Workgroup: Network Working Group
Internet-Draft:
draft-ietf-privacypass-auth-scheme-05
Published: 5 August 2022
Intended Status: Standards Track
Expires: 6 February 2023
Authors: T. Pauly   S. Valdez   C. A. Wood
Apple Inc.   Google LLC   Cloudflare

The Privacy Pass HTTP Authentication Scheme

Abstract

This document defines an HTTP authentication scheme that can be used by clients to redeem Privacy Pass tokens with an origin. It can also be used by origins to challenge clients to present an acceptable Privacy Pass token.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Privacy Pass tokens are unlinkable authenticators that can be used to anonymously authorize a client (see [ARCHITECTURE]). A client possessing such a token is able to prove that it was able to get a token issued by a token issuer -- based on some check from a token issuer, such as authentication or solving a CAPTCHA -- without allowing the relying party redeeming the client's token (the origin) to link it with issuance flow.

Different types of authenticators, using different token issuance protocols, can be used as Privacy Pass tokens.

This document defines a common HTTP authentication scheme ([RFC7235]), PrivateToken, that allows clients to redeem various kinds of Privacy Pass tokens.

Clients and relying parties interact using this scheme to perform the token challenge and token redemption flow. Clients use a token issuance protocol to actually fetch tokens to redeem.

```
 Client                             Relying Party (Origin)

    <------------------------------ Challenge \
                                               |
+----------------------------------\           |
|                                  |           |
|  Issuance Protocol               |           |
|                                  |           |
+----------------------------------/           |
                                               |
     Redemption -------------------------- >  /
```

Figure 1: Token Architectural Components

In addition to working with different token issuance protocols, this scheme supports optionally associating tokens with origin-chosen contexts and specific origin names. Relying parties that request and redeem tokens can choose a specific kind of token, as appropriate for its use case. These options allow for different deployment models to prevent double-spending, and allow for both interactive (online challenges) and non-interactive (pre-fetched) tokens.

1.1. 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.

Unless otherwise specified, this document encodes protocol messages in TLS notation from [TLS13], Section 3.

This document uses the terms "Client", "Origin", "Issuer", "Issuance Protocol", and "Token" as defined in [ARCHITECTURE]. It additionally uses the following terms in more specific ways:

* Issuer key: Keying material that can be used with an issuance protocol to create a signed token.

* Token challenge: A requirement for tokens sent from an origin to a client, using the "WWW-Authenticate" HTTP header. This challenge is bound to a specific token issuer and issuance protocol, and may be additionally bound to a specific context or origin name.

* Token redemption: An action by which a client presents a token to an origin, using the "Authorization" HTTP header.

2. HTTP Authentication Scheme

Token redemption is performed using HTTP Authentication ([RFC7235]), with the scheme "PrivateToken". Origins challenge clients to present a token from a specific issuer (Section 2.1). Once a client has received a token from that issuer, or already has a valid token available, it presents the token to the origin (Section 2.2).

2.1. Token Challenge

Origins send a token challenge to clients in an "WWW-Authenticate" header with the "PrivateToken" scheme. This challenge includes a TokenChallenge message, along with information about what keys to use when requesting a token from the issuer.

Origins that support this authentication scheme need to handle the following tasks:

1. Select which issuer to use, and configure the issuer name and token-key to include in WWW-Authenticate challenges.

2. Determine a redemption context construction to include in the TokenChallenge, as discussed in Section 2.1.1.

3. Select the origin information to include in the TokenChallenge. This can be empty to allow fully cross-origin tokens, a single origin name that matches the origin itself, or a list of origin names containing the origin.

The TokenChallenge message has the following structure:

```
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;
```

The structure fields are defined as follows:

* "token_type" is a 2-octet integer, in network byte order. This type indicates the issuance protocol used to generate the token. Values are registered in an IANA registry, Section 5.2. Challenges with unsupported token_type values MUST be ignored.

* "issuer_name" is a string containing the name of the issuer. This is a hostname that is used to identify the issuer that is allowed to issue tokens that can be redeemed by this origin. The string is prefixed with a 2-octet integer indicating the length, in network byte order.

* "redemption_context" is an optional field. If present, it allows the origin to require that clients fetch tokens bound to a specific context, as opposed to reusing tokens that were fetched for other contexts. See Section 2.1.1 for example contexts that might be useful in practice. When present, this value is a 32-byte context generated by the origin. Valid lengths for this field are either 0 or 32 bytes. The field is prefixed with a single octet indicating the length. Challenges with redemption_context values of invalid lengths MUST be ignored.

* "origin_info" is an optional string containing one or more origin names, which allows a token to be scoped to a specific set of origins. The string is prefixed with a 2-octet integer indicating the length, in network byte order. If empty, any non-origin-specific token can be redeemed. If the string contains multiple origin names, they are delimited with commas "," without any whitespace.

When used in an authentication challenge, the "PrivateToken" scheme uses the following attributes:

* "challenge", which contains a base64url-encoded [RFC4648] TokenChallenge value. Since the length of the challenge is not fixed, the base64url data MUST include padding. This MUST be unique for every 401 HTTP response to prevent replay attacks. This attribute is required for all challenges.

