PKCS#11 Extensions Proposal

Bruno Couillard, CTO
Chrysalis-ITS

bcouillard@chrysalis-its.com
www.chrysalis-its.com

Agenda

- PKCS#11 Amendment for Elliptic Curve Cryptography (ECC)
- Trusted Keys
- Mechanism Proliferation
- ASN.1 encoding/decoding
- New Functions

Amendment for ECC

- Generalizes the definition of Public/Private key objects:
  CKK_EC = CKK_ECDSA
- Generalizes the mechanism for key generation:
  CKM_EC_KEY_PAIR_GEN = CKM_ECDSA_KEY_PAIR_GEN
- Generalizes the public key parameters:
  CKA_EC_PARAMS = CKA_ECDSA_PARAMS
- Extends the definition of public key parameters:

```
Parameters ::= CHOICE {
  ecParameters      ECParameters,
  namedCurve  CURVES.&id({CurveNames}),
  implicitlyCA      NULL}
```

- Defines one new return value:
  CKR_KEY_PARAMS_INVALID

Amendment for ECC (cont'd)

- Defines three new key derivation mechanisms :
  - **CKM_ECDH1_DERIVE** is a mechanism for key derivation based on the Diffie-Hellman version of the elliptic curve key agreement scheme, where each party contributes one key pair.
  - **CKM_ECDH1_COFACTOR_DERIVE** is a mechanism for key derivation based on the cofactor Diffie-Hellman version of the elliptic curve key agreement scheme, where each party contributes one key pair.
  - **CKM_ECMQV_DERIVE** is a mechanism for key derivation based the MQV version of the elliptic curve key agreement scheme, where each party contributes two key pairs.

Amendment for ECC (cont'd)

- Defines three new mechanism parameters:
  - **CK_EC_KDF_TYPE** is used to indicate the Key Derivation Function (KDF) applied to derive keying data from a shared secret.
  - **CK_ECDH1_DERIVE_PARAMS** is a structure that provides the parameters to elliptic curve Diffie-Hellman (ECDH) key derivation mechanisms, where each party contributes one key pair.
  - **CK_ECMQV_DERIVE_PARAMS** is a structure that provides the parameters to elliptic curve Menezes-Qu-Vanstone (ECMQV) key derivation mechanisms, where each party contributes two key pairs.
- Adds the support for the wrapping/unwrapping of EC private keys.

Trusted Keys

- Who needs this?
- How:
  - New key attribute – CKA_TRUSTED
  - New semantic for certain operations with trusted keys
  - Need for an authenticated public key wrapping of symmetric keys
  - Need to address initial setup of trust anchor

Mechanisms Proliferation

- Currently, each new schemes requires new mechanisms
  - CKM_MD2_RSA_PKCS, CKM_MD5_RSA_PKCS …
- Should we define “Primitive” mechanisms like:
  - CKM_RSA_PKCS, CKM_MD5
- And allow for “Schemes” to be constructed like:
  - CKS_SIGN_AND_DIGEST
- Where the “Schemes” would have as parameters “Primitive” mechanisms.
- PRO:
  - Allows new “schemes” to be added without changes to the spec.
- CON:
  - Harder to test conformance
  - “Scheme” discovery an issue.

ASN.1 Encoding/Decoding

- More and more crypto functionality requires ASN.1 handling
- Should we allow for PKCS#11 to offer ASN.1 encoding/decoding

New Functions

- Single function call (for use in hardware accelerators):
  - C_SignOnePass
    - Could be used in OCSP responders
    - Could be used for SSL
  - C_DecryptOnePass
    - Could be used in SSL