```
<?xml version="1.0" encoding="UTF-8"?>
```

KeyGen2 XML schema. The primary purpose of KeyGen2 is usage as a browser-based protocol for on-line Provisioning and Management of PKI-based user-keys stored in containers compatible with the SKS (Secure Key Store) specification.

In addition to PKI, KeyGen2 also supports OTP (One Time Password) "seeds" and similar symmetric keys. By "piggybacking" symmetric keys on PKI, the management operations work identical for PKI and symmetric keys.

Utilizing the KeyGen2 generic extension mechanism, "alien" objects like downloadable code and InformationCards tied to specific keys can be issued and maintained without changing the protocol engine.

Historical note: KeyGen2 started its life as a 4-pass stand-alone key-provisioning protocol back in 2007 but ended up as a SKS/KeyGen2 10-pass "combo" three years later.

Properly implemented the current incarnation provides E2ES (End to End Security), VSDs (Virtual Security Domains), and TBP (Transaction Based Provisioning), which is primarily intended for mobile phones with embedded security hardware hosting credentials from multiple and independent issuers.

KeyGen2 may be used in the following modes and associated protocol steps:

- Mandatory: PlatformNegotiation

- Action="create": ProvisioningInitialization KeyCreation
- Action="unlock": ProvisioningInitialization [CredentialDiscovery]
- Action="update": ProvisioningInitialization [CredentialDiscovery] [KeyCreation]
- Action="resume": *

- Mandatory: ProvisioningFinalization *

* Note that a deferred certification process ends after the KeyCreation (Response) and resumes with a PlatformNegotiation (Request) invoked by the client using an URL supplied OOB.

> *Perfection is achieved, not when there is nothing more to add, but when there is nothing left to take away* [Antoine de Saint-Exupery]

Version: BETA
Revised: 28-DEC-2012
Author: Anders Rundgren


```
<xs:schema targetNamespace="http://xmlns.webpki.org/keygen2/beta/20121228#"
           xmlns:xs="http://www.w3.org/2001/XMLSchema"
           xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
           xmlns:ds11="http://www.w3.org/2009/xmldsig11#"
           xmlns:kg2="http://xmlns.webpki.org/keygen2/beta/20121228#"
           elementFormDefault="qualified" attributeFormDefault="unqualified">

  <xs:import namespace="http://www.w3.org/2000/09/xmldsig#"/>

  <xs:import namespace="http://www.w3.org/2009/xmldsig11#"/>
```

The primary (top-level) platform negotiation request object

```
<xs:element name="PlatformNegotiationRequest">
  <xs:complexType>
    <xs:sequence>
```

A request MAY also include an enveloped XML signature covering the entire request object (which MUST be done through an explicit reference to the request's top-level "ID" attribute). The client software MUST verify such a signature before proceeding with the process. The requester is RECOMMENDED to sign using a certificate which already has a trust anchor installed in the potential users' browsers. If the trust anchor is not known, the client software SHOULD warn the user before proceeding. The user SHOULD be able to abort the operation in case of an unknown issuer. A broken signature MUST abort the entire process

```
      <xs:element ref="ds:Signature" minOccurs="0"/>

    </xs:sequence>
```

The requesting issuer SHOULD enquire about capabilities that are not default or even may be unavailable due to export restrictions. An omission performing a proper capability negotiation could in fact lead to run-time errors in the client implementation!

```
    <xs:attributeGroup ref="kg2:basicCapabilities"/>
```

GUI notification

```
    <xs:attribute name="Action" type="kg2:action" use="required"/>
```

"PrivacyEnabled" specifies if the SKS privacy enabled provisioning mode is to be used. Default: false

```
    <xs:attribute name="PrivacyEnabled" type="xs:boolean" use="optional"/>
```

GUI language in preference order. Primarily for public computers

```
    <xs:attribute name="Languages" type="kg2:languageList" use="optional"/>
```

Key container constraints in preference order

```
    <xs:attribute name="KeyContainers" type="kg2:containerList" use="optional"/>
```

This mandatory server-generated item MUST be unique between key provisioning sessions but MUST remain constant among requests within a specific session

```
    <xs:attribute name="ID" type="kg2:sksID" use="required"/>
```

Where to HTTP POST the platform negotiation response. Note that applicable session cookies MUST be submitted in the request as well

```
    <xs:attribute name="SubmitURL" type="kg2:httpURI" use="required"/>
```

If the user aborts the enrollment process the following optional URL can be set to inform the issuer that this is the case. The server is supposed to return a suitable web-page as the response from the HTTP GET operation. Note that applicable session cookies MUST be submitted in the request as well

```
    <xs:attribute name="AbortURL" type="kg2:httpURI" use="optional"/>

  </xs:complexType>
</xs:element>
```

End of platform negotiation request object definition

The primary (top-level) platform negotiation response object

```
<xs:element name="PlatformNegotiationResponse">
  <xs:complexType>
    <xs:sequence>

      Optional hints for the "CertificatePath/Logotype" extension

      <xs:element name="ImagePreference" type="kg2:imageData" minOccurs="0" maxOccurs="unbounded"/>

    </xs:sequence>

    A conforming client MUST respond to special requirements by indicating its matching
    capabilities.  A client MAY also in the case the default (mandatory) is unavailable due to
    export restrictions or marginal implementations, output its reduced capability in order for
    the issuer to either deny credentials (with an appropriate message to user...), or issue
    credentials with a possibly more limited scope.

    In the case a capability is secret, a client MAY not return a capability except for signed
    requests from a known (pre-configured) party

    <xs:attributeGroup ref="kg2:basicCapabilities"/>

    MUST be equal to "PlatformNegotiationRequest/@ID" within a specific session

    <xs:attribute name="ServerSessionID" type="kg2:sksNCName" use="required"/>

  </xs:complexType>
</xs:element>
```

