Workgroup: Network Working Group
Internet-Draft:
draft-pauly-dprive-oblivious-doh-02
Published: 9 October 2020
Intended Status: Standards Track
Expires: 12 April 2021
Authors: E. Kinnear (Apple Inc.) P. McManus (Fastly) T. Pauly (Apple Inc.) C.A. Wood (Cloudflare)

Oblivious DNS Over HTTPS

Abstract

This document describes an extension to DNS Over HTTPS (DoH) that allows hiding client IP addresses via proxying encrypted DNS transactions. This improves privacy of DNS operations by not allowing any one server entity to be aware of both the client IP address and the content of DNS queries and answers.

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 12 April 2021.

Table of Contents

1. Introduction

DNS Over HTTPS (DoH) [RFC8484] defines a mechanism to allow DNS messages to be transmitted in encrypted HTTP messages. This provides improved confidentiality and authentication for DNS interactions in various circumstances.

While DoH can prevent eavesdroppers from directly reading the contents of DNS exchanges, clients cannot send DNS queries and receive answers from servers without revealing their local IP address, and thus information about the identity or location of the client.

Proposals such as Oblivious DNS ([I-D.annee-dprive-oblivious-dns]) increase privacy by ensuring no single DNS server is aware of both the client IP address and the message contents.

This document defines Oblivious DoH, an extension to DoH that permits proxied resolution, in which DNS messages are encrypted so that no DoH server can independently read both the client IP address and the DNS message contents.

This mechanism is intended to be used as one option for resolving privacy-sensitive content in the broader context of Adaptive DNS [I-D.pauly-dprive-adaptive-dns-privacy].

1.1. Specification of Requirements

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.

2. Terminology

This document defines the following terms:

Oblivious Server: A DoH server that acts as either an Oblivious Proxy or Oblivious Target.

Oblivious Proxy: An Oblivious Server that proxies encrypted DNS queries and responses between a client and an Oblivious Target.

Oblivious Target: An Oblivious Server that receives and decrypts encrypted client DNS queries from an Oblivious Proxy, and returns encrypted DNS responses via that same Proxy. In order to provide DNS responses, the Target can be a DNS resolver, be co-located with a resolver, or forward to a resolver.

Throughout the rest of this document, we use the terms Proxy and Target to refer to an Oblivious Proxy and Oblivious Target, respectively.

3. Deployment Requirements

Oblivious DoH requires, at a minimum:

* Two Oblivious Servers, where one can act as a Proxy, and the other can act as a Target.

* Public keys for encrypting DNS queries that are passed from a client through a Proxy to a Target (Section 6). These keys guarantee that only the intended Target can decrypt client queries.

The mechanism for discovering and provisioning the DoH URI Templates and public keys is via parameters added to DNS resource records. The mechanism for discovering the public key is described in Section 5. The mechanism for discovering a DoH URI Template is described in [I-D.pauly-add-resolver-discovery].

4. HTTP Exchange

Unlike direct resolution, oblivious hostname resolution over DoH involves three parties:

1. The Client, which generates queries.

2. The Proxy, which receives encrypted queries from the client and passes them on to a Target.

3. The Target, which receives proxied queries from the client via the Proxy and produces proxied answers.

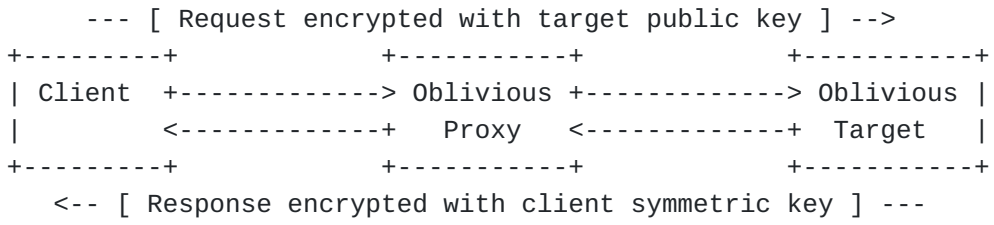

Figure 1: Obvlivious DoH Exchange

4.1. HTTP Request

Oblivious DoH queries are created by the Client, and sent to the Proxy. Requests to the Proxy indicate which DoH server to use as a Target by specifying two variables: "targethost", which indicates the host name of the Target server, and "targetpath", which indicates the path on which the Target's DoH server is running. See Section 4.2 for an example request.

