A házifeladatban alkalmazandó XML struktúra

Absztrakt:

A feladat egy fájl, vagy szövegkódoló készítése. Parancssorból indítható (a helyes szintaxis megadása mellett (http://www.linfo.org/standard_input.html)), és/vagy GUI-val rendelkezik, és a lent megadott követelmények szerint készít egy enkriptált fájlt.

Referencia
http://www.w3.org/TR/xmlenc-core/

File név konvenció
Az XML Encryption szabvány tetszőleges adat (byte-sorozat) titkosítására alkalmas, de nem biztosít eszközöket a file kezeléshez. Az alkalmazások együttműködése érdekében megállapodunk az alábbi file-név konvencióban:

- titkosítandó file neve tetszőleges (pl. abc.def)
- rejtjelezett file neve a fenti minta alapján: abc_def.enc

Dekódoláskor az alkalmazásnak az eredeti néven kell mentenie a nyílt állományt.

Mivel több felhasználótól is jöhet üzenet a különböző felhasználóktől jövő üzenetek külön almappákba kerüljenek, a felhasználók neveivel megjelölve.

Ha két azonos nevű fájl érkezik egy mappába, azt abc_1.def és abc_2.def néven kell elnevezni.

Padding
A titkosítandó adat nem biztos, hogy a blokkméret (16 byte) egész számú többszöröse. Padding-nek nevezzük az adat szükség szerinti kiegészítését. Ezt az OpenSSL kezeli, az XML-ben nem jelenik meg erre vonatkozó információ.

IV (Initial value)
Az alkalmazott algoritmus (128 bites AES, CFB módban) igényel egy véletlen, egyedi, 16 byte-os kezdeti értéket, amire dekódoláskor szükség van. Az IV-t nem kell titkosítani, a szokásos megoldásnak megfelelően a rejtjelezett byte-folyam elé fűzzük az XML-ben.

Base64 kódolás
Mivel az XML-ben nem minden byte engedélyezett, a titkosított adat (IV is) és kulcs Base64 kódolt formában kerül az XML-be.
Bővebben: http://hu.wikipedia.org/wiki/Base64

Megjegyzés: Ügyeljünk azza, hogy bizonyos Base64 enkóderek hajlamosak minden 72. karakter után egy sortörést (\n) rakni, ami nem tartozik az elkódolt adathoz. HA ezek a sortörések bennmaradnak a másik oldalon probléma lehet a dekódolással.

EncryptedType elemtípus
sémadefiníció:

```
<complexType name='EncryptedType' abstract='true'>
  <sequence>
    <element name='EncryptionMethod' type='xenc:EncryptionMethodType'
          minOccurs='0'/>
    <element ref='ds:KeyInfo' minOccurs='0'/>
    <element ref='xenc:CipherData'/>
    <element ref='xenc:EncryptionProperties' minOccurs='0'/>
  </sequence>
  <attribute name='Id' type='ID' use='optional'/>
  <attribute name='Type' type='anyURI' use='optional'/>
```

```
    <attribute name='MimeType' type='string' use='optional'/>
    <attribute name='Encoding' type='anyURI' use='optional'/>
   </complexType>
```

EncryptionMethod is an optional element that describes the encryption algorithm applied to the cipher data. If the element is absent, the encryption algorithm must be known by the recipient or the decryption will fail.

Az EncryptionMethod elemet nem használjuk. Az algoritmust és a paramétereit ismertnek tekintjük.

ds:KeyInfo is an optional element, defined by [XML-DSIG], that carries information about the key used to encrypt the data. Subsequent sections of this specification define new elements that may appear as children of ds:KeyInfo.

A ds:KeyInfo elemet használjuk. Ez határozza meg, hogy mi az a kulcs, amit a dekódoláshoz használni kell. Részletek lejjebb.

CipherData is a mandatory element that provides the encrypted data. It must either contain the encrypted octet sequence as base64 encoded text of the CipherValue element, or provide a reference to an external location containing the encrypted octet sequence via the CipherReference element.

```
   <element name='CipherData' type='xenc:CipherDataType'/>
  <complexType name='CipherDataType'>
    <choice>
      <element name='CipherValue' type='base64Binary'/>
      <element ref='xenc:CipherReference'/>
    </choice>
  </complexType>
```

A CipherData elem CiperValue elemét használjuk.

EncryptionProperties can contain additional information concerning the generation of the EncryptedType (e.g., date/time stamp).

