VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
INFORMATIKOS KATEDRA

Magistro baigimasis darbas

Identifikavimo schemos, naudojančios klaidas taisančius kodus

(Identification schemes based on error-correcting codes)

Atliko:

Dmitrij Orlov .....
(parašas)

Darbo vadovas:

Gintaras Skersys .....
(parašas)

Recenzentas:

Algirdas Mačiulis .....
(parašas)

Vilnius

2006

Tyrinys

Įvadas

Šiuolaikinio žmogaus jau nebenustebinsi kažkokiom gudriom informacijos kodavimo mašinom, ar slaptais ryšio kanalais. Kiekvienas personalinis kompiuteris jau turi pakankamą kiekį įvairių saugumą užtikrinančių priemonių. Mes naudojamės slaptažodžių saugojimo sistemomis, pasirašome siunčiamus laiškus, balsuojame internetu (elektroninio balsavimo apygardos buvo naudojamos Amerikoje per paskutinius prezidento rinkimus), prisijungiame prie specializuotų sisteminių resursų, sudarome slaptus vaizdo ir garso ryšius su savo draugais kitoje pasaulio dalyje, naudojame koduotus ryšio kanalus duomenims ar kitai bankinei arba apskaitos sistemai pasiekti. Bet daug vartotojų dažnai nežino kaip toks programinis arba sisteminis funkcionalumas yra pasiekiamas. Aišku, daugeliui to žinoti ir nereikia, nes šiuolaikinės kriptografinės schemos yra labai sudėtingos ir reikalauja daug matematinių, statistinių, bei kartais ir fizikos žinių. Todėl tam tikrų žinių suvokimas ir įsisavinimas tampa tikrai sunkiu žinių apdorojimu. Tačiau nepaisant visų saugumo sistemų sudėtingumo, jų poreikis su kiekvienais metais vis labiau didėja. Daugiausia šių sistemų populiarumą lemia šiuolaikinių technologijų skvarba į mūsų kasdieninį gyvenimą, o ypač informacinės visuomenės formavimasis ir vystymasis. Prisiminkite, kada paskutinį kartą Jūs padarėte kokį nors pavedimą, arba nupirkote prekes parduotuvėje ir už pirkinį sumokėjote kreditine ar debetine kortele. Tačiau, turbūt, nė karto nepagalvojote, kas užtikrina Jūsų duomenų – sąskaitos, pinigų, konfidencialios informacijos – saugumą. Visgi duomenys perduodami internetiniu kanalu! Kaip gi saugus duomenų perdavimas, ar kitoks informacijos saugumas, tapo mūsų kasdieninio gyvenimo dalimi, ir mes atlikdami vieną ar kitą veiksmą, pasąmonėje tikimės, kad tai, ką mes darome bus ne tik atlikta teisingai, bet ir tais duomenimis negalės pasinaudoti kiti asmenys, kurie gali būti mūsų priešai ar šiaip nedraugiškai nusiteikę žmonės.

Apie saugių ir patikimų sistemų egzistavimą ir jų naudą galima kalbėti labai daug, atsižvelgiant į įvairius sistemų projektavimo, programavimo, diegimo ir kitus aspektus. Tačiau norėtųsi atkreipti dėmesį į tai, kas gi įtakoja tokių sistemų atsiradimą, tiksliau pasakius, kokios žinios mums leidžia, kurti tokias sistemas. Čia mes ir susiduriame su kriptografija, konkrečiau kalbant su matematiniais modeliais, skirtais skaitmeninės informacijos saugumui užtikrinti. Tokie matematiniai modeliai vadinami kriptografinėmis schemomis.

Kriptografinių schemų, kaip ir skirtingų obuolių rūšių, yra daug. Jos skirtos duomenų kodavimui, vartotojų identifikavimui ir autentifikavimui, elektroninių dokumentų pasirašinėjimui, taip pat yra daug kitokių tokių schemų naudojimo sričių. Tačiau kiekvienas

matematinis modelis pagrįstas tam tikrais matematiniais skaičiavimais – tai yra teorija. Ką reiškia matematinis modelis, pagrįstas kažkokia teorija? Šis klausimas yra natūralus, nes žodis teorija yra labai abstraktus ir turi daugybę reikšmių skirtinguose kontekstuose. Kalbant apie matematinius modelius, mes turime omenyje tam tikros matematinės šakos algoritmų sudėtingumą arba neišsprendžiamumą per polinominį laiką. Pavyzdžiui RSA algoritmas pagrįstas didelių skaičių faktorizacijos sudėtingumu, t.y. RSA algoritmas pagrįstas skaičių teorijos faktorizacijos uždavinio sudėtingumu. Šis uždavinys neturi efektyvaus sprendimo, tačiau jei toks atsirastų, RSA schema netektų savo prasmės.

Paskutiniu metu yra daug dirbama ties įvairių kriptografinių sistemų kūrimu ir senų sistemų saugumo patikrinimu. Buvo sukurta daug matematinių modelių, pagrįstų ne tik skaičių teorija. Kaip alternatyva šioms schemoms buvo sukurtos schemos, pagrįstos klaidas taisančių kodų teorija. Šiame darbe bus nagrinėjamos [Ste90] ir [Ste94] darbuose pasiūlytos schemos. Detaliau apžvelgsime pagrindines schemų idėjas, kaip ir teorines kurios buvo pasiūlytos aukščiau išvardintuose darbuose, taip ir praktines, t.y. schemų realizacijos aspektus – atsitiktinės matricos generavimą. Panagrinėsime schemų saugumo aspektus: galimas atakas, įveikimo laiką esant vienodom pradinėm sąlygom. Atsižvelgsime į praktinius aspektus – tokius kaip kodavimo/dekodavimo greičius. Aptarsime galimus schemų patobulinimus.

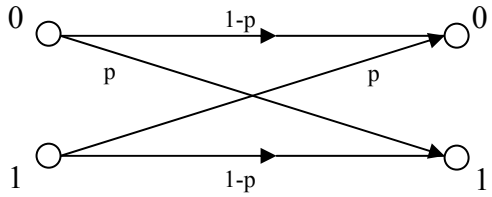
Identifikavimo schemos

Prieš pradedant tirti koki nors gamtos reiškinį ar matematinę problemą, svarbu suvokti, kokie darbai toje srityje jau yra padaryti, su kokiais reiškiniais ar sistemom mums reikės susidurti. Taip pat įdomus ir pats istorinis problemos atsiradimo arba reiškinio pastebėjimo aspektas. Todėl dabar apibudinsime svarbiausius nagrinėjamos srities objektus – analizuojamas schemas. Pateiksime trumpas schemų, įtakojusių nagrinėjamų schemų atsiradimui, aprašymus. Lengvesniam nagrinėjamos srities ir susijusių algoritmų supratimui, trumpai apibrėšime teorines sąvokas ir susijusius taikomus algoritmus.

Klaidas taisantys kodai

Klaidas taisantys kodai buvo sugalvoti, klaidų taisymui perduodant duomenis triukšmingu kanalu. Tarkime turime kokią nors telegrafinę stotį ir liniją Vilnius – Klaipėda, kuria galima siųsti 0 arba 1. Pas mus atėjo vyriškis ir prašo skubiai informuoti savo draugą esanti Klaipėdoje. Pritaikius tam tikras teksto transformavimo taisykles mes siunčiame telegramą į Klaipėdą.

Dažniausiai siunčiant 0 kitame linijos gale mes irgi gausime 0. Tačiau kai mes turime triukšmingą kanalą, Klaipėdoje mes galime gauti ne 0, o 1. Taip pat vietoj 1 kitame laido gale mes galime gauti 0. Tarkime kas šimtasis bitas bus klaidingas, tai reiškia kad kiekvienas simbolius gali būti iškraipytas su tikimybe $p = 1/100$ - tai reiškia kad tikimybė kad kanale įvyks klaida bus p. Žemiau pateiktame paveikslėlyje pavaizduotas aprašomas kanalas, kitaip vadinamas dvejetainiu simetriniu kanalu.



Pav. 1. Dvejetainis simetrinis kanalas.

Jeigu šito kanalų perduodama daug svarbių pranešimų ir mes nenorime kad jie būtų iškarpyti. Pavyzdžiui palydovas, kuris buvo išsiųstas į Marsą, nufotografavo daugeliui girdėtą „kaukę“ ir siunčia nuotraukas į Žemę. Suprantamai antros nuotraukos padaryti gali jau ir ne pavykti, nes palydovas gali tik praskristi pro Marsą, taip pat siunčiant duomenys gali įvykti duomenų iškraipymas. Tai gali įvykti dėl įvairių elektromagnetinių bangų arba dėl sugedusio imtuvo. Nepaisant visų galimų informacijos iškraipymo galimybių ir priežasčių, mes norime, kad siunčiama nuotrauka būtų saugiai gauta Žemėje. Tam užtikrinti mes užkoduojame siunčiamą informaciją.

Tarkime kad siunčiama informacija tai yra 1 ir 0 seka, tada mes galime ją skaidyti į k ilgio blokus $m = m_1 m_2 ..., m_k$, kur $m_i \in \{0,1\}$ ir $1 \le i \le k$. Šitas blokas bus koduojamas kaip vektorius $x = x_1 x_2 ... x_n$, kur $x_i \in \{0,1\}$, $1 \le i \le k$ ir $n \ge k$. Tokie kodo žodžiai x sudaro kodą.

Dabar sukonstruokime paprasčiausią kodą. Tarkime kad pirma kodo dalis yra siunčiamas pranešimas: $x_1 = m_1, m_2 = m_2, ..., x_k = m_k$, o likę $n-k$ bitų bus kontroliniai bitai $x_{k+1}, x_{k+2}, ..., x_n$. Kontrolinius bitus mes galime parinkti taip kad jie tenkintų lygti:

$$H \begin{pmatrix} x_1 \\ x_2 \\ \dots \\ x_n \end{pmatrix} = Hx^T = 0$$

Čia matrica $((n-k) \times n)$ - H, vadinama kodo kontroline matrica ir turį pavidalą:

$$H = [A \mid I_{n-k}]$$

Čia A – kažkokia fiksuota $((n-k)\times k)$ matrica, kurią sudaro 0 ir 1, o I_{n-k} - vienetinė $(n-k)\times(n-k)$ matricą.

Tačiau atsiranda labai natūralus klausimas, o kaip turint pranešimą m gauti jam atitinkanti kodo žodį x? Tarkime, kad mes turime kažkokio kodo kontrolinę matricą H. Tada sukonstruokime matricą G pavidalo:

$$G = [I_k \mid -A^T]$$

Čia I_k yra vienetinė $k \times k$ matrica, o $-A^T$ - atvirkštinė transponuota kontrolinės matricos A matricą. Tokiu atveju, matricą G vadinama klaidas taisančio kodo generuojančia matrica. Ir jeigu mes norime pranešimui m rasti žodį x priklausanti kodui užkoduotam generuojančios matricos G, mums tiesiog reikia paskaičiuoti sekančia išraišką:

$$x = mG$$

Tačiau, ką daryti, kai mes gauname užkoduotą žodį? Tarkime kad žodis $m = m_1 m_2 ..., m_k$ yra užkoduotas į kodo žodį $x = x_1 x_2 ... x_n$, kuris buvo perduotas kanalu. Kadangi mes turime triukšmingą kanalą, tada gautas vektorius $y = y_1, y_2, ..., y_n$ gali skirtis nuo išsiusto vektoriaus x. Tada vektorius $e = y - x = e_1, e_2, ..., e_n$ vadinamas klaidų vektoriumi. Vadinasi dekoderis pagal gautą pranešimą y turi rasti, koks buvo siųstas pranešimas m. Kartais lengviau rasti koks buvo išsiustas pranešimas x. Aišku, dekoderiui pakaktų rasti klaidų vektorių e, kadangi $x = y - e$. Tačiau dekoderis niekados negali žinoti kokose pozicijoje įvyko klaida, tai reiškia, kad jis negali nustatyti, koks buvo vektorius e? Dekoderio strategija – pagal gautą vektorių y, rasti labiausiai tikėtiną klaidų vektorių e. Su prielaida, kad visi kodo žodžiai turi vienodą pasirodymo tikimybę.

Kadangi tikimybė, kad perduodant žodį x įvyks viena klaida didesnė už tikimybę kad įvyks dvi klaidos. Todėl dekodavimo strategija būtų tokia: y dekoduojasi į artimiausią pagal Hemmingo atstumą jam žodį x. Kitaip tariant, parankamas klaidų vektorius e turintis mažiausią svorį. Toks dekodavimas vadinamas dekodavimų į artimiausią žodį.

Hemmingo atstumu tarp vektorių $x = x_1 x_2 ... x_n$ ir $y = y_1, y_2, ..., y_n$ vadinamas pozicijų skaičius, kuriose šie vektoriai skiriasi. O vektoriaus $x = x_1 x_2 ... x_n$ Hemmningo svoris bus skaičius vektoriaus nenulinių koordinačių Akivaizdu, kad $dist(x, y) = wt(x - y)$.

Jeigu mes perrinksim visus 2^k binarinio kodo žodžius ir rasime žodį artimiausia y, toks dekodavimas vadinamas – dekodavimas su pilnu perrinkimu. Šis būdas yra geras, kai mes turime mažus kodus, tačiau, kai k reikšmė yra didelė, toks būdas yra netinkamas. Vienas iš

kodo teorijos uždavinių – paieška dekodavimo algoritmo, kuris skiriasi nuo pateikto aukščiau algoritmo.

Dar vienas iš labai svarbių klaidas taisančio kodo parametrų, tai minimalus atstumas. Klaidas taisančio kodo minimaliu atstumu vadinamas minimalus Hemmingo atstumas tarp visų kodo žodžių:

$$d = \min\{dist(u,v)\} = \min\{wt(u-v)\}$$

Kur $u \in C$, $v \in C$ ir $u \neq v$. C – klaidas taisantis kodas. Klaidas taisantis kodas ilgio n, dimensija k ir minimaliu atstumu d vadinsis C[n,k,d] kodu.

Kodas C[n,k,d] gali ištaisyti $t = \left\lfloor \frac{d-1}{2} \right\rfloor$ klaidų. Ir jeigu d dalinasi iš dviejų, tada kodas gali ištaisyti $t = \frac{d-2}{2}$ klaidų ir aptikti $\frac{d}{2}$ klaidų.

Jei turime C[n,k] kodą virš kūno q, tai kodo klasę sudarys visi vektoriai pavidalo:

$$a + C = \{a + x : x \in C\}$$

Kiekvienas kodas turi $q^{n-k} - 1$ klasę, kurią sudaro q^k vektorių.

