Independent Submission D. Thakore
Request for Comments: 7562 CableLabs
Category: Informational July 2015
ISSN: 2070-1721

Transport Layer Security (TLS) Authorization Using Digital Transmission Content Protection (DTCP) Certificates

Abstract

This document specifies the use of Digital Transmission Content Protection (DTCP) certificates as an authorization data type in the authorization extension for the Transport Layer Security (TLS) protocol. This is in accordance with the guidelines for authorization extensions as specified in RFC 5878. As with other TLS extensions, this authorization data can be included in the client and server hello messages to confirm that both parties support the desired authorization data types. If supported by both the client and the server, DTCP certificates are exchanged in the supplemental data TLS handshake message as specified in RFC 4680. This authorization data type extension is in support of devices containing DTCP certificates issued by the Digital Transmission Licensing Administrator (DTLA).

Status of This Memo

Table of Contents

1. Introduction

The Transport Layer Security (TLS) protocol (see TLS 1.0 [RFC2246], TLS 1.1 [RFC4346], and TLS1 .2 [RFC5246]) is being used in an ever increasing variety of operational environments, the most common among which is its use in securing HTTP traffic [RFC2818]. [RFC5878] introduces extensions that enable TLS to operate in environments where authorization information needs to be exchanged between the client and the server before any protected data is exchanged. The use of these TLS authorization extensions is especially attractive since it allows the client and server to determine the type of protected data to exchange based on the authorization information received in the extensions.

A substantial number of deployed consumer electronics devices, such as televisions, tablets, game consoles, set-top boxes, and other multimedia devices, contain Digital Transmission Content Protection [DTCP] certificates issued by [DTLA]. These DTCP certificates enable secure transmission of premium audiovisual content between devices over various types of links (e.g., DTCP over IP [DTCP-IP]). These DTCP certificates can also be used to verify device functionality (e.g., supported device features).

This document describes the format and necessary identifiers to exchange DTCP certificates within the supplemental data message (see [RFC4680]) while negotiating a TLS session. The DTCP certificates are then used independent of their use for content protection (e.g., to verify supported features) and the corresponding DTCP Authentication and Key Exchange (AKE) protocol. This communication allows either the client, the server, or both to perform certain actions or provide specific services. The actual semantics of the authorization decision by the client/server are beyond the scope of this document. The DTCP certificate, which is not an X.509 certificate, can be cryptographically tied to the X.509 certificate being used during the TLS tunnel establishment by an Elliptic Curve Digital Signature Algorithm (EC-DSA) [DTCP] signature.

1.1. Applicability Statement

DTCP-enabled consumer electronics devices (e.g., televisions, game consoles) use DTCP certificates for secure transmission of audiovisual content. The AKE protocol defined in [DTCP] is used to exchange DTCP certificates and allows a device to be identified and authenticated based on the information in the DTCP certificate. However, these DTCP-enabled devices offer additional functionality (e.g., via HTML5 User Agents or web-enabled applications) that is distinct from its capability to transmit and play audiovisual content. The mechanism outlined in this document allows a DTCP-enabled consumer electronics device to authenticate and authorize using its DTCP certificate when accessing services over the internet; for example, web applications on televisions that can enable value-added services. This is anticipated to be very valuable since there are a considerable number of such devices. The reuse of well-known web security will also keep such communication consistent with existing standards and best practices.

1.2. Conventions

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 [RFC2119].

2. Overview

2.1. Overview of DTCP Certificates

DTCP certificates issued by [DTLA] to DTLA-compliant devices come in three general variations (see Section 4.2.3.1 of [DTCP]):

- Restricted Authentication device certificate format (Format 0): Typically issued to devices with limited computation resources.

- Baseline Full Authentication device certificate format (Format 1): This is the most commonly issued certificate format. Format 1 certificates include a unique DeviceID and device EC-DSA public/private key pair generated by the DTLA. (See Section 4.3 of [DTCP]).

