EÖTVÖS LORÁND TUDOMÁNYEGYETEM

TERMÉSZETTUDOMÁNYI KAR

MSC SZAKDOLGOZAT

Kriptográfiai protokollok

Vak aláírás

Készítette: Szabó Ágnes

Alkalmazott Analízis és Számításmatematikai Tanszék

Témavezető: Villányi Viktória Ildikó, adjunktus

Operációkutatási Tanszék

Budapest, 2016.

Tartalomjegyzék

Köszönetnyilvánítás 3

Bevezető 4

1. Elméleti áttekintés 5

1.1. Digitális aláírás 5

1.2. RSA algoritmus 8

2. Vak aláírás (Blind signature) 13

2.1. A vak aláírás. A vak aláírás alapötlete. 13

2.1.1. Függvények bevezetése 14

2.1.2. Vak aláírás protokollja 14

2.1.3. A vak aláírás egy példán. 15

2.2. Vak RSA aláírás 17

2.2.1. A vak RSA algoritmus gyengesége 17

2.2.2. A vak RSA algoritmus biztonsága 19

2.3. Proxy vak aláírás (Proxy Blind Signature) 21

2.3.1. Nem védett proxy 22

2.3.2. Védett proxy 23

2.3.3. Tulajdonságai és biztonsági analízise 24

2.4. Vakítás a váratlan aláírásokra 27

2.4.1. A faktorizáció elrejtése fizikai biztonsággal 28

2.4.2. Lineáris kivágás és választás (cut-and-choose) a modulokra és a generátorokra. 28

2.5. Vak Schnorr aláírás 30

2.5.1. Véletlen orákulum modell (Random Oracle modell) 30

3. Elektronikus pénz 32

3.1. Követhetetlen elektronikus pénz 33

3.2. Követhetetlen pénzek protokollja 34

3.3. E-pénzek a gyakorlatban. NFC alapú fizetések 36

3.4. Brands protokollja az e-pénzekre. 38

Összefoglaló 42

Irodalomjegyzék 43

Köszönetnyilvánítás

Ezúton szeretnék köszönetet mondani témavezetőmnek, Villányi Viktória Ildikónak, aki rengeteg időt és energiát fordított rám. Türelmével és észrevételeivel végig segített és támogatott engem. Segítsége nélkül nem jöhetett volna létre a dolgozat.

Külön hálával tartozom családomnak és barátaimnak a sok bátorításért és támogatásért, amit egész tanulmányaim során kaptam tőlük.

Bevezető

A dolgozat középpontjában a vak aláírási séma áll. Ez egy olyan aláírási séma, mely segítségével dokumentumokat tudunk hitelesíteni úgy, hogy a hitelesítés közben megóvja azokat az információkat, melyeket nem szeretnénk nyilvánosságra hozni.

Erre egy jó példát David Chaum fogalmazott meg: ez olyan, mintha papír alapon szavazni szeretnénk. Ha bejelöltük a voksunkat a lapon, akkor érvényesíteni kell a szavazólapot, ezért valahogyan el kell „vakítanunk" az információnkat, hogy a hitelesítő ne lássa azt.

Egy másik példa erre a digitális pénzen alapuló rendszerek. Mivel egyre jobban fejlődik a kereskedelem az interneten, ezért egyre jobban oda kell figyelnünk az „információink" biztonságára. Viszont minél jobban fejlődik a technika, annál több biztonsági követelményt kell betartanunk. A szakdolgozat az olyan vak aláírási algoritmusokkal foglalkozik, melyek segítségével banki tranzakciókat tudunk véghezvinni, valamint elektronikus pénzt tudunk generálni, úgy, hogy a megfelelő biztonsági követelmények betartásával meg tudjuk óvni a tulajdonunkat.

A dolgozat felépítése a következő: az első részben a digitális aláírás fogalmával és az RSA algoritmussal foglalkozunk. Kitérünk arra is, hogy milyen biztonsági követelményeket kell betartanunk ahhoz, hogy ne legyen a hamisítónak könnyű dolga.

A második részben a vak aláírásokkal foglalkozunk. Lesz példa arra, hogy milyen algoritmusokból indult a vak aláírás elmélete és, hogy most mennyire fejlődött tovább. Itt is megfogalmazzuk a biztonsági követelményeket és arra is lesz példa, hogy milyen támadások jöhetnek szóba egy ilyen rendszer feltörésénél.

Az utolsó részben az elektronikus pénzről lesz szó. Megnézzük, hogy hogyan tudunk generálni ilyen pénzeket, hogyan tudjuk azt elkölteni és, hogy a bank hogy tudja nyomon követni.

1. Elméleti áttekintés

Mielőtt rátérnénk a vak aláírásokra, típusaira és azok protokolljaira, először nézzük meg annak alapjait, vezessük be a megfelelő jelöléseket és nézzünk meg néhány lényeges fogalmat.

1.1.Digitális aláírás

Mindennapi életünkben egyre fontosabbá válnak a digitális aláírások. Egyre nagyobb területen lehet már hivatalos okiratokat benyújtani, tárolni, hitelesíteni. Ilyenek például az adóbevallások, elektronikus számlázás, elektronikus fizetőrendszer, hatósági illetve ügyvédi eljárások. Nagyon sok információt kell begyűjteni és tárolni a tranzakcióknál, melyeket titokban kell tartani, másrészről egy anonim fizetési rendszer - mint például a bank jegyei és érméi – kialakítása fontos és problémás terület. Ezekben az esetekben használjuk a digitális aláírásokat.

Az ilyen üzenetek csatornán való átküldésének módosítása ellen illetve az üzenet küldője hitelességének megvizsgálására léteznek úgynevezett üzenethitelesítő kódok. Viszont ezeknek a kódoknak az a hibájuk, hogy a szolgáltatások csak a vevő részére adottak. A vevő ebben az esetben egy harmadik felet nem tud meggyőzni arról, hogy az üzenet sértetlen és a küldőtől származik, mivel az üzenethitelesítő kód értéke egy olyan kulcstól függ, amit a küldő és a vevő is ismer. Vagyis a harmadik fél nem tudja megállapítani, hogy az üzenetet a küldő vagy a vevő generálta, a küldő letagadhatja az üzenetet és a vevő ezt nem tudja bizonyítani. Ennek elkerülése érdekében egy olyan mechanizmust kell megvalósítani, ahol az üzenet küldője állíthatja elő az üzenet hitelességére vonatkozó bizonyítékot, viszont mindenki tudja ezt ellenőrizni. Ezt a mechanizmust nevezzük digitális aláírásnak.

A digitális aláírás és a hagyományos aláírás közti különbség az, hogy a digitális nem papíralapú és biztosítja a következő tulajdonságokat:

1. könnyen generálható
2. ne legyen az egyik dokumentumról a másikra áthelyezhető, hamisítható
3. csak a tulajdonosa generálhassa
4. hitelességét bárki ellenőrizhesse

Ha meg szeretnénk ezeket a tulajdonságokat tartani úgy, hogy egy másik féllel nem szeretnénk tudatni, hogy mit ír alá, akkor a digitális aláírás egy módosított változatához

a *vak aláírási technikához* folyamodunk. A dolgozat erről az aláírás típusról fog majd szólni.[1]

A digitális aláírás sémák alapja a nyilvános kulcsú kriptográfia. Két algoritmusból tevődik össze: az első, amely az aláírást generálja, ami az aláíró privát kulcsát, a másik pedig ellenőrzi azt és az aláíró fél nyilvános kulcsát használja. Vagyis legyen:

S_A – az A aláíró privát kulcsával paraméterezett algoritmus bemenete m aláírni kívánt üzenet, kimenete pedig:

$\sigma = S_A(m)$ – digitális aláírás,

V_A – ellenőrző algoritmus, melynek bemenete egy m üzenet és egy σ aláírás és kimenete pedig igaz vagy hamis érték, attól függően, hogy σ teljesítette-e az egyenlőséget.[2]

Legelőször nézzük meg, hogy milyen volt kezdetekben a digitális aláírás elmélete, a mostani protokollokhoz képest mennyire egyszerűbbek voltak ezek.

Nyilvános kulcsú titkosító algoritmus felhasználásával készíthetünk digitális aláírást. Tegyük fel, hogy az A felhasználó szeretne B felhasználónak m rejtett üzenetet küldeni úgy, hogy az aláírását is elhelyezi benne. Ezt megteheti, hogy egy dekódolási lépést hajt végre a titkos kulcsával:

$D_A(m)$ (ezt csak az A képes előállítani).

A B felhasználó titkosítja az üzenetet az A felhasználó nyilvános kulcsával a következő lépéssel:

$$m = E_A(D_A(m)).$$ [3]

Ezekkel a lépésekkel teljesülnek a következők:

1. Az aláírás hitelessége ellenőrizhető, vagyis a B felhasználó meg tud győződni arról, hogy a küldő valójában az A felhasználó volt-e (az A nyilvános kulcsával).
2. Mivel csak az A ismeri a megfelelő titkos kulcsot, így az aláírás nem hamisítható.
3. Az aláírást nem átvihető egy másik dokumentumra.
4. az aláírt dokumentum nem változtatható meg

[1] Folláth J., Huszti A., Pethő A. Informatikai biztonság és kriptográfia, jegyzet, 2010, 162-163 o.
[2] Buttyán L., Vajda I., Kriptográfia és alkalmazásai, Typotex kiadó, Budapest, 2004,158-162 o
[3] Győrfi L., Győri S., Vajda I., Információ- és kódelmélet, Typotex Kiadó, Budapest, 2002, 352 o.

5. az aláírás letagadhatatlan, vagyis ha a B felhasználó egy harmadik személynek be akarja bizonyítani, hogy az üzenetet A felhasználó küldte, akkor az A felhasználó segítsége nélkül is megteheti azt.

Viszont itt az aláírást két ponton tovább érdemes finomítani:

- lenyomatkészítés
- időpecsét alkalmazása

A l*enyomatkészítés* lényege, hogy nem magára az eredeti dokumentumra tesszük az aláírást, hanem annak egy lenyomatára, melyet egy hash függvény (egy nyilvános egyirányú dimenziószűkítő függvény) segítségével végzünk el. Ekkor az aláírt üzenet:

$$[m, D_A(hash(m))].$$

Ez a lépés azért jó, mert tegyük fel, hogy a támadó *m* megfigyelésével ki tudja számítani a *hash(m)*-et, viszont úgy lehet sikeres a támadás, hogy egy olyan *m'* üzenetet talál, amivel a támadó célját eléri: $hash(m) = hash(m')$ és $[m', D_A(hash(m))]$ is hiteles üzenet a B felhasználónak. Viszont egy hash függvény tulajdonsága közzé tartozik, hogy nagyon nehéz hash értékre visszavezethető *m'* üzenetet találni (a skatulya-elv elv alapján lehetséges találni ilyen értékeket, viszont ennek elhanyagolható a valószínűsége), valamint az aláírásra nézve biztosítja, hogy rövid legyen.

Tegyük fel, hogy A felhasználó az aláírásával együtt küldött egy szerződést B felhasználónak, majd meggondolta magát és letagadja, hogy ő bármit is küldött, arra hivatkozik, hogy kitudódott a kulcsa. Ilyen esetek miatt fontos, az *időpecsét* alkalmazása, hogy az aláírás letagadhatatlan legyen.