Oblivious DoH messages have no cache value since both requests and responses are encrypted using ephemeral key material. Clients SHOULD prefer using HTTP methods and headers that will prevent unhelpful cache storage of these exchanges (i.e., preferring POST instead of GET).

Clients MUST set the HTTP Content-Type header to "application/oblivious-dns-message" to indicate that this request is an Oblivious DoH query intended for proxying. Clients also SHOULD set this same value for the HTTP Accept header.

Upon receiving a request that contains a "application/oblivious-dns-message" Content-Type, the DoH server looks for the "targethost" and "targetpath" variables. If the variables are not present, then it is the target of the query, and it can decrypt the query (Section 7). If the variables are present, then the DoH server is acting as a Proxy. If it is a proxy, it is expected to send the request on to the Target using the URI template constructed as "https://targethost/targetpath".

4.2. HTTP Request Example

The following example shows how a client requests that a Proxy, "dnsproxy.example.net", forwards an encrypted message to "dnstarget.example.net". The URI template for the Proxy is "https://dnsproxy.example.net/dns-query{?targethost,targetpath}". The URI template for the Target is "https://dnstarget.example.net/dns-query".

```
:method = POST
:scheme = https
:authority = dnsproxy.example.net
:path = /dns-query?targethost=dnstarget.example.net&targetpath=/dns-quer
accept = application/oblivious-dns-message
cache-control = no-cache, no-store
content-type = application/oblivious-dns-message
content-length = 106

<Bytes containing the encrypted payload for an Oblivious DNS query>
```

The Proxy then sends the following request on to the Target:

```
:method = POST
:scheme = https
:authority = dnstarget.example.net
:path = /dns-query
accept = application/oblivious-dns-message
cache-control = no-cache, no-store
content-type = application/oblivious-dns-message
content-length = 106

<Bytes containing the encrypted payload for an Oblivious DNS query>
```

4.3. HTTP Response

The response to an Oblivious DoH query is generated by the Target. It MUST set the Content-Type HTTP header to "application/oblivious-dns-message" for all successful responses. The body of the response contains a DNS message that is encrypted with the client's symmetric key (Section 7).

The response from a Target MUST set the Content-Type HTTP header to "application/oblivious-dns-message" which MUST be forwarded by the Proxy to the Client. A Client MUST only consider a response which contains the Content-Type header in the response before processing the payload. A response without the appropriate header MUST be treated as an error and be handled appropriately. All other aspects of the HTTP response and error handling are inherited from standard DoH.

4.4. HTTP Response Example

The following example shows a 2xx (Successful) response that can be sent from a Target to a client via a Proxy.

```
:status = 200
content-type = application/oblivious-dns-message
content-length = 154

<Bytes containing the encrypted payload for an Oblivious DNS response>
```

Requests that cannot be processed result in 4xx (Client Error) responses.

5. Configuration and Public Key Discovery

In order to use a DoH server as a Target, the client must know a public key to use for encrypting its queries. This key can be discovered using the SVCB or HTTPSSVC record type ([I-D.ietf-dnsop-svcb-https]) for a name owned by the server.

The Service Binding key name is "odohconfig" (Section 12). If present, this key/value pair contains the public key to use when encrypting Oblivious DoH messages that will be targeted at a DoH server. The format of the key is defined in (Section 6).

Clients MUST only use keys that were retrieved from records protected by DNSSEC [RFC4033] to encrypt messages to a Target.

6. Configuration and Public Key Format

An Oblivious DNS public key configuration is a structure encoded, using TLS-style encoding [RFC8446], as follows:

```
struct {
   uint16 kem_id;
   uint16 kdf_id;
   uint16 aead_id;
   opaque public_key<1..2^16-1>;
} ObliviousDoHConfigContents;

struct {
   uint16 version;
   uint16 length;
   select (ObliviousDoHConfig.version) {
      case 0xff02: ObliviousDoHConfigContents contents;
   }
} ObliviousDoHConfig;

ObliviousDoHConfig ObliviousDoHConfigs<1..2^16-1>;
```

The ObliviousDoHConfigs structure contains one or more ObliviousDoHConfig structures in decreasing order of preference. This allows a server to support multiple versions of Oblivious DoH and multiple sets of Oblivious DoH parameters.

