

DPRIVE
Internet-Draft
Intended status: Informational
Expires: May 7, 2020

J. Livingood
Comcast
A. Mayrhofer
nic.at GmbH
B. Overeinder
NLnet Labs
November 04, 2019

DNS Privacy Requirements for Exchanges between Recursive Resolvers and
Authoritative Servers
draft-lmo-dprive-phase2-requirements-01

Abstract

This document provides requirements for adding confidentiality to DNS exchanges between recursive resolvers and authoritative servers.

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 May 7, 2020.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of

the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction & Scope	2
2. Document Development	3
3. Terminology	3
4. Threat Model and Problem Statement	3
5. Perspectives and Use Cases	4
5.1. The User Perspective and Use Cases	4
5.2. The Operator Perspective and Use Cases	5
5.3. The Implementor / Software Vendor Perspective and Use Cases	7
6. Preliminary Requirements	7
6.1. Mandatory Requirements (Proposed)	7
6.2. Optional Requirements (Proposed)	8
6.3. Working Group Discussion Needed	8
6.4. Prioritization of Requirements	9
6.5. Opportunistic Upgrade to Encryption	9
6.6. Detection of Availability	10
6.7. Resistance to Downgrade Attack	10
6.8. End-User Policy Propagation	11
6.9. Performance and Efficiency	12
7. Security Considerations	12
8. IANA Considerations	12
9. Changelog	12
9.1. lmo-dprive-phase2-requirements-00	12
10. References	12
10.1. Normative References	12
10.2. Informative References	13
10.3. URIs	14
Acknowledgments	14
Authors' Addresses	14

1. Introduction & Scope

The 2018 approved charter of the IETF DPRIVE Working Group [1] contains milestones related to confidentiality aspects of DNS transactions between the iterative resolver and authoritative name servers.

This is also reflected in the DPRIVE milestones [2], which (as of October 2019) contains two relevant milestones:

Develop requirements for adding confidentiality to DNS exchanges between recursive resolvers and authoritative servers (unpublished document).

Investigate potential solutions for adding confidentiality to DNS exchanges involving authoritative servers (Experimental).

This document intends to cover the first milestone for defining requirements for adding confidentiality to DNS exchanges between recursive resolvers and authoritative servers. This may in turn lead to progress in investigating, developing and standardizing potential experimental methods of meeting those requirements.

The motivation for this work is to extend the confidentiality methods used between a user's stub resolver and a recursive resolver to the recursive queries sent by recursive resolvers in response to a DNS lookup (when a cache miss occurs and the server must perform recursion to obtain a response to the query). A recursive resolver will send queries to root servers, to Top Level Domain (TLD) servers, to authoritative second level domain servers and potentially to other authoritative DNS servers and each of these query/response transactions presents an opportunity to extend the confidentiality of user DNS queries.

2. Document Development

TEMPORARY SECTION - WILL BE REMOVED BEFORE PUBLISHING The authors are working on this document via GitHub at <https://github.com/alex-nicat/ietf-dprive-phase2-requirements/>. Feedback via pull requests and issues are invited there. The authors plan to continue developing the document in the lead up to IETF-106, after the draft cut-off date.

3. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

This document also makes use of DNS Terminology defined in [RFC8499]

4. Threat Model and Problem Statement

Currently, potentially privacy-protective protocols such as DoT provide encryption between the user's stub resolver and a recursive resolver. This provides (1) protection from observation of end user DNS queries and responses as well as (2) protection from on-the-wire modification DNS queries or responses (including potentially forcing a downgrade to an unencrypted communication). Of course, observation and modification are still possible when performed by the recursive

resolver, which decrypts queries, serves a response from cache or performs recursion to obtain a response (or synthesizes a response), and then encrypts the response and sends it back to the user's stub resolver.

But observation and modification threats still exist when a recursive resolver must perform DNS recursion, from the root to TLD to authoritative servers. This document specifies requirements for filling those gaps.

5. Perspectives and Use Cases

The DNS resolving process involves several entities. These entities have different interests/requirements, and hence it does make sense to examine the interests of those entities separately - though in many cases their interests are aligned. Four different entities can be identified, and their interests are described in the following sections:

- o Users
- o Operators
- o Implementors / Software Developers
- o Researchers

5.1. The User Perspective and Use Cases

The privacy and confidentiality of Users (that is, users as in clients of recursive resolvers, which in turn forward/resolve the user's DNS requests by contacting authoritative servers) can be improved in several ways. We call this "minimisation of exposure", and there are currently three ways to reduce that exposure:

- o Qname minimisation [RFC7816], reducing the amount of information which is absolutely necessary to resolve a query
- o Aggressive NSEC/local auth cache [RFC8198], reducing the amount of outgoing queries in the first place
- o Encryption, removing exposure of information while in transit

As recursors typically forwards queries received from the user to authoritative servers. This creates a transitive trust between the user and the recursor, as well as the authoritative server, since information created by the user is exposed to the authoritative

server. However, the user has never a chance to identify which data was exposed to which authoritative party (via which path).

Also, Users would want to be informed about the status of the connections which were made on their behalf, which adds a fourth point

Encryption/privacy status signaling

TODO: Actual requirements - what do users "want"? Start below:

5.2. The Operator Perspective and Use Cases

Operators of authoritative services have to provide stable and fast DNS services, and interact with a wide range of clients, not all of them authoritative servers. The operator side actually consists of two sides:

- o The "upstream" facing side of recursive resolvers
- o The "downstream" side of authoritative servers

Those two sides are typically operated by different entities, but many entities operate "both sides". Even though that is discouraged (*TODO* source), the two sides might even be operated on the same nameserver.

- o Maybe different technical perspectives for operators
 - * Intelligence (sharing information)
 - * SLD popularity for marketing
- o Focus initially on Second Level Domains (SLDs) initially
 - * Is there a difference for TLDs vs. SLDs from a "protocol" perspective?
- o Monitoring and aggregated data analysis
- o Signaling provisioning information
 - * New record type for finding authoritative server key and authentication? Use SRV? (Being able to use different servers for serving up DNS-over-{TCP,UDP} vs DNS-over-TLS responses may be valuable.

- * Signal secure transport details (DNS-over-TLS, DNS-over-QUIC, EncryptedSNI, connectionless, etc.), perhaps in an extensible manner? Minimize RTTs and reduce need for trials.
 - * Large provider use cases where the NS names are out of bailiwick for the zone (e.g. small number of distinct NS records serving 100k+ zones)
 - o EDNS client subnet (JL: Not sure ECS crosses the cost/benefit threshold to be included as a requirement and many CDNs that run auth servers will likely say ECS is quite operationally important)
 - o Decide between TLS and connectionless (such as COSE-based messages)
 - o Costs of TLS connection vs. connectionless
 - * Technical solution, e.g. encryption of the DNS query, shouldn't enable an attack vector for DDoS or resource exhaustion. For example, only if the client uses DNS-over-TLS, the upstream query to the authoritative will be over DNS-over-TLS also. If the client uses UDP, the resolver won't invest resources in DNS-over-TLS to prevent a potential resource exhaustion attack.
 - * Reuse connection state (if any) and examine resumption considerations
 - * Minimize server-side state (eg, with session tickets)
 - * Need empirical studies on capacity, traffic, attack vectors
 - * Evaluate impact on architecture and footprint expansion
 - * Analyze optimal persistent connection time/time-out
 - * Analyze optimal number of persistent connections recursive resolvers should maintain
 - * Consider operational concerns with respect to capabilities signaling
 - * Develop a profile that has operational advantages for operators
- *TODO*: Actual requirements - what do operators "want"?

5.3. The Implementor / Software Vendor Perspective and Use Cases

Implementer requirements follows requirements from user and operator perspectives:

- o Non-functional requirements, e.g. diversity of implementations
- o Horizontal vs. vertical scaling, for example similar to http servers
- o Use of DANE [RFC6698] for authentication: strict vs. opportunistic
- o Incremental deployment
- o Cache reuse vs. downgrade? Does the cache need to be partitioned? When can an in-cache answer retrieved via cleartext be served encrypted to a recursive query?
- o (Use of TCP fast open) - but this might be a requirement for the actual encryption protocol

TODO: Actual requirements of implementors - essentially, they follow what Operators need?

6. Preliminary Requirements

The requirements of different interested stakeholders are outlined below. The parenthetical risks and priority levels are intended only to spur discussion. But at a high level the requirements may be summarized as follows:

6.1. Mandatory Requirements (Proposed)

1. Each implementing party should be able to independently take incremental steps to meet requirements without the need for close coordination (e.g. loosely coupled) (low risk, high priority)
2. Implement DoT between a recursive resolver and single level domain authoritative servers (high risk, high priority)
3. Implement DNS privacy protections between a recursive resolver and TLD servers (low risk, low priority)
4. Implement DNS privacy protections between a recursive resolver and the root servers (low risk, low priority)
5. Implement DoT or other DNS privacy protections in a manner that enables operators to perform appropriate performance and security

monitoring, conduct relevant research, etc. (high risk, high priority)

6. Implement QNAME minimisation in all steps of recursion (medium risk, medium priority)
7. The legacy unencrypted DNS protocol (e.g. UDP/TCP port 53) MUST be supported in parallel to DoT (high risk, high priority)
8. Recursive resolvers SHOULD opportunistically upgrade recursive query transmissions to DoT when an authoritative server is detected to support DoT (high risk, high priority)
9. TLS 1.3 (or later versions) MUST be supported and downgrades from TLS 1.3 to prior versions MUST not occur.