Tarkime, kad dabar dekoderis gauna vektorių y. Jeigu šis vektorius nepriklauso kodui, vadinasi jis turi priklausyti kažkokiai kodo klasei. Tada gauname, kad klaidų vektoriai gali būti visi šios klases vektoriai.

Pagal esamą dekodorio strategiją, jo uždavinys būtų iš klasės, kuriai priklauso žodis y, išrinkti mažiausio svorio klaidų vektorių y ir dekoduoti gautą žodį $x = y - e$. Kodo klasės vektorius, turintis mažiausią svorį, vadinamas klasės lyderiu. Jeigu tokių vektorių yra ne vienas, atsitiktinai pasirenkame vieną iš jų ir pavadiname klasės lyderiu.

Vienas iš galimų klaidas taisančių kodų dekodavimo metodų būtų sekančios lentelės sudarymas:

Pranešimas	00...00	00...01	...	11...11
Kodas	00...00	01...11	...	01...00
Klasė 1	00...11	11...00	...	10...10
Klasė 2.	01...10	00...11	...	01...10
...	...	...	...	...
Klasė q^k-1	11...00	10...11	...	11...01

Lentelė 1. Galima dekoderio dekodavimo lentelė.

Pirmą šios lentelės eilutę sudaro galimi pranešimo žodžiai. Antra eilutė – tai atitinkamų pranešimų kodo žodžiai. Kiekvienos toliau einančios eilutės pirmas žodis tai žodis turintis

mažiausią svorį tarp visų dar neįrašytų į lentelę žodžių. O visi kiti eilutės žodžiai yra sumos pirmo eilutės žodžio su atitinkamu kodo žodžiu. Gaunasi, kad kiekviena eilutė atspindi kodo klasę, o pirmas eilutės žodis yra klasės lyderis.

Egzistuoja dar vienas būdas sužinoti ar vektorius priklauso kodui, ar ne. Tam gautą žodį y reikia padauginti iš kodo kontrolinės matricos H ir jeigu gautas vektorius bus nulinis vektorius, tada ir tik tada žodis priklauso kodui. Vektorius $s = Hy^T$ vadinamas žodžio sindromu. Kadangi $y = x + e$, kur $x \in C$, vadinasi $s = Hy^T = Hx^T + He^T = He^T$. O tai reiškia, kad žodžio sindromas yra ne kas kita, o visų kontrolinės matricos stulpelių, kuriose įvyko žodžio iškraipymas, tiesine kombinacija. Taip pat, pagal anksčiau pateikta teiginį, visi vienai klasei priklausantis žodžiai turi vienodą sindromą.

Pagal šį teiginį, galima šiek tiek sumažinti dekoderio naudojamą lentelę. Kaip matome, tam, kad sužinoti kokiai klasei priklauso žodis, mums pakanka išskaičiuoti žodžio sindromą ir pagal jį surasti klasės lyderį. Anksčiau mums reikėjo saugoti visus klasės vektorius, dabar dekoderio lentelėje mums pakanka išsaugoti klasės lyderį ir jam atitinkantį sindromą. Dabar dekoderio lentelė galėtų atrodyti taip:

		Sindromas
Kodas	00...00	00...00
Klasė 1	00...11	10...10
Klasė 2.	01...10	01...10
...	...	...
Klasė q^k-1	11...00	11...01

Lentelė 2. Dekoderio patobulinta dekodavimo lentelė.

Trumpai apibūdinę nagrinėjamą sritį, apžvelkime nagrinėjamas schemas, susijusius straipsnius ir šių schemų atsiradimui įtakojusius schemas.

Pateikime viena pavyzdį. Tarkime turime klaidas taisanti kodą C[4, 2] su generuojančia matrica:

$$C = \begin{bmatrix} 1011 \\ 0101 \end{bmatrix}$$

Tada dekoderio dekodavimo lentelė atrodys taip. Iškart pastebėsim, kad papildomas stulpelis – Sindromas – skirtas supaprastintai lentelei, tačiau, kad vaizdžiau pateikti, kaip atrodo skirtingos dekodavimo lentelės, ir nepaišyti kelių lentelių, apjungsim jas į vieną:

Pranešimas	00	10	01	11	Sindromas
Kodas	0000	1011	0101	1110	$\begin{pmatrix}0\\0\end{pmatrix}$
Klasė 1	1000	0011	1101	0110	$\begin{pmatrix}1\\1\end{pmatrix}$
Klasė 2	0100	1111	0001	1010	$\begin{pmatrix}0\\1\end{pmatrix}$
Klasė 3	0010	1001	0111	1100	$\begin{pmatrix}1\\0\end{pmatrix}$

Lentelė 3. Kodo C[4, 2] dekodavimo lentelė.

Kad suvesti lentelę į pirmą pavidalą, reikia išmesti sindromo stulpelį. Tam, kad suvesti į sutrumpintą – reikia išmesti stulpelius 10, 01, 11.

McEliece viešo rakto kriptoschema

Šis algoritmas yra vienas iš pirmų algoritmų, kuris naudoja klaidas taisančius kodus. Kaip žinome, norint atrinkti užduoto svorio vektorių iš visos tiesinio kodo kodų aibės yra NP-pilna problema. Būtent šio uždavinio sudėtingumu ir remiasi ši kriptoschema.

Tarkime mes turime (n, k) tiesinį binarinį kodą virš kūno GF(2), kurį generuoja matrica G ir kuriam mes turime dekodavimo algoritmą, kuris gali ištaisyti ne daugiau nei t klaidų. Mes atsitiktinai pasirenkame matricą S, kuri turi atvirkštinę matricą, ir perstatų matricą P. Tada trejetas <G, S, P> bus privatus sistemos raktas, o <t, W> - viešas, kur $W = SGP$.
Pati schema yra labai paprasta ir labai panaši į RSA kriptoschemą.

- Kad persiųsti k-bitų pranešimą m, vartotojas – A - užkoduoja pranešimą:

$$c = mW \oplus e,$$

kur e yra atsitiktinis klaidų vektorius, kurio svoris yra nedidesnis už t. Ir išsiunčia pranešimą kitam vartotojui – B.

- B gavęs pranešimą jį dešifruoja,

$$cP^{-1} = mSG \oplus eP^{-1}$$

Tarkime įsilaužėlis nori įveikti McEliece kriptoschemą ir žino viešai prieinamą matricą W ir gali perimti kanalu perduodamą reikšmę c. Norint įveikti šią kriptoschemą įsilaužėlis gali taikyti sekančias taktikas. Pirmoji būtų pabandyti atkurti generuojančia kodo matricą G iš

žinomos W matricos. Antra ataka – tai bandymas sukurti tokį prasmingą pranešimą m', kad $c = m'W + e$.

[Mce78] straipsnio duomenimis, atlikti pirmo tipo ataka yra labai sunku, ypač kai n ir t yra labai dideli (čia n – generuojančios klaidas taisančio kodo matricos stulpelių skaičius, o t – skaičius klaidų, kurias gali ištaisyti klaidas taisantys kodas). Nes esant dideliems n ir t parametrams egzistuoja labai daug galimų matricos G variantų. Nekalbant jau apie P ir S matricų galimus variantus.

Taikant antrą taktiką, galima sulaukti geresnių schemos įveikimo rezultatų. Antros schemos atakos metu, įsilaužėliui reikia surasti tokias m' ir e' reikšmes, kad sąlyga $c = m'W + e'$ būtų teisinga. Tačiau dar viena problema: tokių porų (m', e') gali būti daug, tačiau prasmingų pranešimų tik keletas. Šita problema taip pat yra neišsprendžiama per polinominį laiką.

Pagal anksčiau pateikto straipsnio duomenys rekomenduotini schemos parametrai yra n = 1024, k =524 ir t = 50. Tada operacijų skaičius reikalingas įveikti nagrinėjamą schemą būtų 2^{69}.

Tačiau jau 1986 buvo pasiūlyta kriptoschema, kurioje vietoj klaidas taisančio kodo generuojančios matricos G buvo naudojama kodo kontrolinė matrica H.

Niederreiter kriptoschema

1986 metais savo straipsnyje [Nie86] H. Niederreiter pasiūlė naują idėją – vietoj klaidas taisančio kodo generuojančios matricos nauduoti klaidas taisančio kodo kontrolinę matricą. Dabar bendrai aprašykime kaip veikia pasiūlyta kriptoschema.

Tarkime mes turim (n, k) tiesinį klaidas taisanti kodą C virš GF(q) kūno. H – kodo C kontrolinė matrica. Tada, kaip ir McEliece schemos atveju, mes sugeneruojame klaidas taisančio kodo kontrolinę matricą H, atsitiktinai pasirenkame matricą S, kuri turi atvirkštinę matricą, ir perstatų matricą P. Tada matrica $K = MHP$ bus Niederreier‘io schemos viešas raktas. O schemos slaptą raktą sudarys M, H ir P matricos.

Norint išsiusti pranešimą m su Hemmingo svoriu t, vartotojas skaičiuoja sekančia išraišką $c = Km^T$ ir siunčia užkoduotą žodį c kitam sistemos vartotojui.

Gavėjas gavęs pranešimą c daugina jį iš matricos M atvirkštinės - M^{-1}, ir gauna $HPm^T = H(mP^T)^T$. Toliau paprasčiausiai dekoduojame gautą pranešimą ir gauname vektorių mP^T, iš kurio galime lengvai atstatyti pradinį pranešimą m.

Šitai, kaip ir auščiau minėtai McEliece kriptoschemai, galima taikyti tokias pačias įveikimo taktikas.

Apie skirtingus kriptografinius algoritmus galima kalbėti daug, bet šiame darbe kalbėsime tik apie identifikavimo kriptoschemas.Tačiau, prieš aprašant nagrinėjamas schemas, reikėtų apibudinti dar vieną kriptografinę schemą, kurios idėja buvo pabauduota nagrinėjamose kriptoschemose.

Fiat-Shamir identifikacijos protokolas

Prieš kalbant apie identifikavimo protokolus be paslapties atskleidimo(zero-knowledge), pagrįstus kodų teorijos algoritmų sudėtingumais, reikėtų keletą žodžių tarti ir apie tai, kaip tokie protokolai atsirado ir kokia jų prasmė.

Naršant internetu arba naudojantis kažkokiom paslaugom, kurios reikalauja vartotojo identifikacijos ir autentifikacijos, mes patys nežinodami galime pateikti vertingos informacijos kokiam nors nepatikimam arba priešiškai nusiteikusiam žmogui. Pabandykime truputi pafantazuoti: tarkime, Jūs esate koks nors įtakingas politikas, aišku, kaip visi įtakingi ir garsūs žmonės turite kaip gerų ir patikimų draugų, taip politinių ar kitokio pobūdžio „priešų“. Ir atlikdami vienokias ar kitokias operacijas, pagal paprastus identifikavimo protokolus, patys to nenorėdami paliekate vienokius ar kitokius savo darbo sistemoje požymius. Tai gali būti saugumo įrašai, kuriuose atsispindi Jūsų prisijungimo vardas – pagal kurį nesunku atsekti, kas buvo prisijungęs prie sistemos, arba kokia nors kitokia panašaus pobūdžio informacija. Jei Jūs netgi neatlikote jokių neteisėtų veiksmų, bet pavyzdžiui kalbėjote su prieštaringos reputacijos žmonėm, pasinaudojus interneto technologijomis, toks pokalbis Jus gali stipriai sukompromituoti, jei apie jį sužinotų žurnalistai. Todėl visai logiška sukurti tokias identifikacijos schemas, kuriuos užtikrina ir vartotojo tapatybę ir neatskleidžia jokios slaptos ar nepageidautinos informacijos apie vartotoją. Apie tokius protokolus mes kalbėsime toliau.

Pirmą kartą tokio tipo schema buvo pristatyta 1985 m.Goldwasser, Micali ir Racoff bendrame darbe. O primas protokolas, turintis praktinę galią, buvo pasiūlytas 1987 m. A. Fiat and A. Shamir (Fiat-Shamir identifikacijos schema - tai interaktyvių zero-knowledge įrodymų ir identifikacijos schemų kombinacija) straipsnyje. Naudojant šią schemą vartotojo tapatybė įrodoma be bendrų arba viešų raktų naudojimo algoritmų. Schema pagrįsta kvadratinės šaknies traukimo sudėtingumu, kai mes nežinome skaičiaus n faktorizacijos.

Šios schemos funkcionavimui reikia patikimo asmens Z (kokios nors patikimos organizacijos – karinio štabo, kompanijos gaminančios kreditų korteles, vyriausybės), kurio

vaidmuo – išduoti asmens tapatybės įrodymo kortelę asmeniui, prieš tai fiziškai patvirtinus jo tapatybę. Tačiau čia trečio asmens pagalba ir pasibaigia, tolimesniam schemos darbui - įrodymų generavimui ir patikrinimui – trečio asmens pagalbos nereikės. Be to, pasinaudojant šia schema galima sukurti tapatybės atpažinimo sistemą su neribotu vartotojų skaičiumi. Dar vienas didelis šios schemos pliusas tai, kad vartotojų skaičius sistemoje niekaip neįtakoja sistemos veikimo greičio ir neapkrauna kažkokios fizines duomenų bazės, nes naudojantis šia sistema nereikia kaupti informacijos apie sistemos vartotojus arba turėti bendrą vartotojų sąrašą. Be to, identifikavimo kortelės bus apsaugotas nuo kortelių atgaminimo tapatybę tikrinančios šalies.

Prieš pradedant gaminti tapatybės įrodymo korteles, patikimas asmuo Z, pasirenka modulį n ir hash funkciją f, kuri bet kokią tekstinę eilutę paverčia skaičiumi iš intervalo [0, n). Modulis n tai yra dviejų priminių skaičių p ir q sandauga ir tiktai patikimas asmuo žinos modulio faktorizaciją.

Bendrą informaciją apie schemos duomenis jau įvardinom, dabar aptarkime kortelės duomenų struktūrą. Kai koks nors asmuo paprašys išduoti jam kortelę, kortelių platinimo centras sudarys tam tikrą eilutę **s** į kurią įtrauks vartotojo vardą ir pavardę, gimimo metus, lytį ir dar kokią nors svarbią, jo nuomone, informaciją. Tai gali būti kortelės galiojimo data, apribojimai kortelės naudojimui ir daug daug kitų požymių, kurie priklauso nuo kortelės taikymo srities. Toliau kortelių platinimo centras atlieka sekančius žingsnius:

- Skaičiuoja $v_j = f(s, j)$, kažkokiom mažom j reikšmėm.
- Pasirenka k skirtingų j reikšmių, kurioms egzistuoja kvadratinė šaknis iš v_j^{-1} moduliu n ir paskaičiuoja s_j, kur s_j yra mažiausia v_j^{-1} (mod n) kvadratinė šaknis.
- Sukuria kortelę, į kurią įrašo s ir k s_j reikšmes ir jų indeksus.

