Programozható chipkártyák kriptográfiai alkalmazása[1]

Berta István Zsolt
Mann Zoltán Ádám

A programozható smartcardokról

Az utóbbi években immár Magyarországon is megszokottá vált, hogy az emberek tárcájában különböző kártyák lapulnak. Legismertebb képviselőik a bank- és telefonkártyák, valamint az új diákigazolványok. Az "intelligens kártyák" intelligencia dolgában igen különböző képességekkel rendelkezhetnek. Alapvetően két osztályt különböztethetünk meg: csak adattárolásra alkalmas, illetve önálló számítási és feldolgozási kapacitással is rendelkező kártyákat.

A kártyák piacán egyre nagyobb számban vannak jelen a chipkártyák, melyek mikroelektronikai áramkörök felhasználásán alapulnak. A Bull 1979-ben készítette el első mikroprocesszorral is rendelkező kártyáját, melyen azonban a processzor még külön chipen helyezkedett el. Ez nem bizonyult kellően megbízható megoldásnak. A technikai fejlődés azonban csak a 80-as években tette lehetővé az összes áramkör egyetlen chipre való integrálását.

Egy smartcard programozható, ha rendelkezik processzorral és memóriaegységgel, továbbá képes felhasználói program futtatására. Ezek a kártyák sokcélú eszközök, s az, hogy végül mire is használjuk őket, esetleg csak forgalomba hozásuk után derül ki. E kártyák tulajdonképpen egyenértékűek egy lassú számítógéppel, csupán az input/output perifériákban különböznek. Egyetlen úton tud kommunikálni a külvilággal: a kártyaolvasón keresztül.

Egy hagyományos smartcard „egyszerű" adattároló egységként működik. A kártya kibocsátója definiálhat rajta különféle file-okat, s minden file-hoz megadhatja, milyen jogosultságok, jelszavak szükségesek a hozzáférésükhöz. Ezek után a kártya birtokosa, a felhasználó, magával hordozza a kártyát, s különféle kártyaolvasókba behelyezi azt. Az olvasó műveleteket végez a kártyán lévő adatokkal, olvassa őket, s – jogosultságainak függvényében – esetleg ír is.

Egy programozható kártya esetén más a helyzet. A kártyán lévő adatokon itt nem az olvasó - vagy az olvasóhoz kapcsolódó számítógép – végzi a műveleteket, hanem a kártyán futó szoftver maga. A program üzeneteket kap az olvasón keresztül, azokat dolgozza fel. A kártya kibocsátója a kártyán lévő alkalmazás megírásakor műveleteket definiál a kártyán, s az adatokat később csak ezeken a műveleteken (kapukon) keresztül lehet elérni.

Chipkártyák programozása

A smartcardok technikai okokból „kis" erőforráskészlettel rendelkeznek. Például 8 kilobyte EEPROM már igen soknak mondható. Processzoruk is viszonylag lassú, s egy bizonyos határnál gyorsabb nyilvánvalóan nem lehet, hiszen a műanyag tokban igen nehéz megoldani a kellő hűtést. Ezek után természetes lenne, hogy e kártyákra programot assembly írhatunk.

Csakhogy ez nem így van. A kártyakibocsátók rettegnek attól, hogy elkötelezik magukat egy kártyatípus mellett, s ezzel kiszolgáltatottá válnak az adott kártyagyártóval szemben. Az egymástól eltérő gyártmányú vagy esetleg csupán eltérő típusú kártyáknak gépi kódja s így assembly nyelve gyökeresen különbözhet egymástól.

---

[1] Ezen anyag publikálásra került a **Networkshop2001** konferencián, 2001. április 20.-án. Helyszín: Sopron, Nyugat-Magyarországi Egyetem.

Java Card

A Sun Microsystems a Java nyelvet azért alkotta meg, hogy legyen egy nyelv, amely platformfüggetlen, s mellyel fejlesztett alkalmazások a legkülönbözőbb architektúrájú gépeken futnak. A java appletek – mivel Internetes terjesztésre találták ki őket – kellően kicsik. Mivel egy intelligens smartcard tekinthető számítógépnek is, természetesen vetődik fel a kérdés, hogy miért ne lehetne ez a platform is Java kompatibilis.
A Java Card API is az objektum orientált szemléletet követi. Minden egyes applet, amit a kártyára letöltük, egy-egy objektum, amely önálló életet él. Rendelkezik attribútumokkal és metódusokkal. A külvilág számára látható metódusok pedig argumentumként egy APDU-t (az az adategység, amit a PC küld el a kártyának egy csomagban) kap.
Tehát egy Java Card alkalmazás kártyára telepítésének menete a következő: A programozó megírja a programot, s lefordítja byte-kódra egy tetszőleges java fordítóval. Megteszi erre a célra a teljesen ingyenes JDK is. Ezek után a programot egy – most már kártyaspecifikus – konvertáló programmal lefordítja gépi kódra. S végül egy – szintén kártyafüggő – feltöltőprogrammal elhelyezi művét a smartcardon.

