Szegedi Tudományegyetem

Bolyai Intézet

Diplomamunka

Konkatenált Reed-Solomon kódok optimális paraméterei

Készítette: **Judák Regina**
alkalmazott matematikus MSc

Témavezető: **Dr. Nagy Gábor Péter**
egyetemi docens

2016

Tartalomjegyzék

Bevezetés

Dolgozatomban a következő feladattal foglalkoztam: $N = 3000$ bit hosszúságú csomagokat szeretnénk átküldeni egy $p = 0.1$ paraméterű bináris szimmetrikus csatornán keresztül. Olyan lineáris kódokat keresünk, melyek információs rátája 0.3 felett van és a csomagok küldése során a csomaghiba arány minimális.

Egy rövid kódoláselméleti bevezető után részletesebben is ismertetem majd a feladatot, melynek megoldása érdekében egy konkatenált kódot vizsgáltam, ahol a külső kód egy Reed-Solomon kód, míg a belső kód egy bináris lineáris kód. A belső kód egy bizonyos küszöbértéktől függően, hibajelzésre vagy hibajavításra alkalmazható. Hibajelzés esetén úgynevezett törlések keletkeznek. A törléseket és esetleges további hibákat tartalmazó üzenetet ezután a külső, Reed-Solomon kód dekódolja. A 3. fejezetben ismertetem, hogy hogyan próbáltam meg megoldani a feladatot.

Az előbbi módszert implementáltam is a SageMath matematikai szoftver segítségével, illetve egyéb számolások elvégzésére is ezt a szoftvert használtam. Az implementációt felhasználva szimulációkat készítettem különböző paraméterek esetén, így a számolásokon túl azt is láthatjuk, hogy a valóságban mennyire jól működik a kódolás. Az utolsó fejezetben összefoglaltam a számolások és szimulációk eredményeit, az implementációt pedig a függelékben helyeztem el.

Dave Forney [1] foglalkozott először konkatenált kódokkal. Az általa vizsgált konkatenált kód külső kódja szintén Reed-Solomon kód volt, belső kódnak pedig BCH kódot választott. Ennek a kódnak az optimális paramétereit határozta meg különböző kritériumoknak megfelelően, melyek eltérnek a dolgozatomban tekintett követelményektől. A konkatenált kódot vizsgálta abban az esetben is, amikor a belső kód törléseket is csinálhat. Forney belső kódja egy fix küszöbértéktől függően törölt, vagy javított, míg dolgozatomban ez a küszöbérték több értéket is felvehetett, és ennek az optimális értékét is meghatároztam.

1. fejezet

Alapfogalmak

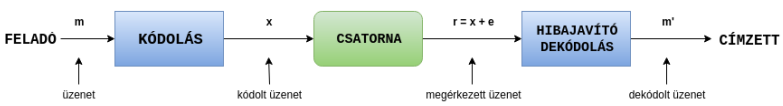
Ebben a fejezetben összefoglaljuk azokat a kódoláselméleti alapfogalmakat, melyekre a dolgozat további részében szükségünk lesz majd.

1.1. Digitális kommunikációs modell

Az 1.1-es ábrán a digitális kommunikációs rendszer alapvető struktúrája látható. A feladó egy zajos csatornán keresztül szeretné eljuttatni üzenetét a címzetthez. Ennek érdekében a továbbítandó $\mathbf{m}$ üzenetet először kódoljuk, majd a kódolt üzenetet küldjük át a csatornán keresztül a címzetthez. A csatornában a zaj hatására hibák keletkezhetnek, vagyis az eredetileg elküldött $\mathbf{x}$ kódolt üzenet helyett egy $\mathbf{r} = \mathbf{x} + \mathbf{e}$ üzenet érkezik meg a címzetthez. Az így megkapott $\mathbf{r}$ üzenet segítségével, először megpróbáljuk kijavítani a csatornában keletkezett hibákat. Ha a hibajavítás megtörtént, akkor már csak a dekódolást kell végrehajtanunk, hogy megkapjuk az eredeti üzenetet.

A dolgozatban bináris szimmetrikus csatornával fogunk dolgozni.

1.1.1. Definíció. Legyen $p \in (0, \frac{1}{2})$ rögzített. Ekkor *p-paraméterű bináris szimmetrikus csatornának* nevezünk egy csatornát, mely a bináris üzeneteket bitenként továbbítja és minden bitet egymástól függetlenül p valószínűséggel helytelenül, míg $1 - p$ valószínűséggel helyesen továbbít. A p paramétert *bithiba aránynak* nevezzük.



1.1. ábra.

Az angol elnevezésekből eredően a bináris szimmetrikus csatornát (binary simmetric channel) röviden *BSC*-nek, a bithiba arányt (bit error ratio) pedig *BER*-nek nevezzük.

A dekódolásnál *maximum likelihood dekódolást* fogunk alkalmazni, vagyis úgy szeretnénk dekódolni az üzeneteket, hogy minimalizáljuk annak a valószínűségét, hogy a dekódolás során rossz kódszóra dekódolunk. Tehát, ha feltesszük, hogy a csatornán az $\mathbf{x}$ kódszót küldtük át és a megérkezett üzenetet $\hat{\mathbf{x}}$-ra dekódoltuk, akkor a

$$p_{hiba} = \mathbf{P}(\hat{\mathbf{x}} \neq \mathbf{x} \mid \mathbf{x})$$

valószínűséget szeretnénk minimalizálni.

1.2. Blokk kódok

A továbbiakban feltesszük, hogy az információt egy olyan Q *ábécé* segítségével kódoljuk, mely q különböző szimbólumot tartalmaz. Ha $C \subset Q^n$ egy nem üres részhalmaz, akkor C-t *kódnak* nevezzük. Ha $q = 2$, akkor *bináris kódról* beszélünk. Egy kódot *blokk kódnak* nevezünk, ha a kódolt információt n hosszú blokkokra tudjuk osztani, úgy hogy ezen blokkok egymástól függetlenül dekódolhatóak. A blokkokat *kódszavaknak* nevezzük, az előbbi n számot pedig, mely megadja, hogy egy blokk milyen hosszú, a *kódszavak hosszának* hívjuk.

1.2.1. Definíció. Ha $\mathbf{x} = (x_0, x_1, ..., x_{n-1}) \in Q^n,\ \mathbf{y} = (y_0, y_1, ..., y_{n-1}) \in Q^n$, akkor az $\mathbf{x}$ és $\mathbf{y}$ kódszavak *Hamming-távolságán* (vagy röviden *távolságán*) a következőt értjük:

$$d(\mathbf{x}, \mathbf{y}) := |\{\ i \mid 0 \leq i \leq n-1,\ x_i \neq y_i\}|,$$

az $\mathbf{x}$ kódszó *súlyának* pedig a $w(\mathbf{x}) := d(\mathbf{x}, \mathbf{0})$ távolságot nevezzük.

Két kódszó távolsága megadja azon pozíciók számát, ahol a kódszavak eltérnek egymástól és így mérhető, hogy a kódszavak mennyire vannak közel egymáshoz. Mint azt már említettük maximum likelihood dekódolást fogunk alkalmazni és BSC esetén akkor lesz a legkisebb annak a valószínűsége, hogy rossz kódszóra dekódolunk, ha az $\mathbf{r}$ üzenetet arra az $\mathbf{x}$ kódszóra dekódoljuk, mely a legkisebb távolságra van $\mathbf{r}$-től. Ezt a módszert *minimum távolság dekódolásnak* nevezzük.

1.2.2. Definíció. Egy $C \subset Q^n$ kód *minimális távolságának* a

$$d(C) := min\left\{d\left(\mathbf{x}, \mathbf{y}\right)\ \mid\ \mathbf{x}, \mathbf{y} \in C,\ \mathbf{x} \neq \mathbf{y}\right\}$$

értéket nevezzük, a kód *minimális súlya* pedig a következő:

$$w(C) := min\left\{w\left(\mathbf{x}\right) \;\mid\; \mathbf{x} \in C,\ \mathbf{x} \neq \mathbf{0}\right\}.$$

1.2.3. Definíció. Legyen

$$\mathbf{B}_r(x) := \left\{\mathbf{y} \in Q^n \mid d(\mathbf{x},\mathbf{y}) \leq r\right\},$$

azaz $\mathbf{B}_r(x)$ egy $\mathbf{x}$ középpontú, r sugarú gömb.

Ha $C \subset Q^n$, $d = d(C)$ és t a legnagyobb olyan egész szám, melyre a $\mathbf{B}_t(c)$ $(\mathbf{c} \in C)$ gömbök diszjunktak, akkor

$$t = \Big\lfloor \frac{d-1}{2} \Big\rfloor.$$

Legyen

$$\rho(C) := max\left\{min\left\{d(\mathbf{x},\mathbf{c}) \mid \mathbf{c} \in C\right\} \mid \mathbf{x} \in Q^n\right\}.$$

Ekkor $\rho = \rho(C)$ az a legkisebb érték, mely esetén a $\mathbf{B}_\rho(\mathbf{c})$ $(\mathbf{c} \in C)$ gömbök lefedik az egész Q^n halmazt. Ha az előbbi t és ρ számok megegyeznek, akkor a kódot *perfekt kódnak* nevezzük, vagyis ha a kód $d = 2t+1$ minimum távolságú és $\forall\ \mathbf{x} \in Q^n$ esetén egyértelműen létezik egy olyan kódszó, mely legfeljebb t távolságra van $\mathbf{x}$-től. Világos, hogy ha a minimum távolság $2t+1$, akkor a kód *t-hibajavító.*

1.3. Lineáris kódok

A továbbiakban $Q = \mathbb{F}_q$ (ahol q egy prímhatvány), azaz $Q^n = \mathbb{F}_q^n$ egy n-dimenziós vektortér.

1.3.1. Definíció. Ha C egy k dimenziós lineáris altere az $\mathbb{F}_q^n$ vektortérnek, akkor C-t *lineáris (n,k)-kódnak* nevezzük.

