

Media Over QUIC
Internet-Draft
Intended status: Standards Track
Expires: 18 April 2026

S. Nandakumar
C. Jennings
Cisco
15 October 2025

Privacy Pass Authentication for Media over QUIC (MoQ)
draft-ietf-moq-privacy-pass-auth-00

Abstract

This document specifies the use of Privacy Pass architecture and issuance protocols for authorization in Media over QUIC (MoQ) transport protocol. It defines how Privacy Pass tokens can be integrated with MoQ's authorization framework to provide privacy-preserving authentication for subscriptions, fetches, publications, and relay operations while supporting fine-grained access control through prefix-based track namespace and track name matching rules.

Issues and PRs can be made against <https://github.com/suhasHere/moq-privacy-pass>.

About This Document

This note is to be removed before publishing as an RFC.

Status information for this document may be found at <https://datatracker.ietf.org/doc/draft-ietf-moq-privacy-pass-auth/>.

Discussion of this document takes place on the Media Over QUIC Working Group mailing list (<mailto:moq@ietf.org>), which is archived at <https://mailarchive.ietf.org/arch/browse/moq/>. Subscribe at <https://www.ietf.org/mailman/listinfo/moq/>. Working Group information can be found at <https://datatracker.ietf.org/wg/moq/>.

Source for this draft and an issue tracker can be found at <https://github.com/moq-wg/moq-transport>.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 18 April 2026.

Copyright Notice

Copyright (c) 2025 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

1. Introduction	3
1.1. Requirements Language	3
2. Privacy Pass Architecture for MoQ	3
2.1. Trust Model	5
3. Privacy Pass Token Integration	5
3.1. Token Types for MoQ Authorization	5
3.2. Token Structure	5
3.2.1. Token Challenge Structure for MoQ	6
3.3. Track Namespace and Track Name Matching Rules	7
3.3.1. Namespace Matching	7
3.3.2. Track Name Matching	7
3.3.3. Matching Algorithm	8
3.4. Token in MOQ Messages	9
3.4.1. SETUP Message Authorization	9
3.4.2. MoQ Operation-Level Authorization	10
4. Example Authorization Flow	10
5. Security Considerations	11
6. IANA Considerations	11
7. References	11
7.1. Normative References	11
7.2. Informative References	12
Authors' Addresses	12

1. Introduction

Media over QUIC (MoQ) [MoQ-TRANSPORT] provides a transport protocol for live and on-demand media delivery, real-time communication, and interactive content distribution over QUIC connections. The protocol supports a wide range of applications including video streaming, video conferencing, gaming, interactive broadcasts, and other latency-sensitive use cases. MoQ includes mechanisms for authorization through tokens that can be used to control access to media streams, interactive sessions, and relay operations.

Traditional authorization mechanisms often lack the privacy protection needed for modern media distribution scenarios, where users' viewing patterns and content preferences should remain private while still enabling fine-grained access control, namespace restrictions, and operational constraints.

Privacy Pass [RFC9576] provides a privacy-preserving authorization architecture that enables anonymous authentication through unlinkable tokens. The Privacy Pass architecture consists of four entities: Client, Origin, Issuer, and Attester, which work together to provide token-based authorization without compromising user privacy. The issuance protocols [RFC9578] define how these tokens are created and verified.

This document defines how Privacy Pass tokens can be integrated with MoQ's authorization framework to provide comprehensive access control for media streaming, real-time communication, and interactive content services while preserving user privacy through unlinkable authentication tokens.

1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2. Privacy Pass Architecture for MoQ

