                    Domain Name Associations (DNA)
         in the Extensible Messaging and Presence Protocol (XMPP)

Abstract

   This document improves the security of the Extensible Messaging and
   Presence Protocol (XMPP) in two ways. First, it specifies how to
   establish a strong association between a domain name and an XML
   stream, using the concept of "prooftypes". Second, it describes how
   to securely delegate a service domain name (e.g., example.com) to a
   target server hostname (e.g., hosting.example.net); this is
   especially important in multi-tenanted environments where the same
   target server hosts a large number of domains.

Copyright Notice

Table of Contents

1. Introduction

   In systems that use the Extensible Messaging and Presence Protocol
   (XMPP) [RFC6120], it is important to establish a strong association
   between the DNS domain name of an XMPP service (e.g., example.com)
   and the XML stream that a client or peer server initiates with that
   service. In other words, the client or peer server needs to verify
   the identity of the server to which it connects. Additionally,
   servers need to verify incoming connections from other servers.

   To date, such verification has been established based on information
   obtained from the Domain Name System (DNS), the Public Key
   Infrastructure (PKI), or similar sources. In particular, XMPP as
   defined in [RFC6120] assumed that Domain Name Associations (DNA) are
   to be proved using the "PKIX prooftype"; that is, the server's proof
   consists of a PKIX certificate that is checked according to the XMPP
   profile of the matching rules from [RFC6125] (and the overall
   validation rules from [RFC5280]), the client's verification material
   is obtained out of band in the form of a trusted root, and secure DNS
   is not necessary.

   By extending the concept of a domain name association within XMPP,
   this document does the following:

   1. Generalizes the model currently in use so that additional
       prooftypes can be defined if needed.

   2. Provides a basis for modernizing some prooftypes to reflect
       progress in underlying technologies such as DNS Security
       [RFC4033].

   3. Describes the flow of operations for establishing a domain name
       association.

   This document also provides guidelines for secure delegation of a
   service domain name (e.g., example.com) to a target server hostname
   (e.g., hosting.example.net). The need for secure delegation arises
   because the process for resolving the domain name of an XMPP service
   into the IP address at which an XML stream will be negotiated (see
   [RFC6120]) can involve delegation of a service domain name to a
   target server hostname using technologies such as DNS SRV records
   [RFC2782]. A more detailed description of the delegation problem can
   be found in [RFC7711]. The domain name association can be verified
   only if the delegation is done in a secure manner.

2. Terminology

   This document inherits XMPP terminology from [RFC6120] and
   [XEP-0220]; DNS terminology from [RFC1034], [RFC1035], [RFC2782], and
   [RFC4033]; and security terminology from [RFC4949] and [RFC5280].
   The terms "reference identity" and "presented identity" are used as
   defined in the "CertID" specification [RFC6125]. For the sake of
   consistency with [RFC7673], this document uses the terms "service
   domain name" and "target server hostname" to refer to the same
   entities identified by the terms "source domain" and "derived domain"
   from [RFC6125].

3. Client-to-Server (C2S) DNA

   The client-to-server case is much simpler than the server-to-server
   case because the client does not assert a domain name; this means
   that verification happens in only one direction. Therefore, we
   describe this case first to help the reader understand domain name
   associations in XMPP.

3.1. C2S Flow

   The following flow chart illustrates the protocol flow for
   establishing a domain name association for an XML stream from a
   client (C) to a server (S) using the standard PKIX prooftype
   specified in [RFC6120].

```
                         |
                 DNS RESOLUTION ETC.
                         |
     +-----------------STREAM HEADERS--------------------+
     |                                                   |
     |                                                   |
     |   C: <stream to='a.example'>                      |
     |                                                   |
     |   S: <stream from='a.example'>                    |
     |                                                   |
     +---------------------------------------------------+
                         |
     +----------------TLS NEGOTIATION--------------------+
     |                                                   |
     |   S: Server Certificate                           |
     |                                                   |
     +---------------------------------------------------+
                         |
             (client checks certificate and
              establishes DNA for a.example)
```
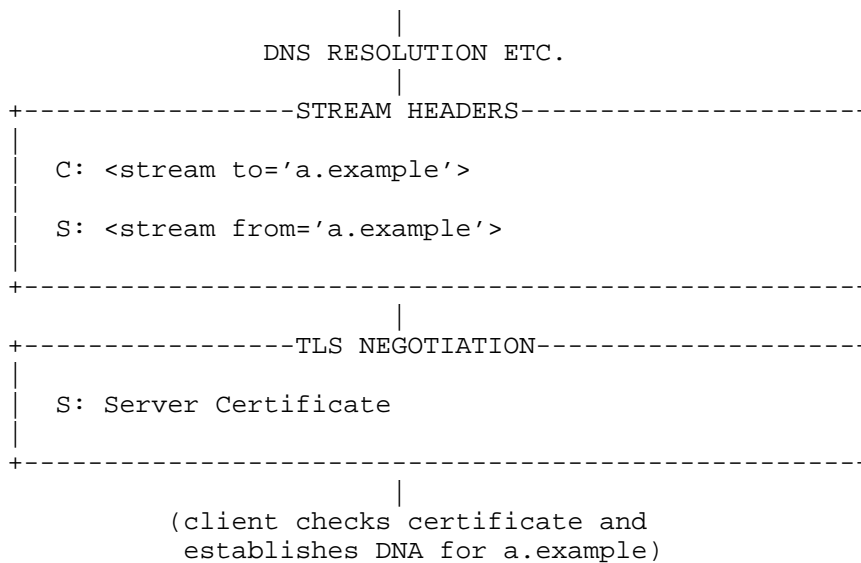
3.2. C2S Description