An ObliviousDoHConfig contains a versioned representation of an Oblivious DoH configuration, with the following fields.

version The version of Oblivious DoH for which this configuration is used. Clients MUST ignore any ObliviousDoHConfig structure with a version they do not support. The version of Oblivious DoH specified in this document is 0xff02.

length The length, in bytes, of the next field.

contents An opaque byte string whose contents depend on the version. For this specification, the contents are an ObliviousDoHConfigContents structure.

An ObliviousDoHConfigContents contains the information needed to encrypt a message under ObliviousDoHConfigContents.public_key such that only the owner of the corresponding private key can decrypt the message. The values for ObliviousDoHConfigContents.kem_id, ObliviousDoHConfigContents.kdf_id, and ObliviousDoHConfigContents.aead_id are described in [I-D.irtf-cfrg-hpke] Section 7. The fields in this structure are as follows:

kem_id
: The HPKE KEM identifier corresponding to public_key. Clients MUST ignore any ObliviousDoHConfig structure with a key using a KEM they do not support.

kdf_id
: The HPKE KDF identifier corresponding to public_key. Clients MUST ignore any ObliviousDoHConfig structure with a key using a KDF they do not support.

aead_id
: The HPKE AEAD identifier corresponding to public_key. Clients MUST ignore any ObliviousDoHConfig structure with a key using an AEAD they do not support.

public_key
: The HPKE public key used by the client to encrypt Oblivious DoH queries.

7. Protocol Encoding

7.1. Message Format

There are two types of Oblivious DoH messages: Queries (0x01) and Responses (0x02). Both messages carry the following information:

1. A DNS message, which is either a Query or Response, depending on context.

2. Padding of arbitrary length which MUST contain all zeros.

They are encoded using the following structure:

```
struct {
   opaque dns_message<1..2^16-1>;
   opaque padding<0..2^16-1>;
} ObliviousDoHMessagePlaintext;
```

Both Query and Response messages use the ObliviousDoHMessagePlaintext format.

```
ObliviousDoHMessagePlaintext ObliviousDoHQuery;
ObliviousDoHMessagePlaintext ObliviousDoHResponse;
```

An encrypted ObliviousDoHMessagePlaintext is carried in a ObliviousDoHMessage message, encoded as follows:

```
struct {
   uint8  message_type;
   opaque key_id<0..2^16-1>;
   opaque encrypted_message<1..2^16-1>;
} ObliviousDoHMessage;
```

The ObliviousDoHMessage structure contains the following fields:

message_type A one-byte identifier for the type of message. Query messages use message_type 0x01, and Response messages use message_type 0x02.

key_id The identifier of the corresponding ObliviousDoHConfigContents key. This is computed as Expand(Extract("", config), "odoh key id", Nh), where config is the ObliviousDoHConfigContents structure and Extract, Expand, and Nh are as specified by the HPKE cipher suite KDF corresponding to config.kdf_id.

encrypted_message An encrypted message for the Oblivious Target (for Query messages) or client (for Response messages).

The contents of ObliviousDoHMessage.encrypted_message depend on ObliviousDoHMessage.message_type. In particular, ObliviousDoHMessage.encrypted_message is an encryption of a ObliviousDoHQuery if the message is a Query, and ObliviousDoHResponse if the message is a Response.

7.2. Encryption and Decryption Routines

Clients use the following utility functions for encrypting a Query and decrypting a Response as described in Section 8.

encrypt_query_body: Encrypt an Oblivious DoH query.

```
def encrypt_query_body(pkR, key_id, Q_plain):
  enc, context = SetupBaseS(pkR, "odoh query")
  aad = 0x01 || len(key_id) || key_id
  ct = context.Seal(aad, Q_plain)
  Q_encrypted = enc || ct
  return Q_encrypted
```

decrypt_response_body: Decrypt an Oblivious DoH response.

```
def decrypt_response_body(context, Q_plain, R_encrypted):
  key, nonce = derive_secrets(context, Q_plain)
  aad = 0x02 || 0x0000 // 0x0000 represents a 0-length KeyId
  R_plain, error = Open(key, nonce, aad, R_encrypted)
  return R_plain, error
```

The derive_secrets function is described below.