* "token-key", which contains a base64url encoding of the public key for use with the issuance protocol indicated by the challenge. Since the length of the key is not fixed, the base64url data MUST include padding. This attribute MAY be omitted in deployments where clients are able to retrieve the issuer key using an out-of-band mechanism.

* "max-age", an optional attribute that consists of the number of seconds for which the challenge will be accepted by the origin.

Clients can ignore the challenge if the token-key is invalid or otherwise untrusted.

The header MAY also include the standard "realm" attribute, if desired. Issuance protocols MAY require other attributes.

As an example, the WWW-Authenticate header could look like this:

```
WWW-Authenticate: PrivateToken challenge=abc..., token-key=123...
```

Upon receipt of this challenge, a client uses the message and keys in the issuance protocol indicated by the token_type. If the TokenChallenge has a token_type the client does not recognize or support, it MUST NOT parse or respond to the challenge. If the TokenChallenge contains a non-empty origin_info field, the client MUST validate that the name of the origin that issued the authentication challenge is included in the list of origin names. Clients MAY have further restrictions and requirements around validating when a challenge is considered acceptable or valid. For example, clients can choose to reject challenges that list origin names for which current connection is not authoritative (according to the TLS certificate).

Caching and pre-fetching of tokens is discussed in Section 2.1.2.

Note that it is possible for the WWW-Authenticate header to include multiple challenges. This allows the origin to indicate support for different token types, issuers, or to include multiple redemption contexts. For example, the WWW-Authenticate header could look like this:

```
WWW-Authenticate: PrivateToken challenge=abc..., token-key=123...,
PrivateToken challenge=def..., token-key=234...
```

Origins should only include challenges for different types of issuance protocols with functionally equivalent properties. For instance, both issuance protocols in [ISSUANCE] have the same functional properties, albeit with different mechanisms for verifying the resulting tokens during redemption. Since clients are free to choose which challenge they want to consume when presented with options, mixing multiple challenges with different functional properties for one use case is nonsensical.

2.1.1. Redemption Context Construction

The TokenChallenge redemption context allows the origin to determine the context in which a given token can be redeemed. This value can be a unique per-request nonce, constructed from 32 freshly generated random bytes. It can also represent state or properties of the client session. Some example properties and methods for constructing the corresponding context are below. This list is not exhaustive.

* Context bound to a given time window: Construct redemption context as SHA256(current time window).

* Context bound to a client location: Construct redemption context as SHA256(client IP address prefix).

* Context bound to a given time window and location: Construct redemption context as SHA256(current time window, client IP address prefix).

An empty redemption context is not bound to any property of the client session. Preventing double spending on tokens requires the origin to keep state associated with the redemption context. The size of this state varies based on the size of the redemption context. For example, double spend state for unique, per-request redemption contexts does only needs to exist within the scope of the request connection or session. In contrast, double spend state for empty redemption contexts must be stored and shared across all requests until token-key expiration or rotation.

Origins that share redemption contexts, i.e., by using the same redemption context, choosing the same issuer, and providing the same origin_info field in the TokenChallenge, must necessarily share state required to enforce double spend prevention. Origins should consider the operational complexity of this shared state before choosing to share redemption contexts. Failure to successfully synchronize this state and use it for double spend prevention can allow Clients to redeem tokens to one Origin that were issued after an interaction with another Origin that shares the context.

2.1.2. Token Caching

Clients can generate multiple tokens from a single TokenChallenge, and cache them for future use. This improves privacy by separating the time of token issuance from the time of token redemption, and also allows clients to avoid any overhead of receiving new tokens via the issuance protocol.

Cached tokens can only be redeemed when they match all of the fields in the TokenChallenge: token_type, issuer_name, redemption_context, and origin_info. Clients ought to store cached tokens based on all of these fields, to avoid trying to redeem a token that does not match. Note that each token has a unique client nonce, which is sent in token redemption (Section 2.2).

If a client fetches a batch of multiple tokens for future use that are bound to a specific redemption context (the redemption_context in the TokenChallenge was not empty), clients SHOULD discard these tokens upon flushing state such as HTTP cookies [COOKIES], or changing networks. Using these tokens in a context that otherwise would not be linkable to the original context could allow the origin to recognize a client.

2.2. Token Redemption

The output of the issuance protocol is a token that corresponds to the origin's challenge (see Section 2.1). A token is a structure that begins with a two-octet field that indicates a token type, which MUST match the token_type in the TokenChallenge structure.

```
struct {
    uint16_t token_type;
    uint8_t nonce[32];
    uint8_t challenge_digest[32];
    uint8_t token_key_id[Nid];
    uint8_t authenticator[Nk];
} Token;
```

The structure fields are defined as follows:

* "token_type" is a 2-octet integer, in network byte order. This value must match the value in the challenge (Section 2.1).

* "nonce" is a 32-octet message containing a client-generated random nonce.

* "challenge_digest" is a 32-octet message containing the hash of the original TokenChallenge, SHA256(TokenChallenge).

