Workgroup: Network Working Group
Internet-Draft:
draft-ietf-privacypass-architecture-04
Published: 1 July 2022
Intended Status: Informational
Expires: 2 January 2023
Authors: A. Davidson J. Iyengar C. A. Wood
LIP Fastly Cloudflare

Privacy Pass Architectural Framework

Abstract

This document specifies the architectural framework for constructing secure and anonymity-preserving instantiations of the Privacy Pass protocol. It provides recommendations on how the protocol ecosystem should be constructed to ensure the privacy of clients, and the security of all participating entities.

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 2 January 2023.

Table of Contents

1. Introduction

Privacy Pass is a protocol for authorization based on anonymous-credential authentication mechanisms. Typical approaches for authorizing clients, such as through the use of long-term cookies, are not privacy-friendly since they allow servers to track clients across sessions and interactions. Privacy Pass takes a different approach: instead of presenting linkable state carrying information to servers, e.g., whether or not the client is an authorized user or has completed some prior challenge, clients present unlinkable proofs that attest to this information.

The most basic Privacy Pass protocol provides a set of cross-origin authorization tokens that protect the client's anonymity during interactions with a server. This allows clients to communicate an attestation of a previously authenticated server action, without having to reauthenticate manually. The tokens retain anonymity in the sense that the act of revealing them cannot be linked back to the session where they were initially issued.

At a high level, Privacy Pass is composed of two protocols: issuance and redemption.

The issuance protocol runs between a Client and two network functions in the Privacy Pass architecture: Attestation and Issuance. These two network functions can be implemented by the same protocol participant, but can also be implemented separately. The Issuer is responsible for issuing tokens in response to requests from Clients. The Attester is responsible for attesting properties about the Client for which tokens are issued. The Issuer needs to be trusted by the server that later redeems the token. Attestation can be performed by the Issuer or by an Attester that is trusted by the Issuer. Clients might prefer to select different Attesters, separate from the Issuer, to be able to use preferred authentication methods or improve privacy by not directly communicating with an Issuer. Depending on the attestation, Attesters can store state about a Client, such as the number of overall tokens issued thus far. As an example of an Issuance protocol, in the original Privacy Pass protocol [PPSRV], tokens were only issued to Clients that solved CAPTCHAs. In this context, the Attester attested that some client solved a CAPTCHA and the resulting token produced by the Issuer was proof of this fact.

The redemption protocol runs between Client and Origin (server). It allows Origins to challenge Clients to present one or more tokens for authorization. Depending on the type of token, e.g., whether or not it is cross-origin or per-origin, and whether or not it can be cached, the Client either presents a previously obtained token or invokes the issuance protocol to acquire one for authorization.

The issuance and redemption protocols operate in concert as shown in the figure below.

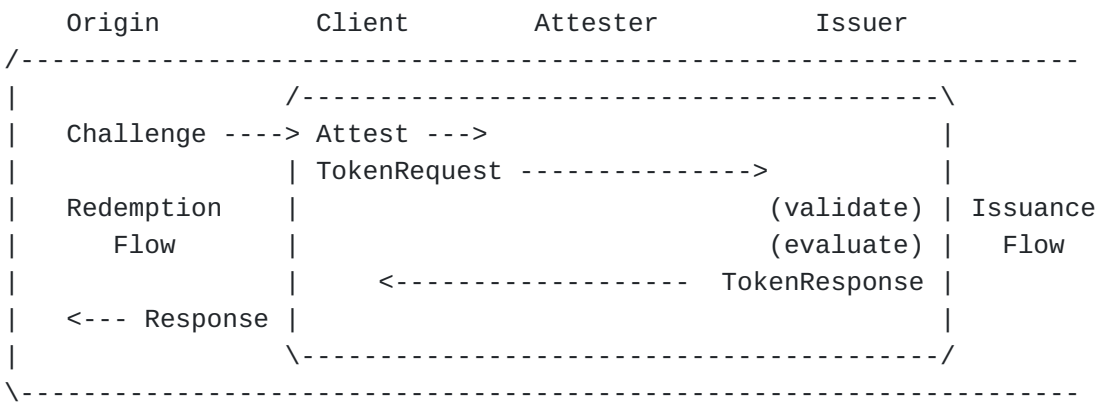

Figure 1: Privacy Pass Architectural Components

This document describes requirements for both issuance and redemption protocols. This document also describes ecosystem considerations that impact the stated privacy and security guarantees of the protocol.

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.

The following terms are used throughout this document.

- Client: An entity that seeks authorization to an Origin.

- Origin: An entity that challenges Clients for tokens.

- Issuer: An entity that issues tokens to Clients for properties attested to by the Attester.

- Attester: An entity that attests to properties of Client for the purposes of token issuance.

3. Architecture

The Privacy Pass architecture consists of four logical entities -- Client, Origin, Issuer, and Attester -- that work in concert as shown in Section 1 for token issuance and redemption. This section describes the purpose of token issuance and redemption and the requirements therein on the relevant participants.

