Mobiltelefon titkosítási algoritmusok

Szakdolgozat

Ota Ayaka

Témavezető: Szabó István
Valószínűségelméleti és Statisztika Tanszék

Eötvös Loránd Tudományegyetem
Természettudományi Kar

2013

Tartalomjegyzék

0.1. Bevezetés

Ennek a szakdolgozatnak a célja a GSM hálózat mobiltelefon titkosítási algoritmusok ismertetése és az elemzése többféle támadás bemutatásával.

A szakdolgozat úgy épül fel, hogy először bevezetjük a shift registereket, a GSM algoritmusok alapját. Ezután a GSM algoritmusok A5/1 és A5/2 leírásairól és a támadási módszerekről lesz szó.

A feedback shift register egy egyszerű módszert nyújt véletlenszerű sorozat generálására, ennek viszont vannak hátrányai. Az A5/1 és A5/2 algoritmusok clockolási mechanizmust alkalmaz, hogy véletlenszerűbb legyen a kimenet, de mégis vannak struktúrális gyengeségek, amik egy hatékonyabb támadási felépítést tesznek lehetővé. A sokféle támadás közül először Goldberg, Wagner és Green támadását([3]), majd Maximov, Johansson és Babbage támadását([5]) és végül Barkan és Biham támadását([6]) vázoljuk.

0.2. Köszönetnyilvánítás

Szeretném megköszönni a témavezetőmnek, Szabó Istvánnak hogy megismertette velem ezt a témakört és irodalom javaslatokkal segített a szakdolgozatom megírásában.

1. fejezet

Shift Registerek: A GSM titkosítási algoritmusok matematikai alapjai

Ebben a fejezetben a shift registereket vezetjük be. A shift register elméletet Solomon W. Golomb alapozta meg és ennek a fejezetnek az anyaga is az ő könyvéből([1]) lett összeállítva.

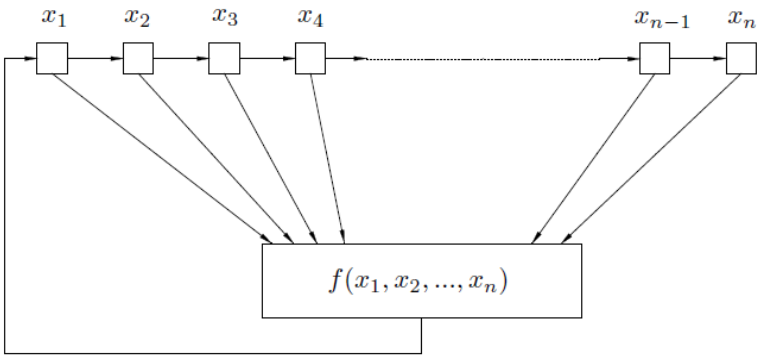
1.1. Shift Registerek bevezetés

Az 1.1. ábra egy általános shift registeret ábrázol feedbackkal. Az $x_1, x_2, ..., x_n$-nel címzett négyzetek bináris(1 vagy 0) tároló elemek. Egy register alapműveletét clockolásnak("clocking") hívjuk, amialatt egy mester óra által meghatározott periódikus időközönként x_i tartalma x_{i+1}-be kerül(clockolással shiftelnek a register tartalmai). Az x_1 új értékét úgy adhatjuk meg, hogy kiszámoljuk valamilyen $f(x_1, x_2, ..., x_n)$ függvényből a register minden jelen pillanatbeli értékeivel. Ezt feedback-nak hívják. A tároló elemek helyeit, amik a következő állapotot befolyásolják, "tap"-oknak hívjuk. Az n bináris tároló elemeket a shift register lépcsőjének("stage") hívjuk, és a tartalmukat(vagy bináris szám vagy n-hosszú bináris vektor) a shift register (belső) állapotának("state") hívjuk. Minden clockolással egy állapot a következő állapotba megy át.

Ha az $f(x_1, x_2, ..., x_n)$ feedback függvényt a következő alakban írhatjuk:

$$f(x_1, x_2, ..., x_n) = c_1x_1 \oplus c_2x_2 \oplus c_3x_3 \oplus ... \oplus c_nx_n,$$

(ahol c_i 0 vagy 1, és $\oplus$ modulo 2 additivitást jelöli) akkor a shift registert lineárisnak hívjuk.



1.1. ábra. Feedback shift register általános ábrája

Egy adott shift register állapotának csak két lehetséges utódja van, mivel csak egy bites bizonytalanság van, ami a feedback függvény által lesz meghatározva. Hasonlóan, egy adott állapotnak csak két elődje van. Egy konkrét feedback függvény kiválasztása egyértelműen határozza meg az utódot, de az elől nem feltételenül egyértelmű.

1.2. Véletlen tulajdonságú sorozatok és lineáris shift registerek

Elektonikában felmerülő problémákban, számítógépeknél, kriptográfiában vagy számos más területen véletlen sorozatra van szükség. Sok esetben hasznos, hogy ha van egyszerű módszer olyan sorozat generálásra, ami véletlennek tűnik, és csak közelebbről vizsgálva derül ki a szabályossága. Előírunk bizonyos tulajdonságokat, ami a véletlenhez kapcsolódik. Olyan sorozatot, ami teljesíti ezeket a tudajdonságot véletlen sorozatnak hívunk.

Először definiáljuk az auto-korreláció függvényt.

1. Definíció. *Legyen* $\{a_n\} = \{a_0, a_1, a_2, ...\}$ *valós sorozat. Ekkor az auto-korreláció függvény* $C(\tau)$*:*

$$C(\tau) = \lim_{N\to\infty} \frac{1}{N} \sum_{n=1}^{N} a_n a_{n+\tau},$$

ha ennek a határértéke létezik.

Speciálisan, ha $\{a_n\}$ periodikus sorozat p periódussal, akkor $C(\tau)$ véges összegre redukálódik:

$$C(\tau) = \frac{1}{p} \sum_{n=1}^{p} a_n a_{n+\tau}$$

Itt τ $\{a_n\}$ fázis shiftjének tekinthető. $C(\tau)$ a sorozat és a fázis shift közötti hasonlóságot méri. Ez a mennyiség mindig $\tau = 0$-ra adja a legmagasabbat, és ha $\{a_n\}$ véletlen sorozat, akkor $C(\tau)$ kis értéket vesz fel minden más τ-ra.

A következőkben periódikus bináris sorozatokat nézünk, amikre teljesül néhány, vagy az összes a következő véletlenszerű tulajdonságok közül:

R-1. Minden periódusban a +1-ek száma majdnem megegyezik a -1-ek számával, $|\sum_{n=1}^{p} a_n| \leq 1$.

R-2. Minden periódusban az ugyanazokból a számokból álló szakaszoknak("run") a fele 1 hosszú, az egy-negyede 2 hosszú, az egy-nyolcada 3 hosszú, stb, amíg az ilyen run-ok száma 1-nél nagyobb. Ráadásul minden hosszúságra a +1-es és a -1-es run-ok száma megegyezik. (Az összes +1-es a run-ok száma megegyezik az összes -1-es run-ok számával, mivel a két típusú run váltakozik.)

R-3. Az auto-korreláció függvény $C(\tau)$ két értékű. Pontosabban:

$$pC(\tau) = \sum_{n=1}^{p} a_n a_{n+\tau} = \begin{cases} p & \text{ha } \tau = 0 \\ K & \text{ha } 0 < \tau < p \end{cases}$$

2. Definíció. *Pszeudo-zaj sorozatnak hívjuk azt a sorozatot, ami R-1, R-2 és R-3 tulajdonságokat teljesíti.*

Most vegyük a bináris lineáris shift registert feedback-kal.(Később shift register alatt shift registert értjük feedback-kal.)

1. Tétel. *Az r-lépcsős shift register állapotainak sorozata periódikus, és a periódusa $p \leq 2^r - 1$*

Bizonyítás A shift register egyes állapotát az előző állapotok határozzák meg. Tehát ha egy állapot ugyanaz mint egy korábbi állapot, akkor a rákövetkező állapotok is ugyanazok, tehát periódikusak. Az r-lépcsős shift registernek összesen 2^r lehetséges állapota van. Így az ismétlődés az első $2^r + 1$ állapotok között történik, és így a periódus: $p \leq 2^r$. Végül, ha a "csupa 0" állapot fordul elő, akkor a rákövetkező állapotok is "csupa 0"-k

lesznek, és a periódus $p = 1$. Így egy rendes hosszúságú periódusba nem számoljuk a "csupa 0" állapotot, és így $p \leq 2^r - 1$. $\square$

Megjegyezzük, hogy a Tétel (1) igaz bármilyen kezdeti állapotú shift registerre.

A most bizonyított tétel éles, azaz minden r-re létezik olyan feedback, hogy $2^r - 1$ hosszú periódus keletkezik.

Érdemes megvizsgálni a shift register egyik elemének, mondjuk az első elemének az állapot változását. Tegyük fel, hogy ezen elem eddigi állapotainak a sorozata: $a_0, a_1, a_2, ..., a_n$. A feedback miatt a_n néhány elemnek a (modulo 2) összege az (n-1)-edik állapotban. Tulajdonképpen az a_n mod 2 összegében minden elem az első elem korábbi állapotaiból kifejezhető. Azaz:

1. Állítás. *a_n teljesíti:*

$$a_n = c_1 a_{n-1} + c_2 a_{n-2} + ... + c_r a_{n-r}$$

ahol $c_1, c_2, ..., c_r$ együtthatók vagy 0-k vagy 1-ek és függetlenek n-től. Az összeadás modulo 2 értendő. Az ilyen összefüggést lineáris rekurziónak hívjuk, és minden $\{a_n\}$-t, ami teljesíti ezt az összefüggést lineáris rekurziós sorozatnak hívjuk.

Megjegyezzük, hogy általában a következők teljesülnek:

1. A második elemnek az állapot változása ugyanaz mint az első elemé, csak 1 állapotnyi különbséggel.

2. Tehát minden elem teljesíti ugyanazt a lineáris rekurziót. Így úgy tekinthető, hogy az egész shift register teljesíti a rekurziót.

3. Definíció. *Ha adott $\{a_n\} = \{a_0, a_1, a_2, ...\}$ a shift register sorozat első elemének állapot változása, akkor definiáljuk a generátor függvénnyét úgy, hogy:*

$$G(x) = \sum_{n=0}^{\infty} a_n x^n$$

A shift register kezdeti állapota úgy is tekinthető, hogy $a_{-1}, a_{-2}, ..., a_{-r}$. Ha az $\{a_n\}$ teljesíti a

$$a_n = \sum_{i=1}^{r} c_i a_{n-i}$$

rekurziót, akkor

$$\begin{aligned} G(x) &= \sum_{n=0}^{\infty} \sum_{i=1}^{r} c_i a_{n-i} x^n = \sum_{i=1}^{r} c_i x^i \sum_{n=0}^{\infty} a_{n-i} x^{n-i} \\ &= \sum_{i=1}^{r} c_i x^i [a_{-i} x^{-i} + ... + a_{-1} x^{-1} + \sum_{n=0}^{\infty} a_n x^n] \end{aligned}$$

Így

$$G(x) = \sum_{i=1}^{r} c_i x^i [a_{-i} x^{-i} + ... + a_{-1} x^{-1} + G(x)]$$

és

$$G(x) - \sum_{i=1}^{r} c_i x^i G(x) = \sum_{i=1}^{r} c_i x^i (a_{-i} x^{-i} + ... + a_{-1} x^{-1}).$$

Más szóval

$$G(x) = \frac{\sum_{i=1}^{r} c_i x^i (a_{-i} x^{-i} + ... + a_{-1} x^{-1})}{1 - \sum_{i=1}^{r} c_i x^i}. \tag{1.1}$$

Ez a kifejezés $G(x)$-et kizárólag az $a_{-1}, a_{-2}, ..., a_{-r}$ kezdeti feltételekkel és a $c_1, c_2, ..., c_r$ feedback együtthatókkal fejezi ki. Valójában a (1.1) kifejezésben a nevező a kezdeti feltételektől is független. $a_{-1} = a_{-2} = ... = a_{1-r} = 0, a_{-r} = 1$ kezdeti feltételekkel a $G(x)$ a következőre redukálódik:

$$G(x) = \frac{c_r}{1 - \sum_{i=1}^{r} c_i x^i} \tag{1.2}$$

4. Definíció. *A következő r-ed fokú polinomot az a_n sorozat és a shift register karakterisztikus polinomjának fogjuk hívni:*

$$f(x) = 1 - \sum_{i=1}^{r} c_i x^i \tag{1.3}$$

Megjegyezzük, hogy itt nem használtuk ki, hogy modulo 2 az összeadás; tehát általános összeadásra is működnek a $G(x)$ fenti képletei. Tétel (1) azt mutatja, hogy a shift register sorozatok periódikusak, és felső korlátot ad a periódusra. A (1.2) egyenlet lehetővé teszi a még pontosabb leírást.

2. Tétel. *Ha egy r-lépcsős shift register sorozat: $A = \{a_n\}$ a $a_{-1} = a_{-2} = ... = a_{1-r} = 0, a_{-r} = 1$ kezdeti feltételeket teljesíti, akkor az A periódusa az a legkisebb pozitív egész szám p, amire az $1 - x^p$ osztja a karakterisztikus polinomot $f(x)$-et modulo 2.*

Bizonyítás Az adott kezdeti feltételek mellett

$$G(x) = \frac{1}{f(x)} = \sum_{n=0}^{\infty} a_n x^n.$$

Ha A-nak a periódusa p, akkor

$$\begin{aligned}
\frac{1}{f(x)} &= (a_0 + a_1 x + ... + a_{p-1} x^{p-1}) + x^p (a_0 + a_1 x + ... + a_{p-1} x^{p-1}) \\
&+ x^{2p} (a_0 + a_1 x + ... + a_{p-1} x^{p-1}) + ... \\
&= (a_0 + a_1 x + ... + a_{p-1} x^{p-1})(1 + x^p + x^{2p} + x^{3p} + ...) \\
&= (a_0 + a_1 x + ... + a_{p-1} x^{p-1})/(1 - x^p).
\end{aligned}$$

Így

$$f(x)[a_0 + a_1x + ... + a_{p-1}x^{p-1}] = 1 - x^p$$

és $f(x)$ osztja $1 - x^p$-t.
Fordítva, ha $f(x)$ osztja $1-x^p$-t, legyen a hányados a maradékos osztás után: $\alpha_0 + \alpha_1x + ... + \alpha_{p-1}x^{p-1}$, ekkor

$$\begin{aligned}
\frac{1}{f(x)} &= \frac{\alpha_0 + \alpha_1x + ... + \alpha_{p-1}x^{p-1}}{1 - x^p} \\
&= (\alpha_0 + \alpha_1x + ... + \alpha_{p-1}x^{p-1})(1 + x^p + x^{2p} + x^{3p} + ...) \\
&= (\alpha_0 + \alpha_1x + ... + \alpha_{p-1}x^{p-1}) + x^p(\alpha_0 + \alpha_1x + ... + \alpha_{p-1}x^{p-1}) + ... \\
&= G(x) = \sum_{n=0}^{\infty} a_nx^n.
\end{aligned}$$

és az egyenletet x szerint rendezve és az együtthatókat összehasonlítva kiderül, hogy $\{a_n\} = \{\alpha_n\}$, így A-nak a periódusa p vagy p-nek egy osztója.
Így az A periódusa az a legkisebb pozitív egész szám p, amire az $1-x^p$ osztja az $f(x)$-et. □

1. Következmény. *A (1.1) egyenlet szerint $G(x) = g(x)/f(x)$, ahol a $g(x)$ foka kisebb, mint az $f(x)$ foka. Ha a shift registernek r lépcsője van, akkor $f(x)$ foka r.*

5. Definíció. *Ha $g(x)$-nek nincs közös osztója $f(x)$-szel, akkor is fennáll a tétel (2) , és a sorozat periódusa az a legkisebb p, amire $1-x^p$ osztja $f(x)$-et. Az ilyen p-t az $f(x)$ kitevőjének hívjuk.*

2. Következmény. *Amikor $g(x) = 1$, ez a tétel (2) maga.*
Másik fontos eset, amikor $f(x)$ irreducibilis. Ebben az esetben $f(x)$-nek nincs közös osztója az alacsonyabb fokú $g(x)$-szel, kivéve amikor $g(x) = 0$, ami a "mindegyik 0" kezdeti feltételnek felel meg. Így amikor $f(x)$ irreducibilis, a shift register sorozat periódusa nem függ a kezdeti feltételektől, a "mindegyik 0" kezdeti feltétel kivételével.