6.2. Optional Requirements (Proposed)

1. Implement DoT between a recursive resolver and TLD servers (low risk, low priority)
2. Implement DoT between a recursive resolver and the root servers (low risk, low priority)
3. DNSSEC validation SHOULD be performed
4. Users SHOULD have a method for signaling their preferences for (1) exclusively using DNS privacy & encryption, (2) preferring DNS privacy & encryption but falling back to un-encrypted DNS as needed, (3) exclusively using un-encrypted DNS, or other preferences. (Possible reference to DNSSEC DO bit?)
5. Authoritative domain administrators SHOULD have a method for signaling their preferences for (1) exclusively using DNS privacy & encryption, (2) preferring DNS privacy & encryption but falling back to un-encrypted DNS as needed, (3) exclusively using un-encrypted DNS, or other preferences. (Possible reference to DNSSEC DO bit?)

6.3. Working Group Discussion Needed

- o Provisioning impacts – operators and vendors say implementation must be zero-provisioning. What does that mean and how should that be articulated as a requirement?
- o Signaling: Provide some method to signal not just binary support DoT / do not support to allow for certain QTYPES or whatever to use DoT while others may not (e.g. an auth server may want to say

in high load that some low risk or low priority queries fallback to unencrypted comms). Is this signaling or negotiation? Perhaps the requirement is ultimately about "Load Shedding" or "Load Management".

- o Trust anchor/authority: Should this depend only on the DNS, such as DANE, or Certification Authorities? See discussion at <https://github.com/alex-nicat/ietf-dprive-phase2-requirements/issues/13>
- o Rather than say DNS privacy methods should we specifically say no ECS (or not fine-grained ECS), and to do QNAME minimization?
- o There is a new signaling draft at <https://tools.ietf.org/html/draft-levine-dprive-signal-00> and a prior one at <https://tools.ietf.org/html/draft-bortzmeyer-dprive-step-2-05> - are these informative for our requirements?
- o Is signaling good and/or necessary.

6.4. Prioritization of Requirements

The preliminary requirements above each have varying levels of risk and so can be prioritized based on that risk. As a result, the highest risk area is the one that involves the greatest potential for surveillance and modification based on the details of the specific step of recursion. This suggests the highest risk and thus highest priority is between a recursive server and first level authoritative server. Lower risks are to TLDs and root servers, with correspondingly lower priority. Support for monitoring and compliance are also high risk since this is operationally critical, and thus should also be considered high priority.

6.5. Opportunistic Upgrade to Encryption

Opportunistically upgrading to use encryption may be the most viable path to deploy new DNS encryption protocols. This may enable deployment to occur incrementally and without tightly coupled coordination across a diverse global group of very different potential implementors.

EDITORIAL NOTE: This paragraph may be unnecessary and could be cut. The exact method by which a recursive resolver determines whether an authoritative server supports DoT has not been specified in this document. But it seems reasonable to imagine that a recursive server might be able to probe authoritative servers on TCP/853 using the DoT protocol and then build a cached list of servers that support DoT so that subsequent queries will upgrade to use DoT (and can fallback if

DoT connections subsequently fail). It seems also possible to imagine a method might exist for an authoritative domain to use a TBD resource record or other method to specify whether DoT is supported.

6.6. Detection of Availability

EDITORIAL NOTE: This section was just moved up. May need some better integration later on.

Recursive resolvers typically communicate with many authoritative nameservers. Not every authoritative nameserver will support DoT and not every recursive resolver will support every requirement. How should a recursive resolver determine whether DoT is supported for example? (There may be multiple ways, or none)

What scope/granularity should such an availability marker have?

- o by zone ("all authoritative nameservers in the "example.net" zone support private queries from resolvers")
- o by identified nameserver ("the nameserver "a.ns.example.net" supports private queries from resolvers")
- o by IP address ("any nameservers that resolve to 192.0.2.13 support private queries from resolvers")

Note that if there is no signal for availability, recursors could still opportunistically try the DNS privacy mechanism, as this is employed by some stub resolvers when they contact their designated recursors.

Should a signal of availability also indicate a preference for privacy over availability? i.e., are there distinct ways to signal "DNS-privacy is available" separately from "Only contact this server via DNS-privacy if you understand this signal (though we may continue to support non-private DNS queries for clients that don't understand it)".

6.7. Resistance to Downgrade Attack

When a connection is opportunistically upgraded to DoT, if a fallback to unencrypted DNS can be possible via a downgrade attack by blocking or modifying TCP/853 communications. In such cases, it may be best to establish a mechanism whereby the authoritative domain can specify their preferred behaviour. This may range from only use DoT and do not fallback to unencrypted DNS, to opportunistically use DoT but fallback in failure, to do not use DoT. The email application layer protocols have similar methods for asserting how email from a

particular domain should be treated, so following some of the lessons learned there is likely a good idea. Compare HSTS [RFC6797]?

6.8. End-User Policy Propagation

EDITORIAL NOTE: This section was just moved up. May need some better integration later on.

Like any multi-party protocol (e.g. SMTP), the end user's preferences or policies might or might not be respected by later hops in the chain. But if we have a way to express those preferences, we offer cooperating resolvers at least an opportunity to respect them.

WG DISCUSS: Is it better to let auth domains assert whether fallback should be permitted or is that an end user preference or both? The email world might suggest the former while the DNSSEC world the latter. Or specify the standardization of the preferences and their communication and leave it to implementors to decide whether or how to treat those signals?

What sorts of preferences or policy might an end-user want to express? for example:

- o do not identify my general location (e.g. don't send my subnet information (ECS) [RFC7871] data about me when talking to authoritative servers), accepting that reduced localization may result in less localized responses from authoritative Content Delivery Network (CDN) servers and thus slower access to content
- o prefer DNS privacy over reduced latency (i.e., do not try to do speedups - try opportunistic privacy first and fall back to cleartext only if that fails)
- o never do non-private authoritative queries on my behalf (for any external queries you need to do to resolve this request, require strict, well-authenticated DNS privacy)

How specifically are these preferences be expressed by the client? (e.g. new EDNS0 [RFC6891] options?) Should the recursor have a way to indicate whether:

- o they are capable of honoring them?
- o they intend to honor them?
- o they did honor them over the course of a specific lookup?

If a resolver merely forwards a request to another recursor, should it also propagate those preferences/policy? if so, how?

This seems similar to [I-D.ietf-uta-smtp-require-tls].

To implement end-user policies, support for signaling of DNS server capabilities is helpful, see for example [I-D.edmonds-dnsop-capabilities].

6.9. Performance and Efficiency

- o Can authoritative server operators limit resource-exhaustion attacks against private DNS mechanisms from having an impact on traditional (non-private) authoritative DNS availability? (JL: seems easy to implement per host connection limits and implement other standard DDoS protections – again for a later BCP doc)
- o What are best practices for authoritative server operators that can minimize latency and unavailability?
- o What are best practices for recursors?

7. Security Considerations

At this point of the document, the authors have not yet discussed security considerations in detail, as the whole document tends to deal with user privacy, which can be considered part of security. :)

8. IANA Considerations

This document has no actions for IANA.

9. Changelog

Note to RFC editor: Remove this entire section before publication.

9.1. lmo-dprive-phase2-requirements-00

Initial version

10. References

10.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.

- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8499] Hoffman, P., Sullivan, A., and K. Fujiwara, "DNS Terminology", BCP 219, RFC 8499, DOI 10.17487/RFC8499, January 2019, <<https://www.rfc-editor.org/info/rfc8499>>.

10.2. Informative References

- [I-D.edmonds-dnsop-capabilities]
Edmonds, R., "Signaling DNS Capabilities", draft-edmonds-dnsop-capabilities-00 (work in progress), July 2017.
- [I-D.ietf-uta-smtp-require-tls]
Fenton, J., "SMTP Require TLS Option", draft-ietf-uta-smtp-require-tls-09 (work in progress), August 2019.
- [RFC6698] Hoffman, P. and J. Schlyter, "The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA", RFC 6698, DOI 10.17487/RFC6698, August 2012, <<https://www.rfc-editor.org/info/rfc6698>>.
- [RFC6797] Hodges, J., Jackson, C., and A. Barth, "HTTP Strict Transport Security (HSTS)", RFC 6797, DOI 10.17487/RFC6797, November 2012, <<https://www.rfc-editor.org/info/rfc6797>>.
- [RFC6891] Damas, J., Graff, M., and P. Vixie, "Extension Mechanisms for DNS (EDNS(0))", STD 75, RFC 6891, DOI 10.17487/RFC6891, April 2013, <<https://www.rfc-editor.org/info/rfc6891>>.
- [RFC7816] Bortzmeyer, S., "DNS Query Name Minimisation to Improve Privacy", RFC 7816, DOI 10.17487/RFC7816, March 2016, <<https://www.rfc-editor.org/info/rfc7816>>.
- [RFC7871] Contavalli, C., van der Gaast, W., Lawrence, D., and W. Kumari, "Client Subnet in DNS Queries", RFC 7871, DOI 10.17487/RFC7871, May 2016, <<https://www.rfc-editor.org/info/rfc7871>>.
- [RFC8198] Fujiwara, K., Kato, A., and W. Kumari, "Aggressive Use of DNSSEC-Validated Cache", RFC 8198, DOI 10.17487/RFC8198, July 2017, <<https://www.rfc-editor.org/info/rfc8198>>.

