

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: 22 October 2022

C. Huitema
Private Octopus Inc.
S. Dickinson
Sinodun IT
A. Mankin
Salesforce
20 April 2022

DNS over Dedicated QUIC Connections
draft-ietf-dprive-dnsquic-12

Abstract

This document describes the use of QUIC to provide transport confidentiality for DNS. The encryption provided by QUIC has similar properties to those provided by TLS, while QUIC transport eliminates the head-of-line blocking issues inherent with TCP and provides more efficient packet loss recovery than UDP. DNS over QUIC (DoQ) has privacy properties similar to DNS over TLS (DoT) specified in RFC7858, and latency characteristics similar to classic DNS over UDP. This specification describes the use of DNS over QUIC as a general-purpose transport for DNS and includes the use of DNS over QUIC for stub to recursive, recursive to authoritative, and zone transfer scenarios.

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 22 October 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	3
2. Key Words	4
3. Document work via GitHub	5
4. Design Considerations	5
4.1. Provide DNS Privacy	5
4.2. Design for Minimum Latency	5
4.3. Middlebox Considerations	6
4.4. No Server-Initiated Transactions	6
5. Specifications	6
5.1. Connection Establishment	7
5.1.1. Draft Version Identification	7
5.1.2. Port Selection	7
5.2. Stream Mapping and Usage	7
5.2.1. DNS Message IDs	9
5.3. DoQ Error Codes	9
5.3.1. Transaction Cancellation	10
5.3.2. Transaction Errors	11
5.3.3. Protocol Errors	11
5.3.4. Alternative error codes	12
5.4. Connection Management	12
5.5. Session Resumption and 0-RTT	13
5.6. Message Sizes	14
6. Implementation Requirements	14
6.1. Authentication	14
6.2. Fallback to Other Protocols on Connection Failure	15
6.3. Address Validation	15
6.4. Padding	16
6.5. Connection Handling	16
6.5.1. Connection Reuse	17
6.5.2. Resource Management	17
6.5.3. Using 0-RTT and Session Resumption	18
6.5.4. Controlling Connection Migration For Privacy	18
6.6. Processing Queries in Parallel	19
6.7. Zone transfer	19
6.8. Flow Control Mechanisms	19
7. Implementation Status	20
7.1. Performance Measurements	21

8. Security Considerations	21
9. Privacy Considerations	22
9.1. Privacy Issues With 0-RTT data	22
9.2. Privacy Issues With Session Resumption	23
9.3. Privacy Issues With Address Validation Tokens	24
9.4. Privacy Issues With Long Duration Sessions	24
9.5. Traffic Analysis	25
10. IANA Considerations	25
10.1. Registration of DoQ Identification String	25
10.2. Reservation of Dedicated Port	25
10.3. Reservation of Extended DNS Error Code Too Early	26
10.4. DNS over QUIC Error Codes Registry	27
11. Acknowledgements	28
12. References	29
12.1. Normative References	29
12.2. Informative References	31
Appendix A. The NOTIFY Service	33
Appendix B. Notable Changes From Previous Versions	33
B.1. Stream Mapping Incompatibility With Draft-02	33
Authors' Addresses	34

1. Introduction

Domain Name System (DNS) concepts are specified in "Domain names - concepts and facilities" [RFC1034]. The transmission of DNS queries and responses over UDP and TCP is specified in "Domain names - implementation and specification" [RFC1035].

This document presents a mapping of the DNS protocol over the QUIC transport [RFC9000] [RFC9001]. DNS over QUIC is referred to here as DoQ, in line with "DNS Terminology" [I-D.ietf-dnsop-rfc8499bis].

The goals of the DoQ mapping are:

1. Provide the same DNS privacy protection as DNS over TLS (DoT) [RFC7858]. This includes an option for the client to authenticate the server by means of an authentication domain name as specified in "Usage Profiles for DNS over TLS and DNS over DTLS" [RFC8310].
2. Provide an improved level of source address validation for DNS servers compared to classic DNS over UDP.
3. Provide a transport that does not impose path MTU limitations on the size of DNS responses it can send.

In order to achieve these goals, and to support ongoing work on encryption of DNS, the scope of this document includes

- * the "stub to recursive resolver" scenario
- * the "recursive resolver to authoritative nameserver" scenario and
- * the "nameserver to nameserver" scenario (mainly used for zone transfers (XFR) [RFC1995], [RFC5936]).

In other words, this document specifies QUIC as a general-purpose transport for DNS.

The specific non-goals of this document are:

1. No attempt is made to evade potential blocking of DNS over QUIC traffic by middleboxes.
2. No attempt to support server-initiated transactions, which are used only in DNS Stateful Operations (DSO) [RFC8490].

Specifying the transmission of an application over QUIC requires specifying how the application's messages are mapped to QUIC streams, and generally how the application will use QUIC. This is done for HTTP in "Hypertext Transfer Protocol Version 3 (HTTP/3)" [I-D.ietf-quic-http]. The purpose of this document is to define the way DNS messages can be transmitted over QUIC.

DNS over HTTP [RFC8484] can be used with HTTP/3 to get some of the benefits of QUIC. However, a lightweight direct mapping for DNS over QUIC can be regarded as a more natural fit for both the recursive to authoritative and zone transfer scenarios which rarely involve intermediaries. In these scenarios, the additional overhead of HTTP is not offset by, e.g., benefits of HTTP proxying and caching behavior.

In this document, Section 4 presents the reasoning that guided the proposed design. Section 5 specifies the actual mapping of DoQ. Section 6 presents guidelines on the implementation, usage and deployment of DoQ.

2. Key Words

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.

3. Document work via GitHub

(RFC EDITOR NOTE: THIS SECTION TO BE REMOVED BEFORE PUBLICATION) The GitHub repository for this document is at <https://github.com/huitema/dnssoquic>. Proposed text and editorial changes are very much welcomed there, but any functional changes should always first be discussed on the IETF DPRIVE WG (dns-privacy) mailing list.

4. Design Considerations

This section and its subsections present the design guidelines that were used for DoQ. Whilst all other sections in this document are normative, this section is informative in nature.

4.1. Provide DNS Privacy

DoT [RFC7858] defines how to mitigate some of the issues described in "DNS Privacy Considerations" [RFC9076] by specifying how to transmit DNS messages over TLS. The "Usage Profiles for DNS over TLS and DNS over DTLS" [RFC8310] specify Strict and Opportunistic Usage Profiles for DoT including how stub resolvers can authenticate recursive resolvers.

QUIC connection setup includes the negotiation of security parameters using TLS, as specified in "Using TLS to Secure QUIC" [RFC9001], enabling encryption of the QUIC transport. Transmitting DNS messages over QUIC will provide essentially the same privacy protections as DoT [RFC7858] including Strict and Opportunistic Usage Profiles [RFC8310]. Further discussion on this is provided in Section 9.

4.2. Design for Minimum Latency

QUIC is specifically designed to reduce protocol-induced delays, with features such as:

1. Support for 0-RTT data during session resumption.
2. Support for advanced packet loss recovery procedures as specified in "QUIC Loss Detection and Congestion Control" [RFC9002].
3. Mitigation of head-of-line blocking by allowing parallel delivery of data on multiple streams.

This mapping of DNS to QUIC will take advantage of these features in three ways:

1. Optional support for sending 0-RTT data during session resumption (the security and privacy implications of this are discussed in later sections).
2. Long-lived QUIC connections over which multiple DNS transactions are performed, generating the sustained traffic required to benefit from advanced recovery features.
3. Mapping of each DNS Query/Response transaction to a separate stream, to mitigate head-of-line blocking. This enables servers to respond to queries "out of order". It also enables clients to process responses as soon as they arrive, without having to wait for in order delivery of responses previously posted by the server.

These considerations are reflected in the mapping of DNS traffic to QUIC streams in Section 5.2.

4.3. Middlebox Considerations

Using QUIC might allow a protocol to disguise its purpose from devices on the network path using encryption and traffic analysis resistance techniques like padding, traffic pacing, and traffic shaping. This specification does not include any measures that are designed to avoid such classification -- the padding mechanisms defined in Section 6.4 are intended to obfuscate the specific records contained in DNS queries and responses, but not the fact that this is DNS traffic. Consequently, firewalls and other middleboxes might be able to distinguish DoQ from other protocols that use QUIC, like HTTP, and apply different treatment.

The lack of measures in this specification to avoid protocol classification is not an endorsement of such practices.

4.4. No Server-Initiated Transactions

As stated in Section 1, this document does not specify support for server-initiated transactions within established DoQ connections. That is, only the initiator of the DoQ connection may send queries over the connection.

DSO does support server-initiated transactions within existing connections. However, DoQ as defined here does not meet the criteria for an applicable transport for DSO because it does not guarantee in-order delivery of messages, see Section 4.2 of [RFC8490].

5. Specifications

5.1. Connection Establishment

DoQ connections are established as described in the QUIC transport specification [RFC9000]. During connection establishment, DoQ support is indicated by selecting the Application-Layer Protocol Negotiation (ALPN) token "doq" in the crypto handshake.

