Implementation security: edhoc-rs

Mališa Vučinić, Thibaut Vandervelden

LAKE @ IETF113

Context

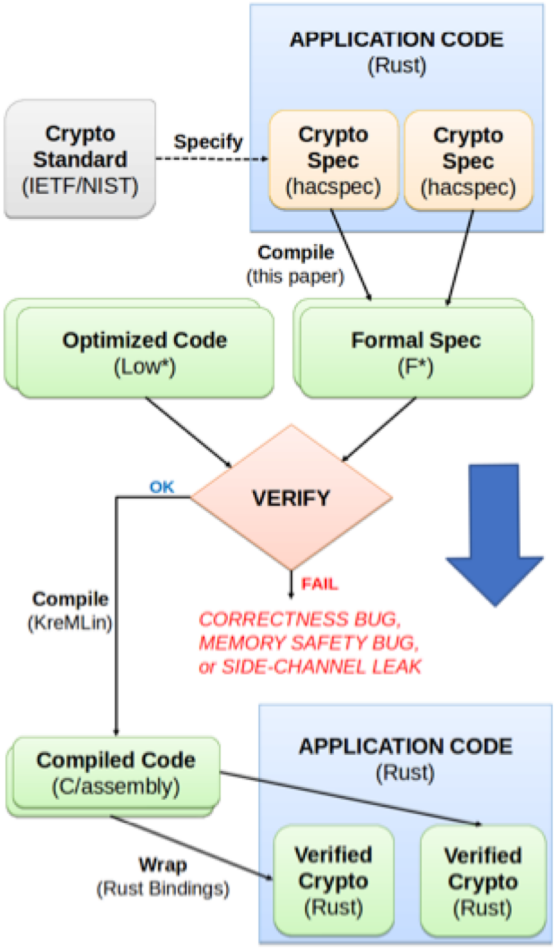
- LAKE targets constrained environments -> embedded systems
- The “embedded programming language” is dominantly C
  - Memory unsafe
- “Protocol is as secure as its implementation”
- Goal: produce implementation with
  - Provable correctness
  - Provable memory safety
  - Side-channel resistance

hacspec [1]

[1] Denis Merigoux, Franziskus Kiefer, Karthikeyan Bhargavan. Hacspec: succinct, executable, verifiable specification for high-assurance cryptography embedded in Rust. https://hal.inria.fr/hal-03176482/document

hacspec Methodology 1/2

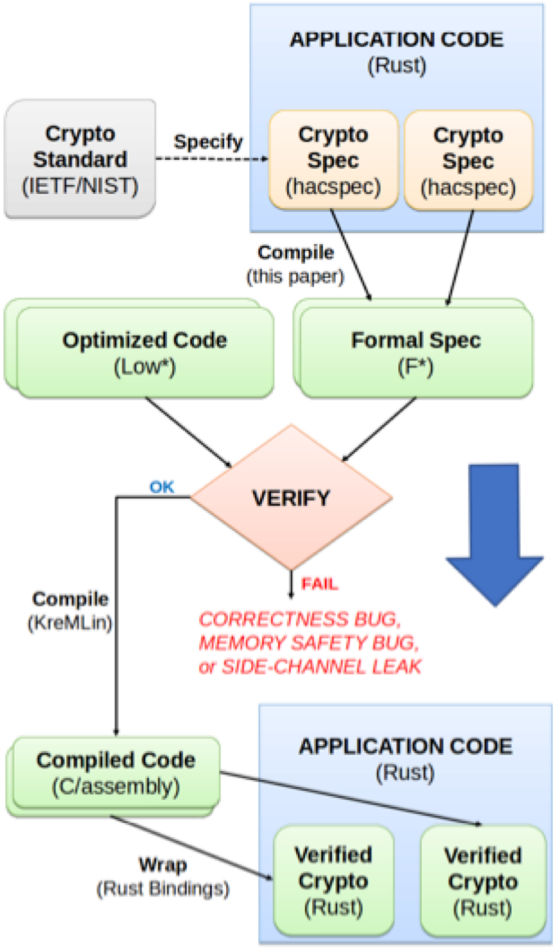
- Specification language is a subset of Rust
- Specification is *executable*
- Specification enables *verification*
- Specification can be compiled to
  1. F*
  2. Coq
  3. Executable



hacspec Methodology 2/2

- Procedure in practice:
  1. Implement draft-ietf-lake-edhoc in Rust/hacspec (manual)
  2. Generate F* model (automatic)
  3. Write and verify Low* implementation against F* (manual)
  4. Compile Low* to verified C (automatic)

https://github.com/hacspec/hacspec



Implementation Goals

- Verifiable code, but for microcontrollers
- “minimal” implementation
  - STAT-STAT Initiator for now
- No dependencies, execute without standard library (i.e. no_std)
- Rely on hardware acceleration where possible
- Initial compilation targets
  - Native
  - CC2538
  - nRF52840
- Portable to other targets

Challenges

- Hacspec relies on Rust's standard library
  - Problematic to port to microcontrollers
- Elliptic curve point representation and API
  - Compact vs compressed vs uncompressed
  - Solved
- Microcontroller hardware abstraction layer support in Rust for popular boards

Current status

https://github.com/openwsn-berkeley/edhoc-rs

Implementation decisions

- Implemented Initiator STAT-STAT
- CCS with integer kid
- Inline CBOR and COSE encoding
- no_std style

Status

- Passes test vectors on native
- Successful interop with californium-edhoc
- WIP: Multi-target build support
- WIP: crypto acceleration in Rust for CC2538

Open Questions and Next Steps

Reflection point

- The implementation serves as a *model* for formal verification
- Important that the implementation models the draft *as close as possible*
- *How to ensure the match between the model and the draft?*

Next Steps

- Complete the port to
  - CC2538
  - nRF52840
- Publish edhoc-rs on crates.io
- Formal verification
  - Generate verified Rust and C code

Acks (in alphabetic order)

- Christian Amsüss
- Karthik Bhargavan
- Franziskus Kiefer
- Denis Merigoux
- Marco Tiloca