Rate-Limited Privacy Pass

draft-privacypass-rate-limit-tokens

Hendrickson, Iyengar, Pauly, Valdez, Wood — IETF 114 — Privacy Pass
Privacy Pass Protocol Recap

Redemption is a consistent/unified API for redeeming tokens, along with the ability to challenge.

Issuance can support multiple types (VOPRF, publicly verifiable, etc). This is the exchange that can be extended or replaced for new deployment models.
Basic Issuance and Redemption

### Origin

$pk_i$

challenge

token

### Client

nonce $\leftarrow \{0,1\}^{256}$

correct = Hash(challenge)

req, inv = Blind($pk_i$, (nonce, context))

authenticator = Finalize($pk_i$, (nonce, context), resp, inv)

### Attester

req

resp

### Issuer

$(sk_i, pk_i)$

resp = BlindSign($sk_i$, req)
Basic Issuance and Redemption

Redemption Protocol

Origin

\[ pk_s \]

\[ \text{challenge} \]

\[ \text{token} \]

Client

\[ \text{nonce} \leftarrow \{0,1\}^{256} \]
\[ \text{context} = \text{Hash}(	ext{challenge}) \]
\[ \text{req, inv} = \text{Blind}(pk_s, (\text{nonce, context})) \]

\[ \text{authenticator} = \]
\[ \text{Finalize}(pk_s, (\text{nonce, context}, \text{resp, inv}) \]

Attester

\[ \text{req} \]

\[ \text{resp} \]

Issuer

\[(sk_s, pk_s)\]

\[ \text{resp} = \text{BlindSign}(sk_s, \text{req}) \]

Origin does not learn anything about Client beyond “can it present a token?”
Basic Issuance and Redemption

Origin

\[ p_k, \]

\[ \text{challenge} \]

\[ \text{token} \]

Client

\[ \text{nonce} \leftarrow \{0,1\}^{256} \]

\[ \text{context} = \text{Hash}((\text{challenge})) \]

\[ \text{req, inv} = \text{Blind}(p_k, (\text{nonce, context})) \]

\[ \text{authenticator} = \text{Finalize}(p_k, (\text{nonce, context}), \text{resp, inv}) \]

Attester

\[ \text{req} \]

\[ \text{resp} \]

Issuer

\[ (s_k, p_k) \]

\[ \text{resp} = \text{BlindSign}(s_k, \text{req}) \]

Issuer does not learn anything about client

Issuance Protocol
Rate Limiting Limitations

Origins don't have a sticky notion (stable mapping) of an *individual* client -- and they shouldn’t! -- leading to shared rate limit contexts (like IP addresses)

Rate limiting is very useful

- Metered paywalls
- Dampens damage from bot farms

How can we add rate limit support to Privacy Pass without compromising on privacy?
Rate Limiting Recap

- Token Bucket
  - (are there tokens available?)
  - Consume tokens
    - Resource Request
      - Yes, service request!
      - Replenish tokens
      - Token Replenish
        - Replenish tokens
      - No, don't service request
Rate Limiting Recap

Token Bucket

1. Identity rate limit context
2. Increment count by number of tokens

<table>
<thead>
<tr>
<th>Mapping</th>
<th>Token Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>1250123010339</td>
<td>N (N + T)</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Yes, service request!

No, don’t service request
Rate Limiting Recap

1. Identity rate limit context
2. Decrement count associated with bucket
3. Process if non-zero, otherwise discard

Mapping | Token Count
---------|-------------
1250123010339 | N (N - 1) > 0?

No, don’t service request
Yes, service request!
Rate Limiting Design Constraints

Origin should only learn whether or not a client exceeded the rate limit

*Otherwise the origin could use this state to track clients*

Origin adoption should be as simple as basic Privacy Pass

*Otherwise little incentive to adopt*

Issuance protocol should be as close to stateless as possible

*Otherwise operating an issuer or attester is complicated (see CT)*

Neither issuer nor attester should be able to identify (client, origin) pairs

*Otherwise issuer or attester could track client browsing history*
Rate-Limited Tokens

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

1. Issuers learn origin associated with a token challenge

   Necessary to determine what rate limit policy to use for the token

2. Attesters learn stable mapping between per-client secret and per-origin secret, without learning only per-origin information

   Necessary to enforce rate limits on a per-client, per-origin basis without learning (client, origin) pairs

3. Token requests may fail if the per-origin rate limit is exceeded
Stable Mappings and Rate Limits

Client

$p_{k_C}, p_{k_I}$

challenge

origin

$s_{k_C}$

token

req

resp

Attester

Compute stable mapping, decrement count, compare against origin limit, accept or reject response accordingly

Issuer

req

resp, $L_{\text{origin}}$

Drop request
Stable Mappings and Rate Limits

Client

$p_kC, pk_I$

challenge

origin

$s_kC$

token

req

resp

Attester

Compute stable mapping, decrement count, compare against origin limit, accept or reject response accordingly

1234 = F(client, origin)

Mapping | Count
---|---
... | ... 
1234 | N → N-1
... | ...

$N - 1 < L_{\text{origin}}$

resp, $L_{\text{origin}}$

Issuer

Drop request

N

13
State Management

Origin keeps issuer’s public key (same as basic Privacy Pass), size $O(1)$

Client maintains state for all origins that request rate-limited tokens, size $O(|S|)$

Attester maintains per-client, per-origin state for all clients, size $O(|C| \times |S|)$

Issuer maintains keys for each origin it supports, size $O(|S|)$

$C$ and $S$ are client and origin sets
Privacy Properties

Origin and Issuer cannot link and two requests to the same client (same as basic Privacy Pass)

Attester learns stable mapping between per-client and per-origin secret, but does not learn the origin identity or when two clients access the same origin

*The attester is already trusted with the client’s IP address, so it is also trusted to handle this sensitive information*
Open Issues

Can (client, origin) access pattern leakage be exploited with additional auxiliary information to reconstruct the origin?

Prevent clients from swapping origin names across redemption and issuance contexts. Binding can be enforced with per-origin keys, or partially-blind signatures in the future.

https://github.com/tfpauly/privacy-proxy/issues/218

Per-origin keys requires an updated approach to key consistency checks
Status

Two interoperable implementations exist

Security analysis complete, though did not capture origin swapping issue

Work underway to update the model using distinct per-origin keys
Questions?  
See ongoing adoption call!
Rate-Limited Privacy Pass

draft-privacypass-rate-limit-tokens