McEliece-koodaussysteemi

Pro gradu -tutkielma
Paula Eerikinharju
Matemaattisten tieteiden tutkinto-ohjelma
Oulun yliopisto
Kevät 2023

Sisällys

1 Johdanto

Tässä tutkielmassa perehdytään McEliece-koodaussysteemiin ja sen eri käyttötarkoituksiin. Lisäksi perehdytään McEliece-koodaussysteemin pohjalta rakennettuun CFS-allekirjoitussysteemiin sekä kahteen eri tapaan hyökätä McEliece-koodaussysteemiin. McEliece-koodaussysteemi on yksi niistä koodaussysteemeistä, joita voidaan käyttää silloin, kun kvanttitietokoneet ovat käytössä. Tämän vuoksi McEliece-koodaussysteemi on tärkeä osa tulevaisuuden kryptografiaa. Tutkielman päälähteenä on käytetty kirjaa *Cryptography: Theory and Practice* (Stinson, D.R. & Paterson M.B., 2019) [10].
Ennen McEliece-koodaussysteemin määrittelyä määritellään tarpeellisia algebran, vektorilaskennan ja matriisilaskennan peruskäsitteitä luvussa 2. Luvun 2 lähteenä on käytetty Oulun Yliopiston kurssien *Algebran perusteet* ja *Matriisiteoria* luentomonisteita.
Luvussa 3 määritellään lineaarinen koodi ja systemaattinen dekoodaus. Lineaarisen koodin systemaattisesta dekoodauksesta esitetään yksi esimerkki. Luvun 3 lähteenä on käytetty Oulun Yliopiston kurssin *Johdatus koodausteoriaan* luentomonistetta.
Luvussa 4 määritellään McEliece-koodaussysteemi ja esitetään kaksi esimerkkiä McEliece-koodaussysteemistä. Esimerkit koskevat McEliece-koodaussysteemiä, joka on muodostettu Goppa-koodin parametrien mukaisesti, ja McEliece-koodaussysteemiä, joka ei ole Goppa-koodi.
Luvussa 5 käsitellään hash-funktioita ja näiden muodostamista. Hash-funktioiden ideaalista toimintaa kuvaava satunnaisoraakkeli-malli esitellään myös tässä luvussa. Luvun 5 lähteenä on käytetty päälähteen lisäksi kirjaa *Introduction to cryptography: Principles and Applications* (Delfs, H. & Knebl, H., 2007) [6].
Luvussa 6 esitellään McEliece-koodaussysteemin pohjalta muodostettu Niederreiter-koodaussysteemi ja Niederreiter-koodaussysteemin pohjalta muodostettu allekirjoitussysteemi CFS. Samalla kerrotaan yleisesti salattujen viestien allekirjoittamisesta ja allekirjoituksen tarkoituksesta. Luvun 6 lähteenä on käytetty päälähteen lisäksi artikkelia *How to Achieve a McEliece-based Digital Signature Scheme* (Courtois, N.T., Finiasz, M. & Sendrier, N., 2001) [5].
Luku 7 käsittelee McEliece-koodaussysteemin turvallisuutta. Ensin verrataan sen turvallisuutta yhden yleisimmistä salaussysteemeistä, RSA:n, turvallisuuteen. Lisäksi käsitellään McEliece-koodaussysteemin murtamista eri menetelmillä. Luvun 7 lähteinä on käytetty päälähteen lisäksi kirjaa *The theory of error correctiong codes* (MacWilliams, F.J. & Sloane, N.J.A., 1977) sekä artikkeleita *Effective attack on the McEliece cryptosystem based on Reed-Muller codes* (Borodin, M.A. & Chizhov, I.V., 2014), *Attacking and defending the*

McEliece cryptosystem (Bernstein, D.J., Lange, T. & Peters, C., 2008) ja *Formalizing O notation on Isabelle/HOL* (Avigad, J. & Donnelly, K., 2004) [4] [3] [2].

2 Peruskäsitteitä

Koodausteoriassa hyödynnetään matemaattisia peruskäsitteitä useilta eri matematiikan osa-alueilta. Suurin osa erilaisista koodausjärjestelmistä pohjautuu algebraan ja matriisiteoriaan. Tässä luvussa määritellään sellaisia algebran ja matriisiteorian käsitteitä, joita tarvitaan myöhemmin koodausteoreettisten systeemien määrittelemiseksi. Tässä luvussa on käytetty lähteenä Oulun Yliopiston kurssin *Algebran perusteet*ja *Matriisiteoria* luentomonisteita.
Ensin määritellään joitakin algebrallisia rakenteita.

2.1 Ryhmä

Ryhmä on algebrallinen rakenne, johon kuuluu yksi joukko ja yksi binäärinen operaatio. Se on yksinkertaisin seuraavaksi määriteltävistä algebrallisista rakenteista, ja myös tärkein, koska sen avulla määritellään monia rakenteeltaan monimutkaisempia algebrallisia rakenteita, kuten esimerkiksi rengas ja kunta. Yksi tutuimmista ryhmistä on esimerkiksi kokonaislukujen joukko yhteenlaskuoperaation kanssa, $(\mathbb{Z}, +)$.

Määritelmä 2.1. Olkoon A joukko ja joukko B sen osajoukko eli $B \subseteq A$. Joukon A *binäärinen operaatio* on kuvaus $* : A \times A \to A$. Jos lisäksi kuvaus $* : (x, y) \Rightarrow x * y$ toteuttaa ehdon $x * y \in B$ kaikilla $x, y \in B$, niin kuvaus $*$ on *suljettu* joukon B suhteen.

Määritelmä 2.2. Olkoon A joukko ja $*$ joukon A suhteen suljettu binäärinen operaatio. Pari $(A, *)$ on *ryhmä*, jos seuraavat ehdot toteutuvat:

1. Neutraalialkio: On olemassa sellainen alkio $e \in A$, että

   $$a * e = e * a = a$$

   kaikilla $a \in A$.

2. Käänteisalkio: Jokaisella alkiolla $a \in A$ on olemassa käänteisalkio $a^{-1} \in A$, joka toteuttaa ehdon

   $$a * a^{-1} = a^{-1} * a = e.$$

3. Assosiatiivisuus: Kaikilla alkioilla $a, b, c \in A$ pätee, että

$$(a * b) * c = a * (b * c).$$

Usein on väliä sillä, operoidaanko ryhmän operaatiolla ryhmän alkioita oikealta vai vasemmalta. Määritellään seuraavaksi ryhmä, jolle ei ole väliä alkioiden järjestyksellä operaatiossa.

Määritelmä 2.3. Olkoon $(A, *)$ ryhmä. Ryhmä on *Abelin ryhmä* silloin, kun se toteuttaa ryhmäkriteereiden lisäksi kommutatiivisuuskriteerin, eli

$$a * b = b * a$$

kaikilla $a, b \in A$.

2.2 Kunta

Kunta on algebrallinen rakenne, johon kuuluu joukon ja kahden binäärisen operaation lisäksi näiden binääristen operaatioiden vastaoperaatiot käänteisalkioiden määrittelyn myötä. Näin ollen kuntaan kuuluu periaatteessa neljä operaatiota. Esimerkiksi rationaalilukujen joukko muodostaa yhdessä yhteen- ja kertolaskuoperaatioiden kanssa kunnan. Kun verrataan kuntaa renkaaseen, huomataan, että esimerkiksi kokonaislukujen joukko ei muodosta yhdessä yhteen- ja kertolaskuoperaatioiden kanssa kuntaa, koska kokonaislukujen joukossa ei ole kaikille alkioille olemassa kertolaskun käänteisalkiota.

Määritelmä 2.4. Olkoon $\mathbb{K}$ joukko, jonka suhteen on suljetut binääriset operaatiot $+$ ja $*$. Kolmikko $(\mathbb{K}, +, *)$ on *kunta*, jos se täyttää seuraavat ehdot:

1. Pari $(\mathbb{K}, +)$ on Abelin ryhmä, jonka neutraalialkio $e = \mathbb{0}_{\mathbb{K}} \in \mathbb{K}$.
2. Pari $(\mathbb{K} \setminus \{\mathbb{0}_{\mathbb{K}}\}, *)$ on Abelin ryhmä, jonka neutraalialkio on $e = \mathbb{1}_{\mathbb{K}} \in \mathbb{K}$.
3. Joukossa $\mathbb{K}$ pätee osittelulait, eli

   $$(a + b) * c = a * c + b * c$$

   ja

   $$a * (b + c) = a * b + a * c$$

   kaikilla $a, b, c \in \mathbb{K}$.

Lause 2.5. *Jäännösluokkajoukon $\mathbb{Z}_p$ ja binääristen operaatioiden $+$ ja $\cdot$ muodostama rengas $(\mathbb{Z}_p, +, \cdot)$ on kunta, jos ja vain jos p on alkuluku.*

2.3 Vektoriavaruus

Vektoriavaruus on yksi lineaarialgebran peruskäsitteistä. Vektoriavaruus koostuu joukosta ja kahdesta operaatiosta; alkioiden summasta ja skalaarilla kertomisesta. Joukko, jonka avulla vektoriavaruus määritellään, koostuu vektoreista. Vektoriavaruuden operaatiot ovat erilaisia verrattuna edellä esiteltyjen algebrallisten rakenteiden binäärisiin operaatiohin, koska vektoriavaruus koostuu vektoreista.

Jotta voidaan määritellä vektoriavaruus, täytyy ensin määritellä vektoriavaruuden tulo-operaatio eli skalaaritulo.

Määritelmä 2.6. Olkoon ryhmä V ryhmän W, joka on Abelin ryhmä, aliryhmä, ja kunta $\mathbb{F}$ kunnan $\mathbb{K}$ alikunta. Operaatiota $\mathbb{K} \times W \to W$ sanotaan ryhmän V *skalaarituloksi* kunnan $\mathbb{F}$ suhteen, jos $fv \in V$ kaikilla $f \in \mathbb{F}$ ja $v \in V$.

Määritelmä 2.7. Olkoon ryhmä $(V, +)$ Abelin ryhmä. Ryhmä V varustettuna skalaaritulolla kunnan $(\mathbb{F}, +, \cdot)$ suhteen on $\mathbb{F}$-kertoiminen *vektoriavaruus*, jos skalaaritulo toteuttaa seuraavat ehdot kaikilla alkioilla $a, b \in \mathbb{F}$ ja $v, w \in V$:

1. $1v = v$, missä 1 on kunnan $\mathbb{F}$ ykkösalkio.
2. $(a + b)v = av + bv$.
3. $a(v + w) = av + aw$.
4. $(a \cdot b)v = a(bv)$.

Kerroinkunnan $\mathbb{F}$ alkioita kutsutaan yleensä skalaareiksi ja vektoriavaruuden V alkioita kutsutaan yleensä vektoreiksi. Jatkossa sekä vektoreiden yhteenlaskulle ja skaalaarien yhteenlaskulle käytetään yhteistä merkintää $+$, vaikka laskutoimitukset voivat olla eri joukkojen laskutoimituksia.

Lause 2.8. *$\mathbb{F}$-kertoimisen vektoriavaruuden V epätyhjä osajoukko H on vektoriavaruuden V aliavaruus, jos ja vain jos ehto*

$$av + bw \in H$$

toteutuu kaikilla $a, b \in \mathbb{F}$ ja $v, w \in H$.

Vektoriavaruuden lineaarinen vapaus on ominaisuus, jota tullaan hyödyntämään erityisesti myöhemmin koodaussysteemejä käsitellessä.

Määritelmä 2.9. Olkoon V $\mathbb{F}$-kertoiminen vektoriavaruus ja S tämän joukon osajoukko, johon kuuluu alkiot $\{s_1, s_2, \ldots, s_n\}$. Osajoukko S on *lineaarisesti vapaa*, jos yhtälöllä

$$a_1 s_1 + a_2 s_2 + \ldots + a_n s_n = 0$$

on ainoastaan yksi ratkaisu kunnassa $\mathbb{F}$, ja tämä ratkaisu on $a_1 = a_2 = \ldots = a_n = 0$. Jos vektoriavaruus ei ole lineaarisesti vapaa, se on *lineaarisesti sidottu*.

Määritelmä 2.10. Vektoriavaruuden V *dimensio* eli $\dim V$ on vektoriavaruuden V kannan vektoreiden lukumäärä. Vektoriavaruuden *kanta* on sellaisten vektoreiden joukko, joiden lineaarikombinaation avulla voidaan esittää kaikki vektoriavaruuden alkiot. Vektoriavaruuden kanta ei ole yksikäsitteinen. Jos $\dim V = n$, niin vektoriavaruuden V jokainen $n+1$ alkion osajoukko on lineaarisesti sidottu.

Määritelmä 2.11. Olkoon V ja W $\mathbb{F}$-kertoimisia vektoriavaruuksia. *Lineaarinen kuvaus* on funktio $f : V \to W$, jolle pätee seuraavat ehdot:

1. $f(v_1 + v_2) = f(v_1) + f(v_2)$ kaikilla $v_1, v_2 \in V$.
2. $f(kv) = kf(v)$ kaikilla $v \in V$ ja $k \in \mathbb{F}$.

2.4 Matriisit

Matriisi on matemaattinen tapa esittää alkioita taulukkomuodossa. Matriisin vaakarivejä kutsutaan riveiksi ja pystyrivejä sarakkeiksi. Matriisien avulla voidaan esittää lineaarisia kuvauksia ja yhtälöryhmiä kätevästi.

Matriiseja käsitellessä perinteiset laskutoimitukset on tarpeen määritellä uudestaan. Määritellään aluksi matriisien laskutoimitukset, summa ja tulo.

Määritelmä 2.12. Olkoon A ja B $(m \times n)$-matriiseja muotoa

$$A = \begin{bmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ a_{21} & a_{22} & \ldots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \ldots & a_{mn} \end{bmatrix} \in \mathbb{F}_{m \times n}$$

ja

$$B = \begin{bmatrix} b_{11} & b_{12} & \ldots & b_{1n} \\ b_{21} & b_{22} & \ldots & b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ b_{m1} & b_{m2} & \ldots & b_{mn} \end{bmatrix} \in \mathbb{F}_{m \times n}.$$

Matriisien A ja B *summamatriisi* $A + B \in \mathbb{F}_{m\times n}$ on muotoa

$$A + B = \begin{bmatrix} a_{11} + b_{11} & a_{12} + b_{12} & \dots & a_{1n} + b_{1n} \\ a_{21} + b_{21} & a_{22} + b_{22} & \dots & a_{2n} + b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} + b_{m1} & a_{m2} + b_{m2} & \dots & a_{mn} + b_{mn} \end{bmatrix}.$$

Määritelmä 2.13. Olkoon $A = (a_{ij})_{m\times n}$ ja $B = (b_{ij})_{n\times s}$. Tällöin matriisien A ja B *tulomatriisi* on matriisi

$$A_{m\times n} \cdot B_{n\times s} = (AB)_{m\times s} = (c_{ij})_{m\times s} \ ,$$

missä

$$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}$$

eli alkio c_{ij} saadaan matriisin A i:nnen vaakarivin ja matriisin B j:nnen pystyrivin pistetulona. Siis

$$AB = \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{pmatrix}_{m\times n} \cdot \begin{pmatrix} b_{11} & \cdots & b_{1s} \\ \vdots & \ddots & \vdots \\ b_{n1} & \cdots & b_{ns} \end{pmatrix}_{n\times s}$$

$$= \begin{pmatrix} \sum_{k=1}^{n} a_{1k} b_{k1} & \cdots & \sum_{k=1}^{n} a_{1k} b_{ks} \\ \vdots & \ddots & \vdots \\ \sum_{k=1}^{n} a_{mk} b_{k1} & \cdots & \sum_{k=1}^{n} a_{mk} b_{ks} \end{pmatrix}_{m\times s}.$$

Esimerkki 2.14. Matriisien $A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$ ja $B = \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix}$ tulo on matriisi

$$AB = \begin{bmatrix} 1\cdot 5 + 2\cdot 7 & 1\cdot 6 + 2\cdot 8 \\ 3\cdot 5 + 4\cdot 7 & 3\cdot 6 + 4\cdot 8 \end{bmatrix} = \begin{bmatrix} 19 & 22 \\ 43 & 50 \end{bmatrix}.$$

Määritelmä 2.15. *Nollamatriisi*, jota yleensä merkitään 0, on matriisi, jonka jokainen alkio on nolla-alkio. *Identiteettimatriisi* on $(n \times n)$-matriisi, joka on muotoa

$$I_n = \begin{bmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 \end{bmatrix} \in \mathbb{F}_{n\times n},$$

missä 1 on ykkösalkio.