WinCard - A Microsoft kártya

A Microsoft hamar előállt a PCSC specifikáció egy referencia implementációval, majd 1998 októberében hivatalosan is bejelentette a Smartcards for Windows rendszert. A tervek szerint a smartcardok szerves részét fogják képezni a jövő Windows-os rendszereinek: szerepet fognak kapni a logon folyamatban, az Outlook részeként az üzenetek hitelesítésében, valamint az elektronikus kereskedelemben.
Ez a kártya is magas szintű nyelven, Visual Basicben programozható. Ebben is, s más dolgokban is a Microsoft-féle kártya természetesen a Microsoft-világhoz illeszkedik. Itt a kártyán nem objektum van, hanem applikáció. Nem metódusai vannak, amiket meghívhatunk, hanem egyetlen belépési pontja van, amit el lehet indítani. Nagyon hasonlít egy exe file-hoz, amely paramétereket kap, s ezek függvényében találja ki, mi a feladata.
Tehát, hogyha egy komplex programot hoz létre a felhasználó, amely egymástól jelentősen eltérő funkciókat tartalmaz, akkor vagy ő hoz létre „metódusokat” az applikációban, vagy több applikációt ír, amelyek egy-egy metódusnak felelnek meg.

Biztonsági megfontolások

A lehetséges támadások alapvetően két csoportra oszthatók aszerint, hogy a támadás a kártya fizikai vagy logikai szintjét veszi célba. Ennek megfelelően beszélhetünk fizikai és logikai biztonságról.

Fizikai biztonság

Egy kártya fizikai manipulálásához rendszerint igen komoly és költséges felszerelésre van szükség (mikroszkóp, lézeres vágóberendezés, mikromanipulátor stb.), ami csak nagyon keveseknek áll rendelkezésre. Ennek ellenére fel kell arra készülni, hogy valakinek sikerül celláról cellára kiolvasni az EEPROM tartalmát. Így a legfontosabb információkat (PIN) a fizikai biztonság ellenére célszerű kódolva (pl egyirányú függvény) tárolni.
A fizikai biztonság megvalósítása a kártyagyártó feladata.

Logikai biztonság

Ha logikai biztonság elleni támadásról beszélünk, a kártyát az I/O portjain keresztül a fizikai specifikációnak megfelelő jelekkel gerjesztjük. A kártya belsejére következtethetünk a visszaadott outputok tartalmából, illetve az input és az output között eltelt időből. A kártya

belsejével nem foglalkozunk (bár belső működéséről/algoritmusairól lehet akár elég jó képünk is). Ezen biztonság megvalósítása a programozó feladata.
A kártya számára gyakran csupán kriptográfiai módszerek jelenthetnek biztonságot. Megfelelő algoritmusok alkalmazása és átgondolt kulcsgondozás mellett az ilyenfajta támadásokból a támadónak nem származhat előnye.

A mi fejlesztésünk

Bevezetés

Mi a Microsoft Smart Card for Windows nevű programozható smartcard béta verziója segítségével hoztunk létre kártyán futó alkalmazásokat. E fejezet elején leírjuk a fejlesztőrendszert, amivel dolgoztunk, majd elkezdjük az általunk készített applikációk ismertetését. Részletes méréseket végeztünk a kártya különféle részeinek sebességéről, s ezek eredményeit is e fejezetben ismertetjük. Ezután bemutatjuk egy alkalmazásunkat, a kártyán futó, kulcsot a kártyában biztonságosan tároló DES programcsomagunkat. Elmagyarázzuk működési elvét, s röviden ismertetjük PC oldali felhasználói felületét. Bemutatjuk, ezen csomag segítségével hogyan valósítottuk meg a hitelesség biztosításának öt fő pillérét.

