Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: 23 April 2026

C. Yun
C. A. Wood
Apple, Inc.
M. Raykova
S. Schlesinger
Google

20 October 2025

Anonymous Token with Hidden Metadata Privacy Pass Issuance and Authentication Protocols
draft-yun-privacypass-athm-00

Abstract

This document specifies the issuance and redemption protocols for tokens based on the Anonymous Tokens with Hidden Metadata (ATHM) protocol.

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://cathieyun.github.io/draft-athm/draft-yun-privacypass-athm.html. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-yun-privacypass-athm/.

Discussion of this document takes place on the Privacy Pass mailing list (mailto:privacy-pass@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/privacy-pass. Subscribe at https://www.ietf.org/mailman/listinfo/privacy-pass/.

Source for this draft and an issue tracker can be found at https://github.com/cathieyun/draft-athm.

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

The Privacy Pass architecture introduced in [ARCHITECTURE] defines an anonymous, single use token scheme. The strongest-anonymity instantiation of a Privacy Pass token scheme carries exactly one bit of information about the client, namely the fact that the client was issued a valid token as a trust signal, and this bit is known to the client who can verify that it received a valid token during issuance.

The architecture further defines options for private or public metadata that can be embeded in the token by the issuer.

While in many applications the trust signals that are attributed to clients via anonymous credentials should be known to them, there are cases where these signals need to remain hidden from the clients as a prerequisite for the effectiveness of the token scheme as a trust conveying mechanism. These settings relate to fraud detection, where allowing the attacker to learn that it has been flagged as fraudulent enables them to adapt and update their attack strategies much more effectively.

An Anonymous Token with Hidden Metadata (ATHM), as specified in [ATHM-SPEC], allows the issuer to embed a fixed number of hidden metadata bits in the issued token. These metadata bits are only readable at token redemption by the party holding the secret key for the scheme. These bits reduce the anonymity properties of the tokens by allowing the issuer to partition clients into as many groups as the domain of metadata, in a secret way. That is why it is important to carefully choose the limit on the allowed metadata bits. The client can verify that the token it was issued does not contain more metadata bits than allowed.

The ATHM tokens provide a way to enable hidden trust signals in an authentication token, while achieving the strongest anonymity properties possible in this setting. Such tokens provide a bridge for using anonymous credentials in fraud applications.

2. Terminology

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.

This document uses the terms Origin, Client, Issuer, and Token as defined in Section 2 of [ARCHITECTURE]. Moreover, the following additional terms are used throughout this document.

* Issuer Public Key: The public key (from a private-public key pair) used by the Issuer for issuing and verifying Tokens.

* Issuer Private Key: The private key (from a private-public key pair) used by the Issuer for issuing and verifying Tokens.

Unless otherwise specified, this document encodes protocol messages in TLS notation from Section 3 of [TLS13]. Moreover, all constants are in network byte order.

3. Protocol Overview

The issuance and redemption protocols defined in this document are built on an anonymous credential construction called ATHM, as specified in [ATHM-SPEC]. ATHM is a privately verifiable token with support for n private metadata buckets.

Unlike the core Privacy Pass token types specified in [ISSUANCE], ATHM tokens are not cryptographically bound to TokenChallenge messages; see [AUTHSCHEME] for details about how this binding typically works. Instead, with ATHM, Clients can request tokens from an Issuer without a preceding TokenChallenge, and present these tokens to the Origin during presentation. This interaction is shown below.

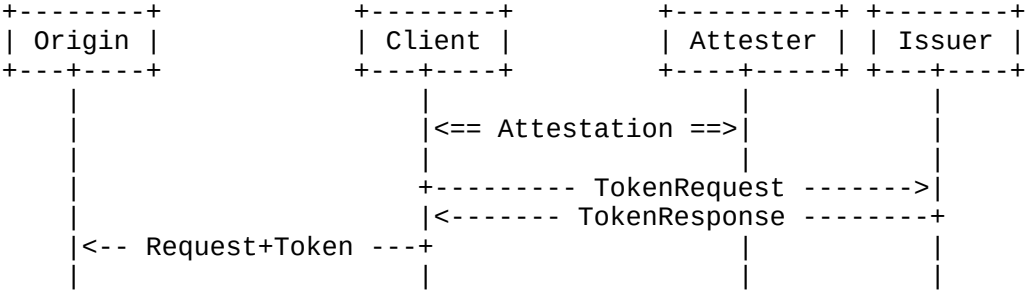

Figure 1: Issuance and Redemption Overview

Unlike the core Privacy Pass protocols, TokenChallenge values are not inputs to the issuance protocol or redemption protocols. As such, ATHM tokens require their own Token format, which is specified in Section 6.