Ahhoz, hogy ne tudja A felhasználó letagadni az aláírást, nem elég az időpont, de szükség van egy megbízható harmadik félre (G). Legyen $S_A(m)$ az A felhasználó által aláírt *m* dokumentum, $V_A(m)$ jelölje az A által aláírt $m = S_A(m)$ dokumentum ellenőrzését az aláírás hitelessége szempontjából. Vizsgáljuk a következő protokollt:

1.	$A \rightarrow G$:	$U = S_A(I, S_A(m)), (I, S_A(m))$
2.	G:	$V_A(U)$,
3.	$G \rightarrow A$:	$W = S_G(T,I, S_A(m))$
	$G \rightarrow B$:	$W = S_G(T,I, S_A(m))$
4.	A:	$V_G(W)$
5.	B:	$V_G(W),I,V_A(S_A(m))$

Az első lépésben A aláírja a dokumentumot, majd kiegészíti egy azonosító fejléccel (*I* – azt jelöli konkrétan, hogy A szeretne B felhasználónak egy aláírt dokumentumot küldeni), majd újra aláírja az eredményt. A második lépésben a harmadik fél (G) ellenőrzi a kapott eredményt az első lépésből, majd kiegészíti az *m* dokumentumot és az *I* azonosító információt egy *T* időpecséttel. Ezt az eredményt aláírja és elküldi mindkét felhasználónak (harmadik lépés). A negyedik lépésben az A felhasználó ellenőrzi a G-től érkezett üzenetet és ha nem ő küldte előzőleg az *I* azonosítójú aláírt dokumentumot, akkor jelzi, hogy kulcsával visszaéltek. Az utolsó lépésben B felhasználó ellenőrzi a G aláírását, az azonosító információt és az A felhasználó aláírását (A felhasználó nyilvános kulcsával kell $V_A(S_A(m))$ vertifikációs lépést végrehajtani).

Megjegyzés: Az utolsó lépésben az *I* azonosító információból derül ki B felhasználó számára, hogy aláírt dokumentumot kapott A felhasználótól.[4]

1.2.RSA algoritmus

Most vizsgáljuk meg részletesen az RSA algoritmusát, mivel ez az alapja a digitális aláírásoknak és a dolgozatban vizsgált protokolloknak.

Az RSA a legelterjedtebb nyilvános kulcsú kriptográfiai algoritmus, mely moduláris hatványozással történik, vagyis kódoláskor az aláírandó üzenetet a kulcs által meghatározott hatványra emeljük, majd a kapott eredményt modulussal elosztjuk és a kapott maradékkal számolunk tovább. Ezt az algoritmust Ronald Rivest, Adi Shamir és Len Adleman fejlesztette ki, mely ma a legelterjedtebb algoritmus.[5]

Az RSA algoritmus három részből tevődik össze, a részletesség kedvéért ezeket az algoritmusokat külön-külön vizsgáljuk: *kulcsválasztás, kódolás* és *dekódolás.*

1. **Kulcsválasztás**, melyet a rendszer minden felhasználója elvégzi, lépései a következők:

- Véletlenszerűen választunk két nagy, nem azonos, prímszámot: p, q (legalább 500 bit bináris méretű legyen).
- Legyen $N = p \cdot q$ és $\varphi(N) = (p-1)(q-1)$, ahol a φ-t **Euler függvény**nek nevezzük.

[4] Györfi L., Győri S., Vajda I., Információ- és kódelmélet, Typotex Kiadó, 2002, 351-354 o.
[5] en. wikipedia.org

- Válasszunk egy $e > 0$ számot, mely $\varphi(N)$-hez relatív prím legyen, vagyis teljesüljön, hogy:

$$(\varphi(N), e) = 1 \qquad (1)$$

- Akkor pontosan egy olyan $1 < d < \varphi(N)$ természetes szám létezik, amely az e inverze:

$$d = e^{-1} \pmod{\varphi(N)} \qquad (2)$$

vagyis:

$$ed = 1 \pmod{\varphi(N)}$$

Az e lesz a nyilvános, a d pedig a titkos exponens. Az (N,e) páros publikus kulcs, mindenki számára elérhető és ezzel titkosítják az üzenetet. Az (N, d) pedig a dekódoláshoz használt páros, mely a titkos kulcs. [6]

2. **Kódolás**.

Ha az A felhasználó üzenetet akar B felhasználónak küldeni, akkor megkeresi a B felhasználó nyilvános kulcsát, az e és N számokat. Ezzel kódolja az üzenetet, így a titkosított üzenet a következő lesz:

$$C = m^e \pmod{N} \quad (3)$$

ahol m az üzenet. B felhasználó megkapja a rejtjeles üzenetet, ami 0 és $N - 1$ közötti, $m_1, m_2, \ldots$ számok sorozata. Ekkor bármely $m < N$ egész számra igaz, hogy:

$$(m^e)^d = m \pmod{N}$$

3. Dekódolás

A B felhasználó dekódolja a megkapott rejtett üzenetet, így a dekódolt szöveg a következő:

$$m = C^d \pmod{N} \quad (4)$$

Például ha B felhasználó alá szeretné írni az m dokumentumot, akkor a következőképpen számítja ki az s aláírást:

[6] Buttyán L., Vajda I., Kriptográfia és alkalmazásai, Typotex Kiadó, Budapest, 2004, 80-81 o.

$$s = m^d \pmod N.$$

A felhasználó megkapja az *m* dokumentumot, az *s* aláírást és a B felhasználó *(N, e)* nyilvános kulcsát. A következő ellenőrző képlettel tudja megállapítani, hogy ezek valóban összetartoznak-e:

$$s^e = m \pmod N.$$

Lemma: a *(3) és (4) formulákkal megadott műveletek egymás inverzei.*

Bizonyítás:

Mivel az (1)-el jelölt formula ekvivalens a $d \cdot e = k \cdot \varphi(N) + 1$ –el valamely *k* egészre, ezért:

$$C^d = (m^e)^d = m^{ed} = m^{k\varphi(N)+1} \pmod N$$

Most vizsgáljuk a következő két esetet:

1. Ha $(m, N) = 1$, akkor az **általánosított Fermat-tétel** ($a^{(p-1)(q-1)} = 1 \pmod{pq}$, ahol *p,q* prímek, $(a,p,q) = 1$) felhasználásával

$$C^d = m^{k\varphi(N)+1} = m(m^{\varphi(N)})^k = m \pmod N,$$

azaz megkaptuk a kívánt eredményt.

2. $(m, N) > 1$ esetén mivel $N = p \cdot q$ és $m < N$, ezért két eset lehetséges: vagy $p|m$ vagy $q|m$. Feltehetjük bármilyen általánosság korlátozása nélkül, hogy $p|m$, így $m = w \cdot p$ felbontás kapható, ahol $(w,N) = 1$.

Ennek felhasználásával:

$$m^{k\varphi(N)+1} = w^{k\varphi(N)+1} p^{k\varphi(N)+1} \pmod N \quad (7).$$

Mivel $(w,N) = 1$, ezért a (7) formula jobb oldalára az általánosított Fermat-tételt alkalmazva kapjuk a következőt:

$$w^{k\varphi(N)+1} = w \pmod N. \quad (8)$$

A **Fermat-tétel** (*Ha c egész szám nem osztható p prímmel, akkor:* $c^{p-1} = 1$) segítségével:

$$p^{k\varphi(N)+1} = p\,(p^{k\varphi(N)}) = p\,(p^{(q-1)})^{k(p-1)} = p \pmod q \quad (9)$$

illetve nyilvánvalóan:

$$p^{k\varphi(N)+1} = 0 = p \pmod p,$$

Így mivel $p|p^{k\varphi(N)+1} - p$ és a (9) formula alapján $q|p^{k\varphi(N)+1} - p$ teljesülnek a $p \neq q$ prímekre, ezért $pq|p^{k\varphi(N)+1} - p$ is fennáll. Vagyis:

$$p^{k\varphi(N)+1} = p \pmod N. \quad (10)$$

Felhasználva a (8) és (10) eredményeket a (7) képletbe helyettesítve kapjuk a bizonyítás eredményét:

$$m^{k\varphi(N)+1} = wp = m \pmod N$$[7]

Néhány **megjegyzést** kell tennünk az RSA jól működésének feltételeiről:

1. A kulcshosszat olyan méretűre kell választanunk, mint az N modulus mérete. Például, ha 2048 bites RSA-ról van szó, akkor m = 2048 bit, p és q pedig két 1024 bites prím. A p és q megválasztásánál két szempontot kell figyelembe venni:

- különbségük is nagy legyen, ugyanis a Fermat faktorizációs algoritmus gyorsan megtalálja az értékeket ellenkező esetben.
- Az e értékét kicsinek kell választani, a d viszont legalább 2048 bites legyen.

2. A prímszámok választására vannak különböző prímtesztek (Például: Rabin-Miller teszt, AKS prímteszt). A p és q értékekre a kulcsgenerálás után már nincs szükség, viszont titokban kell tartani, mert ha a támadóhoz jut, akkor könnyen meghatározhatja a d titkos kulcsot.

3. Az RSA szimmetrikus kulcsú titkosítás, így a titkosítás – dekódolás, aláírás – aláírás ellenőrzése ugyanaz a művelet:

$$(m^e)^d = (m^d)^e \pmod N.$$

Az RSA aszimmetrikus kulcsú titkosítás, mivel a két kulcs e és d nem egyezik meg.

4. Legtöbb esetben az e kulcsra a leggyakrabban választott érték a $2^{16} + 1$. A kiválasztásnál egyetlen szempont, hogy az e relatív prím legyen, ami általában egy rögzített szám. Az e-edik hatványra emelést általában újabb algoritmussal számoljuk (ismételt négyzetreemelés és szorzás), aminek a sebességét az határozza meg, hogy az e bináris alakjában hány 1-es szerepel (például ha e = 65537, annak a bináris alakja: 1000000000000001).[8]

---

[7] Györfi L., Győri S., Vajda I., Információ és kódelmélet, Typotex kiadó, 2002, 328-334 o.
[8] Dr. Berta I. Zs., Nagy e-szignó könyv, Microsec Kft., 2011, 53 o.

5. Az RSA biztonsága azon múlik, hogy milyen gyorsan tudjuk az N számot faktorizálni. Mivel a d csak a $\varphi(N)$ ismeretében tudjuk kiszámítani, aminek a meghatározása olyan nehézségű, mint az N prímtényezőkre bontása. [9][10]

6. A gyakorlatban használt RSA biztonságos változata az RSA-PSS.

[9] Buttyán L., Vajda I., Kriptográfia és alkalmazásai, Typotex kiadó, Budapest, 2004, 83-84 o.
[10] Páles Zs. főszerk., Alkalmazott Matematikai Lapok, 23, 2006, 236 o.

2. Vak aláírás (Blind signature)

Most vizsgáljuk meg részletesen a vak aláírást, amely az alapja a dolgozatban szereplő protokolloknak.

