Eötvös Loránd Tudományegyetem

Természettudományi Kar

Kovacsics Ramóna

Kihívás - Válasz Protokoll

BSc Szakdolgozat

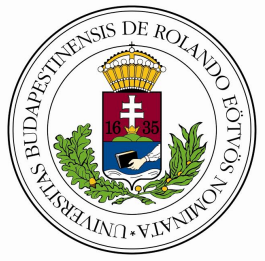
Témavezetők:

Dr. Villányi Viktória Ildikó

és

Dr. Sziklai Péter

Számítógéptudományi Tanszék



Budapest, 2017

Köszönetnyilvánítás

Elsősorban szeretném megköszönni a két konzulensemnek: ***Dr. Villányi Viktóriának***, aki hétről hétre hasznos tanácsokkal, ötletekkel látott el, és ***Dr. Sziklai Péternek*** aki mindig szakított rám időt, ha segítségért fordultam hozzá.

Szeretném megköszönni ***Édesanyámnak***, aki arra biztatott, hogy ne adjam fel, és mindig felhívhattam, ha szükségem volt rá.

Köszönöm ***Édesapámnak***, aki minden egyes alkalommal támogatott engemet, és mindig elmondta nekem, hogy mennyire büszke rám.

Köszönettel tartozom a ***Családom*** többi tagjának, akik hitték, hogy mindenre képes vagyok, és bíztak bennem.

Végső de nem utolsó sorban, köszönöm a ***Barátaimnak***, hogy mindig ott voltak mellettem, ha szükségem volt rájuk.

Tartalomjegyzék

Bevezetés

A Partner-hitelesítés a mindennapjaink része. Az emberek egyre szkeptikusabbak a másikak iránt, a bizalmatlanságukat pedig fokozzák, hogy egyre többet lehet arról hallani, hogy a csalók és szélhámosok hogyan úsznak meg egy-egy esetet, illetve hogyan sikerült átverniük valamelyik embertársukat. Annak érdekében, hogy a csalást minimálisra csökkentsük, különböző azonosítási, hitelesítési protokollokat hoztak létre.

A szakdolgozatom az alapfogalmak definiálása mellett, a különböző hitelesítési protokollokat mutatja be, a Kihívás-válasz protokollra fókuszálva. Az alapvető kérdés, amit vizsgálok, hogy az egyes módszerek mennyire biztonságosak, hogyan zajlik a hitelesítésük, mik szükségesek az algoritmus felépítéséhez.

Minden egyes protokoll mástól lesz egyedi. Valamelyik titkos kulcsot használ, míg a másik hash függvénnyel dolgozik, és van olyan is, ami mindkettőt felhasználja a hitelesítés során. A különböző módszereket más-más területeken alkalmazzák. Például a digitális aláírás az ügyvédek körében, míg az egyszer-használatos jelszavak az internet felületén az elterjedtebbek.

1. fejezet

Partner−hitelesítés nem interaktív módon

A partner−hitelesítések célja, hogy a protokoll résztvevői egymás identitásáról megbizonyosodjanak.

A mai világban egyre nehezebben lehetséges meggyőzni egy másik embert a személyazonosságunkról. A probléma nagyobb kihívást jelent, amikor a két fél (legyen Alice (A) és Bob (B)) ellenfelek, és nem szeretnénk lehetővé tenni, hogy Bob be tudja mutatni magát Alice-ként azután, hogy Alice igazolta saját identitásást és ennek Bob tanúja volt. Azaz Alicenak folyamatosan figyelnie kell arra, hogy ne adjon ki annyi információt magáról, hogy Bob sikeresen meg tudja majd személyesíteni.

Például: Egy átlag embernek van útlevele (amit elég gyakran hamisítanak), bankkártyája (aminek a számát le lehet másolni, vagy pin kódját fel lehet törni), számítógépes/felhasználói jelszava (amiket Hackerek tudnak megtámadni, és sikeresen feltörni), valamint a katonaságnál lehetnek olyan katonai parancsok, esetleg katonai biztonsági rendszerek amik ellenséges kezekbe is kerülhetnek.

Egy másik példa: Tegyük fel, hogy A és B a két fél akik között folyik az azonosítás és B próbálja azonosítani A−t. Legyen T a támadó. A támadó célja, hogy az identitás bizonyítása végére B azt higgye T-ről hogy A. Amennyiben ez sikerül neki, úgy az azonosítás nem volt biztonságos, a támadás pedig sikeres volt.

Alapvetően 3 különböző csoportba sorolhatjuk az azonosítási módszereket:

1. biometriai: Ezeket a módszereket elsősorban személyek identitásának bizonyítására

használjuk. A bizonyítás alapját valamilyen személyen biológiai jellemző (pl. hang, ujjlenyomat, írisz mintázat stb.) képezi.

2. hardver alapú: Ezen módszereknél a bizonyítás alapja valamilyen hardver token (badge, smart kártya stb.).

3. algoritmusos: A bizonyítás alapja valamilyen számítás elvégzésének képessége.

A szakdolgozatomban én csak az algoritmusos módszerekkel foglalkozom, melyekből a következő 3 típust mutatom be:

1. Hitelesítő módszer: Alice képes meggyőzni Bobot, hogy ő valójában Alice, de senki más nem tudja bizonyítani Bobnak hogy ő Alice.

2. Személyazonosság megállapítása : Alice képes meggyőzni Bobot, hogy ő valójában Alice, de Bob senki másnak nem tudja bebizonyítani, hogy ő lenne Alice.

3. Aláíró módszer: Alice képes meggyőzni Bobot, hogy ő valójában Alice, de Bob még saját magának sem tudja bebizonyítani, hogy ő lenne Alice.

1.1. Hitelesítő módszer

A hitelesítő módszer csak külső támadás ellen hasznos, vagyis amikor A és B társak. A hitelesítő és az aláíró séma között a különbség nagyon finom. Míg a hitelesítőnél: B tud létrehozni egy hihető másolatot a feltételezett beszélgetésről, ha jól választja meg a kérdéseket és a válaszokat a párbeszéd alatt, addig az aláírónál: csak a ténylegesen megtörtént beszélgetésről lehet hihető másolatot készíteni. A legtöbb kereskedelmi és katonai felhasználónál a fő probléma, hogy a hamisítások utáni nyomozást valós időben is meg lehessen tenni, és ha valaki hamisítással próbálkozik, akkor azt a rendszer visszautasítsa.

Általában a leggyakrabban használt azonosítási technika a jelszó alapú azonosítás. Ennek oka főleg az, hogy felhasználóbarát, azaz nem igényel extra hardvert, és könnyen kezelhető. A felhasználót a jelszó hitelesíti. A jelszó lényegében egy titok, amit a felhasználónak meg kell adnia az azonosítása során.

Egy alapvető probléma a jelszavakkal az, hogy a túlhasználják, vagyis nem változtatják őket gyakran a felhasználók. Pontosan emiatt, a jelszavas rendszerek biztonsági problémáit főleg a statikusságuk okozzák. Amikor egy jelszót feltörnek, a hitelesítő rendszer nem tudja eldönteni, hogy az igazi felhasználó lép-e be a jelszóval, vagy egy csaló jutott hozzá a jelszóhoz. Ha egy jelszó hosszú időn keresztül nem változik, akkor aki fel akarja törni visszajátszásos támadással fel tudja törni, azaz többszöri próbálkozással. Ennek kieszközölése a "véletlen" biztosítása, vagyis minden azonosítás során kell használni valamilyen véletlen értéket, hogy megszüntessük az állandóságot.

A visszajátszásos támadás mellett, gyakori még a szótár alapú, "ami a rendszer jelszófájljában tárolt információkat és a támadó által generált vagy szintén a rendszerben található szótárfájlokat, a felhasználó jelszavainak off-line megfejtésére" [1]. Ezekből a forrásokból összeállítanak egy listát, melynek a szavait különböző transzformációkkal átalakítják: pl. kis-nagybetű, számok írása a szavak végére. Mivel ezek off-line módon kerülnek kivitelezésre, a támadónak korlátlan ideje van a feltöréséhez.

Megoldás: Olyan jelszót kell választani ami nem egy értelmes szó és nem valamilyen transzformációval jön létre (pl. nem szimmetrikus, nincs benne ismétlődés, nem a nevünk betűiből áll össze). Egy jó jelszó lehet például: Kedvenc könyvünk, első pár sorának kezdőbetűi, plusz hozzáadva egy-két numerikus karaktert.

