IETF 123
CBOR WG meeting
July 24th, 2025

# A Formal Model and Verified Implementation for CBOR, CDDL and COSE

Tahina Ramananandro
taramana@microsoft.com
Microsoft Research
RiSE – Research in Software Engineering

Microsoft

RiSE

everest
Verified End-to-End Secure Transport

# Overview

- EverCBOR: Verified parser and serializer for CBOR
  - Written and proven correct in F*
  - Extracts to C or Rust
- EverCDDL: Verified code generator for CDDL
  - Written and proven correct in F*
  - User provides a CDDL data format description
  - EverCDDL generates C or Rust:
    - Datatype
    - Parser
    - Serializer
  - Enough for COSE and other protocols
- Open-source: https://github.com/project-everest/everparse

ACM CCS 2025 (accepted for publication, to appear)

More detailed presentation at the side meeting

# Proof-Oriented Programming At Scale

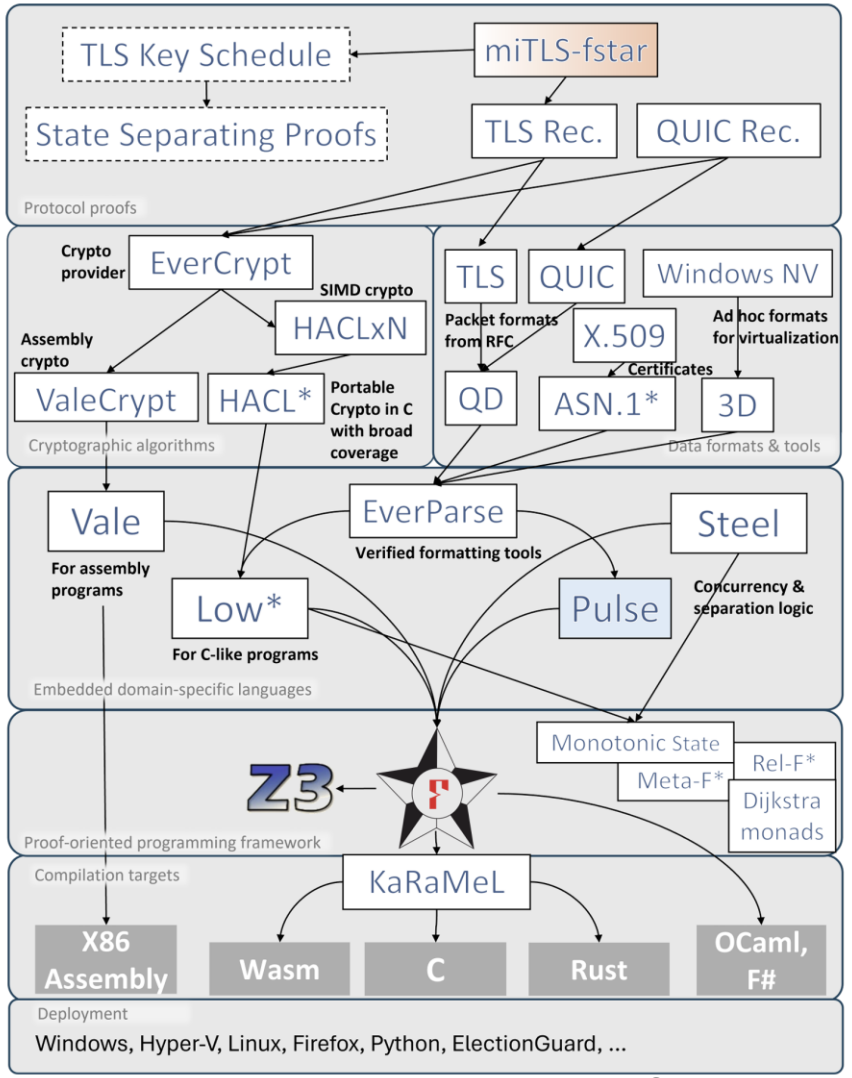
Project Everest (2016-2021)

- ~1 million lines of F* code
- Cryptography, verified parsers, …
- Deployed in Linux, Firefox, Python, Windows, Azure, …
- https://project-everest.github.io/