Néhány szó a kártyáról

A kártya, amivel mi foglalkoztunk, a Microsoft Smartcard of Windows 1.0 nevet viseli, s 1999 májusában bocsátották ki béta verzióként a hozzá tartozó fejlesztőrendszerrel együtt. Néhány nappal e dolgozat beadása előtt kaptuk meg a következő, szeptemberi verziót.
A smartcard az adatbiztonság egyik kulcsa lehet a jövőben. Ez az egyik ok, amiért a Microsoft beleszállt ebbe az üzletbe is. A másik ok pedig az, hogy a Sun már benne van. Voltunk olyan szerencsések, hogy hozzájutottunk egy Microsoft-féle WinCardhoz, s erre alkalmazásokat is fejleszthettünk. Ez a kártya képességeit tekintve a jelen kor csúcsának felel meg.
A WinCardon egy 8 bites RISC AVR MCU processzor van, s rendelkezik emellett 32 kilobyte Flash Program Memoryval az applikációk számára, 32 kilobyte EEPROMmal a tárolandó adatok számára s 1 kilobyte SRAM-mal a változók, dinamikus adatok, stack, stb számára. Rendelkezik továbbá egy ATMEL kriptográfiai műveleteket támogató koprocesszorral is, mely megvalósítja a DES, triple-DES, RSA, SHA és CRC műveleteket.
A WinCard egy intelligens kártya, képes több applikáció tárolására, elkülönítésére. Futtatni viszont egyszerre csak egyet képes. Ugyanakkor használható hagyományos ISO 7816-4 kártyaként a szabványos APDU-kkal. 127 felhasználót tud elkülöníteni egymástól, s rendelkezik filerendszerrel is, amelyben minden file-hoz hozzáférési listákkal adhatjuk meg, ki min milyen műveletet végezhet. A kártya erejét két bástya képezi: az applikációk és a filerendszer.

DES csomag

Az általunk kártyába plántált DES csomag nem más, mint egy futtatható állomány, amely képes az inputját egy beépített titkos kulcs segítségével titkosítani, vagy a titkosított adatból az eredeti adatot visszaállítani. E program a következő utasításokat képes végrehajtani:

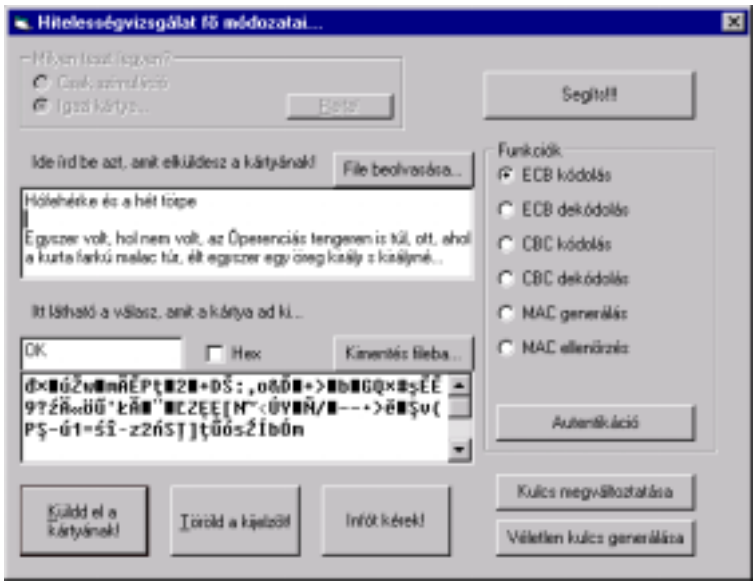
- Bemenet titkosítása: Az "e" parancs hatására a következő 8 byte-ot a kártya titkosítja, s kimenetként ezt küldi ki az outputra.
- Nyílt szöveg visszaállítása: az előző művelet inverze. A "d" parancs hatására a következő 8 byte bemenetből a kártya visszaállítja a nyílt szöveget.

- Kulcs betöltése a kártyába: ennek a műveletnek a segítségével lehet megváltoztatni a kártyában tárolt kulcsot egy a felhasználó által meghatározott értékre. Input: a "l" parancs. Output: a NO_ERROR üzenet.
- Kulcs generálása: szintén a kulcs megváltoztatására szolgál, de itt véletlenszerűen generál egy DES kulcsot. Erre a kártya kripto-koprocesszorának véletlen számgenerátor funkcióját használjuk. A DES kulcs ezután 56 db random bit lesz. Input: az "r" parancs. Output: a NO_ERROR üzenet.

