Alex Davidson, Cloudflare

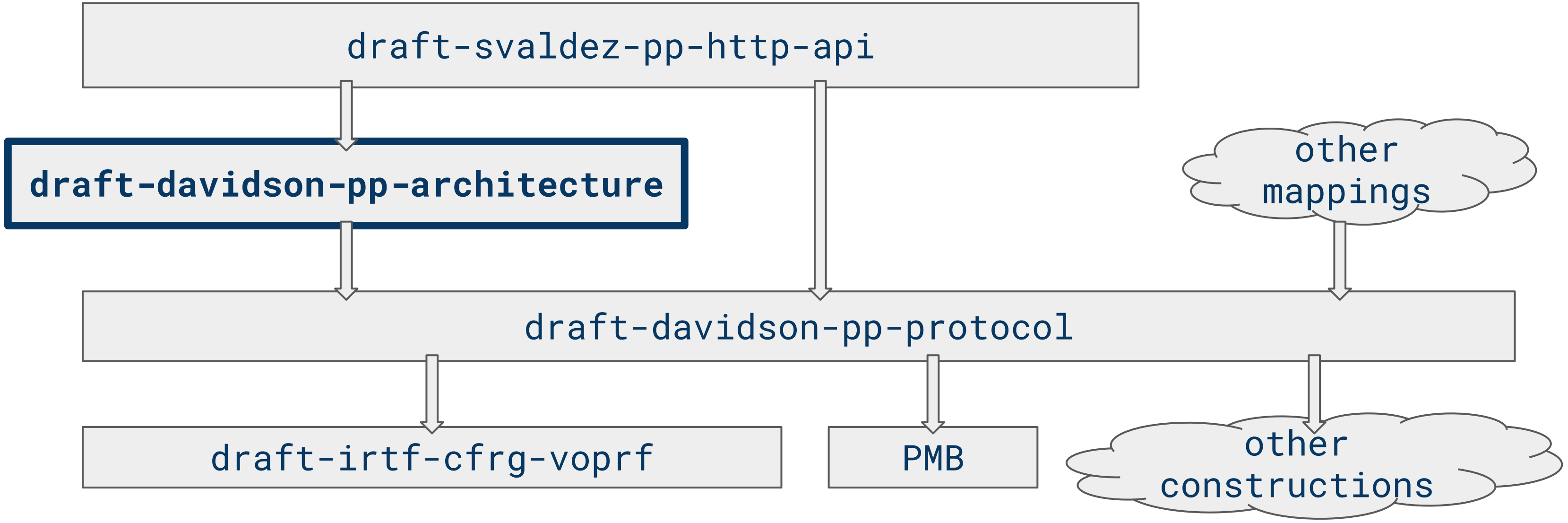
alex.davidson92@gmail.com

Architecture

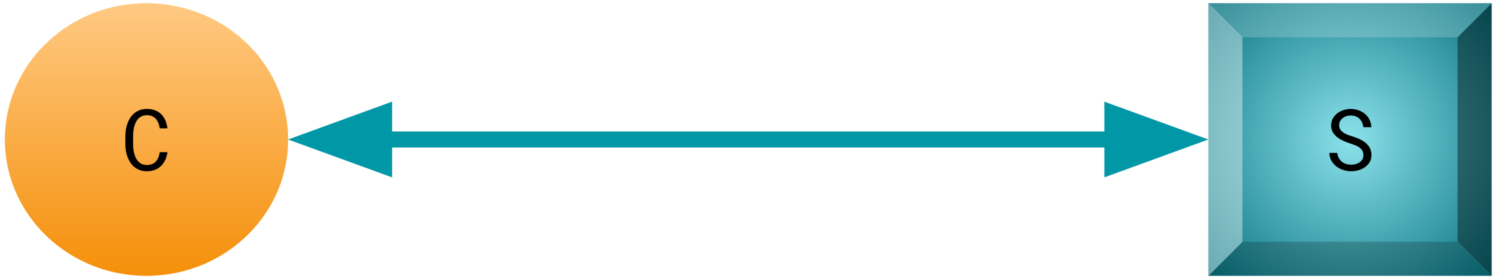
draft-davidson-pp-architecture

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

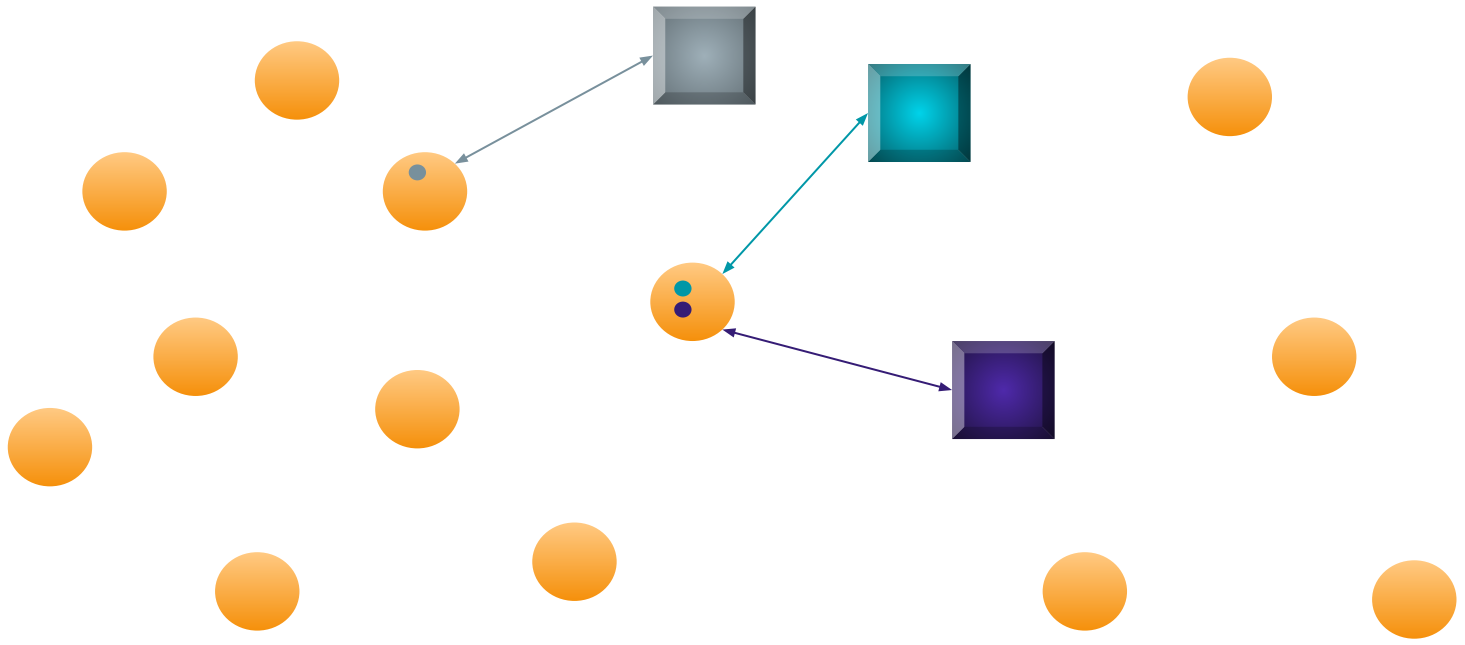
Landscape


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

Protocol



Ecosystem



Document organisation

Important questions

1. Who are the valid token issuers/servers?
2. How are server keys published/audited?
3. How do clients choose which servers to trust?
4. How do clients and servers interact?
5. How do we quantify the privacy of a client?

Allowed servers

1. Who are the valid token issuers/servers?

Ecosystem defined by which servers are supported.

Controlled by where key material is made available to clients.

