# Adatbiztonság 1. KisZH (2010/11 tavaszi félév)

**Ez a dokumentum a Vajda Tanár úr által közzétett fogalomlista teljes kidolgozása az első kiszárthelyire. A tartalomért felelősséget nem vállalok, mindenki saját felelősségére használja! ☺**

**Kolbay Zsolt**

### 1. Szimmetrikus kulcsú rejtjelezés alapelve

Nem biztonságos csatornán akarunk átküldeni egy $m \in M$ üzenetet (nyílt szöveget). Ehhez először titkosítjuk k1 kulccsal az üzenetet: c = $E_{k1}$(m), ahol $c \in C$ a rejtett szöveg. A fogadó fél a k2 kulccsal dekódolja a rejtett szöveget, így megkapja a titkosítatlan üzenetet: m = $D_{k2}$(c). A rejtjelezés szimmetrikus kulcsú, ha k1 = k2. A szimmetrikus kulcsú rejtjelező algoritmusok általában gyorsabbak és kisebb kulcsmérettel is biztonságosak, mint aszimmetrikus társaik, de használatuk esetén a kommunikáló feleknek előzetesen egyeztetniük kell a közös kulcsról. Pl: DES, Caesar-kód.

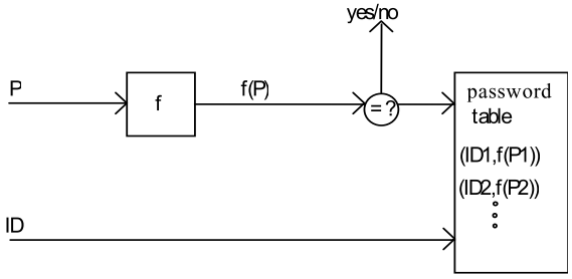
### 2. Aszimmetrikus (nyilvános) kulcsú rejtjelezés alapelve

Nem biztonságos csatornán akarunk átküldeni egy $m \in M$ üzenetet (nyílt szöveget). Ehhez először titkosítjuk k1 kulccsal az üzenetet: c = $E_{k1}$(m), ahol $c \in C$ a rejtett szöveg. A fogadó fél a k2 kulccsal dekódolja a rejtett szöveget, így megkapja a titkosítatlan üzenetet: m = $D_{k2}$(c). A rejtjelezés aszimmetrikus kulcsú, ha $k1 \neq k2$. Az aszimmetrikus kulcsú rejtjelező algoritmusok általában lassabbak, mint szimmetrikus társaik és nagyobb kulcsmérettel dolgoznak (RSA: tipikusan 2048 bit), viszont használatuk esetén a kommunikáló feleknek nem kell egy közös titkos kulcsról megegyezniük (minden fél kulcsának van egy titkos és egy publikus része). Pl: RSA, El-Gamal.

### 3. Hozzáférésvédelem feladata

Garantálja, hogy egy adott erőforráshoz vagy adatokhoz csak a jogosult felhasználók férhetnek hozzá, illetve azon módosító műveleteket csak arra jogosult felhasználók végezhetnek. A jogosultságok halmazát az autorizáció határozza meg. Általában jelszavakat menedzselő, ellenőrző és hozzáférési jogosultságokat kezelő részekből áll. Gyakran kapcsolódik partnerazonosításhoz/azonosító-hitelesítéshez, hiszen amíg a partnerek be nem bizonyították egymásnak kilétüket (autentikáció), addig nincs értelme vizsgálni azt sem, hogy kinek mihez van joga.

### 4. Partnerazonosítás feladata




A kommunikációban résztvevő fél azonosítja magát (pl: felhasználói név megadásával), majd megpróbálja bizonyítani kilétét, ezt tipikusan háromféleképpen teheti meg:

- rendelkezik valamivel (azonosító kártya, mobiltelefon, stb.)
- tud valamit (PIN-kód, szöveges jelszó, stb.)
- valamilyen tőle elválaszthatatlan azonosítóval (ujjlenyomat, retinaminta, stb.)

**5. Integritásvédelem feladata**

Az integritásvédelem azt hivatott biztosítani, hogy a csatornán átküldött üzenetet egy aktív támadó nem tud észrevétlenül módosítani. Néhány lehetséges megoldás: digitális aláírás RSA-val, iterált hash-függvény.

**6. Kulcsgondozás feladatai**

- kulcsgenerálás
- kulcskiosztás
- kulcstárolás
- kulcsfrissítés
- kulcs-visszavonás

**7. Blokk rejtjelezés**

Az üzenetet n bit hosszú blokkokra vágjuk (az utolsó kiegészítjük, ha rövidebb n bitnél) és a blokkokat egyenként kódoljuk. A (bináris) blokkrejtjelező egy $E: \{0,1\}^n \times \{0,1\}^k \longrightarrow \{0,1\}^n$ transzformáció, ami az n bites nyílt blokkot egy k bites kulcs segítségével n bites rejtett blokkba transzformálja. E természetesen invertálható kell hogy legyen, különben nem tudnánk visszanyerni az eredeti üzenetblokkokat.

**8. Kulcsfolyamatos rejtjelezés**

Ellentétben a blokk rejtjelezővel, a kulcsfolyamatos rejtjelezők egy lépésben egy bit vagy néhány bit által meghatározott karakter kerül kódolásra a kulcsbitek felhasználásával (amelyeket általában a rejtjelező generál). A rejtjelező működése időfüggő (állapotfüggő).

**9. Lineáris blokk rejtjelező**

A rejtjelezővel n bites blokkokat (x) kódolunk az alábbi módon: $\underline{y} = \underline{\underline{A}} \times \underline{x} + \underline{b}$
Ahol x, y és b n bites vektorok, A pedig $n \times n$-es mátrix. A kulcs A-ból és b-ből áll, vagyis $(n^2+n)$ bites. Ha egy támadó rendelkezik n darab nyílt-rejtett szöveg párral, akkor megfejtheti a kulcsot és az üzenetet (ismert nyílt szövegű támadás). Tehát egy pazarló (a kulcshoz $n^2+n$ bitet felhasználó) rejtjelező, amely viszonylag könnyen törhető.

**10. Betű-statisztikai alapú rejtjelfejtés**

Statisztikát készítünk a rejtjelezett blokkok (karakterek) gyakoriságáról és a leggyakrabban előfordulókat megpróbáljuk megfeleltetni a nyílt szöveg által használt ábécé leggyakoribb elemeinek (angol írott szövegben például az E és T betűk gyakoriak). Ezeket a párokat felhasználva ismert nyílt szövegű támadást indítunk, megvizsgálva hogy a visszafejtett üzenet értelmes-e. Ha nem, akkor másfajta párosítást alkalmazva támadunk, egészen addig próbálkozva, amíg egy értelmes nyílt szöveget nem kapunk.

