Budapesti Műszaki és Gazdaságtudományi Egyetem

Villamosmérnöki és Informatikai Kar

Elektronikus archiválórendszer fejlesztése PKI alapokon

Szerző:

Kollár Balázs

Konzulensek:

Szigeti Szabolcs, Krasznay Csaba

2005. november 11.

Tartalom

1. Bevezető

A közelmúltban a Magyar Elektronikus Aláírási Szövetség (MELASZ) elkészítette a digitális aláírások formátumára vonatkozó közös ajánlását, mely nemzetközi szabványok [3],[4],[10] alapján ad útmutatást a hazai fejlesztőknek. Ez az ajánlás mérföldkőnek tekinthető az elektronikus iratkezelés hazai elterjedésének szempontjából, amely számos előnye ellenére még várat magára, habár a jogi szabályozás már 2001 óta lehetővé teszi a váltást. Az ajánlás megteremti az alapját az elektronikus iratkezelésre épülő termékek és szolgáltatások hazai piacának. Ugyanis az intézmények iratkezelésük megszervezésekor szabványos felületen kapcsolódó alkalmazások közül válogathatnak, melyek egymást kiegészítve komplex iratkezelő rendszerek kialakítását teszik lehetővé.

Kutatásaim célja kezdetben a PKI rendszerek és a digitális aláírás nemzetközi szabványainak beható tanulmányozása, ezt követően nemzeti megvalósítások, valamint a hazai jogi szabályozás megismerése volt. Később megvizsgáltam a gyakorlati felhasználás lehetőségeit, tájékozódtam a már megvalósult alkalmazásokról.

Vizsgálódásaim eredményeként célul tűztem ki egy saját alkalmazás elkészítését, amely képes a MELASZ ajánlásának megfelelő dokumentumok kezelésére, és amelyre később összetett szolgáltatásokat lehet építeni, különös tekintettel az elektronikus archiválásra. Követtem a korszerű szoftver-technológiai irányvonalakat, így az alkalmazás Web-alapú, több-felhasználós, többrétegű, relációs adatbázisra és XML állományokra épül, hardware platform független és objektum-orientált.

A szoftvert hat hónapja fejlesztem, mely idő alatt kiforrtak a rendszer alacsony szintű funkciói, valamint több magas szintű szolgáltatása is rendelkezésre áll. Ezek a tervek szerint folyamatokban fognak egyesülni.

2. A digitális aláírás matematikai háttere

A digitális aláírás gyakorlati használhatóságához elengedhetetlen, hogy megbízhatóságát tudományos tényekkel lehessen alátámasztani. Mivel a módszer alapja matematikai kutatások eredménye, ez könnyen megtehető. Szeretném röviden bemutatni az aláírás ma elterjedt módszerének elemeit, nevezetesen a lenyomatképző függvényeket, és az aszimmetrikus kódolást. E kettő lépésből áll az ún. „Hash and Sign" módszer, amelyre a mai implementációk épülnek, és amelyet az elfogadott [2] MELASZ szabvány is javasol.

2.1. Lenyomatképző függvények

A lenyomatképző függvények nevüket az ujjlenyomat fogalma után kapták. Ha találunk egy ujjlenyomatot, valószínűsíthetjük, hogy az pontosan egy emberhez tartozik, mivel két embernek nincs hasonló ujjlenyomata, de legalábbis valószínűtlen hogy ilyen párost találnánk. Továbbá ha van tippünk, hogy kié lehet az ujjlenyomat, azt gyorsan le is ellenőrizhetjük egy nyilvántartás vagy az illető segítségével.

A lenyomatképző függvények ugyanerre képesek. A lenyomat esetükben valamiféle szám vagy számsor, a lenyomat tulajdonosa pedig szintén egy számsor, ami a lenyomatnál sokkal hosszabb is lehet. A digitális világban sok mindent leírhatunk számsorokkal, így lenyomatot képezhetünk bármilyen számokkal ábrázolt adathoz, így tipikusan képhez, hanghoz, szerkesztett szöveghez, tömörített fájlhoz. Az ujjlenyomathoz hasonlóan egy erősnek tartott lenyomatképző függvény esetén sem fordulhat elő a gyakorlatban, hogy két különböző számsorhoz ugyanaz a lenyomat tartozzon, sőt, a bemenő számsor egy pici változása is nagymértékű változást eredményez a lenyomatban.

2.2. Aszimmetrikus kriptográfia

Az írott információ elrejtésére már igen régen felmerült az igény. Bizonyítékok már hétezer évvel korábbról, az Óegyiptomi Birodalom idejéből is származnak erre vonatkozóan. Az első komolynak tekinthető, ismert rejtjelező módszer azonban négy évezreddel későbbről származik, és a héber ábécéhez készült. Elve az ábécé visszájára fordítása.

Az alapvető feladat - katonai hasonlattal élve - tulajdonképpen az, hogy úgy írjuk le a mondandónkat a szövetségeseinknek, hogy az ellenség ne tudja azt elolvasni, ha véletlenül hozzá érkezne meg az üzenet. Ennek érdekében az üzenetet mi rejtjelezzük, a

szövetségesünk pedig dekódolja. Ennek sikeréhez mindkét fél tud egy-egy olyan titkot, amit az ellenség reményük szerint nem tud.

A különbség a szimmetrikus és az aszimmetrikus kriptográfia között abban áll, hogy ez a két titok egyforma, vagy különböző (de természetesen valamilyen módon még mindig összetartozó). Előbbi esetben mindkét fél ugyanazzal a kulccsal végzi a rejtjelezést és a feloldást is, utóbbi esetben viszont mindkét félnek két-két kulcsa van (kulcs pár), az egyik nyilvános, a másikat titokban kell tartania. Egy A fél úgy üzenhet rejtjelezve a másik B félnek, hogy B nyilvános kulcsával rejtjelezve küldi el az üzenetet. B ezt a titkos kulcsával dekódolja, és olvassa is. Az aszimmetria abban áll, hogy a nyilvános és a titkos kulcs különböző, és egyikből a másikat támadó nem tudja rekonstruálni.

Tehát, mindkét fél tud rejtjelezve üzenni a másiknak anélkül, hogy közös titkon osztoznának, amit előre egyeztetniük kellene. A nyilvános kulcsokat valamilyen módon ki kell cserélniük egymás között. A cserét nem szükséges titkosítva lebonyolítani, de a kulcs pár és az állítólagos tulajdonos összetartozóságáról meg kell győződni. Hiszen, ha egy támadó a saját nyilvános kulcsára cserélné a másik fél kulcsát, akkor ellophatja titkos üzeneteinket.

2.3. Tanúsítványok

Egy nyilvános kulcs és tulajdonosa összetartozását ún. tanúsítványok rögzítik, melyeket különleges szervezetek, a hitelesítés szolgáltatók bocsátanak ki. Tanúsítványt bármilyen jogi entitás, magánszemély vagy szervezet igényelhet. A kibocsátott tanúsítványok eredetiségét a hitelesítés-szolgáltató a saját digitális aláírásával szavatolja, amelyet ellenőrizve megbizonyosodhatunk a kapott nyilvános kulcs és a tulajdonos összetartozásáról.

Ily módon egy felhasználónak elég a hitelesítés szolgáltatói tanúsítványokat beszereznie ahhoz, hogy különféle felhasználói tanúsítványokat fogadhasson, és azokban megbízhasson.

2.4. A „Hash and Sign” paradigma

Ebben a fejezetben vezetem be a digitális aláírás fogalmát, amely egyrészt a lenyomatképzésre, másrészt az aszimmetrikus rejtjelezés használatának megfordítására épül.

Az első lépésben az aláírandó dokumentumot egy alkalmas lenyomatképző függvényen átengedve megkapjuk a dokumentum lenyomatát, amely az aláírás alapja. A lenyomatképzés azt szavatolja, hogy az aláírás nem egy teljesen másik dokumentumhoz, vagy nem a szóban forgó dokumentum egy kicsit módosított variánsához tartozik. Az első esetben az adja a biztosítékot, hogy két különböző dokumentum lenyomata különböző, a második esetben pedig arra lehet építeni, hogy az alapdokumentum egy pici módosítása is nagy változást okoz annak lenyomatában.

A második lépésben az aláíró fél a dokumentum lenyomatát saját titkos kulcsával rejtjelezi. Ezt hívtam fentebb az aszimmetrikus rejtjelezés megfordításának, hiszen az üzenetrejtjelezéssel ellentétben a titkos kulcsot most rejtjelezésre, nem pedig kikódolásra használjuk fel. Az aszimmetrikus RSA rejtjelező algoritmus esetében ez semmiféle problémát nem okoz. Ez a tulajdonság nem feltétlenül igaz minden aszimmetrikus rejtjelező algoritmusra. Nem minden algoritmussal lehet olyan kulcs párt generálni, amellyel lehet a tulajdonosnak rejtjeles üzeneteket küldeni, valamint digitális aláíró kulcsként is használható.

Az digitális aláírás ezzel a két lépéssel el is készíthető. Érdemes megvizsgálni, hogy milyen biztonságot nyújt az ilyen aláírás. Álljon itt néhány szemléltető magyarázat.

2.4.1. Letagadhatatlanság

Az aláíró a lenyomatot rejtjelezi, amit később le nem tagadhat. Az aláírás rábizonyításához a bizonyító fél képzi a dokumentum lenyomatát, majd az aláírást kirejtjelezi az aláíró nyilvános kulcsával. A nyilvános kulcsot az aláíró fél tanúsítványából lehet megbízható módon megtudni. Ha a képzett és a kirejtjelezett lenyomat megegyezik, azzal bizonyított az aláírás ténye, és a tanúsítványkiadás szabályai miatt az aláírásnak jogi súlya lehet.

2.4.2. Hitelesség

Az aláíró személye egyértelmű is. A matematikai módszerek szavatolják, hogy nem lehetséges egy titkos kulcs nélkül eredetinek látszó aláírást létrehozni.

2.4.3. Integritás

Az integritás követelménye arra vonatkozik, hogy a dokumentum észrevétlenül nem módosítható az aláírás után. A követelmény teljesül a lenyomatképző függvény

felhasználásával. Ugyanis bármilyen kis módosítás a függvény bemenetén megváltoztatja a kimenetet, így az ellenőrzéskor az eredeti és a képzett lenyomat el fog térni egymástól.

3. A jogi környezet

3.1. 2001. évi XXXV. törvény és módosítása

A [6] és [7] törvények (továbbiakban Eat.) rendelkeznek arról, hogy (néhány kivétellel) legalább fokozott biztonságú elektronikus aláírással ellátott elektronikus iratok elfogadhatók ott, ahol jogszabály írásba foglalást ír elő. Ezzel az elektronikus iratok „polgárjogot nyertek” Magyarországon.

Az Eat. kétféle, joghatással bíró elektronikus aláírást definiál:

a) fokozott biztonságú elektronikus aláírás,
b) minősített elektronikus aláírás.

A fokozott biztonságú elektronikus aláírás egyértelműen azonosítja az aláírót, továbbá felismerhetővé teszi, ha a dokumentum tartalma az aláírás elhelyezése óta megváltozott.

A minősített elektronikus aláírás a fokozott biztonságúhoz képest „erősebb”, mivel minősített tanúsítvány tartozik hozzá, és ún. biztonságos aláírás-létrehozó (BALE) eszközzel hozták létre, ennél fogva több joghatás kapcsolódik hozzá.

Az Eat. az elektronikus aláírás gyakorlati használatához négy szolgáltatástípust definiál:

a) elektronikus aláírás hitelesítés-szolgáltatás,
b) időbélyegzés,
c) aláírás-létrehozó eszközön az aláírás-létrehozó adat elhelyezése,
d) elektronikus archiválás szolgáltatás.

