SCTE•ISBE CABLE-TEC EXPO® 2020 VIRTUAL EXPERIENCE » OCTOBER 12-15

2020 Fall Technical Forum SCTE·ISBE • NCTA • CABLELABS®

A Flexible and Scalable Architecture for Over-the-Air Credentials Provisioning

A Technical Paper prepared for SCTE•ISBE by

Alexander Medvinsky
Engineering Fellow
CommScope
6450 Sequence Dr., San Diego CA 92121
+1 858-404-2367
sasha.medvinsky@commscope.com

Dr. Tat Chan
Distinguished System Engineer
CommScope
6450 Sequence Dr., San Diego CA 92121
+1 858-404-3252
tat.chan@commscope.com

Dr. Xin Qiu
Senior Director of Engineering
CommScope
6450 Sequence Dr., San Diego CA 92121
+1 858-404-4212
xin.qiu@commscope.com

Jason Pasion
Senior Manager of Software Engineering
CommScope
6450 Sequence Dr., San Diego CA 92121
+1 858-404-2241
jason.pasion@commscope.com

Table of Contents

List of Figures

1. Introduction

During its lifetime, a device may need to be updated for a variety of reasons. New network access mechanisms or new types of applications and services may be introduced. This can mean that new device digital identities will need to be installed in already deployed devices to enable such new use cases. Examples of new digital identities include new DRM or conditional access credentials for new sources of content, IoT device certificates, credentials for a new copy protection interface such as DTCPv1, DTCPv2, HDCP 1.x, HDCP 2.x, etc.

Provisioning of credentials into devices already connected to the Internet and deployed to individual subscriber homes cannot rely on network or perimeter security. Even when devices are in an enterprise network or in a network operator's domain, it is still prudent to deploy "defense in depth" by providing end-to-end authentication and encryption all the way to the target device, in addition to any perimeter security such as firewalls, IP address filtering and port mapping. Each device and credentials provisioning server need a well-secured root of trust, delivery of credentials should be secured end-to-end and protected against a variety of network-based attacks.

A provisioning system handling different types of credentials has to support a variety of authorization models for different network operators and content providers. Different authorization interfaces are utilized to validate that a legitimate authorized device is being provisioned and that it belongs to a legitimate subscriber authorized for new credentials.

A credentials provisioning system may require a high degree of scalability for large populations of subscribers of premium content or for IoT appliances. Millions of devices may need to be provisioned with new credentials in a relatively short period of time. The worst-case scalability scenario is probably when a DRM or conditional access system is compromised, and every subscriber requires a new set of credentials within a short time period.

This paper describes an architecture of the CommScope credentials provisioning system called Online PKI Update System (OPUS) that has evolved over 10+ years in order to handle a variety of operator-specific and DRM-specific requirements with reliability, flexibility and scalability in mind.

1.1. OPUS System Overview

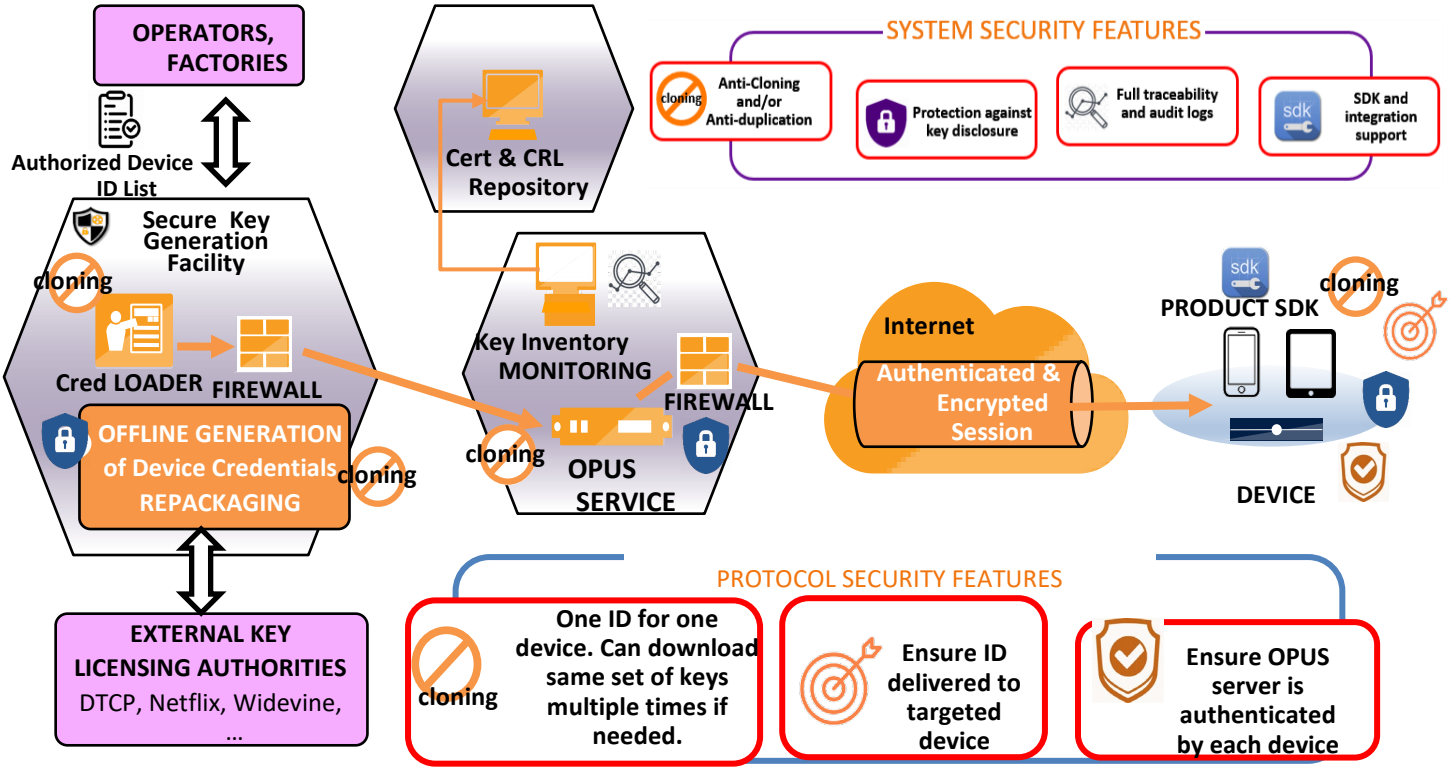


Figure 1 - OPUS System Overview

The OPUS service is accessible by provisioning clients via the Internet. It is protected from malicious attacks by means of a firewall, use of cryptographic specialized hardware, frequent security scanning and patching and other generally recommended security practices. The OPUS service consists of a cluster of frontend machines that are Internet facing to receive and pre-process provisioning requests. Sensitive and secure operations are performed by the backend servers, where the secure database and hardware security modules (HSMs) are located. Additional firewall is installed between the Frontend and Backend to provide additional protection. A hardware-based load balancer is utilized to distribute the workload among the cluster of frontend/backend machine pairs. Device credentials are generated and packaged in an offline secure key generation facility to provide end-to-end security from generation or re-packaging to final consumption on the target device.

