Workgroup: Network Working Group
Internet-Draft:
draft-kpw-iab-privacy-partitioning-00
Published: 21 October 2022
Intended Status: Informational
Expires: 24 April 2023

Authors: M. Kühlewind Ericsson Research
T. Pauly Apple
C. A. Wood Cloudflare

Partitioning as an Architecture for Privacy

Abstract

This document describes the principle of privacy partitioning, which selectively spreads data and communication across multiple parties as a means to improve the privacy by separating user identity from user data. This document describes emerging patterns in protocols to partition what data and metadata is revealed through protocol interactions, provides common terminology, and discusses how to analyze such models.

Discussion Venues

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

Discussion of this document takes place on the Internet Architecture Board Internet Engineering Task Force mailing list (iab@iab.org), which is archived at .

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

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Protocols such as TLS and IPsec provide a secure (authenticated and encrypted) channel between two endpoints over which endpoints transfer information. Encryption and authentication of data in transit is necessary to protect information from being seen or modified by parties other than the intended protocol participants. As such, this kind of security is necessary for ensuring that information transferred over these channels remain private.

However, a secure channel between two endpoints is insufficient for privacy of the endpoints themselves. In recent years, privacy requirements have expanded beyond the need to protect data in transit between two endpoints. Some examples of this expansion include:

- *A user accessing a service on a website might not consent to reveal their location, but if that service is able to observe the client's IP address, it can learn inforamtion about the user's location. This is problematic for privacy since the service can link user data to the user's location.

- *A user might want to be able to access content for which they are authorized, such as a news article, without needing to have which specific articles they read on their account being recorded. This is problematic for privacy since the service can link user activity to the user's account.

- *A client device that needs to upload metrics to an aggregation service might want to be able to contribute data to the system without having their specific contributions being attribued to them. This is problematic for privacy since the service can link client contributions to the specific client.

The commonality in these examples is that clients want to interact with or use a service without exposing too much user-specific or identifying information to that service. In particular, separating the user-specific identity information from user-specific data is necessary for privacy. Thus, order to protect user privacy, it is important to keep identity (who) and data (what) separate.

This document defines "privacy partitioning" as the general technique used to separate the data and metadata visible to various parties in network communication, with the aim of improving user privacy. Partitioning is a spectrum and not a panacea. It is difficult to guarantee there is no link between user-specific identity and user-specific data. However, applied properly, privacy partitioning helps ensure that user privacy violations becomes more technically difficult to achieve over time.

Several IETF working groups are working on protocols or systems that adhere to the principle of privacy partitioning, including OHAI, MASQUE, Privacy Pass, and PPM. This document summarizes work in those groups and describes a framework for reasoning about the resulting privacy posture of different endpoints in practice.

2. Privacy Partitioning

For the purposes of user privacy, this document focuses on user-specific information. This might include any identifying information that is specific to a user, such as their email address or IP address, or data about the user, such as their date of birth. Informally, the goal of privacy partitioning is to ensure that each party in a system beyond the user themselves only has access to one type of user-specific information.

This is a simple application of the principle of least privilege, wherein every party in a system only has access to the minimum amount of information needed to fulfill their function. Privacy partitioning advocates for this minimization by ensuring that protocols, applications, and systems only reveal user-specific information to parties that need access to the information for their intended purpose.

Put simply, privacy partitioning aims to separate *who* someone is from *what* they do. In the rest of this section, we describe how privacy partitioning can be used to achieve this goal.

2.1. Privacy Contexts

Each piece of user-specific information exists within some context, where a context is abstractly defined as a set of data and metadata and the entities that share access to that information. In order to prevent correlation of user-specific information across contexts, partitions need to ensure that any single entity (other than the client itself) does not participate in more than one context where the information is visible.

[RFC6973] discusses the importance of identifiers in reducing correlation as a way of improving privacy:

"Correlation is the combination of various pieces of information related to an individual or that obtain that characteristic when combined... Correlation is closely related to identification. Internet protocols can facilitate correlation by allowing individuals' activities to be tracked and combined over time."

"Pseudonymity is strengthened when less personal data can be linked to the pseudonym; when the same pseudonym is used less often and across fewer contexts; and when independently chosen pseudonyms are more frequently used for new actions (making them, from an observer's or attacker's perspective, unlinkable)."

