IPSec X.509 Certificates

This chapter describes a number of StarOS features that support IPSec certificate management.

The following topics are discussed:

- Multiple Child SA (MCSA) Support, on page 1
- Creating, Signing, and Configuring Certificates, on page 3
- CA Certificate Chaining, on page 4
- Certificate Management Protocol (CMPv2), on page 6
- Online Certificate Status Protocol (OCSP), on page 14
- CRL Fetching, on page 18

Multiple Child SA (MCSA) Support

Overview

A child SA is an Encapsulating Security Payload (ESP) or Authentication header (AH) security association (SA) carrying the secure user traffic. An SA is a "simplex connection"; to achieve bidirectional secure traffic a pair of SAs is required (RFC 5996). To meet this common requirement, IKE explicitly creates SA pairs. An SA pair is referred to as a "Child SA"; one child SA is a pair of IPsec SAs in each direction.

StarOS supports creation up to five child SAs under the crypto template configuration. Child SAs are supported only for IKEv2.

Each child SA should consist of mutually exclusive traffic selectors which are configured via crypto template payloads.

The following traffic selectors would match UDP packets from 198.51.100.66 to anywhere, with any of the four combinations of source/destination ports (100,300), (100,400), (200,300), and (200, 400). Thus, some types of policies may require several Child SA pairs. For instance, a policy matching only source/destination ports (100,300) and (200,400), but not the other two combinations, cannot be negotiated as a single Child SA pair.

```
TSi = ((17, 100, 198.51.100.66-198.51.100.66),(17, 200, 198.51.100.66-198.51.100.66))
TSr = ((17, 300, 0.0.0.0-255.255.255.255),(17, 400, 0.0.0.0-255.255.255.255))
```

The following triggers create Child SAs:

- The initiator of IKE_INIT can start subsequent Child SA creations after the first Child SA creation based on initiator traffic selector (TSi) configuration which calls for multiple Child SAs. StarOS receives CREATE_CHILD_SA request after IKE_AUTH.
- The responder can initiate subsequent Child SA creation after the first child SA creation based on the responder traffic selector configurations (TSr) which calls for multiple Child SAs. StarOS sends CREATE_CHILD_SA request after IKE_AUTH.

Deployment Scenarios

The creation of multiple child SAs helps an operator to segregate and limit the secure traffic into multiple flows. For example, control and data paths between two nodes can be established over two child SAs; the rest of the data between the nodes will bypass IPSec.

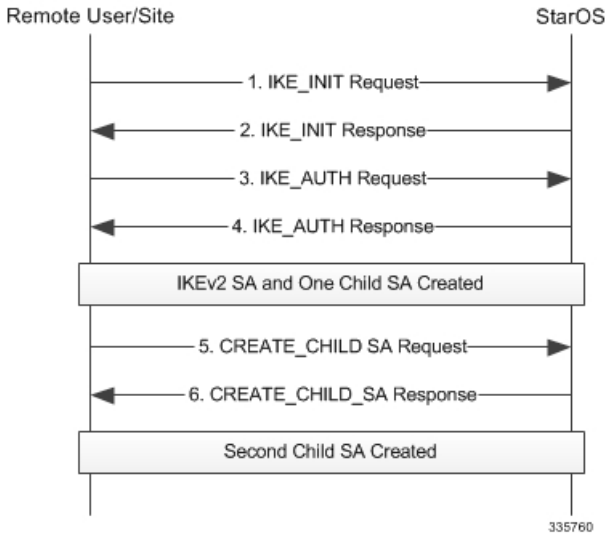
Multiple child SAs may be used for carrying traffic with different class of services (QoS). Similarly, different SAs could be used to carry different traffic with specific security properties. For example, one SA with strongest protection, another with a weaker one, and still another with a proprietary one stipulated by legal, performance or cost needs.

Call Flows

Child SA Creation by Initiator

With crypto template configuration, Child SA creation is initiated by the IKE_INIT initiator through a CREATE_CHILD_SA exchange or by StarOS acting as the responder. The first Child SA is created using the first traffic selector. After creating the first Child SA, the initiator requests the second Child SA using the second traffic selector. The responder completes the creation of the second Child SA.

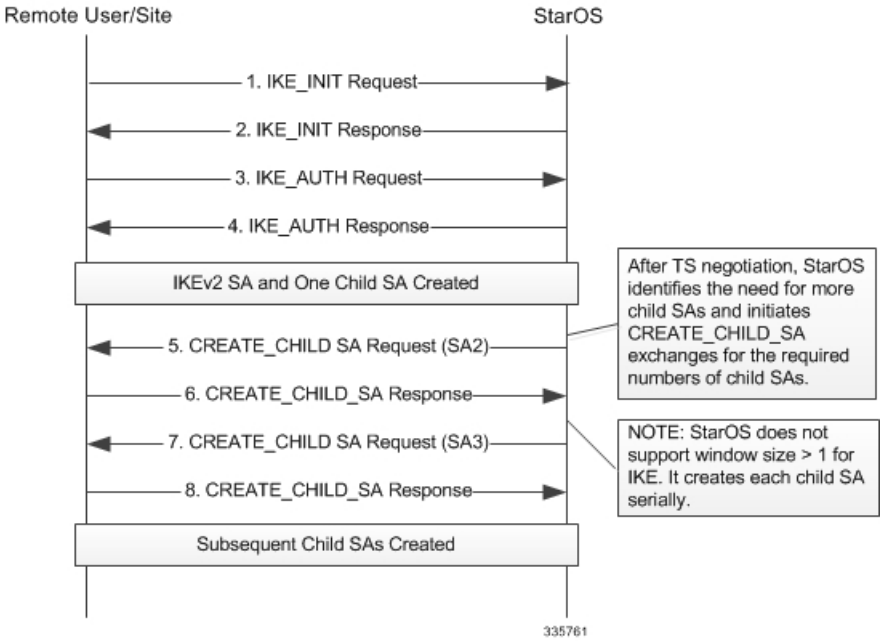
Figure 1: Child SA Creation Initiated by IKE_INIT



Child SA Creation by Responder

After negotiating a transform set (TS), the responder detects the need to create more child SAs to support configured traffic selectors. It sends CREATE_CHILD_SA to create as many child SAs as required to meet the TS configuration. The initiator completes subsequent child SA creations.