10.3. URIs

[1] <https://datatracker.ietf.org/doc/charter-ietf-dprive/>

[2] <https://datatracker.ietf.org/wg/dprive/about/>

Acknowledgments

TODO

Authors' Addresses

Jason Livingood
Comcast

Email: Jason_Livingood@comcast.com

Alexander Mayrhofer
nic.at GmbH

Email: alex.mayrhofer.ietf@gmail.com

Benno Overeinder
NLnet Labs

Email: benno@NLnetLabs.nl

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: May 4, 2020

E. Kinnear
T. Pauly
C. Wood
Apple Inc.
P. McManus
Fastly
November 01, 2019

Adaptive DNS: Improving Privacy of Name Resolution
draft-pauly-dprive-adaptive-dns-privacy-01

Abstract

This document defines an architecture that allows clients to dynamically discover designated resolvers that offer encrypted DNS services, and use them in an adaptive way that improves privacy while co-existing with locally provisioned resolvers. These resolvers can be used directly when looking up names for which they are designated. These resolvers also provide the ability to proxy encrypted queries, thus hiding the identity of the client requesting resolution.

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 May 4, 2020.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents

carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
1.1. Specification of Requirements	4
2. Terminology	4
3. Client Behavior	5
3.1. Discovering Designated DoH Servers	6
3.1.1. Whitelisting Designated DoH Servers	7
3.1.2. Accessing Extended Information	8
3.2. Discovering Local Resolvers	9
3.3. Hostname Resolution Algorithm	10
3.4. Oblivious Resolution	11
3.5. Handling Network Changes	12
4. Server Requirements	12
4.1. Provide a DoH Server	12
4.1.1. Oblivious DoH Proxy	12
4.1.2. Oblivious DoH Target	13
4.1.3. Keying Material	13
4.2. Advertise the DoH Server	13
4.3. Provide Extended Configuration as a Web PvD	13
5. Server Deployment Considerations	15
5.1. Single Content Provider	15
5.2. Multiple Content Providers	15
5.3. Avoid Narrow Deployments	16
6. Local Resolver Deployment Considerations	16
6.1. Designating Local DoH Servers	16
6.2. Local Use Cases	17
6.2.1. Accessing Local-Only Resolvable Content	17
6.2.2. Accessing Locally Optimized Content	18
6.2.3. Walled-Garden and Captive Network Deployments	19
6.2.4. Network-Based Filtering	19
7. Performance Considerations	20
8. Security Considerations	21
9. Privacy Considerations	21
10. IANA Considerations	22
10.1. DoH Template PvD Key	22
10.2. DNS Filtering PvD Keys	22
10.3. DoH URI Template DNS Parameter	23
11. Acknowledgments	23
12. References	23
12.1. Normative References	23
12.2. Informative References	24

Authors' Addresses	25
------------------------------	----

1. Introduction

When clients need to resolve names into addresses in order to establish networking connections, they traditionally use by default the DNS resolver that is provisioned by the local network along with their IP address [RFC2132] [RFC8106]. Alternatively, they can use a resolver indicated by a tunneling service such as a VPN.

However, privacy-sensitive clients might prefer to use an encrypted DNS service other than the one locally provisioned in order to prevent interception, profiling, or modification by entities other than the operator of the name service for the name being resolved. Protocols that can improve the transport security of a client when using DNS or creating TLS connections include DNS-over-TLS [RFC7858], DNS-over-HTTPS [RFC8484], and encrypted Server Name Indication (ESNI) [I-D.ietf-tls-esni].

There are several concerns around a client using such privacy-enhancing mechanisms for generic system traffic. A remote service that provides encrypted DNS may not provide correct answers for locally available resources, or private resources (such as domains only accessible over a private network). Remote services may also be untrusted from a privacy perspective: while encryption will prevent on-path observers from seeing hostnames, client systems need to trust the encrypted DNS service to not store or misuse queries made to it. Further, extensive use of cloud based recursive resolvers obscures the network location of the client which may degrade the performance of the returned server due to lack of proximity at the benefit of improved privacy.

Client systems are left with choosing between one of the following stances:

1. Send all application DNS queries to a particular encrypted DNS service, which requires establishing user trust of the service. This can lead to resolution failures for local or private enterprise domains absent heuristics or other workarounds for detecting managed networks.
2. Allow the user or another entity to configure local policy for which domains to send to local, private, or encrypted resolvers. This provides more granularity at the cost of increasing user burden.
3. Only use locally-provisioned resolvers, and opportunistically use encrypted DNS to these resolvers when possible. (Clients may

learn of encrypted transport support by actively probing such resolvers.) This provides marginal benefit over not using encrypted DNS at all, especially if clients have no means of authenticating or trusting local resolvers.

This document defines an architecture that allows clients to improve the privacy of their DNS queries without requiring user intervention, and allowing coexistence with local, private, and enterprise resolvers.

This architecture is composed of several mechanisms:

- o A DNS record that indicates a designated DoH server associated with a name (Section 3.1);
- o an extension to DoH that allows client IP addresses to be disassociated from queries via proxying ([I-D.pauly-dprive-oblivious-doh]);
- o a DoH server that responds to queries directly and supports proxying (Section 4);
- o and client behavior rules for how to resolve names using a combination of designated DoH resolvers, proxied queries, and local resolvers (Section 3).

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:

Adaptive DNS: Adaptive DNS is a technique to provide an encrypted transport for DNS queries that can be sent directly to a Designated DoH Server, to use Oblivious DoH to hide the client IP address, or to use Direct Resolvers when required or appropriate.

Designated DoH Server: A DNS resolver that provides connectivity over HTTPS (DoH) that is designated as a responsible resolver for a given domain or zone.

Direct Resolver: A DNS resolver using any transport that is provisioned directly by a local router or a VPN.

Exclusive Direct Resolver: A Direct Resolver that requires the client to use it exclusively for a given set of domains, such as private domains managed by a VPN. This status is governed by local system policy.

Oblivious DoH: A technique that uses multiple DoH servers to proxy queries in a way that disassociates the client's IP address from query content.

Oblivious Proxy: A resolution server that proxies encrypted client DNS queries to another resolution server that will be able to decrypt the query (the Oblivious Target).

Oblivious Target: A resolution server that receives encrypted client DNS queries via an Oblivious Proxy.

Privacy-Sensitive Connections: Connections made by clients that are explicitly Privacy-Sensitive are treated differently from connections made for generic system behavior, such as non-user-initiated maintenance connections. This distinction is only relevant on the client, and does not get communicated to other network entities. Certain applications, such as browsers, can choose to treat all connections as privacy-sensitive.

Web PvD: A Web Provisioning Domain, or Web PvD, represents the configuration of resolvers, proxies, and other information that a server deployment makes available to clients. See Section 4.3.

3. Client Behavior

Adaptive DNS allows client systems and applications to improve the privacy of their DNS queries and connections, both by requiring confidentiality via encryption, and by limiting the ability to correlate client IP addresses with query contents. Specifically, the goal for client queries is to achieve the following properties:

1. No party other than the client and server can learn or control the names being queried by the client or the answers being returned by the server.
2. Only a designated DNS resolver associated with the deployment that is also hosting content will be able to read both the client IP address and queried names for Privacy-Sensitive Connections. For example, a resolver owned and operated by the same provider that hosts "example.com" would be able to link queries for

"example.com" to specific clients (by their IP address), since the server ultimately has this capability once clients subsequently establish secure (e.g., TLS) connections to an address to which "example.com" resolves.

3. Clients will be able to comply with policies required by VPNs and local networks that are authoritative for private domains.

An algorithm for determining how to resolve a given name in a manner that satisfies these properties is described in Section 3.3. Note that this algorithm does not guarantee that responses that are not signed with DNSSEC are valid, and clients that establish connections to unsigned addresses may still expose their local IP addresses to attackers that control their terminal resolver even if hidden during resolution.

3.1. Discovering Designated DoH Servers

All direct (non-oblivious) queries for names in privacy-sensitive connections MUST be sent to a server that both provides encryption and is designated for the domain.

Clients dynamically build and maintain a set of known Designated DoH Servers. The information that is associated with each server is:

- o The URI Template of the DoH server [RFC8484]
- o The public HPKE [I-D.irtf-cfrg-hpke] key of the DoH server used for proxied oblivious queries [I-D.pauly-dprive-oblivious-doh]
- o A list of domains for which the DoH server is designated

This information can be retrieved from several different sources. The primary source for discovering Designated DoH Server configurations is from properties stored in a SVCB (or a SVCB-conformant type like HTTPSSVC) DNS Record [I-D.nygren-dnsop-svc-httpssvc]. This record provides the URI Template and the public Oblivious DoH key of a DoH server that is designated for a specific domain. A specific domain may have more than one such record.

In order to designate a DoH server for a domain, a SVCB record can contain the "dohuri" (Section 10). The value stored in the parameter is a URI, which is the DoH URI template [RFC8484].

The public key of the DoH server is sent as the "odohkey" [I-D.pauly-dprive-oblivious-doh].

The following example shows a record containing a DoH URI, as returned by a query for the HTTPSSVC variant of the SVCB record type on "example.com".

```
example.com.      7200  IN HTTPSSVC 0 svc.example.net.  
svc.example.net.  7200  IN HTTPSSVC 2 svc1.example.net. (  
                                dohuri=https://doh.example.net/dns-query  
                                odohkey="..." )
```

Clients MUST ignore any DoH server URI that was not retrieved from a DNSSEC-signed record that was validated by the client [RFC4033].

Whenever a client resolves a name for which it does not already have a Designated DoH Server, it SHOULD try to determine the Designated DoH Server by sending a query for the an SVCB record for the name. If there is no DoH server designated for the name or zone, signaled either by an NXDOMAIN answer or a SVCB record that does not contain a DoH URI, the client SHOULD suppress queries for the SVCB record for a given name until the time-to-live of the answer expires.