DES titkosító rutin természetesen futhatna a PC-n is. Sőt, akkor sokkal gyorsabb is lehetne. Miért jó, hogy kártyán valósítottuk meg? Ahogy végignézzük a fenti négy parancsot, rögtön feltűnik, hogy "hiányzik" közülük egy: a kártyán lévő kulcs kiolvasása a kártyából.

Ez természetesen nem a véletlen műve. A PC-n bárhol helyeznénk is el a kulcsot, egy támadónak lenne esélye arra, hogy megtalálja azt. Feltételezésünk az, hogy a kártya biztosítja azt, hogy a rajta lévő adatokhoz csupán az előre meghatározott műveleteken keresztül lehet hozzáférni. Amennyiben ez tényleg így van, akkor - mivel nem definiáltunk olyan műveletet, hogy “a kulcs kiolvasása” - senki nem fér hozzá a kulcshoz.

Tehát a kulcs a kártyán biztonságban van - legalábbis a kártya előállítójának állítása szerint.



1. Ábra – A DES csomagunk PC-s felhasználói

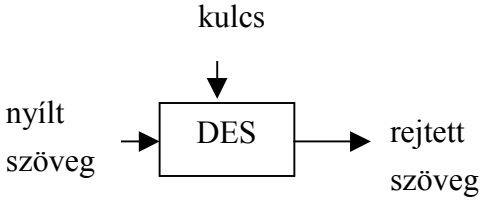
Ki elől van biztonságban? A válasz: mindenki elől. Nemcsak a támadó képtelen hozzáférni a kulcshoz, hanem a kártya gazdája is az. Hiába van a zsebében a kártya, nem képes kinyerni a titkos kulcsot belőle, csak használni tudja azt. Mivel a kártya nem hajlandó kiadni magából a kulcsot, az egyetlen esély annak megszerzésére a DES feltörése. Ezzel a rendszerrel célunk ilyen szintű biztonság létrehozása volt.

Akkor a legtitkosabb valami, ha senki nem ismeri. Az sem, aki beletöltötte a kártyába. Ennek módszere a véletlen kulcs generálása. Ha a kártya felhasználója ezt a parancsot adja ki smartcardjának, akkor ezután a következő ismeretekkel rendelkezik:

- kulcs van a kártyán
- a kulcs, ami a kártyán van, nem azonos azzal a kulccsal, ami eddig volt a kártyán
- a kulcs bitjeinek eloszlása egyenletes

Tehát arra használjuk fel a kártya processzorát, hogy saját maga állítson elő kulcsot, s azt ne adja ki senkinek. Így egyetlen entitásnak - még a kártya kibocsájtójának - sem lehetnek ismeretei a kulcsot illetően. Így e rendszer - a DES ereje és feltételezéseink alapján - biztonságosnak nevezhető.

A hitelesség megállapításának főbb módszerei közül számos megvalósítható DES csomagunk segítségével. Úgymint: hitelességvizsgálat, rejtjelezés, dinamikus jelszavak. S mivel DES esetén titkos kulcsokról van szó, felmerül a kulcsgondozás problémája is, amiben a kártya igencsak hathatós segítségünkre lehet.



2. Ábra – DES doboz

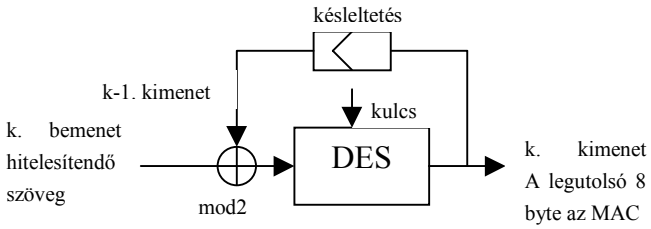
A kártyán egy közönséges DES kódolót valósítottunk meg. A bemenete a 8 byte-os nyílt szöveg, kimenete pedig a szintén 8 byte-os rejtett szöveg. Paramétere a titkos kulcs.
Ezt a rendszert egy PC-s program használja. Ez a program készít a DES elemből egy a gyakorlatban is használható eszközt. Ez a program építkezik a DES dobozból, s egy komplexebb rendszert alakít ki. A PC felelőssége az esetleg hosszú inputot 8 byte hosszú blokkokra tördelni, s őket a fekete doboznak tekintett kártyának elküldeni, majd a rejtett szöveget feldolgozni, s esetleg valamilyen formában a bemenetre visszacsatolni. (pl.: CBC, MAC)
Így egy egyszerű DES elemből egy sok célra felhasználható eszközt készítettünk, amire támaszkodva a hitelességvizsgálat főbb módozatait tekinthetjük át.