Pastaba:

Kad supaprastinti schemos aprašymą, tarkime kad mes visą laiką imame pirmas k reikšmių, tada gaunme indeksus, lygius 1, 2, 3, ... k.

Jei funkcija f nėra tobula, galima iškraipyti s reikšmę pridėjus prie jos kažkokią atsitiktinę eilutę R, kurią pasirenka kortelių platinimo centras.

Mes apibrėžėme bendrus ir kortelės duomenis. Dabar aprašysime identifikacijos schemą. Kad būtų parprasčiau suprasti pačią schemą, įvardinkime du šios schemos dalyvius. Primas dalyvis būtų pati tapatybės kortelė – jai atiteks įrodinėtojo vaidmuo. Antras asmuo, tai kortelės tikrinimo aparatas, kuris savyje turi mikroprocesorių, truputi atminties ir įvedimo/išvedimo interfeisą. Vienintelė informacija, kuri turi būti saugoma šiame įrenginyje, tai modulis n ir funkcija f. Kai kortelė įdedama į įrenginį, ji pasako, kad žino kažkokias

reikšmes s_1 s_k, tačiau neatskleidžia jų reikšmių. Tada tapatybės įrodymui naudojamas sekantis protokolas:

1. A (kortelė) siunčia B (įrenginiui) reikšmę s.
2. B (įrenginys) generuoja $v_j = f(s, j)$, kur $j = 1 \ldots k$
3. A pasirenka atsitiktinį skaičių r_i priklausantį aibei [0, n). Ir siunčia $x_i = r_i^2 \pmod n$ asmeniui B.
4. B gavęs r_i reikšmę išsiunčia A atsitiktinį binarinį vektorių ($e_{i1} \ldots e_{ik}$)
5. A skaičiuoja reikšmę y_i ir siunčia ją B, kur

$$y_i = r_i \prod_{e_{ij}=1} s_i \pmod n$$

6. B gavaus y_i reikšmę paskaičiuoja x_i, kur

$$x_i = y_i^2 \prod_{e_{ij}=1} v_j \pmod n$$

Žingsnius nuo 3 iki 6 reikia pakartoti t kartų.

Protokolas yra įvykdytas, t.y. B priims A identifikacijos įrodymą, jei visi t žingsnių bus sėkmingi. Priešingu atveju, identifikacija bus atmesta. Nesunku įrodyti kad remiantys šio protokolu, vartotojo identifikacija bus atlikta sėkmingai:

$$y_i^2 \prod_{e_{ij}=1} v_j = r_i^2 \prod_{e_{ij}=1} (s_j^2 v_j) = r_i^2 = x_i \pmod n$$

Šios schemos saugumas, t.y. tikimybė kad mus palaikys kitu asmeniu, yra 2^{-kt}. Tarkime kad A nežino s_j reikšmių ir per polinominį laiką negali paskaičiuoti kvadratinės šaknies iš

$$\prod_j^k v_j^{e_j} \pmod n$$

Tada vienintelė galimybė sukčiauti yra atspėti vektorių $\{e_{ij}\}$, bet tokia tikimybė vienai iteracijai 2^{-k}, o tokių iteracijų mes turime t, tai reiškia kad schemos saugumas yra lygus 2^{-kt}. Aprašant šį protokolą pamatėm, kad laikas 2^{kt} yra daug mažesnis už laiką, kurio reikės norint faktorizuoti modulį n. Taip pat, jei asmuo A generuos polinomiškai daug tapatybės įrodymų, tai nepadidins tikimybės įveikti protokolą: A naudoja x_i tik vieną kartą kiekvienai užklausai atsakyti ir vienodos reikšmės x_i gali pasikartoti, jei tik bus atsiųstos vienodos užklausos. Todėl tikimybė įveikti algoritmą nepasikeičia (2^{-kt}).

Algoritmo greičiui padidinti, galima atlikti kai kuriuos veiksmus lygiagrečiai, t.y. A išsiunčia visus x_i, o B siunčia visus e_{ij} ir A siunčia visus y_i. Tačiau dėl techninių priežasčių toks algoritmas jau nebus zero-knowledge.

Tikimybės 2^{-kt}, kad algoritmas bus įveiktas, visiškai pakanka ir visiškai nėra būtina didinti k ir t reikšmes, atsižvelgiant į tolimesnę technikos pažangą. Tikimybės 2^{-20} visiškai užteks, nes niekas neįteiks negaliojančio paso pasienyje, nemėgins įkišti blogos kreditinės kortelės banke, nesieks apgauti policininko, bandyti prasmukti į uždarą teritoriją, kai tikimybė, kad pasiseks atlikti neteisėtus veiksmus, bus lygi vienas prie milijono. Tačiau nacionaliniam saugumui, gal to ir nepakaks, todėl galima sumažinti įsilaužimo tikimybę iki 2^{-30}. Tarkime, kad pagaminti vieną identifikacijos kortelę kainuoja apie 1 dolerį. Vadinasi labai turtingas įsilaužėlis galės pabandyti kelis kartus per dieną ir kaip nors prasmukti nepastebėtas. Tačiau, kad ir kiek turtingas būtų įsilaužėlis, jis gali bandyti įrodyti savo tapatybę, nors ir 1000 kartų per dieną, jam vis tiek prireiks apie 3000 metų kad tą pasekti.

Kad pasiekti saugumo lygį lygų 2^{-20}, reikia pasirinkti k = 5 ir t = 4 (iki 2^{-30}, kiekvieną parametrą reikia padidinti vienetu). Tada tam, kad sugeneruoti arba patikrinti tapatybės įrodymą vidutiniškai prireiks t(k + 2) / 2 operacijų (14). Tarp A ir B bus persiųsta 323 baitai duomenų ir s_j reikšmėms saugoti bus išnaudota 320 baitų ROM'o. Jei mes naudosime e_{ij} vektorius kurių svoris yra 3, kiekvienai iteracijai mes galėsime pasirinkti 988 skirtingus vektorius. Jei iteracijų skaičių sumažinsime iki t = 2, tikimybė įveikti schemą vis tiek bus lygi maždaug vienas prie milijono, o perduodamos informacijos kiekis sumažės iki 165 ir vidutinis daugybos operacijų skaičius – 7.6. Kaip matome, keisdami k,t ir e_{ij} parametrus mes galime modifikuoti schemos saugumą pagal mums reikalingus kriterijus (atsižvelgiant į resursų kiekį). Ir kadangi schemoje naudojamos tik daugybos operacijos, ji yra tinkama realaus laiko sistemoms.

Alternatyvus Fiat-Shamir protokolas

Aukščiau buvo aprašytas pirmasis zero-knowledge identifikacijos protokolas, kuris bovo pasiūlytas Amos Fiat ir Adi Shamir darbe, 1987 m. Ir kaip buvo minėta aukščiau, egzistuoja daug panašių protokolų, bet pagrįstų skirtingais sudėtingais matematiniais uždaviniais. Tačiau šiame ir kituose skyriuose apžvelgsime zero-knowledge identifikacijos protokolus, pagrįstus klaidas taisančių kodų teorija.

Šis identifikacijos protokolas naudoja tik paprastas operacijas. Iš tikro tai bus daugyba iš matricos virš kūno iš dviejų elementų {0, 1}. T.y. mums prireiks tik dviejų operacijų – sudėties ir daugybos moduliu du. O į matricą mes galime žiūrėti kaip į tiesino binarinio klaidas taisančio kodo kontrolinę matricą. Mintis panauduoti klaidas taisančių kodų matricas, prima karta buvo išreikšta S. Harari (1988 m.), bet protokolas, kuris aprašytas žemiau yra ne tik paprastesnis, bet ir žymiai saugesnis.

Prieš pradedant nagrinėti schemos protokolą, trumpai aptarkime bendrus schemos duomenis ir parametrus. Kaip buvo pateikta aukščiau, schema naudoja (n-k, n) matricą - H, kuri yra tiesinio binarinio klaidas taisančio kodo kontrolinė matrica. Be to, ši matrica yra bendra visiems sistemos vartotojams ir dažniausiai sudaroma atsitiktiniu būdu. Dar vienas bendras duomenų masyvas – tai n bitų ilgio žodžių w_1 ... w_q sąrašas, kurio prasmė bus paaiškinta žemiau.

Kiekvienas vartotojas pasirenka slaptą raktą s, kuris yra n bitų ilgio žodis. Tačiau pasirinkto slapto rakto svoris w(s) turi būti lygus p. T.y. vienetų skaičius žodyje s turi būti lygus p. p – taip pat yra schemos bendras parametras. Toliau vartotojas skaičiuoja savo identifikatoriaus reikšmę

$$i = H(s)$$

Detaliau ši operacija bus aprašyta žemiau, čia norisi pabrėžti, kad H(s) užrašas, atspindi

išraišką $H(s) = H \cdot s^T$.

Žodžiai w_1 ... w_q bus naudojami kaip atsitiktinių skaičių generatorius. Tarkime, turime n bitų ilgio žodį z, tada atsitiktinai pasirenkame du sveikus skaičius a ir c, kur a turi būti pirminis su q. Tada galime sugeneruoti pseudo-atsitiktinių skaičių seką:

$$z_j = z \oplus w_{aj + c \bmod q} \; 1 \le j \le q$$

Dabar keletą žodžių reikia tarti ir apie sudėtingiausią šios schemos dalį, tai n ilgio žodžio parašas. N bitų žodžio parašu laikysime seką $H(u_1)$ ... $H(u_l)$, sukurtą iš n ilgio žodžių u_1 ... u_l, kur

$$\begin{cases} u_i \bullet u_j = 0 \\ \oplus_{i=1}^{l} u_i = u \end{cases}$$

Kur $\bullet$ yra loginė daugybos operacija (AND), o $\oplus$ - sudėtis moduliu du (XOR). Aišku, geriausiai sukurti žodžio parašą atsitiktinai suskaidžius žodį į segmentus {1 ... l}, o vėliau praplėsti kiekvieną iš sudarytų dalių iki n ilgio žodžio . Kitaip tariant parašas bus naudojamas kaip hash funkcijos rezultatas, norint paslėpti u_1 ... u_l reikšmes iš kurių jis yra sudarytas. Parašo teisingumą galima patikrinti atskleidus reikšmes u_1 ... u_l, o tada kiekvienas gali patikrinti mūsų tapatybę, padauginus kiekvieną u_j $1 \le j \le l$ iš matricos H.

Dabar aprašysime dar vieną protokolą, kuris įgalina kiekviena vartotoją (A) įrodyti kitam asmeniui (B) savo tapatybę. Protokolą sudaro r etapų, kurie susideda iš sekančių žingsnių:

1. A pasirenka atsitiktinį n bitų ilgio žodį y ir išsiunčia y ir $y \oplus s$ parašus asmeniui B.

2. B patikrina ar $x \equiv x' \oplus i$, kur $i = H(s)$, $x = H(y)$, o $x' = H(y \oplus s)$. Ir jei sąlyga yra teisinga, tai siunčia A atsitiktinį žodį z, kurio ilgis lygus n.
3. A paskaičiuoja $n_j = w(y \oplus z_j)$ ir $m_j = w(y \oplus z_j \oplus s)$, kur w(v) yra vektoriaus v svoris, o z_j yra skaičių seka, sugeneruota naudojant atsitiktinių skaičių generatorių, kuris aprašytas aukščiau. Po to siunčia B parašus kiekvienai iš sugeneruotų sekų n_j ir m_j, kur $1 \le i \le q$, $1 \le j \le q$.
4. B sugeneruoja atsitiktinį skaičių b iš aibės {0, 1, 2}
5. Jei b lygus 1 arba 0, tai A paskelbia parašą žodžiui y‘, kur $y' = y \oplus b * s$. Ir dar paskelbia vieną iš žodžių – n_j, jei b = 0 ir m_j, jei b = 1. Jeigu b = 2, A paskelbia informacija apie sekas n_j ir m_j.
6. Jei b = 0 arba 1, B patikrina ar parašai buvo sudaryti teisingai ir ar skaičiai n_j arba m_j buvo paskaičiuoti tesingai. Jei b = 0, B patikrina sekų parašus ir paskaičiuoja:

$$\mu = \frac{1}{q} \sum_{j=1}^{q} w(n_j - m_j)$$

Ir priima sprendimą apie etapo rezultatą, pagal nelygybę:

$$\mu < 1.07\sqrt{p}$$

Trumpai aptarkime schemos saugumą. Pirmiausia norėtųsi sustoti ties galimomis schemos viešųjų raktų kolizijomis. Tarkime mes turime du raktus s ir s‘ su vienoda h-funkcija. Tada žodis $s \oplus s'$ turi didžiausią svorį lygu 2p. Dabar prisiminkime kad H sugeneruota atsitiktinai, taip, kad klaidas taisantis kodas turi gerą klaidų taisymo galimybę. T.y., yra žinoma, kad tokie kodai turi tenkinti Gilbert-Varshamov sąlygą. Remiantys šia sąlyga, mes turime, kad k = n/2 matricos parametrams, bet koks nenulinis vektorius turės svorį ≈ 0.11n. Tai reiškia, kad p turi būti mažesnis už 0.055n, kad apsaugotų kodą nuo galimų kolizijų. Bet, net jei mes ir neišpildysim šios sąlygos, kolizijos tikimybė yra labai maža.

Kadangi schemos saugumas pagrįstas funkcijos

$$s \to H(s)$$

saugumu, kai jos argumentai yra „geri“ slapti raktai. Kaip mes žinome, kad nustatyti kokiam žodžiui s su svoriu ≤ p atitinka ilgio k žodis i yra NP-pilna problema.

Kad sukompromituoti schemą, nežinant slaptojo rakto, turi būti taikomos sekančios strategijos:

- Paruošti kažkokį y ir reikšmę, kuri yra labai arti $w(y \oplus z_j)$, vietoj m_j. Šiuo atvejų įsilaužėlis tikisi kad B atsiųs 0 arba 2 ir tikimybė, kad jam pasiseks lygi $(2/3)^r$, kur r

yra etapų skaičius. Panaši strategija gali būti taikoma ir $y \oplus s$ atveju. Tada mes gauname tokią pat įsilaužimo tikimybę.

- Antra galimybė - tai paruošti tokius skaičius y ir $y \oplus t$, kur t toks žodis, kad $H(t) = i$. Aišku, t nėra s, bet įsilaužėlis stengsis parinkti tokį t, kad jo ir realaus s skirtumas būtų kuo mažesnis. Jeigu įsilaužėlis manys, kad šiame raunde jam nepasiseks, jis gali grįžti prie pirmo varianto.