End of platform negotiation response object definition

The primary (top-level) provisioning initialization request object

```
<xs:element name="ProvisioningInitializationRequest">
  <xs:complexType>
    <xs:sequence>

      SKS Reference: createProvisioningSession
      "ServerEphemeralKey" holds a server-generated ephemeral key for creating an SKS
      "SessionKey" (shared session key)

      <xs:element name="ServerEphemeralKey">
        <xs:complexType>
          <xs:sequence>

            MUST contain an ECDH public key

            <xs:element ref="ds11:ECKeyValue"/>

          </xs:sequence>
        </xs:complexType>
      </xs:element>

      SKS Reference: createProvisioningSession
      Optional key management key

      <xs:element name="KeyManagementKey" minOccurs="0">
        <xs:complexType>
          <xs:sequence>

            MUST contain an RSA or ECDSA public key

            <xs:choice>

              <xs:element ref="ds:RSAKeyValue"/>
```

```
            <xs:element ref="ds11:ECKeyValue"/>

          </xs:choice>
```

Virtual Machine support is optional and MUST be verified before requested by enquiring the capability attribute "http://xmlns.webpki.org/keygen2/1.0#feature.vm". The VMs identity is supposed to be derived from the "KeyManagementKey" which requires an "Authorization" object created by signing "ServerEphemeralKey" with "KeyManagementKey" using SHA256 as the hash algorithm. Note that VM support is meant to be dealt with by the client platform rather than the SKS. If the VM doesn't exist it should (after the user's consent) be created, otherwise it is opened. A newly created VM should not be made available until the provisioning session has successfully terminated.

```
          <xs:element name="VirtualMachine" minOccurs="0">
            <xs:complexType>
              <xs:attribute name="FriendlyName" type="kg2:friendlyNameType" use="required"/>
              <xs:attribute name="Authorization" type="xs:base64Binary" use="required"/>
            </xs:complexType>
          </xs:element>

        </xs:sequence>
      </xs:complexType>
    </xs:element>
```

See "PlatformNegotiationRequest/Signature"

```
    <xs:element ref="ds:Signature" minOccurs="0"/>

  </xs:sequence>
```

"Algorithm" specifies the algorithm to use for creating an SKS "SessionKey". Currently the only documented algorithm URI is: "http://xmlns.webpki.org/keygen2/1.0#algorithm.sks.s1"

```
  <xs:attribute name="Algorithm" type="xs:anyURI" use="required"/>
```

MUST be equal to "PlatformNegotiationRequest/@ID" within a specific session

```
  <xs:attribute name="ID" type="kg2:sksID" use="required"/>
```

Mandatory time-stamp indicating when the request was issued. The provisioning middleware MAY check it for "sanity"

```
  <xs:attribute name="ServerTime" type="xs:dateTime" use="required"/>
```

"SessionLifeTime" specifies the lifetime of the SKS provisioning session in seconds

```
  <xs:attribute name="SessionLifeTime" use="required">
    <xs:simpleType>
      <xs:restriction base="xs:positiveInteger">
        <xs:minInclusive value="10"/>
        <xs:maxInclusive value="10000000"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:attribute>
```

"SessionKeyLimit" specifies how many "SessionKey" operations the SKS should permit

```
  <xs:attribute name="SessionKeyLimit" use="required">
    <xs:simpleType>
      <xs:restriction base="xs:positiveInteger">
        <xs:minInclusive value="1"/>
        <xs:maxInclusive value="65535"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:attribute>
```

"ClientAttributes" specifies additional client-attributes that may be needed for granting a certificate or be used for inclusion in issued certificates. Currently the only documented parameters have the format: "http://xmlns.webpki.org/keygen2/1.0#clientattribute.*". This request MUST NOT be emitted in the "PrivacyEnabled" mode.

If the server asks for an unknown attribute, the client SHOULD continue anyway but (of course) not provide an element in the response for the particular attribute. See "ProvisioningInitializationResponse/ClientAttribute".

In case it is not obvious that the client supports an attribute, the server SHOULD specify those in "BasicCapabilities@ClientAttributes"

```
<xs:attribute name="ClientAttributes" type="kg2:uriList" use="optional"/>
```

Where to HTTP POST the provisioning session response. Note that applicable session cookies MUST be submitted in the request as well.

This instance of "SubmitURL" is equivalent to SKS' "IssuerURI".

```
<xs:attribute name="SubmitURL" type="kg2:httpURI" use="required"/>
```

```
  </xs:complexType>
</xs:element>
```

End of provisioning initialization request object definition

The primary (top-level) provisioning initialization response object

```
<xs:element name="ProvisioningInitializationResponse">
  <xs:complexType>
    <xs:sequence>
```

"ClientEphemeralKey" holds an SKS-generated ephemeral key for creating an SKS "SessionKey" (shared session key)

```
      <xs:element name="ClientEphemeralKey">
        <xs:complexType>
          <xs:sequence>
```

MUST contain an ECDH public key using the same elliptic curve as "ServerEphemeralKey"

```
            <xs:element ref="ds11:ECKeyValue"/>

          </xs:sequence>
        </xs:complexType>
      </xs:element>
```

"DeviceCertificatePath" holds the SKS device certificate path. Note: This element MUST be defined when "PrivacyEnabled" is false whereas it MUST NOT be defined when "PrivacyEnabled" is true

```
      <xs:element name="DeviceCertificatePath" minOccurs="0">
        <xs:complexType>
          <xs:sequence>
```

MUST contain a valid X509 certificate path[1..n]

```
            <xs:element ref="ds:X509Data"/>

          </xs:sequence>
        </xs:complexType>
      </xs:element>
```