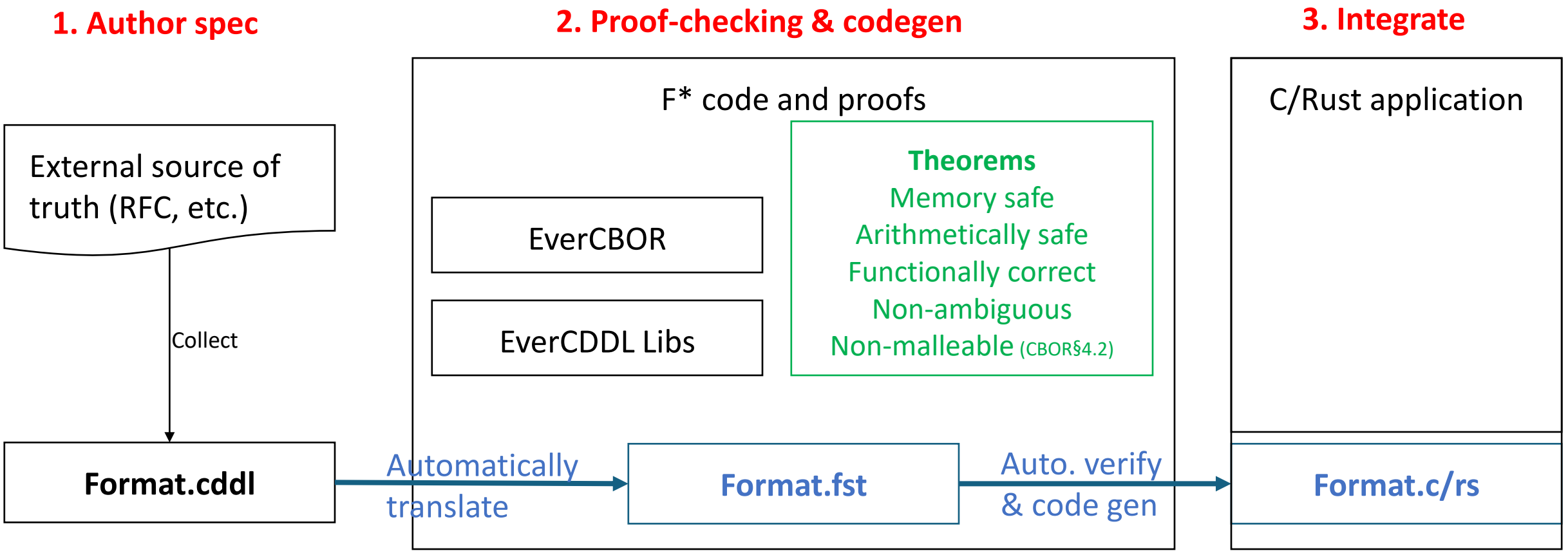
Everest offspring (including F*, EverParse) grown up as active standalone projects

EverCBOR, EverCDDL are part of EverParse

- Broader talk at UFMRG on Thursday 7/24



# EverCBOR and EverCDDL



# Implementation

- Currently, deterministic encoding without floating-point
  - Support for non-deterministic & floating-point in progress
- No heap allocation
  - Validation in constant stack space
  - (impossible for non-deterministic CBOR with maps in map keys)
  - Parsers provide iterators for arrays, maps
- Secure serialization API
  - No preconditions on input values to be serialized
  - Serializer gracefully fails instead of writing ill-formed map, etc.
  - Partial compile-time disjointness checker
- Performance in the ballpark of qcbor, tinycbor
  - Some speedup in map lookup for deterministic encoding
  - Some slowdown due to handling of nesting and lack of specialization
- Compare with zcbor (thanks Thomas Fossati)
  - Perf comparison TODO
  - Does not check map ordering

# EverCDDL Applicability (IETF 123 Hackathon)

- We support most of:
  - COSE (RFC 9052)
  - CWT, EAT (RFC 9711)
  - draft-ietf-ace-edhoc-oscore-profile-08
  - draft-ietf-core-href-23
  - draft-ietf-core-observe-multicast-notifications-12
  - TODO test with draft-lenders-dns-cbor-14 – thanks Martine S. Lenders
  - …
- Some manual rewrites due to currently unsupported features:
  - Syntactic transformations: ~, &
  - JC<_, _>, CBOR_ONLY<_>, other generics
  - Occurrence bounds
  - Disable floating-point
  - Unfold bounded recursion in COSE
- Currently, no support for recursion
  - Potential stack exhaustion attacks
  - Needed by EAT Nested tokens, submodules

# Some inaccuracies in existing CDDL descriptions

```
header_map = {
  ?     1 => int / tstr,   ; algorithm identifier
  ?     2 => [+label],     ; criticality
  ?     3 => tstr / int,   ; content type
  ?     4 => bstr,         ; key identifier
  ?   ( 5 => bstr //       ; IV
        6 => bstr ),       ; Partial IV
  * label => values
}
```

- Missing cuts next to extension tables
  - COSE header_map. Erratum to be submitted
  - EDHOC_Information
  - DPE (DICE Protection Environment, Trusted Computing Group)
  - …
- PEG interpretation
  - COSE Sig_structure, etc.

```
Sig_structure = [
    context : "Signature" / "Signature1",
    body_protected : empty_or_serialized_map,
    ? sign_protected : empty_or_serialized_map,
    external_aad : bstr,
    payload : bstr
]
```

- Some constraints can be added but are not
  - COSE header_map entries for keys 5 and 6 cannot appear together

# EverCBOR, EverCDDL Takeaways

- Interoperable, reasonable implementations with strong formal guarantees
  - No need to hand-write parsers/serializers for CDDL-defined formats anymore
- Improved understanding of the CBOR deterministic encoding and of CDDL
- Clarification needed in CDDL descriptions in some existing standards
  - For use with EverCDDL and other automated code generators

- https://github.com/project-everest/everparse
- Tahina Ramananandro, taramana@microsoft.com