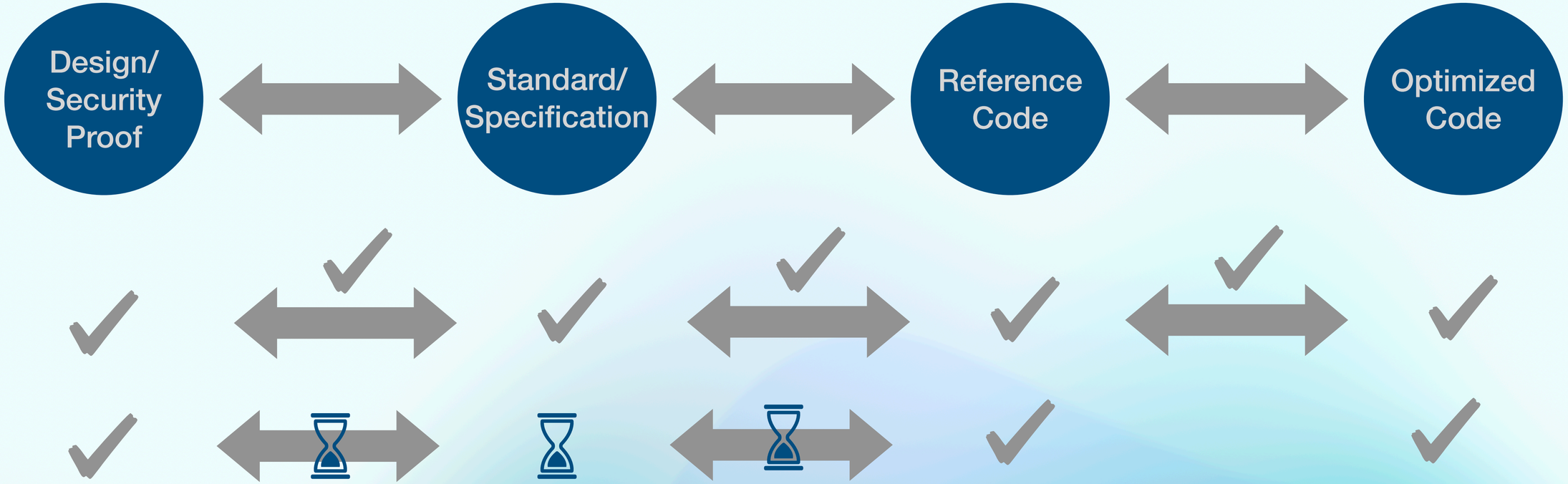
Status Of Proof Chains



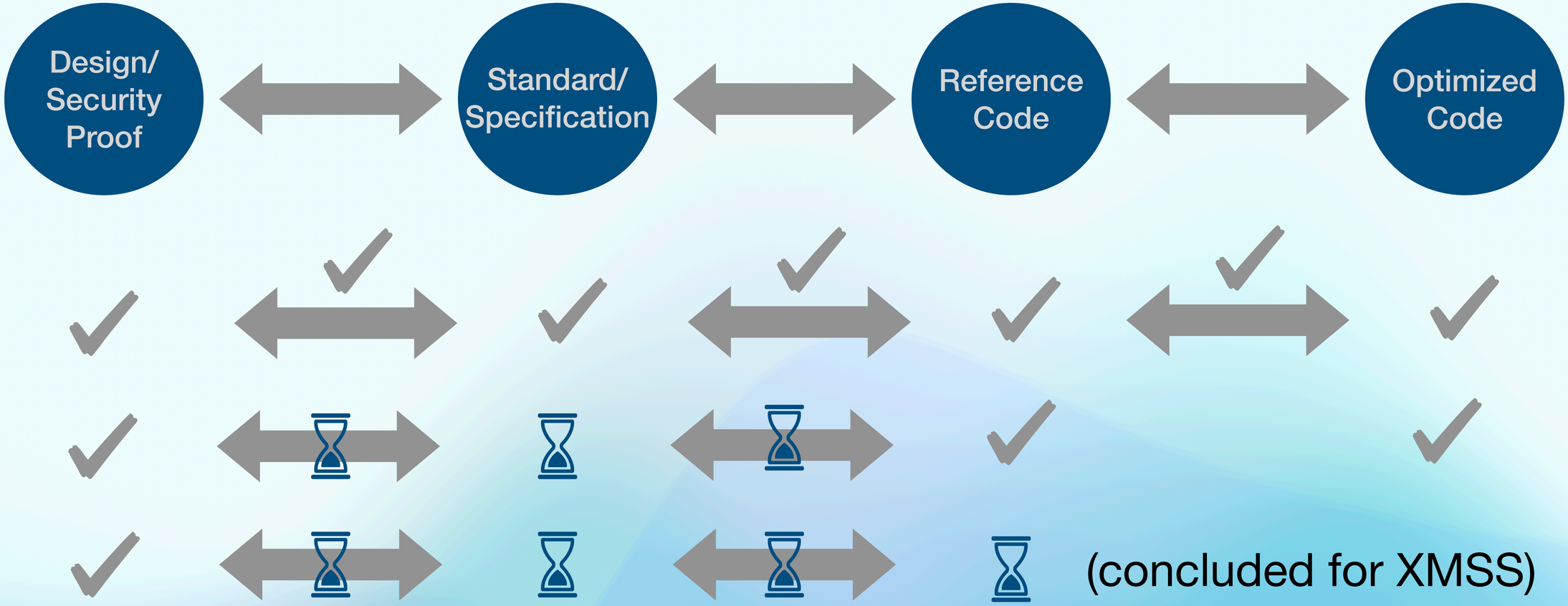
ML-KEM

avx2 code is proved safe, correct, CT and speculative CT.