   The simplified order of events (see [RFC6120] for details) in
   establishing an XML stream from a client (user@a.example) to a server
   (a.example) is as follows:

   1. The client resolves via DNS the service
       _xmpp-client._tcp.a.example.

   2. The client opens a TCP connection to the resolved IP address.

   3. The client sends an initial stream header to the server:

       <stream:stream to='a.example'>

   4. The server sends a response stream header to the client,
       asserting that it is a.example:

       <stream:stream from='a.example'>

   5. The parties attempt TLS negotiation, during which the XMPP server
       (acting as a TLS server) presents a PKIX certificate proving that
       it is a.example.

   6. The client checks the PKIX certificate that the server provided;
       if the proof is consistent with the XMPP profile of the matching
       rules from [RFC6125] and the certificate is otherwise valid
       according to [RFC5280], the client accepts that there is a strong
       domain name association between its stream to the target server
       and the DNS domain name of the XMPP service.

   The certificate that the server presents might not be acceptable to
   the client. As one example, the server might be hosting multiple
   domains and secure delegation as described in Section 6 is necessary.
   As another example, the server might present a self-signed
   certificate, which requires the client to either (1) apply the
   fallback process described in Section 6.6.4 of [RFC6125] or
   (2) prompt the user to accept an unauthenticated connection as
   described in Section 3.4 of [RFC7590].

4. Server-to-Server (S2S) DNA

   The server-to-server case is significantly more complex than the
   client-to-server case, and it involves the checking of domain name
   associations in both directions along with other "wrinkles"
   described in the following sections. In some parts of the flow,
   server-to-server communications use the Server Dialback protocol
   first specified in (the now obsolete) [RFC3920] and since moved to

   [XEP-0220]. See "Impact of TLS and DNSSEC on Dialback" [XEP-0344]
   for considerations when using it together with TLS and DNSSEC. Also,
   "Bidirectional Server-to-Server Connections" [XEP-0288] provides a
   way to use the server-to-server connections for bidirectional
   exchange of XML stanzas, which reduces the complexity of some of the
   processes involved.

4.1. S2S Flow

   The following flow charts illustrate the protocol flow for
   establishing domain name associations between Server 1 (the
   initiating entity) and Server 2 (the receiving entity), as described
   in the remaining sections of this document.

