Updates to and Operational Guidance for the DANE Protocol
draft-ietf-dane-ops-16

Abstract

   This document clarifies and updates the DNS-Based Authentication of
   Named Entities (DANE) TLSA specification (RFC6698) based on
   subsequent implementation experience. It also contains guidance for
   implementers, operators and protocol developers who want to make use
   of DANE records.

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 http://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 February 10, 2016.

Table of Contents

1. Introduction

   The DNS-Based Authentication of Named Entities (DANE) specification
   ([RFC6698]) introduces the DNS "TLSA" resource record type ("TLSA" is
   not an acronym). TLSA records associate a certificate or a public
   key of an end-entity or a trusted issuing authority with the

corresponding Transport Layer Security (TLS) [RFC5246] or Datagram
Transport Layer Security (DTLS) [RFC6347] transport endpoint. DANE
relies on the DNS Security Extensions (DNSSEC, [RFC4033]). DANE TLSA
records validated by DNSSEC can be used to augment or replace the use
of trusted public Certification Authorities (CAs).

The TLS and DTLS protocols provide secured TCP and UDP communication,
respectively, over IP. In the context of this document, channel
security is assumed to be provided by TLS or DTLS. By convention,
"TLS" will be used throughout this document and, unless otherwise
specified, the text applies equally well to DTLS over UDP. Used
without authentication, TLS provides protection only against
eavesdropping through its use of encryption. With authentication,
TLS also protects the transport against man-in-the-middle (MiTM)
attacks.

[RFC6698] defines three TLSA record fields, the first with 4 possible
values, the second with 2, and the third with 3. These yield 24
distinct combinations of TLSA record types. This document recommends
a smaller set of best-practice combinations of these fields to
simplify protocol design, implementation and deployment.

This document explains and recommends DANE-specific strategies to
simplify "virtual hosting", where a single Service Provider transport
endpoint simultaneously supports multiple hosted Customer Domains.

Other related documents that build on [RFC6698] are
[I-D.ietf-dane-srv] and [I-D.ietf-dane-smtp-with-dane].

Section 12 summarizes the normative updates this document makes to
[RFC6698].

1.1. 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
[RFC2119].

The following terms are used throughout this document:

Web PKI: The Public Key Infrastructure (PKI) model employed by
   browsers to authenticate web servers. This employs a set of
   trusted public Certification Authorities (CAs) to vouch for the
   authenticity of public keys associated with a particular party
   (the subject).

   Service Provider: A company or organization that offers to host a
      service on behalf of the owner of a Customer Domain. The original
      domain name associated with the service often remains under the
      control of the customer. Connecting applications may be directed
      to the Service Provider via a redirection resource record.
      Example redirection records include MX, SRV, and CNAME. The
      Service Provider frequently provides services for many customers
      and needs to ensure that the TLS credentials presented to
      connecting applications authenticate it as a valid server for the
      requested domain.

   Customer Domain: As described above, a TLS client may be interacting
      with a service that is hosted by a third party. This document
      refers to the domain name used to locate the service (prior to any
      redirection) as the "Customer Domain".

   TLSA Publisher: The entity responsible for publishing a TLSA record
      within a DNS zone. This zone will be assumed DNSSEC-signed and
      validatable to a trust anchor, unless otherwise specified. If the
      Customer Domain is not outsourcing their DNS service, the TLSA
      Publisher will be the customer themselves. Otherwise, the TLSA
      Publisher may be the operator of the outsourced DNS service.

   public key: The term "public key" is short-hand for the
      subjectPublicKeyInfo component of a PKIX [RFC5280] certificate.

   SNI: The "Server Name Indication" (SNI) TLS protocol extension
      allows a TLS client to request a connection to a particular
      service name of a TLS server ([RFC6066], section 3). Without this
      TLS extension, a TLS server has no choice but to offer a
      certificate with a default list of server names, making it
      difficult to host multiple Customer Domains at the same IP-
      address-based TLS service endpoint (i.e., provide "secure virtual
      hosting").

   TLSA parameters: In [RFC6698] the TLSA record is defined to consist
      of four fields. The first three of these are numeric parameters
      that specify the meaning of the data in the fourth and final
      field. This document refers to the first three fields as "TLSA
      parameters", or sometimes just "parameters" when obvious from
      context.

   TLSA base domain: Per Section 3 of [RFC6698] TLSA records are stored
      at a DNS domain name which is a combination of a port and protocol
      prefix and a "base domain". In [RFC6698] the "base domain" is the
      fully qualified domain name of the TLS server. This document
      modifies the TLSA record lookup strategy to prefer the fully CNAME
      expanded name of the TLS server, provided that expansion is

"secure" (DNSSEC validated) at each stage of the expansion, and
TLSA records are published for this fully expanded name. Thus the
"TLSA base domain" is either the fully CNAME expanded TLS server
name, or otherwise the initial fully qualified TLS server name,
whichever is used in combination with a port and protocol prefix
to obtain the TLSA RRset.

2. DANE TLSA Record Overview

DANE TLSA [RFC6698] specifies a protocol for publishing TLS server
certificate associations via DNSSEC [RFC4033] [RFC4034] [RFC4035].
DANE TLSA records consist of four fields:

The Certificate Usage field: Section 2.1.1 of [RFC6698] specifies 4
   values: PKIX-TA(0), PKIX-EE(1), DANE-TA(2), and DANE-EE(3). There
   is an additional private-use value: PrivCert(255), which, given
   its private scope, shall not be considered further in this
   document. All other values are reserved for use by future
   specifications.

The Selector field: Section 2.1.2 of [RFC6698] specifies 2 values:
   Cert(0), SPKI(1). There is an additional private-use value:
   PrivSel(255). All other values are reserved for use by future
   specifications.

The Matching Type field: Section 2.1.3 of [RFC6698] specifies 3
   values: Full(0), SHA2-256(1), SHA2-512(2). There is an additional
   private-use value: PrivMatch(255). All other values are reserved
   for use by future specifications.

The Certificate Association Data field: Section 2.1.4 of [RFC6698].
   This stores the full value or digest of the certificate or subject
   public key as determined by the matching type and selector
   respectively.

The record type is determined by the values of the first three
fields, which this document refers to as the "TLSA parameters", to
distinguish them from the fourth and last field. The numeric values
of these parameters were given symbolic names in [RFC7218].

In the matching type field, of the two digest algorithms, for now
only SHA2-256(1) is mandatory to implement. Clients SHOULD implement
SHA2-512(2), but servers SHOULD NOT exclusively publish SHA2-512(2)
digests. The digest algorithm agility protocol defined in Section 9
SHOULD be used by clients to decide how to process TLSA RRsets that
employ multiple digest algorithms. Server operators MUST publish
TLSA RRsets that are compatible (see Section 8) with digest algorithm
agility (Section 9).

2.1. Example TLSA record

   In the example TLSA record below:

   _25._tcp.mail.example.com. IN TLSA 2 0 1 (
                              E8B54E0B4BAA815B06D3462D65FBC7C0
                              CF556ECCF9F5303EBFBB77D022F834C0)

   The TLSA Certificate Usage is DANE-TA(2), the selector is Cert(0) and
   the matching type is SHA2-256(1). The last field is the Certificate
   Association Data Field, which in this case contains the SHA2-256
   digest of the server certificate.

3. DANE TLS Requirements

   [RFC6698] does not discuss what versions of TLS are required when
   using DANE records. This document specifies that TLS clients that
   support DANE/TLSA MUST support at least TLS 1.0 and SHOULD support
   TLS 1.2 or later.