1.3.2. Definíció. Egy $k \times n$-es G mátrix a C lineáris (n,k)-kód *generátor mátrixa*, ha a sorai C egy bázisát alkotják. Ha $G = (I_k \mid A_{k,n-k})$ alakú, ahol I_k a $k \times k$-s egységmátrix, akkor G-t *szisztematikus generátormátrixnak* nevezzük.

Ha G a kód egy generátormátrixa, akkor egy $\mathbf{x} \in \mathbb{F}_q^k$ vektort úgy tudunk kódolni, hogy megszorozzuk a G mátrixszal:

$$\mathbf{c} = \mathbf{x}\,G,$$

azaz

$$C = \left\{ \mathbf{x}\, G \mid \mathbf{x} \in \mathbb{F}_q^k \right\}.$$

Ha $G = (I_k \mid A_{k,n-k})$ a kód szisztematikus generátormátrixa, akkor

$$\mathbf{c} = \mathbf{x}\, G = (\mathbf{x} \mid \mathbf{x}\, A_{k,n-k})$$

és ekkor a kódszavak első k szimbólumát *információs szimbólumoknak*, a maradék $n-k$ szimbólumot pedig *paritás ellenőrző szimbólumoknak* nevezzük.

1.3.3. Definíció. A C kód (*információs*) *rátájának* nevezzük az alábbi számot:

$$\mathbf{R}(\mathrm{C}) := \frac{\log_q \mid C \mid}{n} = \frac{k}{n}.$$

1.3.4. Tétel. Ha C egy lineáris kód, akkor $d(C) = w(C)$.

1.3.5. Tétel (Singleton-korlát)**.** Ha C egy lineáris (n,k)-kód, $d = d(C)$ minimum távolsággal, akkor

$$d \leq n - k + 1.$$

1.3.6. Definíció. Ha egy C kód minimum távolsága eléri a Singleton-korlátot, azaz

$$d(C) = n - k + 1,$$

akkor a kódot *MDS-kódnak* (maximum distance separable) nevezzük.

A legfontosabb kód paraméterek a ráta és a minimum távolság. Ezek segítségével tudjuk értékelni a kódunk hatékonyságát, a hibajelző és a hibajavító képességét. A dolgozat későbbi részében ezen paraméterek segítségével próbálunk majd meg minél hatékonyabb kódokat találni.

A lineáris kódok dekódolásához szükségünk lesz még pár további definícióra.

1.3.7. Definíció. Legyen C egy lineáris (n,k)-kód, $G = (I_k \mid A_{k,n-k})$ pedig a kód generátor mátrixa. Az alábbi $(n-k) \times n$-es mátrixot a kód *paritás ellenőrző mátrixának* nevezzük:

$$H := (-A_{k,n-k}^T \mid I_{n-k}).$$

A generátor mátrix és a paritás ellenőrző mátrix ortogonálisak:

$$G\, H^T = (I_k \mid A_{k,n-k})\, (-A_{k,n-k}^T \mid I_{n-k})^T = -A_{k,n-k} + A_{k,n-k} = \mathbf{0}_{k,n-k}.$$

Mivel minden $\mathbf{c} \in C$ kódvektor felírható $\mathbf{c} = \mathbf{x}\,G = (\mathbf{x} \mid \mathbf{x}\,A_{k,n-k})$ alakban valamely $\mathbf{x} \in \mathbb{F}_q^k$ esetén, így adódik a következő:

$$\mathbf{c}\,H^T = (\mathbf{x} \mid \mathbf{x}\,A_{k,n-k})\,(-A_{k,n-k}^T \mid I_{n-k})^T = -\mathbf{x}\,A_{k,n-k} + \mathbf{x}\,A_{k,n-k} = \mathbf{0}.$$

Ezzel ekvivalens, hogy ha $\mathbf{r}\,H^T \neq \mathbf{0}$, akkor $\mathbf{r} \notin C$ és fennáll az alábbi:

$$\mathbf{r}\,H^T = \mathbf{0} \Leftrightarrow \mathbf{r} \in C,$$

azaz adott $n-k$ *paritás ellenőrző egyenlet*, melyet minden kódszónak teljesítenie kell.

Legyen $\mathbf{c} \in C$ az a kódvektor, melyet a csatornán továbbítottak, $\mathbf{r} \in \mathbb{F}_q^n$ pedig az a vektor, amely a továbbítás után megérkezett. Ekkor

$$\mathbf{r} = \mathbf{c} + \mathbf{e},$$

ahol $\mathbf{e} = (e_0, e_1, ..., e_{n-1}) \in \mathbb{F}_q^n$ a *hibavektor*, mely a csatornán való továbbítás közben keletkezett. Ha $e_i = 0$, akkor az i. pozíción nem keletkezett hiba, azonban, ha $e_i \neq 0$, akkor az i. pozíción hiba keletkezett a csatornában.

A megkapott üzenetet a paritás ellenőrző mátrix segítségével fogjuk tudni dekódolni. Minden $\mathbf{r} \in \mathbb{F}_q^n$ vektor esetén meg tudjuk határozni az

$$\mathbf{s} := \mathbf{r}\,H^T \in \mathbb{F}_q^{n-k}$$

vektort, melyet az $\mathbf{r}$ vektor *szindrómájának* nevezünk. A szindróma csupán a hibavektortól függ:

$$\mathbf{s} = \mathbf{r}\,H^T = (\mathbf{c} + \mathbf{e})\;H^T\; = \mathbf{c}\,H^T + \mathbf{e}\,H^T = \mathbf{e}\,H^T,$$

hiszen $\mathbf{c}\,H^T = 0$, mert $\mathbf{c} \in C$.

Ha az $\mathbf{r}$ vektorhoz tartozó szindróma nulla, akkor $\mathbf{r}$ egy kódvektor. Ha $\mathbf{e} = \mathbf{0}$, akkor $\mathbf{r} = \mathbf{c}$, azaz a továbbítás során nem keletkezett hiba így nincs szükségünk hibajavításra, csak dekódolnunk kell az $\mathbf{r}$ vektort, hogy megkapjuk az eredetileg küldött üzenetet. Ha $\mathbf{e} \neq \mathbf{0}$ és $\mathbf{e}\,H^T = \mathbf{0}$ mégis teljesül, akkor a hibát nem tudjuk jelezni.

Ha az $\mathbf{r}$ vektorhoz tartozó szindróma nem nulla, akkor a csatornában hiba keletkezett, melyet jelezni tudunk. Ebben az esetben szükségünk van az $\mathbf{r}$ vektor hibajavító dekódolására, azaz először meg kell találnunk azt a kódszót, melyet eredetileg

küldtek. Ehhez meg kell határoznunk az **e** hibavektort. Az **r** vektor ismeretében ki tudjuk számolni a hozzá tartozó szindrómát, majd az

$$\mathbf{s} = \mathbf{e}\, H^T$$

$n - k$ egyenletből álló egyenletrendszer segítségével meg tudjuk határozni a hibavektort.

Mivel C részcsoportja $\mathbb{F}_q^n$-nek, így vehetjük a C szerinti mellékosztályozását. Két vektor akkor és csak akkor lesz ugyanabban a mellékosztályban, ha a hozzájuk tartozó szindrómák megegyeznek, hiszen

$$\mathbf{x}\, H^T = \mathbf{y}\, H^T \Longleftrightarrow \mathbf{x} - \mathbf{y} \in C.$$

Így, ha a hozzánk megérkezett **r** vektorhoz az **e** hibavektor tartozik, akkor az előbbiek értelmében **r** és **e** azonos mellékosztályban találhatóak. Ahhoz, hogy az **e** hibavektort meghatározzuk, a maximum likelihood dekódolás miatt elegendő a mellékosztályból kiválasztanunk a legkisebb súlyú vektort.

Ha ilyen módon meghatároztuk az **e** vektort, akkor az **r** vektort az

$$\mathbf{r}' = (r'_0, r'_1, ..., r'_{n-1}) = \mathbf{r} - \mathbf{e} \in C$$

vektorra javítjuk, majd $\mathbf{r}'$-t az

$$\mathbf{x}' = (r'_0, r'_1, ..., r'_{k-1}) \in \mathbb{F}_q^k$$

vektorra dekódoljuk.

2. fejezet

A probléma ismertetése

A probléma ismertetése előtt még szükségünk van pár további definícióra. A továbbiakban legyen C egy lineáris (n,k)-kód.

2.0.8. Definíció. Legyen $\mathbf{w} \in C$ és jelölje $\mathbf{P}_{C,w}$ annak a valószínűségét, hogy a $\mathbf{w}$ kódszót helytelenül dekódoljuk:

$$\mathbf{P}_{C,w} = \Pr(\mathbf{z} \in C \setminus \{\mathbf{w}\} \text{ kódszóra dekódolunk} \mid \mathbf{w} \text{ kódszót küldtük}).$$

Legyen továbbá $\mathbf{P}_C$ annak a valószínűsége, hogy a csatornán megérkezett üzenetet helytelenül dekódoljuk:

$$\mathbf{P}_C = \frac{1}{|C|} \sum_{w \in C} \mathbf{P}_{C,w}$$

2.0.9. Definíció. Egy p paraméterű BSC esetén az alábbi függvényt *Shannon-függvénynek*, vagy *entrópia függvénynek* nevezzük

$$h(p) = -p \log_2 p - (1-p) \log_2 (1-p), \qquad (0 \leq p \leq 1).$$

2.0.10. Tétel (Shannon, 1948)**.** Legyen $\epsilon > 0$ és $0 < R < 1 - h(p)$ rögzített. Ekkor elég nagy n esetén létezik olyan C bináris lineáris (n, k)-kód, melyre teljesül, hogy $\mathbf{R}(\mathrm{C}) = \frac{k}{n} \geq R$ úgy, hogy $\mathbf{P}_C < \epsilon$. Továbbá, ha $1 - h(p) < R$, akkor nem létezik ilyen kód.

Tehát Shannon-tételéből adódik, hogy adott p paraméterű BSC esetén létezik olyan kód, mely hibázásának a valószínűsége egészen kicsi és az ilyen kódok rátájára egy felső korlát is adódik az entrópia függvény segítségével.

2.0.11. Definíció. A *csomaghiba arány* (PER - Packet Error Rate) a hibásan átvitt csomagok száma osztva az összes átvitt csomag számával.

