ristretto255 & decaf448

draft-irtf-cfrg-ristretto255-decaf448
what problem are we solving?
2.4 Notation

Let $G$ denote a cyclic group of prime order $p$, 
what kind of elliptic curve?
what kind of elliptic curve?

weierstrass
e.g. secp256k1

edwards
e.g., curve25519, fourq
what kind of elliptic curve?

weierstrass
  e.g. secp256k1

edwards
  e.g., curve25519, fourq

prime order

✅

❌
what kind of elliptic curve?

<table>
<thead>
<tr>
<th></th>
<th>weierstrass</th>
<th>edwards</th>
</tr>
</thead>
<tbody>
<tr>
<td>e.g. secp256k1</td>
<td>![✓]</td>
<td>![✗]</td>
</tr>
<tr>
<td>e.g., curve25519, fourq</td>
<td>![✗]</td>
<td>![✓]</td>
</tr>
<tr>
<td>prime order</td>
<td>![✓]</td>
<td>![✗]</td>
</tr>
<tr>
<td>fastest formulas</td>
<td>![✗]</td>
<td>![✓]</td>
</tr>
</tbody>
</table>
what kind of elliptic curve?

weierstrass  
- prime order  ✔️  ❌  
- fastest formulas  ❌  ✔️  
- complete formulas  😭  ✔️

edwards  
- prime order  ❌  ✔️  
- fastest formulas  ✔️  ✔️  
- complete formulas  ✔️  ✔️
<table>
<thead>
<tr>
<th>Feature</th>
<th>Weierstrass</th>
<th>Edwards</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prime order</td>
<td>✅</td>
<td>❌</td>
</tr>
<tr>
<td>Fastest formulas</td>
<td>❌</td>
<td>✅</td>
</tr>
<tr>
<td>Complete formulas</td>
<td>😭</td>
<td>✅</td>
</tr>
<tr>
<td>Easy in constant time</td>
<td>❌</td>
<td>✅</td>
</tr>
</tbody>
</table>
okay, it’s only a small conceptual mismatch...
...so what’s wrong with a small cofactor?
well...
well...

security analysis for the abstract protocol does not apply to its concrete implementation
security analysis for the abstract protocol does not apply to its concrete implementation

subgroup validation is expensive, negating any speedup
well...

security analysis for the abstract protocol does not apply to its concrete implementation

subgroup validation is expensive, negating any speedup

ad-hoc protocol tweaks are specific to each protocol
tweaks cause subtle “features”
tweaks cause subtle “features”

e.g., rfc8032 does not require ed25519 implementations to agree on whether a signature is valid
tweaks cause subtle “features”

e.g., rfc8032 does not require ed25519 implementations to agree on whether a signature is valid

• different behaviour between batch, single verification
tweaks cause subtle “features”

e.g., rfc8032 does not require ed25519 implementations to agree on whether a signature is valid

- different behaviour between batch, single verification

- very bad for applications involving consensus
tweaks cause subtle “features”

e.g., rfc8032 does not require ed25519 implementations to agree on whether a signature is valid

• different behaviour between batch, single verification

• very bad for applications involving consensus

• also incompatible with hierarchical key derivation
…or catastrophic failures
how do we fix this mismatch?
decaf & ristretto
what are decaf & ristretto?
what are decaf & ristretto?

construction, by mike hamburg, of a prime-order group
what are decaf & ristretto?

collection, by mike hamburg, of a prime-order group
uses a non-prime-order curve internally, no overhead
what are decaf & ristretto?

construction, by mike hamburg, of a prime-order group uses a non-prime-order curve internally, no overhead canonical, non-malleable encoding of group elements
what are decaf & ristretto?

construction, by mike hamburg, of a prime-order group uses a non-prime-order curve internally, no overhead canonical, non-malleable encoding of group elements “batteries included”: a single hash-to-group method
how does this work?
three families of curves
montgomery curves
montgomery curves

\[ M_{B, A} : \quad Bu^2 = v(v^2 + Av + 1) \]
montgomery curves

\[ M_{B,A} : \quad Bu^2 = v(v^2 + Av + 1) \]

support fast “pseudomultiplication” \( v(P) \mapsto v([k]P) \)
montgomery curves

\[ M_{B,A} : \quad Buv^2 = v(v^2 + Av + 1) \]

support fast “pseudomultiplication” \( v(P) \mapsto v(\ell kJP) \)

require few constraints in circuits
edwards curves
edwards curves

\[ E_{a,d} : \ ax^2 + y^2 = 1 + dx^2y^2 \]
Edwards curves

\[ E_{a,d} : ax^2 + y^2 = 1 + dx^2y^2 \]

birationally equivalent to Montgomery curves
edwards curves

\[ E_{a,d} : \ ax^2 + y^2 = 1 + dx^2y^2 \]

birationally equivalent to montgomery curves

fastest known formulas for curve operations
edwards curves

\[ E_{a,d} : \ ax^2 + y^2 = 1 + dx^2 y^2 \]

birationally equivalent to montgomery curves

fastest known formulas for curve operations

formulas allow parallelism inside a curve operation
jacobi quartic curves
jacobi quartic curves

\[ S_{e,A} : \quad \ell^2 = e s^4 + 2 A s^2 + 1 \]
jacobi quartic curves

\[ \mathcal{E}_{e,A} : \quad t^2 = es^4 + 2As^2 + 1 \]

easy to write down 4 points of order 2
jacobi quartic curves

\[ S_{e, A} : \quad t^2 = es^4 + 2As^2 + 1 \]

easy to write down 4 points of order 2

we can efficiently encode \((s, t) \mod S[2]\)
linking curves with isogenies
linking curves with isogenies

\[ E_{a,d} \rightarrow S_{a^2, a-2d} \rightarrow M_{a, 2-4d/a} \rightarrow E_{a', d'} \]
encoding with isogenies
encoding with isogenies

specify an encoding on the Jacobi quartic
encoding with isogenies

specify an encoding on the jacobi quartic

isogenies “transport” the encoding to other curve shapes
encoding with isogenies

specify an encoding on the jacobi quartic

isogenies “transport” the encoding to other curve shapes

extra step to handle cofactor 8 instead of cofactor 4
decaf vs ristretto
decaf vs ristretto
decaf vs ristretto

decaf transports the encoding to Edwards directly
decaf vs ristretto

decaf transports the encoding to Edwards directly.

Ristretto transports the encoding to Edwards via Montgomery + cofactor $8$.
decaf vs ristretto

decaf transports the encoding to edwards directly

ristretto transports the encoding to edwards via montgomery + cofactor 8

either can use any curve internally
concrete parameterizations
ristretto255 can be implemented using curve25519
ristretto255

can be implemented using curve25519

~128-bit security level
ristretto255

can be implemented using curve25519

~128-bit security level

increasing adoption: zk proofs, psi, pake, etc
decaf448
decaf448 can be implemented using edwards448
decaf448 can be implemented using edwards448

~224-bit security level
decaf448

can be implemented using edwards448

~224-bit security level

suitable where ed448 would be used
current status
current status

test vectors achieve “coverage” of all edge cases
current status

test vectors achieve “coverage” of all edge cases

language was rewritten last year to address feedback
current status

test vectors achieve “coverage” of all edge cases

language was rewritten last year to address feedback

added decaf448 parameters
current status

test vectors achieve “coverage” of all edge cases

language was rewritten last year to address feedback

added decaf448 parameters

no outstanding issues, ready to go