A hitelesítés szolgáltató tanúsítványokat bocsát ki. A tanúsítvány kiadásának előfeltétele, hogy az igénylő hitelesen igazolja a személyazonosságát. Ettől fogva a tanúsítvány lejáratáig vagy visszavonásáig az igénylő képes a személyazonosságát elektronikusan is igazolni, valamint képes a tanúsítványhoz tartozó egyedi és titkos aláírás-létrehozó eszközzel iratokon elektronikus aláírást elhelyezni.

A hitelesítés szolgáltató továbbá nyilvántartja, és nyilvánosan elérhetővé teszi a nála kibocsátott tanúsítványok állapotát, díjakat szab a tanúsítvány fenntartásáért, és meghatározza a tanúsítványokkal létrehozott aláírásért vállalt anyagi felelősségének felső határait. A minősített elektronikus aláírásokhoz jellemzően magasabb értékhatárok tartoznak a fokozott elektronikus aláírásokhoz képest.

Az érvényes tanúsítványok hitelességének megőrzése céljából az aláíró köteles haladéktalanul tájékoztatni a hitelesítés szolgáltatót, ha adatai megváltoztak, vagy a titkos magánkulcs kitudódott, esetleg elveszett. Ekkor a hitelesítés-szolgáltató a tanúsítvány érvényességét felfüggeszti, esetleg végleg visszavonja, továbbá ennek tényét közzéteszi. Visszavonás történik akkor is, ha a tanúsítvány érvényességi ideje lejár.

Az időbélyegzés szolgáltató egy dokumentumhoz az aktuális pontos időt rendeli hozzá.

Az archiválás szolgáltató dokumentumokat, és az elektronikus aláírások hosszú távú ellenőrizhetőségéhez szükséges érvényességi láncot tárolja megbízható módon. Továbbá igazolást adhat ki arról, hogy egy elektronikus aláírás a létrehozás pillanatában érvényes volt.

4. A szabványok

4.1. X.509 PKI Certificate and CRL profile (RFC 3280)

Az X.509 egy ITU-T szabvány nyilvános kulcsú infrastruktúra számára. Az első változatot 1988-ban adták ki az X.500 szabványhoz kapcsolódódóan. Mivel az X.500 sosem valósult meg teljesen, valamint szigorú hierarchikus modellje nem illett rá a világhálóra, az IETF átdolgozta az X.509-et az internet rugalmas szerkezetéhez illeszkedően, így az X.509 végül IETF ajánlásként vált ismertté.

Az X.509 megszabja az interneten működő nyilvános kulcsú infrastruktúrában használt tanúsítványok és visszavonási listák szerkezetét, valamint ad egy algoritmust a hitelesítési útvonal érvényességének eldöntésére.

Egy X.509 tanúsítvány alapvető mezői a következők.

- Verziója (Version). A tanúsítvány formátumának verziója.
- Sorszáma (Serial Number). A kibocsátóval együtt azonosítja a tanúsítványt.
- Algoritmus azonosító (Algorithm ID). Annak az algoritmusnak a neve, amellyel a kibocsátó aláírta a tanúsítványt.
- Kibocsátó X.500 neve (Issuer). A sorszámmal együtt azonosítja a tanúsítványt.
- Érvényességi idő (Validity).
- Tárgy (Subject). A tanúsítvány tárgyának X.500 neve. Joghatással bíró tanúsítvány esetén az a jogi vagy magánszemély, akihez a tanúsítvány a kulcspárt hozzárendeli.
- A tanúsított nyilvános kulcs (Subject Public Key Info). A tárgyhoz hozzárendelt nyilvános kulcs értéke, és annak az algoritmusnak az azonosítója, amellyel a kulcs használható.
- Aláíró algoritmus (Certificate Signature Algorithm). Annak az algoritmusnak az azonosítója, amellyel a kibocsátó a tanúsítványt aláírja.
- Aláírás (Certificate Signature). A kibocsátó aláírása a tanúsítványon.

Egy X.509 visszavonási lista szerkezete a következő.

- Verziója (Version). A visszavonási lista formátumának verziója.

- Algoritmus azonosító (Algorithm ID). Annak az algoritmusnak a neve, amellyel a kibocsátó aláírta a visszavonási listát.
- Kibocsátó X.500 neve (Issuer). A kibocsátási idővel együtt azonosítja a visszavonási listát.
- Kibocsátási idő (thisUpdate). A visszavonási lista kibocsátási ideje. A kibocsátó nevével együtt azonosítja a visszavonási listát.
- Következő frissítés (nextUpdate). A következő visszavonási lista kiadásának ideje.
- Visszavont tanúsítványok listája (revokedCertificates). A lista minden eleme kötelezően tartalmazza a visszavont tanúsítvány sorszámát a kibocsátónál, a visszavonás dátumát, valamint választható egyéb mezőket, mint például a visszavonás kiváltó okát.
- Aláíró algoritmus (Certificate Signature Algorithm). Annak az algoritmusnak az azonosítója, amellyel a kibocsátó a visszavonási listát aláírja.
- Aláírás (Certificate Signature). A kibocsátó aláírása a visszavonási listán.

4.2. XML Signature (RFC 3275)

Miután elkészült a dokumentumhoz az elektronikus aláírás, és összegyűltek az egyéb szükséges információk, egységbe kell zárni őket. Ennek eszköze az XML elektronikus aláírás (XMLDSIG), melyet a [3] dokumentum ír le.

Az XML elektronikus aláírás egy XML dokumentum, melynek formája az alábbi.

```
<Signature ID?>
   <SignedInfo>
     <CanonicalizationMethod/>
     <SignatureMethod/>
     (<Reference URI? >
       (<Transforms>)?
       <DigestMethod>
       <DigestValue>
     </Reference>)+
   </SignedInfo>
   <SignatureValue>
  (<KeyInfo>)?
  (<Object ID?>)*
</Signature>
```

1. ábra – XML aláírás szerkezete

Az aláírás a teljes SignedInfo elem lenyomatának a rejtjelezésével keletkezik, és értéke a SignatureValue elembe kerül. Az aláírás tetszőleges számú dokumentumra vonatkozhat, ezek lenyomata külön-külön Reference elemek alá kerül, egy-egy DigestValue elembe.

Az aláírás hordozhatja az aláírás-ellenőrző adatokat is a KeyInfo elemben. A KeyInfo elem közvetlenül nem kerül aláírásra, mert nem része a SignedInfo elemnek. Lehetőség van azonban egy Reference elemen keresztül közvetett módon bevonni az aláírt elemek halmazába.

A Reference elemek opcionálisan hivatkozhatnak arra a dokumentumra, amelynek a lenyomatát hordozzák, sőt, az XML aláírás akár magában is hordozhatja e dokumentumokat az opcionális Object elemekben.

Az aláírás elkészítéséhez először a Reference elemeket kell létrehozni. A dokumentumokat a Transforms tagban rögzített transzformációknak kell alávetni, majd a DigestMethod metódussal el kell készíteni a lenyomatot, és azt a DigestValue elembe kell tenni. Ezután a SignedInfo elemet kell a CanonicalizationMethod elemben rögzített algoritmussal kanonizálni, majd a SignatureMethod elem szerint alá kell írni, az aláírást pedig a SignatureValue elembe kell tenni.

Az aláírás ellenőrzéséhez először a Reference elemekhez tartozó lenyomatokat kell elkészíteni, majd összevetni az XML aláírásban levőkkel. Ha mindegyik lenyomat helyes, akkor a dokumentumok nem módosultak az aláírás óta. Ezután el kell készíteni a SignedInfo elem kanonizált formáját, majd annak lenyomatát, és össze kell vetni a SignatureValue elem értékének feloldásával. A rejtjelezés feloldását a SignatureMethod szerinti algoritmussal lehet elvégezni az aláírás-ellenőrző adat ismeretében.

4.3. XML Advanced Electronic Signature (ETSI TS 101903)

Az XML elektronikus aláírás tartalmára tett eddigi megkötések még nem elegendőek ahhoz, hogy az a gyakorlatban jól használható legyen. További követelmények fogalmazódnak meg, amelyek megoldására különböző elektronikus aláírás formátumokat dolgoztak ki. Ezeket a [4] XAdES szabvány írja le. A meghatározott aláírás-típusok az alábbiak.

1. BES, az alapszintű elektronikus aláírás. A BES tartalmaz egy digitális aláírást, az aláíró tanúsítványát aláírt formában, és egyéb aláírt/nem aláírt opcionális jellemzőket. Ez a formátum megfelel az Eat. fokozott biztonságú aláírásának,

hitelesítést és integritásvédelmet biztosít, de letagadható, mivel nem rögzíti az aláírás időpontját, és így nem bizonyítható, hogy az aláíró tanúsítvány érvényes volt az aláírás létrehozásának időpontjában.

2. EPES, az egyértelműen szabályozott elektronikus aláírás. Az EPES egy alapszintű aláírásra épül, és kiegészíti azt egy aláírási szabályzat azonosítójával, amit kötelező aláírni. Ezáltal egyértelművé teszi az aláírás érvényesítési módját. Az aláírási szabályzat fogalma a dokumentum elején olvasható. Az EPES a BES-hez hasonlóan hiteles, integritásvédett, de letagadható.

3. XAdES-T, az aláírás időpontjával kiegészített elektronikus aláírás. Az XAdES-T az előző két forma egyikét egészíti ki egy (megbízható) időbélyegzés szolgáltatótól származó időpecséttel. A megbízható időbélyeg egy kezdeti lépést jelent az aláírás hosszú távra szóló érvényességének biztosítására. Az időpecsét egy aláírt dokumentum, mely az időbélyegzés-szolgáltató órája által mutatott időből, és egy dokumentum lenyomatából áll. A lenyomatot az ügyfél küldi el a szolgáltatónak, az aláírás pedig a szolgáltató tanúsítványával történik.

4. XAdES-C, a teljes körű érvényesítő adatokkal kiegészített elektronikus aláírás. Az XAdES-T formához képest hivatkozásokat tartalmaz a teljes érvényességi láncra, és a szükséges tanúsítvány-visszavonási listákra. Ezek beszerzése időigényes lehet, ezért ilyen aláírást létrehozni is időt vesz igénybe. Előnye, hogy az aláírás későbbi érvényesítéséhez minden adat rendelkezésre áll. Erre a típusra akkor lehet szükség, ha az aláíró, és esetleg a hitelesítés-szolgáltató tanúsítványának lejárta után is szükség lesz az aláírás ellenőrzésére. Az XAdES-C aláírást nem lehet „azonnal” létrehozni, mivel annak eldöntése, hogy egy tanúsítvány egy időpillanatban érvényes-e, csak egy bizonyos kivárási idő eltelte után lehetséges teljes bizonyossággal, hiszen például a hitelesítés-szolgáltatónak is idő kell a visszavonási listák frissítéséhez. A kivárási idő az [5] ITKTB ajánlásban az elvárt biztonsági szinttől függően 1-3 nap.

5. XAdES-X, az XAdES-C elektronikus aláírás kibővített változata. Három fajtáját határozták meg.
   Az XAdES-X Long nem csak hivatkozásokat tartalmaz az érvényesítő adatokra, hanem magukat az adatokat is magába foglalja. Erre akkor lehet szükség, ha fennáll a veszélye, hogy a tanúsítvány-láncok és visszavonási listák elvesznek.

   a. A XAdES-X Type 1 formátum a XAdES-C-t egészíti ki egy időbélyeggel, mely a teljes elektronikus aláírásra készül el. Ezzel a típussal kezelhetők azok az esetek, amikor kompromittálódik vagy az aláíró tanúsítványt kibocsátó hitelesítés-szolgáltató, vagy az érvényesítő adatokat aláíró szolgáltató aláíró kulcsa. Ez a forma kombinálható az elsővel, ekkor XAdES-X Long Type 1 a neve.

   b. A XAdES-X Type 2 forma szintén a szolgáltatók kulcsának kompromittálódása ellen véd úgy, hogy az egyes érvényesítő adatokra külön-külön tesz időbélyeget. Ez a forma kombinálható az elsővel, ekkor XAdES-X Long Type 2 a neve.