**11. One time pad**

Jelölje X és K az üzenet és kulcs valószínűségi változókat (amelyek egymástól függetlenek), Y pedig a kódolt üzenet valószínűségi változót. Ekkor a rejtjelező transzformáció:

$Y = (X + K) \bmod 2$

Ahol X, Y és K n bites vektorok és az összeadás koordinátánkénti mod 2 összeadással történik. K egyenletes eloszlású az n bites vektorok halmazán (mintha minden bitet egy pénzfeldobással sorsolnánk ki).

A One-Time Pad tökéletes titkosító.

**12. Tökéletes rejtjelezés**

Egy titkosítás tökéletes, ha a nyílt (X) és a titkosított szöveg (Y) statisztikailag függetlenek, vagyis kölcsönös információtartalmuk zérus: I(X, Y) = 0. A támadó a megfigyelt rejtett szövegek alapján, erőforrásaitól függetlenül nem képes a nyílt szöveg megfejtésére. A One-Time Pad tökéletes titkosító.

**13. Shamir háromlépéses protokollja**

Rejtjelezett kommunikációt tesz lehetővé előzetes kulcsegyeztetés nélkül.

X: az üzenet
$E_A$, $E_B$: Alice és Bob kódoló függvényei
$D_A$, $D_B$: Alice és Bob dekódoló függvényei

| | |
|---|---|
| 1. Alice -> Bob: $Y_1 = E_A(X)$ | az üzeneten Alice zára van |
| 2. Bob -> Alice: $Y_2 = E_B(Y_1)$ | az üzeneten már Alice és Bob zára is rajta van |
| 3. Alice -> Bob: $Y_3 = D_A(Y_2)$ | Alice leveszi róla a zárát, már csak Bob-é van rajta |
| 4. Bob: $X = D_B(Y_3)$ | Bob leveszi róla a saját zárát, így előáll az eredeti üzenet |

A helyes működés feltételei:
1. Kommutatív rejtjelező transzformációk: $E_A(E_B(X)) = E_B(E_A(X))$.
2. Lehallgató típusú támadó (egy aktív támadó Bob helyett felrakhatja a saját zárát és így könnyen megfejtheti az üzenetet).

**14. Egyirányú függvény**

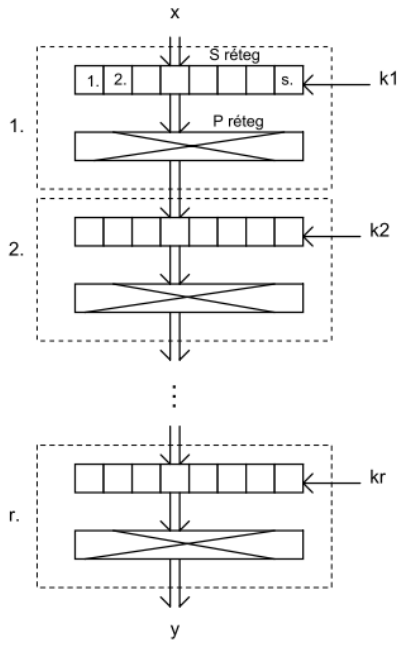
Az üzenetek integritásvédelmére használunk egyirányú függvényeket (kriptográfiai hash-függvényeket), a teljes üzenet vagy egyes blokkjainak lenyomatát képezve vele.
A $h$: $\{0,1\}^n \longrightarrow \{0,1\}^m$ hash-függvény egy n bites blokkhoz egy m bites blokkot rendel. Mivel m < n, ezért h nem invertálható (egy hash-értékhez több őskép is tartozhat).

**15. Helyettesítéses-permutációs rejtjelezés**

Shannon-i elv: erős invertálható transzformáció előállítható több (önmagában gyenge) könnyen implementálható transzformáció sokszori egymás utáni alkalmazásával.

S: helyettesítő réteg
P: permutációs réteg

Pl: DES, AES, 3DES, stb.




**16. S-box balansz tulajdonság**

Az S-doboz transzformációja nem torzítja el egy egyenletes eloszlású bemenet gyakoriság statisztikáját.

$f: \{0,1\}^m \longrightarrow \{0,1\}^n, m \geq n$
$\#\{\boldsymbol{x} \in \{0,1\}^m: f(x) = y\} = \boldsymbol{2^{m-n}} \quad \forall \boldsymbol{y} \in \{0,1\}^n$

**17. S-box nemlinearitás**

Két Boole-függvény ($f, g: \{0,1\}^m \longrightarrow \{0,1\}$) távolsága:
$d(f,g) = \#\{\boldsymbol{x} \in \{0,1\}^m: f(x) \neq g(x)\} = w(f+g)$

Lineáris Boole-függvény: $L_{u,v}(x) = u \cdot x + v \qquad u, x \in \{0,1\}^m$ és $v \in \{0,1\}$
Egy Boole-függvény nem-linearitása: $\mathrm{N(f)} = \min_{u \in \{0,1\}^m, v \in \{0,1\}} d(f, L_{u,v})$
Egy S-doboz nem-linearitása: $\mathrm{N_s(f)} = \min_{w \in \{0,1\}^m, w \neq \underline{0}} N(w \cdot f)$

**18. SPC lavinahatás**

A nyílt szöveg egyetlen bitjének megváltoztatása jelentős változást okoz a rejtett szövegben (a bitjeinek kb. felét megváltoztatja):

$$\frac{1}{2^m} \cdot \sum_{x \in \{0,1\}^m} w(f(x) + f\left(x + e_m^{(i)}\right)) = \frac{n}{2}, \qquad 1 \leq i \leq m, \qquad f: \{0,1\}^m \longrightarrow \{0,1\}^n$$