5.1.1. Draft Version Identification

(RFC EDITOR NOTE: THIS SECTION TO BE REMOVED BEFORE PUBLICATION) Only implementations of the final, published RFC can identify themselves as "doq". Until such an RFC exists, implementations MUST NOT identify themselves using this string.

Implementations of draft versions of the protocol MUST add the string "-" and the corresponding draft number to the identifier. For example, draft-ietf-dprive-dnsquic-00 is identified using the string "doq-i00".

5.1.2. Port Selection

By default, a DNS server that supports DoQ MUST listen for and accept QUIC connections on the dedicated UDP port TBD (number to be defined in Section 10), unless there is a mutual agreement to use another port.

By default, a DNS client desiring to use DoQ with a particular server MUST establish a QUIC connection to UDP port TBD on the server, unless there is a mutual agreement to use another port.

DoQ connections MUST NOT use UDP port 53. This recommendation against use of port 53 for DoQ is to avoid confusion between DoQ and the use of DNS over UDP [RFC1035]. The risk of confusion exists even if two parties agreed on port 53, as other parties without knowledge of that agreement might still try to use that port.

In the stub to recursive scenario, the use of port 443 as a mutually agreed alternative port can be operationally beneficial, since port 443 is used by many services using QUIC and HTTP-3 and thus less likely to be blocked than other ports. Several mechanisms for stubs to discover recursives offering encrypted transports, including the use of custom ports, are the subject of ongoing work.

5.2. Stream Mapping and Usage

The mapping of DNS traffic over QUIC streams takes advantage of the QUIC stream features detailed in Section 2 of [RFC9000], the QUIC transport specification.

DNS query/response traffic [RFC1034], [RFC1035] follows a simple pattern in which the client sends a query, and the server provides one or more responses (multiple responses can occur in zone transfers).

The mapping specified here requires that the client selects a separate QUIC stream for each query. The server then uses the same stream to provide all the response messages for that query. In order that multiple responses can be parsed, a 2-octet length field is used in exactly the same way as the 2-octet length field defined for DNS over TCP [RFC1035]. The practical result of this is that the content of each QUIC stream is exactly the same as the content of a TCP connection that would manage exactly one query.

All DNS messages (queries and responses) sent over DoQ connections MUST be encoded as a 2-octet length field followed by the message content as specified in [RFC1035].

The client MUST select the next available client-initiated bidirectional stream for each subsequent query on a QUIC connection, in conformance with the QUIC transport specification [RFC9000]. Packet losses and other network events might cause queries to arrive in a different order. Servers SHOULD process queries as they arrive, as not doing so would cause unnecessary delays.

The client MUST send the DNS query over the selected stream, and MUST indicate through the STREAM FIN mechanism that no further data will be sent on that stream.

The server MUST send the response(s) on the same stream and MUST indicate, after the last response, through the STREAM FIN mechanism that no further data will be sent on that stream.

Therefore, a single DNS transaction consumes a single bidirectional client-initiated stream. This means that the client's first query occurs on QUIC stream 0, the second on 4, and so on (see Section 2.1 of [RFC9000]).

Servers MAY defer processing of a query until the STREAM FIN has been indicated on the stream selected by the client.

Servers and clients MAY monitor the number of "dangling" streams. These are open streams where the following events have not occurred after implementation defined timeouts:

- * the expected queries or responses have not been received or,

- * the expected queries or responses have been received but not the STREAM FIN

Implementations MAY impose a limit on the number of such dangling streams. If limits are encountered, implementations MAY close the connection.

5.2.1. DNS Message IDs

When sending queries over a QUIC connection, the DNS Message ID MUST be set to zero. The stream mapping for DoQ allows for unambiguous correlation of queries and responses and so the Message ID field is not required.

This has implications for proxying DoQ message to and from other transports. For example, proxies may have to manage the fact that DoQ can support a larger number of outstanding queries on a single connection than e.g., DNS over TCP because DoQ is not limited by the Message ID space. This issue already exists for DoH, where a Message ID of 0 is recommended.

When forwarding a DNS message from DoQ over another transport, a DNS Message ID MUST be generated according to the rules of the protocol that is in use. When forwarding a DNS message from another transport over DoQ, the Message ID MUST be set to zero.

5.3. DoQ Error Codes

The following error codes are defined for use when abruptly terminating streams, and used as application protocol error codes when aborting reading of streams, or immediately closing connections:

DOQ_NO_ERROR (0x0): No error. This is used when the connection or stream needs to be closed, but there is no error to signal.

DOQ_INTERNAL_ERROR (0x1): The DoQ implementation encountered an internal error and is incapable of pursuing the transaction or the connection.

DOQ_PROTOCOL_ERROR (0x2): The DoQ implementation encountered a protocol error and is forcibly aborting the connection.

DOQ_REQUEST_CANCELLED (0x3): A DoQ client uses this to signal that it wants to cancel an outstanding transaction.

DOQ_EXCESSIVE_LOAD (0x4): A DoQ implementation uses this to signal when closing a connection due to excessive load.

DOQ_UNSPECIFIED_ERROR (0x5): A DoQ implementation uses this in the absence of a more specific error code.

DOQ_ERROR_RESERVED (0xd098ea5e): Alternative error code used for tests.

See Section 10.4 for details on registering new error codes.

5.3.1. Transaction Cancellation

In QUIC, sending STOP_SENDING requests that a peer cease transmission on a stream. If a DoQ client wishes to cancel an outstanding request, it MUST issue a QUIC STOP_SENDING, and it SHOULD use the error code DOQ_REQUEST_CANCELLED. It MAY use a more specific error code registered according to Section 10.4. The STOP_SENDING request may be sent at any time but will have no effect if the server response has already been sent, in which case the client will simply discard the incoming response. The corresponding DNS transaction MUST be abandoned.

Servers that receive STOP_SENDING act in accordance with Section 3.5 of [RFC9000]. Servers SHOULD NOT continue processing a DNS transaction if they receive a STOP_SENDING.

Servers MAY impose implementation limits on the total number or rate of request cancellations. If limits are encountered, servers MAY close the connection. In this case, servers wanting to help client debugging MAY use the error code DOQ_EXCESSIVE_LOAD. There is always a trade-off between helping good faith clients debug issues and allowing denial-of-service attackers to test server defenses, so depending on circumstances servers might very well choose to send different error codes.

Note that this mechanism provides a way for secondaries to cancel a single zone transfer occurring on a given stream without having to close the QUIC connection.

Servers MUST NOT continue processing a DNS transaction if they receive a RESET_STREAM request from the client before the client indicates the STREAM FIN. The server MUST issue a RESET_STREAM to indicate that the transaction is abandoned unless

- * it has already done so for another reason or
- * it has already both sent the response and indicated the STREAM FIN.

5.3.2. Transaction Errors

Servers normally complete transactions by sending a DNS response (or responses) on the transaction's stream, including cases where the DNS response indicates a DNS error. For example, a Server Failure (SERVFAIL, [RFC1035]) SHOULD be notified to the client by sending back a response with the Response Code set to SERVFAIL.

If a server is incapable of sending a DNS response due to an internal error, it SHOULD issue a QUIC RESET_STREAM frame. The error code SHOULD be set to DOQ_INTERNAL_ERROR. The corresponding DNS transaction MUST be abandoned. Clients MAY limit the number of unsolicited QUIC Stream Resets received on a connection before choosing to close the connection.

Note that this mechanism provides a way for primaries to abort a single zone transfer occurring on a given stream without having to close the QUIC connection.

5.3.3. Protocol Errors

Other error scenarios can occur due to malformed, incomplete or unexpected messages during a transaction. These include (but are not limited to)

- * a client or server receives a message with a non-zero Message ID
- * a client or server receives a STREAM FIN before receiving all the bytes for a message indicated in the 2-octet length field
- * a client receives a STREAM FIN before receiving all the expected responses
- * a server receives more than one query on a stream
- * a client receives a different number of responses on a stream than expected (e.g., multiple responses to a query for an A record)
- * a client receives a STOP_SENDING request
- * the client or server does not indicate the expected STREAM FIN after sending requests or responses (see Section 5.2).
- * an implementation receives a message containing the edns-tcp-keepalive EDNS(0) Option [RFC7828] (see Section 6.5.2)
- * a client or a server attempts to open a unidirectional QUIC stream

- * a server attempts to open a server-initiated bidirectional QUIC stream
- * receiving a "replayable" transaction in 0-RTT data (for servers not willing to handle this case - see section Section 5.5)

If a peer encounters such an error condition it is considered a fatal error. It SHOULD forcibly abort the connection using QUIC's CONNECTION_CLOSE mechanism, and SHOULD use the DoQ error code DOQ_PROTOCOL_ERROR. In some cases, it MAY instead silently abandon the connection, which uses fewer of the local resources but makes debugging at the offending node more difficult.