Lause 2.16. *Nollamatriisille* $0 \in \mathbb{F}_{m\times n}$ *pätee kaikilla matriiseilla* $A \in \mathbb{F}_{m\times n}$, *että*

$$A + 0 = A.$$

Identiteettimatriisille $I_m \in \mathbb{F}_{m\times m}$ *pätee kaikilla matriiseilla* $A \in \mathbb{F}_{m\times n}$, *että*

$$I_m A = A.$$

Määritelmä 2.17. Matriisin $A = (a_{ij})_{m\times n} \in \mathbb{F}_{m\times n}$ *transpoosi* on matriisi $A^T \in \mathbb{F}_{n\times m}$, joka muodostetaan vaihtamalla alkuperäisen matriisin vaakarivit pystyriveiksi ja pystyrivit vaakariveiksi. Näin ollen matriisin A alkiot ovat samoja kuin matriisin A transpoosin alkiot, joiden pysty- ja vaakarivien paikkojen indeksit on vaihdettu keskenään, eli $a_{ij} = a^T_{ji}$.

Lause 2.18. *Matriisille* $A \in \mathbb{F}_{m\times n}$ *ja sen transpoosille pätee, että*

$$(A^T)^T = A.$$

Määritelmä 2.19. Olkoon A $(n \times n)$-matriisi. Sanotaan, että matriisi on *kääntyvä*, jos on olemassa sellainen $(n \times n)$-matriisi B, että sen ja matriisin A tulo tuottaa aina identiteettimatriisin, eli

$$AB = BA = I_n.$$

Matriisia B kutsutaan tällöin matriisin A *käänteismatriisiksi*.

Määritelmä 2.20. n-alkion *permutaatio* π on bijektiivinen kuvaus

$$\pi : \{1, 2, \ldots, n\} \to \{1, 2, \ldots, n\},$$

jonka voi esittää $(2 \times n)$-matriisina

$$\begin{bmatrix} 1 & 2 & \ldots & n \\ \pi(1) & \pi(2) & \ldots & \pi(n) \end{bmatrix}.$$

Tämän matriisin *permutaatiomatriisi* P_π on $(n \times n)$-matriisi, jonka alkiot ovat kaikkialla muualla nollia, paitsi ykkösiä $\pi(i)$ osoittamilla paikoilla, eli

$$P_\pi = \begin{bmatrix} e_{\pi(1)} \\ e_{\pi(2)} \\ \vdots \\ e_{\pi(n)} \end{bmatrix}.$$

Esimerkki 2.21. Olkoon neljän alkion permutaation π (2×4)-matriisi muotoa

$$\pi = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 3 & 1 & 4 & 2 \end{bmatrix}.$$

Tällöin permutaatiomatriisi P_π on (4×4)-matriisi muotoa

$$P_\pi = \begin{bmatrix} e_{\pi(1)} \\ e_{\pi(2)} \\ e_{\pi(3)} \\ e_{\pi(4)} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{bmatrix}.$$

Kun permutaatiomatriisilla P_π kerrotaan permutaation π alkukuvariviä vasemmalta puolelta, saadaan permutaation π kuvarivi. Merkitään permutaation π alkukuvarivin transpoosia π_1^T ja kuvarivin transpoosia π_2^T. Tämän esimerkin tapauksessa siis

$$P_\pi \pi_1^T = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix} = \begin{bmatrix} 3 \\ 1 \\ 4 \\ 2 \end{bmatrix} = \pi_2^T.$$

Huomautus 2.22. Käytännössä permutaatiomatriisi on identiteettimatriisi, jonka rivit on sekoitettu.

Lause 2.23. *Permutaatiomatriisin P_π transpoosi on sama kuin sen käänteismatriisi.*

Todistus. Olkoon $(n \times n)$-permutaatiomatriisi P_π muotoa

$$P_\pi = \begin{bmatrix} e_{\pi(1)} \\ e_{\pi(2)} \\ \vdots \\ e_{\pi(n)} \end{bmatrix}$$

ja sen transpoosi muotoa

$$P_\pi^T = \begin{bmatrix} e_{\pi(1)} & e_{\pi(2)} & \dots & e_{\pi(n)} \end{bmatrix}.$$

Kun otetaan näiden tulo, saadaan

$$P_\pi \cdot P_\pi^T = \begin{bmatrix} e_{\pi(1)} \\ e_{\pi(2)} \\ \vdots \\ e_{\pi(n)} \end{bmatrix} \cdot \begin{bmatrix} e_{\pi(1)} & e_{\pi(2)} & \dots & e_{\pi(n)} \end{bmatrix} = \begin{bmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 \end{bmatrix} = I_n.$$

Koska permutaatiomatriisin P_π ja sen transpoosin P_π^T tulo on identiteettimatriisi I_n, ne ovat toistensa käänteismatriiseja. □

3 Lineaarinen koodi

Salaussysteemin tarkoitus on lähettää ja vastaanottaa viestejä siten, että ne eivät näy tai käy järkeen muille kuin viestin lähettäjälle ja vastaanottajalle ja että viestin vastaanottaja pystyy tulkitsemaan viestin lähettäjän tarkoittamalla tavalla. Koodaussysteemi toimii niin, että viestin lähettäjä koodaa viestin, jonka hän sitten lähettää vastaanottajalle. Viestin vastaanottaja dekoodaa viestin, jolloin hän pääsee lukemaan viestin lähettäjän tarkoittamassa muodossa. Tässä luvussa on käytetty lähteenä Oulun Yliopiston kurssin *Johdatus koodausteoriaan* luentomonistetta.

3.1 Lineaarinen koodi

Määritelmä 3.1. Olkoon k ja n positiivisia kokonaislukuja siten, että $k \leq n$. *Lineaarinen* $[n, k]$*-koodi* C on vektoriavaruuden $\mathbb{Z}_2^n$ osajoukko, jonka dimensio on k. Koodi C koostuu siis n-pituisista binäärivektoreista, eli vektoreista, joiden kaikki alkiot kuuluvat joukkoon $\{0, 1\}$.

Määritelmä 3.2. Olkoon C lineaarinen $[n, k]$-koodi. Koodin C *generoijamatriisi* G on $(k \times n)$-matriisi, jonka rivit muodostavat koodin C kannan.

Generoijamatriisi G määrittelee koodiin C kuuluvat koodisanat c. Koodisana saadaan, kun kerrotaan vektoria x generoijamatriisilla G. Kaikki koodin C koodisanat c saadaan määritettyä kertomalla kaikki mahdolliset k-mittaiset binäärivektorit tunnetulla generoijamatriisilla G.

Määritelmä 3.3. Olkoon C lineaarinen $[n, k]$-koodi. Koodin C *tarkistusmatriisi* H on $((n - k) \times n)$-matriisi, jolle pätee, että

$$C = \{x \in \mathbb{F}^n \mid xH^T = 0\}.$$

Tarkistusmatriisin H tarkoitus on tarkistaa, kuuluuko jokin vektori koodiin C. Jos vektorin x ja tarkistusmatriisin H tulo tuottaa nollavektorin, vektori x on koodisana. Jos tulo tuottaa jotain muuta kuin nollavektorin, vektori x ei ole koodisana. Tarkistusmatriisin määritelmästä seuraa, että jokaisen generoijamatriisin G rivin tulo jokaisen tarkistusmatriisin H rivin transpoosin kanssa tuottaa aina nolla-alkion.

Määritelmä 3.4. Olkoon vektorit $x, y \in \mathbb{Z}_2^n$, missä $x = (x_1, x_2, \ldots, x_n)$ ja $y = (y_1, y_2, \ldots, y_n)$. Vektoreiden x ja y välinen *Hamming-etäisyys* $d(x, y)$ on niiden indeksien lukumäärä, joiden kohdalla vektoreiden x ja y alkiot eroavat toisistaan, eli

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

Lineaarisia koodeja käsitellessä ollaan usein kiinnostuneita vektoreiden välisestä etäisyydestä etenkin silloin, kun se on pienin mahdollinen.

Määritelmä 3.5. Olkoon C lineaarinen $[n, k]$-koodi. Koodin C *minimietäisyys* $d_{min}C$ on sellaisten vektoreiden x ja y Hamming-etäisyys, joille tämä etäisyys on kaikista kahden vektoreiden yhdistelmistä pienin, eli

$$d_{\min}C = \min\{d(x, y) \mid x, y \in C, x \neq y\}.$$

Lineaarinen $[n, k, d]$-koodi on $[n, k]$-koodi, jonka minimietäisyys on d.

Lause 3.6. *Olkoon C lineaarinen $[n, k]$-koodi, jonka tarkistusmatriisi on H. Koodin C minimietäisyys $d_{min}C = d$, jos ja vain jos jokainen tarkistusmatriisin H $(d-1)$:n sarakkeen joukko on lineaarisesti vapaa ja löytyy sellainen matriisin H d:n sarakkeen joukko, joka on lineaarisesti sidottu.*

Määritelmä 3.7. Olkoon x ja y joukon $\mathbb{Z}_2^n$ vektoreita. Vektorit $x = (x_1, x_2, \ldots, x_n)$ ja $y = (y_1, y_2, \ldots, y_n)$ ovat *ortogonaalisia*, jos niille pätee, että

$$\sum_{i=1}^{n} x_i y_i \equiv 0 \pmod 2.$$

Määritelmä 3.8. Olkoon C lineaarinen $[n, k]$-koodi. Koodin C *duaalikoodi* $C^\perp$ on koodi, joka koostuu vektoreista, jotka ovat ortogonaalisia kaikkien koodin C vektoreiden kanssa, eli

$$C^\perp = \{y \in \mathbb{Z}_2^n \mid x \cdot y^T = 0 \text{ kaikilla } x \in C\}.$$

Duaalikoodin $C^\perp$ generoijamatriisi on sama kuin alkuperäisen koodin C tarkistusmatriisi.

Koodausjärjestelmässä voidaan dekoodata viestejä syndromien avulla.

Määritelmä 3.9. Olkoon C lineaarinen $[n, k]$-koodi, jonka tarkistusmatriisi on H. Tällöin vektorin x *syndromi* on vektori

$$s(x) = xH^T.$$

Syndromi $s(x)$ on nollavektori, jos ja vain jos vektori x on koodin C koodisana, eli $x \in C$.

Syndromien avulla voidaan määrittää se, mikä virhevektori on mukana vastaanotetussa viestissä, kun tunnetaan tarkistusmatriisi ja kaikki mahdolliset virhevektorit. Tällöin on helppo määrittää se koodisana, jonka viestin lähettäjä on halunnut vastaanottajan ottavan vastaan.
Lähimpään naapuriin dekoodaus tarkoittaa sitä, että vastaanotetun viestin y sijaan dekoodataan koodisana c, joka on minimietäisyyden d päässä vastaanotetusta viestistä y.

Lause 3.10. *Olkoon C lineaarinen $[n, k, d]$-koodi. Jos viestissä esiintyy maksimissaan $\frac{d-1}{2}$ virhettä, lähimpään naapuriin dekoodaus korjaa kaikki viestissä esiintyvät virheet.*

Määritelmä 3.11. Olkoon C lineaarinen $[n, k]$-koodi, jonka generoijamatriisi on G. Jos generoijamatriisi on muotoa

$$G = \begin{bmatrix} I_k & P \end{bmatrix},$$

missä matriisi I_k on $(k \times k)$-identiteettimatriisi ja matriisi P on $(k \times (n-k))$-matriisi, on koodi C *systemaattinen koodi*. Systemaattisessa koodissa koodisanassa c k-ensimmäistä alkiota muodostavat koodatun viestisanan m ja loput alkiot ovat tarkistussymboleja.

On olemassa systemaattisia koodeja, joissa koodatun viestisanan m voi myös lukea muusta kohtaa koodisanaa c kuin sen alusta.

Esimerkki 3.12. Olkoon C lineaarinen $[5, 2]$-koodi, jonka generoijamatriisi G on

$$G = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \end{bmatrix}$$

ja tarkistusmatriisi H on

$$H = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 \end{bmatrix}.$$

Matriisi H on tarkistusmatriisi generoijamatriisille G, koska

$$GH^T = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 1 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1+1 & 1+1 & 1+1 \\ 1+1 & 0 & 1+1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$

Tällöin viestiä $x = \begin{bmatrix} 1 & 0 \end{bmatrix}$ vastaava koodisana on

$$xG = \begin{bmatrix} 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 \end{bmatrix}.$$

Tarkistetaan tarkistusmatriisin transpoosin H^T avulla, että vektori $y = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 \end{bmatrix}$ on koodisana. Koska

$$yH^T = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 1 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1+1 & 1+1 & 1+1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix},$$

niin vektori y on koodisana. Jos tarkistusmatriisin transpoosilla H^T kertomisen tuloksena olisi tullut mitä tahansa muuta kuin nollavektori, niin vektori y ei olisi koodisana.

Viestiin tapahtuu lähetyksessä virhe, jota voidaan kuvata virhevektorilla $e = \begin{bmatrix}0 & 1 & 0 & 0 & 0\end{bmatrix}$. Tällöin vastaanottaja vastaanottaa vektorin $y_1 = y + e = \begin{bmatrix}1 & 0 & 0 & 1 & 0\end{bmatrix}$.