Open question: Mitigations against server centralisation?

Key management

2. How are server keys published/audited?

Server key information is stored in independent, public, append-only registries.

Each registry decides which servers to support.

Clients retrieve key information from registries.

Open question: Should we specify such a registry? If so, here or elsewhere?

Key registries

Data:

- Server identifier (e.g. FQDN)
- Ciphersuite
- Public key

Only one valid key permitted at any time, consistent across registries.

Rotation: append new key and invalidate old data.

Client/Server trust

3. How do clients choose which servers to trust?

Clients should only store and redeem tokens with servers that they trust.

Important factors:

- **Does the client trust the key registry?**
- **Reason for initiating issuance/redemption?**

This is a policy question that we do not cover.

Client/Server trust

Implementing client-trust mechanisms:

- Allowlists for key registries
- Allowlists for individual servers?

<u>Open question</u>: How do we assess whether a server is malicious?

Server running modes

4. How do clients and servers interact?

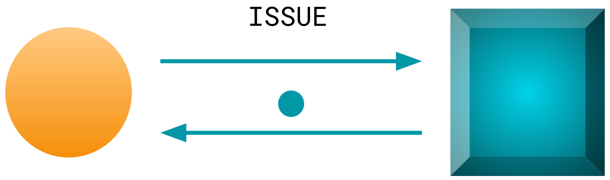
Four running modes.

We define preferred mechanisms for client-server interactions.

Client API is equivalent in most running modes. Tokens are independent of mode.

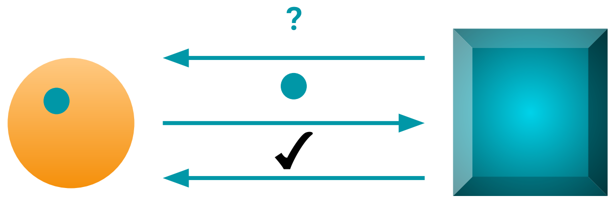
Issuance

We do not explicitly cover issuance running modes in the doc.



Issuance is always a secret key operation, so clients have to receive tokens from a server-authenticated operation.

Redemption: Single-verifier

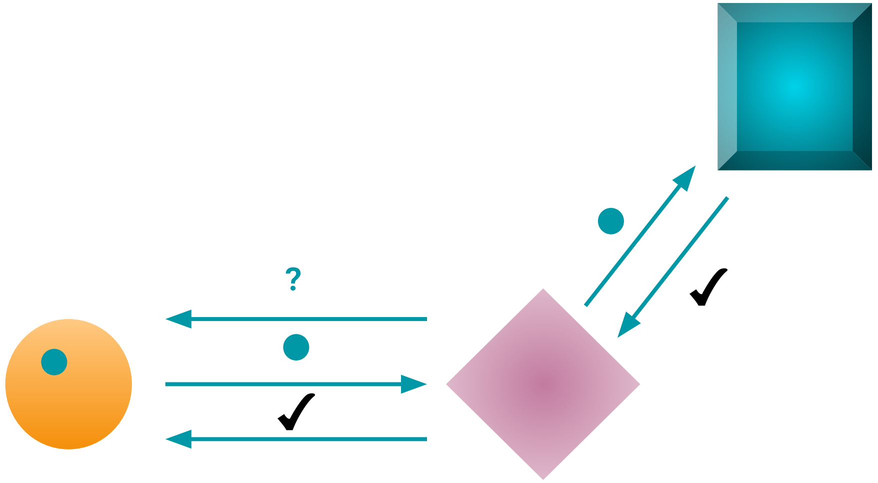


Clients redeem tokens directly with the server that they were issued from, i.e. same FQDN.

