Network Working Group
Internet-Draft
Intended status: Informational
Expires: 14 January 2021

A. Davidson
Cloudflare Portugal

13 July 2020

Privacy Pass: Architectural Framework
draft-davidson-pp-architecture-01

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 14 January 2021.

Table of Contents

1. Introduction

The Privacy Pass protocol provides an anonymity-preserving mechanism for authorization of clients with servers. The protocol is detailed in [draft-davidson-pp-protocol] and is intended for use in the application-layer.

The way that the ecosystem around the protocol is set up can have significant impacts on the stated privacy and security guarantees of the protocol. For instance, the number of servers issuing Privacy Pass tokens, along with the number of registered clients, determines the anonymity set of each individual client. Moreover, this can be influenced by other factors, such as: the key rotation policy used by each server; and, the number of supported ciphersuites. There are also client behavior patterns that can reduce the effective security of the server.

In this document, we will provide a structural framework for building the ecosystem around the Privacy Pass protocol. The core of the document also includes policies for the following considerations.

* How server key material should be managed and accessed.

* Compatible server issuance and redemption running modes and associated expectations.

* How clients should evaluate server trust relationships.

* Security and privacy properties of the protocol.

* A concrete assessment and parametrization of the privacy budget associated with different settings of the above policies.

* The incorporation of potential extensions into the wider ecosystem.

Finally, we will discuss existing applications that make use of the Privacy Pass protocol, and highlight how these may fit with the proposed framework.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

The following terms are used throughout this document.

* Server: An entity that issues anonymous tokens to clients. In symmetric verification cases, the server must also verify tokens. Also referred to as the server.

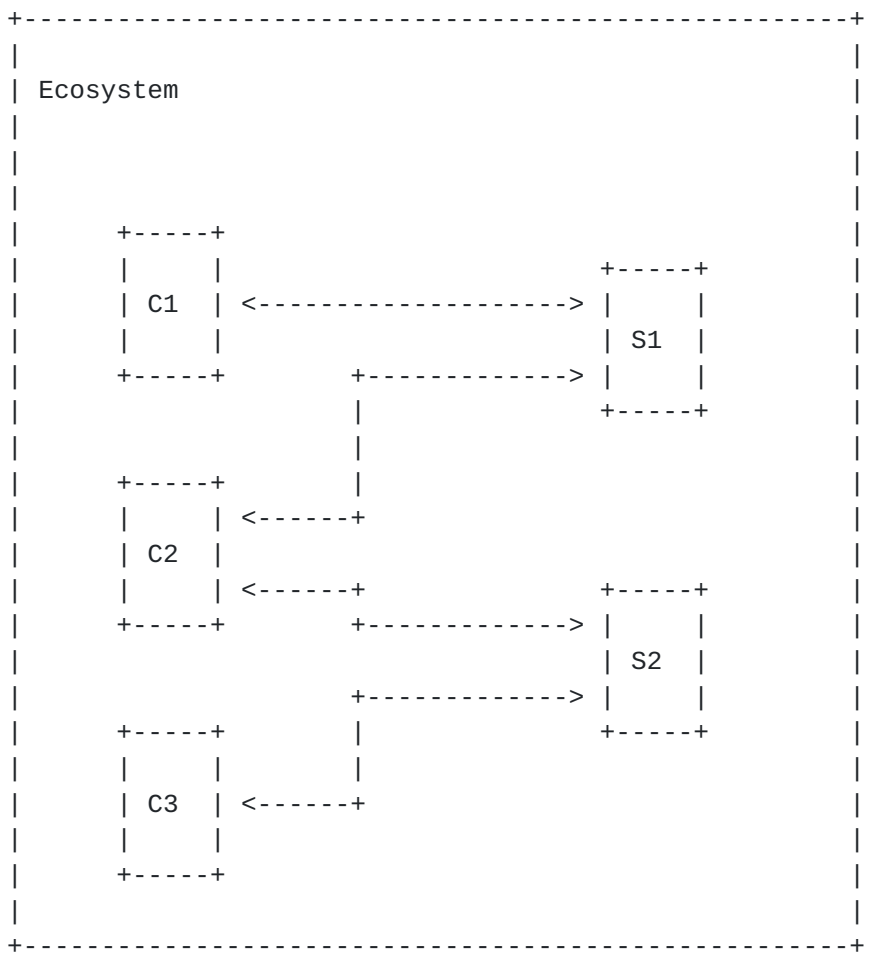
* Client: An entity that seeks authorization from a server.

