Objectives of the editor team (Michel Abdalla, Björn Haase, Julia Hesse)

- RFC document
- Provide public code / scripts for generation of test vectors
- Provide a public reference implementation for some variants.
<table>
<thead>
<tr>
<th>Draft text currently refers generically to “groups” (“groups modulo negation”). =&gt; We plan to change this such that it explicitly refers to groups on elliptic curves only.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Currently two types of elliptic curves are considered in the draft</td>
</tr>
<tr>
<td>- Single-coordinate Diffie-Hellman X448 and X25519 on Edwards/Montgomery curves with small co-factor</td>
</tr>
<tr>
<td>- X-Coordinate-only Diffie-Hellman on Short-Weierstrass curves such as NIST-P256</td>
</tr>
<tr>
<td>- We plan to add also full-group (i.e. both coordinates) support, e.g. for Ristretto25519</td>
</tr>
<tr>
<td>Slight variations between the three alternative implementation variants, specifically regarding the properties imposed on the Map2Point algorithm.</td>
</tr>
<tr>
<td>Explicit security analysis of all resulting slight variations considered mandatory as a first step.</td>
</tr>
</tbody>
</table>
Tight bounds can be established by using the approach used for TBPEKE for GAP SDH and GAP CDH problems
- Most likely a weaker assumption set applies (Strong DH / Strong SDH problems instead of GAP CDH / GAP SDH)

For curves with co-factor, security can be reduced to hardness of the problems on the prime-order subgroup

Important formal requirement for the map: "For any point $G$ not on the image of Map2Point there must be an efficient algorithm for finding an exponent $y$ such that $\text{scalar_mult}(G, y)$ is on the map."
- Property provided by all currently discussed mappings from Hash2Curve and Ristretto25519 (every second exponent $y$ will work on average)
- With this property the SDH problem on the subset of points produced by Map2Point can be reduced to hardness of SDH on the group (or the prime-order subgroup respectively).

Discussion of these aspects not considered suitable for the RFC. A separate paper extensively discussing the above points is in work as a first step. (Currently the main aspect in our weekly meeting in the editor team.)
Presently reference implementation and test vector generation for X25519/Elligator2 available

- Separate implementations in Sagemath, Python and ANSI-C currently still held at https://github.com/BjoernMHaase/AuCPace
- GIT-Repository under the hood of CFRG was setup. CPace-specific code still needs to be transferred to this repository.

Next steps: Similar reference implementation and test vector generation for Short-Weierstrass and Ristretto25519

- At least Sagemath, preferably also generic Python and ANSI-C

Any suggestion / support regarding small self-contained libraries used for reference implementations and test-vector generation for Short-Weierstrass and Ristretto25519 implementations in ANSI-C and Python would be welcome.
We organized our team. Unfortunately we feel slowed down significantly by the pandemic.

Current activity: Provide tight security bounds and reduction to accurately specified assumptions for all possible the tiny implementation differences (e.g. Short-Weierstrass, X448/X25519, Ristretto25519)

Next important step: Scripts for test vectors for all alternative implementations