Szegedi Tudományegyetem

Természettudományi és Informatikai Kar
Bolyai Intézet Geometria Tanszék

Szakdolgozat

McEliece-féle titkosító rendszer

McEliece cryptosystem

Készítette: Spir Anita Elvira
Matematika BSc

Témavezető: Dr. Nagy Gábor Péter

2014

Tartalomjegyzék

1. fejezet

Bevezetés

Informatizálódó társadalmunkban egyre fontosabb szerepet kap az adatcsere biztonságának szavatolása. Ezt részben az adatok titkosításával oldhatjuk meg.

A kriptográfia klasszikus értelmben a titkosítás tudománya, titkosítási rendszerek (kriptorendszerek) felépítésével és biztonsági elemzésével (kriptoanalízisével) foglalkozik. Modern értelemben azonban a kriptográfia fogalma magába foglal olyan aktuális témaköröket is, mint a digitális aláírások, hitelesítések, hash függvények stb.

A kriptográfia fontosságát talán az mutatja a leginkább, hogy napjainkban számos vezető informatikai cég újabb és biztonságosabb titkosítási rendszerek tervezésével és ezek lehetséges alkalmazásaival foglalkozik. Ez pedig egyértelműen hozzájárul a modern kriptográfia robbanásszerű fejlődéséhez. A kvantum-számítógép megépítésének lehetősége mára elérhetőnek tűnő célként lebeg az emberiség előtt.

Az elektronikus kereskedelem ma túlnyomórészt az RSA-t, és a hozzá hasonló, diszkrét-logaritmus problémára alapozott protokollokat használja, mint aszimmetrikus titkosításokat. Ezek a protokollok viszont könnyen feltörhetők egy kvantumszámítógéppel, alkalmazva a Shor-algoritmust.

Az úgynevezett kvantumkriptográfia alapelve a kvantummechanika, azon belül is a szuperpozíció-elmélet és a kétvilág-elmélet. Ez azt állítja, ha egy tárgyat elvesztünk szem elől, több állapotban is lehet. A kvantumkriptográfia használható kriptoanalízis felgyorsítására. Ekkor a gépek, mivel az állapot ismeretlen, egyszerre sokkal több állapotot képesek megvizsgálni.

2. fejezet

Bevezetés a lineáris kódokba

Ebben a fejezetben lineáris kódokra vonatkozó alapismereteket foglalok össze a [6] könyv alapján.

A hibajavító kódok elmélete olyan szituációkban jön elő, amikor egy forrástól egy zajos csatornán keresztül információ áramlik a fogadó félhez. A műholdak által készített képek, például, úgy érkeznek a Földre, hogy a küldendő képet lefedik kis négyzetekkel és egy-egy négyzet 0-tól 63-ig skálázott színkódját küldik el, bináris rendszerben. A 64 színkódot felírják mint egy 6 hosszú 0-1-ekből álló sorozatot, de a hőzaj vagy egyéb zavaró tényezők miatt előfordulhat, hogy egy 1-es 0-ásként érkezik meg, vagy fordítva. Ennek kiszűrése és javítása érdekében a küldendő sorozathoz redundáns 0-1-eket csatolnak, hogy itt is, mint egy hosszabb szónál egy elírt betű, könnyen felismerhető és akár javítható legyen. Ez a folyamat az alábbi ábrán látható:

2.1. A kódolás

Tegyük fel, hogy az információ egy Q ábécé segítségével van kódolva, melyben q darab különböző szimbólum van. Az üzenet a forrás kimeneti sorozatának k hosszúságú szegmense: $a = (a_0, a_1, \ldots, a_{k-1})$. A kódoló az üzenethez kódszavakat rendel: egy Q^k-beli elemhez egy Q^n-beli elemet, egyértelmű megfeleltetéssel. Az a üzenethez

az $x = (x_0, x_1, \ldots, x_{n-1})$ kódszót rendeli. A kódszavak halmazát blokk kódoknak, vagy röviden csak kódnak nevezzük.

2.1.1. Definíció. *A C bináris lineáris kód $GF(2^n)$ egy lineáris altere. Ha C k dimenziós, akkor C egy (n,k) kód.*

2.1.2. Definíció. *A C kód G generátormátrixa egy $k \times n$-es mátrix, mely sorait C egy bázisa alkotja.*

Ha G a C kód egy generátormátrixa, akkor $C = \{aG \mid a \in Q^k\}$. G standard alakú, ha $G = (I_k \ P)$, ahol I_k a $k \times k$ identikus mátrix. Ha a G generátormátrix standard alakú, akkor egy kódszó első k szimbóluma az információ hordó, ezt szabadon megválaszthatjuk, a többi szimbólum determinisztikus, ezek az úgynevezett paritásellenőrző szimbólumok.

2.1.3. Definíció. *Legyen C lineáris kód és $G = (I_k \ P)$ a generátormátrixa. Ekkor a $H^T = (-P^T \ I_{n-k})$ neve a C kód paritásellenőrző mátrixa.*

Ekkor $x \in C$ akkor és csak akkor teljesül, ha $xH = 0$, hisz $GH = 0$.

2.1.4. Definíció. *Ha C lineáris kód és H a paritásellenőrző mátrixa, akkor minden $x \in Q^n$ esetén xH neve x szindrómája.*

2.2. A dekódolás

A dekódolóhoz az $r = x + e$ szó érkezik, ahol $r, e \in Q^n$. Az e vektort hibavektornak nevezzük. A dekódolás eredményeként az a üzenet egy a' becslését kapjuk. A dekódolás két lépésben valósul meg: először a vett szó alapján meg kell állapítani a kódszót, majd elvégezni a kódolás inverzét. Ha kódolásnál a G generátormátrix standard alakú volt, akkor gyakorlatilag dekódoláson csak az első lépést értjük, hisz ekkor az üzenet a kódszó szegmense.

2.2.1. Definíció. *Ha $x \in Q^n$ és $y \in Q^n$, akkor x és y Hamming-távolsága*

$$d(x, y) := |\{i | 1 \leq i \leq n, x_i \neq y_i\}|$$

2.2.2. Definíció. *Ha $x \in Q^n$, akkor x Hamming-súlya $w(x) := d(x, 0)$.*

A Hamming-távolság egy metrika Q^n felett. Ha egy olyan csatornát használunk, amely rendelkezik azzal a tulajdonsággal, hogy az i. pozícióban fellépő hiba nem

befolyásolja a többi pozíció hibáját és a hibás pozícióban fellépő szimbólum azonos valószínűséggel lehet a többi q-1 szimbólum, akkor a Hamming-távolság jó mód a kapott üzenet hibatartalmának mérésére.

A továbbiakban legyen C kód egy nem-üres részhalmaza Q^n-nek. Ha $|C| = 1$, akkor triviális kódról beszélünk.

2.2.3. Definíció. *A nemtriviális C kód minimum-távolsága*

$$min\{d(x, y)|x \in C, y \in C, x \neq y\}$$

2.2.4. Definíció. *A nemtriviális C kód minimum-súlya*

$$min\{w(x)|x \in C, x \neq 0\}$$

2.2.5. Tétel. *Egy C lineáris kódnak a minimum súlya egyenlő a minimum távolságával.*

Bizonyítás. $d(x, y) = d(x - y, 0) = w(x - y)$ és ha $x \in C$ és $y \in C$, akkor $x - y \in C$. □

Hogy a vett szó alapján a teljes hibajavítás lehetséges-e, az függ a kód hibajavító képességétől és a pilanatnyi hibamintától.