6. XAdES-A, az archív érvényesítő adatokkal kiegészített elektronikus aláírás. A XAdES-A formátum valamelyik XAdES-X aláírás-fajtára épülhet, és azokra az esetekre biztosítja az elektronikus aláírás hitelességét, integritását és letagadhatatlanságát, amikor az aláírás bármely eleme meggyengült. Ez lehet például az aláírás elkészítéséhez használt algoritmus, kulcs, tanúsítvány. A XAdES-A egy időbélyeggel látja el a teljes aláírást, amelynek kriptográfiai erőssége mindig elég erős lehet a kor technológiai követelményeihez igazodva. Az Eat. az archiválási-szolgáltatók számára a következőt írja elő.

[6] Eat. 16/G. § (2) A szolgáltató köteles a Hatóság határozata szerinti, elfogadott kriptográfiai algoritmuson alapuló minősített elektronikus aláírást elhelyezni és minősített szolgáltató által kibocsátott időbélyegzőt elhelyezni vagy elhelyeztetni az érvényességi láncon:

1. a szolgáltatási szabályzatban meghatározott időközönként;
2. a határozatban előírt időpontban.

A [5] dokumentum a XAdES-C formátum használatát javasolja a magyar közigazgatás elektronikus kommunikációjában, ha a hosszú távú letagadhatatlanság követelmény. Ekkor az aláíró alkalmazások legalább a XAdES-EPES formátum létrehozását kell támogassák, amelyből majd vagy az aláíró, vagy az aláírást továbbító szolgáltatás, vagy az aláírást fogadó/feldolgozó/ellenőrző címzett készíti el a XAdES-C formátumot, amely már tartalmaz minden szükséges információra vonatkozó hivatkozást a hosszú távú letagadhatatlanság biztosításához.

A dokumentum azért nem javasolja a XAdES-X vagy XAdES-A formátum használatát, mert azokra előreláthatólag még egy évtizedig nem lesz szükség, valamint terjedelmük is jóval nagyobb a XAdES-C változatánál. Ha mégis szükség mutatkozna a használatukra, akkor a meglévő XAdES-C aláírásokból készíthető X vagy A típusú XAdES aláírás, mivel minden szükséges hivatkozás része az aláírásnak.

Az alábbi XML struktúra az XMLDSIG és a különböző XAdES formátumok tartalmát tekinti át:

2. ábra - Az egymásra épülő XAdES formátumok szerkezete

4.4. MELASZ XAdES profil

A Magyar Elektronikus Aláírás Szövetség (MELASZ) 2005. szeptemberében fogadta el az ETSI XAdES szabványra alapuló aláírás profilját. A MELASZ a hazai elektronikus aláíró alkalmazást fejlesztő cégeket tömörítő civil szervezet.

A profil tartalmazza a XAdES szabvány minden kötelező elemét, a választható elemek jelentős részének használatát azonban nem engedi meg. A szűkítést a magyarországi felhasználói és szolgáltatói szempontok figyelembevételével tették meg. Szűkítették a használható XML elemek, valamint a hivatkozható algoritmusok körét. Ezzel egyszerűsödik a profilnak megfelelő alkalmazások megvalósítása, hiszen kevesebb elem és algoritmus kezelésére kell felkészülni, míg az alkalmazás funkcionalitása ezzel nem csökken.

Kikerültek a használható elemek köréből a SignatureProductionPlace, SignerRole, CommitmentTypeIndication, AllDataObjectsTimeStamp, IndividualDataObjectsTimeStamp, AttributeCertificateRefs, AttributeRevocationRefs elemek.

A használható algoritmusok köre is szűkült, így csak a következők használhatók.

- Kanonizálásra csak a megjegyzések nélküli XML kanonizáció.
- Aláírásra csak az SHA-1 lenyomatképzéssel kombinált RSA algoritmus.
- Lenyomatképzésre csak az SHA-1 algoritmus.

Azon transzformációk köre is szűkült, amelyeket az aláírt dokumentumon el lehet végezni.

- Az XPath szűrés mint transzformáció nem megengedett.
- Az Enveloped Signature Transform mint transzformáció nem megengedett.
- Az XSLT transzformáció nem megengedett.

5. A felhasználók

A technológia biztosítja, hogy elektronikusan keletkező iratokat hitelesen alá lehessen írni, és ez az aláírás évek múltán is megőrizze biztonságát. Szabványok biztosítják, hogy az egyes szoftverek képesek legyenek együtt működni.

Az archiválás feladata dokumentumok hosszú távú megőrzése. Ha megvalósul olyan alkalmazás, amely teljesen papírmentes, továbbá a keletkezett iratokat meg kell őrizni, akkor elektronikus iratokat kell archiválni.

Az archiválás egyfelől megbízható adattárolási probléma, amely az állományok fizikai elveszését hivatott elsősorban redundáns tárolással megelőzni. Másfelől a digitális aláírás logikai felépítésében rejlő gyengeségeket kell kezelni, amelyek hosszú távon jelentkezhetnek.

Meggyengülhet valamely kriptográfiai módszer, amelyet az aláíráshoz használtak. Akár a lenyomatképző, akár a kódoló eljárásról kiderülhet, hogy gyengébb, mint amilyennek gondolták, ha valaki talál egy gyengeséget bennük. Innen természetesen még hosszú út vezetne archivált iratok megváltoztatásához, vagy új hamis iratok készítéséhez, de az elvi lehetőség fennáll.

A használt eljárások idővel természetesen avulnak el a számítógépek fejlődésével, hisz minden gyakorlati kriptográfiai eljárás feltörhető nyers számítási erővel. Az ilyen jellegű veszélyek ellen is védekezni kell.

Ha feltételezzük, hogy mindig lesz aktuálisan erős lenyomatképző és aláíró eljárás, akkor az archivált iratokat időközönként meg lehet erősíteni egy-egy újabb, erősebb időbélyeggel, ami tanúsítja, hogy a bélyegzés időpontjában a dokumentumon szereplő aláírások még hitelesek voltak.

5.1. Ügyintézés

Az állami szervekkel kapcsolatos ügyek intézése mindenkit érint, és mindenkinek sok idejét viszi el, melynek nagy részét teszi ki a hivatalok felkeresése, ott az illetékes megtalálása, sorának kivárása. Az ilyen eljárás kiváltására több szolgáltatás is indult a közelmúltban, melyek az elektronikus archiválás nagy felhasználói lehetnek.

Az APEH elektronikus bevallás szolgáltatást indított, amely eleinte csak a nagy adózóknak, később már minden adófizetőnek is elérhetővé vált. Ehhez saját belső

hitelesítés szolgáltatót hozott létre, de hamarosan elfogadja a piaci szolgáltatók minősített tanúsítványait is a bevallás aláírásához.

A magyarorszag.hu is kínál elektronikus ügyindítást regisztráció után. Ehhez alapesetben be kell menni egy okmányirodába, de minősített digitális aláírással rendelkezők teljesen online is regisztrálhatnak. Regisztráció után lehetőség van ügyek indítására, időpontkérésre az okmányirodába, valamint az adóbevallás elektronikus beküldésére.

2005. november 1-jével lép hatályba a Közigazgatási eljárásról szóló törvény (KET), amely kimondja, hogy a hivatalok ügyfelei online kapcsolatba léphetnek az ügyeiket intéző szervekkel egy központi rendszeren keresztül. Így várhatóan egyre kevesebb ügy miatt kell okmányirodába és más hivatalokba bemenni. „Jelenleg már Interneten keresztül igényelhető vállalkozói igazolvány, jogosítvány, lakcímigazolvány, forgalmi, útlevél és egyéb hivatalos irat, 2005 végéig 27-re nő azoknak az okmánytípusoknak a száma, amelyeket Interneten keresztül igényelhet a lakosság.” [9].

5.2. Intézményen belüli iratkezelés

Azokban az intézményekben, amelyekben sok, aláírással hitelesített belső iratot kezelnek, ezeket évekig megőrzik, kézenfekvő lehetőség a belső iratok számítógépes, digitálisan aláírt tárolása. Ekkor megtakarítás érhető el a papír tárolási költségein.

5.3. Elektronikus számlázás

A magyar jog 2004. áprilisa óta engedi meg a számlák csak elektronikus kiállítását, küldését és tárolását. A gyakorlatban nem valósult meg ilyen alkalmazás napjainkig, azonban 2006. január 1-től már az APEH is elfogadja az elektronikus számlákat hitelesnek, így nagy mennyiségű papíralapú számla kiváltása várható a közeljövőben. Elsősorban a nagy szállító cégek lehetnek érdekeltek, hiszen egymás között nagy számlaforgalmat bonyolítanak le. A nagy lakossági ügyfélkörrel rendelkező szolgáltatóknál inkább lassú váltás várható, melynek korlátja a lakossági internet használat aránya.

Ilyen számlakibocsátók a közüzemi szolgáltatók, valamint a nagy lakossági bankok, vagy az ilyen szempontból optimális helyzetben lévő internet szolgáltatók, hisz nekik minden ügyfelük rendelkezik elektronikus elérhetőséggel. Ezek a szervezetek számlák millióit állítják ki évente, amelyek költsége postázással együtt darabonként körülbelül 100Ft. Elektronikus számlázás esetén a szolgáltató fokozott biztonságú digitális aláírással és időpecséttel látja el a számlát a biztonság érdekében. Ennek a költsége a papír-alapú

megoldás 10-30%-a, amit legnagyobb részben az időbélyegzés és az aláíró tanúsítvány díjai tesznek ki. [8] A számlákat a fogadó csak akkor köteles archiválni, amennyiben azt az APEH felé el szeretné számolni. Ez a kötelezettség a vállalkozásokat érinti, így ők az archiválási szolgáltatások lehetséges felhasználói.

6. A kifejlesztett alkalmazás

6.1. A fejlesztés célja

A fejlesztés célja egy olyan alkalmazás létrehozása volt, amely képes a MELASZ ajánlásnak megfelelő aláírás típusokkal kapcsolatos feladatok elvégzésére.

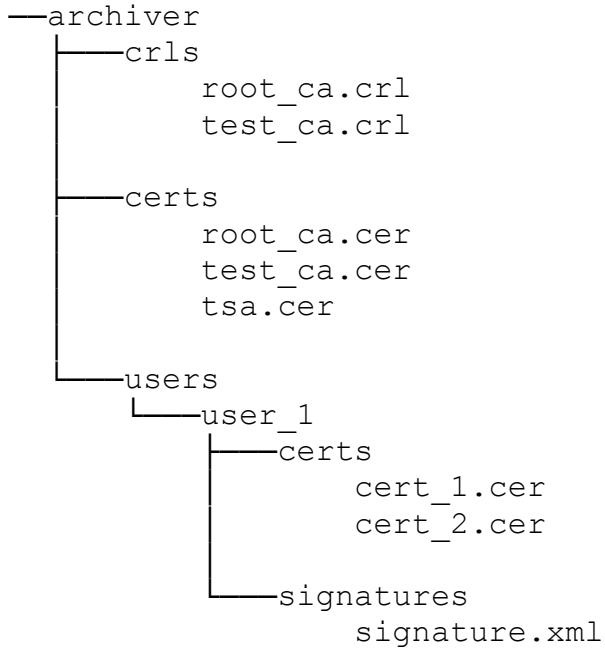
A rendszer

- lehetővé teszi több-felhasználó párhuzamos elérését,
- felhasználói felületét Web böngésző jeleníti meg,
- külön üzemeltető felülettel is rendelkezik.