Dabar aprašysim sistemos parametrų variantus, kurių dėka algoritmas, naudojantis klaidas taisančius kodus, galės užkirsti kelią įsilaužimui. Šie sistemos parametrai pagrįsti tuo, kad įsilaužėlis gali generuoti kodus, kurių svoris yra artimas vidutiniam $H^{-1}(i)$ elementų svoriui.

- Tarkime n=512, k=256, l=4, p=30, q=128. Tarkime įsilaužėlis gali generuoti $H^{-1}(i)$ elementus su vidutiniu svoriu 0.2n. Darant tokią prielaidą, kiekvieną kartą, kai įsilaužėlis bando kažkokią t reikšmę, jis turi galimybę sugeneruoti tinkamą μ, kuri yra artima ribai $1.07\sqrt{30} \approx 5.86$, lygią $9*10^{-5}$ (vienai minutei skaičiavimo). Tai aišku ženkliai nepadidina galimybės sukčiauti. Tada tikimybė sėkmingai įveikti r etapų bus lygi $(0.67)^r$.
- Tarkime sistema turi sekančius parametrus n=1024, k=512, l=8, p=40, q=128, su prielaida kad įsilaužėlis gali generuoti $H^{-1}(i)$ elementus su svoriu artimu 0.12n. Pagal Gilbert-Varshamov lygtį tai beveik optimalus variantas, išskyrus atvejus, kai algoritmas gali atskleisti s reikšmę. Darant tokią prielaidą, galimybė sugeneruoti tinkamą μ artima 6.76 yra $1.03*10^{-3}$. Ir tai irgi neženkliai padidina įsilaužimo efektyvumą, palyginus su pagrindine taktika.

Dar reikėtų įrodyti, kad nė vienas iš sąžiningų sistemos vartotojų nebus atmestas identifikavimo metu. Kaip matome iš schemos tai nėra akivaizdu, kai asmuo B atsiunčia asmeniui A b = 2. Todėl reikėtų atlikti tam tikrus tyrinėjimus, kad įsitikinti, kad taip nebus. Tarkime mes turime atsitiktinį n bitų ilgio žodį t. Tada:

$$m = w(t), m' = w(t \oplus s)$$

mus domina, ne konkrečios m ir m‘ reikšmės, o $w(m - m')$. Jei tarsime, kad

$$T = \sum_{s(i)=1} t(i)$$

tada matosi, kad $w(m - m')$ yra lygus $w(2T - p)$. Dabar mes turime išanalizuoti atsitiktinio kintamojo $w(2T - p)$ reikšmę. Galime paskaičiuoti v matematinę viltį:

$$v = \sqrt{\frac{2p}{\pi}} \approx 0.798\sqrt{p}$$

O jo standartinis nuokrypis,

$$\sigma = \sqrt{p - \frac{2p}{\pi}} \approx 0.603\sqrt{p}$$

Tada, reikšmės $w(n_j - m_j)$, kurios buvo paskaičiuotos 6 žingsnyje, yra lygios $w(2T - p)$ reikšmei, priklausomai nuo to, koks atsitiktinis vektorius t buvo sugeneruotas atsitiktinių reikšmių generatoriumi - $t = y \oplus z_j$. Dabar, pasinaudojant (vidurio reikšmės) teorema, parodysime, kad suskaičiuota μ reikšmė nedaug skiriasi nuo matematinės vilties v. Aišku, tai nėra visiškai teisinga, nes w_j yra fiksuoti, todėl mes neturime nepriklausomų kintamųjų reikšmių. Bet w_j reikšmes yra pasirinktos atsitiktinai. Tada turime:

$$P\{|\mu - v| \geq r\sqrt{p}\} \leq \int_{\frac{r\sqrt{pq}}{\sigma}}^{\infty} e^{-x^2/2}dx \leq \frac{2\sigma}{r}\sqrt{\frac{2}{\pi pq}}e^{-r^2pq/2\sigma^2} \approx \frac{0.962}{r\sqrt{q}}(0.253)^{r^2q}$$

Įstačius r = 0.272, gauname, kad tikimybė gauti μ, kuris viršys $1.07\sqrt{p}$, bus $6.96*10^{-7}$, kai q = 128. Netgi jei etapų skaičius bus 60, tada, teoriškai, vidutinių reikšmių paskaičiavimas vyks 20 kartų. Bendra tikimybė atmesti vartotojo identifikaciją yra $1.4*10^{-5}$. Bet tai lengva įveikti, leidžiant atlikti identifikavimą dar kartą.

Vienas iš didžiausių šios schemos trūkumų - tai atminties kiekis reikalingas šios schemos darbui. Matricos H ir žodžių w_j saugojimui atmintyje vienu atvejų, kai n = 512, reikia 150 kbitų, o saugesniu atveju, kai n = 1024, reikia 630 kbitų. Tačiau schema naudoja paprastas operacijas su bitais, todėl ji gali būti efektyviai realizuota aparatūriškai. Aišku, schemos komunikavimo protokolas yra sudėtingas, bet ne ypač daug skiriasi nuo Fiat-Shamir schemos komunikavimo protokolo. Kadangi n_j reikšmės nuo 256 skiriasi ne daugiau kaip 128-ais, to atveju kai n = 512, tai vieną n_j skaičių galime užkoduoti 8-ais bitais. Tada 4-ių parašų sekai perduoti reikės 4000 bitų ilgio eilutės.

Schemos saugumui padidinti, galima padidinti q ir r reikšmes. Atsitiktinis w_j žodžių generavimas, taip pat teigiamai atsilieps schemos saugumui. Iš kitos pusės padidėja persiunčiamų duomenų kiekis. Be to, sistemos didesniam efektyvumui ir saugumui užtikrinti galima lygiagrečiai atlikti kelis etapus, taip sumažinant etapų skaičių ir laiką, per kurį įsilaužėlis turį įveikti sistemos apsaugą.

Sumažinus p reikšmę, schema gali tapti visiškai nesaugi, nes žodžiai su mažais svoriais gali būti greitai surasti. Todėl schemos autoriai siūlo nauduoti p reikšmes ne mažesnes už 20.

Patobulinta alternatyvi identifikavimo schema

1996 m. aukščiau minėtos schemos autorius, Jacques Stern, pasiūlė naują patobulintą identifikavimo schemą. Schema, kaip ir jos pradmininkai, naudoja (n-k x n) matricą H virš kūno GF(2). Matrica yra bendra visiems schemos dalyviams ir sugeneruota atsitiktiniai. Idealiai, kiekvienas matricos elemento reikšmės iš aibės {0, 1}pasirodymas, turi būti nepriklausomas įvykis. H yra kontrolinė klaidas taisančio kodo matrica su gera klaidų taisymo galia.

Kiekvieno vartotojo registracijos metu, jam išduodamas unikalus raktas s. Raktas yra n bitų ilgio žodis su žinomu svoriu p. Svoris p irgi yra sistemos parametras. Norint identifikuotis vartotojas turi suskaičiuoti savo viešą identifikatorių, taip:

$$i = H \cdot s$$

Raktą galima patalpinti į kokį nors atvirą katalogą arba kitokią lengvai prieinamą vietą. Tam, kad identifikacijos metu bet kokia protokolo šalis galėtų laisvai gauti jai reikiamą informaciją – viešą identifikatorių.

Schema, kaip ir ankščiau siūlyta schema, naudoja parašus. Parašas - tai būdas laikinai paslėpti statinę bitų seką u. Aiškiai matosi, kad šį procesą sudaro dvi dalys – kodavimo ir dekodavimo. Kodavimo dalyje: vektoriaus u reikšmę praleidžia per visiems prieinamą hash-funkciją ir grąžina rezultatą <u> . Dekodavimo dalyje: vartotojas paskelbia u reikšmę, tada kiekvienas iš schemos dalyvių gali lengvai patikrinti <u> reikšmę, pasinaudojus atvira hash-funkcija. Teoriškai, kaip hash-funkciją mes galime nauduoti bet kokią vienakryptę funkciją, kuri užtikrina, kad negalima rasti tokių x ir y žodžių, kad jų hash-funkcijos reikšmės bus lygios <x> = <y>. Tačiau praktiškiau pasinaudoti standartinėmis hash-funkcijomis – tokiomis kaip SHA arba MD5. Pasinaudoję hash-funkcija mes užtikriname kad niekas negalės paskelbti tokios reikšmės y, kad <u> = <y>. Bet hash-funkcijos naudojimas neužtikrina informacijos slaptumo, t.y. mes nesame tikri, kad iš gautos hash-reikšmės negalima atstatyti jokio pradinės informacijos fragmento. Kad užtikrinti ir antrą prielaidą, mums reikia pasinaudoti randomizuota hash-funkcija. Tokios funkcijos yra aprašytos įvairių autorių ir puikiai realizuotos [BER93] straipsnyje. Kitas būdas pridėti atsitiktinumo pasirinktai hash-funkcijai: tai vietoj funkcijos argumento u, ieškoti reikšmę $p \,||\, p \oplus u$, kur u yra mūsų turimas slaptas žodis , p – atsitiktinis n bitų ilgio vektorius, o || yra suliejimo (concatenation) operacija. Šią technologiją pavadinsime – atsitiktiniu hešavimu.

Dabar aprašysime identifikacijos protokolą, kuris leis vartotoją A (įrodinėtojas) įrodyti savo tapatybę vartotojui B (tikrintojas). Protokolą sudaro r raundų, kiekvieną iš kurių sudaro sekantis žingsniai:

- A pasirenka atsitiktinį n bitų ilgio žodį y ir atsitiktinę perstatą σ. σ yra atsitiktinis $\{z\}^n$ vektorius, kur $z \in \{1..n\}$ ir $z_i \neq z_j$, kur $1 \leq i \leq n$, $1 \leq j \leq n$, $i \neq j$. Ir siunčia B sekančius parašus (formulėse σ, tai perstato vektorius paverstas {0, 1} seka. O funkcija y.σ – tai y reikšmė pritaikius perstatą σ):

$$c_1 = \langle \sigma \parallel H(y) \rangle$$

$$c_2 = \langle y.\sigma \rangle$$

$$c_3 = \langle (y \oplus s).\sigma \rangle$$

- Asmuo B siunčia atsitiktinį skaičių $b \in \{0,1,2\}$.
- Jei b = 0, tada A siunčia y ir σ. Jei b = 1, tada A siunčia $y \oplus s$ ir σ. Jei b = 2, tada A atskleidžia y.σ ir s.σ.
- Jei b = 0, tai asmuo B tikrina reikšmės c_1 ir c_2, kurios buvo paskaičiuotos pirmame žingsnyje. B atlieka sekančius skaičiavimus: tarkime kad 3 žingsnyje mes gavome $\tilde{y}$ ir $\tilde{\sigma}$ reikšmes, tada mes skaičiuojame:

$$c_1 = \left\langle \tilde{\sigma} \parallel H(\tilde{y}) \right\rangle$$

$$c_2 = \left\langle \tilde{y} . \tilde{\sigma} \right\rangle$$

Jei b = 1, B patikrina reikšmių c_1 ir c_3 korektiškumą: tarkime kad 3 žingsnyje mes gavome $\tilde{y}$ ir $\tilde{\sigma}$ reikšmes. Iš reiškinio $H(y) = H(y \oplus s) \oplus i$ išplaukia sekantis identiškumo patikrinimai:

$$c_1 = \left\langle \tilde{\sigma} \parallel H(\tilde{y}) \oplus i \right\rangle \text{ ir}$$

$$c_3 = \left\langle \tilde{y} . \tilde{\sigma} \right\rangle$$

Jei b = 3, asmuo B patikrina vektoriaus s.σ svorį w(s.σ) ir įsitikina, kad c_2 ir c_3 paskaičiuoti teisingai: tarkime kad 3 žingsnyje mes gavome $\tilde{y}$ ir $\tilde{s}$ reikšmes. Pirma pasitikrinama ar geras vektoriaus $\tilde{s}$ svoris: $w(\tilde{s}) = p$. Toliau atlieka sekančius skaičiavimus:

$$c_2 = \left\langle \tilde{y} \right\rangle$$

$$c_3 = \left\langle \tilde{y} \oplus \tilde{s} \right\rangle$$

Aišku, jei mes pasinaudosime atsitiktiniu hešavimu: kiekvienas parašas c_j, j = 1,2,3 naudoja atsitiktinį vektorių p_j, j = 1,2,3. Tada reikės šiek tiek pakeisti trečią algoritmo žingsnį:

- Jei b = 0, tada A siunčia y, σ ir p_1, p_2. Jei b = 1, tada A siunčia $y \oplus s$, σ ir p_1, p_3. Jei b = 2, tada A atskleidžia y.σ, s.σ p_2, p_3.

Taip pat atitinkamai reikės pakeisti ir ketvirtą žingsnį, kad identifikavimas būtų sėkmingas.

Kaip ir ankstesnio schemos varianto, Kadangi schemos saugumas pagristas funkcijos

$$s \to H(s)$$

saugumu, kai jos argumentai yra „geri“ slapti raktai.

Kalbant apie schemos efektyvumą, reikia pabrėžti sekančius aspektus:

- Gali pasirodyti, kad schemos darbui reikės daug darbinės atminties, tačiau tai nėra visiškai taip. Iš vienos pusės visos operacijos yra labai paprastos, todėl gali būti efektyviai realizuotos aparatūriškai. Iš kitos pusės, jei bus tik(ir dalinė) programinė schemos realizacija, tai nebūtina saugoti visą H matrica, pakanka išsaugoti tik jos dalis, o kitas matricos reikšmes papildyti atsitiktiniu reikšmių generatoriumi.
- Dviejų asmenų komunikavimo protokolas, palyginus su Fiat-Shamir schema, taip pat nėra labai sudėtingas. Ir persiunčiamų bitų skaičius per vieną raundą nedaug viršija 1000 bitų, kai n = 512.
- Schemos saugumas gali būti padidintas atitinkamai padidinus schemos parametrus n, m ir p. Reikšmės n = 512, m = 256 ir p = 56 yra minimalios galimos schemos parametrų reikšmės.
- Atsižvelgiant į tai, kad vieno raundo įveikimo tikimybė yra 2/3, siūlomas raundų skaičius turi būti nemažesnis už 35 raundus.
- Pagal [FiS87] straipsnį, schemą galima paversti į parašo schemą (Pagal pirmą žingsnį paruošti reikšmes c_1^j, c_2^j, c_3^j, kur j = 1, ... j ir išsiusti jos kartu su pranešimo m tekstu).

Schemų saugumas

Kadangi kriptografinės schemos naudojamos sistemos, verslo, duomenų ir kitokių su žmogaus veikla susijusiu procesų ir objektų saugumui užtikrinti, todėl schemos saugumas yra

vienas iš svarbiausių schemų atributų. Dabar apibendrinsime nagrinėjamų schemų saugumo aspektus.

