XEP-0417: E2E Authentication in XMPP: Certificate Issuance and Revocation

Evgeny Khramtsov
mailto:ekhramtsov@process-one.net
xmpp:xram@zinid.ru

2019-03-29
Version 0.1.0

Status	Type	Short Name
Deferred	Standards Track	NOT_YET_ASSIGNED

This specification defines a way for a certificate authority to serve certificate signing requests via XMPP in order to issue X.509 certificates for the use in end-to-end and c2s SASL EXTERNAL authentication.

Legal

Copyright

This XMPP Extension Protocol is copyright © 1999 – 2024 by the XMPP Standards Foundation (XSF).

Permissions

Permission is hereby granted, free of charge, to any person obtaining a copy of this specification (the ”Specification”), to make use of the Specification without restriction, including without limitation the rights to implement the Specification in a software program, deploy the Specification in a network service, and copy, modify, merge, publish, translate, distribute, sublicense, or sell copies of the Specification, and to permit persons to whom the Specification is furnished to do so, subject to the condition that the foregoing copyright notice and this permission notice shall be included in all copies or substantial portions of the Specification. Unless separate permission is granted, modified works that are redistributed shall not contain misleading information regarding the authors, title, number, or publisher of the Specification, and shall not claim endorsement of the modified works by the authors, any organization or project to which the authors belong, or the XMPP Standards Foundation.

Warranty

NOTE WELL: This Specification is provided on an ”AS IS” BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE.

Liability

In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall the XMPP Standards Foundation or any author of this Specification be liable for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising from, out of, or in connection with the Specification or the implementation, deployment, or other use of the Specification (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if the XMPP Standards Foundation or such author has been advised of the possibility of such damages.

Conformance

This XMPP Extension Protocol has been contributed in full conformance with the XSF’s Intellectual Property Rights Policy (a copy of which can be found at <https://xmpp.org/about/xsf/ipr-policy> or obtained by writing to XMPP Standards Foundation, P.O. Box 787, Parker, CO 80134 USA).

Contents

1 Introduction

E2E Authentication in XMPP (XEP-EAX) specifies certificate requirements for end-to-end authentication. This document describes how such certificates can be obtained directly by an XMPP client from a trusted certificate authority (CA) using the XMPP protocol. This assumes that the CA runs an XMPP server. The CA functionality can be built into the user's server, but this is not a requirement: a client can obtain a certificate from any trusted CA server. In the latter case the user's server should support s2s connectivity with CA servers and, in addition, it may want to trust them if it wishes to accept c2s SASL EXTERNAL authentication (Best Practices for Use of SASL EXTERNAL (XEP-0178) [1]) for users of those certificates as long as the certificates are issued for the users of this server. In order to improve user experience (UX), an account registration and certificate issuance can be combined into a single step if the account's server supports this specification.

2 Requirements

2.1 CA Server Requirements

CA servers MUST NOT allow unencrypted XMPP or HTTP connections.

2.2 CA Certificate Requirements

1. A CA certificate MUST fulfill the requirements outlined in XEP-EAX.
2. Additionally, a CA that manages an XMPP server for certificates issuance using the protocol described herein MUST encode an XMPP address of the XMPP server in its own certificate as an XmppAddr identifier (see Section 13.7.1.4 of RFC 6120 [2]). The node and resource parts of the address MUST be empty (omitted). Although there are several ways to encode domain names in X.509 certificates, an XmppAddr identifier type is chosen to provide an indication that the CA accepts certificate signing requests over XMPP.

2.3 CSR Requirements

The following rules apply to a certificate signing request:

1. It MUST be an ASN.1 CertificateRequest structure which MUST conform to RFC2986.
2. Its subject field SHOULD be empty: a subject of the certificate will be generated by the CA server.

[1] XEP-0178: Best Practices for Use of SASL EXTERNAL <https://xmpp.org/extensions/xep-0178.html>.
[2] RFC 6120: Extensible Messaging and Presence Protocol (XMPP): Core <http://tools.ietf.org/html/rfc6120>.

3. It MUST contain extensionRequest attribute requesting a bare XMPP address encoded within subjectAltName as an XmppAddr identifier type (see Section 13.7.1.4 of RFC 6120 [3]). The XMPP address MUST be the one a client wishes to associate the requested certificate with.

4. It MAY contain other extensionRequest attributes requesting other subjectAltNames such as rfc822Name (email address), a "tel" URI (RFC 3966 [4]) and so on. In this case a client MUST be prepared to be additionally challenged by the CA server to prove possession of the corresponding identities. A client also MUST be prepared that those attributes may be either ignored and omitted in the issued certificate or the whole request rejected.

5. It SHOULD NOT contain other extensionRequest attributes.

3 Glossary

CA Server An XMPP server managed by CA to serve certificate signing requests using the protocol described in this document.

X.509 IBR A procedure of in-band registration of an XMPP account combined with certificate issuance. See X.509 IBR section for details.