Määritetään seuraavaksi mahdollisten virhevektoreiden syndromit. Koska yksikään tarkistusmatriisin H sarakkeista ei ole nollavektori tai ole keskenään samoja, jokainen kahden sarakkeen joukko on lineaarisesti riippumaton. Koska

$$\begin{bmatrix}1\\0\\1\end{bmatrix} = \begin{bmatrix}1\\0\\0\end{bmatrix} + \begin{bmatrix}0\\0\\1\end{bmatrix},$$

on olemassa sellainen kolmen sarakkeen joukko, joka on lineaarisesti riippuvainen. Tällöin siis koodin C minimietäisyys $d_{min}C = 3$. Koodin virheenkorjauskyky on tällöin

$$\frac{d-1}{2} = \frac{3-1}{2} = 1.$$

Painoa yksi olevat mahdolliset virhevektorit ovat tällöin $\begin{bmatrix}1 & 0 & 0 & 0 & 0\end{bmatrix}$, $\begin{bmatrix}0 & 1 & 0 & 0 & 0\end{bmatrix}$, $\begin{bmatrix}0 & 0 & 1 & 0 & 0\end{bmatrix}$, $\begin{bmatrix}0 & 0 & 0 & 1 & 0\end{bmatrix}$ ja $\begin{bmatrix}0 & 0 & 0 & 0 & 1\end{bmatrix}$. Lasketaan näiden virhevektorien syndromit

$$\begin{bmatrix}1 & 0 & 0 & 0 & 0\end{bmatrix} \cdot H^T = \begin{bmatrix}1 & 0 & 1\end{bmatrix}, \begin{bmatrix}0 & 1 & 0 & 0 & 0\end{bmatrix} \cdot H^T = \begin{bmatrix}0 & 1 & 0\end{bmatrix},$$
$$\begin{bmatrix}0 & 0 & 1 & 0 & 0\end{bmatrix} \cdot H^T = \begin{bmatrix}1 & 0 & 0\end{bmatrix}, \begin{bmatrix}0 & 0 & 0 & 1 & 0\end{bmatrix} \cdot H^T = \begin{bmatrix}1 & 1 & 1\end{bmatrix} \text{ ja}$$
$$\begin{bmatrix}0 & 0 & 0 & 0 & 1\end{bmatrix} \cdot H^T = \begin{bmatrix}0 & 0 & 1\end{bmatrix}.$$

Vastaanotetun viestin y syndromi on

$$yH^T = \begin{bmatrix}1 & 0 & 0 & 1 & 0\end{bmatrix} \cdot H^T = \begin{bmatrix}0 & 1 & 0\end{bmatrix} = \begin{bmatrix}0 & 1 & 0 & 0 & 0\end{bmatrix} \cdot H^T,$$

eli saatuun viestiin on tapahtunut virhe, jota kuvaa virhevektori $\begin{bmatrix}0 & 1 & 0 & 0 & 0\end{bmatrix}$. Lähetetty koodisana on tällöin

$$y = y_1 - \begin{bmatrix}0 & 1 & 0 & 0 & 0\end{bmatrix} = \begin{bmatrix}1 & 1 & 0 & 1 & 0\end{bmatrix}.$$

Koodin ollessa systemaattinen, eli generoijamatriisi G on muotoa $\begin{bmatrix}P & I_k\end{bmatrix}$, missä matriisi P on $(k \times (n-k))$-matriisi ja matriisi I_k on $(k \times k)$-identiteetti-

matriisi, vastaanotettu viesti voidaan lukea dekoodatun koodisanan lopusta. Näin ollen vastaanotettu viesti on $x = \begin{bmatrix} 1 & 0 \end{bmatrix}$. Huomataan, että saatiin sama viesti, joka alunperin koodattiin.

Lineaarinen koodi C ei ole aina systemaattinen, koska generoijamatriisi G ei aina ole muotoa $\begin{bmatrix} P & I_k \end{bmatrix}$. Koska generoijamatriisi G ei ole yksikäsitteinen, voidaan valita generoijamatriisiksi G sellainen koodin C kantavektoreista muodostuva matriisi, jossa generoijamatriisiin G sisältyy identiteettimatriisi I_k. Yksinkertaisimmillaan tällainen matriisi saadaan muodostettua muuttamalla alkuperäisen generoijamatriisin G rivien paikkoja niin, että generoijamatriisiin G sisältyy identiteettimatriisi I_k. Generoijamatriisin G tällä tavoin muunnettu versio G' generoi tällöin lineaarisen koodin C', joka on systemaattinen.
Koodin C muuntaminen joksikin systemaattiseksi koodiksi ei ole aina näin yksinkertaista. Seuraavaksi käydään läpi tilanne, jossa generoijamatriisin G rivien vaihtaminen keskenään ei riitä jonkin systemaattisen koodin generoimatriisin muodostamiseksi.

Määritelmä 3.13. Olkoon C_1 ja C_2 lineaarisia $[n, k]$-koodeja, joiden generoijamatriisit ovat G_1 ja G_2. Koodeja C_1 ja C_2 sanotaan *ekvivalenteiksi*, jos koodin C_1 generoijamatriisi G_1 saadaan koodin C_2 generoijamatriisista G_2 muuttamalla sarakkeiden järjestystä. Tällöin merkitään $C_1 \sim C_2$.

Joskus ekvivalentin koodin muodostaminen riittää siihen, että saadaan aikaan systemaattisen koodin generoijamatriisi. Aina ei kuitenkaan saada aikaiseksi systemaattista koodia pelkästään vaihtamalla sarakkeiden järjestystä. Generoijamatriisia G voidaan kuitenkin operoida elementaarisilla vaakarivimuunnoksilla, koska elementaariset vaakarivimuunnokset eivät muuta generoijamatriisin G rivien virittämää aliavaruutta. Elementaarisilla vaakarivimuunnoksilla saadaan generoijamatriisi G sellaiseen muotoon, että sarakkeiden järjestystä muuttamalla saadaan aikaan generoijamatriisi G', joka generoi sellaisen systemaattisen koodin C', joka on ekvivalentti alkuperäisen koodin C kanssa. Näin ollen jokaista ei-systemaattista koodia C vastaa jokin systemaattinen koodi C'.

4 McEliece-koodaussysteemi

4.1 Viestien lähettäminen ja vastaanottaminen

Kun halutaan lähettää sellaisia viestejä, joiden halutaan pysyvän salassa kaikille muille paitsi vastaanottajalle, täytyy lähetettävä viesti salata sellaisella tavalla, että vastaanottaja ymmärtää viestin. Tätä varten viestiä lähetettävään järjestelmään kuuluu erilaisia avaimia. Julkinen avain on sellainen, joka on näkyvillä kaikille, ja salainen avain on jokaisen oma henkilökohtainen avain. Julkisen avaimen on oltava ominaisuuksiltaan sellainen, että sen avulla on hyvin vaikeaa päätellä mitään lähetettävistä viesteistä tai murtaa käytössä olevaa viestien salausjärjestelmää. Salaisten avaimien taas olisi hyvä olla ominaisuuksiltaan sellaisia, että viestejä voidaan avata ja salata systeemin sisällä tehokkaasti, eli mahdollisimman pienellä määrällä yksittäisiä laskutoimituksia.
Kun lähetetään salattuja viestejä erilaisia kanavia pitkin, voi viestiin tulla kohinan tai muun häiriön vuoksi virheitä. Koodaussysteemien avulla on mahdollista salata, lähettää ja avata viestejä niin, että mahdolliset virheet korjaantuvat viestejä avatessa. Koodin ominaisuuksiin kuuluu sen virheenkorjauskyky, mikä tarkoittaa enimmäismäärää virheitä, jotka koodausjärjestelmä pystyy korjaamaan. Jos viestin lähetyksessä tapahtuu virheitä koodin virheenkorjauskykyä suurempi määrä, viestiä ei voida avata.
Lähimpään naapuriin dekoodaus perustuu siihen, että viestin vastaanottaja löytää koodisanan, joka on minimietäisyyden päässä vastaanotetusta viestistä. Vastaanotetun viestin sijaan dekoodataan tämä koodisana, ja toivotaan, että lähetyksessä mahdollisesti tapahtuneet virheet korjaantuvat. Usein koodissa on useampia eri koodisanoja, jotka ovat minimietäisyyden päässä vastaanotetusta viestistä. Tällöin ei ole väliä, mikä koodisana valitaan vastaanotetun viestin tilalle, koska dekoodausalgoritmin suorittamisen jälkeen jäljelle jää oikea alkuperäinen viesti, kunhan valittu koodisana on minimetäisyyden päässä vastaanotetusta viestistä.

4.2 Goppa-koodi

Goppa-koodi on koodi, jonka parametrit n, k ja d noudattavat tiettyjä kaavoja. McEliece-koodaussysteemiä suositellaan käytettäväksi Goppa-koodien kanssa, koska Goppa-koodien dekoodausalgoritmit ovat erityisen tehokkaita, Goppa-koodeja on suhteellisen helppo muodostaa ja on olemassa paljon sellaisia Goppa-koodeja, jotka eivät ole ekvivalentteja ja joiden parametrit ovat samat.

Määritelmä 4.1. *Goppa-koodi* on $[n, k, d]$-koodi, jonka parametrit ovat muotoa $n = 2^m$, $k = n - mt$ ja $d = 2t + 1$ joillekin kokonaisluvuille m ja t.

4.3 McEliece-koodaussysteemi

McEliece-koodaussysteemi on koodaussysteemi, joka perustuu lähimpään naapuriin dekoodaukseen. Lisäksi käytetään matriisilaskentaa julkisen avaimen muodostamiseen. McEliece-koodaussysteemin turvallisuus perustuu siihen, että julkisesta avaimesta on hyvin vaikeaa päätellä koodin C generoijamatriisia G, minkä vuoksi on hyvin vaikeaa päätellä koodisana, joka olisi lähellä haluttua vektoria.

Määritelmä 4.2. Olkoon C lineaarinen $[n, k, d]$-koodi, jonka generoijamatriisi on matriisi G. Olkoon S $(k \times k)$-matriisi, joka on kääntyvä kunnassa $\mathbb{Z}_2$. Olkoon lisäksi P $(n \times n)$-permutaatiomatriisi, $G' = SGP, \mathcal{P} = \mathbb{Z}_2^k$ ja $\mathcal{C} = \mathbb{Z}_2^n$, jolloin voidaan määritellä *McEliece-koodaussysteemi*

$$\mathcal{K} = \{(G, S, P, G')\},$$

missä G' on julkinen avain ja G, S ja P muodostavat yksityisen avaimen. Viesti $x \in \mathbb{Z}_2^k$ koodataan tällöin seuraavasti julkisen avaimen G' avulla:

$$y = xG' + e,$$

missä $e \in \mathbb{Z}_2^n$ on virhevektori.

Yleisesti suositellaan, että McEliece-koodaussysteemiä käytettäessä valittaisiin käytettäväksi Goppa-koodeja, koska niiden dekoodaaminen on tehokasta ja nopeaa erityisesti McEliece-koodaussysteemin tapauksessa. McEliece itse alunperin suositteli käytettäväksi muuttujien m ja t arvoja $m = 10$ ja $t = 50$, jolloin saadaan Goppa-koodin parametrien arvoiksi

$$n = 2^{10} = 1024,$$

$$k = n - mt = 1024 - 50 \cdot 10 = 524$$

ja

$$d = 2t + 1 = 2 \cdot 50 + 1 = 101.$$

Tällöin jokainen viesti on 524-mittainen binäärivektori, koodattu viesti 1024-mittainen binäärivektori ja julkinen avain (524×1024)-binäärimatriisi. Nykyään suositellaan käytettäväksi suurempia parametrien m ja t arvoja tietoturvasyistä. (Delfs, H. & Knebl, H., 2007)

Lause 4.3. *Olkoon $y \in \mathbb{Z}_2^n$ vastaanotettu viesti, joka on koodattu McEliece-koodaussystemillä, jonka julkinen avain on matriisi G' ja yksityinen avain muodostuu matriiseista G, S ja P. Viesti y dekoodataan tällöin seuraavan algoritmin mukaisesti:*

1. *Määritetään $y_1 = yP^{-1}$.*
2. *Dekoodataan y_1, jolloin saadaan $y_1 = x_1 + e_1$, missä $x_1 \in C$.*
3. *Määritetään $x_0 \in \mathbb{Z}_2^k$ siten, että $x_0G = x_1$.*
4. *Määritetään $x = x_0S^{-1}$.*

Todistus. Osoitetaan, että dekoodausalgoritmi toimii.
Alkuperäinen viestivektori x on koodattu laskemalla

$$y = xG' = xSGP.$$

Kun viesti lähetetään, lähetyksessä tapahtuu virhe, jota kuvaa virhevektori e. Vastaanottaja saa viestin siis muodossa

$$y = xSGP + e.$$

Kun suoritetaan dekoodauksen ensimmäinen vaihe, saadaan

$$y_1 = yP^{-1} = (xSGP + e)P^{-1} = xSGPP^{-1} + eP^{-1} = xSG + eP^{-1}.$$

Merkitään vektoria $xSG = x_1$ ja vektoria $eP^{-1} = e_1$. Koska vektori e_1 on virhevektori e kerrottuna permutaatiomatriisilla, virheiden määrä vektorissa ei muutu, vaan ainoastaan niiden paikka vaihtuu. Näin ollen vektori x_1 löydetään lähimpään naapuriin dekoodaamisen periaatteella, kun löydetään sellainen koodisana $c \in C$, joka on koodin C virheenkorjauskyvyn päässä vektorista y_1. Tällöin voidaan merkitä $x_1 = c$. Koska virhevektorin e ja siten myös vektorin e_1 paino on korkeintaan koodin virheenkorjauskyvyn verran, on löydetty koodisana yksikäsitteinen.
Kun löydetään vektori x_0, joka toteuttaa ehdon

$$x_1 = x_0G,$$

saadaan eliminoitua matriisilla G kertominen, eli $x_0 = xS$. Vektori x_0 löytyy esimerkiksi yhtälöryhmän avulla, joka ratkeaa, koska vektorin x_0 pituus on sama kuin generoijamatriisin G rivien lukumäärä, tai käänteismatriisin G^{-1} avulla, jos generoijamatriisi G on neliömatriisi. Matriisin S ollessa kääntyvä, tälle löytyy aina käänteismatriisi S^{-1}. Käänteismatriisin avulla saadaan

$$x_0S^{-1} = xSS^{-1} = x,$$

mikä on alkuperäinen viesti. (Delfs, H. & Knebl, H., 2007) □

Esimerkki 4.4. Olkoon C lineaarinen $[8, 2, 5]$-koodi. Koodi C on Goppa-koodi parametrein $m = 3$ ja $t = 2$, koska $n = 8 = 2^3$, $k = n - mt = 8 - 3 \cdot 2 = 2$ ja $d = 2t + 1 = 2 \cdot 2 + 1 = 5$. Koodin C generoijamatriisi G on

$$G = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}.$$

Muodostetaan McEliece-koodaussysteemi valitsemalla (2×2)-matriisi S siten, että

$$S = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}$$

on kääntyvä matriisi, ja valitsemalla (8×8)-permutaatiomatriisi P siten, että

$$P = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

Tällöin julkinen avain G' on (2×8)-matriisi, joka saadaan laskemalla

$$G' = SGP = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}.$$

Koodataan viesti $x = \begin{bmatrix} 1 & 1 \end{bmatrix} \in \mathbb{Z}_2^2$ käyttäen julkista avainta G', jolloin saadaan

$$xG' = \begin{bmatrix} 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \end{bmatrix}.$$

Oletetaan, että viestin lähetyksessä tapahtuu virhe kahdessa kohdassa viestiä niin, että virhe kuvautuu vektorina $e = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \in \mathbb{Z}_2^8$ lähetettävään viestiin. Vastaanottaja saa viestin muodossa

$$\begin{aligned} y = xG' + e &= \begin{bmatrix} 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \end{bmatrix} + \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \\ &= \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \end{bmatrix}. \end{aligned}$$

Vastaanottaja dekoodaa viestin dekoodausalgoritmin mukaisesti. Ensimmäistä vaihetta varten tarvitaan permutaatiomatriisin P käänteismatriisi P^{-1}. Kun tiedetään, että permutaatiomatriisin käänteismatriisi on permutaatiomatriisin transpoosi, saadaan

$$\begin{aligned} y_1 = yP^{-1} = yP^T &= \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}^T \\ &= \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \end{bmatrix}. \end{aligned}$$

Tiedetään, että koodin C minimetäisyys $d = 5$, jolloin löytämällä vektoria y_1 lähinnä oleva koodin C alkio c voidaan korjata enintään

$$\frac{d-1}{2} = \frac{5-1}{2} = 2$$

virhettä. Koodissa C on koodisana $c = \begin{bmatrix}1 & 0 & 0 & 1 & 1 & 0 & 1 & 0\end{bmatrix}$, jonka Hamming-etäisyys vektoriin y_1 on $d(y_1, c) = 2$, jolloin voidaan merkitä vektoria $c = x_1$. Seuraavaksi etsitään vektori $x_0 = \begin{bmatrix}x_{0_1} & x_{0_2}\end{bmatrix} \in \mathbb{Z}_2^2$, joka toteuttaa ehdon

$$x_1 = x_0 G.$$

Koska generoijamatriisi G ei ole kääntyvä matriisi, ratkaistaan vektori x_0 yhtälöryhmän avulla. Lasketaan vektorin x_0 ja matriisin G tulo, jolloin saadaan yhtälöryhmä

$$\begin{aligned}
&x_1 = x_0 G \\
&\quad \iff \begin{bmatrix}1 & 0 & 0 & 1 & 1 & 0 & 1 & 0\end{bmatrix} = \begin{bmatrix}x_{0_1} & x_{0_2}\end{bmatrix} \cdot \begin{bmatrix}1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1\end{bmatrix} \\
&\quad \iff \begin{bmatrix}x_{0_1} & x_{0_2} & 0 & x_{0_1} & x_{0_1} + x_{0_2} & 0 & x_{0_1} & x_{0_2}\end{bmatrix} \\
&\quad \iff \begin{cases} x_{0_1} = 1 \\ x_{0_2} = 0 \\ x_{0_1} + x_{0_2} = 1 \end{cases} \iff \begin{cases} x_{0_1} = 1 \\ x_{0_2} = 0 \end{cases}.
\end{aligned}$$

Yhtälöryhmän ratkaisuna saatiin vektori $x_0 = \begin{bmatrix}1 & 0\end{bmatrix}$. Algoritmin viimeistä vaihetta varten tarvitaan matriisin S käänteismatriisi. Koska tässä tapauksessa matriisi S on melko pieni, saadaan helposti esimerkiksi yhtälöryhmän avulla käänteismatriisiksi

$$S^{-1} = \begin{bmatrix}1 & 1 \\ 1 & 0\end{bmatrix}.$$

Käänteismatriisin avulla saadaan

$$x = x_0 S^{-1} = \begin{bmatrix}1 & 0\end{bmatrix} \cdot \begin{bmatrix}1 & 1 \\ 1 & 0\end{bmatrix} = \begin{bmatrix}1 & 1\end{bmatrix},$$

mikä on nyt dekoodattu viesti. Huomataan myös, että viesti on sama kuin alun perin koodattu viesti.