A hitelesség biztosításának lehetőségei

Hitelességvizsgálat

A hitelességvizsgálat megvalósítható úgy, hogy ez az információdarabka egy 56 bit hosszú DES kulcs. Amennyiben **A** fél ellenőrizhető hitelességű üzenetet szeretne küldeni **B** félnek, akkor megteheti a következő lépéseket:
1. A titkos kulcs segítségével legenerálja az üzenet nyolc byte hosszú MAC sűrítményét.
2. Az MAC-t odaírja az üzenet végére.
3. Elküldi az így keletkezett új üzenetet.

Ha B fél ellenőrizni szeretné az üzenet hitelességét, nem kell mást tennie, mint leválasztania az üzenet végéről a kapott MAC-t, s a titkos kulcs segítségével újra legenerálnia azt. Mivel ugyanaz a titkos kulcsa, mint A-nak, s az üzenet is ugyanaz, ugyanarra az eredményre kell jutnia, mint A-nak.
Az MAC számításának elmélete a jobb oldali ábrán található. Visszacsatolt rendszerben működik a DES, s a következő kimenetet az aktuális bemenet és az előző kimenet modulo 2 összege adja. Ha a teljes kimenetet vesszük, akkor a kapott eredmény a CBC rejtjelezés, de ha csak az utolsó nyolc byte-ot, akkor azt nevezzük MAC-nek. Az MAC egyfajta kriptográfiai ellenőrző összeg, amit a titkos DES kulcs segítségével állítottunk elő. Az MAC függ a hitelesítendő szöveg minden bitjétől. [Davis-Price1992 *(5.5.)*]



3. Ábra – MAC számítás

Jelen pillanatban a kártyán a rendszerből egyedül a DES doboz és a kulcs van. Voltak próbálkozásaink az egész rendszer kártyára helyezésére is, de jelentős sebességcsökkenés volt akkor tapasztalható. Erről bővebben a mérések fejezetben írunk.
A kártyát kezelő PC-s program felelősségéhez tartozik az input blokkokra tördelése, a kártyás alkalmazás kódoló eszközként való használata, s a az output feldolgozása. Továbbra is kihasználjuk viszont a kártya előnyét: nem ismeretes a PC-s program számára a kártyában tárolt titkos kulcs. Így a kártya szükséges az MAC generálásához vagy ellenőrzéséhez.

Rejtjelezés

DES esetében a rejtjelezést a DES függvény végzi el, melynek paramétere a titkos kulcs. Az inverz transzformáció is elvégezhető a titkos kulcs ismeretében. A DES alapú *rejtjelezésnek*

két struktúrája lehetséges: az ECB és a CBC. Mindkettő olyan DES dobozkára épít, amilyet mi a kártyán kialakítottunk.
Az ECB (electronic codebook) egyszerű blokkrejtjelező. A bemenetet nyolc byte hosszúságú blokkokra tördeljük, s ezeket egyenként rejtjelezzük a DES segítségével.
A CBC (cipher block chaining) [Davis-Price1992 *(4.2)*] egy jóval ravaszabb szerkezetű visszacsatolt rendszer. A következő bemenet mindig a következő nyílt szöveg blokk és az előző rejtett szöveg modulo 2 összegeként áll elő. Így azt érhetjük el, hogy a k. kimenet nemcsak a k. bemenettől függ, hanem függ az összes előző bemenettől is. Tehát ugyanannak a blokknak mindig más és más lesz a képe.

Hozzáférésvédelem

Hozzáférési jogosultság ellenőrzése és azonosítás történhet

- titok segítségével (PIN vagy challange and response)
- tárgy segítségével
- biometriai módszer segítségével

Mi, mivel kártyával foglalkozunk, a tárgy segítségével való hozzáférésvédelmet valósítottuk meg. Ezzel „áttoltuk a lovat a szomszéd utcába”: hogyan azonosítja magát a kártya? Biometriai jellemzői a smartcardoknak nincsenek, s tárgyakat sem birtokolnak, így marad a titok.
A megoldás egyszerű: a rendszer végül is ismeri a jelszót, akárcsak mi. Nincs szüksége rá, hogy megtudja azt. Arra van csupán szüksége, hogy megtudja, ismerjük-e a jelszót. Ezt hogyan érhetjük el? Az általunk megvalósított megoldás a „kihívás és válasz” módszer segítségével a következőképpen működik:

1. A rendszer küld a kártya számára egy kihívást. Ez nem más, mint egy *r* véletlen szám.
2. A kártya megkapja a véletlen számot. Nem csinál vele mást, mint kódolja a *k1* titkos kulcsa segítségével, s visszaküldi $E_{k1}(r)$-et a rendszernek.
3. A rendszer is kódolja *r*-et a saját *k2* titkos kulcsa segítségével, tehát kiszámítja $E_{k2}(r)$-et. Ha a két titkos kulcs megegyezik, a kártyától kapott eredmény is megegyezik majd a saját számításával. Így ha a rendszer azt kapta, amit várt, akkor nyugodtan felismerheti a kártyát.