There are in general two categories of credentials. In the first category, the credentials are generated locally in the secure key generation facility. It is commonly recommended to generate secret/private keys inside the device itself. However, the quality of a random number generator is highly variable across various device categories and developers that are not skilled in security could utilize a poor random number source such as C functions rand() and srand(). Even Linux random number source from /dev/random and /dev/urandom are not considered to be a high-quality random number source and may not be adequate for cryptographic use. Therefore we chose to generate cryptographic material with an HSM inside the secure Key Generation Facility so as to guarantee a sufficient degree of randomness for device credentials. The generated credentials are secured in both the offline Key Generation Facility and

during the transport to the OPUS server and end device so that the risk of compromise prior to the credentials reaching a target device is minimal.

In the second category, the credentials are provided by external parties. The key generation facility in that case only re-packages the credentials to the correct format and applies additional encryption. In some cases, key generation or re-packaging may be based on an authorized device ID list provided by third party such as the Network Operators or Factories. Credentials generated or re-packaged are loaded to the OPUS server through a Credentials Loader application.

The OPUS system utilizes a proprietary request and response message protocol to communicate with provisioning clients. The protocol was evolved over time to provide:

- Flexibility of provisioning any type of device credentials, including but not limited to X.509 device certificates.
- Support for a variety of authorization models some of which that occurred in practice are described in this paper.
- Session security at the application layer with low performance overhead, not requiring additional session setup using a protocol such as TLS or IPsec. Especially considering that each session with an individual device is very short-lived, just for delivery of a single set of device credentials.
- Seamless integration with load balancers, simplified since there is no session setup (OPUS server is stateless)
- Flexibility in terms of adding new cryptographic algorithms

To simplify the integration process, typically an OPUS SDK is built by the CommScope OPUS team and provided to a client device software team. The OPUS SDK handles all the OPUS related processing, including cryptographic operations. This SDK allows fast and easy integration with client teams utilizing the OPUS service and at the same time ensures that the security features on the client side are implemented as designed.

The OPUS solution provides many features crucial to a credential provisioning system, including anti-cloning, strong key protection, reporting, and easy integration. The system is configurable to safely allow the same credential to be downloaded multiple times by the same device in the legitimate cases where the credential on the device may be lost or corrupted. Typically, this is allowed only when that credential can be securely bound to a specific device as was the case in a couple of use cases described in this paper.

In terms of key protection, end-to-end encryption is used in most cases such that device keys are encrypted from generation/re-packaging and decrypted only at the final device. Unique per-device encryption is utilized whenever it is available. In the cases where global encryption is applied to the delivery of device credentials, additional session-based encryption is added by the OPUS server to mitigate the risks of cloning.

Reporting is another important aspect in provisioning. OPUS system maintains logs from key generation/packaging to the actual provisioning transactions, allowing usage reports to be created that may be needed by Network Operators. The logs are crucial for troubleshooting and debugging issues associated with the device credentials provisioning. There is an inventory monitoring service in the system which can generate key inventory reports. When key inventory drops below a preset threshold, a trigger will be created so that responsible parties will be alerted, and more data will be generated or externally acquired and repackaged accordingly.

The OPUS system provides a flexible and scalable solution for secure identity provisioning for devices in the field. In the following, many actual use cases will be discussed in more detail.

2. Existing OPUS Use Cases

2.1. Provisioning of Credentials based on Operator and Model Authorization

The following is a common use of OPUS for field provisioning of DRM keys into deployed digital set-tops and other secure video rendering devices. Each deployed device has been provisioned with a factory-installed unique credential. Most common type of credential in use is an X.509 device certificate chain, issued by either the network operator or device manufacturer.

Netflix credentials in particular require that an operator signs a business agreement with Netflix and opens a corresponding device account for a specific device model and region. An example of this process is documented here: https://openconnect.netflix.com/en/#what-is-open-connect. This process sometimes takes much longer than anticipated and is completed after a particular device model is already in mass production. Therefore, it becomes necessary to provide a secure field provisioning interface to download unique per-device Netflix keys online.[1]

Another common use case is provisioning of Widevine DRM and Attestation Keys into Android and Android TV devices.[2] These Android credentials require for a device vendor to set up a per-model account with Google and when Widevine keys are utilized in conjunction with a Netflix service, then separate accounts for each network operator are also required.

Yet another use case when DRM keys may be provisioned into device following their manufacture is an introduction of a brand-new DRM or copy protection system. DTCP version 2 was introduced in 2017[3] and adds a higher level of content protection which requires a Trusted Execution Environment inside each device and makes use of larger key sizes and stronger cryptographic algorithms than what is defined in DTCP version 1. DTCP version 2 is more suitable for protection of premium 4K and HDR content than DTCP version 1. By the year 2017, there was already a population of secure digital set-top boxes and digital TVs with a sufficient level of HW security and these devices are capable of supporting DTCPv2 and protecting the streaming of 4K and HDR content from a digital set-top box to a digital TV or between server and client set-top boxes. Again, this is a use case for OPUS to update qualified devices with the corresponding DTCPv2 keys.

The following sequence diagram demonstrates the use of OPUS for provisioning of new DRM credentials into each device:

[1] Linux-based and Windows-based Netflix clients are based on Microsoft PlayReady DRM https://www.lightreading.com/cable-video/netflix-taps-playready-as-primary-drm/d/d-id/677491 but still require additional Netflix-specific device keys provisioned into each device. PlayReady DRM keys do not require per-operator Netflix accounts and can be provisioned in the factory ahead of time.
[2] See https://developers.google.com/android-partner/guide/keybox and https://developer.android.com/training/articles/security-key-attestation.
[3] See https://www.dtcp.com/dtcp.aspx

Figure 2 – Provisioning of Credentials Based on Operator and Model Authorization

The above sequence of credentials provisioning steps are as follows:

1. Network operator optionally provides a list of devices requiring an upgrade and new DRM credentials. If this list is incomplete, then this whole sequence may need to be iterated multiple times.

   In majority of cases the operator did not require this list and all devices of a particular model and belonging to a particular operator were allowed to upgrade. But there were some cases when the operator wanted us to upgrade only specific devices based on their list of device identifiers. A device identifier here could be a SOC ID, MAC Address or a Serial Number.

   New types of device identifiers may be supported in the future on a case-by-case basis. Most of the time a new type of device identifier can be supported with no software changes to the OPUS system. A device identifier which is a function of a device certificate (e.g., hash or fingerprint of the device certificate) may be introduced in the future as an OPUS enhancement.

2. Generate the new sets of device credentials (e.g., issue X.509 certificates). Or in other cases which we had to commonly support – obtain and then re-package and re-encrypt 3rd party DRM credentials e.g., Netflix, Google Widevine or Google Attestation Keys. This may require a SOC-specific format that can be later processed by the chip inside the trusted execution environment (TEE).

3. Encryption may utilize either a global HW key known to the SOC TEE environment or unique per-SOC encryption whenever the SOC ID is known in advance. Decryption is often required to occur inside a TEE on the SOC, especially when the new DRM is expected to handle 4K or HDR high value content.

   When the SOC ID is known, it may be utilized to either derive a unique per-SOC key from a global HW key and SOC ID, or utilized to look up a unique One Time Programmable (OTP) key for that SOC in some internal or external database. For CommScope's past utilization of OPUS – we had to handle all of these use cases.