6. Definíció. *Az r-lépcsős shift register által generált sorozatot maximális hosszúságúnak nevezzük, ha a periódusa: $p = 2^r - 1$*

3. Tétel. *Ha az A sorozat maximális hosszúságú, a karakterisztikus polinomja irreducibilis.*

Bizonyítás Mivel az A $2^r - 1$ tagon fut végig amíg visszatér, minden r hosszúságú 0-1 sorozat megtalálható A-ban(Kivéve r hosszú csupa 0). Speciálisan van olyan sorozat, hogy 1 után $r - 1$ darab 0 jön. Ettől a ponttól

tekintve a tétel (2) kezdeti feltétele teljesül. Így az A periódusa az $f(x)$-nek a kitevője.
Tegyük fel, hogy $f(x)$ $f(x) = s(x) \cdot t(x)$ alakban írható. Ekkor

$$\frac{1}{f(x)} = \frac{\alpha(x)}{s(x)} + \frac{\beta(x)}{t(x)}$$

parciális törtekre bontva. Tegyük fel, hogy $s(x)$ és $t(x)$ fokai: $r_1 > 0$ illetve $r_2 > 0$, ahol $r_1 + r_2 = r$ Ekkor $\alpha(x)/s(x)$ egy hatványsor, aminek az együtthatói legfeljebb $2^{r_1} - 1$ periódussal ismétlődnek, és $\frac{\beta(x)}{t(x)}$ egy hatványsor, aminek az együtthatói legfeljebb $2^{r_2} - 1$ periódussal isméltődnek.
Ekkor az $1/f(x) = \alpha(x)/s(x) + \beta(x)/t(x)$ egy hatványsor, aminek az együtthatóinak a periódusa legfeljebbb a legkisebb közös többszöröse az egyes periódusoknak. Mivel ez nem haladhatja meg a periódusok szorzatát, így

$$2^r - 1 \leq (2^{r_1} - 1)(2^{r_2} - 1) = 2^{r_1+r_2} - 2^{r_1} - 2^{r_2} + 1 \leq 2^r - 2 - 2 + 1 = 2^r - 3$$

Ez ellentmondás, tehát nem igaz a feltevésünk, hogy "$f(x) = s(x) \cdot t(x)$ alakban írható". □

Ebben a bizonyításban az $f(x)$ parciális törtekre bontása azt feltételezi, hogy az $s(x)$ és a $t(x)$ különböző osztók. Megjegyezzük, hogy ha $f(x) = s^2(x)$, akkor az $f(x)$ periódusa az $s(x)$ periódusának a kétszerese, így legfeljebb $2(2^{r/2} - 1) < 2^r - 1$. Ilyen módon a többszörös osztók is kezelhetők.
A tétel (3) fordítottja nem igaz. Léteznek irreducibilis polinomok, amik nem felelnek meg maximális hosszúságú sorozatoknak.

Az r lépcsős shift register úgy tekinthető, mint egy r dimenziós vektor. Ekkor a shift register egy lineáris operátor, ami az egyik állapotot a másikba viszi.
Általánosan a shift register mátrixos reprezentálása a következő alakú:

$$M = \begin{pmatrix} c_1 & 1 & 0 & \cdots & 0 \\ c_2 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_{r-1} & 0 & 0 & \cdots & 1 \\ c_r & 0 & 0 & \cdots & 0 \end{pmatrix} \tag{1.4}$$

ahol a főátló felett csupa 1 áll, és az első oszlopban a feedback együtthatók vannak.

Ez a mátrix valóban a shift register clockolást fejezi ki:

$$(a_{n-1}, a_{n-2}, \ldots, a_{n-r}) \begin{pmatrix} c_1 & 1 & 0 & \cdots & 0 \\ c_2 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_{r-1} & 0 & 0 & \cdots & 1 \\ c_r & 0 & 0 & \cdots & 0 \end{pmatrix}$$

$$\begin{aligned} &= (c_1 a_{n-1} + c_2 a_{n-2} + \cdots + c_r a_{n-r}, a_{n-1}, a_{n-2}, \ldots, a_{n-r+1}) \\ &= (a_n, a_{n-1}, \ldots, a_{n-r+1}) \end{aligned}$$

Az M karakterisztikus egyenletét a következőképpen definiáljuk:

$$\begin{aligned} f(\lambda) &= det|M - \lambda I| = \begin{vmatrix} c_1 - \lambda & 1 & 0 & \cdots & 0 \\ c_2 & -\lambda & 1 & \cdots & 0 \\ c_3 & 0 & -\lambda & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_r & 0 & 0 & \cdots & -\lambda \end{vmatrix} \\ &= (-\lambda)^{r-1}(c_1 - \lambda) - c_2(-\lambda)^{r-2} + c_3(-\lambda)^{r-3} \cdots (-1)^r c_r \\ &= -(-\lambda)^r [1 - \frac{c_1}{\lambda} - \frac{c_2}{\lambda^2} - \frac{c_3}{\lambda^3} - \cdots - \frac{c_r}{\lambda^r}] \\ &= \frac{(-1)^{r+1}}{x^r}[1 - (c_1 x + c_2 x^2 + \cdots + c_r x^r)] \end{aligned}$$

ahol $x = 1/\lambda$-val lett helyettesítve.

Tehát $\frac{(-1)^{r+1}}{x^r}$ szorzótól eltekintve az M karakterisztikus egyenlete megegyezik a shift register karakterisztikus polinomjával.

Minden r-ed fokú, modulo 2 irreducibilis polinom osztja az $1 - x^{2^r-1}$ polinomot. Ebből és a tétel (2) -ből könnyen levezethető, hogy:

4. Tétel. *Ha a sorozatnak a karakterisztikus polinomja r-ed fokú és irreducibilis, akkor a sorozat periódusa $(2^r - 1)$-nek az osztója.*

3. Következmény. *Ha $2^r - 1$ prím, akkor minden r-ed fokú irreducibilis polinom egy maximális hosszúságú shift register sorozatnak felel meg.*

Valóban, ebben az esetben $(2^r - 1)$-nek az egyetlen osztója $2^r - 1$ önmaga.

Az r-ed fokú modulo 2 irreducibilis polinomok számára, sőt még a "maximális kitevőjű" r-ed fokú polinomok számára is van explicit képlet. Ezek a

képletek két számelméleti függvényt használnak.
A prím felbontás egyértelműsége szerint minden egész szám felbontható a következő alakra:

$$n = \prod_{i=1}^{k} {p_i}^{\alpha_i}$$

Ekkor az Euler φ -függvény:

$$\phi(n) = \begin{cases} 1 & \text{ha } n = 1 \\ \prod_{i=1}^{k} {p_i}^{\alpha_i - 1}(p_i - 1) & \text{ha } n > 1 \end{cases}$$

és a Möbius μ -függvény:

$$\mu(n) = \begin{cases} 1 & \text{ha } n = 1 \\ 0 & \text{ha } \prod_{i=1}^{k} \alpha_i > 1 \\ (-1)^k & \text{különben, azaz ha az } n \text{ } k \text{ darab különböző prím szorzata} \end{cases}$$

Ekkor modulo 2-ben az r-ed fokú irreducibilis polinomok száma:

$$\Psi_2(r) = \frac{1}{r} \sum_{d|r} 2^d \mu(\frac{r}{d}) \tag{1.5}$$

a maximális kitevőjű r-ed fokú modulo 2 polinomok száma:

$$\lambda_2(r) = \frac{\phi(2^r - 1)}{r} \tag{1.6}$$

5. Tétel. *Legyen az f $(2^r - 1)$-nek az osztója. Ha az f nem osztója $2^s - 1$-nek semmilyen $s < r$-re, akkor az f előáll, mint egy r-ed fokú irreducibilis polinom kitevője. Tehát $\phi(f)/r$ darab r-ed fokú irreducibilis polinom van, aminek a kitevője f.*

6. Tétel. *Ha $f(x) = s(x)t(x)$, ahol $s(x)$-nek nincs közös osztója $f(x)$-szel, akkor az $f(x)$ kitevője az $s(x)$ és $t(x)$ kitevőinek a legkisebb közös többszöröse.*

7. Tétel. *Legyen az $f(x)$ periódusa p, és a $g(x)$ periódusa q. Ha $f(x)$ irreducibilis és $g(x) = [f(x)]^n$, akkor a $g(x)$ periódusa $e(n)$-szerese lesz az $f(x)$ periódusának.*
Azaz $q = e(n)p$, ahol $e(n)$ úgy adható meg, hogy: $e(1) = 1, e(2) = 2, e(3) = 4, e(4) = 4, e(5) = 8, e(6) = 8, e(7) = 8, e(8) = 8$, stb.

Tehát ha van egy polinom, ami egy shift registernek megfelelthető, akkor a periódusát(azaz kitevőjét) úgy határozhatjuk meg, hogy faktorizáljuk

a polinomot irreducibilis polinomok hatványai szerint. Az egyes osztók periódusai a tétel (7) szerint számolhatók, és a szorzatuk periódusa a tétel (6) -ból számolható ki.

Tegyük fel, hogy egy r-lépcsős shift register befutja mind a $2^r - 1$ darab lehetséges állapotot, amíg ismétlődik. Ezek az állapotok 1-től 2^r-1-ig terjedő számok a kettes számrendszerben. A megfelelő maximális hosszúságú shift register sorozatot($\{a_n\}$) úgy tekinthetjük, mint a kettes számrendszerben a számjegyek. Az 1-et páratlan számokra és az 0-t páros számra megfeleltetve 1-től $(2^r - 1)$-ig 2^{r-1} darab páratlan szám van, és $2^{r-1} - 1$ darab páros szám. Tehát bármilyen maximális hosszúságú shift register sorozatban 2^{r-1} darab 1 és $2^{r-1} - 1$ darab 0 van. Azaz:

8. Tétel. *A véletlen tulajdonság R-1 minden maximális hosszúságú shift register sorozatra teljesül. Itt +1-eket és -1-eket 0-nak illetve 1-nek feleltetjük meg.*

A maximális hosszúságú shift register sorozatban: $\{a_n\}$-ben adott r esetén tetszőleges r darab egymást követő számjegy(csupa 0 kivételével) pontosan egyszer fordul elő. Speciálisan, r darab egymást követő 1-es pontosan egyszer fordul elő. Az 1-esek ezen run-ját 0 kell, hogy megelőzze és 0 kell, hogy kövesse, különben más r darab egymást követő 1-esek run-ja is létezne. Az 0-t követő $r - 1$ darab 1-es pontosan egyszer fordul elő. De ezt már számoltuk r hosszú 1-esek run-jaként, amit megelőz egy 0. Hasonlóan $r - 1$ darab 1-est követő 0 is pontosan egyszer fordul elő, és ezt is már megszámoltuk, mert r hosszú 1-esek run-ját 0 kell, hogy kövesse. Így nem létezik $r - 1$ darab 1-esek run-ja.
Tegyük fel, hogy $0 < k < r - 1$. A k hosszúságú 1-esek run-jainak a számát szeretnénk kiszámolni. Tekintsük az r egymást követő számjegyet, ami 0-val kezdődik, k darab 1-essel folytatódik, megint 0 követi, és a maradék $r-k-2$ jegyet tetszőlegesen töltjük fel. Ilyen sorozat megadása 2^{r-k-2}-féleképpen történhet, és ezért ennyi a k hosszúságú 1-esek run-jainak a száma.
Hasonló érvelés igaz a $0 < k < r - 1$ hosszúságú 0-sok run-jainak a számára. Nem létezik r darab egymást követő 0(mivel ez "megállítaná" a shift regis-tert), de az "1-est követő $r - 1$ darab 0"-nak elő kell fordulnia, ezért $r - 1$ hosszú 0-sok run-ja létezik.

Így a maximális hosszúságú shift register sorozat szerkezetét teljesen meghatároztuk, ami az 1-esek run-ját("block") és a 0-sok run-ját("gap") illeti, azaz: ha $0 < k < r - 1$, akkor 2^{r-k-2} darab k-hosszú block és 2^{r-k-2} darab k-hosszú gap létezik. Méghozzá 1 darab $r - 1$-hosszú gap van, és 1 darab r-hosszú block van. A sorozat periódussal($p = 2^r - 1$) kifejezve $(p+1)/2$ darab

run van, a fele block, és a fele gap. A blockok közül a felének a hosszúsága 1, az egy-negyedének a hosszúsága 2, az egy-nyolcadának a hosszúsága 3, stb. és a gapoknál is hasonlóan. Ez addig folytatódik, amíg van egy $r-2$ hosszú block és egy $r-2$ hosszúságú gap. Ezután egy $r-1$ hosszú gap van, és r hosszú block van.
Ezeket az eredményeket a következő formában foglalhatjuk össze:

9. Tétel. *A véletlen tulajdonság R-2 igaz minden maximális hosszúságú shift register sorozatra.*

Legyen $A_1 = \{a_1, a_2, a_3, ...\}$ maximális hosszúságú shift register sorozat $p = 2^r - 1$ periódussal. Legyen $A_2 = \{a_2, a_3, ...\}$,..., $A_p = \{a_p, a_{p+1}, ...\}$. Továbbá legyen $A_0 = \{0, 0, 0, ...\}$.

