DANE
Internet-Draft
Intended status: Standards Track
Expires: April 29, 2015

V. Dukhovni
Unaffiliated
W. Hardaker
Parsons

October 26, 2014

Updates to and Operational Guidance for the DANE Protocol
draft-ietf-dane-ops-07

Abstract

This memo clarifies and updates the DNS-Based Authentication of Named Entities (DANE) TLSA specification 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 April 29, 2015.

Table of Contents

1. Introduction

The DANE TLSA specification ([RFC6698]) introduces the DNS "TLSA" resource record type. TLSA records associate a certificate or a public key of an end-entity or a trusted issuing authority with the corresponding TLS transport endpoint. DNSSEC validated DANE TLSA records can be used to augment or replace the use of trusted public Certification Authorities (CAs).

[RFC6698] defines three TLSA record fields with respectively 4, 2 and 3 currently specified values. These yield 24 distinct combinations of TLSA record types. This memo will recommend a smaller set of best-practice combinations of these fields to simplify protocol design, implementation and deployment.

Implementation complexity also arises from the fact that the TLS transport endpoint is often specified indirectly via Service Records (SRV), Mail Exchange (MX) records, CNAME records or other mechanisms that map an abstract service domain to a concrete server domain. With service indirection there are multiple potential places for clients to find the relevant TLSA records. Service indirection is often used to implement "virtual hosting", where a single Service Provider transport endpoint simultaneously supports multiple hosted domain names. With services that employ TLS, such hosting arrangements may require the Service Provider to deploy multiple pairs of private keys and certificates. The TLS clients, then, signal the desired domain to the TLSA server via the Server Name Indication (SNI) extension ([RFC6066], section 3). This memo provides operational guidelines intended to maximize interoperability between DANE TLS clients and servers.

In the context of this memo, channel security is assumed to be provided by TLS or DTLS. The Transport Layer Security (TLS) [RFC5246] and Datagram Transport Layer Security (DTLS) [RFC6347] protocols provide secured TCP and UDP communication, respectively, over IP. By convention, "TLS" will be used throughout this document and, unless otherwise specified, the text applies equally well to the DTLS over UDP protocol. Used without authentication, TLS provides protection only against eavesdropping through its use of encryption. With authentication, TLS also provides integrity protection and authentication, which protects the transport against man-in-the-middle (MITM) attacks.

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

In Section 12 we summarize 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:

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 must carefully manage any TLS credentials offered to connecting applications to ensure name matching is handled easily by the applications.

Customer Domain: As described above, a TLS client may be interacting with a service that is hosted by a third party. We will refer 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 PKIX certificate with a default list of server names, making it difficult to host multiple Customer Domains at the same IP-addressed based TLS service endpoint (i.e., "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 fourth and final field. To avoid language contortions when we need to distinguish between the first three fields that together define a TLSA record "type" and the fourth that provides a data value of that type, we will call the first three fields "TLSA parameters", or sometimes just "parameters" when obvious from context.

2. DANE TLSA Record Overview

DANE TLSA [RFC6698] specifies a protocol for publishing TLS server certificate associations via DNSSEC [RFC4033] [RFC4034] [RFC4035].

The DANE TLSA specification defines multiple TLSA RR types via combinations of numeric values of the first three fields of the TLSA record (i.e. the "TLSA parameters"). The numeric values of these parameters were later given symbolic names in [RFC7218]. These parameters are:

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.

We may think of TLSA Certificate Usage values 0 through 3 as a combination of two one-bit flags. The low-bit chooses between trust anchor (TA) and end entity (EE) certificates. The high bit chooses between PKIX, or public PKI issued, and DANE, or domain-issued trust anchors:

- When the low bit is set (PKIX-EE(1) and DANE-EE(3)) the TLSA record matches an EE certificate (also commonly referred to as a leaf or server certificate.)

- When the low bit is not set (PKIX-TA(0) and DANE-TA(2)) the TLSA record matches a trust anchor (a Certification Authority) that issued one of the certificates in the server certificate chain.

- When the high bit is set (DANE-TA(2) and DANE-EE(3)), the server certificate chain is domain-issued and may be verified without reference to any pre-configured Certification Authority trust anchor. Trust is based solely on the content of the TLSA records obtained via DNSSEC.

- When the high bit is not set (PKIX-TA(0) and PKIX-EE(1)), the TLSA record publishes a server policy stating that its certificate chain must pass PKIX validation [RFC5280] and the DANE TLSA record is used to signal an additional requirement that the PKIX validated server certificate chain also contains the referenced CA or EE certificate.

The selector field specifies whether the TLSA RR matches the whole certificate (Cert(0)) or just its subjectPublicKeyInfo (SPKI(1)). The subjectPublicKeyInfo is an ASN.1 DER encoding of the certificate's algorithm id, any parameters and the public key data.