* "token_key_id" is an Nid-octet identifier for the the token authentication key. The value of this field is defined by the token_type and corresponding issuance protocol.

* "authenticator" is a Nk-octet authenticator that covers the preceding fields in the token. The value of this field is defined by the token_type and corresponding issuance protocol.

The authenticator value in the Token structure is computed over the token_type, nonce, context, and token_key_id fields.

When used for client authorization, the "PrivateToken" authentication scheme defines one parameter, "token", which contains the base64url-encoded Token struct. Since the length of the Token struct is not fixed, the base64url data MUST include padding. All unknown or unsupported parameters to "PrivateToken" authentication credentials MUST be ignored.

Clients present this Token structure to origins in a new HTTP request using the Authorization header as follows:

```
Authorization: PrivateToken token=abc...
```

For token types that support public verifiability, origins verify the token authenticator using the public key of the issuer, and validate that the signed message matches the concatenation of the client nonce and the hash of a valid TokenChallenge. For context-bound tokens, origins store or reconstruct the contexts of previous TokenChallenge structures in order to validate the token. A TokenChallenge MAY be bound to a specific HTTP session with client, but origins can also accept tokens for valid challenges in new sessions. Origins SHOULD implement some form of double-spend prevention that prevents a token with the same nonce from being redeemed twice. This prevents clients from "replaying" tokens for previous challenges. For context-bound tokens, this double-spend prevention can require no state or minimal state, since the context can be used to verify token uniqueness.

If a client is unable to fetch a token, it MUST react to the challenge as if it could not produce a valid Authorization response.

3. User Interaction

When used in contexts like websites, origins that challenge clients for tokens need to consider how to optimize their interaction model to ensure a good user experience.

Tokens challenges can be performed without explicit user involvement, depending on the issuance protocol. If tokens are scoped to a specific origin, there is no need for per-challenge user interaction. Note that the issuance protocol may separately involve user interaction if the client needs to be newly validated.

If a client cannot use cached tokens to respond to a challenge (either because it has run out of cached tokens or the associated context is unique), the token issuance process can add user-perceivable latency. Origins need not block useful work on token authentication. Instead, token authentication can be used in similar ways to CAPTCHA validation today, but without the need for user interaction. If issuance is taking a long time, a website could show an indicator that it is waiting, or fall back to another method of user validation.

An origin MUST NOT use more than one redemption context value for a given token type and issuer per client request. If an origin issues a large number of challenges with unique contexts, such as more than once for each request, this can indicate that the origin is either not functioning correctly or is trying to attack or overload the client or issuance server. In such cases, a client MUST ignore redundant token challenges for the same request and SHOULD alert the user if possible.

Origins MAY include multiple challenges, where each challenge refers to a different issuer or a different token type, to allow clients to choose a preferred issuer or type.

An origin MUST NOT assume that token challenges will always yield a valid token. Clients might experience issues running the issuance protocol, e.g., because the attester or issuer is unavailable, or clients might simply not support the requested token type. Origins SHOULD account for such operational or interoperability failures by offering clients an alternative type of challenge such as CAPTCHA for accessing a resource.

To mitigate the risk of deployments becoming dependent on tokens, clients and servers SHOULD grease their behavior unless explicitly configured not to. In particular, clients SHOULD ignore token challenges with some non-zero probability. Likewise, origins SHOULD randomly choose to not challenge clients for tokens with some non-zero probability. Moreover, origins SHOULD include random token types in token challenges with some non-zero probability.

4. Security Considerations

The security properties of token challenges vary depending on whether the challenge contains a redemption context or not, as well as whether the challenge is per-origin or not. For example, cross-origin tokens with empty contexts can be replayed from one party by another, as shown below.

```
 Client          Attacker                  Origin

                      <----------- Challenge \
                                             |
   <--------- Challenge                      |
                                             |
   Redemption ---->                          |
                                             |
                      Redemption ----------> /
```

Figure 2: Token Architectural Components

Token challenges that include non-empty origin_info bind tokens to one or more specific origins. As described in Section 2.1, clients only accept such challenges from origin names listed in the origin_info string. Even if multiple origins are listed, a token can only be redeemed for an origin if the challenge has an exact match for the origin_info. For example, if "a.example.com" issues a challenge with an origin_info string of "a.example.com,b.example.com", a client could redeem a token fetched for this challenge if and only if "b.example.com" also included an origin_info string of "a.example.com,b.example.com". On the other hand, if "b.example.com" had an origin_info string of "b.example.com" or "b.example.com,a.example.com" or "a.example.com,b.example.com,c.example.com", the string would not match and the client would need to use a different token.

Context-bound token challenges require clients to obtain matching tokens when challenged, rather than presenting a token that was obtained from a different context in the past. This can make it more likely that issuance and redemption events will occur at approximately the same time. For example, if a client is challenged for a token with a unique context at time T1 and then subsequently obtains a token at time T2, a colluding issuer and origin can link this to the same client if T2 is unique to the client. This linkability is less feasible as the number of issuance events at time T2 increases. Depending on the "max-age" token challenge attribute, clients MAY try to augment the time between getting challenged then redeeming a token so as to make this sort of linkability more difficult. For more discussion on correlation risks between token issuance and redemption, see [I-D.ietf-privacypass-architecture].