The "Jabber" network A publicly available federated network of XMPP servers and clients.

See also Glossary section of XEP-EAX: terminology from there is heavily used in this document.

4 X.509 Elements

4.1 Certificate Elements

An X.509 certificate and a chain of X.509 certificates are represented by <x509-cert/> and <x509-cert-chain/> elements respectively, qualified by 'urn:xmpp:x509:0' namespace. These elements can be included into other XMPP elements such as messages, subscription requests and so on.
Character data of the <x509-cert/> element MUST be a PEM encoded ASN.1 Certificate structure (Section 5.1 of RFC7468) with encapsulation boundaries (BEGIN/END) removed. The <x509-cert/> element MUST NOT contain any child elements.
The <x509-cert-chain/> element MUST contain one or many <x509-cert/> elements. Those elements MUST be ordered: each certificate in the chain is signed by the entity identified by the next certificate in the chain. A root certificate MAY be included in the chain (as the

[3] RFC 6120: Extensible Messaging and Presence Protocol (XMPP): Core <http://tools.ietf.org/html/rfc6120>.
[4] RFC 3966: The tel URI for Telephone Numbers <http://tools.ietf.org/html/rfc3966>.

last element) and an entity performing certification path validation (RFC 5280 [5]) MUST be prepared for this: treating a trusted root certificate in the chain as invalid (because it is self-signed) is a common implementation mistake. However, for the sake of optimization and to avoid trivial bugs, including of a root certificate in the chain is NOT RECOMMENDED.
An <x509-cert-chain/> element MAY possess 'name' attribute. The attribute contains a human readable text that uniquely represents the chain for a user, e.g. a device this certificate chain is assigned to.

Listing 1: Certificate Chain

```
<x509-cert-chain xmlns='urn:xmpp:x509:0'
                 name='Home␣Desktop'>
  <x509-cert>
    MIICQTCCAeagAwIBAgIBATAKBggqhkjOPQQDAjBFMQswCQYDVQQGEwJBVTETMBEG
    A1UECAwKU29tZS1TdGF0ZTEhMB8GA1UECgwYSW50ZXJuZXQgV2lkZ2l0cyBQdHkg
    THRkMB4XDTE5MDMwMjA2MjMyMloXDTQ2MDcxODA2MjMyMlowHzEdMBsGCSqGSIb3
    DQEJARYOdXNlckBsb2NhbGhvc3QwVjAQBgcqhkjOPQIBBgUrgQQACgNCAAQ/5HQB
    OExPNKkiYVNtPTKSItAewVK5ZyvR7bZFkGCDBOPiqOGoabRufF5xVwmHU7aFd3kL
    jjnLz6qR6SEz/rcEo4HvMIHsMAkGA1UdEwQCMAAwHQYDVR0OBBYEFLO9AFgmoQJV
    sbzcfF3gbR+PRh+fMB8GA1UdIwQYMBaAFNetyKStrn2FIcWjsD2F3HAHuhIjMAsG
    A1UdDwQEAwIF4DAdBgNVHSUEFjAUBggrBgEFBQcDAQYIKwYBBQUHAwIwcwYDVR0R
    BGwwaqAcBggrBgEFBQcIBaAQDA51c2VyQGxvY2FsaG9zdIEOdXNlckBsb2NhbGhv
    c3SGOnJlbG9hZDovLzIyMDI3MjIwMjAxODMxOTkzNDg2ODg1NzA0NzEyMTkzNDMy
    MzIyNUB4bXBwLm9yZy8wCgYIKoZIzj0EAwIDSQAwRgIhAOHsOvXmtDJroR0ggL1l
    v+Gh0t6XdNrGc3Lbd+d+LeZAAiEA1Km0ysJgYO6HBEJouPjxE0G9+Ws5SLDuc/0M
    TvzDgXQ=
  </x509-cert>
  <x509-cert>
    MIIB0DCCAXegAwIBAgIJAPYd1dvKJm6qMAoGCCqGSM49BAMCMEUxCzAJBgNVBAYT
    AkFVMRMwEQYDVQQIDApTb21lLVN0YXRlMSEwHwYDVQQKDBhJbnRlcm5ldCBXaWRn
    aXRzIFB0eSBMdGQwHhcNMTkwMzAyMDYyMzIyWhcNNDYwNzE4MDYyMzIyWjBFMQsw
    CQYDVQQGEwJBVTETMBEGA1UECAwKU29tZS1TdGF0ZTEhMB8GA1UECgwYSW50ZXJu
    ZXQgV2lkZ2l0cyBQdHkgTHRkMFYwEAYHKoZIzj0CAQYFK4EEAAoDQgAEYl6sMwVL
    bwGS0c2BLtgaYI5/sQsT+zQ63HfrVdYZ/NqvogSWieQFpkrRCEcewUMVN+5YfA/l
    XnzZRqqdz8kSAKNTMFEwHQYDVR0OBBYEFNetyKStrn2FIcWjsD2F3HAHuhIjMB8G
    A1UdIwQYMBaAFNetyKStrn2FIcWjsD2F3HAHuhIjMA8GA1UdEwEB/wQFMAMBAf8w
    CgYIKoZIzj0EAwIDRwAwRAIgUYLzzS43j55JDmUum41Ixa+p2EslGJqkhC67s1uL
    bWACIEB35gO74quIjqoogP7Hh6L+IdJ8q0yiCS1xxAqmvTjL
  </x509-cert>
</x509-cert-chain>
```