The matching type field specifies how the TLSA RR Certificate Association Data field is to be compared with the certificate or public key. A value of Full(0) means an exact match: the full DER encoding of the certificate or public key is given in the TLSA RR. A value of SHA2-256(1) means that the association data matches the SHA2-256 digest of the certificate or public key, and likewise SHA2-512(2) means a SHA2-512 digest is used. 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 with digest algorithm agility.

2.1. Example TLSA record

In the example TLSA record below:

```
_25._tcp.mail.example.com. IN TLSA PKIX-TA Cert SHA2-256 (
                           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. TLS clients and servers using DANE SHOULD support the "Server Name Indication" (SNI) extension of TLS.

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.

Protocol designers should carefully consider which set of DANE certificate usages to support. Simultaneous support for all four usages is NOT RECOMMENDED for DANE clients. Protocol designers should either use the PKIX-TA(0) and PKIX-EE(1) certificate usages, or use the DANE-TA(2) and DANE-EE(3) usages. If all four usages are deployed together, 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 the PKIX required checks.. In other words, when all four usages are supported, PKIX-TA(2) and PKIX-EE(1) offer only illusory incremental security.

This document recommends that clients should generally support just the DANE-TA(2) and DANE-EE(3) certificate usages. With DANE-TA(2) and DANE-EE(3) clients don't need to track a large changing list of X.509 trust-anchors. Supporting PKIX-TA(0) or PKIX-EE(1) decreases the operational reliability of DANE-based authentication.

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, [I-D.dukhovni-opportunistic-security]), 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. Since the presence and data integrity of TLSA records relies on DNSSEC, PKIX-TA(0) and PKIX-EE(1) TLSA records do not provide additional security. 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 to mitigate the risk of rogue or compromised public CAs issuing unauthorized certificates. This section clarifies the interaction of CT and DANE.

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).

4.3. Transitioning between PKIX-TA/EE and DANE-TA/EE

The choice of preferred certificate usages may need to change as a 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 interval compliance is based solely on the TLSA RRset properties. We also extend [RFC6698] to cover the use of DANE authentication of raw

public keys [I-D.ietf-tls-oob-pubkey] 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.

Similarly, 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 interval of the TLSA record DNSSEC signatures.

With DANE-EE(3) servers that know all the connecting clients are making use of DANE, they need not employ SNI (i.e., the may ignore the client's SNI message) even when the server is known under 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 primary 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.

TLSA records published for DANE servers SHOULD, as a best practice, be "DANE-EE(3) SPKI(1) SHA2-256(1)" records. Since all DANE implementations are required to support SHA2-256, this record type works for all clients and need not change across certificate renewals with the same key. 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.

Another advantage of "DANE-EE(3) SPKI(1)" (with any suitable matching type) TLSA records is that they are compatible with the raw public key TLS extension specified in [I-D.ietf-tls-oob-pubkey]. DANE clients that support this extension can use the TLSA record to authenticate servers that negotiate the use of raw public keys in

place of X.509 certificate chains. 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 compatible 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, this is in conflict with the indicated selector and requires extra logic on clients that not all implementations are expected to provide. Servers SHOULD NOT rely on "DANE-EE(3) Cert(0) Full(0)" TLSA records to publish authentication data for raw public keys.

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.

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:

```
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 tlsa201._dane.example.com.
_443._tcp.www2.example.com.  IN CNAME tlsa201._dane.example.com.
tlsa201._dane.example.com.   IN TLSA 2 0 1 e3b0c44298fc1c14...
```

With usage DANE-TA(2) the server certificates will need to have names that match one of the client's reference identifiers (see [RFC6125]). The 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, the client SHOULD then 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) (these TLSA records are expected to match the digest of a TA certificate or public key), 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.

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) will 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 should not 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 SHOULD include the TA 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, clients are encouraged to 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, 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. This TLSA record matches PKIX-verified trust chains which contain an issuer certificate (root or intermediate) that matches its association data field (typically a certificate or digest).

PKIX-TA(0) also requires more complex coordination 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.

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, a client SHOULD NOT always stop extending the chain when the first 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 the hope that a certificate closer to the root may in fact match the server's TLSA record.

6. Service Provider and TLSA Publisher Synchronization

Complications arise when the TLSA Publisher is not the same entity as the Service Provider. In this situation, the TLSA Publisher and the Service Provider must cooperate 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.

Whenever possible, the TLSA Publisher and the Service Provider should be the same entity. Otherwise, changes in the service certificate chain must be carefully coordinated between the parties involved. Such coordination is difficult and service outages will result when coordination fails.

Having the master 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 must 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. Certificate name checks generally constrain the applicability of TLSA CNAMEs across organizational boundaries to Certificate Usages DANE-EE(3) and DANE-TA(2):

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 DANE-EE SPKI SHA2-256 (
                                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 DANE-TA Cert SHA2-256 (
                                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 DANE-EE SPKI SHA2-256 (
                               8A9A70596E869BED72C69D97A8895DFA
                               D86F300A343FECEFF19E89C27C896BC9 )
_25._tcp.mx2.example.net.   IN TLSA DANE-EE SPKI SHA2-256 (
                               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 must wait for matching TLSA records to be published by its Customer Domains before deploying newly generated keys and certificate chains. In Section 7 below, we describe 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 SNI, 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 a requirement on the TLSA Publisher to ensure that each combination of Certificate Usage, selector and matching type in the server's TLSA RRset MUST include 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 never be the only records published for any given combination of usage, selector and matching type. We describe a TLSA record update algorithm that ensures 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 [I-D.ietf-tls-oob-pubkey] raw public keys, which are only compatible with TLSA records whose Certificate Usage is DANE-EE(3) with selector SPKI(1).

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:

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

; The transitional TLSA RRset published at least 2*TTL seconds
; 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...

; The 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

A more complex involves switching to a trust-anchor or PKIX usage from a chain that is either self-signed, or issued by a private CA and thus not compatible with PKIX. Here the process is to first add TLSA records matching the future chain that is issued by the desired future CA (private or PKIX), but initially with the same parameters as the legacy chain. Then, after deploying the new keys, switch to the new TLSA parameter combination.

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

; A transitional TLSA RRset, published at least 2*TTL before the
; actual key change. The new keys are issued by a DANE-TA(2) CA,
; but for now 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 keys are not an impediment, specify the issuing
; TA of the new keys.
;
_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 combinations 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).

```
; The initial TLSA RRset with 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...

; The 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:

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

- Remove TLSA records matching no longer deployed certificate chains.

- Extend the TLSA RRset with a new combination of parameters (usage, selector and matching type) that is used to generate matching associations for all certificate chains that are published with some other parameter combination.

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. No combination of Certificate Usage, selector and matching type in a server's TLSA RRset should ever 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 TLS 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 any deprecated weaker algorithms that are published for compatibility with less capable clients, but should be ignored when possible. We specify such a protocol below.

Suppose that a DANE TLS client authenticating a TLS server considers digest algorithm "BetterAlg" stronger than digest algorithm "WorseAlg". Suppose further that a server's TLSA RRset contains some records with "BetterAlg" as the digest algorithm. Suppose also that the server adheres to the requirements of Section 8 and ensures that each combination of TLSA parameters contains at least one record that matches the server's current certificate chain (or raw public keys). Under the above assumptions the client can safely ignore TLSA records with the weaker algorithm "WorseAlg", because it suffices to only check the records with the stronger algorithm "BetterAlg".

To make digest algorithm agility possible, all published TLSA RRsets for use with DANE TLS MUST conform to the requirements of Section 8. With servers publishing compliant TLSA RRsets, TLS clients can, for each combination of usage and selector, ignore all digest records except those that employ their notion of the strongest digest algorithm. (The server should only publish algorithms it deems acceptable at all.) The ordering of digest algorithms by strength is not specified in advance; it is entirely up to the TLS client. TLS client implementations SHOULD make the digest algorithm preference ordering a configurable option.

Note, TLSA records with a matching type of Full(0) that publish an entire certificate or public key object play no role in digest algorithm agility. They neither trump the processing of records that employ digests, nor are they ignored in the presence of any records with a digest (i.e. non-zero) matching type.

TLS clients SHOULD use digest algorithm agility when processing the DANE TLSA records of an TLS server. Algorithm agility is to be applied after first discarding any unusable or malformed records (unsupported digest algorithm, or incorrect digest length). Thus, 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.

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 too should be used with caution as they are still larger than necessary. Rather, servers SHOULD publish digest-based TLSA Matching Types in their TLSA records. The complete corresponding certificate should, instead, be transmitted to the client in-band during the TLS handshake, which can be easily verified using the digest value.

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

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. The server name used for this comparison SHOULD be the base domain of the TLSA RRset. 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 DANE-TA Cert SHA2-256  (
                              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).

The semantics of wildcards in server certificates are left to individual application protocol specifications.

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 associated data, DANE clients cannot authenticate the service certificate chain. When authenticated TLS is dictated by the application, the client SHOULD NOT connect to the associated server. If, on the other hand, the use of TLS is "opportunistic", then the client SHOULD generally use the server via an unauthenticated TLS connection, but if TLS encryption cannot be established, the client MUST NOT use the server. Standards for DANE specific to the particular application protocol may modify the above requirements, as appropriate, to specify whether the connection should be established anyway without relying on TLS security, with only encryption but not authentication, or whether to refuse to connect entirely. Application protocols need to specify when to prioritize security over the ability to connect under adverse conditions.

