

Lisa 2. ID-kaardi tarkvara analüüs 02.12.2011

Toni Koivunen, Sauli Pahlman

Riigi Infosüsteemi Amet

reg kp ..20..12..2011.....

reg nr ..1-6/11-6-5-2.....

Background

Estonian eID (EstEID) card is a primary personal identification document. The EstEID-card can be used for issuing digital signatures as well as electronically encrypting documents for only certain EstEID card to read.

1 A vulnerability to adaptive chosen-ciphertext attack

The attack has been reported by Joe-Kai Tsay, Yusuke Kawamoto and Graham Steel to affect EstEID card. Vulnerability to the described attack would allow an attacker with access to a content which is encrypted for an EstEID user, to expose the corresponding plaintext content in a reasonable amount of time, perhaps a couple of days. A succesful attack has several prerequisites:

1.1 The attack takes place by the attacker repeatedly altering the encrypted content, thus the attacker needs to have access to a document or a file which which has been encrypted in a purpose to make only the attack target able to read it.

1.2 The attacker needs access to a decryption oracle which tells him whether the decryption of the altered encrypted document was succesful or not. The number of decryption repetitions to gain access to the corresponding plaintext is at least 20 000. Decrypting data which has been encrypted using the target's public key, requires the corresponding private key. This means that the attacker will need access to the target's EstEID card and the PIN1 code, or a computer where the target's EstEID card is attached with PIN1 given. NB. PIN1 needs to be given only once during the at least 20 000 round decryption session since card will keep the private key unlocked until powered off.

A succesful attack will expose the encrypted message used in phases 1 and 2 below. The secret RSA key which resides on the EstEID chip, is not exposed in the attack.

1.3 Verification of the existence of the vulnerability

The EstEID 3.0 card is a Java based smart card solution. It makes use of APIs provided JavaCard and relies on the security of chosen algorithms and security of the JavaCard API implementation.

Based on [1] and [2], it seems that for an RSA ciphertext to be compatible with the EstEID version 1.0 card, it must be formatted according to PKCS#1 version 1.5. There appears to be no specification for the EstEID 3.0 card. However, based on a source code audit, also EstEID 3.0 is using PKCS#1 version 1.5 as an RSA container. Source code was analysed as follows:

In the APDU processing method specified by JavaCard API we have a case where APDU INS 0x2A (Perform security operation) is given. Once the APDU buffer GB has been unpacked, we are calling compute_digital_signature on it:

```
public void process(APDU apdu) {  
    case (byte)0x2A:    //L_La = perform_security_operation (GB, L_Lc );//- perform
```

security operation (dig sig, hash, decrypt)

```
apdu.setIncomingAndReceive();
    if ( (GB[0] & (byte) 0x0C) != (byte) 0 )
    {
        L_CLA = GB[Offs_CLA];
        unpack_secure_message( GB );
        L _ L a =
compute_digital_signature (GB); //- set security environment
        break;
    }
    apdu.setOutgoingAndSend( (short) 0,
compute_digital_signature (GB) );
    return;
```

In compute_digital_signature (this is used for decryption also) we have a case for requested decryption of APDU data buffer (which contains the symmetric crypto session key):

```
switch ( Util.makeShort( GB[Offs_P1], GB[Offs_P2]) )
{
    case (short)0x8086: //Decrypt
```

...and if PIN1 has been verified (PIN1 validity is cached in a transient/volatile array, thus PIN1 will be "forgotten" only when the card is removed from the reader) we are calling init() (key definition) and doFinal() (data decryption) on the data passed in GB[6:] and storing it in scratch. After that key usage counter is decremented and result stored in GB again.

```
    if ( ! G_bool[bo_PIN1] )
    {
        //if ( G_byte[by_Command_With_Sec_Mess_Key_No] != (byte) 4 )
        ISOException.throwIt(ISO7816.SW_CONDITIONS_NOT_SATISFIED);
        //if ( G_byte[by_SM_TYPE] != SM_PRO_SEC )
        ISOException.throwIt(ISO7816.SW_CONDITIONS_NOT_SATISFIED);
        if ( GB[GB_Offs_Command_With_Sec_Mess_Key_No] != (byte) 4 )
        ISOException.throwIt(ISO7816.SW_CONDITIONS_NOT_SATISFIED);
        if ( GB[GB_Offs_SM_TYPE] != SM_PRO_SEC )
        ISOException.throwIt(ISO7816.SW_CONDITIONS_NOT_SATISFIED);
    }
    if ( CurrKP_1.getPrivate().getType() != KeyBuilder.TYPE_RSA_CRT_PRIVATE )
        ISOException.throwIt(ISO7816.SW_FUNC_NOT_SUPPORTED);
    L_Lc--;
    if ( L_Lc != (short) ( CurrKP_1.getPrivate().getSize() / (short)8 ) )
        ISOException.throwIt( (short) 0x6700 );
    EE_RSA_Cipher.init( CurrKP_1.getPrivate(), Cipher.MODE_DECRYPT);
    L_La = EE_RSA_Cipher.doFinal( GB, (short)6, L_Lc, G_Scratch, (short) 0);
    Decrement_Key_Usage_Counter ( (byte) 1 , Decr_Key_Ref, (short)0, GB );
    Util.arrayCopyNonAtomic( G_Scratch, (short) 0, GB, (short)0, L_La );
    return L_La;
```

As the EE_RSA_Cipher used here is defined as:

```
EE_RSA_Cipher = Cipher.getInstance(Cipher.ALG_RSA_PKCS1, false); //This is a PKCS1v1.5  
padding cipher
```

the card is are vulnerable to a padding oracle attack as described by Tsay et al. In case the decrypted data is not bounded by appropriate padding bytes, an uncaught `javacard.security.CryptoException` is thrown with reason code `CryptoException. ILLEGAL_USE`. This is our oracle call and what makes modular exponentiation of the private key possible, giving the attacker possible access to the session key and the applet vulnerable to adaptive known ciphertext attack.

Based on the above, EstEID 1.0 and 3.0 both are designed to use the vulnerable PKCS#1 version 1.5. Thus this issue should not be considered as a programming error but a possibly risky design decision.

1.4. Eliminating the vulnerability

As the vulnerability exists on the PKCS#1 version 1.5 scheme, the vulnerability cannot be eliminated with a source code fix without moving to use another cryptosystem or removing the encryption/decryption capability from the EstEID. Moving to another cryptosystem, e.g., to PKCS#1 version 2.0 (RSA OAEP) is cryptographically very simple and the JavaCard API has an existing implementation for it. However, the infrastructural transition would require significant effort and would most probably mean losing backwards compatibility to the current versions of the EstEID card and to require some changes to the existing applications based on EstEID.

Also, it's recommended to update the card to require PIN1 in all cases where decryption is performed, or on every Nth decryption. Requiring the PIN1 only once per session is too permissive.

[1] http://support.sk.ee/eng/index.php?_m=knowledgebase&_a=viewarticle&kbarticleid=1401

[2] http://www.id.ee/public/EstEID_kaardi_kasutusjuhend.pdf