In order to bootstrap discovery of Designated DoH Servers, client systems SHOULD have some saved list of at least two names that they use consistently to perform SVCB record queries on the Direct Resolvers configured by the local network. Since these queries are likely not private, they SHOULD NOT be associated with user action or contain user-identifying content. Rather, the expectation is that all client systems of the same version and configuration would issue the same bootstrapping queries when joining a network for the first time when the list of Designated DoH Servers is empty.

3.1.1. Whitelisting Designated DoH Servers

Prior to using a Designated DoH Server for direct name queries on privacy-sensitive connections, clients MUST whitelist the server.

The requirements for whitelisting are:

- o Support for acting as an Oblivious Proxy. Each Designated DoH Server is expected to support acting as a proxy for Oblivious DoH. A client MUST issue at least one query that is proxied through the server before sending direct queries to the server.
- o Support for acting as an Oblivious Target. Each Designated DoH Server is expected to support acting as a target for Oblivious DoH. A client MUST issue at least one query that is targeted at the server through a proxy before sending direct queries to the server.

Designated DoH Servers are expected to act both as Oblivious Proxies and as Oblivious Targets to ensure that clients have sufficient options for preserving privacy using Oblivious DoH. Oblivious Targets are expected to act as Oblivious Proxies to ensure that no Oblivious DoH server can act as only a target (thus being able to see patterns in name resolution, which might have value to a resolver) and require other servers to take on a disproportionate load of proxying.

Clients MAY further choose to restrict the whitelist by other local policy. For example, a client system can have a list of trusted resolver configurations, and it can limit the whitelist of Designated DoH Servers to configurations that match this list. Alternatively, a client system can check a server against a list of audited and approved DoH Servers that have properties that the client approves.

Clients SHOULD NOT whitelist authority mappings for effective top-level domains (eTLDs), such as ".com".

If a client detects at any point after whitelisting a DoH server that the server no longer meets the criteria for whitelisting, such as consistently failing to proxy or receive Oblivious DoH queries, the client SHOULD remove the DoH server from its whitelist.

3.1.2. Accessing Extended Information

When a Designated DoH Server is discovered, clients SHOULD also check to see if this server provides an extended configuration in the form of a Web PvD (Section 4.3). To do this, the client performs a GET request to the DoH URI, indicating that it accepts a media type of "application/pvd+json" [I-D.ietf-intarea-provisioning-domains]. When requesting the PvD information, the query and fragment components of the requested path are left empty. Note that this is different from a GET request for the "application/dns-message" type, in which the query variable "dns" contains an encoded version of a DNS message.

In response, the server will return the JSON content for the PvD, if present. The content-type MUST be "application/pvd+json".

The following exchange shows an example of a client retrieving a Web PvD configuration for a DoH server with the URI Template "https://dnsserver.example.net/dns-query".

The client sends:

```
:method = GET
:scheme = https
:authority = dnsserver.example.net
:path = /dns-query
accept = application/pvd+json
```

And the server replies:

```
:status = 200
content-type = application/pvd+json
content-length = 175
cache-control = max-age=86400
```

<JSON content of the Web PvD>

If the server does not support retrieving any extended PvD information, it MUST reply with HTTP status code 415 (Unsupported Media Type, [RFC7231]).

If the retrieved JSON contains a "dnsZones" array [I-D.ietf-intarea-provisioning-domains], the client SHOULD perform an SVCB record lookup of each of the listed zones on the DoH server and validate that the DoH server is a designated server for the domain; and if it is, add the domain to the local configuration.

3.2. Discovering Local Resolvers

If the local network provides configuration with an Explicit Provisioning Domain (PvD), as defined by [I-D.ietf-intarea-provisioning-domains], clients can learn about domains for which the local network's resolver is authoritative.

If an RA provided by the router on the network defines an Explicit PvD that has additional information, and this additional information JSON dictionary contains the key "dohTemplate" (Section 10), then the client SHOULD add this DoH server to its list of known DoH configurations. The domains that the DoH server claims authority for are listed in the "dnsZones" key. Clients MUST use an SVCB record from the locally-provisioned DoH server and validate the answer with DNSSEC [RFC4033] before creating a mapping from the domain to the server. Once this has been validated, clients can use this server for resolution as described in step 2 of Section 3.3.

See Section 6 for local deployment considerations.

3.3. Hostname Resolution Algorithm

When establishing a secure connection to a certain hostname, clients need to first determine which resolver configuration ought to be used for DNS resolution.

Several of the steps outlined in this algorithm take into account the success or failure of name resolution. Failure can be indicated either by a DNS response, such as SERVFAIL or NXDOMAIN, or by a connection-level failure, such as a TCP reset, TLS handshake failure, or an HTTP response error status. In effect, any unsuccessful attempt to resolve a name can cause the client to try another resolver if permitted by the algorithm. This is particularly useful for cases in which a name may not be resolvable over public DNS but has a valid answer only on the local network.

Given a specific hostname, the order of preference for which resolver to use SHOULD be:

1. An Exclusive Direct Resolver, such as a resolver provisioned by a VPN, with domain rules that include the hostname being resolved. If the resolution fails, the connection will fail. See Section 3.2 and Section 6.
2. A Direct Resolver, such as a local router, with domain rules that are known to be authoritative for the domain containing the hostname. If the resolution fails, the connection will try the next resolver configuration based on this list.
3. The most specific Designated DoH Server that has been whitelisted (Section 3.1.1) for the domain containing the hostname, i.e., the designated DoH server which is associated with the longest matching suffix of the hostname. For example, given two Designated DoH Servers, one for "foo.example.com" and another "example.com", clients connecting to "bar.foo.example.com" should use the former. Note that the matching MUST be performed on entire labels. That is, if "example.com" has a designated DoH server, it can be used for "foo.example.com", but not for "badexample.com". If the resolution fails, the connection can either use an Oblivious DoH resolver (Step 4) or the default resolver (Step 5). Privacy-Sensitive Connections SHOULD NOT skip Step 4. Other connections MAY skip Step 4, based on system policy.
4. Oblivious DoH queries using multiple DoH Servers ([I-D.pauly-dprive-oblivious-doh]). If this resolution fails, Privacy-Sensitive Connections will fail. All other connections will use the last resort, the default Direct Resolvers.

5. The default Direct Resolver, generally the resolver provisioned by the local router, is used as the last resort for any connection that is not explicitly Privacy-Sensitive [RFC2132] [RFC8106].

If the system allows the user to specify a preferred encrypted resolver, such as allowing the user to manually configure a DoH server URI to use by default, the use of this resolver SHOULD come between steps 2 and 3. This ensures that VPN-managed and locally-accessible names remain accessible while all other names are resolved using the user preference.

Resolution on behalf of system traffic, such as interactions required to detect and access a Captive Network Portal, require the use of the default Direct Resolver. System traffic SHOULD have an exception to this algorithm, and only use Steps 2 and 5 (those that use a resolver provisioned by the local network). Further deployment considerations for captive networks and walled-garden networks can be found in Section 6.2.3.

3.4. Oblivious Resolution

For all privacy-sensitive connection queries for names that do not correspond to a Designated DoH Server, the client SHOULD use Oblivious DoH to help conceal its IP address from eavesdroppers and untrusted resolvers.

Disassociation of client IPs from query content is achieved by using Oblivious DoH [I-D.pauly-dprive-oblivious-doh]. This extension to DoH allows a client to encrypt a query with a target DoH server's public key, and proxy the query through another server. The query is packaged with a unique client-defined symmetric key that is used to sign the DNS answer, which is sent back to the client via the proxy.

All DoH Servers that are used as Designated DoH Servers by the client MUST support being both an Oblivious Proxy and an Oblivious Target, as described in the server requirements (Section 4).

Since each Designated DoH Server can act as one of two roles in an proxied exchange, there are $(N) * (N - 1) / 2$ possible pairs of servers, where N is the number of whitelisted servers. While clients SHOULD use a variety of server pairs in rotation to decrease the ability for any given server to track client queries, it is not expected that all possible combinations will be used. Some combinations will be able to handle more load than others, and some will have better latency properties than others. To optimize performance, clients SHOULD maintain statistics to track the performance characteristics and success rates of particular pairs.

Clients that are performing Oblivious DoH resolution SHOULD fall back to another pair of servers if a first query times out, with a locally-determined limit for the number of fallback attempts that will be performed.

3.5. Handling Network Changes

Whenever a client joins a new network, it SHOULD wait to receive local configuration for resolvers before using any Designated DoH servers. The local network might be authoritative for some names, or might require filtering.

Once the local configuration of the new network has been received, the client MAY use Designated DoH configuration that it discovered when associated with another network. These configurations can still be considered valid since they came from DNSSEC-signed records. However, it is possible that different resolver IP addresses would be returned when looking up the designated server on the new network, which can provide a more optimal route through the Internet, so clients SHOULD perform new queries to refresh their mappings by making queries on connection on this new interface.

4. Server Requirements

Any server deployment that provides a set of services within one or more domains, such as a CDN, can run a server node that allows clients to run Adaptive DNS. A new server node can be added at any time, and can be used once it is advertised to clients and can be validated and whitelisted. The system overall is intended to scale and provide improved performance as more nodes become available.

The basic requirements to participate as a server node in this architecture are described below.

4.1. Provide a DoH Server

Each server node is primarily defined by a DoH server [RFC8484] that is designated for a set of domains, and also provides Oblivious DoH functionality. As such, the DoH servers MUST be able to act as recursive resolvers that accept queries for records and domains beyond those for which the servers are specifically designated.

4.1.1. Oblivious DoH Proxy

The DoH servers MUST be able to act as Oblivious Proxies. In this function, they will proxy encrypted queries and answers between clients and Oblivious Target DoH servers.

4.1.2. Oblivious DoH Target

The DoH servers MUST be able to act as Oblivious Targets. In this function, they will accept encrypted proxied queries from clients via Oblivious Proxy DoH servers, and provide encrypted answers using client keys.