Másik megoldás a *salting*. Ez azt jelenti, hogy a jelszavakat egy adott hosszúságú bitsorozattal egészítjük ki, és ezután alkalmazzuk az egyirányú leképezést, ami azt jelenti, hogy a gazdagép nem magát a jelszót tárolja a jelszófájlban, hanem azoknak csak egy lenyomatát a felhasználói azonosítókkal együtt. Így ha a salt mérete r akkor a támadónak 2^r különböző variációt kell kipróbálnia, hogy feltörje a jelszót, ami r méretétől függően elég sok lehetőséget adhat.

[1] Buttyán Levente, Vajda István - Kriptográfia és alkalmazásai

1.2. Egyszer-használatos jelszavak:

Def: Az egyszer-használatos jelszó olyan jelszó, ami abban a pillanatban érvénytelenítődik, amint azt használják.

Ahogy a nevük is mutatja, egyszer lehet őket használni, utána automatikusan törlődnek. Ez egy jó stratégia arra, hogy szembeszálljanak a statikus jelszavak könnyű feltörhetőségével, hiszen sokkal biztonságosabbak lesznek. Ha a támadó sikeresen megfejti, hogy mi volt a korábbi jelszó, hiába próbálja használni, a rendszer már nem fogja neki elfogadni. Amennyiben a támadó hozzáfér az összes korábbi jelszóhoz, akkor sem tudja feltörni a legújabbat, hiszen a korábbi jelszavakból nem lehet kiszámítani mi lesz a következő. Ilyennel találkozhatunk például a bankok netbankár rendszerében.

1.2.1. S/KEY

S/KEY jelszó generálás[2]:

1. Az első lépés egy titkos kulcs készítése, legyen ez a W.

   A titkot létrehozhatja a felhasználó, de akár a számítógép is generálhatja. Bármelyik módon, ha a titkot felfedésre kerül, akkor az S/KEY biztonsága megtört. A kriptográfiai hash függvény legyen H. (A hash függvény egy $H : \{0,1\}^* \to \{0,1\}^n$ függvény, amely egy tetszőleges hosszúságú bitsorozatot egy fix hosszúságú bitsorozatba képez.)

   Ezt a függvényt n-szer alkalmazzuk W-re, ezáltal az eredmény egy hash-lánca lesz az egyszer-használatos jelszavaknak. A jelszó az eredménye a kriptográfiai hash függvény alkalmazásainak:

   $$H_1 := H(W),\ H_2 := H(H_1(W)),\ \ldots\ ,\ H_n := H(H_{n-1}(W)).$$

   Az kezdeti titkot (W) elhagyjuk.

[2]The S/KEY One-Time Password System (RFC 1760)

A felhasználó megkapja az n db jelszót, az alábbi sorrendben, amit a számítógép küldött neki.

$$H_n\ ,\ H_{n-1},\ ...\ ,\ H_2,\ H_1.$$

A jelszavak ($H_1,\ H_2,\ ...\ ,H_{n-1}$) törlődnek a számítógépből. Csak az utolsó jelszót (H_n), ami a felhasználó listájának az élén áll, csak azt tárolja el a szerver.

2. Hitelesítés

   A számítógép végzi a hitelesítést.

   A jelszó generálás után, a felhasználónak van egy papírlapja rajta az n db jelszóval.

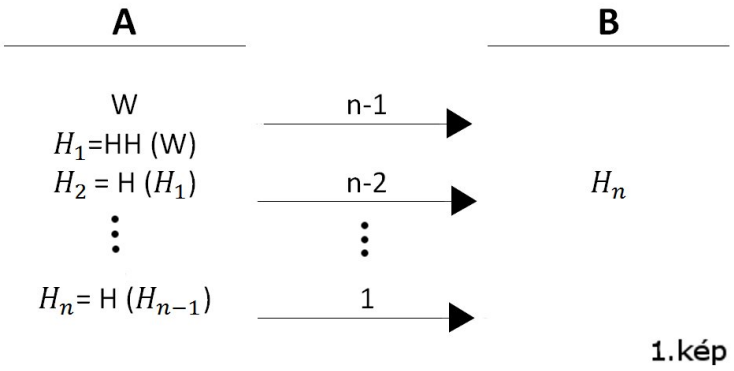
   Ideálisabb, bár kevésbé gyakori, amikor a felhasználónál van egy kicsi, hordozható, biztonságos, nem hálózati számítástechnikai eszköz (program), ami képes visszagenerálni az összes szükséges jelszót, megadva a titkos W-t, és a hash függvény szükséges iterációinak a számát.

   Más esetben, az első jelszó lesz az a jelszó amit a rendszer eltárolt korábban. Az első jelszó nem lesz használva hitelesítés céljából (a felhasználó leírhatja magának). A másodikat használjuk helyette:

   A felhasználó megosztja a szerverrel a 2. jelszót a listáról (ami a H_{n-1}). A szerver megpróbálja kiszámolni a $H(p)$-t, ahol p a jelszót helyettesíti. Ha a $H(p)$ eredménye az első jelszó, (amit a gép eltárolt) akkor a hitelesítés sikeres volt. A rendszer ekkor eltárolja p-t mint érvényes hivatkozás.

   A következő hitelesítésnél, a felhasználó megadja az i. jelszót. (Az utolsó jelszó a listáról az n-dik, ezt generálta ki legelőször a gép : $H_1 = H(W)$, ahol W volt a kezdeti titok). A számítógép kiszámolja H(i. jelszó)-t, és összehasonlítja az $i-1$. jelszóval, amit korábban elraktározott korábbi hivatkozásnak. (1.kép)

3. Biztonság



1.kép

Az S/KEY biztonsága a kriptográfiai hash függvény bonyolultságán alapszik.

Feltételezzük, hogy a támadó megszerezte azt a jelszót, amit mi használtunk egy sikeres hitelesítés során. Tegyük fel hogy ez az i. jelszó volt. Ez a jelszó már használhatatlan a következő hitelesítés során, mert minden jelszót csak egyszer lehet felhasználni.

Az lenne érdekes, ha a támadó az $i-1$. jelszót találná ki, hiszen ez az az egy jelszó, amit használni fogunk a következő hitelesítés során.

Ez az opció igényelné a hash függvény invertálását, vagyis azt hogy kiszámoljuk az $i-1$-dik jelszót, felhasználva az i-et (H($i-1$. jelszó) = i. jelszó), amit még rendkívül nehéz jelenleg megtenni a kriptográfiai hash függvénnyel.

Mindamellett az S/KEY sebezhető az embernek a támadás közepén ha használta őmaga. Szintén sebezhető bizonyos feltételekkel, például ahol a támadó rendszere a hálózaton keresztül megtudja az első $N-1$ karakterét a jelszónak (ahol $N :=$ a jelszó teljes hosszával), ott megalapozza a saját TCP szekcióját a szerveren, és véletlenszerűen próbál ki minden érvényes karaktert behelyezni az N. pozícióba addig amíg megtalálja azt az egyet ami helyes. (Ez a fajta támadás megelőzhető, és elkerülhető.)

(A TCP valójában két számítógépen futó program között, egy adatfolyam megbízható, sorrendhelyes átvitelét biztosítja. Az internet legfontosabb szolgáltatásainak nagy része TCP-n keresztül érhető el.)

Vegyünk egy konkrét példát:

Az egyik repülőtéren az S/KEY protokollt használják, hogy kizárják a nem ott dolgozókat.

A számítógép (S) létrehozza a W titkos kulcsot. A hash függvény legyen H. S kiszámolja a H segítségével az első n db jelszót, amit elküld majd az alkalmazottaknak (legyen Alice (A) az egyik dolgozó.) A jelszavak törlődnek, kivéve az n. . Minden nap új W és H lesz generálva, minden dolgozónak más-más.

Legyen Bob (B) a támadó, aki szeretné feltörni a rendszert és bejutni a reptérre, hogy adatokat lopjon. Ehhez meg kell tudnia a jelszót, amit meg kell adni a számítógépnek a hitelesítés során.

Alice megy dolgozni és ehhez használnia kell a titkos jelszót. Bob ezt megpróbálja megszerezni. Azért, hogy B véletlenül se tudja kitalálni a következő kódot A először az $n-1$. jelszót küldi el a számítógépnek. Mivel S ismeri a hash függvényt, alkalmazza H_{n-1}-re és ha az eredmény megegyezik H_n-nel akkor elfogadja.

Ezután Alice elküldi az $n-2$. jelszót, majd az $n-3$.-at és így tovább míg végül az 1.-t.

Ha Bob tudomást is szerez a H-ról, akkor sem fogja tudni kiszámolni, hogy mi lesz a következő jelszó, mivel a hash függvény invertálása nehéz feladat. Viszont, ha A nem figyel eléggé, és előbb küldi az $n-2$. jelszót mint az $n-1$.-et akkor ha B megszerezte a H-t, ki tudja számolni a következő kódot.

