SPRÁVA ZÁKLADNÍCH REGISTRŮ

Popis eGON služeb

Šifrování fotografií a podpisů pro média služby pro AIS EOP a AIS ECD

Název dokumentu:	Popis eGON služeb	**Verze:**	1.00
Autor:	Správa základních registrů	**Datum aktualizace:**	05. 09. 2017
Účel:	Popis šifrování fotografií a podpisů pro média služby pro AIS EOP a AIS ECD	**Počet stran:**	6

Obsah

1 Účel dokumentu

Účelem tohoto dokumentu je především poskytnout orgánům veřejné moci, obecně uživatelům Základních registrů (typicky IT), jednoduchý a srozumitelný popis jakým způsobem dešifrovat digitální objekty (fotografie a podpisy), které jsou výstupem příslušných eGON služeb ISZR.

Změny provádí SZR.

1.1 Dotčené služby

Šifrování digitálních objektů je použito v eGON službách:

- *E197 – agendaMediaDataCtiAifo*
- *E198 – agendaMediaDataCtiPodleUdaju*

2 Certifikáty a šifrování

Uživatel kompozitních média služeb musí mít pro používání těchto služeb připraven certifikát, se kterým bude AIS EOP nebo AIS ECD šifrovat digitální objekty (fotografie a podpisy) a uživatel je bude dešifrovat.

2.1 Parametry certifikátu

Šifrování je prováděno v AIS vydávajícím digitální objekty na základě veřejné části certifikátu předaného žadatelem o eGON službu na vstupu eGON služby.

Pro použitý certifikát jsou doporučeny a podporovány následující parametry:

- Key Algorithm: RSA, délka 2048
- Signature Algorithm: SHA256withRSA
- Doba platnosti: podle potřeb OVM
- Ostatní údaje vyplnit tak, aby bylo zřejmé, které OVM certifikát používá.

2.2 Šifrování

Pro šifrování se používá **Bouncy Castle API.** V případě média služeb to jsou knihovny od http://www.bouncycastle.org. Na šifrování se použije algoritmus AES 128 CBC.

3 Použití

Uživatel v rámci volání média služeb zasílá veřejnou část certifikátu, kterým mají být fotografie a podpisy šifrovány. Veřejná část certifikátu musí být vyexportována ve formě DER nebo PEM. Média služby jsou testovány na tyto dvě formy veřejné části certifikátu. Uživatel veřejnou část certifikátu zadává do volání média služby v binární podobě.

Privátní klíč certifikátu si uživatel může uložit v libovolné formě.

Dále jsou uvedeny příklady zdrojových kódů pro dešifrování fotografií a podpisů v Java a .NET. Principy z příkladů je možné použít ve vlastním řešení. Při použití tohoto řešení je vhodné vyexportovat privátní klíč ve formátu PKCS #12.

3.1 Dešifrování Java

```
package DecryptExample;

import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.security.KeyStore;
import java.security.PrivateKey;
import java.util.Collection;
import java.util.Enumeration;
import java.util.Iterator;
import org.bouncycastle.cms.CMSEnvelopedDataParser;
import org.bouncycastle.cms.CMSException;
import org.bouncycastle.cms.Recipient;
import org.bouncycastle.cms.RecipientInformation;
import org.bouncycastle.cms.jcajce.JceKeyTransEnvelopedRecipient;

/**
 * Modul pro dešifrování fotografií a podpisů pro kompozitní média služby AIS EOP a AIS ECD
 * Modul závisí na knihovně bcpkix-jdk15on-1.57.jar a bcprov-jdk15on-1.57.jar
 *
 * @author Libor Málek, KOMIX s.r.o
 * @version 1.0, 13.06.2017
 */
public class MediaDeCrypter {

   /**
    * Dešifrování fotografií a podpisů pro kompozitní média služby AIS EOP a AIS ECD
    * @param encryptedData Zašifrovaná data
    * @param file Soubor s privátním klíčem
    * @param password Heslo k souboru s privátním klíčem
    * @return dešifrováná data
    *
    * @throws IOException
    * @throws CMSException
    */
   public byte[] decrypt(
     byte[] encryptedData,
     String file,
     String password
     ) throws IOException, CMSException, Exception {

        CMSEnvelopedDataParser parser = new CMSEnvelopedDataParser(encryptedData);
        RecipientInformation recInfo = getSingleRecipient(parser);
        Recipient recipient = new JceKeyTransEnvelopedRecipient(getPrivateKey(new File(file), password));
        byte[] o_DecryptedData = recInfo.getContent(recipient);

        return o_DecryptedData;
    }

   /**
    * Dešifrování fotografií a podpisů pro kompozitní média služby AIS EOP a AIS ECD
    * @param encryptedData Zašifrovaná data
    * @param file Soubor s privátním klíčem
    * @param password Heslo k souboru s privátním klíčem
    * @return dešifrováná data
    *
    * @throws IOException
    * @throws CMSException
    */
   public byte[] decrypt(
      byte[] encryptedData,
      File file,
      String password) throws IOException, CMSException, Exception {
```

```
        CMSEnvelopedDataParser parser = new CMSEnvelopedDataParser(encryptedData);
        RecipientInformation recInfo = getSingleRecipient(parser);
        Recipient recipient = new JceKeyTransEnvelopedRecipient(getPrivateKey(file, password));
        byte[] o_DecryptedData = recInfo.getContent(recipient);

        return o_DecryptedData;
    }

    private  RecipientInformation getSingleRecipient(CMSEnvelopedDataParser parser) {
        Collection recInfos = parser.getRecipientInfos().getRecipients();
        Iterator recipientIterator = recInfos.iterator();
        if (!recipientIterator.hasNext()) {
            throw new RuntimeException("Could not find recipient");
        }
        return (RecipientInformation) recipientIterator.next();
    }

    /**
     * Práce s privátním klíčem
     * @param file Soubor s privátním klíčem
     * @param password Heslo k souboru s privátním klíčem
     * @return Privátní klíč pro další použití
     *
     * @throws Exception
     */
    private PrivateKey getPrivateKey(File file, String password) throws Exception {
        KeyStore ks = KeyStore.getInstance("PKCS12");
        try (FileInputStream fis = new FileInputStream(file)) {
            ks.load(fis, password.toCharArray());
        }

        Enumeration<String> aliases = ks.aliases();
        String alias = aliases.nextElement();
        return (PrivateKey) ks.getKey(alias, password.toCharArray());
    }
}
```

