Client Application Layer Encryption
draft-deutch-lamps-client-app-encrypt-00

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Status of This Memo

This document specifies an Experimental protocol for the Internet community, and requests discussion and suggestions for improvements. Please refer to the current edition of the "Internet Official Protocol Standards" (STD 1) for the standardization state and status of this protocol. Distribution of this memo is unlimited.

Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

Abstract

The protocol for Client Application Layer Encryption offers organizations a method of securely providing users data with very few authentication steps. This protocol makes use of X.509 public key infrastructure and SHOULD NOT be implemented without transport layer security. The protocol described below helps to ensure that response messages may only be read by the intended recipient.
# Table Of Contents

Abstract ......................................................... 3
1. Introduction .................................................. 3
   1.1 Terminology ................................................. 3
   1.2 Abbreviations .............................................. 3
   1.3 Roles ....................................................... 3
   1.4 Goals ....................................................... 4
   1.5 Motivation ................................................ 4
   1.6 Strengths and Weaknesses ............................... 4
2. Security Considerations ...................................... 5
3. IANA Considerations .......................................... 5
4. Communication Patterns ....................................... 5
   4.1 Initiation ................................................ 5
   4.2 Standard Request ......................................... 5
   4.3 whoami Request ............................................ 7
   4.4 Server Revocation ......................................... 8
References ....................................................... 8
   Normative ..................................................... 8
   Informative .................................................. 8
Appendix A: UML Flow Diagrams ................................. 9
   A.1 Initiation ................................................ 9
   A.2 Standard Request ......................................... 10
   A.3 whoami Request ........................................... 11
Appendix B: Example Requests and Responses .................. 12
   B.1 Initiation ................................................ 12
   B.2 Standard Request ......................................... 15
   B.3 whoami Request ........................................... 20
Author’s Address ................................................ 22
Full Copyright Statement ......................................... 22
Intellectual Property Statement ................................. 22
1. Introduction

This protocol offers a way to reduce the number of network communications that must occur for a system to have confidence in the identity of the requester and reduces the risk in the case of impersonation. This was designed with application programming interfaces in mind.

1.1 Terminology

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

1.2 Abbreviations

CN: Common Name [RFC4514]
CSR: certificate signing request [RFC5280]
DN: Distinguished Name [RFC4514]
GUID: Globally Unique IDentifier [RFC4122]
IaaS: infrastructure as a Service
OU: Organizational Unit [RFC4514]
PaaS: Platform as a Service
SAN: subject alternative name [RFC4514]
SaaS: Software as a Service
TLS: transport layer security [RFC5246]

1.3 Roles

resource owner: The party with rights to the data.

resource server: The object housing the data.

authorization server: The server that fulfills certificate signing requests and catalogs them for validation. All calls to this device should be over TLS with mutual certificate exchange [RFC5246].

client: The object requesting the data.

edge device: The object open to anonymous traffic, terminates TLS [RFC5246], brokers authentication, performs authorization, then forwards data.

origination server: The object that performs processing of the request that results in the response.
1.4 Goals

Minimize exposure of client credentials and data. A client can be authorized and returned a token or other sensitive information with confidence that it cannot be intercepted, even by an internal bad actor. To do this the authorization server must either be a signing authority or have permission to submit certificate signing requests to a signing authority [RFC5280]. The client certificate properties may act as a vehicle for personally identifying information to be passed to the origination server. The private key SHOULD NOT be exported from the client device and therefore the CSR may contain device properties.

1.5 Motivation

Organizations have increased the number of individuals with access to subvert trusted systems with the increase in subcontracting information services i.e. SaaS, PaaS, IaaS, etc; as well as contract workers.

When users’ information is unencrypted is it vulnerable to exploitation. By reducing the occurrences of client data being unencrypted we reduce the opportunity for attack.

1.6 Strengths and Weaknesses

This provides a mechanism for user credentials that may be valid for an undefined period of time. Made possible because the credentials themselves, the private key [RFC5280], never exists outside the users’ (resource owners’) device.

The true proof of identity is in the ability of the client to read the response message. Which makes this mechanism ideal for GET requests but unsuitable for POST, PUT, or DELETE unless accompanied by a secondary authentication mechanism.

