Совместимость объектов rtpkcs11ecp и CryptoAPI

Общая информация

Термины и определения:

rtpkcs11ecp – библиотека, разработанная на базе стандарта PKCS#11, для взаимодействия с устройством Рутокен. Библиотека предоставляет набор функций для выполнения криптографических преобразований независимо от конкретной аппаратной реализации устройства Рутокен и программной среды.
Объекты rtpkcs11ecp – в данной статье этим термином обозначены "открытый", "закрытый" ключи и сертификат.

rtpkcs11ecp

Для обеспечения совместимости с CryptoAPI объекты rtpkcs11ecp должны соответствовать требованиям:

- атрибуты идентификаторов (**CKA_ID)** "открытого", "закрытого" ключа и их сертификата должны иметь одинаковое значение и быть отличны от CKA_ID других объектов rtpkcs11ecp;
- атрибуты "открытого" (**CKA_ENCRYPT**) и "закрытого" (**CKA_DECRYPT**) ключа одновременно должны принимать одно из значений:
  - **TRUE**. Ключ можно использовать для шифрования (расшифрования) данных;
  - **FALSE**. Ключ не используется для шифрования (расшифрования) данных.
- перечень основных атрибутов объектов rtpkcs11ecp должен соответствовать приведенным в разделе "Шаблоны rtpkcs11ecp".

ВНИМАНИЕ!

- Все операции по генерации (импорту) объектов rtpkcs11ecp должны производиться ТОЛЬКО после аутентификации пользователя в устройстве Рутокен.
- Импорт объектов rtpkcs11ecp должен осуществляется в следующем порядке:
  a) "закрытый" ключ;
  b) "открытый" ключ;
  c) сертификат.

CryptoAPI

CryptoAPI для работы с объектами rtpkcs11ecp создает контейнеры.
Один контейнер содержит одну ключевую пару и ее сертификат. Допускается наличие в контейнере только ключевой пары, без сертификата.
Если ключевой паре соответствует несколько сертификатов, в контейнере отображается только последний выданный.
В CryptoAPI реализовано два вида контейнеров:

- **AT_KEYEXCHANGE**, содержащий объекты rtpkcs11ecp со значением атрибутов **CKA_ENCRYPT** и **CKA_DECRYPT** равным **TRUE**. Может использоваться для шифрования (расшифрования) данных и электронной подписи документов;
- **AT_SIGNATURE**, содержащий объекты rtpkcs11ecp со значением атрибутов **CKA_ENCRYPT** и **CKA_DECRYPT** равным **FALSE**. Используется только для электронной подписи документов.

ВНИМАНИЕ!
CryptoAPI не отображает:

- "открытый" или "закрытый" ключ без наличия на устройстве Рутокен его ключевой пары;
- сертификат без его ключевой пары.

Шаблоны rtpkcs11ecp

Шаблоны генерации ключей для криптоалгоритма RSA

Шаблон генерации "закрытого" ключа для расшифрования данных

```
CK_ATTRIBUTE privateKeyTemplate[] =
{
    { CKA_CLASS, &privateKeyObject, sizeof(privateKeyObject)}, //  -
    { CKA_LABEL, &privateKeyLabelRsa, sizeof(privateKeyLabelRsa) - 1}, //
    { CKA_ID, &keyPairIdRsa, sizeof(keyPairIdRsa) - 1}, //    (   ""  "" )
    { CKA_KEY_TYPE, &keyTypeRsa, sizeof(keyTypeRsa)}, //   - RSA
    { CKA_DECRYPT, &attributeTrue, sizeof(attributeTrue)}, //
    { CKA_TOKEN, &attributeTrue, sizeof(attributeTrue)}, //
    { CKA_PRIVATE, &attributeTrue, sizeof(attributeTrue)} //        };
```

Шаблон генерации "открытого" ключа для шифрования данных

```
CK_ATTRIBUTE publicKeyTemplate[] =
{
    { CKA_CLASS, &publicKeyObject, sizeof(publicKeyObject)}, //  -
    { CKA_LABEL, &publicKeyLabelRsa, sizeof(publicKeyLabelRsa) - 1}, //
    { CKA_ID, &keyPairIdRsa, sizeof(keyPairIdRsa) - 1}, //    (   ""  "" )
    { CKA_KEY_TYPE, &keyTypeRsa, sizeof(keyTypeRsa)}, //   - RSA
    { CKA_TOKEN, &attributeTrue, sizeof(attributeTrue)}, //
    { CKA_ENCRYPT, &attributeTrue, sizeof(attributeTrue)}, //
    { CKA_PRIVATE, &attributeFalse, sizeof(attributeFalse)}, //
    { CKA_MODULUS_BITS, &rsaModulusBits, sizeof(rsaModulusBits)} //    };
```