11. Note on DNSSEC Security

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

With the existing public CA 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 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 SHOULD use a registrar lock of 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 SHOULD ensure their registrar publishes a suitable domain transfer policy.

DNSSEC signed RRsets cannot be securely revoked before they expire. Operators should plan accordingly and not generate signatures with excessively long duration periods. For domains publishing high-value keys, a signature lifetime of a few days is reasonable, and the zone should be resigned daily. For domains with less critical data, a reasonable signature lifetime is a couple of weeks to a month, and the zone should be resigned weekly. Monitoring of the signature lifetime is important. If the zone is not resigned in a timely manner, one risks a major outage and the entire domain will become bogus.

12. Summary of Updates to RFC6698

Authors note: is this section needed? Or is it sufficiently clear above that we don't need to restate things here?

- In Section 3 we update [RFC6698] to specify a requirement for clients to support at least TLS 1.0, and to support SNI.

- In Section 5.1 we update [RFC6698] to specify peer identity matching and certificate validity interval based solely on the basis of the TLSA RRset. We also specify DANE authentication of raw public keys [I-D.ietf-tls-oob-pubkey] via TLSA records with Certificate Usage DANE-EE(3) and selector SPKI(1).

- In Section 5.2 we update [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.

- In Section 5.3 and Section 5.4, we explain that PKIX-EE(1) and PKIX-TA(0) are generally NOT RECOMMENDED. With usage PKIX-TA(0) we note that clients may need to processes extended trust chains beyond the first trusted issuer, when that issuer is not self-signed.

- In Section 7, we recommend that DANE application protocols specify that when possible securely CNAME expanded names be used to derive the TLSA base domain.

- In Section 8, we specify 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.

- In Section 9, we propose a digest algorithm agility protocol. [Note: This section does not yet represent the rough consensus of the DANE working group and requires further discussion. Perhaps this belongs in a separate document.]

- In Section 10.1 we recommend 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. TLSA RRsets should have TTLs that are short enough to limit unplanned service disruption to an acceptable duration. For example, TLSA records for SMTP servers with a TTL of approximately an hour are likely sufficiently short. For interactive HTTP services, TLSA record TTLs of approximately 5 minutes may be more appropriate.

The signature validity time for TLSA records should also not be too long. Signed DNSSEC records can be replayed by an MiTM attacker provided the signatures have not yet expired. Shorter signature validity intervals allow for faster invalidation of compromised keys.

14. Security Considerations

Application protocols that cannot make use of the existing public CA PKI (so called non-PKIX protocols), may choose not to implement certain PKIX-dependent 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 plaintext connection. When TLS is not opportunistic, the client MUST NOT connect to the server.

Therefore, when TLSA records are used with protocols where PKIX does not apply, the recommended policy is for servers to not publish PKIX-dependent TLSA records, and for opportunistic TLS clients to use them to enforce the use of (albeit unauthenticated) TLS, but otherwise

treat them as unusable. Of course, when PKIX validation is supported by the application protocol, clients SHOULD perform PKIX validation per [RFC6698].

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 memo 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, March 1997.

[RFC4033] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "DNS Security Introduction and Requirements", RFC 4033, March 2005.

[RFC4034] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Resource Records for the DNS Security Extensions", RFC 4034, March 2005.

[RFC4035] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Protocol Modifications for the DNS Security Extensions", RFC 4035, March 2005.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.

[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, May 2008.

[RFC6066] Eastlake, D., "Transport Layer Security (TLS) Extensions: Extension Definitions", RFC 6066, January 2011.

[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, March 2011.

[RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, January 2012.

[RFC6698] Hoffman, P. and J. Schlyter, "The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA", RFC 6698, August 2012.

[RFC7218] Gudmundsson, O., "Adding Acronyms to Simplify Conversations about DNS-Based Authentication of Named Entities (DANE)", RFC 7218, April 2014.

17.2. Informative References

[I-D.dukhovni-opportunistic-security]
Dukhovni, V., "Opportunistic Security: Some Protection Most of the Time", draft-dukhovni-opportunistic-security-04 (work in progress), August 2014.

[I-D.ietf-dane-smtp-with-dane]
Dukhovni, V. and W. Hardaker, "SMTP security via opportunistic DANE TLS", draft-ietf-dane-smtp-with-dane-12 (work in progress), August 2014.

[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-08 (work in progress), October 2014.

[I-D.ietf-tls-oob-pubkey]
Wouters, P., Tschofenig, H., Gilmore, J., Weiler, S., and T. Kivinen, "Using Raw Public Keys in Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", draft-ietf-tls-oob-pubkey-11 (work in progress), January 2014.

[RFC6962] Laurie, B., Langley, A., and E. Kasper, "Certificate Transparency", RFC 6962, June 2013.

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