AN15034

SCEP Usage on i.MX 8/9 Devices

Rev. 1.0 — 24 June 2026 **Application note**

Document information

Information	Content
Keywords	AN15034, Simple Certificate Enrollment Protocol (SCEP), i.MX 93, i.MX 8, i.MX 9
Abstract	This application note provides the instructions on how to run SCEP on the i.MX devices. This application note is applicable to i.MX8/9 series processors.

1 Introduction

The Simple Certificate Enrollment Protocol (SCEP) is a Public Key Infrastructure (PKI) communication protocol. SCEP uses existing technology including Cryptographic Message Syntax (CMS) and PKCS #10 over HTTP. CMS formerly known as PKCS #7. SCEP automates the process of issuing and renewing digital certificates, enabling devices to obtain X.509 certificates from a Certificate Authority (CA) without manual intervention.

This application note provides the instructions on how to run SCEP on i.MX devices. This application note is applicable to i.MX8/9 series processors.

2 SCEP overview

SCEP is the evolution of the enrollment protocol that Cisco Systems sponsors. This protocol is published as RFC 8894 Simple Certificate Enrolment Protocol in 2020. The following are its main characteristics:

- Request/response architecture based on HTTP (optional HTTPS extension for enhanced security)
- Only supports RSA-based cryptography
- Uses PKCS #10 for Certificate Signing Requests (CSR)
- Employs PKCS #7 for cryptographically signed and encrypted message transport

The following are the SCEP entities:

- SCEP client: the device or application requesting a certificate from CA.
- SCEP CA: the trusted entity that validates requests, signs, and issues digital certificates to authenticated clients.

Figure 1. SCEP System Overview

2.1 SCEP supports operations

The SCEP protocol supports the following general operations:

Table 1. Operation type

Operation type	Direction	Purpose
GetCACert (mandatory)	Client → CA	Retrieve CA certificate
GetCACaps (mandatory)	Client → CA	Query CA capabilities
PKCSReq (mandatory)	Client → CA	Certificate enrollment request
CertPoll	Client → CA	Poll pending request
GetCert	Client → CA	Retrieve issued certificate
RenewalReq	Client → CA	Certificate renewal request
UpdateReq	Client → CA	Certificate update request

The following three are the mandatory operations that SCEP performs. These operations are required for the device certification enrollment.

- GetCACaps
- GetCACert

- PKCSReq

2.1.1 GetCACaps

This operation requests capabilities from a CA.

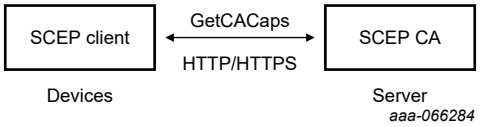

Figure 2. SCEP GetCACaps operation

Table 2. Request/Response

Operations	Describe
Request	The client sends the GetCACaps request using HTTP GET.
Response	The response for a GetCACaps message is a list of CA capabilities, in plain text.

RFC8894 specification defines the following keywords:

- GetNextCACert
- POSTPKIOperation
- Renewal
- AES
- 3DES
- SHA1
- SHA2-256
- SHA2-384
- SHA2-512

2.1.2 GetCACert

To get the CA certificate, the client sends a GetCACert message to the CA.

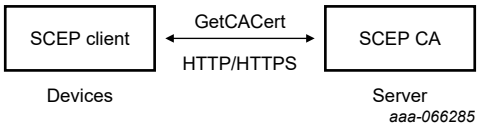

Figure 3. SCEP GetCACert operation

Table 3. Request/Response

Operations	Describe
Request	The client sends the GetCACert request using HTTP GET.
Response	The response consists of a single X.509 CA certificate.

2.1.3 PKCSReq

The PKCSReq operation performs a certificate enrollment transaction.

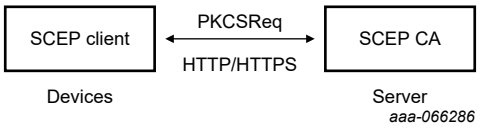

Figure 4. SCEP PKCSReq Operation

Table 4. Request/Response

Operations	Describe
Request	The client sends the enrollment request using HTTP GET.
Response	A signed certificate is held within a special type of PKCS #7 called a “Degenerate Certificates-Only PKCS #7”. It is a special container that can hold one or more X.509 or CRLs, but does not contain a signed or encrypted data payload.