Figure 2: Child SA Creation Initiated by StarOS as Responder



Creating, Signing, and Configuring Certificates

Use the following procedure to create, sign, and configure certificates:

1. Add a file location where the certificates and private keys will be stored:

   ```
   config
      cmp cert-store location path_name
      end
   ```

2. Generate a Certificate Signing Request (CSR):

   ```
   crypto rsa-keygen modulus { 1024 | 2048 | 4096 | 512 } id-type { fqdn
    id fqdn_id | ip id IP_address | rfc822-addr id id_type } subject-name
   subject_string
   ```

   A new private key along with the certificate request will be generated in the configured file location.

3. Use the generated certificate request to apply for a digital identity certificate from the certificate authority (CA).

4. Once the certificate is received, download and configure the certificate file to an accessible path:

   ```
   certificate name name { der url pathname | pem { data pemdata | url pathname
    } private-key pem { [ encrypted ] data pemdata | url pathname [cert-enc]
    [cert-hash-url url patname ] } }
   ```

Notes:

- Use the **no cmp cert-store** command to remove the certificate storage location configuration.
- Use the **no certificate name** *name* command to remove the certificate configuration.
- *pathname* must be in one of the following formats:
  - [file:]{ /flash | /usb1 | /hd-raid }[/*directory*]/*filename*
  - tftp://*host*[:*port*][/*directory*]/*filename*
  - ftp://[*username*[:*password*]@]*host*[:*port*][/*directory*]/*filename*
  - sftp://[*username*[:*password*]@]*host*[:*port*][/*directory*]/*filename*
  - http://[*username*[:*password*]@]*host*[:*port*][/*directory*]/*filename*
- *fqdn_id*, *id_type*, *subject_string* must be an alphanumeric string from 1 to 256 characters
- *IP_address* can be an IPv4 address with dotted-decimal notation, or an IPv6 address with colon-separated hexa-decimal notation.
- *pemdata* must be an alphanumeric string of 1 through 4095 (if private key is not implemented) or 1 through 8191 (if private key is implemented) characters.

Certificate and Private Key Storage

Certificates (configured using a URL) and private keys are stored as a file in a private directory locally. The output of the **show config** command displays the local URL of the certificate (only if the bootup configuration is URL) and private key instead of the data. When the certificate is removed using the **no certificate** *certificate_name* command, the certificate and private key from the local private directory are removed.

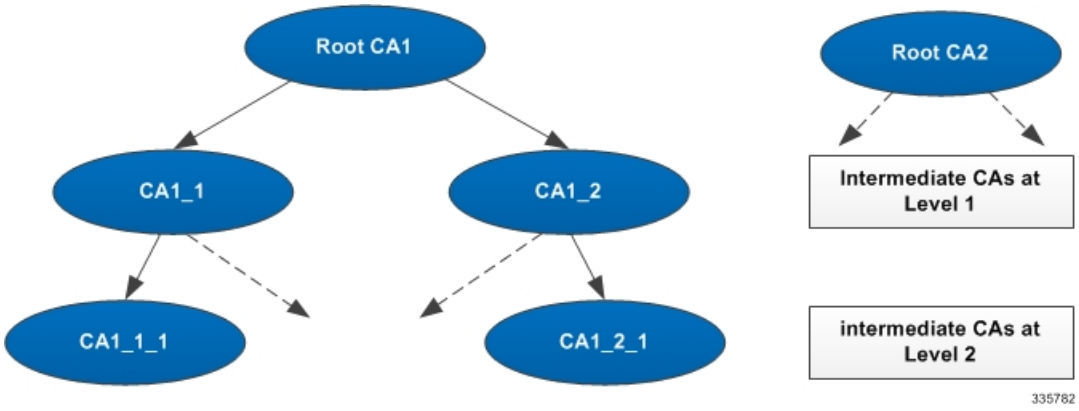
CA Certificate Chaining

Overview

Certificate chaining, also known as hierarchical CA cross certification, is a method by which an entity is authorized by walking a sequence of intermediate As up to the trust-point CA. An intermediate CA is a certification authority under a root CA, which is a self-signed authority.

The sequence of root and intermediate certificates belonging to CA is called a "chain". Each certificate in the chain is signed by the subsequent certificate. In this scheme, the web server certificate (the one that is to be installed on the web server where the user's site is hosted) is signed not by a root certificate directly but by one of the intermediates.

Figure 3: Root and Intermediate CAs in Certificate Chaining



The peer entities may obtain a certificate from any of the root CAs or intermediate CAs. A certificate may be authenticated by walking the chain up to a trust anchor, which may be either an intermediate CA or the root CA in the chain.

When an entity sends its certificate to the peer, it must also send all the certificates in the chain up to the trust anchor requested by the peer, not including the trust anchor certificate itself.

StarOS only supports X.509 Certificate encoding when sending certificates with a maximum certificate chain length of 4. The length of the certificate chain is defined as the number of all certificates in the chain, including the entity and intermediate CA certificates, but excluding the trust anchor certificate.

Deployment Scenarios

StarOS as Responder

Cert. Data in the Payload – Peer Cert. root CA1, StarOS Cert. Intm. CA1_1

1. StarOS sends IKE_SA_INIT to the peer.
2. StarOS sends IKE_SA_INIT to Peer. StarOS includes CERTREQ with Encoding = "X.509 Certificate - Signature" and Certification Authority = "Concatenated hashes of public key info of CA 1_1 and CA1 in any order".
3. Peer sends IKE_AUTH to StarOS. Peer includes CERT with requested encoding type, and an entity certificate issued by CA1. Peer includes CERTREQ with Encoding = "X.509 Certificate - Signature" and Certification Authority = "Hash of public key info of CA1". StarOS authenticates the peer certificate against CA1.
4. StarOS sends IKE_AUTH to Peer. StarOS includes two CERT payloads, with Encoding = "X.509 Certificate - Signature", and certificate data of (1) StarOS and (2) CA1_1.

