Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: July 22, 2016

D. Migault
K. Ma
Ericsson

January 19, 2016

Authentication Model and Security Requirements for the TLS/DTLS Content Provider Edge Server Split Use Case

draft-mglt-lurk-tls-requirements-00

Abstract

In the TLS/DTLS Content provider Edge Server Split use case, a TLS Client uses TLS/DTLS to authenticates the Content Provider while establishing a TLS/DTLS session with the Edge Server. Such authentication scheme is designated as Split Authentication in this document.

In most cases, the Edge Server does not even belong to the Content Provider, but instead to a third party like, for example, a Content Delivery Network. As a result, the Content Provider and the Edge Server must be able to interact and/or share some information. Interactions and shared information constitutes a split authentication model varies with the authentication method involved in the TLS session.

For each TLS/DTLS authentication method, the document provides the associated split authentication model that makes possible a split authentication. The split authentication model is associated to security requirements and an analysis to show it does not introduce any weakness compared to the standard TLS authentication model.

Status of This Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on July 22, 2016.

Table of Contents

1. Introduction

TLS is commonly used by applications to authenticate their counter part. Although the client part of the application and the TLS Client are likely to be hosted on the same node, this is no longer true on the server endpoints. As presented in [draft-mglt-tls-session-key-interface-use-cases], split scenarios considers that the TLS endpoint (Edge Server) and the application endpoint (Content Provider) may be hosted on different nodes that may not even share a common administrative domain.

Authentication of the Content Provider while establishing a TLS/DTLS session with the Edge Server is designated in this document as a split authentication. How split authentication can be performed varies on the TLS authentication method involved between the TLS Client and the Edge Server. The requirements on the information that can be shared between the Content Provider and the Edge Server, as well as the interactions between these two entities constitute a split authentication model.

This document provides a split authentication model for each TLS method. The model is often expresses as a list of requirements. These split authentication models are designed to avoid the leak of secret authentication credentials of the Content Provider, and matched against the standard TLS/DTLS authentication model. More especially, the split authentication models are designed not to introduce vulnerabilities or weakness compared to the standard TLS authentication model.

2. Conventions Used in This Document

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].

3. Terminology

In addition to the terminology used in [draft-mglt-tls-session-key-interface-use-cases], the current document defines

Split Authentication : designates the case when a TLS Client authenticates the Content Provider, while establishing a TLS session with a Edge Server. This is a three party authentication as the Edge Server may not necessarily belong to the Content Provider, nor share authentication credentials - like private keys for example - with the Content Provider. Use

cases can be found in [draft-mglt-tls-session-key-interface-use-cases]

Split Authentication Model : designates the information that can be shared as well as the interactions between the Edge Server and the Content Provider in order to enable the split authentication. In most cases, each TLS authentication method comes with a specific authentication model.

Standard TLS Authentication Model : designates the standard authentication model for TLS, i.e. between the TLS Client and the TLS Server.

DH: Diffie Hellman

ECDH Elliptical Diffie Hellman

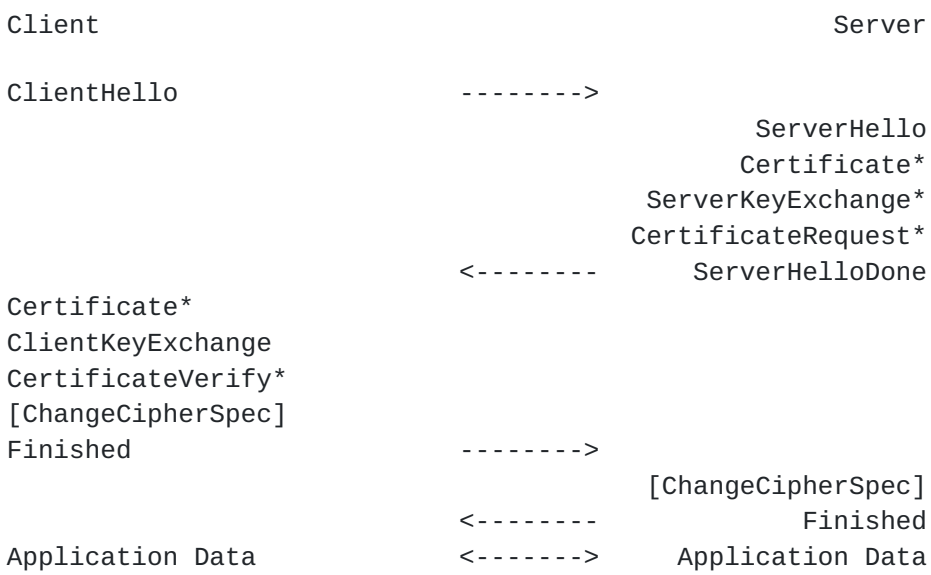
4. Handshake Authentication Methods for DTLS1.2 / TLS1.2

TLS has been designed for end-to-end security. This means that the TLS Client is expected to authenticate and set up a secure channel with the other end point of the communication designated as TLS Server.