A certificate chain may be obtained and/or stored as a so called "PEM file" (formalized by RFC7468). In this case the content of this file is trivially mapped to the <x509-cert-chain/> element and vice versa. See also Storage Format.

[5] RFC 5280: Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile <http://tools.ietf.org/html/rfc5280>.

4.2 CSR Element

A certificate signing request (RFC2986) is represented as an <x509-csr/> element qualified by 'urn:xmpp:x509:0' namespace. Character data of the element MUST be a PEM encoded ASN.1 CertificateRequest structure (Section 7 of RFC7468) with encapsulation boundaries (BEGIN/END) removed. An <x509-csr/> element MUST NOT contain any child elements. The <x509-csr/> element MUST possess a 'transaction' attribute containing a random value identifying a CSR transaction. An <x509-csr/> element MAY possess a 'name' attribute: it contains a human readable text that is linked to the 'name' attribute of the <x509-cert-chain/> element being issued, e.g. a device the requested certificate chain will be assigned to. This name also MAY be stored by the CA server as a part of a user profile, e.g. to futher include it in the user's certificates listing.

Listing 2: Certificate Request

```
<x509-csr xmlns='urn:xmpp:x509:0'
          transaction='j0CAQYFK4EEAAoFpkrRCEce'
          name='My␣Phone'>
  MIIBSDCB7wIBADAfMR0wGwYJKoZIhvcNAQkBFg51c2VyQGxvY2FsaG9zdDBWMBAG
  ByqGSM49AgEGBSuBBAAKA0IABE470MZk43MAKM/HJEXbVU0cOemftIucZi+2Ug9T
  JK4s2J5AqrL84Fznv1zF5dT1Mu2QcwxxCMWEIC/a7/3UfFigcTBvBgkqhkiG9w0B
  CQ4xYjBgMAkGA1UdEwQCMAAwCwYDVR0PBAQDAgXgMB0GA1UdJQQWMBQGCCsGAQUF
  BwMBBggrBgEFBQcDAjAnBgNVHREEIDAeoBwGCCsGAQUFBwgFoBAMDnVzZXJAbG9j
  YWxob3N0MAoGCCqGSM49BAMCA0gAMEUCIQDrWmWBB5/W5+r1AXh7eQOXBHlAAZ5E
  VdF1wXTWUbc1TwIgWQNht5Xu2Z4zOkvnyfh7+fy4L8EQTH8TclPUDaUO1z8=
</x509-csr>
```

4.3 Signature Element

Given arbitrary data and an X.509 certificate with its private key, a signature of the data is created by computing a signature function from signatureAlgorithm structure of the certificate upon the data and the private key. The result is represented as <x509-signature/> element qualified by 'urn:xmpp:x509:0' namespace. Character data of the element MUST be the Base64 (RFC 4648 [6]) encoded signature. The element MUST NOT contain any child elements.

Listing 3: Signature

```
<x509-signature xmlns='urn:xmpp:x509:0'>
  MIIB8zCCAZqgAwIBAgIBATAKBggqhkjOPQQDAjBFMQswCQYDVQQGEwJBVTETMBEG
  A1UECAwKU29tZS1TdGF0ZTEhMB8GA1UECgwYSW50ZXJuZXQgV2lkZ2l0cyBQdHkg
  THRkMB4XDTE5MDMwMjE1Mzk0N1oXDTQ2MDcxODE1Mzk0N1owHzEdMBsGCSqGSIb3
  DQEJARYOdXNlckBsb2NhbGhvc3QwVjAQBgcqhkjOPQIBBgUrgQQACgNCAARQO9DG
  ZONzACjPxyRF21VNHDnpn7SLnGYvtlIPUySuLNieQKqy/OBc579cxeXU9TLtkHMM
```

[6] RFC 4648: The Base16, Base32, and Base64 Data Encodings <http://tools.ietf.org/html/rfc4648>.

```
  cQjFhCAv2u/91HxYo4GjMIGgMAkGA1UdEwQCMAAwHQYDVR0OBBYEFOf7dRPkxIxf
  z7HWUJ+NPezUiZr+MB8GA1UdIwQYMBaAFL5BMCaVAMvN6fxvJSnb0qpwHT/+MAsG
  A1UdDwQEAwIF4DAdBgNVHSUEFjAUBggrBgEFBQcDAQYIKwYBBQUHAwIwJwYDVR0R
  BCAwHqAcBggrBgEFBQcIBaAQDA51c2VyQGxvY2FsaG9zdDAKBggqhkjOPQQDAgNH
  ADBEAiBtdFGoGZ/TcxeOiQ2cau9irgEEP6kW6yEU81VRmjG6OQIgTxhH3vMvgONn
  BXXI8cAFM5iOo5JDq/TW/pV729SFVwg=
</x509-signature>
```