McEliece-koodaussysteemiä voidaan käyttää, vaikka koodi C ei olisi Goppa-koodi. Seuraavassa esimerkissä näytetään miten McEliece-koodaussysteemi toimii, kun koodi ei ole Goppa-koodi.

Esimerkki 4.5. Olkoon C lineaarinen $[8, 5, 3]$-koodi. Koodin C generoijamatriisi G on

$$G = \begin{bmatrix}0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1\end{bmatrix}.$$

Muodostetaan McEliece-koodaussysteemi valitsemalla (5×5)-matriisi S siten, että

$$S = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 \end{bmatrix}$$

on kääntyvä matriisi, ja valitsemalla (8×8)-permutaatiomatriisi P siten, että

$$P = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}.$$

Tällöin julkinen avain G' on (5×8)-matriisi, joka saadaan laskemalla

$$G' = SGP = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \end{bmatrix}.$$

Koodataan viesti $x = \begin{bmatrix} 0 & 1 & 1 & 1 & 0 \end{bmatrix} \in \mathbb{Z}_2^5$ käyttäen julkista avainta G',

jolloin saadaan

$$xG' = \begin{bmatrix}0 & 1 & 1 & 1 & 0\end{bmatrix} \cdot \begin{bmatrix}0 & 1 & 1 & 0 & 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0\\ 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1\\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1\\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0\end{bmatrix} = \begin{bmatrix}1 & 1 & 1 & 0 & 1 & 0 & 0 & 0\end{bmatrix}.$$

Oletetaan, että viestin lähetyksessä tapahtuu virhe yhdessä kohdassa viestiä niin, että virhe kuvautuu vektorina $e = \begin{bmatrix}0 & 0 & 0 & 0 & 0 & 1 & 0 & 0\end{bmatrix} \in \mathbb{Z}_2^8$ lähetettävään viestiin. Vastaanottaja saa viestin muodossa

$$\begin{aligned} y = xG' + e &= \begin{bmatrix}1 & 1 & 1 & 0 & 1 & 0 & 0 & 0\end{bmatrix} + \begin{bmatrix}0 & 0 & 0 & 0 & 0 & 1 & 0 & 0\end{bmatrix} \\ &= \begin{bmatrix}1 & 1 & 1 & 0 & 1 & 1 & 0 & 0\end{bmatrix}. \end{aligned}$$

Vastaanottaja dekoodaa viestin dekoodausalgoritmin mukaisesti. Ensimmäistä vaihetta varten tarvitaan permutaatiomatriisin P käänteismatriisi. Kun tiedetään, että permutaatiomatriisin käänteismatriisi on permutaatiomatriisin transpoosi, saadaan

$$\begin{aligned} y_1 = yP^{-1} = yP^T &= \begin{bmatrix}1 & 1 & 1 & 0 & 1 & 1 & 0 & 0\end{bmatrix} \cdot \begin{bmatrix}0 & 0 & 1 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0\\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1\\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0\\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0\end{bmatrix}^T \\ &= \begin{bmatrix}1 & 1 & 1 & 0 & 1 & 1 & 0 & 0\end{bmatrix} \cdot \begin{bmatrix}0 & 0 & 1 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0\\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0\\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1\\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0\\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0\end{bmatrix} \\ &= \begin{bmatrix}1 & 1 & 1 & 0 & 0 & 1 & 0 & 1\end{bmatrix}. \end{aligned}$$

Tiedetään, että koodin C minimietäisyys $d = 3$, jolloin löytämällä vektoria y_1 lähinnä oleva koodin C alkio c voidaan korjata enintään

$$\frac{d-1}{2} = \frac{3-1}{2} = 1$$

virhettä. Koodissa C on koodisana $c = \begin{bmatrix}1 & 1 & 1 & 0 & 0 & 1 & 0 & 0\end{bmatrix}$. Koodisanan c Hamming-etäisyys vektoriin y_1 on $d(y_1, c) = 1$, joten merkitään vektoria $c = x_1$. Seuraavaksi etsitään vektori $x_0 = \begin{bmatrix}x_{0_1} & x_{0_2} & x_{0_3} & x_{0_4} & x_{0_5}\end{bmatrix} \in \mathbb{Z}_2^5$, joka toteuttaa ehdon

$$x_1 = x_0 G.$$

Yksi tapa ratkaista tämä on selvittää matriisin G käänteismatriisi G^{-1}, mutta koska matriisi G ei ole kääntyvä syystä, että se ei ole neliömatriisi, näin ei voida tehdä. Koska vektori x_0 tiedetään kuuluvan joukkoon $\mathbb{Z}_2^5$, eli sen pituus on 5 ja sen kaikki alkiot ovat joko 1 tai 0, saadaan vektori x_0 selvitettyä yhtälöryhmän avulla. Ratkaistaan vektorin ja matriisin tulo, jolloin saadaan yhtälöryhmä

$$
\begin{aligned}
&x_1 = x_0 G \\
&\quad \iff \begin{bmatrix}1 & 1 & 1 & 0 & 0 & 1 & 0 & 0\end{bmatrix} = x_0 \cdot G \\
&\quad \iff \begin{bmatrix}x_{0_1} & x_{0_2} & x_{0_3} & x_{0_4} & x_{0_5}\end{bmatrix} \cdot \begin{bmatrix}0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1\end{bmatrix} \\
&\quad \iff \begin{bmatrix} x_{0_2} + x_{0_3} + x_{0_5} & x_{0_3} + x_{0_4} + x_{0_5} & x_{0_1} + x_{0_2} + x_{0_3} & x_{0_2} + x_{0_5} & x_{0_1} + x_{0_4} \\ x_{0_2} + x_{0_3} + x_{0_5} & x_{0_1} + x_{0_2} + x_{0_4} & x_{0_2} + x_{0_4} + x_{0_5} \end{bmatrix} \\
&\quad \iff \begin{cases} x_{0_2} + x_{0_3} + x_{0_5} = 1 \\ x_{0_3} + x_{0_4} + x_{0_5} = 1 \\ x_{0_1} + x_{0_2} + x_{0_3} = 1 \\ x_{0_2} + x_{0_5} = 0 \\ x_{0_1} + x_{0_4} = 0 \\ x_{0_2} + x_{0_3} + x_{0_5} = 1 \\ x_{0_1} + x_{0_2} + x_{0_4} = 0 \\ x_{0_2} + x_{0_4} + x_{0_5} = 0 \end{cases} \iff \begin{cases} x_{0_3} = 1 \\ x_{0_4} + x_{0_5} = 0 \\ x_{0_1} + x_{0_2} = 0 \\ x_{0_2} = x_{0_5} \\ x_{0_1} = x_{0_4} \\ x_{0_2} = 0 \\ x_{0_4} = 0 \end{cases} \\
&\quad \iff \begin{cases} x_{0_1} = 0 \\ x_{0_2} = 0 \\ x_{0_3} = 1 \\ x_{0_4} = 0 \\ x_{0_5} = 0 \end{cases} .
\end{aligned}
$$

Tällöin yhtälöryhmän ratkaisu on vektori $x_0 = \begin{bmatrix}0 & 0 & 1 & 0 & 0\end{bmatrix}$. Algoritmin

viimeistä vaihetta varten tarvitaan matriisin S käänteismatriisi. Esimerkiksi Gaussin ja Jordanin menetelmällä saadaan käänteismatriisiksi

$$S^{-1} = \begin{bmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 \end{bmatrix}.$$

Käänteismatriisin avulla saadaan

$$x = x_0 S^{-1} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 1 & 1 & 0 \end{bmatrix},$$

mikä on dekoodattu viesti. Huomataan myös, että viesti on sama kuin alun perin koodattu viesti.

5 Hash-funktiot

Kun tietoa lähetetään ei-turvallista reittiä pitkin, on mahdollista, että viesti muuttuu matkan varrella jonkin ulkopuolisen tekijän toimesta (esimerkiksi virus tai tiedonsiirtovirhe). Jotta voidaan varmistua, ettei lähetettyä viestiä ole käsitelty sen lähettämisen jälkeen, on kehitetty hash-funktiot. Hash-funktion avulla lasketaan lähetettävästä viestistä tiiviste, jonka avulla vastaanottaja voi tarkistaa, onko viestiin tehty muutoksia lähettämisen jälkeen. Hash-funktion tiivisteen pituus riippuu hash-funktiosta, eli eri pituisten viestien samalla hash-funktiolla lasketut tiivisteet ovat saman mittaisia. Hash-funktio suunnitellaan niin, että sen tuottaman tiivisteen avulla ei voi päätellä alkuperäistä viestiä, eikä kahdella eri viestillä ole samanlaista tiivistettä. Teoriassa tämä ei ole mahdollista, mutta käytännössä voidaan suunnitella hash-funktio, joka toteuttaa nämä ominaisuudet nykytietokoneiden laskentatehon asettamissa rajoissa.
Käytännössä hash-funktio toimii niin, että viestin lähettäjä laskee hash-funktion avulla tiivisteen lähetettävästä viestistä, ja lähettää viestin lisäksi tämän tiivisteen. Vastaanottaja voi tarkistaa sen, ettei viestiin ole tehty muutoksia lähettämisen jälkeen, laskemalla samalla hash-funktiolla tiivisteen vastaanottamastaan viestistä. Jos vastaanottajan laskema tiiviste on sama kuin vastaanotetun viestin yhteydessä oleva tiiviste, viestiin ei ole tehty muutoksia. Jos vastaanottajan laskema tiiviste on kuitenkin eri kuin vastaanotetun viestin yhteydessä oleva tiiviste, on viestiin tehty muutoksia lähettämisen jälkeen. (Delfs, H. & Knebl, H., 2007)

5.1 Satunnaisoraakkeli-malli

Satunnaisoraakkeli-malli (eng. *random oracle*) on ideaali hash-funktiosta. Satunnaisoraakkeli-malli ei ole teoriassa saavutettavissa, mutta käytännössä hash-funktiot pyritään suunnittelemaan niin, että ne toteuttavat satunnaisoraakkeli-mallin ominaisuudet käytännön sovelluksissa.
Satunnaisoraakkeli-mallin pääidea on se, että ainoa tapa määrittää vektorin x hash-funktion tiiviste $h(x)$ on laskea hash-funktion tiiviste tälle vektorille. Tämä tarkoittaa sitä, että useita eri vektoreita ja näiden tiivisteitä vertaamalla ei voida päätellä halutun vektorin hash-tiivistettä. Lisäksi satunnaisoraakkeli-mallissa hash-funktio ei ole tiedossa, vaan käyttäjillä on tiedossaan ainoastaan eri vektoreita ja näiden tiivisteitä. Näin ollen ainoa tapa selvittää vektorin x hash-tiiviste $h(x)$ on etsiä se vektoreiden ja tiivisteiden joukosta.
Käytännössä ei luonnollisesti ole mahdollista käyttää tällaista mallia, koska

mahdollisia vektoreita ja tiivisteitä on niin paljon, että kaikkien vaihtoehtojen läpikäymiseen menisi todella paljon aikaa. Hyvä hash-funktio kuitenkin käyttäytyy samaan tapaan kuin satunnaisoraakkeli-malli; vektoria on lähes mahdotonta päätellä sen tiivisteestä, vaikka monen muun vektorin tiivisteet olisivat tiedossa, ja tiivisteet näyttäisivät muodostuvan satunnaisesti. (Delfs, H. & Knebl, H., 2007)

5.2 Merkle-Damgård-konstruktio

Merkle-Damgård-konstruktio on tapa muodostaa kompressiofunktion avulla sellainen hash-funktio, joka toteuttaa törmäysehdon, mikä tarkoittaa sitä, että kahden eri vektorin hash-funktion arvo ei ole sama millään kahdella eri vektorilla. Kompressiofunktio on sellainen funktio, joka tuottaa vektorin tiivisteen, eli esimerkiksi tuottaa n:n pituisia vektoreita $(n + r)$:n pituisista vektoreista. Törmäysehdon toteuttavaa funktiota voidaan kutsua törmäysvapaaksi funktioksi. Merkle-Damgård-konstruktio pienentää hash-funktion määrittelemisen ongelman siihen, että keksitään törmäysvapaa kompressiofunktio f hash-funktion määrittelemiseksi. (Delfs, H. & Knebl, H., 2007)

Määritelmä 5.1. Olkoon

$$f : \{0,1\}^{n+r} \to \{0,1\}^n$$

törmäysvapaa kompressiofunktio, jonka kompression määrä on r. Funktio f siis käsittelee $(n + r)$:n pituisia binäärivektoreita ja tuottaa n:n pituisia binäärivektoreita. Funktion f avulla määritellään hash-funktio

$$h : \{0,1\}^* \to \{0,1\}^n.$$

Olkoon $m \in \{0,1\}^*$ satunnaismittainen viestivektori. Hash-funktio konstruoidaan seuraavan menetelmän mukaisesti:

1. Täydennetään viestivektori m niin, että sen pituus on r:n monikerta. Täydennys tehdään niin, että viestivektorin m perään lisätään yksi alkio 1, jonka jälkeen lisätään niin monta alkiota 0, että viestivektorin m pituus on r:n monikerta. Jos viestivektorin m pituus on valmiiksi r:n monikerta, täydennys tehdään silti. Täydennettyä viestivektoria merkitään m'.

2. Vektori m' jaetaan r:n mittaisiin osiin, eli

$$m' = m_1 \| m_2 \| \ldots \| m_k, \; m_i \in \{0,1\}^r, 1 \leq i \leq k.$$

Lisätään vielä loppuun vektori m_{k+1}, johon tallennetaan alkuperäisen viestivektorin m pituus ennen täydennystä. Jos binäärimuotoinen esitys viestivektorin m pituudesta ei ole r:n monikerta, täytetään loput paikat alkioilla 0. Näin vektori m' tulee muotoon

$$m' = m_1 \| m_2 \| \dots \| m_k \| m_{k+1}.$$

3. Iterointi aloitetaan vektorista $v_0 \in \{0,1\}^n$, joka on jokin joukon $\{0,1\}^n$ vektori, joka valitaan riippumatta viestivektorista m. Vektori v_i lasketaan seuraavasti:

$$v_i = f(v_{i-1} \| m_i), \ 1 \leq i \leq k+1.$$

4. Viestin m hash-funktion arvo on tällöin viimeinen iteraation arvo, eli

$$h(m) = v_{k+1}.$$

Merkle-Damgård-konstruktion avulla on tehty useita paljon käytettyjä hash-funktioita, kuten MD5, SHA-1 ja SHA-2.