A digitális aláírásokat két típusba sorolhatjuk. Az első, amikor egy A felhasználó pontosan tudja, hogy mit ír alá, ő írja alá, és ő küldi az üzenetet. A másik eset pedig, amikor egy A felhasználóval alá kell íratnunk egy dokumentumot, úgy, hogy nem tudja mi van az üzenetben. Az ilyen aláírásokat nevezzük *vak aláírásoknak*. Ilyen aláírást használnak szavazásoknál, amikor úgy kell hitelesíteni a dokumentumot, hogy nem tudjuk, hogy a szavazó kire rakta fel a voksát vagy például a bankoknál a pénzhitelesítés, amikor a pénzt úgy kell hitelesíteni, hogy annak sorszámát titokban kell tartani. A vak aláírás a kriptográfia egy új ága. A következőkben ezzel fogunk foglalkozni, hogy milyen részei vannak, hogyan használatosak, milyen tulajdonságai vannak. Egy példán keresztül vizsgáljuk meg folyamatát, használatát.

2.1.A vak aláírás. A vak aláírás alapötlete.[11]

Tegyük fel, hogy egy felhasználó, aki legyen egy szavazó is akár, szavazatát érvényesíteni szeretné. A következő lépések azt illusztrálják, hogy papír alapon ahogyan működne a vak aláírás algoritmusa:

1. egy szavazó bejelöli voksát a szavazócédulán, majd beleteszi egy átlátszatlan, indigós borítékba
2. a borítékot beleteszi egy külső borítékba, melyre ráírja a hitelesítő címét és a saját címét, amire hitelesítés után szeretné vissza kapni a levelet, majd elpostázza a hitelesítőnek
3. a hitelesítő, amikor megkapja felnyitja a borítékot, majd az indigós borítékot aláírja, úgy, hogy azt nem nyitja ki
4. miután aláírta, beleteszi az indigós borítékot egy másik borítékba, majd visszaküldi a kapott borítékon szereplő címre
5. mikor a szavazó megkapta a borítékot, akkor leellenőrzi az aláírást. Kiveszi az indigós borítékból a szavazati lapot, melyen látszania kell az aláírásnak.
6. ha mindent rendben talál, akkor a szavazati cédulát egy új borítékba teszi és visszaküldi a hitelesítőnek, úgy, hogy a borítékra már nem írja rá a saját címét, hogy ne tudják beazonosítani őt.

[11]D. Chaum, Blind signatures for untraceable payments, Advances in Cryptology, 1983, 200-201 o.

7. a hitelesítő megkapja a szavazatot, leellenőrzi, hogy rajta van-e a saját aláírása és minden hiteles szavazatot feltölt egy mindenki számára elérhető helyre. Innen a szavazó csak akkor tudja meg, hogy melyik az ő voksa, ha esetleg a dokumentumon hagyott valamilyen kis nyomot.

A vak aláírás azért kapta ezt a nevet, mert a szavazó egy „vakított” üzenetet küld a hitelesítőnek, aki szintúgy „vakon” írja alá a dokumentumot. A különbség a vak aláírás és a digitális aláírás között, hogy a digitális aláírásnál az aláírás pillanatában az aláíró fél tisztában van a dokumentum tartalmával. Viszont az aláírás ellenőrzése mindkét esetben egyformán történik, tehát hiába az első esetben aláírásnál „vak” volt a hitelesítő, az ellenőrzésnél már megtekintheti a dokumentum tartalmát. Vak aláírásnál is az aláíró fél rendelkezik titkos és nyilvános kulccsal. A vak aláírással aláírt dokumentumot, az aláírás érvényességét a nyilvános kulcs segítségével bárki ellenőrizheti.[12]

2.1.1. Függvények bevezetése

Vezessük be a következő három függvényt, mely segítségével felvázolhatjuk a vak aláíró sémát:

1. Legyen *s'* az aláíró függvény, melyet csak az aláíró ismer, és legyen az inverze *s* függvény, mely nyilvános. Ebből kifolyólag:

$$m = s(s'(m))$$

2. Legyen egy *c* „ingázó” vagy „vakító” függvény és inverze *c'*, melyeket csak a szolgáltató ismer, és érvényes rá a következő összefüggés:

$$c'(s'(c(m))) = s'(m)$$

3. Végül legyen *r* egy ellenőrző függvény legyen, mely ellenőrzi az aláírás érvényességét.

2.1.2. Vak aláírás protokollja

A vak aláírás protokollja hasonló a magyarázó példa folyamatában leírtakkal.

1. A szavazó választ egy *m* üzenetet, aláírja vagyis vakítja – *c(m)* - , majd továbbítja az aláírónak.

[12] Folláth J., Huszti A., Pethő A., Informatikai biztonság és kriptográfia, Kempelen Farkas Tankönyvtár, 2011
http://www.tankonyvtar.hu/

2. A szavazó választ egy random *m* üzenetet (ez lesz *r(m)*), majd vakítja ezt (*c(m)*), és elküldi az aláírónak.

3. Az aláíró aláírja a vakított üzenetet az *s'* felhasználásával, majd ezt az *s'(c(m))* „értéket" visszaküldi a szavazónak.

4. A szavazó ellenőrzi az aláírt dokumentumot a *c'* függvény segítségével és megkapja az aláírt üzenetet:

$$c'(s'(c(m))) = s'(m)$$

5. Bárki megnézheti az aláíró által létrehozott *s'(m)* üzenetet az aláíró nyilvános kulcsával (*s*) és leellenőrizheti az *r(s(s'(m)))* segítségével.

Digitális aláírásnál - bárki ellenőrizheti, hogy az aláírt üzenet - *s'(m)*- az aláíró privát kulcsát, *s'*-t, használja-e.

Vak aláírásnál– az aláíró nem tud semmit a levelezésről az $s'(m_i)$ és $s'(c(m_i))$ között.

Aláírás megőrzésnél – a szolgáltatónak létre kell hozni legfeljebb egy megfosztott aláírást minden aláíró általi aláírt üzenetre (azaz $s'(c(m_1))\ldots s'(c(m_n))$ választott *c, c'* és m_i mellett kivitelezhetetlen, hogy létezik egy *s'(y)*, hogy *r(y) és* $y \neq m_i$). Ez egy szokásos kriptográfiai művelet, amely kizárja azt, hogy ugyanazt a véletlen számot generáljuk.

2.1.3. A vak aláírás egy példán.

Lenyomozhatatlan fizetési rendszer

Most vizsgáljuk meg egy példán a vak aláírás működését egy fizetési tranzakción keresztül és hogyan alakít ki lenyomozhatatlan fizető rendszereket. Legyenek a szereplők: egy bank, egy fizető és egy kedvezményezett. A banki tranzakciónak a részletes lépései a következők lesznek:

1. A fizető választ egy *m* üzenetet és a *c* „vakító" függvény segítségével „elvakítja" az üzenetet.
2. A fizető továbbítja a *c(m)* üzenetet a banknak.
3. A bank aláírja ezt, *s'(c(m))*, és leterheli a fizető számláját.
4. A bank visszaküldi az aláírt *s'(c(m))* üzenetet a fizetőnek.
5. A fizető leveszi a „vakítást", így megkapja az aláírt üzenetet:

$$c'(s'(c(m))) = s'(m).$$

6. A fizető leellenőrzi az *s* nyilvános kulcs segítségével a következő műveletet:

$$s(s'(m)) = m.$$

Ha ez az egyenlőség hamis, akkor megszakítja a tranzakciót.

7. Ha az egyenlőség fennáll, akkor folytatja a tranzakciót a kedvezményezettnek.

8. A kedvezményezett leellenőrzi az ellenőrző *r* függvénnyel az $r(s(s'(m)))$ tranzakciót és megszakítja, ha hibás.

9. A kedvezményezett visszajelez a banknak.

10. A bank leellenőrzi az $r(s(s'(m)))$ formát és megszakítja, ha hamis.

11. A bank hozzáadja a törölt megjegyzések listájához tranzakciót és megáll, ha az már a listán van.

12. A bank jóváírja a kedvezményezett számlát.

13. A bank tájékoztatja a kedvezményezettet az elfogadásról.[13]

[13] D. Chaum, Blind signatures for untraceable payments, Advances in Cryptology, 1983, 201-203 o.

2.2. Vak RSA aláírás

Ez a protokoll egyesíti a Chaum vak aláírási protokollját és az RSA algoritmust, melyet a fentiekben bemutattunk. Ez az algoritmus az első elektronikus pénz rendszerének az alapja.

Hasonlóan az RSA algoritmuséhoz először bontsuk fel a sémát három részre: *kulcsgeneráló, aláíró* és *ellenőrző* algoritmusra, és vizsgáljunk meg őket részletesen külön-külön.

A **kulcsgenerálás** minden lépése teljesen megegyezik az eredeti RSA algoritmusáéval. Röviden összefoglalva:

- választunk két nagy prímet: p,q.
- kiszámoljuk az $N = p \cdot q$ és $\varphi(N) = (p-1)\cdot(q-1)$ értékeket
- választunk $\varphi(N)$-hez egy $e > 0$ számot, melyhez létezik egy d érték, ami az e inverze.
- ezekből (N, e) lesz majd a publikus, (N, d) pár pedig a titkos kulcs.

Az **aláíró algoritmus** legyen a következő: vegyünk egy tetszőleges m üzenetet, egy d titkos kulcsot, amiből kiszámítunk egy s aláírást, vagyis:

$$sign_d(m) = s,$$

ahol

$$s = m^d (mod\ N).$$

Ezután (m, s) pár segítségével **leellenőrizzük** az aláírás érvényességét. Ha teljesül az $m = s^e\ (mod\ N)$ egyenlőség, akkor az aláírásunk érvényes.[14]

2.2.1. A vak RSA algoritmus gyengesége

A fenti algoritmus lenne a legkézenfekvőbb és legegyszerűbb, viszont az alap RSA algoritmusnak több gyengesége is van, így ez a kombinált algoritmus is „megörökölte" tőle. Az egyik, hogy *alakítható*, vagyis, ha a támadó rendelkezésére áll (m_1, s_1) és (m_2, s_2) aláírt üzenetek, akkor új aláírt üzeneteket tud kialakítani. Ugyanis: $m_3 = m_1 m_2\ (mod\ N)$ üzenet aláírása: $s_3 = s_1 s_2\ (mod\ N)$, hiszen

$$s_3 = m_1^d\ m_2^d = (m_1 m_2)^d\ (mod\ N).$$

[14]Esti Rahmawati Agustina and Is Esti Firmanesa, Implementation of RSA Blind Signature on CryptO-0N2 Protocol, International Journal of Computer , 2013, pp 886-888.

Másik hamisítás az *egzisztenciális hamisítás*, például a támadó választ egy *s* értéket, kiszámítja $m = s^e \pmod N$ értéket, ahol e és n nyilvánosak. Így a *d* ismerete nélkül generál egy *(m, s)* aláírt üzenetet, bár ez a támadó által nem szabályozható, bármilyen bitsorozat lehet.

