KAUNO TECHNOLOGIJOS UNIVERSITETAS

MATEMATIKOS IR GAMTOS MOKSLŲ FAKULTETAS

Indrė Starevičiūtė

PREKIŲ KILMĖS PATVIRTINIMO QR KODO KRIPTOGRAFINĖ SAUGUMO SISTEMA

Baigiamasis magistro projektas

Vadovas
Prof. dr. E. Sakalauskas

KAUNAS, 2016

KAUNO TECHNOLOGIJOS UNIVERSITETAS

MATEMATIKOS IR GAMTOS MOKSLŲ FAKULTETAS

PREKIŲ KILMĖS PATVIRTINIMO QR KODO KRIPTOGRAFINĖ SAUGUMO SISTEMA

Baigiamasis magistro projektas
Taikomoji matematika (kodas 621G10003)

Vadovas
Prof. dr. E. Sakalauskas
2016 05 30

Recenzentas
Doc. dr. A. Aleksa
2016 06 01

Projektą atliko
Indrė Starevičiūtė
2016 05 30

KAUNAS, 2016

KAUNO TECHNOLOGIJOS UNIVERSITETAS

Matematikos ir gamtos mokslų fakultetas

(Fakultetas)

Indrė Starevičiūtė

(Studento vardas, pavardė)

Taikomoji matematika, 621G10003

(Studijų programos pavadinimas, kodas)

„Prekių kilmės patvirtinimo QR kodo kriptografinė saugumo sistema“

AKADEMINIO SĄŽININGUMO DEKLARACIJA

2016-05-30
Kaunas

Patvirtinu, kad mano, **Indrės Starevičiūtės**, baigiamasis projektas tema „Prekių kilmės QR kodo kriptografinė saugumo sistema“ yra parašytas visiškai savarankiškai ir visi pateikti duomenys ar tyrimų rezultatai yra teisingi ir gauti sąžiningai. Šiame darbe nei viena dalis nėra plagijuota nuo jokių spausdintinių ar internetinių šaltinių, visos kitų šaltinių tiesioginės ir netiesioginės citatos nurodytos literatūros nuorodose. Įstatymų nenumatytų piniginių sumų už šį darbą niekam nesu mokėjęs.

Aš suprantu, kad išaiškėjus nesąžiningumo faktui, man bus taikomos nuobaudos, remiantis Kauno technologijos universitete galiojančia tvarka.

(vardą ir pavardę įrašyti ranka) *(parašas)*

Starevičiūtė Indrė. Prekių kilmės patvirtinimo QR kodo kriptografinė saugumo sistema. Magistro baigiamasis projektas / vadovas prof. dr. Eligijus Sakalauskas; Kauno technologijos universitetas, Matematikos ir gamtos mokslų fakultetas.

Mokslo kryptis ir sritis: matematika, fiziniai mokslai

Reikšminiai žodžiai: *QR kodas, ECDSA, RSA, elektroninis parašas, kriptografinė sistema*

Kaunas, 2016. 63 p.

SANTRAUKA

Gyvenant išmanaus vartojimo laikais yra aktualu informaciją talpinti virtualioje erdvėje. Vis greičiau auganti išmaniųjų telefonų rinka siūlo puikius marketingo sprendimus, tokius kaip mobiliosios etiketės (QR kodai), kurios leidžia susieti fizinį objektą su atitinkamais interneto resursais. Tačiau vykdant tam tikrą duomenų judėjimą virtualioje erdvėje reikia užtikrinti vartotojo pasitikėjimą, išsaugant duomenų failo autentiškumą. Todėl šio darbo metu buvo analizuojamas dviejų pasirinktų kriptografinių sistemų saugumas, tiriant QR kodo prekių kilmės patvirtinimo atvejį. Prieš analizuojant duomenis yra svarbu susipažinti su kriptografinių duomenų aspektais – matematinės schemos, vadinamos elektroniniu parašu, panaudojimo galimybėmis. Taip pat svarbu apžvelgti QR kodo infrastruktūrą. Atlikus trumpą teorinę analizę, darbui buvo pasirinktos dvi kriptografinės sistemos (RSA ir ECDSA – modernūs ir saugūs algoritmai, kurie yra plačiai naudojami įvairiuose kriptografiniuose tyrimuose), jos išanalizuotos bei palyginti jų saugūs kriptografiniai parametrai. Nusistačius parametrų reikšmes buvo realizuoti parametrų generavimo ir tikrinimo algoritmai naudojant OpenSSL atvirojo kodo paketą. Susikūrus tinkamą prekės kilmės duomenų failą, jam buvo sugeneruoti atitinkami QR kodai, talpinantys ne tik naudingą duomenų failo informaciją, bet ir kriptografinius duomenis. Galiausiai, buvo palyginti abiejų pasirinktų kriptografinių sistemų gauti QR kodų talpinamos informacijos kiekiai, geometriniai plotai ir nuskaitymo patikimumo galimybės bei pasirinkta viena – ECDSA – efektyvesnė kriptografinė sistema, užtikrinanti informacijos autentiškumą ir integralumą.

Starevičiūtė Indrė. QR Code Cryptographic Security System While Analyzing Certification Of Commodity Origin: Master‘s thesis in applied mathematics / supervisor prof. dr. Eligijus Sakalauskas. The Faculty of Mathematics and Natural Sciences, Kaunas University of Technology.

Research area and field: mathematics, physical sciences

Key words: *QR code, ECDSA, RSA, digital signature, cryptographic system*

Kaunas, 2016. 63 p.

SUMMARY

Due to rapid growth of media and communication technology, now sharing information in cyberspace can be defined as a common thing. The growing smartphone market has excellent marketing solutions under offer such as QR codes, which allows to link the physical object to the relevant internet resources. However, to ensure consumer confidence while sharing some information through the insecure media, it is important to preserve the authenticity of the data file. This is the reason the security of two selected cryptographic systems had been analyzed when investigating the true origin of the commodity. Before analyzing the data, it is important to get acquainted with usability of such aspect of the cryptographic data as mathematical scheme, known as the digital signature. It is also significant to examine all facilities of QR code. A brief theoretical analysis helped to decide on two cryptographic systems (RSA and ECDSA) that are known as a modern and extremely secure crypto-algorithms, currently internationally accepted for online transactions and many other policies. In the present work, they both had been analyzed and compared taking safe cryptographic parameters of each into account. After setting the values of parameters, algorithms of generation and verification had been realized using the OpenSSL open source package. Once correct origins of data files had been established, appropriate QR codes were generated that contained not only the necessary information of the data file, but also the cryptographic data. Finally, after the discussion on contained information quantities, geometric plots and scanning reliability of both selected cryptographic systems, decision on one more effective system (ECDSA) which appeared to ensure better authenticity and integrity of information was made.

TURINYS

LENTELIŲ SĄRAŠAS

PAVEIKSLŲ SĄRAŠAS

ĮVADAS

Gyvename išmanaus vartojimo laikais, todėl sukurti tinkamiausią pasiūlymą tampa vis sudėtingesniu iššūkiu marketingo specialistams. Šių dienų vartotojai yra išsiskiriantys savo reiklumu, žingeidumu ir aktyvumu. Šiuolaikinis vartotojas – tai protingas, praktiškas vartotojas, kuris nori įsitraukti ir dalyvauti procese, turėti galimybę komunikuoti bei lengvai dalintis informacija. Toks vartotojas daugiausiai informacijos gauna virtualioje aplinkoje, todėl marketingo sprendimus tikslinga nukreipti būtent į virtualią aplinką.

Lietuvoje viena greičiausiai auganti yra išmaniųjų telefonų rinka, kurios vartotojų skaičius vis didėja. Mobiliosios etiketės – tai naujos technologijos, leidžiančios susieti bet kokį fizinio pasaulio objektą su atitinkamais interneto resursais, siekiant susietą objektą praplėsti ar virtualiai papildyti [1]. Pavyzdžiui, įsivaizduokite straipsnį žurnale su gale teksto pavaizduota "mobiliąja etikete". Ją, nuskanavęs mobiliuoju telefonu, skaitytojas gali sužinoti daugiau papildomos informacijos apie straipsnį: pamatyti paveiksliukų, vaizdo reportažų, stebėti užkulisinę informaciją ir pan.

QR kodai – tai viena paprasčiausių ir nekainuojančių priemonių, kaip perduoti reikalingą informaciją greitai tik mobiliojo telefono pagalba. Tai fiksuoto duomenų kiekio laikmena, skirta užkoduotai informacijai identifikuoti. Palyginti su paprastu brūkšniniu kodu, QR kodai daug talpesni informacijos atžvilgiu ir turi didesnę praktinę naudą. QR kode gali būti sukonfigūruotas internetinės svetainės adresas, kontaktiniai duomenys ir kita norima informacija. Jie gali būti naudojami bet kur: ant paties produkto, ant kvito, plakato ar skrajutės, vizitinės kortelės ar internetinės svetainės. Tačiau turi būti pakankamo dydžio, kad QR kodų skaitytuvu būtu galima nuskanuoti koduojančius taškus. Kiekvienam naudotojui reikia turėti skanuojančią programą (jos parsisiuntimas nemokamas), o įmonei QR kodo sukūrimas kainuos tik darbuotojo darbo laiką, nes internete yra nemokamos svetainės QR kodo konfigūracijai. Šiuo metu (ypač užsienyje) įvairiose mugėse, verslo konferencijose yra atsisakoma įprastinių reklaminių priemonių, kaip plakatai, skrajutės, informaciniai žurnalai. Priėjus prie stendo vartotojas skanuojasi QR kodą ir visą pristatomąją medžiagą akimirksniu turi savo mobiliajame telefone [2].

Tačiau, reikia nepamiršti, kad vykdant tam tikrą duomenų judėjimą virtualioje erdvėje, vienas iš svarbiausių uždavinių yra išsaugoti duomenų failo autentiškumą ir taip užtikrinti vartotojo pasitikėjimą. Tam yra naudojami ypač saugūs standartiniai kriptografijos metodai, užtikrinantys apsaugą nuo duomenų informacijos pakeitimo. Skaitmeninės žinutės ar dokumento autentiškumo patvirtinimui yra naudojama matematinė schema, vadinama skaitmeniniu arba elektroniniu parašu (toliau – e. parašas). E. parašo panaudojimo tikslas yra aptikti duomenų klastojimą. Toks autentifikavimo mechanizmas suteikia

galimybę duomenų failo kūrėjui, kaip apsaugą prie turimos informacijos, prijungti tam tikrą kodą, kuris yra vadinamas e. parašu [3].

Taigi, QR kode yra patalpinti kriptografiniai duomenys kartu su pagrindine duomenų failo informacija. E. parašo informacija užtikrina duomenų failo vientisumą ir apsaugo kuriamą failą nuo klastojimo bei informacijos pakeitimo [4]. Kriptografinės informacijos dalyje taip pat yra naudojama santraukos funkcija, kuri laikoma labai svarbiu ir galingu įrankiu siekiant sukurti metodus, apsaugančius informacijos autentiškumą [1]. Kriptografiniai duomenys negali egzistuoti be matematiškai susijusių raktų poros, kurie užtikrina informacijos autentiškumą ir neišsiginamumą. Tuomet, galiausiai, turint visus kriptografinius duomenis ir panaudojant tam tikrą specialią išmanaus telefono programėlę, nuskaitomas sukurtas QR kodas ir taip patikrinimas e. parašo autentiškumas [4].

Šio darbo pagrindinis tikslas – išanalizuoti dviejų pasirinktų kriptografinių sistemų saugumą, tiriant QR kodo prekių kilmės patvirtinimo atvejį. Tikslui pasiekti, pirmiausia, darbe bus atliekama kriptografinių duomenų analizė, apžvelgiama QR kodo sandara, veikimo principai, talpinamos informacijos kiekiai bei informacijos kodavimo būdai, išnagrinėjamos panaudojimo ir generavimo galimybės. Vėliau, pasirinkus dvi kriptografines sistemas, jas išanalizavus bei palyginus jų parametrus (santraukos funkcijų, raktų, e. parašo ilgius) bus realizuoti kriptografinių parametrų generavimo algoritmai. Susikūrus tinkamą prekės kilmės duomenų failą, jam bus sugeneruoti atitinkami QR kodai. Galiausiai, palyginus gautus abiejų pasirinktų kriptografinių sistemų QR kodų talpinamos informacijos kiekius, geometrinius plotus ir nuskaitymo patikimumo galimybes, bus pasirenkama viena – efektyvesnė kriptografinė sistema, užtikrinanti informacijos autentiškumą ir integralumą.

1. LITERATŪROS APŽVALGA

1.1. SANTRAUKOS FUNKCIJŲ ANALIZĖ

1.1.1 SANTRAUKOS FUKCIJA

Santraukos funkcija [5] – tai kriptografinė transformacija, iš bet kokio ilgio pranešimo sugeneruojanti fiksuoto ilgio duomenų bloką, kuris yra vadinamas santrauka (angl. *hash value* arba *digest*).

Toliau santraukos funkciją žymėsime h, o pranešimo X santrauką – $h(X)$. Pagrindinė santraukos funkcijų savybė – spartus (efektyvus) pranešimo santraukos apskaičiavimas ($X \rightarrow h(X)$) ir itin sudėtingas (ar net neįmanomas) pranešimo atkūrimas iš santraukos ($h(X) \rightarrow X$).

