Cryptography vizsgajegyzet

Áron Erdélyi

2021.09.22.

Contents

1 Beugró

1.1 Kis Fermat tétel

Tétel (Kis Fermat tétel)**.**

$$a^{p-1} \equiv 1 \mod p,$$

ahol p prímszám és $a \in [1, p-1]$ egész szám.

Bizonyítás. Írjuk fel a következő számokat:

$$a, 2a, 3a, \ldots, (p-1) \cdot a.$$

Indirekt módszerrel tegyük fel, hogy a felírt számok modulo p van kettő ugyan olyan eredmény:

$$i \cdot a \equiv j \cdot a \mod p \quad i \neq j \mod p.$$

$$(i-j) \cdot a \equiv 0 \mod p.$$

Ez azt jelentené, hogy p-vel osztható $(i-j) \cdot a$. Mivel p prímszám, ebből az következik, hogy vagy a, vagy $(i-j)$ osztható p-vel. Mivel $a \in [1, p-1]$ és $i \neq j \mod p$ ezért sem a, sem $i-j$ sem osztható p-vel. Tehát a felsorolt számok között nincsen egyenlő modulo p.

Vegyük a felsorolt számok halmazát és írjuk fel az első halmaz modulo p halmazát:

$$\{a, 2a, \ldots, (p-1) \cdot a \mod p\} = \{1, 2, \ldots, (p-1)\}.$$

Írjuk fel a két halmaz összes elemének a szorzatát modulo p:

$$a \cdot 2a \cdot \cdots \cdot (p-1) \cdot a \equiv 1 \cdot 2 \cdot \cdots \cdot (p-1) \mod p$$

$$(p-1)! \cdot a^{p-1} \equiv (p-1)! \mod p$$

$$a^{p-1} \equiv 1 \mod p$$

□

A tétel segítségével prímszámtesztelést lehet végrehajtani. Itt a tesztelni kívánt számot jelöljük p-vel. Végig próbáljuk $2 \leq a \leq p-1$ számokkal az előző tételben leírt ekvivalenciát, és ha nem tartja $\forall a \in [2, p-1]$-re, akkor p nem prím.

Vannak olyan p nem prím számok, amire $\forall a \in [2, p-1]$-re tartja az ekvivalenciát. Ezeket a számokat Carmichael számoknak nevezzük. Példa: $561 = 3 \cdot 11 \cdot 17$.

1.2 Diffie-Hellman kulcsegyeztetés

1.2.1 Nyilvános kulcsú titkosítási eljárás

A nyilvános kulcsú titkosítási eljárás arról szól, hogy az üzenet címzettjének elég a publikus kulcsát ismerni. Ennek a kulcsnak a tudatában bárki küldhet üzenetet a címzettnek, úgy hogy az üzenet dekódolása csak a címzettnek lehetséges.

Egy nyilvános kulcsú titkosítási rendszer két visszafejthetetlen algoritmus családból áll:

$$E_K : \{M\} \to \{C\},$$

$$D_K : \{C\} \to \{M\},$$

véges $\{M\}$ és $\{C\}$ üzenet terek fölött úgy, hogy

1. $\forall K \in \{K\}$, E_K a D_K inverze,
2. $\forall K \in \{K\}$, $M \in \{M\}$ és $C \in \{C\}$ E_K és D_K könnyen kiszámolható,
3. Majdnem minden $K \in \{K\}$-ra, minden könnyen kiszámítható, D_K-val egyenértékű algoritmus számítási szempontból nem levezethető E_K-ból.
4. $\forall K \in \{K\}$-ból kiszámítható az E_K, D_K inverz páros.

1.2.2 Nyilvános kulcs megosztási eljárás

A nyilvános kulcs megosztási eljárás a nyilvános hálózatokon való szimetrikus kulcs kommunikálására szolgáló eljárás. Ezzel az eljárással úgy tud két fél kulcsot egyeztetni, hogy egy nyílvános hálózaton senki más nem tudja meg a kulcsot.

Az új technika a logaritmus kiszámításának nehézségeit használja fel egy véges $GF(q)$ field fölött, ahol q prímszám szerepel.

$$Y = \alpha^X \mod q, \quad 1 \leq X \leq q-1,$$

ahol α a $GF(q)$ rögzített primitív eleme, akkor X-re Y logaritmusaként hivatkozunk az α bázishoz, mod q:

$$X = \log_\alpha Y \mod q, \quad 1 \leq Y \leq q-1.$$

Y kiszámítása X-ből egyszerű amely maximum $2 \cdot \log_2 q$ szorzásból áll, hiszen a hatványozás hatékony. Például $X = 18$,

$$Y = \alpha^{18} = (((\alpha^2)^2)^2)^2 \cdot \alpha^2.$$

Viszont X kiszámítása Y-ból sokkal nehezebb feladat. Jól választott q esetén $\sqrt{q}$ számítás kell.

Ezek alapján a kulcsegyeztetés a következő. Minden fehlasználó választ magának egy $X_i \in [1, q-1]$ számot. Ezt titokban tartja, de

$$Y_i = \alpha^{X_i} \mod q$$

értéket közzéteszi. Amikor i és j kommunikálni szeretne, akkor a

$$K_{ij} = \alpha^{X_i X_j} \mod q = Y_j^{X_i} \mod q = Y_i^{X_j} \mod q$$

közös kulcsot használják.

1.3 RSA algoritmus

Az (e, n) nyilvános kulccsal való M üzenet titkosításához a következő lépéseket kell végrehajtani (e és n pozitív egész számok).

1. Az üzenetet 0 és $n-1$ közötti egész számmal reprezentáljuk.
2. Titkosítjuk az üzenetet úgy, hogy az e-edik hatványra emeljük modulo n.
3. Visszafejtéshez a titkosított üzenetet a d-edik hatványra emeljük, modulo n.

Tehát a titkosító és a visszafejtő algoritmusok a következők:

$$C \equiv E(M) \equiv M^e \mod n,$$

$$D(C) \equiv C^d \mod n.$$

A titkosító kulcs tehát egy pozitív egész szám pár (e, n). Hasonlóan a visszafejtő kulcs is egy pozitív eglsz szám pár (d, n). Minden felhasználó nyilvánossá teszi a titkosító kulcsát és az ehhez tartozó visszafejtő kulcsot pedig titokban tartja.

1.3.1 Kulcspár generálás

Először kiszámoljuk n-t két nagy, random prímszám szorzataként:

$$n = p \cdot q.$$

Bár n publikus, p-t és q-t eldugjuk a nyílvánosság elől, mert a prímtényezőkre való bontás nehéz feladat.

Ezután kiválasztjuk d-t, ami egy nagy relatív prím $(p-1) \cdot (q-1)$-el. Azaz

$$\gcd(d, (p-1) \cdot (q-1)) = 1.$$

Végül kiszámoljuk e-t, ami a d multiplikatív inverze modulo $(p-1) \cdot (q-1)$:

$$e \cdot d \equiv 1 \mod (p-1) \cdot (q-1).$$

1.3.2 A generált kulcspár helyessége

Bármely M pozitív egész számú üzenetre, amely n-nel relatív prím,

$$M^{\phi(n)} \equiv 1 \mod n.$$

Itt $\phi(n)$ azt a pozitív eglsz számot jelenti, ahány n-nél kisebb relatív prímje van n-nek. A p prím számokra,

$$\phi(p) = p - 1.$$

Mivel $n = p \cdot q$ két prím szorzata, ezért

$$\phi(n) = \phi(p) \cdot \phi(q) = (p-1) \cdot (q-1) = n - (p+q) + 1.$$

Mivel d és $\phi(n)$ relatív prímek, van multiplikatív inverze az egész számok modulo $\phi(n)$ gyűrűn:

$$e \cdot d \equiv 1 \mod \phi(n).$$

$$M^{e \cdot d} \equiv M^{k \cdot \phi(n)+1} \equiv M \mod n.$$

Példa.

$$p = 7,\ q = 13,\ n = 7 \cdot 13 = 91.$$

$$d = 65,\ \gcd(65, 72) = 1.$$

$$e \cdot d \equiv 1 \mod \phi(n) \implies e = 41.$$

$$M = 15 \implies C \equiv M^e \mod n = 15^{41} \mod 91 = 71.$$

$$C^d \mod n = 71^{65} \mod 91 = 15 = M.$$

1.4 Euklideszi algoritmus és kibővített Euklideszi algoritmus

1.4.1 Euklideszi algoritmus

Két egész szám legnagyobb közös osztóját úgy tudjuk meghatározni, hogy prímtényezőkre bontjuk a számokat és a legnagyobb közös osztó a prímtényezőkre való bontások metszetének szorzata lesz. Az Euklideszi algoritmus arra ad választ, hogy mi két egész szám legnagyobb közös osztója prímtényezőkre való bontás nélkül.

Lemma. *Tegyük fel, hogy*

$$a = q \cdot b + r \implies \gcd(a, b) = \gcd(b, r).$$

Bizonyítás. A következő állítások igazak:

$$d|a \wedge d|b \implies d|r$$

$$d|b \wedge d|r \implies d|a$$

Tehát a és b közös osztói megegyeznek b és r közös osztóival. Tehát $\gcd(a, b) = \gcd(b, r)$. □

Tétel. *Legyen $a > b$, ekkor*

$$a = q_1 \cdot b + r_1 \quad 0 \leq r_1 < b$$

$$b = q_2 \cdot r_1 + r_2 \quad 0 \leq r_2 < r_1$$

$$r_1 = q_3 \cdot r_2 + r_3 \quad 0 \leq r_3 < r_2$$

$$\vdots$$

$$r_{s-1} = q_s \cdot r_s$$

$$\gcd(a, b) = r_s.$$

Bizonyítás. Az előző Lemmát alkalmazzuk minden sorra:

$$\gcd(a, b) = \gcd(b, r_1) = \gcd(r_1, r_2) = \cdots = \gcd(r_{s-1}, r_s) = \gcd(r_s, 1) = r_s.$$

□

Példa.

$$a = 55,\ b = 34$$
$$55 = 1 \cdot 34 + 21$$
$$34 = 1 \cdot 21 + 13$$
$$21 = 1 \cdot 13 + 8$$
$$13 = 1 \cdot 8 + 5$$
$$8 = 1 \cdot 5 + 3$$
$$5 = 1 \cdot 3 + 2$$
$$3 = 1 \cdot 2 + 1$$
$$2 = 2 \cdot 1$$
$$\gcd(55, 32) = 1.$$

1.4.2 Kibővített Euklideszi algoritmus

Ez az algoritmus választ ad nekünk arra a kérdésre, hogy mi a inverze modulo p.

