Oblivious Pseudorandom Functions (OPRFs) using Prime-Order Groups

Alex Davidson
Armando Faz Hernández
Nick Sullivan
Christopher Wood

Sources
github.com/cfrg/draft-irtf-cfrg-voprf

Datatracker
datatracker.ietf.org/doc/draft-irtf-cfrg-voprf
OPRF: Oblivious PseudoRandom Function

Two-party 1-RRT protocol between a server and a client.

Client holds some input $x$  
Server holds a key $k$

Both parties cooperate in computing:

\[ y = \text{PRF}(k, x) \]

**Oblivious**  
Client learns $y$, but nothing about $k$  
Server does not learn anything about $x$ or $y$

**Verifiable**  
Client can verify that $y$ was computed using $k$.  
Server commits to the key $k$, and gives a proof.
**Goal:** Client gets $\text{output} = \text{PRF}^\text{SKS}(\text{input})$.

**Client**($pkS$, $\text{input}$, $\text{info}$) **Server**($skS$, $pkS$)

blind, blindedElement = $\text{Blind}(\text{input})$

blindenedElement

evaluatedElement, proof = $\text{Evaluate}(skS, pkS, \text{blindedElement})$

evaluatedElement, proof

unblindedElement = $\text{Unblind}(\text{blind}, \text{evaluatedElement}, \text{blindedElement}, pkS, \text{proof})$

**output** = $\text{Finalize}(\text{input}, \text{unblindedElement}, \text{info})$
Latest Changes

- Ciphersuites for ristretto255 and decaf448.
- Updated hash-to-group and hash-to-scalar details hashing to groups.
- Additive Blinding (faster blinding).
- Complete specification of ciphersuite parameters.
- Test vectors available.
- Editorial improvements.
OPRF: Proposed API

Client:
  SetupClient
  Blind
  Unblind
  VerifyProof
  Finalize

Server:
  SetupServer
  Evaluate
  VerifyFinalize*
  FullEvaluate*
Ciphersuites

Document provides suites based on elliptic curves.

<table>
<thead>
<tr>
<th>Suite Identifier</th>
<th>Group</th>
<th>Hash Function</th>
<th>Security Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>0001</td>
<td>ristretto255</td>
<td>SHA-512</td>
<td>128</td>
</tr>
<tr>
<td>0002</td>
<td>decaf448</td>
<td>SHA-512</td>
<td>224</td>
</tr>
<tr>
<td>0003</td>
<td>P-256</td>
<td>SHA-256</td>
<td>128</td>
</tr>
<tr>
<td>0004</td>
<td>P-384</td>
<td>SHA-512</td>
<td>192</td>
</tr>
<tr>
<td>0005</td>
<td>P-521</td>
<td>SHA-512</td>
<td>256</td>
</tr>
</tbody>
</table>
Implementations

Reference Code: Sage/Python test vector generator

https://github.com/cfrg/draft-irtf-cfrg-voprf/tree/master/poc

Other Implementations:
- Go:
  - bytemare/voprf
  - cloudflare/circl
  - alxdavids/voprf
- Rust:
  - alxdavids/voprf
- C:
  - BoringSSL
Questions?

Ready for RGLC?