Specification for DNS over Transport Layer Security (TLS)

Abstract

   This document describes the use of Transport Layer Security (TLS) to
   provide privacy for DNS. Encryption provided by TLS eliminates
   opportunities for eavesdropping and on-path tampering with DNS
   queries in the network, such as discussed in RFC 7626. In addition,
   this document specifies two usage profiles for DNS over TLS and
   provides advice on performance considerations to minimize overhead
   from using TCP and TLS with DNS.

   This document focuses on securing stub-to-recursive traffic, as per
   the charter of the DPRIVE Working Group. It does not prevent future
   applications of the protocol to recursive-to-authoritative traffic.

Status of This Memo

   This is an Internet Standards Track document.

   This document is a product of the Internet Engineering Task Force
   (IETF). It represents the consensus of the IETF community. It has
   received public review and has been approved for publication by the
   Internet Engineering Steering Group (IESG). Further information on
   Internet Standards is available in Section 2 of RFC 5741.

   Information about the current status of this document, any errata,
   and how to provide feedback on it may be obtained at
   http://www.rfc-editor.org/info/rfc7858.

Copyright Notice

Table of Contents

1. Introduction

   Today, nearly all DNS queries [RFC1034] [RFC1035] are sent
   unencrypted, which makes them vulnerable to eavesdropping by an
   attacker that has access to the network channel, reducing the privacy
   of the querier. Recent news reports have elevated these concerns,
   and recent IETF work has specified privacy considerations for DNS
   [RFC7626].

   Prior work has addressed some aspects of DNS security, but until
   recently, there has been little work on privacy between a DNS client
   and server. DNS Security Extensions (DNSSEC) [RFC4033] provide
   response integrity by defining mechanisms to cryptographically sign
   zones, allowing end users (or their first-hop resolver) to verify
   replies are correct. By intention, DNSSEC does not protect request
   and response privacy. Traditionally, either privacy was not
   considered a requirement for DNS traffic or it was assumed that
   network traffic was sufficiently private; however, these perceptions
   are evolving due to recent events [RFC7258].

   Other work that has offered the potential to encrypt between DNS
   clients and servers includes DNSCurve [DNSCurve], DNSCrypt
   [DNSCRYPT-WEBSITE], Confidential DNS [CONFIDENTIAL-DNS], and IPSECA
   [IPSECA]. In addition to the present specification, the DPRIVE
   Working Group has also adopted a proposal for DNS over Datagram
   Transport Layer Security (DTLS) [DNSoD].

   This document describes using DNS over TLS on a well-known port and
   also offers advice on performance considerations to minimize
   overheads from using TCP and TLS with DNS.

   Initiation of DNS over TLS is very straightforward. By establishing
   a connection over a well-known port, clients and servers expect and
   agree to negotiate a TLS session to secure the channel. Deployment
   will be gradual. Not all servers will support DNS over TLS and the
   well-known port might be blocked by some firewalls. Clients will be
   expected to keep track of servers that support TLS and those that
   don't. Clients and servers will adhere to the TLS implementation
   recommendations and security considerations of [BCP195].

   The protocol described here works for queries and responses between
   stub clients and recursive servers. It might work equally between
   recursive clients and authoritative servers, but this application of
   the protocol is out of scope for the DNS PRIVate Exchange (DPRIVE)
   Working Group per its current charter.

This document describes two profiles in Section 4 that provide
different levels of assurance of privacy: an opportunistic privacy
profile and an out-of-band key-pinned privacy profile. It is
expected that a future document based on [TLS-DTLS-PROFILES] will
further describe additional privacy profiles for DNS over both TLS
and DTLS.

An earlier draft version of this document described a technique for
upgrading a DNS-over-TCP connection to a DNS-over-TLS session with,
essentially, "STARTTLS for DNS". To simplify the protocol, this
document now only uses a well-known port to specify TLS use, omitting
the upgrade approach. The upgrade approach no longer appears in this
document, which now focuses exclusively on the use of a well-known
port for DNS over TLS.

2. Key Words

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT",
"SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this
document are to be interpreted as described in RFC 2119 [RFC2119].

3. Establishing and Managing DNS-over-TLS Sessions

3.1. Session Initiation

By default, a DNS server that supports DNS over TLS MUST listen for
and accept TCP connections on port 853, unless it has mutual
agreement with its clients to use a port other than 853 for DNS over
TLS. In order to use a port other than 853, both clients and servers
would need a configuration option in their software.