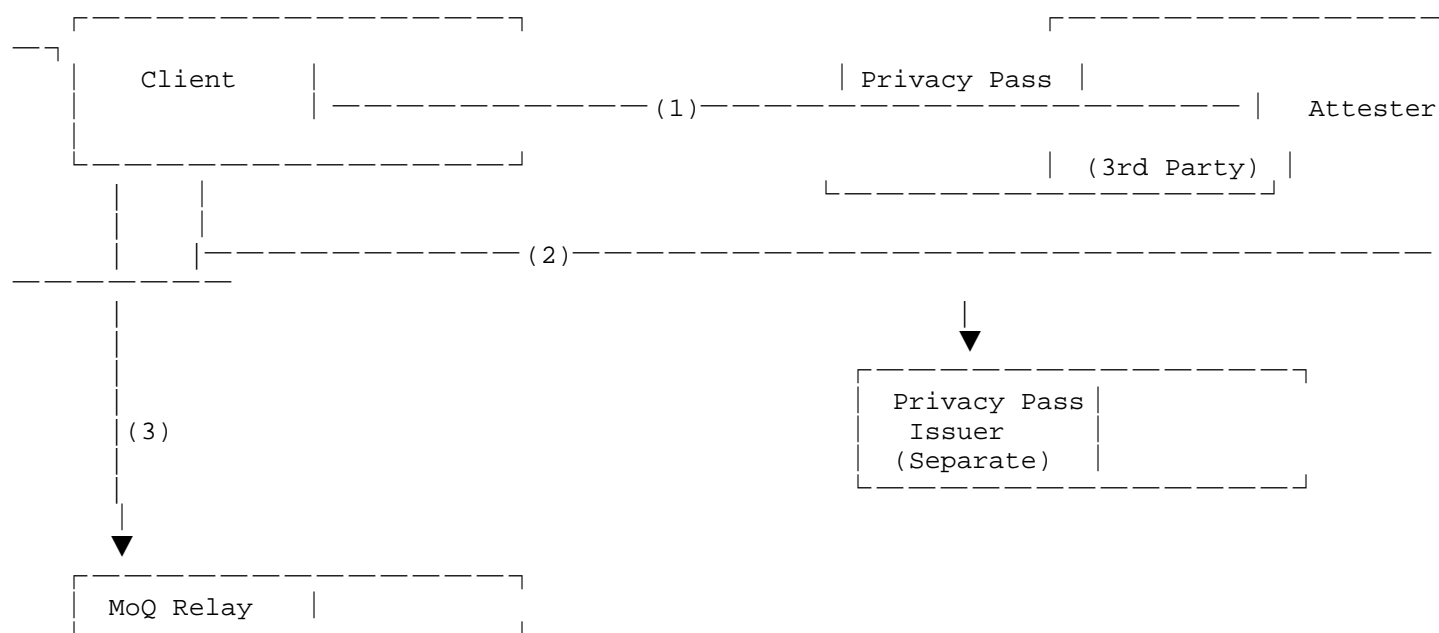
The Privacy Pass MoQ integration involves the following entities and their interactions:

- * *Client*: The MoQ client requesting access to media content. The client is responsible for obtaining Privacy Pass tokens through the attestation and issuance process, and presenting these tokens when requesting MoQ operations.

- * ***MoQ Relay/Origin***: The MoQ relay server or origin that forwards media content and requires authorization. The relay validates Privacy Pass tokens, enforces access policies, and forwards authorized requests to origins or other relays. Relays maintain configuration for trusted issuers and validate token signatures and metadata.
- * ***Privacy Pass Issuer***: The entity that issues Privacy Pass tokens to clients after successful attestation. The issuer operates the token issuance protocol, manages cryptographic keys, and may implement rate limiting. The issuer creates tokens with appropriate MoQ-specific metadata.
- * ***Privacy Pass Attester***: The entity that attests to properties of clients for the purposes of token issuance. The attester verifies client credentials, subscription status, or other eligibility criteria. Common attestation methods include username/password, OAuth, device certificates, or other authentication mechanisms.

In the below deployment, the MoQ relay and Privacy Pass issuer are operated by different entities to enhance privacy through separation of concerns:

Separated Issuer and Relay Architecture



- (1) Client attestation with separate Attester
- (2) Token issuance from separate Issuer to Client
- (3) Client requests media access with token from relay

However, in certain deployments the MoQ relay and Privacy Pass issuer may be operated by the same entity to simplify key management and policy coordination. The details of such an integrated architecture is TBD.

2.1. Trust Model

The architecture assumes the following trust relationships based on the Privacy Pass architecture [RFC9576]:

- * Clients trust issuers not to collude with attester or break unlinkability guarantees
- * Relays trust issuers to properly validate client eligibility before issuing tokens
- * Issuers trust attesters to accurately verify client eligibility

3. Privacy Pass Token Integration

This section describes how Privacy Pass tokens are integrated into the MoQ transport protocol to provide privacy-preserving authorization for various media operations.

3.1. Token Types for MoQ Authorization

This specification uses the existing Privacy Pass token types defined in [RFC9578]:

- * *Token Type 0x0001 (VOPRF(P-384, SHA-384))*: Privately verifiable tokens using Verifiable Oblivious Pseudorandom Function for deployments requiring issuer-only validation capability.
- * *Token Type 0x0002 (Blind RSA (2048-bit))*: Publicly verifiable tokens using blind RSA signatures for deployments requiring distributed validation across multiple relays.