Ezek a támadások elkerülése érdekében bevezették a kriptográfiai hash függvények használatát az ilyen aláírások generálásához.[15]

Az algoritmus javításához először fogalmazzuk meg a hash függvények definícióját: a hash függvények olyan eljárások, melyekkel bármilyen hosszúságú adatot adott hosszúságra képezhetünk le:

$$h: \{0,1\}^* \rightarrow \{0,1\}^N.$$ [16]

Jelen esetünkben egy tetszőleges méretű üzenetre egy hash függvényt alkalmazunk, melynek eredményeként egy fix méretű hash értéket (üzenetkivonatot) kapunk.

Most nézzük, hogy az előző algoritmusban hol kell használnunk a hash függvényt.

Annak érdekében, hogy az üzenet aláírását elvégezzük a következőket kell végrehajtanunk:

1. a felhasználó egy véletlen *c (mod N)* értékkel vakítja az *m'* üzenetet, vagyis a vakított üzenet a következő lesz:

$$m = h(m')c^e \pmod N$$

2. a felhasználó ezt elküldi a banknak például, aki kiszámolja az aláírást:

$$s = m^d \pmod N$$

3. a felhasználó miután megkapta az aláírást, leveszi a „vakítást" az üzenetről, vagyis megszorozza multiplikatív inverzével és így megkapja az eredeti m' üzenetet

4. a kapott *s* aláírásból kiszámítja az eredeti üzenet aláírását a következő kongruenciával:

$$s' = sc^{-1} \pmod N$$

5. az *(m', s')* üzenet és az aláírás segítségével az ellenőrzés során kiszámítja az ellenőrző fél a *h(m')* és $(s')^e \pmod N$ értékeket és leellenőrzi, hogy kongruensek-e.

Ha a fenti lépések szabályosak, akkor a következők teljesülnek:

---

[15] Folláth J., Huszti A., Pethő A., Informatikai biztonság és kriptográfia, Kempelen Farkas Tankönyvtár, 2011
http://www.tankonyvtar.hu/

[16] en.wikipedia.org

$(s')^e = (sc^{-1})^e = (m^d c^{-1})^e = m^{de} c^{-e} = mc^{-e} = b^e h(m')b^{-e} = h(m') \pmod N$

2.2.2. A vak RSA algoritmus biztonsága

Ahogy fent a bevezető részben említettük, az RSA paramétereinek generálásánál sok mindenre kell odafigyelnünk, különben a támadónak egyszerű dolga lesz a hamisítás során.

Véletlenszám-generálás biztonsági kockázatai[17]

Az egyik legalapvetőbb hibának számít a biztonságban az, ha nem megfelelően generáljuk az értékeket. Sok esetben nem magában a protokollban van a hiba, hanem a kulcs generálását nem megfelelően végezzük, vagyis ha nem véletlen számokkal generáljuk kulcsunkat nagyobb esély van arra, hogy gyorsabban feltörjük a sémát.

Digitális algoritmusokkal csakis pseudovéletlen bitsorozatot lehet generálni. Ezek azért áll-véletlen számok, mert megközelítik a véletlen számok jellemzőit. A véletlen bitsorozatok a pseudovéletlen bitsorozatoktól polinom időben nem megkülönböztethetők, viszont nem teljesen azok.

A tökéletes véletlen számsorozatok használatakor az is egy problémát jelent, hogy a felhasznált bitsorozatokon az üzenet küldőjének és fogadójának meg kellene egyeznie mielőtt elkezdik használni az adott sémát. A véletlen bitsorozatokat a protokoll mindkét résztvevőjének ismernie kellene, tehát ezeket a bitsorozatokat hatalmas kapacitású számítógépeken kell biztonságosan tárolni, hogy a protokoll ismételt alkalmazásakor rendelkezésre álljanak.

Egyik ismert támadást fizikai eszközök segítségével valósították meg, miszerint egy fizikai zajgenerátor jelét mintavételezik és annak digitalizált eredményét tekintették véletlen számnak. Az elektromágnesen tér rontja a véletlenszámsorok statisztikai jellemzőit, így nagyobb valószínűséggel sikeres a támadás.

Prímszámokkal kapcsolatos biztonsági kockázatok

A kulcsgenerálási algoritmus megköveteli, hogy a prímszámokat, amelyeket választunk azok nagy és véletlenül választottak legyenek. A biztonság kedvéért

[17] Kadlecsik J., Az informatikai hálózati infrastruktúra biztonsági kockázatai és kontrolljai, 2004, 269 o. http://www.cert.hu/sites/default/files/MTA1_print.pdf

olyannak kell lenniük, hogy semmiféle prímszám táblázatban ne legyen rögzítve, ne lehessen tárolni ezeket, mivel egy *prímtényezős felbontásos támadás* során ezek gyorsan megkereshetők. Olyan prímteszteket alkalmaznak manapság, melynek segítségével gyorsan találhatunk megfelelő prímszámokat.[18] Például:

1. *Fermat-teszt*:

- válasszunk egy véletlen a egész számot az [1, m) intervallumból
- ha $a^{m-1} = 1 \pmod m$, akkor m valószínűleg prím, különben összetett szám

2. *Rabin-Miller-teszt:*

- írjuk fel $(m - 1)$-et $2^k n$ alakban, ahol n páratlan szám
- az $[1, m)$ intervallumból válasszunk egy véletlen egész a értéket
- ha az $a^n - 1$, $a^n + 1$, $a^{2n} + 1$, … , $a^{2^{k-1}n} + 1$ szám egyike sem osztható m-mel, akkor m összetett, ellenkező esetben m valószínűleg prím[19]

[18]Kadlecsik J., Az informatikai hálózati infrastruktúra biztonsági kockázatai és kontrolljai, 2004, 269 o. http://www.cert.hu/sites/default/files/MTA1_print.pdf

[19] Rónyai L., Ivanyos G., Szabó R., Algoritmusok, Typotex Kiadó, 2005, 317-319 o.

2.3. Proxy vak aláírás (Proxy Blind Signature)[20]

A proxy vak aláírások a csoport aláírások közé sorolhatók. Ez abban különbözik az eddigiektől, hogy nem maga a felhasználó ír alá, hanem egy megbízottja, vagyis a proxy aláíró. A csoport aláírásoknál a csoport tagja, aki a csoport nevében ír alá bizonyítani tudja, hogy a csoporthoz tartozik, viszont személyazonossága nem derül ki. A nyilvános kulcs segítségével pedig ellenőrizni tudjuk az aláírás érvényességét.[21]

A vak aláírás biztosítja az e-pénzek anonimitását. Két fő tulajdonsággal rendelkeznek ezek a pénzek: követhetetlenek és összeköthetetlenek. Amikor egy felhasználó engedélyezni akarja egy ilyen érme küldését, teljesítenie kell a vak aláíró rendszer követelményeit. A következőkben azt vizsgáljuk, hogy egy ilyen séma milyen feltételek mellett és hogyan képes működni.

Az érmék előállításánál a felhasználó készít egy vak aláírást a banknak a vak aláíró séma segítségével. Ennek a birtokosa csak a felhasználó, melyet a bank nem tud felismerni. Bármikor, ha a felhasználó vissza szeretné vonni az érméjét, akkor szüksége van előállítani egy proxy vak aláírást az aláíró bank nevében. Vagyis a proxy aláíró egy megbízott, aki az eredeti aláíró nevében ír alá.

Ebben a protokollban egy eredeti aláíró elküldi az eredeti aláírását egy másik, proxy, aláírónak, úgy, hogy a proxy aláíró alá tudjon írni minden üzenetet az eredeti aláíró nevében és a hitelesítő ellenőrizni tudja és különbséget tudjon tenni az eredeti aláírás és a proxy aláírás között. Erre két típusú séma van kidolgozva, amelyek biztosítják a letagadhatatlanságot és a hamisítatlanságot:

1. *nem védett proxy*: a proxy és az eredeti aláíró együttesen tudnak egy érvényes proxy aláírást generálni.
2. *védett proxy*: csak a proxy tud generálni egy érvényes proxy aláírást.

A proxy vak aláírás bemutatása előtt először vezessük be a paramétereket és jelöléseket. Legyen:

p, q: *biztonságos prímeknek* vagy *Sophie German* prímeknek is nevezzük, ami azt jelenti, hogy ha egy k szám prímszám, akkor $2k + 1$ is prím

g: Z_p^* egy eleme, melynek rendje q

[20] Sunder LAl and Amit Kumar Awasthi, Proxy Blind Signature Scheme, Cryptology ePrint Archive, Report 2003/072, ,pp 1-4.

[21] Horváth A., Az elektronikus pénz, mint az elektronikus kereskedelmet támogató speciális fizetési rendszer, Ph.D. értekezés, 2007.

x_A: az eredeti aláíró (A) titkos kulcsa

y_A: az A nyilvános kulcsa, ahol $y_A = g^{x_A} (mod\ p)$

$h(.)$: biztonságos egyirányú hash függvény

Most vizsgáljuk meg külön-külön a proxy aláírás két típusát.

2.3.1. Nem védett proxy

A protokollt három szakaszra bontjuk, melyek a következők:

Proxy szakasz

1. *Proxy generálása*:

Az eredeti A felhasználó véletlenszerűen kiválaszt egy $k \in Z_q^*$ számot, ahol $k \neq 1$ és kiszámolja a következőket:

$$r = g^k (mod\ p),$$

$$s = x_A + kr\ (mod\ q)\ ,$$

$$y_p = g^s\ (mod\ p)$$

2. *Proxy kézbesítése:*

Az eredeti aláíró, A, elküldi a proxy aláírónak – aki legyen B - az *(s, r)* párt egy biztonságos úton, majd a B aláíró y_p nyilvános kulcsot generál belőle.

3. Proxy ellenőrzése:

Miután a B aláíró megkapta az *(s, r)* párt és a nyilvános kulcsot y_p-t, leellenőrzi a titkos kulcs érvényességét a következő kongruenciával:

$$y_p = g^s = y_A r^r\ (mod\ p)$$

Ha *(s, r)* pár teljesíti a feltételt, akkor B aláíró elfogadja, mint érvényes proxyt, ha nem teljesíti, akkor elutasítja és kér egy másik kulcsot vagy megállítja a műveletet.

Aláírási szakasz

1. B aláíró választ a véletlen $K \in Z_q^*$ számot, ahol $K \neq 1$, majd kiszámolja a következőt:

$$R = g^K\ (mod\ p),$$

melyet elküld a C felhasználónak (vevő).

2. C felhasználó kiválaszt véletlenszerűen két számot $\alpha,\beta \in Z_q^*$, majd kiszámolja a következő kongruenciát:

$$r' = R\, g^{-\alpha} y_p^{-\beta} \pmod p$$

Ha $r' = 0$, akkor választ egy másik α és β számot, különben kiszámolja:

$$e' = h(r' \oplus m) \pmod q,$$

$$\text{és } e = e' + \beta \pmod q,$$

majd C felhasználó elküldi az e értéket a B aláírónak.

3. Miután B aláíró az e értéket kiszámolja:

$$s' = K - s\, e \pmod q$$

majd ezt elküldi C-nek.

4. C felhasználó kiszámolja az S_p értéket, ami a következő:

$$S_p = s' - \alpha \pmod q$$

Az (m, S_p, e') hármas lesz a proxy vak aláírás.

