Media Over QUIC
Internet-Draft
Intended status: Standards Track
Expires: 23 April 2026

S. Nandakumar
C. Jennings
Cisco
T. Meunier
Cloudflare Inc.
20 October 2025

Privacy Pass Authentication for Media over QUIC (MoQ)
draft-ietf-moq-privacy-pass-auth-01

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.

About This Document

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

The latest revision of this draft can be found at https://moq-wg.github.io/privacy-pass/draft-ietf-moq-privacy-pass-auth.html. 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/privacy-pass.

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/.

Copyright Notice

Table of Contents

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

Privacy Pass Terminology defined in Section 2 of [RFC9576] is reused here. 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. 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.

2.1. Joint Attester and Issuer

In the below deployment, the MoQ relay and Privacy Pass issuer are operated by different entities to enhance privacy through separation of concerns. This corresponds to Section 4.4 of [RFC9576].

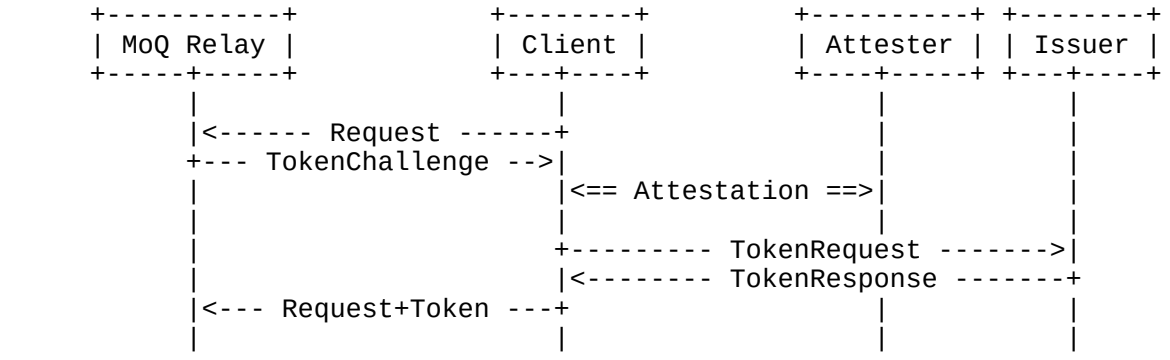


Figure 1: Separated Issuer and Relay Architecture

In certain deployments the MoQ relay and Privacy Pass issuer may be operated by the same entity to simplify key management and policy coordination. This is the Privacy Pass deployment described in Section 4.2 of [RFC9576].

2.2. Shared Origin, Attester, Issuer with a Reverse Flow

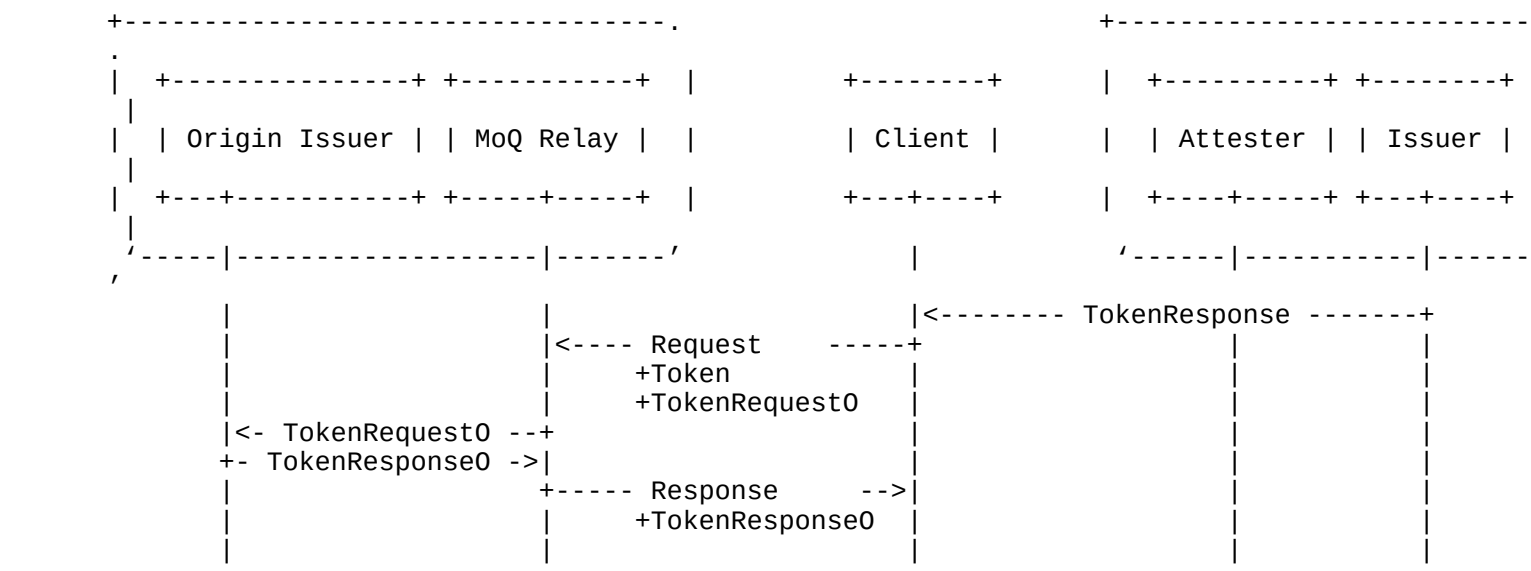
The flow described above can be used to bootstrap a shared origin-attester-issuer flow, as described in Section 4.2 of [RFC9576]. The MoQ relay plays all role, allowing it to use privately verifiable token types registered in [PRIVACYPASS-IANA].