3.2 Dešifrování .NET

```
using System;
using System.IO;
using System.Security;
using System.Security.Cryptography.Pkcs;
using System.Security.Cryptography.X509Certificates;

namespace DecryptExample
{
  /// <summary>
  /// Priklad desifrovani zpravy
  /// </summary>
  public static class DecryptExample
  {
    /// <summary>
    /// Priklad desifrovani zpravy - Main.
    /// </summary>
    public static void Main()
    {
      try
      {
        byte[] decryptedMessage = EnvelopedCmsX509Decrypt(@"fotoEncrypted", @"test.p12", "3155");
        File.WriteAllBytes(@"fotoDecrypted.jpg", decryptedMessage);
      }
      catch (Exception e)
      {
          // Logging and error handling
          Console.WriteLine(e);
      }
    }
```

```
    /// <summary>
    /// Desifrovani EnvelopedCms zpravy
    /// </summary>
    /// <param name="encryptedDataFile">cesta k zakodovane zprave</param>
    /// <param name="certificateFile">cesta k souboru s certifikatem prijemce zpravy</param>
    /// <param name="password">heslo k certifikatu prijemce zpravy ulozenemu v PFX/P12</param>
    /// <returns>byte[] - dekodovana zprava</returns>
    private static byte[] EnvelopedCmsX509Decrypt(
      string encryptedDataFile,
      string certificateFile,
      string password)
    {
      // Overeni - zakodovany soubor existuje
      if (string.IsNullOrWhiteSpace(encryptedDataFile))
        throw new ArgumentNullException(nameof(encryptedDataFile));
      if (!File.Exists(encryptedDataFile))
        throw new FileNotFoundException(encryptedDataFile);

      // Overeni - certifikat s PrivateKey existuje
      if (string.IsNullOrWhiteSpace(certificateFile))
        throw new ArgumentNullException(nameof(certificateFile));
      if (!File.Exists(certificateFile))
        throw new FileNotFoundException(certificateFile);

      // Je zadane heslo
      if (string.IsNullOrWhiteSpace(password))
        throw new ArgumentNullException(nameof(password));

      // Dotahneme data
      byte[] data = File.ReadAllBytes(encryptedDataFile);
      using (SecureString ss = new SecureString())
      {
        // Zkonvertujeme heslo do SecureString formatu
        foreach (char keyChar in password) ss.AppendChar(keyChar);

        // Nacteme certifikat z disku
        X509Certificate2 cert = new X509Certificate2(certificateFile, ss);

        // Pripravime desifrovani
        EnvelopedCms envelopedCms = new EnvelopedCms();

        // Dekodujeme zpravu
        envelopedCms.Decode(data);

        // Desifrujeme zpravu pro prijemce
        X509Certificate2Collection collection = new X509Certificate2Collection();
        collection.Add(cert);

        // Vratime dekodovany obsah zpravy
        envelopedCms.Decrypt(collection);
        return envelopedCms.Encode();
      }
    }
  }
}
```

4 Odkazy na další dokumenty

4.1 Egon služby

SZR_popis_eGON_služeb_E197_agendaMediaDataCtiAifo.docx

SZR_popis_eGON_služeb_E198_agendaMediaDataCtiPodleUdaju.docx