10. Tétel. *Az $A_0, A_1, ..., A_p$ sorozatok Abel-csoportot alkotnak tagonkénti modulo 2 összeadásra nézve. (A "tagonkénti modulo 2 összeadás" azt jelenti, hogy ha $B = \{b_1, b_2, b_3, ...\}$ és $C = \{c_1, c_2, c_3, ...\}$ akkor $B + C = \{b_1 + c_1, b_2 + c_2, b_3 + c_3, ...\}$).*

Bizonyítás Bármely i-re $A_i + A_0 = A_i$ és $A_i + A_i = A_0$. Így A_0 a csoport eleme, és A_i önmagának az ellentetje.
Legyen R a lineáris rekurzió, amit az A_i teljesít. Ekkor az R-et $A_2, ..., A_p$ is teljesítik, és A_0 is triviálisan teljesíti. Mivel az R lineáris, $A_i + A_j$ is teljesíti az R-et, amikor A_i és A_j is teljesíti. Így $A_i + A_j$-t is az R definiálja az első r darab taggal. Bármilyen legyen ez az r darab tag, a 2^r darab sorozat: $A_0, A_1, ..., A_p$ közül pontosan az egyik első r darab tagjával fog megegyezni. Így a két sorozat összege is benne van a csoportban.
A kommutativitás és az asszociativitás igaz a sorozatokra, mivel igaz a szokásos modulo 2 összeadásra. □

Legyen $\{b_n\}$ az $\{a_n\}$-ből származó sorozat úgy, hogy: $b_n = 1 - 2a_n$. Azaz 0 helyébe 1, és 1 helyébe -1 kerül. Ez úgy is írható, hogy: $b_n = e^{i\pi a_n}$.
Legyen $A_0, A_1, ..., A_p$ a fenti módon definiált. 0-t 1-re és 1-et -1-re cseréljük, így kapjuk $B_0, B_1, ..., B_p$-t. Ekkor az A_i modulo 2 összeadása a B_i szorzásával megegyezik. Mivel $A_i + A_j = A_k$, $B_i B_j = B_k$ következik, ahol a szorzás $B_i B_j$ tagonként történik. Méghozzá, a $B_k = B_0 = \{1, 1, 1, ...\}$ kivételével, ami csak akkor történik, ha $i = j$, a B_k $(p-1)/2$ darab 1-et és $(p+1)/2$ darab -1-et tartalmaz.
A $\{b_n\}$ auto-korreláció függvénye:

$$C(\tau) = \frac{1}{p} \sum_{n=1}^{p} b_n b_{n+\tau} = \begin{cases} 1 & \text{ha } \tau = 0 \\ -1/p & \text{ha } 0 < \tau < p, \end{cases}$$

mivel $\{b_n b_{n+\tau}\}$ egy $B_i \cdot B_j$ típusú sorozat, ami viszont B_k típusú, és B_0-nak p-vel több +1-e van a -1-ekhez képest, és a többi B_i-nek eggyel kevesebb +1-e van, mint -1-e. Tehát

11. Tétel. *Minden maximális hosszúságú shift register sorozat az R-3 véletlen tulajdonságot teljesíti.*

Az eddig leírt auto-korrelációs tulajdonságot a csoport karakterek elméletével lehet összekapcsolni.
A B_0, B_1,..., B_p Abel-csoport csak másodrendű elemeket tartalmaz(azaz $B_i^2 = B_0$ minden i-re). Általában egy ilyen csoport r darab másodrendű csoport direkt szorzata, és a nagy csoport karakter táblázata az r darab kis csoport karakter táblázata. Így a $G = \{B_0, B_1, ..., B_p\}$ csoport karakter táblázata csupa 1-ek- és -1-ekből áll. A G csoport karakter táblázatát úgy kapjuk, hogy $B_0, B_1, ..., B_p$ sorozatokat oszloposan egymás mellé rakjuk és efölé a csupa 1-et tartalmazó fő karaktert írjuk vízszintesen. Ebből a nézőpontból a tétel (11) a csoport karakterek ortogonalitási relációjának egy átfogalmazása.

Így a tétel (10) megfordítását is be tudjuk bizonyítani:

12. Tétel. *Legyen $S_1 = \{s_1, s_2, s_3, ...\}$ bármilyen mod 2 sorozat p periódussal, amire $S_1 + S_i = S_k$, ahol $S_i = \{s_i, s_{i+1}, s_{i+2}, ...\}$, és vagy $S_k = \{s_k, s_{k+1}, s_{k+2}, ...\}$, vagy $S_k = \{0, 0, 0, ...\}$. Ekkor $p = 2^r - 1$ valamilyen r egész számra, és S_1 egy maximális hosszúságú shift register sorozat.*

Bizonyítás $S_0, S_1, ..., S_p$ Abel-csoportot alkot a tagonkénti mod 2 összeadásra nézve, amiben az egyes elemnek a rendje 2. Az ilyen csoport r darab 2 rendű csoport direkt összege, tehát $p + 1 = 2^r$.
Továbbá bármilyen r darab nem nulla elem egy bázisát alkotja ennek a csoportnak. Speciálisan $S_{r+1} = \sum_{i=1}^{r} c_i S_i$. Komponensenként nézve, ez egy r-ed rendű rekurziós formulája az S_1-nek. Tehát S_1 egy shift register sorozat $p + 1 = 2^r$ maximális hosszúsággal. □

Tehát a maximális hosszúságú shift register sorozatok R-1-től R-3-ig minden véletlen tulajdonságot teljesítenek.

Legyen $f(x)$ r-ed fokú modulo 2 polinom, és $f(0) = 1$(azaz a konstans tag 1). Ekkor

$$\frac{1}{f(x)} = \sum_{n=0}^{\infty} a_n x^n,$$

ahol az $\{a_n\}$ sorozat shift register által generált. A tétel (1) miatt az $\{a_n\}$ sorozat periódikus. Legyen a periódusa p. Ekkor tétel (2) miatt $f(x)$ osztója

az $(1-x^p)$-nek.
Mivel $f(x)$ osztja az $(1-x^p)$-t, minden $f(x)$ gyöke az $1-x^p$-nek is gyöke. Az $(1-x^p)$-nek a gyökeit p-edik egységgyöknek hívják.
Így minden shift register polinom valamilyen p-edik egységgyök által kielégített egyenlet. (A többi polinom mod 2 $x^p \cdot f(x)$ alakú, tehát $x=0$ is gyök az egységgyökön kívül.) Az irreducibilis polinom periódusa mindig páratlan, mert 2^r-1-et osztja. Fordítva, minden páratlan szám oszt 2^r-1 típusú számot. Így a (2^r-1)-edik egységgyököket érdemes megvizsgálni.

A komplex síkon a p-edik egységgyököket a következő alakban írhatjuk: $z=e^{2n\pi i/p}$, ahol $n=1,2,...,p$. Ha n/p egy nem egyszerűsíthető tört, akkor a megfelelő p-edik gyököt primitívnek hívjuk. A p-edik primitív egységgyökök száma $\phi(p)$. Ha az n/p-t egyszerűsítjük amíg lehet, minden p-edik gyök primitív q-adik gyök, ahol q p-nek valamilyen osztója.

A p-edik körosztási polinom gyökei a primitív egységgyökök. Ez a polinom explicit képlettel is megadható:

$$C_p(x)=\prod_{d|p}(x^d-1)^{\mu(p/d)} \tag{1.7}$$

ahol a kitevőben a Möbius μ-függvény áll. A $C_p(x)$ foka $\phi(p)$ és irreducibilis.

Szeretnénk meghatározni, hogy hogyan faktorizálható a $C_p(x)$ modulo 2. Mivel a $C_p(x)$ osztói p periódusú irreducibilis polinomok, azaz $1-x^p$-t osztják, de $1-x^d$-t nem, ahol d kisebb mint p, hiszen az ilyen osztókat a (1.7) -ben kihagytuk.
Speciálisan a $C_{2^r-1}(x)$ osztói a maximális hosszúságú shift register sorozatnak a karaktrisztikus polinomjai. Tudjuk, hogy ezen polinomok foka r. Ezeknek a szorzata, azaz $C_{2^r-1}(x)$-nek a foka $\phi(2^r-1)$.
Tehát $\phi(2^r-1)/r$ darab maximális kitevőjű r-ed fokú polinom van, ahonnan kijön a (1.6) képlet:

$$\lambda(r)=\phi(2^r-1)/r$$

Minden r-ed fokú irreducibilis polinom osztja az $1-x^{2^r-1}$-et. A periódusuk osztja 2^r-1-et (miközben nem oszt semmilyen kisebb 2^s-1 alakú számot). Ha t 2^r-1 osztója, de nem oszt semmilyen kisebb 2^s-1 számot, akkor a fentiekhez hasonlóan $\phi(t)/r$ darab t-kitevőjű irreducibilis polinom van. Így

$$\Psi_2(r)=\sum_{\substack{t|(2^r-1)\\ t\nmid(2^s-1)}}\frac{\phi(t)}{r}$$

vagyis

$$\Psi_2(r)=\frac{1}{r}\sum_{d|r}(2^d-1)\mu(\frac{r}{d}),$$

ahol a következő összefüggést használtuk:

$$\sum_{t|2^r} \phi(t) = 2^r - 1.$$

Mivel

$$\sum_{d|r} \mu(\frac{r}{d}) = \sum_{d|r} \mu(d) = 0 \quad \text{ha} \quad r > 1,$$

ebből kijön a korábban bevezetett (1.5) egyenlet:

$$\Psi_2(r) = \frac{1}{r} \sum_{d|r} 2^d \mu(\frac{r}{d}).$$

A $C_{2^r-1}(x)$ osztóinak a számát meghatároztuk. De még nem vizsgáltuk meg, hogy az $e^{2n\pi i/p}$ gyökei hogyan oszlanak meg a maximális periódusú polinomok gyökeinek a halmazaira.
Az 1-től $p-1$-ig terjedő egész számok, amiknek nincs közös osztója a p-vel Abel-csoportot alkotnak modulo p szorzásra nézve. Az $1, 2, 4, ..., 2^r - 1$ számok részcsoportot alkotnak r darab elemmel. Ez a részcsoport, bármilyen más csoport elemmel szorozva, egy mellékosztályt alkot.

$C_p(x)$ bármilyen mod 2 osztójának a gyökei ε^n számok, ahol n mod p multiplikatív csoport egy rögzített mellékosztályán fut végig, és ε egy primitív p-edik egységgyök.

Tegyük fel, hogy $A = \{a_n\}$ a darab +1-et és b darab -1-et tartalmaz. Először

$$p = a + b$$

másodszor

$$\sum_{n=1}^{p} a_n = a - b$$

harmadszor

$$\sum_{\tau=1}^{p} pC(\tau) = \sum_{\tau=1}^{p} \sum_{n=1}^{p} a_n a_{n+\tau} \tag{1.8}$$

$$= \sum_{n=1}^{p} a_n \sum_{\tau=1}^{p} a_{n+\tau} = (a-b)^2 \tag{1.9}$$

Másrészt

$$\sum_{\tau=1}^{p} pC(\tau) = pC(0) + \sum_{\tau=1}^{p-1} pC(\tau) = p + K(p-1) \tag{1.10}$$

(1.9) - és (1.10) -ből

13. Tétel. *Ha A az R-3-at teljesíti, akkor*

$$K = \frac{(a-b)^2 - p}{p-1}.$$

Ez egy szükséges feltétel, amit a fázison kívüli K értéknek teljesítenie kell.

Megjegyezzük, hogy ha $a - b = 1$, akkor $K = -1$ és fordítva. Így R-1 és R-3 maga után vonja, hogy:

$$C(\tau) = \begin{cases} 1 & \text{ha } \tau = 0\ , \\ -1/p & \text{ha } 0 < \tau < p \end{cases}$$

amit a maximális hosszúságú shift register sorozatok teljesítenek.

Ha $a = b$, akkor $K = -p/(p-1)$, ami sose egész szám $p > 2$-re.
Azaz az R-3-at sose teljesíti a sorozat, ha ugyanannyi +1 és -1 van(kivéve, ha $p = 2$).

Másik szükséges feltétel a sorozatnak hogy R-3-at teljesítjen az, hogy K- és p-nak ugyanolyan paritású legyen. Azaz $K + p$-nak párosnak kell lennie.

14. Tétel. *Ha A teljesíti az R-3-at, akkor* $K \equiv p \pmod 2$.

Bizonyítás $p = a + b$. A fázison kívüli korreláció felépítéséről tegyük fel, hogy a $\sum a_n a_{n+\tau}$ $1 \cdot 1$-et y-szor, és $-1 \cdot -1$-et z-szer tartalmazza. Ekkor $1 \cdot -1$ $a - y$-szor fordul elő, mivel az első tényező pontosan a-szor +1. De $1 \cdot -1$ $b - z$-szer fordul elő, mivel a második tényező pontosan b-szer -1. Így $a - y = b - z$. Ráadásul $-1 \cdot 1$ $a - y = b - z$-szer fordul elő. Végül:

$$K = \sum_{n=1}^{p} a_n a_{n+\tau} = y - (a-y) - (b-z) + z = 2(y+z) - (a-b)$$

modulo 2-re redukálva $K \equiv a + b \equiv p \pmod 2$ □

Tegyük fel, hogy a bináris sorozat: $A = \{a_n\}$ teljesíti az R-3-at.
Legyen q bármilyen egész szám, ami relatív prím p-vel. Ekkor a $q, 2q, 3q, ..., pq$ ugyanaz a számok mod p, mint $1, 2, 3, ..., p$ a sorrendtől eltekintve. Így $a_q, a_{2q}, ..., a_{pq}$ egy permutációja az $a_1, a_2, ..., a_p$-nek, és a két sorozat összege megegyezik.
Ekkor

$$\sum_{n=1}^{p} a_{qn} a_{qn+\tau} = \sum_{n=1}^{p} a_n a_{n+\tau}$$

vagyis

$$\sum_{n=1}^{p} a_{qn}a_{qn+\tau} = \begin{cases} p & \text{ha } \tau = 0 \\ K & \text{ha } 0 < \tau < p \end{cases}$$

15. Tétel. *Ha $\{a_n\}$ teljesíti R-3-at, akkor $\{a_{qn}\}$ is, feltéve hogy q relatív prím a periódusra, p-re.*

Ez úgy is fogalmazható, hogy: ha egy sorozat R-3-at teljesíti, akkor minden "valódi decimation(tizedelés)"-ja is teljesíti. Itt "tizedelés" azt jelenti, hogy a sorozatból minden q-edik elemet kiválasztjük. "Valódi" azt jelenti, hogy q relatív prím a periódussal p-vel.