Mi a DES csomagra támaszkodva valósítottuk meg ezt a dinamikus jelszó kezelést. Így a jelszó (a *k1* kulcs) nem más, mint egy 56 bit hosszú DES kulcs.
A PC-s program a következőket végzi el, ha az autentikáció gombra kattintunk:

1. generál egy véletlen számot
2. kódolja a számot a kártya segítségével
3. kódolja a számot a saját titkos kulcsa segítségével
4. összehasonlítja a két eredményt

A két eredmény két 64 bit hosszú szám. Annak valószínűsége, hogy két 64 bites számt véletlenül megegyezik, nagyon kicsi. Így ez az azonosítási módszer elég biztonságosnak (a DESsel azonos biztonságúnak) tekinthető.

Digitális aláírás

A digitális aláírás nyilvános kulcsú élő rendszerekben létező módszer. Eredeti elképzelésünk az volt, hogy kidolgozunk a kártyán egy nyilvános s egy titkos kulcsú kriptográfiát alkalmazó csomagot. Míg az utóbbi próbálkozásunkat hosszas küzdelem után végül siker koronázta, az előbbi sajnos kudarcba fúlt. Ennek fő oka az volt, hogy mind a kártyához tartozó fejlesztőeszköz, mind pedig annak dokumentációja még béta verzió, s a végleges változat nem készült el. A kártya már eljutott a végleges változatig, de ahhoz, hogy kódot írjuk rá, szükségünk lett volna a fejlesztőkörnyezetre, amivel kapcsolatban komoly dokumentáltsági

hibákba s hiányosságokba ütköztünk. Így a kártya RSA funkcióját nem voltunk képesek működésre bírni.
Találtunk viszont egy titkos kulcsú protokollt, amely lehetővé teszi a digitális aláírást. Ezt meg lehet valósítani DES segítségével. [Schneier1996 *(2.6.)*]
A módszer lényege a következő:
A, **B**, **C**, **D**, és a többiek személyek, akik hitelesen akarnak kommunikálni egymással. Mindannyian rendelkeznek saját titkos kulccsal. **T** egy kitüntetett személy, akiben a többiek mind megbíznak, s megosztották vele titkos kulcsukat.
A hiteles üzenetet kíván küldeni **B**-nek. A következő lépések történnek ekkor:
1. **A** titkosítja az üzenetét a kulcsával, s elküldi **T**-nek. Beleírja az üzenetébe azt is, hogy ő **B**-nek kíván üzenni.
2. **T** ismeri **A** kulcsát, kibontja az üzenetet. Elolvassa azt, majd hozzáteszi azt, hogy tanúsítja, hogy az üzenet **A**-tól jött. (ezt onnan tudja, hogy **A** kulcsával kódolták)
3. **T** titkosítja az üzenetet **B** kulcsával, s elküldi **B**-nek.
4. **B** tudja, az üzenet **T**-től jött, mert **B** kulcsát csak **T** ismeri rajta kívül. Az üzenetben pedig benne van, hogy **A** küldte, s ezt **T** írta, **T** pedig soha sem hazudik.

Ez a módszer megvalósítható PC-s program csomagunkkal, de tulajdonképpen az MAC generálás funkciót használja (vagy a CBC/ECB rejtjelezések egyikét), tehát nem különbözik a többitől.

