DNS Zone Transfer-over-TLS
draft-ietf-dprive-xfr-over-tls-12

Abstract

DNS zone transfers are transmitted in clear text, which gives attackers the opportunity to collect the content of a zone by eavesdropping on network connections. The DNS Transaction Signature (TSIG) mechanism is specified to restrict direct zone transfer to authorized clients only, but it does not add confidentiality. This document specifies the use of TLS, rather than clear text, to prevent zone content collection via passive monitoring of zone transfers: XFR-over-TLS (XoT). Additionally, this specification updates RFC1995 and RFC5936 with respect to efficient use of TCP connections, and RFC7766 with respect to the recommended number of connections between a client and server for each transport.

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 27 November 2021.
Table of Contents

1. Introduction ................................................. 3
2. Document work via GitHub ................................. 6
3. Terminology ................................................. 6
4. Threat Model ............................................... 6
5. Design Considerations for XoT .......................... 7
6. Connection and Data Flows in Existing XFR Mechanisms .... 8
   6.1. AXFR Mechanism ........................................ 8
   6.2. IXFR Mechanism ........................................ 10
   6.3. Data Leakage of NOTIFY and SOA Message Exchanges .... 11
       6.3.1. NOTIFY ........................................... 11
       6.3.2. SOA ............................................. 12
7. Updates to existing specifications ....................... 12
   7.1. Update to RFC1995 for IXFR-over-TCP ................. 13
   7.2. Update to RFC5936 for AXFR-over-TCP ................. 14
   7.3. Updates to RFC1995 and RFC5936 for XFR-over-TCP .... 14
       7.3.1. Connection reuse ................................ 14
       7.3.2. AXFRs and IXFRs on the same connection ....... 15
       7.3.3. XFR limits ....................................... 15
       7.3.4. The edns-tcp-keepalive EDNS0 Option ........... 15
       7.3.5. Backwards compatibility .......................... 16
   7.4. Update to RFC7766 ..................................... 16
8. XoT specification ............................................. 17
   8.1. Connection establishment ............................. 17
   8.2. TLS versions .......................................... 17
   8.3. Port selection ......................................... 18
   8.4. High level XoT descriptions .......................... 18
   8.5. XoT transfers .......................................... 18
   8.6. XoT connections ........................................ 20
   8.7. XoT vs ADoT ............................................ 21
   8.8. Response RCODES ....................................... 22
   8.9. AXoT specifics ......................................... 22
       8.9.1. Padding AXoT responses ......................... 22
DNS has a number of privacy vulnerabilities, as discussed in detail in [I-D.ietf-dprive-rfc7626-bis]. Stub client to recursive resolver query privacy has received the most attention to date, with standards track documents for both DNS-over-TLS (DoT) [RFC7858] and DNS-over-HTTPS (DoH) [RFC8484], and a proposal for DNS-over-QUIC [I-D.ietf-dprive-dnsoquic]. There is ongoing work on DNS privacy requirements for exchanges between recursive resolvers and authoritative servers [I-D.ietf-dprive-phase2-requirements] and some suggestions for how signaling of DoT support by authoritative nameservers might work. However, there is currently no RFC that specifically defines recursive to authoritative DNS-over-TLS (ADoS).
[I-D.ietf-dprive-rfc7626-bis] established that stub client DNS query transactions are not public and needed protection, but on zone transfer [RFC1995] [RFC5936] it says only:

"Privacy risks for the holder of a zone (the risk that someone gets the data) are discussed in [RFC5936] and [RFC5155]."

In what way is exposing the full contents of a zone a privacy risk? The contents of the zone could include information such as names of persons used in names of hosts. Best practice is not to use personal information for domain names, but many such domain names exist. The contents of the zone could also include references to locations that allow inference about location information of the individuals associated with the zone’s organization. It could also include references to other organizations. Examples of this could be:

* Person-laptop.example.org
* MX-for-Location.example.org
* Service-tenant-from-another-org.example.org

Additionally, the full zone contents expose all the IP addresses of endpoints held in the DNS records which can make reconnaissance and attack targeting easier, particularly for IPv6 addresses or private networks. There may also be regulatory, policy or other reasons why the zone contents in full must be treated as private.

Neither of the RFCs mentioned in [I-D.ietf-dprive-rfc7626-bis] contemplates the risk that someone gets the data through eavesdropping on network connections, only via enumeration or unauthorized transfer as described in the following paragraphs.

Zone enumeration is trivially possible for DNSSEC zones which use NSEC; i.e. queries for the authenticated denial of existence records allow a client to walk through the entire zone contents. [RFC5155] specifies NSEC3, a mechanism to provide measures against zone enumeration for DNSSEC signed zones (a goal was to make it as hard to enumerate a DNSSEC signed zone as an unsigned zone). Whilst this is widely used, it has been demonstrated that zone walking is possible for precomputed NSEC3 using attacks such as those described in [NSEC3-attacks]. This prompted further work on an alternative mechanism for DNSSEC authenticated denial of existence - NSEC5 [I-D.vcelak-nsec5] - however questions remain over the practicality of this mechanism.
[RFC5155] does not address data obtained outside zone enumeration (nor does [I-D.vcelak-nsec5]). Preventing eavesdropping of zone transfers (this document) is orthogonal to preventing zone enumeration, though they aim to protect the same information.

[RFC5936] specifies using TSIG [RFC8945] for authorization of the clients of a zone transfer and for data integrity, but does not express any need for confidentiality, and TSIG does not offer encryption.

Section 8 of the NIST guide on ‘Secure Domain Name System (DNS) Deployment’ [nist-guide] discusses restricting access for zone transfers using ACLs and TSIG in more detail. It also discusses the possibility that specific deployments might choose to use a lower level network layer to protect zone transfers, e.g., IPSec.

It is noted that in all the common open source implementations such ACLs are applied on a per query basis (at the time of writing). Since requests typically occur on TCP connections authoritatives must therefore accept any TCP connection and then handling the authentication of each zone transfer (XFR) request individually.

Because both AXFR (authoritative transfer) and IXFR (incremental transfer) are typically carried out over TCP from authoritative DNS protocol implementations, encrypting zone transfers using TLS [RFC8499], based closely on DoT [RFC7858], seems like a simple step forward. This document specifies how to use TLS (1.3 or later) as a transport to prevent zone collection from zone transfers.

This document also updates the previous specifications for zone transfers to clarify and extend them, mainly with respect to TCP usage:

* RFC1995 (IXFR) and RFC5936 (AXFR) are both updated to add further specification on efficient use of TCP connections

* Section 6.2.2 of RFC7766 (DNS Transport over TCP - Implementation Requirements) is updated with a new recommendation about the number of connections between a client and server for each transport.
2. Document work via GitHub

[THIS SECTION TO BE REMOVED BEFORE PUBLICATION] The Github repository for this document is at https://github.com/hanzhang0116/hzpa-dprive-xfr-over-tls (https://github.com/hanzhang0116/hzpa-dprive-xfr-over-tls). 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.

3. Terminology

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

Privacy terminology is as described in Section 3 of [RFC6973].

DNS terminology is as described in [RFC8499]. Note that as in [RFC8499], the terms 'primary' and 'secondary' are used for two servers engaged in zone transfers.

DoT: DNS-over-TLS as specified in [RFC7858]

XFR-over-TCP: Used to mean both IXFR-over-TCP [RFC1995] and AXFR-over-TCP [RFC5936].

XoT: XFR-over-TLS mechanisms as specified in this document which apply to both AXFR-over-TLS and IXFR-over-TLS

AXoT: AXFR-over-TLS

IXoT: IXFR over-TLS

4. Threat Model

The threat model considered here is one where the current contents and size of the zone are considered sensitive and should be protected during transfer.

The threat model does not, however, consider the existence of a zone, the act of zone transfer between two entities, nor the identities of the nameservers hosting a zone (including both those acting as hidden primaries/secondaries or directly serving the zone) as sensitive information. The proposed mechanism does not attempt to obscure such information. The reasons for this include:
* much of this information can be obtained by various methods, including active scanning of the DNS

* an attacker who can monitor network traffic can relatively easily infer relations between nameservers simply from traffic patterns, even when some or all of the traffic is encrypted (in terms of current deployments)

The model does not consider attacks on the mechanisms that trigger a zone transfer, e.g., NOTIFY messages.