Cert. Data in the Payload – Peer Cert. Intm CA1_1, StarOS Certificate root CA1

1. StarOS sends IKE_SA_INIT to the peer.
2. Peer sends IKE_SA_INIT to StarOS. This message includes CERTREQ with Encoding = "X.509 Certificate - Signature" and Certification Authority = "Hash of public key info of CA1_1 and CA1 in any order".

3. StarOS sends IKE_AUTH to peer. StarOS includes one CERT payload with requested encoding type, and the entity certificate issued by CA1. StarOS includes CERTREQ with Encoding = "X.509 Certificate - Signature" and Certification Authority = "Hash of public key info of CA1".
4. Peer sends IKE_AUTH to StarOS. Peer includes two CERT payloads, with Encoding = "X.509 Certificate - Signature", and (1) the entity certificate data, and (2) certificate data of CA1_1.

StarOS as Initiator

Cert. Data in the Payload – Peer Cert. root CA1, StarOS Cert. Intm. CA1_1

1. StarOS sends IKE_SA_INIT to the peer.
2. Peer sends IKE_SA_INIT to StarOS. This message includes CERTREQ with Encoding = "X.509 Certificate - Signature" and Certification Authority = "Hash of public key info of CA1".
3. StarOS sends IKE_AUTH to peer. StarOS includes two CERT payloads with requested encoding type, and (1) an entity certificate issued by CA1_1, and (2) a certificate of CA1_1. StarOS includes CERTREQ with Encoding = "X.509 Certificate - Signature" and Certification Authority = "Hash of public key info of CA1 and CA1_1 in any order".
4. Peer sends IKE_AUTH to StarOS. Peer includes one CERT payload, with Encoding = "X.509 Certificate - Signature", and the entity certificate data.

Cert. Data in the Payload – Peer Cert. Intm CA1_1, StarOS Certificate root CA1

1. StarOS sends IKE_SA_INIT to the peer.
2. Peer sends IKE_SA_INIT to StarOS. This message includes CERTREQ with Encoding = "X.509 Certificate - Signature" and Certification Authority = "Hash of public key info of CA1_1 and CA1 in any order".
3. StarOS sends IKE_AUTH to peer. StarOS includes one CERT payload with requested encoding type, and the entity certificate issued by CA1. StarOS includes CERTREQ with Encoding = "X.509 Certificate - Signature" and Certification Authority = "Hash of public key info of CA1".
4. Peer sends IKE_AUTH to StarOS. Peer includes two CERT payloads, with Encoding = "X.509 Certificate - Signature", and (1) the entity certificate data, and (2) certificate data of CA1_1.

External Interfaces

Support for "Hash & URL" of certificates/bundle requires HTTP or FTP interfaces to download the data which is implemented separately. OCSP verification of certificates also includes a TCP connection to the OCSP server during verification.

Certificate Management Protocol (CMPv2)

Overview

In cryptography, a public key certificate (also known as a digital certificate or identity certificate) is an electronic document which uses a digital signature to bind a public key with an identity information, such as

the name of a person or an organization, their address, and so forth. The certificate can be used to verify that a public key belongs to an individual. The Certificate Management Protocol (CMP) is an Internet protocol used for obtaining X.509 digital certificates in a public key infrastructure (PKI). It is described in RFC 4210.

StarOS implements the subset of CMPv2 functions:

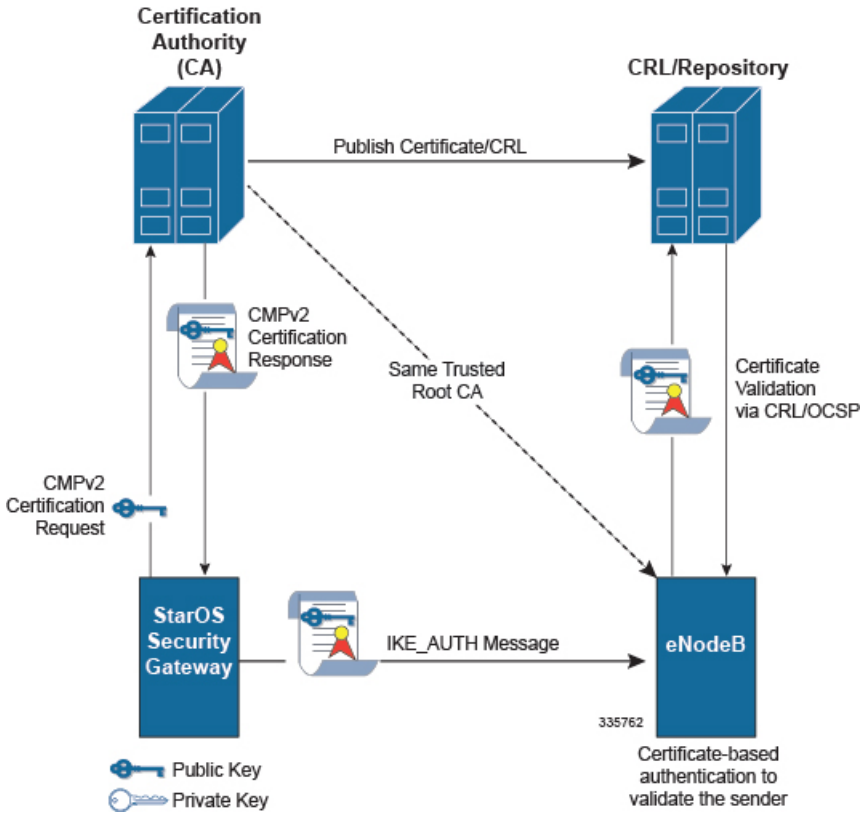
- **Key pair and X.509 certificate request generation**: The StarOS security gateway acts as an end entity as described in RFC 4210. The gateway generates the X.509 public and private key pair for authentication during IKE AUTH. It generates the public and private keys using OpenSSL libraries. The generated private key is saved locally on the management card, and the public key is embedded in the generated X.509 certificate request. The key uses RSA encryption; SHA-1 with RSA encryption is used on the Hashing function for the generated certificate. Certificate requests are sent to the Certificate Authority (CA) or Registration Authority (RA) during the certification process via CMPv2.
- **Initial certificate request transaction (ir and ip)**: A certificate request triggers the CMPv2 messaging to get the first certificate certified by the Certification Authority (CA). This CMPv2 transaction is identified by the Certification Request and Certification Response messages (ir and ip). At the end of this transaction the security gateway may receive the certificate signed by the CA in the response message. This certificate is then saved in the management card and is also propagated to the packet processing cards via internal messaging. The IKEv2 tunnel creation done at the packet processing cards requires this certificate for the IKE_AUTH transaction.
- **Certificate enrolment (cr and cp)**: This CMPv2 transaction obtains additional certificates certified by CA after the initial certification is done. The security gateway triggers additional certificate enrolment. The additional certificates are saved and used in a manner similar to the initial certificate.
- **Polling request and response (pollReq and pollRep)**: The ip, cp or the kup message received from the CA may contain a status code of "waiting". This indicates that the CA is still evaluating the certificate request and will require more time to sign the certificate. In this case the security gateway sends a pollReq message to the CA. The pollRep message from the CA may either contain the signed certificate or indicate a status of "waiting" again. If the pollRep message contains the certificate, it is treated as an ip/cp/kup message with a signed certificate and all relevant actions are taken. The security gateway also supports a CLI command to manually trigger polling for any request.
- **Certificate update transaction (kur and kup)**: This key pair update transaction re-certifies or updates a public/private key pair of the certificate after or before its validity expires. The Key Update Request (kur) message is sent to the CA with a certificate having a new public key, and the CA sends a Key Pair Update Response (kup) message with the signed certificate. The security gateway also supports two mechanisms to update an existing certificate:
  - **Manual Update**: The gateway sports a CLI command to trigger the certificate update transaction.
  - **Auto update**: The gateway can be configured to automatically trigger a certificate update a specified number of days before the certificate expires.
  - For both manual and automatic updates, the updated certificate is saved on the management card and propagated to the packet processing cards.

