VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
INFORMATIKOS KATEDRA

Magistro baigiamasis darbas

Autentifikavimo schemos, naudojančios klaidas taisančius kodus

Authentication Schemes Using Error-Correcting Codes

Atliko:

Svetlana Krasnik (parašas)

Darbo vadovas:

doc. Gintaras Skersys (parašas)

Recenzentas:

doc. Vilius Stakėnas (parašas)

Vilnius

2006

Turinys

1. Įvadas

Autentiškumo užtikrinimo problema yra viena iš pagrindinių kriptografijos problemų.

Autentifikavime išskiriami trys pagrindiniai veikėjai: siuntėjas, gavėjas ir oponentas. Siuntėjas siunčia pranešimą nesaugiu kanalu gavėjui. Gavėjas, gavęs pranešimą, nori būti užtikrintas, kad tas pranešimas nebuvo sufalsifikuotas ir buvo siųstas siuntėjo, o ne kieno nors kito. Trečias veikėjas – oponentas – nori apgauti arba suklaidinti gavėją, modifikuodamas originalų pranešimą arba apsimesdamas siuntėju. Egzistuoja keli metodai šiai problemai išspręsti, tokie kaip specialios kriptografinės schemos, tarp jų ir skaitmeninis parašas. Čia taip pat galima paminėti ir autentifikavimo kodus.

Tarp klaidas taisančių kodų ir autentifikavimo kodų egzistuoja ryšys: autentifikavimo kodai gali būti sukonstruoti iš klaidas taisančių kodų ir atvirkščiai.

Literatūros apžvalgoje yra aprašyta autentifikavimo schema, aprašyti autentifikavimo kodai bei jų ryšys su klaidas taisančiais kodais, taip pat pateiktas būdas, kaip gali būti sukonstruotas autentifikavimo kodas iš klaidas taisančio kodo.

Šiame darbe siekiama ištirti su kokiais klaidas taisančiais kodais gaunami geriausi autentifikavimo kodai. Iš pradžių bus suformuluoti kriterijai, pagal kuriuos bus lyginami autentifikavimo kodai ir sprendžiama, ar jie yra efektyvūs. Taip pat bus nustatyti reikalavimai, kuriuos turi tenkinti klaidas taisantys kodai, kad iš jų sukonstruoti autentifikavimo kodai būtų efektyvūs. Atsižvelgiant į nustatytus reikalavimus, siekiama pasiūlyti konkretų klaidas taisantį kodą, iš kurio gali būti sukonstruotas efektyvus autentifikavimo kodas. Taip pat planuojama išbandyti praktiškai autentifikavimo kodo konstravimo procesą ir remiantis gautais teoriniais bei praktiniais rezultatais suformuluoti rekomendacijas, kurios praverstų renkantis klaidas taisantį kodą, norint sukonstruoti efektyvų autentifikavimo kodą.

2. Literatūros apžvalga

Šiame skyrelyje pateikiama literatūros, susijusios su klaidas taisančių kodų naudojimu autentifikavimo schemose, apžvalga.

2.1 Klaidas taisančių kodų panaudojimas kriptografijoje

Kriptografija (remiantis [Sta02])– tai mokslas, nagrinėjantis matematines priemones ryšio saugumui užtikrinti.

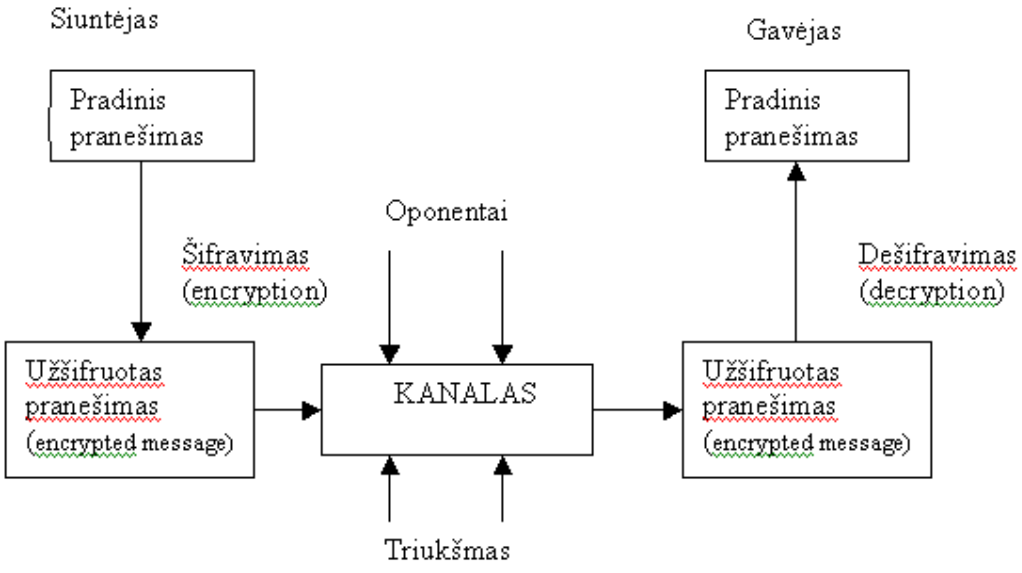
Pagrindiniai kriptografijos uždaviniai yra šie:

- perduodamos informacijos slaptumo užtikrinimas,
- perduodamos informacijos vientisumo ir autentiškumo patvirtinimas,
- ryšio subjektų identiteto patikrinimas,
- išsižadėjimo paneigimas.

Taigi, pagrindinę kriptografijos problemą galima suformuluoti taip:

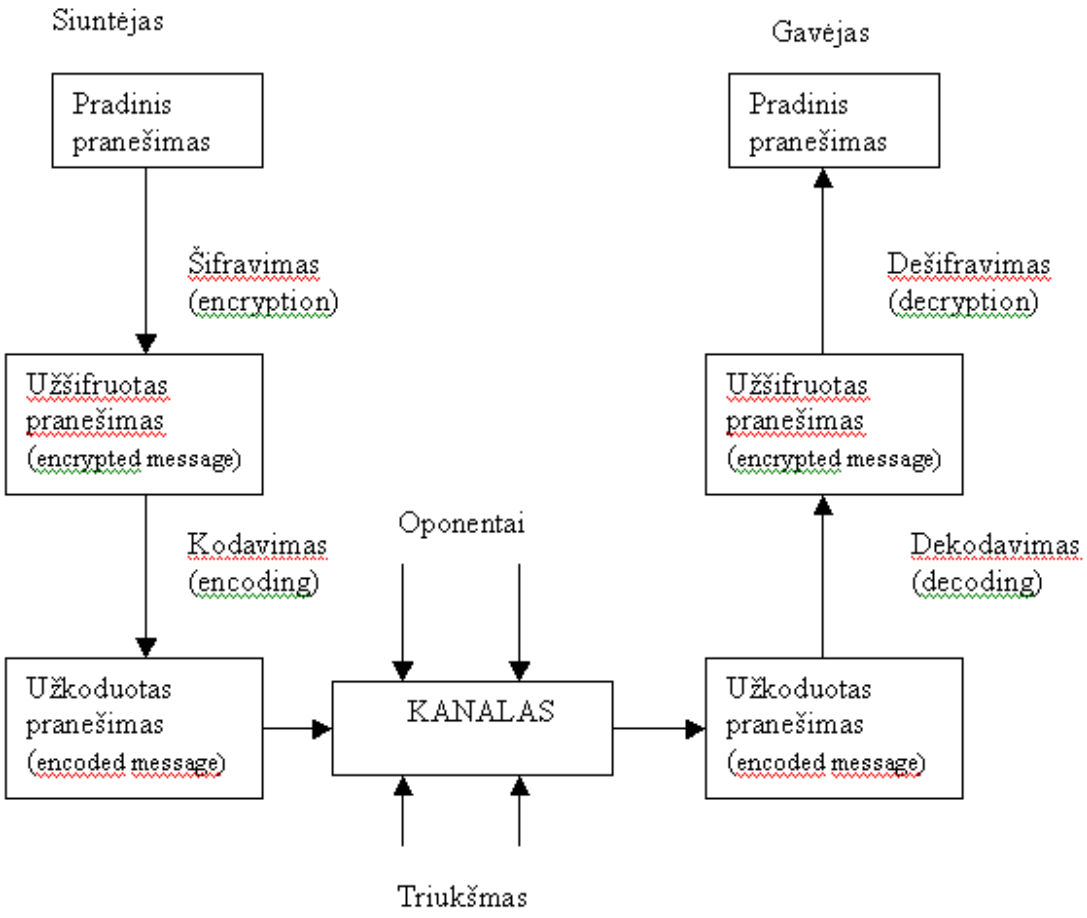
Siuntėjas nori nusiųsti pranešimą gavėjui taip, kad kiti nesužinotų pranešimo turinio.

Kriptografiniai metodai, leidžiantys išspręsti šią problemą, remiasi tokia idėja (pagal [Ada91]): pranešimą reikia užšifruoti (encrypt) ir siųsti kanalu užšifruotą pranešimą. Dešifruoti (decrypt) pranešimą gali tik teisėtas gavėjas, vadinasi, tik jis sužinos pranešimo turinį. Šią idėją schematiškai galima pavaizduoti taip:



Pav.1 Kriptografinė schema pranešimo slaptumui užtikrinti

Kanale dažnai būna triukšmas, kuris gali paveikti pranešimą, t.y. jį transformuoti. Net nežymių pakeitimų pasekme gali būti tai, kad gavėjas nesugebės teisingai dešifruoti pranešimą. Vienas iš būdų šią problemą išspręsti – tai naudoti klaidas taisančius kodus, kurie gali aptikti ir ištaisyti klaidas iš kanalo išėjusiame pranešime. Reikia suderinti kriptografijos schemą su klaidas taisančiu kodu. Sprendimas pavaizduotas pav.2.



Pav.2. Kriptografinė schema su klaidas taisančiu kodu

Siuntėjas užšifruoja pranešimą ir prieš siunčiant jį kanalu, dar užkoduoja pagal vieno iš klaidas taisančių kodų taisykles(encode). Pranešimas keliauja kanalu, kuriame gali būti triukšmas. Kai gavėjas gauna pranešimą, jis, naudodamas klaidas taisančio kodo dekodavimo taisyklę, dekoduoja pranešimą (tuo pačiu yra ištaisomos klaidos, jei jos yra) ir jau ištaisytam pranešimui taiko kriptografinės schemos dešifravimo taisyklę, kad gautų pradinį tekstą.

Yra du būdai naudoti klaidas taisančius kodus kriptografijoje:

1) Naudoti klaidas taisanti kodą klaidų ištaisymui (apie tai buvo šnekėta aukščiau, žr. Pav.2);

2) Šifravimas su klaidas taisančiu kodu (remiantis [Ada91]).

Kiekvienas dvejetainis tiesinis (n,k) kodas K gali būti naudojamas šifravimui (encryption).

Įveskime tokius žymėjimus:

H – kontrolinė matrica,

n – kodo žodžio ilgis,

w – kodo žodis,

s – sindromas.

Kiekvienam kodo žodžiui w galima paskaičiuoti sindromą s:

$$H*w^T = s^T.$$

Sindromo ilgis lygus n-k, kadangi $H_{(n-k)*n} * w^T_{n*1} = s^T_{(n-k)*1} \Rightarrow s_{(n-k)}$.

Iš viso turėsime 2^k žodžių. Visi žodžiai w, kurių sindromai lygūs sudaro klasę.

Pradinį pranešimą reikia išskaidyti n-k ilgio blokais s. Kiekvienam blokui galima parinkti atsitiktinai žodį w iš klasės visų žodžių su sindromu s, kitaip tariant kiekvieną bloką s mes užšifruojame žodžiu w. Taigi, pradinis n-k ilgio žodis šifruojamas n ilgio žodžiu:

$$\text{Informacijos lygis} = \frac{n-k}{n} .$$

Dešifruoti lengva: gauname žodį w ir ieškome jį atitinkantį sindromą s, kuris ir bus - dešifruotas žodis.

Klaidas taisantys kodai gali būti naudojami ir viešojo rakto kriptosistemose, ir privataus rakto kriptosistemose.

2.1.1 Klaidas taisančių kodų naudojimas viešo rakto kriposistemose

Priminkime, kad viešo rakto kriptosistemose šifravimo taisyklė (arba šifravimo raktas, encryption key) yra viešai skelbiamas, todėl bet kas panorėjęs gali užšifruoti pranešimą ir nusiųsti gavėjui. Reikia pabrėžti, kad šifravimas tokiose sistemose nėra sunkus procesas, kadangi šifravimo raktai yra visiems žinomi. Tačiau dešifravimas viešo rakto kriptosistemose yra sudėtingas, praktiškai neįmanomas atlikti, nežinant slapto dešifravimo rakto. Taigi niekas, išskyrus gavėją, kuriam buvo adresuotas pranešimas, negali dešifruoti pranešimo. Pats gavėjas

gali lengvai dešifruoti jam adresuotą pranešimą ir jį perskaityti, kadangi jis žino slaptą dešifravimo raktą.

Egzistuoja keli klaidas taisančių kodų panaudojimo viešo rakto kriptosistemose būdai, tačiau mes pateiksime čia vieną kaip pavyzdį.

2.1.1.1 McEliece viešo rakto kriptosistema

Paimkime t klaidų taisantį tiesinį kodą virš GF(2), kurio generuojanti matrica yra G_{k*n}.

Priminkime, kad kodas vadinamas t klaidas taisančiu kodu, jei naudojant minimalaus atstumo taisyklę jis teisingai dekoduoja žodį, kuriame siuntimo metu įvyko ne daugiau, kaip t iškraipymų.

Kodo koeficientas lygus $\frac{k}{n}$.

Tegul (pagal [RN87])

S - atsitiktinai pasirinkta k*k nevienetinė matrica, taip vadinamas šifratorius,

P – atsitiktinai pasirinkta n*n perstatos matrica,

G’ – vieša generuojanti matrica, G’ = SGP,

M – pradinis ilgio k pranešimas,

C – ilgio n šifras,

Z – atsitiktinai parinktas ilgio n klaidų vektorius, kurio Hamingo svoris lygus t.

Raktai:

(t, G’) – viešas raktas,

(S, G, P) – privatus raktas.

Šifravimas

Pradinis tekstas suskaidomas ilgio k žodžiais M, kiekvienas žodis užšifruojamas tokiu būdu:

C = MG’ + Z.

Panagrinėkime plačiau, kas atliekama šiame žingsnyje.

Kadangi G’ = SGP, tai C = MG’ + Z = MSGP + Z = M’GP +Z, kur M’ =MS.

Dešifravimas

Gavėjas gauna šifrą C.

Dešifravimas susideda iš kelių žingsnių:

1) Gavėjas skaičiuoja

$$C' = CP^T = (M'GP + Z)P^T = M'G + ZP^T = M'G+Z',$$

kur $Z' = ZP^T$.

2) Dekodavimas ir klaidų taisymas.

   Šiame žingsnyje gali būti taikomi įvairūs dekodavimo algoritmai, pvz., Petersono algoritmas, Barlekamp-Massey algoritmas ir kiti. Dekoduojant, kartu ištaisomos klaidos, jei jos yra. Šio žingsnio rezultate, mes gauname $M' = MS$.

3) Atgaminamas pradinis pranešimas:

$M = M'S^{-1}$.

Tretieji asmenys arba priešai žino užšifruotą pranešimą C ir G', bei šifravimo būdą. Tačiau, jie negali dešifruoti C tol, kol nesužinos privatų raktą – matricas S, G ir P, kurios buvo pasirinktos atsitiktinai iš daugelio galimybių.

Panagrinėkime galimas atakas([Til98], [RN87]).

Ataka 1.

Oponentas gali bandyti atspėti matricas S ir P, kad galėtų paskaičiuoti generuojančią matricą G iš G'. Jei jam pavyktų tai padaryti, jis galėtų dešifruoti pranešimą M. Tačiau matricų S ir P parinkimo variantų skaičius yra labai didelis, todėl oponentui bus ypač sunku parinkti teisingas S ir P matricas.

Ataka 2.

Oponentas galėtų palyginti šifrą C su visais žodžiais, generuotais su matrica G', ir rasti arčiausią žodį. Taigi, oponentas sužinos MG', o iš čia jis gali atstatyti ir M (Gauso eliminavimo būdu). Šiai atakai prireiks 2^k palyginimų.