6.2. A rendszer adatmodellje

A rendszerben kezelt hosszú élettartamú adatok tárolása két módon történik. A fájlokban érkező adatokat a fájlrendszerben tárolja a rendszer, így azok a megszokott, kiforrott, kényelmes formában állnak a fejlesztők és a rendszer üzemeltetői részére. A rendszer egyéb adatait relációs adatbázisban tárolja.

6.2.1. Tárolás fájlrendszerben



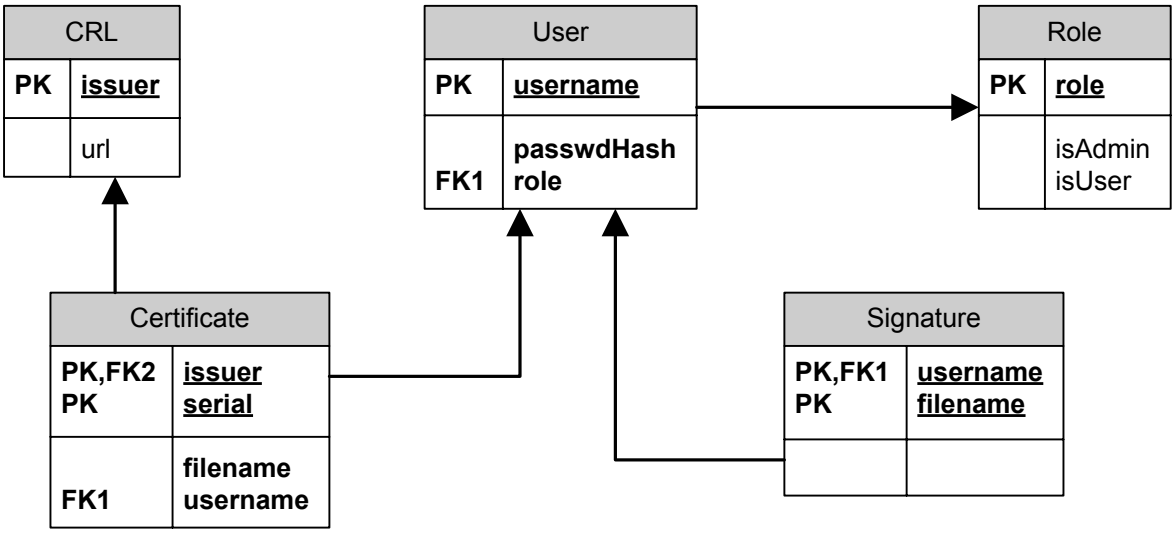
3. ábra - A fájlok tárolásának szerkezete

A fájlrendszerben tárolt állományok könyvtárakba rendezettek. A *crls* könyvtár tartalmazza az egyes szolgáltatók által kibocsátott tanúsítvány-visszavonási listákat. A *certs* könyvtár a szolgáltatók tanúsítványait tartalmazza, amelyekkel a felhasználók tanúsítványait, a visszavonási listákat, vagy az időbélyegeket írják alá.

Az egyes felhasználói könyvtárak a *users* könyvtárban kapnak helyet. Ezeken belül a *certs* könyvtár a felhasználói tanúsítványokat, míg a *signatures* könyvtár a felhasználó által feltöltött aláírásokat gyűjti össze.

6.2.2. Tárolás relációs adatbázisban

A rendszer alapvető működési adatait relációs adatbázis tárolja, melynek adatmodellje a következő ábrán látható.



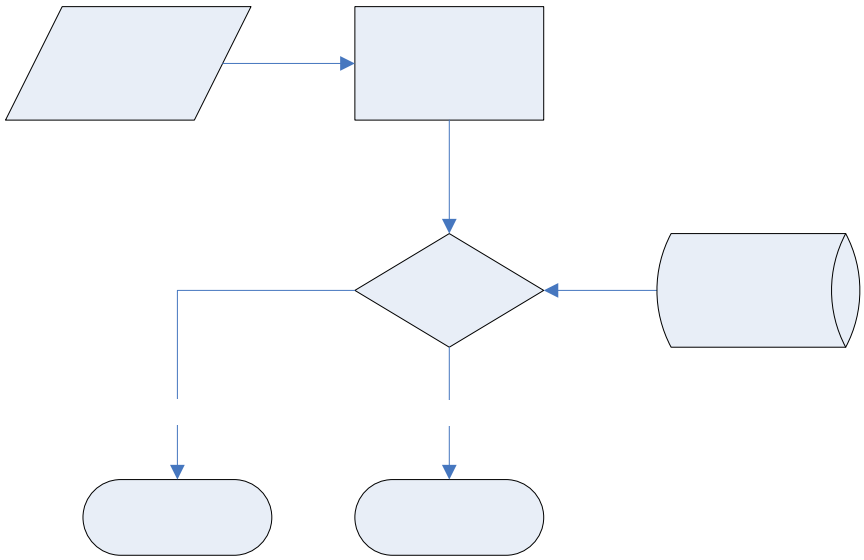
4. ábra A rendszer adatmodellje

A *Certificate, CRL, Signature* táblák adminisztratív célokat szolgálnak. Rajtuk keresztül tartja nyilván a rendszer, hogy milyen tanúsítványok és aláírások érkeztek, és hogy milyen tanúsítvány visszavonási listákat kell beszerezni, tárolni. Ezen három entitást alapvetően diszken, fájlokban tárolja a rendszer, ez adja a táblák adminisztratív jellegét.

A *Role* tábla megadja a lehetőséget a felhasználók jogosultságainak szabályozására. Például, a szoftver karbantartó felületét csak a rendszergazdai jogosultsággal rendelkező felhasználók érhetik el.

A *User* tábla a rendszer felhasználóit és szerepköreiket tartalmazza. Látható, hogy egy felhasználónak több aláírt dokumentuma és több tanúsítványa is lehet. Ezeken kívül

jellemzi a jelszavának lenyomata és a szerepköre. A jelszó kriptográfiai lenyomatának tárolása biztonsági szempontból fontos. Elkerülhető vele a jelszavak lista-szerű tárolása a rendszerben, amely megoldás kockáztatja a lista teljes ellopását. A lenyomat tulajdonságai megegyeznek a második fejezetben leírtakkal. Tehát egy jelszó lenyomata gyorsan képezhető, de a lenyomatból nem állítható vissza a jelszó. Biztosított továbbá, hogy két különböző jelszó lenyomata nem fog megegyezni (ütközni). Így elegendő az adatbázisban a jelszavak listája helyett a lenyomatok listáját tárolni, jelszó ellenőrzéskor pedig az ellenőrizendő jelszó lenyomatát kell képezni és összevetni a tárolt lenyomattal.



5. ábra - Jelszó ellenőrzése

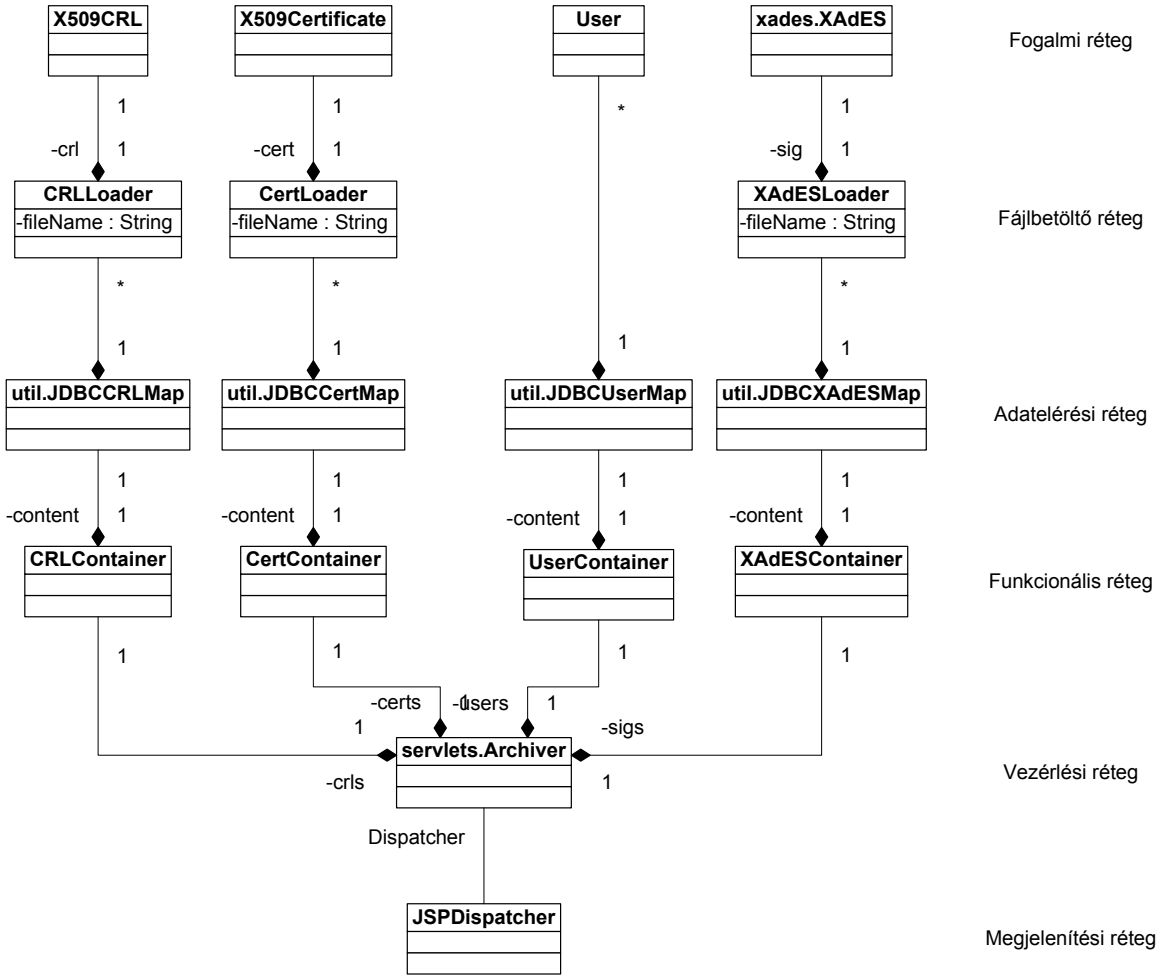
Beírt jelszó Leny

Meg

nem

6.3. A rendszer szerkezete

A rendszer szerkezete a következő ábrán látható. A struktúra elsősorban az adattárolás követelményei után szervezett.



6. ábra - A rendszer szerkezete

6.3.1. Fogalmi réteg

A legfelső szinten helyezkednek el a fogalmi modell elemei, a tanúsítvány visszavonási lista, a tanúsítvány, a felhasználó és az aláírás. Közöttük az előző fejezetben leírt kapcsolati viszonyok állnak fenn.

6.3.2. Fájlbetöltő réteg

A tárolásuk miatt fájlrendszerhez kötődő objektumok kezelését a *Loader objektumok végzik el. Szolgáltatásaik a következők.

- Tárolják a fájlok elérési útját.

- Betöltik a fájlt a Java virtuális gépbe, így hagyományos objektumként kezelhető.
- Elmentik az objektumot a fájlba, ha megváltozott.
- Törlik a fájlt, ha szükséges.

6.3.3. Adatelérési réteg

A harmadik réteg vonja be a relációs adatbázist a tárolásba. Az adatbázis tartalmazza az előző fejezetben szereplő táblákat, amelyek leírják az egyes visszavonási listák, tanúsítványok, felhasználók és aláírások egymás közötti viszonyát.