Deployment Scenarios

In a 4G network the data between the eNodeB and the MME/SGW is sent via a security gateway. The network between the security gateway and the MME/SGW is a trusted network of the vendor. The network between

the eNodeB and security gateway may be a public network requiring the establishment of an IPSec tunnel between eNodeB and the security gateway through which data is sent.

Figure 4: CMPv2 Deployment Scenario



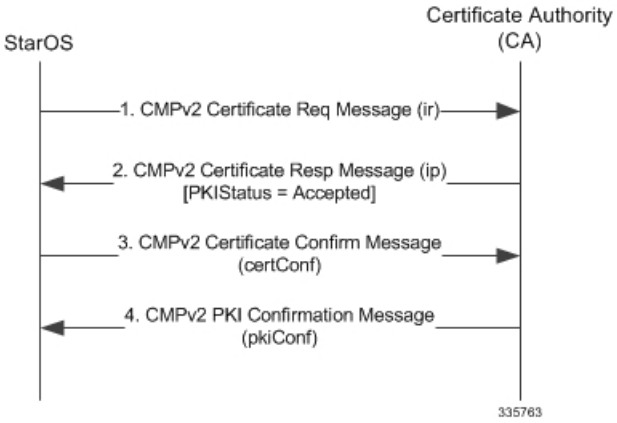
The IKEv2 protocol is used to establish the IPSec tunnel between eNodeB and the MME/SGW. Certificate-based authentication is performed during stage 2 of the IKEv2 exchange (RFC 4306). The security gateway sends its own X.509 certificate to the eNodeB in the IKE_AUTH message's CERT payload. This certificate is validated at the eNodeB and is used to decrypt the AUTH payload to authenticate the security gateway.

CMPv2 is the online mechanism for generating public and private keys and obtaining the certificate signed by a CA.

Call Flows

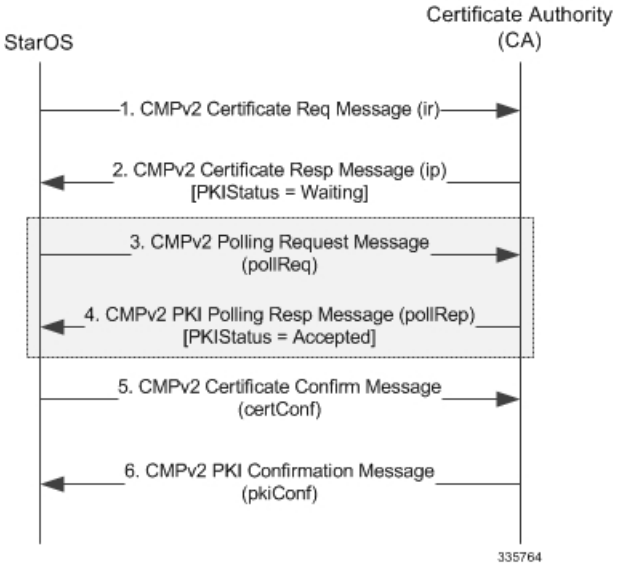
Initial Certification Request

Figure 5: Call Flow: Successful Initial Certification Request



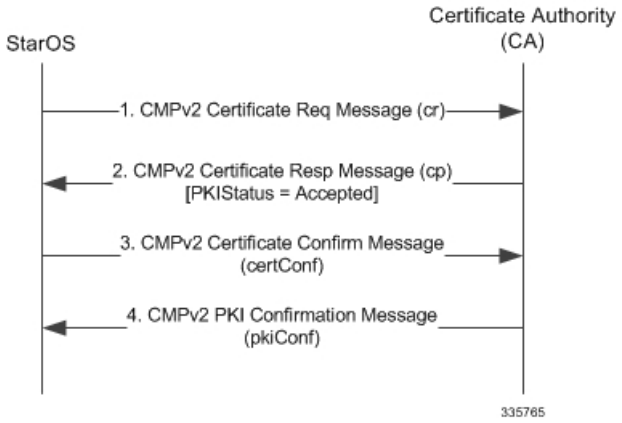
Initial Certification Request with Polling