It is noted that the restrictions on use of the above EDNS(0) options has implications for proxying message from TCP/DoT/DoH over DoQ.

5.3.4. Alternative error codes

This specification suggests specific error codes in Section 5.3.1, Section 5.3.2, and Section 5.3.3. These error codes are meant to facilitate investigation of failures and other incidents. New error codes may be defined in future versions of DoQ, or registered as specified in Section 10.4.

Because new error codes can be defined without negotiation, use of an error code in an unexpected context or receipt of an unknown error code MUST be treated as equivalent to DOQ_UNSPECIFIED_ERROR.

Implementations MAY wish to test the support for the error code extension mechanism by using error codes not listed in this document, or they MAY use DOQ_ERROR_RESERVED.

5.4. Connection Management

Section 10 of [RFC9000], the QUIC transport specification, specifies that connections can be closed in three ways:

- * idle timeout
- * immediate close
- * stateless reset

Clients and servers implementing DoQ SHOULD negotiate use of the idle timeout. Closing on idle timeout is done without any packet exchange, which minimizes protocol overhead. Per Section 10.1 of [RFC9000], the QUIC transport specification, the effective value of the idle timeout is computed as the minimum of the values advertised by the two endpoints. Practical considerations on setting the idle timeout are discussed in Section 6.5.2.

Clients SHOULD monitor the idle time incurred on their connection to the server, defined by the time spent since the last packet from the server has been received. When a client prepares to send a new DNS query to the server, it SHOULD check whether the idle time is sufficiently lower than the idle timer. If it is, the client SHOULD send the DNS query over the existing connection. If not, the client SHOULD establish a new connection and send the query over that connection.

Clients MAY discard their connections to the server before the idle timeout expires. A client that has outstanding queries SHOULD close the connection explicitly using QUIC's CONNECTION_CLOSE mechanism and the DoQ error code DOQ_NO_ERROR.

Clients and servers MAY close the connection for a variety of other reasons, indicated using QUIC's CONNECTION_CLOSE. Client and servers that send packets over a connection discarded by their peer might receive a stateless reset indication. If a connection fails, all the in progress transaction on that connection MUST be abandoned.

5.5. Session Resumption and 0-RTT

A client MAY take advantage of the session resumption and 0-RTT mechanisms supported by QUIC transport [RFC9000] and QUIC TLS [RFC9001], if the server supports them. Clients SHOULD consider potential privacy issues associated with session resumption before deciding to use this mechanism and specifically evaluate the trade-offs presented in the various sections of this document. The privacy issues are detailed in Section 9.1 and Section 9.2, and the implementation considerations are discussed in Section 6.5.3.

The 0-RTT mechanism MUST NOT be used to send DNS requests that are not "replayable" transactions. In this specification, only transactions that have an OPCODE of QUERY or NOTIFY are considered replayable and therefore other OPCODES MUST NOT be sent in 0-RTT data. See Appendix A for a detailed discussion of why NOTIFY is included here.

Servers MAY support session resumption, and MAY do that with or without supporting 0-RTT, using the mechanisms described in Section 4.6.1 of [RFC9001]. Servers supporting 0-RTT MUST NOT immediately process non-replayable transactions received in 0-RTT data, but instead MUST adopt one of the following behaviours:

- * Queue the offending transaction and only process it after the QUIC handshake has been completed, as defined in Section 4.1.1 of [RFC9001].
- * Reply to the offending transaction with a response code REFUSED and an Extended DNS Error Code (EDE) "Too Early", using the extended RCODE mechanisms defined in [RFC6891] and the extended DNS errors defined in [RFC8914]; see Section 10.3.
- * Close the connection with the error code DOQ_PROTOCOL_ERROR.

5.6. Message Sizes

DoQ Queries and Responses are sent on QUIC streams, which in theory can carry up to 2^{62} bytes. However, DNS messages are restricted in practice to a maximum size of 65535 bytes. This maximum size is enforced by the use of a two-octet message length field in DNS over TCP [RFC1035] and DNS over TLS [RFC7858], and by the definition of the "application/dns-message" for DNS over HTTP [RFC8484]. DoQ enforces the same restriction.

The Extension Mechanisms for DNS (EDNS) [RFC6891] allow peers to specify the UDP message size. This parameter is ignored by DoQ. DoQ implementations always assume that the maximum message size is 65535 bytes.

6. Implementation Requirements

6.1. Authentication

For the stub to recursive resolver scenario, the authentication requirements are the same as described in DoT [RFC7858] and "Usage Profiles for DNS over TLS and DNS over DTLS" [RFC8310]. [RFC8932] states that DNS privacy services SHOULD provide credentials that clients can use to authenticate the server. Given this, and to align with the authentication model for DoH, DoQ stubs SHOULD use a Strict authentication profile. Client authentication for the encrypted stub to recursive scenario is not described in any DNS RFC.

For zone transfer, the authentication requirements are the same as described in [RFC9103].

For the recursive resolver to authoritative nameserver scenario, authentication requirements are unspecified at the time of writing and are the subject on ongoing work in the DPRIVE WG.

6.2. Fallback to Other Protocols on Connection Failure

If the establishment of the DoQ connection fails, clients MAY attempt to fall back to DoT and then potentially clear text, as specified in DoT [RFC7858] and "Usage Profiles for DNS over TLS and DNS over DTLS" [RFC8310], depending on their privacy profile.

DNS clients SHOULD remember server IP addresses that don't support DoQ. Mobile clients might also remember the lack of DoQ support by given IP addresses on a per-context basis (e.g. per network or provisioning domain).

Timeouts, connection refusals, and QUIC handshake failures are indicators that a server does not support DoQ. Clients SHOULD NOT attempt DoQ queries to a server that does not support DoQ for a reasonable period (such as one hour per server). DNS clients following an out-of-band key-pinned privacy profile ([RFC7858]) MAY be more aggressive about retrying after DoQ connection failures.

6.3. Address Validation

Section 8 of [RFC9000], the QUIC transport specification, defines Address Validation procedures to avoid servers being used in address amplification attacks. DoQ implementations MUST conform to this specification, which limits the worst case amplification to a factor 3.

DoQ implementations SHOULD consider configuring servers to use the Address Validation using Retry Packets procedure defined in Section 8.1.2 of [RFC9000], the QUIC transport specification. This procedure imposes a 1-RTT delay for verifying the return routability of the source address of a client, similar to the DNS Cookies mechanism [RFC7873].

DoQ implementations that configure Address Validation using Retry Packets SHOULD implement the Address Validation for Future Connections procedure defined in Section 8.1.3 of [RFC9000], the QUIC transport specification. This defines how servers can send NEW_TOKEN frames to clients after the client address is validated, in order to avoid the 1-RTT penalty during subsequent connections by the client from the same address.

6.4. Padding

Implementations MUST protect against the traffic analysis attacks described in Section 9.5 by the judicious injection of padding. This could be done either by padding individual DNS messages using the EDNS(0) Padding Option [RFC7830] or by padding QUIC packets (see Section 19.1 of [RFC9000]).

In theory, padding at the QUIC packet level could result in better performance for the equivalent protection, because the amount of padding can take into account non-DNS frames such as acknowledgements or flow control updates, and also because QUIC packets can carry multiple DNS messages. However, applications can only control the amount of padding in QUIC packets if the implementation of QUIC exposes adequate APIs. This leads to the following recommendation:

- * if the implementation of QUIC exposes APIs to set a padding policy, DNS over QUIC SHOULD use that API to align the packet length to a small set of fixed sizes.
- * if padding at the QUIC packet level is not available or not used, DNS over QUIC MUST ensure that all DNS queries and responses are padded to a small set of fixed sizes, using the EDNS(0) padding extension as specified in [RFC7830].

Implementation might choose not to use a QUIC API for padding if it is significantly simpler to re-use existing DNS message padding logic which is applied to other encrypted transports.

In the absence of a standard policy for padding sizes, implementations SHOULD follow the recommendations of the Experimental status "Padding Policies for Extension Mechanisms for DNS (EDNS(0))" [RFC8467]. While Experimental, these recommendations are referenced because they are implemented and deployed for DoT, and provide a way for implementations to be fully compliant with this specification.

6.5. Connection Handling

"DNS Transport over TCP - Implementation Requirements" [RFC7766] provides updated guidance on DNS over TCP, some of which is applicable to DoQ. This section provides similar advice on connection handling for DoQ.

6.5.1. Connection Reuse

Historic implementations of DNS clients are known to open and close TCP connections for each DNS query. To amortize connection setup costs, both clients and servers SHOULD support connection reuse by sending multiple queries and responses over a single persistent QUIC connection.

In order to achieve performance on par with UDP, DNS clients SHOULD send their queries concurrently over the QUIC streams on a QUIC connection. That is, when a DNS client sends multiple queries to a server over a QUIC connection, it SHOULD NOT wait for an outstanding reply before sending the next query.

6.5.2. Resource Management

Proper management of established and idle connections is important to the healthy operation of a DNS server.

