Hashing to Elliptic Curves

draft-sullivan-cfrg-hash-to-curve

Nick Sullivan (nick@cloudflare.com)
Christopher A. Wood (cawood@apple.com)

CFRG
Background

Hashing to elliptic curves is common

- Simple Password Exponential Key Exchange [Jablons96]
- Password Authenticated Key Exchange [BMP00]
- Boneh-Lynn-Shacham signatures [BLS01]
- Verifiable Random Functions (VRFs) [draft-irtf-cfrg-vrf]
- Privacy Pass [https://privacypass.github.io]
Try-and-Increment

1. $\text{ctr} = 0$
2. $h = \text{"INVALID"}$
3. While $h$ is "INVALID" or $h$ is EC point at infinity:
   A. $\text{CTR} = \text{I2OSP}(\text{ctr}, 4)$
   B. $\text{ctr} = \text{ctr} + 1$
   C. $\text{attempted\_hash} = \text{Hash}(m \mid \mid \text{CTR})$
   D. $h = \text{RS2ECP}(\text{attempted\_hash})$
   E. If $h$ is not "INVALID" and cofactor > 1, set $h = h^{\text{cofactor}}$
4. Output $h$

Make sure $h$ is in the prime order subgroup
(Non-)Requirements

Requirements

• Constant-time
• ...

Non-requirements

• Invertible
# Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Icart</td>
<td>$q = 2 \mod 3$</td>
</tr>
<tr>
<td>SWU</td>
<td>None</td>
</tr>
<tr>
<td>Simplified SWU</td>
<td>$q = 3 \mod 4$</td>
</tr>
<tr>
<td>Elligator2</td>
<td>q is large, has a point of order 2, and $j$-invariant $\neq 1728$</td>
</tr>
</tbody>
</table>
Interface & Notation

$$H2C(\alpha) : \{0, 1\}^+ \rightarrow E$$

- $\alpha = \text{arbitrary input}$
- $q = \text{prime order of base field}$
- $u = \text{point of order 2 (Elligator2)}$
- $f(x) = \text{curve equation}$
- $H(\alpha) = \text{hash to prime order subgroup}$
Icart

\[ t = H(\alpha) \]
\[ v = \left( (3A - t^4) / 6t \right) \]
\[ x = \left( v^2 - b - (t^6 / 27) \right)^{1/3} + (t^2 / 3) \]
\[ y = tx + v \]

Output \((x, y)\)
Elligator2

\[ r = H(\alpha) \]
\[ d = -A/(1 + ur^2) \]
\[ e = f(d)^{(p-1)/2} \]
\[ u = ed - (1-e)A/u \]

Output \((u, f(u))\)
(Current) Recommendations

<table>
<thead>
<tr>
<th>Curve</th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>P-256</td>
<td>Simplified SWU</td>
</tr>
<tr>
<td>P-384</td>
<td>Icart</td>
</tr>
<tr>
<td>Curve25519</td>
<td>Elligator2</td>
</tr>
<tr>
<td>Curve448</td>
<td>Elligator2</td>
</tr>
</tbody>
</table>
Open Tasks

• Complete cost analysis

• Add SWU details and implementation

• Include security reductions where possible

• Interface details: octet strings to integer point encodings

• Produce verifiable implementations

• Clarify mappings that are reversible — this is not always desirable!
Open Issues

- Always multiply by cofactor?
- How close to indistinguishable from random points is needed?
Simplified SWU

\[ t = H(\alpha) \]
\[ x = -t^2 \]
\[ x_2 = \left(-\frac{b}{a}\right) \cdot (1 + \frac{1}{(t^2 + t)}) \]
\[ x_3 = t \cdot x_2 \]
\[ h_2 = f(x_2) \]
\[ h_3 = f(x_3) \]

Output \((x_2, h_2^{(q+1)/4})\) if \(h_2\) is square, else \((x_3, h_3^{(q+1)/4})\)