Az EncryptionProperties elemet nem használjuk.

Id is an optional attribute providing for the standard method of assigning a string id to the element within the document context.

Az Id, Type, MimeType, Encoding attribútumokat nem használjuk.

Az EcryptedType absztrakt típus, amiből az EncryptedData és az EncryptedKey elem származtatható.

The EncryptedData element is the core element in the syntax. Not only does its CipherData child contain the encrypted data, but it's also the element that replaces the encrypted element, or serves as the new document root.

```
  <element name='EncryptedData' type='xenc:EncryptedDataType'/>
  <complexType name='EncryptedDataType'>
    <complexContent>
      <extension base='xenc:EncryptedType'>
      </extension>
    </complexContent>
  </complexType>
```

A titkosított XML file root eleme az EncryptedData lesz.

Az EncryptedKey elem elhelyezésére, valamint az abból dekódolható kulcs és az általa rejtjelezett adat összekapcsolására a szabvány több lehetőséget kínál. Mi az alábbi struktúrát használjuk:

```
<EncryptedData ...>
   <KeyInfo>
      <EncryptedKey ... Recipient="a">...</EncryptedKey>
      <EncryptedKey ... Recipient="b">...</EncryptedKey>
      <EncryptedKey ... Recipient="c">...</EncryptedKey>
   </KeyInfo>
   ....
</EncryptedData>
```

A szabvány lehetővé teszi, hogy az EncryptedKey a titkosított adatot tartalmazó EncryptedData KeyInfo elmének gyermek eleme legyen. Mi ezt a megoldást alkalmazzuk olyan módon, hogy a több címzett részére titkosított szimmetrikus kulcs egy-egy EncryptedKey elemben található. Erre utal a Recipient attribútum.

The EncryptedKey element is used to transport encryption keys from the originator to a known recipient(s). It may be used as a stand-alone XML document, be placed within an application document, or appear inside an EncryptedData element as a child of a ds:KeyInfo element. The key value is always encrypted to the recipient(s).

```
<element name='EncryptedKey' type='xenc:EncryptedKeyType'/>
<complexType name='EncryptedKeyType'>
 <complexContent>
  <extension base='xenc:EncryptedType'>
   <sequence>
    <element ref='xenc:ReferenceList' minOccurs='0'/>
    <element name='CarriedKeyName' type='string' minOccurs='0'/>
   </sequence>
   <attribute name='Recipient' type='string' use='optional'/>
  </extension>
 </complexContent>
</complexType>
```

A ReferenceList és CarriedKeyName elemeket nem használjuk. A Recipient attribútum a szimmetrikus kulcs rejtjelezéséhez használt publikus kulcs magánkulcs párját birtokló csapat neve. A publikus kulcsot tartalmazó file neve egyezzen meg a csapat nevével és legyen *.pem típusú. Ezt a file-t minden csapatnak fel kell helyeznie a tárgy honlapjára (az OpenSSL beszámoló idejére).

A KeyInfo elemet az XML-DSIG szabvány specifikálja (http://www.w3.org/TR/2001/PR-xmldsig-core-20010820/):

```
<element name="KeyInfo" type="ds:KeyInfoType"/>
<complexType name="KeyInfoType" mixed="true">
 <choice maxOccurs="unbounded">
  <element ref="ds:KeyName"/>
  <element ref="ds:KeyValue"/>
  <element ref="ds:RetrievalMethod"/>
  <element ref="ds:X509Data"/>
  <element ref="ds:PGPData"/>
  <element ref="ds:SPKIData"/>
  <element ref="ds:MgmtData"/>
  <any processContents="lax" namespace="##other"/>
  <!-- (1,1) elements from (0,unbounded) namespaces -->
 </choice>
```

```
    <attribute name="Id" type="ID" use="optional"/>
  </complexType>
```

Ezt a struktúrát egészíti ki az XML Encryption szabvány az EncryptedKey elemmel. Az Id attribútumot nem használjuk, a választható elemek közül pedig a KeyName elemben a szimmetrikus kulcsot titkosító kulcspárra való hivatkozást helyezünk el csapatnév.pem file-név formájában.

Ellenőrzés módja:
Az elkészítendő szoftver a parancssorban vagy máshogy megadott *.pem file-nevekkel (nem kell létezniük a file-oknak), fiktív (lehet fix is) rejtjelezett adatokkal létrehoz egy *.enc kiterjesztésű file-t. Megfordítva: egy valid *.enc file-ból a képernyőre kiírja a *.pem file-neveket és a rejtjelezett adatokat.