Schemų saugumas pagrįstas funkcijos

$$s \to H(s)$$

„saugumu“, kai jos argumentai yra „geri“ slapti raktai. Visas atakas į kriptoschemas, pagrįstas kodų teorija, galima suskirstyti į tokias grupes:

- Dekodavimo atakos - bandymai ištaisyti atsitiktinio klaidas taisančio kodo klaidas, arba kitaip tariant, šios atakos metu stengiamasi surasti tokį vektorių x, kad jo sindromas atitiktų vartotojo slapto rakto sindromui.
- Struktūrinės atakos: atakos, sutelktos į kodo struktūros atskleidimą.

Atsižvelgiant į [CHA95] pateiksime skaitinius nagrinėjamų schemų parametrų įvertinimus:

- n = 512, m = 256, p = 56
- n = 768, m = 384, p = 84
- n = 1024, m = 512, p = 110

Tačiau reikia pabrėžti, kad parametrai pasirinkti atsižvelgiant tik į tiesioginio perrinkimo ataką, kai slaptą reikšmę s bando nustatyti iš žinomos kontrolinės matricos H ir sindromo i.

Čia pateikti parametrai atitinka tiesinius kodus su kodo koeficientu, lygiu ½. [CHA95] šaltinio duomenimis, pagal žinomus algoritmus galimybė, kad paslaptis bus atskleista, lygi $O(\tau^k)$, kur aukščiau nurodytom reikšmėm $\tau = 1{,}18$. Tada parinktom minimaliom reikšmėm τ^k bus apytikriai lygus 2^{70}. Identifikuotis sistemoje nežinant slapto rakto arba neturint galimybės atlikti dekodavimo arba struktūros atakas, galima sekančiais būdais:

- Paruošti y ir σ reikšmes ir pakeisti nežinomą s reikšmę vektoriumi t, kurio svoris yra p.Tada įsibrovėlis tikisi, kad atsiųsta b reikšmė bus lygi 0 arba 2. Jei b lygi 0, įsilaužėlis atskleidžia y ir σ reikšmes, o jei 2 – tai bus atskleistos y. σ ir t. σ reikšmės. Tai reiškia kad su tikimybe (2/3) įsibrovėlis gali įveikti vieną raundą ir su tikimybe $(2/3)^r$ visus schemos raundus.
- Panašią strategiją galima taikyti ir $y \oplus s$ reikšmei. Tada įsibrovėlis tikisi gauti b reikšmes 1 ir 2 atitinkamai.
- Galima paruošti σ, y ir $y \oplus t$, kur t yra kažkoks žodis nelygus s ir H(t) = i ir w(t) ≠ p. Tada bus tikimasi gauti 0 ir 1 reikšmes.

Aišku, galima bandyti keisti strategijas priklausomai nuo kiekvieno raundo, tačiau tikimybė įveikti schemą tokiu būdu vis tiek lieka ta pati. Tokio tipo atakos vadinamos – schemos protokolo atakomis.

T. Johansson ir F. Jonsson savo darbe [JJ02] išanalizavę populiariausius algoritmus, kurie priskiriami dekodavimų atakų grupei, ir parodė, kad iš visų žinomų dekodavimo atakų efektyviausia Canteaut ir Chabaud [CC98] pasiūlyta ataka. Savo darbe jie pasiūlė dar vieną žodžių su mažais svoriais paieškos algoritmą. Tačiau dabar, modifikuotas Canteaut ir Chabaud algoritmas vis tiek lieka efektyviausiu dekodavimo tipo algoritmu.

Iš atakų, priskiriamų kodo struktūros atskleidimo grupei, efektyviausia yra [Sen00] pasiūlyta ataka, tačiau jos efektyvumas mažesnis už [CC98] pasiūlytą ataką, todėl pagrindinis akcentas vertinant schemas sutelkiamas į dekodavimo atakas.

Dabar pateiksime dekodavimo atakai įvykdyti reikiamo operacijų skaičiaus įvertinimus. Iškart reikia pastebėti, kad [Cha95] tai ne algoritmas, o straipsnis, kuriame pateikti kriptografinių schemų įveikimų įvertinimai.

	Kriptoschema	
Dekodavimo algoritmas	McEliece	Stern
[Ste89]	$2^{69.9}$	$2^{73.5}$
[CC98]	$2^{64.2}$	$2^{69.9}$
[Cha95]	$2^{68.1}$	$2^{72.2}$

Lentelė 4. Schemų įveikimo faktorių lentelė.

Iš lentelės 3 duomenų matome, kad Canteaut ir Chabaud algoritmas yra vienas iš efektyviausių dabar egzistuojančių dekodavimo algoritmų. Būtent šituo algoritmu ir remsimės savo darbe.

Taip pat pagal lentelės 2 duomenis, pateiktų algoritmų, kaip McEliece ir Stern atveju, rezultatai pagerėjo maždaug 2^5 kartų. Aišku, tai yra rimtas žingsnis schemos įveikimo link, tačiau 2^{64} McEliece ir 2^{69} Stern schemos darbo faktorius vis tiek yra didelis.

Apibendrinus schemų saugumą, būtų labai naudinga pakalbėti apie schemų realizaciją ir jų greitaveiką.

Schemų palyginimas ir realizacija

Kaip matome, nagrinėjamos schemos yra tarpusavyje labai panašios: naudoja klaidas taisančius kodus, neatskleidžia informacijos apie identifikuojamą asmenį (zero-knowledge), naudoja kontrolinę H kodo matricą, kiekvieno vartotojo registravimo metu jam suteikiamas unikalus raktas su žinomu svoriu p, Taip pat schemos turi panašų pranešimų atskleidimo mechanizmą. Tačiau pirma schema turį specifinį žodžių w_1 ... w_q rinkinį, kuris naudojamas kaip atsitiktinių skaičių generatorius. Tokio tipo žodžių, beje kaip ir tokio tipo atsitiktinių žodžių generatoriaus, nėra. Tačiau panašų principą galima rasti šių schemų „protėvyje“ A.

Fiat ir A.Shamir darbe [FiS87], kur naudojamos v_j reikšmės, kurios nėra vektoriai, tačiau skirtos informacijai paslėpti. Antra schema naudoja atsitiktinę perstatą – σ: atsitiktinis $\{z\}^n$ vektorius, kur $z \in \{1..n\}$ ir $z_i \neq z_j$, kur $1 \leq i \leq n$, $1 \leq j \leq n$, $i \neq j$.

Reikėtų pastebėti, kad abi schemos nauduoja ne generuojančią kodo matricą G, kaip McEliece kriptoschema [Mce78], o kontrolinę kodo matricą H, kurią pirma kartą pasiūlė nauduoti H. Niederreiter savo darbe [Nie86] 1986 m. Vėliau Robert H. Deng Yuan Xing Li ir Xin Mei Wang įrodė, kad saugumo aspektu, su tais pačiais parametrais, abi šios idėjos yra ekvivalenčios. [DW94]. Tačiau A. Canteaut ir F. Chabaud savo darbe [CC98] parodė, kad Niederreiter pasiūlyta kriptoschema turi kelis privalumus, tokius kaip slapto rakto dydis ir operacijų skaičius užkoduojant duomenis.

Kaip buvo minėta aukščiau, abu algoritmai naudoja atsitiktines kontrolines matricas, todėl svarbu schemų realizavimo etape atkreipti dėmesį į atsitiktinių matricų generavimo algoritmus.

Atsitiktinių matricų generavimas

Kalbant apie atsitiktinių matricų, turinčių atvirkštinę matricą, generavimą, į galvą iškart ateina vienas iš paprasčiausių būdų sugeneruoti atsitiktinę matricą – tiesiog paimti ir atsitiktinai sugeneruoti matricą pseudo atsitiktinių skaičių generatoriaus pagalba. T.y. kiekvieno matricos elemento generavimą galima traktuoti kaip nepriklausomą įvykį, todėl atsitiktinės matricos generavimas susives į n^2, kur n yra kvadratinės matricos eilučių/stulpelių skaičius, atsitiktinių skaičių generavimą. Taigi paprasčiausias algoritmas gelėtų atrodyti taip:

1. Sugeneruoti atsitiktinę matricą M.
2. Paskaičiuoti matricos M determinantą, ir jei jis nelygus nuliui, tai mes gavome reikalingą matricą. Kitaip pereiname prie pirmo žingsnio.

Tačiau toks algoritmas nėra deterministinis, nes mes nežinome po kiek žingsnių mes gausime reikalingą matricą. Nors Yuang-Xian Li, Da-Xing Li, Chuan-Kun Wu savo darbe [YDC92] įrodė, kad sėkmingo atsitiktinės matricos generavimo virš kūno GF(2) tikimybė didesnė už 0,25 ir teigė kad mums prireiks maždaug 4 kartus pergeneruoti matricą.

Tarkime, kad $P(\omega)$ - tai tikimybė, kad atsitiktinę matricą pavyko sugeneruoti ne daugiau kaip po k kartų, kur ω - tai sėkmingo atsitiktinės matricos sugeneravimo įvykis. Tarkime, kad tikimybė sugeneruoti atsitiktinę matricą yra p = 0,25, o priešingo įvykio tikimybė p‘ = 0,75. Tada:

$$P(\omega) = \sum_{i=0}^{\omega-1} p * p'^{i}$$

Gauname, kad $P(\omega=4) \approx 0.68$. Tikrai su didele tikimybe mums pavyks sugeneruoti tinkamą atsitiktinę matricą, tačiau mes nesame tikri, kad mums tikrai pasiseks ir mes sugeneruosim matricą po 4, 5, 50 iteracijos žingsnio. Dar didelę įtaką generavimui turi (pseudo) atsitiktinių skaičių generatorius. Taigi, pasiūlytas algoritmas yra tikrai paprastas, tačiau nedeterministinis.

Norint pasinauduoti kitokiu deterministiniu atsitiktinių matricų generavimo algoritmu, reikės įdėti daugiau pastangų į algoritmo realizaciją, tačiau tai turi savų privalumų. Vienas iš tokių algoritmų buvo pasiūlytas D. Randall straipsnyje [Ran93]. Šį algoritmą panagrinėkime atidžiau. Be to jis naudojamas ir nagrinėjamų schemų realizacijoje.

Algoritmas veikia rekursiviai, kiekviename žingsnyje taisant reikiamą eilutę/stulpelį kiekvienoje iš matricų. Rekursijos metodu surandame mažesnę matricą, kuri tenkina apibrėžtas savybes, ir grįžtant padidiname matricos dimensiją.

Pagrindinė algoritmo idėja - tai sugeneruoti dvi matricas, kurių sandauga yra ieškoma matrica. Pirma matrica yra ieškoma matrica, o antra - tiesinė pirmos matricos transformacija. Kaip sugeneruoti reikiamą matricą? Kad atsakyti į šį klausimą apibrėžkime kai kuriuos algoritmų parametrus. Įsivaizduokime vektorių e_1 su viena nenuline koordinate - <1, 0, 0, ..., 0> ir jį atitinkantį minorą (1,1). Tai mes galime apibendrinti. Tarkime, turime vektorių e_r <0, 0, .., 1, ..., 0>, kur $w(e_r) = 1$, w – vektoriaus svoris ir r yra vienetuko pozicijos numeris, ir (1, r) atitinkamas minoras. Tarkime, turime atsitiktinį vektorių $v \in \Gamma \setminus \{0\}$, kur r yra r-toji pirmojo vienetuko pozicija atsitiktiniame vektoriuje v. Kol kas tai gali atrodyti painu dėl dvigubo r apibrėžimo, tačiau, kai pristatysime algoritmą, viskas paaiškės. Kaip buvo kalbėta aukščiau, ieškomą matricą sudaro atsitiktinai generuojamų matricų sandauga $A*T$. Sekančiu žingsniu sugeneruojame atsitiktinį vektorių v, apskaičiuojame vektoriaus v pirmo vienetuko poziciją, sugeneruojame atitinkamą e_r vektorių su vienetu r-joje pozicijoje ir kiekviename žingsnyje matricos A viršutinei eilutei priskirsime e_r reikšmę, o matricai T atsitiktinio vektoriaus v reikšmę.

NonSing(n):
Begin
  Gen(A, T, n);
  Return(A*T);
End.

Gen(A, T, n):
Begin
  Jei f =1 tada
    Tegul $A_{1,1} = 1$;
    Pasirenkame $T_{1,1} \in_u \Gamma \setminus \{0\}$;
  Kitaip
    Pasirenkame $v \in_u \Gamma^n \setminus \{0^n\}$ toks kad $v \neq 0^n$;
    Tegul *r* bus pirma nenulinė vektoriaus *v* koordinatė;
    // Priskirsim reikšmes matricai A
      Tegul pirma matricos *A* eilutė lygi e_r.
    // Priskirkim rekšmes matricai T
      Tegul r-toji matricos eilute ligi *v*;
      Kiekvienam $1 \leq i \leq n$, kur $i \neq r$, $T_{i,r} = 0$;
    Gen(A[1,r], T[r,r], n-1);
End;

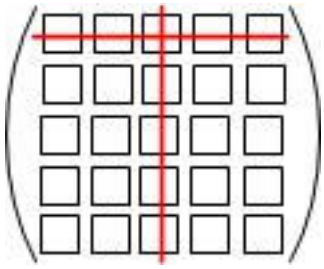
Kadangi šis algoritmas naudoja rekursiją, matricos generavimas gali būti neefektyvus resursų atžvilgių, nes rekursija gali reikalauti didelio atminties kiekio. Taigi, pateiksiu patobulintą generavimo algoritmą, be rekursijos naudojimo:

Tegul *i* einamoji matricos A eilutė, t.y. algoritmo darbo pradžioje ji ligi 1 ir didinama su kiekviena iteracija. Trumpai apibūdinkime mūsų algoritmą:

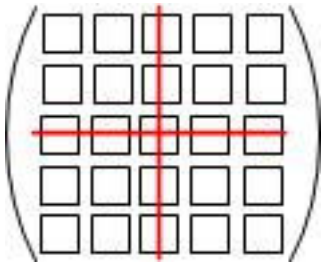
1. Jei *i = n-1*, tada i matricos *A[n, ‚paskutinis neišbrauktas stulpelis‘] = 1* ir T*[‚paskutinis neišbrauktas stulpelis‘,‚paskutinis neišbrauktas stulpelis‘] = 1*. Baigti darbą.
2. Generuojame atsitiktinį vektorių *v* ilgio *n-i*.
3. Surandame pirmą nenulinę koordinatę *r*.
4. Į pirmąją neišbrauktą matricos *A* eilutę *i* įrašome e_r vektorių.
5. Į matricos *T* r-tąją eilutę įrašome vektorių *v*.
6. Uždedame požymį, kad eilutė/stulpelis yra išbraukta(s).
7. Padidiname *i* vienetu ir einame prie pirmo punkto.