4.1.3. Keying Material

In order to support acting as an Oblivious Target, a DoH server needs to provide a public HPKE [I-D.irtf-cfrg-hpke] key that can be used to encrypt client queries. This key is advertised in the SVCB record [I-D.pauly-dprive-oblivious-doh].

DoH servers also SHOULD provide an ESNI [I-D.ietf-tls-esni] key to encrypt the Server Name Indication field in TLS handshakes to the DoH server.

4.2. Advertise the DoH Server

The primary mechanism for advertising a Designated DoH Server is a SVCB DNS record (Section 3.1). This record MUST contain both the URI Template of the DoH Server as well as the Oblivious DoH Public Key. It MAY contain the ESNI key [I-D.ietf-tls-esni].

Servers MUST ensure that any SVCB records are signed with DNSSEC [RFC4033].

4.3. Provide Extended Configuration as a Web PvD

Beyond providing basic DoH server functionality, server nodes SHOULD provide a mechanism that allows clients to look up properties and configuration for the server deployment. Amongst other information, this configuration can optionally contain a list of some popular domains for which this server is designated. Clients can use this list to optimize lookups for common names.

This set of extended configuration information is referred to as a Web Provisioning Domain, or a Web PvD. Provisioning Domains are sets of consistent information that clients can use to access networks, including rules for resolution and proxying. Generally, these PvDs are provisioned directly, such as by a local router or a VPN. [I-D.ietf-intarea-provisioning-domains] defines an extensible configuration dictionary that can be used to add information to local PvD configurations. Web PvDs share the same JSON configuration format, and share the registry of keys defined as "Additional Information PvD Keys".

If present, the PvD JSON configuration MUST be made available to clients that request the "application/pvd+json" media type in a GET request to the DoH server's URI [I-D.ietf-intarea-provisioning-domains]. Clients MUST include this media type as an Accept header in their GET requests, and servers MUST mark this media type as their Content-Type header in responses. If the PvD JSON format is not supported, the server MUST reply with HTTP status code 415 [RFC7231].

The "identifier" key in the JSON configuration SHOULD be the hostname of the DoH Server itself.

For Web PvDs, the "prefixes" key within the JSON configuration SHOULD contain an empty array.

The key "dnsZones", which contains an array of domains as strings [I-D.ietf-intarea-provisioning-domains], indicates the zones that belong to the PvD. Any zone that is listed in this array for a Web PvD MUST have a corresponding SVCB record that defines the DoH server as designated for the zone. Servers SHOULD include in this array any names that are considered default or well-known for the deployment, but is not required or expected to list all zones or domains for which it is designated. The trade-off here is that zones that are listed can be fetched and validated automatically by clients, thus removing a bootstrapping step in discovering mappings from domains to Designated DoH Servers.

Clients that retrieve the Web PvD JSON dictionary SHOULD perform an SVCB record query for each of the entries in the "dnsZones" array in order to populate the mappings of domains. These MAY be performed in an oblivious fashion, but MAY also be queried directly on the DoH server (since the information is not user-specific, but in response to generic server-driven content). Servers can choose to preemptively transfer the relevant SVCB records if the PvD information retrieval is done with an HTTP version that supports PUSH semantics. This allows the server to avoid a round trip in zone validation even before the client has started requested SVCB records. Once the client requests an SVCB record for one of the names included in the "dnsZones" array, the server can also include the SVCB records for the other names in the array in the Additional section of the DNS response.

This document also registers one new key in the Additional Information PvD Keys registry, to identify the URI Template for the DoH server Section 10. When included in Web PvDs, this URI MUST match the template in the SVCB DNS Record.

Beyond providing resolution configuration, the Web PvD configuration can be extended to offer information about proxies and other services offered by the server deployment. Such keys are not defined in this document.

5. Server Deployment Considerations

When servers designate DoH servers for their names, the specific deployment model can impact the effective privacy and performance characteristics.

5.1. Single Content Provider

If a name always resolves to server IP addresses that are hosted by a single content provider, the name ought to designate a single DoH server. This DoH server will be most optimal when it is designated by many or all names that are hosted by the same content provider. This ensures that clients can increase connection reuse to reduce latency in connection setup.

A DoH server that corresponds to the content provider that hosts content has an opportunity to tune the responses provided to a client based on the location inferred by the client IP address.

5.2. Multiple Content Providers

Some hostnames may resolve to server IP addresses that are hosted by multiple content providers. In such scenarios, the deployment may want to be able to control the percentage of traffic that flows to each content provider.

In these scenarios, there can either be:

- o multiple designated DoH servers that are advertised via SVCB DNS Records; or,
- o a single designated DoH server that can be referenced by one or more SVCB DNS Records, operated by a party that is aware of both content providers and can manage splitting the traffic.

If a server deployment wants to easily control the split of traffic between different content providers, it ought to use the latter model of using a single designated DoH server that can better control which IP addresses are provided to clients. Otherwise, if a client is aware of multiple DoH servers, it might use a single resolver exclusively, which may lead to inconsistent behavior between clients that choose different resolvers.

5.3. Avoid Narrow Deployments

Using designated DoH servers can improve the privacy of name resolution whenever a DoH server is designated by many different names within one or more domains. This limits the amount of information leaked to an attacker observing traffic between a client and a DoH server: the attacker only learns that the client might be resolving one of the many names for which the server is designated (or might be performing an Oblivious query).

However, if a deployment designates a given DoH server for only one name, or a very small set of names, then it becomes easier for an attacker to infer that a specific name is being accessed by a client. For this reason, deployments are encouraged to avoid deploying a DoH server that is only designated by a small number of names. Clients can also choose to only whitelist DoH servers that are associated with many names.

Beyond the benefits to privacy, having a larger number of names designate a given DoH server improves the opportunity for DoH connection reuse, which can improve the performance of name resolutions.

6. Local Resolver Deployment Considerations

A key goal of Adaptive DNS is that clients will be able to use Designated DoH Servers to improve the privacy of queries, without entirely bypassing local network authority and policy. For example, if a client is attached to an enterprise Wi-Fi network that provides access and resolution for private names not generally accessible on the Internet, such names will only be usable when a local resolver is used.

In order to achieve this, a local network can advertise itself as authoritative for a domain, allowing it to be used prior to external servers in the client resolution algorithm Section 3.3.

6.1. Designating Local DoH Servers

If a local network wants to have clients send queries for a set of private domains to its own resolver, it needs to define an explicit provisioning domain [I-D.ietf-intarea-provisioning-domains]. The PvD RA option SHOULD set the H-flag to indicate that Additional Information is available. This Additional Information JSON object SHOULD include both the "dohTemplate" and "dnsZones" keys to define the local DoH server and the domains over which it claims authority.

In order to validate that a local resolver is designated for a given zone, the client SHOULD issue a SVCB record query for the names specified in the PvD information, using the DoH server specified in the PvD information. If there is no SVCB record for a name that points to the DoH server that can be validated using DNSSEC, the client SHOULD NOT automatically create a designation from the domain name to DoH server. See specific use cases in Section 6.2 for cases in which a local resolver may still be used.

Although local Designated DoH Servers MAY support proxying Oblivious DoH queries, a client SHOULD NOT select one of these servers as an Oblivious Proxy. Doing so might reveal the client's location to the Target based on the address of the proxy, which could contribute to deanonymizing the client. Clients can make an exception to this behavior if the DoH server designated by the local network is known to be a non-local service, such as when a local network configures a centralized public resolver to handle its DNS operations.

6.2. Local Use Cases

The various use cases for selecting locally-provisioned resolvers require different approaches for deployment and client resolution. The following list is not exhaustive, but provides guidance on how these scenarios can be achieved using the Adaptive DNS algorithm.

6.2.1. Accessing Local-Only Resolvable Content

Some names are not resolvable using generic DNS resolvers, but require using a DNS server that can resolve private names. This is common in enterprise scenarios, in which an enterprise can have a set of private names that it allows to be resolved when connected to a VPN or an enterprise-managed Wi-Fi network. In this case, clients that do not use the locally-provisioned resolver will fail to resolve private names.

In these scenarios, the local network SHOULD designate a local DoH server for the domains that are locally resolvable. For example, an enterprise that owns "private.example.org" would advertise "private.example.org" in its PvD information along with a DoH URI template. Clients could then use that locally-configured resolver with names under "private.example.org" according to the rules in Section 6.1.

In general, clients SHOULD only create designated DoH server associations when they can validate a SVCB record using DNSSEC. However, some deployments of private names might not want to sign all private names within a zone. There are thus a few possible deployment models:

- o "private.example.org" does have a DNSSEC-signed SVCB record that points to the local DoH server. The client requests the SVCB record for "private.example.org" using the local DoH server that is specified in the PvD information, and from that point on uses the local DoH server for names under "private.example.org".
- o Instead of signing "private.example.org", the deployment provides a DNSSEC-signed SVCB record for "example.org", thus steering all resolution under "example.org" to the local resolver.
- o No DNSSEC-signed SVCB record designates the local server. In this case, clients have a hint that the local network can serve names under "private.example.org", but do not have a way to validate the designation. Clients can in this case try to resolve names using external servers (such as via Oblivious DoH), and then MAY fall back to using locally-provisioned resolvers if the names do not resolve externally. This approach has the risk of exposing private names to public resolvers, which can be undesirable for certain enterprise deployments. Alternatively, if the client trusts the local network based on specific policy configured on the client, it can choose to resolve these names locally first. Note that this approach risks exposing names to a potentially malicious network that is masquerading as an authority for private names if the network cannot be validated in some other manner.