2.2.6. Definíció. *A C kód hibajavító képessége t, ha képes javítani tetszőleges olyan e hibamintát, melynek Hamming-súlya legfeljebb t, azaz* $w(e) \leq t$.

2.2.7. Tétel. *A t hibát javító C kód d minimális távolságára érvényes a következő összefüggés:* $d \geq 2t + 1$.

Bizonyítás. Legyen x_1 a leadott és r a vett szó, $r = x_1 + e$, ahol $w(e) \leq t$ fennáll, továbbá legyen x_2 egy tetszőleges másik kódszó. Tudjuk, hogy a Hamming-távolság egy metrika, így a háromszögegyenlőtlenség miatt $d(x_2, r) \geq d(x_1, x_2) - d(x_1, r) \geq d - t$, hiszen $d(x_1, x_2) \geq d$ és $d(x_1, r) \leq t$. A t hibát javító kódra tetszőleges $w(e) \leq t$ hibaminta és x_2 kódszó esetén $d(x_1, r) < d(x_2, r)$. Mivel $d(x_2, r) \geq d - t$, és $d(x_2, r) = d - t$, illetve $d(x_1, r) = t$ a legrosszabb esetben, ezért a $d - t > t$ összefüggést kapjuk, ahonnan a $d \geq 2t + 1$ állítás következik. □

Megfordítva, ha $d \geq 2t + 1$, akkor a kód t hibát képes javítani. A fentebb leírt dekódolási szabály szerint meg kell állapítani a vett szóhoz a legközelebb eső kódszót, s azt tekintjük a dekódolás eredményének. A kódszavak nagy száma miatt

azonban nincs mód a vett szónak a kód szavaitól való távolságát megállapítani, s ennek alapján dekódolni.

A dekódolás leggyakoribb módja a szindróma-dekódolás. Az első lépés a vett szó szindrómájának megállapítása, ami alapján második lépésként döntünk a hibamintára. A hibaminta ismeretében korrigálhatjuk a vett szót. A szindrómának a hibamintára történő leképezési módját táblázatba szokás foglalni, az úgynevezett standard elrendezési táblázatba.

2.2.8. Definíció. *Valamely e hibaminta által generált mellékosztály az $e + x$, $x \in C(n, k)$ vektor halmaza.*

Lineáris kódok esetén adott mellékosztály elemeihez azonos szindróma tartozik. Az $e = 0$ zérus hibavektorhoz tartozó mellékosztály a $C(n, k)$ kóddal azonos. Ha egy e hibaminta $e = e' + x$ alakba írható fel, akkor a két hibaminta azonos mellékosztályt generál.

2.2.9. Definíció. *Azonos mellékosztályba tartozó hibaminták közül a legkisebb súlyú hibaminta a mellékosztályvezető.*

A standard elrendezési táblázat a következő struktúrájú:

Szindróma	Mellékosztályvezető	További mellékosztály elemek		
$s^{(0)}$	$e^{(0)}$	$x^{(1)}$		$x^{(2^k-1)}$
$s^{(1)}$	$e^{(1)}$	$x^{(1)} + e^{(1)}$		$x^{(2^k-1)} + e^{(1)}$
$\vdots$	$\vdots$	$\vdots$	$\dots$	$\vdots$
$s^{(2^{n-k}-1)}$	$e^{(2^{n-k}-1)}$	$x^{(1)} + e^{(2^{n-k}-1)}$		$x^{(2^k-1)} + e^{(2^{n-k}-1)}$

A $w(e^{(i+1)}) \geq w(e^{(i)})$, $e^{(0)} = 0$, $i = 0, 1, \dots, 2^{n-k} - 2$ a szokásos sorrend. Könnyen látható, hogy a táblázat elemei különbözők. Egy soron belül ez nyilvánvaló, ezért tegyük fel, hogy $e^{(i)} + x^{(j)} = e^{(k)} + x^{(m)}$, ahol $i \neq k$. Mivel ebből $e^{(i)} = e^{(k)} + x^{(m)} - x^{(j)} = e^{(k)} + x^{(n)}$ következik, ahol $x^{(j)}, x^{(m)}, x^{(n)} \in C(n, k)$, ezért $e^{(i)}$-nek is az $e^{(k)}$ mellékosztályvezetőjű sorban kellene lennie, ami ellentmond a kiindulási feltevésnek.

Az $e^{(i)}, i = 1, 2, \dots, 2^{n-k} - 1$ mellékosztályvezetőket javítható hibamintáknak nevezzük, ugyanis, ha az r vett szó szindrómája $s^{(i)}$, akkor $\hat{x} = r - e^{(i)} = r + e^{(i)}$ kódszóra döntünk. A szindrómadekódolásnak ezt a módját táblázattal történő dekódolásnak nevezzük.

3. fejezet

Néhány bonyolultságelméleti alapfogalom

Ebben a fejezetben bonyolultságelméleti alapfogalmakat tekintek át a [4] könyv alapján.

Egy adott problémára különféle megoldási algoritmusok léteznek. A cél ezen algoritmusok futásidejének matematikai becslése, a bemeneti adatmennyiség függvényében. Így egy hiteles választ kapunk arra, hogy az adott problémára melyik a leggyorsabb megoldási algoritmus.

A futásidő matematikai értelmezéséhez az algoritmust lebontjuk elemi lépésekre, úgynevezett bitműveletekre. Legyen egy bitművelet két bit összeadása a műveletet kísérő esetleges 1-es átvitelével. Egy algoritmus elméleti futásideje legyen az algoritmus által elvégzett bitműveletek száma, ami a bemeneti adat függvénye.

3.0.10. Definíció. *Legyenek* $f, g : \mathbb{N} \to \mathbb{R}$ *függvények úgy, hogy* $f(n), g(n) \geq 0, \forall n \geq n_0$. *Azt mondjuk, hogy* $f(n) = \mathcal{O}(g(n))$ *ha* $\exists c \in \mathbb{R}_+$ *és* $\exists n_1 \in \mathbb{N}$ *úgy, hogy* $0 \leq f(n) \leq cg(n), \forall n \geq n_1$. *Azaz elég nagy értékekre a g egy konstans erejéig nagyobb f-nél, vagy másképpen: a g konstans erejéig gyorsabban növekszik f-nél.*

Ha B az A algoritmus bemeneti adatait jelöli és $n(B)$ ezek összanagyságát bitekben, akkor legyen $f_A(B)$ a megfelelő elméleti futásidő.

3.0.11. Definíció. *Azt mondjuk, hogy egy A algoritmus futásideje (bonyolultsága)* $\mathbb{O}(g(n))$, *ha* $\exists n_0 \in \mathbb{N}$ *és* $c \in \mathbb{R}_+$ *úgy, hogy* $f_A(B) \leq cg(n)$ $\forall B$ *bemenetre, melyre* $n(B) = n \geq n_0$.