Kulcsgondozás

A kulcsgondozás jelenti kulcsok
- generálását
- tárolását
- továbbítását

RSA kulcs előállítása a kártyán a sebességviszonyok miatt reménytelen vállalkozás lenne. DES kulcsot viszont minden gond nélkül generálhatunk, itt nincsen szükség különösebb biztonsági megfontolásokra a kulcs természetét illetően. Generálunk 8 db random byte-ot, s ez lesz a mi DES kulcsunk. Azzal, hogy véletlen biteket választunk, nem követünk el nagy hibát, hiszen a DES 2^{56} db kulcsa között összesen 16 a gyenge kulcs. [Schneier1996 *(8.1)*]
A fő gond itt nem a véletlen nyolc byte-tal van, hanem azzal, hogy valóban véletlennek tekinthetjük-e azt, amit a kártya kiad. Erről nincsenek információink. Azt tudjuk csak kijelenteni, hogy a kártya dokumentációja nem mond lehetőséget arra, hogy a kártyából a véletlen szám generátor aktuális állását kinyerjük. Sőt, jól megtervezett kártya esetén a támadónak véletlen szám generálásához (s ennek megismeréséhez) sincsen joga.
A kulcsok tárolására adhatunk egy rövid választ: Ez a smartcard technológia lényege. A smartcardokat tulajdonképpen kulcsok tárolására és hordozására találták ki. Ők valójában nem mások, mint ezt a célt szolgáló biztonságos eszközök. A dolog lényege annak biztosítása, amit a biztonságról szóló fejezetben leírtunk.
A következő pontoknak kell teljesülnie ahhoz, hogy a kulcs ne kerülhessen illetéktelen kezekbe:
- Ne ismerje senki illetéktelen a kulcsot, már korábbról, mielőtt az a kártyába bekerült. – Ez szervezési kérdés. (Véletlen kulcs generálása esetén azt csak a kártya ismeri.)
- Ne lehessen a kulcsot fizikai eszközökkel megszerezni! – Ez a kártyagyártó felelőssége.
- Ne lehessen a kulcsot a kártya filerendszeréhez való közvetlen hozzáféréssel kiolvasni! – Az alkalmazás tervezőjének és a gyártónak a közös felelőssége...
- Ne lehessen a kulcsot az alkalmazástól megkapni! – Ez az alkalmazás tervezőjének a feladata. Ha nem írjuk ki a kulcsot soha a kimenetre, az nem fog magától kikerülni oda.

A kulcs továbbítása történhet:

- Smartcard segítségével, tehát a kulcsot hordozó személy a zsebében szállítja a kártyát.
- A PC irányába a kártyából: ilyen a mi programunkban nem történhet. A kulcs csak a PC-ből megy a kártya felé, de ez az előző pont anyaga.
- Hálózaton keresztül: a mi programunk ilyet sem tesz. Ezen egyébként segíteni lehetne megfelelő kódolás használatával.

Összefoglalás, munkánk értékelése

A mi eredményeink

Célunk a WinCard s a hozzá tartozó fejlesztőrendszer lehetőségeinek áttekintése, s a hitelesség biztosítás megvalósításának smartcardos lehetőségeinek felmérése volt. Ezek megvalósításához kifejlesztettünk egy DES kódolást-dekódolást megvalósító, s kulcsokat gondozó kártya-PC programcsomagot. Ennek segítségével próbáltuk megvalósítani a hitelesség biztosításának öt fő pillérét, melyek:

- hitelességvizsgálat
- rejtjelezés
- hozzáférésvédelem - dinamikus jelszavak
- digitális aláírás
- kulcsgondozás

Ezek közül hármat közvetlenül megvalósítottunk. Kártyánk képes rejtjelezni, s a programcsomag képes hitelességet ellenőrizni, illetve a PC dinamikus jelszó segítségével azonosítani a kártyát.
A kulcsgondozás viszont nem egy aktív funkció, amit egy rendszer képes „megtenni”. Ez egy problémakör, egy gondolkodásmód, melynek figyelembe vétele létfontosságú. Kártyánk védi, s ki nem adja a kulcsot senkinek. A kulcsot csak használni (esetleg generáli) lehet, megismerni nem.
Probléma egyedül a digitális aláírás megvalósítása körül történt, ugyanis a fejlesztőrendszer dokumentáltsági hibái folytán nem tudtuk beüzemelni a kártya RSA funkcióit. Klasszikus digitális aláírás megvalósításához viszont nyilvános kulcsú titkosírásra van szükség.
Leírtunk viszont egy három résztvevős protokollt, mely digitális aláírást valósít meg titkos kulcsú rendszerben, s az általunk készített programcsomagot ennek bármely szereplője használhatja.

