Armando Faz, Cloudflare

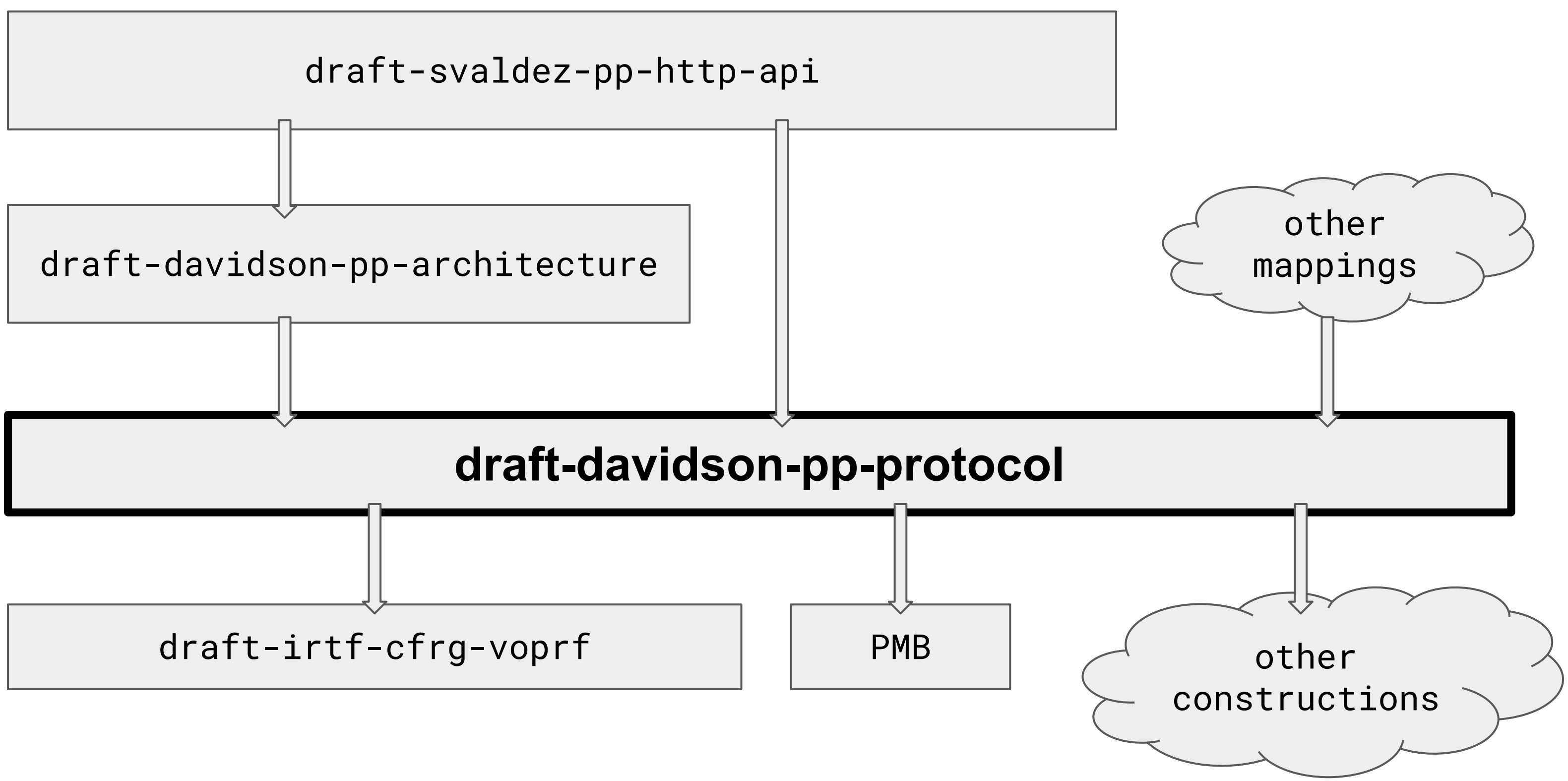
armfazh@cloudflare.com

Privacy Pass: The Protocol

draft-davidson-pp-protocol

https://github.com/alxdavids/privacy-pass-ietf

Privacy Pass Landscape



Content

Definitions

Sec 4.4 RFC 2196

“...
Authorization refers to the process of granting privileges to processes and, ultimately, users.

This differs from **authentication** in that authentication is the process used to identify a user.

Once identified (reliably), the privileges, rights, property, and permissible actions of the user are determined by authorization.
...”

Privacy Pass Requirements

Objective

"Provides a performant, application-layer mechanism for token creation and anonymous redemption."

Security Guarantees

Unlinkability. An issuer cannot link a redeemed token to one of N previously-created tokens using the same key with probability non-negligibly larger than 1/N.