4. Load the generated new DRM credentials onto the OPUS server which is accessible online by devices that are being upgraded. Device credentials were generated in an offline air-gapped facility and so this transfer may be done by a human operator with removable media.

5. At some point prior to a new device model starting to submit requests to OPUS, an OPUS administrator must obtain and install the device root CA and device sub-CA that will be trusted by the OPUS server for submitting requests for device credentials. There can be multiple trusted CA certificate chains configured for the same (operator ID, credentials type) combination since a particular operator may have several different device models with different factory-installed device certificates from different issuers that all require the same new credentials to be downloaded in the field.

   This process needs to be repeated for each new credential type – even if it is the same device model requesting a new credential. For more flexible authorization rules, an operator could require the same device model to utilize a different certificate chain for downloading a different type of device credential. There shouldn't be a limit on the number of CAs installed on the OPUS server, covering any number of authorized device models and types of credentials.

Besides configuring the list of trusted CA certificates, there are other miscellaneous configuration parameters that should be set for each credential type. For example, the type of the key agreement algorithm (e.g., Diffie-Hellman or Elliptic Curve Diffie-Hellman), key size or Elliptic Curve ID, etc.

This step is required for all use cases but for simplicity is not shown on subsequent diagrams.

6. A project manager, possibly a member of customer's operations team, is given a go ahead that the new credentials are now ready to be downloaded.

7. The project manager coordinates a device update to prepare devices for a new credentials download. In this example, device update involves an authenticated software update where the new software has the logic to connect to the OPUS server and request the new credentials (in step 8 below). Well ahead of this step CommScope builds an OPUS client SDK for each different device platform to make it easier to complete this software update.

   In other cases, devices may already have the latest software capable of connecting to the OPUS server but were waiting for some sort of a trigger such as e.g., a TR-069 configuration message. It is not advisable to wait until the end user tries to access content requiring the new DRM and then begin the credentials download – to avoid any potential delays that are visible to the user.

   Section 3 on scalability shows that OPUS is really a cluster of servers capable of very large throughput, but still – there is a lot of unpredictability in the Internet performance and it is best to install DRM credentials in advance of the need to use them.

8. If the software update is required and has not yet taken place, the software is downloaded, verified and installed during this step.

   A software update is required under a variety of circumstances such as:
   - A credentials update was not anticipated during the initial device manufacturing and the corresponding interface to the OPUS server was not implemented at that time.
   - Operator's device authorization mechanisms for new credential updates were not defined at the time of device production or had undergone some changes since then.
   - A field update for new credentials was anticipated prior to manufacture, but a strategic decision was made to accelerate device production schedule and defer the interface to download credentials to a future software update.
   - Any software in the device that is related to the new credentials incurs a per-device license fee and is therefore deferred to a software update and only for specific end users that subscribe to a specific service.

   But in some cases a device may already have the right software to interface to OPUS and this step could be replaced with a trigger message to begin a credentials' download.

   Typically, after the software update is validated and installed, the device will reboot, and restart and it may undergo secure boot (not shown on the figure).

9. Device submits a request for new DRM credentials. This request must be authenticated, and we normally rely on factory-installed certificate and private key for that purpose. This device should have implemented secure boot such that all software running in the device is authenticated. It is also recommended to utilize a TEE for this purpose.

Over the years, we had to handle some exceptions where for example a device with no public key credentials was being upgraded with a DRM application that requires a unique device certificate. Sometimes a device has credentials that are by contract restricted to protect only a specific interface (e.g., DTCP-IP or HDCP copy protection or credentials for a specific DRM) and so still – there were no device credentials available to authenticate a request to OPUS.

Under those circumstances, a software update to a device (in steps 6 & 7) included a software-protected signing key which can be mathematically encoded using whitebox techniques and a software protected decryption key which can also be whitebox-encoded. Here, a device signs the request with its software protected signing key if a more secure option is not available. And associated device certificate issued to the corresponding public key is also attached.

This message includes a unique device identifier as a separate parameter and may also be included in the device certificate attached to the request. Device identifier is validated against the authorized device ID list (see the next step).

This request also includes a key agreement public key of the device in order to establish a one-time session encryption key – in addition to encryption that already occurred offline in step 3. The device generates its key agreement public/private keypair and the public key is included in this request message.

Typical key agreement algorithms include Diffie-Hellman (DH) and Elliptic Curve Diffie-Hellman (ECDH). OPUS protocol is configurable and allows for introduction of new key agreement algorithms as needed.

10. Here, the request is authenticated by checking certificate chain and signature of the requesting device. Authorization is also verified using multiple configurable options:
    - Check if Operator ID in the request is authorized for this type of credential
    - Check if a particular device model is authorized for this type of a credential
    - Check if this specific device has its ID in the authorized device ID list
    - Check a CRL to make sure that the factory-installed certificate is not revoked. There may be multiple CRLs, but OPUS will check a specific CRL corresponding to the End Device's certificate in the request. Not all device certificates include a crlDistributionPoint extension so the server may need to be configured with the corresponding URL manually.

11. After all authentication and authorization checks passed, the OPUS server will either look up a new device credential based on the authorized device ID, or if allowed by configuration, find the next unbound globally encrypted credential.

    The server will generate its own key agreement keypair and include its public key in the response that it is preparing. It will also generate a one-time session key based on server's key agreement private key and the client device's key agreement public key in the request message. This session key is utilized to add an extra layer of encryption, preventing the reuse of the same encrypted credentials in any other OPUS session.

    In the case that encryption in step 3 was done with a unique device key, the reuse or copying of those encrypted credentials in another device is already prevented and this extra layer of session key encryption may be skipped for improved performance and scalability.

12. The OPUS server adds its own signature and certificate chain and sends back a reply containing the server's key agreement public key and the encrypted device credentials.

13. Device receives the response, validates OPUS server's signature and certificate chain and then proceeds to decrypt, validate and install its new credentials. These new credentials should be decrypted inside a secure TEE environment and then re-encrypted using a device unique key to bind these new credentials to the specific device.

2.2. Provisioning Credentials Based on Proof of Subscription

Pay TV subscribers may lease or purchase a local DVR that will record subscriber's favorite digital content and this content can later be streamed to subscriber's secondary rendering devices, including additional client set-top boxes, mobile phones and tablets. This subscriber may want to watch content recorded on the main DVR on her tablet connected over a local Wi-Fi network while in a different part of the house from where the DVR is located. Alternatively, a subscriber would like to receive live or pre-recorded content directly to her tablet, cellphone or laptop via a content distribution network while traveling, away from home.