In this scenario, the MoQ relay origin would accept tokens signed by two issuers:

1. Type 0x0002 token signed by the bootstrap issuer from Section 2.1

2. Type 0x0001, 0x0005, or 0xE5AC tokens signed by its own issuer.

With [PRIVACYPASS-ARC], the flow would look as follow



TokenRequestO and TokenResponseO are part of a reverse flow. The client request a new token/credential to the origin. It allows the client to exchange its initial 0x0002 Token against a privately verifiable token issued by the origin.

TokenRequestO should correspond to the associated privately verifiable token definition. These are listed in Section 3.1.

All privately verifiable scheme allow to amortise token issuance cost, making them more compelling in a streaming case. This is specified in Section 5 of [PRIVACYPASS-BATCHED].

When using 0xE5AC, TokenRequestO is a CredentialRequest defined in Section 7.1 of [PRIVACYPASS-ARC].

2.3. Trust Model

The architecture assumes the following trust relationships based on Section 3 of [RFC9576]:

* Issuer and attesters do not collude to guarantee unlinkability properties

* 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 below existing Privacy Pass token types:

Publicly verifiable token types

* 0x0002 (Blind RSA (2048-bit)): Defined in Section 6 of [RFC9578]. Uses blind RSA signatures ([RFC9474]) for deployments requiring distributed validation across multiple relays.

Privately verifiable token types

* 0x0001 (VOPRF(P-384, SHA-384)): Defined in Section 6 of [RFC9578]. Uses VOPRF ([RFC9497]) for deployments where the origin is the issuer. Issuance can be batched as defined in Section 5 of [PRIVACYPASS-BATCHED].

* 0x0005 (VOPRF(ristretto255, SHA-512)): Defined in Section 8.1 of [PRIVACYPASS-BATCHED]. Uses VOPRF ([RFC9497]) for deployments where the origin is the issuer. Issuance can be batched as defined in Section 5 of [PRIVACYPASS-BATCHED].

* 0xE5AC (ARC(P-256)): Anonymous Rate Limit Credentials Token using [ARC]. Tokens are presented by clients based on an issued credential and up to a presentation_limit.