- Extended Full Authentication device certificate format (Format 2): This is issued to devices that possess additional functions (e.g., additional channel ciphers, specific device properties). The presence of these additional functions is indicated by the device capability mask as specified in Section 4.2.3.2 of [DTCP]. Format 2 certificates also include a unique DeviceID and device EC-DSA public/private key pair generated by the DTLA (see Section 4.3 of [DTCP]).

The mechanism specified in this document allows only Formats 1 and 2 DTCP certificates to be exchanged in the supplemental data message since it requires the use of the EC-DSA private key associated with the certificate.

2.2. Overview of SupplementalData Handshake

Figure 1 illustrates the exchange of the SupplementalData message during the TLS handshake as specified in [RFC4680] (repeated here for convenience):

```
     Client                                               Server

     ClientHello (with extensions) -------->

                                      ServerHello(with extensions)
                                                SupplementalData*
                                                     Certificate*
                                               ServerKeyExchange*
                                              CertificateRequest*
                                 <--------        ServerHelloDone

     SupplementalData*
     Certificate*
     ClientKeyExchange
     CertificateVerify*
     [ChangeCipherSpec]
     Finished                    -------->
                                               [ChangeCipherSpec]
                                 <--------               Finished
     Application Data            <------->       Application Data

     *  Indicates optional or situation-dependent messages that are
        not always sent.

     [] Indicates that ChangeCipherSpec is an independent TLS
        protocol content type; it is not a TLS handshake message.
```

Figure 1: TLS Handshake Message Exchange with SupplementalData

2.3. Overview of Authorization Extensions

[RFC5878] defines two authorization extension types that are used in the ClientHello and ServerHello messages and are repeated below for convenience:

```
      enum {
        client_authz(7), server_authz(8), (65535)
      } ExtensionType;
```

A client uses the client_authz and server_authz extensions in the ClientHello message to indicate that it will send client authorization data and receive server authorization data, respectively, in the SupplementalData messages. A server uses the extensions in a similar manner in its ServerHello message. [RFC5878] also establishes a registry that is maintained by IANA to register authorization data formats. This document defines a new authorization data type for both the client_authz and server_authz extensions and allows the client and server to exchange DTCP certificates in the SupplementalData message.

2.4. Overview of SupplementalData Usage for Authorization

Section 3 of [RFC5878] specifies the syntax of the supplemental data message when carrying the authz_data message that is negotiated in the client_authz and/or server_authz types. This document defines a new authorization data format that is used in the authz_data message when sending DTCP Authorization Data.

3. DTCP Authorization Data Format

3.1. DTCP Authorization Type

The DTCP Authorization type definition in the TLS Authorization Data Formats registry is:

```
dtcp_authorization(66);
```

3.2. DTCP Authorization Data

The DTCP Authorization Data is used when the AuthzDataFormat type is dtcp_authorization. The syntax of the authorization data is:

```
struct {
    opaque random_bytes[32];
} RandomNonce;

struct {
    opaque RandomNonce nonce;
    opaque DTCPCert<0..2^24-1>;
    opaque ASN.1Cert<0..2^24-1>;
    opaque signature<0..2^16-1>;
} dtcp_authz_data;
```

RandomNonce is generated by the server and consists of 32 bytes generated by a high-quality, secure random number generator. The client always sends back the server-generated RandomNonce in its dtcp_authz_data structure. The RandomNonce helps the server in detecting replay attacks. A client can detect replay attacks by associating the ASN.1 certificate in the dtcp_authz_data structure with the certificate received in the Certificate message of the TLS handshake, so a separate nonce for the client is not required.

DTCPCert is the sender's DTCP certificate. See Section 4.2.3.1 of the DTCP Specification [DTCP].

ASN.1Cert is the sender's certificate used to establish the TLS session, i.e., it is sent in the Certificate or ClientCertificate message using the Certificate structure defined in Section 7.4.2 of [RFC5246].