It is noted that simply using XoT will indicate a desire by the zone owner that the contents of the zone remain confidential and so could be subject to blocking (e.g., via blocking of port 853) if an attacker had such capabilities. However this threat is likely true of any such mechanism that attempts to encrypt data passed between nameservers, e.g., IPsec.

5. Design Considerations for XoT

The following principles were considered in the design for XoT:

* Confidentiality. Clearly using an encrypted transport for zone transfers will defeat zone content leakage that can occur via passive surveillance.

* Authentication. Use of single or mutual TLS (mTLS) authentication (in combination with access control lists (ACLs)) can complement and potentially be an alternative to TSIG.

* Performance.

  - Existing AXFR and IXFR mechanisms have the burden of backwards compatibility with older implementations based on the original specifications in [RFC1034] and [RFC1035]. For example, some older AXFR servers don’t support using a TCP connection for multiple AXFR sessions or XFRs of different zones because they have not been updated to follow the guidance in [RFC5936]. Any implementation of XoT would obviously be required to implement optimized and interoperable transfers as described in [RFC5936], e.g., transfer of multiple zones over one connection.

  - Current usage of TCP for IXFR is sub-optimal in some cases i.e. connections are frequently closed after a single IXFR.
6. Connection and Data Flows in Existing XFR Mechanisms

The original specification for zone transfers in [RFC1034] and [RFC1035] was based on a polling mechanism: a secondary performed a periodic query for the SOA (start of zone authority) record (based on the refresh timer) to determine if an AXFR was required.

[RFC1995] and [RFC1996] introduced the concepts of IXFR and NOTIFY respectively, to provide for prompt propagation of zone updates. This has largely replaced AXFR where possible, particularly for dynamically updated zones.

[RFC5936] subsequently redefined the specification of AXFR to improve performance and interoperability.

In this document we use the term "XFR mechanism" to describe the entire set of message exchanges between a secondary and a primary that concludes in a successful AXFR or IXFR request/response. This set may or may not include

* NOTIFY messages
* SOA queries
* Fallback from IXFR to AXFR
* Fallback from IXFR-over-UDP to IXFR-over-TCP

The term is used to encompass the range of permutations that are possible and is useful to distinguish the ‘XFR mechanism’ from a single XFR request/response exchange.

6.1. AXFR Mechanism

The figure below provides an outline of an AXFR mechanism including NOTIFYs.
1. An AXFR is often (but not always) preceded by a NOTIFY (over UDP) from the primary to the secondary. A secondary may also initiate an AXFR based on a refresh timer or scheduled/triggered zone maintenance.

2. The secondary will normally (but not always) make a SOA query to the primary to obtain the serial number of the zone held by the primary.

3. If the primary serial is higher than the secondary’s serial (using Serial Number Arithmetic [RFC1982]), the secondary makes an AXFR request (over TCP) to the primary after which the AXFR data flows in one or more AXFR responses on the TCP connection. [RFC5936] defines this specific step as an ‘AXFR session’ i.e. as an AXFR query message and the sequence of AXFR response messages returned for it.
[RFC5936] re-specified AXFR providing additional guidance beyond that provided in [RFC1034] and [RFC1035] and importantly specified that AXFR must use TCP as the transport protocol.

Additionally, sections 4.1, 4.1.1 and 4.1.2 of [RFC5936] provide improved guidance for AXFR clients and servers with regard to re-use of TCP connections for multiple AXFRs and AXFRs of different zones. However [RFC5936] was constrained by having to be backwards compatible with some very early basic implementations of AXFR. For example, it outlines that the SOA query can also happen on this connection. However, this can cause interoperability problems with older implementations that support only the trivial case of one AXFR per connection.

6.2. IXFR Mechanism

The figure below provides an outline of the IXFR mechanism including NOTIFYs.

```
Secondary                   Primary
 NOTIFY                      <----- NOTIFY ---->  UDP
                                  -----------
                                  NOTIFY Response

                                 <----- SOA Request ---->  UDP or TCP
                                 -----------
                                 SOA Response

                                 <----- IXFR Request ---->  UDP or TCP
                                 -----------
                                 IXFR Response
                                 (Zone data)

                                 <----- IXFR Request ---->  Retry over
                                 -----------  TCP if
                                 IXFR Response  required
                                 (Zone data)

                                 <----- IXFR Request ---->  ---
                                 -----------
                                 IXFR Response
                                 (Zone data)
```

Figure 2. IXFR Mechanism
1. An IXFR is normally (but not always) preceded by a NOTIFY (over UDP) from the primary to the secondary. A secondary may also initiate an IXFR based on a refresh timer or scheduled/triggered zone maintenance.

2. The secondary will normally (but not always) make a SOA query to the primary to obtain the serial number of the zone held by the primary.

3. If the primary serial is higher than the secondaries serial (using Serial Number Arithmetic [RFC1982]), the secondary makes an IXFR request to the primary after which the primary sends an IXFR response.

[RFC1995] specifies that Incremental Transfer may use UDP if the entire IXFR response can be contained in a single DNS packet, otherwise, TCP is used. In fact it says:

"Thus, a client should first make an IXFR query using UDP."

So there may be a fourth step above where the client falls back to IXFR-over-TCP. There may also be a additional step where the secondary must fall back to AXFR because, e.g., the primary does not support IXFR.

However it is noted that most widely used open source authoritative nameserver implementations (including both [BIND] and [NSD]) do IXFR using TCP by default in their latest releases. For BIND, TCP connections are sometimes used for SOA queries but in general they are not used persistently and close after an IXFR is completed.

6.3. Data Leakage of NOTIFY and SOA Message Exchanges

This section presents a rationale for considering encrypting the other messages in the XFR mechanism.

Since the SOA of the published zone can be trivially discovered by simply querying the publicly available authoritative servers, leakage of this resource record (RR) via such a direct query is not discussed in the following sections.

6.3.1. NOTIFY

Unencrypted NOTIFY messages identify configured secondaries on the primary.

[RFC1996] also states:
"If ANCOUNT>0, then the answer section represents an unsecure hint at the new RRset for this (QNAME,QCLASS,QTYPE).

But since the only QTYPE for NOTIFY defined at the time of this writing is SOA, this does not pose a potential leak.

6.3.2. SOA

For hidden XFR servers (either primaries or secondaries), an SOA response directly from that server only additionally leaks the degree of SOA serial number lag of any downstream secondary of that server.

7. Updates to existing specifications

For convenience, the term ‘XFR-over-TCP’ is used in this document to mean both IXFR-over-TCP and AXFR-over-TCP and therefore statements that use that term update both [RFC1995] and [RFC5936], and implicitly also apply to XoT. Differences in behavior specific to XoT are discussed in Section 8.

Both [RFC1995] and [RFC5936] were published sometime before TCP was considered a first class transport for DNS. [RFC1995], in fact, says nothing with respect to optimizing IXFRs over TCP or re-using already open TCP connections to perform IXFRs or other queries. Therefore, there arguably is an implicit assumption that a TCP connection is used for one and only one IXFR request. Indeed, many major open source implementations take this approach (at the time of this writing). And whilst [RFC5936] gives guidance on connection re-use for AXFR, it pre-dates more recent specifications describing persistent TCP connections (e.g., [RFC7766], [RFC7828]), and AXFR implementations again often make less than optimal use of open connections.

Given this, new implementations of XoT will clearly benefit from specific guidance on TCP/TLS connection usage for XFR, because this will:

* result in more consistent XoT implementations with better interoperability
* remove any need for XoT implementations to support legacy behavior for XoT connections that XFR-over-TCP implementations have historically often supported

Therefore this document updates both the previous specifications for XFR-over-TCP ([RFC1995] and [RFC5936]) to clarify that
* Implementations MUST use [RFC7766] (DNS Transport over TCP –
  Implementation Requirements) to optimize the use of TCP
  connections.