Ataka 3.

Ši ataka panaši į antrą ataką. Oponentas gali rasti šifrui C artimiausią žodį, skaičiuojant sindromą. Toliau tęsti kaip Ataka 2. Šiam būdui reikia mažiau bandymų, nei antrai atakai, kadangi šiuo atveju bandymu skaičius yra proporcingas sindromų skaičiui ($\approx 2^{n-k}$).

Ataka 4.

Tegul

$M = m_1\ m_2\ m_3 .... m_k$

$C = c_1\ c_2\ c_3 ..... c_k ..... c_n$

$Z = z_1\ z_2\ z_3 ..... z_k ..... z_n$

$G' = [\ G_{ij}'\ ],\ i = 1, ..., k$

$j = 1, ..., n$

(t klaidas taisančio kodo generuojanti matrica)

Oponentas gali bandyti spręsti tokią lygčių sistemą:

$c_1 = m_1 G_{11}' + m_2 G_{21}' + . . . + m_k G_{k1}' + z_1$

$c_2 = m_1 G_{12}' + m_2 G_{22}' + . . . + m_k G_{k2}' + z_2$

... ...

... ...

$c_n = m_1 G_{1n}' + m_2 G_{2n}' + . . . + m_k G_{kn}' + z_n$

Tam, kad išspręsti k nežinomųjų (m_1 , m_2 , m_3, ... m_k) lygčių sistemą, oponentui reikia apytiksliai k^3 žingsnių.

Kadangi t yra mažesnis, nei n-k, tai oponentas gali bandyti atspėti k lygčių, kuriose nėra klaidų, t.y. jis gali parinkti k lygčių iš n galimų ir spręsti pasirinktas lygtis su prielaida, kad jose nėra klaidų. Tikimybė, kad k parinktų lygčių neturi klaidų yra:

$$P_k = \prod_{i=0}^{k-1} (1 - \frac{t}{n-i}).$$

Oponentas gali parinkinėti ir spręsti lygtis iki tol, kol negaus prasmingos M. Taigi, lygčių parinkimas ir sprendimas gali kartotis $(P_k)^{-1}$ kartų. Todėl, vidutiniškai oponentas turės atlikti $k^3 * (P_k)^{-1}$ žingsnių (tačiau čia neįskaičiuoti žingsniai, skirti patikrinimui, ar M turi prasmę).

Ketvirta ataka reikalauja mažiausiai žingsnių, nei pirmos trys atakos, tačiau vis tiek žingsnių skaičius lieka pakankamai didelis.

Ką tik aprašytoje McEliece viešo rakto kriptosistemoje siūloma naudoti Goppa kodus. Tačiau ši kriptosistema reikalauja daug papildomų skaičiavimų, be to blokų ilgis turi būti gana didelis, kad sistema galėtų efektyviai ištaisyti daug klaidų ($n \approx 1000$ bitų, $t \approx 50$). Todėl buvo pasiūlytas dar vienas sprendimas – McEliece viešo rakto kriptosistemos modifikacija – privataus rakto kriptosistema, naudojanti klaidas taisančius kodus. Modifikuota sistema labai panaši į pirmąją, bet vieša matrica G' modifikuotoje sistemoje laikoma privačia. Kitame skyrelyje pateikiamas detalesnis modifikuotos McEliece kriptosistemos aprašymas.

2.1.1.2 Modifikuota McElice kriptosistema

Kaip jau buvo minėta, ši kriptosistema (remiantis [RN87]) buvo pasiūlyta siekiant sumažinti skaičiavimų kiekį (būtent šifravimo ir dešifravimo metu). Be to, šioje kriptosistemoje numatytas paprastesnių klaidas taisančių kodų naudojimas, tokių, kaip Hemingo kodai arba BCH kodai su atstumu 5.

Visos matricos S, G, P ir G' – laikomos paslaptyje, kad būtų pasiektas didesnis saugumo lygis. Taigi, (S, G, P, G') – privatus raktas.

Šifravimas

Visų pirma skaičiuojama G' = SG, kur

S – k*k nevienetinė matrica,

G – k*n klaidas taisančio kodo generuojanti matrica,

G' – k*n kodavimo matrica.

Toliau šifruojamas pradinis pranešimas tokiu būdu:

C = (MG' + Z)P, kur

M – pradinis ilgio k pranešimas,

C – užkoduotas ilgio n pranešimas,

P – n*n perstatos matrica,

Z – atsitiktinai parinktas klaidų vektorius.

Dešifravimas

Panagrinėkime šifravimo algoritmą:

$C = (MG' + Z)P = MG'P + ZP = MSGP + ZP = M'GP + ZP$, kur $M' = MS$.

Dešifravimas gali būti atliekamas, naudojant privatų raktą. Dešifravimui reikės S^{-1}, H^T (gali būti išskaičiuota iš G, kadangi $G H^T = 0$) ir P^T. Dešifravimas susideda iš trijų žingsnių:

1) Reikia paskaičiuoti C' :

$$C' = C P^T = M'G + Z$$

2) Šiame žingsnyje ištaisomos klaidos M'. Šiam tikslui gali būti naudojama sindromų lentelė. Tada skaičiuojamas sindromas:

$$C' H^T = M' G H^T + Z H^T = Z H^T \text{ (Sindromas)} .$$

Lentelėje ieškomas gautas sindromas ir nustatomas jį atitinkantis klaidų vektorius, pagal kurį ištaisomos klaidos M'.

3) Atgaminamas pradinis pranešimas:

$$M = M'S^{-1}.$$

Praeitame skyrelyje buvo pateikta bendra schema, kaip gali būti panaudoti klaidas taisantys kodai kriptografijoje. Taigi, klaidas taisančius kodus galima derinti su kriptografiniais protokolais: naudoti klaidas taisanti kodą klaidų ištaisymui (žr. Pav.2) ir šifravimas su klaidas taisančiu kodu. Šiuos du būdus galima panaudoti autentiškumo užtikrinimo užduočiai spręsti.

Pirmo būdas: naudoti kriptografinį protokolą, kuris užtikrina autentiškumą (pvz. Šnoro schemą, Fiat-Šamiro schema), o klaidas taisantį kodą naudoti tik klaidų taisymui. Šiuo atveju klaidas taisantis kodas autentiškumo užtikrinime tiesiogiai nedalyvauja, kadangi tai atlieka autentifikavimo protokolas, o klaidas taisantis kodas užtikrina klaidų prevenciją.

Antras būdas (pavyzdžiu gali būti McElice viešo rakto kriptosistema, kurios aprašymas pateiktas 2.1.1.1 skyrelyje) leidžia sukurti autentifikavimo sistemą, naudojant klaidas taisančius kodus, kuri pasižymės tokiomis savybėmis:

- šifravimas nėra sudėtingas,
- gavėjas žinodamas privatų raktą (matricas S, G, P) gali lengvai dešifruoti pranešimą,
- apsaugo nuo apsimetimo ir turinio pakeitimo atakų.

Be šių dviejų būdų egzistuoja kitos galimybės užtikrinti autentiškumą. Tai autentifikavimo kodai (A-kodai). Panagrinėkime juos plačiau.

2.2 Autentifikavimo kodai

Autentifikavimo kodai (arba A-kodai) buvo sukurti apsimetimo ir turinio pakeitimo atakų prevencijai (pagal [Jon03]).

Priminkime, kad apsimetimo ataka – tai oponento bandymas apsimesti siuntėju ir nusiųsti gavėjui apgaulingą pranešimą, o turinio pakeitimo ataka – tai oponento bandymas perimti ir padaryti pakeitimus originaliame pranešime, o modifikuotą pranešimą persiųsti toliau gavėjui.

Atkreipkime dėmesį į tai, kad tie kodai, kuriuos toliau nagrinėsime, neužtikrina informacijos slaptumo, t.y. jie apsaugo nuo apsimetimo ir turinio pakeitimo atakų, bet oponentas žino kokia informacija siunčiama kanalu.

Įveskime tokius žymėjimus:

P_I - didžiausia apsimetimo atakos sėkmės tikimybė,

P_S - didžiausia turinio pakeitimo atakos sėkmės tikimybė,

P_D - didžiausia pavykusio sukčiavimo tikimybė.

Bendru **A-kodu** laikomas ketvertas (S, E, M, g), kur

S – baigtinė pradinių būsenų arba pradinių pranešimų aibė,

E – baigtinė kodavimo taisyklių aibė,

M – baigtinė pranešimų aibė,

g: S*E → M - kiekvienam elementui e iš aibės E paverčia aibę S į ilgio |S| aibę M.

Kitaip tariant, kiekviena taisyklė e∈ E apibrėžia atvaizdį S į M.

P_I ir P_S išraiškos tokios:

$$P_I = \max_{m \in M} P \text{ (m – priimtas)},$$

$$P_S = \max_{\substack{m,m' \in M \\ m \neq m'}} P(\text{m' – priimtas vietoj m}),$$

sąlyga m ≠ m' reiškia, kad m' bus dekoduota kitaip, nei m (t.y. dekodavę m' mes gausime kitą pradinį pranešimą, kuris skirsis nuo to, kuris buvo užkoduotas).

Apgavystės tikimybė P_D apibrėžiama taip:

$$P_D = \max (P_I , P_S).$$

Jei A-kodo pranešime $m \in M$ figūruoja pradinis pranešimas s, tai toks kodas vadinamas **sistematiniu A-kodu,** kai kuriose literatūros šaltiniuose dar vadinamas Dekarto A-kodu (Cartesian A-code).

Pateikime atskirą sistematinių kodų apibrėžimą:

Sistematiniu A-kodu laikomas ketvertas (S, E, Z, f), kur

S – baigtinė pradinių pranešimų arba pradinių būsenų aibė,

E – baigtinė kodavimo taisyklių aibė,

Z – autentifikatorių aibė,

f: $S*E \rightarrow Z$ - kiekvienam $e \in E$ ir $s \in S$ turime pranešimą $m = (s,z) \in S*Z$, kur $z = f(s,e)$.

Kitaip tariant, e nusako atvaizdį $S \rightarrow Z$.

Elementas z vadinamas autentifikatoriumi.

Visų pranešimų aibe M galima apibrėžti tokiu būdu:

$$M = \{ m = (s, f(s,e)): s \in S, e \in E\}.$$

Kaip matome, sistematinio kodo pranešimo sudedamoji dalis – neužkoduotas pradinis pranešimas s.

Kodavimą galime aprašyti taip:

$$Z=e(S),$$

t.y. $\forall$ e nusako atvaizdi $S \rightarrow Z$.

Taigi, $\forall (s,z) \in (S,Z)$, $E(s,z) = \{ e \in E : e(S)=z \}$.

Kodavimo unikalumą galima apibrėžti taip:

$E(s,z) \cap E(s,z') = \varnothing$, $z \neq z$

$\forall \; s \in S$, $E = \bigcup_{z} E(s,z)$.

Gavėjas gali patikrinti gauto pranešimo (s,z) autentiškumą tokiu būdu:

Reikia patikrinti, ar $z = f(s,e)$(arba $e(s) = z$), kur e – (slapta) kodavimo taisykle, žinoma tik siuntėjui ir gavėjui.

Tarkime, kad oponentas turi galimybę siųsti naujus pranešimus į kanalą ir modifikuoti pranešimus, kuriuos siuntėjas siunčia gavėjui. Be to, tarkime, kad oponentas žino viską apie autentifikavimo schemą išskyrus konkrečią kodavimo taisyklę, kurią tuo laiko momentu naudoja siuntėjas ir gavėjas.

Sistematiniams kodams kitaip apibrėžiami P_I ir P_S:

$$P_I = \max_{m \in M, s \in S} P((s,z) - \text{priimtas}) = \max_{m \in M, s \in S} \frac{|E(s,z)|}{|E|}$$

Turinio pakeitimo ataka:

Oponentas perima pranešimą m = (s,z), pakeičia jį kitu pranešimu m'=(s',z'), $s \neq s'$ ir siunčia toliau gavėjui.

$$P_S = \max_{\substack{(s',z') \\ s' \neq s}} P((s',z') - \text{priimtas} | (s,z) - \text{buvo siųstas}) =$$

$$= \max_{\substack{(s,z),(s',z') \in S*Z \\ s' \neq s}} \frac{|\{e \in E \mid e(s) = z, e(s') = z'\}|}{|\{e \in E \mid e(s) = z\}|}.$$

Sistematiniams kodams teisinga tokia teorema:

Teorema. Bet kuriam sistematiniam kodui teisinga tokia nelygybė:

$$P_S \geq P_I .$$

Įrodymas pateiktas [JKS96].

2.2.1 Ryšys tarp autentifikavimo kodų ir klaidas taisančių kodų

Buvo pastebėta (remiantis [JKS94]), kad tarp A-kodų ir klaidas taisančių kodų egzistuoja ryšys – iš A-kodo galima sukonstruoti klaidas taisantį kodą ir atvirkščiai, iš klaidas taisančio kodo galima sukonstruoti A-kodą. Panagrinėkime šį ryšį plačiau.

Pažymėkime n = |E|, q=|Z|.

Atkreipkime dėmesį, kad mus domi keturi pagrindiniai parametrai:

- apsimetimo atakos sėkmės tikimybė P_I ,
- turinio pakeitimo atakos sėkmės tikimybė P_S ,
- rakto ilgis,
- A-kodo kardinalumas (pradinių pranešimų skaičius).

2.2.1.1 A-kodo konstravimas iš klaidas taisančio kodo

Šiame skyrelyje pateiktas aprašymas, kaip gali būti sukonstruotas sistematinis autentifikavimo kodas (S, E, Z) iš klaidas taisančio kodo (n, M, d).

Priminkime, kad klaidas taisančio kodo parametrai (n, M, d) reiškia:

n – kodo ilgis (kodo žodžio ilgis),

M – kodo dydis,

d – kodo minimalus Hemingo atstumas.

Paimkime bet kurį (n, M, d) kodą C virš kūno GF(q), tenkinantį sąlygą

$$c \in C \Rightarrow \forall \lambda \in GF(q) \text{ teisinga } [c + \lambda * \underline{1}] \in C,$$

kur $\underline{1}$ – vektorius (žodis) (1,1,.....,1).

Apibrėžkime ekvivalentumo sąryšį, kurį žymėsime "~":

$c \sim c'$ tada ir tik tada, kai $c - c' = \lambda \underline{1}$, $\lambda \in GF(q)$, $c,c' \in C$.

Pagal ~ sąryšį ir λ reikšmes visus kodo C vektorius galima suskirstyti į klases po q elementų kiekvienoje klasėje:

į vieną klasę patenka vektoriai c, c' tokie, kad $c - c' = \lambda_i \underline{1}$, $\lambda_i \in GF(q)$, i = 1..q.

A-kodo aibė S yra klaidas taisančio kodo aibės M poaibis. Į aibę S įeina M/q vektorių – po vieną atstovą iš kiekvienos klasės. Taigi, aibė S – tai mūsų konstruojamo A-kodo pradinių būsenų aibė, kurios kardinalumas lygus M/q.

Pažymėkime GF(q) elementus α_1, α_2,..., α_q. Apibrėžkime q-narių vektorių aibę V, kurios ilgis yra nq:

$$V = \{v^{(s)} = (s + \alpha_1 * \underline{1}, s + \alpha_2 * \underline{1}, \ldots, s + \alpha_q * \underline{1}) \mid s \in S\}.$$

Vektorių $v^{(s)}$ galima pavaizduoti ir taip:

$$v^{(s)} = (e_1(s), e_2(s), \ldots, e_{nq}(s)).$$

A-kodo kodavimo taisyklų aibė E bus aibės V vektorių koordinačių aibė. Taigi, |K| = nq. Pranešimo s autentikatorius z skaičiuojamas paprastai - imama e-oji vektoriaus $v^{(s)}$ koordinatė, kur e – naudojamas raktas, $e \in E$:

$$z = f(s, e) = (v^{(s)})_e.$$

Pabandykime įvertinti P_I ir P_S.