Tétel. *Az előző tételből,*

$$r_1 = a - q_1 \cdot b$$
$$r_2 = b - q_2 \cdot r_1 = b - q_2 \cdot (a - q_1 \cdot b)$$
$$r_3 = r_1 - q_3 \cdot r_2 = a - q_1 \cdot b - q_3 \cdot (b - q_2 \cdot (a - q_1 \cdot b))$$
$$\vdots$$
$$r_s = r_{s-2} - q_s \cdot r_{s-1} = q_a \cdot a + q_b \cdot b.$$

Ha $\gcd(a, b) = 1$*, akkor*

$$1 = q_a \cdot a + q_b \cdot b.$$

Nézzük meg az előző egyenletet modulo b:

$$1 \equiv q_a \cdot a \mod b.$$

Ebben az esetben q_a az a multiplikatív inverze modulo b.

Példa. *Az előző példát felhasználva:*

$$21 = 1 \cdot 55 - 1 \cdot 34$$
$$13 = 34 - 21 = 34 - 1 \cdot (55 - 1 \cdot 34) = -1 \cdot 55 + 2 \cdot 34$$
$$8 = 21 - 13 = 55 - 1 \cdot 34 - 1 \cdot (34 - 1 \cdot (55 - 1 \cdot 34)) = 2 \cdot 55 - 3 \cdot 34$$
$$5 = 13 - 8 = 34 - 1 \cdot (55 - 1 \cdot 34) - 1 \cdot (55 - 1 \cdot 34 - 1 \cdot (34 - 1 \cdot (55 - 1 \cdot 34))) = -3 \cdot 55 + 5 \cdot 34$$
$$3 = 5 \cdot 55 - 8 \cdot 34$$
$$2 = -8 \cdot 55 + 13 \cdot 34$$
$$1 = 13 \cdot 55 - 21 \cdot 34$$

Vagy táblázatos formában csak az együtthatókat tárolva:

	55	*34*
55	*1*	*0*
34	*0*	*1*
21	*1*	*-1*
13	*-1*	*2*
8	*2*	*-3*
5	*-3*	*5*
3	*5*	*-8*
2	*-8*	*13*
1	*13*	*-21*

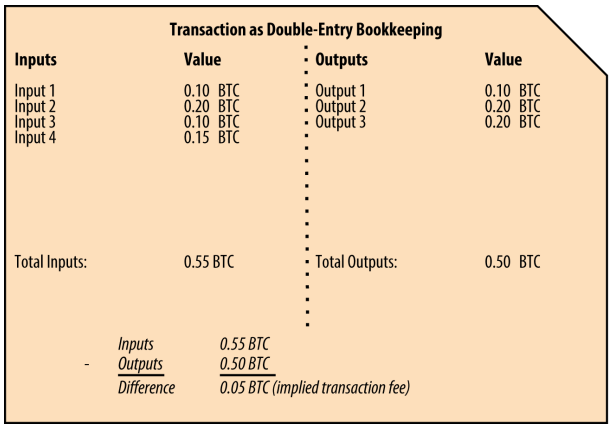
$$1 = 13 \cdot 55 \mod 34$$

Tehát az 55 multiplikatív inverze modulo 34 az 13.

2 Bitcoin

2.1 Bitcoin tranzakciók

Egy bitcoin tranzakcióban külön eltároljuk a bemeneteket és a kimeneteket.



Ez azért van, mert a bitcoinok csak egészben használhatók fel. Tehát össze kell szedni annyi bitcoint, ami legalább a tranzakció értéke, és a maradékot pedig magunknak visszajáróként kiutaljuk.

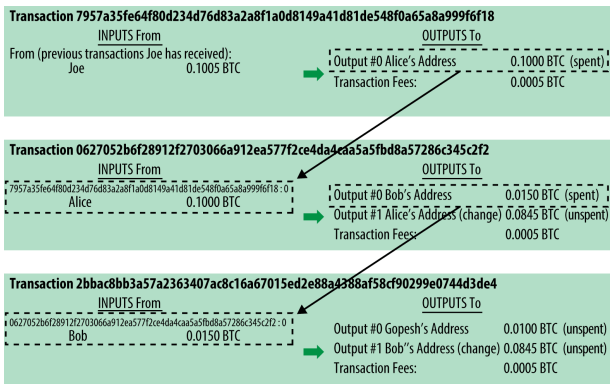
Az inputok összege valamivel több, mint az outputok összege. A különbségét a bányász kapja meg bányászati díjként. Ez azt is megakadályozza, hogy ne lehessen elárasztani a rendszert felesleges tranzakciókkal.

Olyan tranzakció is elképzelhető, hogy sok kis bitcoint váltok be egy nagy értékűre. Vagy ennek a fordítottja, nagy értékű bitcoint szét lehet osztani sokfelé.

2.1.1 Tranzakció készítése

Sok olyan hely van, ahol a teljes blokklánc tárolva van. A felhasználható bitcoin, az, amit megkaptunk valahonnan, de még nem volt tranzakció inputja. Ezt általában satoshiban (0.00000001 BTC) tárolják.

Ezután létrehozzuk az outputok listáját, az előzőekben leírtak alapján. Ha ez kész van, akkor a pénztárca továbbítja ezeket egy vagy több nodehoz. Amint egy node kap egy valid tranzakciót, ezt továbbítja az összes többi, vele kapcsolatban álló nodenak. Ezt a továbbítási technikát "flooding"-nak nevezzük. Pár másodperc alatt eléri a legtöbb nodeot a világon.



A tranzakciós láncban egy bitcoin eredetét lehet visszakeresni, a létrehozásáig.

2.2 Bányászás

A tranzakció azáltal válik hitelessé, hogy bekerül egy blokkba. Ezeknek a blokkoknak az összeállítása és hitelesítése a bányászok feladata. A bitcoincan való megbízást az adja, hogy egy új blokk hitelesítéséhez nagyon nagy számítási kapacitás kell, de egyszerűen leellenőrizhető (Proof Of Work).

A bitcoin cányászat nehézsége úgy van beállítva, hogy körülbelül 10 percenként lehet kibányászni egy blokkot. A bányászás során a bányászok sok hash függvényt hajtnak végre. Az első bányász, aki kibányássza a blokkot, közzéteszi és ezzel megnyeri ezt a "kört".

Egy blokk hitelesítését úgy lehet elérni, hogy a blokk header hashe kisebb lesz mint egy megadott - ú.n. target value - szám. A headerben nincs túl sok dolog, ami változtatható, egyedül a nonce, ami inkrementálható.

A target value csükkentésével egyre nehezebb feladattá lehet tenni a bányászást. Tehát ha nő a bányászok száma, és a hatékonyság, akkor a rendszer lejjebb viszi a target valuet úgy, hogy átlagosan 10 perc legyen kibányászni egy blokkot. Ez 2016 blokkonként van újrakalkulálva (kb két hét).

Ha egy bányász nyer egy adott blokk kibányászásában, akkor jutalmul kap valamennyi bitcoint. Ez a jelenleg forgalomban lévő bitcoinok számától függ, és ahogy egyre több bitcoin lesz, egyre kevesebbet kapnak a bányászok. Ez kb 21 milliárd bitcoinnál fog tetőzni, annál több bitcoin nem lesz a rendszerben. Eredetileg 50 BTC járt egy blokkért. Ez az érték minden 210000 blokk kibányászása után (kb 4 év) megfeleződik.

Amint valaki megfejt egy blokkot, minden bányász elkezd az új blokkon dolgozni. Új tranzakciók folyamatosan folynak be a hálózatba, ezekből válogathatnak a bányászok, bányászati ár függvényében. Ezekhez a tranzakciókhoz a bányász hozzáadja a saját magának kiutalásra kerülő bányászati díjat, ami az új bitcoinokból (jelenleg blokkonként 6.25 BTC), valamint a tranzakciós díjakból áll.

2.3 Node típusok

- **Reference Client (Bitcoin core):** Pénztárca, bányász, teljes Blockchain adatbázis és network routing node a bitcoin P2P hálózaton.
- **Full Block Chain Node:** Tartalmazza a teljes blockchain adatbázist valamint a network routing nodeot.
- **Solo miner:** Tatalmaz bányász funkciókat a teljes blockchain adatbázissal és egy network nodeal együtt.
- **Lightweight (SVP) wallet:** Pénztárca és network node funkciókat tartalmaz.

A bitcoin hálózat ilyen típusú nodeokból épül fel. Vannak speciális hálózati protokollal rendelkező részhalmazok, amelyek bányász poolok.

Minden tárcához tartozik privát kulcs, amiből kiszámítható a publikus kulcs, amelynek a lenyomata a bitcoin cím. Ezek fordított irányba nem előállíthatók.