3.1. Redemption Protocol

The redemption protocol is a simple challenge-response based authorization protocol between Client and Origin. Origins prompt Clients with a token challenge and, if possible, Clients present a valid token for the challenge in response. The context in which an Origin challenges a Client for a token is referred to as the redemption context. This context includes all information associated with the redemption event, such as the timestamp of the event, Client visible information (including the IP address), and the Origin name.

The challenge controls the type of token that the Origin will accept for the given resource. As described in [HTTP-Authentication], there are a number of ways in which the token may vary, including:

- Issuance protocol. The token identifies the type of issuance protocol required for producing the token. Different issuance protocols have different security properties, e.g., some issuance protocols may produce tokens that are publicly verifiable, whereas others may not have this property.

- Issuer identity. Tokens identify which issuers are trusted for a given issuance protocol.

- Redemption context. Tokens can be bound to a given redemption context, which influences a client's ability to pre-fetch and cache tokens. For example, an empty redemption context always allows tokens to be issued and redeemed non-interactively, whereas a fresh and random redemption context means that the redeemed token must be issued only after the client receives the challenge. See Section 2.1.1 of [HTTP-Authentication] for more details.

- Per-origin or cross-origin. Tokens can be constrained to the Origin for which the challenge originated, or can be used across Origins.

Depending on the use case, Origins may need to maintain state to track redeemed tokens. For example, Origins that accept cross-origin across shared redemption contexts tokens SHOULD track which tokens have been redeemed already in those redemption contexts, since these tokens can be issued and then spent multiple times in response to any such challenge. See Section 2.1.1 of [HTTP-Authentication] for discussion.

Origins that admit cross-origin tokens bear some risk of allowing tokens issued for one Origin to be spent in an interaction with another Origin. If tokens protected with resources are unique to a single Origin, then said Origin MUST NOT admit cross-origin tokens for authorization.

3.2. Issuance Protocol

The issuance protocol embodies the core of Privacy Pass. It takes as input a challenge from the redemption protocol and produces a token, as shown in the figure below.

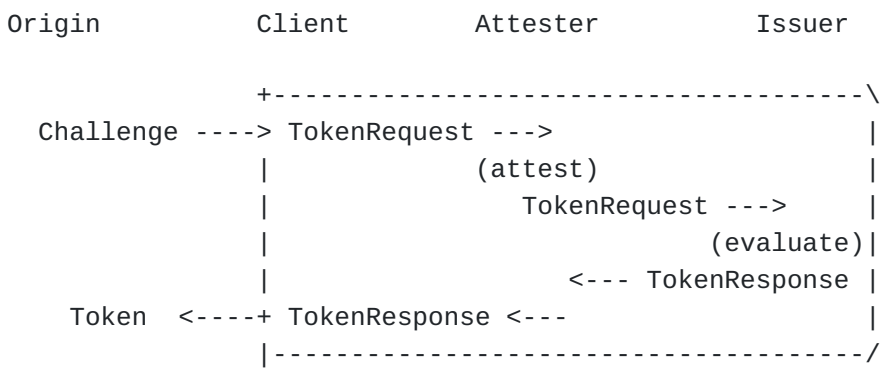

Figure 2: Issuance Overview

Clients interact with the Attester and Issuer to produce a token in response to a challenge. The context in which an Attester vouches for a Client during issuance is referred to as the attestation context. This context includes all information associated with the issuance event, such as the timestamp of the event and Client visible information, including the IP address or other information specific to the type of attestation done.

Each issuance protocol may be different, e.g., in the number and types of participants, underlying cryptographic constructions used when issuing tokens, and even privacy properties.

Clients initiate the Token issuance protocol using the challenge, a randomly generated nonce, and public key for the Issuer. The Token issuance protocol itself can be any interactive protocol between Client, Issuer, or other parties that produces a valid authenticator over the Client's input, subject to the following security requirements.

1. Unconditional input secrecy. The issuance protocol MUST NOT reveal anything about the Client's private input, including the challenge and nonce, to the Attester or Issuer. The issuance protocol can reveal the Issuer public key for the purposes of determining which private key to use in producing the token. A result of this property is that the redemption flow is unlinkable from the issuance flow.

2. One-more forgery security. The issuance protocol MUST NOT allow malicious Clients or Attesters (acting as Clients) to forge tokens offline or otherwise without interacting with the Issuer directly.

3. Concurrent security. The issuance protocol MUST be safe to run concurrently with arbitrarily many Clients.

Each Issuance protocol MUST come with a detailed analysis of the privacy impacts of the protocol, why these impacts are justified,

and guidelines on how to deploy the protocol to minimize any privacy impacts.

Clients obtain the Issuer public key directly from the Origin using the process described in [HTTP-Authentication]. Clients MAY apply some form of key consistency check to determine if this public key is consistency and correct for the specified Issuer. See [CONSISTENCY] for example mechanisms. Depending on the deployment, the Attester may be able to assist the Client in applying these consistency checks across clients.