3.2. Token Structure

Privacy Pass tokens used in MoQ MUST follow the structure defined in [RFC9577] for the PrivateToken HTTP authentication scheme. The token structure includes:

- * *Token Type*: 2-byte identifier specifying the issuance protocol used
- * *Nonce*: 32-byte client-generated random value for uniqueness

- * ***Challenge Digest***: 32-byte SHA-256 hash of the TokenChallenge
- * ***Token Key ID***: Variable-length identifier for the issuer's public key
- * ***Authenticator***: Variable-length cryptographic proof bound to the token

3.2.1. Token Challenge Structure for MoQ

MoQ-specific TokenChallenge structures use the default format defined in [RFC9577] with MoQ-specific parameters in the origin_info field:

```
struct {  
    uint16_t token_type;  
    opaque issuer_name<1..2^16-1>;  
    opaque redemption_context<0..32>;  
    opaque origin_info<0..2^16-1>;  
} TokenChallenge;
```

For MoQ usage, the origin_info field contains MoQ-specific authorization scope information encoded as a UTF-8 string with the following format:

TODO: Define origin_info to be binary format

```
moq-scope = operation ":" namespace-pattern [ ":" track-pattern ]  
operation = "subscribe" / "fetch" / "publish" / "announce"  
namespace-pattern = exact-match / prefix-match  
track-pattern = exact-match / prefix-match  
exact-match = namespace/name-string  
prefix-match = namespace/name-string"*"
```

Examples:

- * subscribe:sports.example.com/live/* - Subscribe to any track under live sports
- * fetch:vod.example.com/movies/action* - Fetch video-on-demand action content
- * publish:meetings.example.com/meeting/m123/audio/opus48000 - Publish content for meeting m123

3.3. Track Namespace and Track Name Matching Rules

This specification defines prefix-based matching rules for track namespaces and track names to enable fine-grained access control while maintaining privacy.

3.3.1. Namespace Matching

Track namespace matching supports three modes:

***Exact Match*:**

- * Pattern: "example.com/live/sports/soccer"
- * Matches: Only the exact namespace example.com/live/sports/soccer

***Prefix Match*:**

- * Pattern: "example.com/live/sports/*"
- * Matches: Any namespace starting with example.com/live/sports/
- * Examples: example.com/live/sports/soccer, example.com/live/sports/tennis

3.3.2. Track Name Matching

Track name matching within authorized namespaces follows the same pattern:

***Exact Match*:**

- * Pattern: "video"
- * Matches: Only tracks named exactly video

***Prefix Match*:**

- * Pattern: "video*"
- * Matches: Any track name starting with video
- * Examples: video-med, video-high, video-low

3.3.3. Matching Algorithm

When a MoQ relay receives a request with a Privacy Pass token, it performs the following validation steps to determine whether to authorize the requested operation:

1. Extract the Privacy Pass token from the MoQ control message (SETUP, SUBSCRIBE, FETCH, PUBLISH, or ANNOUNCE)
2. Verify the token signature using the appropriate issuer public key based on the token type:
 - * For Token Type 0x0001 (VOPRF): Use the issuer's private validation key
 - * For Token Type 0x0002 (Blind RSA): Use the issuer's public verification key
3. Validate that the token has not been replayed by checking:
 - * Token nonce uniqueness within the issuer's replay window
 - * Token expiration timestamp (if present in token metadata)
4. Extract the MoQ-specific authorization scope from the token's `origin_info` field:
 - * Authorized operation type (subscribe, fetch, publish, announce)
 - * Namespace pattern (exact match or prefix match)
 - * Track name pattern (exact match or prefix match, optional)
5. Verify that the requested MoQ operation matches the operation specified in the token scope:
 - * SUBSCRIBE operations require "subscribe" scope
 - * FETCH operations require "fetch" scope
 - * PUBLISH operations require "publish" scope
 - * ANNOUNCE operations require "announce" scope
6. Apply namespace/name matching rules based on the pattern type:

- * If Exact Match, the requested namespace/name MUST exactly equal the pattern
- * If Prefix Match, the requested namespace/name MUST start with the pattern prefix

Access is granted to the requested resource if and only if ALL of the following conditions are met:

- * Token signature verification succeeds
- * Token nonce has not been previously used (replay protection)
- * Token has not expired (if applicable)
- * Requested operation matches token operation scope
- * Requested namespace matches token namespace pattern
- * Requested track name matches token track name pattern (if specified)

else, authorization error is returned to the requesting client.