2.4 Blockchain

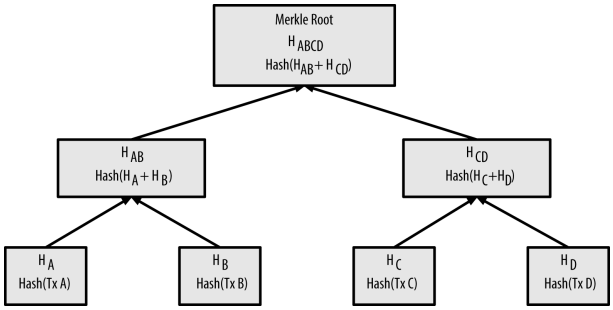
Minden blokkot a hash alapján lehet egyértelműen azonosítani. Ezt a blokk headerből lehet generálni. A blokk header 80 byte adatot tartalmaz. Ebben benne van az előző blokk header hashe, a blokk timestampje, a nehézség, a noce, s Merkle root, valamint a tranzakciók hashe. A tranzakciók nagysága kb 1Mbyte.

A blockchain szekvenciális lánc jellege miatt visszakövethető az ú.n. genesis blokkig.

2.4.1 Merkle Root

A headerben van egy összefoglaló a tranzakciókról a Merkle fa segítségével. A Merkle fa egy bináris hash fa arra van használva, hogy hatákonyan megerősítse nagy halmazok integritását. N hashelt adat elem esetén egy Merkle fa segítségével $2 \cdot \log_2 N$ számítással meg lehet állapítani, hogy az adat benne van-e a fában.

Lightweight nodeoknál a tranzakció hitelesítése a block headereken és a Merkle hash path alapján történik.



2.4.2 Header

A header létrehozásához az alábbi hat adatra van szükség:

- Verzió (4 byte)
- Előző block header hashe (32 byte)
- Merkle root (32 byte)
- Timestamp (4 byte)
- Target value (4 byte)
- Nonce (4 byte)

Bányászásnál nem csak a nonce érték változhat, hanem például a blokk timestampjével is lehet valamilyen szinten variálni a hash-t.

Az általános megegyezés, hogy az a blokklánc az aktuális, amelyik a leghosszabb. Ez azzal is jár, hogy lehetnek másodlagos láncok, amelyek akár át is vehetik a "vezetést", ha gyorsabban, több blokkot bányásznak ki a másodlagos láncba, mint az elsődlegesbe.

3 Véletlenszám generátorok

3.1 Álvéletlen generátorok

3.1.1 Excel RAND fv.

Első elem:

$$r(1) = 9821 \cdot 0.5 + 0.211327 \text{ törtrésze.}$$

Továbbiak:

$$r(n+1) = 9821 \cdot r(n) + 0.211327 \text{ törtrésze.}$$

Be lehet állítani, hogy a rendszeróra alapján más magértékkel kezdődjön.

Hagyományos statisztikai tesztek:

- 1-esek, 0-k gyakorisága
- 00, 01, 10, 11 mintág gyakorisága
- 0, 00, 000, 0000, stb minták gyakorisága
- autokorreláció

Erre vonatkozóan standardnak tekinthető véletlenszám generátor tesztek jöttek létre, például FIPS 140, Diehard teszt, stb.

3.1.2 Wichmann-Hill algoritmus

$X(0),\ Y(0),\ Z(0) = 1$ és 30000 közötti egész

$$X(n) = 171 \cdot X(n-1) \mod 30269$$

$$Y(n) = 172 \cdot Y(n-1) \mod 30307$$

$$Z(n) = 170 \cdot Z(n-1) \mod 30323$$

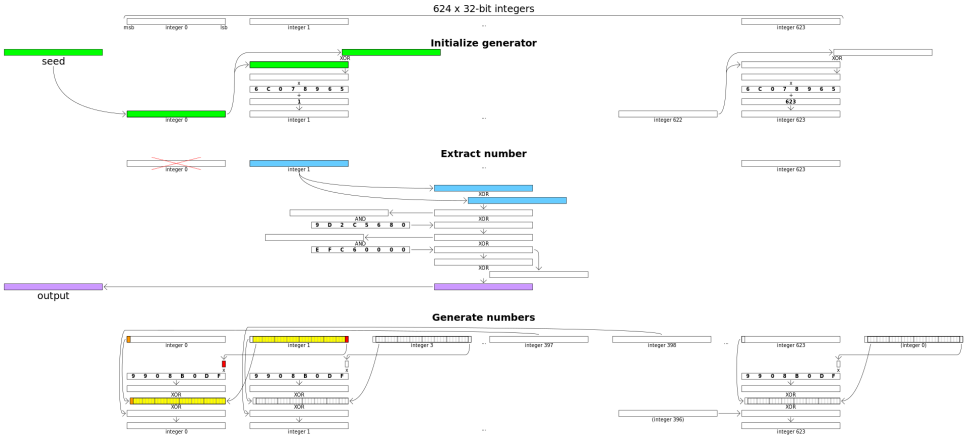
$$r(n) = X(n)/30269 + Y(n)/30307 + Z(n)/30323 \text{ törtrésze.}$$

Ennek hosszú ciklusa van és megfelel a FIPS-nek és a Diehard tesztnek is.

3.1.3 Mersenne Twister

Legelterjedtebb véletéengenerátor (nem kriptográfiai célra).

32 bites magérték generálása, majd ebből létrehozunk 624 darab 32 bites egész számot, majd ebből további egyszerű bitszintű shiftelésekkel és műveletekkel hozahtó létre a következő állapot, amiből egy 32 bites kimenet generálható.



Ennek a ciklusideje $2^{624 \cdot 32} \approx 2^{20000}$.

3.1.4 MS CryptGenRandom

Kiinduló érték képzés az alábbi adatok összefűzésével és hashelésével:

- Jelenlegi process ID
- Jelenlegi thread ID
- 32bit tick count system bootolás óta
- Jelenlegi lokális dátum és idő
- Lokális idővel kapcsolatos metaadatok
- A rendszer hardwarejének metaadatai

A függvény hívásakor megadható a byteok száma, illetve egyéb opcionális entropy bemenetek.

3.1.5 FIPS186-2

Digitális aláírásszabvány, amely előírásokat tartalmaz véletlen számok generálására is:

$$XKEY(1) : seed$$

véletlen bitek (r) generálásának lépései:

$$XVAL(n) = XKEY(n) + XSEED(n)$$
$$r(n) = SHA1(IV, XVAL(n))$$
$$XKEY(n+1) = XKEY(n) + r(n) + 1$$

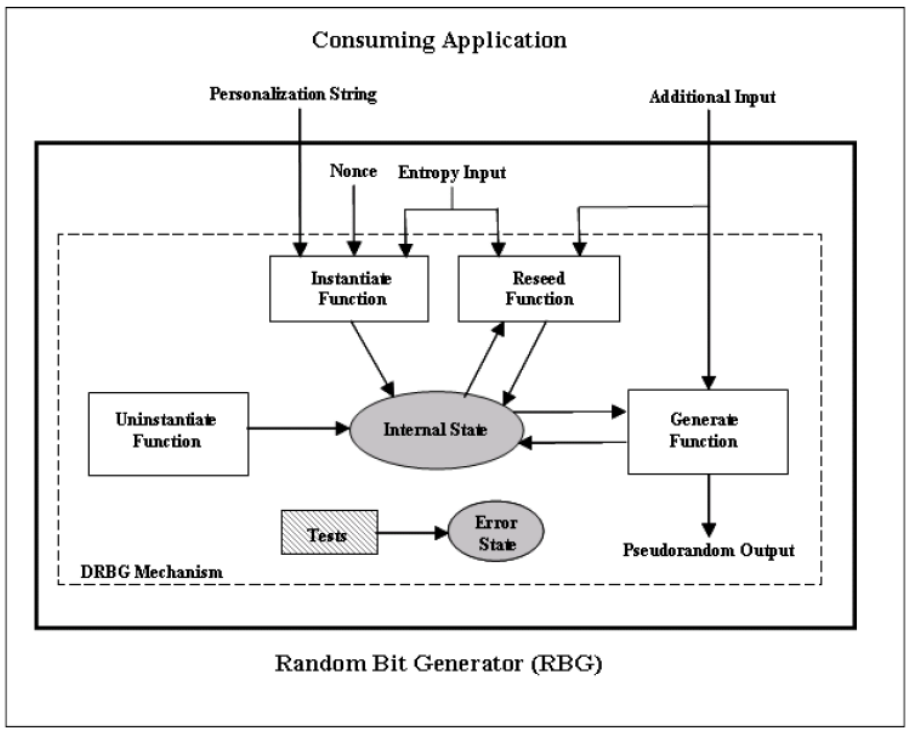
Itt $XSEED(n)$ opcionális további seed biteket, IV az állandó initial valuet jelenti.

3.2 Kriptográfiailag biztonságos determinisztikus random bit generátorok

3.2.1 Fogalmak

- **RBG:** Random bit generátor
- **NRBG:** Nem-determinisztikus random bit generátor
- **DRBG:** Determinisztikus random bit generátor (Pszeudorandom bit generátor)

3.2.2 Funkcionális modell



- Entrópia:
  - magérték létrehozására
  - titkos
  - nem a hossza a meghatározott, hanem a bizonytalansága
- Egyéb bemenet:
  - nonce, személyre szabott érték, stb.
  - nem feltétlenül titkos
- Állapot: Minden belső adat, ami a következő állapot számításához kell
- Műveletek:
  - Egyed létrehozása (instantiate)
  - Új magérték létrehozása (reseed)
  - Törlés (uninstantiate)
  - Gyártás (generate)
- Teszt

Elvárás:

- **Backtracking resistance:** Ismertté vált állapot alapján a korábbi állapotok nem állíthatók elő, illetve a korábbi kimenet nem különböztethető meg a véletlentől. Ez mindig elvárható.
- **Prediction resistance:** Ismertté vált állapot alapján későbbi állapotok nem állíthatók elő, valamint későbbi kimenetek nem különböztethetők meg a véletlentől. Ez nem várható el mindig. Ha szükséges, akkor reseeding kell minden alkalommal.

Az új állapot előállítása hasheléssel, vagy titkosítással történik. Ezen kívül még a Blum–Blum–Shub algoritmust és blokk rejtjelezőket is használhatunk.