Az öt pillér elvi megvalósíthatósága egy mai smartcardon

A smartcardok biztonságrechnikai alkalmazások terén magasan felülmúlják a PC-ket. Fő hártányuk a sebességben rejlik. Elég hamar nyilvánvalóvá vált számunkra, hogy kriptográfiai műveletek terén csakis a kártya processzorában implementált műveletekre támaszkodhatunk. Elvileg lehetséges lenne őket például függvényként megvalósítani, de ezek sebessége kritikán aluli lenne.
A másik szűk keresztmetszet a PC-kártya kommunikáció. Ez ugyanis lassú. Ez a továbbiakban is soros lesz, ugyanis a kártyán lévő kontaktusok működését leíró szabványok csak ilyet tesznek lehetővé. Így komoly sebességnövekedés itt nem jöhet szóba.
A rejtjelező funkciót egy kártya képes ellátni, természetesen a fenti korlátozásokkal. Sajnos az egész nyílt szöveget el kell juttatnunk a kártyára, s ez így lassú. Mi 100 byte-os nagyságrendben mozgó inputot még ésszerű idő alatt tudtunk titkosítani. Hitelességvizsgálat és digitális aláírás esetén jobb a helyzet. Itt megtehetjük, hogy először a PC-n tömörítvényt képzünk a nyílt szövegből, s utána azt írjuk alá a kártyával. A hozzáférésvédelem és a

kulcsgondozás alig ütköznek problémába. Mindkettő olyan eljárás, amelyre már régen alkalmaznak kártyákat, igaz, programozható képességüket még nem használják ki. Úgy is lehet mondani, hogy a kártyákat eredetileg erre a két módszerre találták ki. Mindkét esetben kicsi az adatforgalom, s kevés számításra van szükség, így az új ötletek is viszonylag könnyen implementálhatók.

Smartcardok: jelen és jövő

Igaz, hogy a smartcardok gyenge pontjai, a kommunikációs sebesség, a tárkapacitás és a számítási sebesség sokat fejlődtek, de valószínűleg mindig is alul fogják múlni a PC-k teljesítményét. Bizonyos dolgok elvi korlátokba is ütköznek.
A kommunikáció a kártya és a PC között sorosan valósul meg (ez a kontaktusok specifikációjából következik). Így ez eleve nem lehet különösen gyors. Az adat tárolása jelenleg nem illékony memóriával történik, (hogy ne legyen benne mechanika), ez pedig igen drága. A processzor esetében pedig komoly hőelvezetési problémák merülnek fel az órajelfrekvencia növelése esetén. Ráadásul minden alkatrészt egyetlen mikrochipen kell megvalósítani, ami növeli a nehézségeket. Nem tudjuk, meddig fognak nőni e paraméterek, de a PC-ket nem érhetik utol.
Az intelligens smartcardok igen széles lehetőségeket nyitnak meg elektronikus biztonságtechnikai alkalmazások előtt, s ráadásul ezeket a lehetőségeket viszonylag alacsony áron jelentik. Nem szükséges ugyanis külön hardvert, külön mikrochipet gyártani minden egyes alkalmazáshoz, hanem általános célú, programozható eszközökön lehetséges a fejlesztés, amelyeket nagy példányszámban alacsony áron lehet előállítani. Egy kártya felprogramozásához pedig egy olvasón, egy PC-n s szoftveren kívül semmi sem szükséges. Így nem csak nagy multik, hanem kisebb cégek is képesek lehetnek saját smartcard alapú szoftverek kidolgozására, fejlesztésére.

Irodalomjegyzék

Általános kriptográfiai munkák:
D.W. Davies – W. L. Price: Security for Computer Networks. John Wiley & Sons, 1992.
Bruce Schneier: Applied Cryptography. John Wiley & Sons, 1996.
Gustavus J. Simmons (Szerk.): Contemporary Cryptology. IEEE Press, 1992.
Györfi-Vajda: A hibajavító kódolás és a nyilványos kulcsú titkosítás elemei. Budapest, 1991.

Smartcardokkal kapcsolatos munkák:
W. Rankl – W. Effing: Smart Card Handbook. John Wiley & Sons, 1997.
J. L. Zoreda – J. M. Oton: Smart Cards. Artech House, 1994.

A Microsoft fejlesztőkörnyezettel kapcsolatos információk forrása:
Windows Smart Card Development Kit Help
http://www.microsoft.com/security/tech/smartcards

Egyéb, smartcardokkal kapcsolatos társaságok:
Bull: http://www.cp8.bull.net
Java Card: http://java.sun.com/products/javacard/htmldoc
M.U.S.C.L.E.: http://www.linuxnet.com
PC/SC Workgroup: http://www.smartcardsys.com
OpenCard: http://www.opencard.org