5 CA Server Selection

Both an XMPP server and a client are supposed to maintain a list of trusted CA certificates. This list MAY be preconfigured or dynamically obtained from a trusted source such as the one described in Section 5 of XEP-EAX, or MAY be a mix of both. In principle, a client MAY choose any CA server extracted from its own list of CA certificates to send a certificate signing request to. However, if a client also wishes to use the certificate for SASL EXTERNAL authentication with its server, it needs to pick a CA server from a mutually trusted CA certificate. For doing this, it MAY retrieve a list of CA certificates from the server and choose a CA server from a mix of the server's list and its own list. The following subsections address the latter use case. If a client has an already registered account and wishes to obtain a certificate for the use in e2e authentication only it MUST directly follow the protocol described in Certificate Issuance section.

5.1 Determining Server Support

5.1.1 Service Discovery Features

An XMPP server willing to disclose its own list of trusted CA certificates to already registered accounts MUST advertise 'urn:xmpp:x509:0' feature. In addition, if it accepts certificates issued by CAs from its list in c2s SASL EXTERNAL authentication, it MUST append an <identity/> element of category 'auth' and type 'cert'. Note that advertising either the feature or the identity alone provides very little knowledge (if any) to a client, so servers are RECOMMENDED to advertise either both of them or none.

Listing 4: Server Advertising X.509 Authentication Support

```
<iq type='get'
    to='shakespeare.lit'
    id='features'>
  <query xmlns='http://jabber.org/protocol/disco#info'/>
</iq>

<iq type='result'
    from='shakespeare.lit'
    id='features'>
```

```
  <query xmlns='http://jabber.org/protocol/disco#info'>
    <identity category='server' type='im'/>
    <identity category='auth' type='cert'/>
    ...
    <feature var='urn:xmpp:x509:0'/>
  </query>
</iq>
```

5.1.2 Stream Features

An XMPP server that supports certificate issuance during account registration MUST report that by offering the SASL EXTERNAL mechanism and by including <x509-register/> element qualified by 'urn:xmpp:x509:0' namespace in <stream:features/> element. A server MUST NOT include the feature alone and a client MUST ignore the feature if the SASL EXTERNAL mechanism is not offered. Note that the SASL EXTERNAL mechanism is only offered for TLS encrypted streams.

Listing 5: Server Advertising X.509 IBR Support

```
<stream:features>
  <mechanisms xmlns='urn:ietf:params:xml:ns:xmpp-sasl'>
    <mechanism>EXTERNAL</mechanism>
  </mechanisms>
  <x509-register xmlns='urn:xmpp:x509:0'/>
</stream:features>
```

5.2 CA List Retrieval

Once the server support is determined, a list of CA certificates MAY be retrieved from the server by sending an IQ request containing an empty <x509-ca-list/> element qualified by 'urn:xmpp:x509:0' namespace:

Listing 6: CA List Request

```
<iq type='get'
    to='shakespeare.lit'
    id='ca-list'>
  <x509-ca-list xmlns='urn:xmpp:x509:0'/>
</iq>
```

The server responds with an unordered list of <x509-cert/> elements included in an <x509-ca-list/> element:

Listing 7: CA List Response

```
<iq type='result'
    from='shakespeare.lit'
    id='ca-list'>
  <x509-ca-list xmlns='urn:xmpp:x509:0'>
    <x509-cert>
      ... PEM encoded ASN.1 Certificate structure ...
    </x509-cert>
    ...
    <x509-cert>
      ... PEM encoded ASN.1 Certificate structure ...
    </x509-cert>
  </x509-ca-list>
</iq>
```

Note that the important difference, except semantics, between <x509-cert-chain/> and <x509-ca-list/> elements is ordering of their <x509-cert/> elements.
The <x509-ca-list/> element MUST NOT be empty. Upon receiption of an empty <x509-ca-list/> element, a client SHOULD treat it as a server bug or misconfiguration and SHOULD proceed as if the server didn't support c2s SASL EXTERNAL authentication at all.