"ClientAttribute" holds an optional set of attributes requested by the server in "ProvisioningInitializationRequest@ClientAttributes". Multi-valued attribute types MAY

```
        be repeated.  There MUST NOT be any "ClientAttribute" elements when "PrivacyEnabled" is true
      <xs:element name="ClientAttribute" minOccurs="0" maxOccurs="unbounded">
        <xs:complexType>
          <xs:attribute name="Type" type="xs:anyURI" use="required"/>
          <xs:attribute name="Value" type="xs:string" use="required"/>
        </xs:complexType>
      </xs:element>

        MUST hold an enveloped XML symmetric-key signature created by the SKS "signProvisioningSessionData" method and use "ID" as the "Reference" URI attribute
      <xs:element ref="ds:Signature"/>

    </xs:sequence>

      "ID" holds the SKS-generated ClientSessionID
    <xs:attribute name="ID" type="kg2:sksID" use="required"/>

      MUST be equal to "ProvisioningInitializationRequest/@ID"
    <xs:attribute name="ServerSessionID" type="kg2:sksNCName" use="required"/>

      Copy of "ProvisioningInitializationRequest/@ServerTime"
    <xs:attribute name="ServerTime" type="xs:dateTime" use="required"/>

      The provisioning client's local clock is a mandatory item
    <xs:attribute name="ClientTime" type="xs:dateTime" use="required"/>

      The "Attestation" attribute holds the output from the SKS "createProvisioningSession" method
    <xs:attribute name="Attestation" type="xs:base64Binary" use="required"/>

      The "ServerCertificateFingerprint" attribute MUST be populated with the SHA256 hash of the associated server certificate if the "ProvisioningInitializationResponse" was invoked over HTTPS
    <xs:attribute name="ServerCertificateFingerprint" type="xs:base64Binary" use="optional"/>

  </xs:complexType>
</xs:element>

        End of provisioning initialization response object definition


        The primary (top-level) [optional] credential discovery request object

<xs:element name="CredentialDiscoveryRequest">
  <xs:complexType>
    <xs:sequence>

      "LookupSpecifier" holds an issuer-signed search object
      <xs:element name="LookupSpecifier" maxOccurs="unbounded">
        <xs:complexType>
          <xs:sequence>

            Optional search filter
            <xs:element ref="kg2:SearchFilter" minOccurs="0"/>
```

```
                MUST contain the issuer KM signature
                <xs:element ref="ds:Signature"/>

              </xs:sequence>

              MUST be be linked to by the enveloped signature element
              <xs:attribute name="ID" type="kg2:sksID" use="required"/>

              Freshness nonce to bind discovery to the session
              <xs:attribute name="Nonce" type="xs:base64Binary" use="required"/>

            </xs:complexType>
          </xs:element>

          See "PlatformNegotiationRequest/Signature"
          <xs:element ref="ds:Signature" minOccurs="0"/>

        </xs:sequence>

        MUST be equal to "PlatformNegotiationRequest/@ID" within a specific session
        <xs:attribute name="ID" type="kg2:sksID" use="required"/>

        MUST be equal to"ProvisioningInitializationResponse/@ID" within a specific session
        <xs:attribute name="ClientSessionID" type="kg2:sksNCName" use="required"/>

        Where to HTTP POST the credential discovery response.  Note that applicable session
        cookies MUST be submitted in the request as well
        <xs:attribute name="SubmitURL" type="kg2:httpURI" use="required"/>

    </xs:complexType>
</xs:element>

                End of [optional] credential discovery request object definition


                The primary (top-level) [optional] credential discovery response object

<xs:element name="CredentialDiscoveryResponse">
    <xs:complexType>
        <xs:sequence>

          The number of "LookupResult" objects MUST match the number of "LookupSpecifier"
          objects
          <xs:element name="LookupResult" maxOccurs="unbounded">
            <xs:complexType>
              <xs:sequence>

                Matching credentials...
                <xs:element ref="kg2:MatchingCredential" minOccurs="0" maxOccurs="unbounded"/>

              </xs:sequence>

              MUST match "LookupSpecifier@ID"
              <xs:attribute name="ID" type="kg2:sksID" use="required"/>
```

```
            </xs:complexType>
          </xs:element>

        </xs:sequence>
```

"ID" holds the SKS-generated ClientSessionID

```
        <xs:attribute name="ID" type="kg2:sksID" use="required"/>
```

MUST be equal to "ProvisioningInitializationRequest/@ID"

```
        <xs:attribute name="ServerSessionID" type="kg2:sksNCName" use="required"/>

    </xs:complexType>
  </xs:element>
```

End of [optional] credential discovery response object definition

The primary (top-level) key initialization request object

```
  <xs:element name="KeyCreationRequest">
    <xs:complexType>
      <xs:sequence>

        <xs:choice maxOccurs="unbounded">
```

A key that requires no PIN code

```
          <xs:element ref="kg2:KeyEntry"/>
```

One or more keys that are protected by a specific PIN policy. In addition to the PIN there is a master PUK code that can unlock keys that have been locked due to the use of erroneous PIN codes

```
          <xs:element ref="kg2:PUKPolicy"/>
```

One or more keys that are protected by a specific PIN policy

```
          <xs:element ref="kg2:PINPolicy"/>
```

A key that is protected by the same PIN as the device. This option (which MAY be supported by the device), is mainly intended for keys belonging to an organization that manages the device

```
          <xs:element ref="kg2:DevicePINProtection"/>

        </xs:choice>
```

See "PlatformNegotiationRequest/Signature"

```
        <xs:element ref="ds:Signature" minOccurs="0"/>

      </xs:sequence>
```