Figure 6: Call Flow: Successful Initial Certification Request with Polling



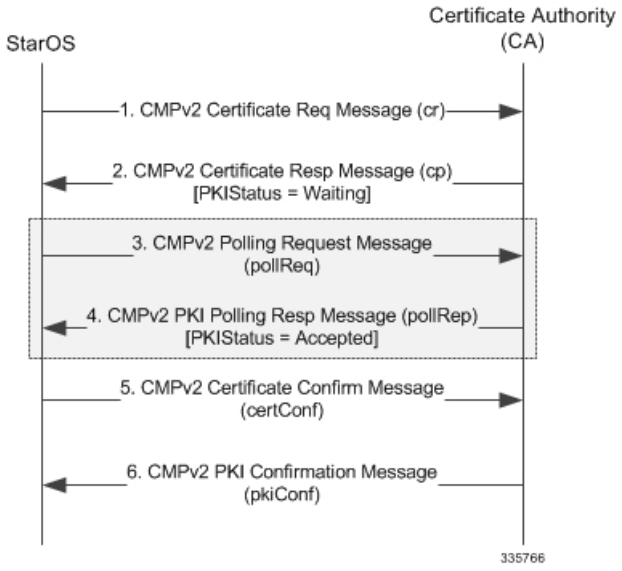
Enrollment Request

Figure 7: Call Flow: Successful Enrollment Request



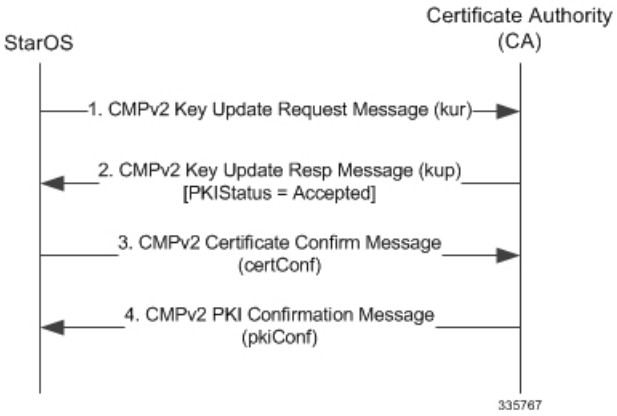
Enrollment Request with Polling

Figure 8: Call Flow: Successful Enrollment Request with Polling



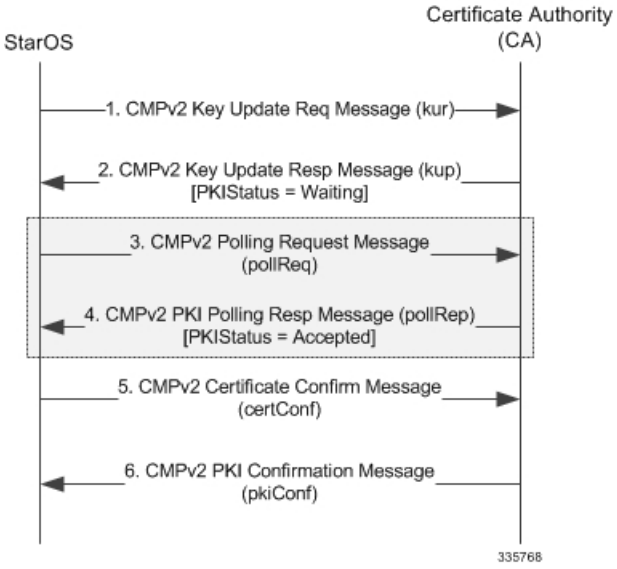
Certificate Update (Manual and Auto)

Figure 9: Call Flow: Successful Certificate Update



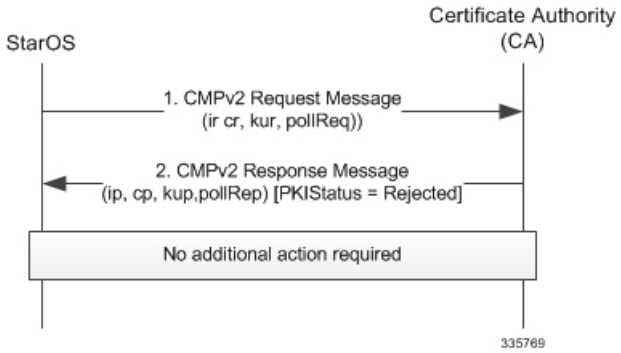
Certificate Update (Manual and Auto) with Polling

Figure 10: Call Flow: Successful Certificate Update with Polling



Failure Response Handling (ip/cp/kup/pollRep)

Figure 11: Call Flow: Failure Response Handling



CLI Commands

Important The commands described below appear in the CLI for this release. However, they have not been qualified for use with any current Cisco StarOS gateway products.

Exec Mode Commands

cmp initialize modulus

Triggers an Initial Certification Request (CR) after generating a public and private key pair, as well as an X.509 certificate to be included in the CR.

cmp intialize modulus *mod_type* **cert-name** *name* **subject-name** *"subject_string"* **ca-psk** *key* **ca-root** *ca_name* **ca-url** *url*

Refer to the *Command Line Interface Reference* for a complete description of this command and its keywords.

cmp enroll current-cert

Triggers a Certification Request (CR) after generating a public and private key pair, as well as an X.509 certificate to be included in the CR for a second certificate from the same Certificate Authority (CA).

cmp enroll current-cert *old-cert-name* **modulus** *mod_type* **subject-name** *"subject_string"* **cert-name** *name* **ca-root** *ca_name* **ca-url** *url*

Refer to the *Command Line Interface Reference* for a complete description of this command and its keywords.

cmp update current-cert

Triggers a Key Update Request after generating a public and private key pair, as well an X.509 certificate to be included in the Key Update Request for a certificate that is about to expire. This is a Certificate Management Protocol v2 command.

cmp update current-cert *old-cert-name* **modulus** *mod_type* **ca-root** *ca_name* **ca-url** *url*

Refer to the *Command Line Interface Reference* for a complete description of this command and its keywords.

cmp fetch current-cert -name

This command is only applicable for the ASR 9000 platform. CMPv2 operations are performed only on one Virtual Services Module (VSM) in the chassis. The certificates along with the private key file and the root certificate are stored on the supervisor card. When invoked on other VSMs in the chassis, this command reads the certificate, private key and the root certificate from the supervisor card.