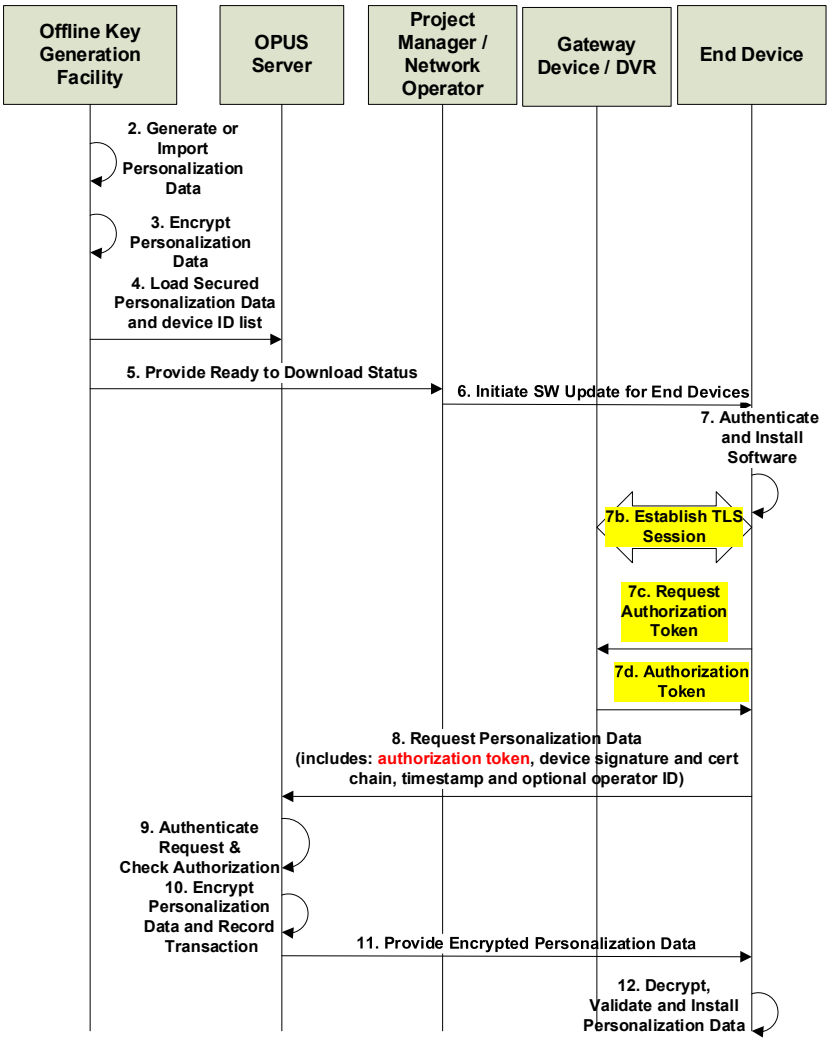


Figure 3 – Provisioning of Credentials Based on Proof of Subscription

The above sequence of credentials provisioning steps is similar to the sequence described in section 2.1 with the following differences associated with device authorization:

- Optional step 1 where an authorized device ID list is provided by the operator is not there.

- A subscriber leases a DVR from an operator and pays for an additional service that allows this subscriber to obtain DVR content or even content direct from the operator's content distribution network on some small number of additional personal devices authorized by the operator, including mobile phones, tablets, PCs, etc.

- To prove that this subscriber has a DVR and is authorized for additional content streaming or download services to a personal device, steps 7b, 7c and 7d were added:

- 7b: establish a 2-way authenticated TLS session. The end device and DVR both have pre-installed certificates that can be used for this purpose. DVR would typically be the TLS server and end device is the client.
- 7c: Device requests an authorization token
- 7d: DVR checks subscriber authorization for content sharing with additional devices and if authorized, signs and returns the authorization token. DVR is separately configured by the operator to enable this service.
    - Authorization step in the DVR may include checking the limit on the number of end devices or the DVR may have its own authorized device ID list received from the operator to make sure that this end device is authorized.

Some examples of additional credentials provisioned into the end device include DTCP-IP, X.509 device certificate, Netflix or Widevine. (Netflix and Widevine DRM are typically utilized to secure content delivery direct from the content provider or the network operator but not from the DVR.)

2.3. Provisioning of New Credentials with Offline Matching Existing Device ID

In some cases a network operator may decide to switch to a new Digital Rights Management system that requires new device credentials such as for example device X.509 certificate chain and private keys. Other DRM systems require their own proprietary DRM credentials in a proprietary format that may need to be installed after devices are already manufactured. In other cases, the operator completes their business agreements with a DRM provider after devices already start shipping and so are shipped lacking the necessary DRM credentials.

An operator may require that such a DRM upgrade is done for a specific set of devices and device IDs that are registered in their network and provides the manufacturer with an authorized device ID list requiring such an upgrade. A device identifier attached to the new device credentials may be for example a MAC Address and it may be included as an X.509 certificate subject name attribute as was the case in the CommScope's experience. Another example of provided device identifiers is a list of SOC identifiers which may be used to encrypt each new DRM credential with a SOC-specific symmetric key.

One way to address these use cases is to process an authorized device ID list in a secure offline facility where we have access to all the factory-provisioned device certificates. For each device ID in the authorized list we look up the original factory-provisioned device certificate and use the included public key to encrypt the newly generated DRM credentials of that device. The X.509 certificate that is part of the new credentials is signed in an offline facility using a Certificate Authority approved by the DRM provider and the device ID in the authorized list is included in the new certificate. Encryption is unique to a specific device and is end-to-end, providing a very high level of security. Clear device credentials are not exposed anywhere outside of the secure offline facility and the device itself.

Alternatively, a SOC ID may be used to look up or derive a SOC-specific and hardware-protected symmetric key and utilized to encrypt the newly generated device credentials. In this case, encryption will still be unique to a specific device and is end-to-end, providing the same high level of security.

CommScope has utilized the deployed OPUS system to perform such device upgrades for multiple (4+) operators which made a decision to switch their whole network from Digital Rights Management System #1 to Digital Rights Management System #2 or in other cases concluded their DRM agreements after devices were already manufactured and delivered to that operator. In each case, the DRM upgrade was successfully completed according to plan.

The following diagram focuses on a use case where each new credential is an X.509 certificate and private keys are encrypted using a public key from a factory-installed certificate:

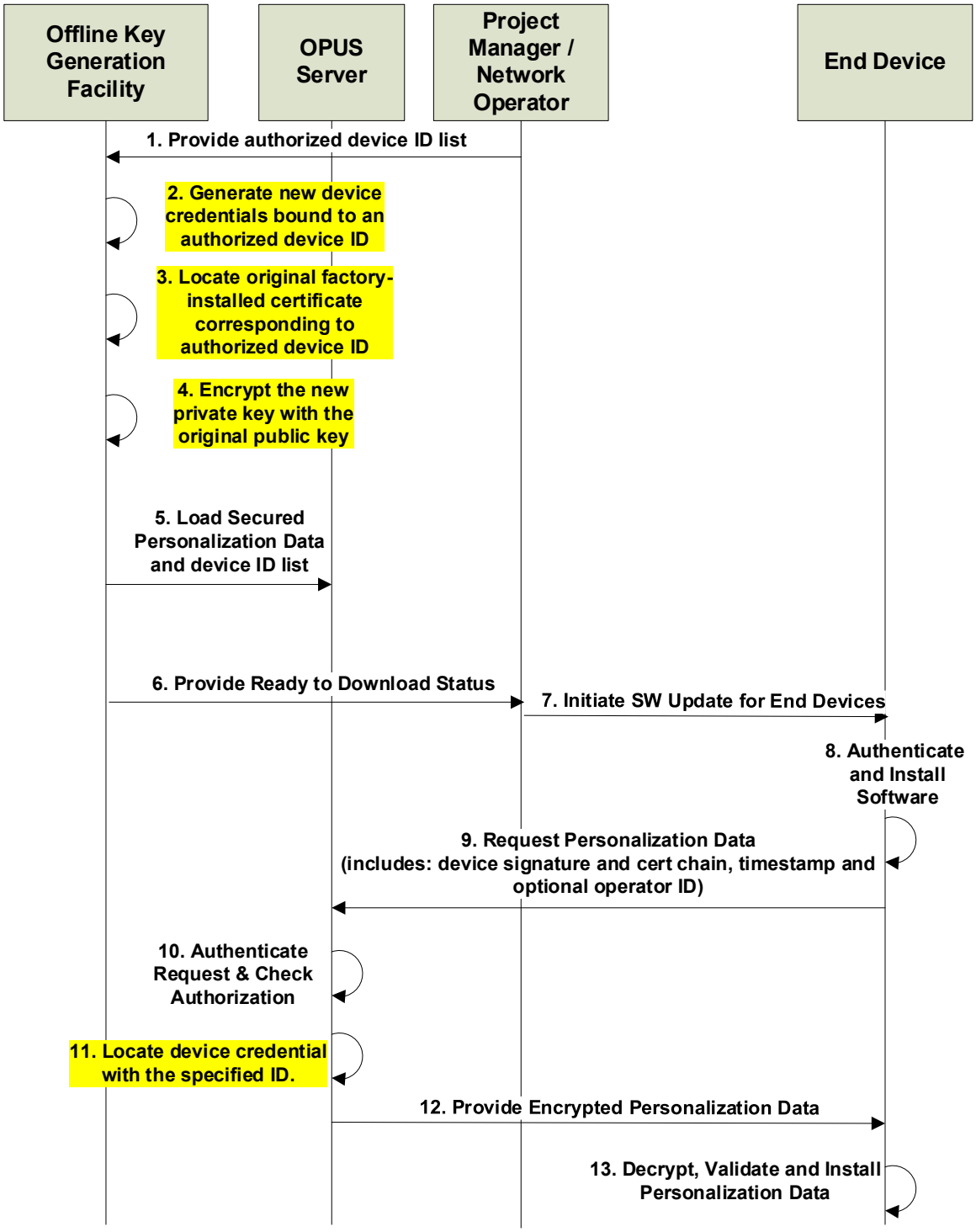


Figure 4: Provisioning of Credentials Matching Existing Device ID

1. Network operator provides a list of authorized device identifiers such as MAC Addresses that are required to be present in the new device certificates.

2. Generate the new device credentials consisting of a public/private keypair and a device certificate chain. The newly generated device certificate includes one of the authorized device identifiers on the list. Certificate is signed using a Certificate Authority that is authorized for use with the new DRM system.

3. Locate the original factory-installed certificate in an offline database. The original certificate may contain the same device identifier. If the original certificate contains a different identifier, then the authorized device ID list provided in step #1 needs to include pairs of (factory-cert-device-ID, new-cert-device-ID) for each device. That way, factory-cert-device-ID can be used to locate the original factory-installed device certificate. Private key corresponding to that factory-installed certificate is not needed.

4. A public key is extracted from the factory-installed certificate and utilized to encrypt the new private key, generated as part of the new device credentials for the new DRM. If RSA public keys are utilized, due to size limitations the RSA public key is utilized to indirectly "wrap" the larger RSA private key structure. A standard RSA wrapping mechanism is specified in [5].

5. Uniquely encrypted device credentials for each ID in the authorized device ID list are loaded to the OPUS server. Encryption was performed in an offline facility and therefore this step requires manual action such as transfer of the keys on removable media. Authorized device ID can be implicit – each authorized device ID is included in one of the uploaded device credentials.

6. A project manager, possibly a member of customer's operations team, is given a go ahead that the new credentials are now ready to be downloaded.

7. The project manager coordinates a device update to prepare devices for a new credentials download. This may be done via an authenticated software update to the end device. (See a more detailed description on step #6 in section 2.1.)

8. If device received a software update needed for the DRM credentials update in step #7, then the software update is authenticated in this step prior to installation.

9. The device submits a request for new credentials to the OPUS server. In this use case, the device includes the new device ID that it requires in the new credential and the request is signed using the factory-installed device certificate and private key.

10. The OPUS server verifies the signature and certificate chain of the request. Some authorization may be performed here, such as verifying the network operator's ID in the request to make sure the operator is authorized for this DRM update.

11. Locate the new device credentials matching the device ID in the request. This provides additional implicit authorization – devices that did not have their device ID included in the list provided during step #1 will not have their certificate loaded on the OPUS server. Either that means that device is not authorized for the DRM upgrade – or the operator simply missed providing the identifier of this device.

12. The encrypted device credentials located on the OPUS server are returned to the device in this step. It isn't necessary to apply additional session-based encryption here since the new credentials are encrypted for a specific target device. But the message is signed and anti-replay protection may be needed in the absence of a key agreement protocol such as DH or ECDH.

13. Device receives, verifies and installs its new credentials. Decryption and re-encryption are likely required in order to store the new DRM credentials using DRM-specific encryption. But in some cases it may be sufficient to save the encrypted private key in the exact format as it was received.

In practice, it turned out that it was very hard for operators provide a complete authorized device ID list in step #1 for all devices that required the DRM upgrade. Some devices may have their device identifier modified in repair. Or the operator may be missing some identifiers of newly shipped devices for which they have not yet received a shipping notice. In our practice, this is an iterative process and for each operator we received the authorized device ID list in 5-10 parts and so this whole process was repeated 5-10 times per operator.

One way to handle this upgrade is to have the OPUS server keep track of errors of requested device IDs that were not found in its list, deliver this list of missing IDs to the operator which would then determine if those are legitimate devices that were missed during the previous DRM upgrade iterations. And then repeat the whole process with the previously missing device IDs.

OPUS system has also been utilized in very similar use cases as the above with a few variations, where:

- The new DRM credentials are in a DRM-proprietary format and may not include X.509 device certificates
- Authorized device ID list consists of SOC IDs and DRM credentials are encrypted with a SOC-specific symmetric key instead of a public key from a factory-installed certificate

The sequence of entity interactions looks the same as in the above diagram with minor variations in the type of device identifiers and encryption that was deployed.