1.3. Aláíró módszer

Az életünk során, szinte minden fontos iratot aláírásunkkal kell hitelesíteni. Például: banki ügyek intézésénél, a postán egy csomag átvételénél, szerződéskötésnél, de akár amikor a

futár hoz nekünk egy ajánlott levelet. Alá kell írni, hogy ezzel bizonyítsuk, hogy mi vettük át, vagy hogy mi szeretnénk a szerződésmódosítást.

Személyigazolványon és diákigazolványon is rajta van az szignatúránk. Ennek elsődleges indoka az, hogy amikor egy új szerződést vagy hivatalos papírt írunk alá, akinek aláírjuk nem ismeri a saját kézjegyünket, így nem tudja, hogy hamisítjuk-e vagy valóban a miénk-e. Ilyenkor szokták kérni a személyigazolvány bemutatását amin az eredeti kézírásunk van rajta.

Az aláírással a probléma az, hogy könnyen lehet hamisítani. Mivel egy aláírás sosem néz ki teljesen ugyanúgy, még magától attól a személytől sem akié, emiatt elég csak egy nagyon hasonló aláírást odaírni a dokumentumra és máris hitelesítve van. Ha kapunk egy levelet, meg kell bizonyosodnunk arról, hogy valóban az a személy küldte-e aki feladta, továbbá azt is, hogy ha valaki más kezébe került a levél, az nem módosított-e rajta. Az aláírás és az üzenet nem tartozik össze. Ha valaki megszerezte az elküldött levelünket, akkor az üzenetet és az aláírást szétvághatja, majd az utóbbit felhasználhatja egy új, általa írt levélben. Ezzel ellentétben a digitális aláírásnál, a dokumentum és az aláírás összetartozik, így ha a támadó az előbbi módszert alkalmazza, az aláírás és az új dokumentum között nem lesz meg az a megfeleltetés mint az eredetinél.

Annak érdekében, hogy biztonságosabbá tegyék az aláírásos hitelesítést létrehozták a *digitális aláírás*t. A digitális aláírás egy nyilvános kulcsú rendszer, amivel a hagyományos aláírást tudjuk helyettesíteni. Egy jó digitális aláírás, mindent tud amit a hagyományos, sőt sokkal többet is: A digitális aláírás csak logikailag kapcsolódik az aláírt dokumentumhoz, így ha valaki megpróbálja lemásolni, könnyen lebukhat. Ebből kifolyólag, mivel nem lehet hamisítani, ha aláírtunk valamit, később nem tudjuk letagadni sem.

Ezt a fajta módszert elsősorban főleg ügyvédek szokták alkalmazni. A digitális aláírásokat a programkódok aláírására is gyakran használják.

Vegyünk egy példát:

Aladár megírta az új saját programját, amit végül aláír. Ha ez a programkód eljut Évához, Évának a digitális aláírás egyértelműen igazolja, hogy az adott szoftvert, prog-

ramot Aladár írta, valamint az is garantálja, hogy a programkód a kibocsátás óta nem változott.

Abban az esetben ha a Éva bármely, a szoftver által generált problémát azonosít be, akkor feltételezhető, hogy az adott szoftver kódját senki más nem változtatta meg, azaz Aladár a felelős, a program hibájáért.

A digitális aláírás sémája 3 algoritmusra épül:

1. Kulcsgeneráló algoritmus: A feladó generál egy saját titkos kulcsot a lehetséges titkos kulcsok közül. Ezzel az algoritmussal megkapja a helyes nyilvános kulcsot is.

$$(P_k, S_k) \leftarrow (I^n)$$

2. Aláírást generáló algoritmus: Az üzenet és a titkos kulcs együtt megalkotja az aláírás.

$$Sign_{S_k}(m) \rightarrow \sigma$$

3. Aláírást ellenőrző algoritmus: Az üzenet a nyilvános kulcs és a megkapott kódolt aláírás segítségével vissza tudja fejteni.

$$Verify_{P_k}(m, \sigma) \begin{cases} 1 & \text{ha hiteles} \\ 0 & \text{ha hamisított az aláírás} \end{cases}$$

Jelentéktelen, hogy elutasítja vagy elfogadja a algoritmus végén a kapott választ, a hitelességét mindig meg tudja állapítani. Ha valahol egy külső ember megváltoztatott valamit, az ki fog derülni az azonosítás során.

Az aláíró módszer[3]:

[3]Amos Fiat and Ami Shamir - How To Prove Yourself

B szerepe a kölcsönható személyazonosság megállapításban passzív, de szükségszerű. B elküld egy véletlenszerű $e_{i,j}$ mátrixot ami nem tartalmaz információt, de meggátolja az A általi előre nem látható csalást. Ezt a módszert, hogy átalakítsuk aláíró módszerre, B szerepét kicseréljük egy f függvényre és követjük a következő protokollt:

Hogy aláírjuk az m üzenetet:

1. A kiválaszt tetszőleges $r_1, ..., r_t \in [0, n) - t$ és kiszámolja az $\mathrm{x}_i = r_i^2 \ (mod\ n)$-t.
2. A kiszámolja $f(m, x_i, ..., x_t)$-t és használja az első kt karaktert $e_{i,j}$ megfelelő értékeiként $(1 \leq i \leq t\ ,\ 1 \leq j \leq k)$.
3. A kiszámolja az $y_i = r_i \prod_{e_{i,j}=1} s_j \ (mod\ n)$-t ahol $i = 1, ..., t$ és s_j a v_j^{-1} legkisebb négyzetgyöke, $j = 1, ..., k$, majd elküldi B-nek: $I, m, e_{i,j}$ és az összes y_i-t.

Hogy B visszafejtse A aláírását az m üzeneten:

1. B kiszámolja a $v_j = f(I, j)$-t ahol $j = 1, ..., k$.
2. B kiszámolja $z_i = y_i^2 \prod_{e_{i,j}} v_j$ (mod n)-t ahol $i = 1, ..., t$.
3. B visszafejti az első kt karaktert az $f(m, z_1, ..., z_t)$-el az $e_{i,j}$-ből.

Biztonság : A protokoll biztonsága függ attól, hogy n eléggé nagy legyen és f valóban véletlenszerűen választott függvény legyen. Következésképpen nem lesz olyan támadás, ami feltöri a sémát bármilyen n-re vagy f-re, kivéve ha a támadó könnyedén tud faktorizálni.

Lemma: Ha A és B követik a protokollt, B mindig visszakapja az aláírást ha az az eredeti volt.

Bizonyítás:

A definíciók alapján:

$$z_i = y_i^2 \prod_{e_{i,j}=1} v_j = r_i^2 \prod_{e_{i,j}=1} (s_j^2 v_j) = r_i^2 = x_i \ (\mathrm{mod\ n})$$

és így

$$f(m, zi, ..., z_t) = f(m, x_1, ..., x_t).$$

□

Lemma: A választ egy sajátos aláírást az összes lehetséges aláírás közül az m üzenethez egyenletes valószínűségi eloszlással.

Bizonyítás:

Adott az aláírás, $e_{i,j}$ mátrix és az y_i értékek. Ekkor lehetséges visszaalakítani $r_1^2, ..., r_k^2$ (mod n)-t egyedülálló módon és $r_1, ..., r_k$ pontosan 4^k különböző esete van.

Amikor A véletlenszerűen választja ki r_i-t a különböző aláírásokat egyenlő valószínűséggel választja ki.

□

1.3.1. Bonyolultság

A tervezett aláíró sémában, az ellenfél tudja előre, hogy az aláírását el fogják-e fogadni érvényesnek, és így tud próbálkozni 2^{kt} véletlenszerű r_i-vel. Emiatt megtalálhatja a megfelelő aláírást amit elküldhet B-nek.

A kt szorzatot érdemes legalább 72-re megemelni, amikor a személyazonossági módszer helyett aláíró sémát használunk.

Ha $k = 9$ és $t = 8$ a választásunk,akkor az megkíván 2^{-72} biztonsági szintet. Egy titkos kulcsot el tudunk tárolni 576 byte-on a ROM-ban, és minden egyes aláírás 521 byte-ot igényel. A moduláris szorzás átlagértékei ennél a választásnál $\frac{t(k+2)}{2}$ ami egyenlő 44-el. (lásd. táblázat)

Ha megkettőzzük a kulcs méretét 1152 byte-ra ($k = 18$) csökkenteni tudjuk minden egyes aláírás méretét 265 byte-al (t= 4) anélkül hogy megváltoztatnánk a 2^{-72} biztonsági szintet. Ahhoz hogy optimalizáljuk a szorzások rendjét hogy kiszámoljuk a t részszorzatot, csökkenteni tudjuk az átlagszámukat 32-re. Ez csak a 4 százaléka az RSA aláíró séma által igényelt szorzatok számának.