3.0.12. Definíció. *Egy n bites bemenetű algoritmus polinomiális, ha a bonyolultsága* $\mathbb{O}(n^k)$. *Speciálisan az algoritmus konstans, lineáris, négyzetes, illetve köbös, ha*

a bonyolultsága rendre $\mathbb{O}(1), \mathbb{O}(n), \mathbb{O}(n^2)$ *illetve* $\mathbb{O}(n^3)$*. Egy algoritmust exponenciálisnak nevezünk, ha bonyolultsága* $\mathbb{O}(2^{poly(n)})$*.*

Az alábbi táblázat összehasonlítja a különböző algoritmusok effektív futásidejét:

Algorimus	Komplexitás	Műveletek száma $n = 10^6$-ra	A műveletek elvégzéséhez szükséges idő (10^6 művelet/s)
konstans	$\mathbb{O}(1)$	1	1 μs
lineáris	$\mathbb{O}(n)$	10^6	1 s
négyzetes	$\mathbb{O}(n^2)$	10^{12}	1,6 nap
köbös	$\mathbb{O}(n^3)$	10^{18}	32000 év
exponenciális	$\mathbb{O}(2^n)$	10^{301030}	$10^{301006}\times$ a világegyetem becsült kora

A fő kérdés, hogy egy feladatnak van-e polinomiális megoldási algoritmusa. A válasz a feladatokat úgynevezett bonyolultsági osztályokba sorolja.

3.0.13. Definíció. *Egy feladat P osztályú, ha van rá polinomiális megoldási algoritmus. Az ilyen feldatokat még könnyűnek (rövid idő alatt megoldhatónak) mondjuk. Egy feladat nehéz, ha nincs P-ben.*

3.0.14. Definíció. *Egy feladat NP osztályú, ha egy általunk megadott megoldási tipp polinomiális idő alatt leellenőrizhető.*

Például ha a feldat $N \in \mathbb{N}$ faktorizálása, akkor egy felbontási tipp egyszerű szorzással leellenőrizhető.

3.0.15. Definíció. *Egy feladat NP-teljes, ha NP osztályú és minden NP-feladat megoldása polinomiális idő alatt visszavezethető ezen feladat megoldására.*

Példák NP-teljes feladatokra:

- az utazóügynök probléma: adva van n város, illetve az útiköltség bármely két város között, és meg kell keresnünk a legolcsóbb utat egy adott városból indulva, amely minden várost pontosan egyszer érint, majd a kiindulási városba ér vissza
- a hátizsák-probléma szerint, ha adott $\nu_1, \ldots, \nu_n$ térfogatú tárgyak és egy V zsáktérfogat, válasszunk ki tárgyakat úgy, hogy össztérfogatuk V legyen

Világos, hogy P $\subseteq$ NP (a tipp P-ben üres). Ha igazolni tudnánk, hogy egy NP-teljes feldat P-ben van, akkor P=NP. Azonban éppen az ellenkezőjét sejtik, így az NP-teljes feladatok a legesélyesebbek arra, hogy számításilag kivitelezhetetlenek legyenek.

4. fejezet

A lineáris kódok dekódolása NP-teljes probléma

Berlekamp és McEliece [1] cikkükben megmutatták, hogy az általános lineáris kódok dekódolása és egy lineáris kód súlyainak meghatározása NP-teljes probléma. Ebben a fejezetben ezt a cikket ismertetem.

Legyen $C(n,k)$ egy n hosszú, k dimenziós, lineáris kód bináris szimmetrikus csatornán. Ha y a vett szó, akkor a szindróma $s = yH$, ahol H az $n \times (n-k)$ paritásellenőrző mátrix és a fogadó fél legjobb becslése a küldött szóról $x = y + z_0$, ahol z_0 az $s = zH$ egyenlőség minimális súlyú megoldása. Egy általános kód esetén, ahhoz, hogy a $zH = s$ egyenlethez megtaláljunk egy minimális súlyú megoldást, végig kell néznünk az összes 2^k darab megoldást. Tehát a triviális algoritmus, mely adott H-hoz és s-hez keresi a legkisebb súlyú megoldást, exponenciális.

A C kód olyan vektorokból áll, melyek teljesítik az $xH = 0$ egyenletet. Sok alkalmazásnál fontos tudni azt, hogy egy adott w súly esetén a C kód tartalmaze w súlyú kódot. Ebben az esetben is, a kérdéssel foglalkozó ismert algoritmusok futásideje exponenciális.

A fent említett problémákat ekvivalens módon megfogalmazhatjuk a következőképpen:

A. Mellékosztály súlya:

   Bemenet: egy A bináris mátrix, egy y bináris vektor és egy w nemnegatív egész

   Tulajdonság: létezik egy x vektor, melynek a Hamming-súlya $\leq w$ úgy, hogy $xA = y$

B. Altér súlya:

Bemenet: egy bináris A mátrix és egy w nemnegatív egész

Tulajdonság: létezik egy x vektor, melynek Hamming-súlya w úgy, hogy $xA = 0$

A következőkben be lesz bizonyítva, hogy valószínűtlen, hogy valaha is lesz ennél alacsonyabb futásidejű algoritmus a problémák megoldására, ugyanis a fent leírt problémák az NP-teljes problémák osztályába tartoznak. A bizonyítás a problémák egy harmadik NP-teljes problémára való visszavezetése.

C. Háromdimenziós párosítás

Bemenet: egy $U \subseteq T \times T \times T$ részhalmaz, ahol T egy véges halmaz

Tulajdonság: létezik egy $W \subseteq U$ halmaz úgy, hogy $|W| = |T|$ és W bármely két elemére igaz, hogy egyik koordinátájukban sem egyeznek meg

Az U részhalmaz elemei rendezett számhármasok a T halmazból. Illusztrációként tekintsük a következő két példát, mindkettőt a $T = \{1, 2, 3, 4\}$ és $|U| = 6$ esetben:

	Első példa	Második példa
1.)	(1,2,1)	(1,2,1)
2.)	(1,3,2)	(1,3,2)
3.)	(2,1,4)	(2,1,4)
4.)	(2,2,3)	(2,2,3)
5.)	(3,1,1)	(3,2,1)
6.)	(4,4,4)	(4,4,4)

Az első példa számhármasainak a halmaza teljesíti a megkívánt tulajdonságot, ugyanis a 2.), 4.), 5.) és 6.) sorszámú számhármasok halmaza megfelelő "párosítás" lesz. A második példa nem teljesíti a tulajdonságot.