2.4. Provisioning of New Credentials with Offline Matching Existing Device ID for Wireless Devices

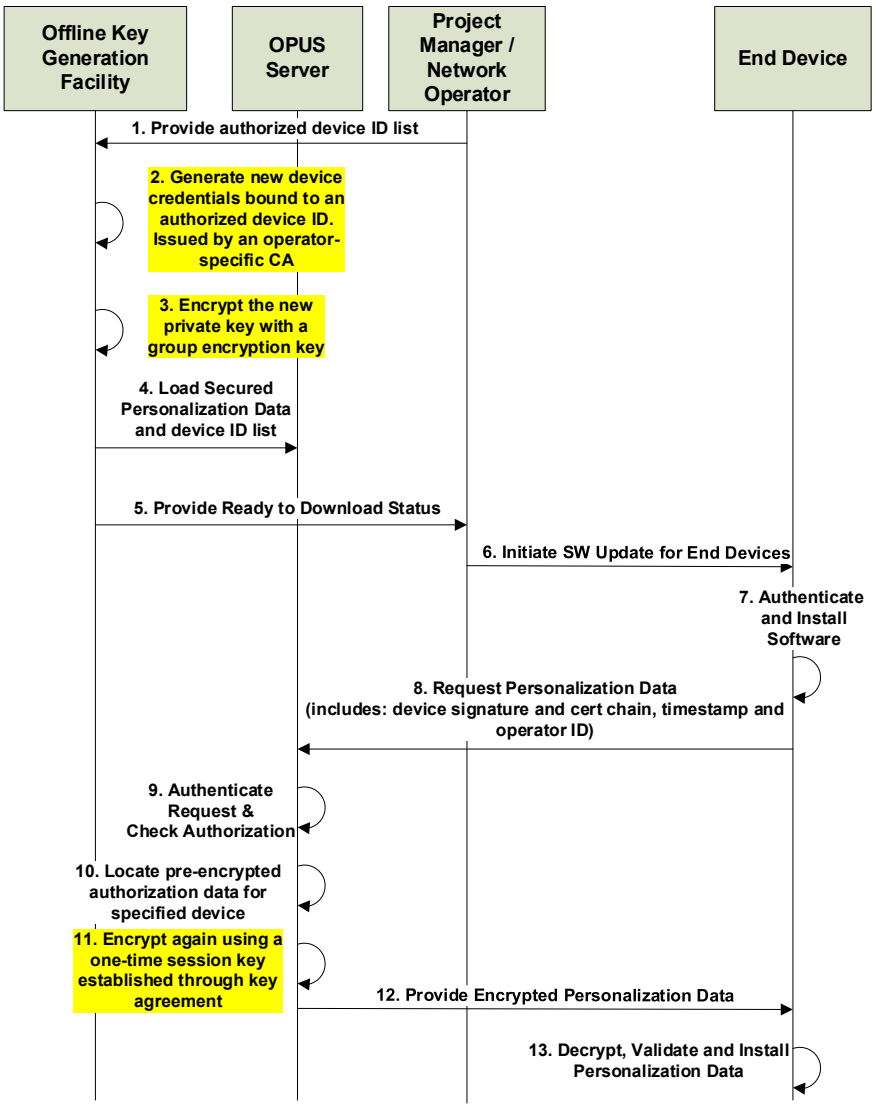
In some cases, a wireless network device such as a router, base station, radio point or radio point controller requires an X.509 device certificate that is utilized in a lower layer protocol security such as for Thread, IKE/IKEv2/IPsec or MACsec. In those cases, it is common to require a device identifier such as the MAC Address to be present in those certificates. Such certificates can be installed in the factory, except when an operator wants to install their own credentials that chain to an operator-specific Root CA.

A device manufacturer may not know network operator's identity prior to manufacture or equipment may change ownership between different operators when a portion of operator's network is sold to a new operator. Such use cases can also be addressed with OPUS providing field provisioning of new device credentials in a very similar manner as the use case described in section 2.3, but with some simplifications.

Encryption in the offline facility is done with a group key which doesn't require locating the original factory-installed certificate. Instead, the OPUS server performs a key agreement protocol to generate a unique session key for an extra layer of encryption thus preventing cloning attacks and reuse of the same encrypted credentials in multiple devices.

The use case below was in practice considered for a wireless network device upgrade, although the same OPUS profile can be applied to field provisioning of any kind of new credentials – when each new credential is targeted to a specific device and a list of the corresponding device identifiers is known in advance. It has some similarity to the use case describe in section 2.3, except that here the OPUS operator may not have access to the factory-provisioned device credentials.

This is illustrated in the following diagram:



The above sequence of credentials provisioning steps is described in more details below:

1. Project Manager or Network Operator provides an authorized device ID list (list of device identifiers for devices requiring new credentials). Each such list is associated with a specific Network Operator, used to determine which CA is used to issue the corresponding device certificates. In practice, this step is iterated periodically. For example, the project manager sends a list for all new devices shipped within a pre-determined time period.

2. Generate the new sets of device credentials (e.g. X.509 certificates) with the device identifiers provided in the list. The CA used to issue the certificates may be specific to the Network Operator.

3. Encrypt each of the device private keys using a group/model-specific symmetric key or a key derived from such a key.

4. Load the authorized device ID list and new credentials onto the OPUS server. The new credentials are identified by a Personalization Type ID that is specific to the type of credential and Network Operator combination. Since the credentials were generated in an offline air-gapped facility, data loading may be performed by a human operator using removable media.

5. The Project Manager or Network Operator is notified that the credentials have been loaded and therefore ready to be consumed by the devices on the list.

6. This step describes the process of the Project Manager or Network Operator triggering the software update to the device for requesting the new credential. This step is the same or similar to step 6 in the section 2.1.

7. In this step, the device verifies and installs the software update necessary to interface to the OPUS server, if it has not been done yet.

8. Similar to the previous use cases, the device submits a request to the OPUS server for new credential, identified by a Personalization Data Type ID. Since the new private key to be provisioned is globally encrypted, a unique session key will be created by the server (in a later step) to mitigate the risk of cloning. Therefore the device generates a key agreement key pair for the request and includes the key agreement public key in the request. In addition, the device also includes an Operator ID for additional validation by the server. The request includes a device signature and a factory-installed certificate.

9. Upon receiving the provisioning request from a device, the OPUS server verifies the signature on the device using the attached device certificate extracted from the request. The server makes sure that the device certificate used is chained to a pre-configured CA for that Personalization Data Type ID. Next, the server extracts the Device ID from the device certificate and verifies if it is on the authorized device ID list. The server would reject the request if any of the verification steps fail.

10. Next, the OPUS server looks up the credential pre-generated for that particular device, based on the Device ID. The credential includes the new device certificate and globally encrypted private key.

11. Since the private key is encrypted globally, OPUS server applies another layer of encryption on top of it using a session key derived from the key agreement algorithm. To do this, the OPUS server extracts the key agreement public key from the request, and generates its own key agreement key pair.

12. The OPUS server sends a response to the device including the new credential (the new device certificate and corresponding private key, which is now double encrypted with the session key), and the server's key agreement public key, needed for the device to derive the same one-time session key. The server also signs the response using its server certificate.

13. After receiving the response, the device verifies the signature of the message using the provided server certificate. The device also makes sure the server certificate chains to a trusted CA embedded in the provisioning software installed earlier. The device then derives the one-time session key using the provided server key agreement public key and its own private key. The device then removes both layers of encryption over the private key, utilizing the one-time session key and then the global encryption key. At this point the device may perform further validation to make sure the certificate is properly formatted and chained to the expected Root CA certificate. It also makes sure the private key and certificate match each other. After all the validation, the device can proceed to store the new credentials.

    To avoid cloning, the private key must be stored encrypted using a unique key tied to the device rather than using a global key. The device derives such a unique key from a global key and a device or SOC identifier. Or a unique device key may have already been provisioned (e.g. into secure OTP memory). The device uniquely encrypts the new private key and saves it into persistent storage along with the associated device certificate.