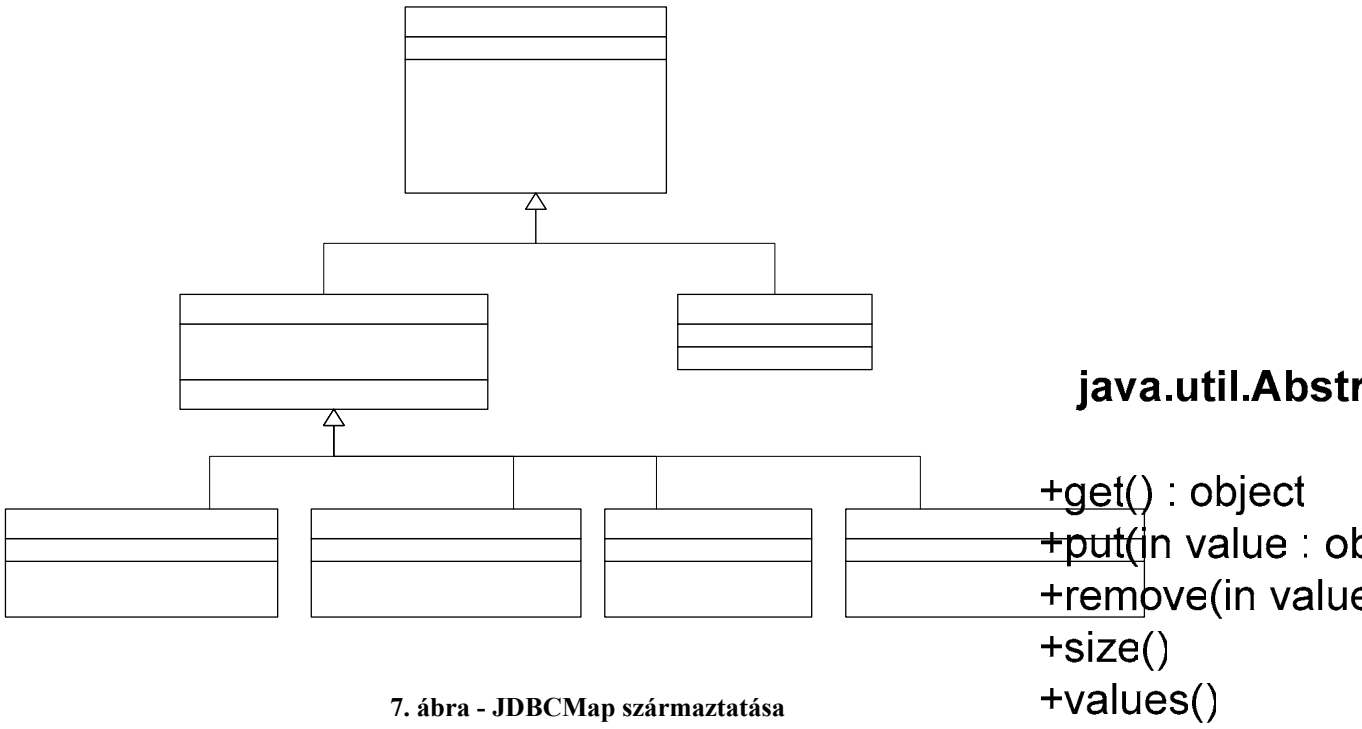
A fejlesztés korai szakaszában az adattárolás nem adatbázis-alapon, hanem objektumsorosítással történt. Erről az eljárásról részletes ismertető található ezen fejezet technológiákról szóló alfejezetében. A rendszer az adatokat a Java gyűjtemény keretrendszerének HashMap objektumaiban tárolta, amelyeket változtatásuk esetén sorosítással a lemezre mentett, majd a rendszer indulásakor betöltötte őket, így azok a lemezen mindig tükrözték a rendszer aktuális állapotát.

Amikor az objektumsorosítás helyett adatbázist kellett a rendszer „alá” beépíteni, alapvető cél volt hogy a változás hatása a kód lehető legszűkebb körét érintse, hisz minden változtatás a rendszerben egyben potenciális hibaforrás is. A relációs adattáblák felfoghatók (kulcs,érték) párok halmazaként, ahol minden sor egy ilyen pár. Kulcs lehet a tábla oszlopainak valamelyik olyan tulajdonságú részhalmaza, amely egyértelműen megkülönbözteti a sorokat. A kulcshoz rendelt érték pedig a tábla összes többi oszlopából képzett halmaz lehet. Másképp, a relációs tábla egy olyan *leképezést* definiál, amely kulcsokhoz értékeket rendel. Így a memóriabeli HashMap leképezése könnyen helyettesíthető adattábla-alapú leképezéssel.

Az implementáció szintjén végül a következő megoldás mellett döntöttem. A HashMap osztály közvetlen ősosztályából, az AbstractMap osztályból származtatok le egy olyan saját (JDBCMap) osztályt, amely relációs adatbázis-kezelővel áll kapcsolatban. A kapott objektumokat tároláskor „szétszedi”, majd beteszi egy tábla egy sorába, visszakéréskor pedig a kulcs alapján megkeresi az adott sort, majd abból az objektumot újra „összerakja”. A kapcsolódáshoz a Java szabványos JDBC technológiáját használtam.

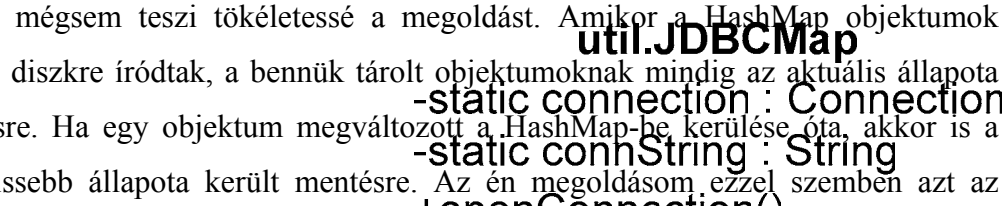
Az implementációban a JDBCMap osztály tartalmazza a kapcsolódáshoz szükséges változókat és metódusokat. A további származtatott osztályok az egyes tárolandó osztályokra nézve specifikus tárolásokat valósítanak meg. Bennük van pontosan rögzítve,

hogy hogyan kell az különféle objektumokat relációs táblába „szétszedni”, és onnan „összerakni”.

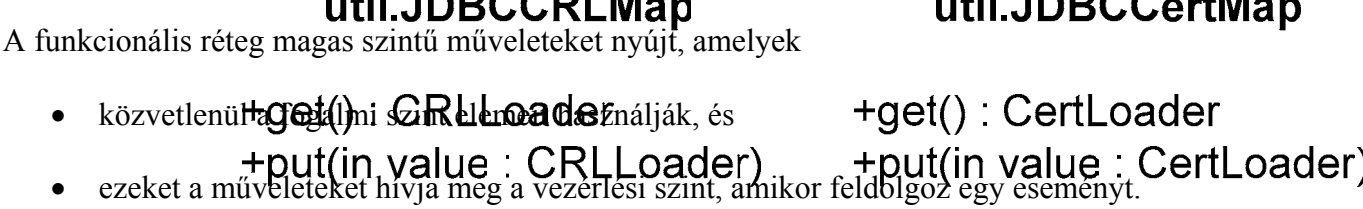


7. ábra - JDBCMap származtatása

Ahogyan látható az ábrából, a szoftver ilyen módosítása majdnem tökéletesen átlátszó a kód többi része szempontjából. A létrehozott négy különféle osztály behelyettesíthető a régebbi HashMap objektumok helyére. Felmerül azonban egy elvi probléma a használat során, amely mégsem teszi tökéletessé a megoldást. Amikor a HashMap objektumok sorosítással a diszkre íródtak, a bennük tárolt objektumoknak mindig az aktuális állapota került mentésre. Ha egy objektum megváltozott a HashMap-be kerülése óta, akkor is a mindig legfrissebb állapota került mentésre. Az én megoldásom ezzel szemben azt az állapotot menti, amelyik a put() metódus hívásakor fennállt. Ha utána az objektum tartalma változik, újabb put() hívás nélkül az nem kerül mentésre. Ennyi módosítás tehát szükséges volt a kód többi részében, így a megoldás nem lett tökéletesen átlátszó.



6.3.4. Funkcionális réteg



A funkcionális réteg magas szintű műveleteket nyújt, amelyek

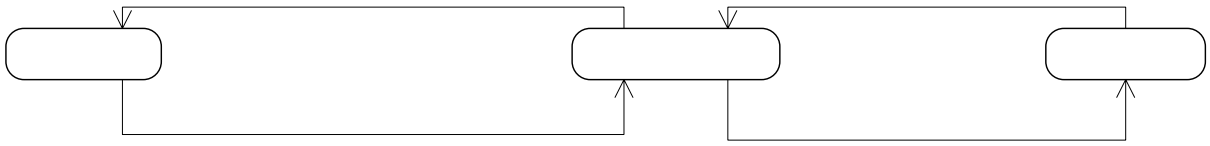
- közvetlenül a fogalmi szint elemeit használják, és
- ezeket a műveleteket hívja meg a vezérlési szint, amikor feldolgoz egy eseményt.

Az egyes entitásokat, és a rajtuk értelmezett műveleteket a következő táblázat foglalja össze. A frissítés művelete a visszavonási listára vonatkozik, melyet a hozzá tartozó URL címről tölti le a rendszer. A mentés művelete azon entitásokra értelmezett, amelyek adatbázisban tárolt adatai megváltozhatnak.

	Tanúsítvány-visszavonási lista	Tanúsítvány	Felhasználó	Aláírás
Felvétel	X	X	X	X
Kikérés	X	X	X	X
Összes kikérése	X	X	X	X
Eltávolítás	X	X	X	X
Frissítés	X			
Mentés	X		X	X

6.3.5. Vezérlési réteg

A vezérlési réteg az alkalmazás dinamikus viselkedésének központi eleme. Központi szerepét a következő ábra szemlélteti.



8. ábra - Vezérlés

A vezérlés fogadja a beérkező kérést, majd annak tartalmától függően módosítja a modellt a funkcionális réteg metódusain keresztül. A módosított modellt lekérdezi, majd visszaküldi a kérőnek a megváltozott nézetet. Az implementációban a nézet egy böngészőprogramban megjelenő HTML oldal, a vezérlést Java Servlet kód végzi, a modell pedig az előzőekben bemutatott Java objektumokból áll. Utóbbi úgy készült el, hogy kizárólag J2SE technológiákat használjon. Ennek az előnye az, hogy a modell újrafelhasználható más szerkezetű rendszerekben, például könnyen építhető rá Webszolgáltatás (WebService), vagy átalakítható EJB modellé.

Fontos jellemzője a vezérlési rétegnek, hogy itt jelenik meg először a bejelentkezett felhasználó fogalma. Ilyenből több is lehet egyidejűleg, megkülönböztetett kiszolgálásuk az ún. munkameneteken keresztül történik.

6.3.6. Megjelenítési réteg

A megjelenítési réteg formázza a felhasználónak küldött választ. Bemenetként kapja a modell megváltozott paramétereit, a kimenete pedig egy HTML oldal. Az implementációban a megjelenítést JSP technológiát használó oldalak végzik. Ezek olyan futtatható fájlok, amelyekben statikus betétek is lehetnek. Például a HTML oldal fejléce gyakran statikus elem, míg az oldalon szereplő listák előállítása a kapott paraméterektől függ. Az oldal létrehozását befolyásoló paramétereket a vezérlési réteg szervlet objektuma állítja be.

6.4. Az alkalmazott technológiák

6.4.1. Adatelérési réteg

Az adatelérés két technológiát használ. Az aláírásokat XML fájlokban kezeli, az adatbázist pedig JDBC-n keresztül éri el.

