Use the Secure Key Injection Protocol in Certificate Manager

This article describes the process of how to use a secure key injection protocol (SKIP) for constrained devices in Smart ID Certificate Manager (CM). Such a device is only required to generate an initial factory key pair and the rest of the required key pairs are generated and provided by Certificate Manager.

The protocol consists of a single request and response exchange.

- The request contains a single PKCS#10 encoded certificate signing request with the initial factory public key.
- The response contains a CMS SignedData type with the generated key pairs and issued certificates.

Expand/Collapse All

This article is valid for CM 8.3 and later.

Related information

- Smart ID Certificate Manager

Security services in the secure key injection protocol

The secure key injection protocol provides the following security services:

Confidentiality	The protocol provides for uninterrupted encryption of the private keys between the source HSM that generates the key pairs and the target device. The private keys are encrypted in the HSM with ECDH key agreement between the initial key pair on the target device and an ephemeral key pair generated by Certificate Manager (CM).
Authentication	The origin of the generated keys are authenticated by issuing a certificate for the ephemeral public key.
Integrity	The integrity of the generated private keys for the target device is provided by digitally signing the encrypted private keys with the ephemeral private key.

Specification

The initial factory key pair and the ephemeral key pair must support both key agreement and digital signature. They must also be generated from the same domain parameters.

Supported EC curves for factory and ephemeral key pairs

- brainpoolP256r1
- brainpoolP320r1
- brainpoolP384r1
- brainpoolP512r1
- secp256r1
- secp384r1,
- secp521r1

Supported EC curves for generated device key pairs

- brainpoolP256r1
- brainpoolP320r1
- brainpoolP384r1
- brainpoolP512r1
- Ed25519
- secp256r1
- secp384r1
- secp521r1

Key agreement algorithm

- ECDH (CKM_ECDH1_DERIVE)

Private key encryption algorithm

- AES/CBC/PKCS5Padding (CKM_AES_CBC_PAD), with 256 bits key

Secure key package encoding

- CMS (RFC 5652 [https://tools.ietf.org/html/rfc5652]) SignedData type

Process description

The generated and encrypted device private keys are encoded and signed as a CMS SignedData type.

The content is an ASN.1 encoded list of the generated device key pairs, `KeyPairContainers`, where:

- the public key is an encoded `SubjectPublicKeyInfo` (RFC 5280 (https://tools.ietf.org/html/rfc5280) and
- the private key is an encoded `EncryptedPrivateKeyInfo` (RFC 5958 (https://tools.ietf.org/html/rfc5958)

1. The public key is used to match a key pair to the corresponding issued certificate:

Match key pair to issued certificate

```
KeyPairContainers ::= SEQUENCE OF KeyPairContainer
KeyPairContainer ::= SEQUENCE {
public SubjectPublicKeyInfo,
encryptedPrivate EncryptedPrivateKeyInfo
}
```

1. An initial factory key pair is generated in the target device, using one of the supported EC curves listed in section "Key types and algorithms".
2. A PKCS#10 request for this public key is created and signed with the private key and sent to Certificate Manager.

The key pairs are generated so that the private keys cannot be exported in clear text from the HSM.

1. CM generates an EC ephemeral key pair for the key agreement process, using the same curve as selected for the initial factory public key in the request.
2. An AES-256 key is derived in the HSM using the ephemeral private key and the initial factory public key, using the ECDH algorithm (CKM_ECDH1_DERIVE).
3. CM generates the key pairs for the target device. For each device key pair, the AES-256 key is used to wrap the private key, using algorithm AES/CBC/PKCS5Padding (CKM_AES_CBC_PAD).
4. The device key pairs are encoded as a list of `KeyPairContainer` defined above.
5. The encoded key pair list is signed with the ephemeral private key, as a CMS SignedData type. The signer is identified with the `subjectKeyIdentifier` of the ephemeral public key, since the certificate is issued later in the process.
6. All key handles in the HSM, of the AES key and the ephemeral and device key pairs, are destroyed.

1. CM issues certificates for the initial factory public key, the ephemeral public key and the device p ublic keys.
2. The issued certificates are added to the CMS SignedData type.

1. The secure key package is decoded as a CMS SignedData type.
2. The issuer of the received certificates, including the certificate for the ephemeral public key, is ve rified to a trusted root.
3. The signature of the SignedData is verified. The certificate with the ephemeral public key for the verification is identified with the `subjectKeyIdentifier` value in the SignerInfo and the subje ct key identifier extension in the certificate.
4. The AES-256 key is derived using the initial factory private key and the received ephemeral publi c key, using the ECDH algorithm (CKM_ECDH1_DERIVE).
5. For each received key pair, the AES-256 key is used to unwrap the encrypted private key into the target device, using algorithm AES/CBC/PKCS5Padding (CKM_AES_CBC_PAD).