An implementation of DoQ SHOULD follow best practices similar to those specified for DNS over TCP [RFC7766], in particular with regard to:

- * Concurrent Connections (Section 6.2.2 of [RFC7766], updated by Section 6.4 of [RFC9103])
- * Security Considerations (Section 10 of [RFC7766])

Failure to do so may lead to resource exhaustion and denial of service.

Clients that want to maintain long duration DoQ connections SHOULD use the idle timeout mechanisms defined in Section 10.1 of [RFC9000], the QUIC transport specification. Clients and servers MUST NOT send the edns-tcp-keepalive EDNS(0) Option [RFC7828] in any messages sent on a DoQ connection (because it is specific to the use of TCP/TLS as a transport).

This document does not make specific recommendations for timeout values on idle connections. Clients and servers should reuse and/or close connections depending on the level of available resources. Timeouts may be longer during periods of low activity and shorter during periods of high activity.

6.5.3. Using 0-RTT and Session Resumption

Using 0-RTT for DNS over QUIC has many compelling advantages. Clients can establish connections and send queries without incurring a connection delay. Servers can thus negotiate low values of the connection timers, which reduces the total number of connections that they need to manage. They can do that because the clients that use 0-RTT will not incur latency penalties if new connections are required for a query.

Session resumption and 0-RTT data transmission create privacy risks detailed in Section 9.2 and Section 9.1. The following recommendations are meant to reduce the privacy risks while enjoying the performance benefits of 0-RTT data, subject to the restrictions specified in Section 5.5.

Clients SHOULD use resumption tickets only once, as specified in Appendix C.4 to [RFC8446]. By default, clients SHOULD NOT use session resumption if the client's connectivity has changed.

Clients could receive address validation tokens from the server using the NEW_TOKEN mechanism; see Section 8 of [RFC9000]. The associated tracking risks are mentioned in Section 9.3. Clients SHOULD only use the address validation tokens when they are also using session resumption, thus avoiding additional tracking risks.

Servers SHOULD issue session resumption tickets with a sufficiently long lifetime (e.g., 6 hours), so that clients are not tempted to either keep connection alive or frequently poll the server to renew session resumption tickets. Servers SHOULD implement the anti-replay mechanisms specified in Section 8 of [RFC8446].

6.5.4. Controlling Connection Migration For Privacy

DoQ implementations might consider using the connection migration features defined in Section 9 of [RFC9000]. These features enable connections to continue operating as the client's connectivity changes. As detailed in Section 9.4, these features trade off privacy for latency. By default, clients SHOULD be configured to prioritize privacy and start new sessions if their connectivity changes.

6.6. Processing Queries in Parallel

As specified in Section 7 of [RFC7766] "DNS Transport over TCP - Implementation Requirements", resolvers are RECOMMENDED to support the preparing of responses in parallel and sending them out of order. In DoQ, they do that by sending responses on their specific stream as soon as possible, without waiting for availability of responses for previously opened streams.

6.7. Zone transfer

[RFC9103] specifies zone transfer over TLS (XoT) and includes updates to [RFC1995] (IXFR), [RFC5936] (AXFR) and [RFC7766]. Considerations relating to the re-use of XoT connections described there apply analogously to zone transfers performed using DoQ connections. One reason for re-iterating such specific guidance is the lack of effective connection re-use in existing TCP/TLS zone transfer implementations today. The following recommendations apply:

- * DoQ servers MUST be able to handle multiple concurrent IXFR requests on a single QUIC connection
- * DoQ servers MUST be able to handle multiple concurrent AXFR requests on a single QUIC connection
- * DoQ implementations SHOULD
 - use the same QUIC connection for both AXFR and IXFR requests to the same primary
 - send those requests in parallel as soon as they are queued i.e. do not wait for a response before sending the next query on the connection (this is analogous to pipelining requests on a TCP/TLS connection)
 - send the response(s) for each request as soon as they are available i.e. response streams MAY be sent intermingled

6.8. Flow Control Mechanisms

Servers and Clients manage flow control using the mechanisms defined in Section 4 of [RFC9000]. These mechanisms allow clients and servers to specify how many streams can be created, how much data can be sent on a stream, and how much data can be sent on the union of all streams. For DNS over QUIC, controlling how many streams are created allows servers to control how many new requests the client can send on a given connection.

Flow control exists to protect endpoint resources. For servers, global and per-stream flow control limits control how much data can be sent by clients. The same mechanisms allow clients to control how much data can be sent by servers. Values that are too small will unnecessarily limit performance. Values that are too large might expose endpoints to overload or memory exhaustion. Implementations or deployments will need to adjust flow control limits to balance these concerns. In particular, zone transfer implementations will need to control these limits carefully to ensure both large and concurrent zone transfers are well managed.

Initial values of parameters control how many requests and how much data can be sent by clients and servers at the beginning of the connection. These values are specified in transport parameters exchanged during the connection handshake. The parameter values received in the initial connection also control how many requests and how much data can be sent by clients using 0-RTT data in a resumed connection. Using too small values of these initial parameters would restrict the usefulness of allowing 0-RTT data.

7. Implementation Status

(RFC EDITOR NOTE: THIS SECTION TO BE REMOVED BEFORE PUBLICATION) This section records the status of known implementations of the protocol defined by this specification at the time of posting of this Internet-Draft, and is based on a proposal described in [RFC7942].

1. AdGuard launched a DoQ recursive resolver service in December 2020. They have released a suite of open source tools that support DoQ:
 1. AdGuard C++ DNS libraries (<https://github.com/AdguardTeam/DnsLibs>) A DNS proxy library that supports all existing DNS protocols including DNS-over-TLS, DNS-over-HTTPS, DNSCrypt and DNS-over-QUIC (experimental).
 2. DNS Proxy (<https://github.com/AdguardTeam/dnsproxy>) A simple DNS proxy server that supports all existing DNS protocols including DNS-over-TLS, DNS-over-HTTPS, DNSCrypt, and DNS-over-QUIC. Moreover, it can work as a DNS-over-HTTPS, DNS-over-TLS or DNS-over-QUIC server.
 3. CoreDNS fork for AdGuard DNS (<https://github.com/AdguardTeam/coredns>) Includes DNS-over-QUIC server-side support.
 4. dnslookup (<https://github.com/ameshkov/dnslookup>) Simple command line utility to make DNS lookups. Supports all known DNS protocols: plain DNS, DoH, DoT, DoQ, DNSCrypt.

2. Quicdoq (<https://github.com/private-octopus/quicdoq>) Quicdoq is a simple open source implementation of DoQ. It is written in C, based on Picoquic (<https://github.com/private-octopus/picoquic>).
3. Flamethrower (<https://github.com/DNS-OARC/flamethrower/tree/dns-over-quic>) is an open source DNS performance and functional testing utility written in C++ that has an experimental implementation of DoQ.
4. aioquic (<https://github.com/aiortc/aioquic>) is an implementation of QUIC in Python. It includes example client and server for DoQ.

7.1. Performance Measurements

To the authors' knowledge, no benchmarking studies comparing DoT, DoH and DoQ are published yet. However, anecdotal evidence from the AdGuard DoQ recursive resolver deployment (<https://adguard.com/en/blog/dns-over-quic.html>) indicates that it performs similarly (and possibly better) compared to the other encrypted protocols, particularly in mobile environments. Reasons given for this include that DoQ

- * Uses less bandwidth due to a more efficient handshake (and due to less per message overhead when compared to DoH).
- * Performs better in mobile environments due to the increased resilience to packet loss
- * Can maintain connections as users move between mobile networks via its connection management

8. Security Considerations

A Threat Analysis of the Domain Name System is found in [RFC3833]. This analysis was written before the development of DoT, DoH and DoQ, and probably needs to be updated.

The security considerations of DoQ should be comparable to those of DoT [RFC7858]. DoT as specified in [RFC7858] only addresses the stub to recursive resolver scenario, but the considerations about person-in-the-middle attacks, middleboxes and caching of data from clear text connections also apply for DoQ to the resolver to authoritative server scenario. As stated in Section 6.1 the authentication requirements for securing zone transfer using DoQ are the same as those for zone transfer over DoT, therefore the general security considerations are entirely analogous to those described in [RFC9103].

DoQ relies on QUIC, which itself relies on TLS 1.3 and thus supports by default the protections against downgrade attacks described in [BCP195]. QUIC specific issues and their mitigations are described in Section 21 of [RFC9000].

9. Privacy Considerations

The general considerations of encrypted transports provided in "DNS Privacy Considerations" [RFC9076] apply to DoQ. The specific considerations provided there do not differ between DoT and DoQ, and are not discussed further here. Similarly, "Recommendations for DNS Privacy Service Operators" [RFC8932] (which covers operational, policy, and security considerations for DNS privacy services) is also applicable to DoQ services.

QUIC incorporates the mechanisms of TLS 1.3 [RFC8446] and this enables QUIC transmission of "0-RTT" data. This can provide interesting latency gains, but it raises two concerns:

1. Adversaries could replay the 0-RTT data and infer its content from the behavior of the receiving server.
2. The 0-RTT mechanism relies on TLS session resumption, which can provide linkability between successive client sessions.