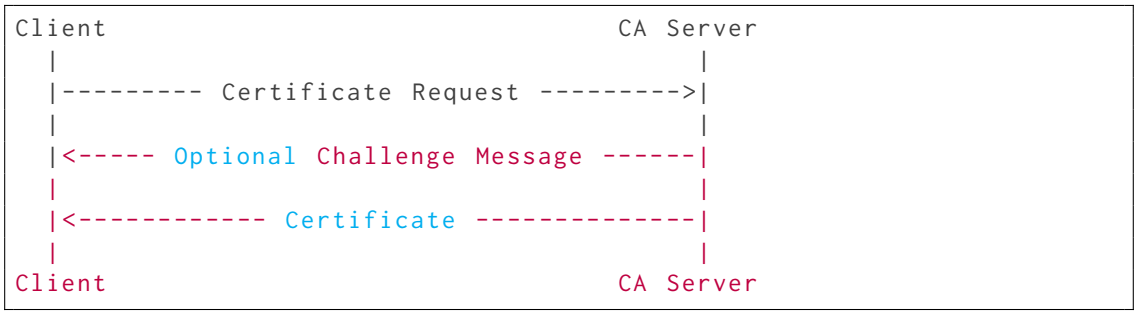
5.3 Merging CA Lists

Once a remote list of CA certificates is retrieved from the server, a client MAY merge it with its own local list and then choose an appropriate CA certificate from this mix. A client is free to use any merging algorithm. The simpliest way to do this is to take an intersection of the remote and local lists. If the result is an empty list, a client MAY apply more sofisticated algorithms, such as checking if there are intermediate CA certificates in the remote list whose are signed by some CA from the local list. In any case, prior to merging, a client MUST filter out certificates from both lists which don't contain an XmppAddr identifier (see Additional CA Certificate Requirements for the explanation). When merging is completed, a client proceeds as follows:

- If the result is not an empty list, a client picks from this list whatever CA certificate it finds appropriate, extracts an XMPP server address from an XmppAddr identifier of this certificate and sends a certificate signing request to this server as described in Certificate Issuance or X.509 IBR depending on whether the client has an already registered account or not.
- If the result is an empty list, a client SHOULD proceed as if the server didn't support c2s SASL EXTERNAL authentication at all.

6 Certificate Issuance

The certificate issuance protocol described in this section is designed to work in the presence of network, server and client failures. This in particular means that the use of Stream Management (XEP-0198) [7] is not assumed, because it's unavailable at legacy servers and during in-band registration. The certificate request is performed as a transaction consisting of an IQ request followed by an optional challenge message and then an IQ response. A transaction diagram is shown below:



To request a certificate, a client generates an ASN.1 CertificateRequest structure following the rules from CSR Requirements section. Note that a client encodes its XMPP address (or the address it wishes to register) as an XmppAddr inside extensionRequest attribute of the structure. The generated structure MUST be retained until successful completion of a transaction. If errors, disconnections or crashes are detected, the same structure MUST be reused for every new transaction (even if another CA server is picked for a retry). Failing to do so during X.509 IBR MAY result into an account deadlock for a prolonged period of time (see Preallocation). Even when a client has an already registered account, the above requirement protects it from issuing unnecessary certificates (whose number is limited by certificate authorities as outlined in XEP-EAX-CAR).

6.1 Certificate Request

Once a CA certificate is selected, a target XMPP server address is extracted from an XmppAddr identifier of this certificate. The generated ASN.1 CertificateRequest structure is then used to form an <x509-csr/> element as specified under section CSR Element. The <x509-csr/> element MUST possess a 'transaction' attribute containing a random value identifying this CSR transaction. It MAY contain a 'name' attribute. The <x509-csr/> element is then included into IQ request for transmission. The 'to' attribute of the IQ stanza MUST be set to the target CA server's address.

Listing 8: Certificate Request

```
<iq type='get'
```

[7] XEP-0198: Stream Management <https://xmpp.org/extensions/xep-0198.html>.

```
    to='ca.shakespeare.lit'
    id='csr'>
  <x509-csr xmlns='urn:xmpp:x509:0'
            transaction='4UGObuJYf7yY8ucndbmH'
            name='My Second Phone'>
    ... PEM encoded ASN.1 CertificateRequest structure ...
  </x509-csr>
</iq>
```

Upon receiption of a certificate request the CA server MUST check that the bare XMPP address in 'from' attribute matches the value of XmppAddr of the CertificateRequest structure.
The CA server then decides to either issue a certificate, challenge a client or generate an error. If it has an already issued certificate for this CSR, it MUST respond with the certificate without challenging a client. If it has received another request with the same CSR during a challenge procedure, it MUST abort the running procedure, destroy an internal transaction state and process the request within a new transaction.

6.2 Certificate Challenge

The CA server MAY challenge a certificate request. It MAY do so for many reasons, for example, it MAY want to identify a human user in order to prevent massive creation of certificates by a single person. Another possible case is when the CA server detected some errors (e.g. too many issued certificates) and wants the user to perform some actions in order to resolve the problem. To do so the CA server responds with an <x509-challenge/> element. The element MUST possess 'uri' attribute containing an URI. It also MUST possess a 'transaction' attribute with the value copied from a 'transaction' attribute of the original <x509-csr/> element. The <x509-challenge/> element MUST contain exactly one <x509-signature/> element carring a signature computed upon concatenation of the values from 'transaction' and 'uri' attributes using the CA certificate (see Signature Element). The challenge element is then included into message stanza for transmission. The value of 'to' attribute of the message MUST be copied from the value of 'from' attribute of the IQ request.
In this version of the protocol the URI MUST be an HTTPS URL. A client is supposed to open this URL in a web browser for a user to process the challenge. The content of the URL is opaque to a human user and thus SHOULD NOT be rendered in a client's user interface.