We assume that all protocol messages are encoded into raw byte format before being sent. We use the TLS presentation language [RFC8446] to describe the structure of the data that is communicated and stored.

3. Ecosystem participants

The Privacy Pass ecosystem refers to the global framework in which multiple instances of the Privacy Pass protocol operate. This refers to all servers that support the protocol, or any extension of it, along with all of the clients that may interact with these servers.

The ecosystem itself, and the way it is constructed, is critical for evaluating the privacy of each individual client. We assume that a client's privacy refers to fraction of users that it represents in the anonymity set that it belongs to. We discuss this more in Section 7.



In the above diagram, the arrows indicate the open channels between a client and a server. An open channel indicates that a client accepts Privacy Pass tokens from this server.

If no channel exists, this means that the client chooses not to accept tokens from (or redeem tokens with) that particular server. We discuss the roles of servers and clients further in Section 3.1 and Section 3.2, respectively.

3.1. Servers

Generally, servers in the Privacy Pass ecosystem are entities whose primary function is to undertake the role of the "server" in [draft-davidson-pp-protocol]. To facilitate this, the server MUST hold a Privacy Pass protocol keypair at any given time. The server public key MUST be made available to all clients in such a way that key rotations and other updates are publicly visible. The server MAY also require additional state for ensuring this. We provide a wider discussion in Section 4.

Note that, in the core protocol instantiation from [draft-davidson-pp-protocol], the redemption phase is a symmetric protocol. This means that the server is the same server that ultimately processes token redemptions from clients. However, plausible extensions to the protocol specification may allow public verification of tokens by entities which do not hold the secret Privacy Pass keying material. We highlight possible client and server configurations in Section 5.

The server must be uniquely identifiable by all clients with a consistent identifier.

3.2. Clients

Clients in the Privacy Pass ecosystem are entities whose primary function is to undertake the role of the "Client" in [draft-davidson-pp-protocol]. Clients are assumed to only store data related to the tokens that it has been issued by the server. This storage is used for constructing redemption requests.

Clients MAY choose not to accept tokens from servers that they do not trust. See Section 6 for a wider discussion.

3.2.1. Client identifying information

Privacy properties of this protocol do not take into account other possibly identifying information available in an implementation, such as a client's IP address. Servers which monitor IP addresses may use this to track client redemption patterns over time. Clients cannot check whether servers monitor such identifying information. Thus, clients SHOULD minimize or remove identifying information where possible, e.g., by using anonymity-preserving tools such as Tor to interact with servers.

4. Key management framework

The key material and protocol configuration that a server uses to issue tokens corresponds to a number of different pieces of information.

* The ciphersuite that the server is using.

* The public keys that are active for the server.

For reasons that we address later in Section 7, the way that the server publishes and maintains this information impacts the effective privacy of the clients. In this section, we describe the main

policies that need to be satisfied for a key management system in a Privacy Pass ecosystem.

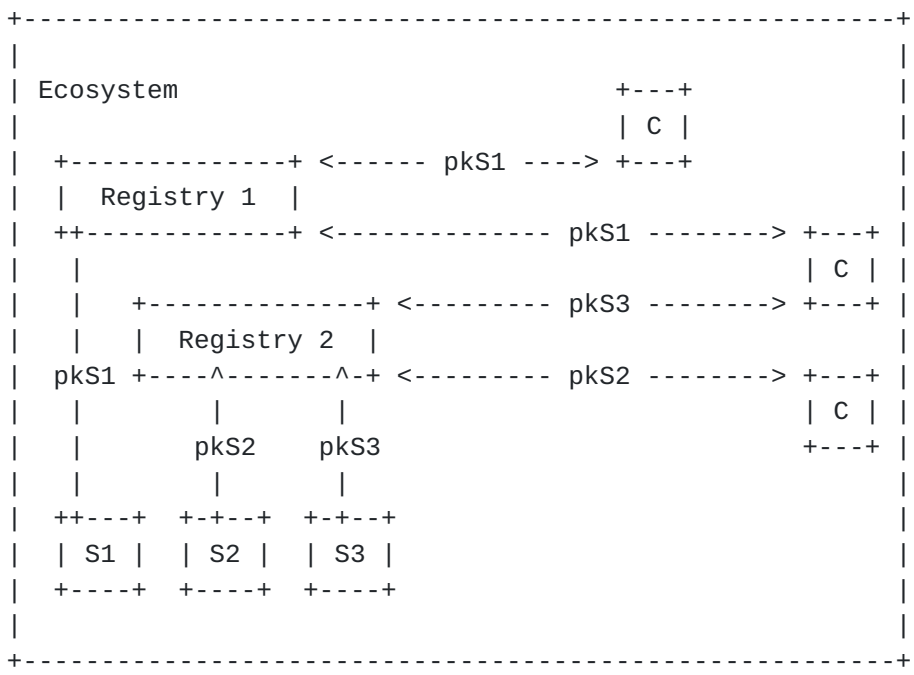
Note that we only specify a set of guidelines and recommendations for operating a public key registry in this document. Actual specification of such a registry and how it operates will be covered elsewhere.