These issues are developed in Section 9.1 and Section 9.2.

9.1. Privacy Issues With 0-RTT data

The 0-RTT data can be replayed by adversaries. That data may trigger queries by a recursive resolver to authoritative resolvers. Adversaries may be able to pick a time at which the recursive resolver outgoing traffic is observable, and thus find out what name was queried for in the 0-RTT data.

This risk is in fact a subset of the general problem of observing the behavior of the recursive resolver discussed in "DNS Privacy Considerations" [RFC9076]. The attack is partially mitigated by reducing the observability of this traffic. The mandatory replay protection mechanisms in TLS 1.3 [RFC8446] limit but do not eliminate the risk of replay. 0-RTT packets can only be replayed within a narrow window, which is only wide enough to account for variations in clock skew and network transmission.

The recommendation for TLS 1.3 [RFC8446] is that the capability to use 0-RTT data should be turned off by default, and only enabled if the user clearly understands the associated risks. In the case of DoQ, allowing 0-RTT data provides significant performance gains, and

there is a concern that a recommendation to not use it would simply be ignored. Instead, a set of practical recommendations is provided in Section 5.5 and Section 6.5.3.

The specifications in Section 5.5 block the most obvious risks of replay attacks, as they only allow for transactions that will not change the long-term state of the server.

The attacks described above apply to the stub resolver to recursive resolver scenario, but similar attacks might be envisaged in the recursive resolver to authoritative resolver scenario, and the same mitigations apply.

9.2. Privacy Issues With Session Resumption

The QUIC session resumption mechanism reduces the cost of re-establishing sessions and enables 0-RTT data. There is a linkability issue associated with session resumption, if the same resumption token is used several times. Attackers on path between client and server could observe repeated usage of the token and use that to track the client over time or over multiple locations.

The session resumption mechanism allows servers to correlate the resumed sessions with the initial sessions, and thus to track the client. This creates a virtual long duration session. The series of queries in that session can be used by the server to identify the client. Servers can most probably do that already if the client address remains constant, but session resumption tickets also enable tracking after changes of the client's address.

The recommendations in Section 6.5.3 are designed to mitigate these risks. Using session tickets only once mitigates the risk of tracking by third parties. Refusing to resume a session if addresses change mitigates the incremental risk of tracking by the server (but the risk of tracking by IP address remains).

The privacy trade-offs here may be context specific. Stub resolvers will have a strong motivation to prefer privacy over latency since they often change location. However, recursive resolvers that use a small set of static IP addresses are more likely to prefer the reduced latency provided by session resumption and may consider this a valid reason to use resumption tickets even if the IP address changed between sessions.

Encrypted zone transfer (RFC9103) explicitly does not attempt to hide the identity of the parties involved in the transfer, but at the same time such transfers are not particularly latency sensitive. This means that applications supporting zone transfers may decide to apply the same protections as stub to recursive applications.

9.3. Privacy Issues With Address Validation Tokens

QUIC specifies address validation mechanisms in Section 8 of [RFC9000]. Use of an address validation token allows QUIC servers to avoid an extra RTT for new connections. Address validation tokens are typically tied to an IP address. QUIC clients normally only use these tokens when setting up a new connection from a previously used address. However, clients are not always aware that they are using a new address. This could be due to NAT, or because the client does not have an API available to check if the IP address has changed (which can be quite often for IPv6). There is a linkability risk if clients mistakenly use address validation tokens after unknowingly moving to a new location.

The recommendations in Section 6.5.3 mitigates this risk by tying the usage of the NEW_TOKEN to that of session resumption, though this recommendation does not cover the case where the client is unaware of the address change.

9.4. Privacy Issues With Long Duration Sessions

A potential alternative to session resumption is the use of long duration sessions: if a session remains open for a long time, new queries can be sent without incurring connection establishment delays. It is worth pointing out that the two solutions have similar privacy characteristics. Session resumption may allow servers to keep track of the IP addresses of clients, but long duration sessions have the same effect.

In particular, a DoQ implementation might take advantage of the connection migration features of QUIC to maintain a session even if the client's connectivity changes, for example if the client migrates from a Wi-Fi connection to a cellular network connection, and then to another Wi-Fi connection. The server would be able to track the client location by monitoring the succession of IP addresses used by the long duration connection.

The recommendation in Section 6.5.4 mitigates the privacy concerns related to long duration sessions using multiple client addresses.

9.5. Traffic Analysis

Even though QUIC packets are encrypted, adversaries can gain information from observing packet lengths, in both queries and responses, as well as packet timing. Many DNS requests are emitted by web browsers. Loading a specific web page may require resolving dozens of DNS names. If an application adopts a simple mapping of one query or response per packet, or "one QUIC STREAM frame per packet", then the succession of packet lengths may provide enough information to identify the requested site.

Implementations SHOULD use the mechanisms defined in Section 6.4 to mitigate this attack.

10. IANA Considerations

10.1. Registration of DoQ Identification String

This document creates a new registration for the identification of DoQ in the "Application Layer Protocol Negotiation (ALPN) Protocol IDs" registry [RFC7301].

The "doq" string identifies DoQ:

Protocol: DoQ

Identification Sequence: 0x64 0x6F 0x71 ("doq")

Specification: This document

10.2. Reservation of Dedicated Port

For both TCP and UDP port 853 is currently reserved for 'DNS query-response protocol run over TLS/DTLS' [RFC7858].

However, the specification for DNS over DTLS (DoD) [RFC8094] is experimental, limited to stub to resolver, and no implementations or deployments currently exist to the authors' knowledge (even though several years have passed since the specification was published).

This specification proposes to additionally reserve the use of UDP port 853 for DoQ. QUIC version 1 was designed to be able to co-exist with other protocols on the same port, including DTLS, see Section 17.2 of [RFC9000]. This means that deployments that serve DNS over DTLS and DNS over QUIC (QUIC version 1) on the same port will be able to demultiplex the two due to the second most significant bit in each UDP payload. Such deployments ought to check the signatures of future versions or extensions (e.g., [I-D.ietf-quic-bit-grease]) of QUIC and DTLS before deploying them to serve DNS on the same port.

IANA is requested to update the following value in the "Service Name and Transport Protocol Port Number Registry" in the System Range. The registry for that range requires IETF Review or IESG Approval [RFC6335].

Service Name: domain-s

Port Number: 853

Transport Protocol(s): UDP

Assignee: IESG

Contact: IETF Chair

Description: DNS query-response protocol run over DTLS or QUIC

Reference: [RFC7858][RFC8094] This document

Additionally, IANA is requested to update the Description field for the corresponding TCP port 853 allocation to be 'DNS query-response protocol run over TLS' and to remove [RFC8094] from the TCP allocation's Reference field for consistency and clarity.

(UPDATE ON IANA REQUEST: THIS SENTENCE TO BE REMOVED BEFORE PUBLICATION) Review by the port experts on 13th December 2021 determined that the proposed updates to the existing port allocation were acceptable and will be made when this document is approved for publication.

10.3. Reservation of Extended DNS Error Code Too Early

IANA is requested to add the following value to the Extended DNS Error Codes registry [RFC8914]:

INFO-CODE: TBD

Purpose: Too Early

Reference: This document

10.4. DNS over QUIC Error Codes Registry

IANA [SHALL add/has added] a registry for "DNS over QUIC Error Codes" on the "Domain Name System (DNS) Parameters" web page.

The "DNS over QUIC Error Codes" registry governs a 62-bit space. This space is split into three regions that are governed by different policies:

- * Permanent registrations for values between 0x00 and 0x3f (in hexadecimal; inclusive), which are assigned using Standards Action or IESG Approval as defined in Section 4.9 and Section 4.10 of [RFC8126]
- * Permanent registrations for values larger than 0x3f, which are assigned using the Specification Required policy ([RFC8126])
- * Provisional registrations for values larger than 0x3f, which require Expert Review, as defined in Section 4.5 of [RFC8126].

Provisional reservations share the range of values larger than 0x3f with some permanent registrations. This is by design, to enable conversion of provisional registrations into permanent registrations without requiring changes in deployed systems. (This design is aligned with the principles set in Section 22 of [RFC9000].)

Registrations in this registry MUST include the following fields:

Value: The assigned codepoint.

Status: "Permanent" or "Provisional".

Contact: Contact details for the registrant.

In addition, permanent registrations MUST include:

Error: A short mnemonic for the parameter.

Specification: A reference to a publicly available specification for the value (optional for provisional registrations).

Description: A brief description of the error code semantics, which MAY be a summary if a specification reference is provided.

Provisional registrations of codepoints are intended to allow for private use and experimentation with extensions to DNS over QUIC. However, provisional registrations could be reclaimed and reassigned for another purpose. In addition to the parameters listed above, provisional registrations MUST include:

Date: The date of last update to the registration.