   TLS clients using DANE MUST support the "Server Name Indication"
   (SNI) extension of TLS ([RFC6066]). Servers MAY support SNI and
   respond with a matching certificate chain, but MAY also ignore SNI
   and respond with a default certificate chain. When a server supports
   SNI but is not configured with a certificate chain that exactly
   matches the client's SNI extension, the server SHOULD respond with
   another certificate chain (a default or closest match). This is
   because clients might support more than one server name, but can only
   put a single name in the SNI extension.

4. DANE Certificate Usage Selection Guidelines

   As mentioned in Section 2, the TLSA certificate usage field takes one
   of four possible values. With PKIX-TA(0) and PKIX-EE(1), the
   validation of peer certificate chains requires additional pre-
   configured CA trust anchors that are mutually trusted by the
   operators of the TLS server and client. With DANE-TA(2) and DANE-
   EE(3), no pre-configured CA trust anchors are required and the
   published DANE TLSA records are sufficient to verify the peer's
   certificate chain.

   Standards for application protocols that employ DANE TLSA can specify
   more specific guidance than [RFC6698] or this document. Such
   application-specific standards need to carefully consider which set
   of DANE certificate usages to support. Simultaneous support for all
   four usages is NOT RECOMMENDED for DANE clients. When all four
   usages are supported, an attacker capable of compromising the
   integrity of DNSSEC needs only to replace server's TLSA RRset with

one that lists suitable DANE-EE(3) or DANE-TA(2) records, effectively
bypassing any added verification via public CAs. In other words,
when all four usages are supported, PKIX-TA(2) and PKIX-EE(1) offer
only illusory incremental security over DANE-TA(2) and DANE-EE(3).

Designs in which clients support just the DANE-TA(2) and DANE-EE(3)
certificate usages are RECOMMENDED. With DANE-TA(2) and DANE-EE(3)
clients don't need to track a large changing list of X.509 trust-
anchors in order to successfully authenticate servers whose
certificates are issued by a brand new or not widely trusted CA.

The DNSSEC TLSA records for servers MAY include both sets of usages
if the server needs to support a mixture of clients, some supporting
one pair of usages and some the other.

4.1. Opportunistic Security and PKIX usages

When the client's protocol design is based on Opportunistic Security
(OS, [RFC7435]), and the use of authentication is based on the
presence of server TLSA records, it is especially important to avoid
the PKIX-EE(1) and PKIX-TA(0) certificate usages.

When authenticated TLS is used opportunistically, based on the
presence of DANE TLSA records, and no secure TLSA records are
present, unauthenticated TLS is used if possible, and otherwise
perhaps even cleartext. However, if usable secure TLSA records are
published then authentication MUST succeed. Also, outside the
browser space, there is no pre-ordained canon of trusted CAs, and in
any case there is no security advantage in using PKIX-TA(0) or PKIX-
EE(1) when the DANE-TA(2) and DANE-EE(3) usages are also supported
(as an attacker who can compromise DNS can replace the former with
the latter).

Authentication via the PKIX-TA(0) and PKIX-EE(1) certificate usages
is more brittle, the client and server need to happen to agree on a
mutually trusted CA, but with opportunistic security the client is
just trying to protect the communication channel at the request of
the server, and would otherwise be willing to use cleartext or
unauthenticated TLS. Use of fragile mechanisms (like public CA
authentication for some unspecified set of trusted CAs) is not
sufficiently reliable for an opportunistic security client to honor
the server's request for authentication. Opportunistic security
needs to be unintrusive and to require few, if any, work-arounds for
valid and yet mismatched peers.

With the PKIX-TA(0) and PKIX-EE(1) usages offering no more security,
but being more prone to failure, they are a poor fit for
opportunistic security and SHOULD NOT be used in that context.

4.2. Interaction with Certificate Transparency

   Certificate Transparency (CT) [RFC6962] defines an experimental
   approach that could be used to mitigate the risk of rogue or
   compromised public CAs issuing unauthorized certificates. This
   section clarifies the interaction of the experimental CT and DANE.
   This section may need to be revised in light of any future standards
   track version of CT.

   When a server is authenticated via a DANE TLSA RR with TLSA
   Certificate Usage DANE-EE(3), the domain owner has directly specified
   the certificate associated with the given service without reference
   to any public certification authority. Therefore, when a TLS client
   authenticates the TLS server via a TLSA record with usage DANE-EE(3),
   CT checks SHOULD NOT be performed. Publication of the server
   certificate or public key (digest) in a TLSA record in a DNSSEC
   signed zone by the domain owner assures the TLS client that the
   certificate is not an unauthorized certificate issued by a rogue CA
   without the domain owner's consent.

   When a server is authenticated via a DANE TLSA record with TLSA usage
   DANE-TA(2) and the server certificate does not chain to a known
   public root CA, CT cannot apply (CT logs only accept chains that
   start with a known public root). Since TLSA Certificate Usage DANE-
   TA(2) is generally intended to support non-public trust anchors, TLS
   clients SHOULD NOT perform CT checks with usage DANE-TA(2).

   With certificate usages PKIX-TA(0) and PKIX-EE(1), CT applies just at
   it would without DANE. TLSA records of this type only constrain
   which CAs are acceptable in PKIX validation. All checks used in the
   absence of DANE still apply when validating certificate chains with
   DANE PKIX-TA(0) and PKIX-EE(1) constraints.

4.3. Switching from/to PKIX-TA/EE to/from DANE-TA/EE

   The choice of preferred certificate usages may need to change as an
   application protocol evolves. When transitioning between PKIX-TA/
   PKIX-EE and DANE-TA/DANE-EE, clients begin to enable support for the
   new certificate usage values. If the new preferred certificate
   usages are PKIX-TA/EE this requires installing and managing the
   appropriate set of CA trust anchors. During this time servers will
   publish both types of TLSA records. At some later time when the vast
   majority of servers have published the new preferred TLSA records,
   clients can stop supporting the legacy certificate usages.
   Similarly, servers can stop publishing legacy TLSA records once the
   vast majority of clients support the new certificate usages.

5. **Certificate-Usage-Specific DANE Updates and Guidelines**

   The four Certificate Usage values from the TLSA record, DANE-EE(3),
   DANE-TA(2), PKIX-EE(1) and PKIX-TA(0), are discussed below.

5.1. **Certificate Usage DANE-EE(3)**

   In this section the meaning of DANE-EE(3) is updated from [RFC6698]
   to specify that peer identity matching and that validity period
   enforcement is based solely on the TLSA RRset properties. This
   document also extends [RFC6698] to cover the use of DANE
   authentication of raw public keys [RFC7250] via TLSA records with
   Certificate Usage DANE-EE(3) and selector SPKI(1).

   Authentication via certificate usage DANE-EE(3) TLSA records involves
   simply checking that the server's leaf certificate matches the TLSA
   record. In particular, the binding of the server public key to its
   name is based entirely on the TLSA record association. The server
   MUST be considered authenticated even if none of the names in the
   certificate match the client's reference identity for the server.
   This simplifies the operation of servers that host multiple Customer
   Domains, as a single certificate can be associated with multiple
   domains, without having to match each of the corresponding reference
   identifiers.