2.0.12. Megjegyzés. Egy csomagot akkor tekintünk hibásnak, ha legalább egy bit hibás benne.

A dolgozat további részében **PER** fogja jelölni a $\mathbf{P}_C$ valószínűséget, azaz annak a valószínűségét, hogy egy kódszó továbbítása során hiba keletkezett, míg $\mathbf{PER_N}$ fogja jelölni annak a valószínűségét, hogy egy N bit hosszúságú üzenet továbbítása során keletkezett hiba.

Egy lineáris (n, k)-kód esetében, ha ismerjük a **PER** értékét, akkor a $\mathbf{PER_N}$ valószínűséget az alábbi összefüggéssel tudjuk kiszámolni:

$$\mathbf{PER_N} = 1 - (1 - \mathbf{PER})^{\lceil \frac{N}{k} \rceil}.$$

A dolgozat írása közben a következő problémával foglalkoztam: egy $p = 0.1$ paraméterű bináris szimmetrikus csatornán keresztül szeretnénk átküldeni $N = 3000$ bit hosszúságú csomagokat. Ehhez olyan C bináris lineáris kódokat szeretnénk találni, melyekre teljesül hogy $\mathbf{R}(\mathrm{C}) > 0.3$ és a csomaghiba valószínűség, $\mathbf{PER_{3000}}$ a lehető legkisebb. A **PER** és $\mathbf{PER_{3000}}$ valószínűségeket szeretnénk szimulációkkal igazolni, így a kódoknak hatékony dekódoló algoritmussal kell rendelkezniük.

2.0.13. Definíció. A **PER** valószínűségekre vonatkozó szimulációk eredményét **sPER**, míg a $\mathbf{PER_{3000}}$ valószínűségekre vonatkozó szimulációk eredményét $\mathbf{sPER_{3000}}$ fogja jelölni.

Shannon tétele egy felső korlátot ad számunkra a keresendő kódok rátájára:

$$\mathbf{R}(\mathrm{C}) < 1 - h(0.1) = 0.531$$

Feladat

Találjunk olyan C kódokat, melyekre teljesülnek az alábbiak:

- $0.3 < \mathbf{R}(\mathrm{C}) < 0.531$
- $\mathbf{PER_{3000}} \longrightarrow min$

2.0.14. Megjegyzés. A fenti feladatot kielégítő kódokat "jó" kódoknak fogjuk nevezni.

3. fejezet

A vizsgált kódolás

3.1. Reed-Solomon kódok

A *Reed-Solomon kódok* egy nagyon fontos csoportját alkotják a hibajavító kódoknak. Eredetileg *Irving S. Reed* és *Gustave Solomon* nevéhez fűződik a kód (1960). Több különböző eljárás létezik a kódolásra és így a kódszavak halmazának definiálására, de a dolgozatban az eredeti koncepciót ismertetem. Az implementált kódolásnál a *Peterson* által kifejlesztett szindróma dekódolás törléses változatát alkalmaztam. A Peterson féle dekódolási eljárást nem ismertetem teljes részletességgel, csupán a törlések dekódolására is alkalmas módosított algoritmust ismertetem. (Az olvasó azonban megtalálja a teljes leírást például [2] 5.4.1-es fejezetében.)

3.1.1. A kód konstrukciója

Legyen q egy prímhatvány, $\mathbb{F}_q$ primitív elemét pedig jelölje α. Legyen $n := q-1$, $k \in \mathbb{N}$ pedig legyen úgy rögzítve, hogy $0 \leq k \leq n$ teljesüljön. Jelölje $\mathcal{P}_k$ az $\mathbb{F}_q[x]$-beli, k-nál alacsonyabb fokú polinomok halmazát, azaz

$$\mathcal{P}_k := \{p \in \mathbb{F}_q[x] \mid deg(p) < k\}.$$

A Reed-Solomon kód kódszavainak halmazát a következőképpen definiáljuk:

$$C := \{(p(1), p(\alpha), p(\alpha^2), ..., p(\alpha^{n-1})) \mid p \in \mathcal{P}_k\}.$$

3.1.1. Tétel. A C kód egy lineáris (n,k)-kód, melynek minimum távolsága $d(C) = n-k+1$, azaz MDS-kód.

Legyen $ev : \mathcal{P}_k \longrightarrow \mathbb{F}_q^n$ a következő lineáris leképezés:

$$ev(p) := (p(1), p(\alpha), p(\alpha^2), ..., p(\alpha^{n-1})).$$

Legyen $\mathbf{m} = (m_0, m_1, ..., m_{k-1}) \in \mathbb{F}_q^k$ az üzenet, melyet kódolni szeretnénk. Az $\mathbf{m}$ vektorhoz hozzárendeljük a $p_m(x) \in \mathcal{P}_k$ polinomot a következőképpen:

$$p_m(x) := \sum_{i=0}^{k-1} m_i x^i.$$

Az $\mathbf{m}$ üzenet kódolásához vennünk kell a hozzátartozó $m(x)$ polinomot, majd kiszámolni $ev(p_m)$-et.

A kód generátormátrixát könnyen felírhatjuk az előbbi lineáris leképezés segítségével. Nem kell mást tennünk, mint $\mathcal{P}_k$ egy bázisát vennünk, majd minden elemére alkalmazni a lineáris leképezést és így meg is kapjuk C egy bázisát. Vegyük tehát a $\mathcal{P}_k$-beli természetes bázist: $\{1, x, x^2, ..., x^{k-1}\}$ és az ev leképezés melletti képeiket írjuk be egy mátrix soraiba:

$$G = \begin{pmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & \alpha & \alpha^2 & \dots & \alpha^{n-1} \\ 1 & \alpha^2 & \alpha^{2\cdot 2} & \dots & \alpha^{2\cdot(n-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \alpha^{k-1} & \alpha^{(k-1)\cdot 2} & \dots & \alpha^{(k-1)\cdot(n-1)} \end{pmatrix}$$

Az így kapott $G \in \mathbb{F}_q^{k\times k}$ mátrix lesz a C kód generátormátrixa, hiszen soraiban C bázisának elemei vannak. A G mátrix $i.$ sorának $j.$ eleme tehát a következő: $G_{ij} = \alpha^{(i-1)(j-1)}$. A generátormátrix segítségével az $\mathbf{m}$ üzenetet az alábbi módon tudjuk kódolni:

$$\mathbf{c} = \mathbf{m}\ G.$$

Az 1.2 fejezetben leírtak alapján, ha $d(C) = 2v_h + 1$, akkora kód legfeljebb v_h db hibát tud javítani. Tudjuk, hogy $d(C) = n-k+1$, így kapjuk, hogy a kód legfeljebb $v_h = \lfloor \frac{n-k}{2} \rfloor$ hibát tud javítani.

3.1.2. Dekódolás törlések esetén

Ha a dekódolandó üzenet bizonyos helyein tudjuk, hogy hiba van, akkor azokat a helyeket törléseknek nevezzük és tekinthetünk rájuk úgy, mintha azokon a helyeken ?-ek lennének. Legyen tehát $\mathbf{r} \in (\mathbb{F}_q \cup \{?\})^n$ a dekódolandó üzenet. Az alábbi algoritmus segítségével dekódolni tudjuk a törléseket tartalmazó üzeneteket.

1. Legyen v_t a törlések száma, és jegyezzük fel egy vektorba, hogy mely pozíciókon voltak törlések:

$$T = (T_1, T_2, ..., T_{v_t}),$$

majd pedig az $\mathbf{r}$ vektorban írjunk a törlések helyére 0-kat.

2. A célunk az, hogy megtaláljuk azt a $\mathbf{c} \in \mathbb{F}_q^n$ kódszót, melyet eredetileg küldtek. Ehhez meg kell keresnünk az $\mathbf{e} \in \mathbb{F}_q^n$ hibavektort, melyre teljesül, hogy:

$$\mathbf{r} = \mathbf{c} + \mathbf{e}.$$

Legyen $v_h := \lfloor \frac{n-k-v_t}{2} \rfloor$ és $v := v_t + v_h$. Ekkor v_h jelöli a maximálisan javítható hibák számát. Feltesszük, hogy ennyi hibát tartalmaz az $\mathbf{r}$ vektor, a számolásoknál nem fog gondot okozni, ha ennél mégis kevesebb hiba lenne benne. A hibák és törlések együttes számát pedig v fogja jelölni. Tegyük fel hogy az i_k $(1 < k \leq v)$ pozíciókon vannak a törlések és a hibák. Legyen

$$X_k := \begin{cases} \alpha^{T_k}, & \text{ha } k \leq v_t \\ \alpha^{i_k}, & \text{ha } v_t < k \leq v \end{cases} \qquad (1 < k \leq v),$$

illetve $E_k := e_{i_k}$ $(1 < k \leq v)$. (Ekkor feltesszük, hogy az i_k $(1 < k \leq v_t)$ pozíciókon vannak a törlések.) Az X_k jelöli az úgynevezett *hibahelyeket*, míg E_k a *hibaértékeket*. A törlések helyeit tudjuk, így az X_k értékek közül ismerünk v_t darabot, míg az E_k értékek mindegyike ismeretlen. A továbbiakban ezeket fogjuk meghatározni.

3. Számoljuk ki a *szindrómákat* az $\mathbf{r}$ vektor segítségével:

$$S_i := \sum_{j=1}^{n} r_j (\alpha^i)^j \qquad (1 < i \leq n-k).$$

A szindrómák csupán a hibavektortól függenek, így teljesül az alábbi:

$$S_i = \sum_{k=1}^{v} e_{i_k} (\alpha^i)^{i_k} = \sum_{k=1}^{v} E_k X_k^i \qquad (1 < i \leq n-k). \tag{3.1.1}$$

Itt kapunk egy $n-k$ egyenletből álló egyenletrendszert. Ha X_k-k ismertek lennének, akkor ez az egyenletrendszer lineáris lenne és az E_k ismeretleneket már könnyen megtudnánk határozni.