Igazolási szakasz

A proxy vak aláírás hitelesítője (vevője) kiszámítja a következő értékeket:

$$e'' = h(g^{Sp} y_p^{e'} \pmod p \oplus m) \pmod q,$$

ahol y_p nyilvános kulcs, melyet az első szakaszban számítottunk ki. Ebben az esetben e'' akkor és csak is akkor egyenlő e' értékkel, ha (m, S_p, e') értékek valós proxy aláírás.

2.3.2. Védett proxy

Ebben az esetben csak a proxy aláíró generál egy é proxy aláírást, vagyis az előző algoritmust a következőképpen kell módosítanunk:

Proxy rész

1. *Proxy generálása:*

Az eredeti aláíró, A, választ egy $k \in Z_q^*$ véletlen számot, ahol $k \neq 1$, majd kiszámolja a következőket:

$$r = g^k \pmod p,$$

$$\sigma = x_A + k\, r \pmod q,$$

$$y_p = g^\sigma\, y_B \pmod p$$

ahol $y_B = g_B^x \pmod p$, ami a B aláíró nyilvános kulcsa lesz.

2. *Proxy kézbesítése:*

Az eredeti aláíró elküldi a (σ, r) párt a proxy aláírónak, vagyis B-nek, valamilyen biztonságos úton.

3. *Proxy ellenőrzése és módosítása:*

Miután megerősítette a B aláíró a (σ, r) párok érvényességét, azután megváltoztatja a proxy kulcsot a következőre:

$$s = \sigma + x_B \pmod q.$$

A többi részek megegyeznek a nem védett proxy algoritmus megfelelő részeivel.

2.3.3. Tulajdonságai és biztonsági analízise[22]

1. Az *aláírás ellenőrzése* részben különböző kongruenciát használunk, hogy leellenőrizzük az eredeti aláírás és a proxy aláírás hitelességét. Tehát az eredeti aláírás megkülönböztethető a proxy aláírástól.

2. Egy valós proxy aláírást, s, megadni kötelező. Lehetetlen érvényes aláírást generálni x_B és s vagy mindkét érték ismerete nélkül. Így a proxy aláírás nem lehet hamis. Tehát az eredeti aláíró létrehozza az s aláírást, és nincs ismeretünk x_B-ről, a proxy védelmi részben. Tehát a proxy aláíró nem tudja letagadni a proxy aláírást, amit ő hozott létre.

3. A nyilvános kulcs - y_p - az eredeti aláíró kulcsa y_A által kerül kiszámításra. Így az eredeti aláíró nem tudja letagadni a szerepét. A proxy aláíró nyilvános kulcsa is részt vesz a nyilvános kulcs generálásában (a proxy védelme rész). Tehát a proxy aláíró az aláírás alapján azonosítható.

[22] Sunder LAl and Amit Kumar Awasthi, Proxy Blind Signature Scheme, Cryptology ePrint Archive, Report 2003/072, pp 3-4.

Egyes biztonsági támadásokat - részletes leírást a cikkben olvashatunk [23] - eltávolítottak a sémából. Ezek a támadások a következők:

Az ellenőrző egyenlet a megadott cikkben:

$$e = h(g^S y_B^{-e} y_A^e u \,||\, m) \pmod q,$$

mely biztosítja az A és B, mindkét aláíró résztvételét és ezért az *(m, u, s, e)* értékek érvényes proxy aláírás a B által az A nevében. Mindkét aláíró nyilvános kulcs bevonása az egyetlen módja annak, hogy elismerjék, hogy a B valóban egy proxy aláírója A aláírónak.

Legyen egy R hamisító vagy vevő bebizonyíthatja, hogy *(m, u, s, e)* egy érvényes proxy vak aláírás valamely F aláírótól, habár F lehet nem adta meg az aláírási jogát mindenkinek. Ekkor a következő történhet:

Amikor a hamisító R kapcsolatba lép „B" aláíróval, akkor

$$u' = (r\, y_A^r)^{-e+b} y_F^{-e} \pmod q$$

kiszámítása helyett a következőt számítja:

$$u = (r\, y_A^r)^{-e+b} y_A^{-e} \pmod q.$$

Viszont nincs már egyenlet, amelyre hatással lenne ez a hamisítás.

Most a hamisító megbizonyosodhat arról, hogy az *(m, u', s, e)* érvényes proxy vak aláírása-e az F aláírónak az ellenőrző egyenlet segítségével:

$$e = h(g^S y_B^{-e} y_F^e u \,||\, m) \pmod q,$$

mely biztosítja az F aláíró részvételét a vak aláírásban.

A második részben a vevő bebizonyíthatja, hogy a D aláíró előállított –e egy érvényes proxy vak aláírást az A nevében az ellenőrzés során. Ezt a következőképpen számítja ki:

$$u'' = (r\, y_A^r)^{-e+b} y_A^{-e}\, y_B^{-e}\, y_D^e \pmod q.$$

és a módosított egyenlet a következő lesz:

$$e = h(g^S y_D^{-e} y_A^e u \,||\, m) \pmod q$$

[23] Z. Tan, Z. Liu and C. Tang, Digital Proxy Blind Signature Scheme Based on DLP and ECDLP, MM Research Preprints, No 21. , Beijing, 2002, pp 212-217.

mely biztosítja, hogy az *(m, u”, s, e)* érvényes proxy vak aláírás a D aláíró által, melyet az A aláíró nevében állított ki, habár A aláíró soha nem ruházhatja át D aláíróra a proxy kulcsát.

Ezek a hamisítások nem lesznek függetlenek *u*-tól, így korlátozni kell őket vagy teljesen eltüntetni a számításból. Az eredeti protokollban az *u* nem jelenik meg, így a rendszer teljesen védve van az ilyen hamisításoktól.

2.4. Vakítás a váratlan aláírásokra[24]

Az eddig ismert aláírási rendszerek sok számítást vettek igénybe, viszont vannak olyan alkalmazások, melyekben ezek a követelmények teljesíthetetlenek. Most egy olyan új vak aláírási technikát vizsgálunk meg, amely lehetővé teszi korlátlan számú aláírás ellenőrzését egy szerény, de állandó mennyiségű számítással. Ez az új technika bizonyos egységek használatán alapul, melyeknek generátoroknak nevezünk.

Ez az új eljárás is az RSA algoritmusán alapul. Fontos kikötése, hogy a generátorokat és az aláírásokat is nyilvánosságra kell hozni, meg kell adni előre. Attól függően, hogy ez a követelmény mennyire teljesül, a vak váratlan aláírási protokoll a következő három forma egyikével rendelkezik:

1. Az első és legegyszerűbb formája ezeknek a protokolloknak a következő:

$$t = [m]g^k \ (mod\ N)$$

$$t' = [m\ g^k]^{d_i} \ (mod\ N)$$

$$m' = [(m\ g^k)^{d_i}]g^{-kd_i} \ (mod\ N)$$

ahol: m a „vakításra” váró üzenet; t a vakított üzenet; t' a t-ből számított aláírás; m' a nem vak, aláírt üzenet; N a nyilvánosan ismert modulus; e_i és d_i a nyilvános és titkos aláírás exponensek és végül k a szolgáltató titkos, vakító kulcsa, melyet egységesen választunk $\{1,\ldots,n^2\}$-ből. A kocka zárójelben szereplő elemek a bemenő adatok, kimenete pedig a bal oldalon található. A három kimenő adat a megadott sorrendben: vakítás, aláírás és „kivakítás”.

2. A másodig protokollforma a következő:

$$t = [m]\ g^{k_1}\ k_2 (mod\ N),$$

$$t' = [m\ g^{k_1} k_2]^{\ d_i} \ (mod\ N),$$

$$m' = [(m\ g^{k_1}\ k_2)^{\ d_i}]\ g^{-d_i\ k_1}\ k_2 \ (mod\ N),$$

ahol a vakító kulcs a k_1, amit úgy választunk, ahogy a fentiekben a k-t, viszont a k_2 vakító kulcsot egységesen csak a $\{1, n-1\}$ párból.

3. A harmadik és egyben legfontosabb forma a következő:

$$t = [m]g_1^{k_1}\ g_2^{k_2} \ldots\ g_r^{k_r} \ (mod\ N)$$

[24] D. Chaum, Blinding for Unanticipated Signatures, Advances in Cryptology, Lecture Notes in Computer Science, pp 227-232

$$t' = [m\, g_1^{k_1}\, g_2^{k_2} \ldots g_r^{k_r}]^{\,d_i} \pmod N$$

$$m' = [(m\, g_1^{k_1}\, g_2^{k_2} \ldots g_r^{k_r\ d_i})]\, g^{-d_i\ k_1} g^{-d_i\ k_1} \ldots . g^{-d_i\ k_r} \pmod N$$

ahol a k_i titkos vakító kulcsok, melyeket úgy kapjuk, mint a fentiekben a k kulcsot.

2.4.1. A faktorizáció elrejtése fizikai biztonsággal

Mint tudjuk a faktorizációt csak a bank ismeri, amit titokban kell tartani. Most azzal fogunk foglalkozni, hogy a faktorizáció felfedése nélkül hogyan tudnánk biztonságosan használni az aláírást.

Egy lehetséges megoldás lehetővé teszi, hogy bárki benyújtson az aláírónak egy fizikai eszközt a faktorizációs teszt lebonyolítására. Az aláíró titkos paramétereket adna egy készülékhez, ami lehetővé teszi az üzenet logikai következményeinek („ellenőrzött" vagy „nem ellenőrzött") elküldését a megfelelő személynek (K). Ezt olyan módon kell elvégezni, hogy a K személyt meg kell győzni arról, hogy az aláíró nem hamisította meg az aláírást, miközben nem szivárogtatja ki a készülék titkos paramétereit. Egyik módja ennek az intézkedésnek a következő: a megadott személy (K) létrehoz egy c' számot véletlenszerűen, alkalmaz egy nyilvánosan ismert, egyirányú f függvényt, mely biztosítja a következő egyenlőség fennállását: $c = f(c')$; megadja a készülékben a c' és N értékeit; megadja a c-t az aláírónak a készülék nyilvános értékeivel együtt. Azután az aláíró elkülöníti a készüléket a K személytől és megadja a titkos paramétereket (p és q). Ha az eredmény „*ellenőrzött*" lesz, akkor az eszköz kimenete c' lesz. Az aláíró aztán f függvényt alkalmazza az eszköz kimenetelére és leellenőrzi, hogy vajon az eredmény c lesz-e, és ha visszatér c' a K személyhez. A készülék úgy van kialakítva az illetéktelen beavatkozás ellen, hogy nehéz feladat legyen előállítani megfelelő c'-t a megadott időintervallumban (kivéve, ha a generátor tesztje „ellenőrzött" részben van).

Egy másik út a szolgáltatók bizalmához a generátorok alkalmasságában van, hogy ne tudja őket manipulálni senki, beleértve az aláírókat is. A generátorok egy véletlenszerű folyamat által vannak választva - például az imént leírtakban is így van -, így az összeköthetetlenség feltételét segíti.

