Rate-Limited Issuance

draft-ietf-privacypass-rate-limit-tokens

Tommy Pauly, Chris Wood, Steven Valdez, Scott Hendrickson, Jana Iyengar

Agenda

Recap of differences from Basic tokens

Open Issues

Rate-Limited Tokens

Rate-limited tokens *extend* the basic issuance protocol with new properties:

1. Attester maintains counters for client + anonymized origin
   - Attesters learn ***stable mapping*** between per-client secret and per-origin secret, without learning only per-origin information
2. Issuer provides a rate limit to enforce when issuing tokens
   - Issuers learn origin associated with a token challenge, encrypted with HPKE
3. Attesters fail requests if the per-origin rate limit is exceeded

Changes from basic type

Request
Response
Encrypted Request
Response
Challenge (Token Key + Encapsulation Key)
Response (Token)
Issuer
Attester
Client
Server

Challenge flow adds a per-issuer HPKE key to use to encrypt token requests

Changes from basic type

Request
Response
Encrypted Request
Response
Challenge (Token Key + Encapsulation Key)
Response (Token)
Issuer
Attester
Client
Server

Issuance protocol adds
1. An HPKE-encrypted inner request that contains Origin name
2. An anonymous origin name the Attester uses to keep counts
3. A token limit per time that the issuer communicates to the Attester
4. A blinded key signature to prove that the Client doesn't cheat

Basic Token Request

```
struct {
   uint16_t token_type = 0x0002
   uint8_t token_key_id;
   uint8_t blinded_msg[Nk];
} TokenRequest;
```

Rate-Limited Token Request

```
struct {
   uint16_t token_type = 0x0003;
   uint8_t request_key[Npk];
   uint8_t issuer_encap_key_id[32];
   uint8_t encrypted_token_request<1..2^16-1>;
   uint8_t request_signature[Nsig];
} TokenRequest;
```

Client-to-Attester

HPKE-protected request to Issuer

Signature using blinded key pair to validate origin uniqueness

```
struct {
  uint8_t token_key_id;
  uint8_t blinded_msg[Nk];
  uint8_t padded_origin_name<0..2^16-1>;
} InnerTokenRequest;
```

Client-to-Issuer
HPKE Encrypted

Adds origin name to basic request struct

Linkability with Malicious Issuer

Issue #6

Malicious Issuer reuses Issuer Origin Secret across different origins

- This causes the Attester to trip the check to prevent a Client from pretending that one origin is actually two

Current protocol has the Attester fail the token request immediately in this case

- This failure can be used as a signal on the redemption path to identify the client.

Proposed Fix

Issue #6

Attesters should not silently drop requests due to mapping collision

- Silent drops allow malicious Issuers to leak a signal
- Silent drops also don't penalize malicious Clients on future requests

Instead, Attesters need to flag these collisions to re-evaluate trust in Issuers or Clients

- A malicious Issuer will cause collisions across Clients; if so, the Attester should start blocking this Issuer
- A malicious Client can similarly be blocked for future token requests

Sidebar

Document should go deeper on how the Attester trusts an Issuer, in addition to causing collisions

- Acceptable lengths of Issuer Policy Window (the time for the rate limit)
- Acceptable rate limit values
- Number of origins served to create an anonymity set

Alternative Fix: ZKP Anonymous Origin

A Zero-Knowledge proof was suggested as an alternate approach

- Rather than relying on the Attester confirming the values from the Client and Issuer, the Client encrypts the origin to the issuer and provides a ZKP the Attester can verify.

Would rely on new crypto that isn't specified yet.

More analysis of protocol impact would be needed.

Recommended as a candidate for future token types

- Could be bundled with other features that would require Attester state

Other Open Issues

#1 - One Bit Token Key ID

Simplifies consistency checks, limits key rotation schemes.

#2 - Hide Issuer Rate Limit from Attester

Potential future extension/token type?

#5 - Rate limit against non-challenger origin

Expand discussion around origin_info with multiple origins.

#8 - Clarify Attester anomaly detection/reaction

Related to sidebar on issue #6, add expanded discussion in spec.

Next Steps

Update document as discussed

Track CFRG dependency (draft-irtf-cfrg-signature-key-blinding)

Questions?