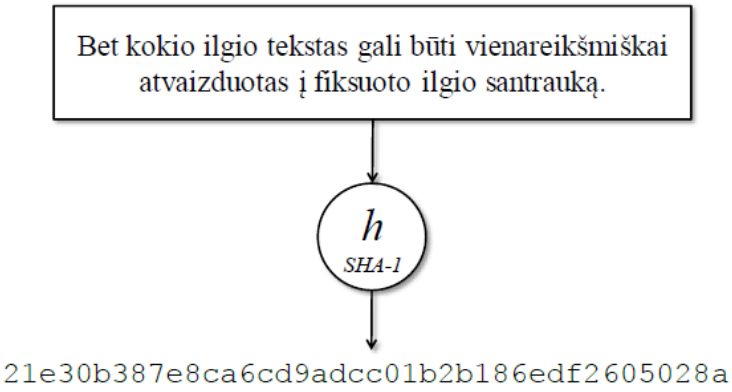
Santraukos funkcijos algoritmai veikia taip, kad net dėl menkiausių duomenų pakeitimų gaunama visiškai skirtinga santraukos reikšmė. Tai neleidžia (ar bent padaro gerokai sudėtingiau) kenkėjams rasti kito pranešimo, kurio santrauka būtų tokia pat. Situacija, kai randami du pranešimai, kurių santraukos funkcijos sutampa, vadinama kolizija. Saugios santraukos funkcijos turi būti atsparios kolizijai.

Skiriamos šios pagrindinės saugių santraukos funkcijų savybės [5]:

- Pirmavaizdžio atsparumas: turint santrauką $h(X)$, neįmanoma skaičiuojant rasti ją atitinkančio pranešimo X.
- Antrojo pirmavaizdžio atsparumas: turint pranešimą X ir jo santrauką $h(X)$, skaičiuojant neįmanoma rasti pranešimo $X' \neq X$, kurio santrauka sutaptų su X, t. y. $h(X') = h(X)$.
- Kolizijos atsparumas: skaičiuojant neįmanoma rasti tokių dviejų pranešimų, kurių santraukos sutaptų.

Kriptografijoje santraukos funkcijos taip pat naudojamos pranešimų autentiškumui užtikrinti ir net slaptažodžiams apsaugoti. Norint santraukos funkciją naudoti pranešimo autentiškumui užtikrinti, santrauka turi būti siunčiama atskirai nuo pranešimo saugiu kanalu arba turi būti naudojama kaip e. parašo sudedamoji dalis.

Tam, kad nereikėtų saugoti paties slaptažodžio, kai kuriose sistemose saugoma tik slaptažodžio santrauka. Vartotojui įvedus slaptažodį, apskaičiuojama jo santrauka ir patikrinama, ar ji sutampa su esančia duomenų bazėje. Jei santraukos sutampa, vartotojui suteikiama prieiga prie sistemos, naudojant įvestą slaptažodį. Tokiu principu veikia *Microsoft Windows* operacinių sistemų registracija.



1.1 pav. Kriptografinė santraukos funkcija [5]

1.1 paveikslėlyje pateiktame pavyzdyje 91 simbolio, arba 728 bitų (ASCII koduotės atveju), ilgio pranešimas atvaizduojamas į 160 bitų, arba 20 simbolių, šešioliktainiu formatu santrauką, naudojant SHA-1 algoritmą. Gauta santrauka yra lyg pirštų atspaudai, pagal kuriuos nustatomas pradinis sakinys. Kolizijos atsparumo savybė rodo, kad beveik neįmanoma rasti kito pranešimo, kurio santrauka šešioliktainiu formatu taip pat būtų lygi

`21e30b387e8ca6cd9adcc01b2b186edf2605028a`.

Jei pranešimas yra tik vienas žodis, naudojant tą patį santraukos algoritmą jis bus atvaizduotas į tokio pat ilgio santrauką kaip ir ilgesnis pranešimas. Pavyzdžiui, slaptažodžio „raktas“ SHA-1 santrauka šešioliktainiu formatu:

`1bd8c54db1ed4adb595d4400a9ad5d4de3734774`.

Jei kenkėjas norėtų rasti santrauką atitinkantį slaptažodį, jam tektų perrinkti visus galimus variantus ir kiekvienu atveju apskaičiuoti bei palyginti santraukos funkcijos reikšmę. Todėl santraukos funkcija laikoma nesaugia, jei per priimtiną laiko tarpą [5]:

- įmanoma atkurti nežinomą pranešimą, turint jo santrauką;
- galima rasti du tokius pranešimus, kurių santraukos funkcijos sutaptų;
- galima parinkti tokį pranešimą, kurio santrauka sutaptų su kito turimo pranešimo santrauka.

1.1.2 SANTRAUKOS FUNKCIJŲ ALGORITMAI

Šiame poskyryje trumpai aptarsime esamus ir labiausiai naudojamus santraukos funkcijų algoritmus.

Galime suskirstyti labiausiai žinomas ir dažniausiai sutinkamas santraukos funkcijas į dvi grupes, iš kiekvienos išskiriant atitinkamas santraukos funkcijas [6]:

- MD: MD2, MD4, MD5
- SHA: SHA1, SHA224, SHA256, SHA384, SHA512

Toliau trumpai aptarsime ir išanalizuosime kiekvienos iš funkcijų parametrus ir palyginsime jų savybes bei panaudojimo efektyvumą [6].

MD2 (angl. *Message-Digest algorithm*). Pirmoji MD5 algoritmo versija, sukurta profesoriaus Ronaldo Rivesto ir skirta naudoti didelės apimties duomenų parašui sukurti.

- MD2 maišos algoritmas yra optimizuotas 8 bitų kompiuteriams.
- Pranešimas papildomas taip, jog jo ilgis dalintųsi iš 16 B (128b), į pabaigą pridedama 16 B kontrolinė suma.
- Gautam pranešimui skaičiuojamas maišos rezultatas.
- Santrauka – 128 bitų ilgio.
- Buvo pasiūlytas metodas galintis sukurti kolizijas, jei nenaudojama kontrolinė suma.

Po kelių metų R. Riverstas pasiūlė naujesnę MD2 algoritmo versiją – **MD4**.

- MD4 maišos algoritmas yra optimizuotas 32 bitų kompiuteriams.
- Pranešimas papildomas taip, kad jo ilgis dalintųsi iš 512b (kartu su 64b pranešimo ilgiu, pridėtu į pabaigą).
- Pranešimas apdorojamas 512b blokais.
- Santrauka – 128 bitų ilgio.
- Labai greitai buvo aptikta efektyvių atakų, metodikos, kurios gali surasti kolizijas per kelias minutes.

Galiausiai buvo R. Riversto buvo patobulintas MD4 algoritmas ir pavadintas **MD5.**

- MD5 maišos algoritmas yra optimizuotas 32 bitų kompiuteriams.
- Naudoja 512b blokus, pranešimą papildo kaip ir MD4.
- Santrauka – 128 bitų ilgio.
- Netinka dokumentų parašams.

Taigi, tai gana populiari santraukos funkcija, apskaičiuojanti 128 bitų ilgio parašą. Nors MD5 algoritmas (kaip ir kiti algoritmai) gali apskaičiuoti parašą nuo begalinio skaičiaus įeinamų duomenų, galimų sugeneruotų kodų (parašų) skaičius yra baigtinis – 2 128. Todėl jau seniai buvo žinoma, kad kolizijos egzistuoja. Anksčiau tai nebuvo aktualu dėl reliatyviai silpnų kompiuterių pajėgumo, tačiau dabar MD5 algoritmas laikomas nesaugus. Tai parodė 2004 m. atrastas pažeidžiamumas.

SHA (angl. *Secure Hash Algorithm*). Saugus maišos algoritmas iš pranešimo, kuris yra mažesnis už 264 bitų, generuoja 160 bitų kodą.

1995 m. NIST pasiūlė **SHA-1** maišos algoritmą. SHA-1 yra populiarus algoritmas, naudojamas įvairiose programose ir protokoluose, tokiuose kaip TLS, SSL, PGP, SSH, S/MIME, IPSec. SHA-1 buvo laikomas pažeidžiamo MD5 algoritmo įpėdiniu.

- Maišos rezultatas – 160 bitų ilgio
- Veikimo principas panašus į MD4 ar MD5 algoritmus
- Patvirtintas kaip FIPS standartas (FIPS PUB 180-1)
- Labai populiarus, pakeitė MD5

SHA-1 algoritmo kodas yra skaičiuojamas taip:

- pradinis tekstas suskirstomas i N blokų po 512 bitų (64 baitus);
- jei paskutiniame M_n bloke trūksta informacijos iki 512 bitų, bloko gale pridedamas 1 ir tiek 0, kad butų užpildytas blokas paliekant 64 bitus pradinio teksto ilgio išsaugojimui bitais;
- Kiekvienas žodis apdorojamas per 80 žingsnių
- Apdorojus visus blokus, vidinę būseną vaizduoja penkios 4 baitų (32b) reikšmės: H_0, ..., H_4
- Maišos rezultatas: $H_0 \| H_1 \| H_2 \| H_3 \| H_4$

Taip pat yra **SHA-1** santraukos funkcijos atmainų, skaičiuojančių skirtingo dydžio bitų kodus: **SHA256**, **SHA384**, **SHA512**, atitinkamai skaičiuoja 256, 384 ir 512 bitų santraukų kodus. SHA-224 ir SHA-384 yra tiesiog „sutrumpintos“ atitinkamų ilgesnių santraukų versijos, skaičiuojamos naudojant kitas konstantas. Visos SHA-1 santraukos funkcijos atmainos apibendrintai vadinamos SHA-2 santraukos funkcijomis.

1.1 lentelėje galime palyginti aptartų santraukos funkcijų algoritmus, matydami visus anksčiau išvardintus parametrus vienoje vietoje:

1.1 lentelė. Santraukos funkcijų palyginimas [7]

Pavadinimas	Bloko dydis (bitais)	Žodžio ilgis (bitais)	Santraukos rezultatas (bitais)	Ciklai	Standarto sukūrimo metai
MD4	512	32	128	48	1990
MD5	512	32	128	64	1992
SHA-0	512	32	160	80	1993
SHA-1	512	32	160	64	1995
SHA-224	512	32	224	64	2004
SHA-256	512	32	256	80	2002
SHA-384	1024	64	384	80	2002
SHA-512	1024	64	512	10	2002

Taip pat, yra naudinga atkreipti dėmesį į maišos funkcijų gyvavimo ciklus (žr. 1.2 lentelė). Lentelėje žalia spalva nuspalvinti langeliai reiškia, kad santraukos funkcija yra nepažeista ir tinkama naudoti; geltona – galimas dalinis naudojimas, t. y. aptikti tik tam tikti funkcijos pažeidimai, kurie susilpnina funkcijos saugumą; raudona – santraukos funkcijos naudoti negalima, dėl aptiktų ir įrodytų funkcijos pažeidimų.

1.2 lentelė. Santraukos funkcijų gyvavimo ciklai [6]

h(*X*)	1990	1991	1992	1993	1994	1995	1996	1997	1998	1999	2000	...	2003	2004	...	2009
MD4																
MD5																
MD2																
SHA-0																
SHA-1																
SHA-2 šeima																

Iš pateiktos lentelės, matome, kad SHA-1 maiša nuo 2004 metų iš 2^{80} „liko“ tik 2^{69} saugos bitų ir yra tinkama naudoti. SHA-2 šeimos funkcijos yra netgi saugesnės už SHA-1, tačiau jos nėra tokios populiarios ir tyrinėjamos. Tuo tarpu MD funkcijos yra laikomos nesaugiomis.

1.2. KRIPTOGRAFINĖS VIEŠOJO RAKTO ŠIFRAVIMO SISTEMOS

Apibrėžkime kelias pagrindines sąvokas, kurios bus naudojamos viso darbo metu [8]:

- Kriptografija – mokslas, kuriantis ir nagrinėjantis įvairias šifravimo sistemas.
- Šifravimas – tai duomenų kodavimas, norint paslėpti jų turinį.
- Iššifravimas – užšifruoto duomenų turinio atstatymas.
- Tekstograma – pradiniai duomenys, kuriuos norime užšifruoti.
- Šifrograma – gauti duomenys, atlikus užšifravimą.

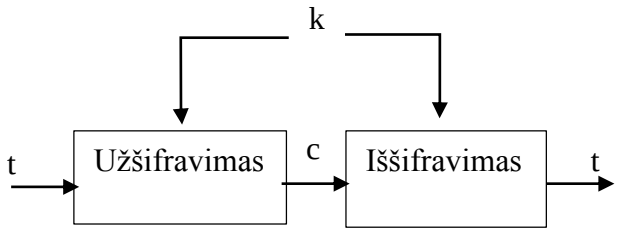
Taigi, kriptografijoje egzistuoja du pagrindiniai šifravimo/iššifravimo algoritmų tipai, kurie yra pritaikomi kuriant kriptografines e. balsavimo sistemas [9]:

1) Simetrinė šifravimo sistema.
2) Asimetrinė šifravimo sistema.