A request to update the date on any provisional registration can be made without review from the designated expert(s).

The initial contents of this registry are shown in Table 1 and all entries share the following fields:

Status: Permanent

Contact: DPRIVE WG

Specification: Section 5.3

Value	Error	Description
0x0	DOQ_NO_ERROR	No error
0x1	DOQ_INTERNAL_ERROR	Implementation error
0x2	DOQ_PROTOCOL_ERROR	Generic protocol violation
0x3	DOQ_REQUEST_CANCELLED	Request cancelled by client
0x4	DOQ_EXCESSIVE_LOAD	Closing a connection for excessive load
0x5	DOQ_UNSPECIFIED_ERROR	No error reason specified
0xd098ea5e	DOQ_ERROR_RESERVED	Alternative error code used for tests

Table 1: Initial DNS over QUIC Error Codes Entries

11. Acknowledgements

This document liberally borrows text from the HTTP-3 specification [I-D.ietf-quic-http] edited by Mike Bishop, and from the DoT specification [RFC7858] authored by Zi Hu, Liang Zhu, John Heidemann, Allison Mankin, Duane Wessels, and Paul Hoffman.

The privacy issue with 0-RTT data and session resumption were analyzed by Daniel Kahn Gillmor (DKG) in a message to the IETF "DPRIVE" working group [DNS0RTT].

Thanks to Tony Finch for an extensive review of the initial version of this draft, and to Robert Evans for the discussion of 0-RTT privacy issues. Early reviews by Paul Hoffman and Martin Thomson and interoperability tests conducted by Stephane Bortzmeyer helped improve the definition of the protocol.

Thanks also to Martin Thomson and Martin Duke for their later reviews focussing on the low level QUIC details which helped clarify several aspects of DoQ. Thanks to Andrey Meshkov, Loganaden Velvindron, Lucas Pardue, Matt Joras, Mirja Kuelewind, Brian Trammell and Phillip Hallam-Baker for their reviews and contributions.

12. References

12.1. Normative References

- [RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, DOI 10.17487/RFC1034, November 1987, <<https://www.rfc-editor.org/info/rfc1034>>.
- [RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, DOI 10.17487/RFC1035, November 1987, <<https://www.rfc-editor.org/info/rfc1035>>.
- [RFC1995] Ohta, M., "Incremental Zone Transfer in DNS", RFC 1995, DOI 10.17487/RFC1995, August 1996, <<https://www.rfc-editor.org/info/rfc1995>>.
- [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>>.
- [RFC5936] Lewis, E. and A. Hoenes, Ed., "DNS Zone Transfer Protocol (AXFR)", RFC 5936, DOI 10.17487/RFC5936, June 2010, <<https://www.rfc-editor.org/info/rfc5936>>.
- [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>>.

- [RFC7301] Friedl, S., Popov, A., Langley, A., and E. Stephan, "Transport Layer Security (TLS) Application-Layer Protocol Negotiation Extension", RFC 7301, DOI 10.17487/RFC7301, July 2014, <<https://www.rfc-editor.org/info/rfc7301>>.
- [RFC7766] Dickinson, J., Dickinson, S., Bellis, R., Mankin, A., and D. Wessels, "DNS Transport over TCP - Implementation Requirements", RFC 7766, DOI 10.17487/RFC7766, March 2016, <<https://www.rfc-editor.org/info/rfc7766>>.
- [RFC7830] Mayrhofer, A., "The EDNS(0) Padding Option", RFC 7830, DOI 10.17487/RFC7830, May 2016, <<https://www.rfc-editor.org/info/rfc7830>>.
- [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>>.
- [RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <<https://www.rfc-editor.org/info/rfc8126>>.
- [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>>.
- [RFC8310] Dickinson, S., Gillmor, D., and T. Reddy, "Usage Profiles for DNS over TLS and DNS over DTLS", RFC 8310, DOI 10.17487/RFC8310, March 2018, <<https://www.rfc-editor.org/info/rfc8310>>.
- [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>>.
- [RFC8914] Kumari, W., Hunt, E., Arends, R., Hardaker, W., and D. Lawrence, "Extended DNS Errors", RFC 8914, DOI 10.17487/RFC8914, October 2020, <<https://www.rfc-editor.org/info/rfc8914>>.

- [RFC9000] Iyengar, J., Ed. and M. Thomson, Ed., "QUIC: A UDP-Based Multiplexed and Secure Transport", RFC 9000, DOI 10.17487/RFC9000, May 2021, <<https://www.rfc-editor.org/info/rfc9000>>.
- [RFC9001] Thomson, M., Ed. and S. Turner, Ed., "Using TLS to Secure QUIC", RFC 9001, DOI 10.17487/RFC9001, May 2021, <<https://www.rfc-editor.org/info/rfc9001>>.
- [RFC9103] Toorop, W., Dickinson, S., Sahib, S., Aras, P., and A. Mankin, "DNS Zone Transfer over TLS", RFC 9103, DOI 10.17487/RFC9103, August 2021, <<https://www.rfc-editor.org/info/rfc9103>>.

12.2. Informative References

- [BCP195] Sheffer, Y., Holz, R., and P. Saint-Andre, "Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", BCP 195, RFC 7525, May 2015.
- Moriarty, K. and S. Farrell, "Deprecating TLS 1.0 and TLS 1.1", BCP 195, RFC 8996, March 2021.
- <<https://www.rfc-editor.org/info/bcp195>>
- [DNS0RTT] Kahn Gillmor, D., "DNS + 0-RTT", Message to DNS-Privacy WG mailing list, 6 April 2016, <<https://www.ietf.org/mail-archive/web/dns-privacy/current/msg01276.html>>.
- [I-D.ietf-dnsop-rfc8499bis] Hoffman, P. and K. Fujiwara, "DNS Terminology", Work in Progress, Internet-Draft, draft-ietf-dnsop-rfc8499bis-03, 28 September 2021, <<https://www.ietf.org/archive/id/draft-ietf-dnsop-rfc8499bis-03.txt>>.
- [I-D.ietf-quic-bit-grease] Thomson, M., "Greasing the QUIC Bit", Work in Progress, Internet-Draft, draft-ietf-quic-bit-grease-02, 10 November 2021, <<https://www.ietf.org/archive/id/draft-ietf-quic-bit-grease-02.txt>>.
- [I-D.ietf-quic-http] Bishop, M., "Hypertext Transfer Protocol Version 3 (HTTP/3)", Work in Progress, Internet-Draft, draft-ietf-quic-http-34, 2 February 2021, <<https://www.ietf.org/archive/id/draft-ietf-quic-http-34.txt>>.

- [RFC1996] Vixie, P., "A Mechanism for Prompt Notification of Zone Changes (DNS NOTIFY)", RFC 1996, DOI 10.17487/RFC1996, August 1996, <<https://www.rfc-editor.org/info/rfc1996>>.
- [RFC3833] Atkins, D. and R. Austein, "Threat Analysis of the Domain Name System (DNS)", RFC 3833, DOI 10.17487/RFC3833, August 2004, <<https://www.rfc-editor.org/info/rfc3833>>.
- [RFC6335] Cotton, M., Eggert, L., Touch, J., Westerlund, M., and S. Cheshire, "Internet Assigned Numbers Authority (IANA) Procedures for the Management of the Service Name and Transport Protocol Port Number Registry", BCP 165, RFC 6335, DOI 10.17487/RFC6335, August 2011, <<https://www.rfc-editor.org/info/rfc6335>>.
- [RFC7828] Wouters, P., Abley, J., Dickinson, S., and R. Bellis, "The edns-tcp-keepalive EDNS0 Option", RFC 7828, DOI 10.17487/RFC7828, April 2016, <<https://www.rfc-editor.org/info/rfc7828>>.
- [RFC7873] Eastlake 3rd, D. and M. Andrews, "Domain Name System (DNS) Cookies", RFC 7873, DOI 10.17487/RFC7873, May 2016, <<https://www.rfc-editor.org/info/rfc7873>>.
- [RFC7942] Sheffer, Y. and A. Farrel, "Improving Awareness of Running Code: The Implementation Status Section", BCP 205, RFC 7942, DOI 10.17487/RFC7942, July 2016, <<https://www.rfc-editor.org/info/rfc7942>>.
- [RFC8094] Reddy, T., Wing, D., and P. Patil, "DNS over Datagram Transport Layer Security (DTLS)", RFC 8094, DOI 10.17487/RFC8094, February 2017, <<https://www.rfc-editor.org/info/rfc8094>>.
- [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>>.
- [RFC8490] Bellis, R., Cheshire, S., Dickinson, J., Dickinson, S., Lemon, T., and T. Pusateri, "DNS Stateful Operations", RFC 8490, DOI 10.17487/RFC8490, March 2019, <<https://www.rfc-editor.org/info/rfc8490>>.
- [RFC8932] Dickinson, S., Overeinder, B., van Rijswijk-Deij, R., and A. Mankin, "Recommendations for DNS Privacy Service Operators", BCP 232, RFC 8932, DOI 10.17487/RFC8932, October 2020, <<https://www.rfc-editor.org/info/rfc8932>>.