3.2.3 Blum–Blum–Shub generátor

Matematikalilag bizonyított, hogy jó generátor, csak elég lassú. Lépései:

1. p és q két nagy Blum prím generálása
2. $n : (p \cdot q$
3. Választunk egy $s \in [1, n-1]$ random seedet.
4. $x_0 := s^2 \mod n$
5. Az állapot szekvenciája $x_i := x_{i-1}^2 \mod n$ és $z_i := \mathit{parity}(x_i)$.
6. A kimenet $z_1, z_2, \ldots,$

ahol a $\mathit{parity}(x_i) = R_2(x_i)$.

Példa.

$$n = p \cdot q = 7 \cdot 19 = 133, \quad s = 100.$$
$$x_0 = 100^2 \mod 133 = 25.$$
$$x_1 = 25^2 \mod 133 = 93$$
$$x_2 = 93^2 \mod 133 = 4$$
$$x_3 = 4^2 \mod 133 = 16$$
$$x_4 = 16^2 \mod 133 = 123$$

Ezekből a kimenet: $1, 0, 0, 1$.

4 Adatfolyam rejtjelezők

A szimmetrikus kulcsó rejtjelezők két nagy családja a blokk rejtjelezők és az adatfolyam rejtjelezők. A blokk rejtjelezők tetszőleges hosszúságú üzenetet felszeletelnek előre meghatározott hosszúságú blokkokra, majd ezeket a blokkokat titkosítják (helyettesítés, vagy felcserélés).

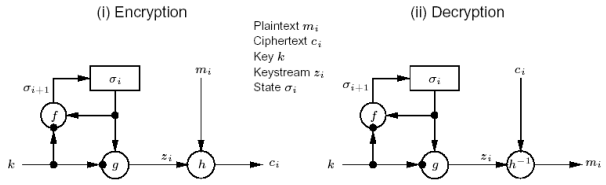
Az adatfolyam rejtjelezők könnyen összekeverhetők a véletlenszám generátorokkal, hasonlóan működnek. A lényege, hogy szimbólumonként rejtjelez, szimbólumról szimbólumra változó kulccsal. Például a Vernam rejtjelező, amely kizáró vagy kapcsolatba hozza a kulcsot és az üzenetet, majd a visszafejtésnél uyan ezt meg kell csinálni az eredeti üzenet visszakapásához.

4.1 Osztályozás

4.1.1 Szinkron

Álvéletlen bitsorozat a nyílt szövegtől független. Szinkronizálni kell, de nincs hibaterjedés bit módosulás esetén. Támadási lehetőség nyílik, mert bit módosítás esetén a rejtjelezett üzenetben a nyílt üzenet bitet átbillenti.

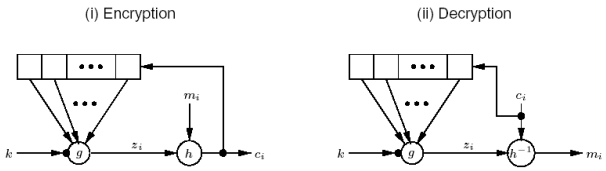
Leggyakrabban kizáró vagy operátor.



Minden jó álvéletlen generátor jó adatfolyam kódoló is elvileg. A gyorsaság fontosabb az adatfolyam rejtjelező esetében. Véletlenszerűség fontosabb az álvéletlen generátor esetében. Újrainicializáció az adatfolyam rejtjelező esetében fontosabb. Backtracking resistance az álvéletlen generátor esetében fontosabb.

4.1.2 Önszinkronizáló

Álvéletlen bitsorozat a rejtjelezett üzenet bitjeiből lesz kiszámolva. Szinkronizálja magát, de van hibaterjedés.



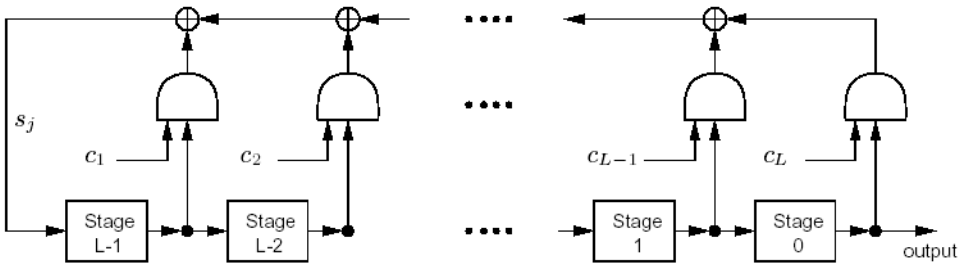
4.1.3 További osztályozás

- LFSR (linear feedback shift register) alapú
  - Hardware megvalósíthatóság
  - Nagy periódus
  - Jó statisztikai tulajdonságok
- Blokk rejtjelező alapú: minden blokk rejtjelezőhöz bizonyos működési mód esetén
  - CTR (counter)
  - OFB (output feedback)
  - CFB (cipher feedback)
- Számelméleti probléma alapú: Blum–Blum–Shub
- Dedikált:
  - RC4
  - TRIVIUM

– SALSA
– CHACHA
– stb...

4.2 LFSR adatfolyam rejtjelezők

Jól ismert és matematikailag jól leírható műveletet hajt végre a lineárisan visszacsatolt shift regiszter.



Visszacsatolási karakterisztikus polinom:

$$C(D) = 1 + c_1 D + c_2 D^2 + \cdots + c_L D^L.$$

A lineáris működés nem biztosít kellően biztonságos kulcsfolyamot, ezért más konstrukciókban, ahol több LFSR-nek vesszük nemlineárisan a kimeneteit.

4.3 RC4 algoritmus

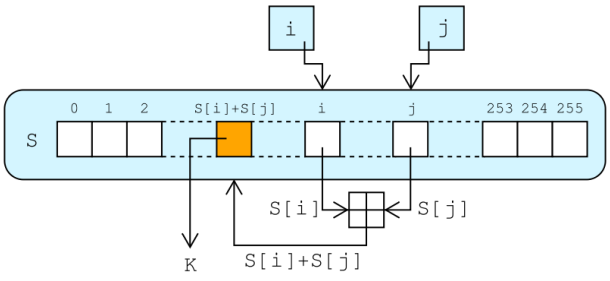
Volt licenszdíja, de mégis ez lett az egyik legelterjedtebb adatfolyam rejtjelező volt sokáig (WiFi, SSL/TLS, SS, PPP). Ez az algoritmus ma már nem biztonságos. Maga a folyamat olyan mint egy kártyapakli megkeverése.

Algorithm 1: Bit generálás

```
for i from 0 to 255 do
    S[i] := i
end
j:=0
for i from 0 to 255 do
    j:=(j+S[i]+key[i mod keylength]) mod 256
    swap(&S[i],&S[j])
end
```

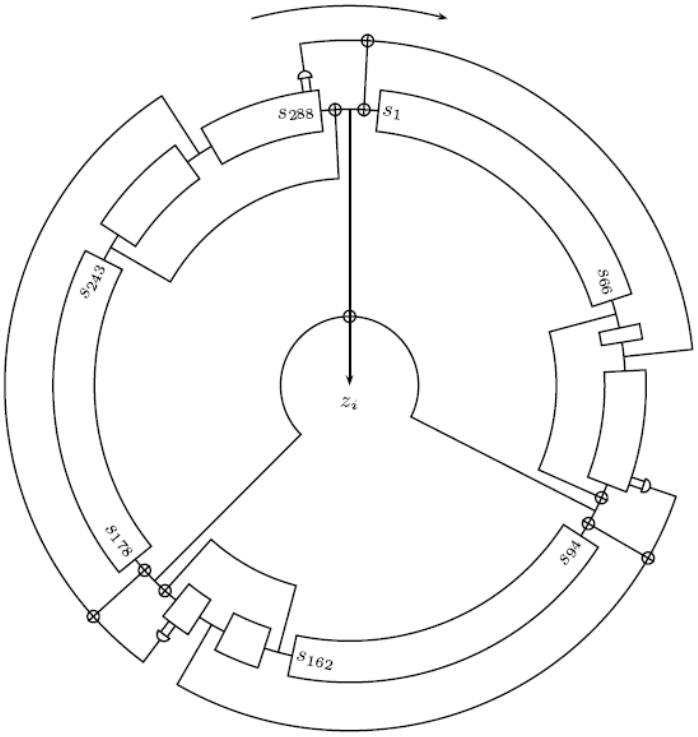
Algorithm 2: Bit generálás

```
i:=0
j:=0
while GeneratingOutput do
    i:=(i+1) mod 256
    j:=(j+S[i]) mod 265
    swap(&S[i],&S[j])
    output S[(S[i]+S[j]) mod 256]
end
```



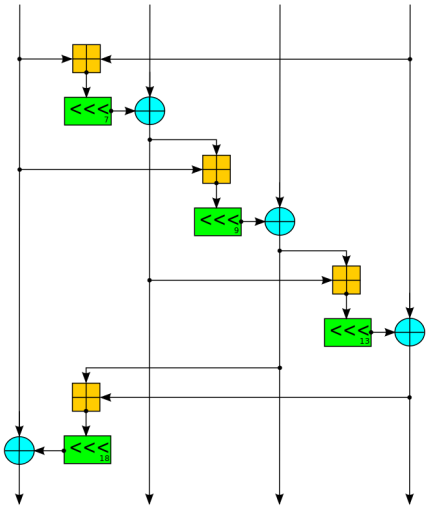
4.4 TRIVIUM

Hardware hatékony algoritmus, amely 288 állapotbitből áll.



4.5 SALSA, CHACHA

Software hatékony algoritmus. Az állapotvektor 16 db 32 bites szóból áll. Az algoritmus 4 párhuzamosítható körben egyszerre négy szóval végzi el az alábbi műveleteket:



Ahol $\oplus$ az bit szerinti kizáró vagy operáció, $\boxplus$ 32 bites összeadás mod 2^{32} és <<< pedig bit rotáció. Ennek a CHACHA nevű változatát a google használja: Android, Chrome.