4.1. Public key registries

Server's must provide their public keys to clients along with details about the cryptographic ciphersuite that they are using. In Section 7, we address the importance of providing clients with sources of truth for learning the server's key configuration.

In particular, server key material MUST be publicly available in a tamper-proof data structure, which we refer to as a key registry. A registry must be globally consistent. Clients using the same registry should coordinate in some way to ensure they have a consistent view of said registry. This can be done via gossiping or some other mechanism. The exact mechanism for this coordination will be described elsewhere. It is assumed there will be at least one such mechanism.

It is RECOMMENDED that each key registry is an append-only data structure, such as a Merkle Tree. The key registry should be operated independently of any server that publishes key material to the registry. This ensures that any client can make better judgements on whether to trust the registry and, transitively, each server.



While there may be multiple key registries for a given ecosystem, a server MUST only publish its key material to a single registry. This ensures that the server is keeping a consistent view of its key material.

4.2. Key rotation

Token issuance associates all issued tokens with a particular choice of key. If a server 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, servers MUST only use one private key for issuing tokens at any given time. Servers may use two or more keys for redemption to allow servers for seamless key rotation.

Key rotations must be limited in frequency for similar reasons. See Section 9 for guidelines on what frequency of key rotations are permitted.

4.3. Ciphersuites

Since a server is only permitted to have a single active issuing key, this implies that only a single ciphersuite is allowed per issuance period. If a server wishes to change their ciphersuite, they MUST do so during a key rotation.

5. Server running modes

We provide an overview of some of the possible frameworks for configuring the way that servers run in the Privacy Pass ecosystem. In short, servers may be configured to provide symmetric issuance and redemption with clients. While some servers may be configured as proxies that accept Privacy Pass data and send it to another server that actually processes issuance or redemption data. Finally, we also consider instances of the protocol that may permit public verification.

The intention with providing each of these running modes is to cover the different applications that utilize variants of the Privacy Pass protocol. We RECOMMEND that any Privacy Pass server implementation adheres to one of these frameworks.

5.1. Single-Verifier

The simplest way of considering the Privacy Pass protocol is in a setting where the same server plays the role of server and verifier, we call this "Single-Verifier" (SV).

Let S be the server, and C be the client. When S wants to issue tokens to C, they invoke the issuance protocol where C generates their own inputs, and S uses their secret key skS. In this setting, C can only perform token redemption with S. When a token redemption is required, C and S invoke the redemption phase of the protocol, where C uses an issued token from a previous exchange, and S uses skS to validate the redemption.

5.2. Delegated-Verifier

In this setting, each client C obtains issued tokens from a derver S via the issuance phase of the protocol. The difference is that C can prove that they hold a valid authorization with any verifier V. We still only consider S to hold their own secret key. We name this mode "Delegated-Verifier" (DV).

When C interacts with V, V can ask C to provide proof of authorization to the separate server S. The first stage of the redemption phase of the protocol is invoked between C and V, which sees C send an unused redemption token to V. This message is then used in a redemption exchange between V and S, where V plays the role of the Client. Then S sends the result of the redemption verification to V, and V uses this result to determine whether C has a valid token.

5.3. Asynchronous-Verifier

This setting is inspired by recently proposed APIs such as [TrustTokenAPI]. It is similar to the DV configuration, except that the verifiers V no longer interact with the server S. Only C interacts with S, and this is done asynchronously to the authorization request from V. Hence "Asynchronous-Verifier" (AV).