2.4.2. Lineáris kivágás és választás (cut-and-choose) a modulokra és a generátorokra.

Egy másik megközelítés szerint egy leendő aláíró bejelenti az n_i modulokat , $1 \leq i \leq k$ és legyenek a megfelelő generátorok g_i, minden egyes „*kivágásra*”. És legyen az egyetlen *n* modulus (*választás*), aminek következtében nyilvánosságra kell hozni a p_i*-1* és q_i*-1* faktorizációját, ahol $1 \leq l \leq k$ és $l \neq i$, minden j modulusra, kivéve a i-ik egy kiválasztottra. Ezután bárki használhatja az ellenőrző technikákat a fent leírtak szerint (amelyik a titok birtokában támaszkodik rá), hogy ellenőrizze a többi modulusokat és megfelelő generátorokat, ily módon a valószínűsége $1\text{-}j^{-1}$, hogy a kiválasztott modulusok is ellenőrizve legyenek. Egyik ilyen választás lehet egy nyilvános pénzfeldobási esemény is akár. Másik útja az i meghatározásának, hogy minden egyes személy egy b’ ad meg véletlenszerűen egy $b = f(b')$ megfelelő bijekcióval, ahol *f* egy egyirányú függvény. Ezután minden egyes személy közzé teszi a *b* értékét. Miután minden *b,* majd *b’* nyilvánosságra került, leellenőrzik, így a hozzáadott modulok száma j, és az index i. Ennek a technikának a hátránya, ami az előző esetben is az, hogy nem „nyitott” új résztvevők számára, ha a „*kivágás és választás*” rész befejeződött.

Erre az algoritmusra nézzünk meg egy példát. Ha e-érmét szeretnénk kivenni, a felhasználó generál n darab érmét, ezeket „vakítja” faktorral és elküldi a banknak. A bank kiválaszt ebből n-1 darabot és megkéri a felhasználót, hogy bizonyítsa be, hogy ismeri a faktorokat, amivel vakolta az érméket. Ha a felhasználó ezeket bemutatta, akkor a bank leellenőrzi az érmék formátumait. Azt az egy darab érmét, ami nem lett felfedve, azt aláírja. Ebből kifolyólag ez a módszer jó arra, hogy sok számítást elkerüljön, mivel annak az esélye, hogy pont az az egy darab érme formátuma rossz az 1/n.

2.5. Vak Schnorr aláírás[25]

Ezt a transzformációt használták először az elektronikus pénz rendszerében „cut-and-choose” nélkül. Az algoritmusa a következő:

1. Legyen két nagy prímszámunk p és q, hogy $q \mid p-1$. Ezeket együtt teszik közzé a g-vel (Z/pZ)-ből.

2. Az aláíró készít egy kulcspárt $x \in Z/qZ$ és $y = g^{-x} \pmod p$-t. Közzé teszi y-t.

3. Ha egy felhasználó szeretne m üzenetre egy vak aláírást kapni. Annak érdekében, hogy kiadja ezt az aláírást, az aláíró választ egy véletlen $k \in Z \backslash qZ$ értéket, majd a saját kiszámolt értékét elküldi, ami a következő:

$$r = g^k \pmod p.$$

4. A felhasználó elvakítja ezt, két véletlenül választott $\alpha, \beta \in Z/qZ$ értékkel. A vakítás a következőképpen számolandó:

$$r' = rg^{-\alpha}y^{-\beta} \pmod p$$

majd kiszámolja az $e' = h(m,r') \pmod q$ és $e = e' + \beta \pmod q$ értéket .

5. Az utóbbit az aláírónak, aki visszaküldi az s értéket a következőkből kiszámítva:

$$g^s y^e = r \pmod p.$$

6. Ezt leellenőrzi az $s' = s-\alpha \pmod q$ segítségével.

(e', s') egy érvényes Schnorr aláírása az m üzenetnek, ha kielégíti a következő egyenletet:

$$e' = h(m, g^{s'}y^{e'} \bmod p).$$

2.5.1. Véletlen orákulum modell (Random Oracle modell)[26]

1993-ban Bellare és Rogaway létrehozott egy modellt, amely lehetővé teszi a biztonság igazolását különböző titkosítási eljárásokban.

Ez a modell biztosítja a Schnorr aláírás biztonságát. Véletlen orákulum alatt olyan függvényt értünk, mely mindenki által hozzáférhető, így a támadó is megkaphatja,

[25] D. Pointcheval and Jacques Stern, Provably Secure Blind Signature Schemes, Lecture Notes in Computer Science, Advances in Cryptology,1996, pp 2-3.

[26] D. Pointcheval and Jacques Stern, Provably Secure Blind Signature Schemes, Lecture Notes in Computer Science, Advances in Cryptology, 1996, pp 3.

ugyanazon az inputra ugyanazon outputot ad, és különböző inputra egyenletesen választott outputot ad.[27]

Informálisan úgy kell elképzelni, hogy a hash függvény egy függvényhalmazból véletlenszerűen kiválasztásra kerül és egy orákulum megkérdezése során határozzák meg a hash értékét, vagyis matematikailag nem létezik formula vagy algoritmus meghatározására. A valós életben nem létezik ilyen függvény, viszont szerkeszteni lehet egy olyan hash függvényt, mely hasonló a véletlen orákuluméhoz. Formálisan így írható le:

Legyen $h : X \rightarrow Y$ hash függvény, ahol $h \leftarrow Func_{X,Y}$ (függvényhalmaz) és legyen $X_0 \subseteq X$. Feltételezzük, hogy $h(x)$ érték csak az $x \in X_0$ esetekben került meghatározásra. Ha a hash függvény viselkedése hasonló a véletlen orákuluméhoz, akkor fennáll, hogy $Pr[h(x) = y] = 1/M$, minden $x \in X \backslash X_0$ és $y \in Y$ elemre.[28]

[27] Buttyán L. , Vajda I., Kriptográfia és alkalmazásai, Typotex Kiadó, 2004, 393 o.
[28] Márton Gy., Titkosítási rendszerek CCA-biztonsága, Doktori értekezés, 2013, 33 o.

3. Elektronikus pénz

Egyre nagyobb jelentőséggel bír a kereskedelem az interneten és az elektronikus pénz használata, így egyre nagyobb aktív kutatási területet von maga után. Az első kriptográfiai elképzelést az elektronikus pénzről David Chaum vezette be. Az volt a célja, hogy megalkossa a papírpénz elektronikus változatát, mely azonos tulajdonságokkal bír, anonim legyen és a bank ellenőrzése alatt tartsa. Az volt az elképzelése, hogy az anonimitás megtartásának a módja az, hogy az érmék használatát egybe kell kötnie a vak aláírásokkal. Amikor egy felhasználó vásárolni akar elektronikus pénzzel, akkor azt jelzi a bank felé, a bank visszaküldi az elektronikus pénzt, amely vakon alá van írva. Ezután a felhasználó el tudja költeni őket kijelölt üzletekben. [29]

A vak aláírásoknak az a céljuk, hogy biztosítsák azt az eszközt, amellyel a felhasználók alá tudják írni a pénzt úgy, hogy a bank később ne tudja beazonosítani ezeket. Ez a technika hatékony online esetben. De ha a kifizetés off-line-ban van, akkor nincs közvetlen módja annak, hogy megakadályozzák, hogy egy felhasználó lemásoljon egy érmét és felhasználja kétszer. Ezt a hamisítványt nevezik „double spending”-nek vagy dupla kiadásnak.

A következő eredményeket az e-coinok kutatásában Chaum, Fiat és Naor mutatta be, miszerint minden érme ugyanolyan anonim, mint eddig, kivéve a dupla kiadásoknál, akkor az érme azonosságát felfedik. Valamilyen speciális formát ad az érméknek, mivel a felhasználó ezt létrehozza, de a banknak kell ellenőriznie, hogy az érme formátuma megfelelő-e.

Vagyis az anonimitásnak két tulajdonságát különböztetünk meg, melyeket a rendszernek teljesítenie kell:

1. *követhetetlen*: Az érmék és a felhasználók között a kapcsolatot nem lehet nyomon követni
2. *összeköthetetlen*: Vagyis egy felhasználó érméi között nem lehet semmiféle kapcsolatot felfedezni.

Chaum, Fiat és Naor alkalmazta először a cut-and-choose technikát, melynek protokollját fent már bemutattuk. A bank megjelöl minél több érmét, majd véletlenszerű választással kéri a felhasználót, hogy hozzák nyilvánosságra néhányat közülük. A

[29] Horváth A., Az elektronikus pénz, mint az elektronikus kereskedelmet támogató speciális fizetési rendszer, Ph.D. értekezés, Budapest, 2007. 65-66 o.

hátránya ennek a technikának, hogy növeli a kommunikációs terhelést a bank és a felhasználó között és a szükséges helyet a tárolt érméknek. Sok fejlesztést próbáltak kialakítani cut-and-choose módszer nélkül. Legújabb kutatásoknál a feltétlen anonimitást kritizálták a pénzmosás vagy más lehetséges bűncselekmény miatt, és így új kutatási területek jelentek meg a letéti alapú rendszerek miatt.[30]

Az elektronikus pénz fajtái:[31]

1. *Azonosítható e-pénz* - olyan információkat tartalmaz, amely segítségével azonosítani tudjuk azt a személyt, aki a bankból felvette a pénzt.

2. *Anonim e-pénz* – az ilyen pénzek lenyomozhatatlanok, mivel egyszer ha valaki felvette a bankszámláról, annak a mozgása nem követhető, mert a tranzakcióknak nem marad nyomuk.

3. *offline/online e-pénz* - az offline anonim pénz vagy az igazi digitális e-pénz használatánál nem kell folyamatosan a bankkal kapcsolatban állni, mivel úgy is lebonyolítható a tranzakció. Az online használatú e-pénzek viszont ezeknek az ellentettje.

3.1. Követhetetlen elektronikus pénz[32]

Chaum vezette be először a feltétel nélküli követhetetlen elektronikus pénz fogalmát. De mi gátolja meg abban az embereket, hogy egy érméből többet létrehozzon és különböző üzletekben fizessen vele? Erre az online elszámolás az egyik lehetséges megoldás, bár ez elég drága.

Feltétele, hogy elektronikus pénzt generálnia bárkinek nehéz feladat legyen, kivéve, ha együttműködik a bankkal. A lenyomozhatatlan elektronikus aláírásra az RSA digitális aláírás sémát javasolták. Ennek a pénznek a formája a következő:

$$(x, f(x)^{1\backslash 3} \ (mod\ N)),$$

ahol N valamilyen összetett szám, melynek a faktorizációját csak a bank ismeri, f pedig egy erre alkalmas egyirányú függvény. A protokoll az e-pénz kiadására vonatkozólag a következőkben foglalhatók össze:

[30] D. Pointcheval and Jacques Stern, Provably Secure Blind Signature Schemes, Lecture Notes in Computer Science, Advances in Cryptology, pp 1-2.
[31] http://ecopedia.hu/elektronikus-penz
[32] D. Chaum, A. Fiat, M. Naor, Untraceable Electronic Cash, Advances in Cryptology, 1990, pp 319-327.