TLS provides multiple KeyExchangeAlgorithm to authenticate the TLS Server by the TLS Client. Current authentication methods for DTLS 1.2 [RFC6347] TLS 1.2 [RFC5246] are dhe_dss, dhe_rsa, dh_anon, rsa, dh_dss and dh_rsa. [RFC4492] defines the additional ecdh_ecdsa, ecdhe_ecdsa, ecdh_rsa, ecdhe_rsa and ecdh_anon. In addition, [RFC4279] defines psk, dhe_psk and rsa_psk.

For each authentication method, this section provides a brief description of the authentication method. The authentication involves multiple credentials usually associated to the Content Provider. For each of these credential this section specifies whether it can be shared with the Edge Servers or not, and if not what information the Content Provider needs to provide to the Edge Server so the TLS session can be agreed between the TLS Client and the Edge Server.

Authentication in TLS1.2 is performed during the Handshake Protocol -- see section 7.3 in [RFC5246]. The messages involved in the authentication are Certificate, the ServerKeyExchange, the client Certificate, and the ClientKeyExchange message. Not all of them are always involved, and the following sections provides a high level description on how authentication is performed with the different methods. Figure 1 illustrates the various messages exchanges during a full handshake protocol in TLS1.2.



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

Figure 1: TLS1.2 Full Handshake

The purpose of the authentication in TLS is that the TLS Client and the TLS Server can agree on a master_secret that will be used to derive all necessary keys to secure the channel between the TLS Client and the TLS Server. This master_secret is derived from a pre_master agreed between the TLS Client and the TLS Server. [RFC5246] and [RFC7627] define different ways to generate the master_secret from the pre_master.

For information, in [RFC5246], the master_secret is generated as follows:

```
   master_secret = PRF(pre_master_secret, "master secret",
                       ClientHello.random + ServerHello.random)
                       [0..47];

   where:
   struct {
               uint32 gmt_unix_time;   # 4 bytes
               opaque random_bytes[28];
           } Random;
```

master_secret

[RFC7627] defines the Extended Master Secret Extension where the "master_secret" is defined as follows:

```
master_secret = PRF(pre_master_secret, "extended master secret",
                        session_hash)
                        [0..47];
where:
   - session_hash = Hash(handshake_messages)
   - handshake_messages is the concatenation of all the exchanged
     Handshake structures, as defined in Section 7.4 of [RFC5246].
   - Hash is as defined in Section 7.4.9 of [RFC5246]
```

Note that in TLS1.2 section 5 mentions "New cipher suites MUST explicitly specify a PRF and, in general, SHOULD use the TLS PRF with SHA-256 or a stronger standard hash function." This means that TLS1.2 provides the necessary cryptographic agility that allow the use of different hash function to generate the master secret from the premaster secret is collision free.

4.1. RSA Authentication

4.1.1. Standard TLS Authentication Description

When key exchange method chosen by the TLS Server is rsa, the TLS Server provides a ServerCertificate message that contains the public RSA key. This RSA key will be used for encryption. The TLS Client checks the public key provided by the certificate is associated to the requested entity, and then checks the binding between the RSA public key and the Content provider is certified by a trusted Certification Authority. This later operation requires that the TLS Client and the TLS Server share a common trusted Certification Authority as well as the TLS Client is able to check the signature embedded in the certificate. More specifically, the TLS must support the necessary hash and signature algorithms to check the certificate. The TLS Client does not inform the TLS Server what are the TLS Client's trusted CA, on the other hand, it provides the supported hash and signature - see Section 4.6.

The TLS Client sends the EncryptedPreMasterSecret, a premaster encrypted with the RSA public key of the TLS Server - provided in the Certificate - in a ClientKeyExchange message. The TLS Client does not provide any Certificate message.

The TLS Server is considered authenticated, if it can provide a proof of ownership of the private key associated to the public key provided in the certificate. In the case of the rsa key exchange algorithm, the TLS Server, needs the private key to decrypt the premaster secret in order to derive the master secret and all session keys, and have the same sessions keys as the TLS Client. Without the private key the TLS Server will not be able to decrypt the premaster secret and

thus not be able to agree with the TLS Client the session keys. In other words, the TLS session cannot be established.

4.1.2. Split Authentication Model

This section lists the requirements to use the RSA authentication in a split authentication model. The requirements regarding the credential information shared between the Content Provider and the Edge Server are:

REQ1: The Content Provider SHOULD share the RSA public key with the Edge Server. The RSA public key is public information and is conveyed in the ServerCertificate message.