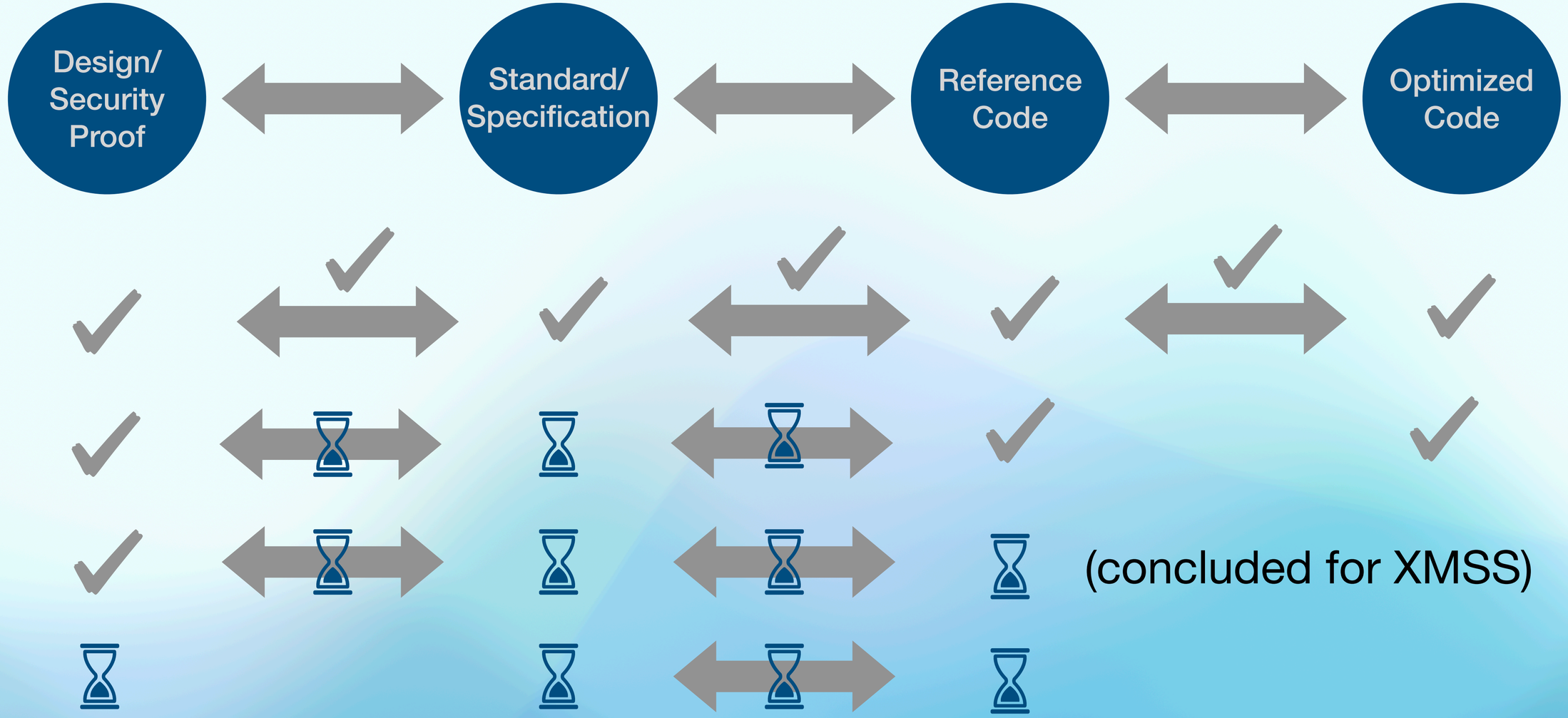
Status Of Proof Chains



Status Of Proof Chains



Status Of Proof Chains



Agenda

- Introducing formosa-crypto
- Formally verified cryptography landscape
- The formosa-crypto approach
- Where we are for the PQ FIPS standards
- **Outreach and industrial adoption**

mbb@fc.up.pt

Outreach and Adoption

- Formosa Crypto code is open-source (cf. formosa-crypto.org)
  - tools: EasyCrypt, Jasmin have their own repositories
  - code: published via PQ Code Package (https://github.com/pq-code-package)
  - proofs: via dedicated repositories (https://github.com/formosa-crypto)
- Industrial adoption
  - Signal (next talk), PQShield (next slide)

PQShield

- Co-steer the formosa-crypto eco-system
- Tool development
  - Lead EasyCrypt development
  - Enable integration with Jasmin
  - Integration with third-party tools: Frama-C, CryptoLine, CAS, SMT solvers
- Apply formosa-crypto methodology to in-house projects:
  - C implementations, masked implementations, integration of cryptographic co-processors

Agenda

- Introducing formosa-crypto
- Formally verified cryptography landscape
- Where we are for the FIPS standards
- Outreach and industrial adoption

Questions?

mbb@fc.up.pt