Deployments SHOULD use the one of the first two approaches (signing their records) whenever possible; the case of providing unsigned names is only described as a possibility for handling legacy enterprise deployments. Clients SHOULD choose to ignore any locally designated names that are not signed unless there is a specific policy configuration on the client.

6.2.2. Accessing Locally Optimized Content

Other names may be resolvable both publicly and on the local resolver, but have more optimized servers that are accessible only via the local network. For example, a Wi-Fi provider may provide access to a cache of video content that provides lower latency than publicly-accessible caches.

Names that are hosted locally in this way SHOULD use a designation with a DNSSEC-signed SVCB record for the name. If a client discovers that a local resolver is designated for a given name, the client SHOULD prefer using connections to this locally-hosted content rather than names resolved externally.

Note that having a DNSSEC-signed designation to the local resolver provides a clear indication that the entity that manages a given name has an explicit relationship with the local network provider.

6.2.3. Walled-Garden and Captive Network Deployments

Some networks do not provide any access to the general Internet, but host local content that clients can access. For example, a network on an airplane can give access to flight information and in-flight media, but will not allow access to any external hosts or DNS servers. These networks are often described as "walled-gardens".

Captive networks [I-D.ietf-capport-architecture] are similar in that they block access to external hosts, although they can provide generic access after some time.

If a walled-garden or captive network defines a PvD with additional information, it can define zones for names that it hosts, such as "airplane.example.com". It can also provide a locally-hosted encrypted DNS server.

However, if such a network does not support explicitly advertising local names, clients that try to establish connections to DoH servers will experience connection failures. In these cases, system traffic that is used for connecting to captive portals SHOULD use local resolvers. In addition, clients MAY choose to fall back to using direct resolution without any encryption if they determine that all connectivity is blocked otherwise. Note that this comes with a risk of a network blocking connections in order to induce this fallback behavior, so clients might want to inform users about this possible attack where appropriate, or prefer to not fall back if there is a concern about leaking user data.

6.2.4. Network-Based Filtering

Some networks currently rely on manipulating DNS name resolution in order to apply content filtering rules to clients associated with the network. Using encrypted DNS resolvers that are not participating in this filtering can bypass such enforcement. However, simply blocking connections for filtering is indistinguishable from a malicious attack from a client's perspective.

In order to indicate the presence of filtering requirements, a network deployment can add the "requiredDNSFiltering" and "dnsFilteredZones" keys to its PvD information. The "dnsFilteredZones" entry can contain an array of strings, each of which is a domain name that the network requires clients to resolve using the local resolver. If the array contains the string ".", it

indicates the network requires filtering for all domains. If "requiredDNSFiltering" is present with a boolean value of true, the network is indicating that it expects all client systems to send the names indicated by "dnsFilteredZones" to the local resolver. If "requiredDNSFiltering" is not present or set to false, then the filtering service is considered to be optional for clients that want to use it as a service to enforce desired policy.

Clients that receive indication of filtering requirements SHOULD NOT use any other resolver for the filtered domains, but treat the network as claiming authority. However, since this filtering cannot be authenticated, this behavior SHOULD NOT be done silently without user consent.

Networks that try to interfere with connections to encrypted DNS resolvers without indicating a requirement for filtering cannot be distinguished from misconfigurations or network attacks. Clients MAY choose to avoid sending any user-initiated connections on such networks to prevent malicious interception.

7. Performance Considerations

One of the challenges of using non-local DNS servers (such as cloud-based DoH servers) is that recursive queries made by these servers will originate from an IP address that is not necessarily geographically related to the client. Many DNS servers make assumptions about the geographic locality of clients to their recursive resolvers to optimize answers. To avoid this problem, the client's subnet can be forwarded to the authoritative server by the recursive using the EDNS0 Client Subnet feature. Oblivious DoH discourages this practice for privacy reasons. However, sharing this subnet, while detrimental to privacy, can result in better targeted DNS resolutions.

Adaptive DNS splits DoH queries into two sets: those made to Designated DoH Servers, and those made to Oblivious DoH servers. Oblivious queries are sensitive for privacy, and can encounter performance degradation as a result of not using the client subnet. Queries to designated DoH servers, on the other hand, are sent directly by clients, so the client IP address is made available to these servers. Since these servers are designated by the authority for the names, they can use the IP address subnet information to tune DNS answers.

Based on these properties, clients SHOULD prefer lookups via Designated DoH Servers over oblivious mechanisms whenever possible. Servers can encourage this by setting large TTLs for SVCB records and using longer TTLs for responses returned by their Designated DoH

Server endpoints which can be more confident they have accurate addressing information.

8. Security Considerations

In order to avoid interception and modification of the information retrieved by clients using Adaptive DNS, all exchanges between clients and servers are performed over encrypted connections, e.g., TLS.

Malicious adversaries may block client connections to all DoH or Oblivious DoH services as a Denial-of-Service (DoS) measure. Clients which cannot connect to any proxy may, by local policy, fall back to unencrypted DNS if this occurs.

9. Privacy Considerations

Clients must be careful in determining to which DoH servers they send queries directly without proxying. A malicious DoH server that can direct queries to itself can track or profile client activity. In order to avoid the possibility of a spoofed SVCB record designating a malicious DoH server for a name, clients MUST ensure that such records validate using DNSSEC [RFC4033].

Even servers that are officially designated can risk leaking or logging information about client lookups. Such risk can be mitigated by further restricting the list of DoH servers that are whitelisted for direct use based on client policy.

Using Oblivious DoH reduces the risk that a single DoH server can track or profile a client. However, clients should exercise caution when using Oblivious DoH responses from resolvers that do not carry DNSSEC signatures. An adversarial Oblivious Target resolver that wishes to learn the IP address of clients requesting resolution for sensitive domains can redirect clients to unique addresses of its choosing. Clients that use these answers when establishing TLS connections may then leak their local IP address to chosen server. Thus, when Oblivious DoH answers are returned without DNSSEC, Privacy-Sensitive Connections concerned about this attack SHOULD conceal their IP address via a TLS- or HTTP-layer proxy or some other tunneling mechanism.

An adversary able to see traffic on each path segment of a DoH or Oblivious DoH query (e.g., from client to proxy, proxy to target, and target to an authoritative DNS server) can link queries to specific clients with high probability. Failure to observe traffic on any one of these path segments makes this linkability increasingly difficult. For example, if an adversary can only observe traffic between a

client and proxy and egress traffic from a target, then it may be difficult to identify a specific client's query among the recursive queries generated by the target.

10. IANA Considerations

10.1. DoH Template PvD Key

This document adds a key to the "Additional Information PvD Keys" registry [I-D.ietf-intarea-provisioning-domains].

JSON key	Description	Type	Example
dohTemplate	DoH URI Template [RFC8484]	String	"https://dnsserver.example.net/dns-query{?dns}"

10.2. DNS Filtering PvD Keys

This document adds a key to the "Additional Information PvD Keys" registry [I-D.ietf-intarea-provisioning-domains].

JSON key	Description	Type	Example
requireDNSFiltering	A flag to indicate that the network requires filtering all DNS traffic using the provisioned resolver.	Boolean	true
dnsFilteredZones	A list of DNS domains as strings that represent domains that can be filtered by the provisioned resolver.	Array of Strings	["."]

Any network that sets the "requireDNSFiltering" boolean to false but provides "dnsFilteredZones" advertises the optional service of filtering on the provisioned network.

An "." in the "dnsFilteredZones" array represents a wildcard, which can be used to indicate that the network is requesting to filter all

names. Any more specific string represents a domain that requires filtering on the network.

10.3. DoH URI Template DNS Parameter

If present, this parameters indicates the URI template of a DoH server that is designated for use with the name being resolved. This is a string encoded as UTF-8 characters.

Name: dohuri

SvcParamKey: TBD

Meaning: URI template for a designated DoH server

Reference: This document.

11. Acknowledgments

Thanks to Erik Nygren, Lorenzo Colitti, Tommy Jensen, Mikael Abrahamsson, Ben Schwartz, Ask Hansen, Leif Hedstrom, Tim McCoy, Stuart Cheshire, Miguel Vega, Joey Deng, Ted Lemon, and Elliot Briggs for their feedback and input on this document.

12. References

12.1. Normative References

- [I-D.ietf-intarea-provisioning-domains]
Pfister, P., Vyncke, E., Pauly, T., Schinazi, D., and W. Shao, "Discovering Provisioning Domain Names and Data", draft-ietf-intarea-provisioning-domains-08 (work in progress), October 2019.
- [I-D.ietf-tls-esni]
Rescorla, E., Oku, K., Sullivan, N., and C. Wood, "Encrypted Server Name Indication for TLS 1.3", draft-ietf-tls-esni-04 (work in progress), July 2019.
- [I-D.irtf-cfrg-hpke]
Barnes, R. and K. Bhargavan, "Hybrid Public Key Encryption", draft-irtf-cfrg-hpke-00 (work in progress), July 2019.

- [I-D.nygren-dnsop-svcb-httpssvc]
Schwartz, B., Bishop, M., and E. Nygren, "Service binding and parameter specification via the DNS (DNS SVCB and HTTPSSVC)", draft-nygren-dnsop-svcb-httpssvc-00 (work in progress), September 2019.
- [I-D.pauly-dprive-oblivious-doh]
Kinnear, E., Pauly, T., Wood, C., and P. McManus, "Oblivious DNS Over HTTPS", draft-pauly-dprive-oblivious-doh-00 (work in progress), October 2019.
- [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>>.
- [RFC7231] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content", RFC 7231, DOI 10.17487/RFC7231, June 2014, <<https://www.rfc-editor.org/info/rfc7231>>.
- [RFC7858] Hu, Z., Zhu, L., Heidemann, J., Mankin, A., Wessels, D., and P. Hoffman, "Specification for DNS over Transport Layer Security (TLS)", RFC 7858, DOI 10.17487/RFC7858, May 2016, <<https://www.rfc-editor.org/info/rfc7858>>.
- [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>>.