REQ2: The Content Provider MUST NOT share the private RSA key with the Edge Server. The RSA private key is associated to the Content Provider and SHOULD be kept in a secure place.

The RSA private key is necessary to the Edge Server to decrypt the premaster secret so the TLS Server and the TLS Client can set the session keys. These operations can only be performed by the owner of the private key, that is in our case the Content Provider. Thus, the requirements regarding interactions between the Content Provider and the Edge Server are:

REQ3: The Content Provider MUST provide RSA decryption facilities to the Edge Servers. The Edge Server SHOULD be able to provide the EncryptedPremasterSecret - as well as additional necessary parameters - and be returned the clear text premaster or the master secret so the Edge Server and the TLS Client can agree and set the TLS channel.

4.1.3. Security Analysis

This section provides a security analysis of the split authentication model versus the standard TLS authentication model with RSA used as an authentication method. The purpose of this section is to show that the Content Provider in the split authentication model does not leak more information on its secret credential than the TLS Server does in the standard TLS model.

In the split authentication model, the Content Provider receives the EncryptedPremasterSecret and returns the cleartext premaster. This is exactly the information provided by the TLS Server to the TLS Client in the TLS standard authentication model.

The Edge Server can perform a chosen cipher text attack as it can send ciphered text to the Content Provider and get the corresponding

clear text. Such attack cannot be performed by the TLS Client in the standard TLS authentication model as the TLS Client generate the premaster and does not get the clear text premaster from the TLS Server, and retrieving the premaster from the session key is believe to be unfeasible.

Another difference between the split authentication model and the standard TLS model is that unlike the TLS Server, the Content Provider cannot check whether the computation of the premaster leads to an effective TLS session. This may be detected by the Edge Server and not by the Content Provider. On the other hand, the Content Provider centralizes the operations of all TLS Servers, which provides the ability to detect orchestrated or cipher text attacks. Also some of the discussion are discussed in the architecture document, on a model comparison, one can say that the split authentication model exposes the Content Provider to a cipher text attack, which can be prevented by the appropriated choice of the RSA parameters:

REQ4: RSA parameters MUST be chosen to make cipher text attack infeasible.

REQ5: The Content Provider MUST be able to provide the master secret or the extended master secret instead of the premaster. This could mitigate the cipher text attack as it is believe to be infeasible to retrieve the premaster secret from the master or extended master secret.

REQ6: The Content Provider SHOULD be able to provide the premaster secret.

In order to protect the TLS session between the TLS Client and the Edge Server, the master secret or the premaster secret must not be disclosed. Follows the security requirements on the Content Provider / Edge Server channel:

REQ7: The communication between the Edge Server and the Content Provider MUST be authenticated and encrypted.

As a result, the split authentication model is not expected to leak more information than the standard TLS authentication model as long as RSA parameters are appropriately chosen.

4.2. DH_DSS, DH_RSA, ECDH_ECDSA, ECDH_RSA

4.2.1. Standard TLS Authentication Description

With Diffie Hellman fixed values, the TLS Server provides a ServerCertificate which contains a certificate that contains the fixed DH or ECDH value. The DH or ECDH fix value is authenticated by the Certification Authority with a specific signature algorithm - RSA, DSS or ECDSA for example. As the DH or ECDH keys are used to establish a shared secret from public values, the purpose of the key is limited to the key agreement and the certificate is expected to have the key usage set to key agreement - e.g. the key is not used for signing or encrypting.

In order to compute the shared secret, the TLS Server expects to receive the TLS Client DH / ECDH counter part fix value. Optionally, the TLS Server can send a Certificate Request with the type corresponding to the key agreement method: rsa_fixed_dh, dss_fixed_dh, rsa_fixed_ecdh, ecdsa_fixed_ecdh.

Upon receiving the ServerCertificate message and optionally the CertificateRequest from the TLS Server, the TLS Client sends back a ClientCertificate with the same type as the ServerCertificate (rsa_fixed_dh, dss_fixed_dh, rsa_fixed_ecdh, ecdsa_fixed_ecdh).

Once the TLS Client and the TLS Server have exchanged their fixed DH / ECDH value, the pre_master can be derived by both the TLS Client and the TLS Server by combining the public and private DH / ECDH key. The premaster is then agreed between the TLS Server and the TLS Client and session keys can be derived from both the TLS Server and TLS Client.

4.2.2. Split Authentication Model

This section lists the requirements to use the DH_DSS, DH_RSA, ECDH_ECDSA, ECDH_RSA authentication in a split authentication model. The requirements regarding the credential information shared between the Content Provider and the Edge Server are:

REQ8: The Content Provider SHOULD share the DH or ECDH public key with the Edge Server. The DH or ECDH public key is public information and is conveyed in the ServerCertificate message.

REQ9: The Content Provider MUST NOT share the private DH or ECDH key with the Edge Servers. The DH or ECDH private key is associated to the Content Provider and SHOULD be kept in a secure place.