Targets use the following utility functions in processing queries and producing responses as described in Section 9.

setup_query_context: Set up an HPKE context used for decrypting an Oblivious DoH query.

```
def setup_query_context(skR, key_id, Q_encrypted):
  enc || ct = Q_encrypted
  context = SetupBaseR(enc, skR, "odoh query")
  return context
```

decrypt_query_body: Decrypt an Oblivious DoH query.

```
def decrypt_query_body(context, key_id, Q_encrypted):
  aad = 0x01 || len(key_id) || key_id
  enc || ct = Q_encrypted
  Q_plain, error = context.Open(aad, ct)
  return Q_plain, error
```

derive_secrets: Derive keying material used for encrypting an Oblivious DoH response.

```
def derive_secrets(context, Q_plain):
  odoh_secret = context.Export("odoh secret", 32)
  odoh_prk = Extract(Q_plain, odoh_secret)
  key = Expand(odoh_prk, "odoh key", Nk)
  nonce = Expand(odoh_prk, "odoh nonce", Nn)
  return key, nonce
```

encrypt_response_body: Encrypt an Oblivious DoH response.

```
def encrypt_response_body(R_plain, answer_key, answer_nonce):
  aad = 0x02 || 0x0000 // 0x0000 represents a 0-length KeyId
  R_encrypted = Seal(answer_key, answer_nonce, aad, R_plain)
  return R_encrypted
```

8. Oblivious Client Behavior

Let M be a DNS message (query) a client wishes to protect with Oblivious DoH. When sending an Oblivious DoH Query for resolving M to an Oblivious Target with ObliviousDoHConfigContents config, a client does the following:

1. Create an ObliviousDoHQuery structure, carrying the message M and padding, to produce Q_plain.

2. Deserialize config.public_key to produce a public key pkR of type config.kem_id.

3. Compute the encrypted message as Q_encrypted = encrypt_query_body(pkR, key_id, Q_plain), where key_id is as computed in Section 7. Note also that len(key_id) outputs the length of key_id as a two-byte unsigned integer.

4. Output a ObliviousDoHMessage message Q where Q.message_type = 0x01, Q.key_id carries key_id, and Q.encrypted_message = Q_encrypted.

The client then sends Q to the Proxy according to Section 4.1. Once the client receives a response R, encrypted as specified in Section 9, it uses decrypt_response_body to decrypt R.encrypted_message and produce R_plain. Clients MUST validate R_plain.padding (as all zeros) before using R_plain.dns_message.

9. Oblivious Target Behavior

Targets that receive a Query message Q decrypt and process it as follows:

1. Look up the ObliviousDoHConfigContents according to Q.key_id. If no such key exists, the Target MAY discard the query, and if so, it MUST return a 400 (Client Error) response to the Proxy. Otherwise, let skR be the private key corresponding to this public key, or one chosen for trial decryption.

2. Compute context = setup_query_context(skR, Q.key_id, Q.encrypted_message).

3. Compute Q_plain, error = decrypt_query_body(context, Q.key_id, Q.encrypted_message).

4. If no error was returned, and Q_plain.padding is valid (all zeros), resolve Q_plain.dns_message as needed, yielding a DNS message M. Otherwise, if an error was returned or the padding was invalid, return a 400 (Client Error) response to the Proxy.

5. Create an ObliviousDoHResponseBody structure, carrying the message M and padding, to produce R_plain.

6. Compute answer_key, answer_nonce = derive_secrets(context, Q_plain).

7. Compute R_encrypted = encrypt_response_body(R_plain, answer_key, answer_nonce). The key_id field used for encryption is empty, yielding 0x0000 as part of the AAD. Also, the Seal function is that which is associated with the HPKE AEAD.

8. Output a ObliviousDoHMessage message R where R.message_type = 0x02, R.key_id = nil, and R.encrypted_message = R_encrypted.

The Target then sends R in a 2xx (Successful) response to the Proxy according to Section 4.3. The Proxy forwards the message R without modification back to the client as the HTTP response to the client's original HTTP request.

10. Compliance Requirements

In the absence of an application profile standard specifying otherwise, a compliant Oblivious DoH implementation MUST support the following HPKE cipher suite:

*KEM: DHKEM(X25519, HKDF-SHA256) (see [I-D.irtf-cfrg-hpke], Section 7.1)

*KDF: HKDF-SHA256 (see [I-D.irtf-cfrg-hpke], Section 7.2)

*AEAD: AES-128-GCM (see [I-D.irtf-cfrg-hpke], Section 7.3)

11. Security Considerations

DISCLAIMER: this is a work in progress draft and has not yet seen significant security analysis.