Oponentas, norėdamas apsimesti siuntėju ir nusiųsti pranešimą s gavėjui žino, kad aibėje $(s + \alpha_1 * \underline{1}, s + \alpha_2 * \underline{1}, \ldots, s + \alpha_q * \underline{1})$ kiekvienas simbolis pasitaiko su vienodą tikimybe. Todėl tikimybė, kad oponentas pasirinks teisingą autentikatorių lygi 1/q.

Jei oponentas norės perimti pranešimą (s, z), kur $z = (v^{(s)})_e$, ir pakeisti jį kitu pranešimu (s', z'), jis turi rasti pranešimą $s' \neq s$ tokį, kad vektoriaus $v^{(s')}$ koordinatėse parinktas autentikatorius z' užimtų kuo daugiau pozicijų. Kadangi žodžio ilgis yra n, o Hemingo atstumas yra d, tai z' galės užimti n-d pozicijų, nes s ir s' yra skirtingi kodo žodžiai. Todėl sėkmingo turinio pakeitimo tikimybė yra (n-d)/n = 1 – d/n.

Taigi, sukonstruoto autentifikavimo kodo parametrai yra tokie:

pradinių pranešimų skaičius |S| = M/q,

rakto ilgis |E| = nq,

apsimetimo atakos sėkmės tikimybė $P_I = 1/q$,

turinio pakeitimo atakos sėkmės tikimybė $P_S = 1 - d/n$.

2.3 Autentifikavimas su arbitravimu

Aukščiau nagrinėti autentifikavimo kodai apsaugo siuntėją ir gavėją nuo galimų oponento neteisėtų veiksmų – apsimetimo ir turinio pakeitimo atakų. Reikia pabrėžti, kad šiame modelyje daroma prielaida, kad siuntėjas ir gavėjas pasitiki vienas kitu, t.y. jie nesukčiauja, o sukčiauti gali tik oponentas. Tačiau tai ne visai atitinka realybę, nes ne visada dvi komunikuojančios šalys pasitiki viena kita. Pavyzdžiui, siuntėjas siunčia pranešimą gavėjui, bet vėliau neigia, kad buvo išsiuntęs pranešimą. Gavėjas savo ruožtu irgi gali teigti , kad gavo pranešimą iš siuntėjo, bet iš tikrųjų siuntėjas nieko nesiuntė. Šiai problemai spręsti yra pasiūlytas *autentifikavimo modelis su arbitru* (pagal [Jon98]), kuriame laikoma, kad ir siuntėjas, ir gavėjas, ir oponentas gali sukčiauti. Šis modelis papildytas dar vienu veikėju – arbitru, kuris žino privačius raktus ir nesukčiauja. Autentifikavimo kodai, kuriose numatytas arbitravimas vadinami *autentifikavimo kodais su arbitravimu*, arba A^2- *kodais*. Panagrinėkime juos plačiau. Pradžiai pateikiamas detalesnis autentifikavimo modelio su arbitravimu aprašymas (žr. 2.3.1 skyrelį), galimos atakos(žr. 2.3.1.1 skyrelį), pateikiamas A^2-kodo apibrėžimas(žr. 2.3.1.2 skyrelį).

2.3.1 Autentifikavimo modelis su arbitravimu.

Autentifikavimo modelyje su arbitravimu dalyvauja keturi dalyviai: siuntėjas, gavėjas, oponentas ir arbitras. Siuntėjas nori nusiųsti gavėjui pradinį pranešimą, kuris dar vadinamas pradine būsena. Gavėjas, gavęs pranešimą, nori būti užtikrintas, kad tas pranešimas nebuvo sufalsifikuotas ir buvo siųstas siuntėju, o ne kuo nors kitu. Tai pasiekiama tokiu būdu: pradinė būsena s iš aibės visų galimų pradinių būsenų S užkoduojama pranešimu m iš aibės visų galimų pranešimų M. Pranešimas m siunčiamas kanalu. Slapta kodavimo taisyklė e_t, pagal kurią iš

pradinės būsenos s gaunamas pranešimas m, parenkama iš aibės visų galimų kodavimo taisyklių E_T. Taigi, siuntėjas naudoja funkciją f tokią, kad:

$$f: S^* E_T \rightarrow M.$$

Atkreipkime dėmesį, kad turi būti tenkinama tokia sąlyga: gavėjas turi vienareikšmiškai nustatyti pradinę būseną iš gauto pranešimo. Ši sąlyga užrašoma taip:

$$f(s, e_t) = f(s', e_t) \Rightarrow s = s'.$$

Gavėjas, gavęs pranešimą, turi patikrinti ar šis pranešimas nebuvo suklastotas. Šiam tikslui gavėjas naudoja savo slaptą kodavimo taisyklę e_r iš aibės galimų kodavimo taisyklių E_R ir funkciją g, kuri leidžia nustatyti, ar pranešimas $m \in M$ autentiškas, t.y. nesufalsifikuotas ir buvo siųstas siuntėju.

$$g: M^* E_R \rightarrow S \cup \{\text{SUKČIAVIMAS}\},$$

$f(s, e_t) = m \Rightarrow g(m, e_r) = s$ – ši savybė turi būti teisinga visoms galimoms poroms (E_T, E_R).

Arbitras – prižiūrintis dalyvis, kuris žino visą informaciją, visus raktus, įskaitant E_T ir E_R. Tačiau jis nedalyvauja komunikavimo procese, kadangi jo pareiga spręsti ginčus tarp siuntėjo ir gavėjo, jei ginčai iškyla. Kaip jau buvo minėta, arbitras nesukčiauja[1].

Kodavimo taisyklės E_T ir E_R gali būti paskirstytos keliais būdais. Vienas būdas būtų toks: gavėjas parenka savo kodavimo taisyklę E_R, slaptai persiunčia ją arbitrui, o arbitras jau išskaičiuoja kodavimo taisyklę E_T ir persiunčia ją siuntėjui. Galima daryti paprasčiau: arbitras parenka abi kodavimo taisykles E_T, E_R ir siunčia jas atitinkamai siuntėjui ir gavėjui.

Šiame modelyje galimos penkios skirtingos atakos, kurios aprašytos 2.3.1.1 skyrelyje.

2.3.1.1 Autentifikavimo modelio su arbitravimu atakos

a) Oponento apsimetimo ataka

   Oponentas siunčia pranešimą gavėjui. Ataka pavyksta, jei gavėjas priima oponento siųstą pranešimą kaip autentišką.

b) Oponento turinio pakeitimo ataka

[1] Bendresniame autentifikavimo modelyje ši prielaida nėra daroma ir arbitras irgi gali sukčiauti

Oponentas perima originalų pranešimą ir padaro jame pakeitimus, modifikuotą pranešimą persiunčia toliau gavėjui. Ataka pavyksta, jei gavėjas priima oponento siųstą pranešimą kaip autentišką.

c) Siuntėjo apsimetimo ataka

Siuntėjas siunčia pranešimą gavėjui. Tačiau tas pranešimas nėra iš tų, kuriuos gali sugeneruoti siuntėjas, naudodamas savo kodavimo taisyklę. Ataka pavyksta, jei gavėjas priima siuntėjo siųstą pranešimą kaip autentišką.

d) Gavėjo apsimetimo ataka

Gavėjas tvirtina, kad gavo pranešimą iš siuntėjo. Ataka pavyksta, jei nustatoma, kad pranešimas galėtų būti sugeneruotas siuntėju, naudojant jo kodavimo taisyklę.

e) Gavėjo turinio pakeitimo ataka

Gavėjas gauna pranešimą iš siuntėjo, bet tvirtina, kad gavo kitokį pranešimą. Ataka pavyksta, jei nustatoma, kad tas kitas pranešimas galėtų būti sugeneruotas siuntėju, naudojant jo kodavimo taisyklę.

Visose aprašytose atakose manoma, jog apgavikas naudoja optimalią strategiją, parinkdamas pranešimą, kitaip tariant, apgavikas parenka tokį pranešimą, kuris maksimizuoja atakos sėkmės tikimybę.

Įveskime tokius žymėjimus:

P_I - oponento apsimetimo atakos sėkmės tikimybė,

P_S - oponento turinio pakeitimo atakos sėkmės tikimybė,

P_T - siuntėjo apsimetimo atakos sėkmės tikimybė,

P_{RI} - gavėjo apsimetimo atakos sėkmės tikimybė,

P_{RS} - gavėjo turinio pakeitimo atakos sėkmės tikimybė.

Tada sėkmingos atakos tikimybė P_D gali būti išreikšta taip:

$$P_D = \max(P_I, P_S, P_T, P_{RI}, P_{RS}).$$

Dabar galime pateikti formalų A^2-kodo apibrėžimą (žr. 2.3.1.2 skyrelį).

2.3.1.2 Autentifikavimo kodo su arbitravimu apibrėžimas

Kaip jau buvo minėta aukščiau, autentifikavimo kodas (A-kodas) gali būti apibrėžiamas ketvertu A(S, E, M, f), kur f : S*E → M. Panašiai apibrėžiamas ir A^2 -kodas.

A^2 -kodas apibrėžiamas šešetu A^2 (S, M, E_T, E_R, f, g), kur

S – pradinių būsenų arba pradinių pranešimų aibė,

M – aibė galimų užkoduotų pranešimų,

E_T - aibė galimų siuntėjo kodavimo taisyklių,

E_R - aibė galimų gavėjo kodavimo taisyklių,

f: S* E_T → M – siuntėjo funkcija,

g: M* E_R → S ∪ {SUKČIAVIMAS} – gavėjo funkcija.

A^2 -kodas vadinamas lygiateisiu, jei $P_I = P_S = P_T = P_{RI} = P_{RS}$.

Buvo įrodyta, kad A^2 -kodo P_D = 1/q, vadinasi kodavimo taisyklių aibių kardinalumai turi tenkinti šias nelygybes:

$$|E_R| \geq q^3 \text{ ir } |E_T| \geq q^4.$$

Jei $P_D = 1/q$ ir $|E_R| = q^3$, $|E_T| = q^4$, tai A^2 -kodas vadinamas absoliučiai lygiateisiu.

Panagrinėkime gavėjo apsimetimo ir turinio pakeitimo atakų atvejį, kai gavėjas tvirtina, jog gavo pranešimą iš siuntėjo, kurio iš tikrųjų siuntėjas nesiuntė. Šiai problemai išspręsti siuntėjas galėtų pasirašyti pradinį pranešimą s∈ S, kurį jis nori nusiųsti gavėjui. Dabar, jei gavėjas norės apsimesti, jog gavo pranešimą iš siuntėjo, jis turės mokėti pasirašyti, kaip siuntėjas, t.y. sukurti siuntėjo parašą. Parašo įvedimas gali būti tradicinių autentifikavimo kodų panaudojimu:

S*E→ M = (S, α).

Pranešimas m turės pavidalą (s, α), kur α = α(s, e) – parašas. Taigi, siuntėjas pradinį pranešimą s∈S paverčia kitu "pradiniu pranešimu" z∈Z, tokiu pat kaip s, tik su įtrauktu parašu. Šis naujas pradinis pranešimas z = (s, α) gali būti siunčiamas gavėjui naudojant dar vieną autentifikavimo kodą, kad apsisaugoti nuo oponento apsimetimo ir oponento turinio pakeitimo atakų. Taigi, siuntėjo sugeneruotas pranešimas turės tokį pavidalą:

m =(s, α(s, e_t), β(s, e_t)) = (s, α, β).

Gavėjas, gavęs pranešimą m ir norėdamas patikrinti autentiškumą, tikrina tik ar teisingas β. Tačiau, jei gavėjas tvirtina, kad gavo pranešimą iš siuntėjo, kurio siuntėjas nesiuntė, tai gavėjas turi mokėti sugeneruoti parašą α.

Toks dvejų A-kodų sujungimas leidžia apsisaugoti nuo oponento apsimetimo, oponento turinio pakeitimo, gavėjo apsimetimo ir gavėjo turinio pakeitimo atakų. Tačiau nėra jokios siuntėjo apsimetimo atakos prevencijos. Tam, kad išspręsti ir šią problemą, reikia pamodifikuoti antrą aprašytos schemos dalį taip, kad siunčiamas pranešimas būtų tokio pavidalo:

$$m =(s, \alpha, \gamma) = (s, \alpha, \gamma(s, \alpha, e_r)).$$

Tai reiškia, kad gavėjas, tikrindamas pranešimo autentiškumą, tikrina dar ir ar $\gamma = \gamma(s, \alpha, e_r)$.

Jei pranešimas teisingai sugeneruotas ir α - tikras siuntėjo parašas, tai tenkinama tokia lygybė:

$$\forall s \in S \quad \beta(s, e_t) = \gamma(s, \alpha, e_r).$$

Reikia pabrėžti, kad $e_t \in E_T$ ir $e_r \in E_R$ turi būti parinkti taip, kad tenkintų šią lygybę (apie $e_t \in E_T$ ir $e_r \in E_R$ parinkimą buvo šnekėta aukščiau, žr. 4.1 skyrelį). Dabar jei siuntėjas norės sukčiauti, tai ne tik turės pakeisti savo parašą, bet ir įvesti pakeitimus $\gamma(s, \alpha, e_r)$, o tai jau bus sunku.

Pavyzdys:

Laikykime, kad $S = s$, $E_T = (e_1, e_2, e_3, e_4)$ ir $E_R = (f_1, f_2, f_3)$, kur $e_i, f_i, s \in F_2$.

Tegul siuntėjo parašo funkcija bus tokia: $\alpha(S, E_T) = e_1 + s*e_2$, o $\beta(S, E_T) = e_3 + s* e_4$.

Taigi, siuntėjas generuoja tokį pranešimą:

$$m = (s, e_1 + s*e_2, e_3 + s* e_4).$$

Tegul gavėjo funkcija γ bus tokia: $\gamma(S, \alpha, E_R) = f_1 + \alpha*f_2 + s* f_3$. Taigi, gavėjas priima pranešimus, kurių pavidalas yra toks:

$$m = (s, \alpha, f_1 + \alpha*f_2 + s* f_3).$$

Kodavimo taisyklės turi būti parinktos taip, kad būtų tenkinama tokia lygybė:

$$\beta(S, E_T) = \gamma(S, \alpha, E_R),$$

arba $e_3 + s* e_4 = f_1 + \alpha*f_2 + s* f_3$

$$e_3 + s* e_4 = f_1 + (e_1 + s*e_2)*f_2 + s* f_3$$

arba $e_3 = f_1 + e_1* f_2,$

$$e_4 = f_3 + e_2*f_2.$$

3. Autentifikavimo kodų ir klaidas taisančių kodų pagrindinės sąvokos

Trumpai priminkime kas yra autentifikavimo kodas, klaidas taisantis kodas ir kokią prasmę turi jų parametrai.

Autentifikavimo kodai (arba A-kodai) buvo sukurti apsimetimo ir turinio pakeitimo atakų prevencijai. Priminkime, kad apsimetimo ataka – tai oponento bandymas apsimesti siuntėju ir nusiųsti gavėjui apgaulingą pranešimą, o turinio pakeitimo ataka – tai oponento bandymas perimti ir padaryti pakeitimus originaliame pranešime, o modifikuotą pranešimą persiųsti toliau gavėjui.

Atkreipkime dėmesį į tai, kad tie kodai, kuriuos toliau nagrinėsime, neužtikrina informacijos slaptumo, t.y. jie apsaugo nuo apsimetimo ir turinio pakeitimo atakų, bet oponentas žino kokia informacija siunčiama kanalu.

Priimta žymėti:

P_I - didžiausia apsimetimo atakos sėkmės tikimybė,

P_S - didžiausia turinio pakeitimo atakos sėkmės tikimybė,

P_D - didžiausia pavykusio sukčiavimo tikimybė.

P_I ir P_S išraiškos tokios:

$$P_I = \max_{m \in M} P(\text{m – priimtas}),$$

$$P_S = \max_{\substack{m,m' \in M \\ m \neq m'}} P(\text{m' – priimtas vietoj m}),$$

sąlyga $m \neq m'$ reiškia, kad m' bus dekoduota kitaip, nei m (t.y. dekodavę m' mes gausime kitą pradinį pranešimą, kuris skirsis nuo to, kuris buvo užkoduotas).

Apgavystės tikimybė P_D apibrėžiama taip:

$P_D = \max(P_I, P_S)$.