The DH or ECDH private key is necessary to the Edge Server to decrypt the premaster secret so the TLS Server and the TLS Client can set the

session keys. These operations can only be performed by the owner of the private key, that is in our case the Content Provider. Thus, the requirements regarding interactions between the Content Provider and the Edge Server are:

REQ10: The Content Provider SHOULD provide the Edge Server facilities to compute the DH or ECDH shared secret, the premaster, the premaster or the extended master to the Edge Server.

4.2.3. Security Analysis

This section provides a security analysis of the split authentication model versus the standard TLS authentication model with DH_DSS, DH_RSA, ECDH_ECDSA, ECDH_RSA used as an authentication method. The purpose of this section is to show that the Content Provider in the split authentication model does not leak more information on its secret credential than the TLS Server does in the standard TLS model.

When the Content Provider receives the public key of the TLS Client, it may return the shared DH / ECDH secret to the Edge Server. In that case, the Edge Server is exactly aware of the shared secret as well as the respective public keys of the Content Provider and the TLS Client. The situation differs from a node intercepting the DH / ECDH exchange as this node never gets the resulting secret. On the other hand, such information are also exchanged between the TLS Server and the TLS Client in the TLS standard authentication model.

As a result, interactions between the Edge Server and the Content Provider is not expected to leak more information then the TLS Server leak information to the TLS Client in [RFC5246].

Similarly to Section 4.1.3 the Content Provider can hardly distinguish the premaster that are being requested in order to perform an attack to the premasters that ends up into a TLS session. As a result,

REQ11: DH / ECDH parameters MUST be chosen to make guessing attacks infeasible.

REQ12: The Content Provider MAY provide the master secret or the extended master secret instead of the premaster. This could mitigate the cipher text attack as it is believe to be infeasible to retrieve the premaster secret from the master or extended master secret.

In order to protect the TLS session between the TLS Client and the Edge Server, the master secret or the premaster secret must not be

disclosed. Follows the security requirements on the Content Provider / Edge Server channel:

REQ13: The communication between the Edge Server and the Content Provider MUST be authenticated and encrypted.

As a result, the split authentication model is not expected to leak more information than the standard TLS authentication model as long as DH / ECDH parameters are appropriately chosen.

4.3. DH_anon, ECDH_anon

4.3.1. Standard TLS Authentication Description

With anonymous DH or ECDH - as opposed to fixed DH - the Certificate message is not appropriated to carry the DH or ECDH parameters, as they are not expected to be signed by a CA. As a result, the TLS Server provides the DH / ECDH parameters in a ServerDHParams structure or a ECParameters carried by a ServerKeyExchange message.

Upon receipt of the ServerKeyExchange, the TLS Client responds with a ClientKeyExchange with the associated DH / ECDH parameters.

Once the TLS Client and the TLS Server have exchanged the fixed DH / ECDH value, the pre_master is agreed between the TLS Server and the TLS Client and session keys can be derived from both the TLS Server and TLS Client.

4.3.2. Split Authentication Model

This section lists the requirements to use the DH_anon or ECDH_anon authentication in a split authentication model. The requirements regarding the credential information shared between the Content Provider and the Edge Server are:

REQ14: The DH or ECDH public and private keys SHOULD be generated by the Edge Servers. In other words, the DH or ECDH keys used SHOULD NOT be generated and associated to the Content Provider. As no authentication is provided to the TLS Client, there is no need to use private information of the Content provider.

4.3.3. Security Analysis

This section provides a security analysis of the split authentication model versus the standard TLS authentication model with DH_anon or ECDH_anon used as an authentication method. The purpose of this section is to show that the Content Provider in the split

authentication model does not leak more information on its secret credential than the TLS Server does in the standard TLS model.

The DH_anon and ECDH_anon authentication methods do not involve any authentication credentials from the Content Provider, so the risks of leakage of Content Provider authentication are not considered in this case.

As a result, the split authentication model is not expected to leak more information than the standard TLS authentication model.

4.4. DHE_DSS, DHE_RSA, ECDHE_RSA, ECDHE_ECDSA

4.4.1. Standard TLS Authentication Description

With ephemeral DH or ECDH - as opposed to fixed DH - the Certificate message is not appropriated to carry the DH / ECDH parameters. In fact, with ephemeral DH / ECDH, the DH / ECDH parameters are not signed by a CA. Instead, they are signed by a signature-capable certificate, which has been signed by the CA.