Context separation is foundational to privacy partitioning and reducing correlation. As an example, consider an unencrypted HTTP session over TCP, wherein the context includes both the content of the transaction as well as any metadata from the transport and IP headers; and the participants include the client, routers, other network middleboxes, intermediaries, and server.

```
+-------------------------------------------------------------------+
| Context A                                                         |
|  +--------+                +-----------+              +--------+  |
|  |        |------HTTP------|           |--------------|        |  |
|  | Client |                | Middlebox |              | Server |  |
|  |        |------TCP-------|           |--------------|        |  |
|  +--------+      flow      +-----------+              +--------+  |
|                                                                   |
+-------------------------------------------------------------------+
```

Figure 1: Diagram of a basic unencrypted client-to-server connection with middleboxes

Adding TLS encryption to the HTTP session is a simple partitioning technique that splits the previous context into two separate contexts: the content of the transaction is now only visible to the client, TLS-terminating intermediaries, and server; while the metadata in transport and IP headers remain in the original context. In this scenario, without any further partitioning, the entities that participate in both contexts can allow the data in both contexts to be correlated.

```
+-------------------------------------------------------------------+
| Context A                                                         |
|  +--------+                                           +--------+  |
|  |        |                                           |        |  |
|  | Client |-------------------HTTPS-------------------| Server |  |
|  |        |                                           |        |  |
|  +--------+                                           +--------+  |
|                                                                   |
+-------------------------------------------------------------------+
| Context B                                                         |
|  +--------+                +-----------+              +--------+  |
|  |        |                |           |              |        |  |
|  | Client |-------TCP------| Middlebox |--------------| Server |  |
|  |        |       flow     |           |              |        |  |
|  +--------+                +-----------+              +--------+  |
|                                                                   |
+-------------------------------------------------------------------+
```

Figure 2: Diagram of how adding encryption splits the context into two

Another way to create a partition is to simply use separate connections. For example, to split two separate HTTP requests from one another, a client could issue the requests on separate TCP connections, each on a different network, and at different times; and avoid including obvious identifiers like HTTP cookies across the requests.

```
+-------------------------------------------------------------------+
| Context A                                                         |
|  +--------+                +-----------+              +--------+  |
|  |        | IP A           |           |              |        |  |
|  | Client |-------TCP------| Middlebox |--------------| Server |  |
|  |        |       flow A   |     A     |              |        |  |
|  +--------+                +-----------+              +--------+  |
|                                                                   |
+-------------------------------------------------------------------+
| Context B                                                         |
|  +--------+                +-----------+              +--------+  |
|  |        | IP B           |           |              |        |  |
|  | Client |-------TCP------| Middlebox |--------------| Server |  |
|  |        |       flow B   |     B     |              |        |  |
|  +--------+                +-----------+              +--------+  |
|                                                                   |
+-------------------------------------------------------------------+
```

Figure 3: Diagram of making separate connections to generate separate contexts

2.2. Context Separation

In order to define and analyze how various partitioning techniques work, the boundaries of what is being partitioned need to be established. This is the role of context separation. In particular, in order to prevent correlation of user-specific information across contexts, partitions need to ensure that any single entity (other than the client itself) does not participate in contexts where both identities are visible.

Context separation can be achieved in different ways, e.g. over time, across network paths, based on (en)coding, etc. The privacy-oriented protocols described in this document generally involve more complex partitioning, but the techniques to partition communication contexts still employ the same techniques:

1. Encryption allows partitioning of contexts within a given network path.

2. Using separate connections across time or space allow partitioning of contexts for different application transactions.

These techniques are frequently used in conjunction for context separation. For example, encrypting an HTTP exchange might prevent a network middlebox that sees a client IP address from seeing the user account identity, but it doesn't prevent the TLS-terminating server from observing both identities and correlating them. As such, preventing correlation requires separating contexts, such as by using proxying to conceal a client IP address that would otherwise be used as an identifier.

3. A Survey of Protocols using Partitioning

The following section discusses currently on-going work in the IETF that is applying privacy partitioning.