4. Tekintsük az alábbi egyenletrendszert:

$$\begin{pmatrix} X_1^{-1} & X_1^{-2} & \dots & X_1^{-v} \\ X_2^{-1} & X_2^{-2} & \dots & X_2^{-v} \\ \vdots & \vdots & \ddots & \vdots \\ X_{v_t}^{-1} & X_{v_t}^{-2} & \dots & X_{v_t}^{-v} \\ S_v & S_{v-1} & \dots & S_1 \\ S_{v+1} & S_v & \dots & S_2 \\ \vdots & \vdots & \ddots & \vdots \\ S_{v+v_h} & S_{v+v_h-1} & \dots & S_{v_h} \end{pmatrix} \begin{pmatrix} \Lambda_1 \\ \Lambda_2 \\ \vdots \\ \Lambda_{v_t} \\ \Lambda_{v_t+1} \\ \Lambda_{v_t+2} \\ \vdots \\ \Lambda_v \end{pmatrix} = \begin{pmatrix} -1 \\ -1 \\ \vdots \\ -1 \\ -S_{v+1} \\ -S_{v+2} \\ \vdots \\ -S_{v+v_h} \end{pmatrix}$$

Ennek megoldásaként megkapott $\Lambda_1, \Lambda_2, ..., \Lambda_v$ értékek segítségével konstruálhatunk egy polinomot a következőképpen:

$$\Lambda(x) = \prod_{k=1}^{v}(1 - xX_k) = 1 + \Lambda_1 x + \Lambda_2 x^2 + ... + \Lambda_v x^v.$$

3.1.2. Megjegyzés. Mivel minden S_i esetén $i \leq n - k$, így kapjuk, hogy

$$v + v_h \leq n - k \qquad \Longleftrightarrow \qquad v_t + 2v_h \leq n - k,$$

amiből már látható, hogy a 2. pontban miért lett $v_h = \lfloor \frac{n-k-v_t}{2} \rfloor$.

5. Ez előzőleg konstruált $\Lambda(x)$ polinomot *hibakereső polinomnak* nevezzük, melynek a gyökei pontosan a hibahelyek inverzei:

$$\Lambda(X_k^{-1}) = 0 \qquad (1 \leq k \leq v).$$

Így az eddig még ismeretlen X_k $(v_t < k \leq v)$ értékeket megkapjuk, ha megkeressük a $\Lambda(x)$ polinom gyökeit, majd vesszük az inverzüket.

6. Ezután vegyük a hibahelyek logaritmusát és megkapjuk, hogy mely helyeken vannak a hibák a megkapott üzenetben:

$$i_k = \log_\alpha X_k = \log_\alpha \alpha^{i_k} \qquad (v_t < k \leq v).$$

7. A hibahelyek ismeretében már megtudjuk oldani a (3.1.1) lineáris egyenletrendszert és így megkapjuk a hibák értékeit.

3.2. Konkatenált kódok

Shannon tétele szerint, ha egy kód rátája egy bizonyos korlát alatt marad, akkor a kód hibázásának a valószínűsége exponenciálisan csökken a blokkok méretében. Ahhoz tehát, hogy a hibázás valószínűsége kicsi legyen meglehetősen hosszú kódokra van szükségünk, melyek dekódolása igen bonyolult lehet. Dave Forney 1966-os cikkében [1] egy olyan kódolást vezetett be, amely hibázásának a valószínűsége nem csak exponenciálisan csökken a blokkok méretében, hanem ezzel egy időben a dekódolás polinom időben végrehajtható.

A Forney által bevezetett kódot *konkatenált kódnak* nevezzük, melynek alapötlete igen egyszerű, egy belső kód és egy külső kód kombinálásán alapszik.

A $C_{\mathcal{I}}$ belső kód egy N hosszúságú, K dimenziós blokk kód az A ábécé felett, míg a $C_{\mathcal{O}}$ külső kód egy n hosszúságú, k dimenziós blokk kód a B ábécé felett, ahol $|B| = |A|^K$. Ekkor létezik egy $\varphi : B \to A^K$ bijekció,melynek segítségével egy $b \in B$ elemet azonosítani tudunk egy $\mathbf{a} \in A^K$ vektorral. A két kód egymásutáni alkalmazásával kapjuk a $C := C_{\mathcal{O}} \circ C_{\mathcal{I}} = C_{\mathcal{O}} \circ_{\varphi} C_{\mathcal{I}}$ konkatenált kódot, mely nN hosszúságú és kK dimenziós.

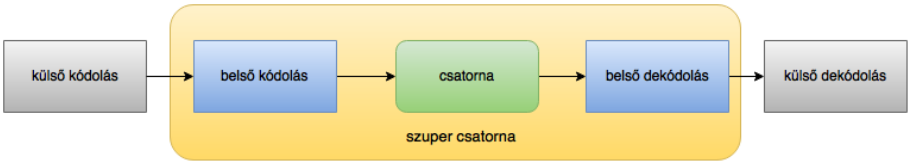
Konkatenált kód	
Belső kód:	$C_{\mathcal{I}} : A^K \to A^N$
Külső kód:	$C_{\mathcal{O}} : B^k \to B^n$
Konkatenált kód:	$C : A^{kK} \to A^{nN}$

Legyen $\mathbf{m} = (m_1, m_2, ..., m_k) \in B^k$ a továbbítandó üzenet, melyet az alábbi módon fogunk kódolni a konkatenált kód segítségével:

$$C(\mathbf{m}) = (C_{\mathcal{I}}(m'_1), C_{\mathcal{I}}(m'_2), ..., C_{\mathcal{I}}(m'_n)),$$

ahol $\mathbf{m}' = (m'_1, m'_2, ..., m'_n) = C_{\mathcal{O}}(\mathbf{m})$. Tehát először kódoljuk az üzenetet a külső kód segítségével, majd pedig a belső kód segítségével. Dekódolásnál pont fordítva járunk el. A csatornán megérkezett üzenetet először a belső kóddal dekódoljuk, majd pedig külső kóddal.

A 3.1 ábra szemlélteti a konkatenált kódok működését. A sárga keretben lévő részre tekinthetünk úgy, mint egy *szuper csatornára*, melyen a külső kóddal kódolt üzeneteket küldjük át és a belső kóddal való dekódolás után a csatornából megérkező üzeneteket a külső kóddal dekódoljuk.



3.1. ábra.

3.3. Konkatenált kód Reed-Solomon külső kóddal

Ebben a fejezetben ismertetem azt a konkatenált kódot, amellyel dolgoztam. A belső kód egy l hosszúságú, f dimenziós bináris lineáris kód, míg a külső kód egy n hosszúságú, k dimenziós Reed-Solomon kód az $\mathbb{F}_q$ test felett, ahol $q = 2^f$.

Ekkor a $\varphi : \mathbb{F}_q \to \mathbb{F}_2^f$ bijekció egy $\mathbb{F}_2$-lineáris leképezés lesz, aminek köszönhetően a konkatenált kód is lineáris kód lesz. Egy $u \in \mathbb{F}_q$ elem felírható a következő alakban:

$$u = \sum_{i=0}^{f-1} v_i \, \alpha^i, \text{ ahol } v_i \in \{0, 1\}.$$

Ekkor a $\varphi : \mathbb{F}_q \to \mathbb{F}_2^f$ lineáris leképezés a következő:

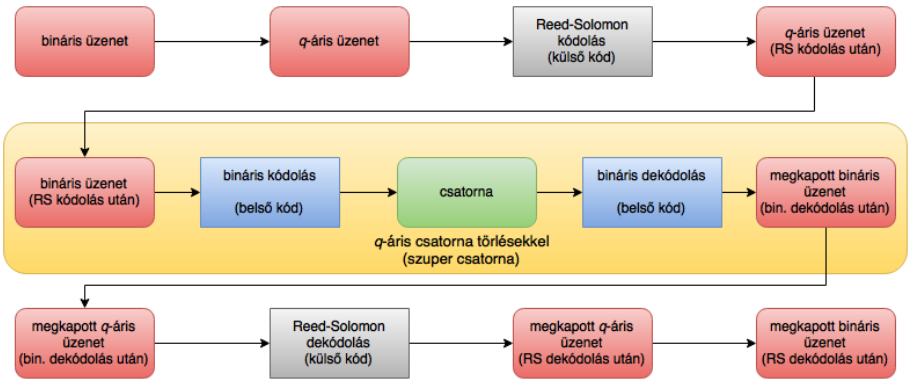
$$\varphi(u) = (v_0, v_1, ..., v_{f-1}) \in \mathbb{F}_2^f.$$

Ha pedig $\mathbf{v} = (v_0, v_1, ..., v_{f-1}) \in \mathbb{F}_2^f$, akkor

$$\varphi^{-1}(\mathbf{v}) = \sum_{i=0}^{f-1} v_i \, \alpha^i \in \mathbb{F}_q.$$

Konkatenált kód Reed-Solomon külső kóddal

Belső kód:	$C_{\mathcal{I}} : \mathbb{F}_2^f \to \mathbb{F}_2^\ell$
Külső kód (Reed-Solomon):	$C_{\mathcal{O}} : \mathbb{F}_q^k \to \mathbb{F}_q^n$
Konkatenált kód:	$C : \mathbb{F}_2^{fk} \to \mathbb{F}_2^{ln}$



3.2. ábra.

A 3.2-es ábra egy áttekintést ad a kódoláshoz, melyet az alábbiakban részletesen ismertetek.

Legyen $C_{\mathcal{I}}$ egy bináris lineáris (l,f)-kód a $G_{\mathcal{I}}$ szisztematikus generátor mátrixszal és a $H_{\mathcal{I}}$ paritás ellenőrző mátrixszal. Tehát $C_{\mathcal{I}} \leqslant \mathbb{F}_2^l$ és $dim(C_{\mathcal{I}}) = f$. A hibajavító dekódolás viselkedése egy $t \in \mathbb{N}$ küszöbértéktől függ, melyet úgy kell megválasztani, hogy $2 \leq t < d$ teljesüljön, ahol $d := d(C_{\mathcal{I}})$ jelöli a $C_{\mathcal{I}}$ kód minimális távolságát.