Bendru **A-kodu** laikomas ketvertas (S, E, M, g), kur

S – baigtinė pradinių pranešimų aibė,

E – baigtinė kodavimo taisyklių aibė (čia kodavimo taisyklės prasmė analogiška slaptam raktui kriptografijoje),

M – baigtinė pranešimų aibė,

g: $S*E \rightarrow M$ - kiekvienam elementui e iš aibės E paverčia aibę S į galios $|S|$ aibę M. Kitaip tariant, kiekviena taisyklė $e \in E$ apibrėžia atvaizdį S į M.

Mes naudosime sistematinius (arba Dekarto) autentifikavimo kodus, kurie pasižymi tuo, kad tokio kodo pranešime $m \in M$ figūruoja pradinis pranešimas s, kur M – baigtinė pranešimų aibė.

Taigi, sistematiniu A-kodu laikomas trejetas (S, E, Z), kur

S – baigtinė pradinių pranešimų arba pradinių būsenų aibė,

E – baigtinė kodavimo taisyklių aibė,

Z – autentifikatorių aibė.

Funkcija $f: S*E \rightarrow Z$ - kiekvienam $e \in E$ ir $s \in S$ turime pranešimą $m = (s,z) \in S*Z$,
kur $z = f(s,e)$. Kitaip tariant, e nusako atvaizdį $S \rightarrow Z$.
Elementas z vadinamas autentifikatoriumi.

Visų pranešimų aibe M galima apibrėžti tokiu būdu:

$$M = \{ m = (s, f(s,e)): s \in S, e \in E\}.$$

Kaip matome, sistematinio kodo pranešimo sudedamoji dalis – neužkoduotas pradinis pranešimas s.

Kodavimą galime aprašyti taip:

$$Z=e(S),$$

t.y. $\forall$ e nusako atvaizdi $S \rightarrow Z$.
Taigi, $\forall (s,z) \in (S,Z)$, $E(s,z) = \{ e \in E : e(S)=z \}$.

Kodavimo unikalumą galima apibrėžti taip:

$E(s,z) \cap E(s,z') = \varnothing$, $z \neq z$

$\forall \; s \in S$, $E = \bigcup_{z} E(s,z)$.

Taigi, siuntėjas siunčia kanalu pranešimą iš aibės $M = \{ m = (s, f(s,e)): s \in S, e \in E\}$.

Gavėjas gali patikrinti gauto pranešimo (s,z) autentiškumą tokiu būdu:
Reikia patikrinti, ar $z = f(s,e)$ (arba $e(s) = z$), kur e – (slapta) kodavimo taisykle, žinoma tik siuntėjui ir gavėjui.

Klaidas taisantys kodai naudojami galimiems informacijos iškraipymams ištaisyti, tam, kad būtų galima atstatyti pirminę informaciją, kuri buvo siunčiama triukšmingu kanalu. Prieš siunčiant informaciją, ji yra užkoduojama, t.y. pridedama papildoma informacija, kuri dekodavimo metu leidžia ištaisyti klaidas, jei jos įvyko. Kodas vadinamas t klaidas taisančiu kodu, jei naudojant minimalaus atstumo taisyklę jis teisingai dekoduoja žodį, kuriame siuntimo metu įvyko ne daugiau kaip t iškraipymų.

Priminkime, kad klaidas taisantis kodas žymimas (n, M, d), kur:

n – kodo ilgis (kodo žodžio ilgis),

M – kodo dydis,

d – kodo minimalus Hemingo atstumas.

Naudojant minimalaus atstumo dekodavimo taisyklę (n, M, d) kodas gali ištaisyti visas klaidas, jeigu jų įvyko ne daugiau kaip $t = \left\lfloor \frac{d-1}{2} \right\rfloor$.

Kodas C vadinamas tiesiniu, jei C yra tiesinės erdvės F_p^n poerdvis, kur

p – pirminis skaičius,

n – natūralusis skaičius, $n \geq 1$,

F_p - kūnas,

F_p^n - visų ilgio n vektorių, kurių koordinatės priklauso kūnui F_p aibė.

Tiesinis kodas žymimas [n, k], kur

n – kodo ilgis,

k – kodo dimensija (bazės vektorių skaičius).

4. Autentifikavimo kodo konstravimas iš klaidas taisančio kodo

Šiame skyrelyje aprašytos taisykles, pagal kurias bus konstruojamas autentifikavimo kodas iš klaidas taisančio kodo (remiantis [Til98]).

Tarkime, turime klaidas taisantį kodą (n, M, d) ir iš jo konstruosime sistematinį autentifikavimo kodą (S, E, Z).

Paimkime bet kurį (n, M, d) kodą C virš kūno GF(q), tenkinantį sąlygą

$$c \in C \Rightarrow \forall \lambda \in GF(q) \text{ teisinga } [c + \lambda * \underline{1}] \in C,$$

kur $\underline{1}$ – vektorius (žodis) (1,1,.....,1).

Apibrėžkime ekvivalentumo sąryšį, kurį žymėsime "~":

$c \sim c'$ tada ir tik tada, kai $c - c' = \lambda \underline{1}$, $\lambda \in GF(q)$, $c,c' \in C$.

Pagal ~ sąryšį ir λ reikšmes visus kodo C vektorius galima suskirstyti į klases, kur į kiekvieną klasę įeis po q elementų.

Apibrėžkime aibę S, į kurią įeina po vieną atstovą iš kiekvienos klasės. Aibė S bus kodo C poaibis. Į aibę S įeina M/q vektorių – po vieną atstovą iš kiekvienos klasės. Taigi, aibė S – tai mūsų konstruojamo A-kodo pranešimų aibė, kurios kardinalumas lygus M/q.

Pažymėkime GF(q) elementus α_1, α_2,..., α_q. Apibrėžkime q-narių vektorių aibę V, kurios vektorių ilgis yra nq:

$$V = \{v^{(s)} = (s + \alpha_1 * \underline{1}, s + \alpha_2 * \underline{1}, \ldots, s + \alpha_q * \underline{1}) \mid s \in S\}.$$

Vektorių $v^{(s)}$ galima pavaizduoti ir taip:

$$v^{(s)} = (e_1(s), e_2(s), \ldots, e_{nq}(s)).$$

A-kodo kodavimo taisyklų aibė E bus aibės V vektorių koordinačių aibė. Taigi, |E| = nq. Pranešimo s autentikatorius z skaičiuojamas paprastai - imama e-oji vektoriaus $v^{(s)}$ koordinatė, kur e – naudojamas raktas, $e \in E$:

$$z = f(s, e) = (v^{(s)})_e.$$

5. Efektyvaus autentifikavimo kodo konstravimas

Mūsų tikslas yra ištirti su kokiais klaidas taisančiais kodais gaunami geriausi autentifikavimo kodai naudojant aukščiau aprašytas konstravimo taisykles (žr. 4 skyrelį). Bet prieš tai reikia suformuluoti kriterijus, pagal kuriuos bus lyginami autentifikavimo kodai.

Vertinant autentifikavimo kodą, atsižvelgiama į apsimetimo atakos sėkmės tikimybę P_I ir turinio pakeitimo atakos sėkmės tikimybę P_S. Mes norime turėti tokį autentifikavimo kodą, kurio P_I ir P_S reikšmės yra kuo mažesnės.

Kitame skyrelyje aprašyta kaip gali būti apskaičiuojamos P_I ir P_S reikšmės bei jų sąryšis su klaidas taisančio kodo parametrais.

5.1 Klaidas taisančio kodo ir autentifikavimo kodo parametrų sąryšis

Įvertinant P_I ir P_S reikšmes pasinaudosime tokia teorema ([JKS94]):

Teorema 1. Tarkime, turime kodą C su parametrais (n, M, d) tenkinantį sąlygą $c \in C \Rightarrow \forall \lambda \in$ GF(q) teisinga $[c + \lambda * \underline{1}] \in C$, $\lambda \in$ GF(q). Tada egzistuoja atitinkamas autentifikavimo kodas su tokiais parametrais:

$|S| = Mq^{-1}$, $|E| = nq$, $P_I = 1/q$, $P_S = 1- d/n$.

Toks autentifikavimo kodas gaunamas naudojant 4 skyrelyje aprašytą konstravimo būdą.

Čia galima paminėti, kad egzistuoja dar vienas būdas sukonstruoti autentifikavimo kodą iš klaidas taisančio kodo, naudojant gimiringas transformacijas. Šis būdas aprašytas [XT99]. Tačiau mes jo nenagrinėsime, kadangi šiuo būdu sukonstruoti a-kodai turi prastesnes P_I ir P_S reikšmes, t.y. apsimetimo atakos sėkmės tikimybė apskaičiuojama taip pat: $P_I = 1/q$, o turinio pakeitimo atakos sėkmės tikimybė yra didesnė: $P_S = 1 - (q-1)d/qn$. Todėl naudosime konstravimo būdą, aprašytą 4 skyrelyje.

Taigi, remiantis Teorema 1, mes galime iš anksto apskaičiuoti ir įvertinti apsimetimo ir turinio pakeitimo atakų sėkmės tikimybes. Mes turime, kad

$$P_I = 1/q, \qquad P_S = 1- d/n.$$

Kaip matome, P_I ir P_S reikšmės priklauso nuo klaidas taisančio kodo parametrų n, d ir q reikšmių.

Remiantis [JKS96], kiekvienam autentifikavimo kodui teisinga tokia lema:

Lema 1. Kiekvienam A-kodui teisinga

$$P_I \geq \frac{|S|}{|B|} \quad \text{ir} \quad P_S \geq \frac{|S|-1}{|B|-1},$$

kur S – autentifikavimo kodo pradinių pranešimų aibė,

B – autentifikavimo kodo pranešimų aibė. Tai yra tie pranešimai kurie siunčiami kanalu, kiekvienas toks pranešimas turi informacinę dalį ir autentikatorių.

Taigi, autentifikavimo kodo P_I ir P_S reikšmės priklauso nuo jo parametrų S ir B. S ir B reikšmės savo ruožtu priklauso nuo klaidas taisančio kodo, kurį naudojant konstruojamas autentifikavimo kodas, parametrų. Pavyzdžiui, iš aprašyto konstravimo būdo (žr. 4 skyrelį) mes matome, kad konstruojamo A-kodo pranešimų aibės B kardinalumas lygus M/q, kur M – klaidas taisančio kodo dydis.

Taigi, žinodami sąryšį tarp klaidas taisančio kodo parametrų ir konstruojamo autentifikavimo kodo P_I ir P_S reikšmių, mes galime iškelti reikalavimus klaidas taisančiam kodui, kuriuos jis turi tenkinti, kad iš jo sukonstruotas autentifikavimo kodas būtų efektyvus. Šie reikalavimai aprašyti toliau.

5.2 Reikalavimai klaidas taisančiam kodui

Kaip jau minėjome, mes norime turėti tokį autentifikavimo kodą, kurio P_I ir P_S reikšmės yra kuo mažesnės. Kadangi P_I = 1/q ir P_S= 1- d/n (žr. 5.1 skyrelį), tai klaidas taisančio kodo parametrai d, n ir q turi tenkinti šiuos reikalavimus:

q turi būti kuo didesnis, nes jo reikšmė yra atvirkščiai proporcinga P_I reikšmei,

d/n santykis turi būti kuo didesnis, tada P_S reikšmė bus mažesnė.

Sukonkretinkime P_I ir P_S reikšmes, t.y. nustatykime mums priimtinas P_I ir P_S. Laikykime, kad mus tenkins autentifikavimo kodai, kurių $P_I \leq 0.1$ ir $P_S \leq 0.1$. Čia galima paminėti, kad literatūroje daugiausia nagrinėjami atvejai, kai P_I arba P_S yra lygūs $\frac{1}{2}$, tačiau praktiškai mus tokie kodai netenkintų, todėl mes stengsimės rasti autentifikavimo kodą su geresniais P_I ir P_S.

5.3 Tinkamų klaidas taisančių kodų parinkimas

Kaip buvo minėta praeitame skyrelyje, mus domina tokie kodai, kurių d/n santykis ir q yra kuo didesni, nes nuo šių parametrų priklauso apsimetimo atakos sėkmės tikimybę P_I ir turinio pakeitimo atakos sėkmės tikimybę P_S. Tokiu būdu, pavyzdžiui kodai virš kūno GF(2) arba GF(3) netinka, nes tada apsimetimo atakos sėkmės tikimybę P_I bus per didelė: $\frac{1}{2}$ ir $\frac{1}{3}$ atitinkamai.

Panagrinėkime kai kurias kodų šeimas ir nustatykime, ar tų šeimų kodų parametrai tenkina nustatytus reikalavimus ir ar tinka autentifikavimo kodų konstravimui. Išrinkime iš jų labiausiai tinkančias kodų šeimas. Nagrinėsime Hammingo kodus, Reed-Muller kodus, BCH kodus ir Reed-Solomon kodus.

5.3.1 Hemingo kodai

Hemingo kodas $Ham_r(q)$ virš baigtinio kūno GF(q), $r \geq 2$, yra tiesinis ilgio $n = \frac{q^r - 1}{q - 1}$ kodas, kurio kontrolinės matricos stulpeliai yra visi galimi ilgio r skirtingi nenuliniai vektoriai. Kontrolinės matricos dydis yra $r \times (q^r - 1)/(q-1)$. Kodas turi tokius parametrus:

$$[\frac{q^r-1}{q-1}, \frac{q^r-1}{q-1} - r, 3\ ].$$

Kaip matome, Hemingo kodo minimalus atstumas d = 3. Įvertinant mus dominantį d/n santykį, gauname d/n = 3/n. Kadangi mes norime, kad d/n būtų kuo didesnis, o d/n = 3/n, tai reiškia, kad n reikšmė turi būti kuo arčiau 3, t.y n turi būti lygus 4. Tačiau tokia n reikšmė mus netenkina: visų pirma n = 4 per mažas kodo ilgis, be to P_S = 1- d/n = 1 – ¾ = 0.25 – per didelė turinio pakeitimo atakos sėkmės tikimybė.

Taigi, Hemingo kodų šeima netinka efektyvių autentifikavimo kodų konstravimui.

5.3.2 Reed-Muller kodai

Reed-Muller kodai – tai dvejetainiai kodai, kurie yra žymimi R(r,m) ir turi tokius parametrus:

ilgis $n = 2^m$,

minimalus atstumas $d = 2^{m-r}$,

ištaiso $2^{m-r-1} - 1$ klaidų.

Reed-Muller kodu, kurio ilgis yra n ir laipsnis r (= 0, 1, …, m) yra laikomas dvejetainis kodas R(r, m), turintis visus dvejetainius ilgio n žodžius, kurių kaip Boolean polinomų didžiausias laipsnis yra r.

Kaip jau buvo minėta, dvejetainiai kodai mums netinka, nes iš tokių kodų sukonstruotų autentifikavimo kodų apsimetimo atakos sėkmės tikimybę $P_I = \frac{1}{2}$ yra per didelė. Todėl panagrinėkime apibendrintus Reed-Muller kodus (Generalized Reed-Muller codes) virš $GF(q^m)$, kurie leidžia pasirinkti didesnį q ir tuo pačiu sumažinti P_I.

Apibendrintas q-narinis laipsnio r ir ilgio n Reed-Muller kodas virš $GF(q^m)$ žymimas RM_{F_q} (r, m), kur $q = p^k$, p –pirminis, $0 \le r \le m(q-1)$. Remiantis [PW04] kodas RM_{F_q} (r, m) turi tokius parametrus:

Kodo ilgis $n = q^m$, minimalus atstumas $d = (q - r)q^{m-1}$, kai r < q.

Įvertinkime santykį $\frac{d}{n}$:

$$\frac{d}{n} = \frac{(q - r)q^{m-1}}{q^m} = \frac{(q-r)q^m}{q^m q} = \frac{q-r}{q}.$$

Mes turime, kad $0 \le r \le m(q-1)$. Į gautą išraišką įstatykime didžiausią r reikšmę:

$$\frac{q-r}{q} = \frac{q-m(q-1)}{q} = \frac{q-qm+m}{q} = \frac{q}{q} - \frac{qm}{q} + \frac{m}{q} = 1 - m + \frac{m}{q}.$$

Įvertinkime P_S:

$$P_S = 1 - d/n = 1 - (1 - m + \frac{m}{q}) = 1 - 1 + m - \frac{m}{q} = m - \frac{m}{q}.$$

Kad P_S reikšmė būtų mažesnė už 1, reikia imti m = 1, didesnės m reikšmės netinka. Todėl, mus tenkintų apibendrintas $RM_{F_q}(r, 1)$ kodas. Remiantis [PW04] $RM_{F_q}(r, m)$ kodas, kai m = 1 yra Reed-Solomon RS(r) kodas, t.y. $RS(r) = RM_{F_q}(r, 1)$.