Az egyedi azonosítás és az aláírás módszer egy sajátos jellemzője hogy lehetséges megváltoztatni a biztonsági szintet azután hogy a kulcs is meg lett változtatva. [4]

k	t	titkos kulcs mérete byteokban	aláírás mérete bytokban	standard átlag	optimalizált átlag	v_i átlag
1	64	64	4608+9	108	108	1
2	128	128	2304+9	72	64	2
3	192	192	1536+9	60	49	3
4	256	256	1152+9	54	46	4
6	384	384	768+9	48	41	6
8	512	512	576+9	45	45	8
9	576	576	512+9	44	44	9
12	768	768	384+9	42	35	12
18	1152	1152	256+9	40	32	17
24	1536	1536	192+9	39	28	21
36	2304	2304	128+9	38	30	24
72	4608	4608	64+9	37	37	36

[4] Amos Fiat and Adi Shamir - How To Prove Yourself page:193

1.3.2. Biztonságos digitális aláírás

Vegyünk egy (G,S,V) nyilvános kulcsú aláíró sémát:

1. Kulcsgeneráló algoritmus:

   $G : 1^n \to (pk, sk)$ ahol pk (public key) a nyilvános, sk (secret key) a titkos kulcs.

2. Aláíró algoritmus: S: $sk \times \{0,1\}^* \to Y$ ahol Y az aláírások tere, $\{0,1\}^*$ pedig az üzenet tere.

3. Ellenőrző algoritmus: V: $pk \times \{0,1\}^* \times Y \to \{0,1\}$ ahol az 1 kimenet a sikeres hitelesítést, a 0 pedig a sikertelent jelenti.

Biztonságos one-time aláírás:

Legyen $f : \{0,1\}^n \to \{0,1\}^n$ egyirányú permutáció. Egy $m \in \{0,1\}^l$ üzenet aláírásához véletlenszerűen generálunk $2l$ darab külön-külön n bit nagyságú blokkot. Ezek fogják az sk titkos kulcsot alkotni.

A blokkok f szerinti leképezésével fogjuk megkapni a $2l$ db pk nyilvános kulcsot.

Azaz:

$$sk = ((r_{1,0}, r_{1,1}), ..., (r_{l,0}, r_{l,1}))$$
$$pk = ((f(r_{1,0}), f(r_{1,1})), ..., (f(r_{l,0}), f(r_{l,1})))$$

Az aláírást bitenként készítjük el: ha az $m_i = 0$, akkor az aláírás r_{i0}, ha pedig $m_i = 1$ akkor r_{i1}. Következésképpen:

$$S(m) = (r_{l,m_1}, ..., r_{l,m_l})$$

V algoritmusa ellenőrzi az aláírás hitelességét és helyességét, felhasználva a pk nyilvános kulcsot, f leképezést, és az $[m, S(m)]$ inputot.

Tétel: Ha a támadó legalább 2 kéréssel fordul az aláíró orákulumhoz, továbbá $2 \leq l$***, akkor a feltörés valószínűsége*** 1***.***

Bizonyítás:

T támadó aláírást kér az orákulumtól az 1^l és a 0^l üzenetekre, amivel azonnal megszerzi a titkos kulcsot.

Mivel feltettük hogy $2 \leq l$, ezért a támadó írhat egy új üzenetet, amelyet alá is tud majd írni.

□

Definíció: Egy (G,S,V) aláírási séma (t,q,ε)-biztonságú, ha ∀ Z^S hatékony algoritmus esetén, t idő alatt, a S aláíró orákulumnak q kérdést tehet fel, annak a valószínűsége, hogy előállít egy új, korábban nem kérdezett üzenet-aláírás párt, legfeljebb ε.

$$Pr\{V_{pk}(m', y') = 1, \text{ ahol } (m', y') \leftarrow Z^S(pk)\} \leq \varepsilon$$

Tétel: Ha a támadónak csak egy kérése lehet és** f **egy** (t, ε)**-biztonságú egyirányú permutáció, valamint** $1 \leq l$**, akkor az aláírás séma** $(t, 1, 2\varepsilon)$**- biztonságú.[5]

1.4. RSA algoritmus

Az RSA eljárás egy nyílt kulcsú aszimmetrikus titkosító algoritmus. Az eljárásának az alapja a moduláris számelmélet valamint a prímszámelmélet. Napjainkban az egyik legelterjedtebb titkosítási forma.

Az algoritmus:

1. Kulcsválasztás:

   (a) Választunk két nagy prím számot : p,q ahol $p \neq q$. (Nagy szám a legalább 500 bit méretű bináris szám)

   (b) Legyen $N = pq$ modulus, és legyen $\varphi(N) = (p-1) * (q-1)$. Választunk egy olyan k számot, ami relatív prím $\varphi(N)$-hez. (Azaz relatív prím $(p-1)$-hez és $(q-1)$-hez is.)

[5]Buttyán Levente, Vajda István - Kriptográfia és alkalmazásai

(c) Kiszámoljuk k $(mod\ \varphi(N))$. (Keresünk egy olyan l számot amire teljesül, hogy: $kl = 1\ (mod\ \varphi(N)) : 1 < l < \varphi(N)\ )$

2. Titkosítás:

   Az algoritmushoz szükségszerű egy olyan hely, ahol minden felhasználónak megtalálható a nyilvános kulcsa. A kulcs nyilvános részét (N, k), feltöltjük erre a helyre, a titkos részét (l, p, q) pedig magunknál tartjuk. Mivel az (l, p, q) számhármas titokban marad, így a $\varphi(N)$ is.

   Ha Alice szeretne küldeni Bobnak egy titkos üzenetet, akkor megkeresi Bob nyilvános kulcsát, és előkódolja az üzenetet. Alice üzenete valamilyen karaktersorozatból áll. Ezt kellene úgy átalakítani, olyan nem negatív egészek sorozatára, melyekre teljesülni fog, hogy kisebbek mint N_B, ahol N_B jelöli Bob modulusát. Ezt a folyamatot minden felhasználó ismeri.

   Az előkódolt üzeneten hajtja végre Alice a rejtjelezést, sorban minden számon. Ha az előkódolt szövegnek a soron kövekező száma s, akkor a hozzá tartozó rejtjeles szám :

   $$x = E_B(s) = s^{k_B} \pmod{N_B}, \text{ ahol } k_B \text{ a Bobhoz tartozó } k \text{ szám.}$$

3. Visszafejtés:

   Miután Bob megkapta a kódolt üzenetet Alicetől, ami $x_1, x_2, ..$ sorozat, ahol x_i 0 és N_B közötti egész szám, a dekódolást a sorozat elemein külön-külön hajtja végre. Ha a soron következő szám x akkor az előkódolt üzenet:

   $$s = D_B(x) = y^{l_B} \pmod{N_B}, \text{ ahol } l_B \text{ Bob } k_B\text{-hez tartozó inverze.}$$

   Ekkor kapunk egy $s_1, s_2, ...$ sorozatot melyre ha az előkódolás inverzét alkalmazzuk akkor visszakapjuk az eredeti üzenetet.

   Tétel: Bármely s egész számra fenn áll hogy:

$$(s^k)^l = x \ (mod \ N)$$

Bizonyítás:

A k $(mod\ \varphi(N))$ inverze l, emiatt $\exists v \in \mathbf{Z} : kl = v\varphi(N) + 1$

Elég belátni: $s = s^{v\varphi(N)+1}$ mod(N).

Legyen u tetszőleges prím. Ekkor tetszőleges s és r esetén:

$$s = s^{r(u-1)+1} \ (\text{mod } u) \tag{1}$$

Ha u nem osztója s-nek, akkor a Fermát-tételt felhasználva:

$$(s^{u-1})^r = 1^r = 1 \ (\text{mod } u) \Rightarrow s = s^{r(u-1)+1} \ (\text{mod } u) \text{ teljesül.}$$

Ha u osztója s-nek akkor:

$$s = 0 = s^{r(u-1)+1} \ (\text{mod } u).$$

Általános esetben: $(p-1) \mid \varphi(N)$ és $(v-1) \mid \varphi(N)$, az (1) miatt:

$$s = s^{v\varphi(N)+1} \ (\text{mod } p)$$
$$s = s^{v\varphi(N)+1} \ (\text{mod } q)$$

teljesül. Ebből következik:

$$p \mid s^{v\varphi(N)+1} - s$$
$$q \mid s^{v\varphi(N)+1} - s.$$

Azaz

$$N \mid s^{v\varphi(N)+1} - s.$$

□

2. fejezet

Partner−hitelesítés interaktív módon

2.1. Kihívás-válasz protokoll (challenge and response)

Alice hitelesíteni akarja magát Bobnak. A két fél megegyezik egy titkos f függvényben, amit csak ők ismernek.