Oblivious DoH aims to keep knowledge of the true query origin and its contents known to only clients. As a simplified model, consider a case where there exists two clients C1 and C2, one proxy P, and one target T. Oblivious DoH assumes an extended Dolev-Yao style attacker which can observe all network activity and can adaptively compromise either P or T, but not C1 or C2. Once compromised, the attacker has access to all session information and private key material. (This generalizes to arbitrarily many clients, proxies, and targets, with the constraints that not all targets and proxies are simultaneously compromised, and at least two clients are left uncompromised.) The attacker is prohibited from sending client identifying information, such as IP addresses, to targets. (This would allow the attacker to trivially link a query to the corresponding client.)

In this model, both C1 and C2 send an Oblivious DoH queries Q1 and Q2, respectively, through P to T, and T provides answers A1 and A2. The attacker aims to link C1 to (Q1, A1) and C2 to (Q2, A2), respectively. The attacker succeeds if this linkability is possible without any additional interaction. (For example, if T is compromised, it may return a DNS answer corresponding to an entity it controls, and then observe the subsequent connection from a client, learning its identity in the process. Such attacks are out of scope for this model.)

Oblivious DoH security prevents such linkability. Informally, this means:

1. Queries and answers are known only to clients and targets in possession of the corresponding response key and HPKE keying material. In particular, proxies know the origin and destination of an oblivious query, yet do not know the

plaintext query. Likewise, targets know only the oblivious query origin, i.e., the proxy, and the plaintext query. Only the client knows both the plaintext query contents and destination.

2. Target resolvers cannot link queries from the same client in the absence of unique per-client keys.

Traffic analysis mitigations are outside the scope of this document. In particular, this document does not recommend padding lengths for ObliviousDoHQuery and ObliviousDoHResponse messages. Implementations SHOULD follow the guidance for choosing padding length in [RFC8467].

Oblivious DoH security does not depend on proxy and target indistinguishability. Specifically, an on-path attacker could determine whether a connection a specific endpoint is used for oblivious or direct DoH queries. However, this has no effect on confidentiality goals listed above.

11.1. Denial of Service

Malicious clients (or proxies) may send bogus Oblivious DoH queries to targets as a Denial-of-Service (DoS) attack. Target servers may throttle processing requests if such an event occurs. Additionally, since Targets provide explicit errors upon decryption failure, i.e., if ciphertext decryption fails or if the plaintext DNS message is malformed, Proxies may throttle specific clients in response to these errors.

Malicious Targets or Proxies may send bogus answers in response to Oblivious DoH queries. Response decryption failure is a signal that either the proxy or target is misbehaving. Clients can choose to stop using one or both of these servers in the event of such failure. However, as above, malicious Targets and Proxies are out of scope for the threat model.

11.2. General Proxy Services

Using DoH over anonymizing proxy services such as Tor would also achieve the desired goal of separating query origins from their contents. However, there are several reasons why such systems are undesirable in comparison Oblivious DoH:

1. Tor is also meant as a generic connection-level anonymity system, and thus seems overly complex and costly for the purpose of proxying individual DoH queries. In contrast, Oblivious DoH is a lightweight extension to standard DoH, implemented as an application-layer proxy, that can be enabled as a default mode for users which need increased privacy.

2. As a one-hop proxy, Oblivious DoH encourages connection-less proxies to mitigate client query correlation with few round-trips. In contrast, multi-hop systems such as Tor often run secure connections (TLS) end-to-end, which means that DoH servers could track queries over the same connection. Using a fresh DoH connection per query would incur a non-negligible penalty in connection setup time.

12. IANA Considerations

12.1. Oblivious DoH Message Media Type

This document registers a new media type, "application/oblivious-dns-message".

Type name: application

Subtype name: oblivious-dns-message

Required parameters: N/A

Optional parameters: N/A

Encoding considerations: This is a binary format, containing encrypted DNS requests and responses, as defined in this document.

Security considerations: See this document. The content is an encrypted DNS message, and not executable code.

Interoperability considerations: This document specifies format of conforming messages and the interpretation thereof.