2.2 SCEP secure message objects

The basic building block of all secured SCEP messages is the SCEP pkiMessage. It consists of a PKCS #7 SignedData content type. The PKCS #7 SignedData is a general enveloping mechanism that enables both signed and encrypted transmission of arbitrary data. The SCEP pkiMessage applies both enveloping and signing transformations to protect both the integrity of its end-to-end transaction information and the confidentiality of its information portion.

Figure 5 shows the typical PKCS #7 SignedData Structure.

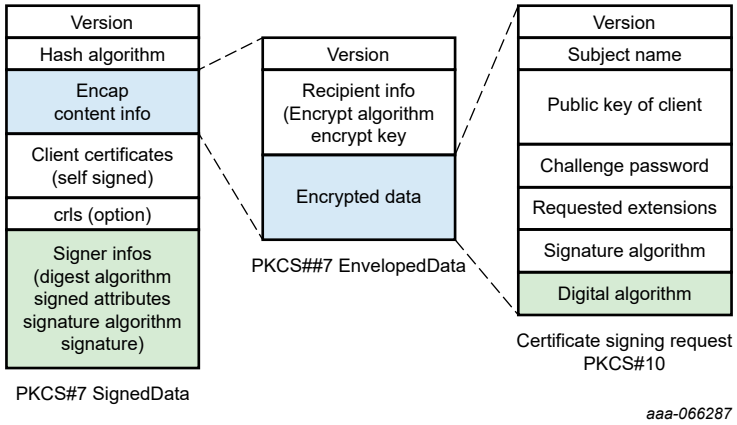

Figure 5. PKCS #7 SignedData Structure

2.3 SCEP message object processing

Creating a SCEP message consists of several stages. The system first encrypts the content and then signs it. The main SCEP message object processing includes the following:

PKCS #7 Wrap (Client → CA)

1. Reads the client certificate request as messageData.
2. Creates inner PKCS #7, encrypts messageData with the CA RSA public key, and produces PKCS #7 EnvelopedData.
3. Creates outer PKCS #7, signs the PKCS #7 EnvelopedData combined with attributes using the client RSA private key, and produces PKCS #7 SignedData.

PKCS #7 Unwrap (CA → Client)

1. Reads the outer PKCS #7 SignedData.

2. Verifies the signature in PKCS #7 SignedData with the CA RSA public key.
3. Decrypts the inner PKCS #7 EnvelopedData with the client RSA private key to extract the original messageData.

2.4 Known limitations

The following are the known limitations of SCEP:

- RSA-only support can be restrictive for modern deployments that require elliptic curve cryptography.
- SCEP RSA operations use PKCS #1 v1.5 padding for both encryption (via PKCS7_encrypt) and signatures (via X509_sign). PKCS #1 v1.5 padding is the traditional padding scheme but has known security vulnerabilities such as Bleichenbacher padding Oracle attacks. For improved security in modern implementations, migrate to RSA-PSS for signature operations and RSA-OAEP for encryption. However, it requires protocol extensions beyond the current SCEP specification.

3 Running SCEP

SCEP deployment typically involves a SCEP server and SCEP clients.

- SCEP server: runs on a Windows or Linux server machine that acts as the CA.
- SCEP clients: to request automatically and obtain digital certificates for secure authentication and communication, SCEP clients run on embedded devices, such as IoT devices, routers, or industrial controllers.

3.1 Setup SCEP server

Deploy a SCEP server on a Windows or Linux machine using one of several available CA solutions that support SCEP, such as the open source OpenCA/OpenSCEP. For this implementation, choose an open source SCEP server written in Go language for its simplicity and cross-platform compatibility.

Compiling SCEP server source code on x86 host machine:

```
git clone https://github.com/micromdm/scep.git
cd scep
make
```

Note: *To compile the SCEP server, you need a Go compiler and standard tools such as Git, make.*

The binaries are compiled in the current directory and named after the architecture, that is, `scepclient-linux-amd64` and `scepserver-linux-amd64`.

Perform the following to create a CA and private key:

```
mkdir test & cd test
../scepserver-linux-amd64 ca -init
```

The initial CA files `ca.pem` and `ca.key` are created in folder `depot`.

Startup SCEP server, for example listening to port 2016:

```
../scepserver-linux-amd64 -depot depot -port 2016 -debug -allowrenew 0
```

3.2 Setup SCEP client

A SCEP client is a software application or tool that implements the client-side of the SCEP protocol to automate the process of obtaining and managing digital certificates from a CA.

