KeyGen2/SKS Scenario – Remote Key Unlock

Keys may lockup due to repeated use of the wrong PIN. If a key is also protected by a PUK the PUK code can be distributed (OOB) to the user for unlock. The following scenario describes remote unlock *that does not rely on a PUK* but on a KMK (Key Management Key) that the issuer deployed during the provisioning of the key. This latter has in the SKS architecture been coined VSD (Virtual Security Domain) and is *the foundation for keeping issuers securely separated from each other*. For detailed information see the *SKS specification*.

The user is assumed to browse to an issuer-specific URL which starts a KeyGen2 operation. How the user is authenticated to the issuer service is out of scope for KeyGen2 but for unlock operations the SKS identity may be sufficient, it at least fully identifies the container.

The following "printout" of a KeyGen2 session shows how remote unlock can be performed.

Pass #1: PlatformNegotiationRequest

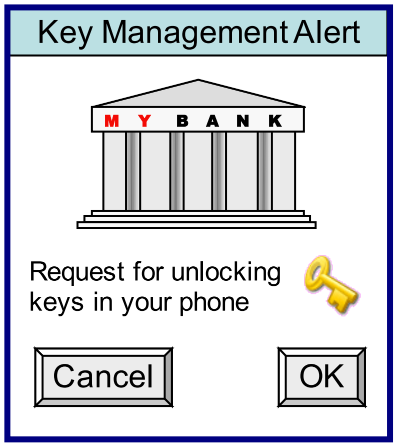
```
<?xml version="1.0" encoding="UTF-8"?>
<PlatformNegotiationRequest Action="unlock" ID="S-12c84bd26b886bedfa8fe0f2e21"
                            SubmitURL="http://issuer.example.com/platform"
                            xmlns="http://xmlns.webpki.org/keygen2/beta/20101204#">
   <IssuerLogotype Height="150"
                   ImageFingerprint="Shtd/s6ukm70ZPjUpQ+5fPW0kWHalE2gRqvH6s4vGaI="
                   LogotypeURL="http://issuer.example.com/images/logo.png"
                   MimeType="image/png"
                   Width="200"/>
   <BasicCapabilities/>
</PlatformNegotiationRequest>
```

Pass #2: PlatformNegotiationResponse

```
<?xml version="1.0" encoding="UTF-8"?>
<PlatformNegotiationResponse ServerSessionID="S-12c84bd26b886bedfa8fe0f2e21"
                             xmlns="http://xmlns.webpki.org/keygen2/beta/20101204#">
   <BasicCapabilities/>
</PlatformNegotiationResponse>
```

Comment: The `PlatformNegotiation` request-response pair is mandatory and alerts the user that a service wants to perform an operation in an SKS key container. In addition, algorithm and feature requirements and preferences are negotiated.   In this case the `Action` attribute was set to "unlock" as this is the only thing the issuer *declared* it intends to perform.

SKS crypto involved: none, only capability requests may be performed at this stage.



Pass #3: ProvisioningInitializationRequest

```
<?xml version="1.0" encoding="UTF-8"?>
<ProvisioningInitializationRequest ID="S-12c84bd26b886bedfa8fe0f2e21"
                                   ServerTime="2010-11-27T09:32:57+01:00"
                                   Algorithm="http://xmlns.webpki.org/keygen2/1.0#algorithm.sks.s1"
                                   SessionKeyLimit="50"
                                   SessionLifeTime="10000"
                                   SubmitURL="http://issuer.example.com/provsess"
                                   xmlns="http://xmlns.webpki.org/keygen2/beta/20101204#"
                                   xmlns:ds11="http://www.w3.org/2009/xmldsig11#">
  <ServerEphemeralKey>
    <ds11:ECKeyValue>
      <ds11:NamedCurve URI="urn:oid:1.2.840.10045.3.1.7"/>
      <ds11:PublicKey>BMKnN7n7VyP/91qOEVAQ/x....wi2J4e44PwRMrjY0=</ds11:PublicKey>
    </ds11:ECKeyValue>
  </ServerEphemeralKey>
</ProvisioningInitializationRequest>
```