Depending on the use case, issuance may require some form of Client anonymization service similar to an IP-hiding proxy so that Issuers cannot learn information about Clients. This can be provided by an explicit participant in the issuance protocol, or it can be provided via external means, e.g., through the use of an IP-hiding proxy service like Tor. In general, Clients SHOULD minimize or remove identifying information where possible when invoking the issuance protocol.

Issuers MUST NOT issue tokens for Clients through untrusted Attesters. This is important because the Attester's role is to vouch for trust in privacy-sensitive Client information, such as account identifiers or IP address information, to the Issuer. Tokens produced by an Issuer that admits issuance for any type of attestation cannot be relied on for any specific property. See Section 3.2.1 for more details.

3.2.1. Attester Role

Attestation is an important part of the issuance protocol. Attestation is the process by which an Attester bears witness to, confirms, or authenticates a Client so as to verify a property about the Client that is required for Issuance. Examples of attestation properties include, though are not limited to:

- Capable of solving a CAPTCHA. Clients that solve CAPTCHA challenges can attest to this capability for the purposes of being ruled out as a bot or otherwise automated Client.

- Client state. Clients can be associated with state and the attester can attest to this state. Examples of state include the number of issuance protocol invocations, the client's geographic region, and whether the client has a valid application-layer account.

- Trusted device. Some Clients run on trusted hardware that are capable of producing device-level attestation statements.

Each of these attestation types have different security properties. For example, attesting to having a valid account is different from attesting to be running on trusted hardware. In general, Attesters should accept a limited form of attestation formats.

Each attestation format also has an impact on the overall system privacy. For example, the number of users in possession of a single class of trusted device might be lesser than the number of users that can solve CAPTCHAs. Similarly, requiring a conjunction of attestation types could decrease the overall anonymity set size. For example, the number of Clients that have solved a CAPTCHA in the past day, have a valid account, and are running on a trusted device is lesser than the number of Clients that have solved a CAPTCHA in the past day. Attesters should not admit attestation types that result in small anonymity sets.

3.2.2. Issuer Role

Issuers MUST be uniquely identifiable by all Clients with a consistent identifier. In a web context, this identifier might be the Issuer host name. As discussed later in Section 5, ecosystems that admit a large number of Issuers can lead to privacy concerns for the Clients in the ecosystem. Therefore, in practice, the number of Issuers should be bounded. The actual Issuers can be replaced with different Issuers as long as the total never exceeds these bounds. Moreover, Issuer replacements also have an effect on client anonymity that is similar to when a key rotation occurs. See Section 5 for more details about maintaining privacy with multiple Issuers.

3.2.2.1. Key Management

To facilitate issuance, the Issuer MUST hold an Issuance key pair at any given time. The Issuer public key MUST be made available to all Clients in such a way that key rotations and other updates are publicly visible. The key material and protocol configuration that an Issuer uses to produce tokens corresponds to a number of different pieces of information.

- The issuance protocol in use; and

- The public keys that are active for the Issuer.

The way that the Issuer publishes and maintains this information impacts the effective privacy of the clients; see Section 5 for more details. The fundamental requirement for key management and discovery is that Issuers must be unable to target specific clients

with unique keys without detection. There are a number of ways in which this might be implemented:

- Servers use a verifiable, tamper-free registry from which clients discover keys. Similar to related mechanisms and protocols such as Certificate Transparency [RFC6962], this may require external auditors or additional client behavior to ensure the registry state is consistent for all clients.

- Clients use an anonymity-preserving tool such as Tor to discover keys from multiple network vantage points. This is done to ensure consistent keys to seemingly different clients.

- Clients embed Issuer keys into software.

As above, specific mechanisms for key management and discovery are out of scope for this document.

3.2.2.2. Key Rotation

Token issuance associates all issued tokens with a particular choice of key. If an Issuer issues tokens with many keys, then this may harm the anonymity of the Client. For example, they would be able to map the Client's access patterns by inspecting which key each token they possess has been issued under.

To prevent against this, Issuers MUST only use one private key for issuing tokens at any given time. Servers MAY use one or more keys for redemption to allow Issuers for seamless key rotation.

Servers may rotate keys as a means of revoking tokens issued under old or otherwise expired keys. Alternatively, Issuers may include expiration information as metadata alongside the token; See Section 3.2.3 for more discussion about metadata constraints. Both techniques are equivalent since they cryptographically bind expiration to individual tokens. Key rotations should be limited in frequency for similar reasons.

3.2.3. Metadata

Certain instantiations of the issuance protocol may permit public or private metadata to be cryptographically bound to a token. As an example, one trivial way to include public metadata is to assign a unique issuer public key for each value of metadata, such that N keys yields log2(N) bits of metadata. The total amount of metadata bits included in a token is the sum of public and private metadata bits.