Forney a t küszöbértéket nem változtatta vizsgálatai során, hanem fixen mindig a $t = \lfloor \frac{d-1}{2} \rfloor$ értékkel dolgozott. Ebben az esetben a kódszavak körüli t sugarú gömbök biztosan diszjunktak lesznek, míg a mi esetünkben ez nem minden esetben lesz így, ami egy kicsit bonyolítja a számolást, mint azt majd látni fogjuk, de előfordulhatnak olyan kódok is, ahol a kódszavak nem teljesen szimmetrikusan helyezkednek el és így érdemes lehet megnézni más küszöbértékekre is a kódolási eljárást.

Legyen $q := 2^f$, $n := q - 1$, $k \in \mathbb{N}$ pedig olyan, hogy $0 < k < n$. Ekkor tekintsük a $C_{\mathcal{O}} \leqslant \mathbb{F}_q^n$, k-dimenziós Reed-Solomon kódot a $G_{\mathcal{O}}$ generátormátrixszal, továbbá jelölje α az $\mathbb{F}_q$ primitív elemét.

3.3.1. A kódolás menete

Legyen $\mathbf{x} = (x_0, x_1, ..., x_{f \cdot k-1}) \in \mathbb{F}_2^{f \cdot k}$ a bináris üzenet, melyet továbbítani szeretnénk. Az $\mathbf{x}$ üzenetet a következő négy lépésen keresztül fogjuk kódolni:

1. Az $\mathbf{x} \in \mathbb{F}_2^{f \cdot k}$ vektort átalakítjuk egy $\hat{\mathbf{x}} = (\hat{x}_0, \hat{x}_1, ..., \hat{x}_{k-1}) \in \mathbb{F}_q^k$ vektorrá. Jelölje $\mathbf{x_j} \in \mathbb{F}_2^f$ az $\mathbf{x}$ vektor j. f-hosszúságú blokkját, azaz

$$\mathbf{x_j} := (x_{j \cdot f}, x_{j \cdot f+1}, ..., x_{(j+1) \cdot f-1}) \qquad (j = 0, 1, ..., k-1).$$

Az átalakításhoz alkalmazzuk a φ^{-1} leképezést:

$$\hat{x}_j = \varphi^{-1}(\mathbf{x_j}) \qquad (j = 0, 1, ..., k-1).$$

2. Az $\hat{\mathbf{x}} \in \mathbb{F}_q^k$ vektort ezután kódoljuk a külső, Reed-Solomon kód segítségével:

$$\hat{\mathbf{y}} = (\hat{y}_0, \hat{y}_1, ..., \hat{y}_{n-1}) = G_{\mathcal{O}} \cdot \hat{\mathbf{x}} \in \mathbb{F}_q^n.$$

3. Az $\hat{\mathbf{y}} \in \mathbb{F}_q^n$ q-áris vektort egy $\mathbf{y} = (y_0, y_1, ..., y_{f \cdot n-1}) \in \mathbb{F}_2^{f \cdot n}$ bináris vektorrá alakítjuk a φ leképezés segítségével. Jelölje $\mathbf{y_j} \in \mathbb{F}_2^f$ az $\mathbf{y}$ vektor j. f-hosszúságú blokkját, azaz

$$\mathbf{y_j} := (y_{j \cdot f}, y_{j \cdot f+1}, ..., y_{(j+1) \cdot f-1}) \qquad (j = 0, 1, ..., n-1).$$

Ekkor

$$\mathbf{y_j} = \varphi(\hat{y}_j) \qquad (j = 0, 1, ..., n-1).$$

4. Ezután az $\mathbf{y} \in \mathbb{F}_2^{f \cdot n}$ vektort kódoljuk a $C_{\mathcal{I}}$ belső kód segítségével egy $\mathbf{z} = (z_0, z_1, ..., z_{l \cdot n-1}) \in \mathbb{F}_2^{l \cdot n}$ vektorrá. Mivel $\mathbf{y} \in \mathbb{F}_2^{f \cdot n}$, így feltudjuk osztani n db f komponensből álló blokkra. Az így kapott blokkokat külön-külön kódolni tudjuk a $G_{\mathcal{I}}$ $f \times l$-es generátormátrix segítségével. Így lesz n db l komponensből álló blokkunk, melyekből kapunk egy $n \cdot l$ komponensből álló vektort.
Tehát

$$\mathbf{z} = (\mathbf{y_0} \cdot G_{\mathcal{I}}, \mathbf{y_1} \cdot G_{\mathcal{I}}, ..., \mathbf{y_{n-1}} \cdot G_{\mathcal{I}}) \in \mathbb{F}_2^{l \cdot n},$$

ahol

$$\mathbf{y_i} = (y_{i \cdot f}, y_{i \cdot f+1}, ..., y_{(i+1) \cdot f-1}) \qquad (i = 0, 1, ..., n-1).$$

Így az $\mathbf{x} \in \mathbb{F}_2^{f \cdot k}$ üzenetet a $\mathbf{z} = (z_0, z_1, ..., z_{l \cdot n-1}) \in \mathbb{F}_2^{l \cdot n}$ vektorra kódoltuk, melyet ezután átküldünk a csatornán és a címzetthez az $\mathbf{r} = (r_0, r_1, ..., r_{l \cdot n-1}) \in \mathbb{F}_2^{l \cdot n}$ vektor érkezik meg.

3.3.2. A dekódolás menete

A címzetthez megérkezett $\mathbf{r} \in \mathbb{F}_2^{l \cdot n}$ vektort ezután dekódolnunk kell, melyet a következő négy lépésen keresztül tudunk megtenni:

1. Először a belső kóddal hajtjuk végre a dekódolás. A belső kód viselkedése a t küszöbértéktől függ. Az 1.3-as fejezetben a lineáris kódok dekódolásánál ismertetett szindróma dekódolás segítségével fogjuk dekódolni az $\mathbf{r} \in \mathbb{F}_2^{l \cdot n}$ vektort.
   Jelölje $\mathbf{r_i} \in \mathbb{F}_2^l$ az $\mathbf{r}$ vektor i. l-hosszúságú blokkját, azaz

$$\mathbf{r_i} := (r_{i \cdot l}, r_{i \cdot l+1}, ..., r_{(i+1) \cdot l-1}) \in \mathbb{F}_2^l \qquad (i = 0, 1, ..., n-1).$$

   Ekkor legyen $\mathbf{s_i} := \mathbf{r_i} H_{\mathcal{I}}^T \in \mathbb{F}_2^f$ az $\mathbf{r_i}$-hez tartozó szindróma, $\mathbf{e_i} \in \mathbb{F}_2^l$ pedig a hozzá tartozó hibavektor. A t küszöbértéktől függően ezután kétféleképpen folytathatjuk a dekódolás menetét:

   (a) Ha $w(\mathbf{e_i}) \leq t$, azaz a t küszöbértéknél kevesebb hiba található az $\mathbf{r_i}$ blokkban, akkor a következőt tesszük: *javítjuk a hibákat*, tehát az $\mathbf{r'_i} := \mathbf{r_i} - \mathbf{e_i} \in \mathbb{F}_2^l$ vektort az

$$\mathbf{y'_i} := (r'_{i,0}, r'_{i,1}, ..., r'_{i,f-1}) \in \mathbb{F}_2^f$$

   vektorra dekódoljuk.

   (b) Ha viszont $w(\mathbf{e_i}) > t$, azaz a t küszöbértéknél több hiba található az $\mathbf{r_i}$ blokkban, akkor a következőt tesszük: a belső kóddal nem végzünk hibajavítást az $\mathbf{r_i}$ blokkon, csupán *hibajelzést adunk*, hogy az i. blokk biztosan hibás. Ebben az esetben

$$\mathbf{y'_i} := (r_{i,0}, r_{i,1}, ..., r_{i,f-1}) \in \mathbb{F}_2^f$$

   és feljegyezzük a blokk sorszámát. Ezen blokkok helyén úgynevezett *törlések* fognak kialakulni.

   Ha a két lehetőség valamelyikét végrehajtottuk minden $\mathbf{r_i}$ $(i = 0, 1, ..., n-1)$ blokkra, akkor kapunk egy

$$\mathbf{y'} := (\mathbf{y'_0}, \mathbf{y'_1}, ..., \mathbf{y_{n-1}}') \in \mathbb{F}_2^{f \cdot n}$$

   vektort és lesz egy vektorunk, mely a törlési pozíciókat tartalmazza:

$$T = (T_1, T_2, ..., T_{v_t}).$$

2. Az $\mathbf{y'} \in \mathbb{F}_2^{f \cdot n}$ vektort ezután átalakítjuk egy $\mathbf{y''} \in \mathbb{F}_q^n$ vektorrá a kódolás 1. lépésénél látottakhoz hasonlóan.

3. Az $\mathbf{y}'' \in \mathbb{F}_q^n$ vektort ezután a külső kóddal dekódoljuk, méghozzá a Reed-Solomon féle törléses dekódolási eljárást fogjuk alkalmazni. Ennek érdekében, ha $i \in T$, akkor legyen $y_i'' = 0$, azaz a törlési pozíciók mindegyikén innentől kezdve 0 fog szerepelni. Ezután elvégezhetjük a Reed-Solomon dekódolást, melynek eredményeképpen kapunk egy

$$\mathbf{x}'' = (x_0'', x_1'', ..., x_{k-1}'') \in \mathbb{F}_q^k$$

vektort.

4. Utolsó lépésben az $\mathbf{x}'' \in \mathbb{F}_q^k$ q-áris vektort átalakítjuk az

$$\mathbf{x}' = (x_0', x_1', ..., x_{f\cdot k-1}') \in \mathbb{F}_2^{f\cdot k}$$

bináris vektorrá, a kódolás 3. lépésénél látottakhoz hasonlóan.

Így a címzetthez megérkezett $\mathbf{r} \in \mathbb{F}_2^{l\cdot n}$ vektort a dekódolás végrehajtása után az $\mathbf{x}' = (x_0', x_1', ..., x_{f\cdot k-1}') \in \mathbb{F}_2^{f\cdot k}$ vektorra dekódoltuk.

A 3.2-es ábra jól összefoglalja a kódolás és a dekódolás lépéseit. A középső sorra tekinthetünk egy q-áris csatornaként, melyben törlések keletkezhetnek.