$\phi(p)$ darab q $0 < q < p$ szám van, ami relatív prím p-vel, és ahogy már néztük, ezek Abel-csoportot(G) alkotnak a modulo p szorzásra nézve.
Legyen $A_1 = \{a_n\}$ egy p periódusú bináris sorozat, ami R-3-at teljesíti. Tétel (15) miatt $A_q = \{a_{an}\}$ is teljesíti R-3-at minden q-ra a csoportban. Legyen C_0 q-knak a halmaza G-ben, amire A_q csak egy fázis shiftja az A_1-nek(azaz lényegében nem különbözik). Ekkor C_0 egy részcsoportja G-nek, mivel ha $\{a_{qn}\}$ és $\{a_{rn}\}$ csak fázis shiftjai $\{a_n\}$-nek, akkor $\{a_{qrn}\} = \{a_{q(rn)}\}$ és a fázis shiftnek a fázis shiftje is csak fázis shift.
Jelöljük C_0 mellékosztályait $C_1, C_2, ..., C_m$-mel.

16. Tétel. *$\{a_{qn}\}$ és $\{a_{rn}\}$ sorozatok csak fázis shiftban különbözik akkor és csak akkor, ha q és r ugyanabba a C_0 mellékosztályba tartoznak.*

Bizonyítás Ha q és r ugyanabba a C_0 mellékosztályba tartoznak, akkor csak egy tényezővel különböznek C_0-tól. Legyen ez a tényező e : $q = er$. Mivel e C_0-hoz tartozik, $\{a_{en}\}$ csak egy fázis shiftje $\{a_n\}$-nek. Hasonlóan $\{a_{qn}\} = \{a_{e(rn)}\}$ és $\{a_{qn}\}$ is egy fázis shiftje $\{a_{rn}\}$-nek.
Fordítva, ha $\{a_{qn}\}$ fázis shiftja $\{a_{rn}\}$-nek, legyen r^{-1} az r inverze G-ben, és legyen $qr^{-1} = e$. Ekkor $q = er$. Mivel $\{a_{qn}\} = \{a_{e(rn)}\}$ csak egy fázis shiftja $\{a_{rn}\}$-nek, tehát e-nek C_0-hoz kell tartoznia. Ekkor q és r csak egy C_0 tényezővel különböznek, és így ugyanabba a C_0 mellékosztályba tartoznak. □

Ha $p = 2^r - 1$ és $C_0 = \{1, 2, 4, ..., 2^{r-1}\}$, akkor a tétel (16) -beli mellékosztály szerkezete ugyanaz, mint a körosztási mellékosztályoké az 5.5-ben. Megjegyezzük, hogy ez mindig fenn áll a maximális hosszúságú shift register sorozatokra, de először nézzük a tételt:

17. Tétel. *Ha $A = \{a_n\}$ teljesíti R-3-at, és C_0 olyan q elemek részcsoportja, hogy $\{a_{qn}\}$ csak egy fázis shiftja $\{a_n\}$-nak akkor létezik A-nak olyan fázis shiftje: $B = \{b_n\}$, amit C_0 invariánsan hagy.*

Bizonyítás Tegyük fel, hogy C_0-nak van primitív eleme q. Azaz, $C_0 = \{1, q, q^2, ..., q^n\}$, ahol a kitevőket modulo p-ben egyszerűsítjük. Tegyük fel, hogy $\{a_{qn}\} = \{a_{n+\alpha}\}$. Ha $\alpha = 0$, akkor triviális. Különben mindig kiválasztható $\{b_n\} = \{a_{n+\alpha}\}$ úgy, hogy $\{b_{qn}\} = \{b_n\}$. Ekkor $\{b_{q^2n}\} = \{b_{q(qn)}\} = \{b_{qn}\} = \{b_n\}$, és ez hasonlóan minden $C(0)$ eleme q-ra is igaz. Így C_0 invariánsan hagyja $B = \{a_{n+\tau}\}$-t.
Végül, ha C_0-nak két generátora van, a fenti eljárást kétszer alkalmazhatjuk. □

18. Tétel. *Egy A sorozat amit teljesíti R-3-at p periódussal, felbontható +1-ek és -1-ek mellékosztályaira.*

Bizonyítás Tétel (17) miatt feltehető, hogy C_0 teljesen invariánsan hagyja az A-t. Legyen $A = \{a_0, a_1, a_2, ..., a_p\}$. Ha q is benne van a C_0-ban, akkor

$$\begin{aligned} a_1 &= a_q = a_{q^2} = ..., \\ a_2 &= a_{2q} = a_{2q^2} = ..., \\ a_3 &= a_{3q} = a_{3q^2} = ..., stb. \end{aligned}$$

Azaz, ha a_1 +1, akkor $a_q, a_{q^2}, ...$ tagok is +1-ek, és $a_c = 1$ minden c-re C_0-ban. Hasonlóan, ha α és β ugyanabba a mellékosztályba C_j-be tartoznak, akkor $a_\alpha = a_\beta$. □

Ha $\{a_n\}$ maximális hosszúságú lineáris shift register sorozat $p = 2^r - 1$ periódussal, akkor $C_0 = 1, 2, 4, ..., 2^{r-1}$, mivel ez adja az egyetlen mellékosztály felbontást, ahol a mellékosztályok száma $\lambda_2(r) = (1/r)\phi(2^r - 1)/r$.
Ebből shift register sorozatokat felépíthetünk mellékosztályok egymásra rakásával ("superposition").

2. fejezet

GSM titkosítási algoritmusok

2.1. GSM algoritmusok bevezetés

A GSM(Global System for Mobile communication) a legterjedetebb mobil hálózat, és több, mint egy milliárd ember által használt.
A GSM mobil rendszer SIM(Subscriber Identity Module)-nak nevezett kriptográfiai hardvert használ. A GSM hálózat biztonságát úgy tervezték, hogy csak az "air interface" legyen megvédve. Air interface védésnek két célja van: védeni a használókat(főleg titkosítás által), és megvédeni a hálózatot a jogtalan hozzáféréstől(a SIM kriptográfiai hitelesétésével).
A hálózat SIM hitelesítése a mobil és a hálózat beszélgetése előtt történik meg. Miután a mobil azonosítja magát, a hálózat tud hitelesítési folyamatot kezdeni. A folyamat alapvetően challenge-válasz séma, ami a mobil és a hálózat között előre megosztott titkos kulcson(K_i) alapszik. A sémában a hálózat küld egy 128 bites véletlen számot($RAND$) challenge-ként a mobilnak. A mobil átküldi a $RAND$-ot a SIM-nek, ami megszámolja a választ: $SRES = A3(K_i, RAND)$-ot. (Itt $A3$ egy egyirányű függvény.) Ezután a mobil átküldi a $SRES$-et a hálózatnak, és a hálózat összehasonlítja ezt a saját maga által előre kiszámolt SRES értékkel. A beszélgetésnek a titkos kulcsa: $K_c = A8(K_i, RAND)$ a hitelesétéssel párhuzamosan készül el, ahol $A8$ is egy egyirányú függvény.
A beszélgetést innentől kezdve K_c-vel lehet titkosítani, és így a mobil és a hálózat kölcsönösen "hitelesítve" marad, mivel ugyanazt a titkos kulcsot használják. Azonban a hálózat szabályozza a titkosítást, ami nem kötelező, így a támadó könnyen meg tudja személyesíteni a hálózatot egy hamis "base station"-nal, ami nem titkosít.

A GSM titkosítási algoritmusának A5/1-et és A5/2-t használják. A5/1

az eredetileg tervezett algoritmus, A5/2-t pedig a nem-OECD(Organization for Economic Co-operation and Development) országok alkalmazták. A5/1 és A5/2 algoritmusokat titkosan tartották, de mindkettő algoritmust "reverse engineer"-ezték egy konkrét GSM mobiltelefonnal(Briceno, 1999 [2]). Mostanában fedezték fel hogy az A5/2 egyáltalán nem biztonságos, és a nem-OECD országok A5/1-re váltanak. 2002-ben A5/3-at az A5 algoritmusok családjába sorolták. Az A5/3 a KASUMI nevű block-cipheren alapul, és harmadik generációs hálózatban alkalmazzák. A5/3 nem shift register alapú, és ebben a szakdolgozatban nem tárgyaljuk.

2.2. A5/2

A5/2 stream ciphernek van 64 bites kulcsa: K_c és 22 bites publikus kezdeti értéke: COUNT. Jelöljük a COUNT értékét f-fel. Az A5/2 belső állapot négy maximális hosszúságú Lineáris Feedback Shift Registerből(LFSR) áll: R1, R2, R3 és R4, amik 19, 22, 23 illetve 17 bitesek. (A maximális hosszúságú LFSR-t a 1.2 alfejezet (1.4) definícióban definiáltuk.) A 2.1. ábra mutatja az A5/2 algoritmus lényegét, melyet a [7] PhD dolgozata alapján készítettem. Amielőtt a regisztert clockolják, kiszámolják a feedback-ot. Azután a registert egy bittel jobbra shiftelik(a legjobboldalibb bitet kidobják), és a feedback a legbaloldalibb helyen lesz tárolva(0 hely).

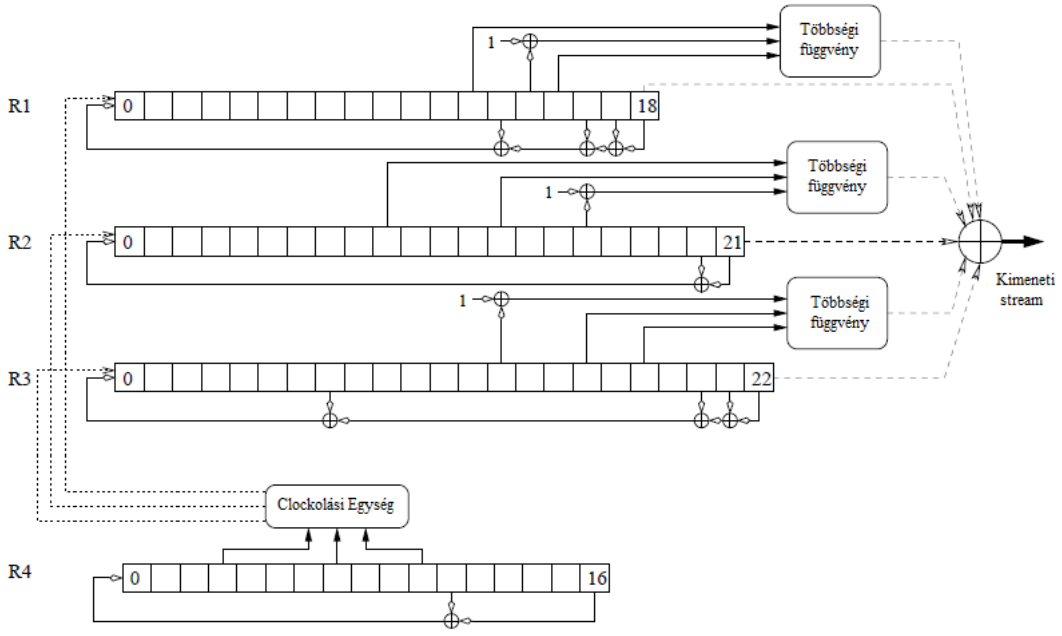
K_c-vel és f-fel inicializálják az A5/2-et a következő négy lépésben.
$K_c[i]$-vel jelölük a K_c i-edik bitjét, $f[i]$-vel az f i-edik bitjét, és $i = 0$-val a legkevésbé szignifikáns("significant") bitet.
Ekkor az inicializálás(key setup) lépései:

1. Állítjuk úgy, hogy: $R1 = R2 = R3 = R4 = 0$
2. $i = 0$-tól 63-ig
   - Mind a négy registert clockoljuk.
   - $R1[0] \leftarrow R1[0] \oplus K_c[i]$; $R2[0] \leftarrow R2[0] \oplus K_c[i]$;
     $R3[0] \leftarrow R3[0] \oplus K_c[i]$; $R4[0] \leftarrow R4[0] \oplus K_c[i]$.
3. $i = 0$-tól 21-ig
   - Mind a négy registert clockoljuk.
   - $R1[0] \leftarrow R1[0] \oplus f[i]$; $R2[0] \leftarrow R2[0] \oplus f[i]$;
     $R3[0] \leftarrow R3[0] \oplus f[i]$; $R4[0] \leftarrow R4[0] \oplus f[i]$.

4. Állitjuk a biteket, azaz: $R1[15] \leftarrow 1$, $R2[16] \leftarrow 1$, $R3[18] \leftarrow 1$; $R4[10] \leftarrow 1$.

Jelöljük az inicializálás(key setup) utáni belső állapotot $(R1, R2, R3, R4) = keysetup(K_c, f)$-fel. Megjegyezzük, hogy a key setup lineáris K_c-ben és f-ben(ha $R1[15]$-, $R2[16]$-, $R3[18]$- és $R4[10]$-et nem mindig 1-re állítjuk).
A5/2 ciklusban működik, ahol minden ciklus végén létrehoz egy kimeneti bitet. Minden ciklusban R1, R2 és R3 registerek között 2 vagy 3 lesz clockolva, R4 három bit értékei szerint. Ezután R4-et clockolnak. Az egyes ciklus elején a $R4[3]$, $R4[7]$ és $R4[10]$ bekerül a clockoló egységbe. A clockoló egység a biteken végrehajt egy többségi függvényt. Ezután a registerek lesznek clockolva a következőek szerint: R1-et clockolnak akkor és csak akkor, ha $R4[10]$ a többséggel megegyezik. R2-öt akkor és csak akkor clockolnak, ha $R4[3]$ megegyezik a többséggel. R3-at akkor és csak akkor, ha $R4[7]$ megegyezik a többséggel. Ezen clockolások után R4-et clockolnak, és a kimeneti bitet R1-, R2-, és R3-ból számolják úgy, hogy XOR műveletet végeznek az egyes registerek legjobboldalibb bitjén és az egyes register többségi értékén(Fig 3.1). Megjegyezzük, hogy a többségi függvény a bemenetnek kvadratikus függvénye: $maj(a, b, c) = a * b \oplus b * c \oplus c * a$. így a kimeneti bit R1, R2 és R3-nak kvadratikus függvénye.
A legelső 99 bitnyi kimenetet kidobják, és a következő 228 bitnyi kimenetet használják kimenetként(keystream). A kimeneti 228 bitet(a keystream) két részre osztják: az első 114 bitet hálózat-mobil irányú kapcsolat titkosítási keystreamként használják, és a másik 114 bitet használják a mobil-hálózat irányú kapcsolat titkosítására. A titkosítás az üzenet és a keystream bitenkénti XOR művelet elvégzéseként történik meg.
Megjegyezzük, hogy az A5/2 az A5/1 szerkezetére épül. Az A5/1 és A5/2 R1, R2 és R3 feedback függvényei ugyanazok. Az A5/2 inicializálási eljárása is hasonlít az A5/1-ére, azzal a különbséggel, hogy az A5/2 R4-et is inicializálja, és minden registerben egy bitet 1-re állítanak inicializálás után. Ezzel ellentétben az A5/1 nem használ R4-et, és nem állít át semmilyen bitet. Ezután A5/2 99 bitet kidob az kimenetből, ezzel ellentétben A5/1 100 bitet dob ki. A clockolás szerkezete is ugyanaz, csak a clockolás bemenete A5/2 esetén R4-ből van, míg A5/1 esetén R1-, R2- és R3-ból.