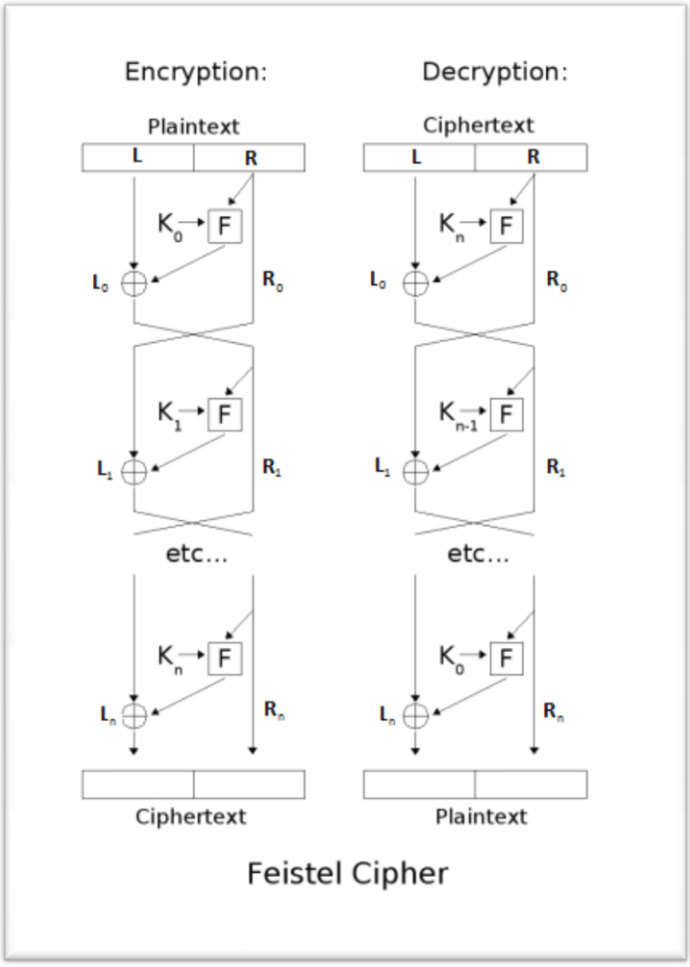
**19. DES Feistel technika**

A nyílt szövegblokkot két egyforma részre vágjuk: L, R.
Sorozatban végrehajtjuk rajta az F transzformációt.

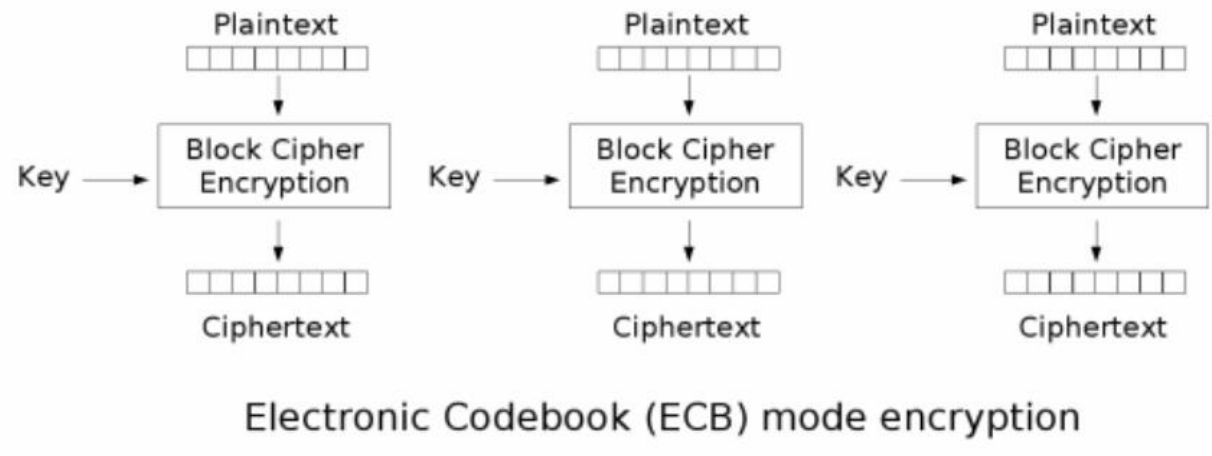
Az (i+1). iteráció során:
az aktuális kulcsot a kulcsütemező szolgáltatja: $K_i$
$L_{i+1} = R_i$
$R_{i+1} = L_i \oplus F(R_i, K_i)$

Ez invertálható, tetszőleges F-re, mert:
$L_i = R_{i+1} \oplus F(L_{i+1}, K_i)$
$R_i = L_{i+1}$

A dekódolás egyszerű: a rejtett szöveget be kell adni a rejtjelező bemenetére, csak az iterációk kulcsainak sorrendjét kell megfordítani.



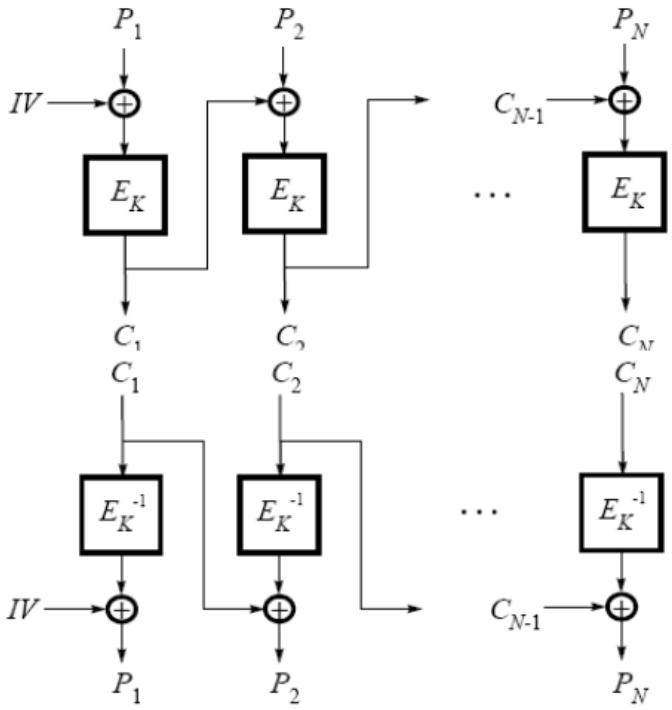

**20. ECB mód blokkséma**



**21. ECB mód biztonság**

- a nyílt szöveg mintáit nem rejti el
- a blokkrejtjelező bemenete nem randomizált
- szótár (kódkönyv) alapú támadás lehetséges
- a rejtjeles blokkok felcserélhetők