This use case has the security advantage that the signing CA is hosted in the offline key generation facility and not on OPUS. However, a drawback is that system response time may not satisfy the operational requirements imposed by the Network Operator. After the authorized device ID list is provided to the offline key generation facility, there will be a lead time for the new credentials to be available for a device to download. In some cases, Network Operator may impose a short turnaround time, say, one day, which may not be satisfied easily with the many steps (some manual) for this process. The next use case can be used in such scenarios.

2.5. Provisioning of New Credentials with Online Matching Existing Device ID

The approach for wireless device credentials provisioning that is described in section 2.4 was seriously considered but as it turned out – after an operator submits a new order for wireless devices, the new certificates have to be made available almost immediately. While it is a more secure approach in section 2.4 where none of the Certificate Authorities are operated online, it did not meet the practical performance constraints.

Therefore, in practice we addressed this use case with a different system architecture that enables the OPUS server to issue new operator-specific certificates. The OPUS server is sufficiently hardened where the certificate issuer's private key is protected in the HSM at all times and device private keys do not need to be exposed on the server. The OPUS server is also sufficiently hardened and is protected behind a firewall.

Just as in the previous section, this use case is not limited only to the wireless device credentials, although in practice that's where it was applied.

This use case is illustrated in the diagram below:

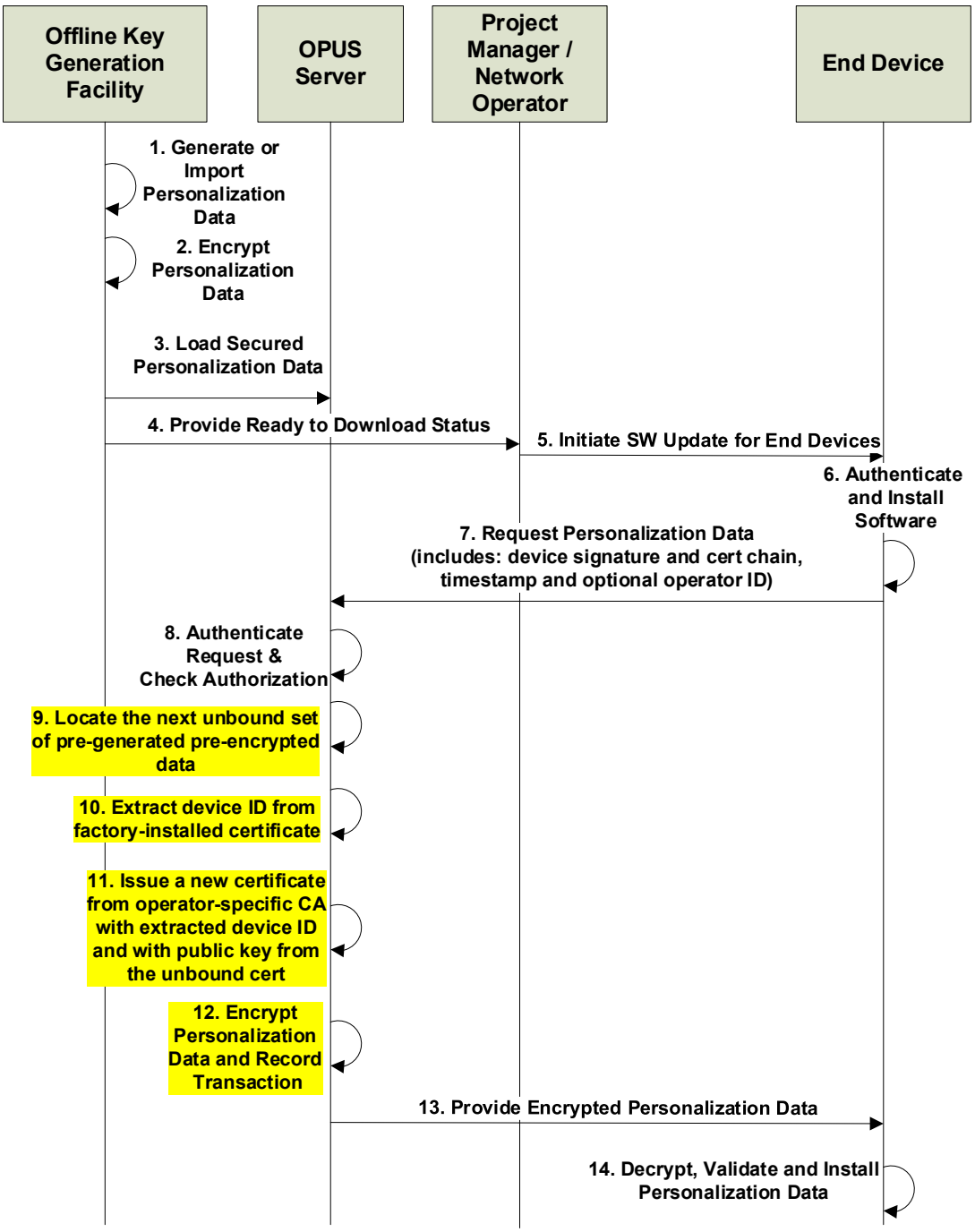


Figure 5 – Provisioning of Credentials Online Matching Existing Device ID

The above sequence of credentials provisioning steps is explained in the following:

1. A set of device credentials, called the Origin Credentials, are pre-generated in the offline key generation facility. Each record of Origin Credential includes a device private key and a corresponding Origin Certificate issued using an identifier that is not the final device identifier.

This is because the list of authorized device IDs is not yet known at this point. The generation may be triggered by production volume estimation provided by a Project Manager.

2. The device private key is encrypted just as in Section 2.4, using a group/model-specific symmetric key or a key derived from such a key. This allows the device private keys to be pre-encrypted end-to-end from generation to the device. The OPUS server does not need to have access to the clear device private key and therefore the exposure of the device private keys is minimized.

3. The Origin Credentials are loaded to the OPUS server. These credentials are identified by a Personalization Type ID that is specific to the type of credential but not necessarily specific to the Network Operator. Note that at this point, the credentials are not yet bound to a specific device.

4. The Project Manager or Network Operator is notified that the credentials have been loaded, and provisioning can proceed.

5. This step describes the process of the Project Manager or Network Operator triggering the software update to the device for requesting the new credential. This step is the same or similar to step 6 in the section 2.1.

6. In this step, the device verifies and installs the software update necessary to interface to the OPUS server, if it has not been done yet.

7. In this step, the device submits a request to the OPUS server similar to step 8 of section 2.4. Again, since the device private key is globally encrypted, a one-time session key will be used to add an additional layer of encryption. Note also that in this use case, the request must include the Operator ID, which will be used by the server to determine which CA to use for issuing the final certificate.

8. Upon receiving the request, the OPUS server authenticates the request and checks for authorization, similar to the previous case. However, as an authorized device ID list is not utilized here, the OPUS server only verifies that the factory-provisioned device certificate in the request is issued by a trusted CA pre-configured for this specific Personalization Data Type and for the specified Operator ID.

9. OPUS server retrieves the next available unbound set of Origin Credentials, which includes an Origin Certificate and a corresponding globally-encrypted device private key.

10. OPUS server then extracts the device ID from the factory-installed certificate in the request. This device ID will be used in the final certificate.