Taigi, apibendrinti Reed-Muller kodai mums tinka, kai turime $RM_{F_q}(r, 1)$, o tai yra Reed-Solomon kodų atvejis. Reed-Solomon kodai bus aprašyti vėliau.

5.3.3 BCH kodai

Ilgio n BCH kodu su "konstrukciniu" atstumu δ (virš kūno GF(q))vadinamas ciklinis kodas, kurio generuojantis polinomas yra:

$g(x) = MBD(P^k(x), P^{k+1}(x), \ldots, P^{k+\delta-2}(x))$,

kur k – sveikasis skaičius,

$P^i(x)$ – polinomas, kurio šaknys yra kūno $GF(q^m)$ primityvūs elementai α (α^k, α^{k+1},…, $\alpha^{k+\delta-2}$) .

Kodas turi tokius parametrus:

$n = q^m - 1$,

$d \ge \delta$,

$k = n - \deg g(x)$.

BCH kodai leidžia mums pasirinkti δ ir sukonstruoti atitinkamą BCH kodą. Todėl mes turime galimybę konstruoti BCH kodą su mus tenkinančiais n, k, d parametrais. Vadinasi, BCH kodai gali būti naudojami efektyvių autentifikavimo kodų konstravimui.

5.3.4 Reed-Solomon kodai

Reed-Solomon kodai yra BCH kodų atskiras atvejis.

Reed-Solomon kodas virš GF(q) – tai ilgio n = q - 1 BCH kodas (q niekada nelygus 2). Kodo generuojantis polinomas yra $g(x) = (\alpha^{m+1} + x)(\alpha^{m+2} + x)\ldots(\alpha^{m+\delta-1} + x)$, kur m – sveikasis skaičius (dažniausiai imama m = 0), α – kūno GF(q) primityvus elementas. Čia δ - taip vadinamas "konstrukcinis" kodo atstumas (designed distance).

Reed-Solomon kodo dimensija $k = n - \deg g(x) = n - \delta + 1$,
minimalus atstumas $d = n - k + 1$. Įstatę į minimalaus atstumo lygtį k išraišką, gauname:
$d = n - (n - \delta + 1) = \delta$. Vadinasi, nuo parinkto "konstrukcinio" kodo atstumo priklauso kodo minimalus atstumas, o tai reiškia, kad mes galime parinkti tokius Reed-Solomon kodo parametrus, kad iš jo būtų galima sukonstruoti efektyvų autentifikavimo kodą.

Taip pat čia svarbu paminėti, kad literatūroje patariama naudoti Reed-Solomon kodus arba išplėstinius Reed-Solomon kodus efektyvių autentifikavimo kodų konstravimui.

6. Tinkamo Reed-Solomon kodo paieška

Kadangi, kaip jau buvo minėta praeitame skyrelyje, literatūroje patariama naudoti Reed-Solomon kodus autentifikavimo kodų konstravimui, parinksime Reed-Solomon kodo parametrus taip, kad toks kodas tiktų sukonstruoti efektyvų autentifikavimo kodą.

Paimkime kūną GF(q), kur $q = 2^r$. Reed-Solomon q-arinį kodą virš $GF(2^r)$ galima lengvai paversti dvinariu ([Ada91]), o to mums gali prireikti, norint išbandyti sukonstruotą autentifikavimo kodą praktiškai (programuojant ir pan.). Mums reikia parinkti parametrus r, δ, k taip, kad d/n būtų kuo didesnis, o kodo ilgis nebūtų per didelis. Taip pat mes turime, kad n priklauso nuo r, t.y. $n = 2^r - 1$, o d priklauso nuo δ, t.y $d = \delta$.

Tegul r = 4, δ = 14, m = 0. Paimkime kodą RS(16, 14) su generuojančiu polinomu
$g(x) = (\alpha + x)(\alpha^2 + x)(\alpha^3 + x)\ldots(\alpha^{13} + x)$ virš kūno GF(16), kur GF(16) sudarytas naudojant polinomą $x^4 + x + 1$.

Kodo RS(16, 14) parametrai yra tokie:

$n = 2^4 - 1 = 15$,

$k = 2^4 - 14 = 2$,

d = 14,

$|C| = (2^4)^2 = 256$.

Taigi, kodas turi 256 žodžius. Konstruojant autentifikavimo kodą, turėsime suskirstyti visus žodžius į ekvivalentumo klases pagal λ reikšmes, $\lambda \in GF(16)$ (žr. 4 skyrelį). Konstruojamas autentifikavimo kodas turės tokias P_I ir P_S reikšmes:

$P_I = 1/16 = 0.0625$,

$P_S = 1 - 14/15 = 1/15 \approx 0.0667$.

Kaip matome, P_I ir P_S reikšmės mus tenkina, todėl siūloma naudoti šį kodą autentifikavimo kodo konstravimui. Kitame skyrelyje pateiktas aprašymas autentifikavimo kodo konstravimo iš pasiūlyto RS(16, 14) kodo procesas.

6.1 Autentifikavimo kodo konstravimas iš RS(16, 14) kodo

Šiame skyrelyje pateiksime aprašymą, kaip praktiškai konstruojamas autentifikavimo kodas iš klaidas taisančio kodo (šiuo atveju mes turime RS(16, 14) kodą). Aprašomo proceso rezultate mes gausime konkretų autentifikavimo kodą su konkrečiais (S, E, Z) parametrais.

Tam, kad sukonstruoti autentifikavimo kodą, mums reikia žinoti visus kodo RS(16, 14) žodžius. Kaip jau buvo minėta, kodas RS(16, 14) turi 256 žodžius. Raskime juos. Prieš tai mums reikia žinoti visus kūno GF(16), kuris sudarytas naudojant polinomą $x^4 + x + 1$, vektorius bei rasti kodo generuojančią matricą. Turime generuojanti polinomą $g(x) = (\alpha + x)(\alpha^2 + x)(\alpha^3 + x)\ldots(\alpha^{13} + x)$, kur α - primityvus elementas. Atlikus tam tikrus skaičiavimus, kurie pateikti priede a_kodas_is_RS.mws, mes gauname generuojančią matricą :

$$G := \begin{bmatrix} \alpha, \alpha+\alpha^2, \alpha^3+\alpha+\alpha^2, \alpha^2+\alpha^3+1, \alpha^3+1+\alpha, \alpha^2+1+\alpha, \alpha^2+\alpha^3, \alpha^3+1, 1+\alpha, \alpha^2, \alpha+\alpha^3, 1+\alpha^2, \alpha^3, 1, 0 \\ 0, \alpha, \alpha+\alpha^2, \alpha^3+\alpha+\alpha^2, \alpha^2+\alpha^3+1, \alpha^3+1+\alpha, \alpha^2+1+\alpha, \alpha^2+\alpha^3, \alpha^3+1, 1+\alpha, \alpha^2, \alpha+\alpha^3, 1+\alpha^2, \alpha^3, 1 \end{bmatrix}$$

Padauginus visus kūno GF(16) elementus iš generuojančios matricos, gauname kodo žodžių aibę ,kuri pateikta *Priede 1*.

Gautas kodas tenkina sąlygą $c \in C \Rightarrow \forall \lambda \in GF(q)$ teisinga $[c + \lambda * \underline{1}] \in C$, taigi galime pradėti konstruoti autentifikavimo kodą.

Suskirstykime kodo C žodžius į klases pagal ekvivalentumo sąryšį “~”:

$c \sim c'$ tada ir tik tada, kai $c - c' = \lambda \underline{1}$, $\lambda \in GF(q)$, $c,c' \in C$ (žr. 4 skyrelį). Mūsų atveju λ turi 16 reikšmių:

$0, 1, \alpha, \alpha^2, \alpha^3, 1+\alpha, \alpha+\alpha^2, \alpha^2+\alpha^3, \alpha^3+1+\alpha, 1+\alpha^2, \alpha+\alpha^3, \alpha^2+1+\alpha, \alpha^3+\alpha+\alpha^2,$
$1+\alpha+\alpha^2+\alpha^3, \alpha^2+\alpha^3+1, \alpha^3+1$

Kaip jau buvo minėta, pagal $\sim$ sąryšį ir λ reikšmes visus kodo C vektorius galima suskirstyti į klases, kur į kiekvieną klasę įeis po q elementų. Taigi, turėsime $256/q = 256/16 = 16$ klasių, kurios pateiktos *Priede 1*.

Sukonstruokime aibę S, į kurią įeina po vieną atstovą iš kiekvienos klasės. Tai bus mūsų konstruojamo A-kodo pradinių pranešimų aibė.

S =

$[[\alpha^3+1+\alpha, 1+\alpha^2, \alpha+\alpha^3, \alpha^2+1+\alpha, \alpha^3+\alpha+\alpha^2, 1+\alpha+\alpha^2+\alpha^3, \alpha^2+\alpha^3+1,$
$\alpha^3+1, 1, \alpha, \alpha^2, \alpha^3, 1+\alpha, \alpha+\alpha^2, \alpha^2+\alpha^3], [1, 1+\alpha, \alpha^2+1+\alpha, 1+\alpha+\alpha^2+\alpha^3,$
$\alpha^2+\alpha^3, \alpha+\alpha^3, \alpha+\alpha^2, \alpha^2+\alpha^3+1, \alpha^3, \alpha, 1+\alpha^2, \alpha^3+1+\alpha, \alpha^2, \alpha^3+1, 0], [$
$\alpha^3+1+\alpha, \alpha^3+1+\alpha, \alpha^3+1+\alpha, \alpha^3+1+\alpha, \alpha^3+1+\alpha, \alpha^3+1+\alpha, \alpha^3+1+\alpha,$
$\alpha^3+1+\alpha, \alpha^3+1+\alpha, \alpha^3+1+\alpha, \alpha^3+1+\alpha, \alpha^3+1+\alpha, \alpha^3+1+\alpha, \alpha^3+1+\alpha,$
$\alpha^3+1+\alpha], [\alpha^3+1+\alpha, \alpha^2, \alpha^3+1, 0, 1, 1+\alpha, \alpha^2+1+\alpha, 1+\alpha+\alpha^2+\alpha^3, \alpha^2+\alpha^3,$
$\alpha+\alpha^3, \alpha+\alpha^2, \alpha^2+\alpha^3+1, \alpha^3, \alpha, 1+\alpha^2], [1, \alpha^2+1+\alpha, \alpha^3+1+\alpha, 0, 1+\alpha^2,$
$1+\alpha+\alpha^2+\alpha^3, \alpha^3, \alpha+\alpha^2, \alpha^3+1, \alpha^2, \alpha^2+\alpha^3+1, \alpha^2+\alpha^3, \alpha^3+\alpha+\alpha^2, \alpha+\alpha^3, \alpha]$
$, [0, \alpha^2, \alpha^2+\alpha^3, 1+\alpha+\alpha^2+\alpha^3, \alpha^3+1, 1+\alpha^2, \alpha^3+\alpha+\alpha^2, \alpha^3+1+\alpha, 1, \alpha+\alpha^2,$
$\alpha^3, \alpha^2+1+\alpha, \alpha+\alpha^3, 1+\alpha, \alpha], [1, \alpha^3+1+\alpha, \alpha^2+\alpha^3, \alpha, \alpha^2+\alpha^3+1, 0, \alpha^3+1,$
$\alpha^3, \alpha+\alpha^3, \alpha^3+\alpha+\alpha^2, \alpha+\alpha^2, 1+\alpha^2, 1+\alpha, 1+\alpha+\alpha^2+\alpha^3, \alpha^2], [\alpha^3+1+\alpha, 0,$
$1+\alpha^2, 1+\alpha+\alpha^2+\alpha^3, \alpha^3, \alpha+\alpha^2, \alpha^3+1, \alpha^2, \alpha^2+\alpha^3+1, \alpha^2+\alpha^3, \alpha^3+\alpha+\alpha^2,$
$\alpha+\alpha^3, \alpha, 1, \alpha^2+1+\alpha], [0, \alpha^3, \alpha^3+1+\alpha, \alpha^2+\alpha^3+1, 1, \alpha+\alpha^3, 1+\alpha+\alpha^2+\alpha^3,$
$1+\alpha^2, \alpha, \alpha^2+\alpha^3, 1+\alpha, \alpha^3+\alpha+\alpha^2, \alpha^2+1+\alpha, \alpha+\alpha^2, \alpha^2], [1, 0, \alpha, \alpha+\alpha^2,$
$\alpha^3+\alpha+\alpha^2, \alpha^2+\alpha^3+1, \alpha^3+1+\alpha, \alpha^2+1+\alpha, \alpha^2+\alpha^3, \alpha^3+1, 1+\alpha, \alpha^2, \alpha+\alpha^3,$
$1+\alpha^2, \alpha^3], [0, 1+\alpha, 1+\alpha^2, \alpha^3+1, \alpha, \alpha^2+1+\alpha, \alpha^2+\alpha^3+1, \alpha+\alpha^3, \alpha^2,$
$\alpha^3+1+\alpha, \alpha+\alpha^2, 1+\alpha+\alpha^2+\alpha^3, \alpha^3+\alpha+\alpha^2, \alpha^2+\alpha^3, \alpha^3], [\alpha^3+1+\alpha,$
$\alpha^2+1+\alpha, \alpha^2+\alpha^3, \alpha^3+1, 1+\alpha, \alpha^2, \alpha+\alpha^3, 1+\alpha^2, \alpha^3, 1, 0, \alpha, \alpha+\alpha^2,$
$\alpha^3+\alpha+\alpha^2, \alpha^2+\alpha^3+1], [1, \alpha^3, \alpha^3+1, \alpha^3+1+\alpha, 1+\alpha+\alpha^2+\alpha^3, \alpha^2+1+\alpha, \alpha^2,$
$\alpha, \alpha^3+\alpha+\alpha^2, 1+\alpha^2, 0, \alpha+\alpha^3, \alpha^2+\alpha^3+1, 1+\alpha, \alpha^2+\alpha^3], [1, \alpha+\alpha^2, \alpha^3,$
$\alpha^2+1+\alpha, \alpha+\alpha^3, 1+\alpha, \alpha, 0, \alpha^2, \alpha^2+\alpha^3, 1+\alpha+\alpha^2+\alpha^3, \alpha^3+1, 1+\alpha^2,$
$\alpha^3+\alpha+\alpha^2, \alpha^3+1+\alpha], [0, 1+\alpha^2, 1+\alpha+\alpha^2+\alpha^3, \alpha^3, \alpha+\alpha^2, \alpha^3+1, \alpha^2,$
$\alpha^2+\alpha^3+1, \alpha^2+\alpha^3, \alpha^3+\alpha+\alpha^2, \alpha+\alpha^3, \alpha, 1, \alpha^2+1+\alpha, \alpha^3+1+\alpha], [1, \alpha^2+\alpha^3,$
$1+\alpha^2, \alpha^2, \alpha+\alpha^2, \alpha, \alpha+\alpha^3, \alpha^3+1, 1+\alpha+\alpha^2+\alpha^3, 1+\alpha, \alpha^3, \alpha^2+\alpha^3+1,$
$\alpha^2+1+\alpha, 0, \alpha^3+\alpha+\alpha^2]]$

Taigi, |S| = 16.

Dabar mums reikia apibrėžti q-narių vektorių aibę V, kurios vektorių ilgis yra nq:

$$V = \{v^{(s)} = (s + \alpha_1 * \underline{1}, s + \alpha_2 * \underline{1}, \ldots, s + \alpha_q * \underline{1}) \mid s \in S\},$$

$\alpha_1, \alpha_2, \ldots, \alpha_q$ yra kūno GF(q) elementai.

Mūsų atveju q = 16, todėl $V = \{v^{(s)} = (s + \alpha_1 * \underline{1}, s + \alpha_2 * \underline{1}, s + \alpha_3 * \underline{1}, \ldots, s + \alpha_{16} * \underline{1} \mid s \in S\}$, kur $\alpha_1 = 0, \alpha_2 = 1, \alpha_3 = \alpha, \alpha_4 = \alpha^2, \ldots, \alpha_{16} = \alpha^3 + 1$.