4. fejezet

A probléma megoldása

4.1. A megoldás ismertetése

Az előző fejezetben ismertetett konkatenált kód segítségével a feladatunkat átfogalmazhatjuk a következő alakba:

Feladat

Határozzuk meg az optimális f, l, t és k paramétereket úgy, hogy az alábbiak teljesüljenek:

- $\mathbf{PER} < 0.01$ és $\mathbf{R}(\mathrm{C}) \to max$
- $0.3 < \mathbf{R}(\mathrm{C})$ és $\mathbf{PER_{3000}} \to min$

Az optimális paramétereket a következő lépéseken keresztül határoztuk meg:

1. lépés

Rögzített f és l értékek esetén a SageMath szoftver segítségével belső kódnak vettük a legjobb ismert lineáris (l, f)-kódot. Fix f és l esetén a külső kód hossza $n = 2^f - 1$.

2. lépés

Legyen $d_{\mathcal{I}} := d(C_{\mathcal{I}})$. Ekkor minden $t \in [2, d_{\mathcal{I}} - 1]$ küszöbértékre kiszámoltuk az alábbi valószínűségeket:

$$
\begin{aligned}
p_0 &= \Pr(C_{\mathcal{I}} \text{ helyesen dekódol egy szimbólumot}) \\
p_1 &= \Pr(C_{\mathcal{I}} \text{ töröl egy szimbólumot}) \\
p_2 &= \Pr(C_{\mathcal{I}} \text{ helytelenül dekódol egy szimbólumot})
\end{aligned}
$$

Tegyük fel, hogy az $\mathbf{y} \in \mathbb{F}_2^l$ vektor érkezett meg a csatornán keresztül. Ekkor az előző valószínűségek a következőképp alakulnak:

$$
\begin{aligned}
p_0 &= \frac{1}{|C_{\mathcal{I}|}} \sum_{\mathbf{c} \in C_{\mathcal{I}}} \Pr(\mathbf{y}\text{-t } C_{\mathcal{I}} \text{ a } \mathbf{c} \text{ kódszóra dekódolja} \mid \text{a } \mathbf{c} \text{ kódszót küldtük}) \\
p_2 &= \frac{1}{|C_{\mathcal{I}|}} \sum_{\mathbf{c} \in C_{\mathcal{I}}} \sum_{\mathbf{c}' \in C_{\mathcal{I}}} \Pr(\mathbf{y}\text{-t } C_{\mathcal{I}} \text{ a } \mathbf{c}' \text{ kódszóra dekódolja} \mid \text{a } \mathbf{c} \text{ kódszót küldtük}) \\
p_1 &= 1 - p_0 - p_2
\end{aligned}
$$

1. eset

Ha $t \leq \frac{d(C_{\mathcal{I}})-1}{2}$, akkor $\mathbf{y}$-t helyesen dekódoljuk, ha a $\mathbf{c}$ kódszó körüli t sugarú gömbben helyezkedik el:

$$
p_0 = \frac{1}{|C_{\mathcal{I}|}} \sum_{\mathbf{c} \in C_{\mathcal{I}}} \sum_{\mathbf{y} \in B_t(\mathbf{c})} p^{d(\mathbf{c},\mathbf{y})} (1-p)^{l-d(\mathbf{c},\mathbf{y})}.
$$

Az $\mathbf{y}$ vektort helytelenül dekódoljuk, ha valamely $\mathbf{c}' \neq \mathbf{c}$ kódszó körüli t sugarú gömbben helyezkedik el:

$$
p_2 = \frac{1}{|C_{\mathcal{I}|}} \sum_{\mathbf{c} \in C_{\mathcal{I}}} \sum_{\mathbf{c}' \in C_{\mathcal{I}} \setminus \{\mathbf{c}\}} \sum_{\mathbf{y} \in B_t(\mathbf{c}')} p^{d(\mathbf{c},\mathbf{y})} (1-p)^{l-d(\mathbf{c},\mathbf{y})}.
$$

Törlünk, ha az $\mathbf{y}$ vektor egyik kódszó körüli t sugarú gömbben sem helyezkedik el:

$$
p_1 = 1 - p_0 - p_2.
$$

4.1.1. Megjegyzés. Ha az $\mathbf{y} \in \mathbb{F}_2^l$ vektort a $\mathbf{c} \in C_{\mathcal{I}}$ vektorra javítjuk, akkor az $\mathbf{y} + \mathbf{d} \in \mathbb{F}_2^l$ vektort, ahol $\mathbf{d} \in C_{\mathcal{I}}$, a $\mathbf{c} + \mathbf{d} \in C_{\mathcal{I}}$ vektorra fogjuk javítani, mivel a két vektor szindrómája megegyezik és így a hozzájuk tartozó hibavektorok is azonosak:

$$
\mathbf{y}H^T = (\mathbf{c} + \mathbf{e})H^T = \mathbf{c}H^T + \mathbf{e}H^T = \mathbf{0} + \mathbf{e}H^T = \mathbf{e}H^T = \mathbf{s}
$$

$$
(\mathbf{y} + \mathbf{d})H^T = \mathbf{y}H^T + \mathbf{d}H^T = \mathbf{s} + \mathbf{0} = \mathbf{s}.
$$

Speciálisan, ha $\mathbf{c} = \mathbf{0}$, akkor a $\mathbf{0}$ körüli t sugarú gömb elemei lesznek jól javítva és minden kódszó körüli gömböt megkaphatunk úgy, hogy eltoljuk a $\mathbf{0}$ körüli gömböt az adott kódszóval:

$$
B_t(\mathbf{d}) = \{\mathbf{y} + \mathbf{d} \mid \mathbf{y} \in B_t(\mathbf{0})\}.
$$

Ebből kifolyólag elég lesz a $c = \mathbf{0}$ esetre kiszámolnunk a valószínűségeket:

$$\begin{aligned} p_0 &= \sum_{\mathbf{y} \in B_t(\mathbf{0})} p^{d(\mathbf{0},\mathbf{y})}(1-p)^{l-d(\mathbf{0},\mathbf{y})} \\ &= \sum_{\mathbf{y} \in B_t(\mathbf{0})} p^{w(\mathbf{y})}(1-p)^{l-w(\mathbf{y})}, \end{aligned}$$

$$\begin{aligned} p_2 &= \sum_{\mathbf{c}' \in C_{\mathcal{I}} \setminus \{\mathbf{0}\}} \sum_{\mathbf{y} \in B_t(\mathbf{c}')} p^{d(\mathbf{0},\mathbf{y})}(1-p)^{l-d(\mathbf{0},\mathbf{y})} \\ &= \sum_{\mathbf{c}' \in C_{\mathcal{I}} \setminus \{\mathbf{0}\}} \sum_{\mathbf{y} \in B_t(\mathbf{c}')} p^{w(\mathbf{y})}(1-p)^{l-w(\mathbf{y})} \\ &= \sum_{\mathbf{c}' \in C_{\mathcal{I}} \setminus \{\mathbf{0}\}} \sum_{\mathbf{z} \in B_t(\mathbf{0})} p^{w(\mathbf{z}+\mathbf{c}')}(1-p)^{l-w(\mathbf{z}+\mathbf{c}')}. \end{aligned}$$

2. eset

Ha $t > \frac{d(C_{\mathcal{I}})-1}{2}$, akkor egy kicsit bonyolultabb a számolás, mert a kódszavak körüli t sugarú gömbök nem minden esetben diszjunktat. Ebben az esetben minden $\mathbf{c} \in C_{\mathcal{I}}$ kódszó esetén kiszámoljuk a hozzá tartozó cellát, ami azokat a vektorokat tartalmazza, amiket a $\mathbf{c}$ kódszóra javítunk:

$$cell(\mathbf{c}) = \{\mathbf{y} \in \mathbb{F}_2^l \mid \mathbf{y}\text{-t a } \mathbf{c} \text{ kódszóra javítjuk}\}$$

Az előző esethez képest annyi módosítással kell élnünk, hogy a $\mathbf{0}$ körüli t sugarú gömb helyett a gömb és a $\mathbf{0}$-hoz tartozó cella metszetével dolgozunk:

$$B_t^{cut}(\mathbf{0}) := B_t(\mathbf{0}) \cap cell(\mathbf{0}).$$

Tehát a valószínűségek az alábbi módon számolhatóak ki:

$$\begin{aligned} p_0 &= \sum_{\mathbf{y} \in B_t^{cut}(\mathbf{0})} p^{w(\mathbf{y})}(1-p)^{l-w(\mathbf{y})} \\ p_2 &= \sum_{\mathbf{c}' \in C_{\mathcal{I}} \setminus \{\mathbf{0}\}} \sum_{\mathbf{z} \in B_t^{cut}(\mathbf{0})} p^{w(\mathbf{z}+\mathbf{c}')}(1-p)^{l-w(\mathbf{z}+\mathbf{c}')} \\ p_1 &= 1 - p_0 - p_2 \end{aligned}$$

3. lépés

Legyen X_i $(i = 1, 2, ..., n)$ az alábbi véletlen változó:

$$X_i := \begin{cases} 0, & \text{ha } C_{\mathcal{I}} \text{ helyesen dekódol} \\ 1, & \text{ha } C_{\mathcal{I}} \text{ töröl} \\ 2, & \text{ha } C_{\mathcal{I}} \text{ hibásan dekódol} \end{cases}$$

Legyen

$$v_t := \text{ a belső kód törléseinek száma,}$$

$$v_h := \text{ a belső kód hibáinak száma,}$$

és legyen Y az alábbi véletlen változó:

$$Y := X_1 + X_2 + ... + X_n.$$

Ekkor Y tulajdonképpen a belső kód törléseinek plusz kétszer a belső kód hibáinak a számát adja meg, azaz

$$Y = v_t + 2v_h.$$

Az X_i véletlen változók függetlenek egymástól és az Y véletlen változó nagy n esetén normális eloszlást követ:

$$\mu := E(X_i) = p_1 + 2p_2$$

$$\sigma := D(X_i) = \sqrt{\mu^2 p_0 + (1-\mu)^2 p_1 + (2-\mu)^2 p_2}$$

$$E(Y) = n\mu$$

$$D(Y) = \sqrt{n}\sigma$$

$$Y \sim \mathcal{N}(n\mu, n\sigma^2)$$

4. lépés

Legyen $k = k(t)$ olyan, hogy

$$\Pr(Y > n - k) = 0.01,$$

azaz a külső kód dimenzióját úgy szeretnénk megválasztani, hogy a belső kód törléseinek plusz kétszer a hibáinak száma az csupán 1 % valószínűséggel legyen magasabb, mint $n-k$. Ha $v_t+2v_h < n-k$, akkor a külső Reed-Solomon kód biztosan javítani tudja az összes hibát, így ha a k paramétert a fenti módon választjuk meg, akkor az esetek legalább 99 %-ában a külső kód minden hibát ki fog tudni javítani.