Listing 9: CA Server Sends Challenge

```
<message type='normal'
         from='ca.shakespeare.lit'
         to='romeo@montague.lit/orchard'>
  <x509-challenge xmlns='urn:xmpp:x509:0'
                  transaction='4UGObuJYf7yY8ucndbmH'
                  uri='https://ca.shakespeare.lit/csr/cOemft/8EQTH8'>
    <x509-signature>
      ... Base64 encoded signature ...
```

```
    </x509-signature>
  </x509-challenge>
</message>
```

In the above example the signature is computed upon '4UGObu-JYf7yY8ucndbmHhttps://ca.shakespeare.lit/csr/cOemft/8EQTH8'.
Upon receiption of a challenge a client MUST follow these rules:

1. A client checks that the value of 'from' attribute matches the CA server address.
2. A client checks that the value of 'transaction' attribute matches the identifier of the running transaction.
3. A client verifies the signature using the public key of the CA certificate.

If all the checks have passed, a client spawns an URI handler and waits for the certificate response. Otherwise, a client MAY ignore the message or MAY reply with a corresponding stanza error.

6.3 Certificate Response

When the CA server successfully issued a certificate it MUST respond with an IQ result containing the full certificate chain represented as an <x509-cert-chain/> containing the issued certificate represented as an <x509-cert/> element. Note that according to the defined ordering, this certificate MUST always be the first element in the chain. The server MUST NOT respond with an empty <x509-cert-chain/> element. If the original <x509-csr/> element has possessed a 'name' attribute, its value MUST be copied to 'name' attribute of <x509-cert-chain/> element.

Listing 10: Certificate Response

```
<iq type='result'
    from='ca.shakespeare.lit'
    to='romeo@montague.lit/orchard'
    id='csr'>
  <x509-cert-chain xmlns='urn:xmpp:x509:0'
                   name='My Second Phone'>
    <x509-cert>
      ... PEM encoded ASN.1 Certificate structure ...
    </x509-cert>
    ...
    <x509-cert>
      ... PEM encoded ASN.1 Certificate structure ...
    </x509-cert>
  </x509-cert-chain>
</iq>
```

Upon receiption of a response matching the request a client proceeds as follows:

- It MUST destroy internal IQ and transaction states.
- It MUST check that the response has arrived from the requested CA server.
- It MUST check that the response carries <x509-cert-chain/> element which contains at least one <x509-cert/> element.
- It MUST check that all certificates in the chain are valid according to the rules outlined in XEP-EAX.
- It MUST perform certification path validation (RFC 5280 [8]) to check that the certificate chain is indeed signed by the requested CA.

If all the checks succeed, the transaction is considered to be completed. At this point a client MAY release the ASN.1 CertificateRequest structure.
If either of the checks fails, a client MUST behave as if it received an error response with a permanent condition (see Certificate Request Error section).

6.4 Certificate Request Error

If the CA server refuses to issue a certificate it MUST generate a corresponding stanza error. In this case <error/> element MUST possess 'by' attribute with the value of the CA server's address. If the error is generated due to challenge failure, <error/> element MUST contain <x509-challenge-failed/> element qualified by 'urn:xmpp:x509:0' namespace.

Listing 11: CA Server Error

```
<iq type='error'
    from='ca.shakespeare.lit'
    to='romeo@montague.lit/orchard'
    id='csr'>
  <error type='auth'
         by='ca.shakespeare.lit'>
    <forbidden xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
    <x509-challenge-failed xmlns='urn:xmpp:x509:0'/>
  </error>
</iq>
```

When a client receives an error response, it considers the transaction as failed and MUST destroy internal IQ and transaction states.
In the case of a temporary failure, a client MAY repeat the request to the same CA server. In

[8] RFC 5280: Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile <http://tools.ietf.org/html/rfc5280>.

the case of a permanent failure, a client MUST choose another CA server if it has decided to retry. In both cases, the 'name' attribute and character data of <x509-csr/> element of the new request MUST be the same. However, a client MUST generate new values for 'transaction' attribute of <x509-csr/> element and for 'id' attribute of the IQ stanza.
A client MUST NOT process an URI from <gone/> and <redirect/> error conditions and MUST treat these errors as permanent failures.

6.5 Certificate Request Timeout

If a client detects a request timeout, i.e. neither challenge nor response have arrived in the assumed time, it MUST behave as if it received an error response with a temporary condition (see Certificate Request Error section).