5 WiFi biztonság

OSI modell:

1. **Fizikai réteg:** jel fizikai átvitele a médiumon.
2. **Adatkapcsolati réteg:** Pont-pont kapcsolat az eszközök és a hálózati csomópontok között. Címzés egyedi MAC cím alapján. Függ a hálózat típusától (Ethernet vagy WLAN).
3. **Hálózati réteg:** IP csomagok közlekednek a forrás és a cél között.

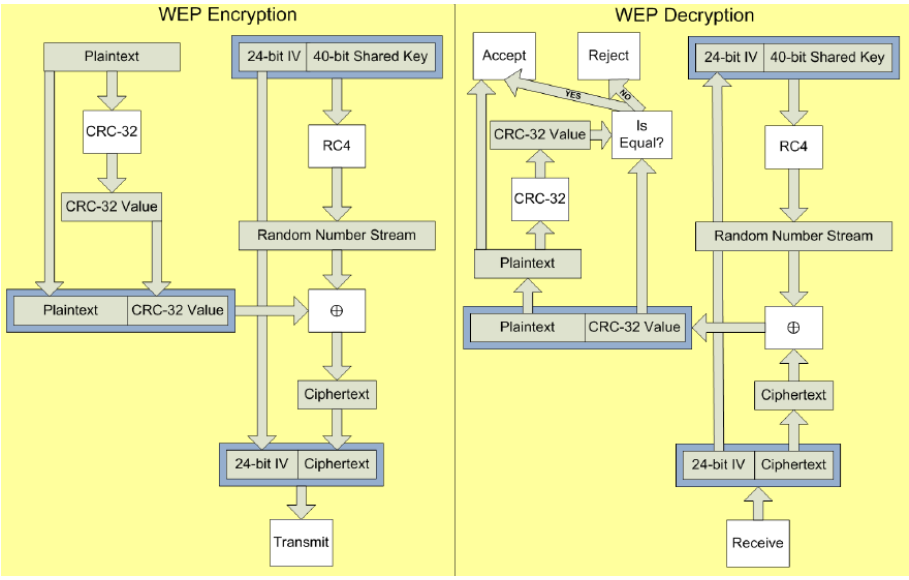
WiFi biztonsága 2004 körül lett fontos, ekkor kijött a WPA2 (WiFi Protected Access) szabvány idáig a WEP volt az elfogadott szabvány.

Üzenet esetén MSDU-kra (MAC Service Data Unit) bontjuk az özeneteket, melyeket headerrel és ellenőrzőösszeggel ellátva küldünk tovább, a fizikai réteg felé. Az MSDU titkosítás esetén megnő 8 byteal. Így a MAC Protocol Data Unit (MPDU): MSDU + header (max. 30 byte) + ellenőrzőösszeg (4 byte).

5.1 WEP titkosítás

WEP (Wired Equivalent Privacy) titkosítás használható előre megosztott kulcs alapján. Ez valójában autentikációt is biztosít. Biztonságosabb, mint a shared key authentication.

Eredetileg 40 bites kulcs + 24 bit IV, később 104 bit kulcs + 24 bit IV. A WEP RC4 algoritmussal titkosít. A kulcsfolyam generálása Vernam rejtjelező.



5.1.1 Ellenőrzőösszeg sérülékenysége

Érvényes ciphertext hozható létre a ciphertext módosításával. Ha egy lehallgatott, érvényes ciphertext adat mezőjét megváltoztatjuk és az alábbi módon származtatjuk a módosított ciphertext-et:

$$C' = C \ XOR \ ([\text{változás}, CRC-32(\text{változás})]),$$

akkor a fogadó félnél az integritás ellenőrzés nem mutatja ki a módosítást. Ez a probléma abból ered, hogy a CRC-32 leképezés és a Veram rejtjelezés is lineáris függvény.

5.1.2 IV ismétlődés

24 bites az IV. Pár ezer frame-ben valószínűleg van két azonos IV. Azonos IV esetén megyegyeznek a kulcsfolyamok, így a két ciphertext különbsége a két cleartext különbségével.

5.1.3 Frame beszúrás

DoS támadás intézhető egy érvényes MPDU üzenet ismétlésével. Az AP elfogadja az üzenetet, mivel az IV véletlenszerű, így nem zátható ki, az ismétlődés. Kicsomagolja a tartalmát, majd továbbadja a magasabb rétegek felé.

5.1.4 Kulcs management

Elvileg lehetséges lenne minden kliensnek külön kulcsot adni, de ez bonyolult. A gyakorlat az, hogy minden kliens ugyanazt a kulcsot kapja. Akinek megvan a kulcs, minden kommunikációt dekódolni tud. Egymás üzeneteit is tudják dekódolni a WLANon belül. Ha egy eszköz kompromitálódik, mindent le tud hallgatni.

5.2 Shared Key Authentication

- Kliens kapcsolódni akar
- AP véletlen kihívást küld
- Kliens visszaküldi az IV-t, és a WEP([kihívás, chechsum],[IV,shared key])
- AP dekódol, összevet, nyugtáz.

5.3 WEP – SKA sérülékenység

Ha a támadó ismeri a kihívást, és a WEP([kihívás, checksum],[IV,shared key])-t akkor a kulcsfolyam számítható a Vernam típusú adatfolyam rejtjelező miatt. Ugyan azzal az IV-vel később autentikálhat. Az a probléma, hogy az IV-t a kliens határozza meg és nyíltan küldi el.

Ha a titkosítás is be van állítva, bár sikeresen autentikálni tud, de kommunikálni nem fog tudni a támadó. Tilos használni az SKA-t titkosítás nélkül, mert nem biztosít autentikációt.

5.3.1 WEP – SK felderítése az AP támadásával

ARP (Adress resolution protocol) az IP cím és a MAC között teremt kapcsolatot. A rejtjelezett frame első 16 bytejából 16 byte key stream meghatározható. Ez a frame beszúrást teszi lehetővé. Ugyan azt az ARP kérést számtalanszor elküldi a támadó, amire mindig új IV-vel kap választ, amiből mindig kinyerhető a 16 byte key stream. Megfelelő mennyiségű key stream alapján a shared key megállapítható. Szükséges idő: 1 perc.

5.3.2 WEP – SK felderítése a kliens támadásával

Bekapcsolt kliens közelében kell lenni. A kliens folyamatosan keresi az elérhető AP-ket. A támadó lehallgatja az SSID-t és megszemélyesíti az AP-t, küld egy kihívást. A kliens elküldi a titkosított kihívást. AP visszaküldi, hogy sikeres autentikáció. A kliens újabb titkosított üzeneteket küld. Néhány óra alatt a kulcs megfejtéséhez szükséges mennyiségű adat keletkezik.

5.4 WPA, WPA2

2004-ben a hibák miatt egy átmeneti (WPA) és egy végleges (WPA2) megoldást alkalmaztak. A WPA még RC4-el működött, de a WPA2 már az AES blokk rejtjelező segítségével rejtjelezett.

Újítás volt mindkettőnél, hogy van egy kulcs hierarchia, tehát nem minden kommunikációnál ugyan az a shared key van használva. Nem csak az AP hitelesít, hanem a kliens is. Valamint a hálózati hitelesítésből származik a kulcs.

6 GSM biztonság

6.1 GSM biztonsági célok és képességek

Az operátor szempontjából fontos

- Az ügyfél azonosítása a számlázás érdekében
- Csalás elketülése (nem fizet, más fizet)
- A szolgáltatás működőképességének végelme
- Védelem a többi operátortól

Ügyfelek szempontjából fontos, hogy legyen biztonság, hiszen nem szeretnénk, hogy mások lehallgathassák a veszélgetéseinket. A kialakított rendszer képességei:

- Az ügyfél kulcsa független a készüléktől, tehát a telefon lecserélése nem veszélyezteti a biztonságot.
- Az ügyfél anonimitásának védelme, tehát a rádiós kommunikáció lehallgatása nem fedi fel az ügyfél kilétét.
- A készülék letiltható, azaz a készülék is nyilván van tartva - elveszés, kompromozáció esetére.
- Az ügyfél hitelességének ellenőrzése, azaz hogy ki használja a szolgáltatást.
- Védelem a lehallgatás ellen.

6.2 GSM architektúra

6.2.1 GSM Mobile Station

- Mobile equipment (ME):
  - Fizikai telefonkészülék
  - azonosító: IMEI - International Mobile Equipment Identity
- Subscriber Identity Module (SIM): Intelligens kártya (Smart card):
  - Azonosítók
  - Kulcsok
  - Algoritmusok
  - Operátor preferenciák
  - Telefonszámok, SMS-ek, stb tárolása

6.2.2 SIM azonosítók

IMSI (International Mobile Subscriber Identity): Mobil ország kód, mobil hálózati kód, Mobil Subscriber azonosító.

TMSI (Temporary Mobile Subscriber Identity).

LAI (Local Area Identity): Aktuális bázisállomás adatai.

Az ügyfélnek van adatbázisa, ami alapján összekapcsolható az ügyfél az azonosítóval. Tehát ez nem a telefonszám, hanem a SIM azonosítója.

6.2.3 Sim kulcsok, algoritmusok

Kulcsok:

- K_i (Subscriber Authentication Key): Ügyfélkulcs
- K_c (Session key): Beszélgetés titkosító kulcsa
- CKSN (Ciphering Key Sequence Number): nem kell mindig új session key-t létrehozni, hanem ezen a számon keresztül hivatkozható egy már korábban létrehozott.
- Pin (Personal Identity Number): SIM védelme.
- PUK: PIN feloldása

Algoritmusok:

- A3: autentikáció
- A8: session kulcs létrehozása

6.2.4 Base Station Subsystem (BSS)

Base Transceiver Station (BTS):

- Bázisállomás: Egy cella rádió adó-vevője és a vezeték nélküli kommunikációs protokoll kezelése.
- a BTS azonosítója a Cell Global Identification (CGI), ami a Location Area Identity (LAI) és a Cell Identity összefűzése.