       ; Multiple client domains hosted by example.net service provider:
       ;
       www.example.com.   IN CNAME ex-com.example.net.
       www.example.org.   IN CNAME ex-org.example.net.
       ;
       ; In the provider's DNS zone, a single certificate and TLSA
       ; record supports multiple client domains, greatly simplifying
       ; "virtual hosting".
       ;
       ex-com.example.net.   IN A 192.0.2.1
       ex-org.example.net.   IN A 192.0.2.1
       _443._tcp.ex-com.example.net. IN CNAME tlsa._dane.example.net.
       _443._tcp.ex-org.example.net. IN CNAME tlsa._dane.example.net.
       tlsa._dane.example.net.   IN TLSA 3 1 1 e3b0c44298fc1c14...

   Also with DANE-EE(3), the expiration date of the server certificate
   MUST be ignored. The validity period of the TLSA record key binding
   is determined by the validity period of the TLSA record DNSSEC
   signatures. Validity is reaffirmed on an ongoing basis by continuing
   to publish the TLSA record and sign the containing zone, rather than
   via dates set in stone in certificate. The expiration becomes a
   reminder to the administrator that it is likely time to rotate the
   key, but missing the date no longer causes an outage. When keys are

rotated (for whatever reason) it is important to follow the
procedures outlined in Section 8.

If a server uses just DANE-EE(3) TLSA records, and all its clients
are DANE clients, the server need not employ SNI (i.e., it may ignore
the client's SNI message) even when the server is known via multiple
domain names that would otherwise require separate certificates. It
is instead sufficient for the TLSA RRsets for all the domain names in
question to match the server's default certificate. For application
protocols where the server name is obtained indirectly via SRV, MX or
similar records, it is simplest to publish a single hostname as the
target server name for all the hosted domains.

In organizations where it is practical to make coordinated changes in
DNS TLSA records before server key rotation, it is generally best to
publish end-entity DANE-EE(3) certificate associations in preference
to other choices of certificate usage. DANE-EE(3) TLSA records
support multiple server names without SNI, don't suddenly stop
working when leaf or intermediate certificates expire, and don't fail
when a server operator neglects to include all the required issuer
certificates in the server certificate chain.

More specifically, it is RECOMMENDED that at most sites TLSA records
published for DANE servers be "DANE-EE(3) SPKI(1) SHA2-256(1)"
records. Selector SPKI(1) is chosen because it is compatible with
raw public keys ([RFC7250]) and the resulting TLSA record need not
change across certificate renewals with the same key. Matching type
SHA2-256(1) is chosen because all DANE implementations are required
to support SHA2-256. This TLSA record type easily supports hosting
arrangements with a single certificate matching all hosted domains.
It is also the easiest to implement correctly in the client.

Clients that support raw public keys can use DANE TLSA records with
certificate usage DANE-EE(3) and selector SPKI(1) to authenticate
servers that negotiate the use of raw public keys. Provided the
server adheres to the requirements of Section 8, the fact that raw
public keys are not compatible with any other TLSA record types will
not get in the way of successful authentication. Clients that employ
DANE to authenticate the peer server SHOULD NOT negotiate the use of
raw public keys unless the server's TLSA RRset includes DANE-EE(3)
SPKI(1) TLSA records.

While it is, in principle, also possible to authenticate raw public
keys via "DANE-EE(3) Cert(0) Full(0)" records by extracting the
public key from the certificate in DNS, extracting just the the
public key from a "3 0 0" TLSA record requires extra logic on clients
that not all implementations are expected to provide. Servers that
wish to support [RFC7250] raw public keys need to publish TLSA

records with a certificate usage of DANE-EE(3) and a selector of
SPKI(1).

While DANE-EE(3) TLSA records are expected to be by far the most
prevalent, as explained in Section 5.2 DANE-TA(2) records are a valid
alternative for sites with many DANE services. Note however, that
virtual hosting is more complex with DANE-TA(2). Also with DANE-
TA(2) server operators MUST ensure that the server is configured with
a sufficiently complete certificate chain and need to remember to
replace certificates prior to their expiration dates.

5.2. Certificate Usage DANE-TA(2)

This section updates [RFC6698] by specifying a new operational
requirement for servers publishing TLSA records with a usage of DANE-
TA(2): such servers MUST include the trust-anchor certificate in
their TLS server certificate message unless all such TLSA records are
"2 0 0" records that publish the server certificate in full.

Some domains may prefer to avoid the operational complexity of
publishing unique TLSA RRs for each TLS service. If the domain
employs a common issuing Certification Authority to create
certificates for multiple TLS services, it may be simpler to publish
the issuing authority as a trust anchor (TA) for the certificate
chains of all relevant services. The TLSA query domain (TLSA base
domain with port and protocol prefix labels) for each service issued
by the same TA may then be set to a CNAME alias that points to a
common TLSA RRset that matches the TA. For example:

```
; Two servers, each with its own certificate, that share
; a common issuer (trust-anchor).
;
www1.example.com.          IN A 192.0.2.1
www2.example.com.          IN A 192.0.2.2
_443._tcp.www1.example.com.  IN CNAME tlsa._dane.example.com.
_443._tcp.www2.example.com.  IN CNAME tlsa._dane.example.com.
tlsa._dane.example.com.      IN TLSA 2 0 1 e3b0c44298fc1c14...
```

The above configuration simplifies server key rotation, because while
the servers continue to receive new certificates from a CA matched by
the shared (target of the CNAMEs) TLSA record, servers certificates
can be updated without making any DNS changes. As the list of active
issuing CAs changes, the shared TLSA record will be updated (much
less frequently) by the administrators who manage the CAs. Those
administrators still need to perform TLSA record updates with care as
described in Section 8.

With usage DANE-TA(2) the server certificates will need to have names
that match one of the client's reference identifiers (see [RFC6125]).
When hosting multiple unrelated client domains (that can't all appear
in a single certificate), such a server SHOULD employ SNI to select
the appropriate certificate to present to the client.

5.2.1. Recommended record combinations

TLSA records with matching type Full(0) are NOT RECOMMENDED. While
these potentially obviate the need to transmit the TA certificate in
the TLS server certificate message, client implementations may not be
able to augment the server certificate chain with the data obtained
from DNS, especially when the TLSA record supplies a bare key
(selector SPKI(1)). Since the server will need to transmit the TA
certificate in any case, server operators SHOULD publish TLSA records
with a matching type other than Full(0) and avoid potential DNS
interoperability issues with large TLSA records containing full
certificates or keys (see Section 10.1.1).

TLSA Publishers employing DANE-TA(2) records SHOULD publish records
with a selector of Cert(0). Such TLSA records are associated with
the whole trust anchor certificate, not just with the trust anchor
public key. In particular, when authenticating the peer certificate
chain via such a TLSA record, the client SHOULD apply any relevant
constraints from the trust anchor certificate, such as, for example,
path length constraints.

While a selector of SPKI(1) may also be employed, the resulting TLSA
record will not specify the full trust anchor certificate content,
and elements of the trust anchor certificate other than the public
key become mutable. This may, for example, enable a subsidiary CA to
issue a chain that violates the trust anchor's path length or name
constraints.

5.2.2. Trust anchor digests and server certificate chain

With DANE-TA(2), a complication arises when the TA certificate is
omitted from the server's certificate chain, perhaps on the basis of
Section 7.4.2 of [RFC5246]:

> The sender's certificate MUST come first in the list.
> Each following certificate MUST directly certify the one
> preceding it. Because certificate validation requires
> that root keys be distributed independently, the
> self-signed certificate that specifies the root
> certification authority MAY be omitted from the chain,
> under the assumption that the remote end must already
> possess it in order to validate it in any case.