A KeyGen2 provisioning process MAY [in a controlled way] be temporarily interrupted and then be resumed at a later stage like required by some certification schemes. The attribute below informs the client software if the current process is to be terminated after the "KeyCreationResponse" message has been sent. The resumed session MUST use Action="resume". Default: false

```
      <xs:attribute name="DeferredCertification" type="xs:boolean" use="optional"/>
```

```
      MUST be equal to "PlatformNegotiationRequest/@ID" within a specific session
      <xs:attribute name="ID" type="kg2:sksID" use="required"/>

      MUST be equal to"ProvisioningInitializationResponse/@ID" within a specific session
      <xs:attribute name="ClientSessionID" type="kg2:sksNCName" use="required"/>

      Where to HTTP POST the key operation response.  Note that applicable session cookies
      MUST be submitted in the request as well
      <xs:attribute name="SubmitURL" type="kg2:httpURI" use="required"/>

      "Algorithm" is mandatory.  Currently the only documented key generation and attestation
      algorithm is: "http://xmlns.webpki.org/keygen2/1.0#algorithm.sks.k1"
      <xs:attribute name="Algorithm" type="xs:anyURI" use="required"/>

    </xs:complexType>
  </xs:element>

                End of key initialization request object definition

                The primary (top-level) key initialization response object

  <xs:element name="KeyCreationResponse">
    <xs:complexType>
      <xs:sequence>

        The keys to be certified
        <xs:element ref="kg2:PublicKey" maxOccurs="unbounded"/>

      </xs:sequence>

      MUST be equal to "PlatformNegotiationResponse/@ID" within a specific session
      <xs:attribute name="ID" type="kg2:sksID" use="required"/>

      MUST be equal to "PlatformNegotiationRequest/@ID" within a specific session
      <xs:attribute name="ServerSessionID" type="kg2:sksNCName" use="required"/>

    </xs:complexType>
  </xs:element>

                End of key initialization response object definition

                The primary (top-level) provisioning finalization request object

  <xs:element name="ProvisioningFinalizationRequest">
    <xs:complexType>
      <xs:sequence>

        The now certified keys.  Note: optional
        <xs:element ref="kg2:CertificatePath" minOccurs="0" maxOccurs="unbounded"/>

        SKS Reference: postUnlockKey
```

```
        <xs:element name="UnlockKey" type="kg2:targetKey" minOccurs="0" maxOccurs="unbounded"/>

        SKS Reference: postDeleteKey
        <xs:element name="DeleteKey" type="kg2:targetKey" minOccurs="0" maxOccurs="unbounded"/>

        See "PlatformNegotiationRequest/Signature"
        <xs:element ref="ds:Signature" minOccurs="0"/>

      </xs:sequence>

      MUST be equal to "ProvisioningInitializationResponse/@ID" within a specific session
      <xs:attribute name="ClientSessionID" type="kg2:sksNCName" use="required"/>

      MUST be equal to "PlatformNegotiationRequest/@ID" within a specific session
      <xs:attribute name="ID" type="kg2:sksID" use="required"/>

      Where to POST the response
      <xs:attribute name="SubmitURL" type="kg2:httpURI" use="required"/>

      SKS Reference: closeProvisioningSession
      <xs:attribute name="MAC" type="xs:base64Binary" use="required"/>

      SKS Reference: closeProvisioningSession
      <xs:attribute name="Nonce" type="xs:base64Binary" use="required"/>

    </xs:complexType>
  </xs:element>

  End of provisioning finalization request object definition

  The primary (top-level) provisioning finalization response object

  <xs:element name="ProvisioningFinalizationResponse">
    <xs:complexType>

      MUST be equal to "PlatformNegotiationResponse/@ID" within a specific session
      <xs:attribute name="ID" type="kg2:sksID" use="required"/>

      MUST be equal to "PlatformNegotiationRequest/@ID" within a specific session
      <xs:attribute name="ServerSessionID" type="kg2:sksNCName" use="required"/>

      SKS Reference: closeProvisioningSession
      <xs:attribute name="Attestation" type="xs:base64Binary" use="required"/>

    </xs:complexType>
  </xs:element>

  End of provisioning finalization object definition

  Object representing basic algorithm/capability requirements/support
```

```
<xs:attributeGroup name="basicCapabilities">
```

> If non-mandatory algorithms are preferred, the following attribute can host a list of such. If the response does not contain a requested algorithm it means that it is not supported

```
  <xs:attribute name="Algorithms" type="kg2:uriList" use="optional"/>
```

> "ClientAttributes" are things that are supported by the SKS or the platform the SKS is running in

```
  <xs:attribute name="ClientAttributes" type="kg2:uriList" use="optional"/>
```

> Arbitrary extension objects MAY be requested. Although most clients are likely to for example support HOTP ("http://xmlns.webpki.org/keygen2/1.0#provider.ietf-hotp"), application support is outside of the core SKS specification and SHOULD be requested rather than taken for granted. A provisioning client MAY dynamically check for application availability

```
  <xs:attribute name="Extensions" type="kg2:uriList" use="optional"/>

</xs:attributeGroup>
```

> Base-object for representing a certified public key, path, and optional attributes

```
<xs:element name="CertificatePath">
  <xs:complexType>
    <xs:sequence>
```

> SKS Reference: setCertificatePath. The certificate collection MAY contain additional certificates belonging to the same certificate-path. There MUST NOT be any "holes" or "extras" in the path but it does not need to go up to the root. The actual order of certificates is insignificant. NOTE: Unlike TLS server-certificates, the KeyGen2 scheme does not put any constraints on issued certificates with respect to being known or trusted by the client platform. That is, all technically valid certificates MUST be accepted without warnings. Individual certificates in excess of 16KB MAY though rejected

```
      <xs:element ref="ds:X509Data"/>

      <xs:choice minOccurs="0">
```