A5/2 technikai részlete: TDMA frame fogalma
A támadás a TDMA frame fogalmát használja, és itt ezt vázlatosan bevezetjük.
A TDMA(Time Division Multiple Access) azt engedi, hogy legfeljebb 8 különböző mobiltelefont szolgálhasson egy csatorna. Az egyes mobil "time slot"-ban továbbítja az információkat, ami 15/26 milliszekundumig tart. A TDMA



2.1. ábra. A5/2 belső szerkezete

frame száma rögzített mindegyik slotra, amíg a következő TDMA frame-et kap. Az egyes TDMA frame 8 time slotból áll, és az egyes slot 114 bitnyi információt tud továbbítani. Tehát az egyes mobiltelefonnak TDMA frame-onként 114 bit jut, azaz 24.7 Kbit/másodperc.
A COUNT a TDMA frame számából adódik.

2.3. A5/1

Az A5/1 stream cipher egy 64 bites session kulcsot(K_c) és egy 22 bites publikus frame számot(f) használ. A GSM kommunikáció frame-mel történik, ahol minden 4.6 millimásodpercben továbbítanak egy frame-ot. Minden frameben A5/1 session kulcsa és frame száma inicializálódik. A keletkező 228 bites kimenetet(keystream) két részre osztják: az első fele a hálózatról a mobil felé menő adat titkosítására, és a második fele a mobiltelefonról a hálózatra menő adat titkosításra szolgál. A titkosítás az adat és a megfelelő keystream részének a XOR-ozásával történik.
A5/1-nek 64 bites belső állapota van, ami maximális hosszúságú 3 darab Lineáris Feedback Shift Registerekből(LFSR) áll. (A maximális hosszúságú

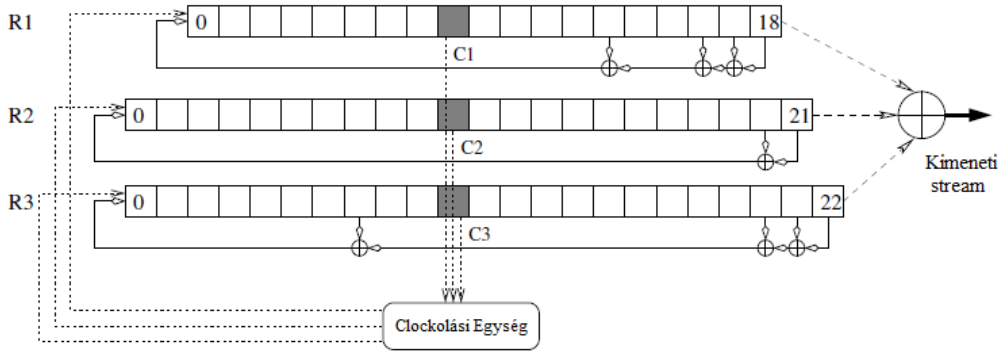
LFSR-t a 1.2 alfejezet (1.4) definícióban definiáltuk.) A 2.2. ábra mutatja az A5/1 algoritmus lényegét, melyet a [7] PhD dolgozata alapján készítettem.
Clockolás alatt az egyes registerek feedbackját kiszámolják(a feedback tapoknak a XOR-ozásaként), aztán shiftelnek a registert jobbra egy bittel(kidobva a legjobboldalibb bitet), és a feedbackot a legbaloldalibb helyen tárolják(zérus hely). A registereket szabályosan clockolnak inicializálás alatt K_c-vel és f-fel(key setup), és szabálytalanul a keystream generáláskor, ahogy azt később megmagyarázzuk.
A5/1-et K_c-vel és f-fel inicializálják három lépésben. Jelöljük a K_c-nek az i-edik bitjét $K_c[i]$-vel, f-nek az i-edik bitjét $f[i]$-vel, és $i = 0$-val a legkevésvé szignifikáns bitet.
Ekkor az A5/1 iniciálizásása(key setup) lépései:

1. Állítjuk úgy, hogy: $R1 = R2 = R3 = 0$
2. $i = 0$-tól 63-ig
   - Mind a három registert clockoljuk.
   - $R1[0] \leftarrow R1[0] \oplus K_c[i]$; $R2[0] \leftarrow R2[0] \oplus K_c[i]$; $R3[0] \leftarrow R3[0] \oplus K_c[i]$.
3. $i = 0$-tól 21-ig
   - Mind a három registert clockoljuk.
   - $R1[0] \leftarrow R1[0] \oplus f[i]$; $R2[0] \leftarrow R2[0] \oplus f[i]$; $R3[0] \leftarrow R3[0] \oplus f[i]$.

A key setup utáni belső állapotot $(R1, R2, R3) = keysetup(K_c, f)$-val jelöljük.

Megjegyezzük, hogy a key setup a K_c és f bitekben lineáris, azaz miután a key setup kész, a belső állapot minden bitje K_c és f-nek a meghatározott helyének a bitjeinek a XOR-ozása. A keystream generálás ciklusokban történik, ahol az egyes ciklusban létrejön egy kimenet bit. A ciklusban R1-, R2- és R3 szabálytalanul clockol(ahogy az később lesz megmagyarázva), és azután a három registernek a legjobboldalibb bitjeinek a XOR-ozást adja ki(Fig 4.1). A kimenet első 100 bitjét kidobják(0,...,99 bitek), azaz 100,...,327 biteket használják a GSM-ben.
A szabálytalan clockolás úgy működik, hogy az egyes registereknek van speciális clockoló tap-ja("clocking tap") a közepénél($R1[8]$, $R2[10]$ és $R3[10]$). A clockolás algoritmusa:

1. A három clockoló tapnak a többségi függvényét kiszámoljuk.



2.2. ábra. A5/1 belső szerkezete

2. Aztán registert clockoljuk akkor és csak akkor, ha a clockoló tapja megegyezik a többséggel.

Megjegyezzük, hogy A5/1 egyes ciklusaiban vagy 2 vagy 3 registert clockolnak(mivel legalább 2 bit megegyezik a többséggel). Feltételezve, hogy a clockolási tapok egyenletesen vannak elosztva, az egyes registerek 1/4 valószínűséggel helyben maradnak, és 3/4 valószínűséggel lesznek clockolva.

3. fejezet

A GSM titkosítás néhány gyengesége

Ennek a fejezetnek a megértését segítheti a 4.2. alfejezetbeli jelölések és elnevezések listája, illetve a vázlatosabb összefoglalás az 4.1. alfejezetben.

3.1. Goldberg, Wagner és Green A5/2 támadása

Először nézzük Goldberg, Wagner és Green A5/2 támadását. [3]

Az első megfigyelés, amin ez a támadás alapul az, hogy mivel $R4[10]$-et "1"-re állítják át az inicializálás utolsó lépésben, az R4-nek ugyanaz az értéke az inicializálás után, mint előtte, attól függetlenül, hogy a COUNT $f[10]$ bit értéke 0 vagy 1.

Mivel R4 szabályozza az R1, R2 és R3 clockolásait, ezek a clockolások függetlenek az $f[10]$ értékétől. A TDMA frame száma és a COUNT közötti fix permutációt figyelembe véve, két frame szükséges, amik $26 \cdot 51 = 1326$ TDMA frame-re(körülberül 6 másodperc) vannak egymástól, és az első frame-nek az $f[10]$ bitje 0. Megjegyezzük, hogy az első frame-nek az $f[10]$ bitje 1 is lehet, ekkor a támadónak várnia kell még legfeljebb 6 másodpercet, hogy az $f[10]$ értéke 0 legyen. A támadó nem tud használni olyan frame-et mint az első frame, ahol $f[10] = 1$, mivel a carry miatt más COUNT bitje lesz megváltoztatva, így az R4 register más lesz a két frameben.

A támadás a következő: Legyenek a két frame számának COUNT értékei f_1 illetve f_2, keystream-jei k_1 illetve k_2. A key setup utáni(a 99 clockolás előtti) első frame registerének értékei legyenek $R1_1, R2_1, R3_1$ illetve $R4_1$.

Hasonlóan jelöljük a második frame-nek a kezdeti belső allapotát, azaz a második frameben a key setup utáni registerek értékei legyenek $R1_2, R2_2, R2_2, R3_2$

illetve $R4_2$. Megjegyezzük, hogy a speciális választása f_1-nek és f_2-nek biztosítja, hogy $R4_1 = R4_2$. Jelöljük ezt az értéket $R4$-gyel. Azonban a többi registerek nem egyenlők, mivel az inicializálási folyamat lineáris f_1-ben, f_2-ben, $R1_1, R2_1, R3_1$ illetve $R1_2, R2_2, R2_2, R3_2$ különbségében és f_1 és f_2 különbségében.
Ezek a különbségek rögzítettek. Így az $R1_1$, $R2_1$ és $R3_1$ a következőképpen fejezhető ki a $\delta_1, \delta_2, \delta_3$ konstans vektorokkal: $R1_1 = R1_2 \oplus \delta_1, R2_1 = R2_2 \oplus \delta_2, R3_1 = R3_2 \oplus \delta_3$.
Most nézzük, hogy ha adott R4 értéke, akkor a $k_1 \oplus k_2$ keystream különbség lineáris $R1_1$-, $R2_1$- és $R3_1$-ben. (k_i az i-edik frame kimeneti keystream-jét jelöli.) Adott R4-re az egész registernek a clockolását tudjuk(és ez megegyezik a két frameben, mivel $R4_1 = R4_2$). Legyenek l_1, l_2 és l_3 a clockolási számok, amivel az R1, R2 és R3 registereket clockolták az i-edik ciklus után. Tehát az i-edik ciklus után az első frameben a registerek értékei $L1^{l_1} \cdot R1_1$, $L2^{l_2} \cdot R2$ és $L3^{l_3} \cdot R3$, ahol $L1$, $L2$, $L3$ mátrixok fejezik ki az adott registerek clockolását. (A mátrixos reprezentálást az 1.2 alfejezet (1.4) -nél vezettük be.) Hasonlóan az i-edik ciklus után a második frameben a registerek értékei $L1^{l_1} \cdot (R1_1 \oplus \delta_1)$, $L2^{l_2} \cdot (R2 \oplus \delta_2)$ és $L3^{l_3} \cdot (R3 \oplus \delta_3)$.

Definiáljuk a $g_i(\cdot)$ függvényt úgy, hogy $g_i(Ri)$ az Ri register kimenet bitje. Ekkor $g_1(R1) \oplus g_2(R2) \oplus g_3(R3)$ az A5/2 kimeneti bitje, ha a registerek belső állapotai rendre R1, R2 és R3.
A $g_1(\cdot)$, $g_2(\cdot)$ és $g_3(\cdot)$ kvadratikus függvények(mivel a többségi függvény egyszer lett alkalmazva).
Goldberg, Wagner és Green észrevette, hogy a kimeneti bitek különbsége kifejezhető az első frame belső állapotának a lineáris függvényeként. Az i-edik ciklus kimeneti bitjeinek a különbsége úgy adható meg, hogy:

$$g_1(L1^{l_1} \cdot R1_1) \oplus g_1(L1^{l_1} \cdot R1_1 \oplus \delta_1) \oplus g_2(L2^{l_2} \cdot R2_1) \oplus g_2(L2^{l_2} \cdot R1_2 \oplus \delta_2) \oplus$$
$$g_3(L3^{l_3} \cdot R3_1) \oplus g_3(L3^{l_3} \cdot R1_3 \oplus \delta_3) = g_{\delta_1}(L1^{l_1} \cdot R1_1) \oplus g_{\delta_2}(L2^{l_2} \cdot R2_1) \oplus g_{\delta_3}(L3^{l_3} \cdot R3_1)$$

ahol a $g_{\delta_i}(\cdot)$ függvény úgy definiálható, hogy $g_{\delta_i}(x) = g_i(x) \oplus g_i(x \oplus \delta_i)$. A $g_{\delta_i}(\cdot)$ függvények lineárisak. Így a kimeneti különbség is lineáris $R1_1$-, $R2_2$-, és $R3_3$-ban. Még meg kell vizsgálni, hogy ha adott egy kvadratikus függvény $g(x_1, ..., x_n)$ és $\Delta = \Delta_1, ..., \Delta_n$, akkor

$$g_\Delta = g(x_1, ..., x_n) \oplus g(x_1 \oplus \Delta_1, x_2 \oplus \Delta_2, ..., x_n \oplus \Delta_n)$$

lineáris x_1-, ..., x_n-ben, ahol $x_i, \Delta_i \in \{0, 1\}$.
Mivel g kvadratikus, ezért úgy írható, hogy

$$g(x_1, ..., x_n) = \sum_{1 \le i,j \le n} a_{i,j} x_i x_j \oplus a_{0,0}$$

ahol $a_{i,j} \in \{0,1\}$ rögzített adott g-re. Így

$$\begin{aligned} g_\delta &= \sum_{1\le i,j\le n} a_{i,j}(x_i x_j \oplus (x_i \oplus \Delta_i)(x_j \oplus \Delta_j)) \\ &= \sum_{1\le i,j\le n} a_{i,j}(x_i x_j \oplus x_i x_j \oplus x_i\Delta_j \oplus \Delta_i x_j \oplus \Delta_i\Delta_j)) \\ &= \sum_{1\le i,j\le n} a_{i,j}(x_i\Delta_j \oplus \Delta_i x_j \oplus \Delta_i\Delta_j) \end{aligned}$$

A végső kifejezés lineáris x_1-, ..., x_n-ben, adott Δ_1-, ..., Δ_n-re.
Tehát ha adott R4 és $k_1 \oplus k_2$, akkor a kezdeti belső állapot $R1_1$ $R2_1$ és $R3_1$ visszanyerhető egy lineáris egyenletrendszer megoldásával. K_c visszanyerhető a kezdeti állapotból $(R1_1, R2_1, R3_1, R4_1)$-ből és f_1-ből A5/2 key setupjának a megfordításával. Mivel R4 ismeretlen, a tamadónak találgatni kell mind a 2^{16} lehetséges értékét az R4-nek, és minden értékre megoldani a lineáris egyenletet, amíg konzisztens megoldást nem talál.
Létezik gyorsabb megoldás hogy ha R4 helyes értékeire szűrünk. Az R1, R2 és R3 kezdeti belső állapotai 61 bitesek(ne felejtsük, hogy R1, R2 és R3 3 bitjei 1-re vannak állítva).
Így $k_1 \oplus k_2$-ből 61 bit szükséges, hogy K_c-t visszanyerjük, míg $k_1 \oplus k_2$ 114 bit hosszú. Tehát meg lehet adni egy túlhatározott lineáris rendszert, aminek a megoldása a belső állapot. A 114 - 61 = 53 darab összefüggő egyenlet nullázódik a Gauss elimináció során. Ezek az egyenletek függnek R4 értékétől, így minden R4 értékére 53 egyenletet írhatunk fel: $V_{R4} \cdot (k_1 \oplus k_2) = 0$, ahol V_{R4} egy 53×114 bites mátrix, és a 0 egy 53 elemű oszlop vektor 53 darab 0-val. A redundancia használható a rossz R4 értékek kiszűrésére, azáltal, hogy leellenőrizzük, hogy $V_{R4} \cdot (k_1 \oplus k_2) = 0$ teljesül-e. Átlagosan két skaláris szorzat szükséges(53 egyenletből), hogy a rossz R4 értéket kizárjuk. Mivel R4-nek 2^{16} lehetséges értéke van, egy átlagos támadás ideje körülberül 2^{16} skaláris szorzat elvégzése plusz az egyenlet rendszer egyszeri megoldásának időtartama.
Egy egyszerű 32-bites személyi számítógépes implementáció, ahol minden lehetséges V_{R4} rendszer előre be van töltve a memóriába, $2^{16}(16 \cdot 114)/8 = 2^{16} \cdot 228$ byteot(körülberül 15 MB-nyi volatile memória) használ el, és egy pár milliszekundum CPU időt igényel(2GHz személyi számitógépen), hogy az R4 helyes értékét kiszűrje. Ha már az R4-et kitaláltuk, megoldhatjuk a lineáris egyenlet rendszert erre a speciális R4-re, hogy visszanyerhessük $R1_1$-, $R2_1$- és $R3_1$-et.
Ilyen egyenlet rendszerek tárolása a Gauss elimináció után körülberül $2^{16} \cdot 64 \cdot 114/8 = 2^1 6 \cdot 912$ byteot, azaz körülberül 60 MB memóriát igényel. Megjegyezzük, hogy ezt a memóriát merevlevezen tárolhatjuk, és R4-gyel indexelhetjük. Adott R4-re a releváns rendszer a volatile memóriából hozható el.