   With TLSA Certificate Usage DANE-TA(2), there is no expectation that
   the client is pre-configured with the trust anchor certificate. In
   fact, client implementations are free to ignore all locally
   configured trust anchors when processing usage DANE-TA(2) TLSA
   records and may rely exclusively on the certificates provided in the
   server's certificate chain. But, with a digest in the TLSA record,
   the TLSA record contains neither the full trust anchor certificate
   nor the full public key. If the TLS server's certificate chain does
   not contain the trust anchor certificate, DANE clients will be unable
   to authenticate the server.

   TLSA Publishers that publish TLSA Certificate Usage DANE-TA(2)
   associations with a selector of SPKI(1) or using a digest-based
   matching type (not Full(0)) MUST ensure that the corresponding server
   is configured to also include the trust anchor certificate in its TLS
   handshake certificate chain, even if that certificate is a self-
   signed root CA and would have been optional in the context of the
   existing public CA PKI.

   Only when the server TLSA record includes a "DANE-TA(2) Cert(0)
   Full(0)" TLSA record containing a full trust-anchor certificate, is
   the trust-anchor certificate optional in the server's TLS certificate
   message. Only in this case, the client MUST also be able to verify
   the server's certificate chain via a trust-anchor provided via DNS
   rather than via the TLS handshake server certificate message.

5.2.3. Trust anchor public keys

   TLSA records with TLSA Certificate Usage DANE-TA(2), selector SPKI(1)
   and a matching type of Full(0) publish the full public key of a trust
   anchor via DNS. In section 6.1.1 of [RFC5280] the definition of a
   trust anchor consists of the following four parts:

   1. the trusted issuer name,

   2. the trusted public key algorithm,

   3. the trusted public key, and

   4. optionally, the trusted public key parameters associated with the
       public key.

   Items 2-4 are precisely the contents of the subjectPublicKeyInfo
   published in the TLSA record. The issuer name is not included in the
   subjectPublicKeyInfo.

   With TLSA Certificate Usage DANE-TA(2), the client may not have the
   associated trust anchor certificate, and cannot generally verify

   whether a particular certificate chain is "issued by" the trust
   anchor described in the TLSA record.

   When the server certificate chain includes a CA certificate whose
   public key matches the TLSA record, the client can match that CA as
   the intended issuer. Otherwise, the client can only check that the
   topmost certificate in the server's chain is "signed by" the trust
   anchor's public key in the TLSA record. Such a check may be
   difficult to implement, and cannot be expected to be supported by all
   clients.

   Thus, servers cannot rely on "DANE-TA(2) SPKI(1) Full(0)" TLSA
   records to be sufficient to authenticate chains issued by the
   associated public key in the absence of a corresponding certificate
   in the server's TLS certificate message. Servers employing "2 1 0"
   TLSA records, MUST include the corresponding trust-anchor certificate
   in their certificate chain.

   If none of the server's certificate chain elements match a public key
   specified in a TLSA record, and at least one "DANE-TA(2) SPKI(1)
   Full(0)" TLSA record is available, it is RECOMMENDED that clients
   check whether the topmost certificate in the chain is signed by the
   provided public key and has not expired, and in that case consider
   the server authenticated, provided the rest of the chain passes
   validation including leaf certificate name checks.

5.3. Certificate Usage PKIX-EE(1)

   This Certificate Usage is similar to DANE-EE(3), but in addition PKIX
   verification is required. Therefore, name checks, certificate
   expiration, certificate transparency, etc., apply as they would
   without DANE.

5.4. Certificate Usage PKIX-TA(0)

   This section updates [RFC6698] by specifying new client
   implementation requirements. Clients that trust intermediate
   certificates MUST be prepared to construct longer PKIX chains than
   would be required for PKIX alone.

   TLSA Certificate Usage PKIX-TA(0) allows a domain to publish
   constraints on the set of PKIX certification authorities trusted to
   issue certificates for its TLS servers. A PKIX-TA(0) TLSA record
   matches PKIX-verified trust chains which contain an issuer
   certificate (root or intermediate) that matches its certificate
   association data field (typically a certificate or digest).

PKIX-TA(0) requires more complex coordination (than with DANE-TA(2)
or DANE-EE(3)) between the Customer Domain and the Service Provider
in hosting arrangements. Thus, this certificate usage is NOT
RECOMMENDED when the Service Provider is not also the TLSA Publisher
(at the TLSA base domain obtained via CNAMEs, SRV or MX records).

TLSA Publishers who publish TLSA records for a particular public root
CA, will expect that clients will only accept chains anchored at that
root. It is possible, however, that the client's trusted certificate
store includes some intermediate CAs, either with or without the
corresponding root CA. When a client constructs a trust chain
leading from a trusted intermediate CA to the server leaf
certificate, such a "truncated" chain might not contain the trusted
root published in the server's TLSA record.

If the omitted root is also trusted, the client may erroneously
reject the server chain if it fails to determine that the shorter
chain it constructed extends to a longer trusted chain that matches
the TLSA record. Thus, when matching a usage PKIX-TA(0) TLSA record,
while no matching certificate is found, a client MUST continue
extending the chain even after any locally trusted certificate is
found. If no TLSA records have matched any of the elements of the
chain, and the trusted certificate found is not self-issued, the
client MUST attempt to build a longer chain in case a certificate
closer to the root matches the server's TLSA record.

[6]. Service Provider and TLSA Publisher Synchronization

Whenever possible, the TLSA Publisher and the Service Provider should
be the same entity. Otherwise, they need to coordinate changes to
ensure that TLSA records published by the TLSA Publisher don't fall
out of sync with the server certificate used by the Service Provider.
Such coordination is difficult and service outages will result when
coordination fails.

Publishing the TLSA record in the Service Provider's zone avoids the
complexity of bilateral coordination of server certificate
configuration and TLSA record management. Even when the TLSA RRset
has to be published in the Customer Domain's DNS zone (perhaps the
client application does not "chase" CNAMEs to the TLSA base domain),
it is possible to employ CNAME records to delegate the content of the
TLSA RRset to a domain operated by the Service Provider.

Only Certificate Usages DANE-EE(3) and DANE-TA(2) work well with TLSA
CNAMEs across organizational boundaries. With PKIX-TA(0) or PKIX-
EE(1) the Service Provider would need to obtain certificates in the
name of Customer Domain from a suitable public CA (securely
impersonate the customer), or the customer would need to provision

the relevant private keys and certificates at the Service Provider's
systems.

Certificate Usage DANE-EE(3): In this case the Service Provider can
   publish a single TLSA RRset that matches the server certificate or
   public key digest. The same RRset works for all Customer Domains
   because name checks do not apply with DANE-EE(3) TLSA records (see
   Section 5.1). A Customer Domain can create a CNAME record
   pointing to the TLSA RRset published by the Service Provider.

Certificate Usage DANE-TA(2): When the Service Provider operates a
   private certification authority, the Service Provider is free to
   issue a certificate bearing any customer's domain name. Without
   DANE, such a certificate would not pass trust verification, but
   with DANE, the customer's TLSA RRset that is aliased to the
   provider's TLSA RRset can delegate authority to the provider's CA
   for the corresponding service. The Service Provider can generate
   appropriate certificates for each customer and use the SNI
   information provided by clients to select the right certificate
   chain to present to each client.

