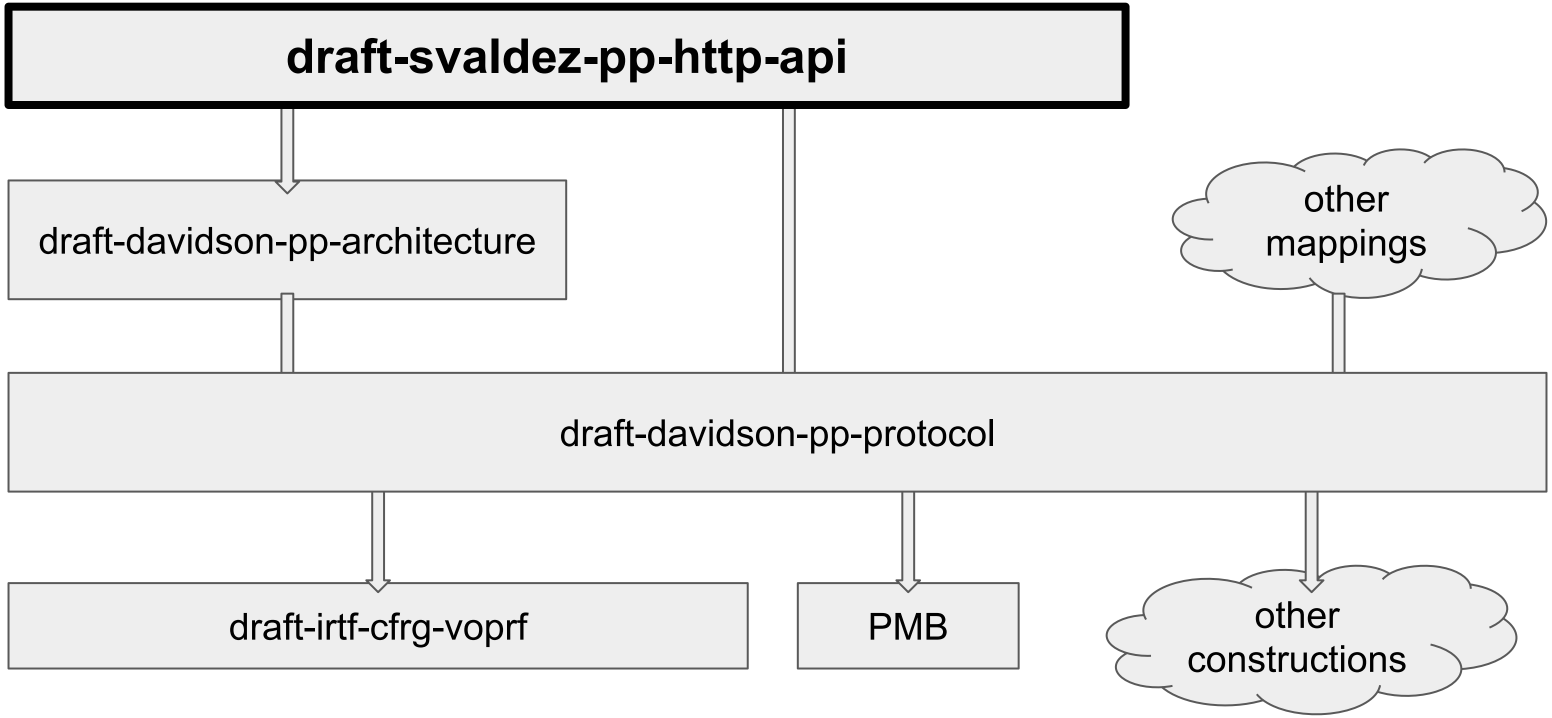
# Privacy Pass: HTTP API

IETF 108 – Virtual – 2020-07

Steven Valdez - svaldez@google.com

## Privacy Pass Landscape


draft-svaldez-pp-http-api
draft-davidson-pp-architecture
other mappings
draft-davidson-pp-protocol
draft-irtf-cfrg-voprf
PMB
other constructions

## pp-http-api

- https://tools.ietf.org/html/draft-svaldez-pp-http-api-01
- HTTP Wrapping of pp-protocol
- Key Management
- Issuance Protocol
- Redemption Protocol

## HTTP Wrapping

- Sec-Privacy-Pass Header
  - Minimal HTTP Parsing by Issuer
  - Server can dispatch header contents to Privacy Pass server
- Structured Header (https://tools.ietf.org/html/draft-ietf-httpbis-header-structure)
  - Message Type
  - Message Data
- Allows for Issuance/Redemption to potentially move off .well-known endpoints.