   A simple S2S scenario would be as follows:

```
                           |
                  DNS RESOLUTION ETC.
                           |
     +-------------STREAM HEADERS-------------------+
     |                                              |
     |   A: <stream from='a.example' to='b.example'>  |
     |                                              |
     |   B: <stream from='b.example' to='a.example'>  |
     |                                              |
     +----------------------------------------------+
                           |
     +-------------TLS NEGOTIATION------------------+
     |                                              |
     |   B: Server Certificate                      |
     |   B: Certificate Request                     |
     |   A: Client Certificate                      |
     |                                              |
     +----------------------------------------------+
                           |
          (A establishes DNA for b.example)
                           |
```
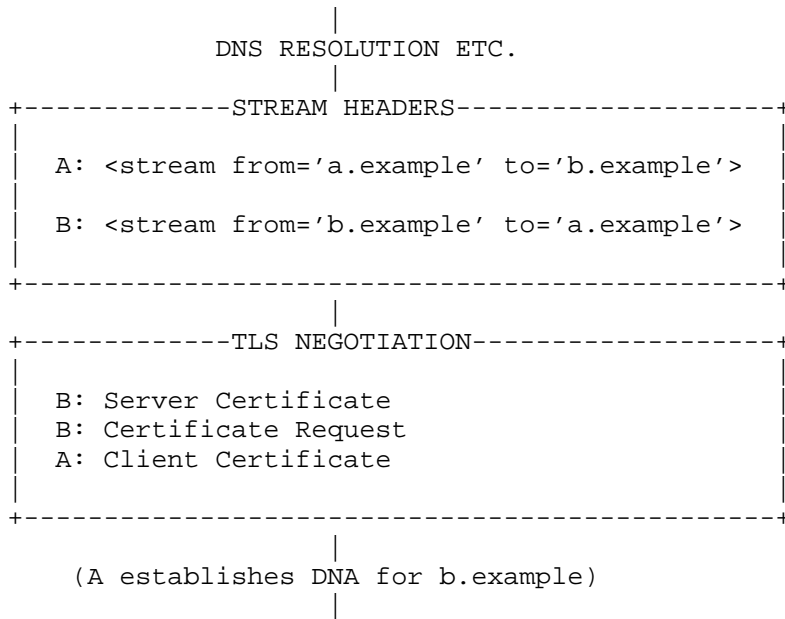
After the domain name association has been established in one
direction, it is possible to perform mutual authentication using the
Simple Authentication and Security Layer (SASL) [RFC4422] and thus
establish domain name associations in both directions.

```
                            |
   +-------------AUTHENTICATION-------------------+
   |                  |                           |
   |        {valid client certificate?} --+      |
   |                  |                    |      |
   |                  | yes           no   |      |
   |                  v                    |      |
   |            SASL EXTERNAL              |      |
   |             (mutual auth)             |      |
   |     (B establishes DNA for a.example) |      |
   +-----------------------------------|---------+
                                       |
```
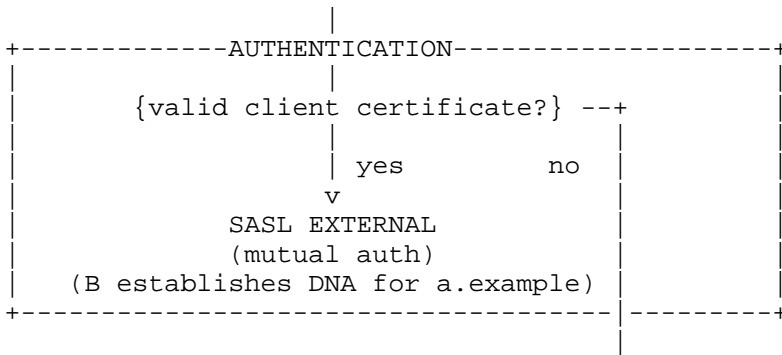
However, if mutual authentication cannot be completed using SASL, the
receiving server needs to establish a domain name association in
another way. This scenario is described in Section 4.3.

```
                               |
            +-----------------+
            |
      (Section 4.3: No Mutual PKIX Authentication)
            |
            | B needs to establish DNA
            | for this stream from a.example,
            | so A asserts its identity
            |
   +----------DIALBACK IDENTITY ASSERTION----------+
   |                                               |
   |  A: <db:result from='a.example'               |
   |              to='b.example'>                   |
   |        some-dialback-key                       |
   |     </db:result>                               |
   |                                               |
   +-----------------------------------------------+
                        |
```

```
                DNS RESOLUTION ETC.
                        |
    +-------------STREAM HEADERS------------------+
    |                                             |
    |   B: <stream from='b.example' to='a.example'> |
    |                                             |
    |   A: <stream from='a.example' to='b.example'> |
    |                                             |
    +---------------------------------------------+
                        |
    +-------------TLS NEGOTIATION------------------+
    |                                             |
    |   A: Server Certificate                     |
    |                                             |
    +---------------------------------------------+
                        |
    +----------DIALBACK IDENTITY VERIFICATION-------+
    |                                             |
    |   B: <db:verify from='b.example'            |
    |                 to='a.example'              |
    |                 id='...'>                   |
    |        some-dialback-key                    |
    |      </db:verify>                           |
    |                                             |
    |   A: <db:verify from='a.example'            |
    |                 to='b.example'              |
    |                 type='valid'                |
    |                 id='...'>                   |
    |                                             |
    +---------------------------------------------+
                        |
        (B establishes DNA for a.example)
                        |
```
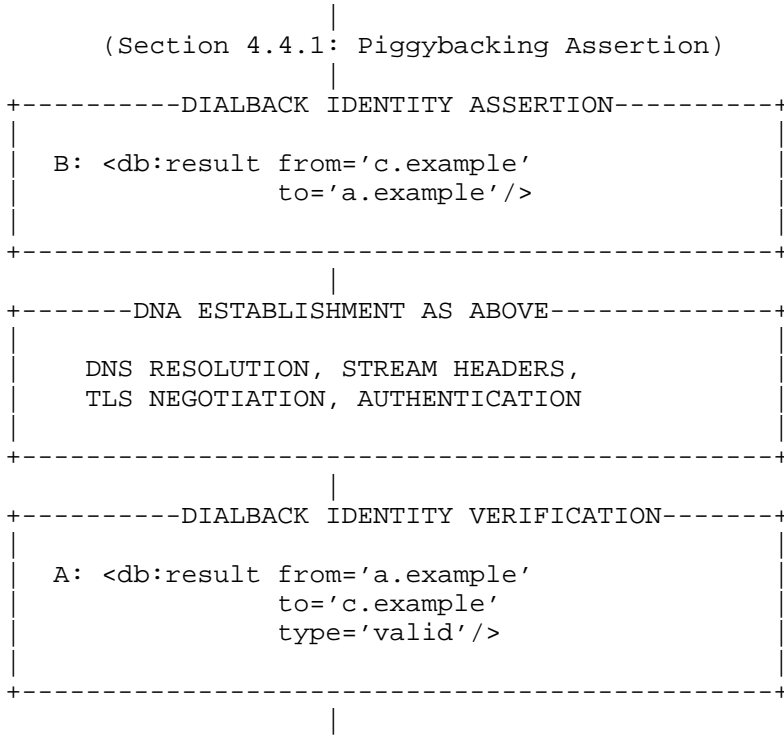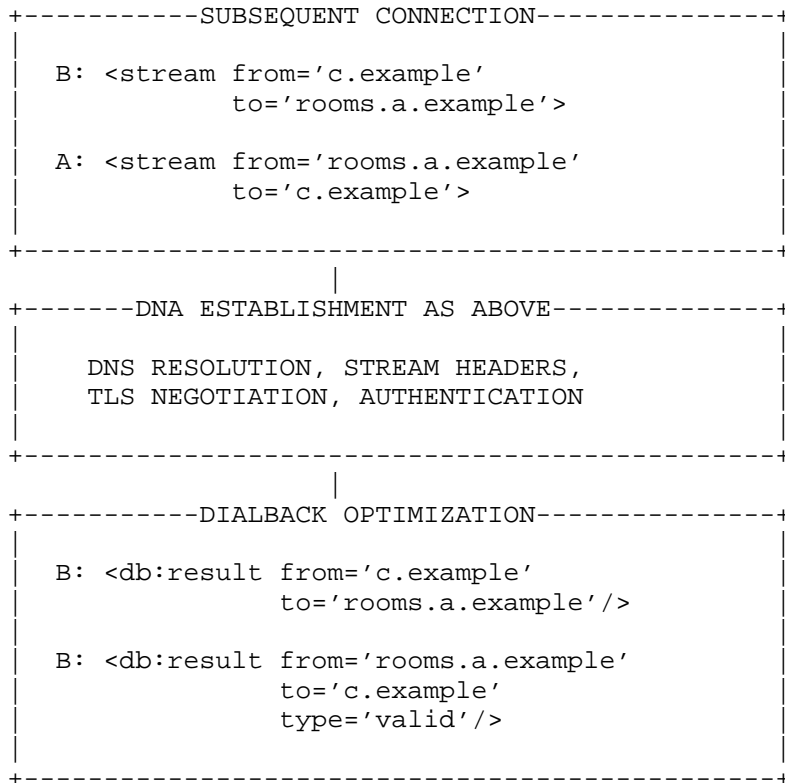
   If one of the servers hosts additional service names (e.g., Server 2
   might host c.example in addition to b.example and Server 1 might host
   rooms.a.example in addition to a.example), then the servers can use
   Server Dialback "piggybacking" to establish additional domain name
   associations for the stream, as described in Section 4.4.

