Oblivious Pseudorandom Functions (OPRFs) using Prime-Order Groups

Alex Davidson, Armando Faz Hernández, Nick Sullivan, Christopher Wood
draft-irtf-cfrg-voprf@ietf.org

**OPRF: Oblivious Pseudorandom Function**

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

Client holds some input \( x \)  
Server holds a private key \( k \)

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

**Oblivious**

Client learns \( y \), but nothing about \( k \).  
Server does not learn anything about \( x \) or \( y \).

**Verifiable**

Client verifies proof that PRF was computed with \( k \).  
Server commits to the key \( k \), and gives a proof.
What is new?

Blinding mechanism:
- Verifiable mode uses Additive blinding (secure as the key is committed)
- Base mode uses Multiplicative blinding

Explicit errors:
- Deserialization of objects
- Documentation and API considerations

Adopt SHAKE-256 for decaf448 ciphersuite

Generalize DLEQ proof functionality
- Use domain separation tags for its use in other protocols

Update test vectors and editorial changes
Adding public metadata proposal

\[
\text{Blind}(pk, t, x) \\
r \leftarrow r Z_B ; \ B \leftarrow H(x)^r \\
\text{Return } ((r, t, x), B)
\]

\[
\text{BlindEval}(sk, t, B) \\
\ k \leftarrow sk + H(t) \\
\ Z \leftarrow B^{1/k} \\
\ \pi \leftarrow \Sigma R.\text{Prove}(k, (g, g^k, Z, B)) \\
\ \text{Return } (Z, \pi)
\]

\[
\text{UnblindFinalize}(Z, \pi; (r, t, x)) \\
\ N \leftarrow (Z)^{1/r} \\
\ \text{Require } \Sigma R.\text{Ver}((g, (g^{H(t)} \cdot pk), Z, B), \pi) \\
\ Z \leftarrow H(t, x, N) \\
\ \text{Return } Z
\]

Blind evaluation for our 3H PO-PRF construction. All three algorithms have implicit input the parameters \(pp = (p, g, G)\) that describe the group used. The NIZK uses relation \(R = \{(g, U, V, W), (\alpha) : U = g^\alpha \land W = V^\alpha\}\).

From: Partially oblivious PRFs (PO-PRFs) [TCRSTW21] (https://eprint.iacr.org/2021/864)
Protocol:

\[
\begin{align*}
\text{Client} & \quad \text{Server} \\
\text{ctx} \leftarrow \text{SetupClient}(\text{suiteId}) & \quad \text{ctx, sk} \leftarrow \text{ServerSetup}(\text{suiteId}) \\
\text{r, blinded} \leftarrow \text{Blind}(\text{input}) & \\
\text{eval} \leftarrow \text{Evaluate}(\text{sk, blinded, serverMetadata, clientMetadata}) & \\
\text{unblinded} \leftarrow \text{Unblind}(\text{r, eval}) & \\
\text{output} \leftarrow \text{Finalize}(\text{input, Unblinded, serverMetadata, clientMetadata}) & \\ 
\end{align*}
\]
Considerations

- Metadata can be of any length
- Metadata can be added by server and client
- Does not work with additive blinding
- Works well with batching
Questions?

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

Data Tracker: datatracker.ietf.org/doc/draft-irtf-cfrg-voprf

Issues: github.com/cfrg/draft-irtf-cfrg-voprf/issues