cmp fetch current-cert *old-cert-name* **ca-root** *ca_name*

Refer to the *Command Line Interface Reference* for a complete description of this command and its keywords.

cmp poll cert-name

Triggers a pollReq for the specified certificate.

cmp poll current-cert *old-cert-name*

Global Configuration Mode Commands

cmp auto-fetch

Use this command to add a fetch configuration for each certificate for which automatic update is required. This is a Certificate Management Protocol v2 command.

cmp auto-fetch current-name *cert_name* **ca-root** *ca_name* **time** *days*

Refer to the *Command Line Interface Reference* for a complete description of this command and its keywords.

cmp cert-store location

Use this command to add a file location on /flash disk where the certificates and private keys will be stored. This is a Certificate Management Protocol v2 command.

cmp cert-store location *pathname* **[key reuse]**

Refer to the *Command Line Interface Reference* for a complete description of this command and its keywords.

cmp cert-trap time

Defines when an SNMP MIB certificate expiry trap should be sent as the number of hours before expiration.

cmp cert-trap time *hours*

Refer to the *Command Line Interface Reference* for a complete description of this command and its keywords.

show and clear Commands

show cmp outstanding-req

Displays details regarding outstanding Certificate Management Protocol v2 requests.

show cmp outstanding-req

Refer to the *Statistics and Counters Reference* for a description of the information output by this command.

show cmp statistics

Displays statistics related to Certificate Management Protocol v2 functions.

show cmp statistics

Refer to the *Statistics and Counters Reference* for a description of the information output by this command.

show cmp history

Displays historical information for the last 100 Certificate Management Protocol v2 transactions.

show cmp history

Refer to the *Statistics and Counters Reference* for a description of the information output by this command.

clear cmp cert-name

Clears information stored for the specified IPSec Certificate Management Protocol v2 (CMPv2) certificate.

clear cmp cert-name *cert_name*

Refer to the *Command Line Interface Reference* for a complete description of this command and its keywords.

clear cmp statistics

Clears statistics for IPSec Certificate Management Protocol v2 (CMPv2) certificates.

clear cmp statistics

Refer to the *Command Line Interface Reference* for a complete description of this command and its keywords.

Online Certificate Status Protocol (OCSP)

Overview

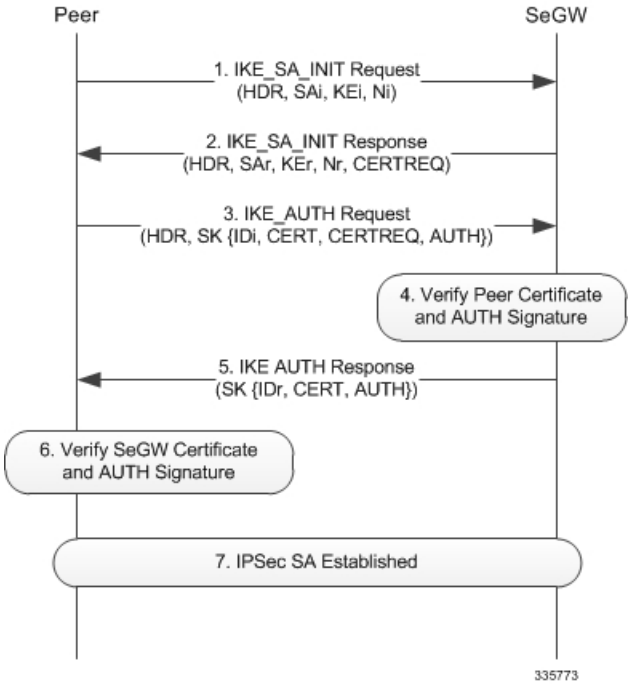
Certificates are used to establish peer identity. A certificate is issued by a trusted CA for a limited period. The validity period is an integral part of the signed certificate. Gateways exchanging certificates for establishing identity and trust check the certificate validity during the transaction. A certificate can be revoked at any instance of time (Well before the expiry of the certificate validity period). It is therefore very important to know the status of a certificate.

Online Certificate Status Protocol (OCSP) provides facility to obtain timely information on the status of a certificate (RFC 2560). OCSP messages are exchanged between a gateway and an OCSP responder during a certificate transaction. The responder immediately provides the current status of the presented certificate. The status can be good, revoked or unknown. The gateway can then proceed based on the response.

Deployment Scenarios

OCSP responders may be part of the CA/RA server or can be a separate entity authorized by the CA. The security gateway requires connectivity to this responder for status information.

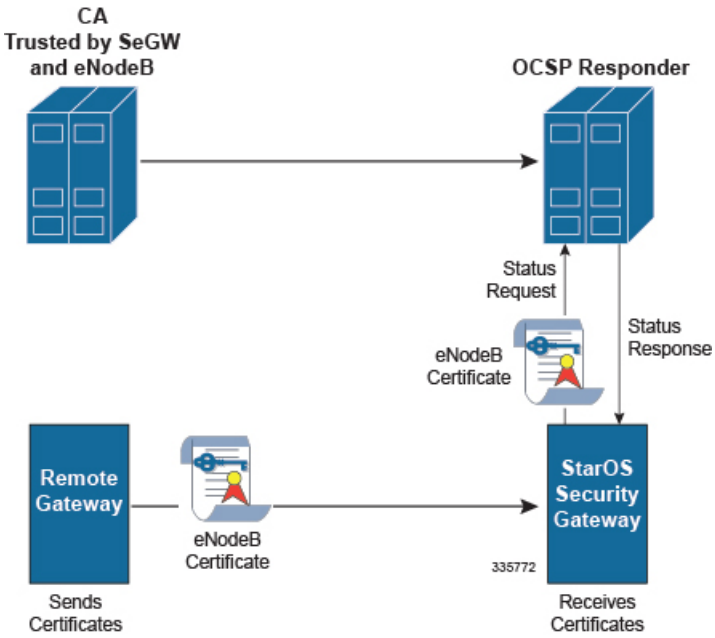
Figure 12: Call Flow: IKE Exchange



When the remote gateway presents a certificate, the security gateway forwards this certificate to the OCSP responder and queries it for the revocation status. The OCSP responder replies with the corresponding status information.