> SKS Reference: importSymmetricKey
> Optional imported symmetric key

```
        <xs:element name="SymmetricKey">
          <xs:complexType>
            <xs:simpleContent>
              <xs:extension base="xs:base64Binary">
```

> Mandatory MAC (Message Authentication Code)

```
                <xs:attribute name="MAC" type="xs:base64Binary" use="required"/>

              </xs:extension>
            </xs:simpleContent>
          </xs:complexType>
        </xs:element>
```

> SKS Reference: importPrivateKey
> Optional imported private key

```
        <xs:element name="PrivateKey">
          <xs:complexType>
            <xs:simpleContent>
              <xs:extension base="xs:base64Binary">
```

> Mandatory MAC (Message Authentication Code)

```
            <xs:attribute name="MAC" type="xs:base64Binary" use="required"/>

          </xs:extension>
        </xs:simpleContent>
      </xs:complexType>
    </xs:element>

  </xs:choice>

  <xs:choice minOccurs="0" maxOccurs="unbounded">
```

SKS Reference: addExtension
Some credentials like OTP seeds need additional attributes to work. These are supplied as typed "PropertyBags" holding "Property" name-value elements

```
    <xs:element ref="kg2:PropertyBag"/>
```

SKS Reference: addExtension
Optional logotypes that can be used in various GUIs. The "Type" URI indicates a logotype's usage and SHOULD be communicated during the platform negotiation phase. Logotypes MAY be ignored by the client software. Currently KeyGen2 supports the following predefined type URIs:

- "http://xmlns.webpki.org/keygen2/1.0#logotype.icon"
- "http://xmlns.webpki.org/keygen2/1.0#logotype.card"
- "http://xmlns.webpki.org/keygen2/1.0#logotype.list"
- "http://xmlns.webpki.org/keygen2/1.0#logotype.application"

"Icon" is intended to be useful in a selection window where multiple credentials are featured as a map of icons. "Card" is the shape needed by for example Microsoft's InformationCards. "List" is meant to be used in credential listings where you need a minute image together with explanatory text. "Application" is a shape that could be used in applications where the logotype is useful for branding/recognition purposes like in OTP systems.

Logotype recommendations:
Logotypes should not have framing borders or extra margins unless these are integral parts of the actual logotype image.
Logotypes should render nicely on light backgrounds.
Logotype "surroundings" should preferably be transparent to accommodate white to light grey backgrounds. If there are shadows that should migrate with the background, PNG files with alpha channels are recommended

```
    <xs:element name="Logotype">
      <xs:complexType>
        <xs:simpleContent>
          <xs:extension base="kg2:imageType">
```

Mandatory extension type attribute

```
            <xs:attributeGroup ref="kg2:extensionAttrs"/>

          </xs:extension>
        </xs:simpleContent>
      </xs:complexType>
    </xs:element>
```

SKS Reference: addExtension

```
    <xs:element name="Extension">
      <xs:complexType>
        <xs:simpleContent>
          <xs:extension base="xs:base64Binary">
```

Mandatory extension type attribute

```
            <xs:attributeGroup ref="kg2:extensionAttrs"/>

          </xs:extension>
        </xs:simpleContent>
      </xs:complexType>
    </xs:element>
```

```
            SKS Reference: addExtension

            <xs:element name="EncryptedExtension">
              <xs:complexType>
                <xs:simpleContent>
                  <xs:extension base="xs:base64Binary">

                    Mandatory extension type attribute

                    <xs:attributeGroup ref="kg2:extensionAttrs"/>

                  </xs:extension>
                </xs:simpleContent>
              </xs:complexType>
            </xs:element>

          </xs:choice>

          <xs:choice minOccurs="0">

            SKS Reference: postUpdateKey

            <xs:element name="UpdateKey" type="kg2:targetKey"/>

            SKS Reference: postCloneKeyProtection

            <xs:element name="CloneKeyProtection" type="kg2:targetKey"/>

          </xs:choice>

        </xs:sequence>

        SKS Reference: setCertificatePath.ID
        A unique key-ID which MUST match the corresponding
        "KeyCreationResponse/PublicKey/@ID"

        <xs:attribute name="ID" type="kg2:sksID" use="required"/>

        SKS Reference: setCertificatePath.MAC

        <xs:attribute name="MAC" type="xs:base64Binary" use="required"/>

        If the issuer wants the user to "install" the uppermost certificate as a "trust anchor" it may
        set this attribute to "true".  Note that a conforming client implementation MUST make this
        setting as a user GUI option.  It is an error if the uppermost certificate is not a CA
        certificate but it doesn't have to be a self-signed root.  Note that the provisioning process is
        independent of the user's choice.  Default: false

        <xs:attribute name="TrustAnchor" type="xs:boolean" use="optional"/>

      </xs:complexType>

      Constraints (that can be enforced by schema-directed validation)

      All Extension type URIs for a key MUST be a unique

      <xs:unique name="ExtensionTypeUnique">
        <xs:selector xpath="*"/>
        <xs:field xpath="@Type"/>
      </xs:unique>

    </xs:element>

    Object for representing properties
```

```
<xs:element name="PropertyBag">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="Property" maxOccurs="unbounded">
        <xs:complexType>

          The unique name of the property

          <xs:attribute name="Name" use="required">
            <xs:simpleType>
              <xs:restriction base="xs:string">
                <xs:minLength value="1"/>
                <xs:maxLength value="255"/>
              </xs:restriction>
            </xs:simpleType>
          </xs:attribute>

          The value of the property

          <xs:attribute name="Value" type="xs:string" use="required"/>

          By default values are read-only but they may be declared as read/writable as well

          <xs:attribute name="Writable" type="xs:boolean" use="optional"/>

        </xs:complexType>
      </xs:element>
    </xs:sequence>

    Mandatory extension attributes

    <xs:attributeGroup ref="kg2:extensionAttrs"/>

  </xs:complexType>

  Property names within a PropertyBag MUST be unique

  <xs:unique name="Unique.Property">
    <xs:selector xpath="kg2:Property"/>
    <xs:field xpath="@Name"/>
  </xs:unique>

</xs:element>

Base-object for representing a created public key

<xs:element name="PublicKey">
  <xs:complexType>
    <xs:sequence>

      SKS Reference: createKeyEntry.PublicKey

      <xs:choice>

        <xs:element ref="ds11:ECKeyValue"/>

        <xs:element ref="ds:RSAKeyValue"/>

      </xs:choice>

    </xs:sequence>

    SKS Reference: createKeyEntry.Attestation

    <xs:attribute name="Attestation" type="xs:base64Binary" use="optional"/>
```