A cél érdekében írjuk fel az U halmaz számhármasait, mint egy $|U| \times 3\,|T|$ bináris incidencia mátrixot, amelyben minden sor megfelel egy adott számhármasnak és minden sor 3 súlyú. Minden 1-es feleljen meg az adott számhármas egy elemének. Az első példa incidencia mátrixa a következő:

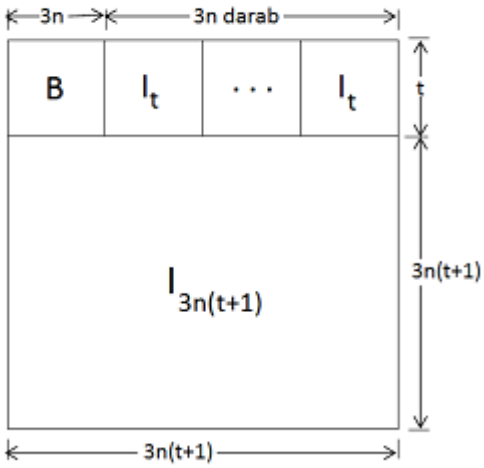
	1	2	3	4	1	2	3	4	1	2	3	4
121:	1	0	0	0	0	1	0	0	1	0	0	0
132:	1	0	0	0	0	0	1	0	0	1	0	0
214:	0	1	0	0	1	0	0	0	0	0	0	1
223:	0	1	0	0	0	1	0	0	0	0	1	0
311:	0	0	1	0	1	0	0	0	1	0	0	0
444:	0	0	0	1	0	0	0	1	0	0	0	1

Tekintve ezt a mátrixot, a háromdimenziós párosítás probléma megfelel $|T|$ darab sor létezésének, melyek kettes számrendszerbeli összege $111\ldots111$.
A C. NP-teljes probléma visszavezetése az A. illetve B. problémára:

C. $\rightarrow$ A. Tegyük fel, hogy van polinomiális algoritmusunk az A. problémára. Legyen adott egy $U \subseteq T \times T \times T$ bemenet a C. problémához és legyen A az $|U| \times 3\,|T|$-s incidencia mátrixa a fenti módon. Futtassuk az A. probléma feltételezett algoritmusát az A, $y = (111\ldots111)$, $w = |T|$ bemenetekkel. Ekkor megkapjuk a választ hogy létezik-e egy legfeljebb w súlyú x vektor, melyre $xA = (111\ldots111)$, azaz ki lehet-e választani $\leq w$ darab sort, melyek összege a csupa 1-esből álló vektor. Fontos észrevétel, hogy w-nél kevesebb sor nem elegendő, hisz minden sorból 3 darab egyest kapunk és $3w$ darabb 1-esre van szükségünk, valamint w darab sor is pontosan akkor elég, ha az 1-esek különböző helyeken vannak, hisz ha két 1-es egymás fölött áll az incidencia mátrixban, akkor az összegük 0.

Ez alapján polinomiális idő alatt megkapjuk a választ, hogy létezik-e megfelelő párosítás. Tehát az A. polinomiális algoritmusának léte maga után vonja C. egy polinomiális algoritmusának létét. Ez azt jelentené, hogy minden NP-teljes problémának van polinomiális algoritmusa. Ez bizonyítja, hogy a mellékosztály súlya nevű probléma is NP-teljes.

C. $\rightarrow$ B. Az ötlet ugyanaz: szerkesszünk egy A mátrixot a B. algoritmusához. Legyen a B mátrix a fent leírt $|t \times 3n|$-es incidecia mátrix és A legyen a következő alakú:



Tegyük fel, hogy B.-nek van egy polinomiális idejű megoldási algoritmusa. Ha ezt az algoritmust futtatjuk az A mátrix és $w = 3n^2 + 4n$ bemenetre, akkor polinomiális idő alatt választ kapunk arra, hogy van-e az eredeti számhármasok halmazán

párosítás.

Bizonyításként legyen x egy vektor mely kielégíti az $xA = 0$ egyenletet. A könnyebb mátrixszorzás érdekében legyen $x = (x_0\ x_1)$, ahol x_0 t hosszú, x_1 pedig $3n(t+1)$ hosszú. Ekkor $(x_0\ x_1)A = (y\ x_0 \dots x_0) + x_1 = 0$, ahol szükségszerűen y vektor hossza $3n$ és $3n$ darab x_0 vektor van. Ebből következik, hogy $x_1 = (y\ x_0 \dots x_0)$, azaz $x = (x_0\ y\ x_0 \dots x_0)$

Ezek alapján, a következő egyenlőséghez jutunk: $|x| = |y| + (3n+1)\,|x_0|$, ahol $|x|$ az x vektor Hamming-súlya. Mivel $0 \leq |y| \leq 3n$, ezért $|x_0|$ és $|y|$ egyértelműen meghatározhatók $|x|$-ből: $|x|$-et osztva $3n+1$-el, $|x_0|$ a hányados és $|y|$ a maradék. Konkrétan, ha $|x| = 3n^2 + 4n$, akkor $|x_0| = n$ és $|y| = 3n$, azaz x_0 megmutatja mely n darab sort választottam ki, hogy megkapjam az $y = (1 \dots 1)$ vektort.

Tehát az A paritásellenőrző mátrixú kód akkor és csak akkor tartalmaz egy $3n^2 + 4n$ súlyú szót, ha a számhármasok halmazában van párosítás. Ez bizonyítja, hogy az altér súlya nevű probléma is NP-teljes.

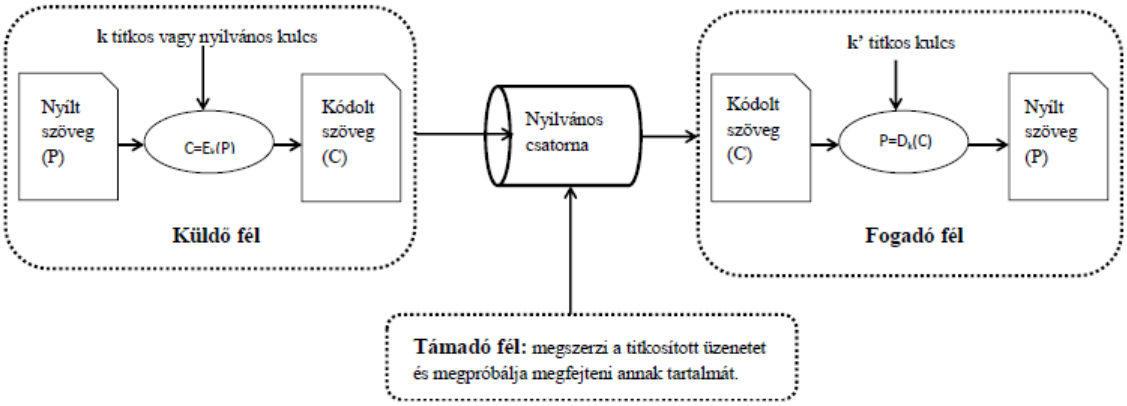
5. fejezet

McEliece-féle titkosító rendszer

Ezt a fejezetet a [4] könyv, illetve a [5] könyv 11. fejezete alapján készítettem.

5.1. Kriptográfiai alapfogalmak

Egy általános titkosítási rendszer az alábbi sémával vázolható:



A séma szerint a küldő fél az eredeti P üzenetet (plain text) a k kulcstól függően E_k titkosítási eljárással kódolja (encryption). A kódolás előtti eredeti üzenetet még nyílt üzenetnek vagy nyílt szövegnek is nevezzük, a titkosítási eljárást pedig kriptorendszernek vagy kódnak. A titkosítási eljárással a küldő fél előállítja a $C = E_k(P)$ tikosított szöveget (cypher text). Ezt egy nyilvános csatornán a fogadó félnek továbbítja, aki a titkosított C üzenetet a k' kulcstól függő $D_{k'}$ dekódoló eljárással (decryption) megfejti, visszakapva az eredeti P üzenetet.