If an attacker captured the CSR then they would be in a position to build a response the client would accept, however the attacker would also have to impersonate the edge device in order to impersonate the authorization and origination servers. Conversely, if an attacker impersonates the edge device without the CSR on file then any response would appear malformed.

Because these certificates are not used in TLS negotiation the client is not required to share it at the device layer. This allows the credentials to be owned exclusively by the application within the clients’ device, reducing the opportunity for another application running on the same device to steal the private key or impersonate the organization’s application to the authorization server by reading their response.

To mitigate risk of attacks some error messages must simulate successful responses reducing feedback to legitimate consumers with malformed requests.
2. Security Considerations

This document defines a protocol for authenticating and authorizing users for access to protected data and the secure delivery of responses.

3. IANA Considerations

No IANA considerations

4. Communication Patterns

The following sections describe the various transactions that make up this protocol.

4.1 Initiation

For this flow the client is also the resource owner, and the authorization server is also both resource server and origination server.

The client must use a method acceptable to the edge device to prove their identity [RFC6749] [RFC7617], preferably initial registration. At the conclusion of this proving the client should have packaged their CSR and sent it to the edge device.

The edge device shall then forward the identity information with the CSR and the cipher used for the TLS to the authorization server.

The authorization server shall store the CSR in association with the user identity and return a response of the GUID of the CSR record encrypted by a certificate generated from the CSR using the cipher negotiated between the client and the edge device. This cipher is used to ensure it is one the client knows, to be sure it is one that the resource server knows; the edge device and resource servers should be configured to maintain the same list (remember in this flow the resource server is also the authorization server).

The edge device shall then return the encrypted response to the client.

The client must decrypt the response with their private key [RFC5280] used to generate the CSR and store the GUID and key for future use.
4.2 Standard Request

For this flow the client is also the resource owner. These credentials are sufficient if this Request is a read only operation or a create that produces data that is only usable after the client has read the response (proving that they are the resource owner), such as token generation where the token is returned in the response payload body or a request to a processing queue which must be followed by an execution call using the queue identifier from the response. These credentials should be supplemented by a secondary mechanism if this request is expected to result in any data changes.

The client shall send their GUID with the request to the edge device.

The edge device should forward the GUID to the authorization server in the form of a validation request. The edge device may forward the request to the origination server without performing this step, which would be bad practice because it increases the opportunity for capture, message replay, and in that case the origination server would need to call the authorization server increasing its client list and therefore attack surface.

The authorization server shall reply to the validation request with a client certificate generated by the CSR associated with the GUID. The certificate should only be valid long enough to fulfill the request.

If the edge device receives a response from the validation call to the authorization server that is not a client certificate then the edge device should return an object large enough to be mistaken for an encrypted response to the untrusted client. If authentication is successful then the edge device should then forward the client request with the certificate and the negotiated cipher to the origination server without the GUID.

If an internal bad actor captured a request with the client’s certificate or GUID and used it to send a request then they would be unable to read the response. Additionally, the certificate should have an extremely short validity period in which this request would be valid.

The origination server should validate the certificate by issuer, subject, and expiration. No CRL is required as the certificate validity should only ever be long enough for one request. This enables the origination server to perform fine grained authentication with high confidence without any external calls. The origination server may be or make calls to the resource server(s) providing the certificate and not the cipher, aggregating data as required. The identity of the certificate is taken from the SAN if present; wherein the CN is the resource owner, the DC is the organization of the servers, and any OUs represent allowed scope(s).
The absence of the cipher informs any resource server(s) that their response should not be encrypted by the user’s certificate. This request should be over TLS and should use mutual certificate exchange [RFC5246] because the client’s certificate in this request is not for authentication, it is present as a form of query. These requests are from the origination server to the resource server(s) as evidenced by the origination server’s need to read the response.

The origination server shall encrypt the response intended for the client using the client’s certificate and the cipher provided by the edge device ensuring that only the client is able to decrypt it. The origination server then returns this response to the edge device.

The edge device shall forward the response to the client.

The client shall use their private key to decrypt the response.