When V invokes a redemption for C, C then invokes a redemption exchange with S in a separate session. If verification is carried out successfully by S, S instead returns a Signed Redemption Record (SRR) that contains the following information:

```
"result": {
  "timestamp":"2019-10-09-11:06:11",
  "verifier": "V",
},
"signature":sig,
```

The "signature" field carries a signature evaluated over the contents of "result" using a long-term signing key for the server S, of which the corresponding public key is well-known to C and V. This would need to be published alongside other public key data for S. Then C can prove that they hold a valid authorization from S to V by sending the SRR to V. The SRR can be verified by V by verifying the signature, using the well-known public key for S.

Such records can be cached to display again in the future. The server can also add an expiry date to the record to determine when the client must refresh the record.

5.4. Public-Verifier

We consider the case where client redemptions can be verified publicly using the server public key. This allows for defining extensions of Privacy Pass that use public-key cryptography to allow public verification.

In this case, the client C obtains a redemption token from S. The redemption token is publicly verifiable in the sense that any entity that knows the public key for S can verify the token. This running mode is known as "Public-Verifier" (PV).

5.5. Bounded number of servers

Each of the configurations above can be generalized to settings where a bounded number of servers are allowed, and verifiers can invoke authorization verification for any of the available servers.

As we will discuss later in Section 7, configuring a large number of servers can lead to privacy concerns for the clients in the ecosystem. Therefore, we are careful to ensure that the number of servers is kept strictly bounded. The actual servers can be replaced with different servers as long as the total never exceeds this bound. Moreover, server replacements also have an effect on client anonymity that is similar to when a key rotation occurs. server so replacement should only be permitted at similar intervals.

See Section 7 for more details about maintaining privacy with multiple servers.

6. Client-Server trust relationship

It is important, based on the architecture above, that any client can determine whether it would like to interact with a given server in the ecosystem. Note that this decision must be taken before a client issues a valid redemption to the server, since redemptions reveal the anonymity set that the client belongs to.

This judgement can be based on a multitude of factors, associated with the way that a server presents itself in the ecosystem. A non-exhaustive list of server characteristics that a client MAY want to check are the following.

* Which key registry a server posts their key updates to.

* How frequent key updates are issued, and which ciphersuite they use.

* The reason given to initiate the redemption.

To aid client trust decisions, a server can publish a "Privacy Pass policy" that documents the procedures that the server uses to ensure that client privacy is respected. If a server does not publish such a document then the client may choose to use its own judgement, or to reject the server altogether.

It should be noted that the client trust decision can be made apriori by specifying an allow-list of all servers that it accepts tokens from. This means that these checks do not have to be performed online.

7. Privacy considerations

In the Privacy Pass protocol [draft-davidson-pp-protocol], redemption tokens intentionally encode very little information beyond which key was used to sign them. The protocol intentionally uses components that provide cryptographic guarantees of this fact. However, even with these guarantees, the way that the ecosystem is constructed can be used to identify clients based on this limited information.

The goal of the Privacy Pass ecosystem is to construct an environment that can easily measure (and maximize) the relative anonymity of any client that is part of it. An inherent feature of being part of this ecosystem is that any client can only remain private relative to the entire space of users 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 servers and servers 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.

7.1. Server 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 server. When a server 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, a server could introduce a key rotation policy that forces clients into small key cycles. Thus, reducing the size of the anonymity set for these clients.

We RECOMMEND that servers should only invoke key rotation for fairly large periods of time such as between 1 and 12 weeks. Key rotations represent a trade-off between client privacy and continued server security. Therefore, it is still important that key rotations occur on a fairly regular cycle to reduce the harmfulness of a server key compromise.

With an active user-base, a week gives a fairly large window for clients to participate in the Privacy Pass protocol and thus enjoy the anonymity guarantees of being part of a larger group. The low ceiling of 12 weeks prevents a key compromise from being too destructive. If a server realizes that a key compromise has occurred then the server should sample a new key, and upload the public key to

the key registry; invoking any revocation procedures that may apply for the old key.

7.2. Large numbers of servers

Similarly to the server rotation dynamic that is raised above, if there are a large number of servers then segregation can occur. In the FV, AV and PV running modes (Section 5), a verifier OV can trigger redemptions for any of the available servers. Each redemption token that a client holds essentially corresponds to a bit of information about the client that OV can learn. Therefore, there is an exponential loss in anonymity relative to the number of servers that there are.