By default, a DNS client desiring privacy from DNS over TLS from a
particular server MUST establish a TCP connection to port 853 on the
server, unless it has mutual agreement with its server to use a port
other than port 853 for DNS over TLS. Such another port MUST NOT be
port 53 but MAY be from the "first-come, first-served" port range.
This recommendation against use of port 53 for DNS over TLS is to
avoid complication in selecting use or non-use of TLS and to reduce
risk of downgrade attacks. The first data exchange on this TCP
connection MUST be the client and server initiating a TLS handshake
using the procedure described in [RFC5246].

DNS clients and servers MUST NOT use port 853 to transport cleartext
DNS messages. DNS clients MUST NOT send and DNS servers MUST NOT
respond to cleartext DNS messages on any port used for DNS over TLS
(including, for example, after a failed TLS handshake). There are
significant security issues in mixing protected and unprotected data,

and for this reason, TCP connections on a port designated by a given
server for DNS over TLS are reserved purely for encrypted
communications.

DNS clients SHOULD remember server IP addresses that don't support
DNS over TLS, including timeouts, connection refusals, and TLS
handshake failures, and not request DNS over TLS from them for a
reasonable period (such as one hour per server). DNS clients
following an out-of-band key-pinned privacy profile (Section 4.2) MAY
be more aggressive about retrying DNS-over-TLS connection failures.

3.2. TLS Handshake and Authentication

Once the DNS client succeeds in connecting via TCP on the well-known
port for DNS over TLS, it proceeds with the TLS handshake [RFC5246],
following the best practices specified in [BCP195].

The client will then authenticate the server, if required. This
document does not propose new ideas for authentication. Depending on
the privacy profile in use (Section 4), the DNS client may choose not
to require authentication of the server, or it may make use of a
trusted Subject Public Key Info (SPKI) Fingerprint pin set.

After TLS negotiation completes, the connection will be encrypted and
is now protected from eavesdropping.

3.3. Transmitting and Receiving Messages

All messages (requests and responses) in the established TLS session
MUST use the two-octet length field described in Section 4.2.2 of
[RFC1035]. For reasons of efficiency, DNS clients and servers SHOULD
pass the two-octet length field, and the message described by that
length field, to the TCP layer at the same time (e.g., in a single
"write" system call) to make it more likely that all the data will be
transmitted in a single TCP segment ([RFC7766], Section 8).

In order to minimize latency, clients SHOULD pipeline multiple
queries over a TLS session. When a DNS client sends multiple queries
to a server, it should not wait for an outstanding reply before
sending the next query ([RFC7766], Section 6.2.1.1).

Since pipelined responses can arrive out of order, clients MUST match
responses to outstanding queries on the same TLS connection using the
Message ID. If the response contains a Question Section, the client
MUST match the QNAME, QCLASS, and QTYPE fields. Failure by clients
to properly match responses to outstanding queries can have serious
consequences for interoperability ([RFC7766], Section 7).

3.4. Connection Reuse, Close, and Reestablishment

   For DNS clients that use library functions such as "getaddrinfo()"
   and "gethostbyname()", current implementations are known to open and
   close TCP connections for each DNS query. To avoid excess TCP
   connections, each with a single query, clients SHOULD reuse a single
   TCP connection to the recursive resolver. Alternatively, they may
   prefer to use UDP to a DNS-over-TLS-enabled caching resolver on the
   same machine that then uses a system-wide TCP connection to the
   recursive resolver.

   In order to amortize TCP and TLS connection setup costs, clients and
   servers SHOULD NOT immediately close a connection after each
   response. Instead, clients and servers SHOULD reuse existing
   connections for subsequent queries as long as they have sufficient
   resources. In some cases, this means that clients and servers may
   need to keep idle connections open for some amount of time.

   Proper management of established and idle connections is important to
   the healthy operation of a DNS server. An implementor of DNS over
   TLS SHOULD follow best practices for DNS over TCP, as described in
   [RFC7766]. Failure to do so may lead to resource exhaustion and
   denial of service.

   Whereas client and server implementations from the era of [RFC1035]
   are known to have poor TCP connection management, this document
   stipulates that successful negotiation of TLS indicates the
   willingness of both parties to keep idle DNS connections open,
   independent of timeouts or other recommendations for DNS over TCP
   without TLS. In other words, software implementing this protocol is
   assumed to support idle, persistent connections and be prepared to
   manage multiple, potentially long-lived TCP connections.

