

Privacy Pass Authorization for MOQ

draft-ietf-moq-privacy-pass-auth-03

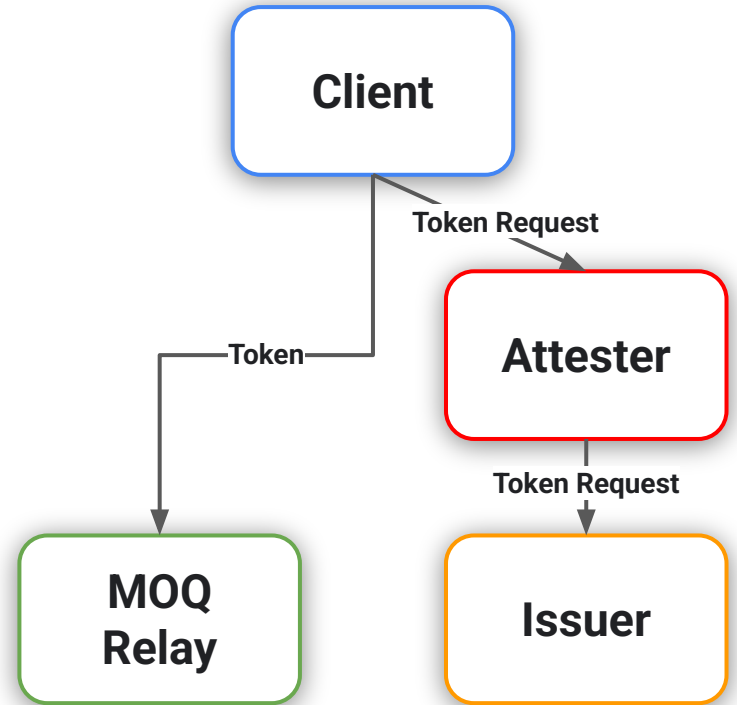
IETF 125

Authors: Suhas Nandakumar, Cullen Jennings, Thibault Meunier

What is Privacy Pass ?

Core Concept

- **Unlinkable Tokens** - Cannot link issuance and redemption
- **Blind Signatures** - Issuer signs without seeing token content
- **Single-use Tokens** - Prevents replay while preserving privacy
- **Separation of Concerns** - Attester, Issuer, and Origin can be different entities



How Privacy Pass Helps MOQ

Benefits for Media Streaming

- **Anonymous Subscriptions** - Users can subscribe to content without revealing viewing patterns
- **Fine-grained Access Control** - Tokens encode scopes for namespaces and track names
- **Relay Trust Separation** - Relays validate tokens without seeing user identity
- **Continuous Authorization** - Batched tokens enable long-lived sessions without re-attestation

Privacy

Viewing patterns remain private:
a relay cannot link subscriptions
to user identity

Scalability

Local token validation eliminates
issuer bottleneck for
authorization checks

Flexibility

Supports existing token types:
Blind RSA, VOPRF, ARC

Reverse Flow: Three-Phase Authorization

Phase 1: Bootstrap Token

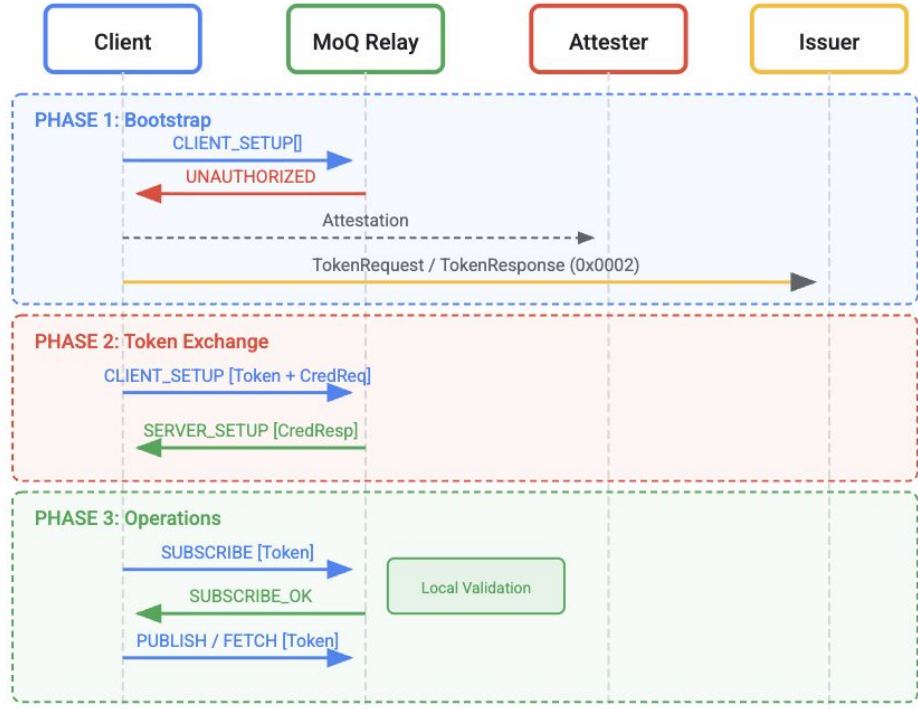
Client obtains publicly verifiable token (0x0002) from external issuer via attestation

Phase 2: Token Exchange

Client presents bootstrap token + credential request; relay issues privately verifiable tokens/credentials

Phase 3: Normal Operations

Client uses derived tokens for SUBSCRIBE, PUBLISH, FETCH operations



Reverse Flow Benefits

Why use Reverse Flow?