Since Certificate messages are not appropriated to carry the DH / ECDH parameters, the TLS Server provides the DH / ECDH parameters in a ServerDHParams structure or a ECParameters carried by a ServerKeyExchange message. This is the same structure as for anonymous DH / ECDH. In order to authenticate, a proof of ownership is added. This proof of ownership takes the form of a signature of a combination of the DH / ECDH parameters associated with the ClientHello.random and the ServerHello.random. The exact structure signed_params for DHE_RSA or ECDHE_RSA is defined in section 7.4.3 of [RFC5246], or Signature for ECDSA_ECDSA is defined in section 5.4 of [RFC4492]. In order to check the signature associated to the DH / ECDH parameters, the TLS Server provides the necessary public key associated to the TLS Server. These keys are carried in a ServerCertificate signed by a CA. These certificates are used to check the signature only.

Upon receipt of the ServerKeyExchange and the ServerCertificate, the TLS Client provides the DH / ECDH parameters to the TLS Server via the ClientDiffieHellmanPublic defined in section 7.4.7.2 of [RFC5246] or ClientECDiffieHellmanPublic structure defined in section 5.7 of [RFC4492]. Both structures are conveyed by the ClientKeyExchange message. This message does not carry any signatures that enable the TLS Server to authenticate the TLS Client. The TLS Client can be authenticated by providing a signing key in a ClientCertificate. The key is signed by a CA. On the other hand, the proof of ownership of the key by the TLS Client is provided by sending a CertificateVerify.

The CertificateVerify message is the generic mechanism to authenticate the TLS Client over the global TLS exchange.

4.4.2. Split Authentication Model

This section lists the requirements to use the DHE_DSS, DHE_RSA, ECDHE_RSA, ECDHE_ECDSA authentication in a split authentication model. The requirements regarding the credential information shared between the Content Provider and the Edge Server are:

REQ15: The Content Provider SHOULD share the public key with the Edge Server. The public key is used for signing the DH ECDH parameters. The public key is respectively a DSS public key when DHE_DSS is selected, a RSA public key when ECDHE_RSA is selected or a ECDSA public key when ECDHE_ECDSA is selected. The public key is public information and is conveyed in the ServerCertificate message.

REQ16: The Content Provider MUST NOT share the private RSA DSS or ECDSA key with the Edge Server. The private key is associated to the Content Provider and SHOULD be kept in a secure place.

The DSA, RSA or ECDSA private key is necessary to the Edge Server to sign the DH or ECDH parameters sent by the Edge Servers to the TLS Clients. These operations can only be performed by the owner of the private key, that is in our case the Content Provider. Thus, the requirements regarding interactions between the Content Provider and the Edge Server are:

REQ17: The Content Provider MUST provide DSS, RSA or ECDSA signing facilities to the Edge Servers. The Edge Server SHOULD be able to provide the extended parameters to be signed (that is the signed_params ClientHello.random, ServerHello.random and ServerKeyExchange.params) or their hash - as well as additional necessary parameters - and be returned the signed value so the Edge Server can send the ServerKeyExchange message to the TLS Client and the TLS Client.

[DISCUSSION: the DHE_DSS / DH_DSS are mentioned for TLS1.2 but do not seems to have specific actions associated to. I expected similar behavior for dhe_dss as for dhe_rsa, but section A.4.2. "Server Authentication and Key Exchange Messages" specifies no specific payload when dhe_dss is selected. I would like to clarify how DHE_DSS should be considered.]

4.4.3. Security Analysis

This section provides a security analysis of the split authentication model versus the standard TLS authentication model with DHE_DSS, DHE_RSA, ECDHE_RSA, ECDHE_ECDSA used as an authentication method. The purpose of this section is to show that the Content Provider in the split authentication model does not leak more information on its secret credential than the TLS Server does in the standard TLS model.

The Content provider is expected to provide some signatures associated to a given clear text. Such information provided by the Content Provider to the Edge Server and the TLS Client in the split authentication model are exactly the same information provided by the TLS Server to the TLS Client in the standard TLS authentication model.

The main difference between the two models is that the Content Provider does not have any means to check whether the corresponding signature effectively result in a running session or is part of an attack. In order to mitigate the surface of such attacks:

REQ18: Signature scheme and associated parameters MUST be chosen to be resistant to adaptive chosen cleartext attacks, i.e. the computation of signatures MUST NOT reveal information of the private key during the lifetime of the private key.

REQ19: The Edge Server SHOULD provide the complete clear text instead of the hash to be signed. In fact, providing the complete clear text provide the opportunity of the Content Provider to check what is being signed. With secure signature scheme it is believe that a collision with another clear text content has a very low probability. On the other hand, if the Edge Server simply provides the hash of the clear text, the Content Provider does not have any means to check what it is signing. In this case, the Content Provider relies on the Edge Server to provide appropriated content and legitimated content, which implies a strong trust relationship between the Content Provider and the Edge Server.