In IKE exchange (During the AUTH phase) the remote certificate is present in the CERT payload of the IKE message.

Figure 13: OCSP Status Request



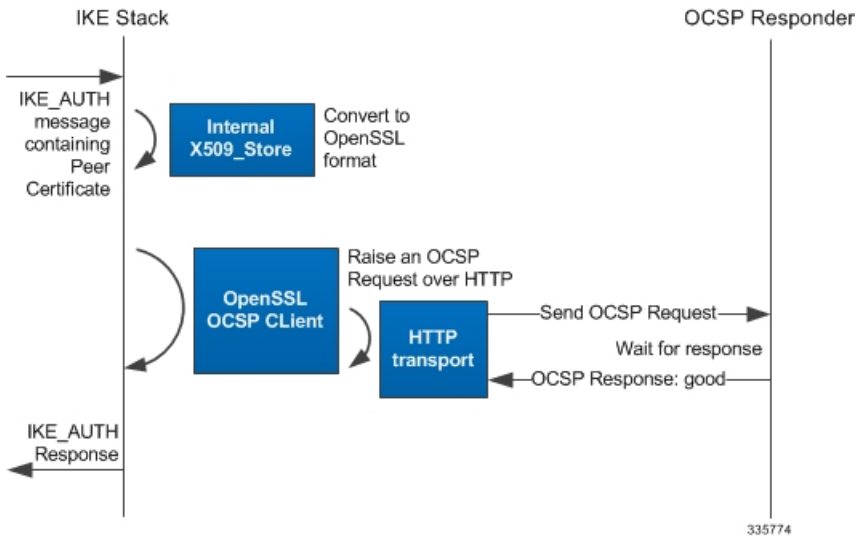
The security gateway passes this certificate along with its issuer certificate (trusted by security gateway) to the OCSP responder. IKE exchange is suspended (after step 3) until the response from the OCSP responder arrives. The OCSP request is initiated only when the presented certificate has the OCSP responder URL. If the URL is absent the OCSP request is not initiated.

If an OCSP response fails or if there is any error while contacting the responder, the certificate is validated with the CRL. Authentication is failed if an error is encountered while verifying with OCSP and or via a Certificate Revocation List (CRL).

Call Flows

Successful OCSP Response

Figure 14: Call Flow: Successful OCSP Response

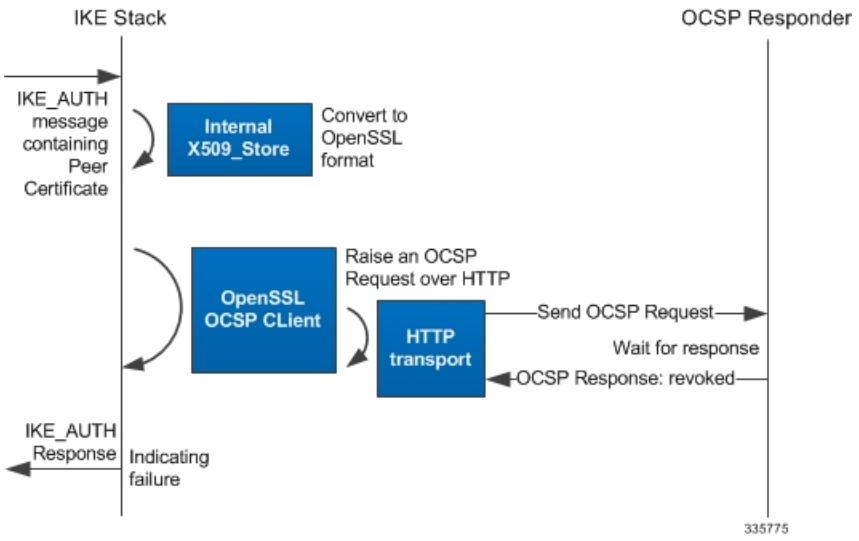


The peer certificate is obtained as CERT payload in the IKE message. The received certificate is converted to the OpenSSL format. This certificate is then passed to the OpenSSL OCSP client along with the X509_STORE to from an OCSP request. A connection to the OCSP responder is established and the request is sent.

On receipt of the response the IKE_AUTH transaction continues.

Revoked OCSP Response

Figure 15: Call Flow: Revoked OCSP Response



In this case fallback to CRL would be implemented for validating the user certificate. If this fails then the IKE_AUTH is aborted and a notification message is sent indicating authentication failure.

External Interface

The OCSP client to the OCSP responder interaction occurs over HTTP. A TCP socket connection is established to the OCSP responder. This connection is taken down once the OCSP response is received. The connection is also taken down as part of the cleanup after the setup timer expires.

CLI Commands

Important The commands described below appear in the CLI for this release. However, they have not been qualified for use with any current Cisco StarOS gateway products.

Context Configuration Mode

OCSP must be enabled in a crypto map or crypto template.

For a crypto map the configuration sequence is:

```
configure
  context ctxt_name
    crypto map template_name { ikev2-ipv4 | ikev2-ipv6 }
      ocsp [ nonce ]
```

For a crypto template the configuration sequence is:

```
configure
  context ctxt_name
    crypto template template_name ikev2-dynamic
      ocsp [ nonce ]
```

Refer to the *Command Line Interface Reference* for a complete description of these commands and their keywords.

CRL Fetching

Overview

CRLs (Certificate Revocation Lists) are issued periodically by the CA. This list contains the serial number of all the certificates that are revoked. An operator can verify the status of a certificate using a CRL. A CRL can be fetched via LDAPv3 from a CRL issuer (Trusted by CA).

When configured, this function also re-fetches the CRL once it expires in the cache. If the CRL is obtained from a CRL Distribution Point (CDP), StarOS defers the CRL fetch until the tunnel is established.

The CDP extension is read from the certificate for all protocols including HTTP, FTP, LDAPv3 and.CDP File.