Шаблон генерации "закрытого" ключа для ЭП

```
CK_ATTRIBUTE privateKeyTemplate[] =
{
    { CKA_CLASS, &privateKeyObject, sizeof(privateKeyObject)}, //  -
    { CKA_LABEL, &privateKeyLabelRsa, sizeof(privateKeyLabelRsa) - 1}, //
    { CKA_ID, &keyPairIdRsa, sizeof(keyPairIdRsa) - 1}, //    (   ""  "" )
    { CKA_KEY_TYPE, &keyTypeRsa, sizeof(keyTypeRsa)}, //   - RSA
    { CKA_DECRYPT, &attributeFalse, sizeof(attributeFalse)}, //
    { CKA_TOKEN, &attributeTrue, sizeof(attributeTrue)}, //
    { CKA_PRIVATE, &attributeTrue, sizeof(attributeTrue)} //        };
```

Шаблон генерации "открытого" ключа для ЭП

```
CK_ATTRIBUTE publicKeyTemplate[] =
{
    { CKA_CLASS, &publicKeyObject, sizeof(publicKeyObject)}, //  -
    { CKA_LABEL, &publicKeyLabelRsa, sizeof(publicKeyLabelRsa) - 1}, //
    { CKA_ID, &keyPairIdRsa, sizeof(keyPairIdRsa) - 1}, //    (   ""  "" )
    { CKA_KEY_TYPE, &keyTypeRsa, sizeof(keyTypeRsa)}, //   - RSA
    { CKA_TOKEN, &attributeTrue, sizeof(attributeTrue)}, //
    { CKA_ENCRYPT, &attributeFalse, sizeof(attributeFalse)}, //
    { CKA_PRIVATE, &attributeFalse, sizeof(attributeFalse)}, //
    { CKA_MODULUS_BITS, &rsaModulusBits, sizeof(rsaModulusBits)} //    };
```

Шаблоны импорта сертификата и ключей для криптоалгоритма RSA

Шаблон импорта "закрытого" ключа для расшифрования данных

```
CK_ATTRIBUTE privateKeyTemplate[] =
{
    { CKA_CLASS, &privateKeyObject, sizeof(privateKeyObject)}, //  -
    { CKA_LABEL, &privateKeyLabelRsa, sizeof(privateKeyLabelRsa) - 1},
    { CKA_ID, &keyPairIdRsa, sizeof(keyPairIdRsa) - 1}, //    (   ""  "" )
    { CKA_KEY_TYPE, &keyTypeRsa, sizeof(keyTypeRsa)}, //   - RSA
    { CKA_DECRYPT, &attributeTrue, sizeof(attributeTrue)}, //
    { CKA_TOKEN, &attributeTrue, sizeof(attributeTrue)}, //
    { CKA_PRIVATE, &attributeTrue, sizeof(attributeTrue)}, //
    { CKA_MODULUS, &rsaModulus, sizeof(rsaModulus)}, //  n
    { CKA_PUBLIC_EXPONENT, &pubExp, sizeof(pubExp)}, //   e
    { CKA_PRIVATE_EXPONENT, &privExp, sizeof(privExp)}, //   d
    { CKA_PRIME_1, &prime1, sizeof(prime1)}, //   p
    { CKA_PRIME_2, &prime2, sizeof(prime2)}, //   q
    { CKA_EXPONENT_1, &exp1, sizeof(exp1)}, //   d   p-1
    { CKA_EXPONENT_2, &exp2, sizeof(exp2)}, //   d   q-1
    { CKA_COEFFICIENT, &coeff, sizeof(coeff)}, // q^(-1) mod p
}
```