   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. Current work in this area may also
   assist DNS-over-TLS clients and servers in selecting useful timeout
   values [RFC7828] [TDNS].

   Clients and servers that keep idle connections open MUST be robust to
   termination of idle connection by either party. As with current DNS
   over TCP, DNS servers MAY close the connection at any time (perhaps
   due to resource constraints). As with current DNS over TCP, clients
   MUST handle abrupt closes and be prepared to reestablish connections
   and/or retry queries.

When reestablishing a DNS-over-TCP connection that was terminated, as
discussed in [RFC7766], TCP Fast Open [RFC7413] is of benefit.
Underlining the requirement for sending only encrypted DNS data on a
DNS-over-TLS port (Section 3.2), when using TCP Fast Open, the client
and server MUST immediately initiate or resume a TLS handshake
(cleartext DNS MUST NOT be exchanged). DNS servers SHOULD enable
fast TLS session resumption [RFC5077], and this SHOULD be used when
reestablishing connections.

When closing a connection, DNS servers SHOULD use the TLS close-
notify request to shift TCP TIME-WAIT state to the clients.
Additional requirements and guidance for optimizing DNS over TCP are
provided by [RFC7766].

4. Usage Profiles

   This protocol provides flexibility to accommodate several different
   use cases. This document defines two usage profiles: (1)
   opportunistic privacy and (2) out-of-band key-pinned authentication
   that can be used to obtain stronger privacy guarantees if the client
   has a trusted relationship with a DNS server supporting TLS.
   Additional methods of authentication will be defined in a forthcoming
   document [TLS-DTLS-PROFILES].

4.1. Opportunistic Privacy Profile

   For opportunistic privacy, analogous to SMTP opportunistic security
   [RFC7435], one does not require privacy, but one desires privacy when
   possible.

   With opportunistic privacy, a client might learn of a TLS-enabled
   recursive DNS resolver from an untrusted source. One possible
   example flow would be if the client used the DHCP DNS server option
   [RFC3646] to discover the IP address of a TLS-enabled recursive and
   then attempted DNS over TLS on port 853. With such a discovered DNS
   server, the client might or might not validate the resolver. These
   choices maximize availability and performance, but they leave the
   client vulnerable to on-path attacks that remove privacy.

   Opportunistic privacy can be used by any current client, but it only
   provides privacy when there are no on-path active attackers.

4.2. Out-of-Band Key-Pinned Privacy Profile

   The out-of-band key-pinned privacy profile can be used in
   environments where an established trust relationship already exists
   between DNS clients and servers (e.g., stub-to-recursive in
   enterprise networks, actively maintained contractual service

relationships, or a client using a public DNS resolver). The result
of this profile is that the client has strong guarantees about the
privacy of its DNS data by connecting only to servers it can
authenticate. Operators of a DNS-over-TLS service in this profile
are expected to provide pins that are specific to the service being
pinned (i.e., public keys belonging directly to the end entity or to
a service-specific private certificate authority (CA)) and not to a
public key(s) of a generic public CA.

In this profile, clients authenticate servers by matching a set of
SPKI Fingerprints in an analogous manner to that described in
[RFC7469]. With this out-of-band key-pinned privacy profile, client
administrators SHOULD deploy a backup pin along with the primary pin,
for the reasons explained in [RFC7469]. A backup pin is especially
helpful in the event of a key rollover, so that a server operator
does not have to coordinate key transitions with all its clients
simultaneously. After a change of keys on the server, an updated pin
set SHOULD be distributed to all clients in some secure way in
preparation for future key rollover. The mechanism for an
out-of-band pin set update is out of scope for this document.

Such a client will only use DNS servers for which an SPKI Fingerprint
pin set has been provided. The possession of a trusted pre-deployed
pin set allows the client to detect and prevent person-in-the-middle
and downgrade attacks.

However, a configured DNS server may be temporarily unavailable when
configuring a network. For example, for clients on networks that
require authentication through web-based login, such authentication
may rely on DNS interception and spoofing. Techniques such as those
used by DNSSEC-trigger [DNSSEC-TRIGGER] MAY be used during network
configuration, with the intent to transition to the designated DNS
provider after authentication. The user MUST be alerted whenever
possible that the DNS is not private during such bootstrap.