StarOS initiates a CRL download in the following scenarios:

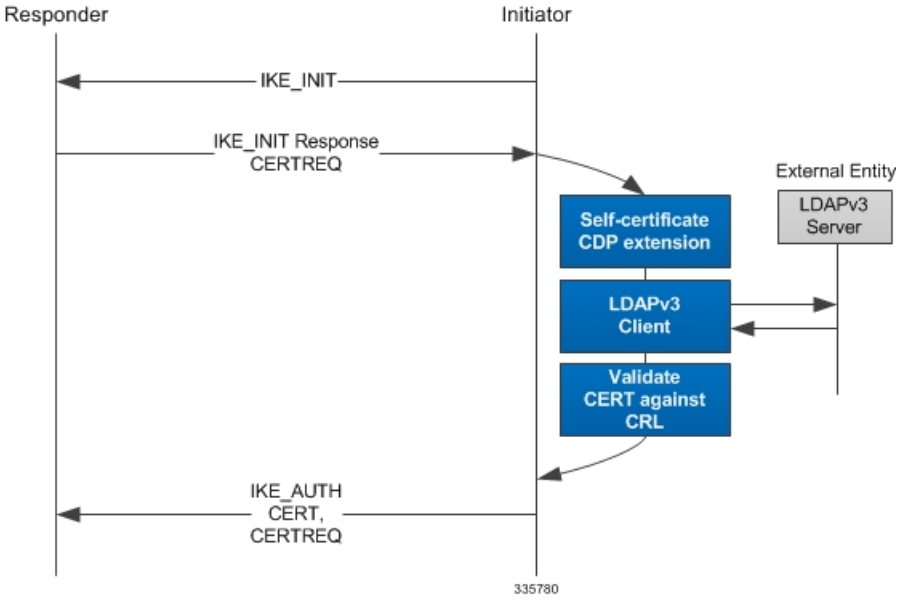
- User configuration via the CLI binds the CRL to a crypto map or template.
- During tunnel establishment:
  - The self-certificate CDP extension is used to download its latest CRL.
  - The CDP extension in the peer certificate is used to download its latest CRL.
- If the CRL (downloaded via CLI) expires during the refresh period (user configurable) a new fetch is triggered. If the CRL is obtained from the CDP extension, the fetch is deferred until tunnel establishment using the certificate.

CRL Downloads

Download from CDP Extension of Self-certificate

The following diagram illustrates the downloading of CRL from the self-certificate CDP extension (if present) at the tunnel creation.

Figure 16: Call Flow: CRL Download from CDP Extension of Self-certificate

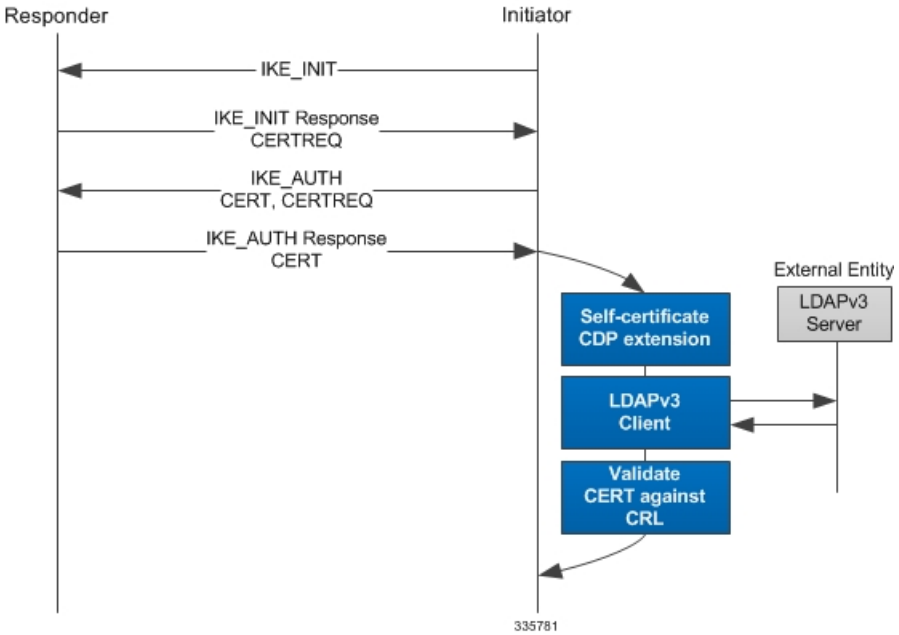


The certificate is then verified against the CRL before it is sent in the CERT payload of the IKE_AUTH message.

Download from CDP Extension of Peer Certificate

The following diagram illustrates peer certificate validations against CRLs. The CRL is fetched based on its CDP extension.

Figure 17: Call Flow: CRL Download from CDP Extension of Peer Certificate



The peer certificate is then verified against the CRL based on its status the IKE_AUTH proceeds.

CLI Commands

Important The commands described below appear in the CLI for this release. However, they have not been qualified for use with any current Cisco StarOS gateway products.

Global Configuration Mode

ca-crl name

This command configures the name and URL path of a Certificate Authority-Certificate Revocation List (CA-CRL).

The configuration sequence is as follows:

```
configure
   ca-crl name name { der | pem } { url url }
    end
```

url supports file pathname, TFTP, FTP, SFTP, HTTP and LDAP protocols.

Refer to the *Command Line Interface Reference* for a complete description of these commands and their keywords.

Context Configuration Mode

ca-crl list

This command is used to bind a CA-CRL to a crypto map or template.

For a crypto map the configuration sequence is:

```
configure
   context ctxt_name
      crypto map template_name { ikev2-ipv4 | ikev2-ipv6 }
         ca-crl list
            ca-crl-name
            end
```

For a crypto template the configuration sequence is:

```
configure
   context ctxt_name
      crypto template template_name ikev2-dynamic
         ca-crl list
            ca-crl name
            end
```

Refer to the *Command Line Interface Reference* for a complete description of these commands and their keywords.

show Commands

This command displays information for Certificate Authority (CA) Certificate Revocation Lists (CRLs) on this system.

show ca-crl { all | name *name* **}**

Refer to the *Statistics and Counters Reference* for a description of the information output by this command.