* Whilst RFC7766 states that 'DNS clients SHOULD pipeline their
  queries' on TCP connections, it did not distinguish between XFRs
  and other queries for this behavior. It is now recognized that
  XFRs are not as latency sensitive as other queries, and can be
  significantly more complex for clients to handle, both because of
  the large amount of state that must be kept and because there may
  be multiple messages in the responses. For these reasons, it is
  clarified here that a valid reason for not pipelining queries is
  when they are all XFR queries i.e. clients sending multiple XFRs
  MAY choose not to pipeline those queries. Clients that do not
  pipeline XFR queries, therefore, have no additional requirements
  to handle out-of-order or intermingled responses (as described
  later), since they will never receive them.

* Implementations SHOULD use [RFC7828] (The edns-tcp-keepalive EDNS0
  Option) to manage persistent connections (which is more flexible
  than using just fixed timeouts).

The following sections include detailed clarifications on the updates
to XFR behavior implied in [RFC7766] and how the use of [RFC7828]
applies specifically to XFR exchanges. They also discuss how IXFR
and AXFR can reuse the same TCP connection.

For completeness, we also mention here the recent specification of
extended DNS error (EDE) codes [RFC8914]. For zone transfers, when
returning REFUSED to a zone transfer request from an 'unauthorized'
client (e.g., where the client is not listed in an ACL for zone
transfers or does not sign the request with a valid TSIG key), the
extended DNS error code 18 (Prohibited) can also be sent.

7.1. Update to RFC1995 for IXFR-over-TCP

For clarity – an IXFR-over-TCP server compliant with this
specification MUST be able to handle multiple concurrent IXoT
requests on a single TCP connection (for the same and different
zones) and SHOULD send the responses as soon as they are available,
which might be out-of-order compared to the requests.
7.2. Update to RFC5936 for AXFR-over-TCP

For clarity - an AXFR-over-TCP server compliant with this specification MUST be able to handle multiple concurrent AXoT sessions on a single TCP connection (for the same and different zones). The response streams for concurrent AXFRs MAY be intermingled and AXFR-over-TCP clients compliant with this specification which pipeline AXFR requests MUST be able to handle this.

7.3. Updates to RFC1995 and RFC5936 for XFR-over-TCP

7.3.1. Connection reuse

As specified, XFR-over-TCP clients SHOULD re-use any existing open TCP connection when starting any new XFR request to the same primary, and for issuing SOA queries, instead of opening a new connection. The number of TCP connections between a secondary and primary SHOULD be minimized (also see Section 7.4).

Valid reasons for not re-using existing connections might include:

* as already noted in [RFC7766], separate connections for different zones might be preferred for operational reasons. In this case, the number of concurrent connections for zone transfers SHOULD be limited to the total number of zones transferred between the client and server.

* reaching a configured limit for the number of outstanding queries or XFR requests allowed on a single TCP connection

* the message ID pool has already been exhausted on an open connection

* a large number of timeouts or slow responses have occurred on an open connection

* an edns-tcp-keepalive EDNS0 option with a timeout of 0 has been received from the server and the client is in the process of closing the connection (see Section 7.3.4)

If no TCP connections are currently open, XFR clients MAY send SOA queries over UDP or a new TCP connection.
7.3.2. AXFRs and IXFRs on the same connection

Neither [RFC1995] nor [RFC5936] explicitly discuss the use of a single TCP connection for both IXFR and AXFR requests. [RFC5936] does make the general statement:

"Non-AXFR session traffic can also use an open TCP connection."

We clarify here that implementations capable of both AXFR and IXFR and compliant with this specification SHOULD

* use the same TCP connection for both AXFR and IXFR requests to the same primary

* pipeline such requests (if they pipeline XFR requests in general) and MAY intermingle them

* send the response(s) for each request as soon as they are available i.e. responses MAY be sent intermingled

For some current implementations adding all the above functionality would introduce significant code complexity. In such a case, there will need to be an assessment of the trade-off between that and the performance benefits of the above for XFR.

7.3.3. XFR limits

The server MAY limit the number of concurrent IXFRs, AXFRs or total XFR transfers in progress, or from a given secondary, to protect server resources. Servers SHOULD return SERVFAIL if this limit is hit, since it is a transient error and a retry at a later time might succeed (there is no previous specification for this behavior).

7.3.4. The edns-tcp-keepalive EDNS0 Option

XFR clients that send the edns-tcp-keepalive EDNS0 option on every XFR request provide the server with maximum opportunity to update the edns-tcp-keepalive timeout. The XFR server may use the frequency of recent XFRs to calculate an average update rate as input to the decision of what edns-tcp-keepalive timeout to use. If the server does not support edns-tcp-keepalive, the client MAY keep the connection open for a few seconds ([RFC7766] recommends that servers use timeouts of at least a few seconds).

Whilst the specification for EDNS0 [RFC6891] does not specifically mention AXFRs, it does say
"If an OPT record is present in a received request, compliant responders MUST include an OPT record in their respective responses."

We clarify here that if an OPT record is present in a received AXFR request, compliant responders MUST include an OPT record in each of the subsequent AXFR responses. Note that this requirement, combined with the use of edns-tcp-keepalive, enables AXFR servers to signal the desire to close a connection (when existing transactions have competed) due to low resources by sending an edns-tcp-keepalive EDNS0 option with a timeout of 0 on any AXFR response. This does not signal that the AXFR is aborted, just that the server wishes to close the connection as soon as possible.

7.3.5. Backwards compatibility

Certain legacy behaviors were noted in [RFC5936], with provisions that implementations may want to offer options to fallback to legacy behavior when interoperating with servers known to not support [RFC5936]. For purposes of interoperability, IXFR and AXFR implementations may want to continue offering such configuration options, as well as supporting some behaviors that were underspecified prior to this work (e.g., performing IXFR and AXFRs on separate connections). However, XoT connections should have no need to do so.

7.4. Update to RFC7766

[RFC7766] made general implementation recommendations with regard to TCP/TLS connection handling:

"To mitigate the risk of unintentional server overload, DNS clients MUST take care to minimize the number of concurrent TCP connections made to any individual server. It is RECOMMENDED that for any given client/server interaction there SHOULD be no more than one connection for regular queries, one for zone transfers, and one for each protocol that is being used on top of TCP (for example, if the resolver was using TLS). However, it is noted that certain primary/secondary configurations with many busy zones might need to use more than one TCP connection for zone transfers for operational reasons (for example, to support concurrent transfers of multiple zones)."
Whilst this recommends a particular behavior for the clients using TCP, it does not relax the requirement for servers to handle 'mixed' traffic (regular queries and zone transfers) on any open TCP/TLS connection. It also overlooks the potential that other transports might want to take the same approach with regard to using separate connections for different purposes.

This specification updates the above general guidance in [RFC7766] to provide the same separation of connection purpose (regular queries and zone transfers) for all transports being used on top of TCP.

Therefore, it is RECOMMENDED that for each protocol used on top of TCP in any given client/server interaction, there SHOULD be no more than one connection for regular queries and one for zone transfers.

As an illustration, it could be imagined that in future such an interaction could hypothetically include one or all of the following:

* one TCP connection for regular queries
* one TCP connection for zone transfers
* one TLS connection for regular queries
* one TLS connection for zone transfers
* one DoH connection for regular queries
* one DoH connection for zone transfers

Section 7.3.1 has provided specific details of reasons where more than one connection for a given transport might be required for zone transfers from a particular client.

8. XoT specification

8.1. Connection establishment

During connection establishment the Application-Layer Protocol Negotiation (ALPN) token "dot" [DoT-ALPN] MUST be selected in the TLS handshake.

8.2. TLS versions

All implementations of this specification MUST use only TLS 1.3 [RFC8446] or later.
8.3. Port selection

The connection for XoT SHOULD be established using port 853, as specified in [RFC7858], unless there is mutual agreement between the secondary and primary to use a port other than port 853 for XoT. There MAY be agreement to use different ports for AXoT and IXoT, or for different zones.

8.4. High level XoT descriptions

It is useful to note that in XoT, it is the secondary that initiates the TLS connection to the primary for a XFR request, so that in terms of connectivity, the secondary is the TLS client and the primary the TLS server.