7 Certificate Revocation

A registered client at any time MAY revoke its certificate. To accomplish this it MUST create an IQ stanza containing <x509-revoke/> element qualified by 'urn:xmpp:x509:0' namespace. The element MUST contain:

- The certificate being revoked represented as <x509-cert/> element.
- An <x509-signature/> element computed upon the ASN.1 DER encoded tbsCertificate structure of the certificate as described in Signature Element.

There MUST be exactly one <x509-cert/> element and exactly one <x509-signature/> element. The IQ stanza MUST be sent to the CA server that has issued the certificate, i.e. extracted from XmppAddr of the corresponding CA certificate.

Listing 12: Certificate Revocation Request

```
<iq type='set'
    to='ca.shakespeare.lit'
    id='revoke'>
  <x509-revoke xmlns='urn:xmpp:x509:0'>
    <x509-cert>
      ... PEM encoded ASN.1 Certificate structure ...
    </x509-cert>
    <x509-signature>
      ... Base64 encoded signature ...
    </x509-signature>
  </x509-revoke>
</iq>
```

The CA server MUST verify the signature using the public key of the certificate, MUST append the certificate's serial to the corresponding CRL and, in the case of success, MUST respond with an empty IQ result:

Listing 13: Certificate Revocation Request

```
<iq type='result'
    from='ca.shakespeare.lit'
    to='romeo@montague.lit/orchard'
    id='revoke'>
```

In the case of failure, the CA server MUST respond with a corresponding stanza error. Depending on the error type, a client MAY either repeat the request or give up.
Upon successful revocation, a client MAY retract the corresponding published item (see Certificates Discovery section).
A client SHOULD revoke all its certificates prior to cancelling the account registration (Section 3.2 of In-Band Registration (XEP-0077) [9]).

8 X.509 IBR

The protocol supports certificate issuance during account registration. Thus the requested certificate can be also used in SASL EXTERNAL authentication with the server where the account is being registered. The rationale for this is at least twofold:

- Consequent creation of an account and then a certificate (for e2e authentication, as described in the sections above) creates burden for users because they typically need to pass a challenge twice: firstly at the local server, and secondly at some CA server. Combining these two steps into one improves user experience in this regard.
- Managing freely available account registration at public servers is not a simple task for operators of these servers. Failing to manage it correctly leads to uncontrolled massive creation of accounts used for SPAM propagation, DoS attacks, etc. and degrades the "Jabber" network as a whole. A server operator may want to delegate a registration and verification procedure to a trusted CA, which is believed to be very good at this task. The operator doesn't lose the userbase in this case: user data and profiles are still located at the operator's server.

The registration protocol described in this section is called X.509 In-Band Registration (X.509 IBR).
A server that supports X.509 IBR MUST report that as specified under section Stream Features.

[9] XEP-0077: In-Band Registration <https://xmpp.org/extensions/xep-0077.html>.

8.1 IBR Client Rules

Once a client has learnt server support from the stream features, it MUST retrieve a list of CA certificates from the server as specified under section CA List Retrieval. The server MUST allow unregistered clients to retrieve this list if it has reported X.509 IBR support. Then a client merges the server's list with its local list as described in section Merging CA Lists and choose a CA certificate from the mix. A client then follows the procedure described in Certificate Issuance section.
Once a certificate is obtained, it is RECOMMENDED to perform SASL EXTERNAL authentication with the server as soon as possible in order for the server to mark the account as registered (see Registration Mark).

8.2 IBR Server Rules

Upon receiption of a certificate request, the server checks that:

1. The IQ stanza possesses 'to' attribute.
2. The ASN.1 CertificateRequest structure inside <x509-csr/> element contains an XMPP address (that the client requests to register, see CSR Requirements).
3. The server is responsible for the domain of the XMPP address.
4. An XMPP address in 'to' attribute is a trusted CA server and the server allows this CA to issue certificates for the users of the domain.
5. There is no already registered or preallocated (see Preallocation) account matching the XMPP address being registered. If the account is preallocated, an ASN.1 CertificateRequest structure from the preallocation MUST match the one from the request.

If either of these checks fails, the server MUST generate a corresponding stanza error. If the error is generated because the account is already registered or preallocated, the error condition MUST be <conflict/>.
If all the checks succeed, the server preallocates an account and routes the request as described below.

8.2.1 Preallocation

In order to prevent registration of the same account by different human users, the server MUST temporary preallocate an account upon receiption of a certificate request and later MUST mark it as permanently registered and/or release the preallocation. The server preallocates an account by storing an association of the XMPP address with the ASN.1 CertificateRequest structure from the request. The server MUST keep an account preallocated for a period long enough for a client to complete the issuance and authentication. The server

MUST NOT allow registration of preallocated accounts using different methods (e.g. In-Band Registration (XEP-0077) [10]).