Base Station Controller (BSC): egy vagy több BTS irányítása és összekapcsolása a hálózat központi elemeivel.

6.2.5 Network Switching Subsystem (NSS)

Bázis állomások között tartja a kapcsolatot, illetve ezeket irányítja.

- Mobile Switching Center (MSC): hívások lebonyolítása
- Home Location Register (HLR): a szolgáltató ügyfelei és aktuális lokációjuk
- Visitor Location Register (VLR): saját és roaming felhasználól az adott szolgáltató területén
- Equipment Identity Register (EIR): Letiltott telefonok adatbázisa (IMEI alapján)
- Authentication Center (AUC): Saját ügyfelek ügyfélkulcsa (K_i)

6.2.6 Ügyfélkulcs kezelése

K_i – Subscriber Authentication Key:

- 128 bites szimmetrikus kulcs
- Ez alapján hitelesíti az operátor
- Ebből származik a telefon és a bázisállomás közötti bizalmas kommunikáció titkosító kulcsa
- Tárolása: SIM, AUC.

A kulcs a SIM-mel átvihető másik telefonra.

Az ügyfél azonosítása az IMSI vagy TMSI alapján történik. Csak az operátor hitelesíti az ügyfelet, fordítva nem. A hitelesítés egy véletlen kihívásra adott válasz ellenőrzése alapján történik, és ebben a lépésben létre jön a session key is.

6.3 A3 és A8 algoritmus

6.3.1 A3 algoritmus: autentikáció

A véletlen kihívásra aláírt válasz generálása.

6.3.2 A8 algoritmus: session key generálása

A véletlen kihívás értékéből a K_c kulcs generálása.

6.3.3 COMP128

A COMP128 algoritmus hozza létre egy lépésben a $SRES$ és K_c értékeket:

- 128 bites hash függvény
- a K_c utolsó 10 bitje: 0, így valójában 54 bites a kulcs.

Hiba: 150000 lekérdezéssel előállítható a K_i kulcs. Új verziók jöttek ki, de SIM kártya cseréje szükséges.

6.4 A5 – titkosító algoritmus

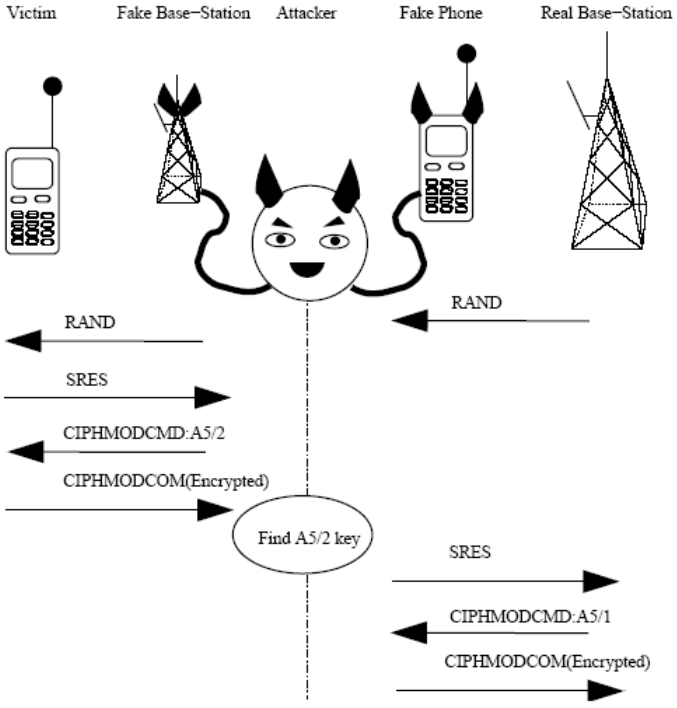
Egy hardwarere optimalizált adatfolyam rejtjelező. Eredetileg titkos volt az algoritmus, majd 1994-ben kisziváárgott. Változatai:

- A5/0: nincs titkosítás – harmadik világ
- A5/1: erős(ebb): nyugat európa, USA
- A5/2: gyenge: szándékosan, többi ország
- A5/3: GPRS, EDGE, 3G

Ez egy szinkron adatfolyam rejtjelező, amely 3 LFSR-t tartalmaz (19, 22, 23 bites). Minden regiszter egyik bitje határozza meg, hogy az adott regiszter léptetése megtörténik-e. Az adott regiszter akkor léptet, ha az órajel bitje megegyezik valamelyik másik órajel bittel.

6.5 Man-in-the-Middle attack

A támadó a bázis felé eszköznek, az eszköz felé bázisnak néz ki. Továbbítja a challenget a telefonnak, majd A5/2-vel kezd el a teleofnnal beszélni, amelyből kiszámolja a K_c-t.



7 Elliptikus görbék

Az elliptikus görbékkel úgyan olyan biztonságos, de kisebb kulcsokat tudunk generálni mint az RSA algoritmussal. Míg az RSA módszerreé inkább exponenciálisan egyre több bitet kell használni, az elliptikus görbékkel generált kulcspár csak lineárisan növekszik. A jelenleg ajánlott biztonsági szinthez az elliptikus görbékkel tizedakkora kulcsot kell generálni, mint RSA-val.

Ezen kívül az elliptikus görbéket prímtényezőkre való bontásra és prímtesztelésre is lehet használni.

Definíció (Folytonos síkban definiált elliptikus görbe)**.** $y^2 = f(x)$ *alakú egyenlettel definiált síkbeli görbe, ahol*

- $f(x)$ *egy* $x^3 + ax + b$ *alakú kifejezés,*
- x, y *valós változók,*
- a, b *egész számok,*
- *a görbe nem szinguláris.*

Az elliptikus görbe diszkriminánsa:

$$D = 4 \cdot a^3 + 27 \cdot b^2$$

$D = 0$ *esetén szinguláris.*

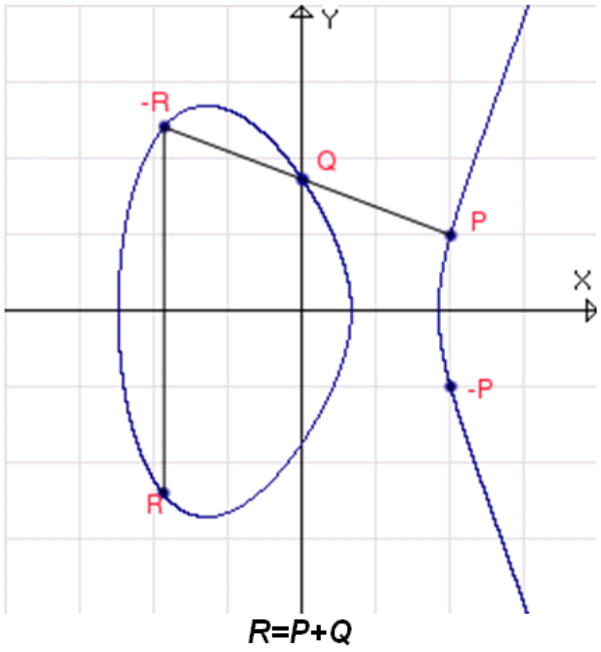
Az érintő és metszéspontok számát tekintve a függőleges egyenes másképpen viselkedik. Legyen a görbe része a függőleges egyenesek ideális (végtelen távoli) pontja is. Ekkor már egyformán viselkednek (az érintő két metszéspontnak számít).

7.1 Műveletek

7.1.1 Összeadás

Az ideális pontról feltételezzük, hogy rajta van minden függőleges egyenesen és hogy az x tengelyre vonatkozó tükörképe önmaga.

A görbe P és Q pontjainak $P + Q$ összege a P, Q pontokon átmenő egyenes és a görbe harmadik metszéspontja $-R$; ennek az x-tengelyre való R tülörképe a $P + Q$ összeg.



Ennek a műveletnek megvan a csoport tulajdonsága:

- Van egységelem: O (ideális pont),
- Van inverz: tükörkép,

- Asszociatív: $(A + B) + C = A + (B + C)$

Ez azért jelentős, mert gyorsan számítha az összeadás. Például:

$$29 \cdot A = 2 \cdot 2 \cdot 2 \cdot 2 \cdot A + 2 \cdot 2 \cdot 2 \cdot A + 2 \cdot 2 \cdot A + A$$

Tétel (Algebrai formula az összegre)**.** *Különböző pontokra:*

$$P : (x_p, y_p), \quad Q : (X_Q, y_Q), \quad R : (x_R, y_R), \qquad R = P + Q$$

$$x_R = m^2 - x_p - x_Q, \quad y_R = m \cdot (x_p - x_R) - y_P, \quad m = \frac{y_P - y_Q}{x_P - x_Q}$$

Ugyan arra a pontra:

$$R = 2 \cdot P$$

$$x_R = m^2 - 2 \cdot x_p, \quad y_R = m \cdot (x_p - x_R) - y_P, \quad m = \frac{3 \cdot x_P^2 + a}{2 \cdot y_p}$$

7.2 Elliptikus görbék mod p

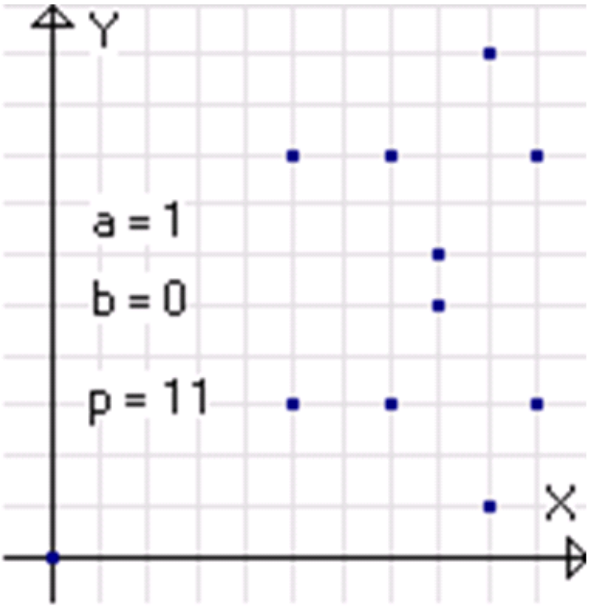
Mostantól $x, y, a, b \mod p$ egészek, ahol p prím. Ez is test, ugyanúgy lehet számolni a görbe pontjaival. Itt is ugyanaz a diszkrimináns, és itt sem lehet $0 \mod p$, azaz nem lehet p-vel osztható.