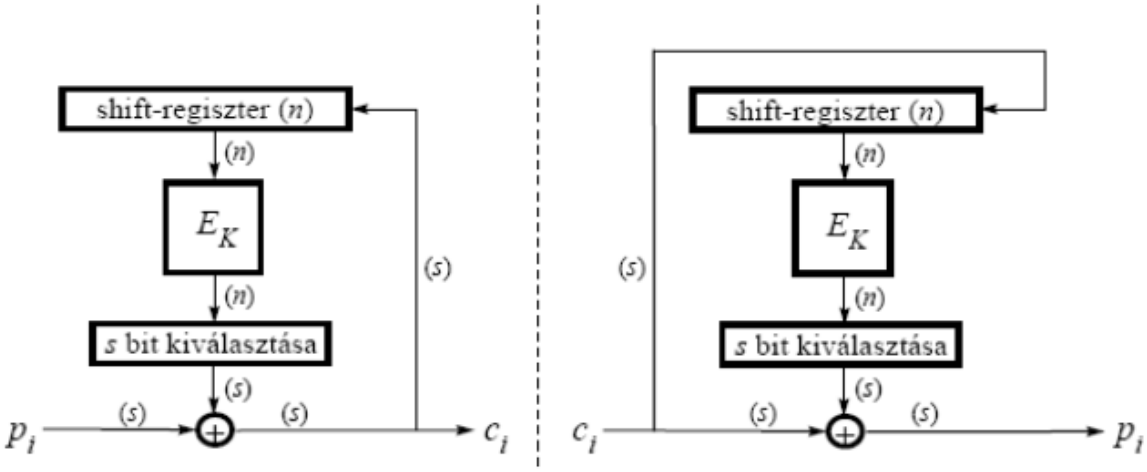
**22. CBC mód blokkséma**




**23. CBC mód biztonság**

+ a nyílt szöveg mintáit rejti az előző rejtjeles blokkal való XOR-olás
+ a blokkrejtjelező bemenetét randomizálja az előző rejtjeles blokkal való XOR-olás (nő a bemenet entrópiája)
+ azonos nyílt szöveget kulönböző IV-vel rejtjelezve különböző rejtjeles szöveget kapunk
+ korlátozott mértékben lehetőséget nyújt a blokkok törlésének, felcserélésnének és beszúrásának detektálására
- kivág-és-beszúr támadások lehetségesek
- azonos rejtjeles blokkokhoz tartozó nyílt blokkok XOR összegét felfedi

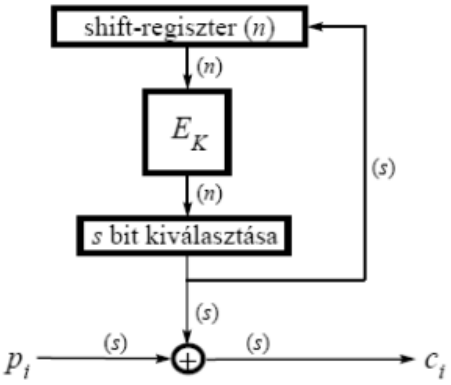
**24. CFB mód blokkséma**




**25. CFB mód biztonság**

+ a nyílt szöveg mintáit elrejti
+ a blokkrejtjelező bemenete véletlen
+ azonos nyílt szöveget különböző IV-vel rejtjelezve különböző rejtjeles szöveget kapunk
+ korlátozott mértékben lehetőséget nyújt a blokkok törlésének, felcserélésnének és beszúrásának detektálására
- utolsó blokk bitjei manipulálhatók

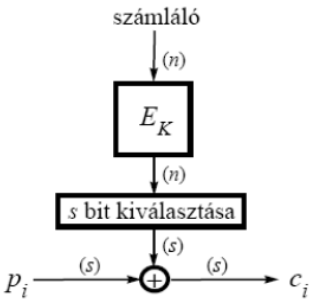
**26. OFB mód blokkséma**




**27. OFB mód biztonság**

+ a nyílt szöveg mintáit elrejti
+ azonos nyílt szöveget különböző IV-vel rejtjelezve különböző rejtjeles szöveget kapunk
+ korlátozott mértékben lehetőséget nyújt a blokkok törlésének, felcserélésnének és beszúrásának detektálására (de nem olyan jól mint CFB módban)
- különböző nyílt szövegeket azonos IV-vel rejtjelezve a nyílt szövegek visszafejthetők
- n-nél kevesebb bites visszacsatolás esetén a generátor periódusideje jelentősen csökken
- a visszaállított nyílt blokkok bitjei manipulálhatók (rejtett blokkok felcserélésével)

**28. CTR mód blokkséma**




**29. CTR mód biztonság**

+ a nyílt szöveg mintáit elrejti
+ azonos nyílt szöveget különböző IV-vel rejtjelezve különböző rejtjeles szöveget kapunk
+ nem nyújt lehetőséget a blokkok törlésének, felcserélésnének és beszúrásának detektálására (de nem olyan jól mint CFB módban)
+ a generátor periódusideje a számláló méretétől függ
- különböző nyílt szövegeket azonos IV-vel rejtjelezve a nyílt szövegek visszafejthetők
- a visszaállított nyílt blokkok bitjei manipulálhatók (rejtett blokkok felcserélésével)

**30. Hibasokszorozódás**

ECB blokkrejtjelezési módban:
- egy bit hiba a rejtjeles szövegben egy teljes blokkot használhatatlanná tesz
- bitbeszúrásból vagy bittörlésből származó hibából nem épül fel

CBC blokkrejtjelezési módban:
- egy bit hiba a rejtjeles szövegben hatással van az aktuális blokkra és az azt követőre
- bitbeszúrásból vagy bittörlésből származó hibából nem épül fel

CFB blokkrejtjelezési módban:
- egy bit hiba a rejtjeles szövegben $^{n}/_{S} + 1$ karaktert érint, amiből $^{n}/_{S}$ használhatatlanná válik
+ bitbeszúrásból vagy bittörlésből származó hibából felépül

OFB/CTR blokkrejtjelezési módban:
+ egy bit hiba a rejtjeles szövegben egy bit hibát generál a visszaállított nyílt szövegben
- bitbeszúrásból vagy bittörlésből származó hibából nem épül fel

**31. RSA kulcs setup**

1. Kiválasztunk két nagy prímszámot: p és q.
2. $m = p \cdot q, \quad \varphi(m) = (p-1) \cdot (q-1)$
3. Válasszunk ki egy tetszőleges $1 < e < \varphi(m)$ számot, amelyre $\big(e, \varphi(m)\big) = 1$
4. Számítsuk ki e inverzét modulo $\varphi(m)$-re: $d = e^{-1}(mod\ \varphi(m))$
5. A kulcs publikus része: $K_p$ = (m, e)
6. A kulcs titkos része: $K_s$ = (d, p, q)

Az x nyílt szöveg kódolása: $y = x^e (mod\ m), \ \ 1 \leq x < m$
Az y rejtett szöveg dekódolása: $x = y^d (mod\ m), \ \ 1 \leq y < m$

**32. Ismételt négyzetre emelés és szorzás**

RSA-val való rejtjelezésnél előnyös az $e = 2^t + 1$ választása a publikus kulcshoz, mert így az $y = x^e (mod\ m)$ transzformáció kiszámolható egyetlen modulo szorzással és t darab négyzetre emeléssel.