Below are example DNS records (assumed "secure" and shown without the
associated DNSSEC information, such as record signatures) that
illustrate both of of the above models in the case of an HTTPS
service whose clients all support DANE TLS. These examples work even
with clients that don't "chase" CNAMEs when constructing the TLSA
base domain (see Section 7 below).

```
; The hosted web service is redirected via a CNAME alias.
; The associated TLSA RRset is also redirected via a CNAME alias.
;
; A single certificate at the provider works for all Customer
; Domains due to the use of the DANE-EE(3) Certificate Usage.
;
www1.example.com.              IN CNAME w1.example.net.
_443._tcp.www1.example.com.  IN CNAME _443._tcp.w1.example.net.
_443._tcp.w1.example.net.    IN TLSA 3 1 1 (
                                   8A9A70596E869BED72C69D97A8895DFA
                                   D86F300A343FECEFF19E89C27C896BC9 )


;
; A CA at the provider can also issue certificates for each Customer
; Domain, and use the DANE-TA(2) Certificate Usage type to
; indicate a trust anchor.
;
www2.example.com.              IN CNAME w2.example.net.
_443._tcp.www2.example.com.  IN CNAME _443._tcp.w2.example.net.
_443._tcp.w2.example.net.    IN TLSA 2 0 1 (
                                   C164B2C3F36D068D42A6138E446152F5
                                   68615F28C69BD96A73E354CAC88ED00C )
```

With protocols that support explicit transport redirection via DNS MX
records, SRV records, or other similar records, the TLSA base domain
is based on the redirected transport end-point, rather than the
origin domain. With SMTP, for example, when an email service is
hosted by a Service Provider, the Customer Domain's MX hostnames will
point at the Service Provider's SMTP hosts. When the Customer
Domain's DNS zone is signed, the MX hostnames can be securely used as
the base domains for TLSA records that are published and managed by
the Service Provider. For example (without the required DNSSEC
information, such as record signatures):

```
; Hosted SMTP service
;
example.com.              IN MX 0 mx1.example.net.
example.com.              IN MX 0 mx2.example.net.
_25._tcp.mx1.example.net.  IN TLSA 3 1 1 (
                                   8A9A70596E869BED72C69D97A8895DFA
                                   D86F300A343FECEFF19E89C27C896BC9 )
_25._tcp.mx2.example.net.  IN TLSA 3 1 1 (
                                   C164B2C3F36D068D42A6138E446152F5
                                   68615F28C69BD96A73E354CAC88ED00C )
```

If redirection to the Service Provider's domain (via MX or SRV
records or any similar mechanism) is not possible, and aliasing of
the TLSA record is not an option, then more complex coordination

between the Customer Domain and Service Provider will be required.
Either the Customer Domain periodically provides private keys and a
corresponding certificate chain to the Provider (after making
appropriate changes in its TLSA records), or the Service Provider
periodically generates the keys and certificates and needs to wait
for matching TLSA records to be published by its Customer Domains
before deploying newly generated keys and certificate chains.
Section 7 below describes an approach that employs CNAME "chasing" to
avoid the difficulties of coordinating key management across
organization boundaries.

For further information about combining DANE and SRV, please see
[I-D.ietf-dane-srv].

7. TLSA Base Domain and CNAMEs

When the application protocol does not support service location
indirection via MX, SRV or similar DNS records, the service may be
redirected via a CNAME. A CNAME is a more blunt instrument for this
purpose, since unlike an MX or SRV record, it remaps the entire
origin domain to the target domain for all protocols.

The complexity of coordinating key management is largely eliminated
when DANE TLSA records are found in the Service Provider's domain, as
discussed in Section 6. Therefore, DANE TLS clients connecting to a
server whose domain name is a CNAME alias SHOULD follow the CNAME
hop-by-hop to its ultimate target host (noting at each step whether
the CNAME is DNSSEC-validated). If at each stage of CNAME expansion
the DNSSEC validation status is "secure", the final target name
SHOULD be the preferred base domain for TLSA lookups.

Implementations failing to find a TLSA record using a base name of
the final target of a CNAME expansion SHOULD issue a TLSA query using
the original destination name. That is, the preferred TLSA base
domain SHOULD be derived from the fully expanded name, and failing
that SHOULD be the initial domain name.

When the TLSA base domain is the result of "secure" CNAME expansion,
the resulting domain name MUST be used as the HostName in the
client's SNI extension, and MUST be the primary reference identifier
for peer certificate matching with certificate usages other than
DANE-EE(3).

Protocol-specific TLSA specifications may provide additional guidance
or restrictions when following CNAME expansions.

Though CNAMEs are illegal on the right hand side of most indirection
records, such as MX and SRV records, they are supported by some

implementations. For example, if the MX or SRV host is a CNAME
alias, some implementations may "chase" the CNAME. If they do, they
SHOULD use the target hostname as the preferred TLSA base domain as
described above (and if the TLSA records are found there, use the
CNAME expanded domain also in SNI and certificate name checks).

8. TLSA Publisher Requirements

This section updates [RFC6698] by specifying that the TLSA Publisher
MUST ensure that each combination of Certificate Usage, selector and
matching type in the server's TLSA RRset includes at least one record
that matches the server's current certificate chain. TLSA records
that match recently retired or yet to be deployed certificate chains
will be present during key rollover. Such past or future records
MUST NOT at any time be the only records published for any given
combination of usage, selector and matching type. The TLSA record
update process described below ensures that this requirement is met.

While a server is to be considered authenticated when its certificate
chain is matched by any of the published TLSA records, not all
clients support all combinations of TLSA record parameters. Some
clients may not support some digest algorithms, others may either not
support, or may exclusively support, the PKIX Certificate Usages.
Some clients may prefer to negotiate [RFC7250] raw public keys, which
are only compatible with TLSA records whose Certificate Usage is
DANE-EE(3) with selector SPKI(1). The only other TLSA record type
that is potentially compatible with raw public keys is DANE-EE(3)
Cert(0) Full(0), but support for raw public keys with that TLSA
record type is not expected to be broadly implemented.

A consequence of the above uncertainty as to which TLSA parameters
are supported by any given client is that servers need to ensure that
each and every parameter combination that appears in the TLSA RRset
is, on its own, sufficient to match the server's current certificate
chain. In particular, when deploying new keys or new parameter
combinations some care is required to not generate parameter
combinations that only match past or future certificate chains (or
raw public keys). The rest of this section explains how to update
the TLSA RRset in a manner that ensures the above requirement is met.

8.1. Key rollover with fixed TLSA Parameters

The simplest case is key rollover while retaining the same set of
published parameter combinations. In this case, TLSA records
matching the existing server certificate chain (or raw public keys)
are first augmented with corresponding records matching the future
keys, at least two TTLs or longer before the the new chain is
deployed. This allows the obsolete RRset to age out of client caches

before the new chain is used in TLS handshakes. Once sufficient time
has elapsed and all clients performing DNS lookups are retrieving the
updated TLSA records, the server administrator may deploy the new
certificate chain, verify that it works, and then remove any obsolete
records matching the no longer active chain:

```
; Initial TLSA RRset
;
_443._tcp.www.example.org. IN TLSA 3 1 1 01d09d19c2139a46...

; Transitional TLSA RRset published at least 2 TTLs before
; the actual key change
;
_443._tcp.www.example.org. IN TLSA 3 1 1 01d09d19c2139a46...
_443._tcp.www.example.org. IN TLSA 3 1 1 7aa7a5359173d05b...

; Final TLSA RRset after the key change
;
_443._tcp.www.example.org. IN TLSA 3 1 1 7aa7a5359173d05b...
```

The next case to consider is adding or switching to a new combination
of TLSA parameters. In this case publish the new parameter
combinations for the server's existing certificate chain first, and
only then deploy new keys if desired:

```
; Initial TLSA RRset
;
_443._tcp.www.example.org. IN TLSA 1 1 1 01d09d19c2139a46...

; New TLSA RRset, same key re-published as DANE-EE(3)
;
_443._tcp.www.example.org. IN TLSA 3 1 1 01d09d19c2139a46...
```

8.2. Switching to DANE-TA from DANE-EE

This section explains how to migrate to a new certificate chain and
TLSA record with usage DANE-TA(2) from a self-signed server
certificate and a DANE-EE(3) SPKI(1) SHA2-256(1) TLSA record. This
example assumes that a new private key is generated in conjunction
with transitioning to a new certificate issued by the desired trust-
anchor.