Nyilvánvalóan az $E_k, D_{k'}$ függvények egymás inverzei kell, hogy legyenek, hiszen

$D_{k'}(C) = D_{k'}(E_k(P)) = P$ és fordítva is igaz $E_k(P) = E_k(D_{k'}(C)) = C$. Ugyanakkor fontos, hogy ezek az eljárások polinomiális idő alatt fussanak.

A képletbe belekerül egy harmadik szereplő is, a támadó fél, aki lehallgatja a nyilvános csatornán továbbított üzenetet és megpróbálja megfejteni annak tartalmát.

A kulcsok titkossága nagyon fontos, mert az E kódolási és D dekódolási eljárások nyilvánosak, a kommunikáció titkossága csupán a sokféleképpen megválasztható k és k' kulcsok titkosságától függ. A kulcs titkossága alapján a titkosítási rendszereket két családba soroljuk: titkos kulcsú (vagy szimmetrikus kulcsú) kriptorendszerek és nyilvános kulcsú (vagy aszimmetrikus kulcsú) kriptorendszerek.

Egy kriptorendszer szimmetrikus kulcsú, ha k és k' is titkos. Ebben az esetben $k \approx k'$, vagyis a két kulcs egyenlő vagy az egyik kulcsból polinomiális idő alatt előállítható a másik kulcs.

Egy titkosítási módszer aszimmetrikus kulcsú, ha k nyilvános és k' titkos. Ebben az esetben nyilván $k \not\approx k'$, tehát k ismerete nem vezethet könnyen k' ismeretéhez.

5.2. Bevezetés a Goppa-kódokba

Minden irreducibilis, $GF(2^m)$-beli t-ed fokú polinomhoz tartozik egy bináris Goppa-kód.

5.2.1. Definíció. *Legyen $g(x)$ egy irreducibilis t-ed fokú polinom $GF(2^m)$ felett. Ekkor*

$$\Gamma(g(x), GF(2^m)) = \{(c_w)_{w \in GF(2^m)} \in \{0,1\}^n \mid \sum_{w \in GF(2^m)} \frac{c_w}{x - w} \equiv 0 \, (mod \; g(x))\}$$

halmaz egy bináris Goppa-kódot definiál, melynek hossza $n = 2^m$, dimenziója $k \geq n - tm$ és minimum távolsága $d \geq 2t + 1$.

A bináris Goppa-kódokhoz létezik gyors futásidejű hibajavító algoritmus. Ez az algoritmus minden $\{0,1\}^n$-beli elemet, ami legfeljebb t koordinátában különbözik egy c kódszótól, ebbe a kódszóba képez. Tehát ha elküldünk egy c kódot és a vett r szó a c kódtól kevesebb mint t koordinátában tér el, akkor a fogadó fél képes előállítani az eredeti c kódot a vett r szóból.

A $GF(2^m)$ test felett körülbelül $2^{mt}/t$ darab t-ed fokú irreducibilis polinom van, így egy véletlenszerűen választott $GF(2^m)$-beli t-ed fokú polinom $1/t$ valószínűséggel

lesz irreducibilis. Ezt az állítást a SAGE [3] program segítségével teszteltem. Véletlenszerűen választottam 1000 darab t-ed fokú polinomot a $GF(2^m)$ véges testből és megszámoltam, hogy ebből hány irreducibilis. Az eredmény a következő táblázatban látható:

	$t=1$	2	3	4	5	6	7	8	9	10
$m=1$	510	375	310	236	212	205	139	145	147	140
2	752	469	359	263	214	192	149	136	119	106
3	875	495	334	248	207	166	154	125	121	112
4	938	505	349	284	198	158	166	119	107	86
5	964	525	367	272	206	165	147	129	119	87
6	987	505	336	253	179	168	131	131	96	117
7	992	481	329	254	192	176	161	136	112	100
8	998	495	342	272	185	157	141	134	125	104
9	997	514	333	262	222	171	148	130	99	104
10	998	491	317	252	170	159	125	136	110	91

Mivel létezik gyors algoritmus az irreducibilitás tesztelésére ([5] 241. oldal), ezért ismételt random választással és teszteléssel tudunk keresni egy t-ed fokú, irreducibilis polinomot $GF(2^m)$ felett, persze nagy t esetén ez az algoritmus nem hatékony, vannak ennél jobban használható algoritmusok is.

5.3. A kriptorendszer felállítása

A fenti tényre és a lineáris kódok dekódolásának NP-teljességére alapozva Robert J. McEliece [2] megfogalmazta a McEliece-féle titkosító rendszert.

1. Minden U felhasználó választ egy Goppa-kódot, melynek hossza $n_U = 2^{m_U}$ és amely képes t_U hibát javítani. Ehhez az U felhasználó véletlenszerűen választ egy t_U-ad fokú irreducibilis $p_U(x)$ polinomot $GF(2^{m_U})$ felett és megszerkeszti a $\Gamma(p_U(x), GF(2^{m_U}))$ Goppa-kódhoz tartozó G_U generátormátrixot. G_U mátrix $k_U \times n_U$ méretű.

2. Az U felhasználó véletlenszerűen választ egy $k_U \times k_U$ méretű sűrű, nemszinguláris S_U mátrixot és egy $n_U \times n_U$ méretű P_U permutációs mátrixot $GF(2)$ felett, majd kiszámolja a $k_U \times n_U$ méretű $G_U^\star = S_U G_U P_U$ mátrixot.

3. Az U felhasználó nyílvánosságra hozza a $G_U^\star$ mátrixot és t_U-t, az irreducibilis polinom fokát. Titkosak maradnak a G_U, S_U és P_U mátrixok.

5.4. Kódolás

Tegyük fel, hogy Alice üzenetet akar küldeni Bobnak. Alice megnézi Bob nyilvános kulcsait: $G_B^\star$ mátrixot és t_B-t. Az üzenetét átalakítja egy k_B hosszúságú m bináris kóddá, majd véletlenszerűen választ egy n_B hosszúságú e hibavektort, melynek súlya maximum t_B. Az m üzenetet lekódolva Alice a következőt küldi Bobnak: $r = mG_B^\star + e$.

5.5. Dekódolás

Bob megkapva az r üzenetet, a titkos P_B permutációs mátrixával elvégzi a következő műveletet:

$$rP_B^{-1} = mG_B^\star P_B^{-1} + eP_B^{-1} = mS_B G_B P_B P_B^{-1} + e' = (mS_B)G_B + e'$$

ahol $e' = eP_B^{-1}$ az e egy permutációja, tehát e' súlya szintén kisebb mint t_B. A $\Gamma(p_B(x), GF(2^{m_B}))$ Goppa-kód dekódoló algoritmusával Bob hatékonyan dekódolhatja rP_B^{-1}-et, megtalálja az e' hibát és megkapja mS_B-t. Ezt jobbról megszorozva a titkos S_B mátrix inverzével visszakapja az Alice által küldött eredeti $m \in \{0,1\}^{k_B}$ üzenetet.

6. fejezet

Példa McEliece-féle titkosító rendszerre

Ebben a fejezetben szintén a [5] könyv 11. fejezete volt a kiindulás, de az abban szereplő Mathematica programokat átírtam a nyílt forráskódú SAGE [3] rendszerre. Ezen kívül dolgozatomban külön kitérek a véges test feletti irreducibilis polinomok, valamint a bináris mátrixok véletlen generálásának problémájára.

