

## Attested Key-Pair Generation with Secure Private-Key Archival

Occasionally credential issuers want (or require) that encryption private keys are archived in order to be able to restore data (AKA "key escrow") in case an encryption-key is lost or is replaced.

This document describes how KeyGen2's key-attestation scheme has been augmented by an end-to-end-secured private-key archival mechanism as well. The concept is as follows: An issuer sends a key-generation request to a client presumably having an SE (Security Element) for keeping user-keys in. The SE responds with attested public keys where each attest consists of a signature over the *generated public key*, *key-usage*, *key-exportability*, and a *nonce*, using an embedded SE device-key as the signature key. A key-attest vouches for that the key-pair actually was generated *inside of the SE*.

By also including a key-archival public key (supplied by the issuer during the request), in the attest signature hash, the issuer gets a cryptographically strong evidence enabling it to detect if something in the transport layer or in the SE middleware temporarily swapped the archival-key in order to steal the private key and then changed it back to cover-up the operation.

A prerequisite for this to be trustworthy is that the entire operation (*key-pair generation*, *private-key archival-encryption*, and *device-key attest-signing*) is carried out inside of the SE *in a fully atomic fashion*. In addition to successful attest-signature validation, the issuer (of course) also has to recognize and trust the SE *device type* identified by the public key certificate received in the response.

For details on the original key-attestation scheme see: <http://webpki.org/papers/keygen2/keygen2-fips140-2.pdf>

Note that although the referred document exploits KeyGen2's DIAS (Device Internal Attestation Signature) single device-key scheme, the core principle is equally applicable to designs relying on separate device-keys for attestations, authentications, and encryptions performed by an SE.

### Request Phase

The following is a simple sample of a KeyGen2 key generation request where the issuer asks for two keys; an encryption key with private-key archival and an authentication without private-key backup (actually KeyGen2 only allows private-key archival for encryption keys).

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

    <KeyPair ID="Key.1" KeyUsage="encryption">
      <RSA KeySize="2048"/>
      <PrivateKeyArchivalKey>
        <ds:X509Data>
          <ds:X509Certificate>MIIB2TCCAUKg ... FLlck8kmFSX</ds:X509Certificate>
        </ds:X509Data>
      </PrivateKeyArchivalKey >
    </KeyPair>

    <KeyPair ID="Key.2" KeyUsage="authentication">
      <RSA KeySize="2048"/>
    </KeyPair>

  </CreateObject>
</KeyOperationRequest>
```

#### IPR Declaration

*This specification is hereby put in the public domain.  
It does to the author's knowledge not infringe on any  
existing patent.*

## Response Phase

Below is a condensed version of the anticipated KeyGen2 response. The only thing that differs from a regular response is the element `EncryptedPrivateKey` which contains a copy of the generated private key in PKCS #8 `PrivateKeyInfo` format, encrypted by a 256-bit random AES key, which in turn is wrapped by the issuer-supplied archival public key. The `EndorsementKey` contains the device public key as well as an enveloped signature over the entire response.

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

  <GeneratedPublicKey ID="Key.1" KeyAttestation="H8USgM2aQG7 ... ZixaYB5lymZ4YVw==">
    <ds:KeyInfo>
      <ds:KeyValue>
        <ds:RSAKeyValue>
          <ds:Modulus>AKu4uVZ8nYvgf ... 4mjS422CqxwAt</ds:Modulus>
          <ds:Exponent>AQAB</ds:Exponent>
        </ds:RSAKeyValue>
      </ds:KeyValue>
    </ds:KeyInfo>
    <EncryptedPrivateKey Format="http://xmlns.webpki.org/keygen2/1.0#format.pkcs8">
      <xenc:EncryptedKey>
        <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#aes256-cbc"/>
        <ds:KeyInfo>
          <ds:KeyName>Key.1.Private</ds:KeyName>
        </ds:KeyInfo>
        <xenc:CipherData>
          <xenc:CipherValue>mWi94IJvPipsM ... Xk/HbWM2w6do=</xenc:CipherValue>
        </xenc:CipherData>
      </xenc:EncryptedKey>
      <xenc:EncryptedKey>
        <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
        <xenc:CipherData>
          <xenc:CipherValue>RbDx5WbNn6 ... D3GjL7qlhD59s=</xenc:CipherValue>
        </xenc:CipherData>
        <xenc:CarriedKeyName>Key.1.Private</xenc:CarriedKeyName>
      </xenc:EncryptedKey>
    </EncryptedPrivateKey>
  </GeneratedPublicKey>

  <GeneratedPublicKey ID="Key.2" KeyAttestation="k6QNThjqMu2 ... dUpRfazW+5lyTA==">
    <ds:KeyInfo>
      <ds:KeyValue>
        <ds:RSAKeyValue>
          <ds:Modulus>AIOwQTjSE8qyZ ... Dfi2WeJdgLQ2k=</ds:Modulus>
          <ds:Exponent>AQAB</ds:Exponent>
        </ds:RSAKeyValue>
      </ds:KeyValue>
    </ds:KeyInfo>
  </GeneratedPublicKey>

  <EndorsementKey KeyAttestationAlgorithm="http://xmlns.webpki.org/keygen2/1.0#algorithm.key-attestation-1">
    <ds:Signature>
      .
      .
    </ds:Signature>
  </EndorsementKey>
</KeyOperationResponse>
```