Gauta V aibė pateikta *Priede 1*. Čia galima paminėti, kad aibė V turi šešiolika ilgio 15 vektorių.

Kaip jau buvo minėta, kiekvieną vektorių $v^{(s)}$ galima pavaizduoti ir taip:

$$v^{(s)} = (e_1(s), e_2(s), \ldots, e_{nq}(s)).$$

A-kodo kodavimo taisyklų aibė E bus aibės V vektorių koordinačių aibė. Taigi, |E| = nq.

E = {1, 2, 3, …, 240}

Pranešimo s autentikatorius z skaičiuojamas paprastai - imama e-oji vektoriaus $v^{(s)}$ koordinatė, kur e – naudojamas raktas, $e \in E$:

$$z = f(s, e) = (v^{(s)})_e.$$

Pavyzdžiui, paimkime pranešimą s =

$[\alpha^3 + 1 + \alpha, \alpha^2, \alpha^3 + 1, 0, 1, 1 + \alpha, \alpha^2 + 1 + \alpha, 1 + \alpha + \alpha^2 + \alpha^3, \alpha^2 + \alpha^3, \alpha + \alpha^3, \alpha + \alpha^2, \alpha^2 + \alpha^3 + 1, \alpha^3, \alpha, 1 + \alpha^2]$

Tada $v^{(s)}$ =

table([0 = [1, 0, α, α + α², α³ + α + α², α² + α³ + 1, α³ + 1 + α, α² + 1 + α, α² + α³, α³ + 1, 1 + α, α², α + α³, 1 + α², α³], 1 = [0, 1, 1 + α, α² + 1 + α, 1 + α + α² + α³, α² + α³, α + α³, α + α², α² + α³ + 1, α³, α, 1 + α², α³ + 1 + α, α², α³ + 1], 2 = [1 + α, α, 0, α², α² + α³, 1 + α + α² + α³, α³ + 1, 1 + α², α³ + α + α², α³ + 1 + α, 1, α + α², α³, α² + 1 + α, α + α³], 3 = [1 + α², α², α + α², α, α + α³, α³ + 1, 1 + α + α² + α³, 1 + α, α³, α² + α³ + 1, α² + 1 + α, 0, α³ + α + α², 1, α² + α³], 4 = [α³ + 1, α³, α + α³, α³ + α + α², α + α², 1 + α², 1 + α, 1 + α + α² + α³, α², 1, α³ + 1 + α, α² + α³, α, α² + α³ + 1, 0], 5 = [α, 1 + α, 1, 1 + α², α² + α³ + 1,

$\alpha^3 + \alpha + \alpha^2, \alpha^3, \alpha^2, 1 + \alpha + \alpha^2 + \alpha^3, \alpha + \alpha^3, 0, \alpha^2 + 1 + \alpha, \alpha^3 + 1, \alpha + \alpha^2,$
$\alpha^3 + 1 + \alpha], 6 = [\alpha^2 + 1 + \alpha, \alpha + \alpha^2, \alpha^2, 0, \alpha^3, \alpha^3 + 1 + \alpha, \alpha^2 + \alpha^3 + 1, 1, \alpha + \alpha^3,$
$1 + \alpha + \alpha^2 + \alpha^3, 1 + \alpha^2, \alpha, \alpha^2 + \alpha^3, 1 + \alpha, \alpha^3 + \alpha + \alpha^2], 7 = [\alpha^2 + \alpha^3 + 1, \alpha^2 + \alpha^3,$
$\alpha^3 + \alpha + \alpha^2, \alpha + \alpha^3, \alpha, 1, \alpha^2 + 1 + \alpha, \alpha^3 + 1 + \alpha, 0, 1 + \alpha^2, 1 + \alpha + \alpha^2 + \alpha^3, \alpha^3,$
$\alpha + \alpha^2, \alpha^3 + 1, \alpha^2], 8 = [\alpha + \alpha^3, \alpha^3 + 1 + \alpha, \alpha^3 + 1, \alpha^2 + \alpha^3 + 1, 1 + \alpha^2, \alpha + \alpha^2, 0,$
$\alpha^2 + \alpha^3, \alpha^2 + 1 + \alpha, \alpha, \alpha^3, 1 + \alpha + \alpha^2 + \alpha^3, 1, \alpha^3 + \alpha + \alpha^2, 1 + \alpha], 9 = [\alpha^2, 1 + \alpha^2,$
$\alpha^2 + 1 + \alpha, 1 + \alpha, \alpha^3 + 1 + \alpha, \alpha^3, \alpha^3 + \alpha + \alpha^2, \alpha, \alpha^3 + 1, \alpha^2 + \alpha^3, \alpha + \alpha^2, 1,$
$1 + \alpha + \alpha^2 + \alpha^3, 0, \alpha^2 + \alpha^3 + 1], 10 = [\alpha^3 + 1 + \alpha, \alpha + \alpha^3, \alpha^3, \alpha^2 + \alpha^3, \alpha^2,$
$\alpha^2 + 1 + \alpha, 1, \alpha^2 + \alpha^3 + 1, \alpha + \alpha^2, 1 + \alpha, \alpha^3 + 1, \alpha^3 + \alpha + \alpha^2, 0, 1 + \alpha + \alpha^2 + \alpha^3, \alpha],$
$11 = [\alpha + \alpha^2, \alpha^2 + 1 + \alpha, 1 + \alpha^2, 1, \alpha^3 + 1, \alpha + \alpha^3, \alpha^2 + \alpha^3, 0, \alpha^3 + 1 + \alpha,$
$\alpha^3 + \alpha + \alpha^2, \alpha^2, 1 + \alpha, \alpha^2 + \alpha^3 + 1, \alpha, 1 + \alpha + \alpha^2 + \alpha^3], 12 = [1 + \alpha + \alpha^2 + \alpha^3,$
$\alpha^3 + \alpha + \alpha^2, \alpha^2 + \alpha^3, \alpha^3, 0, 1 + \alpha, 1 + \alpha^2, \alpha^3 + 1, \alpha, \alpha^2 + 1 + \alpha, \alpha^2 + \alpha^3 + 1, \alpha + \alpha^3,$
$\alpha^2, \alpha^3 + 1 + \alpha, \alpha + \alpha^2], 13 = [\alpha^3 + \alpha + \alpha^2, 1 + \alpha + \alpha^2 + \alpha^3, \alpha^2 + \alpha^3 + 1, \alpha^3 + 1, 1, \alpha,$
$\alpha^2, \alpha^3, 1 + \alpha, \alpha + \alpha^2, \alpha^2 + \alpha^3, \alpha^3 + 1 + \alpha, 1 + \alpha^2, \alpha + \alpha^3, \alpha^2 + 1 + \alpha], 14 = [\alpha^2 + \alpha^3,$
$\alpha^2 + \alpha^3 + 1, 1 + \alpha + \alpha^2 + \alpha^3, \alpha^3 + 1 + \alpha, 1 + \alpha, 0, \alpha + \alpha^2, \alpha + \alpha^3, 1, \alpha^2, \alpha^3 + \alpha + \alpha^2,$
$\alpha^3 + 1, \alpha^2 + 1 + \alpha, \alpha^3, 1 + \alpha^2],$
$15 = [\alpha^3, \alpha^3 + 1, \alpha^3 + 1 + \alpha, 1 + \alpha + \alpha^2 + \alpha^3, \alpha^2 + 1 + \alpha, \alpha^2, \alpha, \alpha^3 + \alpha + \alpha^2, 1 + \alpha^2,$
$0, \alpha + \alpha^3, \alpha^2 + \alpha^3 + 1, 1 + \alpha, \alpha^2 + \alpha^3, 1]$
$])$

Tarkime slapta kodavimo taisyklė e = 119, tada autentikatorius $z = (v^{(s)})_e = \alpha^3 + \alpha + \alpha^2$

Taigi, kanalu siunčiamas pradinis pranešimas ir autentikatorius:

$message := [\alpha^3 + 1 + \alpha, \alpha^2, \alpha^3 + 1, 0, 1, 1 + \alpha, \alpha^2 + 1 + \alpha, 1 + \alpha + \alpha^2 + \alpha^3, \alpha^2 + \alpha^3,$
$\alpha + \alpha^3, \alpha + \alpha^2, \alpha^2 + \alpha^3 + 1, \alpha^3, \alpha, 1 + \alpha^2], \alpha^3 + \alpha + \alpha^2$

Sukonstruoto autentifikavimo kodo apsimetimo atakos sėkmės tikimybė yra

$$P_I = 1/q = 1/16 = 0{,}0625,$$

turinio pakeitimo atakos sėkmės tikimybė yra

$$P_S = 1 - d/n = 1 - 14/15 = 1/15 \approx 0.0667.$$

Taigi, P_I ir P_S rodikliai yra geri. Tačiau čia iškyla viena problema. Kaip matome, kodas RS(16, 14) turi 256 žodžius. Realizuojant autentifikavimo schemą, pradinį pranešimą paversime į dvejetainį formatą, suskaidysime į vektorius ir kiekvieną vektorių užkoduosime, naudojant kodą RS(16, 14). Toliau, į kiekvieną užkoduotą žodį mes galime žiūrėti kaip į autentifikavimo kodo pradinį pranešimą, kuriam reikia paskaičiuoti autentikatorių. Tačiau sukonstruotas autentifikavimo kodas turi tik 16 žodžių pradinių pranešimų aibėje, o kodas RS(16, 14) turi 256

žodžius. Taigi, ne visi RS(16, 14) kodo žodžiai gali būti autentifikavimo kodo pradiniais pranešimais. Siūloma pasinaudoti vienu iš dviejų pateiktų sprendimų:

1) Remiantis autentifikavimo kodo konstravimo iš klaidas taisančio kodo schemą (žr. 4 skyrelį), mes buvom suskirstę vektorius į klases pagal ekvivalentumo sąryšį c ~ c' tada ir tik tada, kai c - c' = $\lambda\underline{1}$, $\lambda \in$ GF(q), c,c' $\in$ C. Taigi, jei pridėsime $\lambda\underline{1}$ vektorių prie pradinio pranešimo, tai gausime vektorių iš tos pačios ekvivalentumo klasės. Mes žinome, kad autentifikavimo kodo pradinių pranešimų aibė susideda iš vektorių – atstovų nuo kiekvienos ekvivalentumo klasės. Todėl mums svarbu gauti tą atstovą, pridedant įvairias λ reikšmes prie pradinio pranešimo. Kodo RS(16, 14) žodžiai buvo suskirstyti į 16 klasių, po 16 vektorių kiekvienoje klasėje. Taigi, pridėjus $\lambda\underline{1}$ vektorių prie pradinio pranešimo, gausime vektorių, įeinanti į autentifikavimo kodo pradinių pranešimų aibę, kuriems mes galime paskaičiuoti autentikatorių. Šiuo atveju siuntėjas siųs gavėjui porą (s, z), kur s – pradinis pranešimas, o z – modifikuoto pranešimo autentikatorius. Gavėjas, žinodamas šią techniką, galės patikrinti autentiškumą, nes vis tiek turės nemodifikuotą pradinį pranešimą.

2) Šis būdas nenumato kodavimo su klaidas taisančiu kodu. Pranešimą, kurį norime išsiųsti, reikia suskaidyti į tam tikro ilgio vektorius (ilgis parenkamas toks, kad galėtumėm gauti lygiai tiek skirtingų bitų kombinacijų, kiek autentifikavimo kodas turi elementų pradinių pranešimų aibėje S. Tokiu būdu mes kiekvienai kombinacijai iš anksto galime vienareikšmiškai priskirti atitinkamą autentifikavimo kodo pradinį pranešimą). Mūsų atveju pranešimą reikėtų skaidyti į 4 bitų vektorius, turėsime 16 įvairių vektorių (bitų kombinacijų) ir kiekvienam vektoriui galima priskirti atitinkamą autentifikavimo kodo pradinį pranešimą, kuriam savo ruožtu galima paskaičiuoti autentikatorių.

7. Pastabos ir rekomendacijos

Panagrinėjus plačiau gauto autentifikavimo kodo savybes bei autentifikavimo kodo konstravimo procesą, galima suformuluoti tam tikras pastabas ir rekomendacijas.

Visų pirma yra svarbu, kad klaidas taisantis kodas, kuriuo naudojantis planuojama sukonstruoti a-kodą, tenkintų Teoremos 1 (žr. skyrelį 5.1) sąlygą: $c \in C \Rightarrow \forall \lambda \in GF(q)$ teisinga $[c + \lambda * \underline{1}] \in C$, $\lambda \in GF(q)$. Tik tada galima suskirstyti kodo žodžius į ekvivalentumo klases. Pastebėta, kad jeigu klaidas taisančiam kodui priklauso vienetinis vektorius ir vektoriai pavidalo $\lambda * \underline{1}$, tai toks kodas tenkins Teoremos 1 minėtą sąlygą.

Skyrelyje 6.1 buvo pasiūlyti sprendimo būdai išspręsti problemai, kai pranešimas, kuriam norima užtikrinti autentiškumą, neįeina į naudojamo autentifikavimo kodo pradinių pranešimų aibę. Siūloma pridedant įvairias λ (tiksliau $\lambda * \underline{1}$) reikšmes prie pradinio pranešimo rasti ekvivalentų pradiniam pranešimui žodį, t.y. iš tos pačios ekvivalentumo klasės. Numatoma, kad šis paieškos procesas turėtų vykti realiame laike, t.y. kiekvieną kartą bus ieškomas ekvivalentus žodis, jei to reikia. Todėl, *didindami q (norėdami turėti mažesnę apsimetimo atakos sėkmės tikimybę $P_I = 1/q$), mes mažiname autentifikavimo proceso greitį, kadangi proporcingai q reikšmei didiname paieškos laiką kiekvienoje ekvivalentumo klasėje.* Antras būdas nenumato tokio paieškos proceso, kadangi pranešimas skaidomas į tam tikro ilgio vektorius, o kiekvienam vektoriui iš anksto priskirtas atitinkamas autentifikavimo kodo pradinis pranešimas. Tačiau naudojant antrą būdą, padidėja autentikatorių skaičius, nes sumažėja autentikuojamo pranešimo ilgis (kadangi savo pranešimą mes skaidome į dalis ir kiekvienai daliai ieškome autentikatoriaus).

Pagal aprašytą konstravimo būdą, į kiekvieną ekvivalentumo klasę įeina q elementų, o tokių klasių bus $\frac{|C|}{q}$. Taigi, kuo daugiau turime kodo žodžių, tuo daugiau turėsime ekvivalentumo klasių. Turėdami fiksuotą q reikšmę, mes galime imti didesnį arba mažesnį klaidas taisantį kodą. Tačiau *sukonstruoto a-kodo apsimetimo atakos sėkmės tikimybė nuo ekvivalentumo klasių skaičiaus nepriklauso, kadangi $P_I = 1/q$, todėl turėti didesnį a-kodą nėra saugiau.* Tačiau *turėti didesnį a-kodą yra efektyviau greičio atžvilgiu.* Kuo daugiau ekvivalentumo klasių turėsime, tuo didesnė tikimybė, kad mūsų pranešimas pateks į a-kodo pradinių pranešimų aibę ir nereikės gaišti laiko ieškant ekvivalentaus žodžio, kuris priklauso a-kodo pradinių pranešimų aibei.

A-kodo saugumui įtakos turi ir klaidas taisančio kodo žodžių ilgis. Remiantis naudojamu konstravimo procesu, praplečiant a-kodo pradinių pranešimų aibę S iki aibės V, kodo žodžių ilgis n turi įtakos aibės V žodžių ilgiui (tiesiškai proporcingas). Kadangi A-kodo kodavimo taisyklių

aibė E yra aibės V vektorių koordinačių aibė, tai kuo ilgesnis aibės V vektorius, arba *kuo ilgesnis klaidas taisančio kodo žodis, tuo daugiau kodavimo taisyklių (raktų) mes turėsime*. Tą patį galima pasakyti ir apie parametrą q, nes aibės V vektoriaus ilgis yra lygus nq.