Legyen $m = 4$, így a Goppa-kód hossza $n = 16$ és legyen $t = 3$, azaz a kód 3 hibát képes javítani. Hozzuk létre a 16 elemű testet és irassuk ki az elemeit.

```
f.<a>=GF(2^m)
for i,x in enumerate(f):
    print i, x.integer_representation(),x
```

Az első oszlop csak sorszámozás, hogy lássuk, hogy megvan mind a 16 elem. A harmadik oszlopban vannak az a generáló elemmel kifejezett elemek. A második oszlop a harmadik oszlopban szereplő elemeknek az egész számú megfelelői.

0	0	0
1	2	a
2	4	a^2
3	8	a^3
4	3	$a+1$
5	6	a^2+a
6	12	a^3+a^2
7	11	a^3+a+1
8	5	a^2+1

9 10 $a^3 + a$
10 7 $a^2 + a + 1$
11 14 $a^3 + a^2 + a$
12 15 $a^3 + a^2 + a + 1$
13 13 $a^3 + a^2 + 1$
14 9 $a^3 + 1$
15 1 1

A Goppa kódhoz szüséges egy 3-ad fokú irreducibilis polinom $GF(16)$ felett. Ezt úgy keressük meg, hogy kiválasztunk egy random elemet a polinogyűrűből, majd leellenőrizzük, hogy irreducibilis-e, ha nem, akkor választunk egy új elemet. Amikor találtunk egy irreducibilis 3-ad fokú polinomot, akkor azt megtartjuk, ez lesz a Goppa kódot generáló polinom.

```
r=PolynomialRing(f,'x')
x=r.gen()
g=r.random_element(t)
counter=0
while not g.is_irreducible():
    g=r.random_element(t)
    counter=counter+1
print counter,g
```

A kiiratásnál az első elem, megmutatja, hogy hányszor kellett újból választanunk egy random elemet, a második magát az irreducibilis polinomot írja ki: 3; $(a+1)*x^3+x^2+(a^3+a^2)*x+a^3+a^2+a+1$

A Goppa kód generálásához szükséges az $x-x_0$ inverzének a megkeresése moduló $g(x)$, ahol $x_0 \in GF(2^m)$.

```
for i,x0 in enumerate(f):
    print i,xgcd(x-x0,g)
```

Az első oszlop ismét csak sorszámozás, a másodikban az első elem a legnagyobb közös osztó, a második kettő pedig az Euklideszi-algoritmusból keletkező tagok.

0 $(1,(a^3+a+1)*x^2+a^3*x+a^3+a,a^3)$
1 $(1,(a^3+a^2+a+1)*x^2+a^3*x+a^3+a,a^2+1)$
2 $(1,(a^2+1)*x^2+a^2*x+a^2,a+1)$
3 $(1,a^2*x^2+(a^3+a+1)*x+a^2,a^3+a^2+1)$
4 $(1,(a^3+a+1)*x^2+(a^2+a)*x,a^3)$
5 $(1,(a^3+a^2+a)*x^2+(a^3+1)*x+a^3+a^2+a,a^3+a+1)$

6 $(1, a^2 * x^2 + a^3 * x + a^3 + 1, a^3 + a^2 + 1)$
7 $(1, (a^3 + a^2 + 1) * x^2 + (a^3 + a^2) * x + a^3 + a^2, a^3 + a)$
8 $(1, (a^2 + a) * x^2 + (a^3 + a^2 + a + 1) * x + a^3 + a^2 + 1, a)$
9 $(1, a^2 * x^2 + (a + 1) * x + a^3 + a^2 + a, a^3 + a^2 + 1)$
10 $(1, a * x^2 + x + a^3 + a^2 + a + 1, a^3 + a^2 + a + 1)$
11 $(1, (a^3 + a^2 + a + 1) * x^2 + a^3 + 1, a^2 + 1)$
12 $(1, (a^2 + a + 1) * x^2 + (a^2 + a + 1) * x + a^2, a^3 + a^2)$
13 $(1, (a^3 + a + 1) * x^2 + (a^3 + a^2 + a) * x, a^3)$
14 $(1, (a^3 + 1) * x^2 + (a^3 + a) * x + a^2 + a + 1, a^2 + a + 1)$
15 $(1, (a + 1) * x^2 + a * x + a^3 + a^2 + a, 1)$

A továbbiakban szükség lesz a $GF(2^m)$ feletti vektortérre. Ezt hozzuk létre és irassuk ki az elemeit.

```
v=f.vector_space()
for i,y in enumerate(v):
    print i,y
```

0 (0, 0, 0, 0)
1 (1, 0, 0, 0)
2 (0, 1, 0, 0)
3 (1, 1, 0, 0)
4 (0, 0, 1, 0)
5 (1, 0, 1, 0)
6 (0, 1, 1, 0)
7 (1, 1, 1, 0)
8 (0, 0, 0, 1)
9 (1, 0, 0, 1)
10 (0, 1, 0, 1)
11 (1, 1, 0, 1)
12 (0, 0, 1, 1)
13 (1, 0, 1, 1)
14 (0, 1, 1, 1)
15 (1, 1, 1, 1)

Valójában csak az $(x - x_0)$ inverzeinek együtthatóira van szükség modulo $g(x)$, ezeket mentsük el egy mátrixba.

```
H1=[xgcd(x-x0,g)[1].coeffs() for x0 in f]
```

$$H1 = \begin{pmatrix} a^3+a & a^3 & a^3+a+1 \\ a^3+a & a^3 & a^3+a^2+a+1 \\ a^2 & a^2 & a^2+1 \\ a^2 & a^3+a+1 & a^2 \\ 0 & a^2+a & a^3+a+1 \\ a^3+a^2+a & a^3+1 & a^3+a^2+a \\ a^3+1 & a^3 & a^2 \\ a^3+a^2 & a^3+a^2 & a^3+a^2+1 \\ a^3+a^2+1 & a^3+a^2+a+1 & a^2+a \\ a^3+a^2+a & a+1 & a^2 \\ a^3+a^2+a+1 & 1 & a \\ a^3+1 & 0 & a^3+a^2+a+1 \\ a^2 & a^2+a+1 & a^2+a+1 \\ 0 & a^3+a^2+a & a^3+a+1 \\ a^2+a+1 & a^3+a & a^3+1 \\ a^3+a^2+a & a & a+1 \end{pmatrix}_{16\times 3}$$

Az együtthatókat a fenti mátrixból a $GF(2^m)$ feletti vektortér segítségével vektorokként mentsük el majd rendezzük mátrixba. Ennek a mátrixnak a transzponálásával megkapjuk a Goppa-kód paritásellenőrző mátrixát.

```
H2=[[v[u.integer_representation()] for u in H1row] for H1row in H1]
H3=[]
for j in range(2^m):
    L=[]
    for i in range(t):
        L.extend(list(H2[j][i]))
    H3.append(L)
H=matrix(H3).transpose()
```

$$H = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \end{pmatrix}_{12\times 16}$$

A paritásellenőrző mátrixból megkapható a Goppa-kód. Ez a kód jelen esetben 16 hosszú, 4 dimenziós és minimum távolsága 7.

```
C=codes.LinearCodeFromCheckMatrix(H)
mdist=C.minimum_distance()
```