[RFC9002] Iyengar, J., Ed. and I. Swett, Ed., "QUIC Loss Detection and Congestion Control", RFC 9002, DOI 10.17487/RFC9002, May 2021, <<https://www.rfc-editor.org/info/rfc9002>>.

[RFC9076] Wicinski, T., Ed., "DNS Privacy Considerations", RFC 9076, DOI 10.17487/RFC9076, July 2021, <<https://www.rfc-editor.org/info/rfc9076>>.

Appendix A. The NOTIFY Service

This appendix discusses why it is considered acceptable to send NOTIFY (see [RFC1996]) in 0-RTT data.

Section 5.5 says "The 0-RTT mechanism SHOULD NOT be used to send DNS requests that are not "replayable" transactions". This specification supports sending a NOTIFY in 0-RTT data because although a NOTIFY technically changes the state of the receiving server, the effect of replaying NOTIFYS has negligible impact in practice.

NOTIFY messages prompt a secondary to either send an SOA query or an XFR request to the primary on the basis that a newer version of the zone is available. It has long been recognized that NOTIFYS can be forged and, in theory, used to cause a secondary to send repeated unnecessary requests to the primary. For this reason, most implementations have some form of throttling of the SOA/XFR queries triggered by the receipt of one or more NOTIFYS.

[RFC9103] describes the privacy risks associated with both NOTIFY and SOA queries and does not include addressing those risks within the scope of encrypting zone transfers. Given this, the privacy benefit of using DoQ for NOTIFY is not clear - but for the same reason, sending NOTIFY as 0-RTT data has no privacy risk above that of sending it using cleartext DNS.

Appendix B. Notable Changes From Previous Versions

(RFC EDITOR NOTE: THIS SECTION TO BE REMOVED BEFORE PUBLICATION)

B.1. Stream Mapping Incompatibility With Draft-02

Versions prior to -02 of this specification proposed a simpler mapping scheme of queries and responses to QUIC stream, which omitted the 2 byte length field and supported only a single response on a given stream. The more complex mapping in Section 5.2 was adopted to specifically cater for XFR support, however, it breaks compatibility with earlier versions.

Authors' Addresses

Christian Huitema
Private Octopus Inc.
427 Golfcourse Rd
Friday Harbor, WA 98250
United States of America
Email: huitema@huitema.net

Sara Dickinson
Sinodun IT
Oxford Science Park
Oxford
OX4 4GA
United Kingdom
Email: sara@sinodun.com

Allison Mankin
Salesforce
Email: allison.mankin@gmail.com

Network Working Group
Internet-Draft
Intended status: Experimental
Expires: 3 October 2021

P. Hoffman
ICANN
P. van Dijk
PowerDNS
1 April 2021

Recursive to Authoritative DNS with Unauthenticated Encryption
draft-ietf-dprive-opportunistic-adotq-02

Abstract

This document describes a use case and a method for a DNS recursive resolver to use unauthenticated encryption when communicating with authoritative servers. The motivating use case for this method is that more encryption on the Internet is better, and some resolver operators believe that unauthenticated encryption is better than no encryption at all. The method described here is optional for both the recursive resolver and the authoritative server. This method supports unauthenticated encryption using the same mechanism for discovery of encryption support for the server as [I-D.rescorla-dprive-adox-latest].

NOTE: The file name for this draft, draft-ietf-dprive-opportunistic-adotq, is now incorrect. This draft only covers unauthenticated encryption, not opportunistic encryption.

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 3 October 2021.

Copyright Notice

Copyright (c) 2021 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	2
1.1. Use Case for Unauthenticated Encryption	3
1.2. Summary of Protocol	3
1.3. Definitions	4
2. Discovering Whether an Authoritative Server Uses Encryption	4
3. Resolving with Encryption	5
3.1. Resolver Session Failures	6
4. Serving with Encryption	7
5. Resolvers Reporting Errors to Authoritative Servers	7
6. IANA Considerations	7
7. Security Considerations	8
8. Acknowledgements	8
9. References	8
9.1. Normative References	8
9.2. Informative References	9
Authors' Addresses	10

1. Introduction

A recursive resolver using traditional DNS over port 53 may wish instead to use encrypted communication with authoritative servers in order to limit snooping of its DNS traffic by passive or on-path attackers. The recursive resolver can use unauthenticated encryption (defined in [RFC7435]) to achieve this goal.

This document describes the use case for unauthenticated encryption in recursive resolvers in Section 1.1. The encryption method with authoritative servers can be DNS-over-TLS [RFC7858] (DoT), DNS-over-HTTPS [RFC8484] (DoH), and/or DNS-over-QUIC [I-D.ietf-dprive-dnsquic] (DoQ), as described in Section 3.

The document also describes a discovery method that shows if an authoritative server supports encryption in Section 2.

See [I-D.rescorla-dprive-adox-latest] for a description of the use case and a proposed mechanism for fully-authenticated encryption.

NOTE: The draft uses the SVCB record as a discovery mechanism for encryption by a particular authoritative server. Any record type that can show multiple types of encryption (currently DoT, DoH, and DoQ) can be used for discovery. Thus, this record type might change in the future, depending on the discussion in the DPRIVE WG.

1.1. Use Case for Unauthenticated Encryption

The use case in this document for unauthenticated encryption is recursive resolver operators who are happy to use encryption with authoritative servers if doing so doesn't significantly slow down getting answers, and authoritative server operators that are happy to use encryption with recursive resolvers if it doesn't cost much. In this use case, resolvers do not want to return an error for requests that were sent over an encrypted channel if they would have been able to give a correct answer using unencrypted transport.

Resolvers and authoritative servers understand that using encryption costs something, but are willing to absorb the costs for the benefit of more Internet traffic being encrypted. The extra costs (compared to using traditional DNS on port 53) include:

- * Extra round trips to establish TCP for every session (but not necessarily for every query)
- * Extra round trips for TLS establishment
- * Greater CPU use for TLS establishment
- * Greater CPU use for encryption after TLS establishment
- * Greater memory use for holding TLS state

This use case is not expected to apply to all resolvers or authoritative servers. For example, according to [RSO_STATEMENT], some root server operators do not want to be the early adopters for DNS with encryption. The protocol in this document explicitly allows authoritative servers to signal when they are ready to begin offering DNS with encryption.

1.2. Summary of Protocol

This summary gives an overview of how the parts of the protocol work together.

- * The resolver discovers whether any authoritative server of interest supports DNS with encryption by querying for the SVCB records [I-D.ietf-dnsop-svcb-https]. As described in [I-D.schwartz-svcb-dns], SVCB records can indicate that a server supports encrypted transport of DNS queries.

NOTE: In this document, the term "SVCB record" is used only for SVCB records that indicate encryption as described in [I-D.schwartz-svcb-dns]. SVCB records that do not have these indicators in the RDATA are not included in the term "SVCB record" in this document.

- * The resolver uses any authoritative server with a SVCB record that indicates encryption to perform unauthenticated encryption.
- * The resolver does not fail to set up encryption if the authentication in the TLS session fails.

1.3. Definitions

The terms "recursive resolver", "authoritative server", and "classic DNS" are defined in [I-D.ietf-dnsop-rfc8499bis].

"DNS with encryption" means transport of DNS over any of DoT, DoH, or DoQ. A server that supports DNS with encryption supports transport over one or more of DoT, DoH, or DoQ.

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. Discovering Whether an Authoritative Server Uses Encryption

A recursive resolver discovers whether an authoritative server supports DNS with encryption by looking for a cached SVCB record for the name of the authoritative server (with "_dns" prefix) with a positive answer. A cached SVCB record with a negative answer indicates that the authoritative server does not support any encrypted transport. Positive and negative responses for SVCB queries are cached the same way as for all other DNS resource records.

See [I-D.rescorla-dprive-adox-latest] for examples of querying for NS records and for SVCB records, and the interpretation of positive answers.

If the cache has no positive or negative answers for any SVCB record for any of a zone's authoritative servers, the resolver MAY send queries for the SVCB records for some or all of the zone's authoritative servers and wait for a positive response so that the resolver can use DNS with encryption for the original query. In this situation, the resolver MAY instead just use classic DNS for the original query but simultaneously queue queries for the SVCB records for some or all of the zone's authoritative servers so that future queries might be able to use DNS with encryption.

Discovery using SVCB records differs between resolvers using unauthenticated encryption and those using fully-authenticated encryption (described in [I-D.rescorla-dprive-adox-latest]). If the resolver is using unauthenticated encryption, the SVCB records do not need to be DNSSEC-signed.

DNSSEC validation of SVCB RRsets used strictly for this discovery mechanism is not mandated.

As described in [I-D.rescorla-dprive-adox-latest], these records may be in the resolver's cache because they came in the Additional section of a query for the NS records of a zone. This document does not rely on that feature being standardized or operationally present to work.