There are two varieties of piggybacking. The first is here called
"assertion".

```
                         |
         (Section 4.4.1: Piggybacking Assertion)
                         |
   +----------DIALBACK IDENTITY ASSERTION----------+
   |                                               |
   |   B: <db:result from='c.example'              |
   |                 to='a.example'/>              |
   |                                               |
   +-----------------------------------------------+
                         |
   +-------DNA ESTABLISHMENT AS ABOVE-------------+
   |                                               |
   |      DNS RESOLUTION, STREAM HEADERS,          |
   |      TLS NEGOTIATION, AUTHENTICATION          |
   |                                               |
   +-----------------------------------------------+
                         |
   +----------DIALBACK IDENTITY VERIFICATION-------+
   |                                               |
   |   A: <db:result from='a.example'              |
   |                 to='c.example'                |
   |                 type='valid'/>                |
   |                                               |
   +-----------------------------------------------+
                         |
```

The second variety of piggybacking is here called "supposition".

```
                            |
          (Section 4.4.2: Piggybacking Supposition)
                            |
    +-----------SUBSEQUENT CONNECTION--------------+
    |                                              |
    |   B: <stream from='c.example'                |
    |              to='rooms.a.example'>           |
    |                                              |
    |   A: <stream from='rooms.a.example'          |
    |              to='c.example'>                 |
    |                                              |
    +----------------------------------------------+
                            |
    +-------DNA ESTABLISHMENT AS ABOVE-------------+
    |                                              |
    |     DNS RESOLUTION, STREAM HEADERS,          |
    |     TLS NEGOTIATION, AUTHENTICATION          |
    |                                              |
    +----------------------------------------------+
                            |
    +-----------DIALBACK OPTIMIZATION--------------+
    |                                              |
    |   B: <db:result from='c.example'             |
    |                 to='rooms.a.example'/>       |
    |                                              |
    |   B: <db:result from='rooms.a.example'       |
    |                 to='c.example'               |
    |                 type='valid'/>               |
    |                                              |
    +----------------------------------------------+
```

4.2. A Simple S2S Scenario

   To illustrate the problem, consider the simplified order of events
   (see [RFC6120] for details) in establishing an XML stream between
   Server 1 (a.example) and Server 2 (b.example):

   1. Server 1 resolves via DNS the service
       _xmpp-server._tcp.b.example.

   2. Server 1 opens a TCP connection to the resolved IP address.

   3. Server 1 sends an initial stream header to Server 2, asserting
       that it is a.example:

       <stream:stream from='a.example' to='b.example'>

4. Server 2 sends a response stream header to Server 1, asserting
    that it is b.example:

    <stream:stream from='b.example' to='a.example'>

5. The servers attempt TLS negotiation, during which Server 2
    (acting as a TLS server) presents a PKIX certificate proving that
    it is b.example and Server 1 (acting as a TLS client) presents a
    PKIX certificate proving that it is a.example.

6. Server 1 checks the PKIX certificate that Server 2 provided, and
    Server 2 checks the PKIX certificate that Server 1 provided; if
    these proofs are consistent with the XMPP profile of the matching
    rules from [RFC6125] and are otherwise valid according to
    [RFC5280], each server accepts that there is a strong domain name
    association between its stream to the other party and the DNS
    domain name of the other party (i.e., mutual authentication is
    achieved).