$$\Pr(Y > n-k) = 0.01 \quad \Longleftrightarrow \quad \Pr(Y \leq n-k) = 0.99 \quad \Longleftrightarrow$$

$$\mathrm{F_Y}(n-k) = 0.99 \quad \Longleftrightarrow \quad n-k = \mathrm{F_Y^{-1}}(0.99) \quad \Longleftrightarrow \quad k = n - \mathrm{F_Y^{-1}}(0.99),$$

tehát kapjuk, hogy a keresett k paraméter értéke a következő:

$$k = \lfloor n - \mathrm{F_Y^{-1}}(0.99) \rfloor.$$

5. lépés

Ha ilyen módon meghatároztuk a k paramétereket, akkor a konkatenált kódok rátáját a következőképpen tudjuk kiszámolni:

$$\mathbf{R}(\mathrm{C}) = \mathbf{R}(C_{\mathcal{I}}) \cdot \mathbf{R}(C_{\mathcal{O}}) = \frac{fk}{ln}$$

Ezután fix f és l esetén úgy választjuk ki az optimális t paramétert , hogy $\mathbf{R}(\mathrm{C})$ maximális legyen, és az ehhez a t értékhez tartozó $k = k(t)$ lesz a Reed-Solomon kód dimenziója.

4.2. Eredmények

Az előzőekben ismertetett módszerrel, a feladat megoldása érdekében elért eredményeket a következőekben ismertetem.

A vizsgálatok során az f és l paraméterek a következőképpen voltak rögzítve:

$$f \in \{6, 7, 8\},$$

$$l \in \{16, 17, 18, 19, 20\}.$$

Ezen értékekre lettek meghatározva tehát a t és k paraméterek a fent leírt módon. Ezzel az átfogalmazott feladat első pontját megoldottuk, a második pont megoldásához további dolgokra van szükségünk.

Adott f, l, t és k esetén a **PER** értékeket az alábbi módon határozhatjuk meg:

$$\mathbf{PER} = 1 - \mathrm{F_Y}(n - k),$$

majd ebből ki tudjuk számolni a $\mathbf{PER_{3000}}$ értékeket is:

$$\mathbf{PER_{3000}} = 1 - (1 - \mathbf{PER})^{\lceil \frac{3000}{fk} \rceil}.$$

Szükségünk van továbbá a kód rátájára:

$$\mathbf{R}(\mathrm{C}) = \frac{fk}{ln}.$$

Itt az fk érték azt jelzi, hogy az ln elküldött bitből hány bit volt az úgynevezett hasznos információ. A mi esetünkben azonban nem csak fk hasznos bitet szeretnénk elküldeni, hanem 3000 bitet. Ezt úgy érjük el, hogy a 3000 bites üzenetet fk méretű blokkokra bontjuk, és úgy fogjuk kódolni. Legyen

$$r := \left\lceil \frac{3000}{fk} \right\rceil,$$

azaz r db blokk segítségével küldjük át a 3000 bitet. Ehhez azonban rln bitet kell a csatornán továbbítanunk, melyből csupán 3000 bit hasznos, azaz erre az esetre is számolni tudunk egy ráta értéket, melyet $\mathbf{R}_{3000}(\mathrm{C})$ jelöl:

$$\mathbf{R}_{3000}(\mathrm{C}) = \frac{3000}{rln}.$$

A 4.2 ábrán található táblázatok tartalmazzák adott f és l paraméterek esetén a kiszámolt optimális t és k értékeket, majd a paraméterekből meghatározott rátákat, a **PER** és $\mathbf{PER_{3000}}$ valószínűségeket, illetve ezen paraméterek szimulált értékeit. A 4.2 ábrán láthatóak a különböző **PER** értékek az optimális paraméterek mellett.

A diagramokról jól látható, hogy az első esetben a legmagasabbak a $\mathbf{PER_{3000}}$ értékek, amikor a belső kód dimenziója $f = 6$. Ebben az esetben a szimulált értékek nem egészen illeszkednek a kiszámolt értékekhez. Ha az első diagramot összevetjük az első táblázattal, akkor látjuk, hogy ebben az esetben nincs olyan kód, melynek a rátája megfelelő lenne számunkra, hiszen minden ráta 0.3 alatt van.

A második esetben, amikor a belső kód dimenziója $f = 7$, akkor egy kicsit jobb a helyzet. A $\mathbf{PER_{3000}}$ értékek $0.03 - 0.05$ körül mozognak, míg az előző esetben $0.07 - 0.1$ körül voltak. Itt már a szimulált eredmények is jobban illeszkednek a kiszámolt értékekhez és az $l = 18$ esetben a kód rátája is egy kicsivel 0.3 felett van.

Az utolsó esetben, amikor a belső kód dimenziója $f = 8$, akkor több jó kódot is találunk. A $\mathbf{PER_{3000}}$ értékek $0.01 - 0.03$ körül mozognak és a szimulált eredmények is szépen illeszkednek a kiszámolt értékekhez. A kódok rátája is megfelelő az $l = 18$ eset kivételével. Láthatjuk a táblázatban, hogy az $l = 19$ esetben az $\mathbf{R}_{3000}(\mathrm{C})$ ráta is 0.3 fölé megy.

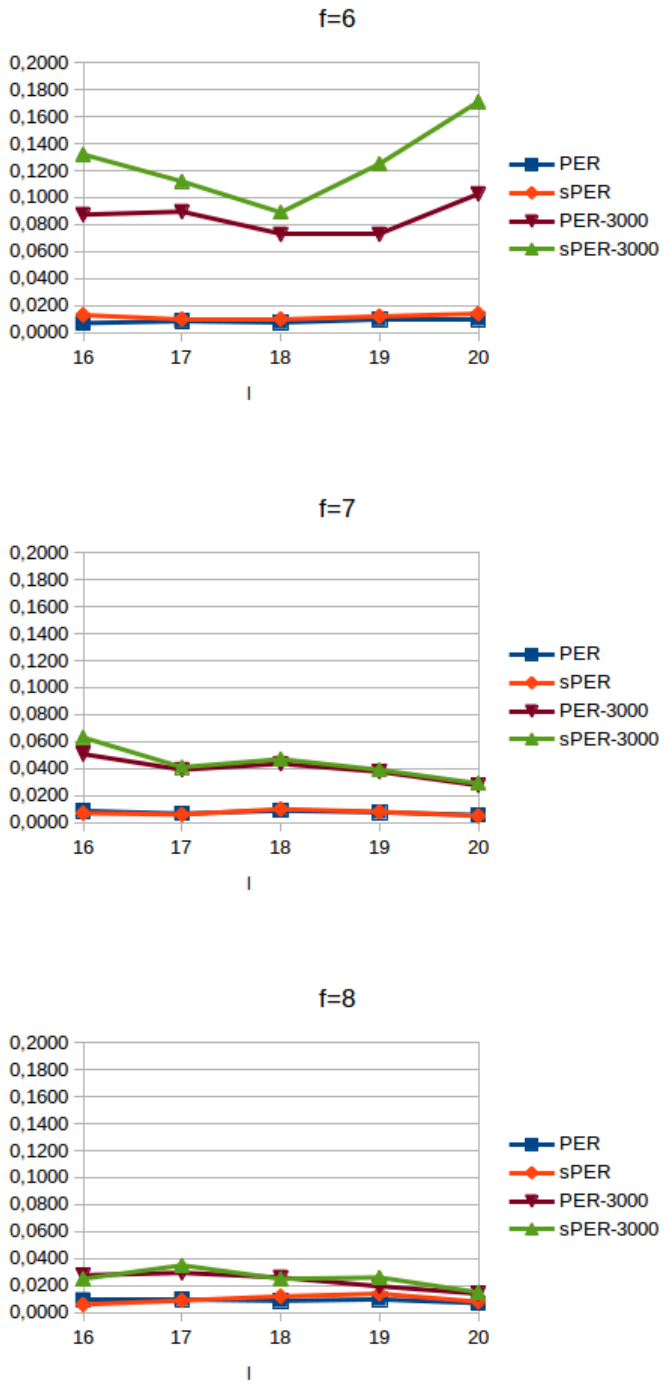
Összegzésként elmondható tehát, hogy a táblázatokban a színessel kiemelt rátájú kódok teljesítik az átfogalmazott feladat második pontját, hiszen rátájuk 0.3 felett van. Ezek közül pedig kiválaszthatjuk azt a kódot, melynek a $\mathbf{PER_{3000}}$ értéke is alacsony.