For example, if there are 32 servers, then OV learns 32 bits of information about the client. If the distribution of server 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 servers at any one time.

In cases where clients can hold tokens for all servers at any given time, a strict bound SHOULD be applied to the active number of servers in the ecosystem. We propose that allowing no more than 4 servers at any one time is highly preferable (leading to a maximum of 64 possible user segregations). However, as highlighted in Section 9, having a very large user base (> 5 million users), could potentially allow for larger values. server 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 servers.

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

7.2.1. Allowing larger number of servers

The bounds on the numbers of servers that we proposed above are very restrictive. This is due to the fact that we considered a situation where a client could be issued (and forced to redeem) tokens for any issuing key.

An alternative system is to ensure a robust strategy for ensuring that clients only possess redemption tokens for a similarly small

number of servers at any one time. This prevents a malicious verifier from being able to invoke redemptions for many servers since the client would only be holding redemption tokens for a small set of servers. When a client is issued tokens from a new server and already has tokens from the maximum number of servers, 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 servers, 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 servers were permitted across the whole system. For example, these 4 servers could be different for each client. Therefore, the selection of servers they possess tokens for is still revealing. Understanding this trade-off is important in deciding the effective anonymity of each client in the system.

7.3. Partitioning of server key material

If there are multiple key registries, or if a key registry colludes with an server, then it is possible to provide a split-view of an server's key material to different clients. This would involve posting different key material in different locations, or actively modifying the key material at a given location.

Key registries should operate independently of server's in the ecosystem, and within the guidelines stated in Section 4. Any client should follow the recommendations in Section 6 for determining whether an server and its key material should be trusted.

7.4. Additional token metadata

In [draft-davidson-pp-protocol], it is permissible to add public and private metadata bits to redemption tokens. The core protocol instantiation that is described does not include additional metadata. However, future instantiations may use this functionality to provide redemption verifiers with additional information about the user.

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

For this reason, the amount of metadata used by an server in creating redemption tokens must be taken into account - together with the bits of information that server's may learn about clients otherwise. We discuss this more in Section 9.

7.5. Tracking and identity leakage

Privacy losses may be encountered if too many redemptions are allowed in a short burst. For instance, in the Internet setting, this may allow delegated or asynchronous verifiers to learn more information from the metadata that the client may hold (such as first-party cookies for other domains). Mitigations for this issue are similar to those proposed in Section 7.2 for tackling the problem of having large number of servers.

In AV, cached SRRs and their associated server public keys have a similar tracking potential to first party cookies in the browser setting. These considerations will be covered in a separate document, detailing Privacy Pass protocol integration into the wider web architecture [draft-svaldez-pp-http-api].

7.6. Client incentives for anonymity reduction

Clients may see an incentive in accepting all tokens that are issued by a server, even if the tokens fail later verification checks. This is because tokens effectively represent a form of currency that they can later redeem for some sort of benefit. The verification checks that are put in place are there to ensure that the client does not sacrifice their anonymity. However, a client may judge the "monetary" benefit of owning tokens to be greater than their own privacy.

Firstly, a client behaving in this way would not be compliant with the protocol, as laid out in [draft-davidson-pp-protocol].

Secondly, acting in this way only affects the privacy of the immediate client. There is an exception if a large number of clients colluded to accept bad data, then any client that didn't accept would be part of a smaller anonymity set. However, such a situation would be identical to the situation where the total number of clients in the ecosystem is small. Therefore, the reduction in the size of the anonymity set would be equivalent; see Section 7.2 for more details.

8. Security considerations

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

8.1. Double-spend protection

All issuing server should implement a robust storage-query mechanism for checking that tokens sent by clients have not been spent before. Such tokens only need to be checked for each server individually. But all servers must perform global double-spend checks to avoid clients from exploiting the possibility of spending tokens more than once against distributed token checking systems. For the same reason, the global data storage must have quick update times. While an update is occurring it may be possible for a malicious client to spend a token more than once.

8.2. Token exhaustion

When a client holds tokens for an server, it is possible for any verifier to invoke that client to redeem tokens for that server. This can lead to an attack where a malicious verifier can force a client to spend all of their tokens for a given server. To prevent this from happening, methods should be put into place to prevent many tokens from being redeemed at once.