A bonyolultság még tovább csökkenthető, ha $k_1 \oplus k_2$ kevesebb bitjét vesszük. Az eddig leírt támadás viszonylag kevés előkészületet igényel, amik az egyenletek kiszámolásából állnak. Az előkészületek egy személyi számítógépen néhány percen belül elvégezhetők.

3.2. Maximov, Johansson és Babbage A5/1 támadása

Ebben az alfejezetben nézzük Maximov, Johansson és Baggabe támadását.

Maximov, Johansson és Babbage támadása a korábbi Ekdahl és Johansson támadáson([4]) alapul.
Ekdahl és Johansson azt fogalmazták meg, hogy az inicializálás linearitása miatt az egyes registerek kezdeti állapotai megadhatók, és ezt felhasználták korrelációs támadás építéséére.
Maximov, Johansson és Babbage olyan korrelációs egyenletet javasoltak, ami két feltevésre épül: a clockolási -és a step feltevésre.
Először bevezetünk néhány jelölést. Legyen S_1, S_2 és S_3 az R1, R2 és R3 kezdeti belső állapotai a helyes K_c-val a keysetup után, ahol a frame száma 0-nak lett választva, azaz $(S_1, S_2, S_3) = keysetup(K_c, 0)$. $i = 1, 2, 3$-ra jelölje az S_i kezdeti állapottól l_i-szer clockolt Ri-nek a kimeneti bitjét $\tilde{S}_i[l_i]$. Hasonlóan legyenek R1, R2 és R3 registereknek a kezdeti belső állapotai a key setup után F_1^j, F_2^j illetve F_3^j , a csupa 0-t kulcsként használva, a frame szám j-vel, azaz $(F_1^j, F_2^j, F_3^j) = keysetup(0, j)$.
$i = 1, 2, 3$-ra jelölje $\tilde{F}_i[l_i]$ az R_i kimenetét miután a kezdeti állapotból: F_i^j-ből l_i-szer clockoltunk.
Ekdahl és Johansson észrevették([4]), hogy a key setup linearitása miatt j frameszám esetén az Ri kezdeti belső állapota úgy adható meg, hogy $S_i \oplus F_i^j$, azaz $keysetup(K_c, j) = keysetup(K_c, 0) \oplus keysetup(0, j) = (S_1 \oplus F_1^j, S_2 \oplus F_2^j, S_3 \oplus F_3^j)$.
Továbbá, a shift register lineáris feedbackja miatt a j frame-es LFSR kimenete(i), miután l_i-szer clockolták a kezdeti állapotból úgy adható meg, hogy: $\tilde{S}_i[l_i] \oplus \tilde{F}_i^j[l_i]$.

Maximov, Johansson és Babbage a következő feltevéseket fogalmazta meg([5]):

1. clockolási feltevés (j, l_1, l_2, t): Adott j frame-es keystream esetén, R1 és R2 registereket pontosan l_1- illetve l_2-szer clocolták a t ciklus végéig. Jelöljük $Pr((l_1, l_2)$t időpontban)-vel azt a valószínűségét, hogy ez a feltevés teljesül.

2. "step" feltevés (j, t): Adott j frame-es keystream esetén, mind R1, mind R2 registereket clockolják a $t + 1$ ciklusban, de R3 helyben marad. Feltéve, hogy a clockoló tapok értékei egyenletesen vannak elosztva, ez a feltévés 1/4 valószínűséggel igaz.

Maximov, Johansson és Babbage észrevették, hogy ezen két feltevés mellett R3 ugyanazt a bitet adja a kimeneti t és $t + 1$ biteknek. Így az R3 hozzájarulása kitörlődik ezen két kimeneti bit különbségéből, és a következő teljesül:

$$(\tilde{S}_1[l_1] \oplus \tilde{S}_2[l_2]) \oplus (\tilde{S}_1[l_1 + 1] \oplus \tilde{S}_2[l_2 + 1]) = \quad (3.1)$$

$$\tilde{Z}^j[t] \oplus \tilde{Z}^j[t + 1] \oplus (\tilde{F_1}^j[l_1] \oplus \tilde{F_2}^j[l_2]) \oplus (\tilde{F_1}^j[l_1 + 1] \oplus \tilde{F_2}^j[l_2 + 1]) \quad (3.2)$$

ahol $\tilde{Z}^j[t]$ a ciphernek a kimeneti bitje a j frame t időpontjában (a két feltevés mellett). Így a $(\tilde{S}_1[l_1] \oplus \tilde{S}_2[l_2]) \oplus (\tilde{S}_1[l_1 + 1] \oplus \tilde{S}_2[l_2 + 1])$ értéke becsülhető az adott keystreamből és a publikus frame számából.
A (3.2) egyenlet 1 valószínűséggel teljesül, ha mind a clockolási feltevés, mind a step feltevés teljesül.
Az egyik vagy mindkét feltevés nem teljesülése esetén úgy becsülhető, hogy a (3.2) egyenlet 1/2 valószínűséggel teljesül(azaz véletlenszerűen teljesül). Tehát a (3.2) egyenlet igaz a következő valószínűséggel:

$$(1 - Pr((l_1, l_2) \text{ t időpontban}))/2 + Pr((l_1, l_2) \text{ t időpontban})((3/4)/2 + 1/4)$$
$$= 1/2 + Pr((l_1, l_2) \text{ t időpontban})/8.$$

Ennél a valószínűségnél az asszimetria (azaz $Pr((l_1, l_2)$t időpontban)/8) tipikusan kétszer vagy háromszor nagyobb, mint Ekdahl és Johansson támadásának az asszimetriája([4]).
Ez a javítás az asszimetriában várhatóan 4-től 10-ig terjedő szorzóval javítja meg a szükséges frame számot, ami ténylegesen megtörtént a Maximov, Johansson és Babbage támadásánál.

A (3.2) egyenletet egyszerűsíthetjük a Barkan és Biham által javasolt jelölésekkel([6]):
$\tilde{S_i}'[l_i]$-vel jelöljük $\tilde{S}_i[l_i] \oplus \tilde{S}_i[l_i + 1]$-t. Hasonlóan jelöljük $\tilde{F_i}'^j[l_i]$-vel $\tilde{F_i}^j[l_i] \oplus \tilde{F_i}^j[l_i + 1]$-t és $\tilde{Z}'^j[t]$-vel $\tilde{Z}^j[t] \oplus \tilde{Z}^j[t + 1]$-t. Így (3.2) egyenlet úgy írható, hogy:

$$(\tilde{S_1}'[l_1] \oplus \tilde{S_2}'[l_2]) = \tilde{Z}'^j[t] \oplus (\tilde{F_1}'^j[l_1] \oplus \tilde{F_2}'^j[l_2]) \quad (3.3)$$

Az látható, hogy a LFSR linearitása miatt $\tilde{S_i}'[l_i]$-t az Ri kimeneteként tekinthetjük az $S_i' = S_i \oplus S_i^+$ kezdeti állapotból való l_i clockolás után, ahol S_i^+

az Ri belső állapotát jelöli, miután a kezdeti állapot S_i-ből egyszer clockolt. Megjegyezzük, hogy az irreducibilis polinom miatt bijekció van S_i és S_i' között (S_i' mint polinom úgy tekinthető, mint S_i $x+1$-gyel való szorzása modulo az irreducibilis polinom, mivel $x+1$ mindig invertálható modulo egy 2-nél nagyobb fokú irreducibilis polinom). Tehát ha egyszer S_i'-t visszanyerjük, akkor könnyen kitalálható S_i is.
Maxinov, Johansson és Babbage észrevették, hogy jobb eredmények jönnek ki, ha S_i' kimenetének d darab egymást követő bitjével(ahol d egy kicsi egész szám) egyidejűleg dolgoztak. Hívjuk (d bites) szimbólumnak a d darab egymást követő bináris sorozatot: $S_i'[l_i] = \tilde{S_i}'[l_i] \| \tilde{S_i}'[l_i+1] \| \cdots \| \tilde{S_i}'[l_i+d-1]$, ahol "$\|$" az összefűzést jelöli.

Minden l_1 és l_2 indexpárra és minden lehetséges $\delta = {S_1}'[l_1] \oplus {S_2}'[l_2]$ szimbólum különbségre definiáljuk az $E_{l_1,l_2}[\delta]$ estimatort mint logaritmusa annak az aposteriori valószínűségnek, hogy ${S_1}'[l_1] \oplus {S_2}'[l_2] = \delta$.
Ekkor $l_1 = 80$- és $l_2 = 83$-ra az $E_{80,83}[0]$ estimator az ${S_1}'[80] \oplus {S_2}'[83] = 0$ valószínűségének a logaritmusa, és $E_{80,83}[1]$ az ${S_1}'[80] \oplus {S_2}'[83] = 1$ valószínűségének a logaritmusa. Megjegyezzük, hogy magasabb d-re jobb estimator várható.

Maximov, Johansson és Babbage támadása három lépésből áll:

1. Az adott keystreamra az estimatorokat kiszámítjuk a fenti korrelációval.

2. Az estimatorokat dekódoljuk rövid intervallumokon belül, és az r darab legvalószínűbb jelöltek listát táblázatban tároljuk.

3. A jelöltek táblázatát sokféle heurisztikával összevetjük, hogy S_1', S_2' és S_3' értékeire a jelölteket visszanyerjük. Így visszanyerjük a kulcs jelölteket.

A második lépésben az egyes intervallum tartalmának lehetséges értékeinek score-ját kiszámítjuk az estimatorok használátával, és az r darab legmagasabb score-ú jelölt listáját táblázatban tároljuk.

Maximov, Johansson és Babbage dekódolták az estimatorokat 11 szimbólum hosszúságú intervallumokra, azaz $S_1'[69,..,79]$-re és $S_2'[69,..,79]$-re.
Minden egyes ilyen intervallumra és minden lehetséges intervallum tartalmának értékére ($s_1'[69,..,79]$ és $s_2'[69,..,79]$) a score-t kiszamították.
Legyen $I = [69,..,79]$. Ekkor $s_1'[I]$ és $s_2'[I]$ $2^{2(11+d-1)}$ darab értéket vehet fel. A jelölt értékére a score-t kiszámoljuk a következő szerint:

$$score(s_1'[I], s_2'[I]) = \sum_{l_1,l_2 \in I} E_{l_1,l_2}[s_1'[l_1] \oplus s_2'[l_2]]$$

Az intervallum r darab legmagasabb score-ú lehetséges értékét táblázatban tároljuk. Ezt az eljárást minden intervallumra elvégezzük.

Minden egyes intervallum érték pár score-jának kiszámítása $|I|^2$ darab elem összeadásával jár, ahol $|I|$ a szimbólumok száma az intervallumban(a mostani esetben $|I| = 11$).
Tehát a támadás második lépésének az időigénye $|I|^2 \cdot 2^{2(|I|+d-1)}$ szorozva az intervallumok számával. (Az S' jelölés bevezetésével ez az időigyény a Maximov, Johnsson és Babbage eredeti támadásánál 4-es szorzóval kevesebb. Ez a javítás az S' és S közötti bijekcióból származik.)

A harmadik lépésben az S'_1, S'_2 és S'_3 értékeket kombináljuk, hogy a kulcs jelöltek listáját megadjuk. A kulcs jelöteket az adott keystreammel ellenőrzzük.

A támadás szimulációját 2.4 GHz-es 256Mb-os RAM Pentium-4 CPU-val végezték, Windows XP Pro SP1 operációs rendszer alatt. 2000 frame-mel végzett szimulációban a támadás első két lépése $d = 1$ esetén 11 másodpercet igényel, és $d = 4$ esetén pedig 40 másodpercet. A szimuláció szerint összességében a támadás néhány másodperctől néhány percig terjedő időt igényel, amíg az adott keystreamból és publikus frame számából a helyes kulcsot visszanyerjük. Az időigény a frame számtól, d-től, a jelöltek lista méretétől(r) és az összevetési heurisztikától függ.

3.3. Barkan és Biham A5/1 támadása

Ez az alfejezet Elad Barkan és Eli Biham "Conditional Estimators: an Effective Attack on A5/1" című cikkére hivatkozik([6]).

Barkan és Biham elöször a Maximov, Johansson és Babbage támadás korrelációs asszimetriáját(lásd az előző alfejezetet) javították azáltal, hogy a step feltevést két esetre bontották és az esetekre feltételes estimatort alkalmazták. Másodszor az R2 register három gyengeségét fogalmazták meg, amivel nagyon hatékony estimatorok dekódolást tették lehetővé.