A unique key-ID which MUST match a "KeyEntry/@ID" in the corresponding "KeyCreationRequest"

```
<xs:attribute name="ID" type="kg2:sksID" use="required"/>

  </xs:complexType>
</xs:element>
```

Base-object for requesting the creation of a key entry and associated key pair

```
<xs:element name="KeyEntry">
  <xs:complexType>
```

SKS Reference: createKeyEntry.ID

```
<xs:attribute name="ID" type="kg2:sksID" use="required"/>
```

SKS Reference: createKeyEntry.ServerSeed
Default: 0 bytes

```
<xs:attribute name="ServerSeed" use="optional">
  <xs:simpleType>
    <xs:restriction base="xs:base64Binary">
      <xs:minLength value="1"/>
      <xs:maxLength value="32"/>
    </xs:restriction>
  </xs:simpleType>
</xs:attribute>
```

SKS Reference: createKeyEntry.BiometricProtection
Default: "none"

```
<xs:attribute name="BiometricProtection" type="kg2:biomProtTypes" use="optional"/>
```

SKS Reference: createKeyEntry.ExportProtection
Default: "non-exportable"

```
<xs:attribute name="ExportProtection" type="kg2:expProtectionTypes" use="optional"/>
```

SKS Reference: createKeyEntry.DeleteProtection
Default: "none"

```
<xs:attribute name="DeleteProtection" type="kg2:delProtectionTypes" use="optional"/>
```

SKS Reference: createKeyEntry.EnablePINCaching
Default: false

```
<xs:attribute name="EnablePINCaching" type="xs:boolean" use="optional"/>
```

SKS Reference: createKeyEntry.AppUsage

```
<xs:attribute name="AppUsage" type="kg2:appUsageTypes" use="required"/>
```

SKS Reference: createKeyEntry.KeyAlgorithm

```
<xs:attribute name="KeyAlgorithm" type="xs:anyURI" use="required"/>
```

SKS Reference: createKeyEntry.KeyParameters

```
<xs:attribute name="KeyParameters" type="xs:base64Binary" use="optional"/>
```

SKS Reference: createKeyEntry.FriendlyName
Default: ""

```
<xs:attribute name="FriendlyName" type="kg2:friendlyNameType" use="optional"/>
```

SKS Reference: createKeyEntry.MAC

```
<xs:attribute name="MAC" type="xs:base64Binary" use="required"/>
```

SKS Reference: createKeyEntry.EndorsedAlgorithm

```
<xs:attribute name="EndorsedAlgorithms" type="kg2:uriList" use="optional"/>
```

```
  </xs:complexType>
</xs:element>
```

```
<xs:attributeGroup name="extensionAttrs">
```

SKS Reference: addExtension.Type

```
  <xs:attribute name="Type" type="xs:anyURI" use="required"/>
```

SKS Reference: addExtension.MAC

```
  <xs:attribute name="MAC" type="xs:base64Binary" use="required"/>
```

```
</xs:attributeGroup>
```

Request sub-object for creating a set of PIN-protected keys associated with a a common master PUK code

```
<xs:element name="PUKPolicy">
  <xs:complexType>
    <xs:sequence>
```

The PIN-protected key groups

```
      <xs:element ref="kg2:PINPolicy" maxOccurs="unbounded"/>
```

```
    </xs:sequence>
```

SKS Reference: createPUKPolicy.ID

```
    <xs:attribute name="ID" type="kg2:sksID" use="required"/>
```

SKS Reference: createPUKPolicy.PUKValue

```
    <xs:attribute name="Value" type="xs:base64Binary" use="required"/>
```

SKS Reference: createPUKPolicy.Format

```
    <xs:attribute name="Format" type="kg2:pinFormats" use="required"/>
```

SKS Reference: createPUKPolicy.RetryLimit

```
    <xs:attribute name="RetryLimit" type="xs:unsignedShort" use="required"/>
```

SKS Reference: createPUKPolicy.MAC

```
    <xs:attribute name="MAC" type="xs:base64Binary" use="required"/>
```

```
  </xs:complexType>
</xs:element>
```