Several simplifying assumptions underlie the "happy path" scenario
just outlined:

o The PKIX certificate presented by Server 2 during TLS negotiation
   is acceptable to Server 1 and matches the expected identity.

o The PKIX certificate presented by Server 1 during TLS negotiation
   is acceptable to Server 2; this enables the parties to complete
   mutual authentication.

o There are no additional domains associated with Server 1 and
   Server 2 (say, a sub-domain rooms.a.example on Server 1 or a
   second domain c.example on Server 2).

o The server administrators are able to obtain PKIX certificates
   issued by a widely accepted Certification Authority (CA) in the
   first place.

o The server administrators are running their own XMPP servers,
   rather than using hosting services.

Let's consider each of these "wrinkles" in turn.

4.3. No Mutual PKIX Authentication

   If the PKIX certificate presented by Server 1 during TLS negotiation
   is not acceptable to Server 2, Server 2 is unable to mutually
   authenticate Server 1. Therefore, Server 2 needs to verify the
   asserted identity of Server 1 by other means.

   1. Server 1 asserts that it is a.example using the Server Dialback
       protocol:

       <db:result from='a.example' to='b.example'>
                some-dialback-key</db:result>

   2. Server 2 resolves via DNS the service
       _xmpp-server._tcp.a.example.

   3. Server 2 opens a TCP connection to the resolved IP address.

   4. Server 2 sends an initial stream header to Server 1, asserting
       that it is b.example:

       <stream:stream from='b.example' to='a.example'>

   5. Server 1 sends a response stream header to Server 2, asserting
       that it is a.example:

       <stream:stream from='a.example' to='b.example'>

   6. The servers attempt TLS negotiation, during which Server 1
       (acting as a TLS server) presents a PKIX certificate.

   7. Server 2 checks the PKIX certificate that Server 1 provided (this
       might be the same certificate presented by Server 1 as a client
       certificate in the initial connection). However, Server 2 does
       not accept this certificate as proving that Server 1 is
       authorized as a.example and therefore uses another method (here,
       the Server Dialback protocol) to establish the domain name
       association.

   8. Server 2 proceeds with Server Dialback in order to establish the
       domain name association. In order to do this, it sends a request
       for verification as described in [XEP-0220]:

          <db:verify from='b.example' to='a.example'
                     id='...'>some-dialback-key</db:verify>

   9. Server 1 responds to this:

          <db:verify from='a.example' to='b.example' id='...' type='valid/>

          allowing Server 2 to establish the domain name association.

   In some situations (e.g., if the Authoritative Server in Server
   Dialback presents the same certificate as the Originating Server), it
   is the practice of some XMPP server implementations to skip steps 8
   and 9. These situations are discussed in "Impact of TLS and DNSSEC
   on Dialback" [XEP-0344].

4.4. Piggybacking

4.4.1. Assertion

   Consider the common scenario in which Server 2 hosts not only
   b.example but also a second domain c.example (often called a
   "multi-tenanted" environment). If a user of Server 2 associated with
   c.example wishes to communicate with a friend at a.example, Server 2
   needs to send XMPP stanzas from the domain c.example rather than
   b.example. Although Server 2 could open a new TCP connection and
   negotiate new XML streams for the domain pair of c.example and
   a.example, that is wasteful (especially if Server 2 hosts a large
   number of domains). Server 2 already has a connection to a.example,
   so how can it assert that it would like to add a new domain pair to
   the existing connection?

   The traditional method for doing so is the Server Dialback protocol
   [XEP-0220]. Here, Server 2 can send a <db:result/> element for the
   new domain pair over the existing stream.

          <db:result from='c.example' to='a.example'>
            some-dialback-key
          </db:result>

   This <db:result/> element functions as Server 2's assertion that it
   is (also) c.example (thus, the element is functionally equivalent to
   the 'from' address of an initial stream header as previously
   described).

In response to this assertion, Server 1 needs to obtain some kind of
proof that Server 2 really is also c.example. If the certificate
presented by Server 2 is also valid for c.example, then no further
action is necessary. However, if not, then Server 1 needs to do a
bit more work. Specifically, Server 1 can pursue the same strategy
it used before:

1. Server 1 resolves via DNS the service
    _xmpp-server._tcp.c.example.

2. Server 1 opens a TCP connection to the resolved IP address (which
    might be the same IP address as for b.example).

3. Server 1 sends an initial stream header to Server 2, asserting
    that it is a.example:

    <stream:stream from='a.example' to='c.example'>

4. Server 2 sends a response stream header to Server 1, asserting
    that it is c.example:

    <stream:stream from='c.example' to='a.example'>

5. The servers attempt TLS negotiation, during which Server 2
    (acting as a TLS server) presents a PKIX certificate proving that
    it is c.example.

6. At this point, Server 1 needs to establish that, despite
    different certificates, c.example is associated with the origin
    of the request. This is done using Server Dialback [XEP-0220]:

    <db:verify from='a.example' to='c.example'
            id='...'>some-dialback-key</db:verify>

7. Server 2 responds to this:

    <db:verify from='c.example' to='a.example' id='...' type='valid/>

    allowing Server 1 to establish the domain name association.

Now that Server 1 accepts the domain name association, it informs
Server 2 of that fact:

    <db:result from='a.example' to='c.example' type='valid'/>

The parties can then terminate the second connection, because it was
used only for Server 1 to associate a stream with the domain name
c.example (the dialback key links the original stream to the new
association).