Upon successful TLS connection and handshake, the client computes the
SPKI Fingerprints for the public keys found in the validated server's
certificate chain (or in the raw public key, if the server provides
that instead). If a computed fingerprint exactly matches one of the
configured pins, the client continues with the connection as normal.
Otherwise, the client MUST treat the SPKI validation failure as a
non-recoverable error. Appendix A provides a detailed example of how
this authentication could be performed in practice.

Implementations of this privacy profile MUST support the calculation
of a fingerprint as the SHA-256 [RFC6234] hash of the DER-encoded
ASN.1 representation of the SPKI of an X.509 certificate.

Implementations MUST support the representation of a SHA-256
fingerprint as a base64-encoded character string [RFC4648].
Additional fingerprint types MAY also be supported.

5. Performance Considerations

DNS over TLS incurs additional latency at session startup. It also
requires additional state (memory) and increased processing (CPU).

Latency: Compared to UDP, DNS over TCP requires an additional round-
   trip time (RTT) of latency to establish a TCP connection. TCP
   Fast Open [RFC7413] can eliminate that RTT when information exists
   from prior connections. The TLS handshake adds another two RTTs
   of latency. Clients and servers should support connection
   keepalive (reuse) and out-of-order processing to amortize
   connection setup costs. Fast TLS connection resumption [RFC5077]
   further reduces the setup delay and avoids the DNS server keeping
   per-client session state.

   TLS False Start [TLS-FALSESTART] can also lead to a latency
   reduction in certain situations. Implementations supporting TLS
   False Start need to be aware that it imposes additional
   constraints on how one uses TLS, over and above those stated in
   [BCP195]. It is unsafe to use False Start if your implementation
   and deployment does not adhere to these specific requirements.
   See [TLS-FALSESTART] for the details of these additional
   constraints.

State: The use of connection-oriented TCP requires keeping
   additional state at the server in both the kernel and application.
   The state requirements are of particular concern on servers with
   many clients, although memory-optimized TLS can add only modest
   state over TCP. Smaller timeout values will reduce the number of
   concurrent connections, and servers can preemptively close
   connections when resource limits are exceeded.

Processing: The use of TLS encryption algorithms results in slightly
   higher CPU usage. Servers can choose to refuse new DNS-over-TLS
   clients if processing limits are exceeded.

Number of connections: To minimize state on DNS servers and
   connection startup time, clients SHOULD minimize the creation of
   new TCP connections. Use of a local DNS request aggregator (a
   particular type of forwarder) allows a single active DNS-over-TLS
   connection from any given client computer to its server.
   Additional guidance can be found in [RFC7766].

A full performance evaluation is outside the scope of this
specification. A more detailed analysis of the performance
implications of DNS over TLS (and DNS over TCP) is discussed in
[TDNS] and [RFC7766].

6. IANA Considerations

IANA has added the following value to 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], and such
a review was requested using the early allocation process [RFC7120]
for the well-known TCP port in this document.

IANA has reserved the same port number over UDP for the proposed DNS-
over-DTLS protocol [DNSoD].

```
Service Name          domain-s
Port Number           853
Transport Protocol(s) TCP/UDP
Assignee              IESG
Contact               IETF Chair
Description           DNS query-response protocol run over TLS/DTLS
Reference             This document
```

7. Design Evolution

Earlier draft versions of this document proposed an upgrade-based
approach to establish a TLS session. The client would signal its
interest in TLS by setting a "TLS OK" bit in the Extensions
Mechanisms for DNS (EDNS(0)) flags field. A server would signal its
acceptance by responding with the TLS OK bit set.

Since we assume the client doesn't want to reveal (leak) any
information prior to securing the channel, we proposed the use of a
"dummy query" that clients could send for this purpose. The proposed
query name was STARTTLS, query type TXT, and query class CH.

The TLS OK signaling approach has both advantages and disadvantages.
One important advantage is that clients and servers could negotiate
TLS. If the server is too busy, or doesn't want to provide TLS
service to a particular client, it can respond negatively to the TLS
probe. An ancillary benefit is that servers could collect
information on adoption of DNS over TLS (via the TLS OK bit in
queries) before implementation and deployment. Another anticipated
advantage is the expectation that DNS over TLS would work over port
53. That is, no need to "waste" another port and deploy new firewall
rules on middleboxes.