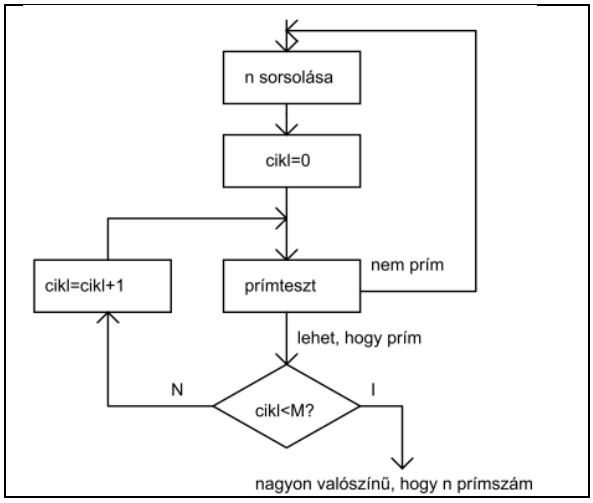
**33. Álprím**

Egy n összetett szám (Fermat-) álprím egy b bázisra nézve, ha

$$b^{n-1} = 1(mod\ n), ahol\ 1 < b < n\ és\ (b,n) = 1$$

Carmichael-szám: olyan összetett szám, amely tetszőleges b bázisra álprím (pl: 561).

**34. Fermat primteszt**




- n sorsolása: első és utolsó bitet 1-re állítjuk, a többit egyenként pénzfeldobás alapon választjuk meg
- N alkalommal végrehajtjuk rajta a prímtesztet: kiválasztjuk n egy tetszőleges b bázisát, ha $b^{n-1} \neq 1(mod\ n)$, akkor biztos, hogy n összetett szám
- annak az esélye, hogy egy (nem Carmichael) szám átmegy egy prímteszten, legfeljebb $^1/_2$, így N alkalommal különböző bázisokra végrehajtva a tesztet, $2^{-N}$ az esélye annak, hogy N összetett szám

**35. Fermat faktorizáció**

Ha $n = p \cdot q$ két prímszám szorzata, ahol p > q, valamint p és q különbsége kicsi, akkor a két prím értéke kiszámítható a Fermat-faktorizációval:
Legyen $t = \frac{p+q}{2}$ $és$ $s = \frac{p-q}{2}$. Ekkor p=t+s és q=t-s, tehát $n = (t+s) \cdot (t-s) = t^2 - s^2$. Ha s kicsi, akkor $t \approx \sqrt{n}$.
Próbáljuk meg kitalálni t értékét: $t_1 = \lfloor\sqrt{n}\rfloor$+1, $t_2 = \lfloor\sqrt{n}\rfloor$+2, $t_3 = \lfloor\sqrt{n}\rfloor$+3, ...
Ellenőrizzük minden $t_i$-re, hogy ($t_i^2$-n) négyzetszám-e. Ha igen, akkor az s-el egyenlő, p és q pedig könnyen kiszámítható.

**36. El-Gamal rejtjelező**

Legyen G egy multiplikatív csoport, g pedig egy generátor eleme.
Titkos kulcsnak válasszuk a $K_s$=x véletlen elemet a $\{1,2,\dots,|G|\}$ halmazból.
Publikus kulcsnak legyen $K_p$={X, G, g}, ahol $X=g^x$.

Az $m \in M$ üzenet rejtjelezése:

- kiválasztunk egy véletlen y elemet az M halmazból.
- legyen $Y=g^y$ és $b = X^y \cdot m$
- ekkor a rejtjelezett üzenet: $E_{K_p}(m) = (Y, b)$

Az (Y, b) üzenet dekódolása:

- $D_{K_s}(Y,b) = \frac{b}{Y^x} = \frac{X^y \cdot m}{Y^x} = \frac{g^{xy} \cdot m}{g^{xy}} = m$

**37. ECDLP**

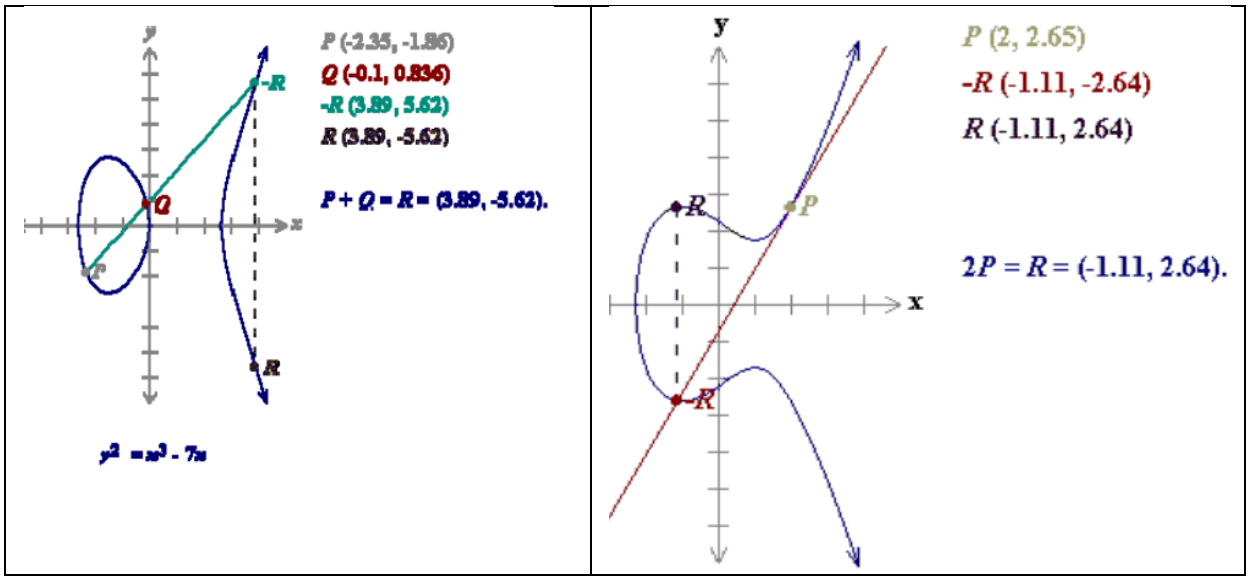
Egy elliptikus görbe pontjai az alábbi egyenlettel definiálhatók:

$$y^2 = x^3 + a \cdot x + b \quad és \quad 4 \cdot a^3 + 27 \cdot b^2 \neq 0$$

Ezeken kívül az elliptikus görbéknek még egy pontjuk van: a végtelen messzeségben lévő **O** pont.