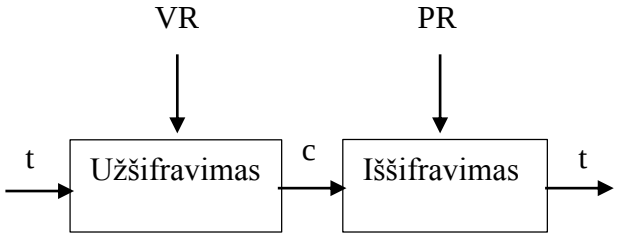
Simetrinė šifravimo sistema [10] – tai slaptojo rakto sistema. Šios sistemos veikimas paremtas vieno rakto sukūrimu, kuris naudojamas tiek duomenų užšifravimui, tiek iššifravimui. Tuo tarpu asimetrinė šifravimo sistema [11] – tai viešojo rakto sistema, kurios pagrindinė savybė ta, kad užšifravimui yra naudojamas viešasis raktas, o iššifruojama – slaptu privačiuoju raktu. Taigi, pagrindinis aptartų šifravimo sistemų skirtumas – simetrinė sistema naudoja vieną, o asimetrinė du skirtingus raktus šifravimo ir iššifravimo procesuose.

Šių sistemų pagrindinis veikimo principas (pavaizduotas 1.2 ir 1.3 pav.):

Tekstograma *t* yra užšifruojama raktu (simetrinės šifravimo sistemos atveju – tai slaptasis raktas *k*, o asimetrinėje sistemoje – tai viešasis raktas *VR*) ir gaunama šifrograma *c*, kuriai dešifruoti vėlgi naudojamas raktas (simetrinėje – tas pats slaptasis raktas *k*, o asimetrinėje – privatus gavėjo raktas *PR*). Atlikus dešifravimą yra gaunama pradinė tekstograma *t*.



1.2 pav. Simetrinė šifravimo sistema [12]



1.3 pav. Asimetrinė šifravimo sistema [13]

Realiose sistemose yra naudojamas abiejų šifravimo sistemų mišinys, t. y. siekiama išnaudoti kiekvienos sistemos pranašumus [14]. Darbo metu daugiau susidursime su asimetrinėmis šifravimo sistemomis. Taigi, plačiau panagrinėkime asimetrinių sistemų raktų veikimo procesą.

Kaip jau išsiaiškinome, asimetrinės šifravimo sistemos veikimas reikalauja dviejų skirtingų raktų kombinacijos. Viešasis raktas yra saugomas viešų raktų faile, todėl jo informacija yra pasiekiama masėms. Tačiau užkoduotą pranešimą su viešuoju raktu gali dešifruoti tik slaptojo rakto savininkas. Taigi, viešo rakto žinojimas nepadeda atgaminti slaptojo rakto. Be to, dažnai yra atliekamas autentifikavimo procesas, kurio metu siuntėjas pasirašo pranešimą naudodamas savo slaptąjį raktą, o gavėjas panaudodamas siuntėjo viešąjį raktą, gali patikrinti parašą ir įsitikinti asmens teisėtumu. Slaptas raktas yra taip pat papildomai šifruojamas, norint apsisaugoti nuo rakto pavogimo. Taigi, asimetrinės šifravimo sistemos veikimo principas yra paremtas tuo, kad nors abu raktai yra matematiškai susiję, tačiau žinant viešąjį raktą, gauti slaptąjį raktą yra neįmanoma [11].

1976 m. W. Diffie ir M. Hellman iškėlė viešojo rakto kriptosistemos idėją ir pasiūlė originalų raktų apsikeitimo algoritmą. Sukurtas algoritmas neleidžia šifruoti siunčiamų pranešimų, tačiau yra sėkmingai pritaikomas raktų generavimo procese [15]. 1978 m. R. Rivest, A. Shamir, L. Adleman sukūrė pirmą praktikoje panaudotą viešojo rakto šifravimo ir e. parašo schemą, dabar žinomą kaip RSA. RSA algoritmas pagrįstas didelių skaičių faktorizacijos (skaidymo į pirminius daugiklius) sudėtingumu. Šiomis dienomis RSA skaitmeninio parašo schema kaip ir pati viešojo rakto kriptosistema yra plačiai tyrinėjama ir naudojama. RSA sistema yra laikoma viena iš esminių blokų augančioje viešojo rakto infrastruktūroje (angl. *Public Key Infrastructure – PKI*). Elektroninėje erdvėje ši technologija yra naudojama kaip būdas, susiejantis įvairius dokumentus, sandorius su tikruoju iniciatoriumi, išlaikant dokumento vientisumą [16].

Kitos viešojo rakto algoritmų klasės pradžia laikomi 1985 m., kai kriptografai V. Miller ir N. Koblitz pristatė elipsinių kreivių kriptosistemą (EKK). Šis algoritmas pagrįstas sveikųjų skaičių faktorizavimo problema. Tobulėjant technologijoms vis daugiau informacijos apdorojama ne personaliniais kompiuteriais, o mobiliaisiais įrenginiais. Tokiu atveju reikalinga naudoti kriptografinę sistemą su „mažais“ parametrais. Būtent tokiomis savybėmis pasižymi elipsinių kreivių kriptografinė sistema, kuri leidžia naudoti trumpesnius raktus aukštesnio lygio saugumui užtikrinti. Elipsinių kreivių kriptosistemoms užtenka „silpnesnės“ techninės įrangos. Dėl mažesnės atminties naudojimo EKK naudojamos delniniuose kompiuteriuose bei išmaniuosiuose telefonuose [17]. Labiausiai paplitęs EKK algoritmas – elipsinės kreivės skaitmeninio parašo algoritmas (angl. *Elliptic Curve Digital Signature Algorithm* – ECDSA).

1.3. E. PARAŠŲ ANALIZĖ

Informacinės technologijos šiais laikais vystosi labai sparčiai, todėl vis dažniau naudojami elektroniniai dokumentai. Kadangi e. dokumentai yra perduodami viešomis ryšio linijomis, kurios gali būti lengvai pažeidžiamos, pagrindinis tikslas naudojant e. dokumentus yra vartotojo pasitikėjimo užtikrinimas. Vienas e. dokumentų autentiškumo ir vientisumo užtikrinimo būdų – elektroninis parašas (toliau e. parašas) [18].

E. parašas – duomenys, kurie įterpiami, prijungiami ar logiškai susiejami su kitais duomenimis pastarųjų autentiškumui patvirtinti ir (ar) pasirašančiam asmeniui identifikuoti [19]. Tai viena iš pagrindinių asimetrinės kriptografijos dalių. Paprastai tariant, e. parašo technologija leidžia naudoti elektroninius dokumentus užtikrinant jų autentiškumą ir vientisumą. Tai bet kokio pavidalo koduota informacija, pagal kurią savo tapatybę vienas kitam gali patvirtinti du kompiuterių sistemos vartotojai [18]. Kitaip tariant, e. parašas yra traktuojamas kaip įrodymas, kad pranešimas yra gautas būtent iš nurodyto siuntėjo [19]. Taip pat, ši technologija leidžia parašo gavėjui, nešališkam trečiajai šaliai, įrodyti, kad parašas yra autentiškas. E. parašas atitinka tradicinį ranka rašytą parašą, tačiau viskas vyksta elektroninėje erdvėje [18].

E. parašas – tai prie duomenų bloko pridėti duomenys arba jų kriptografinė transformacija, leidžianti duomenų bloko gavėjui įrodyti duomenų bloko kilmę bei vientisumą ir apsisaugoti nuo klastojimo. Toks duomenų blokas gali būti atskiras duomenų bazės įrašas, vienas sakinys, visas dokumentas, atskiras failas ar net visas kompiuterio diskinio kaupiklio turinys [18].

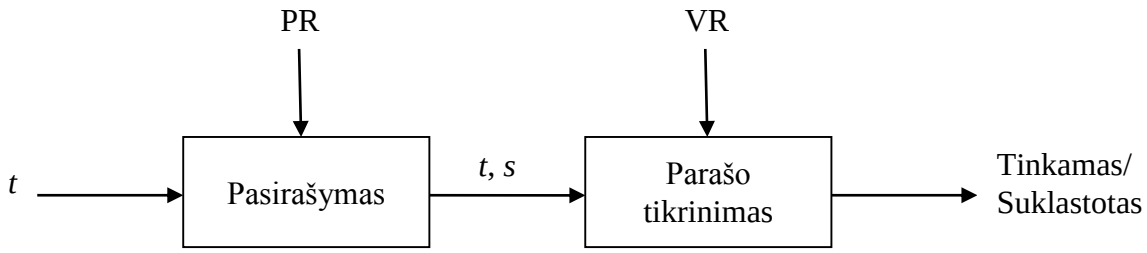
Tam, kad e. parašas būtų laikomas efektyviu, jis turi užtikrinti tiek pranešimo originalumą, tiek teikti teisingą informaciją apie patį siuntėją. Tai suteikia kopijavimo apsaugą, ir tuo pačiu apsaugo ir nuo galimybės gavėjui modifikuoti originalų pranešimą [19]. Taigi, e. parašas užtikrina šias svarbiausias ir pagrindines duomenų savybes [18]:

- vientisumą;
- autentiškumą;
- neišsiginamumą.

Dažnai duomenų vientisumas yra užtikrinamas panaudojant santraukos funkciją (žr. 1.1.1 poskyris). Kadangi, e. parašas grindžiamas asimetrine kriptografija, naudojant viešųjų raktų infrastruktūrą, todėl duomenų autentiškumą užtikrina atitinkamas pasirašiusiojo asmens viešasis raktas. Šis raktas leidžia patikrinti e. parašą. Neišsiginamumas – tai savybė, kuri užtikrina pasirašiusiojo asmens parašo tikrumą, t. y. pasirašęs asmuo negali išsiginti savo parašo. Tai užtikrinama privačiuoju raktu, kuris naudojamas parašo sudarymui [18].

1.3.1 E. PARAŠO SISTEMA

E. parašo sistema apibrėžia tris algoritmus: elektroninių (viešojo ir privačiojo) raktų generavimo, e. parašo formavimo ir parašo tikrinimo. Bendras šių algoritmų veikimo principas pavaizduotas 1.4 paveikslėlyje [20].



1.4 pav. E. parašo sistemos veikimas ([20], 33 p.)

Kaip matome 1.4 paveikslėlyje, pirmiausia yra pasirašomas pranešimas *t* privačiuoju raktu *PR* ir sudaromas e. parašas *s*. Naudojant e. parašą, *s* pridedamas prie pranešimo *t*. Galiausiai atliekamas parašo patikrinimas, panaudojant pasirašiusiojo asmens viešąjį raktą *VR*. Jei gautas tikrinimo funkcijos rezultatas yra teigiamas, tai reiškia, kad pranešimas yra autentiškas ir duomenys nebuvo pakeisti.

Patikima e. parašo schema, užtikrinanti vartotojo autentifikavimą turi tenkinti šias savybes [20]:

- Paraše turi būti siuntėjo tapatybę užtikrinanti informacija. Ši informacija yra susijusi su siuntėjo privačiuoju raktu (e. parašas formuojamas naudojant siuntėjo privatųjį raktą).
- Būtina, kad siuntėjo tapatybę patvirtinančią informaciją būtų galima sudaryti tik žinant siuntėjo privatųjį raktą.
- Gavėjas, turėdamas siuntėjo viešąjį raktą, bet nežinodamas siuntėjo privataus rakto, gali lengvai patikrinti, ar e. dokumentu pasirašyti buvo naudotas siuntėjo privatusis raktas.
- Nežinant siuntėjo privačiojo rakto, neįmanoma suformuoti parašo, kurio patikros funkcijos rezultatas būtų teigiamas.

Taigi, šios savybės užtikrina siunčiamo dokumento vientisumą ir vartotojo autentifikavimą. Tačiau lieka neužtikrintas e. parašo konfidencialumas. Tam yra naudojami šifravimo algoritmai: RSA, ElGamalio arba elipsinių kreivių e. parašo sistemos [21].

1.4. SERTIFIKATAI

Sertifikatas – elektroninis liudijimas, kuris susieja parašo tikrinimo duomenis su pasirašančiu asmeniu ir patvirtina arba leidžia nustatyti pasirašančio asmens tapatybę. Kvalifikuotam sertifikatui keliami papildomi reikalavimai – jį privalo sudaryti Vyriausybės ar jos įgaliotos institucijos nustatytus reikalavimus atitinkantis sertifikavimo paslaugų teikėjas [22].

Tam, kad turėtume pilnavertį viešojo rakto sertifikatą, sugeneruota raktų pora turi būti patvirtinta įgalioto patikimo atstovo. Šią funkciją atlieka sertifikavimo centras, kuris paprastai patikrina vartotojo tapatybę ir savo parašu patvirtina, kad konkretus viešasis raktas priklauso būtent šiam vartotojui. Šis patvirtinimas ir laikomas vartotojo viešojo rakto sertifikatu. Registrų centras žino tik viešąjį raktą, o privatus raktas išlieka slaptas ir žinomas tik pačiam vartotojui [18].

Esminės viešojo rakto infrastruktūros (angl. *Public Key Infrastructure – PKI*), kurioje viešojo rakto kodavimui naudojamas skaitmeninis sertifikatas, funkcijos yra šios [23]:

- autentifikavimas – naudojamas abiejų dalyvaujančių asmenų tapatybės įrodymui;
- integralumas – galimybė įsitikinti, kad siuntimo metu duomenys nebuvo pakeisti;
- konfidencialumas – siunčiama informacija gali būti koduojama siekiant užtikrinti jos saugumą nuo trečiųjų asmenų;
- neatskiriamumas – viešojo rakto infrastruktūra leidžia abiem dalyvaujančioms pusėms saugiai patekti į tinklą ir pasirašyti siunčiamus duomenis.