Esimerkki 5.2. Olkoon f kompressiofunktio, johon syötetään binäärivektoreita, joiden pituus on yhdeksän, ja joka tuottaa binäärivektoreita, joiden pituus on kuusi. Funktio f jakaa binäärivektorin, jonka pituus on yhdeksän, kolmen alkion osiin, joista otetaan ensimmäinen alkio pois. Eli esimerkiksi vektorin $v = \begin{bmatrix} 0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \end{bmatrix} \in \{0,1\}^9$ kompressiofunktion arvo on $f(v) = \begin{bmatrix} 1 & 0 & 1 & 1 & 1 & 0 \end{bmatrix} \in \{0,1\}^6$. Huomataan, että tämä kompressiofunktio ei ole törmäysvapaa, mutta sillä ei ole tämän esimerkin toimivuuden kannalta merkitystä.

Muodostetaan edellä määritellyn kompressiofunktion f ja Merkle-Damgård-konstruktion avulla hash-tiiviste vektorille

$$m = \begin{bmatrix} 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix} \in \{0,1\}^{11}.$$

Koska vektorin m pituus ei ole kolmen monikerta, täydennetään vektori m niin, että lisätään sen perään alkio 1. Näin saadaan vektori

$$m' = \begin{bmatrix} 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \end{bmatrix} \in \{0,1\}^{12}.$$

Vektori m' jaetaan osiin, joiden pituus on kolme, jolloin saadaan

$$\begin{aligned} m' &= m_1 \| m_2 \| \dots \| m_k \\ &= \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \| \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \| \begin{bmatrix} 0 & 1 & 1 \end{bmatrix} \| \begin{bmatrix} 0 & 1 & 1 \end{bmatrix}. \end{aligned}$$

Loppuun lisätään vielä alkuperäisen viestivektorin m pituus, eli luvun 11 binäärimuotoinen esitys, joka on 1011. Koska tämän pituus ei ole kolmen monikerta, niin loppuun lisätään kaksi alkiota 0, jolloin saadaan

$$m' = \begin{bmatrix} 0 & 0 & 1 & \| & 1 & 0 & 0 & \| & 0 & 1 & 1 & \| & 0 & 1 & 1 & \| & 1 & 0 & 1 & \| & 1 & 0 & 0 \end{bmatrix}.$$

Valitaan aloitusvektoriksi $v_0 = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix} \in \{0,1\}^6$. Koska vektori m' on nyt jaettu kuuteen osaan, iteraatiossa on kuusi kierrosta. Viimeisen kierroksen tuottama vektori v_6 on hash-funktion arvo viestivektorille m. Lasketaan vektorit v_i:

$$v_1 = f(v_0 \| m_1) = f(\begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}) = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix},$$

$$v_2 = f(v_1 \| m_2) = f(\begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \end{bmatrix}) = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 \end{bmatrix},$$

$$v_3 = f(v_2 \| m_3) = f(\begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}) = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix},$$

$$v_4 = f(v_3 \| m_4) = f(\begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \end{bmatrix}) = \begin{bmatrix} 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix},$$

$$v_5 = f(v_4 \| m_5) = f(\begin{bmatrix} 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 \end{bmatrix}) = \begin{bmatrix} 0 & 1 & 1 & 1 & 0 & 1 \end{bmatrix},$$

$$v_6 = f(v_5 \| m_6) = f(\begin{bmatrix} 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \end{bmatrix}) = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 \end{bmatrix}.$$

Näin ollen viestin $m = \begin{bmatrix} 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix}$ tiiviste on $h(m) = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 \end{bmatrix}$.

6 McEliece-koodaussysteemi ja allekirjoittaminen

Tässä luvussa esitellään McEliece-koodaussysteemin pohjalta muodostettu allekirjoitus CFS. CFS-allekirjoitusta varten määritellään Niederreiter-koodaussysteemi, joka pohjautuu vahvasti McEliece-koodaussysteemiin. Niederreiter-koodaussysteemistä ja CFS-allekirjoituksesta kerrotaan artikkelissa *How to Achieve a McEliece-based Digital Signature Scheme* (Courtois, Finiasz & Sendrier, 2001). [5]

6.1 Viestin allekirjoittaminen

Kun vastaanottaja dekoodaa vastaanottamansa viestin, ei hänellä ole mitään takeita siitä, että viesti on tullut halutulta lähettäjältä. Tämän ongelman ratkaisuksi on kehitetty erilaisia allekirjoitusjärjestelmiä.
Allekirjoitusjärjestelmät perustuvat siihen, että viestin vastaanottaja pystyy suhteellisen lyhyellä ja yksinkertaisella laskutoimituksella varmistamaan, että viesti on oikealta lähettäjältä. Tätä varten viestin lähettäjä liittää viestin yhteyteen digitaalisen allekirjoituksen, jonka muoto riippuu käytettävästä allekirjoitusjärjestelmästä. Joka tapauksessa allekirjoituksen pituus on huomattavasti viestiä lyhyempi. Allekirjoitus ei ole koodattua tekstiä, vaan koodatun viestin avulla muodostettu merkkijono.
Allekirjoitusjärjestelmä valitaan usein niin, että se toimii käytännöllisesti yhdessä koodaussysteemin kanssa. Toisin sanoen, allekirjoitusjärjestelmä valitaan koodaussysteemin perusteella. Esimerkiksi McEliece-koodaussysteemin kanssa käytetään usein allekirjoitusjärjestelmää, joka pohjautuu syndromeihin ja McEliece-koodausjärjestelmän muunnokseen, Niederreiter-koodaussysteemiin.

6.2 Niederreiter-koodaussysteemi

Määritelmä 6.1. Olkoon C lineaarinen $[n, k, d]$-koodi, jonka tarkistusmatriisi on $((n-k) \times n)$-matriisi H. Olkoon R $((n-k) \times (n-k))$-matriisi, joka on kääntyvä kunnassa $\mathbb{Z}_2$ ja P $(n \times n)$-permutaatiomatriisi. Niederreiter-koodaussysteemin julkinen avain on $((n-k) \times n)$-matriisi $H' = RHP$, ja yksityinen avain muodostuu matriiseista H, R ja P. Viesti $x \in \mathbb{Z}_2^n$ koodataan julkisen avaimen avulla seuraavasti:

$$y = H'x^T.$$

Näin ollen Niederreiter-koodaussysteemin avulla koodattu viesti ei ole koodisana, vaan ennemminkin virherakenne.

6.3 CFS

CFS-allekirjoitussysteemi perustuu Niederreiter-koodaussysteemiin, joka taas perustuu McEliece-koodaussysteemiin. CFS-allekirjoitussysteemi on käytännössä julkisen avaimen koodaussysteemi, jossa koodataan halutun viestivektorin sijasta koodatun viestivektorin hash-tiiviste. CFS-allekirjoitussysteemin turvallisuus perustuu Niederreiter- ja McEliece-koodaussysteemien tapaan siihen, että koodin rakennetta on hankalaa selvittää pelkän julkisen avaimen perusteella.

Määritelmä 6.2. Olkoon D koodattu viesti, eli dokumentti, joka halutaan allekirjoittaa, ja olkoon h hash-funktio, joka palauttaa $(n-k)$-pituisen binäärivektorin. Olkoon lisäksi vektori $s = h(D)$ dokumentti D operoituna hash-funktiolla. Merkitään vektorin s ja indeksin i ketjutusta $\begin{bmatrix} s & \| & i \end{bmatrix}$ ja $s_i = h(\begin{bmatrix} s & \| & i \end{bmatrix})$. CFS-allekirjoitus muodostetaan seuraavan algoritmin avulla:

1. Lasketaan vektoreita s_i aloittaen indeksistä $i = 0$ niin, että löydetään sellainen vektori, joka on dekoodattavissa. Merkitään tätä indeksiä i_0.

2. Etsitään sellainen vektori z, että
$$zH'^T = s_{i_0}.$$

3. Allekirjoitus on muotoa
$$\begin{bmatrix} z & \| & i_0 \end{bmatrix}.$$

Viestin vastaanottaja varmistaa allekirjoituksen avulla, että viesti on oikealta lähettäjältä, seuraavan algoritmin mukaisesti:

1. Lasketaan
$$s_1 = zH'^T$$
julkisella avaimella H'.

2. Lasketaan
$$s_2 = h(\begin{bmatrix} h(D) & \| & i_0 \end{bmatrix})$$
julkisella hash-funktiolla h.

3. Jos $s_1 = s_2$, on viesti oikealta lähettäjältä.

Todistus. Osoitetaan, että viestin lähettäjä voidaan varmistaa allekirjoituksen avulla edellä esitettyä algoritmia noudattaen.
Olkoon H' tiedossa oleva julkinen avain ja $\begin{bmatrix} z & \| & i_0 \end{bmatrix}$ vastaanotettu allekirjoitus. Kun lasketaan vektori s_1, saadaan

$$s_1 = zH'^T = s_{i_0}$$

määritelmän 6.2. kohdan 2 mukaisesti. Vektori $s = h(D)$ saadaan hash-funktion h avulla, ja tämän ketjutus $s_i = h(\begin{bmatrix} s & \| & i \end{bmatrix})$ saadaan hash-funktion h ja indeksin i avulla. Kun on tiedossa dokumentti D ja indeksi i_0, voidaan muodostaa vektori

$$s_2 = h(\begin{bmatrix} h(D) & \| & i_0 \end{bmatrix}) = h(\begin{bmatrix} s & \| & i_0 \end{bmatrix}) = s_{i_0}.$$

Koska $s_1 = s_{i_0} = s_2$, allekirjoituksen tarkistusalgoritmi toimii. □

Esimerkki 6.3. Olkoon C lineaarinen $[6, 3, 3]$-koodi, jonka generoijamatriisi on (3×6)-matriisi

$$G = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 \end{bmatrix}$$

ja tarkistusmatriisi on (3×6)-matriisi

$$H = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 \end{bmatrix}.$$

Koodin C virheenkorjauskyky on $\frac{d_{min}C-1}{2} = \frac{3-1}{2} = 1$, joten kaikissa dekoodattavissa vektoreissa on korkeintaan yksi virhe. Lasketaan kaikkien virhevektoreiden, joiden paino on yksi, syndromit:

$$s_1 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} H^T = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix},$$

$$s_2 = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} H^T = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix},$$

$$s_3 = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} H^T = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix},$$

$$s_4 = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} H^T = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix},$$

$$s_5 = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} H^T = \begin{bmatrix} 0 & 1 & 1 \end{bmatrix},$$

$$s_6 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} H^T = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}.$$

Olkoon lisäksi (3×3)-matriisi

$$R = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 1 \end{bmatrix},$$

joka on kääntyvä kunnassa $\mathbb{Z}_2$, sekä (6×6)-permutaatiomatriisi

$$P = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}.$$

Niederreiter-koodin julkinen avain saadaan laskemalla

$$H' = RHP = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \end{bmatrix}.$$

Olkoon hash-funktio h funktio, joka tuottaa binäärivektoreita, joiden pituus on kolme. Hash-arvo saadaan Merkle-Damgård-konstruktion avulla, kompressiofunktiona funktio $f : \{0,1\}^6 \to \{0,1\}^3$, johon syötetään binäärivektoreita, joiden pituus on kuusi. Kompressiofunktio f toimii niin, että binäärivektori jaetaan kahden alkion mittaisiin osiin, ja näiden osien alkiot lasketaan yhteen modulo kaksi, jolloin saadaan binäärivektori, jonka pituus on kolme. Huomataan, että tämä kompressiofunktio ei ole törmäysvapaa, mutta se ei haittaa tämän esimerkin toimimista.

Jollain tavalla, esimerkiksi McEliece-koodaussysteemillä, on salattu viestivektori, jota kutsutaan dokumentiksi D. Sillä, minkä mittainen dokumentti halutaan allekirjoittaa, ei ole väliä, koska hash-funktio tuottaa kaikenmittaisista viestivektoreista samanmittaisia tiivisteitä. Dokumentti on nyt

$D = \begin{bmatrix}1 & 1 & 1 & 0 & 1 & 1 & 0 & 0\end{bmatrix} \in \{0,1\}^8$ (kts. esimerkki 4.5.), ja se halutaan allekirjoittaa. Lasketaan ensin vektori $s = h(D)$. Koska dokumentin D pituus ei ole kolmen monikerta, täydennetään dokumenttia D lisäämällä loppuun alkio 1. Lisätään loppuun alkuperäisen dokumentin D pituus binäärimuodossa, joka on 1000, ja täydennetään kahdella alkiolla 0, koska binäärimuotoisen esityksen pituus ei ole kolmen monikerta. Merkitään täydennettyä dokumenttia D'. Jaetaan vektori D' kolmen mittaisiin osiin, jolloin saadaan

$$\begin{aligned} D' &= D_1 \| D_2 \| D_3 \| D_4 \| D_5 \\ &= \begin{bmatrix}1 & 1 & 1 & \| & 0 & 1 & 1 & \| & 0 & 0 & 1 & \| & 1 & 0 & 0 & \| & 0 & 0 & 0\end{bmatrix}. \end{aligned}$$

Aloitetaan iterointi vektorista $v_0 = \begin{bmatrix}0 & 0 & 1\end{bmatrix}$, ja lasketaan kaikki viisi kierrosta, jolloin saadaan

$$\begin{aligned} v_1 &= f(v_0 \| D_1) = f(\begin{bmatrix}0 & 0 & 1 & 1 & 1 & 1\end{bmatrix}) = \begin{bmatrix}0 & 0 & 0\end{bmatrix}, \\ v_2 &= f(v_1 \| D_2) = f(\begin{bmatrix}0 & 0 & 0 & 0 & 1 & 1\end{bmatrix}) = \begin{bmatrix}0 & 0 & 0\end{bmatrix}, \\ v_3 &= f(v_2 \| D_3) = f(\begin{bmatrix}0 & 0 & 0 & 0 & 0 & 1\end{bmatrix}) = \begin{bmatrix}0 & 0 & 1\end{bmatrix}, \\ v_4 &= f(v_3 \| D_4) = f(\begin{bmatrix}0 & 0 & 1 & 1 & 0 & 0\end{bmatrix}) = \begin{bmatrix}0 & 0 & 0\end{bmatrix}, \\ v_5 &= f(v_4 \| D_5) = f(\begin{bmatrix}0 & 0 & 0 & 0 & 0 & 0\end{bmatrix}) = \begin{bmatrix}0 & 0 & 0\end{bmatrix}. \end{aligned}$$

Nyt $v_5 = h(D) = s$. Seuraavaksi lasketaan hash-tiivisteitä $h(\begin{bmatrix}s & \| & i\end{bmatrix})$ aloittaen indeksistä $i = 0$, kunnes saadaan sellainen tiiviste, joka on jokin koodin C koodisanan syndromi (eli vastaavan Niederreiter-koodin alkio). Hash-tiivisteet lasketaan samalla tavalla kuin edellä, aloittaen vektorista $v_0 = \begin{bmatrix}0 & 0 & 1\end{bmatrix}$. Nyt

$$h(\begin{bmatrix}s & \| & 0\end{bmatrix}) = h(\begin{bmatrix}0 & 0 & 0 & 0\end{bmatrix}) = \begin{bmatrix}1 & 0 & 0\end{bmatrix},$$

Indeksillä $i = 0$ saatiin Niederreiter-koodin alkio, eli jokin alussa määritellyistä syndromeista. Merkitään tätä vektoria $s_{i_0} = \begin{bmatrix}1 & 0 & 0\end{bmatrix}$. Seuraavaksi etsitään sellainen vektori z, että $zH'^T = s_{i_0}$. Merkitään vektoria $z = \begin{bmatrix}z_1 & z_2 & z_3 & z_4 & z_5 & z_6\end{bmatrix}$ ja muodostetaan yhtälöryhmä

$$zH'^T = s_{i_0} = \begin{bmatrix} z_1 & z_2 & z_3 & z_4 & z_5 & z_6 \end{bmatrix} \cdot \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}$$

$$= \begin{bmatrix} z_2 + z_5 + z_6 & z_1 + z_5 & z_2 + z_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}$$

$$= \begin{cases} z_2 + z_5 + z_6 = 1 \\ z_1 + z_5 = 0 \Rightarrow z_1 = z_5 = 1 \text{ tai } z_1 = z_5 = 0 \\ z_2 + z_3 = 0 \Rightarrow z_2 = z_3 = 1 \text{ tai } z_2 = z_3 = 0 \end{cases}$$

Esimerkiksi vektori $z = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$ toteuttaa ehdon $zH'^T = s_{i_0}$. Vektorista z ja indeksistä $i_0 = 0$ saadaan muodostettua CFS-allekirjoitus, joka on $\begin{bmatrix} z & \| & i_0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$. Allekirjoitus lähetetään dokumentin yhteydessä, joten vastaanottaja vastaanottaa sekä dokumentin että allekirjoituksen.
Vastaanottaja tarkistaa allekirjoituksen avulla, onko viesti oikealta lähettäjältä. Ensin vastaanottaja laskee vektorin s_1

$$s_1 = zH'^T = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}.$$

Sitten vastaanottaja laskee vektorin s_2 hash-funktion avulla

$$s_2 = h(\begin{bmatrix} h(D) & \| & i_0 \end{bmatrix}) = h(\begin{bmatrix} 0 & 0 & 0 & 0 \end{bmatrix}) = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}.$$

Koska $s_1 = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} = s_2$, viesti on oikealta lähettäjältä.