3.1. CONNECT Proxying and MASQUE

HTTP forward proxies, when using encryption, provide privacy partitioning by separating a connection into multiple segments. When connections over the proxy themselves are encrypted, the proxy cannot see the end-to-end content. HTTP has historically supported forward proxying for TCP-like streams via the CONNECT method. More recently, the MASQUE working group has developed protocols to similarly proxy UDP [CONNECT-UDP] and IP packets [CONNECT-IP] based on tunneling.

In a single-proxy setup there is a tunnel connection between the client and proxy and an end-to-end connection that is tunnelled between the client and target. This setup, as shown in the figure below, partitions communication into a Client-to-Proxy context (the transport metadata between the client and the target, and the request to the proxy to open a connection to the target), and a Client-to-Target context (the end-to-end data, which generally would be a TLS-encrypted connection). There is also a Proxy-to-Target context; in case of MASQUE this context only contains any (unprotected) packet header information that is added or modified by the proxy, e.g., the IP and UDP headers.

```
+-------------------------------------------------------------------+
| Client-to-Target Context                                          |
|  +--------+                +-----------+              +--------+  |
|  |        |                |           |              |        |  |
|  | Client |----Proxied-----|   Proxy   |--------------| Server |  |
|  |        |      flow      |           |              |        |  |
|  +--------+                +-----------+              +--------+  |
|                                                                   |
+-------------------------------------------------------------------+
| Client-to-Proxy Context                                           |
|  +--------+                +-----------+                          |
|  |        |                |           |                          |
|  | Client |---Transport----|   Proxy   |                          |
|  |        |     flow       |           |                          |
|  +--------+                +-----------+                          |
|                                                                   |
+-------------------------------------------------------------------+
| Proxy-to-Target Context                                           |
|                            +-----------+              +--------+  |
|                            |           |              |        |  |
|                            |   Proxy   |--Transport---| Server |  |
|                            |           |    flow      |        |  |
|                            +-----------+              +--------+  |
|                                                                   |
+-------------------------------------------------------------------+
```

Figure 4: Diagram of one-hop proxy contexts

Using two (or more) proxies provides better privacy partitioning. In particular, with two proxies, each proxy sees the Client metadata, but not the Target; the Target, but not the Client metadata; or neither.

```
+-------------------------------------------------------------------+
| Client-to-Target Context                                          |
|  +--------+                           +-------+       +--------+  |
|  |        |                           |       |       |        |  |
|  | Client |----------Proxied----------| Proxy |-------| Server |  |
|  |        |           flow            |   B   |       |        |  |
|  +--------+                           +-------+       +--------+  |
|                                                                   |
+-------------------------------------------------------------------+
| Client-to-Proxy B Context                                         |
|  +--------+         +-------+         +-------+                   |
|  |        |         |       |         |       |                   |
|  | Client |---------| Proxy |---------| Proxy |                   |
|  |        |         |   A   |         |   B   |                   |
|  +--------+         +-------+         +-------+                   |
|                                                                   |
+-------------------------------------------------------------------+
| Client-to-Proxy A Context                                         |
|  +--------+         +-------+                                     |
|  |        |         |       |                                     |
|  | Client |---------| Proxy |                                     |
|  |        |         |   A   |                                     |
|  +--------+         +-------+                                     |
|                                                                   |
+-------------------------------------------------------------------+
| Proxy A-to-Proxy B Context                                        |
|                     +-------+         +-------+                   |
|                     |       |         |       |                   |
|                     | Proxy |---------| Proxy |                   |
|                     |   A   |         |   B   |                   |
|                     +-------+         +-------+                   |
|                                                                   |
+-------------------------------------------------------------------+
| Proxy B-to-Target Context                                         |
|                                       +-------+       +--------+  |
|                                       |       |       |        |  |
|                                       | Proxy |-------| Server |  |
|                                       |   B   |       |        |  |
|                                       +-------+       +--------+  |
|                                                                   |
+-------------------------------------------------------------------+
```

Figure 5: Diagram of two-hop proxy contexts

Forward proxying, such as the protocols developed in MASQUE, uses both encryption (via TLS) and separation of connections (via proxy hops that see only the next hop) to achieve privacy partitioning.