Példa.

$$p = 11, \quad a = 1, \quad b = 0$$

$$y^2 \equiv x^3 + x \mod 11$$

$x = y = 0$ *kivételével szimmetrikus:*



y	$y^2 \mod 11$
0	0
1	1
2	4
3	9
4	5
5	3
6	3
7	5
8	9
9	4
10	1

x	y^2	y
0	0	0
1	2	-
2	10	-
3	8	-
4	2	-
5	9	3,8
6	2	-
7	9	3,8
8	3	5,6
9	1	1,10
10	9	3,8

7.2.1 Összeadás művelet

Különböző pontokra:

Tétel (Algebrai formula az összegre)**.**

$$P : (x_p, y_p), \quad Q : (X_Q, y_Q), \quad R : (x_R, y_R), \qquad R = P + Q$$

$$x_R = m^2 - x_p - x_Q \mod p, \quad y_R = m \cdot (x_p - x_R) - y_P \mod p, \quad m = (y_P - y_Q) \cdot (x_P - x_Q)^{-1} \mod p$$

Ugyan arra a pontra:

$$R = 2 \cdot P$$

$$x_R = m^2 - 2 \cdot x_p \mod p, \quad y_R = m \cdot (x_p - x_R) - y_P \mod p, \quad m = (3 \cdot x_P^2 + a) \cdot (2 \cdot y_p)^{-1} \mod p$$

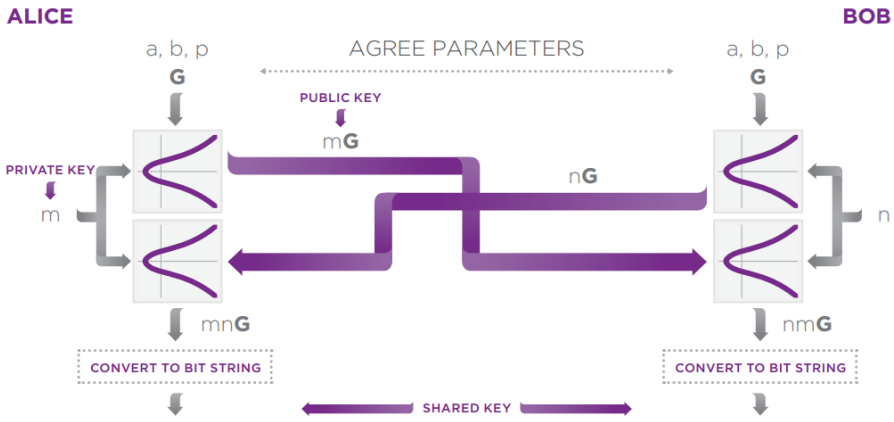
Definíció (Generátor pont)**.** *Azt a pontot, amelyet p-szer összeadva előállít minden pontot, generátor pontnak hívjuk.*

7.2.2 Diszkrét logaritmus

Definíció (Diszkrét logaritmun elliptikus görbék mod p felett)**.** *Diszkrét logaritmusnak nevezzük azt a műveletet, amellyel megállapítjuk, hogy hányszor kell összeadni egy elemet, hogy megkapjuk a keresett értéket.*

Az R alapú diszkrét logaritmusa Q pontnak: $Q = n \cdot R$.

Diffie–Hellman kulcsegyeztetés:



Aláírás: Alice alá akarja írni az üzenetét a privát kulcsával (d_A), és Bob ezt validálni akarja, Alice nyilvános kulcsával ($H_A = d_A G$). Az algoritmus lépései Alice oldaláról:

1. $k \in [1, n-1]$ egy random egész.
2. Kiszámolja a $P = kG$ pontot (G a generátorelem).
3. Kiszámolja az $r = x_P \mod n$ számot.
4. Ha $r = 0$, akkor új k-t választ és kezdi előről.
5. Kiszámolja az $s = k^{-1}(z + rd_A) \mod n$ számot, ahol k^{-1} k multiplikatív inverze modulo n és z az aláírandó üzenet hashe.
6. Ha $s = 0$ választ egy másik k-t és kezdi előről.

Az algoritmu lépései Bob oldaláról:

1. Kiszámolja az $u_1 = s^{-1}z \mod n$ számot.
2. Kiszámolja az $u_2 = s^{-1}r \mod n$ számot.
3. Kiszámolja a $P = u_1 G + u_2 H_A$ pontot.

Akkor valid az aláírás, ha $r = x_P \mod n$.

8 Bináris polinomok

Definíció (Bináris polinom)**.** *A*

$$f(x) = a_0 + a_1x + a_2x^2 + \cdots + a_nx^n$$

alakú polinomokat, ahol $a_i \in \{0, 1\}$*, bináris polinomnak nevezzük.*

Ezeket vektor formában is ábrázolhatjuk, az együtthatók eltárolásával.

Példa.

$$1 \implies \begin{bmatrix}1\end{bmatrix}$$
$$x \implies \begin{bmatrix}1 & 0\end{bmatrix}$$
$$x+1 \implies \begin{bmatrix}1 & 1\end{bmatrix}$$
$$x^2+x \implies \begin{bmatrix}1 & 1 & 0\end{bmatrix}$$

8.1 Műveletek

8.1.1 Összeadás

Az összeadás az együtthatónk összeadásával lehetséges.

Példa.

$$(x^2+x)+(x+1) = x^2+1 \implies \begin{bmatrix}1 & 1 & 0\end{bmatrix} + \begin{bmatrix}0 & 1 & 1\end{bmatrix} = \begin{bmatrix}1 & 0 & 1\end{bmatrix}$$

8.1.2 Szorzás

Szorzásnál összeadásokat fogunk elvégezni. Ahol egyes van, azt eltoljuk az adott egyes helyiértékére és hozzáadjuk a szummához.

Példa.

$$(x^2+x)(x+1) = x^3+x^2+x^2+x = x^3+x$$

8.2 Irreducibilis bináris polinom

Definíció (Irreducibilis bináris polinom)**.** *Akkor mondjuk egy bináris polinomra, hogy irreducibilis, ha csak önmagával és* 1*-el osztható.*

Az első ilyen polinom x. Hasonlóan $x+1$ is irreducibilis. Az egyetlen másodfokú irreducibilis polinom x^2+x+1. Pár szabály megofogalmazhazható irreducibilis polinomok keresésére. Például kell a végére 1 tag, hiszen különben x-szel osztható. Hasonlóan biztosan nem páros darab egyes van benne, mert akkor osztható lesz $x+1$-el. Ezekkel a szabályokkal már meg tudjuk találni a harmadfokú irreducibilis bináris polinomokat, hiszen így más első fokú polinomokkal nem osztható, és két másodfokú osztó már negyedfokú polinomokat adna.

Ezekből a harmadfokú irreducibilis bináris polinomok x^3+x^2+1 és x^3+x+1. A negyedfokú irreducibilis polinomokra is igazak ezek a szabályok, és mivel csak egy másodfokú irreducibilis polinomunk van, ezek közül még ki kell ejteni annak a négyzetét.

Tehát a negyedfokú irreducibilis bináris polinomok x^4+x^3+1, x^4+x+1 valamint $x^4+x^3+x^2+x+1$.

8.3 Bináris polinomok feletti véges test

Ha van egy $m(x)$ modulusunk, amely egy irreducibilis polinom, akkor minden nem 0 polinomnak van egyértelmű inverze.

Példa.

$$m(x) = x^3+x^2+1, \quad 1^{-1} = 1$$

Többi számra kibővített euklideszi algoritmust alkalmazhatunk. Pl. $x^{-1} = ?$

$$x^3+x^2+1 = (x^2+x)\cdot x + 1 \implies x^{-1} = x^2+x.$$

$$(x+1)^{-1} = ? \qquad x^3+x^2+1 = x^2(x+1)+1 \implies (x+1)^{-1} = x^2.$$

Példa.

$$m(x) = x^8 + x^4 + x^3 + x + 1 \Leftrightarrow \begin{bmatrix}1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1\end{bmatrix}$$

$$(x^7 + x + 1)^{-1} = ? \Leftrightarrow \begin{bmatrix}0 & 0 & 0 & 0 & 0 & 0 & 1 & 1\end{bmatrix}^{-1} = ?$$

$$x^8 + x^4 + x^3 + x + 1 = x(x^7 + x + 1) + (x^4 + x^3 + x^2 + 1)$$

$$x^7 + x + 1 = (x^3 + x^2 + 1)(x^4 + x^3 + x^2 + 1) + x$$

$$x^4 + x^3 + x^2 + 1 = (x^3 + x^2 + x)x + 1$$

	$x^8 + x^4 + x^3 + x + 1$	$x^7 + x + 1$
$x^8 + x^4 + x^3 + x + 1$	*1*	*0*
$x^7 + x + 1$	*0*	*1*
$x^4 + x^3 + x^2 + 1$	*1*	x
x	$x^3 + x^2 + 1$	$x^4 + x^3 + x + 1$
1	$1 - (x^3 + x^2 + x)(x^3 + x^2 + 1)$	x^7

$$1 \equiv x^7(x^7 + x + 1) \mod m(x)$$

$$(x^7 + x + 1)^{-1} = x^7 \mod m(x)$$

9 Blokk rejtjelezők

A nyílt szöveget t hosstúságú, A-beli elemekből álló láncokra bontja, és egyszerre egy blokkot rejtjelez. Sokféle algoritmus van és sokfél felhasználási mód.