The DTCPCert and ASN.1Cert are variable-length vectors as specified in Section 4.3 of [RFC5246]. Hence, the actual length precedes the vector's contents in the byte stream. If the ASN.1Cert is not being sent, the ASN.1Cert_length MUST be zero.

dtcp_authz_data contains the RandomNonce, the DTCP certificate, and the optional ASN.1 certificate. This is then followed by the digital signature covering the RandomNonce, the DTCP certificate, and the ASN.1 certificate (if present). The signature is generated using the private key associated with the DTCP certificate and using the Signature Algorithm and Hash Algorithm as specified in Section 4.4 of [DTCP]. This signature provides proof of the possession of the private key by the sender. A sender sending its own DTCP certificate MUST populate this field. The length of the signature field is determined by the Signature Algorithm and Hash Algorithm as specified in Section 4.4 of [DTCP], and so it is not explicitly encoded in the dtcp_authz_data structure (e.g., the length will be 40 bytes for a SHA1+ECDSA algorithm combination).

3.3. Usage Rules for Clients to Exchange DTCP Authorization Data

A client includes both the client_authz and server_authz extensions in the extended client hello message when indicating its desire to exchange dtcp_authorization data with the server. Additionally, the client includes the AuthzDataFormat type specified in Section 3.1 in the extension_data field to specify the format of the authorization data.

A client will receive the server's dtcp_authz_data before it sends its own dtcp_authz_data. When sending its own dtcp_authz_data message, the client includes the same RandomNonce that it receives in the server's dtcp_authz_data message. Clients MUST include its DTCP certificate in the dtcp_authz_data message. A client MAY include its ASN.1 certificate (certificate in the ClientCertificate message) in the ASN.1Cert field of the dtcp_authz_data to cryptographically tie the dtcp_authz_data with its ASN.1Cert being used to establish the TLS session (i.e., sent in the ClientCertificate message).

3.4. Usage Rules for Servers to Exchange DTCP Authorization Data

A server responds with both the client_authz and server_authz extensions in the extended server hello message when indicating its desire to exchange dtcp_authorization data with the client.

Additionally, the server includes the AuthzDataFormat type specified in Section 3.1 in the extension_data field to specify the format of the dtcp_authorization data. A client may or may not include an ASN.1 certificate during the TLS handshake. However, the server will not know that at the time of sending the SupplementalData message. Hence, a server MUST generate and populate the RandomNonce in the dtcp_authz_data message. If the client's hello message does not contain both the client_authz and server_authz extensions with dtcp_authorization type, the server MUST NOT include support for dtcp_authorization data in its hello message. A server MAY include its DTCP certificate in the dtcp_authz_data message. If the server does not send a DTCP certificate, it will send only the RandomNonce in its dtcp_authz_data message. If the server includes its DTCP certificate, it MUST also include its server certificate (sent in the TLS Certificate message) in the certs field to cryptographically tie its dtcp_authz_data with the ASN.1 certificate used in the TLS session being established. This also helps the client in detecting replay attacks.

3.5. TLS Message Exchange with dtcp_authz_data

Based on the usage rules in the sections above, Figure 2 provides one possible TLS message exchange where the client sends its DTCP certificate to the server within the dtcp_authz_data message.

```
   Client                                                   Server

   ClientHello (with extensions) -------->

                                 ServerHello(with extensions)
                               SupplementalData(with Nonce N1)
                                                  Certificate
                                            ServerKeyExchange*
                                           CertificateRequest
                                <--------      ServerHelloDone

   SupplementalData(with Data D1)
   Certificate
   ClientKeyExchange
   CertificateVerify
   [ChangeCipherSpec]
   Finished                     -------->
                                           [ChangeCipherSpec]
                                <--------             Finished
   Application Data             <------->     Application Data

 N1 Indicates a Random nonce generated by server

 D1 Contains dtcp_authz_data populated with the following
   {(N1, DTCP Cert, Client X.509 Cert) Signature over all elements}

 *  Indicates optional or situation-dependent messages that are
    not always sent.

 [] Indicates that ChangeCipherSpec is an independent TLS
    protocol content type; it is not a TLS handshake message.
```

Figure 2: DTCP SupplementalData Exchange

3.6. Alert Messages

This document reuses TLS Alert messages for any errors that arise during authorization processing and reuses the AlertLevels as specified in [RFC5878]. Additionally, the following AlertDescription values are used to report errors in dtcp_authorization processing:

unsupported_extension:
During processing of dtcp_authorization, a client uses this when it receives a server hello message that includes support for dtcp_authorization in only one of client_authz or server_authz but not in both the extensions. This message is always fatal. Note:

Completely omitting the dtcp_authorization extension and/or omitting the client_authz and server_authz completely is allowed and should not constitute the reason that this alert is sent.

certificate_unknown:
During processing of dtcp_authorization, a client or server uses this when it has received an X.509 certificate in the dtcp_authorization data and that X.509 certificate does not match the certificate sent in the corresponding ClientCertificate or Certificate message.