However, at the same time, there was uncertainty whether or not
middleboxes would pass the TLS OK bit, given that the EDNS0 flags
field has been unchanged for many years. Another disadvantage is
that the TLS OK bit may make downgrade attacks easy and
indistinguishable from broken middleboxes. From a performance
standpoint, the upgrade-based approach had the disadvantage of
requiring 1xRTT additional latency for the dummy query.

Following this proposal, DNS over DTLS was proposed separately. DNS
over DTLS claimed it could work over port 53, but only because a non-
DTLS server interprets a DNS-over-DTLS query as a response. That is,
the non-DTLS server observes the QR flag set to 1. While this
technically works, it seems unfortunate and perhaps even undesirable.

DNS over both TLS and DTLS can benefit from a single well-known port
and avoid extra latency and misinterpreted queries as responses.

8. Security Considerations

Use of DNS over TLS is designed to address the privacy risks that
arise out of the ability to eavesdrop on DNS messages. It does not
address other security issues in DNS, and there are a number of
residual risks that may affect its success at protecting privacy:

1. There are known attacks on TLS, such as person-in-the-middle and
    protocol downgrade. These are general attacks on TLS and not
    specific to DNS over TLS; please refer to the TLS RFCs for
    discussion of these security issues. Clients and servers MUST
    adhere to the TLS implementation recommendations and security
    considerations of [BCP195]. DNS clients keeping track of servers
    known to support TLS enables clients to detect downgrade attacks.
    For servers with no connection history and no apparent support
    for TLS, depending on their privacy profile and privacy
    requirements, clients may choose to (a) try another server when
    available, (b) continue without TLS, or (c) refuse to forward the
    query.

2. Middleboxes [RFC3234] are present in some networks and have been
    known to interfere with normal DNS resolution. Use of a
    designated port for DNS over TLS should avoid such interference.
    In general, clients that attempt TLS and fail can either fall
    back on unencrypted DNS or wait and retry later, depending on
    their privacy profile and privacy requirements.

3. Any DNS protocol interactions performed in the clear can be
    modified by a person-in-the-middle attacker. For example,
    unencrypted queries and responses might take place over port 53
    between a client and server. For this reason, clients MAY

discard cached information about server capabilities advertised
in cleartext.

4. This document does not, itself, specify ideas to resist known
traffic analysis or side-channel leaks. Even with encrypted
messages, a well-positioned party may be able to glean certain
details from an analysis of message timings and sizes. Clients
and servers may consider the use of a padding method to address
privacy leakage due to message sizes [RFC7830]. Since traffic
analysis can be based on many kinds of patterns and many kinds of
classifiers, simple padding schemes alone might not be sufficient
to mitigate such an attack. Padding will, however, form a part
of more complex mitigations for traffic-analysis attacks that are
likely to be developed over time. Implementors who can offer
flexibility in terms of how padding can be used may be in a
better position to enable such mitigations to be deployed in the
future.

As noted earlier, DNSSEC and DNS over TLS are independent and fully
compatible protocols, each solving different problems. The use of
one does not diminish the need nor the usefulness of the other.

9. References

9.1. Normative 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,
              <https://www.rfc-editor.org/info/bcp195>.

   [RFC1034] Mockapetris, P., "Domain names - concepts and facilities",
              STD 13, RFC 1034, DOI 10.17487/RFC1034, November 1987,
              <http://www.rfc-editor.org/info/rfc1034>.

   [RFC1035] Mockapetris, P., "Domain names - implementation and
              specification", STD 13, RFC 1035, DOI 10.17487/RFC1035,
              November 1987, <http://www.rfc-editor.org/info/rfc1035>.

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

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

   [RFC5077] Salowey, J., Zhou, H., Eronen, P., and H. Tschofenig,
              "Transport Layer Security (TLS) Session Resumption without
              Server-Side State", RFC 5077, DOI 10.17487/RFC5077,
              January 2008, <http://www.rfc-editor.org/info/rfc5077>.

   [RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security
              (TLS) Protocol Version 1.2", RFC 5246,
              DOI 10.17487/RFC5246, August 2008,
              <http://www.rfc-editor.org/info/rfc5246>.