Egyszerű helyettesítéses és felcseréléses rejttelezők és ezek kombinációi. Még ma is ezek a leghatékonyabb rejtjelezők.

9.1 Jellegzetes blokk rejtjelezők

9.1.1 Szorzat rejtjelező

Helyettesítés, áthelyezés, lineáris leképezés, aritmetikai művelet, moduláris szorzás egymás utáni végrehajtása

9.1.2 Iterált rejtjelező

Egy belső invertálható funkció ismétlése, minden lépésben a kulcsból generált alkulcs alkalmazásával.

9.1.3 Helyettesítéses-permutációs hálózat

Iterált szorzat rejtjelező egyszerű helyettesítésekből és permutációkból.

9.1.4 Feistel rejtjelezők

Iterált rejtjelező, mely azáltal lesz invertálható, hogy minden lépésben tetszőleges f-re:

$$L_i = R_{i-1}$$

$$R_i = L_{i-1} \oplus f(R_{i-1}, K_i)$$

9.1.5 DES

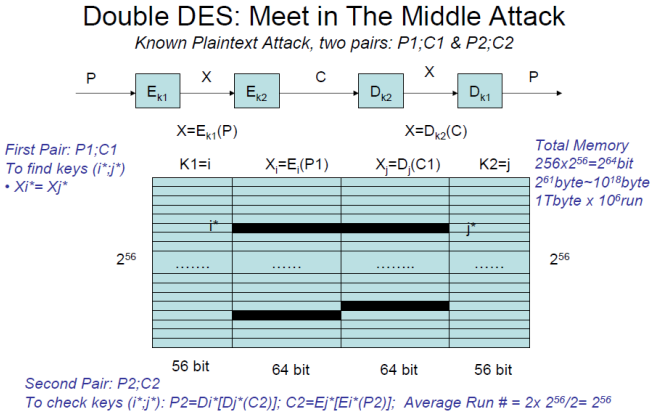
Olyan Feistel rejtjelező, amelynek a blokkmérete 64 bit, a kulcsmérete is 64 bit, amely igazából 56 bit és 8 paritás bit. Ez a rejtjelező 16 iteratív lépésből áll.

2^{56} kulcs lehetséges, amely a mai számítási kapacításokkal kimerítő kereséssel feltörhető, ezért ma már DES-t nem használunk.

Minden rejteltszöveg bit függ minden nyíltszöveg bittől. Egy bit megváltoztatása minden rejtjel bitet $\frac{1}{2}$ valószínőséggel változtat meg. Nem jósolható meg, hogy a rejtjelszöveg bit változtatása milyen hatással lesz a dekódolt szövegre.

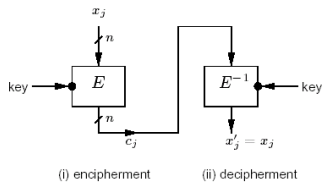
A DES nem csoporttulajdonságú: kompozícióra nem zárt, tehát két különböző kulccsal történő rejtjelezés egymás után nem váltható ki egy harmadik kulccsal történő rejtjelezésre. Emiatt van lehetőség többszörös rejtjelezésre:

- DES: nem biztonságos ($2^5$6 művelet kell a feltöréshez)
- double DES: nincs értelme (meet-in-the-middle támadás, 2^{57} művelet)
- tripple DES: kettő vagy három kulccsal (itt is van MITM támadás csak sokkal számításigényesebb)



9.2 Működési módok

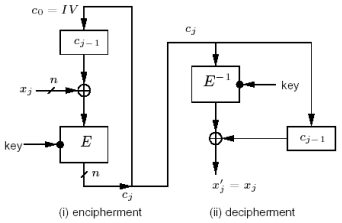
9.2.1 ECB – Electronic codebook



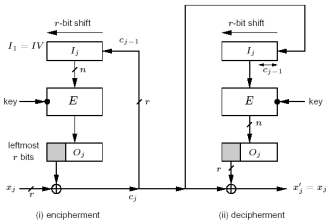
Azonos nyílt szöveg és kulcs esetén azonos rejtjelezett szöveget kapunk. A blokkok függetlenek egymástól és hiba esetén csak egy blokk hibás. Nem biztonságos, korlátozásokkal alkalmazható. A biztonságban véletlen "padding" segíthet.

9.2.2 CBC – Cipher–block chaining

Azonos nyílt szöveg és kulcs eltérő rejtjelezett szöveget eredményez, mert véletlen IV-vel indul az algoritmus. Dekódoláshoz kell az előző rejtjelszöveg. Hiba esetén csak két dekódolt blokk hibás. IV lehet nyílt, de integritása szükséges.



9.2.3 CFB – Cipher Feedback

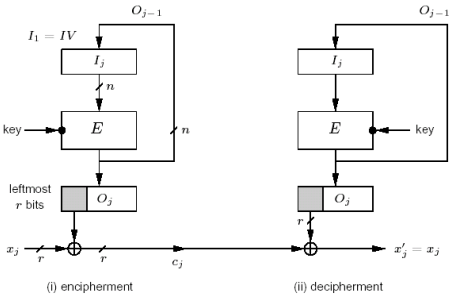


Azonos nyílt szöveg és kulcs esetén eltérő rejtjelezett szöveget eredményez, mert véletlen IV-vel indul az algoritmus. Dekódoláshoz kell az előző néhány rejtjelszöveg. Hiba esetén csak néhány dekódolt blokk hibás. Alacsonyabb hatásfok $r < n$ miatt. IV lehet nyílt.

Önszinkronizáló adatfolyam rejtjelezőnek használható.

9.2.4 OFB – Output Feedback

Azonos nyílt szöveg és kulcs esetén eltérő rejtjelezett szöveget eredményez, mert véletlen IV-vel indul az algoritmus. A dekódoló kulcsfolyam független a rejtjelszövegtől. Bithiba esetén csak egy dekódolt blokk hubás, de egy rejtjelszöveg bit kiesése esetén kiesik a szinkronból. IV lehet nyílt, de változtatni kell a kulcs újboli használatakor. Visszacsatolás helyett lehet számláló is (CTR mode).



Ez használható szinkron adatfolyam rejtjelező kulcsgenerátorának. CFB és OFB módok esetén csak rejtjelezés számítása szükséged, erre érdeme optimalizálni (inverzet nem kell számolni).

9.3 AES rejtjelezők

Minimum feltételek:

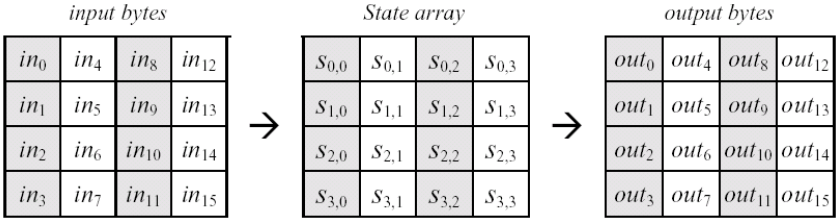
- Nyilvános,
- Licensz mentes,
- Szimmetrikus kulcsú,
- Blokk kódoló,
- Blokk méret: 128 bit,
- Kulcs méret: 128/192/256 bit.

Szempontok:

- SW/HW hatékonyság,
- Memóriaszükséglet,

- Rugalmasság,
- Egyszerűség.

128 bites blokk: 16 byte, 4×4-es tömbbe rendezve:



A byteokat bináris polinomként kezeli:

$$b_7x^7 + b_6x^6 + b_5x^5 + b_4x^4 + b_3x^3 + b_2x^2 + b_1x + b_0.$$

Összeadás és szorzás $GF(2^8)$-ban. AES-ben használt modulus: $m(x) = x^8 + x^4 + x^3 + x + 1$ irreducibilis polinom. 4 byteból, azaz egy oszlopból wor képezhető:

$$a(x) = a_3x^3 + a_2x^2 + a_1x + a_0.$$

Itt az együtthatók hetedfokú bináris polinomok. Erre a struktúrára az összeadás:

$$a(x) + b(x) = (a_3 \oplus b_3)x^3 + (a_2 \oplus b_2)x^2 + (a_1 \oplus b_1)x + (a_0 \oplus b_0)$$

Szorzás modulo $(x^4 + 1)$: $d(x) = a(x) \otimes b(x)$.

$$d(x) = d_3x^3 + d_2x^2 + d_1x + d_0 \qquad \begin{bmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \end{bmatrix} = \begin{bmatrix} a_0 & a_3 & a_2 & a_1 \\ a_1 & a_0 & a_3 & a_2 \\ a_2 & a_1 & a_0 & a_3 \\ a_3 & a_2 & a_1 & a_0 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

Az algoritmus konkrét sorrendje a következő. Minden iterációban helyettesítéssel kezdünk. Ehhez ki kell számítani a byteok inverzét $GF(2^8)$-ben, majd egy leképezés. Ehhez az egész művelethez van egy LUT.

A következő lépés a sor eltolás. Az első sor nem tolódik, a második 1-el, a harmadik 2-vel, a negyedik sor pedig 3 byteal tolódik balra.

A következő lépés az oszlop keverés. Egy oszlopban lévő négy bináris polinomból alkotott harmadfokú polinom kerül megszorzásra a $\{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\}$ polinommal modulo $x^4 + 1$. Ennek az inverz polinomja $\{0b\}x^3 + \{0d\}x^2 + \{09\}x + \{0e\}$.

Az iteráció a kulcs hozzáadásával zárul. Itt minden oszlophoz hozzáadunk egy előre legenerált kulcs vektort.