Műveletek a görbe pontjain:

- előjelváltás (ellentett): **P(x, y)** pont ellentettje **P**-nek az x-tengelyre tükrözött képe, vagyis **R(x,-y)**
- összeadás: legyen a görbe két pontja **P** és **Q**. E két pont összegét jelölje **R=P+Q**. Ekkor a műveletet az alábbiak szerint kell elvégezni:
  - Kössük össze **P**-t és **Q**-t egy egyenessel.
  - Az egyenes egy harmadik pontban metszi a görbét, ez lesz **–R**.
  - Ennek képezzük az ellentetjét és megkapjuk **R**-t.
- **P+P=2P** kiszámítása: húzzuk be a görbe érintőjét a **P** pontba, ez kimetszi **–R**-t, annak vegyük az ellentetjét. Ha **P** az x-tengelyen van akkor **2P=O**.
- **P+(-P)=O**
- **R=P+Q** algebrai módon: $s = \frac{y_P - y_Q}{x_P - x_Q}$, $x_R = s^2 - x_P - x_Q$ és $y_R = s \cdot (x_P - x_R) - y_P$




Elliptikus görbe $F_p$ test felett:

$$y^2 = x^3 + a \cdot x + b \ (mod\ p) \quad és \quad 4 \cdot a^3 + 27 \cdot b^2 \neq 0$$
$$ahol\ p\ prím\ és\ 0 \leq a, b, x, y < p$$

Ekkor az előbbi műveletek továbbra is alkalmazhatók, csak $F_p$-beli értékekkel:

R=P+Q: $s = (y_P - y_Q) \cdot (x_P - x_Q)^{-1} (mod\ p)$

$x_R = s^2 - x_P - x_Q (mod\ p)$

$y_R = s \cdot (x_P - x_R) - y_P (mod\ p)$

Értelmezhető a szorzás művelete is: egy P pont $n \in F_p$-vel való szorzása, P összeadását jelenti n alkalommal.

Az elliptikus görbéken alapuló diszkrét logaritmikus probléma (ECDLP): $R = n \cdot P$, adjuk meg n értékét P és R ismeretében! Ez nehéz feladat, ezt használják ki egyes aláíró és kulccserélő protokollok.

**38. ECDH kulcscsere**

Az elliptikus görbéket használó Diffie-Hellman kulccsere alkalmazása két kommunikáló fél esetén (Alice és Bob):

1. Alice és Bob megegyeznek egy $E$: $y^2 = x^3 + c \cdot x + d (mod\ p)$ görbében és annak egy G pontjában. E és G publikus adatok.
2. Alice és Bob kiválasztanak egy-egy véletlen egész számot, amelyek kisebbek mint G rendje: $a$ és $b$. A generált számukat mindketten titokban tartják.
3. Alice átküldi Bob-nak az $aG$pontot, Bob pedig Alice-nek a $bG$ pontot.
4. Alice beszorozza a-val a Bob-tól kapott pontot, Bob pedig b-vel az Alice-től kapottat, így mindketten megkapják az $abG$ pontot.
5. Ezután a most már mindkettőjük által ismert $abG$ pont tetszőleges tulajdonságait használhatják közös titkos kulcsként (pl: x vagy y koordinátáját).

**39. Digitális aláírás generálása és ellenőrzése**

Az üzenet hitelesítése nyilvános kulcsú titkosító algoritmus felhasználásával (pl: RSA).
Alice: küldő fél
Bob: fogadó fél
X: az üzenet
$D_A$: Alice dekódoló függvénye (a titkos kulcs felhasználásával)
$E_A$: Alice kódoló függvénye (a nyilvános kulcs felhasználásával)

A küldő fél aláírás generálása: Alice a dekódoló transzformációval (a titkos kulcsával) titkosítja az X üzenetet (vagy ha túl hosszú, akkor a hash-kódját), majd az eredeti üzenettel együtt átküldi Bob-nak.
Alice -> Bob: X, $D_A(X)$

A fogadó fél aláírás ellenőrzése: Bob visszafejti az aláírást ($D_A(X)$-et) Alice publikus kulcsának felhasználásával, majd a kapott eredményt összeveti az üzenettel.
Bob: X ?= $E_A(D_A(X))$

**40. Digitális aláírás vs. analóg aláírás**

- Az aláírás hiteles: amikor Bob ellenőrzi az aláírást Alice publikus kulcsával, meggyőződik róla, hogy azt csak Alice küldhette

- Az aláírás nem hamisítható: csak Alice ismeri a saját titkos kulcsát

- Az aláírás nem újrahasznosítható (nem emelhető át egy másik dokumentumhoz): az aláírás a dokumentum függvénye is

- Az aláírt dokumentum nem módosítható: ha módosítják a dokumentumot, az eredeti aláírás nem illeszkedik, és ez detektálható

- Az aláírás letagadhatatlan: Bob vagy egy haramdik fél Alice közreműködése nélkül képes az aláírás ellenőrzésére

**41. Kriptográfiai hash függvény egyirányúsága**

Egyirányú Hash-függvény (One-Way Hash Function): a h hash-függvény egyirányú, ha egy y hash-érték (lenyomat) ismeretében nehéz olyan X' értéket találni, amelyre h(X') = y.

**42. Kriptográfiai hash függvény ütközésmentesége**

Ütközésmentes Hash-függvény (Collision Resistant Hash Function): a h hash-függvény ütközésmentes, ha nehéz két $X, X'$: $X \neq X'$ üzenet találni, amelyekre h(X) = h(X').

**43. Születésnapi paradoxonok**

Egy 365 fős csoportból elég $\sqrt{365} \approx 19$ különböző embert kiválasztani ahhoz, hogy kb. $^1/_2$ valószínűséggel legyen közöttük két olyan, akik ugyanazon a napon születtek.

Tehát ha van egy M méretű halmazunk, akkor abból egy $\sqrt{M}$ nagyságrendű részhalmazt véletlenszerűen kiválasztva nagy a valószínűsége annak, hogy két azonos tulajdonságú elem lesz benne.

