draft-irtf-cfrg-cpace

Michel Abdalla, Björn Haase, Julia Hesse

Status IETF 112, November 2021

- Update Security analysis
- Updates regarding the ID

Status update: Security analysis

Two new/updated papers available online:

[1] https://eprint.iacr.org/2021/114 Security analysis of CPace
[2] https://eprint.iacr.org/2021/696 The ‘quantum annoying’ property of PAKE protocols

Results in a nutshell

- CPace secure for initiator/responder setting with fixed message sequencing
- CPace also secure for parallel setting without enforced message order
- CPace features strong ‘quantum annoying’ properties
- CPace secure with and without pre-established session id *sid*

[2] “The “quantum annoying” property …”

E. Eaton, D. Stebila: https://eprint.iacr.org/2021/696

- Considers security of the idealized CPace_base in the generic-group model.
- Based on a variant of the game-based BPR model.
- Considers a “quantum”-adversary with access to discrete log oracle queries.

> As a preview of our theorem, the probability that an adversary manages to win the game is dominated by a term $(q_C + q_D)/N$ term, where q_C is the number of online interactions, q_D is the number of discrete log oracle queries, and N is the size of the password space. This lines up exactly with the intuitive guarantees we would expect a quantum annoying system to have: guess a password and try using it in an active session, or guess a password and take a discrete logarithm based on it to see if it was the password used in a passively-observed session.

[1] “Security analysis of CPace”

M. Abdalla, B. Haase, J. Hesse: https://eprint.iacr.org/2021/114

Paper now accepted at ASIACRYPT 2021

- Refined security analysis of CPace protocol variants in the associated paper
  - Clarification of security definitions and proofs
  - Both, “initiator-responder” and “parallel” (no message sequence enforced) setting covered
  - Improved readability
  - Clarification of the role of the session id (*sid*)

[1] “Security analysis of CPace”

M. Abdalla, B. Haase, J. Hesse: https://eprint.iacr.org/2021/114

- New Appendix B: Game-Based proof complementing the simulation-based analysis
  - ROR-Security Model extended to the case of the “parallel” protocols
  - Explicit consideration of CPace without pre-established session id values

Conclusion. Our security analysis of CPace in the UC framework in Section 5 requires that, prior to entering the protocol, a pre-established unique session identifier *sid* is available to both parties. As the above game-based security analysis highlights, CPace features also strong albeit different[8] security guarantees if there is no such *pre*-established *sid*.

[1] “Security analysis of CPace”

M. Abdalla, B. Haase, J. Hesse: https://eprint.iacr.org/2021/114

- Security analysis of different variants V that each come with the following methods
  - V.ScSam(): Scalar sampling method
  - V.Gen(str): string -> generator mapping
  - V.ScMul(scalar,generator): Scalar multiplication method
  - V.ScMulVf(scalar, remote_point): Scalar multiplication with point verification
- Appendix G: Security analysis provided for tailored protocol variants for different “ecosystems” with respectively tailored set of the above functions.
  - CPace for the Short-Weierstrass ecosystems (e.g. NIST-P256, Brainpool)
  - CPace for Single-Coordinate ladders on Montgomery curves (e.g. X25519)
  - CPace for idealized group abstractions (e.g. ristretto255, decaf448)

Session identifier

Users are recommended to first agree on a joint session identifier [3]

- Both users should contribute randomness to the *sid*
- Agreement does not require secrecy
- Can potentially be piggy-backed to messages sent by the application

CPace is secure also without *sid* but pre-established *sid* makes a difference regarding …

- UC composability guarantees [1]. Binds a CPace run to one single session.
- The level of “quantum annoyance” guarantees. (See [1] Appendix B, [2] Appendix A.2)

[1] https://eprint.iacr.org/2021/114 Security analysis of CPace
[2] https://eprint.iacr.org/2021/696 The ‘quantum annoying’ property of PAKE protocols
[3] https://eprint.iacr.org/2004/006 Protocol initialization for the framework of universal composability