Public metadata is that which clients can observe as part of the token issuance flow. Public metadata can either be transparent or

opaque. For example, transparent public metadata is a value that the client either generates itself, or the Issuer provides during the issuance flow and the client can check for correctness. Opaque public metadata is metadata the client can see but cannot check for correctness. As an example, the opaque public metadata might be a "fraud detection signal", computed on behalf of the Issuer, during token issuance. In normal circumstances, clients cannot determine if this value is correct or otherwise a tracking vector.

Private metadata is that which clients cannot observe as part of the token issuance flow. Such instantiations may be built on the Private Metadata Bit construction from Kreuter et al. [KLOR20] or the attribute-based VOPRF from Huang et al. [HIJK21].

Metadata may also be arbitrarily long or bounded in length. The amount of permitted metadata may be determined by application or by the underlying cryptographic protocol.

3.2.4. Issuance Protocol Extensibility

The Privacy Pass protocol and ecosystem are both intended to be receptive to extensions that expand the current set of functionalities through new issuance protocols. Each issuance protocol SHOULD come with a detailed analysis of the privacy impacts of the extension, why these impacts are justified, and guidelines on how to deploy the protocol to minimize any privacy impacts. Any extension to the Privacy Pass protocol MUST adhere to the guidelines specified in Section 3.2.2 for managing Issuer public key data.

4. Deployment Considerations

Client uses Privacy Pass to separate attestation context and redemption context. Linking or combining these contexts can reveal sensitive information about the Client, including their identity or browsing history. Depending on the deployment model, separating these contexts can take different forms. The Origin, Attester, and Issuer portrayed in Figure 1 can be instantiated and deployed in a number of different ways. This section covers some expected deployment models and their corresponding security and privacy considerations. The discussion below assumes non-collusion between entities when operated by separate parties. Mechanisms for enforcing non-collusion are out of scope for this architecture.

4.1. Shared Origin, Attester, Issuer

In this model, the Origin, Attester, and Issuer are all operated by the same entity, as shown in the figure below.

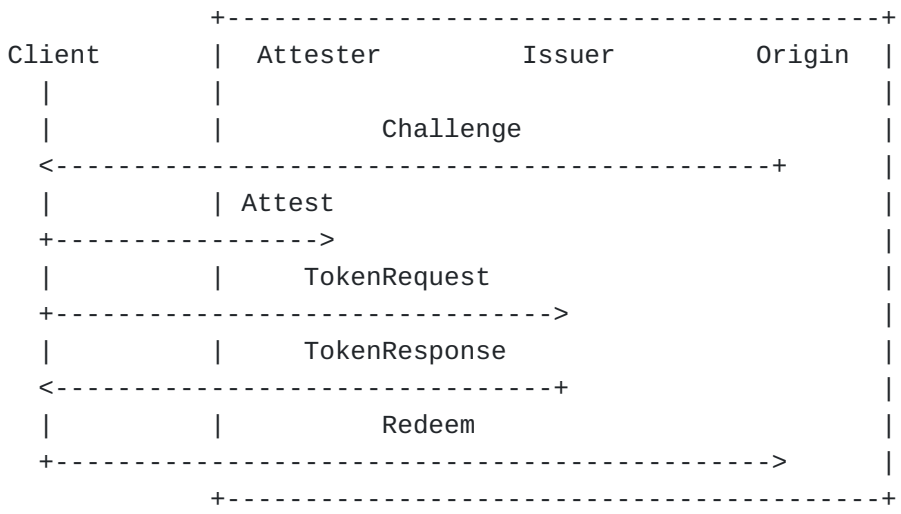

Figure 3: Shared Deployment Model

This model represents the initial deployment of Privacy Pass, as described in [PPSRV]. In this model, the Attester, Issuer, and Origin share the attestation and redemption contexts. As a result, attestation mechanisms that can uniquely identify a Client, e.g., requiring that Clients authenticate with some type of application-layer account, are not appropriate, as they could be used to learn or reconstruct a Client's browsing history.

Attestation and redemption context unlinkability requires that these events be separated over time, e.g., through the use of tokens with an empty redemption context, or over space, e.g., through the use of an anonymizing proxy when connecting to the Origin.

4.2. Joint Attester and Issuer

In this model, the Attester and Issuer are operated by the same entity that is separate from the Origin, as shown in the figure below.