## Issuance Protocol

- Request to well known endpoint with nonces in the Sec-Privacy-Pass header.
- Response with the signed blinded tokens.
- Client parses the Sec-Privacy-Pass header and unblinds the tokens.
- Client stores the key used to sign the tokens, along with each token, partitioned by the issuer origin.

## Redemption Protocol (Generic)

- The client requests the current issuer configuration/commitments.
- If the configuration/commitments are incompatible with the token being redeemed, the redemption fails.
- Otherwise, a request is made to well known endpoint with token and 'info' in the Sec-Privacy-Pass header.
    - Info is generic data that the client wants attached to this redemption.
- Server verifies the token and provides a signature over the 'info' field and the result of the redemption back to the client in the Sec-Privacy-Pass header.
- The client parses the Sec-Privacy-Pass header and verifies the signature over the redemption response.

## Redemption Protocol (Direct)

- Using the Single Verifier/Delegated Verifier server running modes.
- The client sends the PrivacyPass token to the **target** that wants a token via the Sec-Privacy-Pass header.
- The target site will perform a "Generic Redemption" to the specified issuer.
- If the "Generic Redemption" succeeds with a valid redemption response, the **target** can proceed as the client provided a valid token, otherwise it should proceed as if the client didn't provide a token.

## Redemption Protocol (Delegated)

- Using the Asynchronous Verifier server mode
- The client constructs an "info" value containing the current timestamp, the context it is redeeming this token for, and any additional data.
- It then performs a "Generic Redemption" directly to the PrivacyPass issuer.
- The client then sends the resulting Redemption Response to the **target** site.
- The target site verifies the validity of the Redemption Response and checks whether the "info" value corresponds to this action (has a recent timestamp and valid context).
- If valid, the **target** can proceed as the client provided a valid token, otherwise it should proceed as if the client didn't provide a token.
- This Delegated Redemption allows one token redemption to be used multiple times in the same context (a site loading many PrivacyPass-requiring resources) or in cases whether the **target** doesn't want to perform a token redemption for every query and would rather have the client do so (due to QPS or bandwidth concerns).

## Key Management

## HTTP Key Management Requirements

- Anonymity sets dependent on sets of keys used
- Consistent key commitments across different web clients
- Allow for Key Rotation
  - Compromised/Lost Keys
  - Standard rotation windows
- Auditability
- Append-Only Consistent Log

## Optimization: Issuer Configuration

- Issuers send their key commitment to a key registry and receive an inclusion proof.
- Clients fetch latest Key Commitments directly from Issuer
  - Fetched at issuance and redemption time to ensure consistency between issuance and redemption.
  - Clients verify inclusion proof.
- Auditors verify that the key registry is receiving new keys from issuers within approved timespans (not rotating keys every minute).
- Reduces QPS on Key Registry.

## Key Management Alternatives

- Clients performs a HTTP request to the Key Registry.
  - Receives the requested server configuration/key commitments.
- Client Proxy fetches key commitments.
  - All clients to the proxy receive the same key commitments.
  - Proxy should be oblivious to any client state to prevent collusion.
  - For HTTP clients, UAs could provide the proxy/configurable proxy/etc

## Open Questions

- Protocol Endpoints
    - Using .well-known or allowing issuance/redemption to be performed on any server endpoint (as part of other requests to the issuance server)
- Recommendations for Double-Spend protection
    - Eventually consistent
- Auditors/Key Rotation Policies
- Key management strategy
    - Specify particular strategy for this API or add support for any in the architecture doc?
- Maintaining Consistency with Higher-Level APIs
    - W3C Specs