The original "3 1 1" TLSA record supports [RFC7250] raw public keys,
and clients may choose to negotiate their use. Use of raw public
keys rules out the possibility of certificate chain verification.
Therefore, the transitional TLSA record for the planned DANE-TA(2)
certificate chain is a "3 1 1" record that works even when raw public
keys are used. The TLSA RRset is updated to use DANE-TA(2) only

   after the new chain is deployed and the "3 1 1" record matching the
   original key is dropped.

   This process follows the requirement that each combination of
   parameters present in the RRset is always sufficient to validate the
   server. It avoids publishing a transitional TLSA RRset in which "3 1
   1" matches only the current key and "2 0 1" matches only the future
   certificate chain, because these might not work reliably during the
   initial deployment of the new keys.

```
; Initial TLSA RRset
;
_443._tcp.www.example.org. IN TLSA 3 1 1 01d09d19c2139a46...

; Transitional TLSA RRset, published at least 2 TTLs before the
; actual key change.  The new keys are issued by a DANE-TA(2) CA,
; but are initially specified via a DANE-EE(3) association.
;
_443._tcp.www.example.org. IN TLSA 3 1 1 01d09d19c2139a46...
_443._tcp.www.example.org. IN TLSA 3 1 1 7aa7a5359173d05b...

; The final TLSA RRset after the key change.  Now that the old
; self-signed EE key is out of the picture, publish the issuing
; TA of the new chain.
;
_443._tcp.www.example.org. IN TLSA 2 0 1 c57bce38455d9e3d...
```

8.3. Switching to New TLSA Parameters

   When employing a new digest algorithm in the TLSA RRset, for
   compatibility with digest agility specified in Section 9 below,
   administrators SHOULD publish the new digest algorithm with each
   combination of Certificate Usage and selector for each associated key
   or chain used with any other digest algorithm. When removing an
   algorithm, remove it entirely. Each digest algorithm employed SHOULD
   match the same set of chains (or raw public keys).

```
; Initial TLSA RRset with DANE-EE SHA2-256 associations for two keys.
;
_443._tcp.www.example.org. IN TLSA 3 1 1 01d09d19c2139a46...
_443._tcp.www.example.org. IN TLSA 3 1 1 7aa7a5359173d05b...

; New TLSA RRset also with SHA2-512 associations for each key
;
_443._tcp.www.example.org. IN TLSA 3 1 1 01d09d19c2139a46...
_443._tcp.www.example.org. IN TLSA 3 1 2 d9947c35089310bc...
_443._tcp.www.example.org. IN TLSA 3 1 1 7aa7a5359173d05b...
_443._tcp.www.example.org. IN TLSA 3 1 2 89a7486a4b6ae714...
```

8.4. **TLSA Publisher Requirements Summary**

   In summary, server operators updating TLSA records should make one
   change at a time. The individual safe changes are:

   o Pre-publish new certificate associations that employ the same TLSA
      parameters (usage, selector and matching type) as existing TLSA
      records, but match certificate chains that will be deployed in the
      near future.

   o Wait for stale TLSA RRsets to expire from DNS caches before
      configuring servers to use the new certificate chain.

   o Remove TLSA records matching no longer deployed certificate
      chains.

   o Publish TLSA RRsets in which all parameter combinations
      (certificate usage, selector and matching type) present in the
      RRset match the same set of current and planned certificate
      chains.

   The above steps are intended to ensure that at all times and for each
   combination of usage, selector and matching type at least one TLSA
   record corresponds to the server's current certificate chain. Each
   combination of Certificate Usage, selector and matching type in a
   server's TLSA RRset SHOULD NOT at any time (including unexpired
   RRsets in client caches) match only some combination of future or
   past certificate chains. As a result, no matter what combinations of
   usage, selector and matching type may be supported by a given client,
   they will be sufficient to authenticate the server.

9. **Digest Algorithm Agility**

   While [RFC6698] specifies multiple digest algorithms, it does not
   specify a protocol by which the client and TLSA record publisher can
   agree on the strongest shared algorithm. Such a protocol would allow
   the client and server to avoid exposure to deprecated weaker
   algorithms that are published for compatibility with less capable
   clients, but which SHOULD be avoided when possible. Such a protocol
   is specified below.

   This section defines a protocol for avoiding deprecated digest
   algorithms when these are published in a peer's TLSA RRset alongside
   stronger digest algorithms. Note that this protocol never avoids RRs
   with DANE matching type Full(0), as these do not employ a digest
   algorithm that might some day be weakened by cryptanalysis.

Client implementations SHOULD implement a default order of digest
algorithms by strength. This order SHOULD be configurable by the
administrator or user of the client software. If possible, a
configurable mapping from numeric DANE TLSA matching types to
underlying digest algorithms provided by the cryptographic library
SHOULD be implemented to allow new matching types to be used with
software that predates their introduction. Configurable ordering of
digest algorithms SHOULD be extensible to any new digest algorithms.

To make digest algorithm agility possible, all published DANE TLSA
RRsets MUST conform to the requirements of Section 8. Clients SHOULD
use digest algorithm agility when processing the peer's DANE TLSA
records. Algorithm agility is to be applied after first discarding
any unusable or malformed records (unsupported digest algorithm, or
incorrect digest length). For each usage and selector, the client
SHOULD process only any usable records with a matching type of
Full(0) and the usable records whose digest algorithm is considered
by the client to be the strongest among usable records with the given
usage and selector.

Example: a client implements digest agility and prefers SHA2-512(2)
over SHA2-256(1), while the server publishes an RRset that employs
both digest algorithms as well as a Full(0) record.

```
_25._tcp.mail.example.com. IN TLSA 3 1 1 (
                            3FE246A848798236DD2AB78D39F0651D
                            6B6E7CA8E2984012EB0A2E1AC8A87B72 )
_25._tcp.mail.example.com. IN TLSA 3 1 2 (
                            D4F5AF015B46C5057B841C7E7BAB759C
                            BF029526D29520C5BE6A32C67475439E
                            54AB3A945D80C743347C9BD4DADC9D8D
                            57FAB78EAA835362F3CA07CCC19A3214 )
_25._tcp.mail.example.com. IN TLSA 3 1 0 (
                            3059301306072A8648CE3D020106082A
                            8648CE3D0301070342000471CB1F504F
                            9E4B33971376C005445DACD33CD79A28
                            81C3DED1981F18E7AAA76609DD0E4EF2
                            8265C82703030AD60C5DBA6FB8A9397A
                            C0FCF06D424C885D484887 )
```

In this case the client SHOULD accept a server public key that
matches either the "3 1 0" record or the "3 1 2" record, but SHOULD
NOT accept keys that match only the weaker "3 1 1" record.

10. General DANE Guidelines

These guidelines provide guidance for using or designing protocols
for DANE.

10.1. DANE DNS Record Size Guidelines

Selecting a combination of TLSA parameters to use requires careful
thought. One important consideration to take into account is the
size of the resulting TLSA record after its parameters are selected.

10.1.1. UDP and TCP Considerations

Deployments SHOULD avoid TLSA record sizes that cause UDP
fragmentation.