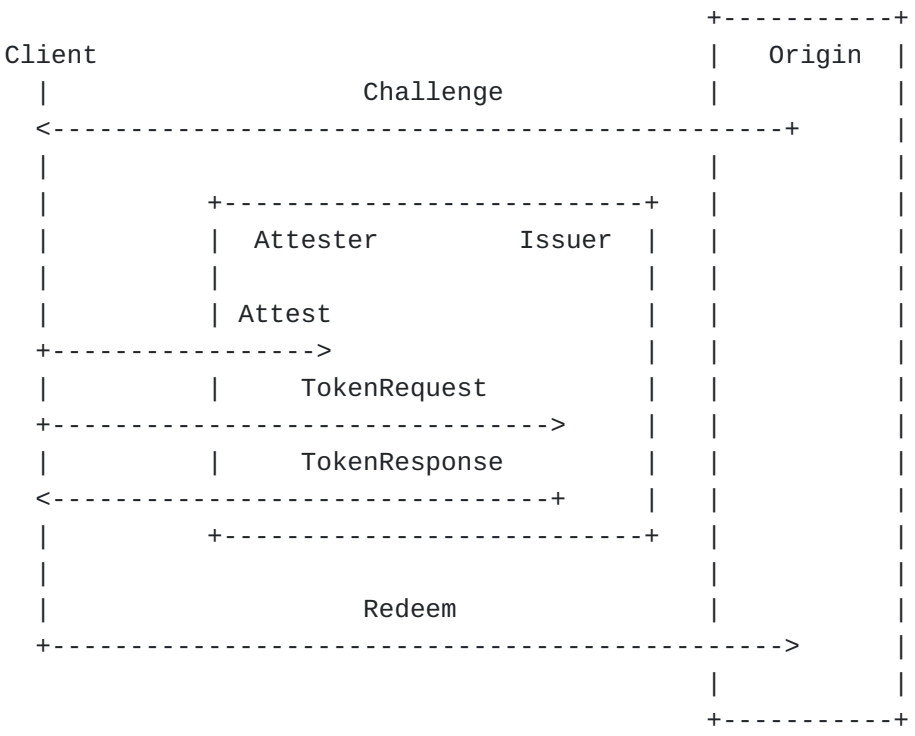

Figure 4: Joint Attester and Issuer Deployment Model

This model is useful if an Origin wants to offload attestation and issuance to a trusted entity. In this model, the Attester and Issuer share attestation context for the Client, which can be separate from the Origin's redemption context.

For certain types of issuance protocols, this model separates attestation and redemption contexts. However, Issuance protocols that require the Issuer to learn information about the Origin, such as that which is described in [RATE-LIMITED], are not appropriate since they could link attestation and redemption contexts through the Origin name.

4.3. Joint Origin and Issuer

In this model, the Origin and Issuer are operated by the same entity, separate from the Attester, as shown in the figure below.

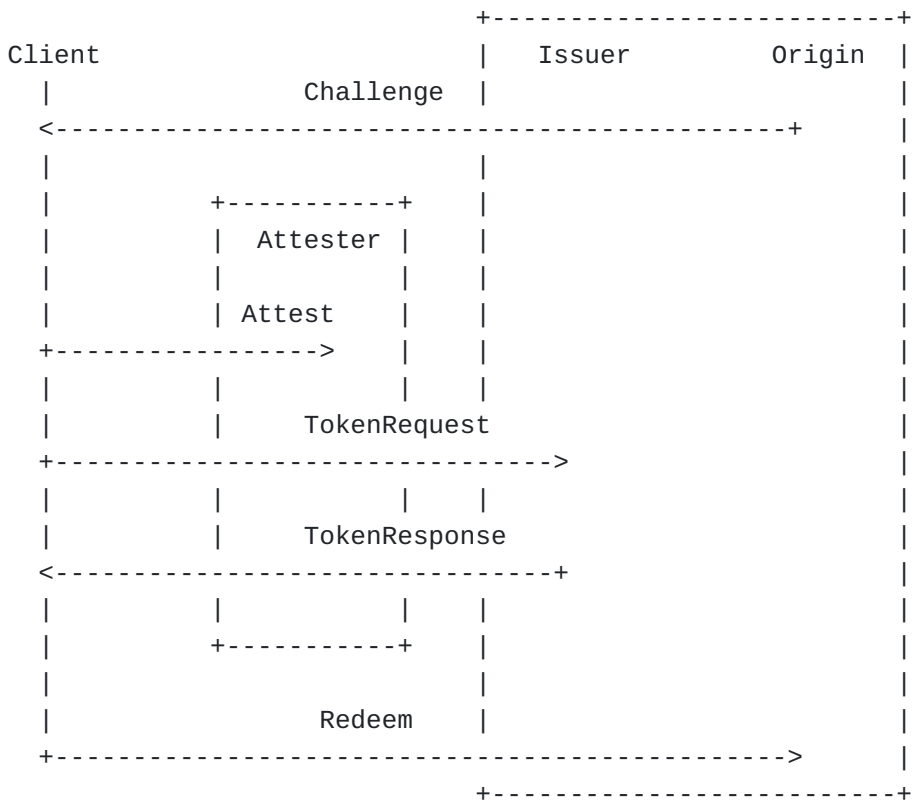

Figure 5: Joint Origin and Issuer Deployment Model

This model is useful for Origins that require Client-identifying attestation, e.g., through the use of application-layer account information, but do not otherwise want to learn information about individual Clients beyond what is observed during the token redemption, such as Client IP addresses.