The figure below provides an outline of the AXoT mechanism including NOTIFYs.
Figure 3. AXoT Mechanism

The figure below provides an outline of the IXoT mechanism including NOTIFYs.
8.5. XoT transfers

For a zone transfer between two end points to be considered protected with XoT all XFR requests and response for that zone MUST be sent over TLS connections where at a minimum:

* the client MUST authenticate the server by use of an authentication domain name using a Strict Privacy Profile, as described in [RFC8310]

* the server MUST validate the client is authorized to request or proxy a zone transfer by using one or both of the following methods:
  - Mutual TLS (mTLS)
- an IP based ACL (which can be either per-message or per-
  connection) combined with a valid TSIG/SIG(0) signature on the
  XFR request

If only one method is selected then mTLS is preferred because it
provides strong cryptographic protection at both endpoints.

Authentication mechanisms are discussed in full in Section 10 and the
rationale for the above requirement in Section 11. Transfer group
policies are discussed in Section 12.

8.6. XoT connections

The details in Section 7 about, e.g., persistent connections and XFR
message handling are fully applicable to XoT connections as well.
However, any behavior specified here takes precedence for XoT.

If no TLS connections are currently open, XoT clients MAY send SOA
queries over UDP or TCP, or TLS.

8.7. XoT vs ADoT

As noted earlier, there is currently no specification for encryption
of connections from recursive resolvers to authoritative servers.
Some authoritatives are experimenting with ADoT and opportunistic
encryption has also been raised as a possibility; it is therefore
highly likely that use of encryption by authoritative servers will
evolve in the coming years.

This raises questions in the short term with regard to TLS connection
and message handling for authoritative servers. In particular, there
is likely to be a class of authoritatives that wish to use XoT in the
near future with a small number of configured secondaries, but that
do not wish to support DoT for regular queries from recursives in
that same time frame. These servers have to potentially cope with
probing and direct queries from recursives and from test servers, and
also potential attacks that might wish to make use of TLS to overload
the server.

[RFC5936] clearly states that non-AXFR session traffic can use an
open TCP connection, however, this requirement needs to be re-
evaluated when considering applying the same model to XoT. Proposing
that a server should also start responding to all queries received
over TLS just because it has enabled XoT would be equivalent to
defining a form of authoritative DoT. This specification does not
propose that, but it also does not prohibit servers from answering
queries unrelated to XFR exchanges over TLS. Rather, this
specification simply outlines in later sections:
* how XoT implementations should utilize EDE codes in response to queries on TLS connections they are not willing to answer (see Section 8.8)

* the operational and policy options that a XoT server operator has with regard to managing TLS connections and messages (see Appendix A)

8.8. Response RCODES

XoT clients and servers MUST implement EDE codes. If a XoT server receives non-XoT traffic it is not willing to answer on a TLS connection, it SHOULD respond with REFUSED and the extended DNS error code 21 - Not Supported [RFC8914]. XoT clients should not send any further queries of this type to the server for a reasonable period of time (for example, one hour) i.e., long enough that the server configuration or policy might be updated.

Historically, servers have used the REFUSED RCODE for many situations, and so clients often had no detailed information on which to base an error or fallback path when queries were refused. As a result, the client behavior could vary significantly. XoT servers that refuse queries must cater for the fact that client behavior might vary from continually retrying queries regardless of receiving REFUSED to every query, or at the other extreme clients may decide to stop using the server over any transport. This might be because those clients are either non-XoT clients or do not implement EDE codes.

8.9. AXoT specifics

8.9.1. Padding AXoT responses

The goal of padding AXoT responses is two fold:

* to obfuscate the actual size of the transferred zone to minimize information leakage about the entire contents of the zone.

* to obfuscate the incremental changes to the zone between SOA updates to minimize information leakage about zone update activity and growth.

Note that the re-use of XoT connections for transfers of multiple different zones slightly complicates any attempt to analyze the traffic size and timing to extract information. Also, effective padding may require state to be kept as zones may grow and/or shrink over time.
It is noted here that, depending on the padding policies eventually developed for XoT, the requirement to obfuscate the total zone size might require a server to create 'empty' AXoT responses. That is, AXoT responses that contain no RR's apart from an OPT RR containing the EDNS(0) option for padding. For example, without this capability the maximum size that a tiny zone could be padded to would theoretically be limited if there had to be a minimum of 1 RR per packet.

However, as with existing AXFR, the last AXoT response message sent MUST contain the same SOA that was in the first message of the AXoT response series in order to signal the conclusion of the zone transfer.

[RFC5936] says:

"Each AXFR response message SHOULD contain a sufficient number of RRs to reasonably amortize the per-message overhead, up to the largest number that will fit within a DNS message (taking the required content of the other sections into account, as described below)."

'Empty' AXoT responses generated in order to meet a padding requirement will be exceptions to the above statement. For flexibility, future proofing and in order to guarantee support for future padding policies, we state here that secondary implementations MUST be resilient to receiving padded AXoT responses, including 'empty' AXoT responses that contain only an OPT RR containing the EDNS(0) option for padding.

Recommendation of specific policies for padding AXoT responses are out of scope for this specification. Detailed considerations of such policies and the trade-offs involved are expected to be the subject of future work.

8.10. IXoT specifics

8.10.1. Condensation of responses

[RFC1995] says condensation of responses is optional and MAY be done. Whilst it does add complexity to generating responses, it can significantly reduce the size of responses. However any such reduction might be offset by increased message size due to padding. This specification does not update the optionality of condensation for XoT responses.
8.10.2. Fallback to AXFR

Fallback to AXFR can happen, for example, if the server is not able to provide an IXFR for the requested SOA. Implementations differ in how long they store zone deltas and how many may be stored at any one time.

Just as with IXFR-over-TCP, after a failed IXFR a IXoT client SHOULD request the AXFR on the already open XoT connection.

8.10.3. Padding of IXoT responses

The goal of padding IXoT responses is to obfuscate the incremental changes to the zone between SOA updates to minimize information leakage about zone update activity and growth. Both the size and timing of the IXoT responses could reveal information.

IXFR responses can vary greatly in size from the order of 100 bytes for one or two record updates, to tens of thousands of bytes for large dynamic DNSSEC signed zones. The frequency of IXFR responses can also depend greatly on if and how the zone is DNSSEC signed.

In order to guarantee support for future padding policies, we state here that secondary implementations MUST be resilient to receiving padded IXoT responses.

Recommendation of specific policies for padding IXoT responses are out of scope for this specification. Detailed considerations of such padding policies, the use of traffic obfuscation techniques (such as ‘dummy’ traffic), and the trade-offs involved are expected to be the subject of future work.

8.11. Name compression and maximum payload sizes

It is noted here that name compression [RFC1035] can be used in XFR responses to reduce the size of the payload, however, the maximum value of the offset that can be used in the name compression pointer structure is 16384. For some DNS implementations this limits the size of an individual XFR response used in practice to something around the order of 16kB. In principle, larger payload sizes can be supported for some responses with more sophisticated approaches (e.g., by pre-calculating the maximum offset required).
Implementations may wish to offer options to disable name compression for XoT responses to enable larger payloads. This might be particularly helpful when padding is used since minimizing the payload size is not necessarily a useful optimization in this case and disabling name compression will reduce the resources required to construct the payload.

9. Multi-primary Configurations

This model can provide flexibility and redundancy particularly for IXFR. A secondary will receive one or more NOTIFY messages and can send an SOA to all of the configured primaries. It can then choose to send an XFR request to the primary with the highest SOA (or based on other criteria, e.g., RTT).

When using persistent connections, the secondary may have a XoT connection already open to one or more primaries. Should a secondary preferentially request an XFR from a primary to which it already has an open XoT connection or the one with the highest SOA (assuming it doesn’t have a connection open to it already)?

Two extremes can be envisaged here. The first one can be considered a ‘preferred primary connection’ model. In this case, the secondary continues to use one persistent connection to a single primary until it has reason not to. Reasons not to might include the primary repeatedly closing the connection, long query/response RTTs on transfers or the SOA of the primary being an unacceptable lag behind the SOA of an alternative primary.

The other extreme can be considered a ‘parallel primary connection’ model. Here, a secondary could keep multiple persistent connections open to all available primaries and only request XFRs from the primary with the highest serial number. Since normally the number of secondaries and primaries in direct contact in a transfer group is reasonably low this might be feasible if latency is the most significant concern.