Tarkime iš pradžių susigeneravo vektorius v = <0, 0, 1, 1, 0>. Tada r = 3 ir į A matricos pirmą eilutę įrašysime reikšmę <0, 0, 1, 0, 0> o į T matricos r-tąją eilutę – v vektoriaus

reikšmę ir išbrauksime atitinkamas eilutes, t.y. tolimesnius veiksmus atliksime su matricos minorais:

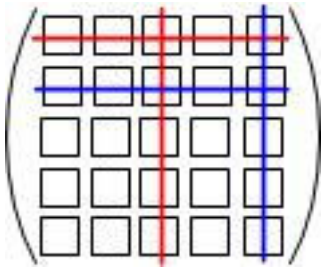


Pav. 2 Matricos A minoras

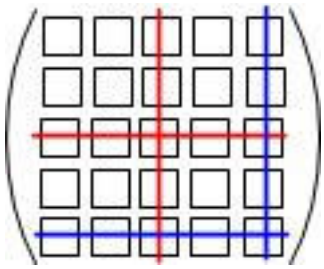


Pav. 3 Matricos T minoras

Tada pereiname prie kitos eilutės, t.y. padidiname i reikšmę 1. Vėl generuojame vektorių v = <0, 0, 0, 1>, surandame r = 4 ir išbraukiame atitinkamas eilutes ir stulpelius. Gauname tokius minorus:



Pav.4 Matricos A minoras



Pav.5. Matricos T minoras

Ir taip toliau, kol kiekvienos kvadratinės matricos minoro dimensija netaps lygi 1. Tada likusiems minorams priskiriame reikšmę 1 ir užbaigiame matricų generavimo procesą.

Kad gauti ieškomą matricą liko tik sudauginti sugeneruotas matricas A ir T.

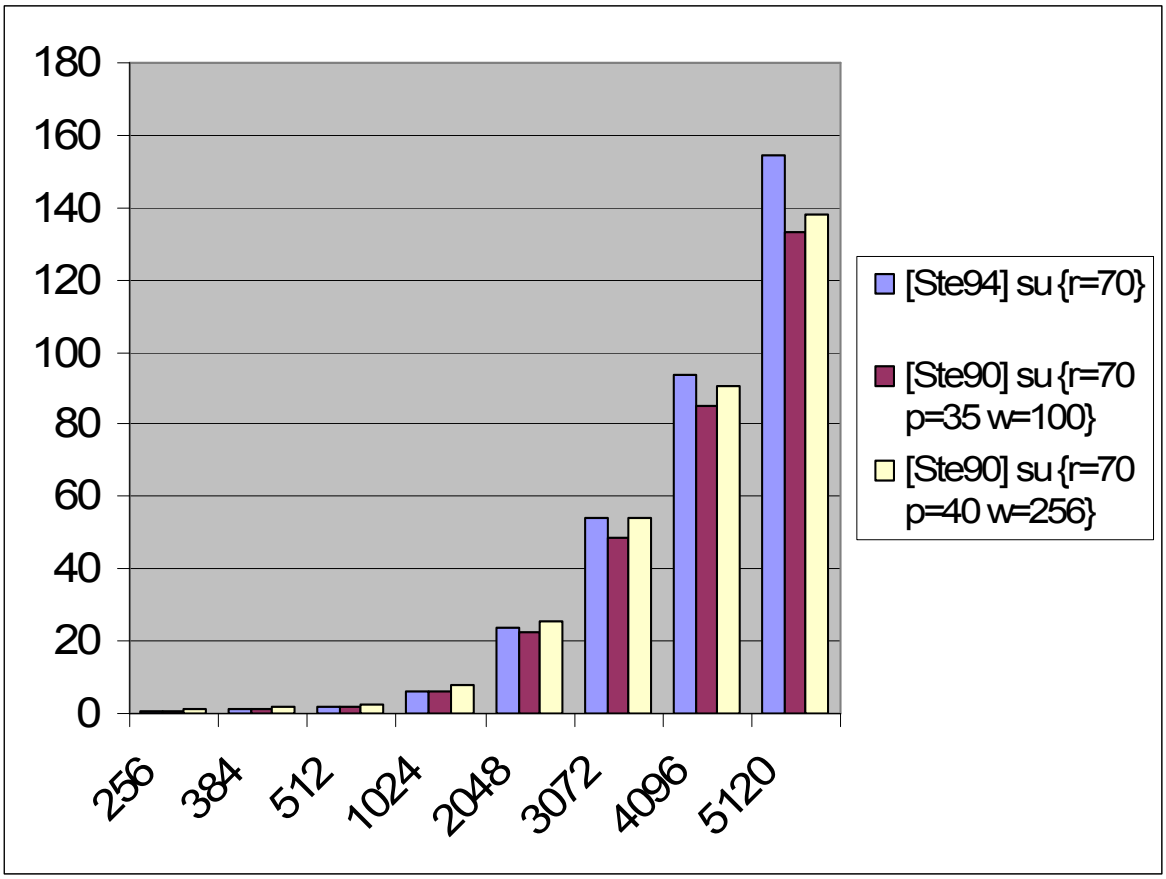
Autoriai įrodė, kad algoritmo sudėtingumas yra $M(n)+O(n^2)$, kur $M(n)$ laikas reikalingas nxn matricų daugybai.

Schemų greitaveika

Sukūrus naują algoritmą arba teoriją visada įdomu, kaip tai veiks praktiškai. Todėl šiame skyrelyje mes ištirsime nagrinėjamų schemų greitaveiką.

Viena iš svarbiausių kriptografinių savybių yra schemos efektyvumas, t. y. kaip greitai schema atpažįsta/užkoduoja/dekoduoja vartotojo informaciją. Mūsų atveju labai svarbus laikas, per kurį schema gali identifikuoti arba atmesti pažeidėją. Aišku, schemos greitaveikos tyrimas turi vykti su prasmingais schemos parametrais, todėl mes pasinaudosim [Ste90] ir [Ste94] pasiūlytais schemų parametrais.

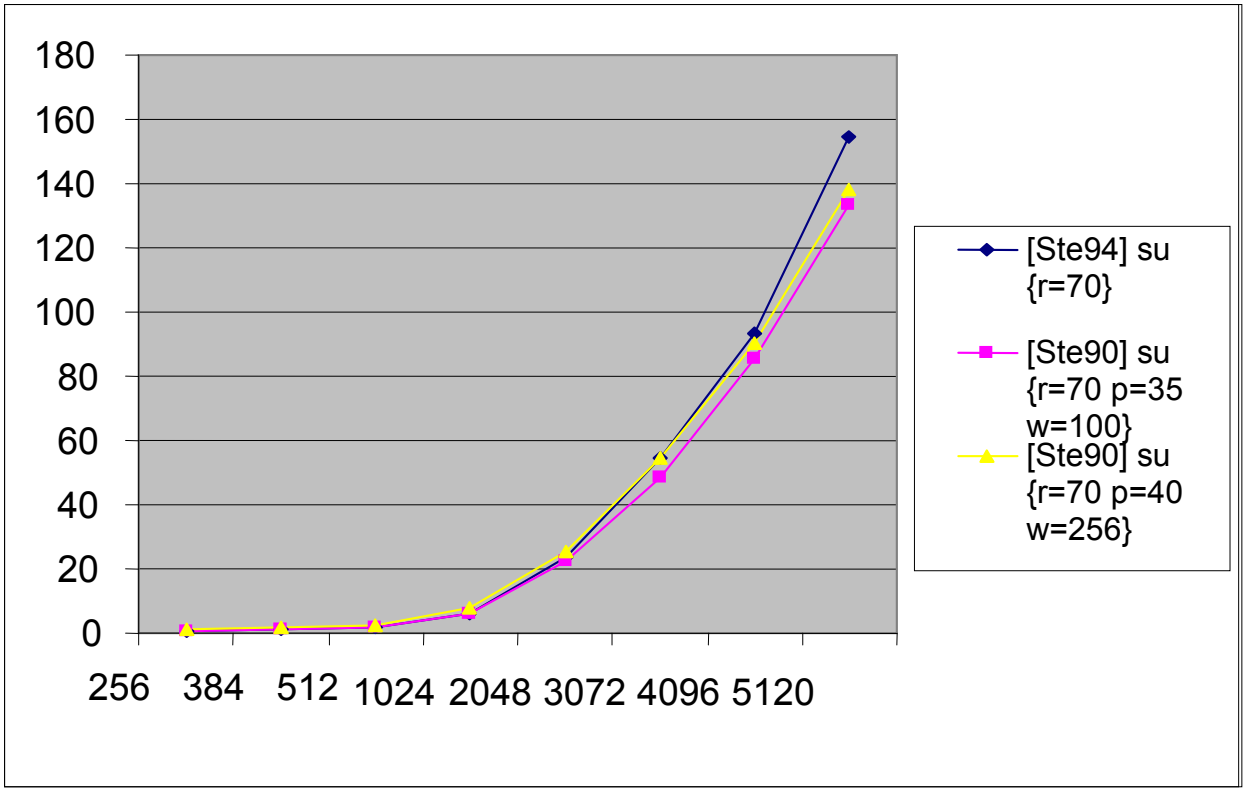
Eksperimento metu patikrinsime schemų efektyvumą naudojant 512x256, 768x384, 1024x512, 2048x1024, 4096x2048, 6144x3072, 8192x4096 ir 10240x5120 dydžio matricas. Tačiau pirma schema turi dar papildomą parametrą – žodžių aibę, kuri naudojama atsitiktinių žodžių generatoriaus. Šis parametras labai svarbus [Ste90] schemos saugumui. Todėl į mūsų tyrimą mes įtraukėme ir šios aibės galią. Taip pat eksperimente dalyvaujančiose schemose naudosime 70 etapų, tam, kad objektyviau įvertinti schemų identifikavimo laiką. Priklausomai nuo pasirinktos užklausos priklauso schemos naudojamų operacijų skaičius. O turint omeny, kad atsakymo pasirinkimas yra nepriklausomas įvykis, galima teigti, kad atlikus daug iteracijų gausime apytikslį algoritmo greičio įvertinimą.



Pav. 6 Laiko sąnaudos [Ste90] ir [Ste94] schemose.

Pagal ašį x yra išvardintos klaidas taisančio kodo kontrolinės matricos dydžiai. Kadangi mes naudojame kodus, kurių efektyvumas yra ½. Todėl nėra būtina vardinti abi matricos dimensijas. T.y. jeigu mes turime pagal x ašį reikšmę 512, tai reiškia, kad kontrolinės matricos dydis bus 1024x512 ir t.t. Pagal y ašį nurodytas vartotojo identifikavimo greitis sekundėm.

Tačiau šio tipo grafike nesimato, kad antras algoritmas su mažomis reikšmėmis – matricomis iki 4096x2048 - lenktų pirmą algoritmą. Tačiau, didinant matricų parametrus, didėja ir reikiamų operacijų skaičius, ir antrasis algoritmas parodo blogesnius rezultatus. Tokį atvejį pademonstruosime dar vienu grafiku ir pateiksime palyginimų rezultatų lentelę.



Pav. 7 Laiko sąnaudos [Ste90] ir [Ste94] schemose.

Šitas grafikas taip pat atspindi vartotojo identifikavimo greičius, tačiau čia mes galime matyti, koks algoritmas su kokiais parametrais pirmauja įvairiuose etapuose. Taip pat pagal x ašį išvardintos kodo kontrolinių matricų dydžiai. O pagal y ašį atvaizduotas vartotojo identifikavimo laikas.

Matricos dydis	[Ste94] (r=70, p=40)	[Ste90](w=100,r=70,p=35)	[Ste90](w=256,r=70,p=40)
512x256	0,484	0,625	1,016
768x384	0,953	1,296	1,75
1024x512	1,625	1,89	2,61
2048x1024	6,031	6,219	7,625
4098x2048	23,5	22,437	25,281
6144x3072	54,323	48,5	54,313
8192x4096	93,609	85,359	90,516
10240x5120	154,453	133,343	138,25

Lentelė 5 Laiko sąnaudos [Ste90] ir [Ste94] schemose.

Pastaba: R išreiškia etapų skaičių, P – vartotojo identifikatoriaus Hemmingo svorį,o W – schemoje [Ste90] naudojamų vektorių aibės galią.

Patobulintos identifikavimo schemos

Visos pateiktos ir nagrinėjamos [Ste90] ir [Ste94] straipsnių schemos sudarytos virš $GF(2^n)$ kūnų. Logiška būtų patyrinėti kaip pasikeis schemų saugumas jei schemos bus sudarytos virš kitų $GF(q^n)$ kūnų. Todėl kitų darbo skyrių, kaip ir viso darbo, pagrindinis akcentas teks tokių schemų sudarymui ir tyrinėjimui.

Aišku, schemos nėra skirtos naudojimui virš kitokių kūnų, todėl mums reikės atitinkamai pakeisti pasiūlytas identifikavimo schemas, kad jos atitiktų mūsų kriterijų.

Pakeistos identifikavimo schemos

Modernizuosim nagrinėjamas schemas taip kad jos veiktų virš $GF(q^n)$ kūnų. Aišku, toks modernizavimas šiek tiek atsilieps schemų identifikavimo greičiui, nes operacijoms virš $GF(2^n)$ kūnų galima nauduoti postūmio ir logines operacijas, tuo tarpų modernizuotoms schemoms tai netiks. Kaip matome iš trečios lentelės duomenų, mūsų realizacijoje modulio dydis (jeigu modulio dydis neviršija 4 baitų) beveik neįtakoja schemos vykdymo proceso:

	q	2	3	5	11	23	419	1357	5501	
M	**r**									**Vidutinis laikas**
64	**70**	0,532	0,62	0,62	0,594	0,62	0,62	0,62	0,62	**0,60575**
128	**70**	2,186	2,314	2,188	2,31	2,344	2,376	2,312	2,344	**2,29675**
256	**70**	8,44	8,966	9,282	9,44	9,156	9,282	9,596	9,812	**9,24675**
512	**70**	32,784	35,812	37,22	37,938	37,594	37,406	37,908	36,47	**36,6415**
1024	**70**	131,904	139,374	148,844	145,188	141,376	143,126	153,688	153,688	**144,6485**
2048	**70**	541,626	571,028	602,844	576,94	574,218	598,624	587,718	580,44	**579,1798**
4096	**70**	2188,688	2222,97	2337,25	2393,596	2320,188	2298,564	2399,028	2411,592	**2321,485**

Lentelė 6 Modifikuotų schemų greitaveikos priklausomybės nuo modulio dydžio analizės lentelė[1].