In this model, attestation and redemption contexts are separate. As a result, any type of attestation is suitable in this model. Moreover, any type of token challenge is suitable assuming there is more than one Origin involved, since no single party will have access to the identifying Client information and unique Origin information. If there is only a single Origin, then per-Origin tokens are not appropriate in this model, since the Attester can learn the redemption context. (Note, however, that the Attester does not learn whether a token is per-Origin or cross-Origin.)

4.4. Split Origin, Attester, Issuer

In this model, the Origin, Attester, and Issuer are all operated by different entities, as shown in the figure below.

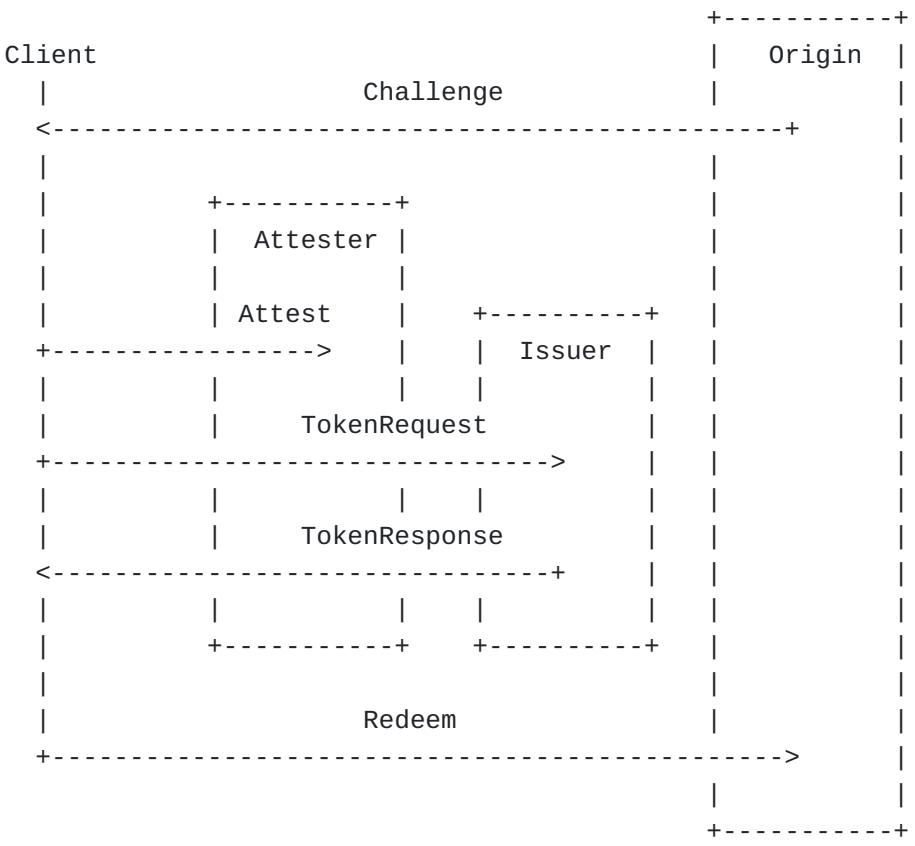

Figure 6: Split Deployment Model

This is the most general deployment model, and is necessary for some types of issuance protocols where the Attester plays a role in token issuance; see [RATE-LIMITED] for one such type of issuance protocol. In this model, the Attester, Issuer, and Origin have a separate view of the Client: the Attester sees potentially sensitive Client identifying information, such as account identifiers or IP addresses, the Issuer sees only the information necessary for Issuance, and the Origin sees token challenges, corresponding tokens, and Client source information, such as their IP address. As a result, attestation and redemption contexts are separate, and therefore any type of token challenge is suitable in this model assuming there is more than a single Origin. As in the Joint Origin and Issuer model in Section 4.3, if there is only a single Origin, then per-Origin tokens are not appropriate.

5. Privacy Considerations

Client uses Private Pass to separate attestation context and redemption context. Depending on the deployment model, this can take different forms. For example, any Client can only remain private relative to the entire space of other Clients using the protocol. Moreover, by owning tokens for a given set of keys, the Client's

anonymity set shrinks to the total number of clients controlling tokens for the same keys.

In the following, we consider the possible ways that Issuers can leverage their position to try and reduce the anonymity sets that Clients belong to (or, user segregation). For each case, we provide mitigations that the Privacy Pass ecosystem must implement to prevent these actions.

5.1. Metadata Privacy Implications

Any metadata bits of information can be used to further segment the size of the Client's anonymity set. Any Issuer that wanted to track a single Client could add a single metadata bit to Client tokens. For the tracked Client it would set the bit to 1, and 0 otherwise. Adding additional bits provides an exponential increase in tracking granularity similarly to introducing more Issuers (though with more potential targeting).

For this reason, the amount of metadata used by an Issuer in creating redemption tokens must be taken into account -- together with the bits of information that Issuer's may learn about Clients otherwise. Since this metadata may be useful for practical deployments of Privacy Pass, Issuers must balance this against the reduction in Client privacy. In general, Issuers should bound the metadata permitted so as to not allow it to uniquely identify each possible user.