   [RFC6234] Eastlake 3rd, D. and T. Hansen, "US Secure Hash Algorithms
              (SHA and SHA-based HMAC and HKDF)", RFC 6234,
              DOI 10.17487/RFC6234, May 2011,
              <http://www.rfc-editor.org/info/rfc6234>.

   [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,
              <http://www.rfc-editor.org/info/rfc6335>.

   [RFC7120] Cotton, M., "Early IANA Allocation of Standards Track Code
              Points", BCP 100, RFC 7120, DOI 10.17487/RFC7120, January
              2014, <http://www.rfc-editor.org/info/rfc7120>.

   [RFC7469] Evans, C., Palmer, C., and R. Sleevi, "Public Key Pinning
              Extension for HTTP", RFC 7469, DOI 10.17487/RFC7469, April
              2015, <http://www.rfc-editor.org/info/rfc7469>.

   [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,
              <http://www.rfc-editor.org/info/rfc7766>.

9.2. Informative References

   [CONFIDENTIAL-DNS]
              Wijngaards, W. and G. Wiley, "Confidential DNS", Work in
              Progress, draft-wijngaards-dnsop-confidentialdns-03, March
              2015.

   [DNSCRYPT-WEBSITE]
              Denis, F., "DNSCrypt", December 2015,
              <https://www.dnscrypt.org/>.

   [DNSCurve] Dempsky, M., "DNSCurve: Link-Level Security for the Domain
             Name System", Work in Progress, draft-dempsky-dnscurve-01,
             February 2010.

   [DNSoD]   Reddy, T., Wing, D., and P. Patil, "DNS over DTLS
             (DNSoD)", Work in Progress, draft-ietf-dprive-dnsodtls-06,
             April 2016.

   [DNSSEC-TRIGGER]
             NLnet Labs, "Dnssec-Trigger", May 2014,
             <https://www.nlnetlabs.nl/projects/dnssec-trigger/>.

   [IPSECA]   Osterweil, E., Wiley, G., Okubo, T., Lavu, R., and A.
             Mohaisen, "Opportunistic Encryption with DANE Semantics
             and IPsec: IPSECA", Work in Progress,
             draft-osterweil-dane-ipsec-03, July 2015.

   [RFC3234] Carpenter, B. and S. Brim, "Middleboxes: Taxonomy and
             Issues", RFC 3234, DOI 10.17487/RFC3234, February 2002,
             <http://www.rfc-editor.org/info/rfc3234>.

   [RFC3646] Droms, R., Ed., "DNS Configuration options for Dynamic
             Host Configuration Protocol for IPv6 (DHCPv6)", RFC 3646,
             DOI 10.17487/RFC3646, December 2003,
             <http://www.rfc-editor.org/info/rfc3646>.

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

   [RFC7258] Farrell, S. and H. Tschofenig, "Pervasive Monitoring Is an
             Attack", BCP 188, RFC 7258, DOI 10.17487/RFC7258, May
             2014, <http://www.rfc-editor.org/info/rfc7258>.

   [RFC7413] Cheng, Y., Chu, J., Radhakrishnan, S., and A. Jain, "TCP
             Fast Open", RFC 7413, DOI 10.17487/RFC7413, December 2014,
             <http://www.rfc-editor.org/info/rfc7413>.

   [RFC7435] Dukhovni, V., "Opportunistic Security: Some Protection
             Most of the Time", RFC 7435, DOI 10.17487/RFC7435,
             December 2014, <http://www.rfc-editor.org/info/rfc7435>.

   [RFC7626] Bortzmeyer, S., "DNS Privacy Considerations", RFC 7626,
             DOI 10.17487/RFC7626, August 2015,
             <http://www.rfc-editor.org/info/rfc7626>.

   [RFC7828] Wouters, P., Abley, J., Dickinson, S., and R. Bellis, "The
              edns-tcp-keepalive EDNS0 Option", RFC 7828,
              DOI 10.17487/RFC7828, April 2016,
              <http://www.rfc-editor.org/info/rfc7828>.

   [RFC7830] Mayrhofer, A., "The EDNS(0) Padding Option", RFC 7830,
              DOI 10.17487/RFC7830, May 2016,
              <http://www.rfc-editor.org/info/rfc7830>.

   [TDNS]   Zhu, L., Hu, Z., Heidemann, J., Wessels, D., Mankin, A.,
              and N. Somaiya, "Connection-Oriented DNS to Improve
              Privacy and Security", 2015 IEEE Symposium on Security and
              Privacy (SP), DOI 10.1109/SP.2015.18,
              <http://dx.doi.org/10.1109/SP.2015.18>.