Request sub-object for creating a set of keys associated with a user PIN policy

```
<xs:element name="PINPolicy">
  <xs:complexType>
    <xs:sequence>
```

```
    <xs:choice>

      One or more keys protected by user-defined PINs governed by a common PIN policy

      <xs:element ref="kg2:KeyEntry" maxOccurs="unbounded"/>

      One or more keys protected by issuer-defined PINs

      <xs:element name="PresetPIN" maxOccurs="unbounded">
        <xs:complexType>

          <xs:sequence>
            <xs:element ref="kg2:KeyEntry"/>
          </xs:sequence>

          SKS Reference: createKeyEntry.PINValue

          <xs:attribute name="Value" type="xs:base64Binary" use="required"/>

        </xs:complexType>
      </xs:element>

    </xs:choice>

</xs:sequence>

SKS Reference: createPINPolicy.ID

<xs:attribute name="ID" type="kg2:sksID" use="required"/>

SKS Reference: createPINPolicy.PUKPolicyHandle
"PUKPolicyHandle" is an implicit and implementation-dependent PUK reference object which is defined by the presence of an enclosing "PUKPolicy" object

SKS Reference: createPINPolicy.UserDefined
"UserDefined" is an implicit variable which is true unless there is a "PresetPIN" object enclosing the "KeyEntry" object

SKS Reference: createPINPolicy.UserModifiable
Default: false for preset PINs, true for user-defined PINs

<xs:attribute name="UserModifiable" type="xs:boolean" use="optional"/>

SKS Reference: createPINPolicy.Format

<xs:attribute name="Format" type="kg2:pinFormats" use="required"/>

SKS Reference: createPINPolicy.RetryLimit

<xs:attribute name="RetryLimit" type="xs:unsignedShort" use="required"/>

SKS Reference: createPINPolicy.Grouping
Default: "none"

<xs:attribute name="Grouping" type="kg2:pinGrouping" use="optional"/>

SKS Reference: createPINPolicy.PatternRestrictions
Default: none

<xs:attribute name="PatternRestrictions" type="kg2:pinPatternList" use="optional"/>

SKS Reference: createPINPolicy.MinLength

<xs:attribute name="MinLength" type="kg2:pinLength" use="required"/>
```

SKS Reference: createPINPolicy.MaxLength

```
<xs:attribute name="MaxLength" type="kg2:pinLength" use="required"/>
```

SKS Reference: createPINPolicy.InputMethod
Default: "any"

```
<xs:attribute name="InputMethod" type="kg2:pinInput" use="optional"/>
```

SKS Reference: createPINPolicy.MAC

```
<xs:attribute name="MAC" type="xs:base64Binary" use="required"/>
```

```
  </xs:complexType>
</xs:element>
```

Request sub-object for creating a key protected by the key "host" device PIN-code

```
<xs:element name="DevicePINProtection">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="kg2:KeyEntry"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:complexType name="imageData">
```

MIME type.

```
  <xs:attribute name="MIMEType" type="kg2:mimeType" use="required"/>
```

Logotype usage type

```
  <xs:attribute name="Type" type="xs:anyURI" use="required"/>
```

Width in pixels. For SVG it holds the "ideal" width

```
  <xs:attribute name="Width" type="xs:positiveInteger" use="required"/>
```

Height in pixels. For SVG it holds the "ideal" height

```
  <xs:attribute name="Height" type="xs:positiveInteger" use="required"/>

</xs:complexType>

<xs:complexType name="targetKey">
```

One of the two IDs identifying a provisioning session

```
  <xs:attribute name="ClientSessionID" type="kg2:sksNCName" use="required"/>
```

The other ID identifying a provisioning session

```
  <xs:attribute name="ServerSessionID" type="kg2:sksNCName" use="required"/>
```

Handle to the end entity certificate (key ID)

```
  <xs:attribute name="CertificateFingerprint" type="xs:base64Binary" use="required"/>
```

Mandatory MAC (Message Authentication Code)

```
  <xs:attribute name="MAC" type="xs:base64Binary" use="required"/>
```

```
    Authorization
    <xs:attribute name="Authorization" type="xs:base64Binary" use="required"/>

</xs:complexType>

<xs:element name="MatchingCredential">
  <xs:complexType>

    The End Entity Certificate
    <xs:attribute name="EndEntityCertificate" type="xs:base64Binary" use="optional"/>

    One of the two IDs identifying a provisioning session
    <xs:attribute name="ClientSessionID" type="kg2:sksNCName" use="required"/>

    The other ID identifying a provisioning session
    <xs:attribute name="ServerSessionID" type="kg2:sksNCName" use="required"/>

    Locked? Default: false
    <xs:attribute name="Locked" type="xs:boolean" use="optional"/>

  </xs:complexType>
</xs:element>

<xs:element name="SearchFilter">
  <xs:complexType>

    Qualifiers.  At least one MUST be defined.  Note that "IssuedBefore" and "IssuedAfter" can
    be used to create a region that should either be included or excluded depending on which
    one is biggest.  The "Subject" and "Issuer" attributes are regular expressions matching
    RFC 2253 names.  Note that the "Issuer" attribute is searched for through the entire path.
    "Email" attributes MUST be searched for in both the subject DN and in subject alt name
    extensions
    <xs:attribute name="Issuer" type="kg2:distinguishedNameExpr" use="optional"/>
    <xs:attribute name="Subject" type="kg2:distinguishedNameExpr" use="optional"/>
    <xs:attribute name="Serial" type="xs:integer" use="optional"/>
    <xs:attribute name="Email" type="xs:string" use="optional"/>
    <xs:attribute name="Policy" type="kg2:oidValue" use="optional"/>
    <xs:attribute name="ExcludedPolicies" type="kg2:oidValueList" use="optional"/>
    <xs:attribute name="IssuedBefore" type="xs:dateTime" use="optional"/>
    <xs:attribute name="IssuedAfter" type="xs:dateTime" use="optional"/>

  </xs:complexType>
</xs:element>

IDs flowing between the SKS and an issuer MUST be 1-32 bytes using a constrained character
set

<xs:simpleType name="sksID">
  <xs:restriction base="xs:ID">
    <xs:pattern value="[0-9,a-z,A-Z,_,\-,\.]*"/>
    <xs:maxLength value="32"/>
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="sksNCName">
  <xs:restriction base="xs:NCName">
    <xs:pattern value="[0-9,a-z,A-Z,_,\-,\.]*"/>
    <xs:maxLength value="32"/>
  </xs:restriction>
</xs:simpleType>
```