In this section, two SCEP client implementations are described on the i.MX 93 device to enable automated certificate enrollment. Both the implementations are based on the open source SCEP client SSCEP - Simple SCEP client for Unix. One implementation uses OpenSSL for standard cryptographic operations. Another implementation uses the OP-TEE secure environment of the device for enhanced security with secure key storage and cryptographic operations in the Trusted Execution Environment (TEE).

3.2.1 SCEP client (sscep)

SSCEP - Simple SCEP client for Unix is a client-only implementation of the SCEP. It is widely used for automated certificate enrollment and management.

Compile the sscep source code on the i.MX 93 device as follows:

```
git clone https://github.com/certnanny/sscep.git
cd sscep
./bootstrap.sh
./configure
make
```

The compiled binary sscep is placed in the current directory.

3.2.2 SCEP client backends with OpenSSL

This section describes how the SCEP client uses OpenSSL as its cryptographic backend to handle RSA key pair generation, CSR creation, and automated SCEP enrollment with the CA.

1. Create an OpenSSL configuration file that enables legacy cryptographic algorithms and providers.

```
root@imx93evk:~# cat ./openssl_legacy.cnf
openssl_conf = openssl_init

[openssl_init]
providers = provider_sect

# List of providers to load
[provider_sect]
default = default_sect
legacy = legacy_sect

[default_sect]
activate = 1

[legacy_sect]
activate = 1
```

2. Generate a 2048-bit RSA private key and create a CSR for the device `device01.example.com` to be submitted to a CA for certificate issuance.

```
root@imx93evk:~# export OPENSSL_CONF=./openssl_legacy.cnf
root@imx93evk:~# openssl genrsa -out private.key 2048
root@imx93evk:~# openssl req -new -key private.key -out request.csr -subj "/
CN=device01.example.com"
```

3. Retrieve the CA certificate from the SCEP server. Then to enroll the device and obtain a signed client certificate, submit the CSR.

```
root@imx93evk:~# ./sscep/sscep getca -u http://<scep server IP>:2016/scep -c
 ca.pem
root@imx93evk:~# ./sscep/sscep enroll -u http://<scep server IP>:2016/scep -c
 ca.pem -k private.key -r request.csr -l client_legacy.crt
```

`client_legacy.crt` is the final client signed certificate.

3.2.3 SCEP client backends with OP-TEE

This section describes how to implement a SCEP client backend using OP-TEE for secure key generation and storage. The approach integrates:

- OP-TEE for secure key storage in the TEE.
- PKCS #11 interface for standardized cryptographic operations.
- pkcs11-tool for RSA key pair generation within the secure world.
- OpenSSL with pkcs11-provider for CSR generation using TEE-protected keys.

This architecture ensures that private keys never leave the secure environment, provides hardware-level protection for the cryptographic identity of the device.

3.2.3.1 Enable OP-TEE

OP-TEE is an open source implementation of the TEE specification that Global Platform defines. It provides a hardware-isolated secure execution environment for protecting sensitive operations like cryptography, authentication, and secure storage, using Arm TrustZone technology to separate trusted code from the normal operating system.

Figure 6 shows the SCEP backend with OP-TEE architecture block diagram.