As discussed in Section 2.1, clients SHOULD discard any context-bound tokens upon flushing cookies or changing networks, to prevent an origin using the redemption context state as a cookie to recognize clients.

Applications SHOULD constrain tokens to a single origin unless the use case can accommodate such replay attacks.

All random values in the challenge and token MUST be generated using a cryptographically secure source of randomness.

5. IANA Considerations

5.1. Authentication Scheme

This document registers the "PrivateToken" authentication scheme in the "Hypertext Transfer Protocol (HTTP) Authentication Scheme Registry" established by [RFC7235].

Authentication Scheme Name: PrivateToken

Pointer to specification text: Section 2 of this document

5.2. Token Type Registry

IANA is requested to create a new "Privacy Pass Token Type" registry in a new "Privacy Pass Parameters" page to list identifiers for issuance protocols defined for use with the Privacy Pass token authentication scheme. These identifiers are two-byte values, so the maximum possible value is 0xFFFF = 65535.

Template:

* Value: The two-byte identifier for the algorithm

* Name: Name of the issuance protocol

*Publicly Verifiable: A Y/N value indicating if the output tokens are publicly verifiable

*Public Metadata: A Y/N value indicating if the output tokens can contain public metadata.

*Private Metadata: A Y/N value indicating if the output tokens can contain private metadata.

*Nk: The length in bytes of an output authenticator

*Nid: The length of the token key identifier

*Reference: Where this algorithm is defined

New entries in this registry are subject to the Specification Required registration policy ([RFC8126], Section 4.6). Designated experts need to ensure that the token type is sufficiently clearly defined to be used for both token issuance and redemption, and meets the common security and privacy requirements for issuance protocols defined in Section 3.2 of [ARCHITECTURE].

This registry also will also allow provisional registrations to allow for experimentation with protocols being developed. Designated experts review, approve, and revoke provisional registrations.

This document defines several Reserved values, which can be used by clients and servers to send "greased" values in token challenges and responses to ensure that implementations remain able to handle unknown token types gracefully (this technique is inspired by [RFC8701]). Implemenations SHOULD select reserved values at random when including them in greased messages. Servers can include these in TokenChallenge structures, either as the only challenge when no real token type is desired, or as one challenge in a list of challenges that include real values. Clients can include these in Token structures when they are not able to present a real token response. The contents of the Token structure SHOULD be filled with random bytes when using greased values.

The initial contents for this registry are defined in the table below.

Value	Name	Publicly Verifiable	Public Metadata	Private Metadata	Nk	Nid	Reference
0x0000	RESERVED	N/A	N/A	N/A	N/A	N/A	This document
0x02AA	RESERVED	N/A	N/A	N/A	N/A	N/A	This document
0x1132	RESERVED	N/A	N/A	N/A		N/A	

Value	Name	Publicly Verifiable	Public Metadata	Private Metadata	Nk	Nid	Reference
					N/A		This document
0x2E96	RESERVED	N/A	N/A	N/A	N/A	N/A	This document
0x3CD3	RESERVED	N/A	N/A	N/A	N/A	N/A	This document
0x4473	RESERVED	N/A	N/A	N/A	N/A	N/A	This document
0x5A63	RESERVED	N/A	N/A	N/A	N/A	N/A	This document
0x6D32	RESERVED	N/A	N/A	N/A	N/A	N/A	This document
0x7F3F	RESERVED	N/A	N/A	N/A	N/A	N/A	This document
0x8D07	RESERVED	N/A	N/A	N/A	N/A	N/A	This document
0x916B	RESERVED	N/A	N/A	N/A	N/A	N/A	This document
0xA6A4	RESERVED	N/A	N/A	N/A	N/A	N/A	This document
0xBEAB	RESERVED	N/A	N/A	N/A	N/A	N/A	This document
0xC3F3	RESERVED	N/A	N/A	N/A	N/A	N/A	This document
0xDA42	RESERVED	N/A	N/A	N/A	N/A	N/A	This document
0xE944	RESERVED	N/A	N/A	N/A	N/A	N/A	This document
0xF057	RESERVED	N/A	N/A	N/A	N/A	N/A	This document

Table 1: Token Types

6. References

6.1. Normative References

[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>.

[RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, DOI 10.17487/RFC4648, October 2006, <https://www.rfc-editor.org/rfc/rfc4648>.

[RFC7235] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Authentication", RFC 7235, DOI 10.17487/RFC7235, June 2014, <https://www.rfc-editor.org/rfc/rfc7235>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/rfc/rfc8126>.

[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>.

6.2. Informative References

[ARCHITECTURE] Davidson, A., Iyengar, J., and C. A. Wood, "Privacy Pass Architectural Framework", Work in Progress, Internet-Draft, draft-ietf-privacypass-architecture-06, 5 August 2022, <https://datatracker.ietf.org/doc/html/draft-ietf-privacypass-architecture-06>.