A kihívás-válasz hitelesítő rendszer működése:

1. B egy véletlen m üzenetet, egy úgynevezett kihívást küld A-nak.
2. A visszaküldi r-t ami m transzformáltja, mégpedig $r = f(m)$.
3. B érvényesíti A válaszát, azaz ellenőrzi.

Példa1:

Vegyünk egy felhasználót, aki be szeretne lépni a bankfiókjába. A netbankok általában többszörösen védettek. A bejelentkezésekhez szükséges egy állandó jelszó (amit 3-5 havonta cserélni kell), valamint emellett használnak egyszer-használatos jelszavakat is. Az egyszer-használatos jelszavaknál a kihívás egy véletlenszerűen generált karaktersorozat (ami legtöbbször 8-10 karakter hosszú szám), és ez össze van kapcsolva egy hitelesítési kísérlettel. A válasz pedig a kapott szám megfelelő helyre beírása. Aki be szeretne jelentkezni meg kell, hogy adja a felhasználónevét, és jelszavát. A beléptető rendszer, küld neki egy karaktersorozatot, amit a felhasználónak be kell írnia a beléptetőfelületnél. Ha elírta, a rendszer elutasítja, továbbá a jelszava lejár, vagyis másodjára már egy másik kóddal tud majd csak belépni, amit hasonlóan tud megszerezni. Továbbá ezekre a jelszavakra gyak-

ran raknak időkorlátot is, azaz a felhasználónak adnak 5-10 percet, hogy felhasználják a kapott jelszót, utána érvénytelenítik.

Minden felhasználó más beléptető kódot kap: A beléptetőkódjával, B nem tud belépni a saját fiókjába.

Példa2: Tegyük fel hogy, a kihívás-válasz rendszerünk a következőképpen néz ki:

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
1	2	3	4	5	6	7	8	9	9	8	7	6	5	4	3	2	1	1	2	3	4	5	6	7	8

Az f függvény a példában megfelel azzal a hozzárendeléssel, hogy :

$$a \to 1, b \to 2 \ ... \ y \to 7 \ z \to 8$$

Ha a Kihívó elküldi a *dinuszaurusz* szót, akkor a helyes válasz, amit vissza kell kapnia: 495418131318

A jelszó alapú hitelesítéseket, főleg emberek azonosítására hitelesítésére használjuk. Amennyiben A jelöli az azonosítani akaró félt B pedig azt aki azonosítja, és A és B képesek bonyolultabb számításokra is, akkor tudnak használni kriptográfiai protokollra épülő hitelesítést is. Ekkor A próbálja meggyőzni B-t hogy birtokában van egy olyan kriptográfiai kulcs, ami egyértelműen bizonyítja A kilétét, azaz B pontosan tudja, hogy ha a kulcs A-nál van, akkor ő csakis A lehet, nem lehet egy T támadó.

Feladat1:

Legyen A egy repülőgép ami szeretne leszállni egy B reptéren, de ehhez azonosítania kell magát.

Amit tudunk:

1. B ismeri A jelszavát, és nincs több közös titkuk.
2. A csak (mod 2) összeadást tud végrehajtani.
3. T támadó lehallgatja A jeleit, viszont B tud úgy jeleket küldeni, hogy azt T ne tudja lehallgatni.

Mi legyen a protokoll?

Megoldás:

Legyen a jelszó p. B kiválaszt egy jelszó méretű r véletlen számot.

A protokoll:

1. $B \rightarrow A : r$
2. $A \rightarrow B : p + r \ (mod\ 2)$

B elküldi az r véletlen számot A-nak, amit T nem tud megszerezni. Ehhez A hozzáadja az előre megbeszél jelszót, és azt küldi vissza B-nek. T-nek sikerül megszereznie, viszont a kapott $p + r$-ből nem tudja kiszámolni magát a jelszót.

A visszajátszásos támadások elkerülése érdekében a kihívás-válasz protokollt szoktuk használni. (A visszajátszásos támadás esetében a támadó a protokoll egyes üzeneteit a protokoll korábbi futásai során lehallgatott megfelelő üzeneteivel helyettesíti.)

A kihívás-válasz protokollnak két fő kategóriája van: szimmetrikus és aszimmetrikus. A protokoll a gyökvonás és a moduláris négyzetgyökvonás nehézségén alapul amikor a n faktorizációja nem ismert.

2.1.1. Szimmetrikus kulcsú rendszerek

Legyen A (Alice) és B (Bob) a két fél, x az üzenet, amit próbálnak egymásnak eljuttatni, K a szimmetrikus kulcs, amit rejtjelezésre használnak.

Szimmetrikus kulcsú hitelesítés rejtjelezéssel:

1. $B \rightarrow A$: N_B
2. $A \rightarrow B$:$\{N_B\}_{K_{AB}}$

B generál egy N_B véletlen számot, és elküldi A-nak. A a használva a K_{AB} kulcsot rejtjelezi N_B-t, ahol K_{AB} csak A és B által ismert szimmetrikus kulcs. B dekódolja A

válaszát, és amennyiben az eredeti N_B-t kapta, akkor A hitelesítette magát. A hitelesítés során feltételezzük, hogy csak A és B tud rejtjelezni K_{AB}-vel.

Egyszerűben: Alice megírja az üzenetet, amit szeretne Bobhoz eljuttatni, majd a közös kulccsal titkosítja és elküldi. Bob a dekódoláshoz, ugyanazt a közös kulcsot fogja használni.

Hitelesítés szempontjából, ugyanúgy működik az algoritmus. A szimmetrikus kulcsú rendszereknél csak az identitását bizonyító és az ellenőrző fél tudja a szimmetrikus kulcsot, így csak ők tudnak egymásnak megadni hiteles kódot.

Szimmetrikus kulcsú hitelesítés dekódolással:

1. $B \to A$: $\{N_B\}_{K_{AB}}$
2. $A \to B$: $\{N_B\}$

B generál egy N_B véletlen számot, rejtjelezi a K_{AB} kulccsal, ahol K_{AB} csak A és B által ismert szimmetrikus kulcs. A kapott eredményt elküldi A-nak, aki dekódolja az üzenetet, majd visszaküldi a eredményt B-nek. B ellenőrzi A válaszát, és ha egyenlő N_B-vel akkor A sikeresen hitelesítette magát. A hitelesítés során feltételezzük, hogy csak A és B tud rejtjelezni K_{AB}-vel.

Továbbiakban legyen C (Cecil) a támadó, aki szeretné magát Alice-ként azonosíttatni Bobbal.

Az üzenethitelesítést legtöbbször üzenethitelesítő kódok alkalmazásával valósítjuk meg. A MAC (message authentication code) egy üzenethitelesítő kód, amit mindenki ismer a hitelesítés során. A MAC-hez tartozó K kulcs lesz a közös kulcs A és B között.

1. B generál egy y véletlen számot, majd elküldi a kihívást A-nak.

2. A kiszámolja $x_1 = MAC_K(y)$-t és a kapott x_1-et visszaküldi. (MAC_k K kulcs hitelesítő kódja, ahol K a titkos kulcs).

3. B kiszámolja $x_2 = MAC_K(y)$-t, és leellenőrzi, hogy az A-tól kapott válasz megegyezike a sajátjával. ($x_1 = x_2$)

Ha C ismeri, vagy megszerzi az y számot, akkor sem tudja kiszámolni $MAC_K(y)$-t, mivel a hitelesítő kódhoz tartozó kulcsot nem ismeri. Ettől függetlenül, így is tud sikeres támadást végrehajtani.

Def: Egy támadás akkor sikeres, ha az üzenet megváltozik, vagy nem az eredeti címzetthez jut el az üzenet.

1. B generál egy y számot, majd elküldi a kihívást A-nak.

2. C lehallgatja azt a csatornát, ahol a hitelesítés folyik és megszerzi az y-t, és ezt visszaküldi B-nek.

3. B kiszámolja $x_1 = MAC_K(y)$-t és elküldi C-nek.

4. C elküldi $x_1 = MAC_K(y)$-t B-nek.

5. B kiszámolja $x_2 = MAC_K(y)$-t, és leellenőrzi, hogy az A-tól kapott válasz megegyezike a sajátjával. ($x_1 = x_2$)

A hitelesítés végére Bob azt fogja hinni, hogy Cecil Alice.
A protokollt biztonságossá tétele:

1. B generál egy y számot, majd elküldi a kihívást A-nak.

2. A kiszámolja $x_1 = MAC_K(ID(A)||y)$-t és elküldi B-nek.

3. B kiszámolja $x_2 = MAC_K(ID(A)||y)$-t, és leellenőrzi, hogy az A-tól kapott válasz megegyezik-e a sajátjával. ($x_1 = x_2$)

A megoldás az $x_1 = MAC_K(ID(A)||y)$ kiszámításában van. Amennyiben C az előző protokollt használva szeretne támadást végrehajtani, B-től az $x_1 = MAC_K(ID(B)||y)$ választ fogja kapni, amit később B el fog utasítani.