7 McEliece-koodaussysteemi ja turvallisuus

Koodaussysteemien turvallisuus perustuu yleensä tekijöihinjaon tai diskreetin logaritmin ongelman hankaluuteen. Koodaussysteemit, jotka hyödyntävät tekijöihinjakoa ja diskreetin logaritmin ongelmaa, ovat suhteellisen helppoja ohjelmoida ja nopeita käyttää. Sen takia tällaiset koodaussysteemit ovatkin hyvin yleisesti käytössä.
McEliece-koodaussysteemin lisäksi on paljon muita koodaussysteemeitä, joiden turvallisuus perustuu johonkin muuhun kuin tekijöihinjakoon tai diskreetin logaritmin ongelmaan. Muita ongelmia, joihin koodaussysteemit voivat perustua, ovat esimerkiksi hilojen tai elliptisten käyrien laskutoimitukset.
Tulevaisuudessa tullaan tarvitsemaan yhä hankalammin murrettavia koodaussysteemejä, koska tietokoneiden tehokkuus kasvaa. Esimerkiksi diskreetin logaritmin ongelmaan perustuvat koodaussysteemit tulevat mahdollisesti olemaan täysin turhia sitten, kun kvanttitietokoneet ovat yleisessä käytössä näiden valtavan laskentatehon vuoksi.

7.1 RSA-koodaussysteemi

Määritelmä 7.1. Olkoon $n = pq$, missä p ja q ovat (suuria) alkulukuja. Tällöin viesti $m \in \mathbb{Z}_n$ salataan seuraavasti:

1. Valitaan sellainen luku e, että

$$\text{syt}(e, (p-1)(q-1)) = 1.$$

2. Valitaan sellainen luku d siten, että

$$\text{syt}(d, (p-1)(q-1)) = 1 \text{ ja}$$

$$de \equiv 1 \mod (p-1)(q-1).$$

3. Salattu viesti v lasketaan alkuperäisestä viestistä m laskemalla

$$v \equiv m^e \mod n.$$

Kun vastaanotetaan viesti $v \in \mathbb{Z}_n$, se avataan seuraavasti:

1. Avattu viesti m saadaan vastaanotetusta viestistä v laskemalla

$$m \equiv v^d \mod n.$$

7.2 RSA vs. McEliece

RSA-koodaussysteemin turvallisuus perustuu sekä tekijöihinjakoon että diskreetin logaritmin ongelmaan. RSA on yksi yleisimpiä käytössä olevia koodaussysteemeitä, ja sen turvallisuus on toistaiseksi ollut riittävän hyvä. Jatkuvasti täytyy kuitenkin pidentää koodaussysteemissä liikkuvien merkkijonojen pituutta. RSA:n turvallisuus käytännössä perustuu siis siihen, että tietyn mittaista merkkijonoa ei vielä ole kyetty jakamaan tekijöihin.
McEliece-koodaussysteemin turvallisuus perustuu siihen, että koodin murtajan on hyvin vaikeaa arvata se koodisana, jonka viestin lähettäjä on halunnut lähettää ilman, että on mitään muita tietoja koodista julkisen avaimen lisäksi. Julkisen avaimen ollessa matriisi G', joka on muodostettu matriisien S, G ja P avulla, murtajan on hyvin vaikeaa määrittää koodin generoijamatriisi, jota tarvitaan koodisanojen määrittämistä varten.
Vaikka RSA ja McEliece on kehitetty samoihin aikoihin, RSA:sta on tullut kansainvälisesti huomattavasti suositumpi koodaussysteemi helppoutensa vuoksi. Kuitenkin tulevaisuudessa, kun tietokoneet kehittyvät tehokkaammiksi, voi olla, että McEliece-koodaussysteemistä tulee aiempaa suositumpi, koska se on vaikeampi murtaa kuin RSA.

7.3 McEliecen murtaminen

McEliece-koodaussysteemi on pystytty murtamaan tietyillä parametrien arvoilla jo aikana ennen kvanttitietokoneita. Tässä luvussa esitellään kaksi McEliece-koodaussysteemin hyökkäysmenetelmää: Reedin-Mullerin koodin avulla tapahtuva hyökkäys ja Stern-hyökkäys.

7.3.1 Minderin-Shokrollahin hyökkäys

Reedin-Mullerin koodi on kehitetty vuonna 1954, mikä tekee siitä yhden vanhimmista koodeista. (MacWilliams, F.J. & Sloane, N.J.A., 1977)

Määritelmä 7.2. Funktiota $f(x) = f(x_1, \ldots, x_m)$, joka saa arvoja 0 ja 1, kutsutaan *totuusarvofunktioksi* (eng. *Boolean function*). Totuusarvofunktio voidaan määritellä *totuustaulukon* avulla. Totuustaulukko on $n = 2^m$ pituinen binäärivektori Ω_f, joka muodostetaan totuusarvofunktion f avulla seuraavasti:

$$\Omega_f = (f(0, \ldots, 0, 0), f(0, \ldots, 0, 1), f(0, \ldots, 1, 0), \ldots, f(1, \ldots, 1, 1)).$$

Määritelmä 7.3. Olkoon n:n pituiset vektorit $v = \begin{bmatrix} v_1 & v_2 & \ldots & v_n \end{bmatrix}$ ja $w = \begin{bmatrix} w_1 & w_2 & \ldots & w_n \end{bmatrix}$. Tällöin *vektoreiden kertolasku komponenteittain* las-

ketaan seuraavasti:

$$vw = \begin{bmatrix} v_1 w_1 & v_2 w_2 & \dots & v_n w_n \end{bmatrix}.$$

Määritelmä 7.4. Olkoon $(m \times n)$-matriisi $\mathcal{M}$ sellainen matriisi, jonka sarakkeet muodostavat lukujen $0, 1, \dots, 2^m - 1$ binääriesitykset. Matriisi $\mathcal{M}$ on tällöin muotoa

$$\mathcal{M} = \begin{bmatrix} \xi_{m0} & \xi_{m1} & \cdots & \xi_{m(n-1)} \\ \xi_{(m-1)0} & \xi_{(m-1)1} & \cdots & \xi_{(m-1)(n-1)} \\ \vdots & \vdots & \ddots & \vdots \\ \xi_{10} & \xi_{11} & \cdots & \xi_{1(n-1)} \end{bmatrix} = \begin{bmatrix} v_m \\ v_{m-1} \\ \vdots \\ v_1 \end{bmatrix} = \begin{bmatrix} s_0 & s_1 & \dots & s_{n-1} \end{bmatrix},$$

missä $v_i = \begin{bmatrix} \xi_{i0} & \xi_{i1} & \dots & \xi_{i(n-1)} \end{bmatrix}$ ja $s_i = \begin{bmatrix} \xi_{mj} & \xi_{(m-1)j} & \dots & \xi_{1j} \end{bmatrix}^T$ ja

$$J = \sum_{i=1}^{m} \xi_{ij} 2^{i-1}, \xi_{ij} \in \{0, 1\}$$

on luvun $j \in \{0, 1, \dots, 2^m - 1\}$ binääriesitys.

Kun $1 \leq r \leq m$, niin 2^m-pituiseksi *r. kertaluvun Reedin-Mullerin koodiksi* eli *RM-koodiksi* sanotaan sitä avaruuden $\mathbb{Z}_2^{2^m}$ aliavaruutta, jonka kantavektoreina ovat $v_0 = \begin{bmatrix} 1 & 1 & \dots & 1 \end{bmatrix}$ ja kaikkien vektoreiden $v_1, v_2, \dots, v_m$ tulot $v_{i_1} v_{i_2} \dots v_{i_k}$, missä $k \leq r$. Tälle koodille käytetään merkintää $RM(r, m)$. Toisin sanoen, r. kertaluvun Reedin-Mullerin koodi $RM(r, m)$ on sellaisten vektoreiden Ω_f joukko, joille $f(w_i)$, missä $w_i \in \mathbb{Z}_2^m$, on totuusarvofunktio. *Nollannen kertaluvun Reedin-Mullerin koodiksi* $RM(0, m)$ sanotaan vektorin v_0 generoimaa toistokoodia. Määritelmästä seuraa, että $RM(m, m) = \mathbb{Z}_2^{2^m}$. Reedin-Mullerin koodin $RM(r, m)$ dimensio on $k = \sum_{i=0}^{r} \binom{m}{i}$, pituus on $n = 2^m$ ja minimietäisyys $d = 2^{m-r}$.

Esimerkki 7.5. Reedin-Mullerin koodin $RM(1, 3)$, jonka pituus on $n = 2^3 = 8$, kantavektorit ovat

$$\begin{aligned} v_0 &= \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}, \\ v_1 &= \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}, \\ v_2 &= \begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \end{bmatrix}, \\ v_3 &= \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}, \end{aligned}$$

jolloin sen generoi matriisi

$$R = \begin{bmatrix} v_0 \\ v_1 \\ v_2 \\ v_3 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

Reedin-Mullerin koodin $RM(2,4)$, jonka pituus on $n = 2^4 = 16$, kantavektorit ovat

$$\begin{aligned}
v_0 &= \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}, \\
v_1 &= \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}, \\
v_2 &= \begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \end{bmatrix}, \\
v_3 &= \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}, \\
v_4 &= \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \\
v_1v_2 &= \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}, \\
v_1v_3 &= \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \\
v_1v_4 &= \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \\
v_2v_3 &= \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \\
v_2v_4 &= \begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \\
v_3v_4 &= \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.
\end{aligned}$$

Huomautus 7.6. Reedin-Mullerin koodi $RM(r,m)$ on kaikkien totuusarvofunktioiden $f(x_1,\ldots,x_m)$ totuustaulukoiden Ω_f muodostaman vektoriavaruuden aliavaruus.

Jatkossa Minderin-Shokrollahin hyökkäykseen liittyen ei tehdä eroa merkinnöissä totuusarvofunktion f ja totuustaulukon Ω_f välillä yksinkertaisuuden säilyttämiseksi.

Hyökkäys McEliece-koodaussysteemiin Reedin-Mullerin koodin avulla perustuu yksityisen avaimen selvittämiseen julkisesta avaimesta. Tällaista hyökkäystä kutsutaan Minderin-Shokrollahin hyökkäykseksi.

Määritelmä 7.7. Olkoon $RM(r,m)$ Reedin-Mullerin koodi, jonka generoijamatriisi on $(k \times n)$-matriisi R. Reedin-Mullerin koodin $RM(r,m)$ pohjalta on muodostettu McEliece-koodi, jonka generoijamatriisi on sama matriisi R,

joka on Reedin-Mullerin koodin $RM(r,m)$ generoijamatriisi. Sekä Reedin-Mullerin koodi $RM(r,m)$ että tämän pohjalta muodostettu McEliece-koodi ovat tuntemattomia. Matriisi R koostuu yksikkövektorista ja kaikista sellaisista totuustaulukoista, jotka muodostuvat seuraavasti:

$$R = \begin{bmatrix} G_0 \\ G_1 \\ \vdots \\ G_r \end{bmatrix},$$

missä $G_0 = \Omega_1 = \begin{bmatrix} 1 & 1 & \dots & 1 \end{bmatrix}$, ja

$$G_1 = \begin{bmatrix} \Omega_{f_m} \\ \Omega_{f_{m-1}} \\ \vdots \\ \Omega_{f_2} \\ \Omega_{f_1} \end{bmatrix}, G_2 = \begin{bmatrix} \Omega_{f_{m-1}}\Omega_{f_m} \\ \Omega_{f_{m-2}}\Omega_{f_{m-1}} \\ \vdots \\ \Omega_{f_1}\Omega_{f_3} \\ \Omega_{f_1}\Omega_{f_2} \end{bmatrix}, G_r = \begin{bmatrix} \Omega_{f_{m-r+1}}\Omega_{f_{m-r+2}} \dots \Omega_{f_m} \\ \Omega_{f_{m-r}}\Omega_{f_{m-r+1}} \dots \Omega_{f_{m-1}} \\ \vdots \\ \Omega_{f_1}\Omega_{f_2} \dots \Omega_{f_{r-1}}\Omega_{f_{r+1}} \\ \Omega_{f_1}\Omega_{f_2} \dots \Omega_{f_{r-1}}\Omega_{f_r} \end{bmatrix}.$$

McEliece-koodin yksityisen avaimen muodostavat generoijamatriisin R lisäksi kääntyvä $(k \times k)$-matriisi H ja $(n \times n)$-permutaatiomatriisi σ, jolloin McEliece-koodin julkinen avain on matriisi $G = HR\sigma$. Koska myös tulomatriisi HR muodostaa Reedin-Mullerin koodin $RM(r,m)$ kannan [4], niin matriisi G on Reedin-Mullerin koodin $RM^\sigma(r,m)$ kanta. *Minderin-Shokrollahin hyökkäyksen* tavoitteena on muodostaa Reedin-Mullerin koodin $RM^\sigma(r,m)$ pohjalta Reedin-Mullerin koodi $RM(r,m)$, jonka generoijamatriisi R on myös hyökkäyksen kohteena olevan McEliece-koodin generoijamatriisi. Tällöin täytyy löytää sellaiset matriisit H' ja σ', joille pätee, että $H'G\sigma' = R$, koska matriisi R on McEliece-koodin generoijamatriisi. Matriisi R osataan koostaa kuten yllä. Matriisit σ' ja H' voidaan löytää seuraavasti:

1. Muodostetaan Reedin-Mullerin koodin $RM^\sigma(r,m)$ avulla koodi $RM^\sigma(r-1,m)$. Reedin-Mullerin koodi $RM^\sigma(r-1,m)$ voidaan muodostaa seuraavasti:

   - Tiedetään [8], että on olemassa sellainen Reedin-Mullerin koodin $RM^\sigma(r-1,m)$ kanta, joka koostuu pelkästään minimipainoisista kantavektoreista, ja Reedin-Mullerin koodin $RM^\sigma(r-1,m)$ minimipainoiset kantavektorit ovat kahden Reedin-Mullerin koodin $RM^\sigma(r,m)$ minimipainoisen kantavektorin tuloja.
   - Etsitään Reedin-Mullerin koodin $RM^\sigma(r,m)$ minimipainoisia kantavektoreita ja lasketaan Reedin-Mullerin koodin $RM^\sigma(r-1,m)$

minimipainoisia kantavektoreita näiden tuloina, kunnes saadaan riittävä määrä Reedin-Mullerin koodin $RM^{\sigma}(r-1,m)$ minimipainoisia kantavektoreita kannan muodostamiseksi.