11. OPUS server issues a final certificate for the device. The final certificate has the extracted device ID in the subject name, and the public key extracted from the Origin Certificate. The final certificate is signed by the OPUS server using a CA determined by the Operator ID provided in the request. This final certificate is now bound to the device. For additional security, the CA private keys are protected in an HSM and therefore never exposed in the clear on the server. Additionally, an operator-specific certificate template may be selected based on an Operator ID. For example, certificate validity period and some subjectName attributes may be determined by an operator-specific template.

12. In this step, the server derives a one-time session key to encrypt the globally-encrypted device private key. This is the same as in the previous use case. The session key can be derived inside the HSM as well to minimize its exposure.

13. The final credential is then sent to the device.

14. The device decrypts, validates, and installs the credentials the same way as in the previous use case.

In practice, this is the selected method utilized by OPUS to provision operator-specific credentials into wireless devices. While still maintaining end-to-end encryption of private keys from the offline key generation facility all the way to the target device, this method enables real-time provisioning of the device credentials and doesn't depend on manual pre-processing and installation of an authorized device ID list.

3. Scalability and Benchmark Results

With a cluster of 2 front end/back end OPUS server pairs, when key agreement is not required (when unique-per device encryption is applied end-to-end), the observed throughput is 30 million transactions per day. The load balancer allows OPUS to increase scalability if necessary and multiple OPUS clusters with their own load balancers may be deployed for different operators or different types of device credentials. The observed performance included hardware acceleration for RSA signatures provided by a hardware security module (in addition to the signing private key protection)

Key agreement protocols such as DH or ECDH may require additional OPUS server pairs to maintain the same level of scalability. This is highly variable depending on the computing power, hardware-based crypto acceleration and a choice of the key agreement algorithm. According to [6], software implementations for ECDH utilizing NIST P256 curve are 3 times faster than 2048-bit classic Diffie-Hellman. ECDH utilizing Curve25519 gains a performance improvement of another factor of 3 over the NIST P256 curve.

For the OPUS design, security of device credentials is the highest priority since additional layers of encryption or larger key sizes can always be offset by additional pairs of FE/BE servers which can be shared across multiple types of device credentials and operators.

4. Future Work

The use case described in section 2.5 provides a lot of flexibility to network operators to provision a device with their own credentials that utilize an operator-specific CA and an operator-specific certificate template. However, that use case requires a device to store a new private key that is downloaded into the device from the OPUS server. And that may not be possible for devices that utilize secure OTP memory to store a private key which cannot be modified following manufacture.

For such future use cases, the OPUS server may issue a new operator-specific certificate for the same public key that is part of a certificate installed at manufacture time and corresponds to a device private key that is burned into fuses and is not updateable. Allowing multiple simultaneously valid certificates for the same public key is not a recommended security practice and in this case we should revoke the previously issued certificate right before issuing a new operator-specific certificate.

Another way to address the limited storage for private keys in the OTP security fuses would be to derive a public/private keypair during device startup from a seed value that is locked in OTP. This can be easily done with the Elliptic Curve crypto system where a private key is a random value and is much more difficult with RSA. With this approach, each new operator-specific certificate can be issued to a new public key derived based on the new operator ID and it will not be necessary to revoke a previous certificate unless it was compromised.

Additional OPUS system flexibility will be addressed with future enhancements, including for example:

- Improving algorithm agility through the following means:
  - OPUS request will include a list of client-supported algorithms and the OPUS server will select the algorithms from that list to protect the corresponding response containing new credentials.
  - In the case that OPUS server finds client's selection of algorithms inadequate, a client will be automatically redirected to a software update with additional/stronger algorithms. After the client has been upgraded, it will retry.
- Introduction of additional types of device identifiers such as for example a certificate fingerprint

5. Conclusion

Today's consumers may own or lease a wide variety of digital entertainment devices all capable of receiving DRM-protected digital content. Different network operators and over the top content services are protected with a variety of DRM systems and a new use case for a particular device model may not be apparent until after those devices are manufactured and delivered to end consumers.

In order to provide maximum entertainment value to consumers, it becomes necessary to provision their devices that are already installed in their home with new DRM credentials. This can be both a daunting scalability challenge and a security concern.

There are many other types of devices such as routers, wireless network access points, radio points and radio point controllers that have similar needs to obtain new cryptographic device credentials after each such device has already been fielded. Such devices may reside in consumer's home network, a private enterprise network or in the network provider's infrastructure. And they may require different types of credentials to be field downloaded.

Furthermore, a credentials provisioning system has to be constantly revised in order to support new use cases involving new authorization models that may be specific to a network operator or to a new industry. This paper demonstrates that a single field provisioning system can be designed with sufficient flexibility and scalability to address such challenges and to handle many use cases.

Abbreviations and Definitions

AES	Advanced Encryption Standard
BE	back end (server)
CA	Certificate Authority
DH	Diffie-Hellman
DRM	digital rights management
DTCP	Digital Transmission Content Protection
DVR	digital video recorder
FE	front end (server)
ECDH	elliptic curve Diffie-Hellman
HDCP	High-Bandwidth Digital Content Protection
HDR	high dynamic range (high quality video content format supported by Ultra HD TVs)
HSM	hardware security module
IETF	Internet Engineering Task Force
IoT	Internet of things
IPsec	Internet Protocol Security
ITU-T	International Telecommunication Union Telecommunication Standardization Sector
MAC	media access control
MACsec	media access control security (specified by the IEEE 802.1AE-2006 standard)
OPUS	Online PKI Update Service
Origin Credential	Device credential generated in the factory which requires at least partial transformation and replacement on the OPUS server prior to being provisioned into a target device.
OTP	one time programmable
PKI	public key infrastructure
RFC	request for comments (an IETF standards document)
RSA	Rivest-Shamir-Adleman (public key cryptosystem)
SDK	software development kit
SOC	system on a chip
SSL	Secure Sockets Layer
Sub-CA	subordinate certificate authority
TEE	trusted execution environment (e.g., ARM TrustZone, or a separate security processor/co-processor with its own memory space)
TLS	Transport Layer Security (IETF-standardized successor to SSL)
X.509	Standardized format for digital public key certificates. It is published as an ITU-T standard and has a corresponding IETF standard, RFC 5280.

Bibliography & References

Ref	Document Name	Authors	Version
[1]	Netflix Open Connect, https://openconnect.netflix.com/en/#what-is-open-connect	Netflix	
[2]	Keybox User's Guide, https://developers.google.com/android-partner/guide/keybox	Google	
[3]	Verifying Hardware Backed Keypairs with Key Attestation, https://developer.android.com/training/articles/security-key-attestation	Google	
[4]	DTCP and DTCP2 specifications, https://www.dtcp.com/dtcp.aspx	DTLA	
[5]	Use of the RSA-KEM Key Transport Algorithm in the Cryptographic Message Syntax (CMS), IETF RFC 5990	J. Randall, B. Kaliski, J. Brainard, S. Turner	Sept, 2010
[6]	eBACS: ECRYPT Benchmarking of Cryptographic Systems, http://bench.cr.yp.to/call-dh.html	VAMPIRE (Virtual Application and Implementation Research Lab)	