Ha mindkét fél azonosíttatni szeretné magát a másikkal akkor a következő protokollokat tudják felhasználni:

1. B generál egy y_1 számot, majd elküldi a kihívást A-nak.
2. A generál egy y_2 számot, és kiszámolja $x_1 = MAC_K(ID(A)||y_1||y_2)$ és elküldi B-nek.
3. B kiszámolja $x'_1 = MAC_K(ID(A)||y_1||y_2)$ és leellenőrzi, hogy az A-tól kapott válasz megegyezik-e a sajátjával ($x_1 = x'_1$). Ezután B kiszámolja $x_2 = MAC_K(ID(B)||y_2)$ és elküldi A-nak.
4. A kiszámolja $x'_2 = MAC_K(ID(B)||y_2)$ és leellenőrzi, hogy az B-től kapott válasz megegyezik-e a sajátjával ($x_2 = x'_2$).

2.1.2. Aszimmetrikus kulcsú rendszerek

A szimmetrikus kulcsúval ellentétben itt nem közös kulcs van, hanem minden résztvevőnek (a példánk szempontjából most a 2 félnek A-nak és B-nek) külön-külön van egy nyilvános és egy titkos kulcspárja. Előny, hogy nincs közös kulcs, így nem is kell előre egyeztetni, hogy mi legyen az. Ha egynél több résztvevő van, akkor a nyilvános kulcsokat egy kulcstárban helyezik el, és ha valakinek szüksége van a másikéra csak kikeresik onnan.

Az aszimmetrikus rendszernek ezt a két fajtáját használjuk :

1. Az egyik amikor bizonyító fél aláírja a kihívó által küldött véletlen számot.
2. A másik pedig amikor titkos kulcsával visszafejti az aszimmetrikusan titkosított véletlen számot.

Digitális aláírás alapú:

1. B generál egy véletlen r_1 számot és ezt elküldi A-nak.

2. A kiszámolja az $x = r_2||r_1||ID(B)||Sign_{SK_B}(r_2||r_1||ID(B))||Cert_A$ ahol $Cert_A$ az A tanúsítványa, SK_A A titkos kulcsa, és r_2 A által generált szám.

3. B ellenőrzi $Cert_A$-t, hogy hiteles-e.

4. B ellenőrzi az aláírás helyességét, azaz valóban A titkos kulcsával lett-e aláírva.

Vegyünk egy olyan algoritmust ahol A ás B kölcsönösen tudják igazolni identitásukat.

1. Az előző algoritmus első 4 lépését végrehajtják.

2. B kiszámolja $x' = r_2||r_1'||ID(A)||Sign_{SK_B}(r_2||r_1'||ID(A))||Cert_B$ üzenetet és visszaküldi A-nak, ahol $Cert_B$ az B tanúsítványa és SK_B B titkos kulcsa.

3. A ellenőrzi $Cert_B$ hitelességét.

4. A ellenőrzi az aláírás hitelességét, azaz hogy B titkos kulcsával írták-e.

2.1.3. Zero-Knowledge Protokoll

Napjainkban használt azonosítási rendszerek, (legyen az jelszó alapú, vagy kihívás-válasz alapú, esetleg aláírásos) részinformációkat szolgáltatnak a külvilágnak a titkos információról, ezzel könnyítve a támadónak a sikeres támadás végbemenését. A kriptográfiában a zero-knowledge protokoll egy módszer arra, hogy az egyik fél bizonyítani tudja a másik félnek, hogy tudatában van egy információnak anélkül, hogy ezt az információt vagy a hozzá vezető utat felfedné. Ez a kihívás-válasz protokoll egyik speciális fajtája. Egyszerűbben : azon kívül, hogy bizonyítanánk az állításunk helyességét, más információt nem adunk ki.

Ahhoz, hogy bizonyítsunk egy olyan állítást ami egy titkos információt igényel a bizonyító féltől (legyen A) úgy, hogy a hitelesítő (legyen B) ne legyen képes bizonyítani az állítást bárki másnak, (hiszen Ő nem ismeri a titkos információt) még akkor sem, miután A sikeresen elvégezte az állításának igazolását. Vegyük észre, hogy ebben az esetben a bizonyításhoz szükséges az a titkos információ, amit nem szeretnénk elárulni, vagyis az

állításunkat nem tudjuk teljes egészében belátni ezzel a módszerrel, hiszen akkor olyan információt szivárogtatunk ki, ami elvenné a módszer lényegét vagyis a titok kiderülne.

Emiatt, hogy ez ne történjen meg, A-nak és B-nek egy olyan kihívást kell választania, amit ha A helyesen megold, akkor elég kicsi valószínűségű legyen az, hogy átveri B-t. Növeli az esélyeket, ha a kihívást kétszer vagy akár többször is elvégezzük, ezzel folyamatosan kizárva annak az esélyét, hogy A-nak csak szerencséje van, azaz hazudik.

Vegyünk 2 példát:

Példa1: Barlang

Van egy barlangunk, aminek van egy bejárata (1.ábra), ahol két irányba lehet elindulni. Legyen az egyik a J mint jobb a másik B mint bal járat, továbbá legyen egy K kiindulási pontunk ami a barlangon kívül van. A barlangunk annyiban speciális, hogy ha bemegyünk J járatba, akkor ki tudunk jönni egyértelműen B járatból és ugyanez visszafele is igaz, vagyis nincsenek mellékutak. Tegyük fel hogy, ennek az alagútnak a közepén van egy ajtó, amit egy titok nyit. (Ez a titok, lehet egy számkód, egy jelszó, egy kulcs, vagy ujjlenyomat olvasó stb.)

Éva azt állítja Aladárnak, hogy Ő ismeri ennek az ajtónak a titkos kódját, de Aladár ezt nem hiszi el. Éva be szeretné bizonyítani, de úgy, hogy közben nem mondja el Aladárnak a titkos kódot. A bizonyítás során, mi kevesebbet fogunk tudni, mint Aladár, vagyis azon kívül, hogy Éva használja-e a titkos kódot, azt sem fogjuk tudni, hogy Aladár és Éva nem-e dolgoznak össze. Pl: Előre megbeszélik, hogy mikor hol menjen be Éva a barlangba.

Aladár a K kiindulási pontnál van, amíg Éva bemegy és eldönti, hogy a J vagy a B bejáratot használja. Ezután Aladár bemegy a két járat találkozásához. Fontos: Aladár semmilyen tudást nem szerzett arról, hogy Éva melyik bejáratot választotta és azáltal, hogy ott áll a bejáratoknál, továbbra sem tud semmilyen plusz információt kapni. (Például: a barlangban nem lehet hallani Éva lépteit.)

Aladár bekiabálja Évának, hogy melyik kijáraton szeretné, hogy Éva kijöjjön. Ezután megvárja, hogy melyik bejáratnál bukkan fel.

Tegyük fel, hogy Aladár a J bejáratot válassza és Éva a B kijáraton jön ki. Ebben az

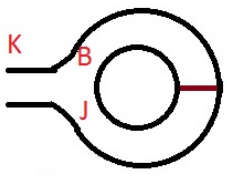
esetben Aladár egyértelműen állíthatja, hogy Éva nem tudja a titkos kódot, hiszen akkor át tudott volna jutni a másik oldalra, tehát hazudott.

Ha Éva a J kijáraton jön ki, akkor 2 lehetősége van Aladárnak: Vagy elhiszi Évának, hogy tudja a kódját az ajtónak, vagy azt mondja, hogy szerencséje volt, hiszen Ő véletlenszerűen választott kijáratot egyenlő eséllyel, így Évának 50 százaléka volt eltaláni, hogy mit fog mondani.

Hogy Aladárnak ne maradjanak kétségei, ezért többször megismétlik a módszert. 20 alkalomnál annak az esélye, hogy Éva mindig eltalálja a megfelelő bejáratot (vagyis annak a valószínűsége, hogy nem tudja a kódot, de mindig jó kijáraton jön ki) $\frac{1}{2^{20}}$ ami jóval kisebb mint 1 százalék. Ha 100-szor ismétlik meg még kisebb az esélye, vagyis Aladár anélkül, hogy tudná a titkos kódot, megbizonyosodott arról, hogy Éva nagy valószínűséggel tényleg ismeri.