4.4.2. Supposition

Piggybacking can also occur in the other direction. Consider the
common scenario in which Server 1 provides XMPP services not only for
a.example but also for a sub-domain such as a Multi-User Chat
[XEP-0045] service at rooms.a.example. If a user from c.example at
Server 2 wishes to join a room on the groupchat service, Server 2
needs to send XMPP stanzas from the domain c.example to the domain
rooms.a.example rather than a.example.

First, Server 2 needs to determine whether it can piggyback the
domain rooms.a.example on the connection to a.example:

1. Server 2 resolves via DNS the service
    _xmpp-server._tcp.rooms.a.example.

2. Server 2 determines that this resolves to an IP address and port
    to which it is already connected.

3. Server 2 determines that the PKIX certificate for that active
    connection would also be valid for the rooms.a.example domain and
    that Server 1 has announced support for dialback errors.

Server 2 sends a dialback key to Server 1 over the existing
connection:

```
<db:result from='c.example' to='rooms.a.example'>
  some-dialback-key
</db:result>
```

Server 1 then informs Server 2 that it accepts the domain name
association:

```
<db:result from='rooms.a.example' to='c.example' type='valid'/>
```

5. Alternative Prooftypes

   The foregoing protocol flows assumed that domain name associations
   were proved using the PKIX prooftype. However, sometimes XMPP server
   administrators are unable or unwilling to obtain valid PKIX
   certificates for all of the domains they host at their servers.
   For example:

   o In order to issue a PKIX certificate, a CA might try to send email
      messages to authoritative mailbox names [RFC2142], but the
      administrator of a subsidiary service such as im.cs.podunk.example
      cannot receive email sent to hostmaster@podunk.example.

   o A hosting provider such as hosting.example.net might not want to
      take on the liability of holding the certificate and private key
      for a tenant such as example.com (or the tenant might not want the
      hosting provider to hold its certificate and private key).

   o Even if PKIX certificates for each tenant can be obtained, the
      management of so many certificates can introduce a large
      administrative load.

   (Additional discussion can be found in [RFC7711].)

   In these circumstances, prooftypes other than PKIX are desirable or
   necessary. As described below, two alternatives have been defined so
   far: DNS-Based Authentication of Named Entities (DANE) and PKIX over
   Secure HTTP (POSH).

5.1. DANE

   The DANE prooftype is defined as follows:

   1. The server's proof consists of either a service certificate or
       domain-issued certificate (TLSA usage PKIX-EE or DANE-EE; see
       [RFC6698] and [RFC7218]).

   2. The proof is checked by verifying an exact match or a hash of
       either the SubjectPublicKeyInfo or the full certificate.

   3. The client's verification material is obtained via secure DNS
       [RFC4033] as described in [RFC7673].

   4. Secure DNS is necessary in order to effectively establish an
       alternative chain of trust from the service certificate or
       domain-issued certificate to the DNS root.

The DANE prooftype makes use of DNS-Based Authentication of Named
Entities [RFC6698], specifically the use of DANE with DNS SRV records
[RFC7673]. For XMPP purposes, the following rules apply:

o If there is no SRV resource record, pursue the fallback methods
   described in [RFC6120].

o Use the 'to' address of the initial stream header to determine the
   domain name of the TLS client's reference identifier (because the
   use of the Server Name Indication extension (TLS SNI) [RFC6066] is
   purely discretionary in XMPP, as mentioned in [RFC6120]).

5.2. POSH

The POSH prooftype is defined as follows:

1. The server's proof consists of a PKIX certificate.

2. The proof is checked according to the rules from [RFC6120] and
    [RFC6125].

3. The client's verification material is obtained by retrieving a
    hash of the PKIX certificate over HTTPS at a well-known URI
    [RFC5785].

4. Secure DNS is not necessary, because the HTTPS retrieval
    mechanism relies on the chain of trust from the public key
    infrastructure.

POSH is defined in [RFC7711]. For XMPP purposes, the following rules
apply:

o If no verification material is found via POSH, pursue the fallback
   methods described in [RFC6120].

o Use the 'to' address of the initial stream header to determine the
   domain name of the TLS client's reference identifier (because the
   use of TLS SNI [RFC6066] is purely discretionary in XMPP, as
   mentioned in [RFC6120]).

The well-known URIs [RFC5785] to be used for POSH are:

o "/.well-known/posh/xmpp-client.json" for client-to-server
   connections

o "/.well-known/posh/xmpp-server.json" for server-to-server
   connections

6. Secure Delegation and Multi-Tenancy

   One common method for deploying XMPP services is multi-tenancy: e.g.,
   XMPP services for the service domain name example.com are actually
   hosted at the target server hosting.example.net. Such an arrangement
   is relatively convenient in XMPP given the use of DNS SRV records
   [RFC2782], such as the following delegation from example.com to
   hosting.example.net:

   _xmpp-server._tcp.example.com. 0 IN SRV 0 0 5269 hosting.example.net

   Secure connections with multi-tenancy can work using the PKIX
   prooftype on a small scale if the provider itself wishes to host
   several domains (e.g., related domains such as jabber-de.example and
   jabber-ch.example). However, in practice the security of
   multi-tenancy has been found to be unwieldy when the provider hosts
   large numbers of XMPP services on behalf of multiple tenants (see
   [RFC7711] for a detailed description). There are two possible
   results: either (1) server-to-server communications to example.com
   are unencrypted or (2) the communications are TLS-encrypted but the
   certificates are not checked (which is functionally equivalent to a
   connection using an anonymous key exchange). This is also true of
   client-to-server communications, forcing end users to override
   certificate warnings or configure their clients to accept or "pin"
   certificates for hosting.example.net instead of example.com. The
   fundamental problem here is that if DNSSEC is not used, then the act
   of delegation via DNS SRV records is inherently insecure.