3.2. Oblivious HTTP and DNS

Oblivious HTTP [OHTTP], developed in the OHAI working group, adds per-message encryption to HTTP exchanges through a relay system. Clients send requests through an Oblivious Relay, which cannot read message contents, to an Oblivious Gateway, which can decrypt the messages but cannot communicate directly with the client or observe client metadata like IP address. Oblivious HTTP relies on Hybrid Public Key Encryption [HPKE] to perform encryption.

Oblivious HTTP uses both encryption and separation of connections to achieve privacy partitioning. The end-to-end messages are encrypted between the Client and Gateway (forming a Client-to-Gateway context), and the connections are separated into a Client-to-Relay context and a Relay-to-Gateway context. It is also important to note that the Relay-to-Gateway connection can be a single connection, even if the Relay has many separate Clients. This provides better anonymity by making the pseudonym presented by the Relay to be shared across many Clients.

```
+-------------------------------------------------------------------+
| Client-to-Target Context                                          |
|  +--------+                           +---------+     +--------+  |
|  |        |                           |         |     |        |  |
|  | Client |---------------------------| Gateway |-----| Target |  |
|  |        |                           |         |     |        |  |
|  +--------+                           +---------+     +--------+  |
|                                                                   |
+-------------------------------------------------------------------+
| Client-to-Gateway Context                                         |
|  +--------+         +-------+         +---------+                 |
|  |        |         |       |         |         |                 |
|  | Client |---------| Relay |---------| Gateway |                 |
|  |        |         |       |         |         |                 |
|  +--------+         +-------+         +---------+                 |
|                                                                   |
+-------------------------------------------------------------------+
| Client-to-Relay Context                                           |
|  +--------+         +-------+                                     |
|  |        |         |       |                                     |
|  | Client |---------| Relay |                                     |
|  |        |         |       |                                     |
|  +--------+         +-------+                                     |
|                                                                   |
+-------------------------------------------------------------------+
```

Figure 6: Diagram of Oblivious HTTP contexts

Oblivious DNS over HTTPS [ODOH] applies the same principle as Oblivious HTTP, but operates on DNS messages only. As a precursor to the more generalized Oblivious HTTP, it relies on the same HPKE cryptographic primatives, and can be analyzed in the same way.

3.3. Privacy Pass

Privacy Pass is an architecture [PRIVACYPASS] and set of protocols being developed in the Privacy Pass working group that allow clients to present proof of verification in an anonymous and unlinkable fashion, via tokens. These tokens originally were designed as a way to prove that a client had solved a CAPTCHA, but can be applied to other types of user or device attestation checks as well. In Privacy Pass, clients interact with an attester and issuer for the purposes of issuing a token, and clients then interact with an origin server to redeeem said token.

In Privacy Pass, privacy partitioning is achieved with cryptographic protection (in the form of blind signature protocols or similar) and separation of connections across two contexts: a "redemption context" between clients an origins (servers that request and receive tokens), and an "issuance context" between clients, attestation servers, and token issuance servers. The cryptographic protection ensures that information revealed during the issuance context is separated from information revealed during the redemption context.

```
+-------------------------------------------------------------------+
| Redemption Context                                                |
|  +--------+         +--------+                                    |
|  |        |         |        |                                    |
|  | Origin |---------| Client |                                    |
|  |        |         |        |                                    |
|  +--------+         +--------+                                    |
|                                                                   |
+-------------------------------------------------------------------+
| Issuance Context                                                  |
|                     +--------+      +----------+      +--------+  |
|                     |        |      |          |      |        |  |
|                     | Client |------| Attester |------| Issuer |  |
|                     |        |      |          |      |        |  |
|                     +--------+      +----------+      +--------+  |
|                                                                   |
+-------------------------------------------------------------------+
```

Figure 7: Diagram of contexts in Privacy Pass

3.4. Privacy Preserving Measurement

The Privacy Preserving Measurement (PPM) working group is chartered to develop protocols and systems that help a data aggregation or collection server (or multiple, non-colluding servers) compute aggregate values without learning the value of any one client's individual measurement. Distributed Aggregation Protocol (DAP) is the primary working item of the group.