f	6							
l	$\mathbf{t_o}$	$\mathbf{k_o}$	**R**	$\mathbf{R_{3000}}$	**PER**	**sPER**	$\mathbf{PER_{3000}}$	$\mathbf{sPER_{3000}}$
16	3	39	0.2321	0.2289	0.0070	0.013	0.0874	0.1320
17	3	49	0.2745	0.2546	0.0085	0.01	0.0896	0.1120
18	3	50	0.2646	0.2646	0.0075	0.01	0.0728	0.0890
19	4	49	0.2456	0.2278	0.0097	0.012	0.1013	0.1250
20	4	49	0.2333	0.2165	0.0098	0.014	0.1027	0.1710

f	7							
l	$\mathbf{t_o}$	$\mathbf{k_o}$	**R**	$\mathbf{R_{3000}}$	**PER**	**sPER**	$\mathbf{PER_{3000}}$	$\mathbf{sPER_{3000}}$
16	2	85	0.2928	0.2461	0.0086	0.0070	0.0507	0.0630
17	2	81	0.2626	0.2316	0.0066	0.0060	0.0392	0.0410
18	3	98	0.3001	0.2625	0.0089	0.0100	0.0436	0.0470
19	3	101	0.2930	0.2487	0.0077	0.0080	0.0377	0.0390
20	3	98	0.2701	0.2362	0.0056	0.0050	0.0276	0.0290

f	8							
l	$\mathbf{t_o}$	$\mathbf{k_o}$	**R**	$\mathbf{R_{3000}}$	**PER**	**sPER**	$\mathbf{PER_{3000}}$	$\mathbf{sPER_{3000}}$
16	2	154	0.3020	0.2451	0.0093	0.0060	0.0277	0.025
17	2	168	0.3100	0.2307	0.0099	0.0090	0.0295	0.035
18	2	161	0.2806	0.2179	0.0088	0.0120	0.0260	0.025
19	3	193	0.3187	0.3096	0.0098	0.0140	0.0195	0.026
20	3	201	0.3153	0.2941	0.0070	0.0080	0.0140	0.015

4.1. ábra. Optimális k, t paraméterek esetén a különböző **PER** értékek



4.2. ábra. A különböző **PER** értékek az optimális paraméterek esetén

A. függelék

Implementáció

```
### BELSO KOD
### f-dimenzio, l-hossz

# a belso kod szindromaihoz tartozo hibavektorokat eltarolja az inner_code_standard_array listaban
def InnerCode_standard_array(l, f):
    global InnerCode, inner_code_genmat, inner_code_checkmat
    global inner_code_mindist, inner_code_standard_array
    InnerCode=best_known_linear_code(l,f,GF(2))
    InnerCode.generator_matrix_systematic()
    inner_code_genmat=InnerCode.generator_matrix_systematic()
    inner_code_checkmat=inner_code_genmat.right_kernel_matrix().transpose()
    inner_code_mindist=InnerCode.minimum_distance()

    inner_code_standard_array=[-1]*(2^(l-f))
    word=vector(GF(2),l)
    for i in range(inner_code_mindist):
              for c in Combinations(l,i):
            word*=0
            for j in c: word[j]=1
            syndrome = vector(word)*inner_code_checkmat
            syndrome = sum(2^i for i in range(l-f) if syndrome[i]!=0)
            if -1==inner_code_standard_array[syndrome]:
                inner_code_standard_array[syndrome]=vector(word)

# kodolas a belso koddal
def INNER_encode(word):
        word=list(word)
        while len(word)%f: word.append(0)
        enc=[]
        for i in range(len(word)/f): enc.extend(vector(word[i*f:(i+1)*f])*inner_code_genmat)
        return vector(enc)

# az l hosszusagu recv vektor dekodolasa a threshold kuszobtol fuggoen
# torles eseten suspect=True es nem javit, egyebkent suspect=False es javit
def inner_decode(recv,threshold):
```

```
36          syndrome=vector(recv)*inner_code_checkmat
37          syndrome=sum(2^i for i in range(l-f) if syndrome[i]!=0)
38          syndrome=inner_code_standard_array[syndrome]
39          if (syndrome==-1) or (syndrome.hamming_weight() > threshold):
40                  suspect = True
41                  return recv[0:f], suspect
42          else:
43                  recv=vector(recv)-syndrome
44                  suspect = False
45                  return recv[0:f], suspect
46
47  # word vektor dekodolasa, melyet l hosszusagu blokkokra bontva az elozo fuggveny dekodol
48  # dec vektorba dekodolja word-ot es egy listat is visszaad a torlesi poziciokkal
49  def INNER_decode(word, threshold):
50          dec=[]
51          erasure_pos=[]
52          for i in range(len(word)/l):
53                  rcv, suspect = inner_decode(word[i*l:(i+1)*l], threshold)
54                  if suspect:
55                          erasure_pos.append(i)
56                          dec.extend(rcv)
57                  else:
58                          dec.extend(rcv)
59          return vector(dec),list(erasure_pos)
60
61
62  ### BINARIS "KONVERTALAS"
63  ### Adott: f, q, alpha
64
65  # a phi lekepezes
66  def push_to_binary(vec):
67          ret=[]
68          for i in vec: ret.extend(i.polynomial().padded_list(f))
69          return vector(ret)
70
71  # az inverz lekepezes
72  def pull_over_q(vec):
73          return vector(sum(alpha^i for i in range(f) if vec[j*f+i]) for j in range(len(vec)/f))
74
75
76  ### REED-SOLOMON KOD
77  ### Adott: q, k - dimenzio, n=q-1 - hossz, alpha
78
79  # q-aris Reed-Solomon torleses hibajavitas
80  def RS_error_erasure_correction(rcv,erasure_pos):
81    #torlesek helyere 0-t irunk
82    for i in erasure_pos:
83        rcv[i]=0
84
85    #v_t a torlesek szama, v_h a hibake, tudjuk, hogy 2*v_h + v_t <= n - k
86    v_t=len(erasure_pos)
87    v_h=floor((n-k-v_t)/2)
88    v=v_h + v_t
```

```
89
90      #szindromak
91      S=[sum(rcv[i]*alpha^(j*i) for i in range(n)) for j in [1..n-k]]
92      S_mat=[[S[v+i-j-1] for j in range(v)] for i in range(v_h)]
93      S_rhs=[-S[v+i] for i in range(v_h)]
94
95      #az ismert torlesi helyek alapjan:
96      Z=[alpha^erasure_pos[i] for i in range(v_t)]
97      Z_mat=[[Z[i]^(-j) for j in [1..v]] for i in range(v_t)]
98      Z_rhs=[-1 for i in range(v_t)]
99
100     #a szindromak es az ismert hibahelyek alapjan:
101     Z_S_rhs=vector(Z_rhs+S_rhs)
102     Z_S_mat=matrix(Z_mat+S_mat)
103
104     #a hibakereso polinom egyutthatoi:
105     try:
106         Lambda=Z_S_mat.solve_right(Z_S_rhs)
107     except:
108         return rcv
109
110     #hibakereso polinom:
111     def Lambda_polfun(j):
112         return 1+sum(Lambda[i]*(alpha^j)^(-i-1) for i in range(v))
113     #gyokei a hibak helyei:
114     err_pos=[j for j in range(n) if Lambda_polfun(j)==0]
115
116     if err_pos==[]:
117         return rcv
118     else:
119         X_mat=matrix([[alpha^(u*j) for u in err_pos] for j in [1..v]])
120         X_rhs=vector([S[i] for i in [0..v-1]])
121
122     try:
123         err_coeff=X_mat.solve_right(X_rhs)
124     except:
125         return rcv
126
127     for i in range(len(err_pos)):
128         rcv[err_pos[i]]-=err_coeff[i]
129     return rcv
130
131 #Reed-Solomon kodolas
132 def RS_encode_over_q(msg):
133         return vector(sum(msg[i]*alpha^(i*j) for i in range(k)) for j in range(n))
134
135 #Reed-Solomon dekodolas
136 def RS_decode_over_q(recv):
137         return vector(sum(recv[i]*alpha^((-j)*i) for i in range(n)) for j in range(k))
138
139
140 ### KONKATENALAS
141
```

```
142 # elokeszulet a kodhoz
143 def initiate_concatenated_code(inner_code_len, inner_code_dim):
144     global f, l, q, n, F, alpha
145     InnerCode_standard_array(l,f)
146     f=inner_code_dim
147     l=inner_code_len
148     q=2^f
149     n=q-1
150     F.<alpha>=GF(q)
151 
152 # msg kodolasa a konkatenalt koddal
153 def concatenated_encode(msg):
154         msg_q=pull_over_q(msg)
155         sent_q=RS_encode_over_q(msg_q)
156         sent_q_2=push_to_binary(sent_q)
157         return INNER_encode(sent_q_2)
158 
159 
160 # recv dekodolasa a konkatenalt koddal a threshold kuszobertektol fuggoen
161 def concatenated_decode_erasure(recv, threshold):
162         recv_q2_inner_code_corrected,era_pos=INNER_decode(recv, threshold)
163         recv_q_inner_code_corrected=pull_over_q(recv_q2_inner_code_corrected)
164         recv_q_rs_corrected=RS_error_erasure_correction(recv_q_inner_code_corrected,era_pos)
165         recv_q=RS_decode_over_q(recv_q_rs_corrected)
166         return push_to_binary(recv_q)
```

Irodalomjegyzék

[1] G. David Forney - *Concatenated Codes*, MIT Press, Cambridge, MA, 1966.

[2] W. Cary Huffman, Vera Pless - *Fundamentals of Error-Correcting Codes*, Cambridge University Press, 2010

[3] J. H. van Lint - *Introduction to Coding Theory*, Third Edition - Springer, 1999

[4] Andre Neubauer, Jurgen Freudenberger, Volker Kuhn - *Coding Theory: Algorithms, Architectures and Applications*, First Edition - Wiley Interscience, 2007

[5] `https://en.wikipedia.org/wiki/Concatenated_error_correction_code`, Elérés ideje: 2016.04.13.

[6] `https://en.wikipedia.org/wiki/Reed%E2%80%93Solomon_error_correction`, Elérés ideje: 2016.04.11.

[7] `http://www.sagemath.org/`, Elérés ideje: 2016.02.11.

[8] `http://www.codetables.de/`, Elérés ideje: 2016.02.10.

Köszönetnyilvánítás

Ezúton is szeretnék köszönetet mondani Dr. Nagy Gábor Péternek, hogy türelmével, hasznos tanácsaival, útmutatásaival mindvégig segítette diplomamunkám elkészülését. Külön köszönettel tartozom neki azért, hogy különös figyelmet fordított arra, hogy a kötelező tananyagokon túlmutató ismeretekre is szert tehessünk tanulmányaink során, melynek hatására választottam diplomamunka témámnak ezt a témát.

Köszönettel tartozom továbbá szüleimnek, hogy tanulmányaim során türelmükkel, megértésükkel, bizalmukkal és bátorításukkal végig mellettem álltak.

Nyilatkozat

Alulírott Judák Regina kijelentem, hogy a diplomamunkámban 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, 2016. május 12.

Judák Regina