If the request is captured between the client and the edge device then a message replay is possible, however the response could only be read by the real client. If a request is captured between the edge device and the origination server then a message replay is possible only until the certificate expires and again, could only be read by the real client. The flow should use TLS throughout to prevent the request from being read between hops.

### 4.3 whoami Request

For this flow the client is also the resource owner and the authorization server is also both resource server and origination server.

The client makes a request to the edge device using their GUID.

The edge device receives the request and forwards the GUID to the authorization server with the negotiated cipher.

The authorization server generates a certificate for the client that expires immediately, encrypts the certificate using itself and the specified cipher, and then returns this as the response to the edge device. If the GUID is not known then an response consisting of a random salt large enough to be reasonably mistaken for an encrypted payload should be returned to the edge device with a HTTP 200 code [RFC7231], this is intended to prevent a dictionary attack from mapping out valid GUIDs.

The edge device forwards the response to the client.

The resource owner must then decrypt the response to read it.
4.4 Server Revocation

In the event that a set of credentials are compromised then the authorization server may be required to revoke them. The resource owner may be required to perform a new initiation to regain access to their account.

References

Normative


Informative


[WSD]    WebSequenceDiagrams software is provided by Hanov Solutions Inc., of Waterloo, Ontario, Canada.  
<https://www.websequencediagrams.com>
Appendix A: UML Flow Diagrams

Each section of this appendix corresponds to the same numbered subsection of this document under section 4. The text between the section heading and the flow graphic represents the flow in sudo-code [WSD]. The diagrams have been simplified from the sudo-code in order to fit this document format.

A.1 Initiation

title Initiation

note over client:
    generate key
    generate CSR
end note
client-->edge device: Registration+CSR
edge device-->+authorization server: ID+CSR+cipher
note over authorization server:
    store CSR
    generate GUID
    encrypt GUID
end note
authorization server-->-edge device: encrypted GUID
edge device-->+client: encrypted GUID
note over client:
    decrypt response
    store GUID
end note

[client]-------------------------------------------------------[client]
    ||                            ^
    registration :            GUID
    and csr                        encrypted
    ||                            :
    \                           :[/edge device]
    ||                            ^
    ID :                           :
    CSR GUID
    cipher    encrypted
    ||                            :
    \                           :[/authorization server]

Deutsch Client Application Layer Encryption [Page 9]
A.2 Standard Request

title Standard Request

client->edge device: request+GUID
edge device->authorization server: GUID
note over authorization server: generate certificate from GUID CSR
authorization server-->edge device: certificate
edge device->origination server: request+certificate+cipher
note over origination server:
certificate validation
authorization
end note
opt if origination server is not resource server
origination server->resource server: server request+certificate
resource server-->origination server: server response
end
note over origination server: encrypt response
origination server->edge device: encrypted response
edge device-->client: encrypted response
note over client: decrypt response

[client]--------------------------------------------------------[client]
  ^
 Request
 GUID encrypted
\`
[end device]--------------------------------------------------------[edge device]
  ^
 GUID : encrypted
 \'
 certificate : 
[end authorization server]----------------------------------[authorization server]
  ^
 request : 
 certificate encrypted
 cipher response
 \`
[end origination server]-------------------------------------------[origination server]
  ^
 server : 
 request+ : 
 certificate : 
    encrypted 
     response 
\`
[end resource server]-------------------------------------------[resource server]
A.3 whoami Request

title whoami Request

client->edge device: GUID
edge device->authorization server: GUID+cipher
note over authorization server:
    generate certificate
    encrypt response
end note
authorization server-->edge device: encrypted response
edge device-->client: encrypted response
note over client: decrypt response

[client]-------------------------------------------------------[client]
  ||   ^
  GUID : 
  || encrypted
  || response
  /:
[edge device]-------------------------------------------------------[edge device]
  ||   ^
  GUID : 
  cipher encrypted
  || response
  /:
[authorization server]---------------------------[authorization server]
  ^
  _generate and_
  encrypt cert

Deutsch Client Application Layer Encryption [Page 11]
Appendix B: Example Requests and Responses

Each section of this appendix corresponds to the same numbered sub-section of this document under section 4. These examples contain elements which fulfill the requirements described above and may be met by other means.