At a high level, DAP uses a combination of cryptographic protection (in the form of secret sharing amongst non-colluding servers) to establish two contexts: an "upload context" between clients and non-colluding aggregation servers wherein aggregation servers possibly learn client identity but nothing about their individual measurement reports, and a "collect context" wherein a collector learns aggregate measurement results and nothing about individual client data.

```
+-------------------------------------+--------------------+
| Upload Context                      | Collect Context    |
|                     +------------+  |                    |
|              +------>   Helper   |  |                    |
| +--------+   |      +------^-----+  |                    |
| |        +---+             |        |   +-----------+    |
| | Client |                 |        |   | Collector |    |
| |        +---+             |        |   +-----+-----+    |
| +--------+   |      +------V-----+  |         |          |
|              +------>   Leader   <------------+          |
|                     +------------+  |                    |
+-------------------------------------+--------------------+
```

Figure 8: Diagram of contexts in DAP

4. Applying Privacy Partioning

Applying privacy partitioning to an existing or new system or protocol requires the following steps:

1. Identify the types of information used or exposed in a system or protocol, some of which can be used to identify a user or correlate to other contexts.

2. Partition data to minimize the amount of user-identifying or correlatable information in any given context to only include what is necessary for that context, and prevent sharing of data across contexts wherever possible.

The most impactful types of information to partition are (a) user identity or identities (such as an account name or IP address) that can be linked and (b) user data (such as the content a user is accessing), which can be often sensitive when combined with user identity. Note that user data can itself be user-identifying, in which case it should be treated as an identifier. For example, Oblivious DoH and Oblivious HTTP partition the client IP address and client request data into separate contexts, thereby ensuring that no entity beyond the client can observe both. Collusing across contexts may reverses this partition process, but can also promote non-user-identifying information to user-identifying. For example, in CONNECT proxy systems that use QUIC, the QUIC connection ID is inherently non-user-identifying since it is generated randomly [QUIC], Section 5.1. However, if combined with another context that has user-identifying information such as the client IP address, the QUIC connection ID can become user-identifying information.

This partitioning process can be applied incorrectly or incompletely. Contexts may contain more user-identifying information than desired, or some information in a context may be more user-identifying than intended. Moreover, splitting user-identifying information over multiple contexts has to be done with care, as creating more contexts can increase the number of entities that need to be trusted to not collude. Nevertheless, partitions can help improve the client's privacy posture when applied carefully.

Evaluating and qualifying the resulting privacy of a system or protocol that applies privacy partitioning depends on the contexts that exist and types of user-identifying information in each context. Such evaluation is helpful for identifying ways in which systems or protocols can improve their privacy posture. For example, consider DNS-over-HTTPS [DOH], which produces a single context which contains both the client IP address and client query. One application of privacy partitioning results in ODoH, which produces two contexts, one with the client IP address and the other with the client query.

Recognizing potential appliations of privacy partitoning requires identifying the contexts in use, the information exposed in a context, and the intent of information exposed in a context. Unfortunately, determing what information to include in a given context is a nontrivial task. In principle, the information contained in a context should be fit for purpose. As such, new systems or protocols developed should aim to ensure that all information exposed in a context serves as few purposes as possible. Designing with this principle from the start helps mitigate issues that arise if users of the system or protocol inadvertently ossify on the information available in contexts. Legacy systems that have ossified on information available in contexts may be difficult to change in practice. As an example, many existing anti-abuse systems depend on some notion of client identity such as client IP address, coupled with client data, to provide value. Partitioning contexts in these systems such that they no longer see the client identity requires new solutions to the anti-abuse problem.

5. Limits of Privacy Partitioning

Privacy Partitioning aims to increase user privacy, though as stated is not a panacea. The privacy properties depend on numerous factors, including, though not limited to:

- Non-collusion across contexts; and

- The type of information exposed in each context.

We elaborate on each below.

5.1. Violations by Collusion

Privacy partitions ensure that only the client, i.e., the entity which is responsible for partitioning, can link all user-specific information together up to collusion. No other entity individually knows how to link all the user-specific information as long as they do not collude with each other across contexts. This is why non-collusion is a fundamental requirement for privacy partitioning to offer meaningful privacy for end-users.

As an example, consider OHTTP, wherein the Oblivious Relay knows the Client identity but not the Client data, and the Oblivious Gateway knows the Client data but not the Client identity. If the Oblivious Relay and Gateway collude, they can link Client identity and data together for each request and response transaction by simply observing the requests in transit.