Tekintsük az R1 és R2 registereket, és tegyük fel, hogy az adott j framere és a t-edik kimeneti bitre a clockolási feltevés igaz. Továbbá tegyük fel, hogy tudjuk az $\tilde{S}_1[l_1 + 10]$ és az $\tilde{S}_2[l_2 + 11]$ értékeit. Barkan és Biham támadása a publikus frame számát(j) használja a clockoló tapoknak(R1-nek $C1 = \tilde{S}_1[l_1 + 10] \oplus \tilde{F_1}^j[l_1 + 10]$-t és R2-nek $C2 = \tilde{S}_2[l_2 + 11] \oplus \tilde{F_2}^j[l_2 + 11]$) a t-edik kimeneti bitjének kitalálására.
Barkan és Biham észrevették, hogy a korrelációs asszimetriája("bias") javítható azáltal, hogy a step feltevést két esetre bontjuk. Az első eset az, amikor

$C1 \neq C2$. A clockolás szerkezete miatt R3-at mindig clockolják vagy R1-gyel vagy R2-vel. A step feltevés nem igaz, és ezért a (3.3) egyenlet igaz az esetek felénél.
Azonban a második esetben, amikor $C1 = C2$, nyerünk egy kettes szorzót a bias-ban. Ebben az esetben mind az R1-et, mind az R2-t clockolják(mivel $c = C1 = C2$ a többség), és R3-at 1/2 valószínűséggel clockolják, feltéltelezve, hogy a clockoló tapok értékei egyenletesen vannak elosztva(R3-at akkor clockolják, ha $C3 = c$). Tehát amikor $C1 = C2$, a step feltevés 1/2 valószínűséggel igaz, míg Maximov, Johansson és Babbage támadásánál csak 1/4 valószínűséggel az.
Barkan és Biham ezeket a megfigyeléseket használták a feltételes estimator("conditional estimator") építésére.
Definiáljuk az l_i indexre a d bites clockolás szimbólumát d-bites stringként: $S_i[l_i] = \tilde{S}_i[l_i] \| \tilde{S}_i[l_i + 1] \| \cdots \| \tilde{S}_i[l_i + d - 1]$, ahol "$\|$" az egymásra fűzést jelöli.

Az $E_{l_1,l_2}[x|Sc]$ feltételes estimatort kiszámoljuk minden lehetséges $Sc = S_1[l_1 + 10] \oplus S_2[l_2 + 11]$ clockolás szimbólum különbségre és minden $x = S'_1[l_1] \oplus S'_2[l_2]$ szimbólum különbségre.
Az $E_{l_1,l_2}[x|Sc]$ estimator annak az aposteriori valószínűségnek a logaritmusa, hogy a szimbólum különbség értéke x, feltéve hogy az adott clockolás szimbólum különbsége Sc. A feltételes estimatorok kiszámítása hasonló a estimatorok kiszámításához, azzal a különbséggel, hogy a clock szimbolúm különbség hatását is beleszámoljuk.

A Barkan és Biham támadása három R2 gyengeségen alapszik.

Az első gyengesége("alignment" tulajdonság) az, hogy az R2 feedback tapja korrelációs egyenlet által becsült bitekkel egybeesik.
Tegyük fel, hogy tudjuk az S_1 értékét. Ekkor minden i indexre a korrelációs egyenlet becsüli az $S_2[i] \oplus S_2[i + 1]$ értékét. Másrészt az R2 lineáris feedback megköveteli, hogy: $S_2[i] \oplus S_2[i + 1] = S_2[i + 22]$(tehát $S'_2[i] = S_2[i + 22]$). Így a korrelációs egyenlet igazából 22 bit távolságra lévő biteket becsül.

A második gyengesége("folding" tulajdonság) az, hogy az R2-nek csak két feedback tapja van, és ezek a tapok szomszédosak. Legyen $X[*]$ a bitek stringje, ami az R2-nek a kimenete. Továbbá legyen a $cost(i, x)$ költségfüggvény, ami minden egyes d-bites x stringhez és egy i indexhez (i az $X[*]$ bitjein fut végig) egy költséget rendel hozzá.
Kiszámoljuk az adott $X[*]$ stringnek az összes költségét(azaz "scoreját") úgy,

hogy:

$$\sum_{i=i_s}^{length(X)-d+1} cost(i, X[i]\|X[i+1]\|\cdots\|X[i+d-1]), \tag{3.4}$$

ahol i_s az első bit, amire számoljuk a scoreját, és $length(X)$ az utolsó bit, amire számoljuk a scoreját. Bevezethetünk egy új $cost'(i,x)$ költség függvényt, ahol az i-t az első 22 indexből vesszük. Az R2 második gyengesége az, hogy az első 22 index-szel számolt $cost'$ megegyezik a (3.4) egyenlettel kiszámolt score-ral(amit minden indexen számoljuk $cost$ függvény használással). $cost'(t,x)$ azonban d'-bites x stringeken értelmezett, ami kicsivel hosszabb, mint d. Minden 22-vel további index(az első 22 indexen túl) az $X[*]$-ban egy bittel több hosszat ad az x-hez, tehát

$$d' = d + \lceil (length(X) - i_s - d + 1 + 1 - 22)/22 \rceil .$$

A d'-bites stringeket reprezentatív szimbólumoknak hívjuk.
Minden i indexre az első 22 index közül az egyenlőség $X[i] \oplus X[i+1] = X[i+22]$ igaz az R2 lineáris feedback tap miatt. Más szóval az i index d'-bites stringje meghatározza a $i+22$ index $(d'-1)$-bites stringjét(ami a XOR különbsége az i index bármely két szomszédos d'-bites stringjének). Ez a string határozza meg az $i+2\cdot 22$ index $(d'-2)$-bites stringjét, és az $i+3\cdot 22$ index $(d'-3)$-bites stringjét, stb. A score-t minden index $cost$ összegéként számolják, a (3.4) egyenlet szerint. Ugyanazt a score értéket kaphatjuk, ha minden $i \in \{i_s, ..., i_s+21\}$-re először összegezük a $cost$-ját minden indexre, ami kongruens i-vel modulo 22, aztán tároljuk az eredményt i index $cost'$ függvényébe, és végül az első 22 indexre: $(i_s, ..., i_s+21)$-re összegezzük a $cost'$-jait.
Így "behajthatjuk"("fold") minden indexre definiált $cost$ függvényt az első 22 indexre definiált $cost'$ függvényre.

A harmadik R2 gyengesége(szimmetria tulajdonság) az, hogy a pont közepén van a clockoló tapja. A folding tulajdonsággal együtt kombinálva az R2 clockoló tapja és a kimeneti tapja között szimmetria alakul ki. A szimmetria tulajdonság hatékony támadást tesz lehetővé a feltételes estimator használatával. Tegyük fel, hogy tudjuk az S_1-et. $S_2[i]$ az R2 kimeneti tapja, amikor $S_2[i+11]$ az R2 clockoló tapja. Amikor $S_2[i+11]$ eléri a kimeneti tapot, $S_2[i+11+11] = S_2[i+22]$ a clockoló tapnél van.
Azonban az i index reprezentatív szimbóluma meghatározza mind az $S_2[i]$ bitjét és az $S_2[i+22]$ bitjét. Tehát a reprezentatív szimbólumokat párokba oszthatjuk, ahol az egyes pár i index reprezentatív szimbólumát és $i+11$

index reprezentatív szimbólumát tartalmazza. Az egyik reprezentatív szimbólum clockolásra szolgál, a másik kimenetre, és fordítva. Így a reprezentatív szimbólum pár az egymásnak a clockolását szabályozza. Ez a reprezentatív szimbólum párba osztás nem lenne lehetséges, ha a clockoló tapok nem lettek volna középen.

Barkan és Biham támadása három lépésből áll.

1. Számoljuk ki a feltételes estimatorokat.

2. Az estimatorokat dekódoljuk és a legvalószínűbb jelölt (S_1, S_2) párokat listázzuk úgy, hogy a legvalószínűbb jelölt párok keresésnek problémáját egy gráfelméleti feladattá alakítjuk át.

3. Az (S_1, S_2) listában az egyes jelöltre visszanyerjük az S_3 jelöltjeit. Minden egyes ilyen jelöltre visszanyerjük a kulcsot, és ezt ellenőrizzük próba titkosításokkal.

Az első lépés feltételes estimatorjait kiszámoljuk az eddig leírtak szerint. A harmadik lépésben, ha adott az (S_1, S_2) jelölt pár, akkor az S_3 lehetséges értékét kitalálhatjuk az (S_1, S_2)-ből és a konkrét frame-nek a keystream-jéből.

A következőben a második lépést vázoljuk.
Ha az (S_1, S_2) jelölt pár értékeit (s_1, s_2)-vel jelöljük, akkor az (s_1, s_2) scoreja:

$$score(s_1, s_2) = \sum_{l_1, l_2} E_{l_1,l_2}[s'_1[l_1] \oplus s'_2[l_2] | s'_1[l_1 + 10] \oplus s'_2[l_2 + 11]]. \qquad (3.5)$$

A legvalószínűbb jelöltek listája olyan $\{(s_1, s_2)\}$ jelöltekből áll, aminek a scorejai a lehető legmagasabbak.
A legvalószínűbb jelöltek listájának kiszámításához úgy alakítjuk át ezt egy gráfelméleti problémává, hogy egy nagy gráffal modellezzük, aminek van egy s forrás csúcsa és egy t nyelő csúcsa, és minden s-ből t-be vezető út megfelel a jelölt (S_1, S_2) értékének úgy, hogy a pár scoreját az út menti élek költség összegének feleltetjük meg. Így a legmagasabb a score-ú út felel meg a pár a legmagasabb score-jának. A Dijkstra algoritmussal megtalálható a legköltségesebb út, mivel az élek súlyai negatívak(valószínűség logaritmusa).

A gráf költség függvényét úgy definiálták, hogy: $cost(i, x|y) = func_{i,s_1}[x|y] = \sum_{l_1} E_{l_1,i}[s'_1[l_1] \oplus x | s_1[l_1+10] \oplus y]$, ami a $cost'(i, x|y)$-be lesz "behajtva"("folded"). Mivel az egyes $i|i + 11$ index két indexet egyesít, ezért az $i|i + 11$-re vezető élnek az i és az $i + 11$ indexek összegét kell tartalmaznia, tehát az él költsége $nsr(i, s'_2[i]|s_2[i + 11]) = cost'(i, s'_2[i]|lsb_{d'}(s_2[i + 11])) + cost'(i + 11, s'_2[i +$

$11]|s_2[i+22])$ ahol $lsb_{d'}(x)$ az x első d' bitje. Megjegyezzük, hogy $s'_2[i+11] = D(s_2[i+11])$ ahol $D(\alpha_1, \alpha_2, \alpha_3, ..., \alpha_{d'}) = (\alpha_1 \oplus \alpha_2, \alpha_2 \oplus \alpha_3, ..., \alpha_{d'-1} \oplus \alpha_{d'})$ az az operátor, ami számolja az XOR különbséget a szomszédoos bit párra. Továbbá az alignment tulajdonság miatt $s_2[i+22] = s'_2[i]$. Tehát $nsr(i, s'_2[i]|s_2[i+11]) = cost'(i, s'_2[i]|lsb_{d'}(s_2[i+11])) + cost'(i+11, D(s'_2[i+11])|s'_2[i])$.

Barkan és Biham támadásának szimulációja 1500 és 2000 frame számokkal történt, továbba $d = 1$, $l_1 \in \{61, ..., 144\}$, $l_2 \in \{70, ..., 135\}$ és $|l_1 - l_2| < 10$-re számolták az estimatorokat. A támadás hármadik lépését 64-bites kulccsal végezték. A szimulációt 1.8GHz-es 512MB-os RAM Pentium-4 Mobile CPU-val vézezték, a Windows XP Cygwin operációs rendszer alatt. Az $\{(s_1, s_2)\}$ listája méretét 5200-ra korlátozva a kulcsot az esetek 64 %-ában találták ki, a korábbi támadások 5 %-a arányához képest. A támadás első lépése körülbelül 7 másodpercig tartott. A második lépés 340 másodpercet igényelt az első párra, miután 1500 pár jelöltet generálták pár másodperc alatt. A harmadik lépésben körülberül 20.4 jelöltet futották át másodpercenként. Tehát a futási idő a listában a helyes pár helyétől függ. A legjobb esetben 350 másodpercet, a legrosszabb esetben 10 percet igényel a támadás.

4. fejezet

Összefoglalás

4.1. Összefoglalás

Egy egyszerű módszer a bináris sorozat generálásra a shift register használata mod 2 feedback-kel. Az így generált sorozat periódikus, és a periódusa nem nagyobb, mint $2^r - 1$, ahol r a shift register lépcsőszáma. Minden $\{a_n\}$ shift register sorozat teljesíti az alábbi lineáris rekurziót:

$$a_n = \sum_{i=1}^{r} c_i a_{n-i},$$

ahol $\{c_i\}$ értéke 0 vagy 1, aszerint, hogy az elem szerelpel-e a feedback-ben vagy sem.

Generátor függvény segítségével belátható a következő:

$$\sum_{n=0}^{\infty} a_n x^n = \frac{g(x)}{f(x)},$$

ahol a számlálóban olyan polinom áll, aminek a foka kisebb, mint r, és a nevező a karakterisztikus polinom: $f(x) = 1 - \sum_{i=1}^{r} c_i x^i$.

A shift register sorozat periódusa p a legkisebb pozitív egész szám, amire az $1 - x^p$ osztható $f(x)$-szel.

$f(x)$ irreducibilitása a szükséges feltétel, hogy az $\{a_n\}$ periódusa $2^r - 1$ legyen.

Egy $\{a_n\}$ bináris sorozat akkor és csak akkor maximális hosszúságú, ha a periódusa: $p = 2^r - 1$. Az r-et az $\{a_n\}$ fokának hívjuk. A szükséges és elégséges feltétel, hogy az $\{a_n\}$ maximális hosszúságú legyen az, hogy $f(x)$ az $1 - x^m$-et osztja $m = p$-re, de olyan pozitív m-re nem, ami kisebb, mint p.

Ha egy sorozat maximális hosszúságú, akkor a karakterisztikus polinomja irreducibilis. Továbbá láttuk, hogy egy polinomról hogyan dönthető el, hogy irreducibilis-e, és ha irreducibilis, akkor maximális kitevőjű-e.

Az Euler függvény segítségével foglamaztuk meg a következő tételt: A $p = 2^r - 1$ hosszúságú különböző (nem tekintjük különböző sorozatoknak, ha csak a kezdő pont különbözik) maximális hosszúságú sorozatok száma $\phi(p)/r$, ahol ϕ az Euler függvény.

Egy maximális hosszúságú shift register sorozat teljesíti mind a három véletlen tulajdonságot, amit kitűztünk. Ez a három tuladonság jellemzi a maximális shift register sorozatokat, amik pszeudo-zaj sorozatok.

Bevezettük a modulo p körosztási mellékosztály fogalmát. Ennek a segítségével építhetünk maximális hosszúságú lineáris rekurziós sorozatokat a mellékosztályok egymásra rakásának módszerével, vagy egy új maximális hosszúságú sorozat képződést a tizedelés("decimation") módszerével.