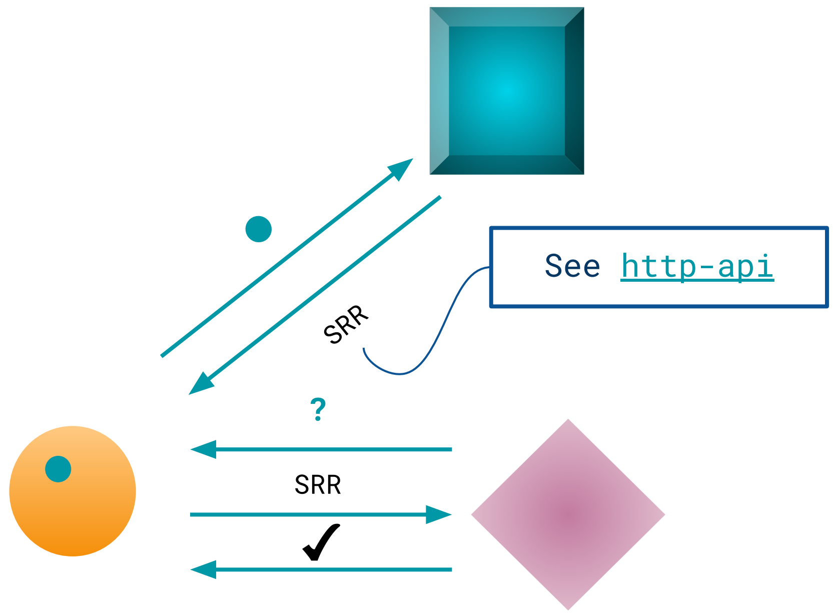
Redemption: Delegated-verifier

Intermediate verifiers proxy valid tokens from clients to appropriate server.

Verifiers can use valid redemption signal.



Redemption: Asynchronous-verifier



Client redemption triggered by verifier. Client retrieves signed redemption record (SRR) directly from issuing server (or cache).

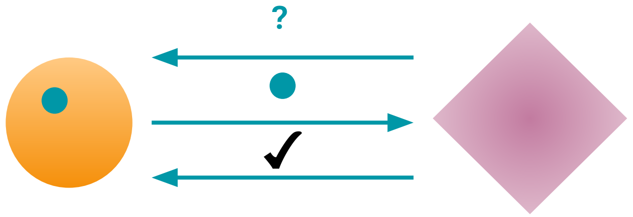
Client reveals SRR to verifier.

Redemption: Public-verifier

Client redemption tokens are publicly verifiable using server's public key.

Currently not supported by protocol functionality.

Potential extension.



Privacy analysis

5. How do we quantify the privacy of a client?

Important factors to consider:

- **# of servers**
- **# of clients accepting tokens for a server**
- **Additional metadata bits inserted in tokens**
- **Frequency of server key rotation**
- **Potential collusion (servers + key registries)**