It is not currently possible to guarantee with technical protocol measure that two entities are not colluding. However, there are some mitigations that can be applied to reduce the risk of collusion happening in practice:

- Policy and contractual agreements between entities involved in partitioning, to disallow logging or sharing of data, or to require auditing.

- Protocol requirements to make collusion or data sharing more difficult.

- Adding more partitions and contexts, to make it increasingly difficult to collude with enough parties to recover identities.

5.2. Violations by Insufficient Partitioning

It is possible to define contexts that contain more than one type of user-specific information, despite effort to do otherwise. As an example, consider OHTTP used for the purposes of hiding client-identifying information for a browser telemetry system. It is entirely possible for reports in such a telemetry system to contain both client-specific telemetry data, such as information about their specific browser instance, as well as client-identifying inforamtion, such as the client's location or IP address. Even though OHTTP separates the client IP address from the server via a relay, the server still learns this directly from the client.

Other relevant examples of insufficient partitioning include TLS and Encrypted Client Hello (ECH) [I-D.ietf-tls-esni] and VPNs. TLS and ECH use cryptographic protection (encryption) to hide information from unauthorized parties, but both clients and servers (two entities) can link user-specific data to user-specific identity (IP address). Similarly, while VPNs hide identity from end servers, the VPN server has still can see the identity of both the client and server. Applying privacy partitioning would advocate for at least two additional entities to avoid revealing both (identity (who) and user actions (what)) from each involved party.

While straightforward violations of user privacy like this may seem straightforward to mitigate, it remains an open problem to determine whether a certain set of information reveals "too much" about a specific user. There is ample evidence of data being assumed "private" or "anonymous" but, in hindsight, winds up revealing too much information such that it allows one to link back to individual clients; see [DataSetReconstruction] and [CensusReconstruction] for more examples of this in the real world, and see Section 7 for more discussion.

6. Impacts of Partitioning

Applying privacy partitioning to communication protocols lead to a substantial change in communication patterns. For example, instead of sending traffic directly to a service, essentially all user traffic is routed through a set of intermediaries, possibly adding more end-to-end round trips in the process (depending on the system and protocol). This has a number of practical implications, described below.

1. Service operational or management challenges. Information that is traditionally passively observed in the network or metadata that has been unintentionally revealed to the service provider cannot be used anymore for e.g. existing security procedures such as application rate limiting or DDoS mitigation. However,

network management techniques deployed at present often rely on information that is exposed by most traffic but without any guarantees that the information is accurate. Privacy partitioning provides an opportunity for improvements in these management techniques by providing opportunities to actively exchange information with each entity in a privacy-preserving way and requesting exactly the information needed for a specific task or function rather then relying on assumption that are derived on a limited set of unintentionally revealed information which cannot be guaranteed to be present and may disappear any time in future.

2. Varying performance effects. Depending on how context separation is done, privacy partitioning may affect application performance. As an example, Privacy Pass introduces an entire end-to-end round trip to issue a token before it can be redeemed, thereby decreasing perormance. In contrast, while systems like CONNECT proxying may seem like they would regress performance, often times the highly optimized nature of proxy-to-proxy paths leads to improved perforamnce. In general, while performance and privacy tradeoffs are often cast as a zero sum game, in reality this is often not the case.

7. Security Considerations

Section 5 discusses some of the limitations of privacy partitioning in practice. In general, privacy is best viewed as a spectrum and not a binary state (private or not). Applied correctly, partitioning helps improve an end-users privacy posture, thereby making violations harder to do via technical, social, or policy means. For example, side channels such as traffic analysis [I-D.irtf-pearg-website-fingerprinting] or timing analysis are still possible and can allow an unauthorized entity to learn information about a context they are not a participant of. Proposed mitigations for these types of attacks, e.g., padding application traffic or generating fake traffic, can be very expensive and are therefore not typically applied in practice. Nevertheless, privacy partitioning moves the threat vector from one that has direct access to user-specific information to one which requires more effort, e.g., computational resources, to violate end-user privacy.

8. IANA Considerations

This document has no IANA actions.