Recommendation of a particular scheme is out of scope of this document, but implementations are encouraged to provide configuration options that allow operators to make choices about this behavior.
10. Authentication mechanisms

To provide context to the requirements in Section 8.5, this section provides a brief summary of some of the existing authentication and validation mechanisms (both transport independent and TLS specific) that are available when performing zone transfers. Section 11 then discusses in more details specifically how a combination of TLS authentication, TSIG and IP based ACLs interact for XoT.

We classify the mechanisms based on the following properties:

* 'Data Origin Authentication' (DO): Authentication that the DNS message originated from the party with whom credentials were shared, and of the data integrity of the message contents (the originating party may or may not be party operating the far end of a TCP/TLS connection in a 'proxy' scenario).

* 'Channel Confidentiality' (CC): Confidentiality of the communication channel between the client and server (i.e. the two end points of a TCP/TLS connection) from passive surveillance.

* 'Channel Authentication' (CA): Authentication of the identity of party to whom a TCP/TLS connection is made (this might not be a direct connection between the primary and secondary in a proxy scenario).

10.1. TSIG

TSIG [RFC8945] provides a mechanism for two or more parties to use shared secret keys which can then be used to create a message digest to protect individual DNS messages. This allows each party to authenticate that a request or response (and the data in it) came from the other party, even if it was transmitted over an unsecured channel or via a proxy.

Properties: Data origin authentication

10.2. SIG(0)

SIG(0) [RFC2931] similarly provides a mechanism to digitally sign a DNS message but uses public key authentication, where the public keys are stored in DNS as KEY RRs and a private key is stored at the signer.

Properties: Data origin authentication

10.3. TLS
10.3.1. Opportunistic TLS

Opportunistic TLS for DoT is defined in [RFC8310] and can provide a defense against passive surveillance, providing on-the-wire confidentiality. Essentially

* clients that know authentication information for a server SHOULD try to authenticate the server
* however they MAY fallback to using TLS without authentication and
* they MAY fallback to using cleartext if TLS is not available.

As such, it does not offer a defense against active attacks (e.g., an on path active attacker on the connection from client to server), and is not considered as useful for XoT.

Properties: None guaranteed.

10.3.2. Strict TLS

Strict TLS for DoT [RFC8310] requires that a client is configured with an authentication domain name (and/or SPKI pinset) that MUST be used to authenticate the TLS handshake with the server. If authentication of the server fails, the client will not proceed with the connection. This provides a defense for the client against active surveillance, providing client-to-server authentication and end-to-end channel confidentiality.

Properties: Channel confidentiality and channel authentication (of the server).

10.3.3. Mutual TLS

This is an extension to Strict TLS [RFC8310] which requires that a client is configured with an authentication domain name (and/or SPKI pinset) and a client certificate. The client offers the certificate for authentication by the server and the client can authenticate the server the same way as in Strict TLS. This provides a defense for both parties against active surveillance, providing bi-directional authentication and end-to-end channel confidentiality.

Properties: Channel confidentiality and mutual channel authentication.
10.4. IP Based ACL on the Primary

Most DNS server implementations offer an option to configure an IP based Access Control List (ACL), which is often used in combination with TSIG based ACLs to restrict access to zone transfers on primary servers on a per query basis.

This is also possible with XoT, but it must be noted that, as with TCP, the implementation of such an ACL cannot be enforced on the primary until an XFR request is received on an established connection.

As discussed in Appendix A, an IP based per connection ACL could also be implemented where only TLS connections from recognized secondaries are accepted.

Properties: Channel authentication of the client.

10.5. ZONEMD

For completeness, we also describe Message Digest for DNS Zones (ZONEMD) [RFC8976] here. The message digest is a mechanism that can be used to verify the content of a standalone zone. It is designed to be independent of the transmission channel or mechanism, allowing a general consumer of a zone to do origin authentication of the entire zone contents. Note that the current version of [RFC8976] states:

"As specified herein, ZONEMD is impractical for large, dynamic zones due to the time and resources required for digest calculation. However, The ZONEMD record is extensible so that new digest schemes may be added in the future to support large, dynamic zones."

It is complementary but orthogonal the above mechanisms; and can be used in conjunction with XoT, but is not considered further here.

11. XoT authentication

It is noted that zone transfer scenarios can vary from a simple single primary/secondary relationship where both servers are under the control of a single operator to a complex hierarchical structure which includes proxies and multiple operators. Each deployment scenario will require specific analysis to determine which combination of authentication methods are best suited to the deployment model in question.
The XoT authentication requirement specified in Section 8.5 addresses the issue of ensuring that the transfers are encrypted between the two endpoints directly involved in the current transfers. The following table summarizes the properties of a selection of the mechanisms discussed in Section 10. The two letter acronyms for the properties are used below and (S) indicates the secondary and (P) indicates the primary.

| Method         | DO(S) | CC(S) | CA(S) | DO(P) | CC(P) | CA(P) |
|----------------+-------+-------+-------+-------+-------+-------|
| Strict TLS     |       |   Y   |   Y   |       |   Y   |       |
| Mutual TLS     |       |   Y   |   Y   |       |   Y   |   Y   |
| ACL on primary |       |       |       |       |       |   Y   |
| TSIG           |   Y   |       |       |   Y   |       |       |

Table 1

Table 1: Properties of Authentication methods for XoT

Based on this analysis it can be seen that:

* Using just mutual TLS can be considered a standalone solution since both end points are cryptographically authenticated

* Using secondary side Strict TLS with a primary side IP ACL and TSIG/SIG(0) combination provides sufficient protection to be acceptable.

Using just an IP ACL could be susceptible to attacks that can spoof TCP IP addresses, using TSIG/SIG(0) alone could be susceptible to attacks that were able to capture such messages should they be accidentally sent in clear text by any server with the key.

12. Policies for Both AXoT and IXoT

Whilst the protection of the zone contents in a transfer between two end points can be provided by the XoT protocol, the protection of all the transfers of a given zone requires operational administration and policy management.

We call the entire group of servers involved in XFR for a particular set of zones (all the primaries and all the secondaries) the ’transfer group’.
In order to assure the confidentiality of the zone information, the entire transfer group MUST have a consistent policy of using XoT. If any do not, this is a weak link for attackers to exploit. For clarification, this means that within any transfer group both AXFRs and IXFRs for a zone MUST all use XoT.

An individual zone transfer is not considered protected by XoT unless both the client and server are configured to use only XoT and the overall zone transfer is not considered protected until all members of the transfer group are configured to use only XoT with all other transfers servers (see Section 13).

A XoT policy MUST specify if

* mutual TLS is used and/or

* a IP ACL and TSIG/SIG(0) combination is used

Since this may require configuration of a number of servers who may be under the control of different operators the desired consistency could be hard to enforce and audit in practice.

Certain aspects of the Policies can be relatively easily tested independently, e.g., by requesting zone transfers without TSIG, from unauthorized IP addresses or over cleartext DNS. Other aspects such as if a secondary will accept data without a TSIG digest or if secondaries are using Strict as opposed to Opportunistic TLS are more challenging.

The mechanics of co-ordinating or enforcing such policies are out of the scope of this document but may be the subject of future operational guidance.

13. Implementation Considerations

Server implementations may want to also offer options that allow ACLs on a zone to specify that a specific client can use either XoT or TCP. This would allow for flexibility while clients are migrating to XoT.

Client implementations may similarly want to offer options to cater for the multi-primary case where the primaries are migrating to XoT.

14. Operational Considerations

If the options described in Section 13 are available, such configuration options MUST only be used in a ‘migration mode’, and therefore should be used with great care.
It is noted that use of a TLS proxy in front of the primary server is a simple deployment solution that can enable server side XoT.

15. IANA Considerations

None.

16. Implementation Status

[TWO 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].

A summary of current behavior and implementation status can be found here: XoT implementation status (https://dnsprivacy.org/wiki/display/DP/DNSPrivacy+Implementation+Status#DNSPrivacyImplementationStatus-XFR/XoTImplementationstatus)

Specific recent activity includes:

1. The 1.9.2 version of Unbound (https://github.com/NLnetLabs/unbound/blob/release-1.9.2/doc/Changelog) includes an option to perform AXoT (instead of AXFR-over-TCP).