The Edge Server can provide the complete cleartext or a hash of the cleartext. An attacker may want to usurpate the Content Provider's identity and provides content associated with the signature of the content provider. In this case, the attacker may generate the malicious content and hash it. By submitting the hash to the Content Provider for signing, the Content Provider has no ways to check the content that is expected to be signed by the Content Provider. In other words, when hashes are submitted for signing, the content check and validation is expected to be left to the Edge Server. On the

other hand, when the whole content is provided to the Content Provider, the Content Provider is able to approve the content before signing it. In this case, the Edge Server is not assigned the role of checking the content. Instead, this role is left to the Content Provider, thus limiting the trust in the Edge Server. This limits the surface of attacks.

When the Edge Server provides the complete content to the Content Provider, an attacker that wants to usurpate the Content Provider's identity has the following possibilities:

a) It can provide a malicious content that is not detected as malicious by the Content Provider, and so get signed by the Content Provider. In order to address such vector of attack, the Content Provider should be provided some means to control the content that are signed. Otherwise, this function is delegated to the Edge Server, and the model implies the Edge Server is a trusted entity. When the clear text is provide instead of hash, the Content Provider has means to check the content its signs. On the other hand the Content Provider cannot check the signed clear text corresponds to an effective and legitimate TLS session.

b) It can generate the malicious content such that its hash matches an already known hash. Such attack is known as a hash collision. To address this vector of attack, the hash function is expected to be resistant to pre-image, second pre-image and collision attack.

c) It can guess the signature given the corresponding between cleartext and their corresponding signatures. In order to address this vector of attack, the signature is expected to be resistant to the adaptive chosen message attack.

Only a) is introduced by the split model. In fact, in the TLS model, the TLS Server signs content it generates and so assumes it is legitimate. As explained, this is not anymore always true in the split model where the Content Provider is not supposed to trust the Edge Server. Vector of attack mentioned in b) and c) are common in the split model and the standard TLS model, as an Edge Server in the split model does not have any additional information than a TLS Client in the TLS model. As a result:

REQ20: The Content Provider MUST be able to sign cleartext content rather than the resulting hash. In addition, the Content Provider MUST verify the cleartext content provided as input to avoid signature usurpation.

REQ21: The Content Provider SHOULD enable hash signature. When the Content Provider chose to sign hashes, the Edge Server MUST be a trusted node.

The signature is not expected to be secret. As a result, the information exchanged between the Content Provider and the Edge Server does not require confidentiality. In addition, the information provided to the Content Provider is not protected so confidentiality is not mandatory. On the other hand, it is of primary importance that only the legitimate Edge Server have access to the signing service.

REQ22: The communication between the Edge Server and the Content Provider MUST be authenticated.

As a result, the split authentication model is not expected to leak more information than the standard TLS authentication model.

4.5. PSK, DHE_PSK, RSA_PSK

4.5.1. Standard TLS Authentication Description

When PSK, DHE_PSK or RSA_PSK [RFC4279] are selected by the TLS Server for authentication, the TLS Server sends the TLS Client ServerKeyExchange message with a psk_identity_hint to indicate the TLS Client the PSK to select. When PSK has been selected, no additional information is provided by the TLS Server. When DHE_PSK is selected, the TLS Server adds the ServerDHParams to the ServerKeyExchange. When RSA_PSK is selected, the TLS Server provides the RSA public key of the TLS Server in a ServerCertificate message. The key will be used for encryption.

Upon receipt of the ServerKeyExchange message, the TLS Client responds with a ClientKeyExchange that contains the psk_identity. When PSK has been selected, no additional information is provided. The premaster includes the PSK designated by the psk_identity, and TLS session keys are derived based on the shared secret. In case the PSK is not known by both the TLS Client and the TLS Server, the premaster will not be agreed upon.

When DHE_PSK is selected, in addition to the psk_identity, the TLS Client provides the DH parameters in a ClientDiffieHellmanPublic structure that is added to the ClientKeyExchange message. The premaster is derived by the TLS Client and the TLS Server and includes both the DH shared secret as well as the PSK itself. Only if the TLS Server and the TLS Client share the same PSK, the premaster is not agreed upon.

When RSA_PSK is selected, in addition to the psk_identity, the TLS Client provides and EncryptedPreMasterSecret structure conveyed by the ClientKeyExchange message. The EncryptedPreMasterSecret contains among other informations a 46-byte random value and the PSK. When receiving this message, the TLS Server decrypts the premaster and check the validity between the PSK and the claimed psk_identity before the premaster is agreed.

To sum up PSK DHE_PSK key exchange methods requires up to five messages to be exchanged: ServerKeyExchange and ClientKeyExchange. RSA_PSK requires the TLS Server to provide an additional ServerCertificate.