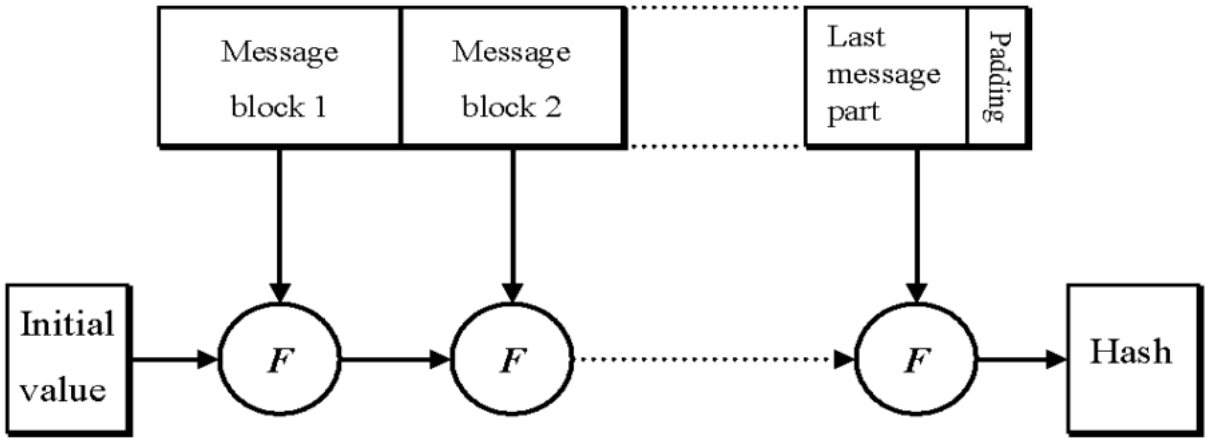
Egy módosított változata: egy M alaphalmazból véletlenszerűen választva egy r méretű U és V részhalmazt, mekkora az esélye annak, hogy a két részhalmaz metszete nem üres? A válasz: r = $\sqrt{M}$ esetén $P \approx 0.95$

**44. Születésnapi paradoxon és hash fv. támadása**

A születésnapi paradoxon segítségével egy hash függvény esetén becsülhető egy ütközés megtalálásának valószínűsége. Ha a H(x) hash-függvény kimenete n bites, akkor $2^{\frac{n}{2}}$ darab véletlenül választott üzenet között nagyjából $^1/_2$ valószínűséggel lesz két egyező.

Védekezés ha n-t kellően nagynak választjuk, akkor egy $2^{\frac{n}{2}}$ méretű halmaz ütközésellenőrzése kivitelezhetetlenné válik.

**45. Iterált kriptográfiai hash fv.**




Az üzenetet egyforma, n bites blokkokra tördeljük (az utolsó blokkot szükség szerint kiegészítjük, lásd DM-padding), majd sorozatban alkalmazzuk rajtuk az F függvényt, mégpedig úgy, hogy a függvény bemenete az adott iterációban az akuális üzenetblokk és az előző függvényhívás eredménye.

F: egyirányú kompressziós függvény, két n bites bemenete és egy n bites kimenete van.

Initial value: kezdeti érték az első F függvényhíváshoz, értéke általában előre rögzített.

Tulajdonságai:

- 1. őskép-ellenálló: egy y hash-értékhez nehéz olyan üzenetet találni, amelynek y lenne a lenyomata
- 2. őskép-ellenálló: egy konkrét X üzenethez nehéz olyan X'≠X-et találni, amelyre h(X) = h(X')
- ütközésellenálló: nehéz két olyan különböző üzenetet találni, amelynek ugyanaz lenne a hash-értéke

**46. DM padding**

Iterált hash-függvény esetén, ha az F kompressziós ütközésellenálló, és az üzenetet kiegészítjük egy olyan blokkal, amely tartalmazza az üzenet hosszát (megfelelő számú 0 bittel kiegészítve), akkor az iterált hash-függvény is ütközésellenálló lesz.

**47. Kihívás és válaszvárás a partnerazonosításban**

Alice úgy hitelesíti magát Bob előtt, hogy bebizonyítja, hogy birtokában van egy kulcsnak amit csak Alice ismerhet. A bizonyításhoz Alice használja ezt a kulcsot, tehét rejtjelez, dekódol, vagy aláír vele valamit. A hitelesítés során Alice egy Bob által frissen generált elemen (kihíváson) alkalmazza a kulcsot, így Alice válasza nem csak a kulcstól függ, hanem a kihívástól is, ezért egy potenciális támadó nem tudja a hitelesítő üzenetet más alkalommal felhasználni.

| | |
|---|---|
| Bob -> Alice: $E_K(V)$ | Bob elküldi Alice-nek a V véletlen elemet az Alice által ismert kulccsal kódolva |
| Alice -> Bob: V (= $D_k(E_k(V))$) | a Bob-tól kapott üzenetet dekódolja a kulccsal és V-t visszaküldi |

**48. Egyirányú függvény és jelszóvédelem**

Alice és Bob kicserél egyeztet PW jelszót (vagy kettőt, ha kétirányú hitelesítést szeretnének). Minden azonosításkor Alice átküldi Bob-nak a [T, Hash(PW, T)] üzenetet, ahol T az aktuális idő. Bob ellenőrzi friss-e az időbélyeg, majd kiszámolja a Hash(PW, T) értéket és összeveti az Alice által küldöttel. Egy aktív támadó így nem képes sem PW megismerésére, sem replay-, vagy pre-play támadásra.

**49. Egyszer használatos jelszó**

Alice hitelesíteni akarja magát Bob előtt. Először generál egy R véletlen elemet, majd n alkalommal alkalmazza rajta az f egyirányú függvényt, majd egyezteti az eredményt és az n számot Bob-al. Ezután n darab egyszer használatos jelszóval rendelkezik amit n alkalommal hasznalhat fel arra, hogy Bob előtt hitelesítse magát.

Inicializálás:

1. Alice: az R véletlen elem generálása
2. Alice: y = $f^{(n)}(R)$ kiszámítása
3. Alice -> Bob: $ID_A$, n, y

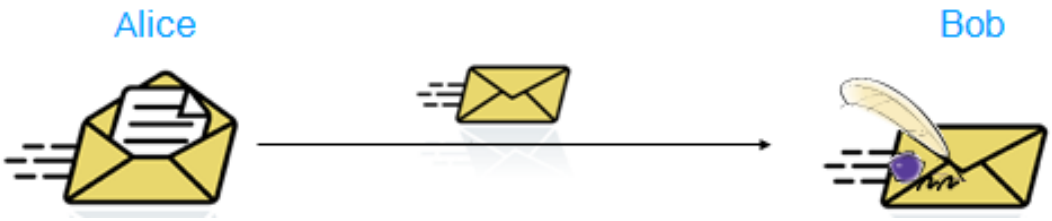
Hitelesítés (az i. kulccsal):

1. Alice -> Bob: $P_i = f^{(n-i)}(R)$
2. Bob: $y \stackrel{?}{=} f^{(i)}(P_i)$ (hitelesítés)

**50. Vak aláírás**

RSA titkosítást alkalmazva, ahol

- e: kódoló kulcs (nyilvános)
- d: dekódoló kulcs (titkos)
- n: két nagy prímszám szorzata (nyilvános, $e \cdot d = 1 \ (mod \ \varphi(n))$)