ATHM is only compatible with deployment models where the Issuer and Origin are operated by the same entity (see Section 4 of [ARCHITECTURE]), as tokens produced from a credential are not publicly verifiable. The details of attestation are outside the scope of the issuance protocol; see Section 4 of [ARCHITECTURE] for information about how attestation can be implemented in each of the relevant deployment models.

4. Configuration

ATHM issuers are configured with key material used for issuance and credential verification. Concretely, Issuers run the KeyGen function from [ATHM-SPEC] to produce a secret and public key, denoted skI and pkI, respectively.

skI, pkI = KeyGen()

The Issuer Public Key ID, denoted issuer_key_id, is computed as the SHA-256 hash of the Verified Issuer Public Key, i.e., issuer_key_id = SHA-256(Serialize(verifiedPublicKey)).

5. Token Issuance Protocol

Issuers provide a Issuer Private and Public Key, denoted skI and pkI respectively, used to produce tokens as input to the protocol. See Section 4 for how these keys are generated.

Clients provide the following as input to the issuance protocol:

* Issuer Request URL: A URL identifying the location to which issuance requests are sent. This can be a URL derived from the "issuer-request-uri" value in the Issuer's directory resource, or it can be another Client-configured URL. The value of this parameter depends on the Client configuration and deployment model. For example, in the 'Joint Origin and Issuer' deployment model, the Issuer Request URL might correspond to the Client's configured Attester, and the Attester is configured to relay requests to the Issuer.

* Issuer name: An identifier for the Issuer. This is typically a host name that can be used to construct HTTP requests to the Issuer.

* Issuer Public Key: pkI, with a key identifier token_key_id computed as described in Section 4.

Given this configuration and these inputs, the two messages exchanged in this protocol to produce a credential are described below.

5.1. Client-to-Issuer Request

Given Issuer Public Key pkI, the Client first verifies the public key to make a verified public key:

verifiedPublicKey = VerifyPublicKeyProof(publicKey, pi)

Next, it creates a token request message using the TokenRequest function from [ATHM-SPEC] as follows:

```
(context, request) = TokenRequest(verifiedPublicKey)
```

The Client then creates a TokenRequest structure as follows:

```
struct {
  uint16_t token_type = 0xC07E; /* Type ATHM(P-256) */
  uint8_t truncated_issuer_key_id;
  uint8_t encoded_request[Nrequest];
} TokenRequest;
```

The structure fields are defined as follows:

* "token_type" is a 2-octet integer.

* "truncated_issuer_key_id" is the least significant byte of the issuer_key_id, the Issuer Public Key ID corresponding to pkI, in network byte order (in other words, the last 8 bits of issuer_key_id). This value is truncated so that Issuers cannot use issuer_key_id as a way of uniquely identifying Clients; see Section 7 and referenced information for more details.

* "encoded_request" is the Nrequest-octet request, computed as the serialization of the request value as defined in [ATHM-SPEC].

The Client then generates an HTTP POST request to send to the Issuer Request URL, with the TokenRequest as the content. The media type for this request is "application/private-token-request". An example request for the Issuer Request URL "https://issuer.example.net/request" is shown below.

[[QUESTION: Should we reuse the same content type for this request, or should we introduce a new content type?]]

```
POST /request HTTP/1.1
Host: issuer.example.net
Accept: application/private-token-response
Content-Type: application/private-token-request
Content-Length: <Length of TokenRequest>

<Bytes containing the TokenRequest>
```

5.2. Issuer-to-Client Response

Upon receipt of the request, the Issuer validates the following conditions:

* The TokenRequest contains a supported token_type equal to value 0xC07E.

* The TokenRequest.truncated_token_key_id corresponds to the truncated key ID of an Issuer Public Key, with corresponding secret key skI, owned by the Issuer.

* The TokenRequest.encoded_request is of the correct size (Nrequest).

If any of these conditions is not met, the Issuer MUST return an HTTP 422 (Unprocessable Content) error to the client.

If these conditions are met, the Issuer then tries to deserialize TokenRequest.encoded_request according to [ATHM-SPEC], yielding request. If this fails, the Issuer MUST return an HTTP 422 (Unprocessable Content) error to the client. Otherwise, if the Issuer is willing to produce a token for the Client with a hidden metadata bucket, denoted hiddenMetadata, the Issuer completes the issuance flow by an issuance response as follows:

```
response = TokenResponse(skI, pkI, request, hiddenMetadata)
```

The Issuer then creates a TokenResponse structured as follows:

```
struct {
   uint8_t encoded_response[Nresponse];
} TokenResponse;
```

The structure fields are defined as follows:

* "encoded_response" is the Nresponse-octet encoded issuance response message, computed as the serialization of response as specified in [ATHM-SPEC].