Unforgeability. Tokens are unforgeable. Clients can not redeem more tokens than those were granted.

Key Commitment. Clients can verify that a token created by an issuer corresponds to a committed keypair.

Extensibility. Protocol allows to be extended.

Protocol Phases

Setup

Server generates keys.
Client fetches server's public key.

Issuance

Client interacts with server to obtain tokens.

Redemption

Client redeems a token with the server as authorization method.

Issuance

Goal: Client obtains **m** tokens from the server.

Client(pkS, **m**) | **Server**(skS, pkS)

iReq = **Generate**(**m**)

iReq →

iResp = **Issue**(pkS, skS, iReq)

← iResp

tokens = **Process**(iResp)

Redemption

Goal: Client redeems a token with the server.

Client(token)	**Server**(skS, pkS)
rReq = **Redeem**(token)
rReq →
rResp = **Verify**(skS, pkS, rReq)
← rResp (✓/✗)
Access Granted/Denied

Instantiation Using VOPRF

An *Oblivious Pseudorandom Function* (OPRF) is a protocol for collaboratively computing

Y = PRF(sK, X)

Oblivious

- Client learns Y, without learning the server's key sK.
- Server learns nothing about client's input X.

Verifiable

- Server commits to the key sK.
- Server can prove to the client that Y was computed using sK.

Correctness

- The pair (X,Y) essentially corresponds to a redemption token.
- During redemption, server checks PRF(sK, X) == Y.

Instantiation Using VOPRF

The VOPRF draft provides constructions based on prime-order groups.

The VOPRF API is used for implementing the Privacy Pass functions.

Generate	→	VOPRF::Blind
Issue	→	VOPRF::Evaluate
Process	→	VOPRF::Unblind
Redeem	→	VOPRF::Finalize
Verify	→	VOPRF::VerifyFinalize

Security Analysis

- Satisfies unlinkability, unforgeability, and verifiability.
- See draft-irtf-cfrg-voprf.

Ciphersuites

Privacy Pass Suite	Security Level	VOPRF Suite
PP(OPRF4)	192	OPRF(P-384, SHA-512)
PP(OPRF2)	224	OPRF(curve448, SHA-512)
PP(OPRF5)	256	OPRF(P-521, SHA-512)
(extensible)	...	...

Note: no suites below 192 bits of security due to
Static Diffie-Hellman Problem, see draft-irtf-cfrg-voprf.

Extensions Policy

- New extension must:
  - Add a new ciphersuite.
  - Instantiate the existing API.
- Security properties must be uphold.
- Specified as separate document in WG, or as part of core protocol instantiations.

Potential extensions:

- PMB protocol (eprint.iacr.org/2020/072).
- Publicly verifiable using blind signatures.

Example: Publicly Verifiable Instantiation

A potential way of instantiating the Privacy Pass API using a blind signature scheme.

Generate → BlindSig::Blind
Issue → BlindSig::Sign
Process → BlindSig::Unblind
Redeem → BlindSig::Redeem
Verify → BlindSig::Verify

Security Analysis

- Based on properties of blind signature scheme.
- More details on mailing list thread.

Summary

Privacy Pass Protocol aligned to the goals of WG

- Unlinkable tokens for anonymous redemption.
- API definition for interoperability.
- Efficient instantiation using VOPRF.
- Ciphersuites for crypto agility.
- Extensible.

Armando Faz, Cloudflare

armfazh@cloudflare.com

Privacy Pass: The Protocol

draft-davidson-pp-protocol

https://github.com/alxdavids/privacy-pass-ietf

Motivation

Servers provide an authorization challenge to clients.

Issues:

- Cookies cannot be used cross-domain.
- Client is frequently challenged.
- Manual intervention, e.g. captchas.
- Bad access experience.
- Server can link client browsing patterns across multiple domains.

API

Server Keys:

- **Private Key:** For issuance of tokens.
- **Public Key:** For client verification of issuance.

Data Structures: Data structures provided for structuring protocol messages.

Functions:

Generate	Client prepares a request for tokens.
Issue	Server token generation.
Process	Client token post-processing.
Redeem	Client prepares a request for token redemption.
Verify	Server determines token validity.

Ciphersuites: Determine the implementation of these core functions

Additional Security Properties

Avoid Double-Spending

Prevents clients from replaying previous requests.

Limit #Tokens per Issuance

Prevents malicious clients to abuse the service, e.g. DoS attacks.

Existing Applications

- Bypassing CAPTCHA challenges [PPSRV]
- Trust Token API [TTA]
- Zero-Knowledge Access Passes [PS]
- Basic Attention Tokens [BAT]
- Token-based Services [OP]