Although DNS over TCP would provide the ability to more easily
transfer larger DNS records between clients and servers, it is not
universally deployed and is still prohibited by some firewalls.
Clients that request DNS records via UDP typically only use TCP upon
receipt of a truncated response in the DNS response message sent over
UDP. Setting the TC bit alone will be insufficient if the response
containing the TC bit is itself fragmented.

10.1.2. Packet Size Considerations for TLSA Parameters

Server operators SHOULD NOT publish TLSA records using both a TLSA
Selector of Cert(0) and a TLSA Matching Type of Full(0), as even a
single certificate is generally too large to be reliably delivered
via DNS over UDP. Furthermore, two TLSA records containing full
certificates will need to be published simultaneously during a
certificate rollover, as discussed in Section 8.1.

While TLSA records using a TLSA Selector of SPKI(1) and a TLSA
Matching Type of Full(0) (which publish the bare public keys without
the overhead of a containing X.509 certificate) are generally more
compact, these are also best avoided when significantly larger than
their digests. Rather, servers SHOULD publish digest-based TLSA
Matching Types in their TLSA records. In which case, the complete
corresponding certificate MUST be transmitted to the client in-band
during the TLS handshake. The certificate (or raw public key) can be
easily verified using the digest value.

In summary, the use of a TLSA Matching Type of Full(0) is NOT
RECOMMENDED and a digest-based matching type, such as SHA2-256(1),
SHOULD be used instead.

10.2. Certificate Name Check Conventions

   Certificates presented by a TLS server will generally contain a
   subjectAltName (SAN) extension or a Common Name (CN) element within
   the subject distinguished name (DN). The TLS server's DNS domain
   name is normally published within these elements, ideally within the
   subjectAltName extension. (The use of the CN field for this purpose
   is deprecated.)

   When a server hosts multiple domains at the same transport endpoint,
   the server's ability to respond with the right certificate chain is
   predicated on correct SNI information from the client. DANE clients
   MUST send the SNI extension with a HostName value of the base domain
   of the TLSA RRset.

   Except with TLSA Certificate Usage DANE-EE(3), where name checks are
   not applicable (see Section 5.1), DANE clients MUST verify that the
   client has reached the correct server by checking that the server
   name is listed in the server certificate's SAN or CN (when still
   supported). The primary server name used for this comparison MUST be
   the TLSA base domain, however additional acceptable names may be
   specified by protocol-specific DANE standards. For example, with
   SMTP both the destination domain name and the MX host name are
   acceptable names to be found in the server certificate (see
   [I-D.ietf-dane-smtp-with-dane]).

   It is the responsibility of the service operator, in coordination
   with the TLSA Publisher, to ensure that at least one of the TLSA
   records published for the service will match the server's certificate
   chain (either the default chain or the certificate that was selected
   based on the SNI information provided by the client).

   Given the DNSSEC validated DNS records below:

   example.com.   IN MX 0 mail.example.com.
   mail.example.com.   IN A 192.0.2.1
   _25._tcp.mail.example.com. IN TLSA 2 0 1 (
                                 E8B54E0B4BAA815B06D3462D65FBC7C0
                                 CF556ECCF9F5303EBFBB77D022F834C0)

   The TLSA base domain is "mail.example.com" and is required to be the
   HostName in the client's SNI extension. The server certificate chain
   is required to be be signed by a trust anchor with the above
   certificate SHA2-256 digest. Finally, one of the DNS names in the
   server certificate is required to be be either "mail.example.com" or
   "example.com" (this additional name is a concession to compatibility
   with prior practice, see [I-D.ietf-dane-smtp-with-dane] for details).

[RFC6125] specifies the semantics of wildcards in server certificates
for various application protocols. DANE does not change how
wildcards are treated by any given application.

10.3. Design Considerations for Protocols Using DANE

When a TLS client goes to the trouble of authenticating a certificate
chain presented by a TLS server, it will typically not continue to
use that server in the event of authentication failure, or else
authentication serves no purpose. Some clients may, at times,
operate in an "audit" mode, where authentication failure is reported
to the user or in logs as a potential problem, but the connection
proceeds despite the failure. Nevertheless, servers publishing TLSA
records MUST be configured to allow correctly configured clients to
successfully authenticate their TLS certificate chains.

A service with DNSSEC-validated TLSA records implicitly promises TLS
support. When all the TLSA records for a service are found
"unusable", due to unsupported parameter combinations or malformed
certificate association data, DANE clients cannot authenticate the
service certificate chain. When authenticated TLS is mandatory, the
client MUST NOT connect to the associated server.

If, on the other hand, the use of TLS and DANE is "opportunistic"
([RFC7435]), then when all TLSA records are unusable, the client
SHOULD connect to the server via an unauthenticated TLS connection,
and if TLS encryption cannot be established, the client MUST NOT
connect to the server.

Standards for opportunistic DANE TLS specific to a particular
application protocol may modify the above requirements. The key
consideration is whether mandating the use of (unauthenticated) TLS
even with unusable TLSA records is asking for more security than one
can realistically expect. If expecting TLS support when unusable
TLSA records are published is realistic for the application in
question, then the application MUST avoid cleartext. If not
realistic, then mandating TLS would cause clients (even in the
absence of active attacks) to run into problems with various peers
that do not interoperate "securely enough". That would create strong
incentives to just disable opportunistic security and stick with
cleartext.

11. Note on DNSSEC Security

Clearly the security of the DANE TLSA PKI rests on the security of
the underlying DNSSEC infrastructure. While this document is not a
guide to DNSSEC security, a few comments may be helpful to TLSA
implementers.

With the existing public CA Web PKI, name constraints are rarely
used, and a public root CA can issue certificates for any domain of
its choice. With DNSSEC, under the Registry/Registrar/Registrant
model, the situation is different: only the registrar of record can
update a domain's DS record in the registry parent zone (in some
cases, however, the registry is the sole registrar). With many
Generic Top Level Domains (gTLDs), for which multiple registrars
compete to provide domains in a single registry, it is important to
make sure that rogue registrars cannot easily initiate an
unauthorized domain transfer, and thus take over DNSSEC for the
domain. DNS Operators are advised to set a registrar lock on their
domains to offer some protection against this possibility.

When the registrar is also the DNS operator for the domain, one needs
to consider whether the registrar will allow orderly migration of the
domain to another registrar or DNS operator in a way that will
maintain DNSSEC integrity. TLSA Publishers are advised to seek out a
DNS hosting registrar that makes it possible to transfer domains to
another hosting provider without disabling DNSSEC.

DNSSEC signed RRsets cannot be securely revoked before they expire.
Operators need to plan accordingly and not generate signatures of
excessively long duration. For domains publishing high-value keys, a
signature lifetime of a few days is reasonable, and the zone can be
re-signed daily. For domains with less critical data, a reasonable
signature lifetime is a couple of weeks to a month, and the zone can
be re-signed weekly.

Short signature lifetimes require tighter synchronization of primary
and secondary nameservers, to make sure that secondary servers never
serve records with expired signatures. They also limit the maximum
time for which a primary server that signs the zone can be down.
Therefore, short signature lifetimes are more appropriate for sites
with dedicated operations staff, who can restore service quickly in
case of a problem.

Monitoring is important. If a DNS zone is not re-signed in a timely
manner, a major outage is likely as the entire domain and all its
sub-domains become "bogus".