5.2. Issuer Key Rotation

Techniques to introduce Client "segregation" can be used to reduce Client anonymity. Such techniques are closely linked to the type of key schedule that is used by the Issuer. When an Issuer rotates their key, any Client that invokes the issuance protocol in this key cycle will be part of a group of possible clients owning valid tokens for this key. To mechanize this attack strategy, an Issuer could introduce a key rotation policy that forces Clients into small key cycles. Thus, reducing the size of the anonymity set for these Clients.

Issuers SHOULD invoke key rotation for a period of time between 1 and 12 weeks. Key rotations represent a trade-off between Client privacy and continued Issuer security. Therefore, it is still important that key rotations occur on a regular cycle to reduce the harmfulness of an Issuer key compromise.

With a large number of Clients, a minimum of one week gives a large enough window for Clients to participate in the issuance protocol and thus enjoy the anonymity guarantees of being part of a larger group. A maximum of 12 weeks limits the damage caused by a key

compromise. If an Issuer realizes that a key compromise has occurred then the Issuer should generate a new key and make it available to Clients. If possible, it should invoke any revocation procedures that may apply for the old key.

5.3. Large Number of Issuers

Similarly to the Issuer rotation dynamic that is raised above, if there are a large number of Issuers, and Origins accept all of them, segregation can occur. For example, if Clients obtain tokens from many Issuers, and Origins later challenge a Client for a token from each Issuer, Origins can learn information about the Client. Each per-Issuer token that a Client holds essentially corresponds to a bit of information about the Client that Origin learn. Therefore, there is an exponential loss in anonymity relative to the number of Issuers that there are.

For example, if there are 32 Issuers, then Origins learn 32 bits of information about the Client if a valid token is presented for each one. If the distribution of Issuer trust is anything close to a uniform distribution, then this is likely to uniquely identify any Client amongst all other Internet users. Assuming a uniform distribution is clearly the worst-case scenario, and unlikely to be accurate, but it provides a stark warning against allowing too many Issuers at any one time.

In cases where clients can hold tokens for all Issuers at any given time, a strict bound SHOULD be applied to the active number of Issuers in the ecosystem. We propose that allowing no more than 4 Issuers at any one time is highly preferable (leading to a maximum of 64 possible user segregations). However, having a very large user base could potentially allow for larger values. Issuer replacements should only occur with the same frequency as config rotations as they can lead to similar losses in anonymity if clients still hold redemption tokens for previously active Issuers.

In addition, we RECOMMEND that trusted registries indicate at all times which Issuers are deemed to be active. If a Client is asked to invoke any Privacy Pass exchange for an Issuer that is not declared active, then the client SHOULD refuse to retrieve the Issuer public key during the protocol.

5.3.1. Allowing More Issuers

The bounds on the numbers of Issuers that this document proposes above are very restrictive. This is because this document considers a situation where a Client could be challenged (and asked to redeem) tokens for any Issuer.

An alternative system is to ensure a robust strategy for ensuring that Clients only possess redemption tokens for a similarly small number of Issuers at any one time. This prevents a malicious verifier from being able to invoke redemptions for many Issuers since the Client would only be holding redemption tokens for a small set of Issuers. When a Client is issued tokens from a new Issuer and already has tokens from the maximum number of Issuers, it simply deletes the oldest set of redemption tokens in storage and then stores the newly acquired tokens.

For example, if Clients ensure that they only hold redemption tokens for 4 Issuers, then this increases the potential size of the anonymity sets that the Client belongs to. However, this doesn't protect Clients completely as it would if only 4 Issuers were permitted across the whole system. For example, these 4 Issuers could be different for each Client. Therefore, the selection of Issuers they possess tokens for is still revealing. Understanding this trade-off is important in deciding the effective anonymity of each Client in the system.

5.3.1.1. Redemption Partitions

Another option to allow a large number of Issuers in the ecosystem, while preventing the joining of a number of different tokens is for the Client to maintain sharded "redemption partitions". This would allow the Client to redeem the tokens it wishes to use in a particular context, while still allowing the Client to maintain a large variety of tokens from many Issuers. Within a redemption partition, the Client limits the number of different Issuers used to a small number to maintain the privacy properties the Client requires. As long as each redemption partition maintains a strong privacy boundary with each other, the verifier will only be able to learn a number of bits of information up to the limits within that "redemption partitions".