The Issuer generates an HTTP response with status code 200 whose content consists of TokenResponse, with the content type set as "application/private-token-response".

```
HTTP/1.1 200 OK
Content-Type: application/private-token-response
Content-Length: <Length of TokenResponse>

<Bytes containing the TokenResponse>
```

5.3. Token Finalization

Upon receipt, the Client handles the response and, if successful, deserializes the content values TokenResponse.encoded_response according to [ATHM-SPEC] yielding response. If deserialization fails, the Client aborts the protocol. Otherwise, the Client processes the response as follows:

```
token = FinalizeToken(context, verifiedPublicKey, request, response)
```

The Client then saves the resulting token structure for use with a future redemption.

6. Token Redemption Protocol

The token redemption protocol presents a Token to the Origin for redemption. This section describes how the Token values are encoded in the redemption protocol and then verified by the Origin.

6.1. Token Structure

```
struct {
    uint16_t token_type = 0xC07E; /* Type ATHM(P-256) */
    uint8_t issuer_key_id[Nid];
    uint8_t token[Ntoken];
} Token;
```

The structure fields are defined as follows:

* "token_type" is a 2-octet integer, in network byte order, equal to 0xC7D3.

* "issuer_key_id" is a Nid-octet identifier for the Issuer Public Key, computed as defined in Section 4.

* "token" is a Ntoken-octet token, set to the serialized token value (see [ATHM-SPEC] for serialization details); see Section 6.2 for more information about how this field is used in verifying a token.

6.2. Token Verification

Verifying a Token requires invoking the VerifyToken function from [ATHM-SPEC] with input skI, pkI, and token in the following way:

```
hiddenMetadata = VerifyToken(skI, pkI, token)
```

This function will fail with an error if the token is invalid. Otherwise, it will return an integer value corresponding to the bucket bound to the token during issuance.

7. Security Considerations

ATHM is a privately verifiable token scheme, and therefore, the Issuer and Origin have joint configuration. The privacy consideration from Section 6 of [ARCHITECTURE] apply to deployments of the ATHM scheme. The Origin-Client unlinkability of the ATHM depends on the size of the hidden metadata, which reduced the size of the anonymity sets allowing the issuer to partition clients according to their token metedata bits. As discussed in Section 6.1 of [ARCHITECTURE] this could enable the Issuer to track as many clients as the domain size of the metadata. Since the metadata is private the assigned anonymity sets to clients remain hidden, e.g., if the issuer is trying to track a small set of client, it can hide which these clients are. As suggested in Section 6.1 of [ARCHITECTURE] each deployment should carefully consider the balance of the utility obtained by the private metadata and the reduction of privacy and chose a setting that most closely alignes with its goals.

8. IANA Considerations

This document updates the "Privacy Pass Token Type" Registry with the following entries.

* Value: 0xC07E
* Name: ATHM(P-256)
* Token Structure: As defined in Section 2.2 of [AUTHSCHEME]
* Token Key Encoding: Serialized as described in Section 4
* TokenChallenge Structure: As defined in Section 2.1 of [AUTHSCHEME]
* Public Verifiability: N
* Public Metadata: N
* Private Metadata: Y
* Nk: 48
* Nid: 32

   * Reference: This document

   * Notes: None

9. Normative References

[ARCHITECTURE]
            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>.

[ATHM-SPEC]
            "*** BROKEN REFERENCE ***".

[AUTHSCHEME]
            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>.

[ISSUANCE] 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>.

[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>.

[TLS13]   Rescorla, E., "The Transport Layer Security (TLS) Protocol
            Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018,
            <https://www.rfc-editor.org/rfc/rfc8446>.

Acknowledgments

Thanks to Melissa Chase for discussion about the ATHM paper. Thanks also to Tommy Pauly, Phillipp Schoppmann and Ghous Amjad for collaboration on the ATHM specifications.

Authors’ Addresses

Cathie Yun
Apple, Inc.

Email: cathieyun@gmail.com

Christopher A. Wood
Apple, Inc.
Email: caw@heapingbits.net

Mariana Raykova
Google
Email: marianar@google.com

Samuel Schlesinger
Google
Email: sgschlesinger@gmail.com