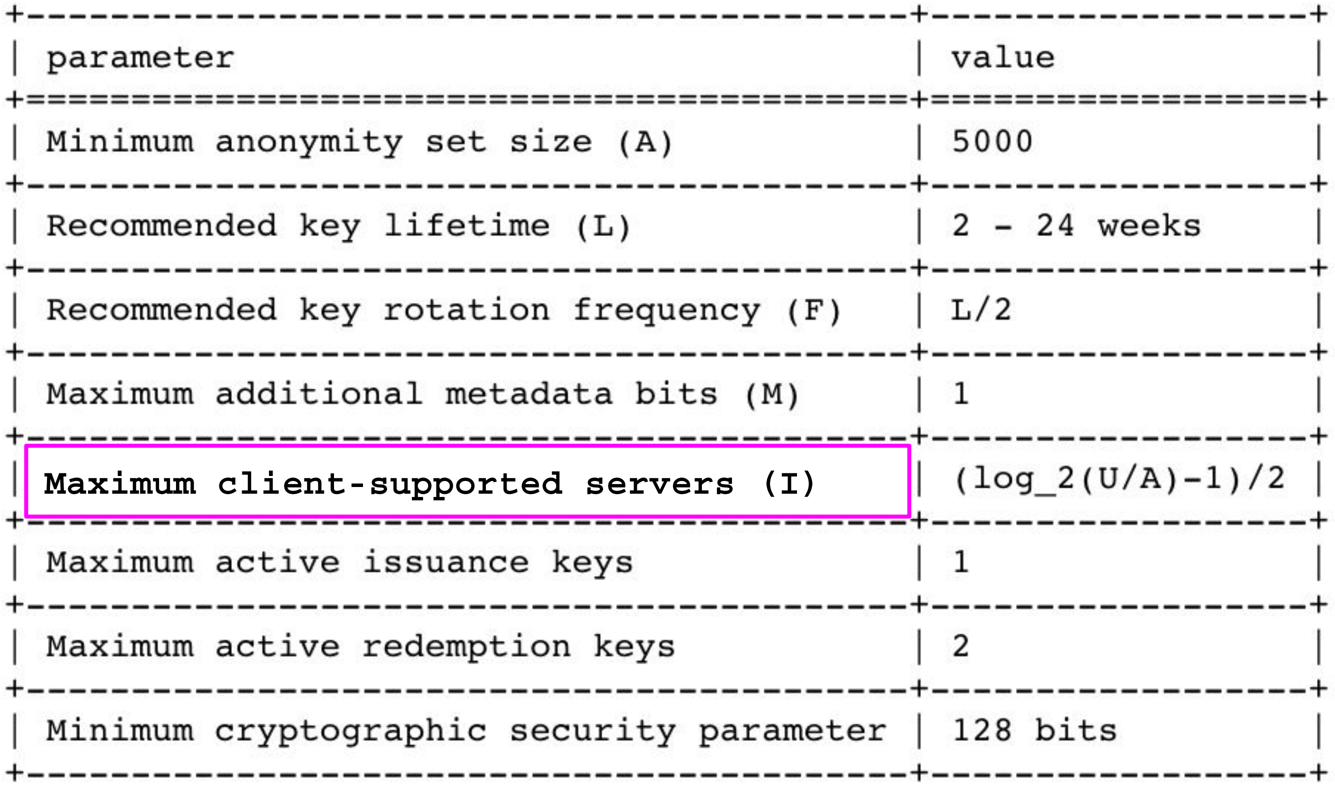
Privacy parameterisation

parameter	value	
Minimum anonymity set size (A)	5000	?
Recommended key lifetime (L)	2 - 24 weeks	?
Recommended key rotation frequency (F)	L/2	
Maximum additional metadata bits (M)	1	
Maximum allowed servers (I)	(log_2(U/A)-1)/2	?
Maximum active issuance keys	1	
Maximum active redemption keys	2	
Minimum cryptographic security parameter	128 bits	

U is total # of users

Privacy parameterisation

Possible way for removing hard limit on # of allowed servers



parameter	value
Minimum anonymity set size (A)	5000
Recommended key lifetime (L)	2 - 24 weeks
Recommended key rotation frequency (F)	L/2
Maximum additional metadata bits (M)	1
Maximum client-supported servers (I)	(log_2(U/A)-1)/2
Maximum active issuance keys	1
Maximum active redemption keys	2
Minimum cryptographic security parameter	128 bits

Extension policy

Any protocol extension must:

- Provide new ciphersuite identifiers
- Update security analysis for protocol
- Update privacy analysis
  - Key management
  - Additional metadata

Summary

Architecture doc analyses Privacy Pass ecosystem.

Advice on server implementation and resulting privacy implications for clients.

Concrete privacy parameterisation for informing policies.

Open questions

Suggestions for mitigating against server centralisation? (Separate doc?)

Should we concretely specify key registry? In this doc, or somewhere else?

Suggestions for how malicious servers & key registries should be detected, and how to react?

Alex Davidson, Cloudflare

alex.davidson92@gmail.com

Architecture

draft-davidson-pp-architecture

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