[COOKIES] Chen, L., Englehardt, S., West, M., and J. Wilander, "Cookies: HTTP State Management Mechanism", Work in Progress, Internet-Draft, draft-ietf-httpbis-rfc6265bis-10, 24 April 2022, <https://datatracker.ietf.org/doc/html/draft-ietf-httpbis-rfc6265bis-10>.

[I-D.ietf-privacypass-architecture] Davidson, A., Iyengar, J., and C. A. Wood, "Privacy Pass Architectural Framework", Work in Progress, Internet-Draft, draft-ietf-privacypass-architecture-06, 5 August 2022, <https://datatracker.ietf.org/doc/html/draft-ietf-privacypass-architecture-06>.

[ISSUANCE] Celi, S., Davidson, A., Faz-Hernandez, A., Valdez, S., and C. A. Wood, "Privacy Pass Issuance Protocol", Work in Progress, Internet-Draft, draft-ietf-privacypass-protocol-06, 6 July 2022, <https://datatracker.ietf.org/doc/html/draft-ietf-privacypass-protocol-06>.

[RFC8701] Benjamin, D., "Applying Generate Random Extensions And Sustain Extensibility (GREASE) to TLS Extensibility", RFC 8701, DOI 10.17487/RFC8701, January 2020, <https://www.rfc-editor.org/rfc/rfc8701>.

Appendix A. Test Vectors

This section includes test vectors for the challenge and redemption functionalities described in Section 2.1 and Section 2.2. Each test vector lists the following values:

* token_type: The type of token issuance protocol, a value from Section 5.2. For these test vectors, token_type is 0x0002, corresponding to the issuance protocol in [ISSUANCE].

* issuer_name: The name of the issuer in the TokenChallenge structure, represented as a hexadecimal string.

* redemption_context: The redemption context in the TokenChallenge structure, represented as a hexadecimal string.

* origin_info: The origin info in the TokenChallenge structure, represented as a hexadecimal string.

* nonce: The nonce in the Token structure, represented as a hexadecimal string.

* token_key: The public token-key, encoded based on the corresponding token type, represented as a hexadecimal string.

* token_authenticator_input: The values in the Token structure used to compute the Token authenticator value, represented as a hexadecimal string.

* token_authenticator: The output Token authenticator which verifies under token_key, represented as a hexadecimal string.

Test vectors are provided for each of the following TokenChallenge configurations:

* TokenChallenge with a single origin and non-empty redemption context

* TokenChallenge with a single origin and empty redemption context

* TokenChallenge with an empty origin and redemption context

* TokenChallenge with an empty origin and redemption context

* TokenChallenge with an empty origin and non-empty redemption context

* TokenChallenge with a multiple origins and non-empty redemption context

These test vectors are below.

```
token_type: 2
issuer_name: 6973737565722e6578616d706c65
redemption_context:
40ff3cdc296a1e823f43b49355df1a2ee4c5f65e5d38ebb3e24ecf4d874997c6
origin_info: 6f726967696e2e6578616d706c65
nonce: 4437fb872eab95b5831a5d01005ee2995e417862ecfd2079ee4c246859a060ae
token_key: 30820252303d06092a864886f70d01010a3030a00d300b060960864801650
3040202a11a301806092a864886f70d010108300b0609608648016503040202a20302013
00382020f003082020a0282020100d730ce8b3ec7336b48a4f5897564d87c87627298f21
ba4bf34e7931142875c0e52c5aef3222d67e86124403e436d0136ebd806de37730427f81
4f7f0485eace93015471d14e56f3824e8bc5fbe44cf67e241c7642ac3a39452a283ff806
84ddbd66929a371d01e50feef1faee7f63f3ceb4b5ceacb939e06a558c2a6bccfd96fb74
16d3edce151bc7b0a6582f0ce99a7c0e7d5793b13d41292105e510e1aa00e082975a1386
6dfaf3a0a51c0dd1ecb64cc55cc607ca1813b5f91fd8e9cb9db18ffd81ac985a6cfdd5cc
2a0b8a5e4e9fa1ea5f149c1662155bb071c95218cae9ae4af613351baf470b1597bb984c
5ea8326f98aff64f72b60bcd035f6b970eb6edd2f9f2180d5aa8a17ed400056af3faa520
4b73c89b4eada6a057dd3dda9d8e18b3a6d2347c1027e2711f21eb7d96fef50cc3dacb2f
5ccc36e4c138ab75953974ade74982f85b91f419654d390378e2ea5aae33f1b4acf534d0
6de2f114acfdd88d6d708f4d2b646a8112b0fe181489916e2ba5c634cdf9b95762d1e120
169482dd27f959132705079fc4a00eee1f353a81c1e810ade20d070d839277169e09150c
08605afe7cea2aec41d2f85c2af7bef5d577343b4385e2c6c159926c1c8267d00433b88b
ad314a5ddcef58936126f1dd8da7b5728da192f54b304e60f4088e5b0620404f82a5939d
975e6714453a533c172c8a9b4b5da976ea60a5aa91fef0203010001
token_authenticator_input: 00024437fb872eab95b5831a5d01005ee2995e417862e
cfd2079ee4c246859a060ae055038620bd58190f057b86af2883352fd9ec612487979b00
74a489aece337e79f9293b4d62e4b4759af064df8fa5759c79ab51a00f692541b26d466d
ab48091
token_authenticator: 9c2fc25cb429a7cfe6e21193b6122ffe18c2c09c1df10dfea3d
155c297ce3f4132d273bf2ad490c41e592219bb253378c21215657905fe713aca31f6ab7
1206c1c872210c71d53a8d9b3ee635cf22c47d518454f9f5a898218ed7aae78414e9d85f
4a62244babdb63accc1257f1f1824493549465a3c63d69e9e307a328121402022a4f1aea
a7e6ff46edf4884b5ab47531b8c949a225a9f5a9c1b7608af0641c2070533402683e6f86
d547b6083d6cfa2a985f4673bf46ae09864d31613acd5a7d61dae7a29133e37093baabe1
20e59714d662162324406403c1fb44312b03c509202041a44dc351ea3659d446ea024e96
23b522171d9af0c8ac81f8a6a0018cb049bda21d12c9783ae6f7057144f8d26699d20f19
023269256ff607ca1ab37b0d95704aa0299e64b3b823c148ff8c46e835c7060dbb3c247e
6034892b3f6b7401fb67366e8afe8267182d6f36bf3618712371e6ae4654c0897f7475d3
9e9c186189162e29a9d8b3e37860457843a1c2fa3cacc133bdb8c7f77aae0ea3a5649300
35a7689f3a24ea726d4506d19f0b1aedb8739cb3fe7177cfaed08c8902162ed530ef19a0
266ca61a1a0b1bfc329fd2d1c1ad307a32f531f5be6faf75d96a49020acca8e37a4cb55f
f4072916711c397daf5bcbd229132b47d10b16f646d1d675cdf58c6f057333b1cbb94b2d
c44320cd2e9cba6f1c33a708ae1bb97f0739a

token_type: 2
issuer_name: 6973737565722e6578616d706c65
redemption_context:
origin_info: 6f726967696e2e6578616d706c65
nonce: 4437fb872eab95b5831a5d01005ee2995e417862ecfd2079ee4c246859a060ae
token_key: 30820252303d06092a864886f70d01010a3030a00d300b060960864801650
```

