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 = PRF(k, x)

Oblivious

Client learns y, but nothing about k

Server does not learn anything about x or y

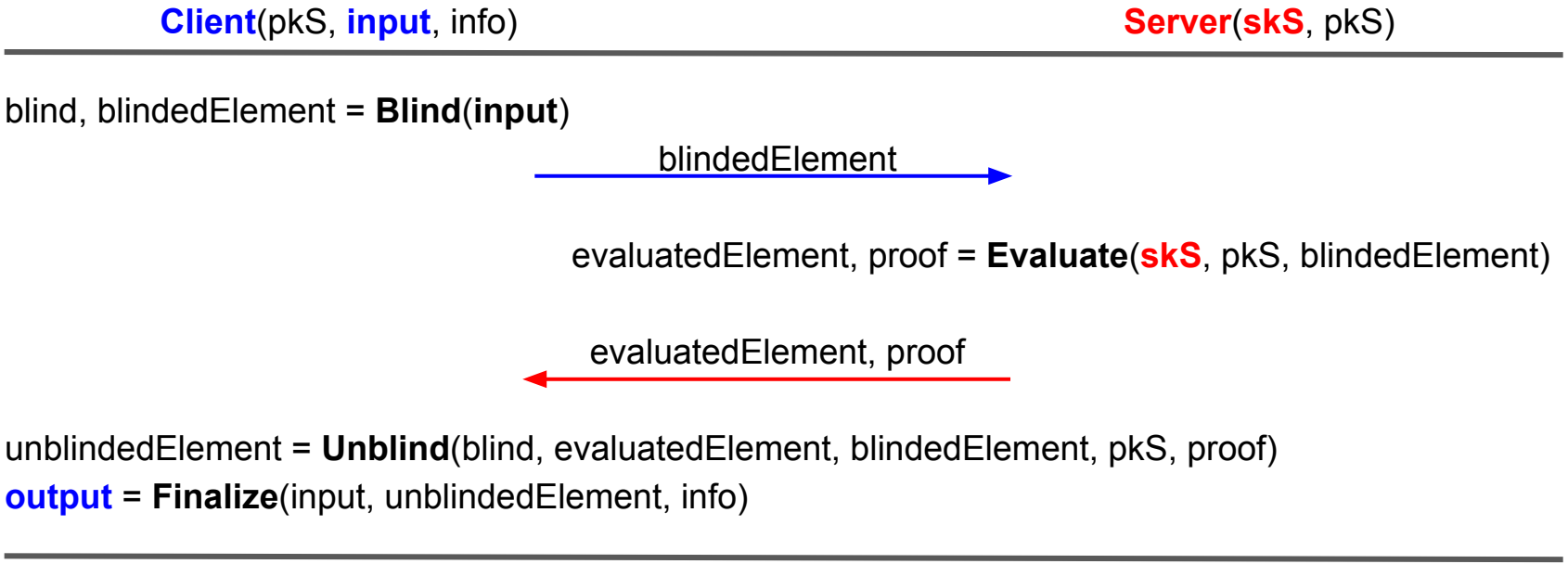
Verifiable

Client can verify that y was computed using k.

Server commits to the key k, and gives a proof.

Protocol

Goal: Client gets output=PRF_{SKS}(input).



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.

Suite Identifier	Group	Hash Function	Security Level
0001	ristretto255	SHA-512	128
0002	decaf448	SHA-512	224
0003	P-256	SHA-256	128
0004	P-384	SHA-512	192
0005	P-521	SHA-512	256

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?