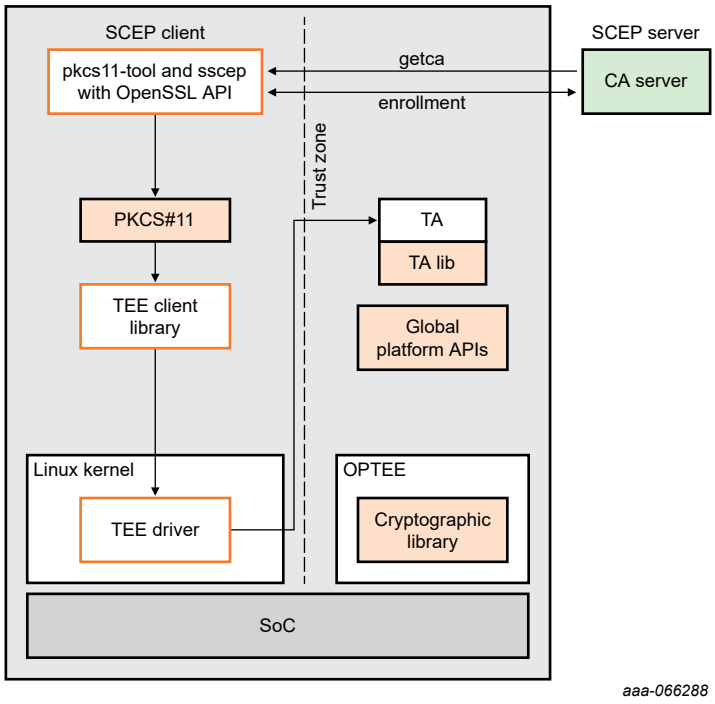

Figure 6. SCEP backend OP-TEE architecture

`pkcs11-tool` is part of the open source project and serves as a testing and administration tool for PKCS #11 modules. It provides functions to manage tokens including initialization, PIN management, and key generation. The tool can perform cryptographic operations like signing, encryption, and key pair generation through the PKCS #11 interface.

When used with OP-TEE, pkcs11-tool communicates with the secure world to perform cryptographic operations.

3.2.3.2 Configure OpenSSL

Setting up OpenSSL to perform cryptography through PKCS #11 interface (libckteec.so) depends on the `pkcs11-provider` module.

`pkcs11-provider` is an OpenSSL 3.x provider that enables OpenSSL to interface with PKCS #11 modules for cryptographic operations. It acts as a bridge between OpenSSL provider API and PKCS #11 libraries.

Create an OpenSSL configuration file that enables PKCS #11 module and providers.

```
root@imx93evk:~# cat openssl_tee.cnf
openssl_conf = openssl_init

[openssl_init]
providers = provider_sect
alg_section = algorithm_sect

# List of providers to load
[provider_sect]
default = default_sect
pkcs11 = pkcs11_sect
legacy = legacy_sect

[default_sect]
activate = 1

[pkcs11_sect]
module = /usr/lib/ossl-modules/pkcs11.so
pkcs11-module-path = /usr/lib/libckteec.so.0
pkcs11-module-cache-keys = false
pkcs11-module-quirks = no-operation-state
pkcs11-module-block-operations = digest mac cipher
activate = 1

[algorithm_sect]
default_properties = ?provider=default

[legacy_sect]
activate = 1
```

Configure the environment to enable OpenSSL to use the OP-TEE PKCS #11 interface for cryptographic operations.

```
root@imx93evk:~# export OPENSSL_CONF=/root/openssl_tee.cnf
root@imx93evk:~# export PKCS11_MODULE_PATH="/usr/lib/libckteec.so.0"
root@imx93evk:~# export PIN="1234"
root@imx93evk:~# export SO_PIN="1234"
root@imx93evk:~# export TOKEN_NAME="token0"
root@imx93evk:~# export PKCS11_PROVIDER_DEBUG=file:/tmp/debug.log
```

3.2.3.3 Generate RSA key pair

To initialize the PKCS #11, pkcs11-tool sets the security officer PIN and user PIN, and creates a secure storage area for cryptographic keys.

```
root@imx93evk:~# pkcs11-tool --init-token --slot-index=1 --label=$TOKEN_NAME --
so-pin $SO_PIN --module $PKCS11_MODULE_PATH
root@imx93evk:~# pkcs11-tool --init-pin --pin $PIN --slot-index=1 --label=
$TOKEN_NAME --so-pin $SO_PIN --module $PKCS11_MODULE_PATH
```

After initialization, it can generate an RSA key pair directly in the secure token. The system identifies the generated keys by labels and IDs. Applications such as OpenSSL use these labels and IDs for signing and encryption operations.

```
root@imx93evk:~# pkcs11-tool --keypairgen --key-type RSA:2048 --label
 "test_rsa_new" --id 1 --login --module $PKCS11_MODULE_PATH --usage-sign  --
usage-decrypt --pin $PIN --slot-index=1
```

3.2.3.4 Convert PKCS #11 URI to PEM

pkcs11-provider provides the `uri2pem` utility that extracts public key information from a PKCS# 11 token and converts it to PEM format. It takes a PKCS #11 URI as input (for example, "pkcs11:token=token0;object=my-key") and outputs the corresponding private key in standard PEM encoding.

```
root@imx93evk:~# python3 pkcs11-provider-main/tools/uri2pem.py --out key.pem
 'pkcs11:model=OP-TEE
%20TA;manufacturer=Linaro;serial=0000000000000001;token=token0;id=
%01;object=test_rsa_new;type=private'
root@imx93evk:~# openssl req -new -key key.pem -subj /O=NXP-CLIENT-521/
CN=10.232.132.242 -out cert.csr
```

3.2.3.5 Run sscep

Retrieve the CA certificate from the SCEP server, then to enroll the device and obtain a signed client certificate, submit the CSR.

```
root@imx93evk:~#  ./sscep/sscep getca -u http://<scep server IP>:2016/scep -c
 ca.pem
root@imx93evk:~#  ./sscep/sscep enroll -u http://<scep server IP>:2016/scep -k
 key.pem -c ca.pem -r cert.csr -l client.crt
```

`client.crt` is the final client signed certificate.