6.4.2. XML kezelés

Az XML leírónyelv a XAdES alapja, így minden aláírás ilyen formában érkezik, készül, és tárolódik a fájlrendszerben.

Kezdetben a World Wide Web Consortium (W3C) által specifikált Document Object Model (DOM) implementációját használtam. Ez a csomag a specifikáció univerzalitása, nyelvfüggetlensége miatt – hasonlóan az OMG CORBA modellhez – nem használja ki a Java speciális előnyeit, amelyek a gyors, hibamentes kódolást segítik elő. Java környezetben használata nehézkes, idegennek hat. Aránytalanul sok időt vitt el az XML-kezelő részek megírása, melyek épp az alkalmazás központi elemét képezik. Ezért másik API-t kerestem.

Végül a JDOM API-t választottam ki, amelynek kialakításánál kihasználták a Java nyelv minden lehetőségét. Az API használja a Java Gyűjtemény Keretrendszert (Collections), és az 1.5-ös Java verzióban megjelenő típusos gyűjteményeket. Használatával jól érthető, jól olvasható, tömör, biztonságos, lényegretörő kódot lehet írni a W3C DOM-hoz képest töredék idő alatt.

Példaként álljon itt egy sokszor idézett kódrészlet. A feladat egy új dokumentum létrehozása egy gyökérelemmel és abban egy szöveges elemmel. A különbség magáért beszél.

W3C DOM megoldás:

```
DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
DocumentBuilder builder = factory.newDocumentBuilder();
Document doc = builder.newDocument();
Element root = doc.createElement("root");
Text text = doc.createText("This is the root");
root.appendChild(text);
doc.appendChild(root);
```

JDOM megoldás:

```
Document doc = new Document(
    new Element("root").setText("This is the root"));
```

6.4.3. Adatelérés objektumsorosítással

Adatbázis helyett kezdetben objektumsorosításra épült a rendszer. A technológia korlátai gyakran jelentkeztek, így használata értékes tapasztalatokkal szolgált.

Az objektumsorosítás a memóriában létező objektumok leképezése bájtok folyamává. Mivel a memóriában az objektumok változatos módon kapcsolódhatnak egymáshoz referenciákon keresztül, ezért elképzelhetők egy hivatkozási gráfként, amelynek csúcsai az objektumok, élei pedig a hivatkozások. Sorosításkor egy objektumot jelölünk ki gyökérként, majd az eljárás megkeresi az összes hivatkozásokon át elérhető objektumot, és azokat is belekódolja a folyamba. Visszaállításkor az eljárás dekódolja a folyamból az összes objektumot, azokat a memóriában létrehozza, majd visszaadja a forrásobjektumra mutató referenciát.

A módszer a betöltés-mentés műveletei túl mást nem támogat, így nagyobb információs rendszer hosszú távú adattárolását megoldani nem alkalmas. Két jellemző gyakorlati probléma merült fel a fejlesztés során.

Az első probléma akkor jelentkezett, amikor az adatokat sorosítással kimentettem egy fájlba, majd változtattam a kiírt osztályok szerkezetén. Például hozzávettem egy mezőt, vagy megváltoztattam egy tag nevét. A visszatöltő eljárás ilyenkor képtelen volt a mentett adatok értelmezésére, hiszen ilyen esetekre – változáskezelésre – nem volt felkészítve. Ekkor a tárolt adatok ugyan megmaradtak, de mivel a program nem tudta őket olvasni, újra be kellett vinni őket.

A második probléma a tesztadatok kézi elkészítésekor jelentkezik. A fejlesztő a tesztelés egyszerűsítése okán szívesen változtat meg adatokat a program két futása között közvetlenül az adatállományokban. Objektumsorosítás esetén ez nem lehetséges, mivel a tárolt adatoknak nincs olyan szabályos szerkezete, amely az olvashatósághoz, módosíthatósághoz elengedhetetlenül szükséges.

Hordozhatóság

Az alkalmazáslogikát, azaz az alsó négy réteget úgy terveztem, hogy hordozható legyen. Ezt a szervlet technológia is támogatja, így az alsó négy réteg ún. POJO, Plain Old Java Object osztályokból áll. Ez azt jelenti, hogy nem jelenik meg benne a futtatókörnyezetre nézve különleges hívás, mondhatni nem is tud az őt körülvevő környezetről. Ezt az alkalmazáslogikát könnyen át lehet ültetni más futtatókörnyezetbe, például komoly alkalmazásszerverbe, amely többféle felületen teheti elérhetővé a modellt. Lehet például egy Webszolgáltatás alapja.

6.4.4. Vezérlési réteg szervlettel

A vezérlési réteg a Sun által kifejlesztett szervlet technológiára épül, amely szerveroldali Java programozást tesz lehetővé. A HTTPServlet osztálynak négy fontos metódusát kell használni.

Az *init()* metódus az első amit a szervlet létrehozója meghív. Az objektum életciklusának további részében a *processRequest()* metódus dolgozza fel az érkező HTTP kéréseket. A metódus szerkezete az eseményvezérelt feldolgozó kódoknál megszokott hosszú switch-case elágazás. Minden kérés típushoz tartozik egy ág.

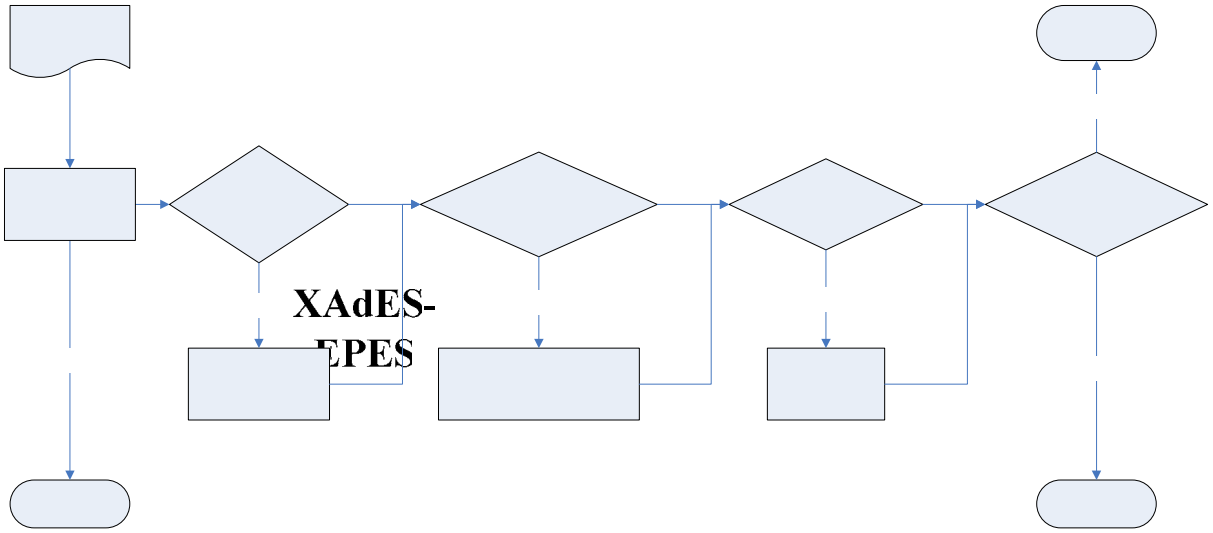
6.4.5. Megjelenítési réteg

A megjelenítési réteg a Sun JSP megoldását használja, amellyel preparált HTML sémák formájában lehet leírni a válaszoldalt.

6.5. Folyamatok

A rendszer a kezelt aláírásokon az alábbi négy folyamatot képes végrehajtani, melyek összhangban vannak a [2] MELASZ szabvány hatodik fejezetével. Az egyes folyamatokat folyamatábrák szemléltetik, a bennük szereplő feldolgozások leírását a következő fejezet írja le részletesen.

6.5.1. Az aláírás fogadása



9. ábra: Az aláírás fogadás utáni ellenőrzése

Amikor egy aláírás bekerülna rendszerbe, számos feltételnek kell megfelelnie, amelyeket a rendszer ellenőriz. A szabvány szerint ezeket a feltételeket az aláírást létrehozó alkalmazásnak kell biztosítania. Ha az elvárt minimális feltételek nem teljesülnek, a folyamat eredménye „érvénytelen”. Megfelelőség esetén három további elem meglétéről kell gondoskodni, amelyeket nem kötelezően az aláírást létrehozó alkalmazás is csatolhat.

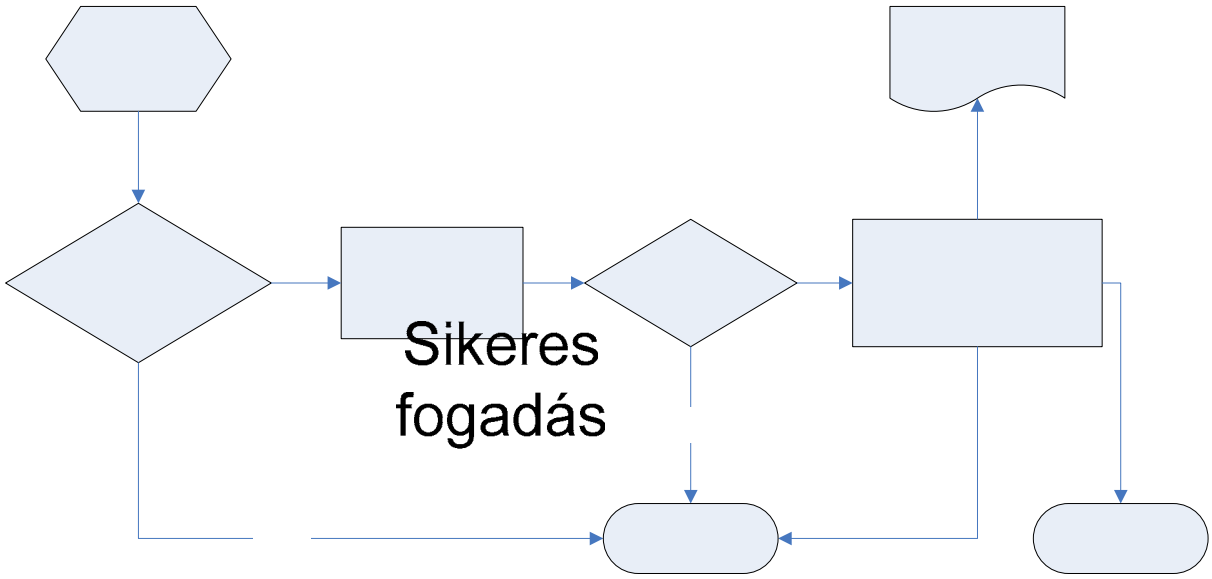
Az első az aláírás időbélyege (SignatureTimeStamp), amely hitelesen igazolja, hogy az aláírás létezett egy bizonyos időpillanat előtt. Ettől számítva kivárási idő letelte után lehet az aláírás kezdeti ellenőrzését, azaz a következő folyamatot végrehajtani. Emiatt célszerű az időbélyegzést minél korábban végrehajtani.

A második elem hivatkozásokat tartalmaz mindazon tanúsítványok visszavonási listájára, amelyek a hitelesítési láncban szerepelnek (CompleteRevocationRefs).

A harmadik elem tartalmazza a hitelességi láncban szereplő tanúsítványokat.

Ha mind a három elemet sikerül csatolni, akkor az aláírás fogadása sikerrel zárult.