   The specification for the use of SRV records with DANE [RFC7673]
   explains how to use DNSSEC for secure delegation with the DANE
   prooftype, and the POSH specification [RFC7711] explains how to use
   HTTPS redirects for secure delegation with the POSH prooftype.

7. Prooftype Model

   In general, a Domain Name Association (DNA) prooftype conforms to the
   following definition:

   prooftype: A mechanism for proving an association between a domain
      name and an XML stream, where the mechanism defines (1) the nature
      of the server's proof, (2) the matching rules for comparing the
      client's verification material against the server's proof, (3) how
      the client obtains its verification material, and (4) whether or
      not the mechanism depends on secure DNS.

   The PKIX, DANE, and POSH prooftypes adhere to this model. (Some
   prooftypes depend on, or are enhanced by, secure DNS [RFC4033] and
   thus also need to describe how they ensure secure delegation.)

Other prooftypes are possible; examples might include TLS with Pretty
Good Privacy (PGP) keys [RFC6091], a token mechanism such as Kerberos
[RFC4120] or OAuth [RFC6749], and Server Dialback keys [XEP-0220].

Although the PKIX prooftype reuses the syntax of the XMPP Server
Dialback protocol [XEP-0220] for signaling between servers, this
framework document does not define how the generation and validation
of Server Dialback keys (also specified in [XEP-0220]) constitute a
DNA prooftype. However, nothing in this document prevents the
continued use of Server Dialback for signaling, and a future
specification (or an updated version of [XEP-0220]) might define a
DNA prooftype for Server Dialback keys in a way that is consistent
with this framework.

8. Guidance for Server Operators

This document introduces the concept of a prooftype in order to
explain and generalize the approach to establishing a strong
association between the DNS domain name of an XMPP service and the
XML stream that a client or peer server initiates with that service.

The operations and management implications of DNA prooftypes will
depend on the particular prooftypes that an operator supports.
For example:

o To support the PKIX prooftype [RFC6120], an operator needs to
   obtain certificates for the XMPP server from a Certification
   Authority (CA). However, DNS Security is not required.

o To support the DANE prooftype [RFC7673], an operator can generate
   its own certificates for the XMPP server or obtain them from a CA.
   In addition, DNS Security is required.

o To support the POSH prooftype [RFC7711], an operator can generate
   its own certificates for the XMPP server or obtain them from a CA,
   but in addition needs to deploy the web server for POSH files with
   certificates obtained from a CA. However, DNS Security is not
   required.

Considerations for the use of the foregoing prooftypes are explained
in the relevant specifications. See in particular Section 13.7 of
[RFC6120], Section 6 of [RFC7673], and Section 7 of [RFC7711].

Naturally, these operations and management considerations are
additive: if an operator wishes to use multiple prooftypes, the
complexity of deployment increases (e.g., the operator might want to
obtain a PKIX certificate from a CA for use in the PKIX prooftype and
generate its own certificate for use in the DANE prooftype). This is

an unavoidable aspect of supporting as many prooftypes as needed in
order to ensure that domain name associations can be established in
the largest possible percentage of cases.

9. IANA Considerations

   The POSH specification [RFC7711] establishes the "POSH Service Names"
   registry for use in well-known URIs [RFC5785]. This specification
   registers two such service names for use in XMPP: "xmpp-client" and
   "xmpp-server". The completed registration templates follow.

9.1. POSH Service Name for xmpp-client Service

   Service name: xmpp-client

   Change controller: IETF

   Definition and usage: Specifies the location of a POSH file
      containing verification material or a reference thereto that
      enables a client to verify the identity of a server for a
      client-to-server stream in XMPP

   Specification: RFC 7712 (this document)

9.2. POSH Service Name for xmpp-server Service

   Service name: xmpp-server

   Change controller: IETF

   Definition and usage: Specifies the location of a POSH file
      containing verification material or a reference thereto that
      enables a server to verify the identity of a peer server for a
      server-to-server stream in XMPP

   Specification: RFC 7712 (this document)

10. Security Considerations

   With regard to the PKIX prooftype, this document supplements but does
   not supersede the security considerations of [RFC6120] and [RFC6125].

   With regard to the DANE and POSH prooftypes, the reader is referred
   to [RFC7673] and [RFC7711], respectively.

   Any future prooftypes need to thoroughly describe how they conform to
   the prooftype model specified in Section 7 of this document.

11. References

11.1. Normative References

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

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

   [RFC2782]   Gulbrandsen, A., Vixie, P., and L. Esibov, "A DNS RR for
               specifying the location of services (DNS SRV)", RFC 2782,
               DOI 10.17487/RFC2782, February 2000,
               <http://www.rfc-editor.org/info/rfc2782>.

   [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>.