3.2. Token Structure

Privacy Pass tokens used in MoQ MUST follow the structure defined in Section 2.2 of [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 Section 2.1 of [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 using TLS presentation language For anonymous credentials, it would make a lot more sense to use redemption_context instead of origin_info, as redemption_context is decided upon at presentation time rather than at issuance time. Question that I don't have the answer yet: are 32-bytes enough? we might need to say something like "first 2 bytes represent the type of rule, then data"

```
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(P-384, SHA-384)): Use the issuer’s private validation key

    * For Token Type 0x0002 (Blind RSA(2048 bits)): 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 granded 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) specified)

else, authorzation 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 Issuer. This uses token with public verifiability.

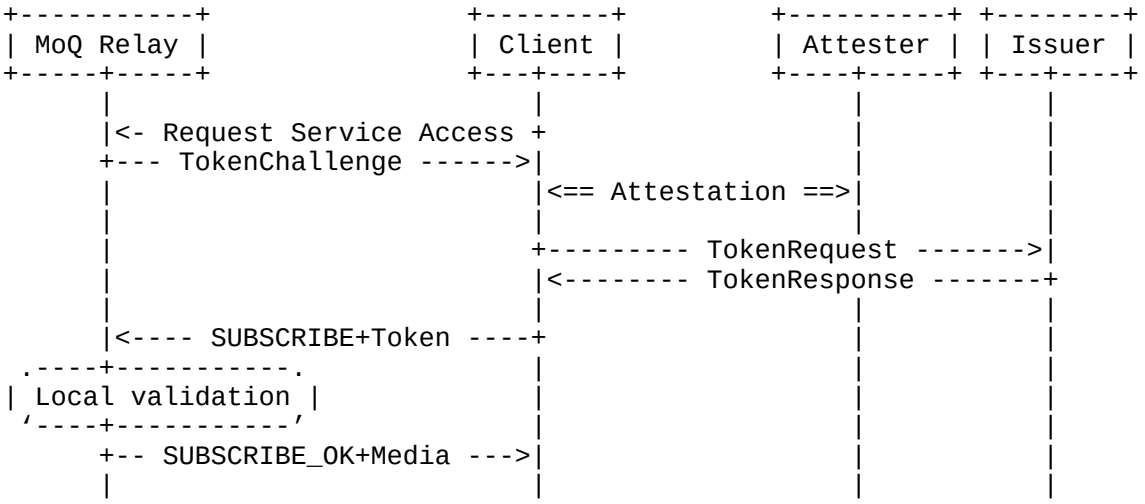


Figure 2: Direct Relay Authorization Flow

5. Security Considerations

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

* Token Replay
* Token harvest
* Key rotation
* Use of TLS

6. IANA Considerations

TODO

* Register namespace?
* New registry for auth_scheme with 0x01 as the first registered auth_scheme

7. References

7.1. Normative References

[ARC] Yun, C. and C. A. Wood, "Anonymous Rate-Limited Credentials", Work in Progress, Internet-Draft, draft-yun-cfrg-arc-01, 6 August 2025, <https://datatracker.ietf.org/doc/html/draft-yun-cfrg-arc-01>.

[MoQ-TRANSPORT] Nandakumar, S., Vasiliev, V., Swett, I., and A. Frindell, "Media over QUIC Transport", Work in Progress, Internet-Draft, draft-ietf-moq-transport-14, 2 September 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-moq-transport-14>.

[PRIVACYPASS-ARC] Yun, C. and C. A. Wood, "Privacy Pass Issuance Protocol for Anonymous Rate-Limited Credentials", Work in Progress, Internet-Draft, draft-yun-privacypass-arc-01, 6 August 2025, <https://datatracker.ietf.org/doc/html/draft-yun-privacypass-arc-01>.

[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>.

[RFC9474] Denis, F., Jacobs, F., and C. A. Wood, "RSA Blind Signatures", RFC 9474, DOI 10.17487/RFC9474, October 2023, <https://www.rfc-editor.org/rfc/rfc9474>.

[RFC9497] Davidson, A., Faz-Hernandez, A., Sullivan, N., and C. A. Wood, "Oblivious Pseudorandom Functions (OPRFs) Using Prime-Order Groups", RFC 9497, DOI 10.17487/RFC9497, December 2023, <https://www.rfc-editor.org/rfc/rfc9497>.

[RFC9576] Davidson, A., Iyengar, J., and C. A. Wood, "The Privacy Pass Architecture", RFC 9576, DOI 10.17487/RFC9576, June 2024, <https://www.rfc-editor.org/rfc/rfc9576>.

[RFC9577] Pauly, T., Valdez, S., and C. A. Wood, "The Privacy Pass HTTP Authentication Scheme", RFC 9577, DOI 10.17487/RFC9577, June 2024, <https://www.rfc-editor.org/rfc/rfc9577>.

[RFC9578] Celi, S., Davidson, A., Valdez, S., and C. A. Wood, "Privacy Pass Issuance Protocols", RFC 9578, DOI 10.17487/RFC9578, June 2024, <https://www.rfc-editor.org/rfc/rfc9578>.

7.2. Informative References

[PRIVACYPASS-BATCHED] Robert, R., Wood, C. A., and T. Meunier, "Batched Token Issuance Protocol", Work in Progress, Internet-Draft, draft-ietf-privacypass-batched-tokens-05, 3 July 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-privacypass-batched-tokens-05>.

[PRIVACYPASS-IANA] "Privacy Pass IANA", n.d., <https://www.iana.org/assignments/privacy-pass/privacy-pass.xhtml>.

[PRIVACYPASS-REVERSE-FLOW] "*** BROKEN REFERENCE ***".

[RFC9458] Thomson, M. and C. A. Wood, "Oblivious HTTP", RFC 9458, DOI 10.17487/RFC9458, January 2024, <https://www.rfc-editor.org/rfc/rfc9458>.

Appendix A. Acknowledgments

TODO acknowledge.

Appendix B. Change Log

RFC Editor's Note: Please remove this section prior to publication of a final version of this document.

B.1. Since draft-ietf-moq-privacy-pass-auth-00

* Add Thibault Meunier as Coauthor

* Add support for Reverse flow to be deploy and scale friendly way to get tokens

Authors' Addresses

Suhas Nandakumar
Cisco
Email: snandaku@cisco.com

Cullen Jennings
Cisco
Email: fluffy@iii.ca

Thibault Meunier
Cloudflare Inc.
Email: ot-ietf@thibault.uk