4. IANA Considerations

This document includes an entry registered in the IANA-maintained "TLS Authorization Data Formats" registry for dtcp_authorization(66). This registry is defined in [RFC5878] and defines two ranges: one is IETF Review, and the other is Specification Required. The value for dtcp_authorization should be assigned via [RFC5226] Specification Required. The extension defined in this document is compatible with Data Transport Layer Security (DTLS) [RFC6347], and the registry assignment has been marked "Y" for DTLS-OK.

5. Security Considerations

The dtcp_authorization data, as specified in this document, carries the DTCP certificate that identifies the associated device. Inclusion of the X.509 certificate being used to establish a TLS Session in the dtcp_authorization data allows an application to cryptographically tie them. However, a TLS Client is not required to use (and may not possess) an X.509 certificate. In this case, the dtcp_authorization data exchange is prone to a man-in-the-middle (MITM) attack. In such situations, a TLS server MUST deny access to the application features dependent on the DTCP certificate or use a double handshake. The double handshake mechanism is also vulnerable to the TLS MITM Renegotiation exploit as explained in [RFC5746]. In order to address this vulnerability, clients and servers MUST use the secure_renegotiation extension as specified in [RFC5746] when exchanging dtcp_authorization data. Additionally, the renegotiation is also vulnerable to the Triple Handshake exploit. To mitigate this, servers MUST use the same ASN.1 certificate during renegotiation as the one used in the initial handshake.

It should be noted that for the double handshake to succeed, any extension (e.g., TLS Session Ticket [RFC5077]) that results in the TLS handshake sequence being modified may result in failure to exchange SupplementalData.

Additionally, the security considerations specified in [RFC5878] and [RFC5246] apply to the extension specified in this document. In addition, the dtcp_authorization data may be carried along with other supplemental data or some other authorization data and that information may require additional protection. Finally, implementers should also reference [DTCP] and [DTCP-IP] for more information regarding DTCP certificates, their usage, and associated security considerations.

6. References

6.1. Normative References

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

[RFC2246] Dierks, T. and C. Allen, "The TLS Protocol Version 1.0", RFC 2246, DOI 10.17487/RFC2246, January 1999, <http://www.rfc-editor.org/info/rfc2246>.

[RFC4346] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.1", RFC 4346, DOI 10.17487/RFC4346, April 2006, <http://www.rfc-editor.org/info/rfc4346>.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <http://www.rfc-editor.org/info/rfc5246>.

[RFC5746] Rescorla, E., Ray, M., Dispensa, S., and N. Oskov, "Transport Layer Security (TLS) Renegotiation Indication Extension", RFC 5746, DOI 10.17487/RFC5746, February 2010, <http://www.rfc-editor.org/info/rfc5746>.

[RFC4680] Santesson, S., "TLS Handshake Message for Supplemental Data", RFC 4680, DOI 10.17487/RFC4680, October 2006, <http://www.rfc-editor.org/info/rfc4680>.

[RFC5878] Brown, M. and R. Housley, "Transport Layer Security (TLS) Authorization Extensions", RFC 5878, DOI 10.17487/RFC5878, May 2010, <http://www.rfc-editor.org/info/rfc5878>.

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

[DTCP] Digital Transmission Licensing Administrator, "Digital Transmission Content Protection Specification", Volume 1, Informational Version, <http://www.dtcp.com/documents/dtcp/info-20130605-dtcp-v1-rev-1-7-ed2.pdf>.