4.5.2. Split Authentication Model

This section lists the requirements to use the PSK, RSA_PSK or the DHE_PSK authentication in a split authentication model. The requirements regarding the credential information shared between the Content Provider and the Edge Server are:

REQ23: The Content Provider MAY share the psk_identity_hints and associated profile with the Edge Servers. This information may be shared after an evaluation that they are public information and that profiles do not leak confidential, critical or privacy related information. In addition the Content Provider must also evaluate how the distribution of the hints profiles may be distributed and updated on the Edge Servers. Hints is definitely public as opposed to the PSK that is secret, so the list of psk_identity_hints associated to profiles may be provided to the Edge Servers. On the other hand, there might be privacy issues related to providing a complete list of hints, as well as management issues associated to maintaining up-to-date lists between multiple Edge Servers.

REQ24: The Content Provider SHOULD be able to provide psk_identity_hints on a per request basis. More especially, the Edge Server SHOULD be able to send a ClientHello message or specific parameters to the Content Provider in order to get the corresponding psk_identity_hint. Such interaction is necessary when privacy and management issues may require the profiles and psk_identity_hints be kept in a safe and centralized place.

REQ25: The Content Provider MUST NOT share the PSK with the Edge Server. The PSK is associated to the Content Provider and MUST be kept in a secure place.

When PSK is selected, the requirements regarding interactions between the Content Provider and the Edge Server are:

REQ26: The Content Provider SHOULD provide the Edge Server master secret generation from the psk_identity as well as additional information. As the premaster conveys the PSK in cleartext, the Content Provider MUST NOT provide the cleartext premaster, but instead should compute the master in order to avoid transmitting the PSK to the Edge Server.

When PSK_RSA is selected, the requirements regarding interactions between the Content Provider and the Edge Server are similar as those to the RSA case except that the PSK SHOULD NOT be returned by the Content Provider. More especially:

REQ27: The Content Provider SHOULD share the RSA public key with the Edge Server. The RSA public key is public information and is conveyed in the ServerCertificate message.

REQ28: The Content Provider MUST NOT share the private RSA key with the Edge Server. The RSA private key is associated to the Content Provider and SHOULD be kept in a secure place.

REQ29: The Content Provider MUST provide RSA decryption facilities to the Edge Servers. The Edge Server SHOULD be able to provide the EncryptedPremasterSecret - as well as additional necessary parameters - and be returned the clear master so the Edge Server and the TLS Client can agree and set the TLS channel. As the premaster conveys the PSK in cleartext, the Content Provider MUST NOT provide the cleartext premaster, but instead should compute the master in order to avoid transmitting the PSK to the Edge Server.

When DHE_PSK is selected, the DH exchange may be performed by the Edge Server or by the Content Provider. When the DH exchange is performed between the Edge Server and the TLS Client, the Edge Server needs to provide the DH share secret - and the psk_identity - to the Content Provider which in turn generates the master secret with its PSK. Note that performing the DH with the Edge Server requires the Edge Server to provide a Certificate with a public key signed by a CA trusted by the TLS Client. This key may also be associated to the Content provider URL. When the exchange is performed between the Content Provider and the TLS Client, the agreed DH shared secret as well as the PSK are only known by the Content Provider, which limits the possibilities for the Edge Server to guess the PSK for example.

When the DH exchange is performed by the Edge Server and the TLS Client, the requirements regarding interactions between the Content Provider and the Edge Server are:

REQ30: The Content Provider MUST be able to generate a master from the shared secret agreed between the Edge Server and the TLS Client and the psk_identity provided by the TLS Client.

REQ31: As the premaster conveys the PSK in cleartext, the Content Provider MUST NOT provide the cleartext premaster, but instead should compute the master in order to avoid transmitting the PSK to the Edge Server. This requirement assumes that the Edge Server is performing the DH exchange.

When the DH exchange is performed between the Content Provider and the TLS Client, the requirements regarding interactions between the Content Provider and the Edge Server are:

REQ32: The Content Provider SHOULD share the private DH key with the Edge Server. These DHParams will be sent to the TLS Client by the Edge Server in the ServerKeyExchange message.

REQ33: The Content Provider SHOULD be able to generate a master from the psk_identity and TLS Client DHParams - in addition to potential additional parameters. As the premaster conveys the PSK in clear, the Content Provider MUST NOT provide the cleartext premaster, but instead should compute the master in order to avoid transmitting the PSK to the Edge Server.

4.5.3. Security Analysis

This section compares the PSK, DHE_PSK, RSA_PSK authentication method as described in [RFC4279] in a standard TLS model to the split model described in this document - that is when it is split between the Edge Server and the Content Provider. The main question considered is how the split model exposes the authentication credential compared to the standard TLS model described in [RFC5246].