3040202a11a301806092a864886f70d010108300b0609608648016503040202a20302013
00382020f003082020a0282020100d730ce8b3ec7336b48a4f5897564d87c87627298f21
ba4bf34e7931142875c0e52c5aef3222d67e86124403e436d0136ebd806de37730427f81
4f7f0485eace93015471d14e56f3824e8bc5fbe44cf67e241c7642ac3a39452a283ff806
84ddbd66929a371d01e50feef1faee7f63f3ceb4b5ceacb939e06a558c2a6bccfd96fb74
16d3edce151bc7b0a6582f0ce99a7c0e7d5793b13d41292105e510e1aa00e082975a1386
6dfaf3a0a51c0dd1ecb64cc55cc607ca1813b5f91fd8e9cb9db18ffd81ac985a6cfdd5cc
2a0b8a5e4e9fa1ea5f149c1662155bb071c95218cae9ae4af613351baf470b1597bb984c
5ea8326f98aff64f72b60bcd035f6b970eb6edd2f9f2180d5aa8a17ed400056af3faa520
4b73c89b4eada6a057dd3dda9d8e18b3a6d2347c1027e2711f21eb7d96fef50cc3dacb2f
5ccc36e4c138ab75953974ade74982f85b91f419654d390378e2ea5aae33f1b4acf534d0
6de2f114acfdd88d6d708f4d2b646a8112b0fe181489916e2ba5c634cdf9b95762d1e120
169482dd27f959132705079fc4a00eee1f353a81c1e810ade20d070d839277169e09150c
08605afe7cea2aec41d2f85c2af7bef5d577343b4385e2c6c159926c1c8267d00433b88b
ad314a5ddcef58936126f1dd8da7b5728da192f54b304e60f4088e5b0620404f82a5939d
975e6714453a533c172c8a9b4b5da976ea60a5aa91fef0203010001
token_authenticator_input: 00024437fb872eab95b5831a5d01005ee2995e417862e
cfd2079ee4c246859a060ae11e15c91a7c2ad02abd66645802373db1d823bea80f08d452
541fb2b62b5898b9f9293b4d62e4b4759af064df8fa5759c79ab51a00f692541b26d466d
ab48091
token_authenticator: 4be4655a33566de7409e7cfdcdb764c251c04138602a046a7d7
1540aa9bcdb34e7df5dfabe17e16a3569f67c36460a79e9e7278b454c4f505580ae99750
b9308022b20aa8807ee054881126d9a4afd134331d0ebb3f9a4f2948731cd0ad2fe468b1
f8c6fcf2d5b9a2991a684b3fb0dec6a3e32fd3335e546f58c2217736683d378076355727
0493eb0f607c7936633a0532d1828dc860f0bf3954c82f0e1ab6089eae92d9d97e3237f2
58c5c82e711bef1682deb6edd19b24bb543f4418825e5d41e126de82f1a4b63321f07c12
3029a7499356fa8bdc11982451c69fa3d1940a4781b646c99e33e83b95810dc1e7a32a25
953ba0a0e37917c9f85bb4f0e7687c826e7138c9a2e71e87644b36c3891b4fec6af02519
aafaa36d559c71d090ea081ceed6cb738ffb730fa5dcbe889362591eb0a89f8ba4057f09
3ca35cc684f3b4cdc7c177f275a9d74a75e98eb395689842a5626d61af84072c9eb858ea
ed7e467b570c771e6530e02dda20b47f6860bb341ca4f849168dc7b6cb8c6743b8113c3e
f09ce9b9b2eaf172204cc26ea7c3de962cc1a851195cf143c9f27cb8f1df219df1209097
5ba657de1d021f2044829a689decf1072e5f79fbef0d3ae6085532f81b32b2a47968b073
2928193894dabb521c3f4c4a6858d43c8abd4695db4e49d3b5d46059032819d8485b0ccb
c3d24c5c72ac3e44d9ff94946cb8f8ca69fb1