- Véglegesíti a dokumentumot: X
- Behelyezi egy átlátszatlan borítékba:
  - egy r < n véletlen érték kiválasztása, ahol (r, n) = 1
  - $Y = r^e \cdot X^e (mod \ n)$
- Y átküldése Bob-nak

- Aláírás: $S = Y^d (mod \ n)$
- Bob nem tudja mit ír alá
- S visszaküldése Alice-nek

Aláírás ellenőrzése:
$r^{-1} \cdot S =$
$r^{-1} \cdot Y^d =$
$r^{-1} \cdot r^{e \cdot d} \cdot X^{e \cdot d} =$
$r^{-1} \cdot r \cdot X = X$

**51. Moduláris négyzetgyökvonás probléma**

Legyen $n = p \cdot q$, ahol p és q két nagy prímszám.
$Ha \ \exists x, amelyre \ x^2 = c \ (mod \ n), akkor \ a \ c \ számot \ kvadratikus \ maradéknak,$
$x \ megoldást \ pedig \ c \ négyzetgyökének \ nevezzük \ modulon \ n.$
Ha csak a c és n értékeket ismerjük, akkor x meghatározása nehéz feladat, de ha n faktorait (p és q) is ismerjük, akkor már könnyű.
Ez felhasználható Zero-Knowledge-protokolloknál, ahol az egyik fél (Alice) azt szeretné bizonyítani a másiknak (Bob), hogy birtokában van egy titoknak, anélkül hogy felfedné azt a titkot.

A Fiat-Shamir partner-hitelesítési protokoll egy kulcskiosztó központot használ, amely generál két véletlen prímszámot, p-t és q-t, majd az $n = p \cdot q$ modulust kiadja Alice-nek és Bobnak (de p-t és q-t nem). Ezután sorsol Alice-nek egy u titkos kulcsot (véletlenszám), és egy $v = u^2 (mod \ n)$ publikus kulcsot.

Ezután ha Alice hitelesíteni akarja magát Bob előtt:

1. Alice kisorsol egy 0 < R < n véletlenszámot, majd elküldi $z = R^2 (mod\ n)$ számot Bob-nak
2. Bob visszaküld egy véletlen b bitet (b = 0 vagy b = 1) Alice-nek

Ha b = 0:

3. Alice elküldi az R számot
4. Bob ellenőrzi a $z = R^2 (mod\ n)$ értéket

Ha b = 1

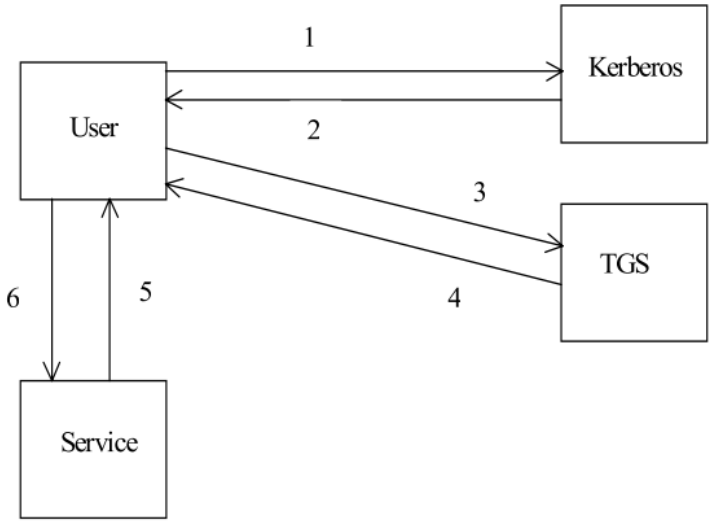
3. Alice elküldi Bob-nak a $w = R \cdot u (mod\ n)$
4. Bob ellenőrzi a $z \cdot v = w^2 (mod\ n)$ értéket

Alice így bizonyítja, hogy ismeri u értékét, Bob azonban csak R és z számokkal együtt tudná u-t kiszámítani, ezek közül csak egyet kap meg.
Ha Alice birtokában van az u titkos kulcsnak, akkor mindkét esetben probléma nélkül el tudja küldeni a megfelelő választ. Ha nem rendelkezik vele és át akarja verni Bob-ot, akkor meg kell próbálnia megjósolni b értékét:

1. Ha b = 0-ra tippel, akkor generál egy tetszőleges R számot, az első lépésben elküldi a négyzetes maradékát Bob-nak, és ha valóban b = 0-t kap vissza, akkor a harmadik lépésben elküldi R-t és sikerült az átverés. Ha b = 1, akkor nem tudja elhitetni Bob-bal. A siker esélye így $^1/_2$.
2. Ha b = 1-re tippel, akkor az első lépésben a $z = R \cdot v^{-1} (mod\ n)$ értéket küldi át, és ha valóban b = 1-et kap vissza, akkor a harmadik lépésben R-t. Ha b = 0, akkor az átverés nem sikerült, mert Alice nem tudja kiszámítani z modulo négyzetgyökét. A siker esélye így $^1/_2$.

Tehát Alice $^1/_2$ eséllyel verheti át Bob-ot. Azonban ha Bob k alkalommal játssza le az 1-4 lépéseket, akkor Alice esélye $2^{-k}$-ra csökken.

**52. Kerberos blokkvázlat**




A rendszer elemei:

- felhasználók
- Kerberos szerver
- jegyszolgáltató szerverek (Ticket Granting System)
- szolgáltatást futtató szerverek

**53. Kerberos ticket**

Egy kliens csak akkor használhat egy szolgáltatást, ha korábban szerzett erre egy jegyet, azaz speciális megbízólevelet. Ezt kell bemutatnia a szervernek, hogy bizonyítsa, valóban jogosult a szolgáltatás használatára. A jegyet tehát egyetlen szerver-kliens pár között használjuk és jegyosztótól (Ticket Granting System) szerezhető be.
A jegy tartalmazza a szerver nevét, a kliens nevét, a kliens IP-címét, az időbélyeget, a lejárati időt és viszonykulcsot.

$$T=\{ServerID, ClientID, ClientIPAddress, TimeStamp, ExpirationTime, SessionKey\}$$

**54. Kerberos authenticator**

A hitelesítő jegyet a kliens csak egyszer használhatja, ha új szolgáltatást akar igényelni, akkor újat kell generálnia. A hitelesítő a következőket tartalmazza: a kliens nevét, a munkaállomás IP-címét, a munkaállomás aktuális idejét. A hitelesítő jegyet a szerverre és kliensre kiadott kapcsolati kulccsal (session key) titkosítják.

$$KA = \{ClientID, ClientIPAddress, TimeStamp\}$$