12.2. Informative References

- [I-D.ietf-capport-architecture]
Larose, K. and D. Dolson, "CAPPORT Architecture", draft-ietf-capport-architecture-04 (work in progress), June 2019.
- [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>>.
- [RFC2132] Alexander, S. and R. Droms, "DHCP Options and BOOTP Vendor Extensions", RFC 2132, DOI 10.17487/RFC2132, March 1997, <<https://www.rfc-editor.org/info/rfc2132>>.

- [RFC8106] Jeong, J., Park, S., Beloeil, L., and S. Madanapalli,
"IPv6 Router Advertisement Options for DNS Configuration",
RFC 8106, DOI 10.17487/RFC8106, March 2017,
<<https://www.rfc-editor.org/info/rfc8106>>.
- [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>>.

Authors' Addresses

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

Email: ekinnear@apple.com

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

Email: tpauly@apple.com

Chris Wood
Apple Inc.
One Apple Park Way
Cupertino, California 95014
United States of America

Email: cawood@apple.com

Patrick McManus
Fastly

Email: mcmanus@ducksong.com

Network Working Group
Internet-Draft
Intended status: Experimental
Expires: 21 August 2022

E. Kinnear
Apple Inc.
P. McManus
Fastly
T. Pauly
Apple Inc.
T. Verma
C.A. Wood
Cloudflare
17 February 2022

Oblivious DNS Over HTTPS
draft-pauly-dprive-oblivious-doh-11

Abstract

This document describes a protocol that allows clients to hide their IP addresses from DNS resolvers via proxying encrypted DNS over HTTPS (DoH) messages. 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.

This experimental protocol is developed outside the IETF and is published here to guide implementation, ensure interoperability among implementations, and enable wide-scale experimentation.

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 21 August 2022.

Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

1. Introduction	2
1.1. Specification of Requirements	3
2. Terminology	3
3. Deployment Requirements	4
4. HTTP Exchange	4
4.1. HTTP Request	5
4.2. HTTP Request Example	6
4.3. HTTP Response	6
4.4. HTTP Response Example	7
4.5. HTTP Metadata	8
5. Configuration and Public Key Format	8
6. Protocol Encoding	9
6.1. Message Format	9
6.2. Encryption and Decryption Routines	11
7. Oblivious Client Behavior	12
8. Oblivious Target Behavior	13
9. Compliance Requirements	14
10. Experiment Overview	14
11. Security Considerations	14
11.1. Denial of Service	16
11.2. Proxy Policies	16
11.3. Authentication	16
12. IANA Considerations	17
12.1. Oblivious DoH Message Media Type	17
13. Acknowledgments	18
14. References	18
14.1. Normative References	18
14.2. Informative References	19
Appendix A. Use of Generic Proxy Services	20
Authors' Addresses	20

1. Introduction

DNS Over HTTPS (DoH) [RFC8484] defines a mechanism to allow DNS messages to be transmitted in HTTP messages protected with TLS. 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 to and receive answers from servers without revealing their local IP address (and thus information about the identity or location of the client) to the server.

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 experimental protocol built on DoH that permits proxied resolution, in which DNS messages are encrypted so that no server can independently read both the client IP address and the DNS message contents.

As with DoH, DNS messages exchanged over Oblivious DoH are fully-formed DNS messages. Clients that want to receive answers that are relevant to the network they are on without revealing their exact IP address can thus use the EDNS0 Client Subnet option [RFC7871], Section 7.1.2 to provide a hint to the resolver using Oblivious DoH.

This mechanism is intended to be used as one mechanism for resolving privacy-sensitive content in the broader context of DNS privacy.

This experimental protocol is developed outside the IETF and is published here to guide implementation, ensure interoperability among implementations, and enable wide-scale experimentation. See Section 10 for more details about the experiment.

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 Proxy: An HTTP server that proxies encrypted DNS queries and responses between a Client and an Oblivious Target, and is identified by a URI template [RFC6570] (see Section 4.1). Note that this Oblivious Proxy is not acting as a full HTTP proxy, but is instead a specialized server used to forward oblivious DNS messages.

Oblivious Target: An HTTP 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:

- * An Oblivious Proxy server, identified by a URI template.
- * An Oblivious Target server. The Target and Proxy are expected to be non-colluding (see Section 11).
- * One or more Target public keys for encrypting DNS queries send to a Target via a Proxy (Section 5). These keys guarantee that only the intended Target can decrypt Client queries.

The mechanism for discovering and provisioning the Proxy URI template and Target public keys is out of scope of this document.

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.

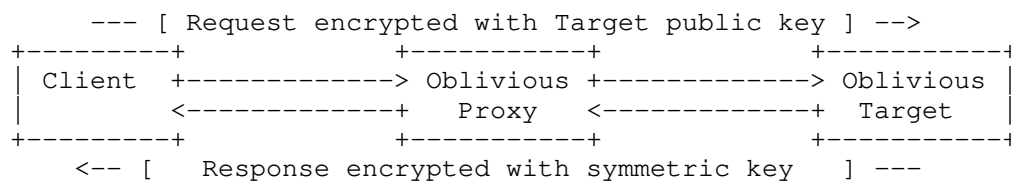


Figure 1: Oblivious DoH Exchange

4.1. HTTP Request

Oblivious DoH queries are created by the Client, and sent to the Proxy as HTTP requests using the POST method. Clients are configured with a Proxy URI Template [RFC6570] and the Target URI. The scheme for both the Proxy URI Template and the Target URI MUST be "https". The Proxy URI Template uses the Level 3 encoding defined in Section 1.2 of [RFC6570], and contains two variables: "targethost", which indicates the host name of the Target server; and "targetpath", which indicates the path on which the Target is accessible. Examples of Proxy URI Templates are shown below:

```
https://dnsproxy.example/dns-query{?targethost,targetpath}  
https://dnsproxy.example/{targethost}/{targetpath}
```

The URI Template MUST contain both the "targethost" and "targetpath" variables exactly once, and MUST NOT contain any other variables. The variables MUST be within the path or query components of the URI. Clients MUST ignore configurations that do not conform to this template. 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. Requests and responses MUST NOT be cached.

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.

A correctly encoded request has the HTTP Content-Type header "application/oblivious-dns-message", uses the HTTP POST method, and contains "targethost" and "targetpath" variables. If the Proxy fails to match the "targethost" and "targetpath" variables from the path, it MUST treat the request as malformed. The Proxy constructs the URI of the Target with the "https" scheme, using the value of "targethost" as the URI host, and the percent-decoded value of "targetpath" as the URI path. Proxies MUST check that Client requests are correctly encoded, and MUST return a 4xx (Client Error) if the check fails, along with the Proxy-Status response header with an "error" parameter of type "http_request_error" [I-D.ietf-httpbis-proxy-status].

Proxies MAY choose to not forward connections to non-standard ports. In such cases, Proxies can indicate the error with a 403 response status code, along with a Proxy-Status response header with an "error" parameter of type "http_request_denied", along with an appropriate explanation in "details".

If the Proxy cannot establish a connection to the Target, it can indicate the error with a 502 response status code, along with a Proxy-Status response header with an "error" parameter whose type indicates the reason. For example, if DNS resolution fails, the error type might be "dns_timeout", whereas if the TLS connection failed the error type might be "tls_protocol_error".

Upon receipt of requests from a Proxy, Targets MUST validate that the request has the HTTP Content-Type header "application/oblivious-dns-message" and uses the HTTP POST method. Targets can respond with a 4xx response status code if this check fails.

4.2. HTTP Request Example

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

```
:method = POST
:scheme = https
:authority = dnspoxy.example
:path = /dns-query?targethost=dnstarget.example&targetpath=/dns-query
accept = application/oblivious-dns-message
content-type = application/oblivious-dns-message
content-length = 106
```

<Bytes containing an encrypted Oblivious DNS query>

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

```
:method = POST
:scheme = https
:authority = dnstarget.example
:path = /dns-query
accept = application/oblivious-dns-message
content-type = application/oblivious-dns-message
content-length = 106
```

<Bytes containing an encrypted 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 an encrypted DNS message; see Section 6.

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.

Proxies forward responses from the Target to client, without any modifications to the body or status code. The Proxy also SHOULD add a Proxy-Status response header with an "received-status" parameter indicating that the status code was generated by the Target.

Note that if a Client receives a 3xx status code and chooses to follow a redirect, the subsequent request MUST also be performed through a Proxy in order to avoid directly exposing requests to the Target.

Requests that cannot be processed by the Target result in 4xx (Client Error) responses. If the Target and Client keys do not match, it is an authorization failure (HTTP status code 401; see Section 3.1 of [RFC7235]). Otherwise, if the Client's request is invalid, such as in the case of decryption failure, wrong message type, or deserialization failure, this is a bad request (HTTP status code 400; see Section 6.5.1 of [RFC7231]).

Even in case of DNS responses indicating failure, such as SERVFAIL or NXDOMAIN, a successful HTTP response with a 2xx status code is used as long as the DNS response is valid. This is identical to how DoH [RFC8484] handles HTTP response codes.