token_type: 2
issuer_name: 6973737565722e6578616d706c65
redemption_context:
origin_info:
nonce: 4437fb872eab95b5831a5d01005ee2995e417862ecfd2079ee4c246859a060ae
token_key: 30820252303d06092a864886f70d01010a3030a00d300b0609608648016503
3040202a11a301806092a864886f70d010108300b0609608648016503040202a20302013
00382020f003082020a0282020100d730ce8b3ec7336b48a4f5897564d87c87627298f21
ba4bf34e7931142875c0e52c5aef3222d67e86124403e436d0136ebd806de37730427f81
4f7f0485eace93015471d14e56f3824e8bc5fbe44cf67e241c7642ac3a39452a283ff806
84ddbd66929a371d01e50feef1faee7f63f3ceb4b5ceacb939e06a558c2a6bccfd96fb74
16d3edce151bc7b0a6582f0ce99a7c0e7d5793b13d41292105e510e1aa00e082975a1386
6dfaf3a0a51c0dd1ecb64cc55cc607ca1813b5f91fd8e9cb9db18ffd81ac985a6cfdd5cc

2a0b8a5e4e9fa1ea5f149c1662155bb071c95218cae9ae4af613351baf470b1597bb984c
5ea8326f98aff64f72b60bcd035f6b970eb6edd2f9f2180d5aa8a17ed400056af3faa520
4b73c89b4eada6a057dd3dda9d8e18b3a6d2347c1027e2711f21eb7d96fef50cc3dacb2f
5ccc36e4c138ab75953974ade74982f85b91f419654d390378e2ea5aae33f1b4acf534d0
6de2f114acfdd88d6d708f4d2b646a8112b0fe181489916e2ba5c634cdf9b95762d1e120
169482dd27f959132705079fc4a00eee1f353a81c1e810ade20d070d839277169e09150c
08605afe7cea2aec41d2f85c2af7bef5d577343b4385e2c6c159926c1c8267d00433b88b
ad314a5ddcef58936126f1dd8da7b5728da192f54b304e60f4088e5b0620404f82a5939d
975e6714453a533c172c8a9b4b5da976ea60a5aa91fef0203010001
token_authenticator_input: 00024437fb872eab95b5831a5d01005ee2995e417862e
cfd2079ee4c246859a060aeb741ec1b6fd05f1e95f8982906aec1612896d9ca97d53eef9
4ad3c9fe023f7a49f9293b4d62e4b4759af064df8fa5759c79ab51a00f692541b26d466d
ab48091
token_authenticator: 31c2ae70c45f171ed822a9397ba844d6ee20d09323491f4f9fb
3db54d7d3c7b403fd8ee1e2eedebd693d2493b3b1973142cd85f54257c009edda7cd5ad5
3cf8a07d8a3252c62da14d688d225727faa294b5ed57bd0913482c845b502fd967c27b92
d7c4ee7566894134fc71999e55073bf9d19f95b10f0d2044bef815dccfa7632903af7fd2
09af17c008c93fe76e6c4dffd90de933d711366ee72adc32d1289205a306de9b15bb6639
9b2e89c7cb129eadf062be9c4fd54b1ffea79840d0451544f30cc4eab6c36a06ad6dac87
741059803aa57006ce5aea4e71e053f4901f9901dd1f9fa489763f1c499fdf8ec1903a31
79259c79f7a496eefbe937f5b6d69f17f2b96b184cfab98dc0e46b2b0f5fb57764f894bf
2025d5f26505d70d3fa8766406d246bc037d588f035ea7230969323cb237da949a87db95
854f09ba24363a608d0a56427fac57907204aa8c57dc29633a36a83cff385f1eefcfffc9
730eda756d80109a20394c21b40ddc3e0121bd08e4a1eae48daa3a3c7a78f682ee208a78
686960c270e0ffd0042f38e9f786276ef01f7bda5ee323692c8de4f590014c4f4ea1f583
bf3db5cee7ba39d612c73535ef488c796ea33d2f9049a3b34cbc7db3d58208a11ceaf1b0
ab7853b817f53a3ff470bd3e353ca9d4365de09b6a70d94ee4e9118a0901013026360e99
64b2d6c51d47d4307328cedff3561d65a5583