Шаблон импорта "открытого" ключа для шифрования данных

```
CK_ATTRIBUTE publicKeyTemplate[] =
{
    { CKA_CLASS, &publicKeyObject, sizeof(publicKeyObject)}, //  -
    { CKA_LABEL, &privateKeyLabelRsa, sizeof(privateKeyLabelRsa) - 1},
    { CKA_ID, &keyPairIdRsa, sizeof(keyPairIdRsa) - 1}, //    (   ""  "" )
    { CKA_KEY_TYPE, &keyTypeRsa, sizeof(keyTypeRsa)}, //   - RSA
    { CKA_ENCRYPT, &attributeTrue, sizeof(attributeTrue)}, //
    { CKA_TOKEN, &attributeTrue, sizeof(attributeTrue)}, //
    { CKA_PRIVATE, &attributeFalse, sizeof(attributeTrue)}, //
    { CKA_MODULUS, &rsaModulus, sizeof(rsaModulus)}, //  n
    { CKA_MODULUS_BITS, &modulusBits, sizeof(modulusBits) }, //   n
        { CKA_PUBLIC_EXPONENT, pubExp, sizeof(pubExp) } }); //
```

Шаблон импорта "закрытого" ключа для ЭП

```
CK_ATTRIBUTE privateKeyTemplate[] =
{
    { CKA_CLASS, &privateKeyObject, sizeof(privateKeyObject)}, //  -
    { CKA_LABEL, &privateKeyLabelRsa, sizeof(privateKeyLabelRsa) - 1},
    { CKA_ID, &keyPairIdRsa, sizeof(keyPairIdRsa) - 1}, //    (   ""  "" )
    { CKA_KEY_TYPE, &keyTypeRsa, sizeof(keyTypeRsa)}, //   - RSA
    { CKA_DECRYPT, &attributeFalse, sizeof(attributeFalse)}, //
    { CKA_TOKEN, &attributeTrue, sizeof(attributeTrue)}, //
    { CKA_PRIVATE, &attributeTrue, sizeof(attributeTrue)}, //
    { CKA_MODULUS, &rsaModulus, sizeof(rsaModulus)}, //  n
    { CKA_PUBLIC_EXPONENT, &pubExp, sizeof(pubExp)}, //   e
    { CKA_PRIVATE_EXPONENT, &privExp, sizeof(privExp)}, //   d
    { CKA_PRIME_1, &prime1, sizeof(prime1)}, //   p
    { CKA_PRIME_2, &prime2, sizeof(prime2)}, //   q
    { CKA_EXPONENT_1, &exp1, sizeof(exp1)}, //   d   p-1
    { CKA_EXPONENT_2, &exp2, sizeof(exp2)}, //   d   q-1
    { CKA_COEFFICIENT, &coeff, sizeof(coeff)}, // q^(-1) mod p
}
```

Шаблон импорта "открытого" ключа для ЭП

```
CK_ATTRIBUTE publicKeyTemplate[] =
{
    { CKA_CLASS, &publicKeyObject, sizeof(publicKeyObject)}, //  -
    { CKA_LABEL, &privateKeyLabelRsa, sizeof(privateKeyLabelRsa) - 1},
    { CKA_ID, &keyPairIdRsa, sizeof(keyPairIdRsa) - 1}, //    (   ""  "" )
    { CKA_KEY_TYPE, &keyTypeRsa, sizeof(keyTypeRsa)}, //   - RSA
    { CKA_ENCRYPT, &attributeFalse, sizeof(attributeFalse)}, //
    { CKA_TOKEN, &attributeTrue, sizeof(attributeTrue)}, //
    { CKA_PRIVATE, &attributeFalse, sizeof(attributeTrue)}, //
    { CKA_MODULUS, &rsaModulus, sizeof(rsaModulus)}, //  n
    { CKA_MODULUS_BITS, &modulusBits, sizeof(modulusBits) }, //   n
        { CKA_PUBLIC_EXPONENT, pubExp, sizeof(pubExp) } }); //
```

Шаблон импорта сертификата

```
CK_ATTRIBUTE certificateTemplate[] =
{
        { CKA_VALUE, 0, 0 },
        { CKA_CLASS, &certificateObject, sizeof(certificateObject)}, //  -
        { CKA_ID, &keyPairIdRsa, sizeof(keyPairIdRsa) - 1},  //
        { CKA_TOKEN, &attributeTrue, sizeof(attributeTrue)}, //
        { CKA_PRIVATE, &attributeFalse, sizeof(attributeFalse)}, //
        { CKA_CERTIFICATE_TYPE, &certificateType, sizeof(certificateType)}, //
        { CKA_CERTIFICATE_CATEGORY, &tokenUserCertificate, sizeof(tokenUserCertificate)}, //    ,

}
```