2. Toistetaan vaihetta 1 niin kauan, kunnes saadaan muodostettua koodi $RM^{\sigma}(1,m)$.

3. Etsitään sellainen permutaatiomatriisi σ', jolle pätee, että

$$RM^{\sigma\cdot\sigma'}(1,m) = RM(1,m).$$

Tämä onnistuu seuraavasti:

- Olkoon G_1 jokin koodin $RM^{\sigma}(1,m)$ generoijamatriisi, joka sisältää yksikkövektorin yhtenä riveistä. Muodostetaan matriisin G_1 avulla matriisi G_1' poistamalla matriisista G_1 yksikkövektori. Nyt matriisi G_1' on $(m \times 2^m)$-matriisi, jonka sarakkeina on kaikki m:n mittaiset binäärivektorit. Matriisissa G_1' ei esiinny samaa saraketta kahta kertaa [4].
- Nyt matriisi σ' on permutaatiomatriisi, jonka tulo matriisin G_1' kanssa tuottaa matriisin, jonka sarakkeina on kaikki m:n pituiset binääriluvut järjestyksessä pienimmästä suurimpaan. Tämä matriisi on Reedin-Mullerin koodin $RM(1,m)$ generoijamatriisi, ja löydetty permutaatiomatriisi σ' on haluttu permutaatiomatriisi.

Näin löydetty permutaatiomatriisi σ' toteuttaa myös ehdon

$$RM^{\sigma\cdot\sigma'}(r,m) = RM(r,m).$$

4. Rekonstruoidaan matriisi H' permutaatiomatriisin σ' avulla. Tämä onnistuu niin, että ratkaistaan matriisi H' yhtälöstä

$$H'G' = R,$$

kun $G' = G\sigma'$.

(Borodin, M.A. & Chizhov, I.V., 2014)

Minderin-Shokrollahin hyökkäys on tällaisenaan melko työlästä. Määritellään seuraavaksi Reedin-Mullerin koodien operaatiot $\odot$ ja $\perp$.

Määritelmä 7.8. Olkoon $RM^{\sigma}(r_1,m)$ ja $RM^{\sigma}(r_2,m)$ Reedin-Mullerin koodeja. Operaatiot $\odot$ ja $\perp$ määritellään seuraavasti:

1.
$$RM^\sigma(r_1, m) \odot RM^\sigma(r_2, m) = RM^\sigma(r_1 + r_2, m).$$

2.
$$RM^\sigma(r_1, m)^\perp = RM^\sigma(m - r_1 - 1, m).$$

Operaatioiden $\odot$ ja $\perp$ avulla saadaan muutettua Minderin-Skrollahin hyökkäystä tehokkaammaksi.

Lause 7.9. *Olkoon McEliece-koodin ja Reedin-Mullerin koodin parametrit määritelty kuten määritelmässä 7.7. Matriisi H' löydetään tehokkaammin parannellun Minderin-Shokrollahin hyökkäyksen avulla seuraavasti:*

1. *Muodostetaan Reedin-Mullerin koodin $RM^\sigma(r, m)$ avulla Reedin-Mullerin koodi $RM^\sigma(d, m)$, missä $d = syt(r, m - 1)$, operaatioiden $\odot$ ja $\perp$ avulla. Tämä onnistuu seuraavasti:*

   - *Lasketaan Reedin-Mullerin koodeja $(RM^\sigma(r, m))^\perp$ niin kauan, että löydetään sellaiset Reedin-Mullerin koodit $RM^\sigma(r_1, m)$ ja $RM^\sigma(r_2, m)$, että $r_1 + r_2 = d$.*
   - *Lasketaan Reedin-Mullerin koodi $RM^\sigma(d, m)$ laskemalla*
   $$RM^\sigma(r_1, m) \odot RM^\sigma(r_2, m).$$

2. *Muodostetaan Reedin-Mullerin koodin $RM^\sigma(d, m)$ avulla Reedin-Mullerin koodi $RM^\sigma(d - 1, m)$ kuten määritelmän 7.7 kohdassa 1.*

3. *Muodostetaan Reedin-Mullerin koodi*
$$\begin{aligned}&((RM^\sigma(d, m))^\perp \odot RM^\sigma(d - 1, m))^\perp \\ &= (RM^\sigma(m - d - 1, m) \odot RM^\sigma(d - 1, m)^\perp \\ &= (RM^\sigma(m - 2, m))^\perp \\ &= RM^\sigma(1, m).\end{aligned}$$

4. *Etsitään sellainen permutaatiomatriisi σ', jolle pätee, että*
$$RM^{\sigma\cdot\sigma'}(1, m) = RM(1, m),$$
kuten määritelmässä 7.7. Tämä permutaatiomatriisi toteuttaa myös ehdon
$$RM^{\sigma\cdot\sigma'}(r, m) = RM(r, m).$$

5. *Rekonstruoidaan matriisi H' permutaatiomatriisin σ' avulla kuten määritelmässä 7.7.*

(Borodin, M.A. & Chizhov, I.V., 2014)

7.3.2 Sternin hyökkäys

Sternin hyökkäys perustuu pienipainoisten vektoreiden löytämiseen hyökkäyksen kohteena olevaa koodia hieman suuremman koodin avulla. Käytännössä, jos hyökkäyksen kohteena oleva McEliece-koodi on $[n, k]$-koodi, niin Sternin hyökkäyksessä käytetään $[n, k+1]$-koodia. Pienipainoisten vektoreiden avulla voidaan löytää alkuperäisen koodin koodisanoja, joiden avulla saadaan selville koodin rakenne. [3]

Määritelmä 7.10. Olkoon $[n, k]$-McEliece-koodi C, johon hyökätään, $w \geq 0$ ja matriisi H $((n-k)\times n)$-tarkistusmatriisi jollekin lineaariselle $[n, k]$-koodille. *Sternin hyökkäyksessä* etsitään tarkistusmatriisin H avulla pienipainoisia koodisanoja $c \in C$, joiden avulla saadaan selville koodin C rakenne. Pienipainoisen koodisanan etsiminen etenee seuraavasti:

1. Valitaan satunnaisesti $(n-k)$-saraketta tarkistusmatriisista H. Näistä sarakkeista valitaan satunnaisesti l-saraketta, jota kutsutaan osajoukoksi Z. Jaetaan loput k sarakkeista satunnaisesti ja tasaisesti osajoukkoihin X ja Y.

2. Muokataan matriisia H elementaarisilla vaakarivimuunnoksilla niin, että valitut $(n-k)$-saraketta saadaan $((n-k)\times(n-k))$-identiteettimatriisin muotoon. Tämä onnistuu silloin, kun alunperin valitut $(n-k)$ saraketta muodostavat kääntyvän matriisin. Jos näin ei ole, algoritmi aloitetaan alusta valitsemalla toiset $(n-k)$-saraketta.

3. Nyt jokainen valituista $(n-k)$:sta sarakkeesta vastaa yhtä yksikäsitteistä riviä matriisissa H, nimittäin sitä riviä, jolla sijaitsee alkio 1 valitussa sarakkeessa. Esimerkiksi, nyt valituista $(n-k)$-sarakkeista ensimmäinen vastaa matriisin H ensimmäistä riviä, toinen valituista $(n-k)$-sarakkeista vastaa matriisin H toista riviä, ja niin edelleen. Näin ollen myös osajoukon Z kaikki l saraketta vastaavat l riviä matriisissa H. Olkoon $0 \leq p \leq \frac{k}{2}$. Jaetaan osajoukot X ja Y p:n kokoisiin osiin. Merkitään osajoukon X p:n kokoisia osajoukkoja A, ja osajoukon Y p:n kokoisia osajoukkoja B. Jokaisessa osajoukossa A valitaan jokaisesta sarakkeesta ne alkiot, jotka ovat osajoukon Z merkitsemillä l riveillä. Nämä lasketaan yhteen, joilloin saadaan l:n pituisia vektoreita $\pi(A)$. Vastaava toistetaan osajoukolle Y.

4. Kun saadaan sellaiset vektorit $\pi(A)$ ja $\pi(B)$, että $\pi(A) = \pi(B)$, lasketaan näitä vastaavat $2p$ saraketta yhteen joukossa $A \cup B$, jolloin saadaan $(n-k)$:n pituinen vektori. Merkitään tätä vektoria v_{AB}. Jos vektorin v_{AB} paino on $w - 2p$, valitaan tätä vektoria vastaavat sarakkeet

$((n-k)\times(n-k))$-identiteettimatriisissa, eli ne sarakkeet, joissa on samassa kohdassa alkio 1 kuin vektorissa v_{AB}.

5. Valitut sarakkeet muodostavat joukon A ja joukon B kanssa w-painoisen koodisanan $c \in C$ niin, että valitut sarakkeet sekä joukot A ja B merkitsevät alkioiden 1 paikkoja n:n pituisessa vektorissa. Valittujen sarakkeiden ja joukkojen A ja B ulkopuolelle jääville paikoille koodisanassa c tulee alkio 0, jolloin koodisana todella on w-painoinen. [3].

Alla olevassa kuvassa esitetään vaihetta 3. Tämän esimerkin tarkoitus on selkeyttää vaihetta 3. Kuvan esimerkissä on valittu osajoukko Z, kun $l = 3$, ja osajoukot X ja Y, kun $k = 4$. Osajoukoista X ja Y on valittu osajoukot A_1 ja A_2 sekä B_1 ja B_2, kun $p = 1$. Todellisessa tilanteessa $p \geq 2$, koska muuten ei voida laskea saatujen l:n pituisten vektoreiden summia. Tähdellä on merkitty ne rivit, joiden alkioita osajoukkojen X ja Y sarakkeissa tarkastellaan. Kuvassa on ympyröity ne alkiot, joista muodostetaan l:n pituinen vektori. Tällä tavoin muodostetuista vektoreista lasketaan summa $\pi(A_1)$.

	Z			Z			Z	X_{A_1}	Y_{B_1}	Y_{B_2}	X_{A_2}
$*$	1	0	0	0	0	0	0	(1)	0	1	0
	0	1	0	0	0	0	0	0	1	1	1
$*$	0	0	1	0	0	0	0	(1)	1	0	0
	0	0	0	1	0	0	0	0	1	1	0
	0	0	0	0	1	0	0	1	0	1	1
$*$	0	0	0	0	0	1	0	(1)	0	0	1
	0	0	0	0	0	0	1	1	0	0	1

(Bernstein, D.J., Lange, T. & Peters, C., 2008)

Koska yhden koodisanan avulla ei vielä saada selville paljoakaan koodin C rakenteesta, vaatii Sternin hyökkäys sen, että pienipainoinen koodisana etsitään useita kertoja. Yllä olevaa algoritmia voidaan tehostaa, kun sitä toistetaan useita kertoja.

Lause 7.11. *Olkoon tarkistusmatriisi H määritelty kuten määritelmässä 7.10. ja ensimmäinen koodisana $c \in C$ löydetty määritelmän 7.10. algoritmin mukaisesti. Sternin hyökkäystä voidaan tehostaa seuraavilla tavoilla:*

1. *Koska kaikkien saman koodin tarkistusmatriisien H muunnos muotoon $H = \begin{bmatrix} I_{n-k} & P \end{bmatrix}$ tuottaa saman matriisin [3], käytetään samaa tarkistusmatriisin H muunnosta kuin ensimmäisen koodisanan etsinnässä. Näin ollen kun valitaan uudet $(n-k)$ saraketta, on huomattavasti helpompaa saada aikaan uusi $((n-k)\times(n-k))$-identiteettimatriisi matriisin*

H alimatriisiksi, koska hyvin todennäköisesti osa sarakkeista sisältää jo valmiiksi vain yhden alkion 1*.*

2. *Osajoukot X, Y ja Z määritellään kuten määritelmässä 7.10. Joukko Z hajautetaan m osaan algoritmin tehostamiseksi. Luku m on uusi parametri algoritmiin, joka on esitetty määritelmässä 7.10.*

3. *Kun lasketaan vektoreita $\pi(A)$ ja $\pi(B)$, hyödynnetään välimuistia, jottei lasketa samoja sarakkeita yhteen useita kertoja. Tämä on todennäköistä, koska osajoukkojen A ja B sarakkeista valitaan l alkiota, joten samoja l:n pituisten vektoreiden yhteenlaskuja tehdään todennäköisesti useita.*

4. *Kun lasketaan parien (A, B), joille $\pi(A) = \pi(B)$, sarakkeita yhteen joukossa $A \cup B$, hyödynnetään jälleen välimuistia, jotta ei lasketa samoja laskuja useita kertoja.*

(Bernstein, D.J., Lange, T. & Peters, C., 2008)

Sternin hyökkäyksen tehostaminen perustuu suurelta osin jo kerran tehtyjen laskutoimitusten hyödyntämiseen uudestaan tulevilla kierroksilla.

7.3.3 Hyökkäysten tehokkuus

Määritellään merkintä $\mathcal{O}(g(x))$, jolla kuvataan laskennallista kompleksisuutta eli bittioperaatioiden lukumäärää tarkasteltavassa laskutoimituksessa.

Määritelmä 7.12. Olkoon f ja g positiivisia reaaliarvoisia funktioita. Jos on olemassa sellaiset positiiviset luvut A ja B, että

$$f(x) \leq Bg(x) \text{ kaikilla } x > A,$$

niin merkitään $f(x) = \mathcal{O}(g(x))$.

Lause 7.13. *Laskennallisen kompleksisuuden $\mathcal{O}$ laskutoimitukset toimivat seuraavasti:*

1. $$\mathcal{O}(f_1 f_2) = \mathcal{O}(f_1)\mathcal{O}(f_2),$$

2. $$\mathcal{O}(f_1 + f_2) = \max\{(f_1), (f_2)\}.$$

Todistus. Laskennallisen kompleksisuuden $\mathcal{O}$ laskutoimituksista kerrotaan artikkelissa *Formalizing O notation on Isabelle/HOL* (Avigad, J., & Donnelly, K, 2004). [2]. □

Määritelmän 7.7. mukaisessa Minderin-Shokrollahin hyökkäyksessä vaiheen 1 kompleksisuus on kaikista vaiheista selvästi suurin [4]. Tämän vuoksi hyökkäystä saadaan paranneltua tehokkaimmin muokkaamalla vaihetta 1 tehokkaammaksi.

Parannellussa Minder-Shokrollahin hyökkäyksessä vaiheita 1 ja 2 on tehostettu laskemalla Reedin-Mullerin koodit $RM^\sigma(d, m)$ ja $RM^\sigma(d-1, m)$ ja muodostamalla Reedin-Mullerin koodi $RM^\sigma(1, m)$ näiden ja Reedin-Mullerin koodin operaatioiden $\odot$ ja $\perp$ avulla. Näin ollen ei tarvitse laskea jokaista uutta Reedin-Mullerin koodia yksi kerrallaan, ja koodiin $RM^\sigma(1, m)$ päädytään todennäköisesti nopeammin.