Pass #4: ProvisioningInitializationResponse

```
<?xml version="1.0" encoding="UTF-8"?>
<ProvisioningInitializationResponse Attestation="MWdfi2wqy+WEI/x56eO1e …. vEWlbj+ruKtw="
                                    ClientTime="2010-11-27T09:32:57+01:00"
                                    ID="C-12c84bd26dcffd1b02a9b75734a"
                                    ServerSessionID="S-12c84bd26b886bedfa8fe0f2e21"
                                    ServerTime="2010-11-27T09:32:57+01:00"
                                    xmlns="http://xmlns.webpki.org/keygen2/beta/20101204#"
                                    xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
                                    xmlns:ds11="http://www.w3.org/2009/xmldsig11#">
  <ClientEphemeralKey>
    <ds11:ECKeyValue>
      <ds11:NamedCurve URI="urn:oid:1.2.840.10045.3.1.7"/>
      <ds11:PublicKey>BMLWM1XoMO6wkUo6u4reeN13HC4D4 …. FY/MHfEpObRHJhFcrOYMZ+MpxjXpo=</ds11:PublicKey>
    </ds11:ECKeyValue>
  </ClientEphemeralKey>
  <DeviceCertificatePath>
    <ds:X509Data>
      <ds:X509Certificate>MIIC2DCCAcCgAwIBAgIGARTWcc7VMA0 ….. Y+xtVD5cD1Gcn7KNdcJfLt</ds:X509Certificate>
    </ds:X509Data>
  </DeviceCertificatePath>
  <ds:Signature>
    <ds:SignedInfo>
      <ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
      <ds:SignatureMethod Algorithm="http://www.w3.org/2001/04/xmldsig-more#hmac-sha256"/>
      <ds:Reference URI="#C-12c84bd26dcffd1b02a9b75734a">
        <ds:Transforms>
          <ds:Transform Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
          <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
        </ds:Transforms>
        <ds:DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>
        <ds:DigestValue>spVTwuSvOjadMf/9PG2lwpa1m4YiMRjWJ71eoVXX9XA=</ds:DigestValue>
      </ds:Reference>
    </ds:SignedInfo>
    <ds:SignatureValue>zySg8QkLwCNwMwfpV6X8cKEcfoJ2thxjKC+IEAeuW7s=</ds:SignatureValue>
    <ds:KeyInfo>
      <ds:KeyName>derived-session-key</ds:KeyName>
    </ds:KeyInfo>
  </ds:Signature>
</ProvisioningInitializationResponse>
```

This is the core SKS operation, creating an authenticated shared session key. It is with the exception of **`ClientTime`** time *running at the SKS level* to cater for E2ES support.

Note: the *issuer is not authenticated by the SKS*, it is authenticated by the *user* (+ SSL cert) in the PlatformNegotiation phase.

Primary SKS method: **`createProvisioningSession()`**

Pass #5: CredentialDiscoveryRequest

```
<?xml version="1.0" encoding="UTF-8"?>
<CredentialDiscoveryRequest ClientSessionID="C-12c84bd26dcffd1b02a9b75734a"
                            ID="S-12c84bd26b886bedfa8fe0f2e21"
                            SubmitURL="http://issuer.example.com/credisc"
                            xmlns="http://xmlns.webpki.org/keygen2/beta/20101204#"
                            xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
  <LookupSpecifier ID="Lookup.1" Nonce="NmFp6mVN99JjWiiwIVQSdudxNI37ZTKW4qfS3Z8N0uY=">
    <SearchFilter Email="john.doe@example.com"/>
    <ds:Signature>
      <ds:SignedInfo>
        <ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
        <ds:SignatureMethod Algorithm="http://www.w3.org/2001/04/xmldsig-more#rsa-sha256"/>
        <ds:Reference URI="#Lookup.1">
          <ds:Transforms>
            <ds:Transform Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
            <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
          </ds:Transforms>
          <ds:DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>
          <ds:DigestValue>pzD0XzkaZLb/aSBiElx+m+dhPk7Do7Z34u/FP0IElEk=</ds:DigestValue>
        </ds:Reference>
      </ds:SignedInfo>
      <ds:SignatureValue>KpWG/aMDDb0GQfviAC35ViBM0e869n....Wr6dfoydZfLNiZBFD42chRbiBVxEzDnvA=</ds:SignatureValue>
      <ds:KeyInfo>
        <ds:KeyValue>
          <ds:RSAKeyValue>
            <ds:Modulus>AMQVfhviyYdQYe+TlJFQwwGytXeWvkyn....ZoMjkx6tAuTMey9SxW0XLaZDcXl</ds:Modulus>
            <ds:Exponent>AQAB</ds:Exponent>
          </ds:RSAKeyValue>
        </ds:KeyValue>
      </ds:KeyInfo>
    </ds:Signature>
  </LookupSpecifier>
</CredentialDiscoveryRequest>
```