B.1 Initiation

The below private key is used to generate the below examples and can be used to execute the client decryption commands:

```
-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: DES-EDE3-CBC,7F58E7878FA4D4A8

98MDLxjgMW5W712ADD1CG2VeAMG/vxmIgpF+F2japv831iSh4WC5LJFPxFKXp+nQ5
L74+xDt1fRSKuFpnBqPnkol11ZrMqK+WT83FSxA0wm5rvfsa9ECSmJMP24z5roAd
+ipyn3v47Vmlu6gjk1wmgj2t2LknrsvXh6CGKc2Aaj3xQie0KyzB6/m2hMc6A5z
nVwhwJi4Fc4J/Zs9+J/4KUFbSdo6b5eS5e4ixeWWTGfzVDjOmRaob0l6x1diGDH3
1khUSpJut0jnisWUvKjUos9AvFt8QISeodiQr5AMCrYGVY0N5BN35hv/mqJHT3g
AH25psCwaTIP44qYU6CQ5RkRxOE2CmJiPvsPj9uox8zosi31CwCUZUIkpwsl7o
DfnucGNPPSa1Jennos5CqY8ySE0BXN/msoXLfcwiA842V5G57TDaysIcHnyG9
J56S2D2QjEO7s9LbUykGutlBOIloxWxyhMK2ku/DjB01QTncUaibWUJ5Y3BwluVVe
8GL3HaorRG+aos8E5y/OvcaE7mTM4iqXKZcRELvl1G+HqCsa1lVgfaLmNdq+8l
qTEHPGFPvo5BQRLCaVv21hd38nBmHFQFHyB5x6jQcAhBuhf8Ns1LJ2740HyxtKd
h/bgrQdQukkAhCRRGFb31iOwM/OUwv/v/w1N1Kp6Z9jhTICQ/RF/86CGAYfdq
1Fv/wsDgkmkmaQWIYqc9drH1X1L2htMSJ4aeJufjFVhY06Ph2ie10UpkFq5i518x
0kfoiU03oAEPKo2zVhH1RN+/x89JccqcmOF0KdXckUaRPhzJuJwvQDNArKPSNKd
6yjD4nB9Jbjnpb1wRr/yj70CA33ilRangFuUFq6gsZaj54o1i8MO2a0x0GdrM8o
eseWFe7od+l6/zbH4Whe0vMDj0UXOqGkCFW1mhP0Yd13LuLddaeUyeJDjp44t2y
pb8FdnfA5IS7xMyXz2XIBZjQctqGrWvPR9o/xloZiUNOZmn6wzmZGznZddVex
s9nM2VopdrPe8n4bxuTRKPyGvATDddY8czq2h8/STGx55PmCvRa111WrlNls8844mq
Q5sv1sw+GnDiA25975D5pq5Ed1Bzy5g610uUwEdQeIOmRswHRBtQz3xmX+
1Hg35WKhTRJpK6ckGCwWcrBk4mKsUq4gqjQbZLNbA92vXb5Cu0jyFMGrz80o
phfixeJWA8w185UPQ9sf8m/S6n5Pnd04Xuhz2J7u/+Um2Xqrrfhf1+mgd1+T5a
PPvKf2veCkVvIpeFw1f2WQZ2GluJTHdb1j71G2QrKrLY7tyO5+qGQsB8y8nwnm
XYgJqjXnzn1r5u055QpXhkJ1B4+x4+0k51vVJN0n1L40+rw/UB8tNFeV8c9cBtPm
B4+zikt+xD2Alu+UwCBOAIANE/xCN950tC60FMiv8j/qzq9+c7DnXnqPz8rvCQ7f
DhH8AlkMxqJOJ9Ifu2MmAMUMqMl35q8oGT8RYcWa/e1JExMxMqOpSY7cwhGUympM
/8ZrWLPo7CynTvK4LaMBzhuZ6m3p348dNR8qmXIkm7rcqXyPv+BvWtp/T2pXVe
-----END RSA PRIVATE KEY-----
```
B.1.1 Client Registration Request