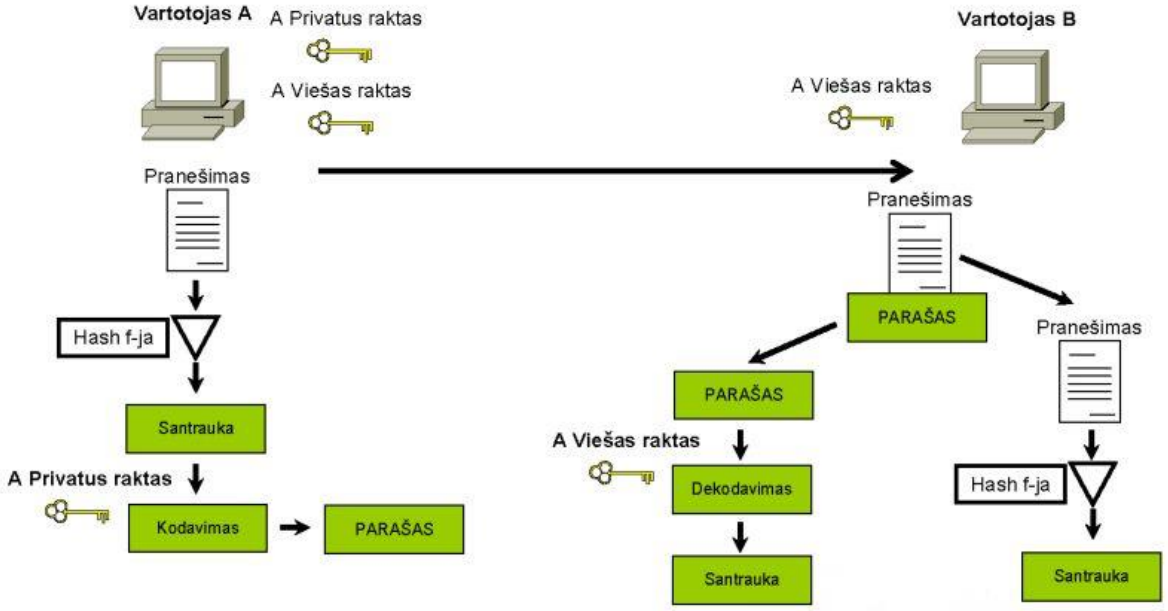
Skaitmeninį sertifikatą sudaro ir skiria sertifikavimo paslaugas teikianti organizacija (angl. *Certification Authority – CA*), pasirašanti jį savo privačiu raktu. Ji taip pat teikia sertifikatų duomenis parašo naudotojams elektroniniams parašams tikrinti [15].

Skaitmeninis sertifikatas paprastai susideda iš [15]:

- savininko viešo rakto;
- savininko vardo;
- viešo rakto galiojimo termino;
- skaitmeninį sertifikatą teikiančios organizacijos pavadinimo;
- skaitmeninio sertifikato serijinio numerio;
- sertifikatą teikiančios organizacijos skaitmeninio parašo.

Taigi, tam, kad sertifikavimo centras galėtų patvirtinti viešąjį raktą, jam reikia nusiųsti registravimo užklausą su nurodytais viešais parametrais: viešuoju raktu bei identifikacine asmens informacija [18].

Tam, kad geriau įsivaizduoti, kaip visame dokumento pasirašymo procese dalyvauja *CA* organizacija, panagrinėkime pavyzdį, kuris iliustruoja vieną iš galimų elektroninio parašo pritaikymo atvejų (žr. 1.5 pav.).



1.5 pav. Dokumento pasirašymas elektroniniu parašu [15]

Kaip matyti 1.5 paveikslėlyje, Vartotojas *A* siunčia pranešimą Vartotojui *B*. Vartotojo *A* programinė įranga siunčiamam pranešimui pritaiko santraukos funkciją ir sugeneruoja viena kryptimi užkoduotą teksto santrauką. Ši užšifruojama Vartotojo *A* privačiu raktu ir taip suformuojamas e. parašas. Vartotojas *B*, gavęs pranešimą, atskiria jo tekstą nuo parašo ir tekstui taiko santraukos funkciją, taip sugeneruodamas pranešimo santrauką. Tuo tarpu parašas iššifruojamas Vartotojo *A* viešuoju raktu ir taip gaunama teksto antroji santrauka. Vartotojo *B* programinė įranga palygina ar pirmoji santrauka (gauta panaudojus santraukos funkciją) ir antroji santrauka (iššifravus viešuoju raktu) yra identiškos. Jei jos identiškos, vadinasi, Vartotojo *A* pranešimas pasiekė Vartotoją *B* nepakeistas ir Vartotojas *B* yra tikras, kad pranešimą jam siuntė Vartotojas *A* [15].

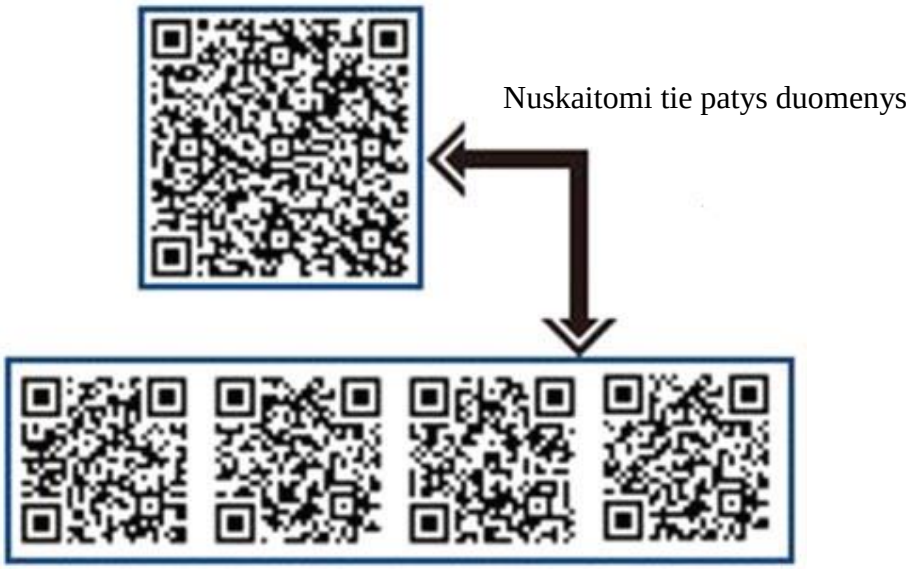
Paprasčiausias e. parašo taikymo atvejis yra naudojant X.509 sertifikatą, kur gavėjo rekvizitus bei viešąjį raktą pasirašo išdavėjas ir parašas lieka sertifikate. Taip yra užtikrinamas sertifikato vientisumas. Tuo tarpu šakninio išdavėjo sertifikate yra pasirašomi nuosavi rekvizitai bei viešasis raktas ir gaunamas taip vadinamas *self-signed* sertifikatas. Toks sertifikatas nėra saugus, kadangi jį galima lengvai falsifikuoti. Todėl jo autentiškumas yra užtikrinamas šiek tiek kitokiu būdu – imama iš patikimo šaltinio ir įdiegiama vartotojo kompiuteryje. Faktiškai taip pareiškiamas pasitikėjimas tiek išdavėju, tiek jo išduodamais sertifikatais [18].

1.5. QR KODO ANALIZĖ

QR kodas – tai matricos kodas (arba dviejų dimensijų barkodas). Paprasčiau tariant, tai įprastas kvadratas tik su kodu viduje, matomas vis dažniau mūsų aplinkoje (laikraščiuose, žurnaluose, plakatuose, kavinėse, interneto svetainėse ir t. t.). Raidės "QR" yra santrumpa angliško žodžių junginio *Quick Response*, reiškiančio *g*reitą reagavimą. Pavadinimas atspindi pagrindinę šio kodo panaudojimo savybę – jis buvo specialiai sukurtas taip, kad jo iššifravimas būtu atliekamas dideliu greičiu. Šis grafinis arba taškinis kodas yra sukurtas 1994 metais japonų firmos *Denso-Vawe* ir yra šifruojamas specialių aplikacijų, kurias galime įdiegti į telefoną, nešiojamąjį kompiuterį ar elektroninę užrašų knygutę, turinčius fotokamerą. Šie kodai iš pradžių buvo sukurti naudoti automobilių pramonėje, tačiau ilgainiui QR kodas tapo mobiliųjų etikečių (žymių) standartu Azijoje. Japonijoje daiktų žymėjimai pasinaudojus QR kodo mobiliosiomis etiketėmis yra atliekami daugiau kaip 50 milijonų kartų per dieną ir šis skaičius nepaliaujamai didėja [1].

1.5.1 QR KODO SANDARA

QR kodas susideda iš juodų modulių sudėliotų į kvadrato formą baltame fone. Dažniausiai tai nespalvotas, iš juodos ir baltos spalvų sudarytas kvadratas, kuriame yra atitinkamai išdėstyta kitų mažų kvadratėlių iš kurių susidaro tinklelis su užšifruota informacija [1] (žr. 1.6 pav.).



1.6 pav. QR kodo sandara [1]

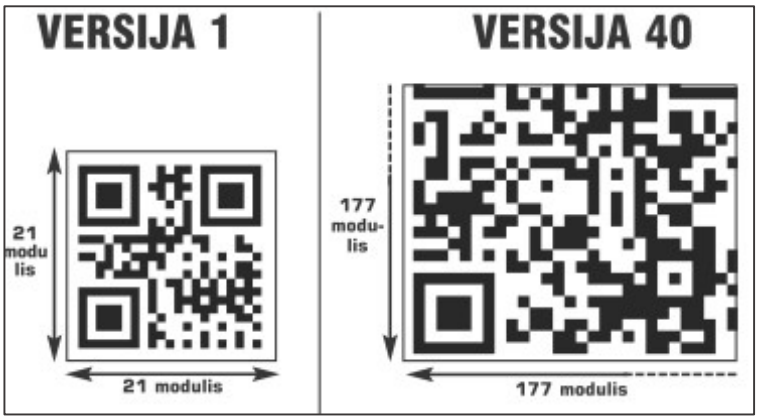
Užšifruojama informacija gali būti tekstas, internetinė nuoroda ar bet kokia kita informacija. Pačios QR kodo etiketės paprasčiausiai tik talpina duomenis. Taigi, QR kodų paskirtis – saugoti informaciją. QR kodo etiketėse yra talpindama įvairi informacija, kurią QR skeneriai, mobilūs telefonai

su foto kamera bei išmanieji telefonai, nesunkiai gali nuskaityti. QR kodo etiketės gali savyje talpinti daugiau informacijos nei įprasti vienos dimensijos barkodai, tačiau jos turi ir savo technologines ribas. Todėl QR kodas yra apibrėžiamas, kaip fiksuoto dydžio duomenų failas [1].

QR kodą galima panaudoti ant įvairių daiktų, todėl jis gali būti ypač praktiškai panaudotas įvairiems marketingo sprendimams, kaip pavyzdžiui, vienai iš dažniausiai kasdieniniame gyvenime sutinkamų – reklamos sferai [1] (žr. 1.7 pav.).

1.7 pav. QR kodo panaudojimas šiuolaikiniame marketinge [1]

Kodas gali būti ir kitų pasirinktų spalvų, tačiau svarbu, kad jos būtų kontrastingos. Vienas toks mažas kvadratėlis yra vadinamas moduliu. Kiek modulių bus patalpinta kvadrate, tiek informacijos bus galima į tą kodą ir surašyti. Tačiau modulių negalime pridėti tiek, kiek norime. Tam yra nustatytos QR kodo versijos (nuo 1 iki 40): mažiausias QR kodas (versija 1) ir didžiausias QR kodas (versija 40) (žr. 1.8 pav.). Mažiausioje versijoje gali tilpti 441 modulis (t. y. 21 x 21). Kiekviena versija didėjanti vienetu yra ekvivalenti elementų padauginimui keturiais vienetais, iš abiejų matmenų pusių. Pavyzdžiui, 2 versijos matmenys būtų 25 x 25 (1-os versijos (4+) 21 x 21 (+4)). Didžiausia, 40-ta versija – iki 31329 modulių, kas yra 177 x 177. Jei informacijos kiekis nėra didelis, kodas gali būti ir itin mažas, tačiau kuo informacijos daugiau – tuo jis didesnis, o jo tinklelis yra smulkesnis [24].



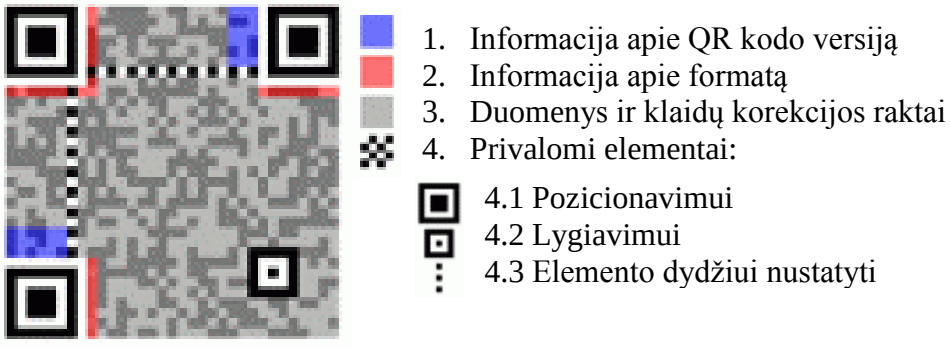
1.8 pav. QR kodo dydis [1]

Taigi, galime išskirti QR kodo privalumus prieš standartinį barkodą [24]:

- Didelė informacijos talpa.
- Mažam informacijos kiekiui gali būti naudojamas daug mažesnio dydžio barkodas nei standartinis brūkšninis, kadangi jame informacija yra saugoma tiek horizontaliai, tiek vertikaliai.
- QR kodas turi savyje ir klaidų taisymo savybę. Jei šis kodas kažkiek pažeistas ar purvinas – vis vien bus įmanoma atkoduoti jame esančią informaciją.
- Skaitymas gali būti iš bet kurios pusės. 360 laipsnių kampu.
- Gali būti saugoma net dvejetainio tipo informacija.