For example, it may be possible to cache a redemption for the entity that is invoking a token redemption. If the verifier requests more tokens then the client simply returns the cached token that it returned previously. This could also be handled by simply not redeeming any tokens for verification if a redemption had already occurred in a given time window.

In AV, the client instead caches the SRR that it received in the asynchronous redemption exchange with the server. If the same verifier attempts another redemption request, then the client simply returns the cached SRR. The SRRs can be revoked by the server, if need be, by providing an expiry date or by signaling that records from a particular window need to be refreshed.

8.3. Avoiding server centralization

[[OPEN ISSUE: explain potential and mitigations for server centralization]]

9. Protocol parametrization

We provide a summary of the parameters that we use in the Privacy Pass protocol ecosystem. These parameters are informed by both privacy and security considerations that are highlighted in Section 7 and Section 8, respectively. These parameters are intended as a single reference point for those implementing the protocol.

Firstly, let U be the total number of users, I be the total number of servers. We let M be the total number of metadata bits that are allowed to be added by any given server. Assuming that each user accept tokens from a uniform sampling of all the possible servers, as a worst-case analysis, this segregates users into a total of 2^I buckets. As such, we see an exponential reduction in the size of the anonymity set for any given user. This allows us to specify the privacy constraints of the protocol below, relative to the setting of A.

parameter	value
Minimum anonymity set size (A)	5000
Recommended key lifetime (L)	2 - 24 weeks
Recommended key rotation frequency (F)	L/2
Maximum additional metadata bits (M)	1
Maximum allowed servers (I)	(log_2(U/A)-1)/2
Maximum active issuance keys	1
Maximum active redemption keys	2
Minimum cryptographic security parameter	128 bits

Table 1

9.1. Justification

We make the following assumptions in these parameter choices.

* Inferring the identity of a user in a 5000-strong anonymity set is difficult.

* After 2 weeks, all clients in a system will have rotated to the new key.

In terms of additional metadata, the only concrete applications of Privacy Pass that use additional metadata require just a single bit. Therefore, we set the ceiling of permitted metadata to 1 bit for now, this may be revisited in future revisions.

The maximum choice of I is based on the equation 1/2 * U/2^(2I) = A. This is derived from the fact that permitting I servers lead to 2^I segregations of the total user-base U. Moreover, if we permit M = 1, then this effectively halves the anonymity set for each server, and thus we incur a factor of 2I in the exponent. By reducing I, we limit the possibility of performing the attacks mentioned in Section 7.

We must also account for each user holding issued data for more then one possible active keys. While this may also be a vector for monitoring the access patterns of clients, it is likely to unavoidable that clients hold valid issuance data for the previous key epoch. This also means that the server can continue to verify redemption data for a previously used key. This makes the rotation period much smoother for clients.

For privacy reasons, it is recommended that key epochs are chosen that limit clients to holding issuance data for a maximum of two keys. By choosing F = L/2 then the minimum value of F is a week, since the minimum recommended value of L is 2 weeks. Therefore, by the initial assumption, then all users should only have access to only two keys at any given time. This reduces the anonymity set by another half at most.

Finally, the minimum security parameter size is related to the cryptographic security offered by the protocol that is run. This parameter corresponds to the number of operations that any adversary has in breaking one of the security guarantees in the Privacy Pass protocol [draft-davidson-pp-protocol].

9.2. Example parameterization

Using the specification above, we can give some example parameterizations. For example, the current Privacy Pass browser extension [PPEXT] has nearly 300000 active users (from Chrome and Firefox). As a result, log_2(U/A) is approximately 6 and so the maximum value of I should be 3.

If the value of U is much bigger (e.g. 5 million) then this would permit I = (log_2(5000000/5000)-1)/2 ~= 4 servers.

9.3. Allowing more servers

Using the recommendations in Section 7.2.1, it is possible to tolerate larger number of servers if clients in the ecosystem ensure that they only store tokens for a small number of them. In particular, if clients limit their storage of redemption tokens to the bound implied by I, then prevents a malicious verifier from triggering redemptions for all servers in the ecosystem.