Kaip matome, didėjant q didėja saugumas (raktų skaičius didėja, P_I mažėja), tačiau sulėtėja pats autentifikavimo kodo naudojimo procesas.

8. Išvados

Darbe buvo atliekamas tyrimas ir siekiama nustatyti su kokiais klaidas taisančiais kodais gaunami efektyvūs autentifikavimo kodai. Iš pradžių buvo suformuluoti kriterijai, pagal kuriuos buvo vertinami ir lyginami autentifikavimo kodai. Taip pat buvo nustatyti bendri reikalavimai, kuriuos turi tenkinti klaidas taisantys kodai, kad iš jų sukonstruoti autentifikavimo kodai būtų efektyvūs. Atsižvelgiant į nustatytus reikalavimus, buvo nagrinėjamos kelios kodų šeimos (Hemingo, BCH, Reed-Muller, Reed-Solomon) ir ieškomos labiausiai tinkančios. Iš jų buvo atrinktas ir pasiūlytas konkretus klaidas taisantis kodas, iš kurio gali būti sukonstruotas efektyvus autentifikavimo kodas. Autentifikavimo kodo konstravimas buvo išbandytas praktiškai ir aprašytas pažingsniui, to rezultate buvo gautas konkretus a-kodas, tenkinantis iškeltus reikalavimus apsimetimo atakos sėkmės tikimybei ir turinio pakeitimo atakos sėkmės tikimybei. Remiantis gautais teoriniais ir praktiniais rezultatais, bei išnagrinėjus autentifikavimo kodo konstravimo procesą, buvo suformuluotos pastabos ir rekomendacijos, kurios galėtų praversti renkantis kodą, norint sukonstruoti autentifikavimo kodą kai svarbios tam tikros konstruojamojo autentifikavimo kodo savybės (pvz., vienu atveju gali būti svarbus greitis, kitu atveju – saugumas).

Summary

To begin with, there is a relationship between error-correcting codes and authentication codes. The thing is, that authentication codes can be constructed using error-correcting codes and visa versa.

The objective of the work was to determine what error-correcting codes are the most suitable for constructing effective authentification codes. Error-correcting codes and authentication codes were overviewed and construction method was described. Then some criteria for evaluating authentication codes were proposed in order to be able to compare authentication codes and choose the most suitable ones. Also some basic requirements for error-correcting codes were established. Error-correcting codes must satisfy above-mentioned requirement in order to be able to produce effective authentication code. According to that, some families of error-correcting codes were studied (such as Hamming codes, BCH codes, Reed-Muller codes, Reed-Solomon codes) and as the result the most suitable family was chosen and a concrete code was offered.

A construction of authentication code using error-correcting code was tried practically as well. According to acquired practical and theoretical information some comments and recommendations were proposed, which could help when choosing an error-correcting code in order to construct an authentication code if some features of authentication code are important (speed, security, etc.)

Priedas 1. *Autentifikavimo kodo konstravimas iš RS(16, 14) kodo*

RS(16, 14) kodo žodžiai (patekta tik dalis, pilna aibė pateikta kompaktiniame diske)

$$
\begin{aligned}
&[\,[\,\alpha^3+1+\alpha,\ 1+\alpha^2,\ \alpha+\alpha^3,\ \alpha^2+1+\alpha,\ \alpha^3+\alpha+\alpha^2,\ 1+\alpha+\alpha^2+\alpha^3,\ \alpha^2+\alpha^3+1,\\
&\quad \alpha^3+1,\ 1,\ \alpha,\ \alpha^2,\ \alpha^3,\ 1+\alpha,\ \alpha+\alpha^2,\ \alpha^2+\alpha^3\,],\ [\,1,\ 1+\alpha,\ \alpha^2+1+\alpha,\ 1+\alpha+\alpha^2+\alpha^3,\\
&\quad \alpha^2+\alpha^3,\ \alpha+\alpha^3,\ \alpha+\alpha^2,\ \alpha^2+\alpha^3+1,\ \alpha^3,\ \alpha,\ 1+\alpha^2,\ \alpha^3+1+\alpha,\ \alpha^2,\ \alpha^3+1,\ 0\,],\ [\\
&\quad \alpha^3+1+\alpha,\ \alpha^3+1+\alpha,\ \alpha^3+1+\alpha,\ \alpha^3+1+\alpha,\ \alpha^3+1+\alpha,\ \alpha^3+1+\alpha,\ \alpha^3+1+\alpha,\\
&\quad \alpha^3+1+\alpha,\ \alpha^3+1+\alpha,\ \alpha^3+1+\alpha,\ \alpha^3+1+\alpha,\ \alpha^3+1+\alpha,\ \alpha^3+1+\alpha,\ \alpha^3+1+\alpha\,],\\
&\quad [\,0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0\,],\\
&\quad [\,1,\ 1,\ 1,\ 1,\ 1,\ 1,\ 1,\ 1,\ 1,\ 1,\ 1,\ 1,\ 1,\ 1,\ 1,\ 1\,],\ [\,\alpha^3+1+\alpha,\ \alpha^2,\ \alpha^3+1,\ 0,\ 1,\ 1+\alpha,\\
&\quad \alpha^2+1+\alpha,\ 1+\alpha+\alpha^2+\alpha^3,\ \alpha^2+\alpha^3,\ \alpha+\alpha^3,\ \alpha+\alpha^2,\ \alpha^2+\alpha^3+1,\ \alpha^3,\ \alpha,\ 1+\alpha^2\,],\ [\,0,\\
&\quad \alpha,\ \alpha+\alpha^2,\ \alpha^3+\alpha+\alpha^2,\ \alpha^2+\alpha^3+1,\ \alpha^3+1+\alpha,\ \alpha^2+1+\alpha,\ \alpha^2+\alpha^3,\ \alpha^3+1,\ 1+\alpha,\\
&\quad \alpha^2,\ \alpha+\alpha^3,\ 1+\alpha^2,\ \alpha^3,\ 1\,],\ [\,1,\ \alpha^2+1+\alpha,\ \alpha^3+1+\alpha,\ 0,\ 1+\alpha^2,\ 1+\alpha+\alpha^2+\alpha^3,\ \alpha^3,\\
&\quad \alpha+\alpha^2,\ \alpha^3+1,\ \alpha^2,\ \alpha^2+\alpha^3+1,\ \alpha^2+\alpha^3,\ \alpha^3+\alpha+\alpha^2,\ \alpha+\alpha^3,\ \alpha\,],\ [\,\alpha^3+1+\alpha,\ \alpha^3+1,\\
&\quad \alpha^2+\alpha^3+1,\ 1+\alpha^2,\ \alpha+\alpha^2,\ 0,\ \alpha^2+\alpha^3,\ \alpha^2+1+\alpha,\ \alpha,\ \alpha^3,\ 1+\alpha+\alpha^2+\alpha^3,\ 1,\\
&\quad \alpha^3+\alpha+\alpha^2,\ 1+\alpha,\ \alpha+\alpha^3\,],\ [\,0,\ \alpha^2,\ \alpha^2+\alpha^3,\ 1+\alpha+\alpha^2+\alpha^3,\ \alpha^3+1,\ 1+\alpha^2,\\
&\quad \alpha^3+\alpha+\alpha^2,\ \alpha^3+1+\alpha,\ 1,\ \alpha+\alpha^2,\ \alpha^3,\ \alpha^2+1+\alpha,\ \alpha+\alpha^3,\ 1+\alpha,\ \alpha\,],\ [\,1,\ \alpha^3+1+\alpha,\\
&\quad \alpha^2+\alpha^3,\ \alpha,\ \alpha^2+\alpha^3+1,\ 0,\ \alpha^3+1,\ \alpha^3,\ \alpha+\alpha^3,\ \alpha^3+\alpha+\alpha^2,\ \alpha+\alpha^2,\ 1+\alpha^2,\ 1+\alpha,\\
&\quad 1+\alpha+\alpha^2+\alpha^3,\ \alpha^2\,],\ [\,\alpha^3+1+\alpha,\ 0,\ 1+\alpha^2,\ 1+\alpha+\alpha^2+\alpha^3,\ \alpha^3,\ \alpha+\alpha^2,\ \alpha^3+1,\ \alpha^2,\\
&\quad \alpha^2+\alpha^3+1,\ \alpha^2+\alpha^3,\ \alpha^3+\alpha+\alpha^2,\ \alpha+\alpha^3,\ \alpha,\ 1,\ \alpha^2+1+\alpha\,],\ [\,0,\ \alpha^3,\ \alpha^3+1+\alpha,\\
&\quad \alpha^2+\alpha^3+1,\ 1,\ \alpha+\alpha^3,\ 1+\alpha+\alpha^2+\alpha^3,\ 1+\alpha^2,\ \alpha,\ \alpha^2+\alpha^3,\ 1+\alpha,\ \alpha^3+\alpha+\alpha^2,\\
&\quad \alpha^2+1+\alpha,\ \alpha+\alpha^2,\ \alpha^2\,],\ [\,1,\ 0,\ \alpha,\ \alpha+\alpha^2,\ \alpha^3+\alpha+\alpha^2,\ \alpha^2+\alpha^3+1,\ \alpha^3+1+\alpha,\\
&\quad \alpha^2+1+\alpha,\ \alpha^2+\alpha^3,\ \alpha^3+1,\ 1+\alpha,\ \alpha^2,\ \alpha+\alpha^3,\ 1+\alpha^2,\ \alpha^3\,],\ [\,\alpha^3+1+\alpha,\ 1,\ \alpha+\alpha^2,\ \alpha^3,\\
&\quad \alpha^2+1+\alpha,\ \alpha+\alpha^3,\ 1+\alpha,\ \alpha,\ 0,\ \alpha^2,\ \alpha^2+\alpha^3,\ 1+\alpha+\alpha^2+\alpha^3,\ \alpha^3+1,\ 1+\alpha^2,\\
&\quad \alpha^3+\alpha+\alpha^2\,],\ [\,0,\ 1+\alpha,\ 1+\alpha^2,\ \alpha^3+1,\ \alpha,\ \alpha^2+1+\alpha,\ \alpha^2+\alpha^3+1,\ \alpha+\alpha^3,\ \alpha^2,\\
&\quad \alpha^3+1+\alpha,\ \alpha+\alpha^2,\ 1+\alpha+\alpha^2+\alpha^3,\ \alpha^3+\alpha+\alpha^2,\ \alpha^2+\alpha^3,\ \alpha^3\,],\ [\,1,\ 1+\alpha^2,\\
&\quad \alpha^2+\alpha^3+1,\ \alpha^3+\alpha+\alpha^2,\ \alpha^3,\ \alpha^2,\ 1+\alpha+\alpha^2+\alpha^3,\ \alpha+\alpha^3,\ 0,\ \alpha^2+1+\alpha,\ \alpha^3+1,\\
&\quad \alpha+\alpha^2,\ \alpha^3+1+\alpha,\ \alpha,\ 1+\alpha\,],\ [\,\alpha^3+1+\alpha,\ 1+\alpha,\ 0,\ \alpha+\alpha^2,\ \alpha+\alpha^3,\ 1,\ \alpha^2,\\
&\quad \alpha^3+\alpha+\alpha^2,\ \alpha^3+1,\ \alpha^2+1+\alpha,\ \alpha^3,\ 1+\alpha^2,\ \alpha^2+\alpha^3,\ \alpha^2+\alpha^3+1,\ 1+\alpha+\alpha^2+\alpha^3\,],\ [\\
&\quad 0,\ \alpha+\alpha^2,\ \alpha+\alpha^3,\ 1,\ \alpha^2,\ \alpha^3+\alpha+\alpha^2,\ \alpha^3+1,\ \alpha^2+1+\alpha,\ \alpha^3,\ 1+\alpha^2,\ \alpha^2+\alpha^3,\\
&\quad \alpha^2+\alpha^3+1,\ 1+\alpha+\alpha^2+\alpha^3,\ \alpha^3+1+\alpha,\ 1+\alpha\,],\ [\,1,\ 1+\alpha+\alpha^2+\alpha^3,\ 0,\ \alpha^2+\alpha^3+1,\\
&\quad \alpha^3+\alpha+\alpha^2,\ 1+\alpha+\alpha^2+\alpha^3,\ \alpha^2+\alpha^3+1\,],\ [\,\alpha^3+1+\alpha,\ \alpha,\ 1+\alpha,\ 1,\ 1+\alpha^2,\\
&\quad \alpha^2+\alpha^3+1,\ \alpha^3+\alpha+\alpha^2,\ \alpha^3,\ \alpha^2+1+\alpha+\alpha^3,\ \alpha^3+\alpha,\ 0,\ \alpha^2+1+\alpha,\ \alpha^3+1,\\
&\quad \alpha+\alpha^2\,],\ [\,\alpha^3+1,\ \alpha^3+1,\ \alpha^3+1,\ \alpha^3+1,\ \alpha^3+1,\ \alpha^3+1,\ \alpha^3+1,\ \alpha^3+1,\ \alpha^3+1,\ \alpha^3+1,\\
&\quad \alpha^3+1,\ \alpha^3+1,\ \alpha^3+1,\ \alpha^3+1,\ \alpha^3+1\,]\,]
\end{aligned}
$$

Kodo žodžių suskirstymas į klases

Turime 16 klasių, kiekvienoje klasėje po 16 narių. Priede pateikta tik 1 klasė, visos 16 klasių pateiktos kompaktiniame diske.