Ha van egy harmadik ember (Cecil) akit odaállítunk Aladár mellé és feltételezzük, hogy Ő csak megfigyel és nem kérdezhet semmit, akkor Cecil csak annyit fog látni, hogy ha Aladár azt kiáltja hogy *Jobb* akkor Éva onnan jön ki, ha pedig azt hogy *Bal* akkor pedig onnan. Ezáltal nem fogja megtudni a titkos kódot az ajtóhoz, sőt elhinni sem fogja, hogy Éva ténylegesen tudja, mivel azt az esetet, hogy Aladár és Éva összedolgozik nem zárhatjuk ki.

Abban az esetben ha Évára tesznek egy kamerát, a protokoll elveszíti a lényegét, hiszen a titok kiderül.



1.ábra

Példa2: Gráf izomorfia

Legyen adott két gráf: G_1 és G_2, Alice (A) szeretné meggyőzni Bobot (B), arról hogy a két gráf izomorf. (Két gráf izomorf egymással ha létezik olyan π permutáció, amellyel az egyik gráf pontjai és élei kölcsönösen egyértelműen és illeszkedéstartóan megfeleltethetők

a másik gráf pontjainak, éleinek.) Azaz Alice szeretné bebizonyítani, hogy ismeri a π permutációt melyre: $\pi(G_1) = G_2$. Természetesen Alice elküldheti Bobnak a π-t, de az nem lenne zero-knowledge. A feladat, hogy anélkül győzze meg Bobot, hogy bármit is elárulna a π-ről.

A protokoll a következő:

1. $A \to B$: A választ véletlenszerűen egy φ permutációt és egy $b \in \{0, 1\}$ bitet és kiszámolja $H = \varphi(G_b)$-t, majd elküldi H-t B-nek.

2. $B \to A$: B választ egy $b' \in \{0, 1\}$ bitett és elküldi A-nak.

3. $A \to B$: A elküldi a ε permutációt B-nek ahol:

$$(\varepsilon)) \begin{cases} \varphi & \text{ha } b = b' \\ \varphi\pi^{-1} & \text{ha } b = 0 \text{ , } b' = 1 \\ \varphi\pi & \text{ha } b = 1 \text{ , } b' = 0 \end{cases}$$

4. B csak akkor fogadja el a A állítását, ha $H = \varepsilon(G_{b'})$.

A protokoll végére B mindig el tudja majd dönteni, hogy a két gráf izomorf-e.

Egy zero- knowledge protokollnak az alábbi tulajdonságokat mindig teljesítenie kell:

1. Teljesség: Ha az állítás igaz, akkor az aki állítja meg tudja győzni a társát egy hihető bizonyítékkal.

2. Megbízhatóság: Ha az állítás hamis, akkor nincs olyan algoritmus amivel meg lehetne győzni valakit az ellenkezőjéről.

3. Nem adunk információt a titokról: Ha az állítás igaz, akkor semmi mást nem tudunk meg az állításról csak kizárólag annyit, hogy igaz. Vagyis tudni az állítást, csupán csak azt mutatja hogy a bizonyító tudja a titkot.

Egy Zero- Knowledge protokoll a modulo szerinti négyzetgyökvonás nehézségére épül, ahol a modulus egy összetett szám (n=pq). Ha b $\in$ N $0 < b < n$, ami az $x^2 = c$ (mod n) egyenlet megoldása akkor a b megoldás a c négyzetgyöke (mod n), a c számot pedig kvadratikus maradéknak nevezzük. Ha $n = p$ ahol p prím, akkor létezik megoldás a c-re, és ezek a b és a p-b lesznek. Mivel ekkor egy p test feletti az egyenlet, emiatt maximum 2 különböző megoldásunk lehet (speciális eset ha $b = p/2$).

Tétel: Ha $n = p$ ***prímszám, akkor az egyenlet pontosan*** $s = \frac{p-1}{2}$ ***különböző*** c ***értékre oldható meg.***

Bizonyítás:

Tegyük fel hogy, b megoldása az egyenletnek, ekkor $p - b$ is megoldása lesz, azaz a kvadratikus maradékok száma legfeljebb s lehet.

Elég belátni azt hogy:

$$\forall b_1 \neq b_2\text{-re, } 0 < b_1 < b_2 < s \text{ esetén } b_1^2 \neq b_2^2 \text{ (mod p)}$$

Tegyük fel hogy, $\exists\ b_1, b_2$ amelyre : $b_1^2 = b_2^2$ (mod p) teljesül, tehát a $p|(b_2^2 - b_1^2)$. Ebből következik hogy:

$$p|(b_2 - b_1) \text{ vagy } p|(b_2 + b_1)$$

ami nem következhet be, hiszen :

$$0 < (b_2 - b_1) < p$$

valamint

$$0 < (b_2 + b_1) < p\ .$$

□

Ha n= pq (p és q is prímszám), akkor ha $\exists$ b megoldás a c-re, akkor c-nek négy megoldása van : b_1, $n - b_1$, b_2, $n - b_2$, ahol $0 < b_1, b_2 < n$. A megoldásokat az $x^2 = c$ $(mod\ p)$ és az $x^2 = c$ $(mod\ q)$ egyenletek megoldásainak felhasználásával kaphatjuk meg. Ha az

$x^2 \equiv a \ (mod\ n)$ kongruenciának létezik megoldása, akkor a megoldások kiszámításához létezik polinomiális idejű algoritmus. (A számításokhoz szükség van a kínai maradéktétel használatára). Ha az n modulus faktorai nem ismertek, akkor a kongruencia megoldása nehéz probléma.

Állítás: Ha ismerjük p-t és q-t akkor az $x^2 = c$ (mod n) egyenletnek a megoldásait polinomiális időben ki tudjuk számítani, hiszen a prímszám egyenletek megoldásainak kiszámítására ismerünk polinomiális idejű algoritmust. Ha viszont p és q ismeretlenek, akkor ez nehéz feladat.

Bizonyítás:

Legyen d : $0 < d < n$ egy véletlen szám, számítsuk ki d^2 (mod n) és ezt jelöljük c-vel.
Keressük meg a $x^2 = c \ (mod\ n)$ egyenlet egy megoldását, és jelöljük d'-vel.
Ebben az esetben:

$$d^2 - d'^2 = 0 \text{ (mod n)}$$
$$n|(d - d')(d + d').$$

Mivel d-t véletlenszerűen választottuk ki az elején, emiatt $\frac{1}{2}$ annak az esélye, hogy:

$$d = d' \text{ vagy } d = n - d'.$$

Ebből következik

$$d - d' = 0 \text{ vagy } d + d' = n.$$

Ekkor egyértelmű, hogy n osztója az $(d - d')(d + d')$ szorzatnak.
Viszont annak is $\frac{1}{2}$ a valószínűsége $d - d' \neq 0$ (mod n), ekkor

$$|d - d'| < n \text{ és } d + d' < 2n$$

miatt mindegy, hogy a LNKO(n,$|d - d'|$) vagy a LNKO(n,$d + d'$) számításával n faktorizálásához jutunk, amiről tudjuk, hogy nehéz feladat.

□

2.1.4. Fiat-Shamir-protokoll

A protokollhoz szükséges egy kulcsgeneráló központ, ami véletlenszerűen választ két darab prímszámot, és ezek szorzata lesz az n modulus.

Vegyünk egy bevezető példát:

Amikor A megpróbál belépni a hitelesítő rendszerbe, a központtól (legyen most B) kap egy titkos (u) és egy nyilvános kulcsot (v). A titkos kulcs, egy véletlenszerűen generált szám, és a nyilvános kulcs az megegyezik a titkos kulcs négyzetével modulo n, vagyis $v = u^2 \ (mod\ n)$.

A protokoll:

Legyen R egy A által generált véletlen szám ($0 < R < n$), amíg b-t, B állítja elő, a második lépés előtt.

1. $A \rightarrow B : z = R^2 \ (mod\ n)$
2. $B \rightarrow A : b$
3. Ha $b = 0$, akkor $A \rightarrow B : R$

   Ha $b = 1$, akkor $A \rightarrow B : w = Ru \ (mod\ n)$

Miután a 3. lépés után, B az alábbi ellenőrzést hajtja végre:
Ha $b = 0$ volt, akkor ellenőrzi : $R^2 \ (mod\ n) = \mathrm{z})$
Ha $b = 1$ volt, akkor ellenőrzi : $zv = w^2 \ (mod\ n)$

Legyen C támadó, aki szeretné A-t megszemélyesíteni. Hogyan próbálkozhat vele?