A házifeladatban alkalmazandó OpenSSL parancsok

A csapatok kötelesek időben egyeztetni egymással a parancsok pontos paraméterezését illetően. Kérdéseket a gyakorlatvezetőknek fel lehet tenni.

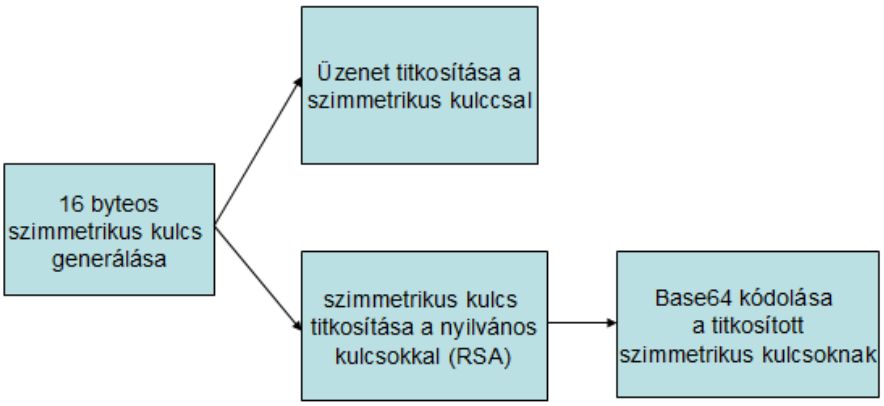
Tanúsítvány elkészítése

2048 bites RSA aszimmetrikus kulcspár és tanúsítvány generálása self-signed módon, de egyébként tetszőlegesen:

```
req -x509 -nodes -newkey
```

A file kiterjesztése: .pem, a file neve a csapatra utaljon.

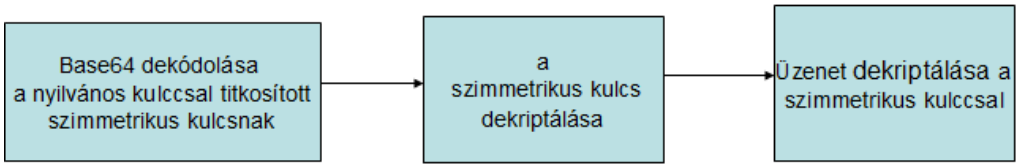
Kódolás



Az egyes lépésekben használt openssl függvények:

1. 16 byte-os szimmetrikus kulcs generálása: rand
2. Üzenet titkosítása szimmetrikus kulccsal: enc -aes-128-cfb
   Kezdeti érték (IV) alkalmazása kötelező, meghatározása tetszőleges módon történik. Az IV nyílt formában kerül befűzésre a rejtjelezett byte-folyam elé az XML struktúránál megadott módon.
3. Szimmetrikus kulcs titkosítása a publikus kulcsokkal: rsautl
4. Base64 kódolása a publikus kulcsokkal titkosított szimmetrikus kulcsnak: enc -base64

Dekódolás



Az egyes lépésekben használt openssl függvények:

- Base64 kódolt, publikus kulccsal titkosított szimmetrikus kulcs Base64 dekódolása: enc -base64
- Szimmetrikus kulcs megfejtése a privát kulccsal: rsautl -decrypt
- Az eredeti üzenet megfejtése: enc -aes-128-cfb

Összefoglalva

1. A feladat egy olyan fájl, vagy szövegkódoló készítése, ami vagy parancssorból indítható (a helyes szintaxis megadása mellett (http://www.linfo.org/standard_input.html)!!), és/vagy GUI-val rendelkezik, és a fent megadott követelmények szerint készít egy enkriptált fájlt.

2. A GUI pluszpont, de csak ha igényes, fájlkódolás esetén további pluszpont, ha az oprendszer fájlkezelőjébe van integrálva (context menu).

3. A fájl átküldése a csapaton belül történhet:

- Másolással (pendrive, e-mail)

- TCP kapcsolaton keresztül (pluszpontért)

4. Az elkódolt fájlt azonban bármely másik csapatnak, vissza kell tudnia fejteni. Egyéb esetben a teljes feladatra csak fél pontszám adható plusz pontok nélkül.