9. Informative References

[CensusReconstruction] "The Census Bureau's Simulated Reconstruction-Abetted Re-identification Attack on the

2010 Census", n.d., <https://www.census.gov/data/academy/webinars/2021/disclosure-avoidance-series/simulated-reconstruction-abetted-re-identification-attack-on-the-2010-census.html>.

[CONNECT-IP] Pauly, T., Schinazi, D., Chernyakhovsky, A., Kühlewind, M., and M. Westerlund, "IP Proxying Support for HTTP", Work in Progress, Internet-Draft, draft-ietf-masque-connect-ip-03, 27 September 2022, <https://datatracker.ietf.org/doc/html/draft-ietf-masque-connect-ip-03>.

[CONNECT-UDP] Schinazi, D. and L. Pardue, "HTTP Datagrams and the Capsule Protocol", RFC 9297, DOI 10.17487/RFC9297, August 2022, <https://www.rfc-editor.org/rfc/rfc9297>.

[DataSetReconstruction] Narayanan, A. and V. Shmatikov, "Robust De-anonymization of Large Sparse Datasets", 2008 IEEE Symposium on Security and Privacy (sp 2008), DOI 10.1109/sp.2008.33, May 2008, <https://doi.org/10.1109/sp.2008.33>.

[DOH] Hoffman, P. and P. McManus, "DNS Queries over HTTPS (DoH)", RFC 8484, DOI 10.17487/RFC8484, October 2018, <https://www.rfc-editor.org/rfc/rfc8484>.

[HPKE] Barnes, R., Bhargavan, K., Lipp, B., and C. Wood, "Hybrid Public Key Encryption", RFC 9180, DOI 10.17487/RFC9180, February 2022, <https://www.rfc-editor.org/rfc/rfc9180>.

[I-D.ietf-tls-esni] Rescorla, E., Oku, K., Sullivan, N., and C. A. Wood, "TLS Encrypted Client Hello", Work in Progress, Internet-Draft, draft-ietf-tls-esni-15, 3 October 2022, <https://datatracker.ietf.org/doc/html/draft-ietf-tls-esni-15>.

[I-D.irtf-pearg-website-fingerprinting] Goldberg, I., Wang, T., and C. A. Wood, "Network-Based Website Fingerprinting", Work in Progress, Internet-Draft, draft-irtf-pearg-website-fingerprinting-01, 8 September 2020, <https://datatracker.ietf.org/doc/html/draft-irtf-pearg-website-fingerprinting-01>.

[ODOH] Kinnear, E., McManus, P., Pauly, T., Verma, T., and C.A. Wood, "Oblivious DNS over HTTPS", RFC 9230, DOI 10.17487/RFC9230, June 2022, <https://www.rfc-editor.org/rfc/rfc9230>.

[OHTTP] Thomson, M. and C. A. Wood, "Oblivious HTTP", Work in Progress, Internet-Draft, draft-ietf-ohai-ohttp-05, 26

September 2022, <https://datatracker.ietf.org/doc/html/draft-ietf-ohai-ohttp-05>.

[PRIVACYPASS] Davidson, A., Iyengar, J., and C. A. Wood, "The Privacy Pass Architecture", Work in Progress, Internet-Draft, draft-ietf-privacypass-architecture-08, 17 October 2022, <https://datatracker.ietf.org/doc/html/draft-ietf-privacypass-architecture-08>.

[QUIC] Iyengar, J., Ed. and M. Thomson, Ed., "QUIC: A UDP-Based Multiplexed and Secure Transport", RFC 9000, DOI 10.17487/RFC9000, May 2021, <https://www.rfc-editor.org/rfc/rfc9000>.

[RFC6973] Cooper, A., Tschofenig, H., Aboba, B., Peterson, J., Morris, J., Hansen, M., and R. Smith, "Privacy Considerations for Internet Protocols", RFC 6973, DOI 10.17487/RFC6973, July 2013, <https://www.rfc-editor.org/rfc/rfc6973>.

Acknowledgments

TODO acknowledge.

Authors' Addresses

Mirja Kühlewind
Ericsson Research

Email: mirja.kuehlewind@ericsson.com

Tommy Pauly
Apple

Email: tpauly@apple.com

Christopher A. Wood
Cloudflare

Email: caw@heapingbits.net