A GSM hálózat az A5/1 és az A5/2 nevű shift register alapú algoritmust használja titkosításra. Az A5/2 esetén négy(R1, R2, R3 és R4), az A5/1 esetén három(R1, R2 és R3) maximális hosszúságú lineáris feedback shift registert használnak. Az A5/1 és A5/2 ciklusokban működnek. Az egyes ciklus menete: elöször betöltjük a kulcsot(K_c) és a frame számot(f) a regisztekbe. A registerek inicializálása(key setup) a kulccsal és a frame számával történik. Az inicializálás alatt a registerek szabályosan lesznek clockolva a kulcssal és a frame számmal. (Az A5/2 esetén az inicializálás végső lépésében az egyes register egy bitje 1-re lesz állítva.) Az egyes ciklusban az egyes registerek feedbackjét kiszámolják, aztán shiftelik a registert jobbra egy bittel(kidobva a legjobboldalibb bitet), és a feedbacket a legbaloldalibb helyen tárolják. Ezután az R1, R2 és R3 registerek közül 2 vagy 3 szabálytalanul lesz clockolva:

- A5/2 esetén R4 három bitjén(clockoló egység) végzett többségi függvény értéke szerint, és
- A5/1 esetén az R1, R2 és R3 egy meghatározott bitjének(clockoló tap) értékei szerint.

Az A5/2 esetén ezután R4 is clockolva lesz. Az egyes ciklus végén egy bites kimenet keletkezik. A kimenetből az A5/2 esetén a legelső 99 bitet, az A5/1 esetén a legelső 100 bitet dobják ki. A következő 228 bitnyi kimenetet(keystream) használják titkosításra.

Goldberg, Wagner és Green támadása azt használja ki, hogy az $R4[10]$-et 1-re állitják át az inicializálás utólsó lépésében. Így R4-nak ugyanaz az értéke van az inicializálás után is, az $f[10]$ értékétől függetlenülen. Tehát az R1, R2 és R3 clockolásai is függetlenek az $f[10]$ értékétől. A kimeneti bitek különbsége az első frame belső állapotának lineáris függvényeként kifejezhető.

Tehát adott R4 értékre és a kimeneti bitek különbségére($k_1 \oplus k_2$) az első frameben az R1, R2 és R3 belső állapotai($(R1_1, R2_1, R3_1, R4_1)$) visszanyerhetők azáltal, hogy megoldunk egy lineáris egyenletrendszert. K_c visszanyerhető az $(R1_1, R2_1, R3_1, R4_1)$-ből és az f_1-ből A5/2 key setupjának a megfordításával. Mivel R4 ismeretlen, a támadónak vagy találgatnia kell mind a 2^{16} lehetséges értékét az R4-nek, vagy R4 helyes értékeire szűrnie. Ha az első módszert alkalmazzuk, minden értékre meg kell oldani a lineáris egyenletet, amíg konzisztens megoldást nem találunk. A második módszer esetében a redundanciát kihasználva felépítünk egy túlhatározott lineáris egyenletrendszert, és így gyorsítjuk a számolást. A redundancia azon alapul, hogy: R1, R2 és R3 61 bitesek, és $k_1 \oplus k_2$ 114 bites. A K_c rekonstruáláshoz pedig csak 61 bit kell, és ezért 114 - 61 = 53 darab összefüggő egyenlet nullázódik a Gauss elimináció során.

A támadás előkészítése (az egyenletek kiszámolása) egy személyi számítógépen néhány percen belül elvégezhetők.

Maximov, Johansson és Babbage támadása korrelációs egyenleten alapszik, ami a következő két feltevésre épül: clockolási feltevés: feltesszük, hogy az R1 és R2 registert megadott számokkal clockolták a t-edik ciklusig. step feltevés: azt teszi fel, hogy az adott ciklusban ($t + 1$-edik ciklus) mind R1-et és R2-t clockolták, de R3 helyben maradt.

Maximov, Johansson és Babbage észrevették, hogy ezen két feltevés mellett R3 ugyanazt a bitet adja a kimeneti t és $t + 1$ biteknek.

Ezzel az észrevétellel belátható, hogy a kimenet bit különbség az adott keystreamból és publikus frame számából becsülhető.

Ezen kimenet bit különbségből viszont az estimatorok kiszámolhatók, továbbá a registerek állapotára a legvalószínűbb jelöltek listája elkészíthető. Végül ezekből a kulcsot vissza tudjuk nyerni.

A két feltevéssel 2 vagy 3 szorzóval erősebb korrelációs asszimetriát(azaz $Pr((l_1, l_2)$t időpontban)/8-t) nyertek. Ezzel a korábbi Ekdahl és Johansson támadáshoz képest a szükséges frame számát javították 4-től 10-ig terjedő szorzóval.

A szimuláció szerint összességében a támadás néhány másodperctől néhány percig terjedő időt igényel. Az időigény a frame számtól, d-től, a jelöltek lista méretétől(r) és az összevetési heurisztikától függ.

Barkan és Biham elöször a Maximov, Johansson és Babbage támadás korrelációs asszimetriáját javították tovább azáltal, hogy a step feltevést két esetre bontották és az esetekre feltételes estimatort alkalmazták. Másodszor az R2 register három gyengeségét fogalmazták meg, amivel nagyon hatékony estimatorok dekódolást tették lehetővé. Az első gyengesége ("alignment"

tulajdonság) hasznosítja azt, hogy a korrelációs egyenlet egybeesik az R2 feedback tapjával. A második gyengesége ("folding" tulajdonság) azt használja, hogy R2-nek csak két feedback tapja van, és szomszédosak. A folding tulajdonság használatával az estimatorokat optimálisan tudták dekódolni. Továbbá legvalószínűbb jelölt $\{(S_1, S_2)\}$ párok listájának kiszámolására hatékony módszert mutattak. Így adott S_1- és S_2-ből tudjuk visszanyerni az S_3-at a keystream-ből. Az R3 harmadik gyengesége (szimmetria tulajdonság) azon alapszik, hogy az R2 clockoló tapja pontosan középen van, ami a folding tulajdonsággal együtt a clockoló és a kimeneti tapok közötti szimmetriát hoz létre. A szimulációjuk szerint az új támadásuknak magasabb a siker aránya, gyorsabb, és kevés előszámolást igényel a korábbi tamádásokhoz képest.

4.2. Jelölések és elnevezések listája

clockolás	Egy register alapművelet, amialatt periódikus időközönként x_i tartalma x_{i+1}-be kerül
tap	a shift register tároló elemének a helye, ami a shift register következő állapotát befolyásolja
feedback tap	a shift register tároló elemének a helye, ami a feedback függvény által a shift register következő állapotát befolyásolja
clockoló tap	a shift register tároló elemének a helye, ami a shift registerek clockolását befolyásolja az A5/1- vagy A5/2-ben
clockoló egység	a clockoló tapok halmaza
$R1, R2, R3$	az A5/1- és A5/2-ben használt maximális hosszúságú lineáris feedback shift register vagy a shift register állapota
$R4$	az A5/2-ben használt maximális hosszúságú lineáris feedback shift register
$frame$	amivel történik a GSM beszélgetés. Megadott időközönként továbbítják az új frame-et, amivel inicializálják a kulcsot és a frame számot
K_c	titkos kulcs
f	frame szám
$K_c[i]$	K_c i-edik bitje
$f[i]$	f i-edik bitje
$i = 0$	legkevésbé szignifikáns bit
$keysetup$	az A5/1 illetve A5/2 inicializálása (K_c-vel és f-fel történik meg)
$keysetup(K_c, f)$	a keysetup utáni registerek belső allapotai (a K_c és f függvénye)
$keystream$	Az A5/1 illetve A5/2 kimenete, amit titkosításra használnak
Ri_j	a j-edik frameben az Ri register keysetup utáni állapota
f_i	az i-edik frame száma
$g_i(Ri)$	a függvény, ami az Ri register kimenet bitjét adja (3.1. alfejezet)

Li	Ri register clockolást kifejező mátrix (3.1. alfejezet)
l_i	ahányszor Ri registert clockolták az i-edik ciklus után (3.1. alfejezet)
δ_i	konstansok, amire $Ri_1 = Ri_2 \oplus \delta_i$ tejesül (3.1. alfejezet)
$g_{\delta_i}(\cdot)$	függvény, amire $g_{\delta_i}(x) = g_i(x) \oplus g_i(x \oplus \delta_i)$ tejesül (3.1. alfejezet)
k_i	i-edik frame kimeneti keystream-je (3.1. alfejezet)
S_i	az Ri állapota a keysetup után, 0 frame számmal inicializálva (3.2. alfejezet)
$\tilde{S}_i[l_i]$	az Ri kimenete az S_i állapottól való l_i-szeres clockolás után (3.2. alfejezet)
F_i^j	Ri kezdeti állapota, j frame számmal, csupa 0 kulccsal (3.2. alfejezet)
$\tilde{Z}^j[t]$	az A5/1 kimeneti bitje j frame t időpontjában, mind a két (clockolási és step) feltevés mellett (3.2. alfejezet)
$Pr((l_1, l_2)$ t idopontban)	annak a valószínűsége, hogy a clockolási feltevés igaz az adott paraméterekkel(3.2. alfejezet)
bias	a korrelacios asszimetria (3.2. alfejezet)
korrelációs asszimetria	az algoritmus statisztikai gyengesége (nem teljesen véletlenszerű szabályossága az algoritmusnak) (3.2. alfejezet)
$\tilde{S_i}'[l_i]$	jelölés a $\tilde{S}_i[l_i] \oplus \tilde{S}_i[l_i + 1]$-re (3.2. alfejezet)
$\tilde{F_i}'^j[l_i]$	jelölés a $\tilde{F_i}^j[l_i] \oplus \tilde{F_i}^j[l_i + 1]$-re (3.2. alfejezet)
$\tilde{Z}'^j[t]$	jelölés a $\tilde{Z}^j[t] \oplus \tilde{Z}'^j[t]$-ra (3.2. alfejezet)
S_i^+	az Ri állapota, miután a S_i-ből egyszer clockolták (3.2. alfejezet)
S_i'	jelölés a $S_i \oplus S_i^+$-ra (3.2. alfejezet)
szimbólum	d bites szimbólumnak hívjuk a d darab egymást követő bitet (3.2. alfejezet)
d	a szimbólum méret jelölése bitekben (3.2. alfejezet)
$E_{l_1,l_2}[\delta]estimator$	logaritmusa annak az aposteriori valószínűségnek, hogy $S_1'[l_1] \oplus S_2'[l_2] = \delta$ (3.2. alfejezet)
clockolás szimbólum	az l_i indexre a d bites clockolás szimbólum egy d-bites string, ami a következő alakú: $S_i[l_i] = \tilde{S}_i[l_i] \| \tilde{S}_i[l_i + 1] \| \cdots \| \tilde{S}_i[l_i + d - 1]$, ahol "$\|$" az egymásra fűzést jelöli. (3.3. alfejezet)

Sc	egy adott clockolás szimbólum különbsége $Sc = S_1[l_1 + 10] \oplus S_2[l_2 + 11]$ (3.3. alfejezet)
$E_{l_1,l_2}[x\|Sc]$ *feltételes estimator*	logaritmusa annak az aposteriori valószínűségének, hogy $Sc = S_1[l_1 + 10] \oplus S_2[l_2 + 11]$ és $x = S'_1[l_1] \oplus S'_2[l_2]$ (3.3. alfejezet)
$X[*]$	az R2 kimeneti bit string-je (3.3. alfejezet)
i_s	az $X[*]$ első indexe (3.3. alfejezet)
$cost(i, x)$	d-bites x stringen és az i indexen értelmezett költségfüggvény (3.3. alfejezet)
$cost'(i, x)$	$cost(i, x)$-ből úgy kapott függvény, hogy az i-t csak az első 22 indexből vesszük (3.3. alfejezet)
reprezentatív szimbólum	*cost*-ról *cost'*-re való áttérés során a string hossz is d-ről d'-re változik. Ezt a d'-bites stringet hívjuk reprezentatív szimbólumnak. (3.3. alfejezet)
score	a költségek összege. $X[*]$ stringen (3.4). (s_1, s_2)-re (3.5) szerint definiált. (3.3. alfejezet)
$lsb_{d'}(x)$	az x első d' bitje (3.3. alfejezet)
$nsr(i, x)$	a következőképpen definiált él költség függvény: $nsr(i, s'_2[i]\|s_2[i + 11]) = cost'(i, s'_2[i]\|lsb_{d'}(s_2[i + 11])) + cost'(i + 11, s'_2[i + 11]\|s_2[i + 22])$ (3.3. alfejezet)
$func_{i,s_1}[x]$	a $cost(i, x)$ értéke, amit úgy definiálunk, hogy: $\sum_{l_1} E_{l_1,i}[s'_1[l_1] \oplus x]$ (3.3. alfejezet)
$D(\alpha_1, \alpha_2, ..., \alpha_{d'})$	az operátor, ami kiszámolja az XOR különbséget a bemenet szomszédos bit párjaira: $D(\alpha_1, \alpha_2, ..., \alpha_{d'}) = (\alpha_1 \oplus \alpha_2, \alpha_2 \oplus \alpha_3, ..., \alpha_{d'-1} \oplus \alpha_{d'})$ (3.3. alfejezet)

Irodalomjegyzék

[1] Solomon W. Golomb, Shift Register Sequences, Holden-Day Inc., San Francisco, 1967 (Portions co-authored by L.Welch, R. Goldstein and A. Hales)

[2] Marc Briceno, Ian Goldberg, David Wagner, A pedagogical implementation of the GSM A5/1 and A5/2 "voice privacy" encryption algorithms, http://cryptome.org/gsm-a512.htm, 1999, az internetes cím legutoljára elenőrizve: 2013.05.27.

[3] Ian Goldberg, David Wagner, Lucky Green, The (Real-Time) Cryptanalysis of A5/2, Rump Session of Crypto'99, 1999.

[4] Patrik Ekkdahl, Thomas Johansson, Another Attack on A5/1, IEEE Transactions on Information Theory 49(1), 284-289 oldal, 2003

[5] Alexander Maximov, Thomas Johansson, Steve Babbage, An improved correlation attack on A5/1, proceedings of SAC 2004, LNCS 3357, 1-18 oldal, Springer-Verlag, 2005.

[6] Elad Barkan, Eli Biham, Conditional Estimators: an Effective Attack on A5/1, proceedings of SAC 2005, LNCS 3897, 1-19 oldal, Springer-Verlag, 2006.

[7] Elad Barkan, Cryptanalysis of Ciphers and Protocols, Submitted in partial fulfillment of the Requirements for the Degree of Doctor of Philosophy, Submitted to the Senate of the Technion - Israel Institute of Technology, Adar 5766, 2006