The client generates a private key:

```
openssl genrsa -des3 -out privkey.key 2048
```

Then generate a certificate from the key to designate the expected properties:

```
openssl req -key privkey.key -out client.crt -new -x509
```

Then generates a CSR from the key and certificate:

```
openssl x509 -x509toreq -in client.crt -out client.req -sha256 -signkey privkey.key
```

The request from the client to the Edge Device:

```
POST /registration HTTP/1.1
Host: server.example.com
Content-Type: application/pkcs10

-----BEGIN CERTIFICATE REQUEST-----
MIICrzCCAZcCAQAwajELMAkGA1UEBhMCVVMxCzAJBgNVBAgMAkNBMRUwEwYDVQQH
DAxExZzhwdWxEIENpdHkxHDAAQgNVBAoME0RlZmF1bHQgQ29tcGFueSBmdQxGTAX
BgNVBAMMEEJlbmphbWluIERldXRzY2gwggiMa0GA1UEBhMCMjAxGTAXBgNVBAsM
E0RlZmF1bHQgQ29tcGFueSBmdQxGTAXBgNVBAMMBGB2NlY3JvcGUgU2VsdGFy
-----END CERTIFICATE REQUEST-----
```
B.1.2 Edge Registration Request

The request forwarded to the authorization server with the cipher:

```
POST /registration HTTP/1.1
Host: server.example.com
Content-Type: application/pkcs10
Cipher: ECDHE-RSA-AES256-SHA

-----BEGIN CERTIFICATE REQUEST-----
MIICrzCCA2cCAQAwELMAkGA1UEBhMCVVMxCSAJBgNVBAgMAkNBMRUwEwYDVQQH
DAxZ2dhW0IENpdHxxHDAaBgNVBAoME0RlZmF1bHQgQ29tcGFueSBMdGQxGTAx
BgNVBAMMMEEJlbmphbWluIERldXRzY2wgggEiMA0GCSqGSIb3DQEBAQUAA4IBDwaw
ggEKAoIBAQCPJMQQY1gANVIIreVqV1lp2mw1ASUixRJp4SGPHpsaNJfvHcZBW1
zBvVf9600sC1NasU69QIPeuJAYELdOXYoX2J+5DSN/g3X8p3CMXMd7xpArpx
q6uxevEtMP1kx4XSC7nJspEFOl1FhwxU0Qx5DL5guATafmtRvbbWBNBMRa8
55HCKlCQxx4i0/DMREm0P/7fYRfuwUYWf3KfkuCnwhbmxvFI0PDQfw/q+UhpobV
arxZS++S6j1MdaKh7HLOpHdrrLr8uaN10B3weF6C2EGDxJ2B0v3xEdmxVL7Ch6
GBA7y3amfydZ5F0K13d1gWWYV6m/6E5tAgMBAAAGgADANBkgkchiG9wDBAQsFAACOC
AQEAUnKJBIenLIMF17J3GWl49kK6PNkuc7HRex0Tmso4C7fN7RxIo+6uZrgF0g
met55u+5uepVYGNmph2tg0hYUnUA5Z14fzJeNmX1jBafBUQJ4Yhi6R55yCpzU1CuC
wJSyWWujPpUvf3RsK5kk7LbHKqJntZ8+s3mbUtVVb80VsaWv0zDZs6K+2hXnX
YnP40qZ0mhraYDdDuVVBYJNapzWjNHTXJjhgR0u+xhnX8PugIoUItan/SmFkt/6pvIjgOBX1nBQ04B8S+1+6Er9CrShEX6UCALkd+9BhHeDDi7Zara1YshpOEFr9W
qMHUCVVDTCYzomsqGQ/0/wKF8Hg==
-----END CERTIFICATE REQUEST-----
```
B.1.3 Registration Response

After generating the GUID that identifies the record it shall be encoded using the client certificate:

```
openssl smime -encrypt -binary -aes-256-cbc -in response.txt client.crt
```

Resulting in the encrypted response [RFC5751]:

```
HTTP/1.1 200 OK
Content-Type: text/plain;charset=UTF-8
MIME-Version: 1.0
Content-Disposition: attachment; filename="smime.p7m"
Content-Type: application/x-pkcs7-mime; smime-type=enveloped-data; name="smime.p7m"
Content-Transfer-Encoding: base64
```

B.2 Standard Request

A request to some other services with this added protection.