Klasė 1

table([1 = [$\alpha^3 + 1 + \alpha$, $1 + \alpha^2$, $\alpha + \alpha^3$, $\alpha^2 + 1 + \alpha$, $\alpha^3 + \alpha + \alpha^2$, $1 + \alpha + \alpha^2 + \alpha^3$, $\alpha^2 + \alpha^3 + 1$, $\alpha^3 + 1$, 1, α, α^2, α^3, $1 + \alpha$, $\alpha + \alpha^2$, $\alpha^2 + \alpha^3$], 2 = [$\alpha + \alpha^3$, α^2, $\alpha^3 + 1 + \alpha$, $\alpha + \alpha^2$, $1 + \alpha + \alpha^2 + \alpha^3$, $\alpha^3 + \alpha + \alpha^2$, $\alpha^2 + \alpha^3$, α^3, 0, $1 + \alpha$, $1 + \alpha^2$, $\alpha^3 + 1$, α, $\alpha^2 + 1 + \alpha$, $\alpha^2 + \alpha^3 + 1$], 3 = [$\alpha^3 + 1$, $\alpha^2 + 1 + \alpha$, α^3, $1 + \alpha^2$, $\alpha^2 + \alpha^3$, $\alpha^2 + \alpha^3 + 1$, $1 + \alpha + \alpha^2 + \alpha^3$, $\alpha^3 + 1 + \alpha$, $1 + \alpha$, 0, $\alpha + \alpha^2$, $\alpha + \alpha^3$, 1, α^2, $\alpha^3 + \alpha + \alpha^2$], 4 = [$1 + \alpha + \alpha^2 + \alpha^3$, 1, $\alpha^3 + \alpha + \alpha^2$, $1 + \alpha$, $\alpha + \alpha^3$, $\alpha^3 + 1 + \alpha$, $\alpha^3 + 1$, $\alpha^2 + \alpha^3 + 1$, $1 + \alpha^2$, $\alpha + \alpha^2$, 0, $\alpha^2 + \alpha^3$, $\alpha^2 + 1 + \alpha$, α, α^3], 5 = [$1 + \alpha$, $\alpha^2 + \alpha^3 + 1$, α, $1 + \alpha + \alpha^2 + \alpha^3$, $\alpha + \alpha^2$, $\alpha^2 + 1 + \alpha$, $1 + \alpha^2$, 1, $\alpha^3 + 1$, $\alpha + \alpha^3$, $\alpha^2 + \alpha^3$, 0, $\alpha^3 + 1 + \alpha$, $\alpha^3 + \alpha + \alpha^2$, α^2], 6 = [α^3, $\alpha + \alpha^2$, $\alpha^3 + 1$, α^2, $\alpha^2 + \alpha^3 + 1$, $\alpha^2 + \alpha^3$, $\alpha^3 + \alpha + \alpha^2$, $\alpha + \alpha^3$, α, 1, $\alpha^2 + 1 + \alpha$, $\alpha^3 + 1 + \alpha$, 0, $1 + \alpha^2$, $1 + \alpha + \alpha^2 + \alpha^3$], 7 = [$\alpha^2 + \alpha^3 + 1$, $1 + \alpha$, $\alpha^2 + \alpha^3$, 1, α^3, $\alpha^3 + 1$, $\alpha^3 + 1 + \alpha$, $1 + \alpha + \alpha^2 + \alpha^3$, $\alpha^2 + 1 + \alpha$, α^2, α, $\alpha^3 + \alpha + \alpha^2$, $1 + \alpha^2$, 0, $\alpha + \alpha^3$], 8 = [$\alpha^2 + 1 + \alpha$, $\alpha^3 + 1$, $\alpha + \alpha^2$, $\alpha^3 + 1 + \alpha$, α, $1 + \alpha$, 1, $1 + \alpha^2$, $\alpha^2 + \alpha^3 + 1$, $\alpha^3 + \alpha + \alpha^2$, α^3, α^2, $1 + \alpha + \alpha^2 + \alpha^3$, $\alpha + \alpha^3$, 0], 9 = [0, $\alpha^3 + \alpha + \alpha^2$, 1, $\alpha^2 + \alpha^3$, $1 + \alpha^2$, α^2, $\alpha + \alpha^2$, α, $\alpha + \alpha^3$, $\alpha^3 + 1$, $1 + \alpha + \alpha^2 + \alpha^3$, $1 + \alpha$, α^3, $\alpha^2 + \alpha^3 + 1$, $\alpha^2 + 1 + \alpha$], 10 = [$\alpha^3 + \alpha + \alpha^2$, 0, $1 + \alpha + \alpha^2 + \alpha^3$, α, $\alpha^3 + 1 + \alpha$, $\alpha + \alpha^3$, α^3, $\alpha^2 + \alpha^3$, α^2, $\alpha^2 + 1 + \alpha$, 1, $\alpha^2 + \alpha^3 + 1$, $\alpha + \alpha^2$, $1 + \alpha$, $\alpha^3 + 1$], 11 = [1, $1 + \alpha + \alpha^2 + \alpha^3$, 0, $\alpha^2 + \alpha^3 + 1$, α^2, $1 + \alpha^2$, $\alpha^2 + 1 + \alpha$, $1 + \alpha$, $\alpha^3 + 1 + \alpha$, α^3, $\alpha^3 + \alpha + \alpha^2$, α, $\alpha^3 + 1$, $\alpha^2 + \alpha^3$, $\alpha + \alpha^2$], 12 = [$\alpha^2 + \alpha^3$, α, $\alpha^2 + \alpha^3 + 1$, 0, $\alpha^3 + 1$, α^3, $\alpha + \alpha^3$, $\alpha^3 + \alpha + \alpha^2$, $\alpha + \alpha^2$, $1 + \alpha^2$, $1 + \alpha$, $1 + \alpha + \alpha^2 + \alpha^3$, α^2, 1, $\alpha^3 + 1 + \alpha$], 13 = [$1 + \alpha^2$, $\alpha^3 + 1 + \alpha$, α^2, $\alpha^3 + 1$, 0, 1, $1 + \alpha$, $\alpha^2 + 1 + \alpha$, $1 + \alpha + \alpha^2 + \alpha^3$, $\alpha^2 + \alpha^3$, $\alpha + \alpha^3$, $\alpha + \alpha^2$, $\alpha^2 + \alpha^3 + 1$, α^3, α], 14 = [α^2, $\alpha + \alpha^3$, $1 + \alpha^2$, α^3, 1, 0, α, $\alpha + \alpha^2$, $\alpha^3 + \alpha + \alpha^2$, $\alpha^2 + \alpha^3 + 1$, $\alpha^3 + 1 + \alpha$, $\alpha^2 + 1 + \alpha$, $\alpha^2 + \alpha^3$, $\alpha^3 + 1$, $1 + \alpha$], 15 = [$\alpha + \alpha^2$, α^3, $\alpha^2 + 1 + \alpha$, $\alpha + \alpha^3$, $1 + \alpha$, α, 0, α^2, $\alpha^2 + \alpha^3$, $1 + \alpha + \alpha^2 + \alpha^3$, $\alpha^3 + 1$, $1 + \alpha^2$, $\alpha^3 + \alpha + \alpha^2$, $\alpha^3 + 1 + \alpha$, 1],
16 = [α, $\alpha^2 + \alpha^3$, $1 + \alpha$, $\alpha^3 + \alpha + \alpha^2$, $\alpha^2 + 1 + \alpha$, $\alpha + \alpha^2$, α^2, 0, α^3, $\alpha^3 + 1 + \alpha$, $\alpha^2 + \alpha^3 + 1$, 1, $\alpha + \alpha^3$, $1 + \alpha + \alpha^2 + \alpha^3$, $1 + \alpha^2$]
])

Aibė V(praplėsta aibė S)

(pateiktas tik vienas aibės V elementas, visa aibė pateikta kompaktiniame diske)

$$
\begin{aligned}
&([0=[\alpha^3+1+\alpha,\ 1+\alpha^2,\ \alpha+\alpha^3,\ \alpha^2+1+\alpha,\ \alpha^3+\alpha+\alpha^2,\ 1+\alpha+\alpha^2+\alpha^3,\\
&\alpha^2+\alpha^3+1,\ \alpha^3+1,\ 1,\ \alpha,\ \alpha^2,\ \alpha^3,\ 1+\alpha,\ \alpha+\alpha^2,\ \alpha^2+\alpha^3],\ 1=[\alpha+\alpha^3,\ \alpha^2,\ \alpha^3+1+\alpha,\\
&\alpha+\alpha^2,\ 1+\alpha+\alpha^2+\alpha^3,\ \alpha^3+\alpha+\alpha^2,\ \alpha^2+\alpha^3,\ \alpha^3,\ 0,\ 1+\alpha,\ 1+\alpha^2,\ \alpha^3+1,\ \alpha,\\
&\alpha^2+1+\alpha,\ \alpha^2+\alpha^3+1],\ 2=[\alpha^3+1,\ \alpha^2+1+\alpha,\ \alpha^3,\ 1+\alpha^2,\ \alpha^2+\alpha^3,\ \alpha^2+\alpha^3+1,\\
&1+\alpha+\alpha^2+\alpha^3,\ \alpha^3+1+\alpha,\ 1+\alpha,\ 0,\ \alpha+\alpha^2,\ \alpha+\alpha^3,\ 1,\ \alpha^2,\ \alpha^3+\alpha+\alpha^2],\ 3=[\\
&1+\alpha+\alpha^2+\alpha^3,\ 1,\ \alpha^3+\alpha+\alpha^2,\ 1+\alpha,\ \alpha+\alpha^3,\ \alpha^3+1+\alpha,\ \alpha^3+1,\ \alpha^2+\alpha^3+1,\\
&1+\alpha^2,\ \alpha+\alpha^2,\ 0,\ \alpha^2+\alpha^3,\ \alpha^2+1+\alpha,\ \alpha,\ \alpha^3],\ 4=[1+\alpha,\ \alpha^2+\alpha^3+1,\ \alpha,\\
&1+\alpha+\alpha^2+\alpha^3,\ \alpha+\alpha^2,\ \alpha^2+1+\alpha,\ 1+\alpha^2,\ 1,\ \alpha^3+1,\ \alpha+\alpha^3,\ \alpha^2+\alpha^3,\ 0,\ \alpha^3+1+\alpha,\\
&\alpha^3+\alpha+\alpha^2,\ \alpha^2],\ 5=[\alpha^3,\ \alpha+\alpha^2,\ \alpha^3+1,\ \alpha^2,\ \alpha^2+\alpha^3+1,\ \alpha^2+\alpha^3,\ \alpha^3+\alpha+\alpha^2,\\
&\alpha+\alpha^3,\ \alpha,\ 1,\ \alpha^2+1+\alpha,\ \alpha^3+1+\alpha,\ 0,\ 1+\alpha^2,\ 1+\alpha+\alpha^2+\alpha^3],\ 6=[\alpha^2+\alpha^3+1,\\
&1+\alpha,\ \alpha^2+\alpha^3,\ 1,\ \alpha^3,\ \alpha^3+1,\ \alpha^3+1+\alpha,\ 1+\alpha+\alpha^2+\alpha^3,\ \alpha^2+1+\alpha,\ \alpha^2,\ \alpha,\\
&\alpha^3+\alpha+\alpha^2,\ 1+\alpha^2,\ 0,\ \alpha+\alpha^3],\ 7=[\alpha^2+1+\alpha,\ \alpha^3+1,\ \alpha+\alpha^2,\ \alpha^3+1+\alpha,\ \alpha,\ 1+\alpha,\\
&1,\ 1+\alpha^2,\ \alpha^2+\alpha^3+1,\ \alpha^3+\alpha+\alpha^2,\ \alpha^3,\ \alpha^2,\ 1+\alpha+\alpha^2+\alpha^3,\ \alpha+\alpha^3,\ 0],\ 8=[0,\\
&\alpha^3+\alpha+\alpha^2,\ 1,\ \alpha^2+\alpha^3,\ 1+\alpha^2,\ \alpha^2,\ \alpha+\alpha^2,\ \alpha,\ \alpha+\alpha^3,\ \alpha^3+1,\ 1+\alpha+\alpha^2+\alpha^3,\ 1+\alpha,\\
&\alpha^3,\ \alpha^2+\alpha^3+1,\ \alpha^2+1+\alpha],\ 9=[\alpha^3+\alpha+\alpha^2,\ 0,\ 1+\alpha+\alpha^2+\alpha^3,\ \alpha,\ \alpha^3+1+\alpha,\\
&\alpha+\alpha^3,\ \alpha^3,\ \alpha^2+\alpha^3,\ \alpha^2,\ \alpha^2+1+\alpha,\ 1,\ \alpha^2+\alpha^3+1,\ \alpha+\alpha^2,\ 1+\alpha,\ \alpha^3+1],\ 10=[1,\\
&1+\alpha+\alpha^2+\alpha^3,\ 0,\ \alpha^2+\alpha^3+1,\ \alpha^2,\ 1+\alpha^2,\ \alpha^2+1+\alpha,\ 1+\alpha,\ \alpha^3+1+\alpha,\ \alpha^3,\\
&\alpha^3+\alpha+\alpha^2,\ \alpha,\ \alpha^3+1,\ \alpha^2+\alpha^3,\ \alpha+\alpha^2],\ 11=[\alpha^2+\alpha^3,\ \alpha,\ \alpha^2+\alpha^3+1,\ 0,\ \alpha^3+1,\ \alpha^3,\\
&\alpha+\alpha^3,\ \alpha^3+\alpha+\alpha^2,\ \alpha+\alpha^2,\ 1+\alpha^2,\ 1+\alpha,\ 1+\alpha+\alpha^2+\alpha^3,\ \alpha^2,\ 1,\ \alpha^3+1+\alpha],\ 12=[\\
&1+\alpha^2,\ \alpha^3+1+\alpha,\ \alpha^2,\ \alpha^3+1,\ 0,\ 1,\ 1+\alpha,\ \alpha^2+1+\alpha,\ 1+\alpha+\alpha^2+\alpha^3,\ \alpha^2+\alpha^3,\\
&\alpha+\alpha^3,\ \alpha+\alpha^2,\ \alpha^2+\alpha^3+1,\ \alpha^3,\ \alpha],\ 13=[\alpha^2,\ \alpha+\alpha^3,\ 1+\alpha^2,\ \alpha^3,\ 1,\ 0,\ \alpha,\ \alpha+\alpha^2,\\
&\alpha^3+\alpha+\alpha^2,\ \alpha^2+\alpha^3+1,\ \alpha^3+1+\alpha,\ \alpha^2+1+\alpha,\ \alpha^2+\alpha^3,\ \alpha^3+1,\ 1+\alpha],\ 14=[\\
&\alpha+\alpha^2,\ \alpha^3,\ \alpha^2+1+\alpha,\ \alpha+\alpha^3,\ 1+\alpha,\ \alpha,\ 0,\ \alpha^2,\ \alpha^2+\alpha^3,\ 1+\alpha+\alpha^2+\alpha^3,\ \alpha^3+1,\\
&1+\alpha^2,\ \alpha^3+\alpha+\alpha^2,\ \alpha^3+1+\alpha,\ 1],\\
&15=[\alpha,\ \alpha^2+\alpha^3,\ 1+\alpha,\ \alpha^3+\alpha+\alpha^2,\ \alpha^2+1+\alpha,\ \alpha+\alpha^2,\ \alpha^2,\ 0,\ \alpha^3,\ \alpha^3+1+\alpha,\\
&\alpha^2+\alpha^3+1,\ 1,\ \alpha+\alpha^3,\ 1+\alpha+\alpha^2+\alpha^3,\ 1+\alpha^2]\\
&])
\end{aligned}
$$

Literatūros sąrašas

[Ada91] Jiri Adamek. Foundations of Coding. A Wiley-Interscience Publication JOHN WILEY&SONS, INC. 1991.

[AR98] Yonatan Aumann and Michael O. Rabin. Authentication, Enhanced Security and Error Correcting Codes. H. Krawczyk (Ed.), Advances in Cryptology - CRYPTO'98, 18th Annual International Cryptology Conference, Santa Barbara, California, USA, August 1998, Lecture Notes in Computer Science 1462, p. 299-303, 1998.

[DK] Peng Ding and Jennifer D. Key. Minimum-weight codewords as generators of generalized Reed-Muller codes.

[Gab95] Ernst M. Gabidulin. Public-key cryptosystems based on linear codes. 1995

[JKS94] Johansson, Gregory Kabatianskii, and Bernard Smeets. On the Relation between A-Codes and Codes Correcting Independent Errors. In: Lecture Notes in Computer Science 765, Advances in Cryptology - EUROCRYPT '93, p. 1-11, Springer-Verlag, 1994.

[JKS96] A.Kabatianskii, Ben J.M. Smeets, and T. Johansson. On the cardinality of systematic authentication codes via error-correcting codes. IEEE IT-42(2), pp. 566-578, March 1996.

[Jon03] Thomas Johansson. Authentication codes, URL: www.selmer.uib.no/researchcourse2004/ program/acodes_paper.ps**,** 2003

[Jon98] Thomas Johansson. On the Construction of Perfect Authentication Codes that Permit Arbitration. 1998

[RN87] T.R.N Rao, Kil-Hyun Nam. Private-key Algebraic-coded Cryptosystems. A.M. Odlyzko (Ed.): Advances in Cryptology – CRYPTO'86, LNCS 263, pp. 35-48, 1987

[HLL91] D.G.Hoffman, D.A.Leonard, C.C.Lindner, K.T.Phelps, C.A.Rodger, J.R.Wall. Coding Theory: The Essentials. Dekker, New York, 1991.

[MS77] F.J. MacWilliams, N.J.A. Sloane The Theory of Error Correcting Codes, 1977.

[PW04] Ruud Pellikaan, Xin-Wen Wu. List Decoding of q-ary Reed-Muller Codes. IEEE Trans. Inform. Theory, vol. 50, No. 4, April 2004, pp. 679-682

[Sta02] Vilius Stakėnas. Kriptologija. Paskaitų konspektas, URL: http://www.mif.vu.lt/matinf/asm/vs/pask/crypto/crypto.htm, 2002.

[Sim] Gustavus J. Simmons. Authentication Theory/Coding Theory. G.R. Blakley, David Chaum (Eds.), Advances in Cryptology, Proceedings of CRYPTO 84, Lecture Notes in Computer Science 196, p. 411-431, 1985.

[SS91] Reihameh S. Safavi-Naini and Jennifer R. Seberry. Error-correcting codes for authentication and subliminal channels. IEEE IT-37(1), pp. 13-17, Jan. 1991.

[Til98] Henk C. A. van Tilborg. Coding theory at work in cryptology and vice versa. In: V.S. Pless and W.C. Huffman (Eds.), Handbook of Coding Theory, ch. 14, Elsevier, 1998, pp. 1195-1227.

[XT99] Sheng-bo Xu and Henk van Tilborg. A construction of systematic authentication codes based on Error Correcting Codes. Daniel Augot, Claude Carlet (Eds.), Proceedings of Workshop on Coding and Cryptography, WCC'99, Paris, France, January 1999, INRIA, Paris, pp. 279-289, 1999.