12. Summary of Updates to RFC6698

   o Section 3 updates [RFC6698] to specify a requirement for clients
      to support at least TLS 1.0, and to support SNI.

   o Section 5.1 updates [RFC6698] to specify peer identity matching
      and certificate validity interval based solely on the basis of the
      TLSA RRset. It also specifies DANE authentication of raw public

keys [RFC7250] via TLSA records with Certificate Usage DANE-EE(3)
and selector SPKI(1).

o Section 5.2 updates [RFC6698] to require that servers publishing
   digest TLSA records with a usage of DANE-TA(2) MUST include the
   trust-anchor certificate in their TLS server certificate message.
   This extends to the case of "2 1 0" TLSA records which publish a
   full public key.

o Section 5.3 and Section 5.4 explain that PKIX-EE(1) and PKIX-TA(0)
   are generally NOT RECOMMENDED. This document notes that with
   usage PKIX-TA(0) clients may need to process extended trust chains
   beyond the first trusted issuer, when that issuer is not self-
   signed.

o Section 7 recommends that DANE application protocols specify that,
   when possible, securely CNAME expanded names be used to derive the
   TLSA base domain.

o Section 8 specifies a strategy for managing TLSA records that
   interoperates with DANE clients regardless of what subset of the
   possible TLSA record types (combinations of TLSA parameters) is
   supported by the client.

o Section 9 specifies a digest algorithm agility protocol.

o Section 10.1 recommends against the use of Full(0) TLSA records,
   as digest records are generally much more compact.

13. Operational Considerations

The DNS time-to-live (TTL) of TLSA records needs to be chosen with
care. When an unplanned change in the server's certificate chain and
TLSA RRset is required, such as when keys are compromised or lost,
clients that cache stale TLSA records will fail to validate the
certificate chain of the updated server. Publish TLSA RRsets with
TTLs that are short enough to limit unplanned service disruption to
an acceptable duration.

The signature validity period for TLSA records SHOULD NOT be too
long. Signed DNSSEC records can be replayed by an MiTM attacker
provided the signatures have not yet expired. Shorter signature
validity periods allow for faster invalidation of compromised keys.
Zone refresh and expiration times for secondary nameservers often
imply a lower bound on the signature validity period (Section 11).
See Section 4.4.1 of [RFC6781].

14. Security Considerations

   Application protocols that cannot make use of the existing public CA
   Web PKI, may choose to not implement certain TLSA record types
   defined in [RFC6698]. If such records are published despite not
   being supported by the application protocol, they are treated as
   "unusable". When TLS is opportunistic, the client MAY proceed to use
   the server with mandatory unauthenticated TLS. This is stronger than
   opportunistic TLS without DANE, since in that case the client may
   also proceed with a cleartext connection. When TLS is not
   opportunistic, the client MUST NOT connect to the server.

   Thus, when TLSA records are used with opportunistic protocols where
   the PKIX-TA(0) and PKIX-EE(1) do not apply, the recommended protocol
   design is for servers to not publish such TLSA records, and for
   opportunistic TLS clients to use them to only enforce the use of
   (albeit unauthenticated) TLS, but otherwise treat them as unusable.
   Of course, when PKIX-TA(0) and PKIX-EE(1) are supported by the
   application protocol, clients MUST implement these certificate usages
   as described in [RFC6698] and this document.

15. IANA Considerations

   This specification requires no support from IANA.

16. Acknowledgements

   The authors would like to thank Phil Pennock for his comments and
   advice on this document.

   Acknowledgments from Viktor: Thanks to Tony Finch who finally prodded
   me into participating in DANE working group discussions. Thanks to
   Paul Hoffman who motivated me to produce this document and provided
   feedback on early drafts. Thanks also to Samuel Dukhovni for
   editorial assistance.

17. References

17.1. Normative References

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

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

   [RFC4034] Arends, R., Austein, R., Larson, M., Massey, D., and S.
              Rose, "Resource Records for the DNS Security Extensions",
              RFC 4034, DOI 10.17487/RFC4034, March 2005,
              <http://www.rfc-editor.org/info/rfc4034>.

   [RFC4035] Arends, R., Austein, R., Larson, M., Massey, D., and S.
              Rose, "Protocol Modifications for the DNS Security
              Extensions", RFC 4035, DOI 10.17487/RFC4035, March 2005,
              <http://www.rfc-editor.org/info/rfc4035>.

   [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>.

   [RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S.,
              Housley, R., and W. Polk, "Internet X.509 Public Key
              Infrastructure Certificate and Certificate Revocation List
              (CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008,
              <http://www.rfc-editor.org/info/rfc5280>.

   [RFC6066] Eastlake 3rd, D., "Transport Layer Security (TLS)
              Extensions: Extension Definitions", RFC 6066, DOI
              10.17487/RFC6066, January 2011,
              <http://www.rfc-editor.org/info/rfc6066>.

   [RFC6125] Saint-Andre, P. and J. Hodges, "Representation and
              Verification of Domain-Based Application Service Identity
              within Internet Public Key Infrastructure Using X.509
              (PKIX) Certificates in the Context of Transport Layer
              Security (TLS)", RFC 6125, DOI 10.17487/RFC6125, March
              2011, <http://www.rfc-editor.org/info/rfc6125>.

   [RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer
              Security Version 1.2", RFC 6347, DOI 10.17487/RFC6347,
              January 2012, <http://www.rfc-editor.org/info/rfc6347>.

   [RFC6698] Hoffman, P. and J. Schlyter, "The DNS-Based Authentication
              of Named Entities (DANE) Transport Layer Security (TLS)
              Protocol: TLSA", RFC 6698, DOI 10.17487/RFC6698, August
              2012, <http://www.rfc-editor.org/info/rfc6698>.

   [RFC7218] Gudmundsson, O., "Adding Acronyms to Simplify
              Conversations about DNS-Based Authentication of Named
              Entities (DANE)", RFC 7218, DOI 10.17487/RFC7218, April
              2014, <http://www.rfc-editor.org/info/rfc7218>.

   [RFC7250] Wouters, P., Ed., Tschofenig, H., Ed., Gilmore, J.,
              Weiler, S., and T. Kivinen, "Using Raw Public Keys in
              Transport Layer Security (TLS) and Datagram Transport
              Layer Security (DTLS)", RFC 7250, DOI 10.17487/RFC7250,
              June 2014, <http://www.rfc-editor.org/info/rfc7250>.

17.2. Informative References

   [I-D.ietf-dane-smtp-with-dane]
              Dukhovni, V. and W. Hardaker, "SMTP security via
              opportunistic DANE TLS", draft-ietf-dane-smtp-with-dane-19
              (work in progress), May 2015.

   [I-D.ietf-dane-srv]
              Finch, T., Miller, M., and P. Saint-Andre, "Using DNS-
              Based Authentication of Named Entities (DANE) TLSA Records
              with SRV Records", draft-ietf-dane-srv-14 (work in
              progress), April 2015.

   [RFC6781] Kolkman, O., Mekking, W., and R. Gieben, "DNSSEC
              Operational Practices, Version 2", RFC 6781, DOI 10.17487/
              RFC6781, December 2012,
              <http://www.rfc-editor.org/info/rfc6781>.

   [RFC6962] Laurie, B., Langley, A., and E. Kasper, "Certificate
              Transparency", RFC 6962, DOI 10.17487/RFC6962, June 2013,
              <http://www.rfc-editor.org/info/rfc6962>.

   [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>.

Authors' Addresses

   Viktor Dukhovni
   Unaffiliated

   Email: ietf-dane@dukhovni.org

   Wes Hardaker
   Parsons
   P.O. Box 382
   Davis, CA 95617
   US

   Email: ietf@hardakers.net