1. Alice választ egy véletlen x és r számot, és elküldi a banknak a következővel:

$$B = r^3 f(x) \pmod N$$

2. A bank visszaküldi az $r\, f(x)^{1\backslash 3} \pmod N$ értéket és visszavon tőle egy dollárt.
3. Alice kiszámítja a következőt a B értékből:

$$C = f(x)^{1\backslash 3} \pmod N$$

4. Ha Alice fizetni akar Bobnak egy dollárt, akkor elküldi a következő párt neki:

$$(x, f(x)^{1\backslash 3} \pmod N).$$

5. Bob azonnal hívja a bankot, leellenőrzi, hogy a pénz nincs-e letétbe helyezve, nem használták-e fel már egyszer.

Ennek a protokollnak az előnyei a következők:

1. Mindenki könnyen ellenőrizheti, hogy az érme megfelelő struktúrájú, aláírta-e a bank, de a bank nem tudja hozzá kapcsolni, nem tudja beazonosítani konkrét érmét Alice számlájához, mivel az r le lett mozgatva.
2. A kereskedőnek nem kell minden tranzakciókor felvenni a bankkal a kapcsolatot.
3. Alice csak egyszer tudja felhasználni az érmét, mivel a bank nyomon tudja követni a számlát és be tudja bizonyítani, hogy ha kétszer használt egy érmét.

Most vizsgáljuk meg az alapsémát, ami garantálja a pénz követhetetlenségét - viszont a bank mégis nyomon tudja követni a „repeat spender”-t - és, hogy hogyan kell a protokollt változtatni ahhoz, hogy a bank biztosan be tudja bizonyítani, ha Alice újrahasználja pénzét.

3.2. Követhetetlen pénzek protokollja[33]

A bank először közzétesz egy RSA modulust, aminek a faktorizációját titokban tartja. Legyen f és g két argumentumú ütközésmentes függvény, vagyis ami nem képes arra, hogy ugyanarra a két bemenetre azonos értéket adjon. Elvárjuk még azt is, hogy f véletlen orákulum legyen. Alicenak van egy bankszámlája u számozással, melyet pedig a bank v számozással tárol. Ahhoz, hogy *Alice egy elektronikus pénzt „kapjon”*, a következő protokollal kell a bankhoz fordulnia:

[33] D. Chaum, A. Fiat, M. Naor, Untraceable Electronic Cash, Advances in Cryptology, 1990, pp 319-327.

1. Alice választ a_i, c_i, d_i és r_i,értékeket, ahol $1 \leq i \leq k$. Ezek függetlenek és egyenletes eloszlásúak a (mod N) maradékosztályból.

2. Alice elküldi a banknak a *k* vak „jelölteket”:

$$B_i = r_i^3 . f(x_i, y_i) \ (mod\ N)$$

minden $1 \leq i \leq k$, ahol

$$x_i = g(a_i, c_i),\ y_i = g(a_i \oplus (u \,||\, (v+i)), d_i).$$

3. A bank választ egy véletlenszerű részhalmazát a k/2 vak jelölteknek, a következőképpen indexelve: $R = \{i_j\}$, $1 \leq i_j \leq k$, minden $1 \leq i \leq k/2$-re és továbbítja Alicenak.

4. Alice elküldi az r_i, a_i, c_i és d_i értékeket, ahol minden i az R-ből van és a bank leellenőrzi őket. Egyszerű jelölés kedvéért mi feltételezni fogjuk, hogy:

$$R = \{k/2 + 1, k/2 + 2, \ldots, k\}.$$

5. A bank elküldi Alicenak a következőket:

$$\prod_{i \notin R} B_i^{1/3} = \prod_{1 \leq i \leq k/2} B_i^{1/3} \ (\text{mod N})$$

és feltölt a számlájára egy dollárt. A bank szintén feltölti a *v* számlálót (ami Alice számlája a banknál) *k*-val.

6. Alice ezután könnyen megkapja az elektronikus érméket:

$$C = \prod_{1 \leq i \leq k/2} f(x_i, y_i)^{1/3} (mod\ N)$$

Alice ezután újraindexelheti a „jelölteket” a C-ben, ami legyen: $f(x_1, y_1) < f(x_2, y_2) < \ldots < f(x_{k/2}, y_{k/2})$ majd megnöveli a saját számlálóját.

Nézzük azt az esetet, amikor *Alice fizet egy dollárt Bobnak*. Akkor Alicenak és Bobnak a következőképpen kell eljárniuk:

1. Alice elküldi a *C*-t Bobnak.
2. Bob választ véletlenszerű bináris karaktereket: $z_1, z_2, \ldots z_{k/2}$.
3. Alice a következőképpen válaszol rá: minden $1 \leq i \leq k/2$ -re:

a. ha $z_i = 1$, akkor Alice elküldi Bobnak a_i, c_i, és y_i értékeket.

b. ha $z_i = 0$, akkor Alice elküldi Bobnak x_i, $a_i \oplus (u \,||\, (v+i))$ és d_i értékeket.

4. Bob leellenőrzi, hogy a *C* megfelelő formában van-e és Alice válaszai illeszkednek-e *C*-re.

5. Bob később elküldi a *C*-t és Alice „válaszait” a banknak, mely ellenőrzi azok helyességét és jóváírja a számláját.

A banknak el kell tárolnia *C*-t, a bináris értékeket: $z_1, z_2, \ldots z_k$ (minden $z_i = 1$-re) és az $a_i \oplus (u \mid\mid v)$ (minden $z_i = 0$-ra) értékeket.

Ha Alice ugyanazt a *C* érmét kétszer használja, akkor annak nagy valószínűsége van, hogy nyomon követik: nagy valószínűséggel két különböző boltos küldi el a kiegészítő bináris értékeket legalább egy bit z_i-t, ahol B_i megfelelő formában volt. A bank könnyedén meg tudja keresni az eredményt, hogy biztosítsa, hogy a *C*-t nem használták korábban. Ha Alice a *C*-t kétszer használta, nagy valószínűséggel a banknak megvan mindkét a_i *és* $a_i \oplus (u \mid\mid (v+i))$. Így a bank elkülöníti az *u*-t és lenyomozza a fizetést az Alice számláján.

Lehetséges probléma lehet ezzel a sémával az összejátszás Alice és egy másik boltos, például Charlie között. Például a Bobbal való tranzakciót követően, Alice elkezd egy új tranzakciót Charlieval, és mind Bob és mind Charlie elküldik a banknak ugyanazt az információt; a bank nagy valószínűséggel tudja, hogy az egyikük hazudik, de nem lehet megmondani, hogy melyik, és nem tudja Alice számláján a pénzt nyomon követni.

3.3. E-pénzek a gyakorlatban. NFC alapú fizetések[34]

Most vizsgáljuk meg az elektronikus-pénz használatát a gyakorlatban, hogyan használják, mint fizetési eszközt. Milyen követelményeknek kell megfelelniük, illetve milyen fizetési eszközök vannak.

A mai világban elterjedtek az NFC-k (Near Field Communication - egy rövid hatótávú kommunikációs szabványgyűjtemény okostelefonok illetve hasonló eszközök között, vezeték nélküli kommunikációs szabvány, mely rádiófrekvencia alapú azonosítást használ). Ez lehetővé teszi a kommunikációt az eszközzel kis tartományon belül (például eszköz egymás mellé helyezésével), megszerzéséhez pedig csak egy szoftvert kell letölteni az okostelefonunkra. Ez lehetővé teszi, hogy a felhasználónak nem kell többszöri fizetőeszközöket használni különböző helyeken. Előnye, hogy biztonságos és nincs hamis pénz.

[34] G. Hinterwälder, Ch. T. Zenger , F. Baldimtsi, Efficient E-cash in Practice: NFC-based Payments for Public Transportation Systems, pp 6-10.

Most vizsgáljuk meg ennek működését. Egy transportation payment (*TA*) rendszer a következők közt van kölcsönhatásban: *TA* (automaták, melyek az adatbázishoz vannak csatolva), a felhasználók rendszere (*U*) és egy fizetési gép (*M*), melyek kihelyezett pontokban vannak elhelyezve, mely ha megkapta az érvényes befizetést, a felhasználónak a Transportation rendszerhez hozzáférést adhat. Például egy bolti fizetésnél a következő pontok szerint megy végbe az algoritmus.

Beállítás. Itt a *TA* vagy más megbízható hatóság (authority) előállítja a rendszer nyilvános paramétereit, illetve a *TA* titkos kulcsát.

Számlanyitás. Először a felhasználónak regisztrálnia kell a *TA*-n, majd számlát kell nyitnia, megadja az adatait, ismertetőjegyeit $(L_1, \ldots, L_n)$ és megkapja a nyilvános kulcsát:

$$pk_U = g^{sk_U}.$$

Ezeknek az attribútumoknak a típusait és sorrendjét a *TA* határozza meg a *beállítás* alatt. Egy lehetséges beállítás lehet például az első attribútum (L_1) a felhasználó titkos kulcsa ($sk_V = L_1$), L_3 a kora, stb.

Visszavonás. Ha U vissza szeretné vonni a pénzeket az automatából, akkor legelőször be kell bizonyítania, hogy az ő tulajdonában van a számla. Ezt a digitális aláírásával tudja megtenni, pontosabban Schnorr aláírás technikával, melyet fent bemutattunk.

Kiadások. Hogy egy *U* felhasználó el tudjon költeni egy érmét a fizető automatánál, *U*-nak le kell futtatnia a fizetési protokollt.

Letét. A fizető automata a kiadást elküldi a *TA*-nak. Itt van még egy másik mechanizmus, amely leellenőrzi a dupla kiadásokat (double spending). Ennek érdekében a *TA*-nak van egy másik adatbázisa, ahol a letétben lévő érméket tárolja. Viszont ezt az adatbázist frissíteni kell, mivel nem tárolhatják örökre minden érmét. Így bevezettek egy lejárati időpontot, aminek lejárata után törlődik. Abban az esetben, ha még nem használták fel az érmét, arra vannak különböző ellenőrzések.

Most nézzünk meg egy protokollt ezekre a folyamatokra.

3.4. Brands protokollja az e-pénzekre.[35]

Most nézzük meg, hogy pontosan milyen számításokat kell elvégeznünk a fent említett részekben.

Beállítások. A *TA* vagy más megbízható hatóság választ a G csoportból egy q prímet, és $h, g, g_1, \ldots, g_n$ generátorokat, ahol az n a maximális attribútumok száma, melyre a rendszernek szüksége van. Végül legyen egy $h: \{0, 1\}^* \rightarrow Z_q^*$ hash függvény. A *TA* nyilvános kulcsa legyen $y = g^x$, ahol $x \in Z_q$ a titkos kulcsa.