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 an encrypted Oblivious DNS response>
```

4.5. HTTP Metadata

Proxies forward requests and responses between Clients and Targets as specified in Section 4.1. Metadata sent with these messages could inadvertently weaken or remove Oblivious DoH privacy properties. Proxies **MUST NOT** send any Client-identifying information about Clients to Targets, such as "Forwarded" HTTP headers [RFC7239]. Additionally, Clients **MUST NOT** include any private state in requests to Proxies, such as HTTP cookies. See Section 11.3 for related discussion about Client authentication information.

5. Configuration and Public Key Format

In order to send a message to a Target, the Client needs to know a public key to use for encrypting its queries. The mechanism for discovering this configuration is out of scope of this document.

Servers ought to rotate public keys regularly. It is **RECOMMENDED** that servers rotate keys every day. Shorter rotation windows reduce the anonymity set of Clients that might use the public key, whereas longer rotation windows widen the timeframe of possible compromise.

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 0x0001: 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 0x0001.

`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 Section 7 of [HPKE]. 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.

6. Protocol Encoding

This section includes encoding and wire format details for Oblivious DoH, as well as routines for encrypting and decrypting encoded values.

6.1. 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). Implementations MAY enforce limits on the size of this field depending on the size of plaintext DNS messages. (DNS queries, for example, will not reach the size limit of $2^{16}-1$ in practice.)

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.

6.2. Encryption and Decryption Routines

Clients use the following utility functions for encrypting a Query and decrypting a Response as described in Section 7.

`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, resp_nonce):
    aead_key, aead_nonce = derive_secrets(context, Q_plain, resp_nonce)
    aad = 0x02 || len(resp_nonce) || resp_nonce
    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 8.

`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, resp_nonce):
    secret = context.Export("odoh response", Nk)
    salt = Q_plain || len(resp_nonce) || resp_nonce
    prk = Extract(salt, secret)
    key = Expand(odoh_prk, "odoh key", Nk)
    nonce = Expand(odoh_prk, "odoh nonce", Nn)
    return key, nonce
```

The `random(N)` function returns `N` cryptographically secure random bytes from a good source of entropy [RFC4086]. The `max(A, B)` function returns `A` if `A > B`, and `B` otherwise.

encrypt_response_body: Encrypt an Oblivious DoH response.

```
def encrypt_response_body(R_plain, aead_key, aead_nonce, resp_nonce):
    aad = 0x02 || len(resp_nonce) || resp_nonce
    R_encrypted = Seal(aead_key, aead_nonce, aad, R_plain)
    return R_encrypted
```

7. 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 6. Note also that `len(key_id)` outputs the length of `key_id` as a two-byte unsigned integer.
4. Output an `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 8, it uses `decrypt_response_body` to decrypt `R.encrypted_message` (using `R.key_id` as a nonce) and produce `R_plain`. Clients MUST validate `R_plain.padding` (as all zeros) before using `R_plain.dns_message`.

8. 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 401 (Unauthorized) 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. Create a fresh nonce `resp_nonce = random(max(Nn, Nk))`.
7. Compute `aead_key, aead_nonce = derive_secrets(context, Q_plain, resp_nonce)`.
8. Compute `R_encrypted = encrypt_response_body(R_plain, aead_key, aead_nonce, resp_nonce)`. The `key_id` field used for encryption carries `resp_nonce` in order for Clients to derive the same secrets. Also, the Seal function is that which is associated with the HPKE AEAD.
9. Output an `ObliviousDoHMessage` message `R` where `R.message_type = 0x02`, `R.key_id = resp_nonce`, and `R.encrypted_message = R_encrypted`.

The Target then sends R in a 2xx (Successful) response to the Proxy; see 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. In the event of an error (non 2xx status code), the Proxy forwards the Target error to the Client; see Section 4.3.

9. Compliance Requirements

Oblivious DoH uses HPKE for public key encryption [HPKE]. 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 [HPKE], Section 7.1)
- * KDF: HKDF-SHA256 (see [HPKE], Section 7.2)
- * AEAD: AES-128-GCM (see [HPKE], Section 7.3)

10. Experiment Overview

This document describes an experimental protocol built on DoH. The purpose of this experiment is to assess deployment configuration viability and related performance impacts on DNS resolution by measuring key performance indicators such as resolution latency. Experiment participants will test various parameters affecting service operation and performance, including mechanisms for discovery and configuration of DoH Proxies and Targets, as well as performance implications of connection reuse and pools where appropriate. The results of this experiment will be used to influence future protocol design and deployment efforts related to Oblivious DoH, such as Oblivious HTTP [OHTTP]. Implementations of DoH that are not involved in the experiment will not recognize this protocol and will not participate in the experiment. It is anticipated that use of Oblivious DoH and the duration of this experiment to be widespread.

11. Security Considerations

Oblivious DoH aims to keep knowledge of the true query origin and its contents known only to 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. Note that compromising both P and T is equivalent to collusion between these two parties in practice. 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 could 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 prescribe 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) can send bogus Oblivious DoH queries to targets as a Denial-of-Service (DoS) attack. Target servers can 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 can throttle specific clients in response to these errors. In general, however, Targets trust Proxies to not overwhelm the Target, and it is expected that Proxies either implement some form of rate limiting or client authentication to limit abuse; see Section 11.3.

Malicious Targets or Proxies can 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. Proxy Policies

Proxies are free to enforce any forwarding policy they desire for Clients. For example, they can choose to only forward requests to known or otherwise trusted Targets.

Proxies that do not reuse connections to Targets for many Clients may allow Targets to link individual queries to unknown Targets. To mitigate this linkability vector, it is RECOMMENDED that Proxies pool and reuse connections to Targets. Note that this benefits performance as well as privacy since queries do not incur any delay that might otherwise result from Proxy-to-Target connection establishment.

11.3. Authentication

Depending on the deployment scenario, Proxies and Targets might require authentication before use. Regardless of the authentication mechanism in place, Proxies MUST NOT reveal any Client authentication information to Targets. This is required so Targets cannot uniquely identify individual Clients.

Note that if Targets require Proxies to authenticate at the HTTP- or application-layer before use, this ought to be done before attempting to forward any Client query to the Target. This will allow Proxies to distinguish 401 Unauthorized response codes due to authentication failure from 401 Unauthorized response codes due to Client key mismatch; see Section 4.3.

12. IANA Considerations

This document makes changes to the "Multipurpose Internet Mail Extensions (MIME) and Media Types" registry. The changes are described in the following subsection.

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 encoded as ObliviousDoHMessage values, as defined in Section 6.1.

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; see Section 6.1.

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: N/A

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

Intended usage: COMMON

Restrictions on usage: N/A

Author: Tommy Pauly tpauly@apple.com (mailto:tpauly@apple.com)

Change controller: IETF

Provisional registration? (standards tree only): No

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 Nafeez Ahamed, Elliot Briggs, Marwan Fayed, Frederic Jacobs, Tommy Jensen, Jonathan Hoyland, Paul Schmitt, Brian Swander, Erik Nygren, and Peter Wu for the feedback and input.

14. References

14.1. Normative References

- [HPKE] Barnes, R. L., Bhargavan, K., Lipp, B., and C. A. Wood, "Hybrid Public Key Encryption", Work in Progress, Internet-Draft, draft-irtf-cfrg-hpke-12, 2 September 2021, <<https://www.ietf.org/archive/id/draft-irtf-cfrg-hpke-12.txt>>.
- [I-D.ietf-httpbis-proxy-status] Nottingham, M. and P. Sikora, "The Proxy-Status HTTP Response Header Field", Work in Progress, Internet-Draft, draft-ietf-httpbis-proxy-status-08, 13 October 2021, <<https://www.ietf.org/archive/id/draft-ietf-httpbis-proxy-status-08.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>>.
- [RFC4086] Eastlake 3rd, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, DOI 10.17487/RFC4086, June 2005, <<https://www.rfc-editor.org/info/rfc4086>>.
- [RFC6570] Gregorio, J., Fielding, R., Hadley, M., Nottingham, M., and D. Orchard, "URI Template", RFC 6570, DOI 10.17487/RFC6570, March 2012, <<https://www.rfc-editor.org/info/rfc6570>>.
- [RFC7231] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content", RFC 7231, DOI 10.17487/RFC7231, June 2014, <<https://www.rfc-editor.org/info/rfc7231>>.

- [RFC7235] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Authentication", RFC 7235, DOI 10.17487/RFC7235, June 2014, <<https://www.rfc-editor.org/info/rfc7235>>.
- [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, <<https://www.ietf.org/archive/id/draft-annee-dprive-oblivious-dns-00.txt>>.
- [OHTP] Thomson, M. and C. A. Wood, "Oblivious HTTP", Work in Progress, Internet-Draft, draft-ietf-ohai-ohttp-01, 15 February 2022, <<https://www.ietf.org/archive/id/draft-ietf-ohai-ohttp-01.txt>>.
- [RFC7239] Petersson, A. and M. Nilsson, "Forwarded HTTP Extension", RFC 7239, DOI 10.17487/RFC7239, June 2014, <<https://www.rfc-editor.org/info/rfc7239>>.
- [RFC7871] Contavalli, C., van der Gaast, W., Lawrence, D., and W. Kumari, "Client Subnet in DNS Queries", RFC 7871, DOI 10.17487/RFC7871, May 2016, <<https://www.rfc-editor.org/info/rfc7871>>.

Appendix A. Use of Generic Proxy Services

Using DoH over anonymizing proxy services such as Tor can 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 meant to be a generic connection-level anonymity system, and incurs higher latency costs and protocol complexity for the purpose of proxying individual DNS queries. In contrast, Oblivious DoH is a lightweight protocol built on 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.

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

Tanya Verma
Cloudflare
101 Townsend St
San Francisco,
United States of America
Email: vermatanyax@gmail.com

Christopher A. Wood
Cloudflare
101 Townsend St
San Francisco,
United States of America
Email: caw@heapingbits.net