Because some authoritative servers or middleboxes are misconfigured, requests for unknown RRtypes might be ignored by them. Resolvers should be ready to deal with timeouts or other bad responses to their SVCB queries.

3. Resolving with Encryption

A resolver following this protocol MUST use SVCB records in its cache to decide whether to use classic DNS or encryption to contact authoritative servers for a zone. If any of the SVCB records in the cache for the authoritative servers for a zone are positive responses, the resolver uses any of those servers for encryption. A resolver MUST NOT attempt encryption for a server that has a negative response in its cache for the associated SVCB record.

If all of the SVCB records for the authoritative servers in the cache for a zone are negative responses, the resolver MUST use classic (unencrypted) DNS instead of encryption. Similarly, if none of the SVCB records for the authoritative servers in the cache have information about encrypted services as described in [I-D.schwartz-svcb-dns], the resolver MUST use classic (unencrypted) DNS instead of encryption.

If there are any SVCB records in the cache for the authoritative servers for a zone with a positive response, the resolver MUST try each indicated authoritative server using DNS with encryption until it successfully sets up a connection. The resolver only attempts to use the encrypted transports that are in the associated SVCB record for the authoritative server. Reasons for TLS failures are listed in Section 3.1.

After a DNS with encryption session is set up, the resolver uses that authoritative server for whatever query about the zone it was going to send. If a resolver cannot set up a DNS with encryption session with any of the authoritative servers, it MUST attempt to perform the resolution over classic (unencrypted) DNS as it would have without encryption.

A resolver SHOULD keep a DNS with encryption session to a particular server open if it expects to send additional queries to that server in a short period of time. If the server closes the DNS with encryption session, the resolver can possibly re-establish a DNS with encryption session using encrypted session resumption. [RFC7766] says "both clients and servers SHOULD support connection reuse" for TCP connections, and that advice could apply as well for DNS with encryption even though DNS with encryption has greater overhead for saving state.

Privacy-oriented resolvers (defined in [RFC8932]) following this protocol MUST NOT indicate that they are using encryption because this protocol is susceptible to on-path attacks.

3.1. Resolver Session Failures

The following are the reasons that a DNS with encryption session might fail to be set up:

- * The resolver receives a TCP RST response
- * The resolver does not receive replies to TCP or TLS setup (such as getting the TCP SYN message, the first TLS message, or completing TLS handshakes)
- * The TLS handshake gets a definitive failure
- * The encrypted session fails for reasons other than for authentication, such as incorrect algorithm choices or TLS record failures

4. Serving with Encryption

An operator of an authoritative server following this protocol SHOULD publish SVCB records as described in Section 2. If they cannot publish such records, the security properties of their authoritative servers will not be found. If an operator wants to test serving using encryption, they can publish SVCB records with short TTLs and then stop serving with encryption after removing the SVCB records and waiting for the TTLs to expire.

An operator of authoritative servers for a zone that is following this protocol MAY support encryption towards any IP address on which it offers service for classic DNS on port 53. It is acceptable for such an operator to only offer encryption on some of the named authoritative servers, such as when the operator is determining how far to roll out encrypted service.

A server MAY close an encrypted connection at any time. For example, it can close the session if it has not received a DNS query in a defined length of time. The server MAY close an encrypted session after it sends a DNS response; however, it might also want to keep the session open waiting for another DNS query from the resolver. [RFC7766] says "both clients and servers SHOULD support connection reuse" for TCP connections, and that advice could apply as well for DNS with encryption even though DNS with encryption has greater overhead for saving state.

5. Resolvers Reporting Errors to Authoritative Servers

Resolvers should have a method of telling authoritative servers that there are problems with the encrypted service they are offering. There is a proposal that the DNSOP Working Group adopt [I-D.arends-dns-error-reporting], which would enable such reporting.

((Clearly, more will need to go here.))

6. IANA Considerations

((Update registration for TCP/853 to also include ADoT))

((Maybe other updates for DoH and DoQ))

7. Security Considerations

The method described in this document explicitly allows a resolver to perform DNS communications over traditional unencrypted, unauthenticated DNS on port 53, if it cannot find an authoritative server that advertises that it supports encryption. The method described in this document explicitly allows a resolver using encryption to choose to allow unauthenticated encryption. In either of these cases, the resulting communication will be susceptible to obvious and well-understood attacks from an attacker in the path of the communications.

An authoritative server that wants to only serve data to resolvers that using fully-authenticated encryption as described in [I-D.rescorla-dprive-adox-latest] cannot differentiate between those resolvers and resolvers using the mechanisms described in this document.

8. Acknowledgements

Puneet Sood contributed many ideas to early drafts of this document.

The DPRIVE Working Group has contributed many ideas that keep shifting the focus and content of this document.

9. References

9.1. Normative References

[I-D.ietf-dnsop-rfc8499bis]

Hoffman, P. and K. Fujiwara, "DNS Terminology", Work in Progress, Internet-Draft, draft-ietf-dnsop-rfc8499bis-01, 20 November 2020, <<https://www.ietf.org/archive/id/draft-ietf-dnsop-rfc8499bis-01.txt>>.

[I-D.ietf-dnsop-svcb-https]

Schwartz, B., Bishop, M., and E. Nygren, "Service binding and parameter specification via the DNS (DNS SVCB and HTTPS RRs)", Work in Progress, Internet-Draft, draft-ietf-dnsop-svcb-https-04, 17 March 2021, <<https://www.ietf.org/archive/id/draft-ietf-dnsop-svcb-https-04.txt>>.

- [I-D.rescorla-dprive-adox-latest]
Pauly, T., Rescorla, E., Schinazi, D., and C. A. Wood,
"Signaling Authoritative DNS Encryption", Work in
Progress, Internet-Draft, draft-rescorla-dprive-adox-
latest-00, 26 February 2021,
<[https://www.ietf.org/archive/id/draft-rescorla-dprive-
adox-latest-00.txt](https://www.ietf.org/archive/id/draft-rescorla-dprive-adox-latest-00.txt)>.
- [I-D.schwartz-svcb-dns]
Schwartz, B., "Service Binding Mapping for DNS Servers",
Work in Progress, Internet-Draft, draft-schwartz-svcb-dns-
02, 17 February 2021, <[https://www.ietf.org/archive/id/
draft-schwartz-svcb-dns-02.txt](https://www.ietf.org/archive/id/draft-schwartz-svcb-dns-02.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>>.
- [RFC7435] Dukhovni, V., "Opportunistic Security: Some Protection
Most of the Time", RFC 7435, DOI 10.17487/RFC7435,
December 2014, <<https://www.rfc-editor.org/info/rfc7435>>.
- [RFC7766] Dickinson, J., Dickinson, S., Bellis, R., Mankin, A., and
D. Wessels, "DNS Transport over TCP - Implementation
Requirements", RFC 7766, DOI 10.17487/RFC7766, March 2016,
<<https://www.rfc-editor.org/info/rfc7766>>.
- [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>>.
- [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>>.
- [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>>.

9.2. Informative References

[I-D.arends-dns-error-reporting]

Arends, R. and M. Larson, "DNS Error Reporting", Work in Progress, Internet-Draft, draft-arends-dns-error-reporting-00, 30 October 2020, <<https://www.ietf.org/archive/id/draft-arends-dns-error-reporting-00.txt>>.

[I-D.ietf-dprive-dnssoquic]

Huitema, C., Mankin, A., and S. Dickinson, "Specification of DNS over Dedicated QUIC Connections", Work in Progress, Internet-Draft, draft-ietf-dprive-dnssoquic-02, 22 February 2021, <<https://www.ietf.org/archive/id/draft-ietf-dprive-dnssoquic-02.txt>>.

[RFC8932] Dickinson, S., Overeinder, B., van Rijswijk-Deij, R., and A. Mankin, "Recommendations for DNS Privacy Service Operators", BCP 232, RFC 8932, DOI 10.17487/RFC8932, October 2020, <<https://www.rfc-editor.org/info/rfc8932>>.

[RSO_STATEMENT]

"Statement on DNS Encryption", 2021, <https://root-servers.org/media/news/Statement_on_DNS_Encryption.pdf>.

Authors' Addresses

Paul Hoffman
ICANN

Email: paul.hoffman@icann.org

Peter van Dijk
PowerDNS

Email: peter.van.dijk@powerdns.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.annex-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.

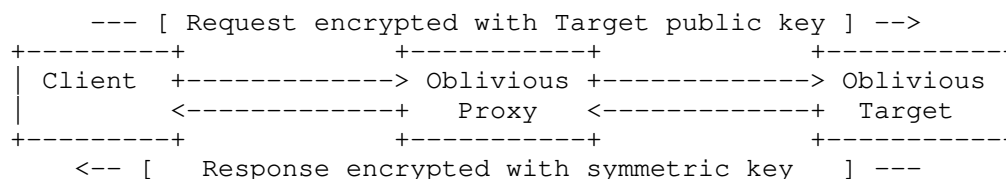


Figure 1: Oblivious DoH Exchange

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 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