2. There are currently open pull requests against NSD to implement

   1. Connection re-use by default during XFR-over-TCP (https://github.com/NLnetLabs/nsd/pull/145)

   2. Client side XoT (https://github.com/NLnetLabs/nsd/pull/149)

3. Version 9.17.7 of BIND contained an initial implementation of DoT, implementation of XoT is planned for early 2021 (https://gitlab.isc.org/isc-projects/bind9/-/issues/1784)

   Both items 1. and 2. listed above require the client (secondary) to authenticate the server (primary) using a configured authentication domain name if XoT is used.

17. Security Considerations

This document specifies a security measure against a DNS risk: the risk that an attacker collects entire DNS zones through eavesdropping on clear text DNS zone transfers.
This does not mitigate:

* the risk that some level of zone activity might be inferred by observing zone transfer sizes and timing on encrypted connections (even with padding applied), in combination with obtaining SOA records by directly querying authoritative servers.

* the risk that hidden primaries might be inferred or identified via observation of encrypted connections.

* the risk of zone contents being obtained via zone enumeration techniques.

Security concerns of DoT are outlined in [RFC7858] and [RFC8310].

18. Acknowledgements

The authors thank Tony Finch, Benno Overeinder, Shumon Huque and Tim Wicinski and many other members of DPRIVE for review and discussions.

The authors particularly thank Peter van Dijk, Ondrej Sury, Brian Dickson and several other open source DNS implementers for valuable discussion and clarification on the issue associated with pipelining XFR queries and handling out-of-order/intermingled responses.

19. Contributors

Significant contributions to the document were made by:

Han Zhang
Salesforce
San Francisco, CA
United States

Email: hzhang@salesforce.com

20. Changelog

[THIS SECTION TO BE REMOVED BEFORE PUBLICATION]

draft-ietf-dprive-xfr-over-tls-12

* Changes from IESG review

* Add section 8.1 on the requirement to use the DoT ALPN

* Modify the one of the options for validation of a client from just an IP ACL to a combination of IP ACL and TSIG/SIG(0)
- Update Abstract and Introduction with clear descriptions of how earlier specifications are updated
- Add reference on NSEC3 attacks
- Justify use of SHOULD in sections 7.3.2 and 7.3.3.
- Clarify the Appendix is non-normative
- Numerous typos and editorial improvements.
- Use xml2rfc v3 (some format changes occur as a result)

draft-ietf-dprive-xfr-over-tls-11
* Fix definition update missed in -10
draft-ietf-dprive-xfr-over-tls-10
* Address issued raised from IETF Last Call
draft-ietf-dprive-xfr-over-tls-09
* Address issued raised in the AD review
draft-ietf-dprive-xfr-over-tls-08
* RFC2845 -> (obsoleted by) RFC8945
* I-D.ietf-dnsop-dns-zone-digest -> RFC8976
* Minor editorial changes + email update
draft-ietf-dprive-xfr-over-tls-07
* Reference RFC7942 in the implementation status section
* Convert the URIs that will remain on publication to references
* Correct typos in acknowledgments
draft-ietf-dprive-xfr-over-tls-06
* Update text relating to pipelining and connection reuse after WGLC comments.
* Add link to implementation status matrix
* Various typos

draft-ietf-dprive-xfr-over-tls-05
* Remove the open questions that received no comments.
* Add more detail to the implementation section

draft-ietf-dprive-xfr-over-tls-04
* Add Github repository
* Fix typos and references and improve layout.

draft-ietf-dprive-xfr-over-tls-03
* Remove propose to use ALPN
* Clarify updates to both RFC1995 and RFC5936 by adding specific sections on this
* Add a section on the threat model
* Convert all SVG diagrams to ASCII art
* Add discussions on concurrency limits
* Add discussions on Extended DNS error codes
* Re-work authentication requirements and discussion
* Add appendix discussion TLS connection management

draft-ietf-dprive-xfr-over-tls-02
* Significantly update descriptions for both AXoT and IXoT for message and connection handling taking into account previous specifications in more detail
* Add use of ALPN and limitations on traffic on XoT connections.
* Add new discussions of padding for both AXoT and IXoT
* Add text on SIG(0)
* Update security considerations
* Move multi-primary considerations to earlier as they are related to connection handling

draft-ietf-dprive-xfr-over-tls-01

* Minor editorial updates

* Add requirement for TLS 1.3. or later

draft-ietf-dprive-xfr-over-tls-00

* Rename after adoption and reference update.

* Add placeholder for SIG(0) discussion

* Update section on ZONEMD

draft-hzpa-dprive-xfr-over-tls-02

* Substantial re-work of the document.

draft-hzpa-dprive-xfr-over-tls-01

* Editorial changes, updates to references.

draft-hzpa-dprive-xfr-over-tls-00

* Initial commit

[-@?I-D.ietf-tls-esni]

21. Normative References


22. Informative References


[I-D.ietf-tls-esni]

[I-D.vcelak-nsec5]

[NSD]

[NSEC3-attacks]


Appendix A.  XoT server connection handling

This appendix provides a non-normative outline of the pros and cons of XoT server connection handling options.

For completeness, it is noted that an earlier version of the specification suggested using a XoT specific ALPN to negotiate TLS connections that supported only a limited set of queries (SOA, XRFs), however, this did not gain support. Reasons given included additional code complexity and the fact that XoT and ADoT are both DNS wire format and so should share the "dot" ALPN.

A.1.  Only listen on TLS on a specific IP address

Obviously a nameserver which hosts a zone and services queries for the zone on an IP address published in an NS record may wish to use a separate IP address for listening on TLS for XoT, only publishing that address to its secondaries.

Pros: Probing of the public IP address will show no support for TLS. ACLs will prevent zone transfer on all transports on a per query basis.

Cons: Attackers passively observing traffic will still be able to observe TLS connections to the separate address.

A.2.  Client specific TLS acceptance

 Primaries that include IP based ACLs and/or mutual TLS in their authentication models have the option of only accepting TLS connections from authorized clients. This could be implemented using a proxy or directly in DNS implementation.
Pros: Connection management happens at setup time. The maximum number of TLS connections a server will have to support can be easily assessed. Once the connection is accepted the server might well be willing to answer any query on that connection since it is coming from a configured secondary and a specific response policy on the connection may not be needed (see below).

Cons: Currently, none of the major open source DNS authoritative implementations support such an option.

A.3. SNI based TLS acceptance

Primaries could also choose to only accept TLS connections based on an SNI that was published only to their secondaries.

Pros: Reduces the number of accepted connections.

Cons: As above. Also, this is not a recommended use of SNI. For SNIs sent in the clear, this would still allow attackers passively observing traffic to potentially abuse this mechanism. The use of Encrypted Client Hello [I-D.ietf-tls-esni] may be of use here.

A.4. Transport specific response policies

Some primaries might rely on TSIG/SIG(0) combined with per-query IP based ACLs to authenticate secondaries. In this case the primary must accept all incoming TLS/TCP connections and then apply a transport-specific response policy on a per query basis.

As an aside, whilst [RFC7766] makes a general purpose distinction in the advice to clients about their usage of connections (between regular queries and zone transfers) this is not strict and nothing in the DNS protocol prevents using the same connection for both types of traffic. Hence a server cannot know the intention of any client that connects to it, it can only inspect the messages it receives on such a connection and make per-query decisions about whether or not to answer those queries.

Example policies a XoT server might implement are:

* strict: REFUSE all queries on TLS connections except SOA and authorized XFR requests

* moderate: REFUSE all queries on TLS connections until one is received that is signed by a recognized TSIG/SIG(0) key, then answer all queries on the connection after that
* complex: apply a heuristic to determine which queries on a TLS connections to REFUSE

* relaxed: answer all non-XoT queries on all TLS connections with the same policy applied to TCP queries

Pros: Allows for flexible behavior by the server that could be changed over time.

Cons: The server must handle the burden of accepting all TLS connections just to perform XFRs with a small number of secondaries. Client behavior to REFUSED response is not clearly defined (see Section 8.8). Currently, none of the major open source DNS authoritative implementations offer an option for different response policies in different transports (but such functionality could potentially be implemented using a proxy).

A.4.1. SNI based response policies

In a similar fashion, XoT servers might use the presence of an SNI in the client hello to determine which response policy to initially apply to the TLS connections.

Pros: This has to potential to allow a clean distinction between a XoT service and any future DoT based service for answering recursive queries.