Pass #6: CredentialDiscoveryResponse

```
<?xml version="1.0" encoding="UTF-8"?>
<CredentialDiscoveryResponse ID="C-12c84bd26dcffd1b02a9b75734a"
                             ServerSessionID="S-12c84bd26b886bedfa8fe0f2e21"
                             xmlns="http://xmlns.webpki.org/keygen2/beta/20101204#">
  <LookupResult ID="Lookup.1">
    <MatchingCredential CertificateFingerprint="hwEPTko3GvE+44Q7jC3f46wyfPGEDRBntyfUUTUY6PA="
                        ClientSessionID="C-12c84bd25e4e4957d83a9514e23"
                        Locked="true"
                        ServerSessionID="S-12c84bd25bc34fbb0c2a435735f"/>
  </LookupResult>
 /CredentialDiscoveryResponse>
```

Comment: In order to perform reliable key management, you need to know if the key in question is actually available since the user may be able to delete keys. In some cases you may not even know exactly what keys that you have issued to a specific SKS and for that purpose credential discovery is indispensable. By requiring KMK-signed lookup requests an issuer can only "see" keys it has actually issued.

The **`CredentialDiscovery`** request-response pair is an optional KeyGen2 step which is *performed at the middleware level*.

Primary SKS methods:

`enumerateProvisioningSessions()`, **`enumerateKeys()`**, **`getKeyAttributes()`**

Pass #7: ProvisioningFinalizationRequest

```
<?xml version="1.0" encoding="UTF-8"?>
<ProvisioningFinalizationRequest ClientSessionID="C-12c84bd26dcffd1b02a9b75734a"
                                 ID="S-12c84bd26b886bedfa8fe0f2e21"
                                 MAC="jgvw7lAXqLoGUuQNGz9GycYWw2rEASsNHJREzcO3FDc="
                                 Nonce="Xx7tm1g2240C55TLb9Yqmxm9FKWv62WXIOb0C7bwzbA="
                                 SubmitURL="http://issuer.example.com/credep"
                                 xmlns="http://xmlns.webpki.org/keygen2/beta/20101204#">
   <UnlockKey Authorization="nIqmN9elBOE5H82E.... bhKtoT92wDYi4D2vgrV9A="
              CertificateFingerprint="hwEPTko3GvE+44Q7jC3f46wyfPGEDRBntyfUUTUY6PA="
              ClientSessionID="C-12c84bd25e4e4957d83a9514e23"
              MAC="Q+AZxTt5LLPCDm79yCuHgoFozXhoGaY0Xm5X/OA6Y/Q="
              ServerSessionID="S-12c84bd25bc34fbb0c2a435735f"/>
</ProvisioningFinalizationRequest>
```

Pass #8: ProvisioningFinalizationResponse

```
<?xml version="1.0" encoding="UTF-8"?>
<ProvisioningFinalizationResponse Attestation="IpTDttCtplnkIiIX7TS5nSZTuJrcO9tt97MGbehdw4o="
                                  ID="C-12c84bd26dcffd1b02a9b75734a"
                                  ServerSessionID="S-12c84bd26b886bedfa8fe0f2e21"
                                  xmlns="http://xmlns.webpki.org/keygen2/beta/20101204#"/>
```

Comment: The **`ProvisioningFinalization`** request-response pair is the mandatory ending of a KeyGen2/SKS session. The request usually contains down-loadable data such as certificates, logotypes, and symmetric keys but can as shown above also comprise of stand-alone key management operations.

These operations are *performed at the SKS level* in order to cater for E2ES and transaction-based operation.

Primary SKS methods: **`pp_unlockKey`**, **`closeProvisioningSession()`**