B.2.1 Standard Client Request

The request to some service:

```
GET /resource HTTP/1.1
Host: server.example.com
CALE-GUID: bec6dc7e-6562-4b1c-b308-6c352e6f8404
```

The client decrypts the response:

```
openssl smime -decrypt -binary -aes-256-cbc -in response.enc -out response.txt -inkey privkey.key
```

Enter pass phrase for privkey.key: password

```
bec6dc7e-6562-4b1c-b308-6c352e6f8404
```
B.2.2 Edge Validation Request

The authentication request to the authorization server:

GET /validate HTTP/1.1
Host: authority.example.com
CALE-GUID: bec6dc7e-6562-4b1c-b308-6c352e6f8404

B.2.3a Authorization Validation Response

Create the signed certificate with minimally applicable validity:

openssl ca -config openssl.cnf -startdate 180731190800Z -enddate 180731190810Z
-keyfile ca.key -cert ca.crt -in client.req -out ./client.crt

A successful response from the authorization server:

HTTP/1.1 200 OK
Content-Type: application/x509

-----BEGIN CERTIFICATE-----
MIIEfDCCA2SgAwIBAgIRAOaxLLnaTZdrituxMDU+EwowDQYJKoZIhvcNAQELBQAw
czELMAkGA1UEBhMCVVMxGjBEQUFwCgYIKoZIhvcNAQELBQADggEBMAwGA1UdEwMC
cwGQYDVQQKEwJVUzELMAkGA1UEBhMCOkEwCwYIKoZIhvcNAQELBQADggEBMAwGA1U
dEQyBCA4AwDQYJKoZIhvcNAQELBQADggEBMAwGA1UdEwMCBAQAwggGcBAwggGcB
-----END CERTIFICATE-----

B.2.3b authentication Validation Error

An unsuccessful response from the authorization server:

HTTP/1.1 403 Forbidden
B.2.3c Edge Device Erroneous Response

A successful appearing response designed to prevent dictionary attack from mapping real user GUIDs (mocking B.2.7).

HTTP/1.1 200 OK
Content-Type: text/plain;charset=UTF-8
MIME-Version: 1.0
Content-Disposition: attachment; filename="smime.p7m"
Content-Type: application/x-pkcs7-mime; smime-type=enveloped-data; name="smime.p7m"
Content-Transfer-Encoding: base64
QXQgdmVybyBlb3MgZXQgYWNjdXNhbXVzIGV0IGIlc3RvIG9kaW8gZGlmbm1z
b3MgZHVaW1lcyBxdWkgYmxhbRpdGlpcyBwcmFlc2VudG1lbSB2b2x1cHRhDHVt
IGRlbGVuXRPiGF0cXVlIGNvcnJlIHRpbHFCb2x1cHRhdGVtIGRlbGVu
bGVzdlhcyBleGNlcnR1cmkgc2VycyB2YWx1ZSBmb3IgZm9yY3VydW5k
chJvdmkiZmFjdGVyc2Vyb2plY3R1cmRleSBlZHUjIjogIjoiMTQ2MjE1ODU2
IGNlZCBmb3IgZm9yY3VydW5kcyBmb3IgZm9yY3VydW5kcyBmb3IgZm9y
bGVzdlhcyBleGNlcnR1cmkgc2VycyB2YWx1ZSBmb3IgZm9yY3VydW5k
chJvdmkiZmFjdGVyc2Vyb2plY3R1cmRleSBlZHUjIjogIjoiMTQ2MjE1ODU2
IGNlZCBmb3IgZm9yY3VydW5kcyBmb3IgZm9yY3VydW5kcyBmb3IgZm9y

Deutsch  Client Application Layer Encryption  [Page 17]
B.2.4 Edge Forwarded Request

The request to some service:

GET /resource HTTP/1.1
Host: server.example.com

Cipher: ECDHE-RSA-AES256-SHA
B.2.5 Aggregation Request

A request from the origin server to another resource server:

GET /aggregate HTTP/1.1
Host: origin.example.com
CALE-PEM: "MIIEfDCCA2SgAwIBAgIRAOaxLLnaTZDrituxMDU+EwowDQYJKoZIhvcNAQELBQAwchezELMAkGA1UEBhMCVVMxGzAZBgNVBAoMC2V4YW1wbGF0ZSBEb2NlMjBOMw4wDgYDVQQDIE9Bc3VwZwYDVQQKEM1SaW4xMDQwHhcNMTgwNzUxMDUwNjEwWhcNMzEwMjIzMTQwNjEwWjEaMBIEFTEwDQYJKoZIhvcNAQELBQAwf8wDgYDVQQDIEFJU0ZwZzABBgNVBAcMDERlZmF1bHQgQ2l0eTEcMBoGA1UdDgQWMBQGCCsGAQUFBwMBBggrBgEFBQcDAjAdBgNVHQ4EFgQUSxqn9ioM+4Im9NWszrg3xvB3Xt4wHwYDVR0PAQH/BAQDAgWgMB0GA1UDQWMBQQIBOg=="

B.2.6 Aggregation Response

A response from a resource server to the origin server:

HTTP/1.1 200 OK
{"foo": "bar"}
B.2.7 Origination Response

The encrypted response from the origination server that will be passed back to the client by the edge device:

```
HTTP/1.1 200 OK
Content-Type: text/plain;charset=UTF-8
MIME-Version: 1.0
Content-Disposition: attachment; filename="smime.p7m"
Content-Type: application/x-pkcs7-mime;
smime-type=enveloped-data; name="smime.p7m"
Content-Transfer-Encoding: base64
```

B.3 whoami Request

B.3.1 Client whoami Request

The request to some service:

```
GET /whoami HTTP/1.1
Host: server.example.com
CALE-GUID: bec6dc7e-6562-4b1c-b308-6c352e6f8404
```

B.3.2 Edge whoami Request

The request forwarded to the authorization server with the cipher:

```
GET /whoami HTTP/1.1
Host: server.example.com
CALE-GUID: bec6dc7e-6562-4b1c-b308-6c352e6f8404
Cipher: ECDHE-RSA-AES256-SHA
```
B.3.3 Authorization whoami Response

Generate the certificate that expires immediately:

```bash
openssl x509 -req -days 0 -in client.req -CA ca.crt -CAkey ca.key -CAserial file.srl -out client.pem
```

The certificate is encrypted with itself using the cipher:

```bash
openssl smime -encrypt -binary -aes-256-cbc -in client.pem
```

HTTP/1.1 200 OK
MIME-Version: 1.0
Content-Disposition: attachment; filename="smime.p7m"
Content-Type: application/x-pkcs7-mime; smime-type=enveloped-data; name="smime.p7m"
Content-Transfer-Encoding: base64

MIIGogYJKoZIhvcNAQcDoIAgQSeAQgWMIIBkgIBADB6MHMxCzAJBgNV
BAYTA1MTQswCQYDVQQIDAJDQTEVMBQGA1UEAxwYVXVsdcCBDaXR5MRQw
EgYDVQQKEg1CCB8GCSqGSIb3DQEHAQRgBCBQAeMGATBCBAMCAwEAGB
kNKCIBCBAMCAwEAGBkNKCIBCAQAwIBAgIIRMFNQ2F0dGhvbGRoaW
lXb3JnMjEwHwYDVQQDDAbTb3JnMjEwHwYDVQQDDABTb3JnMjEwHwYD
QQDDABTb3JnMjEwHwYDVQQDDABTb3JnMjEwHwYDVQQDDABTb3JnMjEw
HwYDVQQDDABTb3JnMjEwHwYDVQQDDABTb3JnMjEwHwYDVQQDDABTb3Jn
MjEwHwYDVQQDDABTb3JnMjEwHwYDVQQDDABTb3JnMjEwHwYDVQQDDAB
Tb3JnMjEwHwYDVQQDDABTb3JnMjEwHwYDVQQDDABTb3JnMjEwHwYDVQQD
```

Deutsch  
Client Application Layer Encryption  
[Page 21]