[DTCP-IP] Digital Transmission Licensing Administrator, "Mapping DTCP to IP", Volume 1, Supplement E, Informational Version, <http://www.dtcp.com/documents/dtcp/info-20130605-dtcp-v1se-ip-rev-1-4-ed3.pdf>.

6.2. Informative References

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, DOI 10.17487/RFC5226, May 2008, <http://www.rfc-editor.org/info/rfc5226>.

[DTLA] Digital Transmission Licensing Administrator, "DTLA", <http://www.dtcp.com>.

[RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818, DOI 10.17487/RFC2818, May 2000, <http://www.rfc-editor.org/info/rfc2818>.

[RFC5077] Salowey, J., Zhou, H., Eronen, P., and H. Tschofenig, "Transport Layer Security (TLS) Session Resumption without Server-Side State", RFC 5077, DOI 10.17487/RFC5077, January 2008, <http://www.rfc-editor.org/info/rfc5077>.

[RFC6042] Keromytis, A., "Transport Layer Security (TLS) Authorization Using KeyNote", RFC 6042, DOI 10.17487/RFC6042, October 2010, <http://www.rfc-editor.org/info/rfc6042>.

Appendix A. Alternate Double Handshake Example

This document specifies a TLS authorization data extension that allows TLS clients and servers to exchange DTCP certificates during a TLS handshake exchange. In cases where the supplemental data contains sensitive information, the double handshake technique described in [RFC4680] can be used to provide protection for the supplemental data information. The double handshake specified in [RFC4680] assumes that the client knows the context of the TLS session that is being set up and uses the authorization extensions as needed. Figure 3 illustrates a variation of the double handshake that addresses the case where the client may not have a priori knowledge that it will be communicating with a server capable of exchanging dtcp_authz_data (typical for https connections; see [RFC2818]). In Figure 3, the client's hello messages includes the client_authz and server_authz extensions. The server simply establishes an encrypted TLS session with the client in the first handshake by not indicating support for any authz extensions. The server initiates a second handshake by sending a HelloRequest. The second handshake will include the server's support for authz extensions, which will result in SupplementalData being exchanged.

Alternately, it is also possible to do a double handshake where the server sends the authorization extensions during both the first and the second handshake. Depending on the information received in the first handshake, the server can decide whether or not a second handshake is needed.

```
     Client                                                        Server

     ClientHello (w/ extensions) -------->                            |0
                                   ServerHello (no authz extensions)  |0
                                                        Certificate*  |0
                                                  ServerKeyExchange*  |0
                                                 CertificateRequest*  |0
                                 <--------           ServerHelloDone  |0
     Certificate*                                                     |0
     ClientKeyExchange                                                |0
     CertificateVerify*                                               |0
     [ChangeCipherSpec]                                               |0
     Finished                    -------->                            |1
                                                  [ChangeCipherSpec]  |0
                                 <--------                  Finished  |1
                                 <--------              HelloRequest  |1
     ClientHello (w/ extensions) -------->                            |1
                                         ServerHello (w/ extensions)  |1
                                                   SupplementalData*  |1
                                                        Certificate*  |1
                                                  ServerKeyExchange*  |1
                                                 CertificateRequest*  |1
                                 <--------           ServerHelloDone  |1
     SupplementalData*                                                |1
     Certificate*                                                     |1
     ClientKeyExchange                                                |1
     CertificateVerify*                                               |1
     [ChangeCipherSpec]                                               |1
     Finished                    -------->                            |2
                                                  [ChangeCipherSpec]  |1
                                 <--------                  Finished  |2
     Application Data            <------->          Application Data  |2

     *  Indicates optional or situation-dependent messages.
```

Figure 3: Double Handshake to Protect SupplementalData

Acknowledgements

The author wishes to thank Mark Brown, Sean Turner, Sumanth Channabasappa, and the Chairs (EKR, Joe Saloway) and members of the TLS Working Group who provided feedback and comments on one or more revisions of this document.

This document derives its structure and much of its content from [RFC4680], [RFC5878], and [RFC6042].

Author's Address

D. Thakore
Cable Television Laboratories, Inc.
858 Coal Creek Circle
Louisville, CO 80023
United States

Email: d.thakore@cablelabs.com