Cons: As above.

Authors’ Addresses

Willem Toorop
NLnet Labs
Science Park 400
Amsterdam

Email: willem@nlnetlabs.nl

Sara Dickinson
Sinodun IT
Magdalen Centre
Oxford Science Park
Oxford
OX4 4GA
United Kingdom

Email: sara@sinodun.com
Shivan Sahib  
Brave Software  
Vancouver, BC  
Canada  

Email: shivankaulsahib@gmail.com

Pallavi Aras  
Salesforce  
Herndon, VA,  
United States  

Email: paras@salesforce.com

Allison Mankin  
Salesforce  
Herndon, VA,  
United States  

Email: allison.mankin@gmail.com
Abstract

This document specifies a way to signal the usage of DoT, and the pinned keys for that DoT usage, in authoritative servers. This signal lives on the parent side of delegations, in DS records. To ensure easy deployment, the signal is defined in terms of (C)DNSKEY.
1. Introduction

Even quite recently, DNS was a completely unencrypted protocol, with no protection against snooping. In the past few years, this landscape has shifted. The connections between stubs and resolvers are now often protected by DoT, DoH, or other protocols that provide privacy.

This document introduces a way to signal, from the parent side of a delegation, that the name servers hosting the delegated zone support DoT, and with which TLS/X.509 keys. This proposal does not require any changes in authoritative name servers, other than (possibly through an external process) actually offering DoT on port 853.
DNS registry operators (such as TLD operators) also need to make no changes, unless they filter uploaded DNSKEY/DS records on acceptable DNSKEY algorithms, in which case they would need to add algorithm TBD to that list.

This document was inspired by, and borrows heavily from, [I-D.bretelle-dprive-dot-for-insecure-delegations].

2. Document work

This document lives on GitHub (https://github.com/PowerDNS/parent-signals-dot/blob/master/draft-vandijk-dprive-ds-dot-signal-and-pin/draft-vandijk-dprive-ds-dot-signal-and-pin.md); 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.

3. Conventions and Definitions

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.

CDNSKEY record as defined in [RFC7344][RFC8078]

DS record as defined in [RFC4034]

DNSKEY record as defined in [RFC4034]
4. Summary

To enable the signaling of DoT a new DNSKEY algorithm type TBD is added. If a resolver with support for TBD encounters a DS record with the DNSKEY algorithm type TBD it MUST connect to the authoritative servers for this domain via DoT. It MUST use the hashes attached to the DS records with DNSKEY algorithm type TBD to check whether the public key supplied by the authoritative nameserver during the TLS handshake is valid. If the DoT connection is unsuccessful or the public key supplied by the server does not match any of the DS digests, the resolver MUST NOT fall back to unencrypted Do53. For resolvers that are willing to probe for protocol support (DNS over HTTPS, DNS over QUIC), a fallback to other encrypted protocols is allowed if they can satisfy the key pin. This means that if a DS for algo TBD is present, and no name servers satisfy the pin requirement, the response returned to the client is SERVFAIL because no name servers for the domain were available to answer the questions.

A domain MAY have more than one DS record with DNSKEY algorithm TBD. A resolver with support for TBD should then try to verify the public key supplied by the authoritative nameserver against every supplied DS record. Multiple records can be used to support multiple DS digest types, multiple TLS key algorithms, different keys for each authoritative, and for key rollovers. In case of an algorithm or key rollover the new DS record should be added to all served domains before the new key is deployed on the authoritatives. To allow for emergency rollovers, having a standby DS record for all domains with a private key securely stored offline can be a valid strategy.

The pseudo DNSKEY record (when considered in wire format) MUST contain the ([RFC4648] 4.) DER SubjectPublicKeyInfo as defined in [RFC5280] 4.1.2.7. Since the cert provided by the TLS server over the wire is already DER encoded this makes for easy validation. (In the DNSKEY presentation format, the Public Key field contains the Base64 encoding of the DER SPKI, which is equivalent to the SPKI in PEM format minus the header and footer.) The pseudo DNSKEY algorithm type TBD is algorithm agnostic, like the TLSA record, since the DER encoded data already contains information about the used algorithm. Algorithm support SHOULD be handled at the TLS handshake level, which means a DNS application SHOULD NOT need to be aware of the algorithm used by its TLS library. The pseudo DNSKEY record MUST NOT be present in the zone. The procedure for hashing the pseudo DNSKEY record is the same as for a normal DNSKEY as defined in RFC4034 5.1.4.
As DNSKEY algorithm TBD is not meant to be used for Zone Signing, the existing ZONE and SEP flags do not mean anything. This specification statically defines the flags value as 257 for optimal compatibility with existing registry operations.

The pseudo DNSKEY type can be used in CDNSKEY and CDS (as defined in [RFC7344]) records. These records MAY be present in the zone.

For those familiar with TLSA ([RFC6698]), key matching for this protocol is identical to that provided by "TLSA 3 1 0" for (C)DNSKEY. For the DS case, key matching is similar to "TLSA 3 1 x" where x is not zero, except that the rest of the (C)DNSKEY, including the owner name, gets prepended before hashing.

5. Example

This section will take you through the various parts of this specification, by example.

We assume that we are working with a domain "example.com." with one name server, "ns.example.com".

5.1. Generating and placing the (C)DNSKEY/DS records

[NOTE: this section uses ‘225’ instead of ‘TBD’ because otherwise the code does not work. We need to fix this before publication.]

We will walk you through the CDNSKEY/DS generation, demonstrating it in terms of basic shell scripting and some common tools.

First, we extract the SubjectPublicKeyInfo:

```bash
dev/null |
```

```
openssl s_client -connect ns.example.com:853 < /dev/null 
| openssl x509 -noout -pubkey > pubkey.pem
```

This gives us a file "pubkey.pem" that looks like this (abridged):

```
-----BEGIN PUBLIC KEY-----
MIICIjANBgkqhkiG9w0BAQEFAAOCAg8AMIICCgKCAgEAxH2a6NxIcw5527b04kKy...
71AWASNoX2GQh7eaQPDD9i8CAwEAAQ==
-----END PUBLIC KEY-----
```

To turns this into a CDNSKEY:

1. remove the header and footer
2. remove all newlines
In other words:

```bash
openssl s_client -connect ns.example.com:853 </dev/null \
   | openssl x509 -noout -pubkey \
   | sed '1d;$d' \
   | tr -d '\n'
```

Then we prepend

e.g.,

```
example.com. IN CDNSKEY 257 3 225
```

so that we end up with

```
example.com. IN CDNSKEY 257 3 225 MIICIj...AAQ==
```

If your registry accepts CDNSKEY, or DNSKEY via EPP, you are done — you can get your DS placed.

To generate the DS, do something like this:

```bash
echo example.com. IN DNSKEY 257 3 225 MIICIj...AAQ== \
   | ldns-key2ds -f -n -2 /dev/stdin
echo example.com. 3600 IN DS 7573 225 2 fcb6...c26c
```

6. Implementation

The subsection titles in this section attempt to follow the terminology from [RFC8499] in as far as it has suitable terms. 'Implementation' is understood to mean both 'code changes' and 'operational changes' here.

6.1. Authoritative server changes

This specification defines no changes to query processing in authoritative servers.

If DoT-signaling DS records are published for a zone, all name servers for the zone (from both the parent-side and child-side NS RRsets) SHOULD offer DoT service on port 853, and when they do, they SHOULD do so using keys present in the DS RRset. However, there are potential cases where this is not possible, like having multiple DNS providers. In this case the name servers that do not support DoT MUST respond with a RST response or similar on the port tcp/853 to prevent name resolution slowdowns.
6.2. Validating resolver changes

If a resolver successfully uses DoT with a nameserver as specified in
this document for one domain, it MAY assume DoT is always available
from that nameserver for questions for another domain. However, it
MUST NOT assume that the connection is properly pinned for that other
domain unless there is a DS record available for that other domain it
is currently resolving.