Lause 7.14. *On olemassa sellainen algoritmi, jonka avulla voidaan muodostaa Reedin-Mullerin koodi $RM^\sigma(r_1 + r_2, m)$, kun tiedetään Reedin-Mullerin koodi $RM^\sigma(r_1, m)$ ja $RM^\sigma(r_2, m)$. Tämän algoritmin kompleksisuus on $\mathcal{O}(n^4)$, kun $n = 2^m$.*

Todistus. Olkoon $\{f_1, f_2, \ldots, f_{k_1}\}$ koodin $RM^\sigma(r_1, m)$ kanta ja $\{g_1, g_2, \ldots, g_{k_2}\}$ koodin $RM^\sigma(r_2, m)$ kanta. Muodostetaan koodin C kanta vektoreiden f_i ja g_j tulojen $f_i g_j$ avulla, kun $1 \leq i \leq k_1$ ja $1 \leq j \leq k_2$, jolloin saadaan

$$\{f_1 g_1, f_1 g_2, \ldots, f_1 g_{k_2}, f_2 g_1, f_2 g_2, \ldots, f_2 g_{k_2}, \ldots, f_{k_1} g_1, \ldots, f_{k_1} g_{k_2}\}.$$

Todistetaan nyt, että edellä konstruoitu koodi on sama kuin koodi $RM^\sigma(r_1 + r_2, m)$. Ensin todistetaan, että $C \subseteq RM^\sigma(r_1 + r_2, m)$. Koska Reedin-Mullerin koodin $RM(r_1, m)$ kannan $\{f_1^{\sigma^{-1}}, f_2^{\sigma^{-1}}, \ldots, f_{k_1}^{\sigma^{-1}}\}$ ja Reedin-Mullerin koodin RM
(r_2, m) kannan $\{g_1^{\sigma^{-1}}, g_2^{\sigma^{-1}}, \ldots, g_{k_2}^{\sigma^{-1}}\}$ tulon $(f_i)^{\sigma^{-1}}(g_j)^{\sigma^{-1}}$, missä $1 \leq i \leq k_1$ ja $1 \leq j \leq k_2$, aste ei ylitä kantojen asteiden summaa, eli

$$\deg((f_i)^{\sigma^{-1}}(g_j)^{\sigma^{-1}}) \leq \deg(f_i)^{\sigma^{-1}} + \deg(g_j)^{\sigma^{-1}} \leq r_1 + r_2,$$

voidaan sanoa, että $C \subseteq RM^\sigma(r_1 + r_2, m)$.

Sitten todistetaan, että $RM^\sigma(r_1 + r_2, m) \subseteq C$. Jotta tämä voidaan todistaa, täytyy etsiä koodin C dimensiolle alaraja. Koska permutaatio-operaatio ei muuta koodin dimensiota, voidaan etsiä dimension alaraja koodille $C^{\sigma^{-1}}$. Tämä koodi koostuu vektoreista $f_i^{\sigma^{-1}} g_j^{\sigma^{-1}} = f_i' g_j', 1 \leq i \leq k_1, 1 \leq j \leq k_2$, missä $\{f_1, f_2, \ldots, f_{k_1}\}$ on koodin $RM^\sigma(r_1, m)$ kanta ja $\{g_1, g_2, \ldots, g_{k_2}\}$ on koodin $RM^\sigma(r_2, m)$ kanta. Oletetaan, että $r_1 \leq r_2$. Todistetaan, että koodi C sisältää lineaarisesti riippumattomat vektorit, jotka ovat muotoa

$\{1, \{x_{j_1}x_{j_2}\ldots x_{j_s}\}\}$, missä $s = 1, \ldots, r_1 + r_2$ ja $1 \leq j_1 < j_2 < \ldots < j_s \leq m$. Tällaisten vektoreiden lukumäärä on

$$\sum_{i=0}^{r_1+r_2} \binom{m}{i} = \dim RM(r_1 + r_2, m).$$

Tällöin $\dim C = \dim C^{\sigma^{-1}} \geq \dim RM(r_1 + r_2, m) = \dim RM^{\sigma}(r_1 + r_2, m)$.

Käsitellään vektoria $f \in \{1, \{x_{i_1}x_{i_2}\ldots x_{i_t}\}\}$, missä $1 \leq t \leq r_1, 1 \leq i_1 < i_2 < \ldots < i_t \leq m$. Tämä vektori kuuluu sekä koodiin $RM(r_1, m)$ että koodiin $RM(r_2, m)$, joten se voidaan esittää muodossa

$$f = \sum_{i=1}^{k_1} \alpha_i f_i' = \sum_{j=1}^{k_2} \beta_j g_j'.$$

Näin ollen myös

$$f = f \cdot f = \sum_{i=1}^{k_1} \alpha_i f_i' \cdot \sum_{j=1}^{k_2} \beta_j g_j' = \sum_{i,j} \alpha_i \beta_j f_i' g_j'.$$

Koska $f_i' g_j' \in C^{\sigma^{-1}}$, niin myös vektori f kuuluu koodiin $C^{\sigma^{-1}}$.

Käsitellään vektoria $f \in \{x_{i_1}x_{i_2}\ldots x_{i_t}\}$, missä $r_1 + 1 \leq t \leq r_1 + r_2, 1 \leq i_1 < i_2 < \ldots < i_t \leq m$. Vektori f voidaan esittää muodossa

$$f = x_{i_1}x_{i_2}\ldots x_{i_{r_1}}x_{i_{r_1+1}}\ldots x_{i_t},$$

missä $x_{i_1}x_{i_2}\ldots x_{i_{r_1}} \in RM(r_1, m)$ ja $x_{i_{r_1+1}}\ldots x_{i_t} \in RM(r_2, m)$, koska $1 \leq t - r_1 \leq r_2$. Näin ollen voidaan merkitä, että

$$x_{i_1}x_{i_2}\ldots x_{i_{r_1}} = \sum_{i=1}^{k_1} \alpha_i f_i'$$

ja

$$x_{i_{r_1+1}}\ldots x_{i_t} = \sum_{j=1}^{k_2} \beta_j g_j',$$

jolloin

$$f = \sum_{i=1}^{k_1} \alpha_i f_i' \cdot \sum_{j=1}^{k_2} \beta_j g_j' = \sum_{i,j} \alpha_i \beta_j f_i' g_j'.$$

Koska $f_i' g_j' \in C^{\sigma^{-1}}$, niin myös vektori f kuuluu koodiin $C^{\sigma^{-1}}$, jolloin voidaan sanoa, että $RM^{\sigma}(r_1 + r_2, m) \subseteq C$. Nyt siis $C = RM^{\sigma}(r_1 + r_2, m)$, jolloin lauseen ensimmäinen osa on todistettu.

Seuraavaksi täytyy estimoida koodin $RM^{\sigma}(r_1+r_2, m)$ konstruoimisen kompleksisuus. Olkoon $\{f_1, f_2, \ldots, f_{k_1}\}$ jokin koodin $RM^{\sigma}(r_1, m)$ kanta ja $\{g_1, g_2, \ldots, g_{k_2}\}$ jokin koodin $RM^{\sigma}(r_2, m)$ kanta, ja olkoon joukko $L = \{h_1, h_2, \ldots, h_k\}$, joka koostuu kaikista lineaarisesti riippumattomista tuloista $f_i g_j$, missä $1 \leq i \leq k_1$ ja $1 \leq j \leq k_2$. Näin ollen joukko L on sama kuin koodi C. Näin ollen koodin C konstruoimiseksi on järkevää muodostaa joukko L sen kannaksi. Joukko L muodostetaan seuraavasti: aloitetaan niin, että $L = \{f_1 g_1\}$ olettaen, että $f_1, g_1 \neq 0$. Lisätään joukkoon L vektori $f_1 g_2$ ja selvitetään joukon L aste muodostamalla joukon L vektoreista matriisi, ja muuttamalla tämä yläkolmiomuotoon. Tähän tarvitaan yksi bittikohtainen operaatio, eli kahden vektorin yhteenlasku. Jos matriisin aste on alle kaksi eli vektorit ovat lineaarisesti riippuvaisia, unohdetaan vektori $f_1 g_2$, valitaan seuraava vektori ja toistetaan sama tarkastelu. Näin ollen on olemassa jokin luku x_1, joka on n:n pituisten vektorien kertolaskujen maksimimäärä, ennen kuin löydetään kaksi toisistaan riippumatonta vektoria. Sitten muodostetaan vastaavasti kolmen vektorin lineaarisesti riippumaton systeemi. Koska valmiiksi on jo kahden vektorin muodostama yläkolmiomuodossa oleva matriisi, kolmen vektorin muodostaman yläkolmiomuodossa olevan matriisin muodostamiseen menee kaksi n:n pituisten vektoreiden yhteenlaskua. Jos on mennyt x_2 yritystä löytää sopiva kolmas vektori, ollaan näin ollen suoritettu $2x_2$ n:n pituisten vektoreiden yhteenlaskua. Kun jatketaan näin, kokonaisen koodin konstruoimiseen menee

$$N = \sum_{i=1}^{k(1,2)-1} (i \cdot x_i)$$

n:n pituisten vektoreiden yhteenlaskua, missä $k(1,2)$ on koodin $RM^{\sigma}(r_1 + r_2, m)$ dimensio. Koska kokeiltujen vektoreiden kokonaismäärä on pienempi kuin tulojen $f_i g_j$ lukumäärä, eli

$$\sum_{i=1}^{k(1,2)-1} x_i \leq k_1 \cdot k_2,$$

voidaan arvioida lukua N seuraavasti:

$$N \leq k(1,2) \cdot \sum_{i=1}^{k(1,2)-1} x_i \leq k(1,2) \cdot k_1 \cdot k_2.$$

Koska $k(1,2), k_1, k_2 \leq n$, niin joukon L muodostamisen kompleksisuus N on pienempi kuin n^3. Koska joukon L muodostamisen kompleksisuus N on maksimissaan n^3, ja koodin C alkioiden maksimipituus on n, niin koodin C kannan muodostamisen kompleksisuus on

$$\mathcal{O}(N \cdot n) = \mathcal{O}(n^3 \cdot n) = \mathcal{O}(n^4).$$

□

(Borodin, M.A. & Chizhov, I.V., 2014)

Seuraus 7.15. *Tästä seuraa, että operaation $RM^\sigma(r_1, m) \odot RM^\sigma(r_2, m)$ kompleksisuus on $\mathcal{O}(n^4)$.*

Lause 7.16. *Reedin-Mullerin koodin $RM^\sigma(m - r - 1, m)$ muodostamisen kompleksisuus on $\mathcal{O}(n^3)$ kun tiedetään Reedin-Mullerin koodi $RM^\sigma(r, m)$, ja n on koodin $RM^\sigma(r, m)$ pituus.*

Todistus. Tiedetään (Borodin, M.A. & Chizhov, I.V., 2014), että koodin $(RM^\sigma(r, m))^\perp = RM^\sigma(m-r-1, m)$ generoijamatriisi $(R^\sigma)^\perp$ saadaan muokkaamalla koodin $RM^\sigma(r, m)$ generoijamatriisia R^σ elementaarisin vaakarivimuunnoksin. Koodin $RM^\sigma(m - r - 1, m)$ generoijamatriisin $(R^\sigma)^\perp$ tällä tavoin laskemisen kompleksisuus on $\mathcal{O}(n^3)$, missä n on koodin $RM^\sigma(r, m)$ pituus.[4] □

Seuraus 7.17. *Tästä seuraa, että operaation $(RM^\sigma(r, m))^\perp$ kompleksisuus on $\mathcal{O}(n^3)$.*

Paranneltu versio Minderin-Shokrollahin hyökkäyksestä tehostaa hyökkäyksen ensimmäistä vaihetta operaatioiden $\odot$ ja $\perp$ avulla. Operaation $\odot$ kompleksisuus on $\mathcal{O}(n^4)$ ja operaation $\perp$ kompleksisuus on $\mathcal{O}(n^3)$. Varmaksi ei kuitenkaan voida sanoa, että se olisi tehokkaampi kaikkiin McEliece-koodeihin tapahtuviin hyökkäyksiin kuin alkuperäinen Minderin-Shokrollahin hyökkäys. Hyökkäyksen ensimmäisen vaiheen tehostaminen kuitenkin tehostaa hyökkäystä melko todennäköisesti. [4]

Alkuperäisessä Sternin hyökkäyksessä vaiheen 2 suorittaminen vaatii jokaisella kierroksella noin

$$\frac{1}{2}(n-k)^3 + k(n-k)^2$$

bittioperaatiota. Parannellussa Sternin hyökkäyksessä saman vaiheen suorittaminen toisesta kierroksesta eteenpäin vaatii noin

$$\frac{k^2(n-k)(n-k-1)(3n-k)}{4n^2}$$

bittioperaatiota, mikä on huomattavasti vähemmän. Tämä johtuu siitä, että parannellussa Sternin hyökkäyksessä hyödynnetään ensimmäisen koodisanan c löytämisen jälkeen ensimmäisellä kierroksella laskettua tarkistusmatriisin H muunnosta. [3]

Tarkistusmatriisin H edellisen kierroksen muunnoksen hyödyntämisen lisäksi voidaan myös muita ensimmäisellä kierroksella tehtyjä operaatioita hyödyntää tulevilla kierroksilla. Alkuperäisessä Sternin hyökkäyksessä vektorien $\pi(A)$ ja $\pi(B)$ laskemiseen tarvitaan

$$2lp\binom{k/2}{p}$$

bittioperaatiota. Koska tarkistusmatriisin H muunnos on nyt sama joka kierroksella, on hyvin todennäköistä, että vektoreita $\pi(A)$ ja $\pi(B)$ laskiessa toistetaan samoja laskutoimituksia. Välimuistin hyödyntäminen tässä vaiheessa pienentää tarvittavien bittioperaatioiden määrää. Samoin vaihetta 4 voidaan tehostaa huomattavasti, kun ei toisteta aikaisemmin tehtyjä laskutoimituksia.

McEliece-koodaussysteemiä voidaan muokata turvallisemmaksi vastaamaan tehokkaita hyökkäyksiä koodaussysteemiä vastaan. Yksinkertainen ja intuitiivisesti järkevä tapa parantaa McEliece-koodaussysteemin turvallisuutta on kasvattaa koodisanojen c pituutta eli valitsemalla suurempi n. Jotta dekoodaus pysyisi myös tehokkaana, kannattaa valita hyvin iso $n = 2^m$, jolloin McEliece-koodaussysteemi pysyisi myös Goppa-koodina.

Viitteet

[1] *Algebran perusteet*, Oulun Yliopisto, luentomoniste, 2022

[2] Avigad, J. & Donnelly, K. (2004), *Formalizing O notation on Isabelle/HOL*, Automated Reasoning: Second International Joint Conference, sivut 357-372, Springer

[3] Bernstein, D.J., Lange, T. & Peters, C. (2008), *Attacking and defending the McEliece cryptosystem*, Post-Quantum Cryptography: Second International Workshop, sivut 31-47, Springer

[4] Borodin, M.A. & Chizhov, I.V. (2014), *Effective attack on the McEliece cryptosystem based on Reed-Muller codes*, Discrete Mathematics and Applications, 24(5), sivut 273–280.

[5] Courtois, N.T., Finiasz, M. & Sendrier, N. (2001), *How to Achieve a McEliece-based Digital Signature Scheme*, ASIACRYPT 2001, sivut 157–174.

[6] Delfs, H. & Knebl, H. (2007), *Introduction to cryptography: Principles and Applications*, s. 137-156, Springer-Verlag Berlin Heidelberg.

[7] *Johdatus koodausteoriaan*, Oulun Yliopisto, luentomoniste, 2022

[8] MacWilliams, F.J. & Sloane, N.J.A. (1977), *The theory of error correctiong codes*, s. 370, North-Holland Pub. Co.

[9] *Matriisiteoria*, Oulun Yliopisto, luentomoniste, 2022

[10] Stinson, D.R. & Paterson M.B. (2019). *Cryptography: Theory and Practice*, s. 353-358, Taylor & Francic Group.