Image definition. An image SHOULD consist of a size-wise device-adapted GIF or PNG picture

```
<xs:complexType name="imageType">
  <xs:simpleContent>
    <xs:extension base="xs:base64Binary">
      <xs:attribute name="MIMEType" type="kg2:mimeType" use="required"/>
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>
```

Action definitions

```
<xs:simpleType name="action">
  <xs:restriction base="xs:string">
    <xs:enumeration value="create"/>
    <xs:enumeration value="unlock"/>
    <xs:enumeration value="update"/>
    <xs:enumeration value="resume"/>
  </xs:restriction>
</xs:simpleType>
```

Application usage definitions

```
<xs:simpleType name="appUsageTypes">
  <xs:restriction base="xs:string">
    <xs:enumeration value="signature"/>
    <xs:enumeration value="authentication"/>
    <xs:enumeration value="encryption"/>
    <xs:enumeration value="universal"/>
  </xs:restriction>
</xs:simpleType>
```

Biometric protection definitions

```
<xs:simpleType name="biomProtTypes">
  <xs:restriction base="xs:string">
    <xs:enumeration value="none"/>
    <xs:enumeration value="alternative"/>
    <xs:enumeration value="combined"/>
    <xs:enumeration value="exclusive"/>
  </xs:restriction>
</xs:simpleType>
```

Delete protection definitions

```
<xs:simpleType name="delProtectionTypes">
  <xs:restriction base="xs:string">
    <xs:enumeration value="none"/>
    <xs:enumeration value="pin"/>
    <xs:enumeration value="puk"/>
    <xs:enumeration value="non-deletable"/>
  </xs:restriction>
</xs:simpleType>
```

Export protection definitions

```
<xs:simpleType name="expProtectionTypes">
  <xs:restriction base="xs:string">
    <xs:enumeration value="none"/>
    <xs:enumeration value="pin"/>
    <xs:enumeration value="puk"/>
    <xs:enumeration value="non-exportable"/>
  </xs:restriction>
</xs:simpleType>
```

PIN policy definitions

```
<xs:simpleType name="pinLength">
  <xs:restriction base="xs:positiveInteger">
    <xs:maxInclusive value="100"/>
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="pinFormats">
  <xs:restriction base="xs:string">
    <xs:enumeration value="numeric"/>
    <xs:enumeration value="alphanumeric"/>
    <xs:enumeration value="string"/>
    <xs:enumeration value="binary"/>
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="pinGrouping">
  <xs:restriction base="xs:string">
    <xs:enumeration value="none"/>
    <xs:enumeration value="shared"/>
    <xs:enumeration value="signature+standard"/>
    <xs:enumeration value="unique"/>
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="pinInput">
  <xs:restriction base="xs:string">
    <xs:enumeration value="any"/>
    <xs:enumeration value="programmatic"/>
    <xs:enumeration value="trusted-gui"/>
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="pinPatternList">
  <xs:restriction>
    <xs:simpleType>
      <xs:list>
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:enumeration value="two-in-a-row"/>
            <xs:enumeration value="three-in-a-row"/>
            <xs:enumeration value="sequence"/>
            <xs:enumeration value="repeated"/>
            <xs:enumeration value="missing-group"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:list>
    </xs:simpleType>
    <xs:minLength value="1"/>
  </xs:restriction>
</xs:simpleType>
```

URI and OID definitions

```
<xs:simpleType name="oidValue">
  <xs:restriction base="xs:string">
    <xs:pattern value="[1-9][0-9]*(\.[1-9][0-9]*)*"/>
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="oidValueList">
  <xs:restriction>
    <xs:simpleType>
      <xs:list itemType="kg2:oidValue"/>
    </xs:simpleType>
    <xs:minLength value="1"/>
  </xs:restriction>
</xs:simpleType>
```

```
<xs:simpleType name="httpURI">
  <xs:restriction base="xs:anyURI">
    <xs:pattern value="https?://.*"/>
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="uriList">
  <xs:restriction>
    <xs:simpleType>
      <xs:list itemType="xs:anyURI"/>
    </xs:simpleType>
    <xs:minLength value="1"/>
  </xs:restriction>
</xs:simpleType>
```

Language list definitions

```
<xs:simpleType name="languageList">
  <xs:restriction>
    <xs:simpleType>
      <xs:list itemType="xs:language"/>
    </xs:simpleType>
    <xs:minLength value="1"/>
  </xs:restriction>
</xs:simpleType>
```

MIME type definition (MUST NOT contain character encoding)

```
<xs:simpleType name="mimeType">
  <xs:restriction base="xs:string">
    <xs:pattern value="[^;]+"/>
  </xs:restriction>
</xs:simpleType>
```

Key container definitions

```
<xs:simpleType name="containerList">
  <xs:restriction>
    <xs:simpleType>
      <xs:list>
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:enumeration value="File"/>
            <xs:enumeration value="TPM"/>
            <xs:enumeration value="SIM"/>
            <xs:enumeration value="External"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:list>
    </xs:simpleType>
    <xs:minLength value="1"/>
  </xs:restriction>
</xs:simpleType>
```

"Subject" attributes MUST be represented as regular expressions compatible with the XML Schema "pattern" facet. Note that the regular expressions MUST be targeted for RFC 4514 encoded data

```
<xs:simpleType name="distinguishedNameExpr">
  <xs:restriction base="xs:string">
    <xs:pattern value="\S.*"/>
  </xs:restriction>
</xs:simpleType>
```

Miscellenous

```
    <xs:simpleType name="friendlyNameType">
      <xs:restriction base="xs:string">
        <xs:minLength value="1"/>
        <xs:maxLength value="100"/>
      </xs:restriction>
    </xs:simpleType>

</xs:schema>
```