A McEliece-féle kriptorendszerhez szükség van a generátor mátrixra, melyet a paritásellenőrző mátrixból kaphatunk meg úgy, hogy megkeressük a mátrix magterét.

```
G1=matrix(transpose(H).kernel().basis())
```

$$G1 = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}_{4\times 16}$$

A kódoláshoz kell egy invertálható 4×4-es mátrix. Ehhez hozzuk létre a megfelelő nagyságú mátrixteret, majd válasszunk egy random mátrixot, ellenőrizzük le, hogy invertálható-e, ha nem akkor újból választunk. Amikor találunk egy invertálható mátrixot, azt megtartjuk. A kiiratásnál az első elem megmutatja, hogy hányszor kellett újból választanunk egy random elemet.

```
mat=MatrixSpace(GF(2),2^m-m*t,2^m-m*t)
smat=mat.random_element()
counter=0
while not smat.is_invertible():
    smat=mat.random_element()
```

```
        counter=counter+1
    counter , matrix(smat)
```

Most 5-ször kellett újból választanunk és a következő mátrixot kaptuk:

$$smat = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \end{pmatrix}_{4\times 4}$$

Szükségünk van még egy 16×16-os permutációs mátrixra, ehhez véletlenszerűen válasszunk egy 16 hosszú permutációt, és alakítsuk mátrixá.

```
perm=Permutations(2^m).random_element()
pmat=perm.to_matrix()
```

$$pmat = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}_{16\times 16}$$

Állítsuk elő a kódoláshoz szükséges mátrixot. Ez a mátrix egyenlő a random invertálható mátrix, a generátormátrix és a permutációs mátrix szorzatával.

```
G=smat*G1*pmat
```

$$G = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \end{pmatrix}_{4\times 16}$$

Válasszuk ki a küldendő üzenetet, ezt is véletlenszerűen, ezért hozzuk létre a $GF(16)$ feletti 4 hosszú vektorokból álló vektorteret és válasszunk innen egy random elemet, ez lesz az üzenet.

```
w=VectorSpace(GF(2),2^m-m*t)
u=w.random_element()
```

$$u = (0, 0, 1, 0)$$

A kódolás során az üzenethez hozzá kell adnunk egy 16 hosszú maximum 5 súlyú hibavektort.

```
err=random_matrix(GF(2),1,2^m,density=mdist/2^m)[0]
```

$$err = (0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0)$$

A küldendő üzenetet megszorozva a fent létrehozott G mátrixal, elküldjük a címzettnek, közben hozzáadódik a hiba, és ezt kapja meg a címzett.

```
u*G+err
```

$$(0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0, 1, 0)$$

6.1. $n \times n$-es nem-elfajuló mátrix

Nézzük meg, hogy a következő ciklus, mely addig választ egy tetszőleges $n \times n$-es mátrixot GF(2)-felett, míg nem-elfajuló mátrixot nem kap, milyen gyorsan ad megoldást.

```
mat=MatrixSpace(GF(2),n,n)
smat=mat.random_element()
while not smat.is_invertible():
    smat=mat.random_element()
```

Ehhez nézzük meg, hogy mi annak a valószínűsége, hogy egy tetszőleges M $n \times n$-es $GF(q)$ feletti mátrix determinánsa nem nulla. A $GF(q)$ feletti, nem-elfajuló mátrixok száma: $(q^n-1)(q^n-q)(q^n-q^2)\dots(q^n-q^{n-1})$, hiszen az első sort q^n-1 féle képpen választhatom meg (minden egyes elemet q féle képpen, de ezek közül le kell

vonnom a csupa 0-ból álló sort). A második sort nem lehet az első sor skalárszorosa, így q^n-ből le kell vonnom q-t. A harmadik sor nem lehet az első két sor lineáris kombinációja, így q^n-ből le kell vonnom q^2-et és így tovább. Mivel $GF(2)$ felett összesen $q^n q^n$ mátrix van, így

$$P(\det(M) \neq 0) = \frac{(q^n-1)(q^n-q)\dots(q^n-q^{n-1})}{q^{n^2}} = \left(1-\frac{1}{q^n}\right)\left(1-\frac{1}{q^{n-1}}\right)\dots\left(1-\frac{1}{q}\right)$$

Ez a produktum kis n értékekre kiszámolható. Ha n nagy, tart a ∞-be, akkor ez a valószínűség megegyezik $\Phi(1/q)$-val, ahol Φ az Euler-függvény [8]:

$$\Phi(p) = \prod_{k=1}^{\infty}(1-p^k)$$

Vegyük a függvény logaritmusát:

$$\ln(\Phi(p)) = \sum_{k=1}^{\infty}\ln(1-p^k)$$

Az $\ln(1-x)$, $x=0$ körüli Taylor-sora: $-\sum_{n=1}^{\infty}\frac{x^n}{n}$,ami abszolút konvergens az $R=1$ konvergencia sugáron belül. Így

$$\ln(\Phi(p)) = -\sum_{k=1}^{\infty}\sum_{n=1}^{\infty}\frac{p^{k^n}}{n}$$

sor $|p|<1$ esetén abszolút konvergens, ezért a szummákat felcserélhetjük, és a második szummából kiemelve $\frac{1}{n}$-t a belső szumma: $\sum_{k=1}^{\infty}p^{n^k}$ egy p^n kvóciensű mértani sorozat, mely egyenlő $\frac{1}{1-p^n}-1=\frac{p^n}{1-p^n}$. Tehát

$$\ln(\Phi(p)) = -\sum_{n=1}^{\infty}\frac{1}{n}\frac{p^n}{1-p^n}$$

A fenti sor a Lambert-sorok [7] egy speciális esete.

$$-\sum_{n=1}^{\infty}\frac{1}{n}\frac{p^n}{1-p^n} = -\sum_{n=1}^{\infty}\frac{p^n}{n}\sum_{k=1}^{\infty}p^{(k-1)n} = -\sum_{n,k=1}^{\infty}\frac{p^{kn}}{n} = -\sum_{d|m,m=1}^{\infty}\frac{p^m}{d} = -\sum_{m=1}^{\infty}p^m\sum_{d|m}\frac{1}{d}$$

Mivel $-\sum_{d|m}\frac{1}{d} = -\sum_{d|m}\frac{d}{m}$, igaz a kövtekező:

$$\ln(\Phi(p)) = -\sum_{n=1}^{\infty}\frac{1}{n}\frac{p^n}{1-p^n} = -\sum_{m=1}^{\infty}\frac{p^m}{m}\sum_{d|m}d$$

Ezt a szummát csökkentjük $0 < p < 1$ esetén, ha $\sum_{d|m} d$ helyett $\sum_{d=1}^{m} d$ írunk:

$$\ln(\Phi(p)) = -\sum_{m=1}^{\infty} \frac{p^m}{m} \sum_{d|m} d \geq -\sum_{m=1}^{\infty} \frac{p^m}{m} \sum_{d=1}^{m} d = -\sum_{m=1}^{\infty} \frac{p^m}{m} \frac{m(m+1)}{2} = -\frac{1}{2} \sum_{m=1}^{\infty} (m+1) p^m$$

$$= -\frac{1}{2}(2q + 3p^2 + 4p^3 + \dots) = -\frac{1}{2}(p^2 + p^3 + p^4 + \dots)' = -\frac{1}{2}\left(\sum_{m=2}^{\infty} p^m\right)'$$

$$= -\frac{1}{2}\left(\frac{1}{1-p} - 1 - p\right)' = -\frac{1}{2}\left(\frac{p^2}{1-p}\right)' = -\frac{1}{2}\frac{2p - p^2}{(1-p)^2} = \frac{1}{2} - \frac{1}{2(1-p)^2}$$