1. C az első lépés szerint jár el, majd megkapja a b bitet. C ismeri A nyilvános kulcsát, így a b értékétől függően, vagy elküldi R-t, amit ismer, hiszen ő állította elő, vagy megpróbál gyököt vonni a zv $(mod\ n)$ értékből, ami szinte kivitelezhetetlen. Ebből következtetve $\frac{1}{2}$ esélye van a sikeres támadásra.

2. C megpróbálja előre kitalálni, hogy mi lesz a b, amit B majd visszaküld neki, és ennek megfelelően választja ki a z értékét.

   (a) Ha a sejtése az, hogy $b = 1$ lesz, akkor $z = R^2 v^{-1}$ $(mod\ n)$-t küldi el, majd a harmadik lépésnél R $(mod\ n\ )$-t

   (b) Ha a sejtése az, hogy $b = 0$, akkor $z = R^2$-et küldi el, majd a harmadik lépésnél, az R $(mod\ n\ )$-t

   Azt viszont előre nem tudja pontosan megmondani, hogy melyik b-t fogja kapni, így $\frac{1}{2}$ eséllyel dönt arról, hogy mi legyen z értéke, vagyis a támadás sikerességének az esélye $\frac{1}{2}$.

A protokollt k-szor megismételve azt kapjuk. hogy annak a valószínűsége, hogy C sikeresen hitelesítteti magát A-ként $\frac{1}{2^k}$.

A Fiat-Shamir azonosítási protokoll biztonsága azon alapszik, hogy a $x^2 \equiv c\ (mod\ n)$ megoldása könnyű ha n faktorait ismerjük, különben nehéz.

A protokollnak két résztvevője van: Alice (A) aki azonosítani szeretné magát, és Bob (B) aki ellenőrzi az azonosítás hitelességét. Bob véletlenszerűen választ két olyan prímszámot (p q), hogy a szorzatok faktorizálása ne legyen könnyű (például a 3 és 5 nem lesz jó), majd kiszámolja az $n = pq$-t, ami a modulus lesz és ezt nyilvánosságra hozza. Alice generál egy $x \in Z_n^*$ számot, majd kiszámolja $y \equiv x^2\ (mod\ n)$-t. Az x-et titokban tartja míg az y-t a nyilvánosságra hozza. Az x lesz Alice titkos kulcsa, az y pedig a nyilvános kulcsa.

A protokoll:

1. A generál egy véletlen $r \in Z_n^*$ számot, majd kiszámolja $k \equiv r^2\ (mod\ n)$ -t és elküldi B-nek.

2. B generál egy véletlen bitet: $c \in \{0, 1\}$, majd elküldi B-nek.

3. A kiszámolja $s \equiv rxc\ (mod\ n)$ -t és elküldi B-nek .

4. B ellenőrzi $s^2 \equiv ty^c\ (mod\ n)$ teljesül-e.

Ha $s^2 \equiv ty^c$ teljesül, akkor az azonosítás sikeres volt, ha nem teljesült, akkor pedig sikertelen. A protokollt többször is meg lehet ismételni, minden egyes alkalommal új r és c számot fog generálni A és B.

A protokoll is tartalmaz kihívás-válasz részt, mégpedig a 2. lépés amikor B generál egy véletlen c számot, az valójában egy kihívás, amit elküld A-nak. Ezután A-nak a 3. pontban a megfelelő választ kell visszaküldenie B-nek, hogy a hitelesítés hiteles legyen. Az utolsó lépésben pedig ellenőrzi B a kapott eredményeket.

Állítás: A protokoll minden szabályosan generált s-et hitelesnek fog találni, azaz B el fogja fogadni, A hitelesítette magát.

Bizonyítás:

□

Ahhoz hogy bizonyítani tudjuk, hogy egy T támadó nem szabályosan generált válasza nem lesz hiteles, azaz a protokoll hamis értéket ad vissza, meg kell vizsgálni hogy hogyan próbálhat támadásokat indítani a protokoll ellen a támadó.

Két különböző eset van:

1. Ha a támadó generál egy $t \in Z_n^*$ B c bitjére, majd megpróbálja kitalálni s -t. Annak a valószínűsége, hogy megtalálja a helyes s-t, elég nagy n esetén kicsi.

2. Ha a támadó megpróbálja előre kitalálni, hogy 0 vagy 1 lesz a c bit, és ennek megfelelően adja meg t és s értéket.

   Ha $c = 1$, T kiválaszt egy véletlen egy s számot és kiszámolja a $t \equiv \frac{s^2}{y} \ (mod\ n)$. Az így kiszámolt értékeket a megfelelő kihívásra elküldi T B-nek. Ha $c = 0$, T választ véletlenszerűen egy $r \in Z_n^*$ számot, és kiszámolja $t \equiv r^2 (mod\ n)$. Az így kiszámolt értéket a 4. lépésben s helyett r-et adja meg.

   Viszont T nem fogja tudni mind a 2 esetet kiszámolni, mivel akkor az x titkos kulcsot ki tudná számolni. Tegyük fel, hogy T valamilyen módonha $c = 0$ meg tudja adni s_0 -t, ha pedig $c = 1$ akkor pedig s_1-et.

Azaz:

$$s_0\text{-t } (s_0 = r) \text{ és } s_1\text{-t } (s_1 = rx) \text{ is ismeri} \Rightarrow x\text{-et } (x = \frac{s_1}{s_2})\text{is ismeri.}$$

Annak a valószínűsége, hogy T a helyes c értéket válassza: $\frac{1}{2}$. Ha megismételjük k-szor a protokollt akkor annak a valószínűsége, hogy mindig helyesen választja ki c-t : $\frac{1}{2^k}$

Feladat3:

Alice ismeri $y = g^x \ (mod\ p)$ diszkrét hatvány g szerinti diszkrét logaritmusát (x-et), és ezt szeretné bebizonyítani Bobnak is anélkül hogy felfedné x-et. (p egy prímszám, g pedig egy primitív elem $(mod\ p)$)Adjunk rá zero-knowledge protokollt!

Megoldás:

Legyen a protokoll a következő:

1. $A \to B : w = g^r \ (mod\ p)$ (r véletlen szám $(mod\ p)$.
2. $B \to A : b$ ($b \in \{0, 1\}$ egy véletlen kihívás).
3. $A \to B : z = r + bx \ (mod\ (p - 1)$.

A 3. lépésben B ellenőrzi, hogy $g^z = wy^b$-vel $(mod\ p)$. A protokollt többször is le lehet futtatni. Minél többször ismételik meg, annál biztosabb lehet Bob abban, hogy Alice ismeri x-et, azaz 2^{-t} valószínűséggel fogja elfogadni Alice állítását a t-dik alkalom után.

A protokoll az RSA-val ellentétben csak szorzást használ, a hatványozás helyett, így sokkal kevesebb számolást igényel.

Összefoglalás

Mindent egybevetve, a mai világban nélkülözhetetlenek a hitelesítő algoritmusok, mind az interaktív mind a nem interaktív formája. Teljesen mindegy, hogy az email-fiókunkat, vagy esetleg a vállalatunk bizalmas papírjait szeretnénk megvédeni, egy biztonságos rendszer kiépítése elkerülhetetlen.

A jelszavak használata egyszerűnek és kényelmesnek tűnik, hiszen csak meg kell jegyezni egy általunk megadott karaktersorozatot és onnantól kezdve védve vagyunk a külső érdeklődőktől. Sajnos nem figyelünk arra, hogy ha nem módosítjuk őket gyakran, a támadóknak megkönnyítjük a dolgukat, és nagy esélyt adunk nekik arra, hogy feltörjék jelszavainkat. Az egyszer-használatos jelszavak ugyan nem kényelmesebbek, de annál biztonságosabbak tudnak lenni.

A zero-knowledge protokoll egy nagyszerű lehetőség arra, hogy a titkunk megosztása nélkül bebizonyítsuk állításunkat. Nem szabad elfelejteni, hogy 100 százalékos bizonyítást sosem tudunk létrehozni, viszont egy adott protokoll többszöri megismétlésével, minimalizálni tudjuk hiba valószínűségét.

Minden egyes protokoll azért jött létre, hogy biztonságosabbá tehessük velük a környezetünket. Amikor kiválasztunk egyet saját magunk számára, mindig mérlegelni kell, hogy milyen célra szeretnénk használni, és mi kell ahhoz, hogy valaki fel tudja törni.

Irodalom jegyzék

1. Buttyán Levente, Vajda István : KRIPTOGRÁFIA ÉS ALKALMAZÁSAI
2. Amos Fiat and Adi Shamir : How To Prove Yourself: Practical Solutions to Identification and Signature Problems
3. Rafael Pass : Theory of Computing : Lecture 18: Zero-Knowledge Proofs