Published specification: This document.

Applications that use this media type: This media type is intended to be used by clients wishing to hide their DNS queries when using DNS over HTTPS.

Additional information: None

Person and email address to contact for further information: See Authors' Addresses section

Intended usage: COMMON

Restrictions on usage: None

Author: IETF

Change controller: IETF

12.2. Oblivious DoH Public Key DNS Parameter

This document adds a parameter ("odohconfig") to the "Service Binding (SVCB) Parameter" registry [I-D.ietf-dnsop-svcb-https]. The allocation request is 32769, taken from the to the First Come First Served range.

If present, the "odohconfig" parameter contains a ObliviousDoHConfigs structure. In wire format, the value of the parameter is an ObliviousDoHConfigs vector, including the redundant length prefix. In presentation format, the value is encoded in [base64].

Name: odohconfig

SvcParamKey: 32769

Meaning: An ObliviousDoHConfigs structure.

Reference: This document.

13. Acknowledgments

This work is inspired by Oblivious DNS [I-D.annee-dprive-oblivious-dns]. Thanks to all of the authors of that document. Thanks to Elliot Briggs, Marwan Fayed, Frederic Jacobs, Tommy Jensen Paul Schmitt, and Brian Swander for the feedback and input.

14. References

14.1. Normative References

[base64] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, DOI 10.17487/RFC4648, October 2006, <https://www.rfc-editor.org/info/rfc4648>.

[I-D.ietf-dnsop-svcb-https] Schwartz, B., Bishop, M., and E. Nygren, "Service binding and parameter specification via the DNS (DNS SVCB and HTTPS RRs)", Work in Progress, Internet-Draft, draft-ietf-dnsop-svcb-https-01, 13 July 2020, <http://www.ietf.org/internet-drafts/draft-ietf-dnsop-svcb-https-01.txt>.

[I-D.irtf-cfrg-hpke] Barnes, R., Bhargavan, K., Lipp, B., and C. Wood, "Hybrid Public Key Encryption", Work in Progress, Internet-Draft, draft-irtf-cfrg-hpke-05, 30 July 2020, <http://www.ietf.org/internet-drafts/draft-irtf-cfrg-hpke-05.txt>.

[I-D.pauly-add-resolver-discovery]
Pauly, T., Kinnear, E., Wood, C., McManus, P., and T. Jensen, "Adaptive DNS Resolver Discovery", Work in Progress, Internet-Draft, draft-pauly-add-resolver-discovery-01, 13 July 2020, <http://www.ietf.org/internet-drafts/draft-pauly-add-resolver-discovery-01.txt>.

[I-D.pauly-dprive-adaptive-dns-privacy]
Kinnear, E., Pauly, T., Wood, C., and P. McManus, "Adaptive DNS: Improving Privacy of Name Resolution", Work in Progress, Internet-Draft, draft-pauly-dprive-adaptive-dns-privacy-01, 1 November 2019, <http://www.ietf.org/internet-drafts/draft-pauly-dprive-adaptive-dns-privacy-01.txt>.

[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>.

[RFC4033] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "DNS Security Introduction and Requirements", RFC 4033, DOI 10.17487/RFC4033, March 2005, <https://www.rfc-editor.org/info/rfc4033>.

[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/info/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/info/rfc8446>.

[RFC8467] Mayrhofer, A., "Padding Policies for Extension Mechanisms for DNS (EDNS(0))", RFC 8467, DOI 10.17487/RFC8467, October 2018, <https://www.rfc-editor.org/info/rfc8467>.

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

14.2. Informative References

[I-D.annee-dprive-oblivious-dns]
Edmundson, A., Schmitt, P., Feamster, N., and A. Mankin, "Oblivious DNS - Strong Privacy for DNS Queries", Work in Progress, Internet-Draft, draft-annee-dprive-oblivious-dns-00, 2 July 2018, <http://www.ietf.org/internet-drafts/draft-annee-dprive-oblivious-dns-00.txt>.

Authors' Addresses

Eric Kinnear
Apple Inc.
One Apple Park Way
Cupertino, California 95014,
United States of America

Email: ekinnear@apple.com

Patrick McManus
Fastly

Email: mcmanus@ducksong.com

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

Email: tpauly@apple.com

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

Email: caw@heapingbits.net