1.5.2 QR KODO VEIKIMO PRINCIPAI

Pirmiausia programa pagal tris kodo matricos kampus (1.9 paveikslėlyje pagal legendą: 4.1 Pozicionavimas) , kurie yra sujungti punktyrine linija (4.3 Elementų dydžiui nustatyti) atpažįsta kodą ir tiksliai nustato, kokio dydžio yra kodo paveikslėlis. To padaryti būtų neįmanoma, jei aplink tris orientacinius kvadratėlius nebūtų balto apvado. Taip pat, yra ir ketvirtasis, mažesnis, kvadratėlis, kitaip dar vadinamas žymelėmis, kuris padeda nustatyti kodo poziciją paveikslėlyje. Jie yra skirti lygiavimui arba kitaip sakant, padeda naudojamai aplikacijai orientuotis erdvėje. Nes žymeles fotografuojant ne tiesiai iš priekio, jos išsikreipia. Žymelių skaičius priklauso nuo paveikslėlio dydžio, kuo didesnis paveikslėlis – tuo daugiau žymelių [24].



1.9 pav. QR kodo sudedamosios dalys [24]

Paveikslėlio pilka spalva pažymėti laukai yra skirti klaidoms taisyti. Duomenų zonoje yra saugomi (užkoduojami) QR brūkšninio kodo duomenys. Duomenys yra užkoduojami ir paverčiami dvejetainiais skaičiais (0 ir 1). Dvejetainiai skaičiai paverčiami į juodas ir baltas ląsteles. Duomenų zonoje esantys duomenys naudoja *Reed-Solomon* klaidų taisymo metodus. Šio algoritmo dėka, galime gauti informaciją net ir tada, jei paveikslėlis yra pažeistas. Tačiau, koks procentas kodinių žodžių (kodinis žodis – 8 bitai) gali būti atkurtas, priklauso nuo korekcijos lygmens [24].

Lygmenys [24]:

L – (angl. *Low*, liet. žemas) gali atkurti iki 7 procentų kodinių žodžių.

M – (angl. *Medium*, liet. vidutinis) gali atkurti iki 15 procentų kodinių žodžių.

Q – (angl. *Quality*, liet. kokybinis) gali atkurti iki 25 procentų kodinių žodžių.

H – (angl. *High*, liet. aukštas) gali atkurti iki 30 procentų kodinių žodžių.

Pavyzdžiui, koduojant 200 simbolių ir pažeidimo atveju, norint atgauti bent 50 simbolių, kas sudaro 25% (200/50 = 1/4), reikėtų pridėti Q lygmenį – 25proc. koduojamų duomenų.

Raudona spalva paveikslėlyje nusako kodo formatą, kuris programai leidžia suprasti, kokią informaciją teks atvaizduoti. Pavyzdžiui, tai galėtų būti paveikslėlis, video, SMS žinutė, puslapio adresas.

Mėlyna spalva pažymėtas kodas pasako apie QR versiją. Yra net 40 skirtingų QR kodų variantų, todėl programa gali tiksliai sužinoti, kokia versija yra naudojama.

Yra sukurti keli standartai, kad viso pasaulio sugeneruoti kodai būtų lengvai perskaitomi [24]:

- JIS X 0510;
- ISO/IEC 18004:2000;
- ISO/IEC 18004:2006.

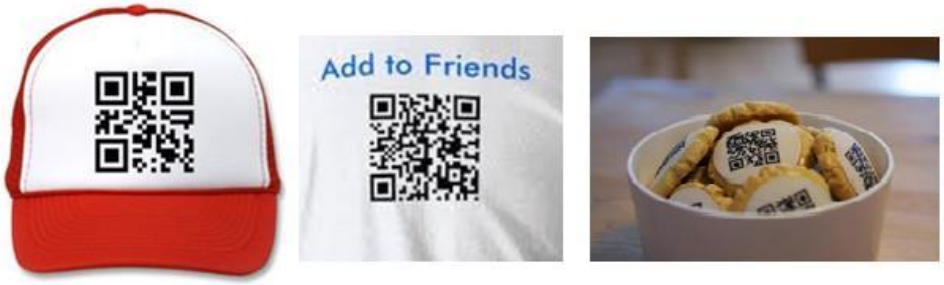
Taip pat reikia paminėti, kad QR kodai turi tris tipus [24]:

1) Statinis kodas – toks QR, kurio užkoduotą informaciją negalima keisti.

2) Dinaminis kodas – toks QR, kurio kodas išlieka toks pats, bet galima keisti užšifruotą nukreipimo nuorodą.
3) Mišrus kodas – tai toks QR kodas, kurį sugeneravę turime sąlygas pakeisti, pagal mūsų poreikius. (pvz.: toks pats kodas gali leisti patekti į 3 skirtingus interneto puslapius tuo pačiu laiko momentu, atsižvelgiant į tai, kokią OS palaiko telefonas).

1.5.3 QR KODO PANAUDOJIMAS

QR kodu galima užkoduoti tekstą, įvairias nuorodas, trumposios žinutės tekstą, elektroninio pašto adresą, kontaktinę informaciją ir pan.



1.10 pav. QR kodo panaudojimo pavyzdžiai

Taip pat QR kodą galima naudoti ant įvairių daiktų, ne tik kompiuteryje ar telefone. Jis gali atsirasti ant marškinėlių, rankinių, sagių, puodelių, kepurių (žr. 1.10 pav.). QR kodas gali pakeisti ne tik įprastą tekstą – jis gali būti ir paveikslėlis ar vaizdo klipas. QR kodas gali slėpti mobiliojo telefono numerį, kuriuo būtų skambinama nuskaičius kodą. QR kodu galima užšifruoti netgi trumposios žinutės tekstą. Tokios žinutės rašymas ir skaitymas užtrunka šiek tiek ilgiau, tačiau tam tikrai situacijai, tai gali būti tikrai įdomus bendravimo būdas. Japonijoje šie kodai ypač populiarūs, jie klijuojami net ant antkapių, kad kiekvienas galėtų sužinoti, ką yra nuveikę čia palaidoti žmonės [25].

Dar viena QR kodo panaudojimas – vizitinėje kortelėje. Dažnai yra ieškoma originalios vizitinės kortelės formos. Štai sprendimas – QR kodas – elegantiškas, minimalistinis ir originalus sprendimas. Šiuolaikinis verslo pasaulis ir jame susiklosčiusi verslo etika vizitines korteles padarė kone būtinu atributu, kuris pradeda kiekvieną dalykinį pokalbį. Nors vizitinės kortelės ir padeda lengviau išsaugoti kontaktus, tačiau šiuolaikinės technologijos ir gyvenimo tempas lemia tai, kad įprastos vizitinės kortelės nebegali tinkamai užtikrinti savo funkcijos, nes tradicinės vizitinės kortelės stokoja interaktyvumo ir norint pasinaudoti jose pateikta informacija būtina perrašinėti pateiktą informaciją į mobilųjį telefoną ar kompiuterio adresų knygą [25].

Šią problemą išsprendžia QR kodas. Vizitinė kortelė su atspausdintu QR kodu Jūsų verslo partneriui suteikia ypač spartų būdą su jumis susisiekti. Pakanka nukreipti išmaniojo telefono kamerą į vizitinę kortelę ir visi kontaktai akimirksniu perkeliami į telefono atminti, suteikiama galimybė skambinti nerenkant numerio ranka, susisiekti el. paštu ir t. t.

Taigi, QR kodu gali būti užšifruota [25]:

- SMS žinutės – tai QR kode saugomas tekstas su telefono numeriu, į kurį bus siunčiama žinutė.

  Nuskaičius QR kodą, turėtų atsidaryti nauja paruošta žinutė, kurią vartotojas galėtų kurti ir siųsti. Kaip pavyzdžiui, norint sukurti nuorodą į numerį "*12345*", kodavimas būtų: "*SMS: 12345*".
- Tekstai – įvairūs tekstai, kurie gali užimti vidutiniškai apie 2 vnt. A4 formato lapų.
- Telefonų numeriai – tik atpažinus programai, kad tai telefono numeris, iš karto bus tam numeriui ir skambinama.

  Koduojant tam tikrą telefono numerį, yra patartina įtraukti tam tikrą priešdėlį (t. y. šalies kodą), tam, kad telefono numeris būtų prieinamas tarptautiniu mastu. Pavyzdžiui, koduoti JAV telefono numerį, *212-555-1212,* reikia koduoti "*tel: +12125551212*".
- El. paštas ir el. laiškai – kurie gali būti saugomi kartu su laiško tema. Patogu naudoti rengiant įvairius konkursus, renginius.

  Norėdami koduoti e-pašto adresą , kaip pavyzdžiui, *sean@example.com,* galima tiesiog koduoti "*sean@example.com*". Tačiau siekiant užtikrinti, kad tekstas būtų pripažįstamas kaip elektroninio pašto adresas, yra patartina sukurti tinkamą e-pašto adreso kreipinį su priekyje nurodytu žodžiu *mailto*: "*mailto: sean@example.com*". Nuskaičius tokį QR kodą yra atveriamas naujas tuščias laiškas su nurodytu adresatu.
- Vizitinės kortelės – tai kode išsaugoti kontaktiniai duomenys, kuriuos programa iškart įtraukia į adresų knygutę.

  Tarkime, kad turime *vCard* (standartinis failo formatas elektroninei vizitinės kortelės versijai) formato vizitinės kortelės kontaktinę informaciją kaip tekstą.

  "*NTT DoCoMo*" yra populiarinamas kompaktiškas "*MeCard*" formatas kontaktinės informacijos kodavimui. Pavyzdžiui, koduojant informaciją: vardas: *Seanas Owen* , adresas: *76 9. Avenue, 4-as aukštas , Niujorkas, NY 10011*, telefono numeris: *212 555 1212*, elektroninio pašto adresas: *srowen@example.com*, brūkšninio kodo informacija atrodytų taip:

"*MECARD:N:Owen,Sean;ADR:76 9th Avenue, 4th Floor, New York, NY10011;TEL:+12125551212;EMAIL:srowen@example.com*".

- Interneto adresai – programai atpažinus, kad tai nuoroda į internetinį puslapį, iškart įjungia naudojamą interneto naršyklę ir pradeda užkrauti numatytą puslapį.

  Tai dažniausiai pasitaikantis brūkšninių kodų taikymo būdas (kai yra koduojamas URL tekstas). Pavyzdžiui, norint užkoduoti interneto adresą: *http://google.com/m*, yra tiesiog koduojamas tikslus URL tekstas iš brūkšninio kodo: "*http://google.com/m*". Reikia nepamiršti įtraukti protokolą ("*http://*"), tam, kad užtikrinti, kad tekstas būtų pripažintas kaip interneto adresas.
- E. parašas – yra sukuriama nuoroda, kuri talpinama QR kode. Nuskenavus kodą yra pasiekiamas atitinkamas internetinis adresas, kuriame patalpintas dokumentas, pasirašytas e. parašu.

  Kiekvienas elektroninis parašas, sukurtas elektroniniu būdu (naudojant *E-Sign*) gauna unikalų anspaudą, kuris yra saugomas internetiniame adrese, kaip pavyzdžiui:

  Tarkime, kad QR kode yra talpinama ši nuoroda:

  http://app.e-sign.co.uk/link?d=wihf8934hf874gf7384gf8heuh&i=78erfrr7yf8yr8r88fy8ef89er

  Nuskaičius QR kodą, įvyksta automatinis nukreipimas į e. parašą atitinkamame puslapyje, kur galima patikrinti parašą ir visą informaciją.
- *Wi-Fi* tinklai – dažnai kavinėse naudojami kodai, kuriuose nurodoma ten veikiančio *Wi-Fi* tinklo prisijungimo duomenys.

  Wi-Fi konfigūracijai yra naudojama "*MECARD*" sintaksė. Skanuojant tokį kodą, pirmiausia yra atpažįstamas vartotojas, o paskui atitinkamai konfigūruojamas įrenginio *Wi-Fi*. Kol kas šia galimybe gali pasinaudoti tik *Android* savininkai.
- Geografiniai duomenys – tai QR kode užšifruotas žemėlapis, kuriame gali būti nurodyta tiksli vieta.

  Pavyzdžiui, užkoduoti *Google* rastą *New York* biurą, kurio koordinatės yra 40,71872 laipsnių šiaurės platumos, 73,98905 laipsnių vakarų ilgumos, taške, esančiame 100 metrų virš biuro, koduotė būtų: "*GEO: 40,71872, -73.98905,100*".
- Paveikslėliai – QR kodu užšifruoti paveikslėliai.
- Vaizdo įrašai – įvairaus pobūdžio vaizdo klipai, ypač tinkantys reklamuoti įvairius gaminius.