8.2.2 Routing

If the server has accepted the request it MUST set 'from' attribute of the IQ stanza with the value of the XMPP address being registered and MUST forward the request towards the CA server. Since the client doesn't yet have an account at the server, the standard routing rules (Section 8.5 of RFC 6121 [11]) cannot be used to route back CA responses. In order to find the corresponding client's stream statelessly, the server MAY append a resource part to the XMPP address in 'from' attribute. The resource MAY contain arbitrary data needed by the server to detect the client's stream location. Note that the data MUST NOT be more than 1023 octets in length (Section 3.4 of RFC 7622 [12]). Prior to forwarding of a CA response to a client, the server MAY remove 'to' attribute from the response, however, this is not strictly speaking needed since a client is supposed not to check its value (see "Implementation Note" of Section 8.1.1.1 of RFC 6120 [13]).

8.2.3 Registration Mark

In general, there is no way for the server to know whether certificate issuance was successful or not: even though the server is able to inspect CA responses, their delivery to a client is not guaranteed. So the only reliable way to mark an account as registered is at the first successful SASL EXTERNAL authentication. When the account is finally marked, the server MUST release the preallocation.

9 Certificates Discovery

A client MAY use local PEP storage (Personal Eventing Protocol (XEP-0163) [14]) in order to publish its certificates so other peers can discover them. It MUST do this by including each certificate chain represented as <x509-cert-chain/> element in a separate pubsub <item/> element and publish each of the items to 'urn:xmpp:x509:0' node. Note well: a single item corresponds to a single certificate chain.

[10] XEP-0077: In-Band Registration <https://xmpp.org/extensions/xep-0077.html>.
[11] RFC 6121: Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence <http://tools.ietf.org/html/rfc6121>.
[12] RFC 7622: Extensible Messaging and Presence Protocol (XMPP): Address Format <http://tools.ietf.org/html/rfc7622>.
[13] RFC 6120: Extensible Messaging and Presence Protocol (XMPP): Core <http://tools.ietf.org/html/rfc6120>.
[14] XEP-0163: Personal Eventing Protocol <https://xmpp.org/extensions/xep-0163.html>.

Listing 14: Publishing Certificate Chain

```
<iq type='set' id='announce1'>
  <pubsub xmlns='http://jabber.org/protocol/pubsub'>
    <publish node='urn:xmpp:x509:0'>
      <item id='304402206242a7b554e2f1a1bd790758'>
        <x509-cert-chain xmlns='urn:xmpp:x509:0'
                         name='Laptop'>
          <x509-cert>
            ... PEM encoded ASN.1 Certificate structure ...
          </x509-cert>
          ...
          <x509-cert>
            ... PEM encoded ASN.1 Certificate structure ...
          </x509-cert>
        </x509-cert-chain>
      </item>
    </publish>
  </pubsub>
</iq>
```

To uniquely identify a certificate chain within the node, 'id' attribute of the <item/> element MUST contain first 16 octets from a signatureValue (Section 4.1.1.3 of RFC 5280 [15]) of the first certificate in the chain, represented in lowercased hexadecimal encoding. For instance, a value of 'id' attribute from the example above corresponds to the signature from the example below.

Listing 15: Certificate Signature

```
30:44:02:20:62:42:a7:b5:54:e2:f1:a1:bd:79:07:58:f7:53:
 22:ba:0a:a5:4a:3f:d8:51:22:38:6c:59:3a:fd:77:d6:07:a4:
 02:20:6c:ac:34:ac:71:f5:4b:ba:58:9f:34:f4:3a:6a:64:31:
 06:72:5e:e9:e6:ea:9d:99:31:e6:a3:08:e6:67:57:c1
```

10 Implementation Notes

10.1 Storage Format

For compatibility with other programs, a client SHOULD store an obtained certificate chain in PEM format (RFC7468) written to a file with ".pem" extension. Alternatively, a client MAY store it in other formats, but SHOULD provide a procedure for exporting in PEM format.

[15] RFC 5280: Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile <http://tools.ietf.org/html/rfc5280>.

10.2 SASL Mechanism Transitioning

When an already registered client detects server support e.g. after software upgrade, it may ask the user to request a certificate and transition to SASL EXTERNAL authentication (although the exact question may not contain these technical details). In order to avoid confusion, a client should check if it has a mutually trusted CA certificate with the server as specified under CA List Retrieval and Merging CA Lists sections before asking for transitioning.

10.3 Mobile OS Considerations

In order to optimize battery consumption some mobile operating systems have very strong limitations for background processes. This may become a problem for a client running a challenge procedure: the procedure is typically interactive and thus the client process may be preempted and killed. A possible workaround is to store the request state in durable storage and, when the challenge is passed and the client process is restarted, consult the storage and repeat the request if needed. Since CA servers are prepared to resend responses for already issued certificates without challenging, a client doesn't need to disturb a human user again in order to receive the certificate.

11 Security Considerations

TODO

12 IANA Considerations

None required.

13 XMPP Registrar Considerations

The urn:xmpp:x509:0 namespace needs to be registered.

14 XML Schema

TODO