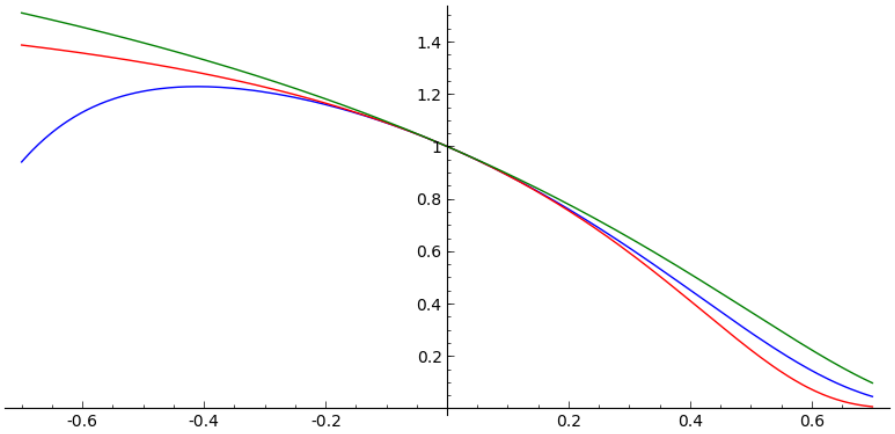
Tehát az $\frac{1}{2} - \frac{1}{2(1-p)^2}$ egy jó alsó korlát $\ln(\Phi(p))$-re.

$\ln(\Phi(p))$ felülről is korlátos:

$$\ln(\Phi(p)) = -\sum_{m=1}^{\infty} \frac{p^m}{m} \sum_{d|m} d \leq -\sum_{m=1}^{\infty} \frac{p^m}{m} m = -\sum_{m=1}^{\infty} p^m = -\left(\frac{1}{1-p} - 1\right) = \frac{p}{p-1}$$

Tehát $0 < p < 1$ esetén:

$$\frac{1}{2} - \frac{1}{2(1-p)^2} \leq \ln \Phi(p) \leq \frac{p}{p-1}$$



6.1. ábra. Korlátok $\Phi(p)$-re $0 < p < 1$ esetén

A $p = 1/q = 1/2$ esetén:

$$-\frac{3}{2} \leq \ln(\Phi(p)) \leq -1$$

$$0,2231 \approx \mathrm{e}^{-\frac{3}{2}} \leq \Phi(p) = P(\det(M) \neq 0) \leq \mathrm{e}^{-1} \approx 0,3678$$

Ezeket a korlátokat a SAGE [3] program segítségével ellenőrizzük le:

```
for n in [1..39]:
    m=MatrixSpace(GF(2),n,n)
    counter=0
    for j in [1..1000]:
        ma=m.random_element()
        if ma.is_invertible():
            counter=counter+1
    print(n, counter)
```

Véletlenszerűen válasszunk 1000 darab $GF(2)$ feletti $n \times n$-es mátrixot és számoljuk meg, hogy ezek közül hány nem-elfajuló. Az eredményt a következő táblázat tartalmazza:

$n=$	1	2	3	4	5	6	7	8	9	10	11	12	13
	521	356	324	309	264	296	259	282	265	301	289	306	308
$n=$	14	15	16	17	18	19	20	21	22	23	24	25	26
	289	273	258	299	297	283	303	293	281	260	306	286	305
$n=$	27	28	29	30	31	32	33	34	35	36	37	38	39
	292	301	282	279	300	279	288	280	291	293	308	279	276

Mivel a $P(\det(M))$ a fent leírt tartományban van, a nem-elfajuló mátrixok keresésére írt eljárásunk várhatóan néhány lépésen belül eredményez egy nem 0 determinánsú $n \times n$-es $GF(2)$ feletti mátrixot.

7. fejezet

Köszönetnyilvánítás

Ezúton szeretnék köszönetet mondani *Dr. Nagy Gábor Péternek*, amiért a szakdolgozatom témaválasztásában, illetve kidolgozásában a kezdetektől fogva segített tanácsaival, útmutatásával, továbbá hogy erre a projektre áldozta az idejét. Segítségével a matematika egy számomra új és érdekes világával ismerkedhettem meg, valamint a SAGE nevű program alapjaival.

Tiszta szívvel köszönöm *szüleimnek* a sok gondoskodást, szeretetet és türelmet ami elkísért a tanulmányaim során. Köszönöm a barátaimnak és szaktársaimnak, hogy együtt éltük át az egyetem jó, bár néha nehéz napjait, valamint köszönöm *Eriknek* a türelmét, segítőkészségét és a pozitív hozzáállását

Végül de nem utolsó sorban szeretném hálásan megköszönni a *Bolyai Intézet* valamennyi oktatójának, dolgozójának azt a lelkiismeretes munkáját, amivel a hallgatók képzését, tanulását és a versenyképes tudás megszerzését támogatják

Nyilatkozat

Alulírott Spir Anita Elvira kijelentem, hogy a szakdolgozatban foglaltak saját munkám eredményei, és csak a hivatkozott forrásokat (szakirodalom, eszközök, stb.) használtam fel. Tudomásul veszem, hogy szakdolgozatomat a Szegedi Tudományegyetem könyvtárában a kölcsönözhető könyvek között helyezik el, és az interneten is nyilvánosságra hozhatják.

Szeged, 2014.05.10

Spir Anita Elvira

Irodalomjegyzék

[1] Elwyn R. Berlekamp, Robert J. McEliece and Henk C.A. van Tilborg, 1978, *On the Inherent Intractability of Certain Coding Problems*, IEEE Transactions on Information Theory, IT-24, pp. 384-386.

[2] R. J. McEliece (January and February 1978). *"A Public-Key Cryptosystem Based On Algebraic Coding Theory"*. DSN Progress Report. 42-44: 114.

[3] W. A. Stein et al., *Sage Mathematics Software (Version 6.1.1)*, The Sage Development Team, 2014, `http://www.sagemath.org`.

[4] Szántó Csaba, Suteu Szöllősi István, 2009, *Kriptográfia*, Kolozsvári Egyetemi Kiadó

[5] Henk C.A. van Tilborg, 2000, *Fundamentals of cryptology*, Kluwer Academic Publishers

[6] Dr. Vajda István, 1982, *Hibajavító kódolás és műszaki alkalmazásai*, Budapesti Műszaki Egyetem

[7] Wikipedia. Lambert series — Wikipedia, The Free Encyclopedia. `http://en.wikipedia.org/w/index.php?title=Lambert_series&oldid=574206576`, [Online; accessed 22-April-2014]

[8] Wikipedia. Euler function — Wikipedia, The Free Encyclopedia. `http://en.wikipedia.org/w/index.php?title=Euler_function&oldid=579547366`, [Online; accessed 22-April-2014]

.