Kaip jau buvo minėta, QR kode gali būti saugomi dvejetainiai duomenys. Todėl ne išimtis yra vaizdo ir garso duomenys, kurie QR kode bus dvejetainio formato. Tačiau, duomenų kiekis, kuris gali

būti saugomas tokio tipo kode yra daugiausiai 3 KB (kai naudojama 40 versija su L korekcijos lygmeniu). Galime įsivaizduoti, kad garsai ar vaizdai, išreikšti su minėtu duomenų kiekiu, yra gana riboti. Tuo atveju, kai QR kodas yra nuskaitomas naudojant mobiliojo telefono fotoaparato funkciją, nuskaitomas duomenų kiekis siekia apie 271 baitą (kai naudojama 10 versija su L korekcijos lygmeniu). Šis skaičius gali svyruoti, priklausomai nuo turimo telefono gamintojo ir jo modelio tipo (fotoaparato funkcijos pajėgumo). Todėl būtų protinga manyti, kad praktiniais tikslais atvaizdai ir garsai negali būti saugomi tiesiogiai QR kode, kai yra naudojamas telefonas duomenims skaityti [24].

Kol kas QR kodų šifravimas nėra populiarus, tačiau natūralu, kad populiarėjant pačiam kodui ir daugėjant jo panaudojimo paskirčių, atsiranda ir poreikis juo perduoti užslaptintas žinutes. Taip pat yra galimybė parsisiųsti tam tikras pritaikytas programėles (pvz.: *Android* operacines sistemas turintiems telefonams – *QR Droid*), su kurių pagalba galima sukurti šifruotą pranešimą. Kitas asmuo, norėdamas perskaityti pranešimą, turi jį nuskaityti ir atkoduoti su ta pačia programa, kuri po nuskaitymo paprašo įvesti slaptažodį, kuris veikia kaip šifro raktas [24].

Lietuvoje vieni iš pirmųjų tokią paslaugą pristatė įmone *Fur LT, UAB*, kuri sukūrė *QRbook.eu* platformą skirtą kurti ir administruoti vizitines korteles internete, kurios susiejamos su įprastomis popierinėmis vizitinėmis kortelėmis ant jų spausdinant QR kodą [1].

Taigi, turint fotokamerą ir specialią QR *Scanner* programėlę išmaniajame telefone galima skenuoti QR kodą. Tai suteikia galimybę "sekti" produktą, jį identifikuoti, reguliuoti laiką, dokumentų valdymą, palaikyti bendrosios rinkodaros principus ir pan. Šie kodai intensyviai pradedami naudoti gamyboje ir žmonių kasdienėje veikloje. Jais žymimos įvairios produktų dalys, o daugelio miestų gyventojai juo naudojasi kasdien, jei laukia autobuso. Jiems tereikia nuskenuoti šį kodą ir jie greitai gauna informaciją, už kelių minučių atvažiuos jiems reikiamas autobusas [1].

1.5.4 QR KODO GENERAVIMO GALIMYBĖS

Standartiškai QR skirtas saugoti skaitinius, tekstinius arba mišrius duomenis (vardan išbaigtumo – taip pat ir Kanji/ Kana, t. y. Japoniškus rašmenis), tačiau tolimesnį jų interpretavimą riboja tik fantazija. Vieni populiaresnių duomenų formatų: URL nuorodos (taip pat ir URI/URN), *vCard* formato kontaktų kortelės, *vCalendar* formato kalendoriaus įrašai, SMS bei MMS žinutės, el. pašto laiškai, *PayPal* greitasis apmokėjimas ir t. t. [24].

Internete yra ypač platus pasirinkimas QR kodo generavimui. Išvardinsime kelias, atsitiktinai pasirinktas, internetinės svetainės, QR kodui kurti:

- [26] – formatai: URL, tekstai, vizitinė kortelė, SMS, skambutis nurodytu telefono numeriu, geografiniai duomenys, el. laiškas, „Wi-Fi“, kalendoriaus įvykis/ renginys.
  Leidžia iš karto matyti pakeistą variantą. Pirmame svetainės plane yra laukelis tekstui įvesti, kuris verčiamas į QR kodą. Galima iškart pasirinkti, kur ir kam QR kodą naudosime bei parsisiųsti jį į kompiuterį.
- [27] – formatai : URL, *Youtube* video, geografiniai duomenys (*GoogleMap*), *Twitter* profilis arba būsenos atnaujinimas, nuoroda į *Facebook* puslapį, *Facebook* „Like“ mygtukas ir daug kitų susijusių su socialiniais tinklais, renginiais, taip pat su „Wi-Fi“ tinklais, netgi su *PayPal* pirkimo nuoroda.
  Galima matyti realiai keičiamą QR kodą, taip pat galima keisti spalvas pagal pateiktą spalvų paletę.
- [28] – platus formatų pasirinkimas, ne tik leidžia kurti, bet kartu ir skenuoja.
- [29] – leidžia sukurti: SMS, geografines vietas, skambučius, „Wi-Fi“, kontakto informaciją, el. laišką. Galima pasirinkti kodo spalvas, leidžia įterpti paveiksliuką į tekstą.
- [30] – leidžia įkelti failą CSV (angl. *comma-separated values*) formatu. Sukuriami atskiri QR kodai kiekvienai duomenų failo eilutei. Ekrane rodo tik du sugeneruotus QR kodus, o pilnai sugeneruotas failas, talpinantis visus sukurtus QR kodus, parsisiunčiamas ZIP formatu.

Dar keletas populiaresnių QR nuskaitymo programų skirtų išmaniesiems telefonams:

- Android – *BeeTag*, *ixMat Scanner*, *Kaywa Reader*, *QuickMark*;
- Apple – *BeeTag*, *i-nigma Reader*, *UpCode*, *QRdeCODE*, *QuickMark*;
- Bada OS – *BeeTag*;
- BlackBerry – *BeeTag*, *i-nigma Reader*, *UpCode*;
- Java palaikantiems – *BeeTag*, *SnapMaze*;
- Palm OS – *BeeTag*;
- Symbian – *BeeTag*, *UpCode*, *QuickMark*;
- Windows – *BeeTag*, *i-nigma Reader*, *Upcode*, *QuickMark*;

QR kodo naudojimas nėra varžomas jokios licencijos, bei jo naudojimas yra aiškiai apibrėžtas ir aprašytas kaip ISO (angl. *International Standard Organization*) standartas [24].

2. MEDŽIAGOS IR TYRIMŲ METODAI

Kaip jau buvo minėta 1.5.4 skyrelyje, QR kodas turi daug panaudojimo būdų, nuo duomenų apie save platinimo iki įvairių marketingo sprendimo verslui. Šiame darbe, kaip pavyzdys bus nagrinėjamas prekės kilmės QR kodas.

2.1. PREKĖS KILMĖS DOKUMENTO DUOMENYS

Produktai, kurie yra perkami parduotuvėse ar prekybos centruose, keliauja ilgą kelią nuo gamintojo per kelis tiekėjus iki kol yra pasiekiama pardavimo vieta, prieinama vartotojui. Dėl šio, daug variacijų turinčio, gamybinio prekių keliavimo proceso, norint galutiniam vartotojui pristatyti tik kokybiškus produktus, gamintojas privalo vykdyti prekių žymėjimą. Tai paprasčiausiai užtikrina prekių autentiškumą [31].

Prieš įsigydamas prekę, dažnas vartotojas nori žinoti tam tikrus prekės parametrus (pavyzdžiui, prekės gamybos vietą, kilmės šalį, kainą ir kt.). Todėl, norint suteikti vartotojams išsamią informaciją apie produktą, QR kodas yra talpinamas ant prekės įpakavimo. Nuskanavus tokį QR kodą vartotojas iškart gauna paaiškinimus apie prekę ir jos privalumus, naudojimo instrukciją, aptarnavimo centrų kontaktinius duomenis arba tiesiog bendrą informaciją apie paslaugas teikiančią kompaniją [31].

Šiame darbe pasirinkto atveju metu, tarkime, kad vartotojas nori įsigyti badmintono raketę. Prieš įsigydamas prekę, pirkėjas nori matyti pagrindinius prekės parametrus: pavadinimą, firmą, kilmės šalį, prekės kodą, svorį ir, žinoma, kainą. Raketės prekės kilmės duomenis galime matyti 2.1 lentelėje.

2.1 lentelė. Duomenų failas

Pavadinimas	Firma	Kilmės šalis	Prekės kodas	Svoris	Kaina
DUORA 10 Racket	YONEX	Japonija	DUO10	88 g	200 €
VOLTRIC Z-FORCE II Racket	BABOLAT	Japonija	VTZF2	83 g	150 €
X-Feel Origin Power Racket	BABOLAT	Prancūzija	XFEOP	85 g	100 €

Kadangi įvestos tik trys prekės, duomenų failo dydis CSV (angl. *comma-separated values*) formatu nėra didelis – 232 baitai, t. y. 1856 bitai. Tačiau toks failas bus pakankamas išsiaiškinti ir išanalizuoti QR kodo kūrimo, generavimo ir patikrinimo principus, siekiant sukurti saugią ir patikimą kriptografinę sistemą.

Toliau kiekvienai duomenų failo eilutei yra sukuriami RSA ir ECDSA raktai, apskaičiuojamos santraukos funkcijos ir e. parašai. Taigi, sekančiuose poskyriuose išnagrinėsime tiriamų kriptografinių sistemų parametrų dydžius, reikalingus saugiam naudojimui. Taip pat, apsirašysime RSA ir ECDSA algoritmus, kurių pagrindu bus atliekamas tyrimas.

2.2. RSA IR ECDSA ANALIZĖ

2.2.1 RSA RAKTŲ GENERAVIMAS IR E. PARAŠO FORMAVIMAS

RSA sistemos pagrindas yra trys tarpusavyje susiję skaičiai. Du iš jų yra visiems žinomi ir sudaro viešąjį raktą, trečiasis yra slaptas ir žinomas tiktai rakto savininkui. Sistema aprašyta remiantis [32] šaltiniu.

Raktų generavimo algoritmas:

1) Generuojami du dideli, maždaug vienodo dydžio, pirminiai skaičiai p ir q.
2) Apskaičiuojama:

$$n = p \cdot q, \tag{2.1}$$

$$\varphi(n) = (p-1)(q-1), \tag{2.2}$$

kur φ – Oilerio funkcijos reikšmė.

3) Atsitiktinai parenkamas skaičius e, toks, kad $1 < e < \varphi$ ir $DBD(e, \varphi) = 1$.
4) Randamas toks d, kad

$$(e \cdot d) mod\ \varphi = 1. \tag{2.3}$$

5) Gauname, kad viešasis raktas $VR = (n, e)$, o privatusis $PR = d$.

E. parašo formavimo algoritmas:

Tarkime, kad vartotojas A pasirašo pranešimą M.

1) Apskaičiuojama pranešimo santrauka:

$$m = H(M), \tag{2.4}$$

kur m atvaizduojama į sveikąjį skaičių iš atkarpos $[0, n-1]$.

2) Apskaičiuojamas parašas:

$$s = m^d\ mod\ n. \tag{2.5}$$

3) Pranešimas siunčiamas kartu su parašu.

E. parašo tikrinimo algoritmas:

Vartotojas B tikrina, ar gautą pranešimą pasirašė A.

1) B pasiima viešąjį A raktą $VR_A = (n, e)$;
2) Apskaičiuoja:

$$m_1 = s^e\ mod\ n. \tag{2.6}$$

3) Apskaičiuojama pranešimo santrauka:

$$m_2 = H(M), \tag{2.7}$$

kur m_2 atvaizduojama į sveikąjį skaičių iš atkarpos $[0, n-1]$.

4) Jei apskaičiuota pranešimo santrauka m_2 sutampa su rezultatu m_1, apskaičiuotu panaudojant gautą parašą bei vartotojo *A* viešąjį raktą, parašas tikras.

Taigi, RSA sistemos saugumo lygis priklauso nuo pasirinktų viešųjų parametrų. Jei *n* (žr. 2.1 formulę) yra pakankamo dydžio, tai potencialus kenkėjas negali perrinkti visų įmanomų tekstogramų *t* reikšmių, kol ras atitinkamą šifrogramą *c*. Taip pat, kuo *n* mažesnis, tuo jį lengviau išskaidyti priminiais dalikliais. Tačiau tam reikalinga efektyvus algoritmas, kuris gali išskaidyti duotą skaičių atitinkamais pirminiais dalikliais. Tokiu atveju RSA sistema taptų nesaugi, kadangi žinant *n* skaidinį $n = p \cdot q$ pasidaro įmanoma iš viešojo rakto *VR* apskaičiuoti privatųjį *PR*:

$$d = e^{-1} mod\ \varphi(n) \qquad \text{(žr. 2.3 formulę)},$$

$$\varphi(pq) = (p-1)(q-1) \qquad \text{(žr. 2.1, 2.2 formules)}.$$

Todėl atsižvelgiant į šias savybes ir esamas technines galimybes, norint užtikrinti saugų sistemos panaudojimą, reikia pasirinkti pakankamai saugius sisteminius parametrus. Jų pasirinkimo galimybės plačiau bus analizuojamos 2.2.3 poskyryje.

2.2.2 ELIPSINIŲ KREIVIŲ RAKTŲ GENERAVIMAS IR E. PARAŠO FORMAVIMAS

Sistema aprašyta remiantis [21], [33] šaltiniais.

Elipsinės kreivės kriptosistemos sisteminius parametrus sudaro elipsinės kreivės modulis *p*, elipsinės kreivės taškai *a* ir *b*, ciklinį pogrupį generuojantis taškas *Q* ir to taško eilė *n*.