6.5.2. Az aláírás kezdeti ellenőrzése



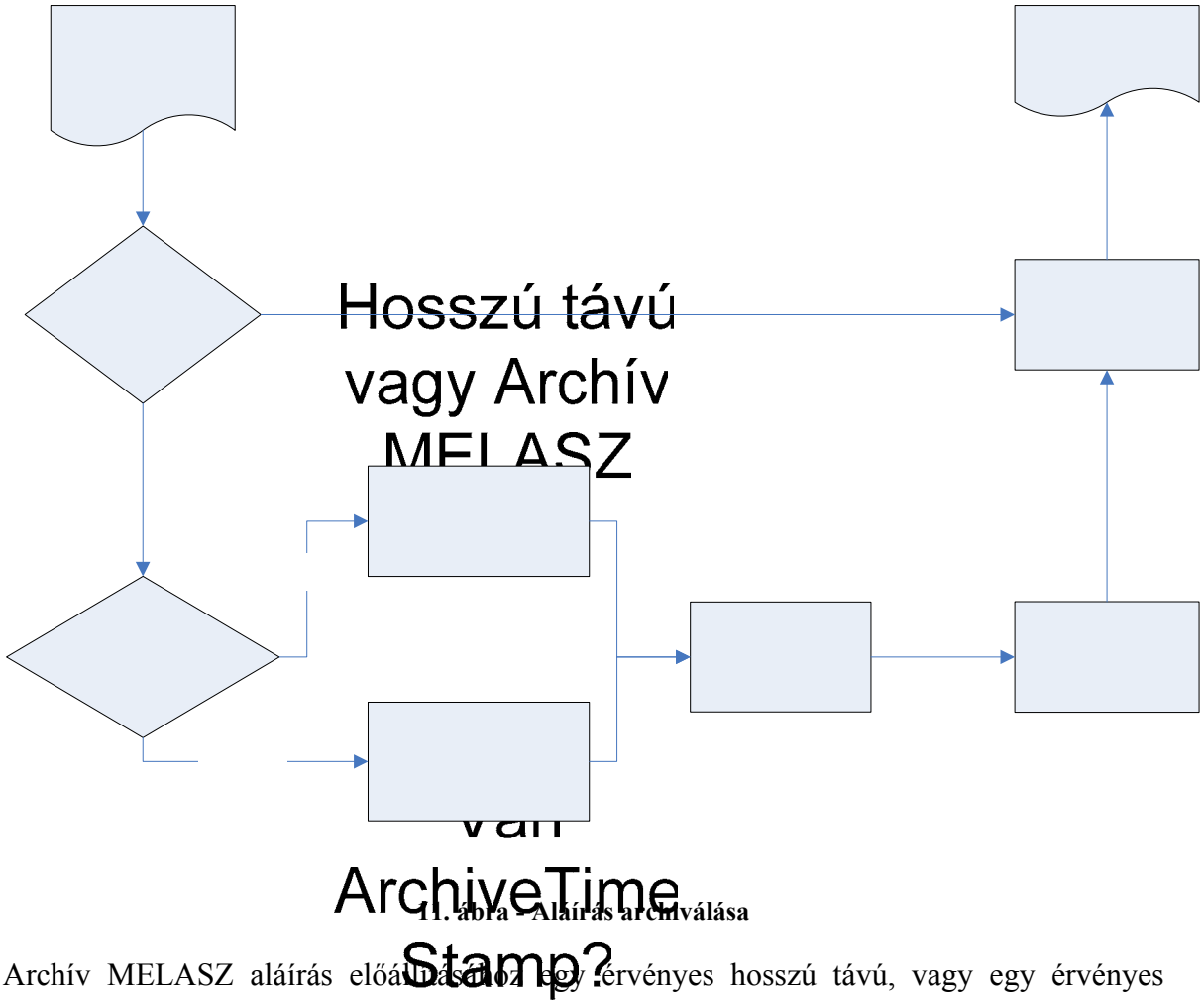
10. ábra - Aláírás kezdeti ellenőrzése

A sikeresen fogadott aláírások feldolgozásának következő lépése az ún. kezdeti ellenőrzés. Ez a folyamat a kivárási időtelte után hajtható végre sikerrel. A kivárási időszak kezdete az aláírás első hiteles időbélyegzésétől számítandó, hossza CRL visszavonási listák használata esetében 24 óra, OCSP azonnali tanúsítvány állapotot szolgáltató szerver használata esetén 30 perc.

Ha a kötelező kivárási idő még nem telt le, az aláírás kezdeti ellenőrzése befejezetlen marad.

Amennyiben a kötelező kivárási idő már eltelt, be kell szerezni a hitelességi láncban szereplő tanúsítványok állapotát igazoló adatokat. Ezután ellenőrzést kell végrehajtani az aláíráson. Ha az ellenőrzés megerősíti az aláírás érvényességét, azaz a kriptográfiai aláírás érvényes, és az aláíró tanúsítványt sem veszítette el az érvényességét a kivárási idő alatt, akkor az aláírás érvényes hosszú távú MELASZ aláírás.

6.5.3. Az aláírás archiválása



11. ábra - Aláírás archiválása

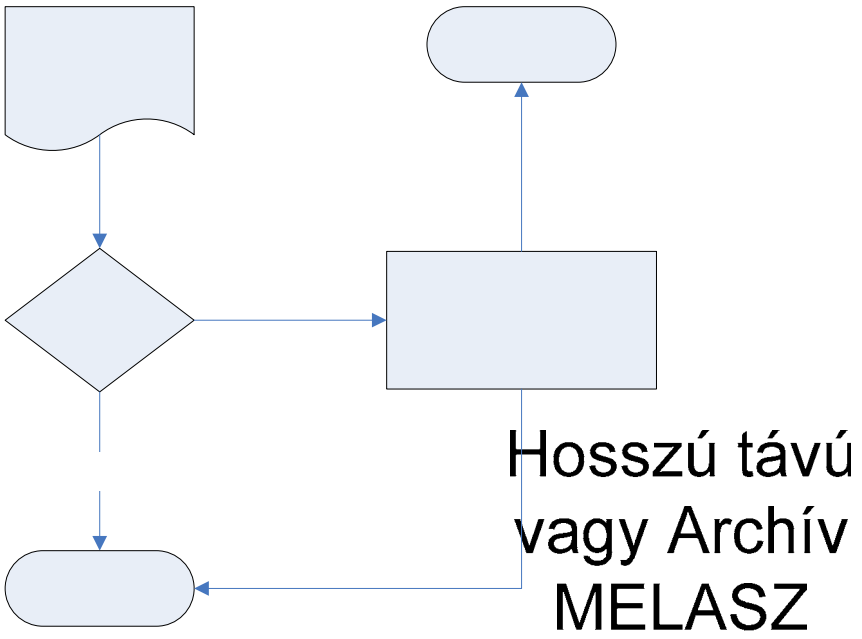
Archív MELASZ aláírás előállításához egy érvényes hosszú távú, vagy egy érvényes archív MELASZ aláírásra van szükség.

Az első eset akkor fordul elő, ha az aláírást először kerül archiválása. Ekkor ki kell egészíteni a visszavonási listájának megfelelő időbélyeggel, amely CRL lista használata esetén egy RefsOnlyTimeStamp elembe kerül, vagy azonnali tanúsítvány állapot információ (OCSP) használata esetén egy SigAndRefsTimeStamp. Ezután hozzá kell adni az aláíráshoz a tanúsítványokat a CertificateValues elemben, majd a visszavonási információkat a RevocationValues elemben. Végül egy archív időbélyeget (ArchiveTimeStamp elemet) kell készíteni a teljes aláírásra. Ez az időbélyeg abban különbözik az előzőektől, hogy az aláírás minden elemét védi ellentétben a többi időbélyeggel, amelyek nem védik az összes csatolt információt.

A második esetben az aláírás már volt archiválva, és a mostani archiválás archív felülbélyegzést jelent. Erre akkor lehet szükség, ha az előző archiváláskor használt

algoritmus vagy kulcs hitelét veszíti. Ebben az esetben csak az archív időbélyeget (ArchiveTimeStamp elemet) kell az aláírásra elhelyezni.

6.5.4. Aláírás utólagos ellenőrzése



12. ábra - Aláírás utólagos ellenőrzése

Aláírás utólagos ellenőrzésekor a már az aláíráshoz csatolt hitelesítő adatok alapján kell annak érvényességét eldönteni. Első lépésben a formátum helyességét kell megvizsgálni. Lényeges, hogy a kötelező mezők megléte és helyes tartalma. Második lépésben az aláírás kriptográfiai ellenőrzését kell elvégezni. Ha az aláírás mindkét ellenőrzésen átmegy, akkor érvényes, ellenkező esetben érvénytelen.

6.6. Magas szintű funkciók

Megfelelő

Ez a fejezet tartalmazza azon funkciók leírását, amelyekre az előző pontban megadott folyamatok épülnek.

6.6.1. Aláírás megfelelőségének vizsgálata fogadáskor

nem

Ez a vizsgálat azokat az elemeket ellenőrzi, amelyeket az aláírást létrehozó alkalmazásnak kell az aláírásban elhelyeznie. Egy helyes aláírásban helyes és végleges tartalommal kell szerepelniük a következő elemeknek, amelyeket az XMLDSIG szabvány definiál.

- SignedInfo elem: Egyrészt hivatkozásokat tartalmaz az aláírt adatok elérhetőségére vonatkozóan, másrészt leírja, hogy a hivatkozott adatok az aláírás során milyen

Érvénytelen

algoritmusokkal milyen feldolgozási lépéseken mentek keresztül. Ilyen lépések a kanonizálás, a lenyomatképzés, és az aláírás.

- SignatureValue elem: Az aláírás base64 algoritmussal kódolt értékét tartalmazza.
- KeyInfo elem: Az aláírás érvényesítéséhez szükséges tanúsítványokat, vagy azokra mutató hivatkozásokat tartalmaz. Választhatóan tanúsítvány-visszavonási információ is elhelyezhető ebben az elemben.

A következő elemeket a XAdES szabvány definiálja, és az aláírásban helyes és végleges tartalommal kitöltve kell szerepelniük.

- SigningTime elem: Az aláíró által állított aláírási időpontot tartalmazza.
- SigningCertificate elem: Az aláírást létrehozó tanúsítványra mutató hivatkozást, valamint biztonsági okokból a tanúsítvány lenyomatát tartalmazza.
- SignaturePolicyIndentifier elem: Hivatkozást tartalmaz az aláírási szabályzatra, amely azon szabályok összessége, amelyeket be kell tartani, hogy az aláírás érvényes legyen.
- DataObjectFormat elem: Az egyes aláírt adatok formátumára vonatkozó információkat tartalmazza.
- CompleteCertificateRefs elem: A hitelesítési lánc elemeire mutató hivatkozásokat tartalmazza.

Az aláírást létrehozó alkalmazás választhatóan a következő három elemet is elhelyezheti az aláírásban.

- SignatureTimeStamp elem.
- CompleteRevocationRefs elem.
- CertificateValues.

6.6.2. SignatureTimeStamp hozzáadása

Ebben a feldolgozási lépésben egy időbélyegzés szolgáltató által kiállított időbélyeget kell az aláíráson elhelyezni. Az időbélyeg elkészítéséhez képezni kell a ds:SignatureValue elemben tárolt aláírás érték lenyomatát, majd azt a szolgáltatónak el kell küldeni. A szolgáltató visszaküldi az időbélyeget, amelyet base64-gyel kódolva kell a SignatureTimeStamp elemben elhelyezni.

6.6.3. CompleteRevocationRefs hozzáadása

Ez a lépés visszavonási információkra vonatkozó hivatkozásokat helyez el az aláírásban. A hivatkozások az aláíró tanúsítványra, valamint az „őt" hitelesítő tanúsítvány láncra vonatkoznak. Egy hivatkozás mutathat időszakosan frissülő CRL listára, vagy azonnali tanúsítvány állapotot szolgáltató (OCSP) szerverre.

Egy CRL hivatkozás tartalmazza a hitelesítés szolgáltató X.500 formátumú nevét, a lista kibocsátásának időpontját, valamint a lista sorszámát, ha a hitelesítés szolgáltató ezt a mezőt kitölti.