token_type: 2
issuer_name: 6973737565722e6578616d706c65
redemption_context:
40ff3cdc296a1e823f43b49355df1a2ee4c5f65e5d38ebb3e24ecf4d874997c6
origin_info:
6f726967696e2e6578616d706c652c6f726967696e322e6578616d706c65
nonce: 4437fb872eab95b5831a5d01005ee2995e417862ecfd2079ee4c246859a060ae
token_key: 30820252303d06092a864886f70d01010a3030a00d300b0609608648016503040202a11a301806092a864886f70d010108300b0609608648016503040202a2030201300382020f003082020a0282020100d730ce8b3ec7336b48a4f5897564d87c87627298f21
3040202a11a301806092a864886f70d010108300b0609608648016503040202a20302013
00382020f003082020a0282020100d730ce8b3ec7336b48a4f5897564d87c87627298f21
ba4bf34e7931142875c0e52c5aef3222d67e86124403e436d0136ebd806de37730427f81
4f7f0485eace93015471d14e56f3824e8bc5fbe44cf67e241c7642ac3a39452a283ff806
84ddbd66929a371d01e50feef1faee7f63f3ceb4b5ceacb939e06a558c2a6bccfd96fb74
16d3edce151bc7b0a6582f0ce99a7c0e7d5793b13d41292105e510e1aa00e082975a1386
6dfaf3a0a51c0dd1ecb64cc55cc607ca1813b5f91fd8e9cb9db18ffd81ac985a6cfdd5cc
2a0b8a5e4e9fa1ea5f149c1662155bb071c95218cae9ae4af613351baf470b1597bb984c
5ea8326f98aff64f72b60bcd035f6b970eb6edd2f9f2180d5aa8a17ed400056af3faa520
4b73c89b4eada6a057dd3dda9d8e18b3a6d2347c1027e2711f21eb7d96fef50cc3dacb2f
5ccc36e4c138ab75953974ade74982f85b91f419654d390378e2ea5aae33f1b4acf534d0
6de2f114acfdd88d6d708f4d2b646a8112b0fe181489916e2ba5c634cdf9b95762d1e120

169482dd27f959132705079fc4a00eee1f353a81c1e810ade20d070d839277169e09150c
08605afe7cea2aec41d2f85c2af7bef5d577343b4385e2c6c159926c1c8267d00433b88b
ad314a5ddcef58936126f1dd8da7b5728da192f54b304e60f4088e5b0620404f82a5939d
975e6714453a533c172c8a9b4b5da976ea60a5aa91fef0203010001
token_authenticator_input: 00024437fb872eab95b5831a5d01005ee2995e417862e
cfd2079ee4c246859a060ae175a86e01410befaee0307ec86990b8a6e1b8192dfca7f0a9
692e06813ec9d199f9293b4d62e4b4759af064df8fa5759c79ab51a00f692541b26d466d
ab48091
token_authenticator: 6b25498e0c809b8c83ec22f6d46a98cd866354ad56b7aa78ef3
fc237ebe9b7fc5ea3346257ddf085166931653d94016e37896df38a767c438d4b2ca440c
47ae62f5e9074f9a72b06bdf103b4f205b20075d07465750a06e463106bb1ff0f9393f00
1ac5377b9ba7edc28b88aeab4e0d4ca9bd101af13967a4d681be02f8f3308224c0115ef8
7ed890e04441486db438538a3a8a82050377d5882001689216c82bb74513ecab47eebaa4
17f030ee887a83f0cc805becc5dd417a3c1c79f828d28068872b0102e8b2fd21743d4aea
4d4bbe8193b496d4167c312c1bea39c5c337701b33f07706daf2e07d8e0ad178878604b3
adcd766ea93e70a70a0b7d1447d20a6af222d8a785db417f462639e89aa57fd664c1c93d
b0b75dc2189345ab83aa823a7b1eb7c3f965fca97780e46d019044dc9583fc3a4e13705c
7c97594c4e983c975f2ac6e5a3a3fe46bc811e9bb46e8f1d2997152d19eff15e2e238f7b
541413f05141d31154e4a59c4b552192ef1d39b0399c5a9b935d4133f4d4e2b3737e7f45
8f92a90719cf5620b05884a74a16db6dbc48b7e819543290cef76d0e761dff156a6800a7
4430a79cba2cc46178ab72b169222fb082f9e05874ccaf0734e2b24315fb2c429c0b1b42
dc6513d76b891b7ce1c9c819303a050c9251aaa8ea2bb61a3bbbfc770ccad6a53dfd29d9
a65f81e91de499d752b29a43294f0cdaf361a

Authors' Addresses

Tommy Pauly
Apple Inc.
One Apple Park Way
Cupertino, California 95014,
United States of America

Email: tpauly@apple.com

Steven Valdez
Google LLC

Email: svaldez@chromium.org

Christopher A. Wood
Cloudflare

Email: caw@heapingbits.net