Számlanyitás. Amikor *U* felhasználó egy számlát akar nyitni a *TA*-nál, először bemutatja az azonosító okmányát, majd kódolja az attribútumokat ($L_1, \ldots, L_n \in Z_q$) I^3-ba. Az L_1 érték, amit a *TA* nem ismer, lesz a felhasználó titkos kulcsa, a nyilvános kulcsa pedig $pk_U = g_1^{L_1}$. A π-t (ami egy bizonyíték *U*-nak, hogy ismeri a megfelelő értékeket) az általános Schnorr logaritmusból számolhatjuk ki. Ezek után a *TA* leellenőrzi ezeket az értékek érvényességét, eltárolja a felhasználó adatait, majd kiszámolja a $z = (Ih)^x$ értéket, melyet elküld *U* felhasználónak. A Brands protokollban attribútumok nélkül csak azt írja elő, hogy a felhasználó bizonyítani tudja a saját titkos kulcsának ismeretét. A számlanyitás protokollját a következőképpen írhatjuk le:

$TA\ (y = g^x)$		$U(pk_U = g_1^{L_1})$
		$I = g_1^{L_1} \ldots g_n^{L_n}$
		$\pi = PK\ \{(\Lambda_1, \ldots, \Lambda_n)\}:$
		$I/pk_U = g_2^{\Lambda_2} \ldots g_n^{\Lambda_n} \wedge$
		$pk_U = g_1^1$
π *ellenőrzőse*	$\xleftarrow{I,\ pkU,\ \pi}$	*Ha* $I\,h \neq 1$
$z = (I\,h)^x$	$\xrightarrow{z}$	z

Visszavonás. Ha *U* felhasználó vissza szeretné vonni *k* érmét, akkor először a *TA*-nak kell igazolnia magát a titkos kulcs ismeretével és meg kell mondani, hogy hány érmét szeretne visszavonni. Ezt egyszerre elérhetjük Schnorr aláírás használatával. A

[35] G. Hinterwälder, Ch. T. Zenger , F. Baldimtsi, Efficient E-cash in Practice: NFC-based Payments for Public Transportation Systems, pp 6-10. (Hinterwalder, és mtsai.,)

TA engedélyezi a visszavonást, ha az aláírás ellenőrzés alatt a felhasználónak helyes a nyilvános kulcsa és ha a elég pénz van a számláján. Majd végezetül a felhasználó lefuttatja a visszavonási protokollt:

$TA\ (y = g^x)$		$U(pk_U = g_1^{L_1})$
$w \in Z_q,\ a = g^w,\ b = (I\ h)^w$	$\xrightarrow{a, b}$	$s \in Z_q^*,\ A = (I\ h)^s,\ z' = z^s$
		$x_0, x_1, \ldots, x_n, u, v \in Z_q$
		$B = A^{x_0} g_1^{x_1} \ldots g_n^{x_n}$
		$a' = a^u g^v,\ b' = b^{su} A^v$
		$c' = H(A, B, z', a', b')$
	$\xleftarrow{e}$	$c = c' / u \pmod q$
$r = cx + w \pmod q$	$\xrightarrow{r}$	$g^r = y^e a\ ?,\ (I\ h)^r = z^c b\ ?$
		$r' = ru + v \pmod q$

Minden érmére *U*-nak el kell tárolnia az *A, B, sign(A, B)* $\cong$ *A, B, z', a', b', r'* értékeket *s* értékkel együtt és az $x_0, \ldots, x_n$ –t, ahol *sign (A, b)* egy vak Chaum-Pedersen aláírás. Annak érdekében, hogy leellenőrizzük az aláírást a következő egyenlőségeket kell megvizsgálni: $g^{r'} = y^{c'} a'$ és $A^{r'} = z'^{c'} b'$.

Költés (Spending). Ha *U* felhasználó elkölt egy érmét, a következő protokollt kell végrehajtani, ami nem tár fel minden attribútumot:

U $pk_U = g_1^{L_1}$		M
	$\xrightarrow{A, B, sign\ (A, B)}$	$A \neq 1$?
$r_1 = -d\ L_1 + x_1 \pmod q$	$\xleftarrow{d}$	$d = H_0\,(A, B, I_M,$

…		dátum/idő)
$r_n = -d\ L_n + x_n \pmod q$		
$R = d/s + x_0 \pmod q$	$\xrightarrow{(R, r1, \dots, rn)}$	$g_1^r \dots g_n^{r_n} h^{-d} = A^{-R} B$?
		Sign (A, B) ellenőrzése

Ebben a fázisban a felhasználó prezentálja az érméket: *A, B, sign (A, B) M*-nek és bizonyítja az attribútumai ismeretéről. A tranzakció és az aláírás érvényességének ellenőrzése után *M* elmenti a fizetési tranzakciót, mely a következőkből áll: *A, B sign(A, B), (R,*$r_1, \dots, r_n$*)* és az időbélyegzőt (dátum/idő).

Mi történik akkor, ha *U* felhasználó ki akar adni egy attribútumot (*Lj*) a költési protokoll folyamata alatt? *U* felhasználónak ekkor nincs szüksége arra, hogy kiszámítsa r_j értéket, ha megkapja a *d* értéket *M*-től. Ehelyett elküldi az L_j és saját „vakító” értékét x_j-t *M*-nek, melyet a lenti protokollban láthatunk. így nagyobb számú attribútumok felfedésével kevesebb a felhasználó által végzett számítás, viszont az adatok mennyisége növekszik.

Letét. Annak érdekében, hogy letétbe helyezzünk egy érmét, M benyújtja a fizetési transcriptet TA-nak. A TA először ellenőrzi az érme érvényességét, majd lekérdezi az adatbázist az összes érméről, hogy ezek az érmék nincsenek-e letétbe helyezve (leellenőrzi a double spending-et). Ezt a következő protokollal teheti meg:

$U(pk_U = g_1^{L_1})$		M
	$\xrightarrow{A, B, sign\ (A, B)}$	$A \neq 1$?
$r_1 = -d\ L_1 + x_1 \pmod q$	$\xrightarrow{A, B, sign\ (A, B)}$	$d = H_0(A, B, I_M,$ *dátum/idő)*
…		
$r_{j-1} = -d\ L_{j-1} + x_{j-1} \pmod q$	$\xrightarrow{A, B, sign\ (A, B)}$	
$r_{j+1} = -d\ L_{j+1} + x_{j+1} \pmod q$	$\xrightarrow{A, B, sign\ (A, B)}$	
…		

$r_n = -d\, L_n + x_n \pmod q$

$$\xrightarrow[A, B, sıgn\ (A, B)]{}$$

$R = d/s + x_0 \pmod q$

$$\xrightarrow[(R, mınden\ rJ)(LJ, XJ)]{}$$

$g_1^r \ldots g_j^{-d\, Lj + Xj} \ldots$

$g_n^{r_n} h^{-d} = A^{-R} B$?

sign (A, B) ellenőrzése

Összefoglaló

Mint láthattunk, minden algoritmusnak vannak feltételei, amit be kell tartanunk ahhoz, hogy megvédjük információnkat, hogy rendszerünk jól működjön és ne legyen könnyű ellopni adatainkat.

A dolgozatban sokat foglalkoztunk az RSA algoritmusával. Erre azért volt szükség, mert a vak aláírás algoritmusai erre támaszkodnak. Külön megvizsgáltuk, hogy hogyan generáljuk a kulcsokat, adtunk példát a kulcsméret nagyságára, megnéztük, hogy milyen összefüggés van köztük. Ezen belül leírtuk, hogy minden véletlenszerűen választott értékére milyen feltételek vannak, milyen tesztek léteznek például a prímszámok kiválasztására. Ezeket a szabályokat betartani kötelező, ha nem akarjuk, hogy gyorsan feltörjék kulcsunkat.

A következőkben külön a vak aláírási sémával foglalkoztunk. Láttuk, hogy vannak olyan esetek, melyekben megbízhatunk valakit, akit proxynak aláírónak nevezünk, hogy helyettünk írjon alá.

Majd végül kitértünk a gyakorlati alkalmazására, az elektronikus pénz generálására. A vak aláírásoknak ebben az a feladatuk, hogy biztosítsák azt, hogy a felhasználó úgy generálja az érméit, hogy a bank ne tudja beazonosítani ezeket, az érmék és a felhasználó között a kapcsolatot ne tudja bárki nyomon követni. Viszont a bank mégis ellenőrzése alatt tartja valamilyen szinten az érméket, mivel ki kell zárnia a double spending-et. Végezetül láthattunk arra is példát, hogy hogyan működik egy NFC alapú kifizetés.

Az elmúlt évszázadban nagyon sokat alakultak, fejlődtek ezek az algoritmusok, viszont van még mit fejleszteni rajtuk, mivel még mindig nem hibátlanok. Minél jobban fejlődik a technika, annál erősebbek a támadások és a hamisítások is, így folyamatosan javítani kell ezeket.

Irodalomjegyzék

1. Buttyán L., Vajda I., Kriptográfia és alkalmazásai. Budapest, Typotex Kiadó, 2004.

2. D. Chaum, Blind Signatures for Untraceable Payments. Advances in Cyptology., 1983.

3. D. Chaum, Blinding for Unanticipated Signatures. Center for Mathematics and Computer Science. Berlin Heidelberg , 1988.

4. D. Chaum, A. Fiat, M. Naor, Untraceable Electronic Cash. Advances in Cryptology, 1990.

5. Dr Berta Zs., Nagy e-szignó könyv. Budapest, Microsec Kft., 2011.

6. A. Esti Rahmawati, F. Is Esti, Implementation of RSA Blind Signature. International Jurnal of computer, 2013.

7. Folláth J., Huszti A., Pethő A., Informatikai Biztonság és Kriptográfia. Debrecen, 2010.

8. Györfi L., Győri S., Vajda I., Információ- és kódelmélet. Budapest, Typotex Kiadó, 2002.

9. G. Hinterwalder, C. Zenger, F. Baldimtsi, Efficient E-cash in Practice: NFC-based Payments for Public Transportation Systems. In Privacy Enhancing Technologies. Springer Berlin Heidelberg.

10. Horváth A., Az elektronikus pénz, mint az elektronikus kereskedelmet támogató speciális fizetési rendszer. Budapest, 2007.

11. Kadlecsik J., Az informatikai hálózati infrastruktúra és biztonsági kockázatai és kontrolljai, Budapest, 2004.

12. Sunder Lal, Amit Kumar Awasthi, Proxy Blind Signature Scheme. Cryptology ePrint Archive., 2003.

13. Márton Gy., Titkosítási rendszerek CCA-biztonsága. Budapest ,Doktori értekezés, 2013.

14. Páles Zs. , RSA. Alkalmazott Matematikai Lapok. 23. Budapest, 2006.

15. D. Pointcheval, J. Stern, Probably Secure Blind Signature Schemes. Advances in Cryptology , 1996.

16. Rónyai L., Ivanyos G., Szabó R., Algoritmusok. Budapest, Typotex Kiadó, 2005.

17. Z. Tan, Z. Liu Z., C. Tang, Digital proxy blind signature scheme based on DLP and ECDLP. Beijing, MM Research Preprints, 2002.

NYILATKOZAT

Név:

ELTE Természettudományi Kar, szak:

NEPTUN azonosító:

Szakdolgozat címe:

A **szakdolgozat** szerzőjeként fegyelmi felelősségem tudatában kijelentem, hogy a dolgozatom önálló munkám eredménye, saját szellemi termékem, abban a hivatkozások és idézések standard szabályait következetesen alkalmaztam, mások által írt részeket a megfelelő idézés nélkül nem használtam fel.

Budapest, 2016 _____

a hallgató aláírása