Taigi, pagrindiniai schemų parametrai dabar bus tokie: **m** = n-k – nusako H matricos dydį. Kadangi mes naudojame matricas, kurių santykis tarp n ir m yra ½, gauname, kad mūsų matricos dydis yra m x 2m. **p** – slapto vartotojo rakto **s** svoris. Originaliose schemose tai buvo vienetų skaičius pasirinktame vektoriuje, kas yra lygu ekvivalentų vektoriaus Hammingo svoriui. **r** – schemos etapų skaičius. Kaip buvo minėta aukščiau, schemos saugumas ir greitaveika tiesiogiai priklauso nuo schemos etapų skaičiaus, todėl šis parametras irgi labai

[1] Lentelėje 6 q – tai GF(q) parametras, m- klaidas taisančio kodo kontrolinės matricos eilučių skaičius, r – schemos etapų skaičius, kadangi anksčiau visos schemų realizacijos naudojo 70 etapų, todėl ir patobulinto schemos naudos 70 etapų. Be to 70 etapų tai yra rekomenduotinas schemų saugumo parametras.

svarbus. Senesnei schemai taip pat svarbus parametras yra atsitiktinio generatoriaus žodžių **w** aibė ir jos galia. Tačiau kaip ir slapto rakto atveju, šie vektoriai taip pat turi būti formuojami virš didesnių kūnų, todėl generuojant šią w aibę .

Pagrindiniai schemų pakeitimai yra susiję su sugeneruotų parašų tikrinimu. Kadangi atliekama daug parašų tikrinimų, kurie sudaryti virš didesnių kūnų, o tikrinimo sąlygos yra nepakeistos, gaunami klaidingi rezultatai. Pavyzdžiui, analizuojant [Ste94] straipsnyje pateiktą schemą, buvo pastebėta, kad parašo tikrinimo metu, kai yra atskleistos σ ir $y \oplus s$ reikšmės, sąlygos niekados negali būti išpildytos. Visų pirma, vietoj sudėties modulių 2, mums reikia naudoti sudėtį moduliu q, - tai yra natūralu. Tačiau lygybės $H(y) = H(y \oplus s) \oplus i$ patikrinti, kaip tai buvo siūloma schemą aprašančiame straipsnyje neišeis, nes lygybė $c_1 = \left\langle \tilde{\sigma} \parallel H(\tilde{y}) \oplus i \right\rangle$ visada duos neigiamą rezultatą, jei q > 2. Pavyzdžiui, moduliu 2, 1+1 = 0, tačiau virš didesnių kūnų tai toli gražu ne tas pats. Todėl šiai sąlygai užtikrinti reikia užtikrinti sekančią lygybę $c_1 = \left\langle \tilde{\sigma} \parallel H(\tilde{y}) \oplus \tilde{i} \right\rangle$, kur $\tilde{i}$ tenkina sąlygą $i \oplus \tilde{i} = 0$, t.y. i ir $\tilde{i}$ yra atvirkštinis vektorius sudėties atžvilgiu. Panaši situacija yra ir su senesne schemos versija. Trumpai apžvelgsime pakeitimus, kurie buvo reikalingi norint nauduoti pirmąją Sterno vartotojų identifikacijos schemą virš $GF(q^n)$ kūnų.

Kaip ir aukščiau aprašytoje schemoje, mums reikės pakeisti kai kuriuos skaičiavimus, kad algoritmas būtų tinkamas nauduoti virš didesnių kūnų.

Taigi, pirmoje pasiūlytoje [Ste90] schemoje reikia atlikti sekančius pakeitimus:

1. Jei atsitiktinai iškritęs skaičius b = 0, tada reikia patikrinti ar visos n_j reikšmės yra paskaičiuotos teisingai. Šio parašo tikrinimo metu kiekvienai n_j reikšmei patikrinti ar $y' = n_j - z_j$, kur $y' = y$
2. Jei atsitiktinai iškritęs skaičius b = 1, tada reikia patikrinti ar visos m_j reikšmės yra paskaičiuotos teisingai. Šio parašo tikrinimo metu kiekvienai m_j reikšmei patikrinti ar $y' = m_j - z_j$, kur $y' = y + s$.

Antroje Stern‘o pasiūlytoje [Ste94] schemoje reikalingi tokie pakeitimai:

1. Jei atsitiktinai iškritęs skaičius b = 1, tada c_1 parašo tikrinimui skaičiuojame sekančia išraiška - $c_1 = \left\langle \tilde{\sigma} \parallel H(\tilde{y}) \oplus \tilde{i} \right\rangle$, kur $\tilde{i}$ tenkina sąlygą $i \oplus \tilde{i} = 0$, i – vartotojo slapto rakto sindromas.

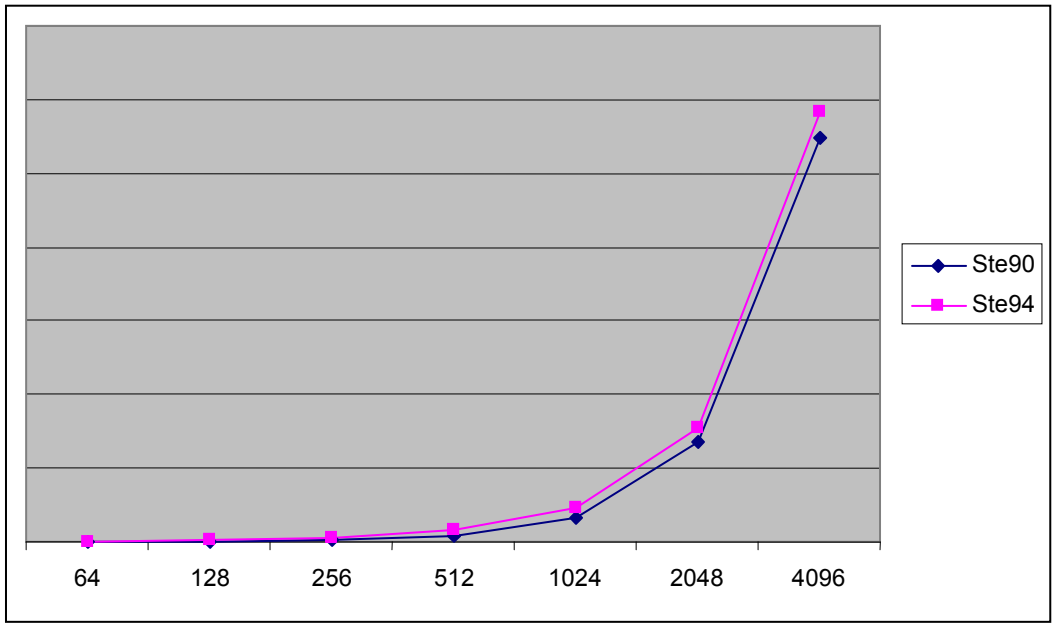
Kaip matome, schemoms pritaikyti prie didesnių kūnų reikia minimalių pakeitimų, tačiau kiek tai yra efektyvu ir kaip tai veikia schemų saugumą bus įvertinta kituose skyriuose.

Patobulintų schemų greitaveika

Kaip buvo minėta anksčiau, schemų greitaveika virš kūnų su skirtingu parametru q esant vienodoms m reikšmėms yra maždaug vienoda, todėl iš visų eksperimentų su vienodais m = n-k dydžiais išvesim vidurkį ir jį naudosime kaip algoritmo laiką su q ir m parametrais. Tada gausime tokią lentelę:

m	Ste90	Ste94
64	0,60	2,45
128	2,29	7,68
256	9,24	19,81
512	36,64	63,87
1024	144,64	185,71
2048	579,17	635,78
4096	2321,48	2557,56

Lentelė 7 Patobulintų algoritmų vartotojų identifikavimo greičiai.



Pav. 8 Grafinis algoritmų greičių palyginimas.

Grafike pagal x ašyje nurodyti kodo kontrolinių matricų dydžiai. Kadangi mes naudojame kodus kurių efektyvumas yra ½. Todėl nėra būtina vardinti abi matricos dimensijas. T.y. jeigu

mes turime pagal x ašį reikšmę 512, tai reiškia, kad kontrolinės matricos dydis bus 1024x512 ir t.t. Pagal y ašyje nurodomas vartotojų identifikavimo greitis.

Kaip matome iš aukščiau pateiktų laiko įverčių, algoritmai yra maždaug lygiaverčiai ir nuo padarytų pakeitimų praktiškai niekas nepasikeitė - pirmas algoritmas vis dar šiek tiek greitesnis už antrą, nes antrame vidutiniškai daugiau atliekama daugybos ir sudėties operacijų, kai tuo tarpu pirmame dominuoja palyginimo operacijos.

Patobulintų schemų saugumas

Kaip buvo minėta aukščiau, norint įveikti šias schemas galima taikyti dvi taktikas:

- Schemos protokolo ataka – kai įsilaužėlis bando paruošti klaidingus parašus.
- Schemos hash funkcijos ataka – kai pagal turimus duomenys bando įveikti schemos hash funkciją $H(s) = H \cdot s^T$.

Pirmo tipo atakos galima išvengti arba sumažinti jos efektyvumą didinant protokolo etapų skaičių, kas pagal Sterno [Ste94] straipsnį yra labai efektyvu, nes tikimybė įveikti protokolą auga eksponentiškai. Todėl pagrindinį dėmesį skirsime antro tipo atakoms. Taip pat, kaip buvo minėta aukščiau, tokio tipo atakoms vykdyti galima išskirti kelias taktikas:

- Kodo struktūros atskleidimo ataka (Struktūrinės).
- Dekodavimo ataka.

Tačiau, žinomų algoritmų realizacijoms kodo struktūros atskleidimo atakoms vykdyti, reikia didesnio operacijų skaičiaus nei dekodavimo tipo atakoms. Be to egzistuoja tokie kodai, pvz. Goppa kodai, kuriems struktūros atskleidimo algoritmai nėra žinomi. Todėl visą dėmesį sutelksim į dekodavimo tipo atakas.

Nagrinėjamų schemų saugumas remiasi $s \to H(s)$ funkcijos invertavimo sudėtingumu. Tarkime, turime slaptą vartotojo raktą $s \in G[n,k,d]$. Tada protokolo vykdymo metu identifikacijos siekiantis vartotojas siunčia savo slapto rakto s hash funkcijos reikšmę i, kur $i = H * s^T$, čia H tai ECC kontrolinė matrica. Gauname m= n-k ilgio hash funkcijos reikšmę, kuri yra ne kas kita, o sindromas. Vadinasi, kad įveikti schemą, įsilaužėliui reikės rasti tokį žodį x, nepriklausantį kodui $G[n,k,d]$, kad $H * x^T \equiv i$ ir $w(x) \equiv p$, kur w() – žodžio Hemingo svoris, o p – algoritmo parametras. Paprasčiausias būdas tam pasiekti, tai perrinkti visus ilgio n žodžius kurių atstumas neviršija d ir žodis nepriklauso kodui. Paprasčiausia ataka, kuri ateina į galvą, tai pilnas perrinkimas. Šiuo metodu reikia perrinkti visus n ilgio žodžius. Patikrinimas ar žodis priklauso kodui gaunasi gana sudėtingas. Žodį

reikia padauginti iš kontrolinės matricos arba turėti paruošus visą kodų lentelę, kas dideliems kodams yra problematiška. Todėl paprasčiausias būdas - perrinkti visus kodo žodžius iš eilės, o žodžiai, kurie priklauso kodui, turės sindromą lygų nuliui.

Dabar panagrinėkime, kas pasikeis, kai mes perrinkimo būdu ieškosime reikiamo žodžio virš skirtingų kūnų. Tam, kad perrinkti visus n ilgio žodžius virš GF(2) kūno, mums reikės išanalizuoti 2^n skirtingų žodžių. Jei tai bus GF(3) kūnas, tai žodžių kiekis bus lygus – 3^n. Tai bendru atveju gauname q^n žodžių. Tokiu atveju didesnių kūnų naudojimas žymiai padidina schemų realizacijų saugumą. Tačiau, kas bus, jei mes pritaikysime algoritmus naudojančius euristikas? Kaip pasikeičia algoritmų saugumas? Ar jis vertas visų pastangų?

Patikrinti kaip pasirinktas kūnas įtakoja dekodavimo atakos operacijų kiekį, naudosime Canteaut ir Chabaud [CC98] pasiūlytą atakos algoritmą. Kaip buvo minėta anksčiau, tai efektyviausias dabar žinomas dekodavimo atakos algoritmas, tačiau jis skirtas kodams virš GF(2) kūnų dekoduoti, nors, kita vertus, jį labai lengvai galima modifikuoti ir pritaikyti bendram atvejui – kodams virš GF(q) kūnų. Dabar trumpai supažindinsime su analizuojamu algoritmu ir pateiksime teorinius operacijų augimo išskaičiavimus.

Trumai aprašysime nagrinėjamą algoritmą:

Inicializacija: Kodo generuojančiai matricai G pritaikome Gauso eliminaciją atsitiktinai pasirinktoje stulpelių aibėje I iš visų kodo N koordinačių. Gausime kodo generuojančią matricą G pavidalu $(\mathrm{I}_{dk}, \mathrm{Z})$.

Algoritmas:

1. Atsitiktinai padaliname aibę I į du poaibius, tokius, kad $\mathrm{I}_1 = \lfloor k/2 \rfloor$ ir $\mathrm{I}_2 = \lceil k/2 \rceil$. Tada Z matricos eilutės suskaidomos į dvi dalis Z_1 ir Z_2. Taip pat atsitiktinai pasirenkame σ dydžio aibę L tokią, kad $x \in L \Rightarrow x \in J = N \setminus \mathrm{I}$
2. Kiekvienai tiesinei π eilučių kombinacijai Λ_1 (arba Λ_2) iš matricos Z_1 (arba Z_2), paskaičiuojame $\Lambda_{1|L}$ ($\Lambda_{2|L}$) reikšmę ir išsaugome ją q^σ ilgio hash lentelėje.
3. Pasinaudojus hash lentele kiekvienai porai (Λ_1, Λ_2), tokiai, kad $\Lambda_{1|L} = \Lambda_{2|L}$, patikriname ar $wt((\Lambda_1 - \Lambda_2)_{|J \setminus L}) = w - 2\pi$. Jei toks žodis rastas, jis dar turi tenkinti tokią sąlygą: $wt(c_{|\mathrm{I}_1}) = wt(c_{|\mathrm{I}_2}) = \pi$ ir $wt(c_{|L}) = 0$.
4. Jeigu tokio vektoriaus neradome, tada išmetame vieną iš matricos I stulpelių ir pakeičiame jį kitu stulpeliu iš Z aibės, tokiu, kad naujo vienetinio vektoriaus y-tojoje pozicijoje irgi būtų 1. Ir pradedame algoritmą nuo pirmo žingsnio.