Taigi, ECDSA sisteminiai parametrai:

- Elipsinė kreivė $E_p(a, b)$, kur modulis *p* turi būti didelis pirminis skaičius.
- Šios kreivės taškas Q, kuris turi generuoti ciklinį pogrupį, kurio eilė n būtų pirminis skaičius.
- Elipsinės kreivės grupės eilė *n*, kuri tenkina nelygybes $n > 2^{160}$ bei $n > 4\sqrt{p}$.

Raktų generavimo algoritmas:

1) Pasirenkamas atsitiktinis skaičius $x, 1 < x < n - 1$.
2) Apskaičiuojama

$$P = xQ. \tag{2.8}$$

3) Viešasis raktas $VR = P$, privatusis raktas $PR = x$.

E. parašo formavimo algoritmas:

1) Pasirenkamas atsitiktinis skaičius $k, 1 \leq k \leq n - 1$.
2) Apskaičiuojami

$$kQ = (x_1, y_1), \tag{2.9}$$

$$r = x_1 \; mod \; n. \tag{2.10}$$

Jei $r = 0$, e. parašas formuojamas iš naujo.

3) Apskaičiuojama

$$k^{-1} \; mod \; n. \tag{2.11}$$

4) Apskaičiuojama e. dokumento *t* santrauka

$$h = H(t). \tag{2.12}$$

5) Apskaičiuojama

$$s = k^{-1}(h + xr) \; mod \; n. \tag{2.13}$$

Jei $s = 0$, e. parašas formuojamas iš naujo.

6) E. dokumento *t* e. parašą sudaro pora (*r*, *s*).

E. parašo tikrinimo algoritmas:

1) Apskaičiuojama gauto e. dokumento *t* santrauka $h = H(t)$.
2) Apskaičiuojama

$$w = s^{-1}\ mod\ n. \tag{2.14}$$

3) Apskaičiuojami

$$u_1 = hw\ mod\ n, \tag{2.15}$$

$$u_2 = rw\ mod\ n. \tag{2.16}$$

4) Apskaičiuojamas

$$X = u_1 Q + u_2 P = (x_1, y_1). \tag{2.17}$$

Jei $X = O$, t. y. X yra be galo nutolęs taškas, tai e. parašas – suklastotas.

Jei $X \neq O$, apskaičiuojamas

$$v = x_1\ mod\ n. \tag{2.18}$$

5) E. parašas yra tikras tada ir tik tada, kai $v = r$.

Aprašytos e. parašo schemos korektiškumo įrodymas:

Iš lygybės

$$s = k^{-1}(h + xr)\ mod\ n \tag{2.19}$$

išreiškę *k*, gauname:

$$k = s^{-1}(h + xr)\ mod\ n = \tag{2.20}$$

$$= s^{-1}h + s^{-1}xr\ mod\ n = \tag{2.21}$$

$$= wh + wxr\ mod\ n = \tag{2.22}$$

$$= u_1 + u_2 x\ mod\ n. \tag{2.23}$$

Tuomet

$$kQ = (u_1 + u_2 x)Q = \tag{2.24}$$

$$= u_1 Q + u_2 xQ = \tag{2.25}$$

$$= u_1 Q + u_2 P \tag{2.26}$$

ir

$$v = r. \tag{2.27}$$

Taigi, ECDSA parametrai yra parenkami taip, kad esami diskretinio logaritmo problemos sprendimo algoritmai būtų nepajėgūs išspręsti problemos per priimtą laiko tarpą [21]. Sekančiame poskyryje išnagrinėsime, kokie ECDSA sistemos parametrai yra laikomi pakankamai saugiais naudoti bei palyginsime juos su prieš tai nagrinėtos RSA sistemos parametrais.

2.2.3 RSA IR ELIPSINIŲ KREIVIŲ E. PARAŠO SISTEMOS REIKALAVIMAI

Kriptografinių sistemų saugumo lygis priklauso nuo pasirinktų viešųjų parametrų [18]. Todėl šioje dalyje palyginsime šiuo metu nustatytus saugius abiejų darbe nagrinėjamų kriptografinių sistemų parametrus. Atlikus analizę galėsime nustatyti, kuri kriptografinė sistema užtikrina pakankamą saugumą su mažesnio dydžio raktų ir e. parašo ilgiais.

Pagal išnagrinėtus šaltinius, e. parašo kriptosistemoms elipsinių kreivių pagrindu (ECDSA) saugumui užtikrinti reikia žymiai trumpesnių raktų nei, pavyzdžiui, RSA sistemai [34]. Tai reiškia, kad, tarkime, norint užtikrinti 80 bitų saugumo lygį (reikės įvykdyti 2^{80} operacijų, norint atrasti privatųjį raktą) RSA formuojamas viešasis raktas turės būti ne mažesnis kaip 1024 bitai, o tuo tarpu ECDSA viešasis raktas tenkins 192 bitų dydį [17]. Pagal nustatytas rekomendacijas, lyginant RSA ir ECDSA raktų ilgius, tam pačiam saugumo lygiui užtikrinti ECDSA naudoja žymiai mažesnio ilgio raktus [34] (žr. 2.2 lentelę). Tai užtikrina greitesnį apskaičiavimą ir mažesnės talpos reikalavimus [17].

2.2 lentelė. RSA ir ECDSA ekvivalentūs raktų ilgiai [17], [34]

RSA raktų ilgis (bitais)	ECDSA raktų ilgis (bitais)
1024	192
2048	224
3072	256

2.2 lentelėje matome pateiktus RSA ir ECDSA tą patį saugumą užtikrinančius raktų dydžius. Naudojamų raktų dydis turi būti saugus, t. y. raktai turi būti pakankamai saugaus dydžio, kad užtikrintų šifruojamų duomenų apsaugą. Tam yra atliekami įvairūs tyrimai, nagrinėjamos galimos atakos ir įsilaužimo galimybės. Šiuo metu 1024 bitų RSA raktų ilgis ir atitinkamai 192 bitų ECDSA raktų ilgis yra laikomi nesaugiais. Tuo tarpu, pakankamai saugus RSA raktų ilgis yra 2048 bitai, o atitinkamas ECDSA raktų ilgis siekia 224 bitus. Taip pat yra atliekama nemažai projektų ir tyrimų ([17], [34]), kur naudojami ir didesni (RSA – 3072 bitai, ECDSA – 256 bitai) saugumą užtikrinantys raktų ilgiai [34].

Tačiau lyginant e. parašo patikrinimo greitį, ECDSA yra lėtesnis negu RSA. Tarkime, 2048 bitų parašo patikrinimas užtrunka ~0,16 milisekundžių, o atitinkamai 244 bitų ECDSA parašo patikrinimas – 8.33 milisekundės (kitus kriptografinių sistemų palyginimus galima rasti [35]). Todėl lyginant visas charakteristikas ECDSA yra greitesnis už RSA raktų generavimo ir parašo kūrimo procedūrose, tačiau RSA veikia greičiau parašo patikrinimo etape (žr. 2.3 lentelę).

2.3 lentelė. RSA ir ECDSA charakteristikų palyginimas [17]

	Saugumas	**Sudėtingumas**	**Panaudojimo sfera**	**Raktų generavimas**	**Vykdymo laikas**	**Tikrinimas**	**Pasirašymas**
RSA	Aukšta	Sveikųjų skaičių faktorizavimas	Personalinis kompiuteris, nešiojamasis kompiuteris, superkompiuteris	Greitas	Lėtas	Greitas	Greitas
ECDSA	Aukšta	Diskretinis algoritmas	Lengvasvoriai prietaisai	Greitesnis	Greitas	Lėtas	Greitas

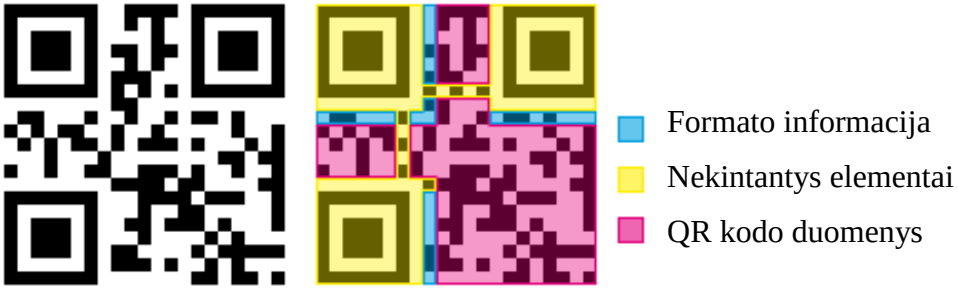
Kita vertus, parašo dydis ECDSA yra 4*t* bitų, kur *t* yra apsaugos lygis matuojamas bitais, tai yra apie 320 bitų, 80 bitų apsaugos lygiui [36]. ECDSA e. parašas susideda iš dviejų komponenčių (*r*, *s*), kurios kiekviena yra 32 baitų ilgio. Tuo tarpu RSA e. parašo dydis yra 2048 bitai [35].

Lyginant abiejų kriptografinių sistemų sertifikatų dydžius, ECDSA taip pat pranoksta RSA. ECDSA reikalingas sertifikatų dydis yra 577 bitų dydžio, o tuo tarpu RSA reikalingi 2048 bitai [37].

Toliau detaliai aprašysime RSA ir ECDSA raktų generavimo, e. parašo formatavimo ir tikrinimo algoritmus tam, kad geriau suprasti algoritmų schemų veikimus ir išsiaiškinti esminius skirtumus. Tačiau iš atliktos analizės galime pastebėti, kad ECDSA kriptografiniai parametrai yra žymiai trumpesni lyginant su RSA kriptosistemos parametrais. Todėl atsižvelgiant į tai, kad QR kodas turi ribotą informacijos tūrį, yra tikėtina, kad naudojant elipsinių kreivių kriptosistemą, naudingos informacijos kiekis QR kode bus didesnis. Kitaip tariant, galėsime sukurti mažesnį QR kodą, kuris talpins didesnį kiekį informacijos, lyginant su RSA kriptosistemos sugeneruotu kodu. Taip pat, kadangi QR kodas turi ribotą informacijos tūrį, tai QR kodo e. parašo sertifikatas turi būti patalpintas serveryje.

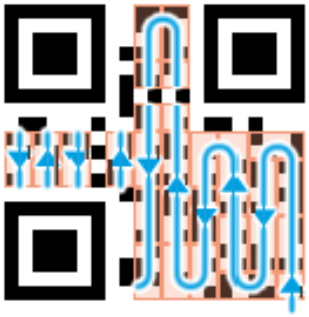
2.3. QR KODO DUOMENŲ TALPOS GALIMYBIŲ ANALIZĖ

QR kodą sudaro trys dalys: formato informacija, nekintantys elementai ir QR kodo duomenys (žr. 2.1 pav.).



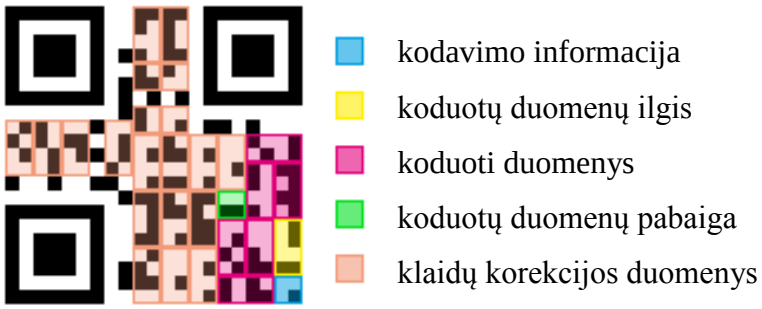
2.1 pav. QR kodo struktūra [38]

Skanuojant QR kodą visų pirma nustatoma formato informacija: klaidų korekcijos lygis bei kaukės šablonas. Formato informacija yra dubliuota, t. y. QR kode saugomos dvi jos kopijos, taip pat ji saugoma nuo klaidų BCH (angl. *Bose, Ray-Chaudhuri, Hocquenghem*) kodu. Egzistuoja 8 skirtingi kaukių šablonai, 6×6 elementų dydžio. Šablonu yra uždengiamas visas kodas (išskyrus formato informaciją ir pozicionavimo elementus), o kodo elementai sutampantys su juodais kaukės elementais – invertuojami. Kodas apeinamas pradedant nuo apatinio dešiniojo kampo, zigzago tvarka, pradedant eiti į viršų (apėjimo būdas labai primena *gyvatėlę*) (žr. 2.2 pav.). Bitų tvarka apeinant: pirmasis – svarbiausias, paskutinis – mažiausiai svarbus, kitaip tariant *MSb-to-LSb*/ *right-to-left*, dviejų elementų pločio, atsižvelgiant į apėjimo tvarką [24].



2.2 pav. Duomenų apėjimas [38]

Pirmieji keturi bitai (0-3) – kodavimo informacija (skaitinis, mišrus, dvejetainis, Kanji/ Kana ir t. t.), tolimesni aštuoni bitai (4-11) nurodo koduojamos žinutės ilgį. Visa tolimesnė informacija žingsniuojant po 8 bitus eilės tvarka: užkoduota žinutė, keturių bitų žinutės pabaigos simbolis, bei klaidų korekcijos informacija pabaigoje (žr. 2.3 pav.).



2.3 pav. Duomenų struktūra [38]

Jei pirmieji keturi bitai indikuoja skaitinį kodavimą, tuomet į 10 bitų sutalpinami 3 skaitmenys. Naudojant mišrų (angl. *alphanumeric*) kodavimą – į 11 bitų telpa 2 simboliai, tačiau šio kodavimo atveju nėra išsaugomas didžiosios/ mažosios raidės požymis. Norint naudoti platesnį simbolių rinkinį, naudojamas dvejetainis (Base256) kodavimas *(*angl. *byte encoding*) kurio atveju vienas simbolis užima 8 bitus [24].