   [RFC4422]   Melnikov, A., Ed., and K. Zeilenga, Ed., "Simple
               Authentication and Security Layer (SASL)", RFC 4422,
               DOI 10.17487/RFC4422, June 2006,
               <http://www.rfc-editor.org/info/rfc4422>.

   [RFC4949]   Shirey, R., "Internet Security Glossary, Version 2",
               FYI 36, RFC 4949, DOI 10.17487/RFC4949, August 2007,
               <http://www.rfc-editor.org/info/rfc4949>.

   [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>.

   [RFC5785]   Nottingham, M. and E. Hammer-Lahav, "Defining Well-Known
               Uniform Resource Identifiers (URIs)", RFC 5785,
               DOI 10.17487/RFC5785, April 2010,
               <http://www.rfc-editor.org/info/rfc5785>.

   [RFC6120]   Saint-Andre, P., "Extensible Messaging and Presence
               Protocol (XMPP): Core", RFC 6120, DOI 10.17487/RFC6120,
               March 2011, <http://www.rfc-editor.org/info/rfc6120>.

   [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>.

   [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>.

   [RFC7673] Finch, T., Miller, M., and P. Saint-Andre, "Using
              DNS-Based Authentication of Named Entities (DANE) TLSA
              Records with SRV Records", RFC 7673,
              DOI 10.17487/RFC7673, October 2015,
              <http://www.rfc-editor.org/info/rfc7673>.

   [RFC7711] Miller, M. and P. Saint-Andre, "PKIX over Secure HTTP
              (POSH)", RFC 7711, DOI 10.17487/RFC7711, November 2015,
              <http://www.rfc-editor.org/info/rfc7711>.

   [XEP-0220] Miller, J., Saint-Andre, P., and P. Hancke, "Server
              Dialback", XSF XEP 0220, August 2014,
              <http://xmpp.org/extensions/xep-0220.html>.

11.2. Informative References

   [RFC2142]   Crocker, D., "Mailbox Names for Common Services, Roles
                and Functions", RFC 2142, DOI 10.17487/RFC2142, May 1997,
                <http://www.rfc-editor.org/info/rfc2142>.

   [RFC3920]   Saint-Andre, P., Ed., "Extensible Messaging and Presence
                Protocol (XMPP): Core", RFC 3920, DOI 10.17487/RFC3920,
                October 2004, <http://www.rfc-editor.org/info/rfc3920>.

   [RFC4120]   Neuman, C., Yu, T., Hartman, S., and K. Raeburn, "The
                Kerberos Network Authentication Service (V5)", RFC 4120,
                DOI 10.17487/RFC4120, July 2005,
                <http://www.rfc-editor.org/info/rfc4120>.

   [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>.

   [RFC6091]   Mavrogiannopoulos, N. and D. Gillmor, "Using OpenPGP Keys
                for Transport Layer Security (TLS) Authentication",
                RFC 6091, DOI 10.17487/RFC6091, February 2011,
                <http://www.rfc-editor.org/info/rfc6091>.

   [RFC6749]   Hardt, D., Ed., "The OAuth 2.0 Authorization Framework",
                RFC 6749, DOI 10.17487/RFC6749, October 2012,
                <http://www.rfc-editor.org/info/rfc6749>.

   [RFC7590]   Saint-Andre, P. and T. Alkemade, "Use of Transport Layer
                Security (TLS) in the Extensible Messaging and Presence
                Protocol (XMPP)", RFC 7590, DOI 10.17487/RFC7590,
                June 2015, <http://www.rfc-editor.org/info/rfc7590>.

   [XEP-0045]   Saint-Andre, P., "Multi-User Chat", XSF XEP 0045,
                February 2012,
                <http://xmpp.org/extensions/xep-0045.html>.

   [XEP-0288]   Hancke, P. and D. Cridland, "Bidirectional
                Server-to-Server Connections", XSF XEP 0288,
                September 2013,
                <http://xmpp.org/extensions/xep-0288.html>.

   [XEP-0344]   Hancke, P. and D. Cridland, "Impact of TLS and DNSSEC on
                Dialback", XSF XEP 0344, March 2015,
                <http://xmpp.org/extensions/xep-0344.html>.

Acknowledgements

   Richard Barnes, Stephen Farrell, and Jonas Lindberg contributed as
   co-authors to earlier draft versions of this document.

   Derek Atkins, Mahesh Jethanandani, and Dan Romascanu reviewed the
   document on behalf of the Security Directorate, the Operations and
   Management Directorate, and the General Area Review Team,
   respectively.

   During IESG review, Stephen Farrell and Barry Leiba provided helpful
   input that led to improvements in the specification.

   Thanks to Dave Cridland as document shepherd, Joe Hildebrand as
   working group chair, and Ben Campbell as area director.

   Peter Saint-Andre wishes to acknowledge Cisco Systems, Inc., for
   employing him during his work on earlier draft versions of this
   document.

Authors' Addresses

   Peter Saint-Andre
   &yet

   Email: peter@andyet.com
   URI: https://andyet.com/

   Matthew Miller
   Cisco Systems, Inc.
   1899 Wynkoop Street, Suite 600
   Denver, CO 80202
   United States

   Email: mamille2@cisco.com

   Philipp Hancke
   &yet

   Email: fippo@andyet.com
   URI: https://andyet.com/