Žingsnių seka nuo 1 iki 4 vadinama algoritmo etapu.

Pagal [CC98] straipsnio autorių išskaičiavimus, bendras operacijų kiekis reikalingas surasti svorio w ir ilgio n kodo C[n, k, d], generuojamo generuojančios matricos G, žodžiui, apskaičiuojamas pagal formulę:

$$W_{\pi,\sigma} = \frac{\Omega_{\pi,\sigma} E(N)}{\mathrm{A}_w} \quad (1.1)$$

Kur $\Omega_{\pi,\sigma}$ yra skaičius operacijų, reikalingas vienam etapui įvykdyti, $E(N)$ - numatomas etapų skaičius, reikalingas surasti ieškomą kodo žodį, A_w - skaičius kodo C[n, k, d] žodžių, kurių svoris yra w.

Kadangi mus domina tik kaip pasikeis reikalingų operacijų skaičius pakeitus kūną, tai mums pakanka išskaičiuoti santykį:

$$\frac{W^q_{\pi,\sigma}}{W^2_{\pi,\sigma}} = \frac{\Omega^q_{\pi,\sigma} E(N)}{\mathrm{A}^q_w} * \frac{\mathrm{A}^2_w}{\Omega^2_{\pi,\sigma} E(N)} \quad (1.2)$$

Kur q > 2, ir atspindi pakeistą GF(q) kūną, $\Omega^2_{\pi,\sigma}$ - operacijų skaičius reikalingas vienam etapui virš GF(2) kūno, $\Omega^q_{\pi,\sigma}$ - operacijų skaičius reikalingas vienam etapui įveikti virš GF(q) kūno. Atitinkamai A^2_w - skaičius kodo žodžių, kurių svoris yra w, kai mes naudojame kodus virš GF(2^n) kūnų, A^q_w - skaičius kodo žodžių, kurių svoris yra w, kai mes naudojame kodus virš GF(q^n) kūnų.

Kadangi pagal algoritmo specifiką numatomas etapų skaičius nepriklauso nuo pasirinkto kūno, (1.2) formulėje E(N) reikšmės susiprastina. Dabar reikia įvertinti tik du dydžius - $\Omega^q_{\pi,\sigma}$, A^q_w.

[CC98] autorių pasiūlytus išskaičiavimus pritaikant mūsų atvejui, kai naudojami GF(q^n) kūnai, gauname sekančią išraišką:

$$\Omega_{\pi,\sigma} = 2\pi\sigma\binom{k/2}{\pi} + 2\pi(n-k-\sigma)\frac{\binom{k/2}{\pi}^2}{q^\sigma} + 32(\pi\binom{k/2}{\pi} + q^\sigma) + \frac{k(n-k)}{2}$$

(1.3)

Kur:

- $\binom{k/2}{\pi}$ - matricos Z1 (atitinkamai Z2) π skirtingų eilučių tiesinių kombinacijų skaičius. Jų paskaičiavimas ir išsaugojimas hash lentelėje reikalauja apytiksliai $\pi\sigma$ operacijų.

- Vidutinis porų (Λ_1, Λ_2), tokių kad $(\Lambda_1 + \Lambda_2)_{|L} = 0$, skaičius lygus $\frac{(\pi^{k/2})^2}{q^\sigma}$, ir kiekvienai iš jų reikia paskaičiuoti gauto $(n-k-\sigma)$ vektoriaus svorį.
- $32(\pi(\pi^{k/2}) + q^\sigma)$ - operacijų skaičius, reikalingas dinamiškai išskirti atmintį hash lentelės reikšmėms saugoti.
- $\frac{k(n-k)}{2}$ - operacijų skaičius reikalingas matricos Z atnaujinimui.

Kadangi mes turime tipo (2mxm) matricas, formulė (1.3) atrodys taip:

$$\Omega_{\pi,\sigma} = 2\pi\sigma(\pi^{k/2}) + 2\pi(k-\sigma)\frac{(\pi^{k/2})^2}{q^\sigma} + 32(\pi(\pi^{k/2}) + q^\sigma) + \frac{k^2}{2} \quad (1.4)$$

Dabar reikia įvertinti A_w^q dydį. Tegul mes turime atsitiktinai parinktą kodą C[n, m] virš GF(q) kūno, w - žodžio c Hemingo svoris, tada aibės $\Psi = \{c : c \in C[n,k] \;\; ir \;\; wt(c) = w\}$ galią A_w^q galima įvertinti pagal:

$$A_w^q = \frac{(q-1)^w \binom{n}{w}}{q^{n-k}} \quad (1.5)$$

Kadangi vidutiniškai visi kodo žodžių svoriai yra pasiskirstę pagal normalaus pasiskirstymo dėsnį, tai reiškia, kad ir visų kodo klasių svoriai vidutiniškai pasiskirstę pagal normalųjį pasiskirstymo dėsnį. (nes $C_i = a + C$, kur C_i yra kodo C klasė). Vadinasi, vidutiniškai visi svorio w ir ilgio n virš kūno GF(q) žodžiai tolygiai pasiskirstę tarp visų kodo klasių. Kadangi visų n ilgio ir w svorio virš kūno GF(q) vektorių skaičius yra baigtinis ir lygus $(q-1)^w \binom{n}{w}$, o visų klasių skaičius yra q^{n-k}, gauname, kad vidutinis analizuojamų žodžių skaičius atsitiktinai parinktame kode yra $\frac{(q-1)^w \binom{n}{w}}{q^{n-k}}$.

Kaip matome iš formulės (1.5), tam tikro svorio w kodo žodžio pasirodymas kažkokiame atsitiktiniame kode skyrėsi priklausomai nuo kodo n ir k parametrų, o taip pat nuo kūno. Taigi, kad efektyviau įvertinti patobulintų schemų saugumus, fiksuosime kodo generuojančios matricos dydį. Pavyzdžiui 1024x512. Tada, atsižvelgiant į [CHA95] straipsnio rekomendacijas, paimsime schemos virš GF(2) rekomenduotiną parametrą p, kur p yra vartotojo slapto rakto svoris, ir pabandysime rasti tokią naujos schemos parametro p' maksimalią reikšmę, kad $A_{p'}^q$ būtų ko artesnė A_p^2.

Žemiau pateiksime nagrinėjamų ir patobulintų algoritmų vidutinio algoritmo darbo faktoriaus padidėjimą. $F_q = \frac{W^q_{\pi,\sigma}}{W^2_{\pi,\sigma}}$ - darbo faktoriaus padidėjimas, parametrai π=2 ir σ=15 - dekodavimo algoritmo [CC98] parametrai. p‘ – patobulintos identifikacijos schemos vartotojo slapto rakto svoris. q – atitinka patobulintos schemos GF(q).

Q	p'	F_q
3	81	33
5	107	10^4
11	137	10^8
23	157	10^{15}
419	199	10^{35}
1357	208	10^{42}
5501	215	10^{52}

Lentelė 8 Binarinio kodo 512x256, su vartotojo slapto rakto svoriu lygu 56, palyginimas[2].

Q	p'	F_q
3	160	33
5	212	10^4
11	273	10^8
23	313	10^{15}
419	397	10^{35}
1357	415	10^{42}
5501	430	10^{52}

Lentelė 9 Binarinio kodo 1024x512, su vartotojo slapto rakto svoriu lygu 110 palyginimas. [3]

Taip pat iš (1.3) formulės galima pastebėti, kad schemos realizacija virš GF(q) kūnų yra sudėtingesnė, nes 2^σ auga žymai lėčiau negu q^σ, kur q > 2. Tačiau yra dar viena priežastis, dėl kurios schemos įveikimas tampa sudėtingesnis – tai atminties kiekis. Pagal [CC98] algoritmą, visos reikšmės saugomos hash lentelėse ir kiekvienos lentelės raktų aibės dydis turi būti q^σ, o tai didėjant q reikšmei tampa „didele našta“, nes reikalauja daug atminties.

[2] Visų nagrinėjamų schemų ECC kodų kontrolinių matricų dydžiai vienodi.

[3] Visų nagrinėjamų schemų ECC kodų kontrolinių matricų dydžiai vienodi.

Išvados

Detaliai apžvelgėme pagrindines schemų [Ste90] ir [Ste94] idėjas: tiek teorines, kurios buvo pasiūlytos aukščiau išvardintuose darbuose, tiek ir praktines, t.y. schemų realizacijos aspektus. Ištyrėme schemų saugumo aspektus: galimas atakas, įveikimo laiką esant vienodom pradinėm sąlygom. Palyginome nagrinėjamas schemas su kitomis žinomomis kriptografinėmis schemomis. Pateikėme schemų su prasmingais schemų parametrais veikimo greičių įvertinimus. Išvardijome nagrinėjamų schemų patobulinimo būdus. Aprašėme patobulintų schemų realizacijos pricipus. Ištyrėme patobulintų schemų saugumo aspektus.

Pagal atliktus tyrimus, patobulintos identifikacijos schemos yra saugesnės, nei originalios Stern identifikacijos schemos, t.y. jos reikalauja atlikti daugiau operacijų ir naudoja daugiau atminties dekodavimo atakai atlikti, tačiau jos sunkiau realizuojamos.

Summary

In this paper we discussed [Ste90] and [Ste94] identification schemes theoretical and practical aspects, such as: security aspect – main attack types, time needed to break any of discussed schemes (with the same parameter values). We compared the discussed identification schemes with other identification schemas based on other mathematical problems. In this paper we computed user identification time with practical parameters of all presented schemas, described possible improvements of our schemes, and examined proposed improvement of schemes.

According to our research, all improved identification schemes are more safe then original Stern proposed identification schemes. They demand more operations and memory resources to perform decoding attack. However implementation of improved schemes is more difficult.

Priedas A. Literatūros sąrašas

[Ste90] Jacques Stern. An Alternative to the Fiat-Shamir Protocol. J.-J. Quisquater, J. Vandewalle (Ed.), Advances in Cryptology - EUROCRYPT '89, Workshop on the Theory and Application of Cryptographic Techniques, Houthalen, Belgium, April 1989, Lecture Notes in Computer Science 434, p. 173-180, 1990.

[Ste94] Jacques Stern. A New Identification Scheme Based on Syndrome Decoding. In: Lecture Notes in Computer Science 773, Advances in Cryptology - CRYPTO '93, p. 13-21, Springer-Verlag, 1994

[FiS87] A. Fiat ir A.Shamir. How to prove yourself: Practical solutions in identification and signatures problems. Proceedings of Eurocrypt 86, Lecture note in Computer Science 263, 181-187.

[Mce78] R. J. McEliece. A Public-Key Cryptosystem Based On Algebraic Coding Theory. JPL DSN Progress Report 42-44, pp. 114-116, April 15, 1978.

[Nie86] H. Niederreiter. Knapsack-type cryptosystems and algebraic coding theory. Probl. Control and Inform. Theory 15(2), pp. 159-165, 1986

[DW94] Robert H. Deng Yuan Xing Li and Xin Mei Wang. On the equivalence of mceliece's and niederreiter's public key cryptosystems. IEEE Trans. on Information Theory, 40(1):271–273, January 1994.

[CC98] A. Canteaut and F. Chabaud. A new algorithm for finding minimum-weight words in a linear code: application to McEliece's cryptosystem and to narrow-sense bch codes of length 511. IEEE Transactions on Information Theory, 1998, pages 367-378.

[BeR93] M.Bellare ir P.Rogaway. Random oracles are practical: a paradigm for designing efficient protocols. Proceedings of the 1st ACM Conference on Computer and Communications Security, 1993.

[YDC92] Yuang-Xian Li, Da-Xing Li, Chuan-Kun Wu. How to generate a random nonsingular matrix in McEliece's public-key cryptosystem. Singapore ICCS/ISITA '92 'Communications on the Move' 16-20 Nov. 1992, Vol. 1, pp. 268 – 269

[Ran93] D. Randall. Efficient generation of random nonsingular matrices. *Random Structures and Algorithms* , 4: 111-118, 1993

[JJ02] Johansson, T.; Jonsson, F. On the complexity of some cryptographic problems based on the general decoding problem. IEEE Transactions on Information Theory, Volume 48, Issue 10, pp. 2669 - 2678, Oct. 2002.

[Sen00] N. Sendrier. Finding the permutation between equivalent codes: the support splitting algorithm. IEEE Transactions on Information Theory, 46(4):1193{1203, July 2000.

[Ste89] J. Stern, "A method for finding codewords of small weight," in Coding Theory and Applications (Lecture Notes in Computer Science), G. Cohen and J.Wolfman, Eds. Berlin, Germany: Springer-Verlag, 1989, vol. 388, pp. 106–113.

[CC98] A. Canteaut and F. Chabaud, "A new algorithm for finding minimumweight words in a linear code: Application to McEliece cryptosystem and to narrow-sense BCH codes of length 511," IEEE Trans. Inform. Theory, vol. 44, pp. 367–378, Jan. 1998.

[Cha95] Florent Chabaud. On the Security of Some Cryptosystems Based on Error-Correcting Codes. In: Lecture Notes in Computer Science 950, Advances in Cryptology - EUROCRYPT '94, p. 131-139, Springer-Verlag, 1995.

[FiS87] A. Fiat ir A.Shamir. How to prove yourself: Practical solutions in identification and signatures problems. Proceedings of Eurocrypt 86, Lecture note in Computer Science 263, 181-187.

[FFS87] U. Feige, A.Fiat, A. Shamir Zero-knowledge proofs or identity Proc. 19th ACM symp. Theory of computing. 1987, 210-217 ir J. Cryptology 1988 77-95.

[GuQ88] L.S. Guillou and J.-J. Quisquater. A practical zero-knowledge protocol fitted to security microprocessors minimizing both transmission and memory. Proceedings of Eurocript 88, Lecture Notes in Computer Science 339, 123-128.

[Oka92] T. Okamoto. Provably secure and practical identification schemes and corresponding signature schemes. Proceedings of Crypto 92, Lecture Notes in Computer Science 740, 31-53.

[OhO88] K. Ohta ir T. Okamoto. A modification of the Fiat-Shamir scheme, Proceedings of Crypto 88, Lecture Notes in Computer Science 403, 232-243.

[Sha89] A. Shamir. An efficient identification scheme base don permuted kernels, Proceedings of Crypto 89, Lecture Notes in Computer Science 435, 606-609..

[Sch91] C.P. Schnorr. Efficient signature generation by smart cards. J. Cryptology, 4 (1991), 161-174.