Verifiable Oblivious Pseudorandom Functions

draft-sullivan-cfrg-voprf

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

CFRG
VRFs

Public-key versions of cryptographic hash functions [1]

- Signers compute the “hash,” and verifiers check its correctness

Applications

- Key transparency [https://github.com/google/keytransparency]

- NSEC5 [https://tools.ietf.org/html/draft-vcelak-nsec5-06]

OPRFs

Related to blind signatures rooted in e-cash schemes [1]

- Verifier requests signature over blinded message
- Signer signs blinded message
- Verifier removes blind from signature and message, yielding a message with valid signature

Obliviousness: Signer learns nothing of message to sign

[1] Blind signatures for untraceable payments
Goal: Make OPRFs verifiable

PRF criteria:
- Given $x$, it is infeasible to compute $y = F(k, x)$ without $k$
- $F$ is indistinguishable from a truly random function

Verifiable: Verifier can verify that $Y = kG$ was used to compute $y = F(k, x)$

Oblivious: Signer does not know which signing operation was used in the computation of $y$
VOPRFs

Blinded message $m'$

“Signature” $s$ of $m'$ with proof

Remove blind
Verify proof

Confirm $(m, s)$

Verify $s$ is a “signature” of $m$

Unlinkable

VOPRFs - CFRG - IETF 101
EC-VOPRF

A VOPRF can be constructed with the following building blocks:

- An elliptic curve
- A one-way function to hash a value to a point on a prime order subgroup of the elliptic curve (H2C)
- A non-interactive zero knowledge discrete log equality proof algorithm for points in this subgroup (DLEQ)
NIZK Discrete Log Equality

\[
\log_G(Y) =? \log_M(Z) \\
(Y = kG, Z = kM)
\]

**DLEQGenerate** \((G, Y, M, Z)\)

\[
\begin{align*}
    r &\leftarrow \mathbb{Z}_p \\
    A &= rG \\
    B &= rM \\
    c &= H(G, Y, Z, A, B) \\
    s &= (r - ck)(\text{mod} p)
\end{align*}
\]

Output \((c, s)\)

**DLEQVerify** \((G, Y, M, Z, (c, s))\)

\[
\begin{align*}
    A' &= sG + cY \\
    B' &= sM + cZ \\
    c' &= H(G, Y, Z, A', B') \quad \text{Output } c' = c'
\end{align*}
\]

Protocol

ECVOPRF Protocol

\[ \begin{align*}
  V & \quad P \\
  r & \leftarrow \mathbb{Z}_p \\
  M & = rH2C(x) \\
  \quad M \\
  Z & = kM = krH2C(x) \\
  (c, s) & = \text{DLEQGenerate}(G, Y, M, Z) \\
  (Z, c, s) & \\
  b & = \text{DLEQVerify}(G, Y, M, Z, (c, s)) \\
  y & = r^{-1}Z = kH2C(x) \\
  \text{If } b = 1, \text{Output}(H(y)), \text{ElseOutput}(\bot)
\end{align*} \]

Note: batched protocol is possible with single DLEQVerify
Confirmation by Prover

(x, y) can be sent to P

P computes $kH2C(x)$ and checks $kH2C(x) =? y$

If these match, P can confirm that x was used in a previous VOPRF prover request

Alternatively, (x, m, PRF(y, m)) can be used for a binding message m
Application: Privacy Pass

One more step
Please complete the security check to access captcha.website
Application: Privacy-Preserving Password Leak Checks

';--have i been pwned?

Check if you have an account that has been compromised in a data breach

email address  pwned?
Foundations

[https://eprint.iacr.org/2014/650.pdf]

Pedersen, “Wallet Databases with Observers.”
[https://link.springer.com/content/pdf/10.1007/3-540-48071-4_7.pdf]


Sullivan et al., “Hashing to Elliptic Curves.”
[draft-sullivan-cfrg-hash-to-curve]