Az OCSP hivatkozás tartalmazza a szolgáltató szerver címét, a tanúsítvány állapotáról szóló igazolás időpontját, valamint az igazolás kriptográfiai lenyomatát a lenyomatképző algoritmus azonosítójával együtt.

6.6.4. CertificateValues hozzáadása

Ebben a lépésben az aláíráshoz kell csatolni az aláíró tanúsítványt, valamint a hitelesítési láncban szereplő tanúsítványokat. Minden tanúsítvány egy EncapsulatedX509Certificate elembe kerül base64 algoritmussal kódolva. Az egyes EncapsulatedX509Certificate elemek a CertificateValues elembe kerülnek.

6.6.5. RevocationValues hozzáadása

Ez a lépés a hiteles tanúsítvány visszavonási információkat tartalmazza az aláíró tanúsítványra és a hitelesítési lánc elemeire vonatkozóan. Az egyes CRL listák vagy OCSP válaszokat base64-gyel kódolva kell az aláíráshoz mellékelni a RevocationValues elembe.

6.6.6. Aláírás érvényességének ellenőrzése

Az első lépés a tanúsítványlánc érvényességének ellenőrzése. Meg kell vizsgálni, hogy a kivárási időn belül a hitelesítési lánc valamely tanúsítványát visszavonták-e. Ha igen, akkor az aláírás érvénytelen.

A második lépés a kriptográfiai ellenőrzés. Az aláírás ds:SignatureValue elemben található értékét kell leellenőrizni, melyhez szükséges az aláírt dokumentum és a hitelesítési lánc minden tanúsítványa. A dokumentum vagy része az aláírásnak, vagy külön áll tőle, de a felhasználónak valamilyen formában mindenképp el kell juttatnia a rendszerbe. A hitelesítési lánc elemeit a CertificateValues elemnek kell tartalmaznia. Ezek

felhasználásával ellenőrizni kell a láncban szereplő aláírásokat. Ha a lánc valamelyik eleme érvénytelen, akkor az aláírás is érvénytelen.

Ha mindkettő ellenőrzés érvényes eredményt ad, akkor az aláírás érvényes.

6.6.7. RefsOnlyTimeStamp hozzáadása

Ez a lépés az aláírás ellenőrző adatokra mutató hivatkozásokat védi időbélyeg elhelyezésével. Akkor használatos, amikor a visszavonási információk CRL típusúak. Egy ilyen időbélyeget biztonságos időbélyegzés szolgáltatónak kell elkészítenie a következő elemekre.

CompleteCertificateRefs, CompleteRevocationRefs.

Az időbélyeg base64-gyel kódolt értékét a RefsOnlyTimeStamp elemben kell elhelyezni.

6.6.8. SigAndRefsTimeStamp hozzáadása

Ez a lépés is időbélyeget helyez el. Abban az esetben használatos, amikor OCSP válasz igazolja egy tanúsítvány állapotát. Ekkor az időbélyeget a következő elemekre kell kérni.

ds:Signature, SignatureTimeStamp, CompleteCertificateRefs, CompleteRevocationRefs.

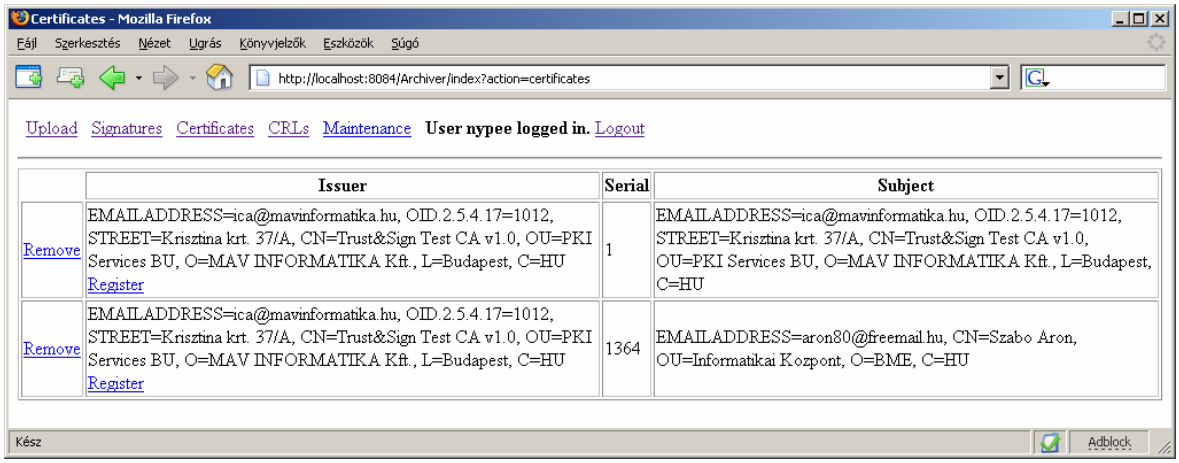
Az időbélyeg base64-gyel kódolt értékét a RefsOnlyTimeStamp elemben kell elhelyezni.

6.6.9. ArchiveTimeStamp hozzáadása

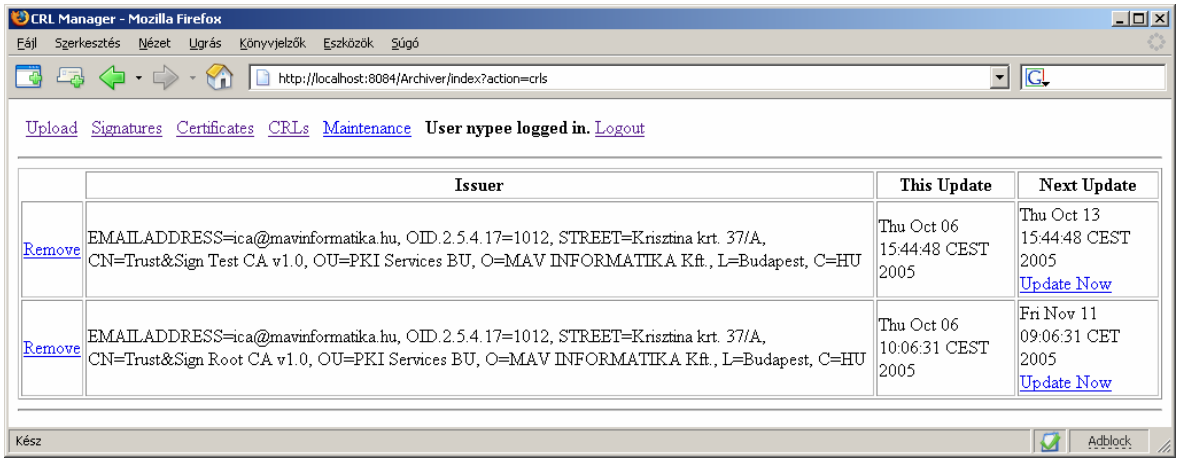
Az archív időbélyeg hozzáadásakor hiteles időbélyegzés szolgáltatóval kell az időbélyeget elkészíttetni az összes olyan elemre, amelyben tárolt információ hitelessége elveszhet a kriptográfiai algoritmusok meggyengülése miatt.

6.7. Képek a programból

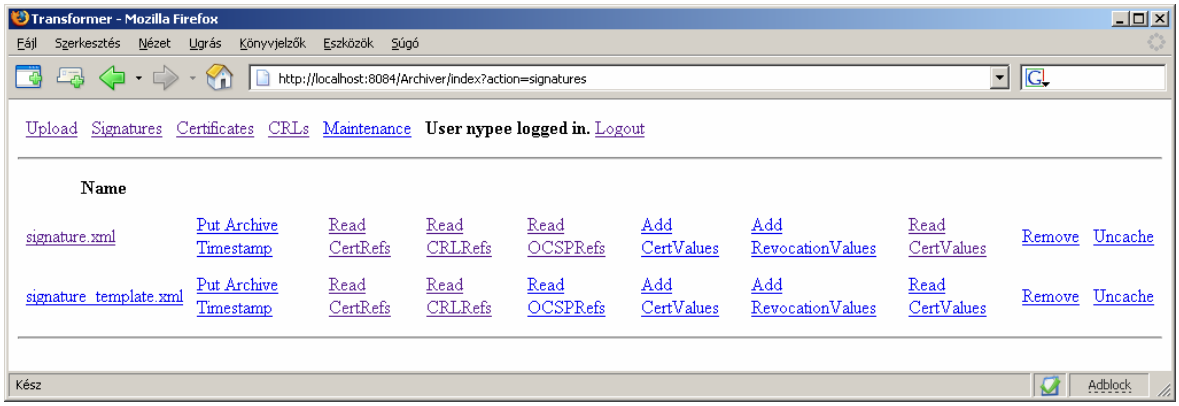
13. ábra - Bejelentkezési képernyő



14. ábra - Tanúsítványok listája



15. ábra - Visszavonási listák



16. ábra – Aláírásokon végezhető műveletek

7. Továbbfejlesztési lehetőségek

Az alkalmazás továbbfejlesztése számos irányba folytatódhat. A folyamatok működését alapos tesztelésnek kell alávetni, valamint kiterjedt együttműködési tesztelést is kell végezni a jelenleg is fejlesztés alatt álló egyéb hazai MELASZ szabványt támogató alkalmazásokkal. Fontos feladat felmérni a rendszer potenciális felhasználóinak körét, valamint az ő részükről felmerülő igényeket.

A fejlesztéseket technológiai oldalról a futtató platform lecserélésével lehet a legjobban támogatni. A mostani webkonténer környezet EJB-alapú alkalmazásszerverre cserélése sok kisebb problémát megoldana. Például az adatelérési réteg című fejezet végén kifejtett, a perzisztenciával kapcsolatos szépséghiba megoldódna a tárolandó objektumok Entity Bean-ekre cserélésével.

Egy komoly alkalmazásszerver használata esetén a rendszer további felületeit is könnyen meg lehetne valósítani. A webes kezelőfelület mellé ki lehetne fejleszteni egy XML WebService felületet is, így a rendszer bármilyen platformon futó másik alkalmazásból elérhetővé válna.

8. Köszönetnyilvánítás

Szeretnék köszönetet mondani:

Dr. Frigó József tanár úrnak, aki előadásai során megismertetett a korszerű szoftverfejlesztés filozófiájával és módszereivel.

Konzulenseimnek, amiért segítettek eligazodni a szabványok és technológiák útveszőiben, majd e hozzám illő feladatot javasolták kidolgozásra, és végül hasznos tanácsokkal segítettek a dolgozat elkészítése során.

9. Irodalom

[1] http://en.wikipedia.org/wiki/History_of_cryptography

[2] Egységes MELASZ formátum elektronikus aláírásokra; verzió: 1.0

[3] RFC 3275 XML-Signature Syntax and Processing

[4] ETSI TS 101 903 XML Advanced Electronic Signatures (XAdES)

[5] Aláírási szabályzatra vonatkozó elvárások a magyar elektronikus közigazgatásban. *forrás:* Információs Társadalom Koordinációs Tárcaközi Bizottság, Elektronikus Közigazgatás Albizottság, http://www.itktb.hu/engine.aspx?page=ias

[6] 2004. évi LV. törvény az elektronikus aláírásról szóló 2001. évi XXXV. törvény módosításáról

[7] 2001. évi XXXV. törvény az elektronikus aláírásról

[8] Lehetőség az e-számlázás http://www.montana.hu/MonWeb/Doc.asp?DocID=9&CikkID=390

[9] Jövőre indul az elektronikus számlázás – Index.hu; 2005 október 13. http://index.hu/tech/ihirek/?main:2005.10.13&239764

[10] RFC 2459 Internet X.509 Public Key Infrastructure Certificate and CRL Profile