10. Extension integration policy

The Privacy Pass protocol and ecosystem are both intended to be receptive to extensions that expand the current set of functionality. As specified in [draft-davidson-pp-protocol], all extensions to the Privacy Pass protocol SHOULD be specified as separate documents that modify the content of this document in some way. We provide guidance on the type of modifications that are possible in the following.

Any such extension should also come with a detailed analysis of the privacy impacts of the extension, why these impacts are justified, and guidelines on changes to the parametrization in Section 9. Similarly, extensions MAY also add new server running modes, if applicable, to those that are documented in Section 5.

Any extension to the Privacy Pass protocol must adhere to the guidelines specified in Section 4 for managing server public key data.

11. Existing applications

The following is a non-exhaustive list of applications that currently make use of the Privacy Pass protocol, or some variant of the underlying functionality.

11.1. Cloudflare challenge pages

Cloudflare uses an implementation of the Privacy Pass protocol for allowing clients that have previously interacted with their Internet challenge protection system to bypass future challenges [PPSRV]. These challenges can be expensive for clients, and there have been cases where bugs in the implementations can severely degrade client accessibility.

Clients must install a browser extension [PPEXT] that acts as the Privacy Pass client in an exchange with Cloudflare's Privacy Pass server, when an initial challenge solution is provided. The client extension stores the issued tokens and presents a valid redemption token when it sees future Cloudflare challenges. If the redemption

token is verified by the server, the client passes through the security mechanism without completing a challenge.

11.2. Trust Token API

The Trust Token API [TrustTokenAPI] has been devised as a generic API for providing Privacy Pass functionality in the browser setting. The API is intended to be implemented directly into browsers so that server's can directly trigger the Privacy Pass workflow.

11.3. Zero-knowledge Access Passes

The PrivateStorage API developed by Least Authority is a solution for uploading and storing end-to-end encrypted data in the cloud. A recent addition to the API [PrivateStorage] allows clients to generate Zero-knowledge Access Passes (ZKAPs) that the client can use to show that it has paid for the storage space that it is using. The ZKAP protocol is based heavily on the Privacy Pass redemption mechanism. The client receives ZKAPs when it pays for storage space, and redeems the passes when it interacts with the PrivateStorage API.

11.4. Basic Attention Tokens

The browser Brave uses Basic Attention Tokens (BATs) to provide the basis for an anonymity-preserving rewards scheme [Brave]. The BATs are essentially Privacy Pass redemption tokens that are provided by a central Brave server when a client performs some action that triggers a reward event (such as watching an advertisement). When the client amasses BATs, it can redeem them with the Brave central server for rewards.

11.5. Token Based Services

Similarly to BATs, a more generic approach for providing anonymous peers to purchase resources from anonymous servers has been proposed [OpenPrivacy]. The protocol is based on a variant of Privacy Pass and is intended to allow clients purchase (or pre-purchase) services such as message hosting, by using Privacy Pass redemption tokens as a form of currency. This is also similar to how ZKAPs are used.

12. References

12.1. Normative References

[draft-davidson-pp-protocol]
Davidson, A., "Privacy Pass: The Protocol", n.d., <https://tools.ietf.org/html/draft-davidson-pp-protocol-00>.

[draft-svaldez-pp-http-api]
Valdez, S., "Privacy Pass: HTTP API", n.d., <https://github.com/alxdavids/privacy-pass-ietf/tree/master/drafts/draft-svaldez-pp-http-api>.

[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/info/rfc2119>.

[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/info/rfc8446>.

12.2. Informative References

[Brave] "Brave Rewards", n.d., <https://brave.com/brave-rewards/>.

[OpenPrivacy]
"Token Based Services - Differences from PrivacyPass", n.d., <https://openprivacy.ca/assets/towards-anonymous-prepaid-services.pdf>.

[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/>.

[PrivateStorage]
Steininger, L., "The Path from S4 to PrivateStorage", n.d., <https://medium.com/least-authority/the-path-from-s4-to-privatestorage-ae9d4a10b2ae>.

[TrustTokenAPI]
Google, ., "Getting started with Trust Tokens", n.d., <https://web.dev/trust-tokens/>.

Appendix A. Contributors

* Alex Davidson (alex.davidson92@gmail.com)

* Christopher Wood (caw@heapingbits.net)

Author's Address

Alex Davidson
Cloudflare Portugal
Largo Rafael Bordalo Pinheiro 29
Lisbon
Portugal

Email: alex.davidson92@gmail.com