draft-irtf-cfrg-cpace

Michel Abdalla, Björn Haase, Julia Hesse

Status IETF 110, March 2021

Status security analysis: completed

Paper available here:
https://eprint.iacr.org/2021/114 - submitted to ACM CCS 2021

Results in a nutshell

- CPace protocol variants in current draft provide strong security guarantees:
    - Composability & adaptive security under sSDH (DDH-type assumption)
    - Map2Point security analyzed without random oracle
        - security-relevant properties are fulfilled by Hash2Curve maps
        - potentially relevant news for VOPRF draft: non-uniform maps suitable and secure
    - CPace using Ristretto 25519 secure as well
    - Cofactor does not impact security

Next steps

Plan for implementation and RFC summarized in appendix F of https://eprint.iacr.org/2021/114.

Objective: Minimize need for adaption when integrating into existing “ eco-systems”

Three “eco-systems” identified => three tailored variants:

- Single-coordinate ladders X25519 (X448)
- Group abstractions Ristretto25519 (Decaf448)
- Short-Weierstrass (e.g. NIST-P256)

X25519 / X448

- Non-uniform Elligator2 map without co-factor clearing
- CPace implementation will need only X25519 (X448) and Elligator2
- Non-uniform sampling of secret scalars (aka. “scalar clamping”) secure
- No need for explicit point verification code:
  Use twist security and check for neutral elements.

=> Existing X25519 / X448 implementations suitable for CPace “as-is”.

Group abstractions Ristretto25519/Decaf448

- Use built-in uniform “map-twice-and-add” construction that comes “batteries included” with the abstraction.
- Slightly non-uniform sampling of secret scalars secure for Curve25519 and Curve448. (e.g. no need for rejection sampling for uniform scalars)
- Point verification enforced by the abstraction’s encode/decode primitives

=> Will make CPace implementation most straight-forward for ristretto25519

(Plan: use go implementation found at https://github.com/FiloSottile/go-cpace-ristretto255 for deriving test vectors)

Short Weierstrass

- Use non-uniform Encode2Curve construction from hash2curve draft
- Conventional point verification mandatory
- Uniform sampling of secret scalars mandatory on most curves, specifically also for NIST-P256. Care needed when re-using libraries for conventional ECDH.
- Use only X-Coordinate of DH results for deriving session key

=> Not considering Y-Coordinate for deriving session keys allows for Single-Coordinate ladder scalar multiplication strategies also for CPace

Status implementations

Several new Implementation prototypes in go and rust online, could provide a basis for Ristretto25519 test vectors for CPace.

We are currently actively looking for partners for CPace integration in different crypto libraries, e.g. as student projects in addition to reference implementations written by the editor team:

- Real-world test for readability and completeness of RFC draft
- Have different implementations for inter-operability tests.

Status test vectors

Currently only X25519-based test vectors available.

Feedback regarding different X25519 variants incorporated:
Implementations “out in the wild” differ with respect to the aspect whether published field elements must be fully reduced to 255 bit.

Interacting with the other CFRG working groups

- Proof strategy for CPace could be transferred for analysis for DH-OPRF function, as e.g. used in OPAQUE:
  - Avoiding the need of modeling the map construction as RO.
  - Will most likely allow for more efficient single-coordinate ladders using non-uniform maps.

- Non-uniform Encode2Curve primitive without cleared co-factor would be desireable from Hash2Curve draft. (Might be re-used for an optimized VOPRF)

- Let’s join forces for reference implementations with VOPRF and Hash2Curve.