3.4. Token in MoQ Messages

Privacy Pass tokens are provided to MoQ relays using the existing MoQ authorization framework with the following adaptations:

3.4.1. SETUP Message Authorization

For connection-level authorization, Privacy Pass tokens are included in the SETUP message's authorization parameter:

```
SETUP {
  Version = 1,
  Parameters = [
    {
      Type = AUTHORIZATION,
      Value = base64url(PrivateTokenAuth)
    }
  ]
}

struct {
  uint8_t auth_scheme = 0x01; // Privacy Pass
  opaque token_data<1..2^16-1>;
} PrivateTokenAuth;
```

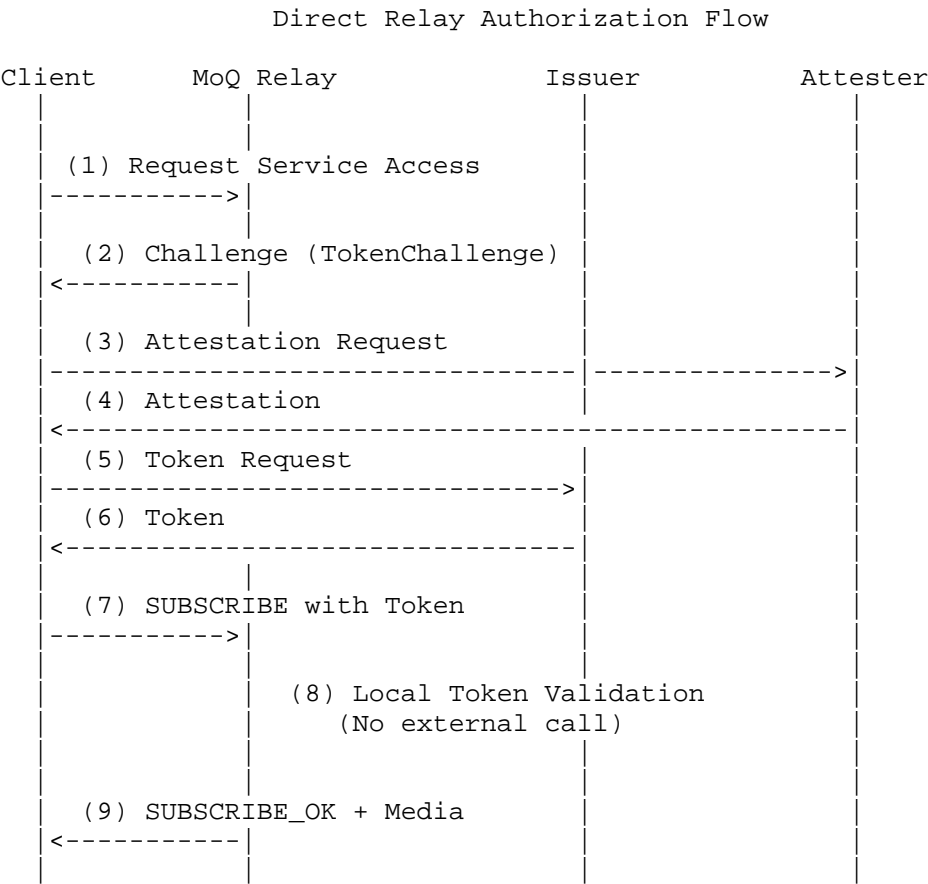
3.4.2. MoQ Operation-Level Authorization

For individual MoQ operation authorization, tokens are included in operation-specific control messages:

```
SUBSCRIBE {  
  Track_Namespace = "sports.example.com/live/soccer",  
  Track_Name = "video",  
  Parameters = [  
    {  
      Type = AUTHORIZATION,  
      Value = base64url(PrivateTokenAuth)  
    }  
  ]  
}
```

4. Example Authorization Flow

Below shows an example deployment scenarios where the relay has been configured with the necessary validation keys and content policies, the relay can verify Privacy Pass tokens locally and deliver media directly without contacting the Origin.



5. Security Considerations

TODO: Add considerations for the security and privacy of the Privacy Pass tokens.

6. IANA Considerations

TODO

7. References

7.1. Normative References

[MoQ-TRANSPORT]
"Media over QUIC Transport", n.d.,
<<https://www.ietf.org/archive/id/draft-ietf-moq-transport-12.txt>>.

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/rfc/rfc2119>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/rfc/rfc8174>>.
- [RFC9576] "The Privacy Pass Architecture", n.d., <<https://www.rfc-editor.org/rfc/rfc9576.txt>>.
- [RFC9577] "The Privacy Pass HTTP Authentication Scheme", n.d., <<https://www.rfc-editor.org/rfc/rfc9577.txt>>.
- [RFC9578] "Privacy Pass Issuance Protocols", n.d., <<https://www.rfc-editor.org/rfc/rfc9578.txt>>.

7.2. Informative References

- [RFC9458] "Oblivious HTTP", n.d., <<https://www.rfc-editor.org/rfc/rfc9458.txt>>.

Authors' Addresses

Suhas Nandakumar
Cisco
Email: snandaku@cisco.com

Cullen Jennings
Cisco
Email: fluffy@iii.ca