Status update: ID

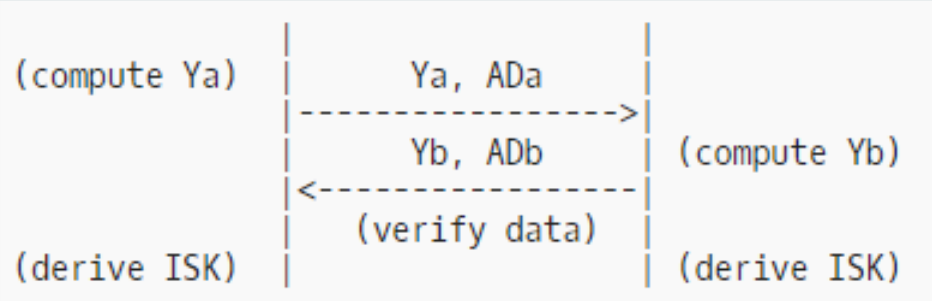
Major rewrite: Focus text for the audience of the "implementer" and refer the theoretically inclined reader to the scientific papers for technical details!
(Working copy of the editor team here)

- Current github version considers both, parallel and initiator/responder setting
- First gives a generic definition of the Protocol
- Then details specific choices to use for the specific "ecosystems"
  - Short-Weierstrass (e.g. NIST-P256, Brainpool)
  - Single-Coordinate ladders on Montgomery curves (e.g. X448, X25519)
  - Idealized group abstractions (e.g. ristretto255, decaf448)
- Test vector generation code is currently rewritten completely.
  Plan: Update ID once this step is completed.

Status update: ID – Associated Data fields AD

Change to the previous state …

- Added additional OPTIONAL associated data fields ADa and ADb to the protocol messages as used within the game-based security analysis [1], Appendix B.



This also allows for applications where un-ambiguous party identifier encodings are not available at protocol start.

[1] https://eprint.iacr.org/2021/114 Security analysis of CPace

Status update: ID - Hashing

Change: Length of hashed substrings now prepended to all subfields prior to hashing

- Important for parsing messages (Ya, ADa) and (Yb, ADb) by the receiver
- Helpful for buffer-overflow sanity checks
- Helpful security-wise:

  Asserts prefix-free encoding [4] for all hash inputs, such that length-extension imperfections of Merkle-Damgard Hash constructions don't become an issue.

  (Security analysis [1,2] model hash functions H: {1,0}* -> {1,0}$^{\lambda}$ as idealized random oracles, and don't consider Merkle-Damgard imperfections!)

[4] https://iacr.org/archive/crypto2005/36210424/36210424.pdf
J.-S. Coron, Y. Dodis, C. Malinaud, P. Puniya: "Merkle Damgard revisited: How to construct a hash function"

Status update: ID - Test Vectors

Current draft on github repo does not yet include the new test vectors.

- Sage code largely rewritten
- Proof of concept implementations in sagemath for the different primitives.
- New: Follow directory structure and build system used for h2c and OPRF drafts.

- Goal: Automatically generate markdown sources for test vectors from sage scripts.
- **New ID-revision upload planned as soon as test-vector generation code works.**

Acknowlegements / feedback appreciated

- **@Christopher Wood: Thank you for your help with XML->Markdown conversion!**

Feedback/Hints appreciated, specifically regarding …

- Readability of “Object-style” notation for “hash primitive” and “group ecosystem” (G.sample_scalar(), G.scalar_mult(), G.DSI, …, H.b_in_bytes, H.s_in_bytes) ?
- Should we explicitly consider both, initiator/responder + parallel version or only focus on one setting for conciseness ?
- Best way for prepending field lengths to octet strings. (“prefix-free encoding”) Current suggestion: encode length as UTF8.
- Github integration of automagical markdown -> .html, .txt conversion.