To support this strategy, the client keeps track of a partition which contains the set of Issuers that redemptions have been attempted against. An empty redemption is returned when the limit has been hit:

```
Client(partition, issuer)                    Issuer(skS, pkS)
-------------------------------------------------------------
if issuer not in partition {
  if partition.length > REDEEM_LIMIT {
    Output {}
    return
  }
  partition.push(issuer)
}
token = store[issuer.id].pop()
req = Redeem(token, info)

                          req
                    ------------------>

                          if (dsIdx.includes(req.data)) {
                            raise ERR_DOUBLE_SPEND
                          }
                          resp = Verify(pkS, skS, req)
                          if resp.success {
                            dsIdx.push(req.data)
                          }

                           resp
                    <------------------
Output resp
```

5.4. Centralization

A consequence of limiting the number of participants (Attesters or Issuers) in Privacy Pass deployments for meaningful privacy is that it forces concentrated centralization amongst those participants. [CENTRALIZATION] discusses several ways in which this might be mitigated. For example, a multi-stakeholder governance model could be established to determine what participants are fit to operate as participants in a Privacy Pass deployment. This is precisely the model used to control the Web's trust model.

Alternatively, Privacy Pass deployments might mitigate this problem through implementation. For example, rather than centralize the role of attestation in one or few entities, attestation could be a distributed function performed by a quorum of many parties, provided that neither Issuers nor Origins learn which attester implementations were chosen. As a result, clients could have more opportunities to switch between attestation participants.

6. Security Considerations

We present a number of security considerations that prevent malicious Clients from abusing the protocol.

6.1. Token Exhaustion

When a Client holds tokens for an Issuer, it is possible for any verifier to invoke that client to redeem tokens for that Issuer. This can lead to an attack where a malicious verifier can force a Client to spend all of their tokens from a given Issuer. To prevent this from happening, tokens can be scoped to single Origins such that they can only be redeemed within for a single Origin.

If tokens are cross-Origin, Clients should use alternate methods to prevent many tokens from being redeemed at once. For example, if the Origin requests an excess of tokens, the Client could choose to not present any tokens for verification if a redemption had already occurred in a given time window.

7. References

7.1. 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>.

[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>.

[RFC8446] 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>.

7.2. Informative References

[CENTRALIZATION] Nottingham, M., "Centralization, Decentralization, and Internet Standards", Work in Progress, Internet-Draft, draft-nottingham-avoiding-internet-centralization-04, 26 June 2022, <https://datatracker.ietf.org/doc/html/draft-nottingham-avoiding-internet-centralization-04>.

[CONSISTENCY] Davidson, A., Finkel, M., Thomson, M., and C. A. Wood, "Key Consistency and Discovery", Work in Progress, Internet-Draft, draft-wood-key-consistency-02, 4 March 2022, <https://datatracker.ietf.org/doc/html/draft-wood-key-consistency-02>.

[HIJK21] Huang, S., Iyengar, S., Jeyaraman, S., Kushwah, S., Lee, C. K., Luo, Z., Mohassel, P., Raghunathan, A., Shaikh,

S., Sung, Y. C., and A. Zhang, "PrivateStats: De-Identified Authenticated Logging at Scale", January 2021, <https://research.fb.com/wp-content/uploads/2021/01/PrivateStats-De-Identified-Authenticated-Logging-at-Scale_final.pdf>.

[HTTP-Authentication] Pauly, T., Valdez, S., and C. A. Wood, "The Privacy Pass HTTP Authentication Scheme", Work in Progress, Internet-Draft, draft-ietf-privacypass-auth-scheme-02, 4 April 2022, <https://datatracker.ietf.org/doc/html/draft-ietf-privacypass-auth-scheme-02>.

[KLOR20] Kreuter, B., Lepoint, T., Orrù, M., and M. Raykova, "Anonymous Tokens with Private Metadata Bit", Advances in Cryptology - CRYPTO 2020 pp. 308-336, DOI 10.1007/978-3-030-56784-2_11, 2020, <https://doi.org/10.1007/978-3-030-56784-2_11>.

[PPEXT] "Privacy Pass Browser Extension", n.d., <https://github.com/privacypass/challenge-bypass-extension>.

[PPSRV] Sullivan, N., "Cloudflare Supports Privacy Pass", n.d., <https://blog.cloudflare.com/cloudflare-supports-privacy-pass/>.

[RATE-LIMITED] Hendrickson, S., Iyengar, J., Pauly, T., Valdez, S., and C. A. Wood, "Rate-Limited Token Issuance Protocol", Work in Progress, Internet-Draft, draft-privacypass-rate-limit-tokens-02, 2 May 2022, <https://datatracker.ietf.org/doc/html/draft-privacypass-rate-limit-tokens-02>.

[RFC6962] Laurie, B., Langley, A., and E. Kasper, "Certificate Transparency", RFC 6962, DOI 10.17487/RFC6962, June 2013, <https://www.rfc-editor.org/rfc/rfc6962>.

Appendix A. Acknowledgements

The authors would like to thank Scott Hendrickson, Tommy Pauly, Benjamin Schwartz, Steven Valdez and other members of the Privacy Pass Working Group for many helpful contributions to this document.

Authors' Addresses

Alex Davidson
LIP
Lisbon
Portugal

Email: alex.davidson92@gmail.com

Jana Iyengar
Fastly

Email: jri@fastly.com

Christopher A. Wood
Cloudflare
101 Townsend St
San Francisco,
United States of America

Email: caw@heapingbits.net