   [TLS-DTLS-PROFILES]
              Dickinson, S., Gillmor, D., and T. Reddy, "Authentication
              and (D)TLS Profile for DNS-over-TLS and DNS-over-DTLS",
              Work in Progress, draft-ietf-dprive-dtls-and-tls-
              profiles-01, March 2016.

   [TLS-FALSESTART]
              Langley, A., Modadugu, N., and B. Moeller, "Transport
              Layer Security (TLS) False Start", Work in Progress,
              draft-ietf-tls-falsestart-02, May 2016.

Appendix A. Out-of-Band Key-Pinned Privacy Profile Example

   This section presents an example of how the out-of-band key-pinned
   privacy profile could work in practice based on a minimal pin set
   (two pins).

   A DNS client system is configured with an out-of-band key-pinned
   privacy profile from a network service, using a pin set containing
   two pins. Represented in HTTP Public Key Pinning (HPKP) [RFC7469]
   style, the pins are:

   o pin-sha256="FHkyLhvI0n70E47cJlRTamTrnYVcsYdjUGbr79CfAVI="

   o pin-sha256="dFSY3wdPU8L0u/8qECuz5wtlSgnorYV2f66L6GNQg6w="

   The client also configures the IP addresses of its expected DNS
   server: perhaps 192.0.2.3 and 2001:db8::2:4.

   The client connects to one of these addresses on TCP port 853 and
   begins the TLS handshake: negotiation of TLS 1.2 with a Diffie-
   Hellman key exchange. The server sends a certificate message with a
   list of three certificates (A, B, and C) and signs the
   ServerKeyExchange message correctly with the public key found in
   certificate A.

   The client now takes the SHA-256 digest of the SPKI in cert A and
   compares it against both pins in the pin set. If either pin matches,
   the verification is successful; the client continues with the TLS
   connection and can make its first DNS query.

   If neither pin matches the SPKI of cert A, the client verifies that
   cert A is actually issued by cert B. If it is, it takes the SHA-256
   digest of the SPKI in cert B and compares it against both pins in the
   pin set. If either pin matches, the verification is successful.
   Otherwise, it verifies that B was issued by C and then compares the
   pins against the digest of C's SPKI.

   If none of the SPKIs in the cryptographically valid chain of certs
   match any pin in the pin set, the client closes the connection with
   an error and marks the IP address as failed.

Acknowledgments

   The authors would like to thank Stephane Bortzmeyer, John Dickinson,
   Brian Haberman, Christian Huitema, Shumon Huque, Simon Joseffson,
   Kim-Minh Kaplan, Simon Kelley, Warren Kumari, John Levine, Ilari
   Liusvaara, Bill Manning, George Michaelson, Eric Osterweil, Jinmei
   Tatuya, Tim Wicinski, and Glen Wiley for reviewing this
   specification. They also thank Nikita Somaiya for early work on this
   idea.

Contributors

   The below individuals contributed significantly to the document:

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

   Email: sara@sinodun.com
   URI: http://sinodun.com

   Daniel Kahn Gillmor
   ACLU
   125 Broad Street, 18th Floor
   New York, NY 10004
   United States

Authors' Addresses

   Zi Hu
   USC/Information Sciences Institute
   4676 Admiralty Way, Suite 1133
   Marina del Rey, CA 90292
   United States

   Phone: +1-213-587-1057
   Email: zihu@outlook.com

   Liang Zhu
   USC/Information Sciences Institute
   4676 Admiralty Way, Suite 1133
   Marina del Rey, CA 90292
   United States

   Phone: +1-310-448-8323
   Email: liangzhu@usc.edu

   John Heidemann
   USC/Information Sciences Institute
   4676 Admiralty Way, Suite 1001
   Marina del Rey, CA 90292
   United States

   Phone: +1-310-822-1511
   Email: johnh@isi.edu

   Allison Mankin
   Independent

   Phone: +1-301-728-7198
   Email: Allison.mankin@gmail.com

   Duane Wessels
   Verisign Labs
   12061 Bluemont Way
   Reston, VA 20190
   United States

   Phone: +1-703-948-3200
   Email: dwessels@verisign.com

   Paul Hoffman
   ICANN

   Email: paul.hoffman@icann.org