A validating resolver that supports this draft will perform the
following actions when a DS record with algorithm TBD is encountered:

1. Connects to the name server on port 853.

2. During TLS handshake, the resolver will extract the
SubjectPublicKeyInfo from the certificate.

3. Construct an in-memory DNSKEY record [RFC4034] section 2 with its
   fields set as follow:
   * Flags: 257
   * Protocol: 3
   * Algorithm: TBD
   * Public Key: The wire-format SubjectPublicKeyInfo

4. Get the list of Digest Type for DS records obtained from the
   parent with algorithm TBD

5. For each digest type from the list, compute the DS record of the
   previously computed DNSKEY, its fields are set as follow:
   * Key Tag: computed from DNS key using [RFC4034] appendix B
   * Algorithm: TBD
   * Digest Type: the current Digest Type we are computing the DS
     for.
   * Digest: Following [RFC4034] section 5.1.4, compute the digest
     of owner name | previously computed DNSKEY’s RDATA.

6. Test the computed DS record against all the supplied DS records
   until a match is encountered.
7. If any computed DS record matches a DS record in the DS record set we got from the parent, the connection is successfully authenticated.

6.3. Stub resolver changes

This specification defines no changes to stub resolvers.

6.4. Zone validator changes

This section covers both the 'online' type of zone validator, such as Zonemaster, and the 'offline full zone' type, such as "validdns" and "dnssec-verify".

Checks for child DNSKEY records based on parent DS records algorithms, and checks for zone RRSIG algorithms based on DNSKEY algorithms, MUST not be applied to algorithm TBD. [NOTE: rephrase this in terms of the Zone Signing column at https://www.iana.org/assignments/dns-sec-alg-numbers/dns-sec-alg-numbers.xhtml (https://www.iana.org/assignments/dns-sec-alg-numbers/ dns-sec-alg-numbers.xhtml) ?]

DNSKEY validity checks MAY verify correct DER syntax in DNSKEY Public Key content when algorithm is TBD.

6.5. Domain registry changes

Any pre-delegation or periodic checks by registries should honor the Zone validator changes from the previous section.

This specification trusts that appearance of TBD in https://www.iana.org/assignments/dns-sec-alg-numbers/dns-sec-alg-numbers.xhtml (https://www.iana.org/assignments/dns-sec-alg-numbers/ dns-sec-alg-numbers.xhtml) will eventually lead registries to accept DS/(C)DNSKEY submissions for algorithm TBD.

Registries that limit the total number of DS records for a delegation SHOULD consider having a separate limit for algorithm TBD DS records, as their management is separate from actual DNSSEC key management.

7. Security Considerations

This document defines a way to convey, authoritatively, that resolvers must use DoT to do their queries to the name servers for a certain zone. By doing so, that exchange gains confidentiality, data integrity, peer entity authentication.
8. Implementation Status

[RFC Editor: please remove this section before publication]

This section records the status of known implementations of the protocol defined by this specification at the time of posting of this document, and is based on a proposal described in [RFC6982]. The description of implementations in this section is intended to assist the IETF in its decision processes in progressing drafts to RFCs. Please note that the listing of any individual implementation here does not imply endorsement by the IETF. Furthermore, no effort has been spent to verify the information presented here that was supplied by IETF contributors. This is not intended as, and must not be construed to be, a catalog of available implementations or their features. Readers are advised to note that other implementations may exist.

According to RFC 6982, "this will allow reviewers and working groups to assign due consideration to documents that have the benefit of running code, which may serve as evidence of valuable experimentation and feedback that have made the implemented protocols more mature. It is up to the individual working groups to use this information as they see fit".

8.1. PoC

Some Proof of Concept code showing the generation of the (C)DNSKEY, and the subsequent hashing by a client (which should match one of the DS records with algo TBD), in Python and Go, is available at https://github.com/PowerDNS/parent-signals-dot/tree/master/poc (https://github.com/PowerDNS/parent-signals-dot/tree/master/poc)

9. Design Considerations

[RFC Editor: please remove this section before publication]

A protocol design is nothing without a clear statement of the constraints it was designed to meet, and perhaps a list of other constraints it meets by accident.

We humbly acknowledge Petr Spacek’s excellent summary of the 'nice properties' this protocol has (https://mailarchive.ietf.org/arch/msg/dns-privacy/_Zf5TGVAcUFPRrQ_7o_NPnmni2s/) as a source of inspiration for this section.

Manu’s DSPKI proposal had the following excellent properties:
* no extra roundtrips (assuming DSPKI came 'for free' with
delegations like DS records do today)

* downgrade resistance

* simple protocol, no indirections

It also had this one very important undesirable property:

* a new RRtype with 'special' behaviour would be pretty much
  impossible to deploy

In various private and public discussions, it was quickly realised
that fitting this into the actual DS record would solve that problem. The first obvious answer to that is 'just assign some numbers and do in DS what DSPKI defined in its own type'. Petr Spacek and others pointed out that this would be incompatible with 'DNSKEY-style' registries, i.e. those that demand DNSKEY, not DS, in their communications (those communications being either EPP, some registry-specific protocol, or CDNSKEY). In other words, a protocol that would not allow the DS to be hashed 'the usual way' from a DNSKEY would not go far, as many registries are slow to update their software even _just_ for a couple of new numbers in an IANA registry.

With that, the puzzle was clear. We need some format to signal and pin DoT with a DNSKEY, in such a way that a DS can be hashed from it without software changes in parties such as registries, and such that that DS is enough for a resolver to validate a TLS connection.

Eventually we realised that a resolver could take the TLS SubjectPublicKeyInfo, construct a 'pseudo' DNSKEY from it, and hash that into a DS. This resolves the one bad property of DSPKI (deployability without changing every auth, resolver, and registry stack in the world).

The design constraints we felt we must meet with this protocol were:

* deployability without demanding massive software changes or even 'flag days'

* downgrade resistance

And we feel we have met those. The other positive properties of DSPKI (simplicity, no extra roundtrips) have been kept intact more by accident than by strong intention.
We can understand that several people are saying that this is hacky (we do not even disagree), and that TLSA should have been used. However, we feel that any TLSA-based protocol we can imagine would be a lot more complex, and therefore prone to breakage which might be hard to debug. It would also be very easy to accidentally introduce chicken-and-egg problems with a more indirect approach. Note that we are responding to imagined TLSA-based protocols here. If a draft appears for a TLSA-based approach to DoT signaling/pinning, we would love to read it. Depending on what that draft looks like, it might even make sense to have that protocol _and_ the protocol described in this document.

The biggest downside to this DS-based protocol is that a change in TLS keys on an auth may require DS updates for thousands or even hundreds of thousands of domains. This issue is partially mitigated by allowing backup keys to be part of those DS sets. Furthermore we hope that efforts from Cloudflare and others for shortening the path between auth operator and domain registrar one day work out. Those efforts are focused on NSset updates and DS updates for DNSSEC validation, but they would also aid key rollovers for this protocol greatly.

10. IANA Considerations


The following entries have been added to the registry:

```
+--------------+----------------+
| Number       | TBD            |
| Description  | DoT signal+pin |
| Mnemonic     | DOTPIN         |
| Zone signing | N              |
| Trans sec.   | N              |
| Reference    | RFC TBD2       |
+--------------+----------------+
```

11. Acknowledgements

The authors would like to thank the following individuals for their useful input: Job Snijders, Maik Zumstrull, Petr Spacek, Pieter Lexis, Ralph Dolmans, Remi Gacogne, Seth Arnold, and Vladimir Cunat.

12. Normative References
13. Informative References

Appendix A.  Document history

A.1.  Changes between -00 and -01

1.  Lots of clarifying text that does not change any semantics, including:
   *  a section on how resolvers would actually use this protocol.
   *  we made it clearer that multiple DS records for a delegation are allowed, and why you would want this.

2.  DNSKEY flags are now set to 257, because it looks like this will make it a lot easier for many registries to accept the records.

3.  Added a 'Design Considerations' section to give some background to why this protocol is what it is.

We have tried to do a review of this protocol against the requirement of the DPRIVE phase 2 document. You can find this review (which might be updated outside of revisions of this draft or the phase 2 draft) in our GitHub repo (https://github.com/PowerDNS/parent-signals-dot/blob/master/draft-vandijk-dprive-ds-dot-signal-and-pin/yardsticks/draft-ietf-dprive-phase2-requirements-01.md).

Authors’ Addresses

Peter van Dijk
PowerDNS
Den Haag
Netherlands

Email: peter.van.dijk@powerdns.com