- **Bootstrapping** - Establish trust with publicly verifiable token – relay is distinct from IdP
- **Rate Limiting** - ARC tokens (0xE5AC) provide presentation_limit for rate control
- **Privacy** - Each token presentation is unlinkable, even from same credential
- **Efficiency** - Batched issuance amortizes cryptographic costs across multiple operations
- **Continuous Auth** - Request new tokens inline with operations for long sessions

```
// Continuous Authorization Timeline with Reverse Flow  
  
Time 0:  CLIENT_SETUP [Token_1, request 1 token] -> SERVER_SETUP [1 token]  
  
Time T:  SUBSCRIBE [Token_2, request 1 token]    -> Relay responds [1 token]  
  
Time 2T: PUBLISH [Token_3, request 1 token]      -> Relay responds [1 token]  
  
// Unlinkable chain - relay cannot correlate operations
```

Changes in -03: Error Handling

New Error Code Registry

Error Code	Name	Description
<code>0x0100</code>	TOKEN_MISSING	No token provided when required
<code>0x0101</code>	TOKEN_INVALID	Token signature verification failed
<code>0x0102</code>	TOKEN_EXPIRED	Token has expired or been revoked
<code>0x0103</code>	TOKEN_REPLAYED	Token nonce has been seen before
<code>0x0104</code>	SCOPE_MISMATCH	Token scope does not authorize this operation
<code>0x0105</code>	ISSUER_UNKNOWN	Token issuer is not trusted by this relay
<code>0x0106</code>	TOKEN_MALFORMED	Token cannot be parsed correctly

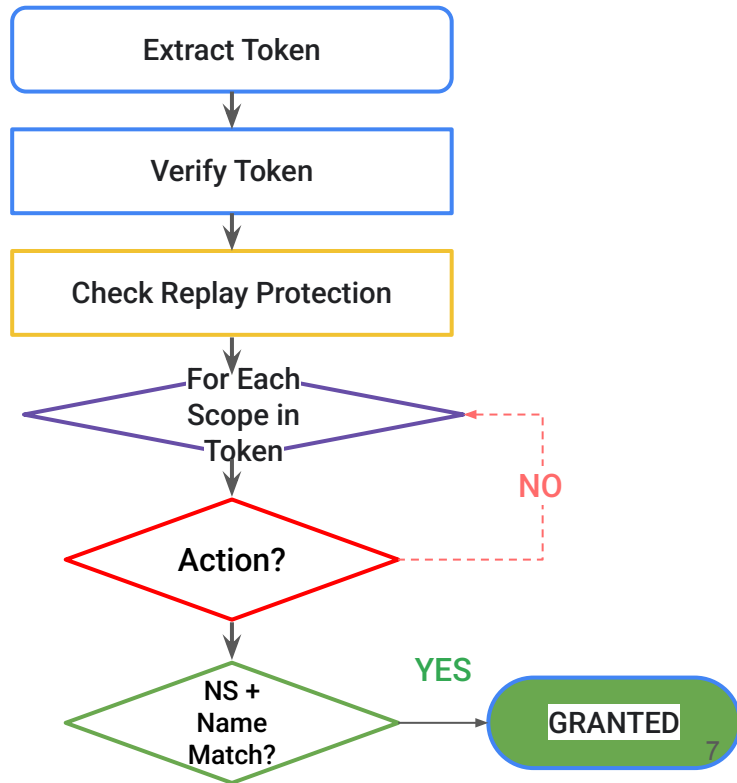
Changes in -03: MoQ Actions & Matching Algorithm

MOQ Actions Registry

Value	Action
0	CLIENT_SETUP
1	SERVER_SETUP
2	PUBLISH_NAMESPACE
3	SUBSCRIBE_NAMESPACE
4	SUBSCRIBE
5	REQUEST_UPDATE
6, 7, 8	PUBLISH, FETCH, TRACK_STATUS

Default policy: **blocked** - all actions denied unless explicitly permitted

Matching Algorithm Flow



Changes in -03: Match Types & Authorization Scopes

Match Types Registry

Value	Type	Description
0	MATCH_EXACT	Value equals pattern exactly
1	MATCH_PREFIX	Value starts with pattern
2	MATCH_SUFFIX	Value ends with pattern
3	MATCH_CONTAINS	Value contains pattern

```
struct {
    MoQAction actions<1..2^8-1>;
    NamespaceMatchRule namespace_match;
    TrackNameMatchRule track_name_match;
} MoQAuthScope;

// Example: Subscribe to live sports
MoQAuthScope {
    actions = [SUBSCRIBE(4)],
    namespace_match = {
        match_type = MATCH_PREFIX(1),
        value = ["sports.example.com", "live"]
    },
    track_name_match = {
        match_type = MATCH_PREFIX(1),
        value = "" // matches all
    }
}
```

Next Steps

New IANA Registries

- **MoQ Privacy Pass Auth Schemes** - PrivateTokenAuth (0x01)
- **MoQ Actions** - CLIENT_SETUP through TRACK_STATUS (0-8)
- **MoQ Match Types** - EXACT, PREFIX, SUFFIX, CONTAINS
- **MoQ Privacy Pass Error Codes** - 0x0100-0x01FF range