All PSK, DHE_PSK, RSA_PSK assumes the PSK owned by the Content Provider MUST NOT be provided to the Edge Server. The difference between the split authentication model and the standard TLS authentication model is that in the standard TLS model, the TLS Server owns the PSK and derives the master secret from the PSK. On the other hand, the split authentication model assumes that the Edge Server is not aware of the PSK while being aware of the master secret. The difference is that the TLS standard model assumes all parties (TLS Server and TLS Client) share the PSK, whether the split model assumes the PSK is not shared between all parties. As a

result, the PSK leakage analysis is out of scope of the standard TLS model and is specific to the split model.

The PSK is used to generate the master secret using a hash function. The leakage of the PSK information in the master depends on the hashing function and its collision free properties. [RFC4279]. As a result, we can assume that the master secret does not leak information about the premaster to the Edge Server.

For PSK_RSA, the authentication credential of the Content provider are a RSA key and a PSK. Section 4.1 addresses the requirements associated to the leakage of the RSA credential. Similarly to the RSA analyzed in Section 4.1.3:

REQ34: RSA parameters MUST be chosen to make cipher text attack infeasible.

For DHE_PSK, the DH exchange may be performed between the TLS Client and the Content Provider or between the TLS Client and the Edge Server. When the DH exchange is performed between the TLS Client and the Content provider, the Content Provider provides the public key used to sign the DHparams structure. In this case the analysis is similar to Section 4.4. When the DH exchange is performed between the Edge Server and the TLS Client, the Content Provider does not share any additional authentication credentials. Similarly to DH / ECDH:

REQ35: DH parameters MUST be chosen to make guessing attacks infeasible.

In order to protect the TLS session between the TLS Client and the Edge Server, the master secret or the premaster secret must not be disclosed. Follows the security requirements on the Content Provider / Edge Server channel:

REQ36: The communication between the Edge Server and the Content Provider MUST be authenticated and encrypted.

As a result, the split authentication model is not expected to leak more information than the standard TLS authentication model.

4.6. Client Hash and Signature

These are provided in the SignatureAndHash which is an Hello Extension. As detailed in [RFC5246], if this extension has not been provided, and rsa has been chosen as KeyExchangeAlgorithm, than the server behaves as if the TLS Client had provided sha1 for the hash function and rsa for the supported signatures. If the TLS Client

does not support sha1, than it has to use the extension to indicate it.

5. Security Considerations

The scope of this document is to avoid that secret information information associated to authentication credentials is spread over multiple Edge Servers. Such models would assume that all Edge Server remain trusted and reliable over time while being exposed on the Internet. Instead the secret information is kept in one secured place own by the Content Provider. Authentication of the Content Provider by the TLS Clients is performed through the Edge Servers. Such model designated in this document as the split model requires some interactions between the Edge Servers and the Content Provider. This document described what are the necessary interactions between the Edge Servers and the Content Provider as well as the information that can be shared between the Edge Servers and the Content Provider and the information that must not be shared and that must remain secret. These interactions as well as the shared and non shared information do not expose the split model to additional risks of secret leakages than in the standard TLS model.

The security associated to the authentication relies on the authentication protocols defined in [RFC5246], [RFC4279] and [RFC4492] and practices provided by [RFC7525]. As a result, security consideration of these document apply.

6. IANA Considerations

There is no IANA considerations in this document.

7. Acknowledgements

8. References

8.1. 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>.

[RFC4279] Eronen, P., Ed. and H. Tschofenig, Ed., "Pre-Shared Key Ciphersuites for Transport Layer Security (TLS)", RFC 4279, DOI 10.17487/RFC4279, December 2005, <http://www.rfc-editor.org/info/rfc4279>.

[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>.

[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>.

[RFC7525] Sheffer, Y., Holz, R., and P. Saint-Andre, "Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", BCP 195, RFC 7525, DOI 10.17487/RFC7525, May 2015, <http://www.rfc-editor.org/info/rfc7525>.

[RFC7627] Bhargavan, K., Ed., Delignat-Lavaud, A., Pironti, A., Langley, A., and M. Ray, "Transport Layer Security (TLS) Session Hash and Extended Master Secret Extension", RFC 7627, DOI 10.17487/RFC7627, September 2015, <http://www.rfc-editor.org/info/rfc7627>.

8.2. Informative References

[RFC4492] Blake-Wilson, S., Bolyard, N., Gupta, V., Hawk, C., and B. Moeller, "Elliptic Curve Cryptography (ECC) Cipher Suites for Transport Layer Security (TLS)", RFC 4492, DOI 10.17487/RFC4492, May 2006, <http://www.rfc-editor.org/info/rfc4492>.

Authors' Addresses

Daniel Migault
Ericsson
8400 boulevard Decarie
Montreal, QC H4P 2N2
Canada

Phone: +1 514-452-2160
Email: daniel.migault@ericsson.com

Kevin Ma J
Ericsson
43 Nagog Park
Acton, MA 01720
USA

Phone: +1 978-844-5100
Email: kevin.j.ma@ericsson.com