4 Acronyms

Table 5 lists the acronyms used in this document.

Table 5. Acronyms

Acronym	Description
3DES	Triple Data Encryption Standard
AES	Advanced Encryption Standard
API	Application Programming Interface

Table 5. Acronyms...*continued*

Acronym	Description
CA	Certificate Authority
CMS	Cryptographic Message Syntax
CSR	Certificate Signing Requests
GET	An HTTP request method used to retrieve data from a web server
HTTPS	Hypertext Transfer Protocol Secure
IoT	Internet Of Things
OpenSSL	Open Secure Sockets Layer
OP-TEE	Open Portable Trusted Execution Environment
PEM	Privacy-Enhanced Mail
PKCS	Public Key Cryptography Standards
PKI	Public Key Infrastructure
RSA	Rivest–Shamir–Adleman
RSA-OAEP	RSA Optimal Asymmetric Encryption Padding
RSA-PSS	RSA Probabilistic Signature Scheme
SCEP	Simple Certificate Enrollment Protocol
SHA	Secure Hash Algorithm
URI	Uniform Resource Identifier
X.509	A standard for public key infrastructure

5 Revision history

Table 6 summarizes revisions to this document.

Table 6. Revision history

Document ID	Release date	Description
AN15034 v.1.0	24 June 2026	Initial public release

6 References

- RFC 8894 Simple Certificate Enrolment Protocol
- SCEP server
- SSCEP - Simple SCEP client for Unix
- pkcs11-provider

Legal information

Definitions

Draft — A draft status on a document indicates that the content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included in a draft version of a document and shall have no liability for the consequences of use of such information.

Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms and conditions of commercial sale of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Terms and conditions of commercial sale — NXP Semiconductors products are sold subject to the general terms and conditions of commercial sale, as published at https://www.nxp.com/profile/terms, unless otherwise agreed in a valid written individual agreement. In case an individual agreement is concluded only the terms and conditions of the respective agreement shall apply. NXP Semiconductors hereby expressly objects to applying the customer's general terms and conditions with regard to the purchase of NXP Semiconductors products by customer.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

Suitability for use in non-automotive qualified products — Unless this document expressly states that this specific NXP Semiconductors product is automotive qualified, the product is not suitable for automotive use. It is neither qualified nor tested in accordance with automotive testing or application requirements. NXP Semiconductors accepts no liability for inclusion and/or use of non-automotive qualified products in automotive equipment or applications.

In the event that customer uses the product for design-in and use in automotive applications to automotive specifications and standards, customer (a) shall use the product without NXP Semiconductors' warranty of the product for such automotive applications, use and specifications, and (b) whenever customer uses the product for automotive applications beyond NXP Semiconductors' specifications such use shall be solely at customer's own risk, and (c) customer fully indemnifies NXP Semiconductors for any liability, damages or failed product claims resulting from customer design and use of the product for automotive applications beyond NXP Semiconductors' standard warranty and NXP Semiconductors' product specifications.

HTML publications — An HTML version, if available, of this document is provided as a courtesy. Definitive information is contained in the applicable document in PDF format. If there is a discrepancy between the HTML document and the PDF document, the PDF document has priority.

Translations — A non-English (translated) version of a document, including the legal information in that document, is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

Security — Customer understands that all NXP products may be subject to unidentified vulnerabilities or may support established security standards or specifications with known limitations. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer's applications and products. Customer's responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer's applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately.

Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP.

NXP has a Product Security Incident Response Team (PSIRT) (reachable at PSIRT@nxp.com) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

NXP B.V. — NXP B.V. is not an operating company and it does not distribute or sell products.

Trademarks

Notice: All referenced brands, product names, service names, and trademarks are the property of their respective owners.

NXP — wordmark and logo are trademarks of NXP B.V.

Contents