Taigi, informacijos formatas, kuris yra suformatuotas, išsaugo maskavimo šabloną ir klaidų korekcijos lygmenį panaudotą simboliui. Maskavimo šablonas naudojamas, kad panaikintų panašius į QR duomenis, kuriuos QR kodų skaitytuvas gali sumaišyti su pozicionavimo žymėmis. Tai gali būti įvairūs kontrastingi ornamentai, reklamose ar QR kodo fone, kuriuos skaitytuvas gali netinkamai suprasti. Tokia maskavimo sistema naudoja privalomus elementus, kaip tarp pozicionavimo elementų esanti „punktyrinė linija“. Kadangi QR kodo pagrindas yra apibrėžtas kaip tinklelis, tai skaitytuvas skenuoja tiek kartų, kol galiausiai aptinka QR kodo paveikslėlį. Moduliai, padedantys maskavimo šablonui yra invertuojami. Tada apsaugant nuo kodo pažeidimų yra formatuojama informacija su Rydo ir Solomono (angl. *Reed-Solomon – RS*) kodu. Tokia maskavimo šablono ir klaidų korekcijos lygmens „pora“ – įtraukiama į kiekvieną QR simbolį [24].

Po kiekvieno indikatoriaus, kuris parenka kodavimo metodą, eina laukas, kuris nusako kiek simbolių yra užkoduota tame metode. Bitų skaičius lauke, priklauso nuo kodavimo ir simbolio versijos [24].

Efektyviam duomenų saugojimui, QR kodas naudoja keturis standartinius kodavimo rėžimus [24]:

- Skaitmeninį – maksimaliai talpina 7089 simboliai;
- Skaitmeninį-raidinį – maksimaliai talpina 4296 simboliai (4,2 KB);
- Dvejetainį – maksimaliai talpina 2953 baitai (2,9 KB);
- *Kanji (*logografinį) – maksimali talpa 1817 simboliai.

Įprasta QR kodo etiketė gali talpinti informaciją iš raidžių ir skaičių, kurios maksimalus kiekis gali būti 4296 simboliai. Dažniausiai kode būna užkoduota informacija iki 3 KB (t. y. daugiau nei 7 tūkstančiai skaičių arba daugiau nei 4 tūkstančiai skaičių ir raidžių) [24].

2.4 lentelė. Kodavimo būdas ir simbolio versijos [24]

Kodavimas	**Versija 1–9**	**Versija 10–26**	**Versija 27–40**
Skaitinis	10	12	14
Raidinis-skaitinis	9	11	13
Dvejetainis	8	16	16
Kanji	8	10	12

2.4 lentelėje matome, kaip bitų skaičius lauke priklauso nuo kodavimo ir simbolio versijos. Norint plačiau paanalizuoti kiekvienos iš keturių išvardintų QR kodo versijų maksimalius talpinamų duomenų kiekius žiūrėti 1 priedą.

Taigi, raidinis-skaitinis kodavimo būdas kaupia žinutę kompaktiškiau, negu baito būdas gali, bet negali sukaupti mažųjų raidžių ir turi tiktai ribotą pasirinkimą skyrybos ženklų, kurie yra pakankami daugumos tinklo adresų. Šio kodavimo būdo simbolių kodus galime matyti 2.5 lentelėje.

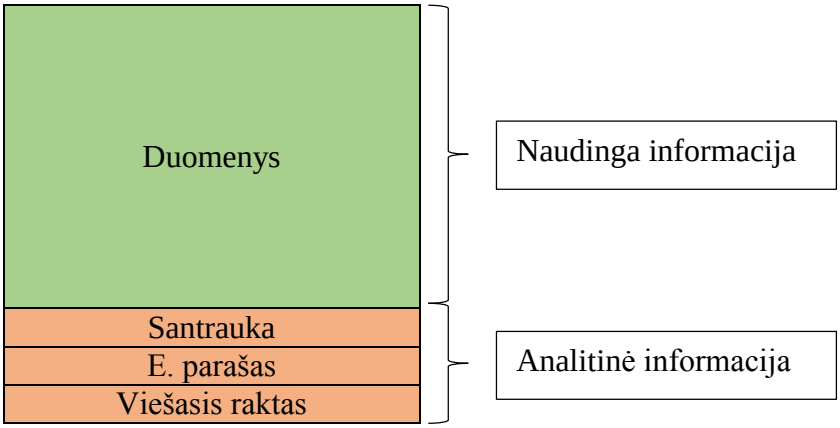
2.5 lentelė. Raidinio-skaitinio kodavimo būdo simbolių kodai [24]

Kodas	Simbolis	Kodas	Simbolis	Kodas	Simbolis	Kodas	Simbolis	Kodas	Simbolis
00	0	09	9	18	I	27	R	36	SP
01	1	10	A	19	J	28	S	37	$
02	2	11	B	20	K	29	T	38	%
03	3	12	C	21	L	30	U	39	*
04	4	13	D	22	M	31	V	40	+
05	5	14	E	23	N	32	W	41	–
06	6	15	F	24	O	33	X	42	.
07	7	16	G	25	P	34	Y	43	/
08	8	17	H	26	Q	35	Z	44	:

Todėl, norint naudoti platesnį simbolių rinkinį, naudojamas dvejetainis kodavimo būdas. Atsižvelgiant į 2.1 poskyryje suformuotą duomenų failą, mūsų duomenys talpina tiek mažąsias, tiek didžiąsias raides, todėl šis faktorius nulems, kad automatiškai QR kodo generavimui bus parinktas dvejetainis kodavimo būdas.

Darbo metu bus naudojamas automatinis QR kodo generavimo būdas, t. y. QR kodas bus sugeneruotas internetiniame puslapyje, pasinaudojant nemokama QR kodo generavimo programėle. Tokiu atveju analizuojamo prekės kilmės QR kodo duomenų failui bus automatiškai parinktas kodavimo būdas (pagal duomenis – dvejetainis) ir versija. Tačiau svarbu paminėti, kad norint užtikrinti duomenų failo saugumą ir vientisumą, prie koduojamos informacijos reikia priskirti kriptografinius duomenis. informacija bus sudaryta iš dviejų tipų duomenų.

Prekės kilmės QR kodo informacinė struktūra vaizduojama 2.4 paveikslėlyje. Paveikslėlis atvaizduoja QR kodo duomenų išsidėstymą, t. y. QR kodas turi talpinti naudingą informaciją (prekės kilmės duomenis) ir analitinę (kriptografinę) informaciją (santraukos funkciją, e. parašą ir viešąjį raktą).



2.4 pav. QR kodo informacinė struktūra

Kadangi darbui bus naudojamas dvejetainis kodavimo būdas, tai QR kodo maksimali talpa, kai duomenys yra koduojami dvejetainiu būdu, gali siekti 2953 baitus t. y. 23624 bitus (2,9 KB).

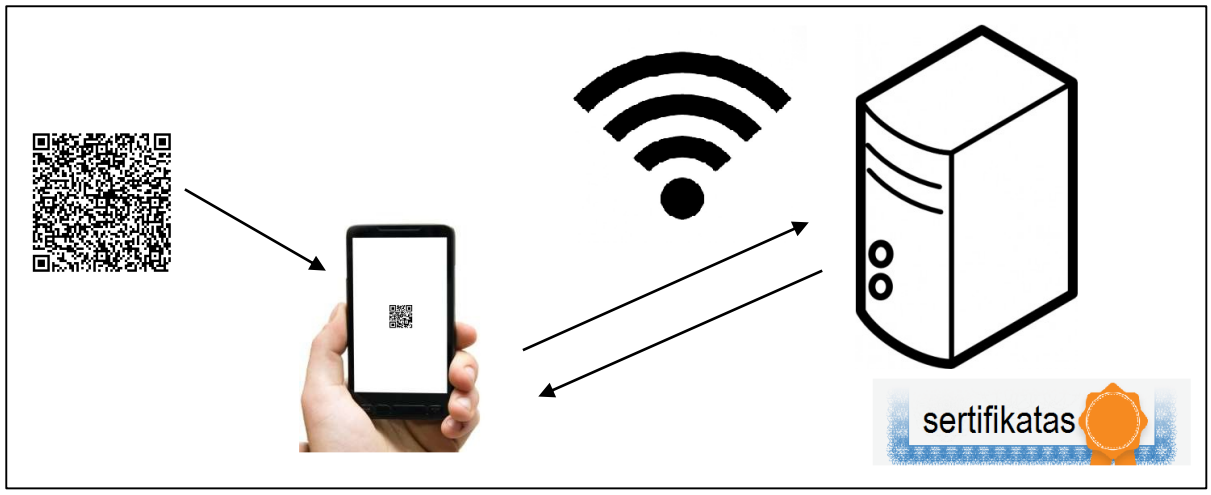
Taip pat, galime pastebėti, kad dėl riboto QR kodo talpinamos informacijos tūrio, formuojamas QR kodo e. parašo sertifikatas turi būti patalpintas serveryje. Taigi, jo į analitinę informaciją netraukiame.

2.4. SIŪLOMA PREKĖS KILMĖS QR KODŲ INFRASTRUKTŪRA

Sukonstravus prekių kilmės dokumento duomenų failą, išanalizavus pasirinktų kriptografinių sistemų algoritmus ir nusistačius jų parametrų dydžius bei pasirinkus QR kodo informacijos kodavimo būdą ir nusistačius šio būdo maksimaliai talpinamos informacijos kiekį, toliau galime apsibrėžti viso darbo infrastruktūros schemą. Schema padės vaizdžiai ir aiškiai matyti bei geriau įsivaizduoti atliekamo tyrimo metodikos pritaikymą.

Taigi, siūloma prekės kilmės QR kodų infrastruktūra susideda iš šių dalių (žr. 2.5 pav.):

1) Prekės duomenų nuskaitymas.
2) Santraukos funkcijos apskaičiavimas telefone.
3) Apskaičiuotų duomenų palyginimas su kriptografiniais duomenimis, esančiais santraukos funkcijos reikšmėje.
4) Parašo patikrinimas.
5) Sertifikato atsiuntimas iš serverio.
6) Sertifikato patikrinimas.



2.5 pav. Siūloma prekės kilmės QR kodų infrastruktūra

Kaip matome 2.5 paveikslėlyje pirmiausia yra nuskaitomi prekės duomenys QR kodo pavidalu, su tam tikra telefone įdiegta QR kodo nuskaitymo programėle (žr. 1.4.3 poskyrį). Tuomet telefone yra apskaičiuojama santraukos funkcijos reikšmė ir palyginama su kriptografiniuose duomenyse esančia santraukos reikšme. Jei santraukos reikšmės sutampa, procesas tęsiamas. Toliau yra patikrinamas parašas ir bevieliu tinklu kreipiamasi į *debesyse* saugomą serverį dėl sertifikato atsiuntimo. Vykdant viešojo rakto sertifikato peržiūrą yra patikrinamas sertifikatas ir patvirtinamas duomenų autentiškumas ir vientisumas.

3. TYRIMŲ REZULTATAI IR JŲ APTARIMAS

Tyrimas buvo atliekamas naudojant OpenSSL paketą [39], parsisiuntus vykdomąjį failą [40]. Darbas su OpenSSL vykdomas naudojant Windows komandų eilutę (*cmd.exe*). Paleidus komandų eilutės langą, aktyvus katalogas turi būti tas, kuriame įkeliamas *openssl.exe*.

3.1. SANTRAUKOS SKAIČIAVIMAS

OpenSSL pakete failų santraukų skaičiavimai atliekami naudojant komandą *dgst* ir nurodant, kuri santraukos funkcija bus naudojama. Dėl vienodumo tiek RSA, tiek ECDSA naudosime tą pačią santraukos reikšmę.

Kiekvienai raketei sukuriame po atskirą tekstinį duomenų failą: *rakete1.txt, rakete2.txt, rakete3.txt*. Atitinkamai kiekvienam iš šių tekstinių failų yra saugoma informacija apie raketę (žr. 5 lentelė).

Skaičiuodami kiekvieno failo SHA1 santrauką vykdome komandą:

openssl dgst –sha1 rakete.txt,

kur *sha1* – naudojimui pasirinkta santraukos funkcija,

rakete.txt – duomenų failas, kuriam kuriama santraukos funkcija.

Gauti rezultatai (šešioliktainiu formatu):

SHA1 (*rakete1.txt*) = 47004505ac7cfb1e7ea182d9b9d79754df962f7f

SHA1 (*rakete2.txt*) = 5b9b71018c773b1bf554ea2902c8d5c27fa3a296

SHA1 (*rakete3.txt*) = 8f5784a909ed6b19256b8424a3fe7244af68fa3a

Gautas SHA1 santraukos funkcijos reikšmes talpiname 2.1 skyrelyje aprašytuose atitinkamuose RSA ir ECDSA duomenų failuose. Toliau, kiekvienai sistemai, turime susigeneruoti raktų poras bei e. parašus.

3.2. RAKTŲ POROS GENERAVIMAS

3.2.1 RSA RAKTŲ POROS GENERAVIMAS

OpenSSL komandos formuojamos kiekvienam raketės duomenų failui atskirai, t. y. kiekvienas failas gauna savo atitinkamą raktų porą. Kadangi tokiu atveju keičiasi tik failo pavadinimas, todėl visas komandas vykdysime vienam failui, o rezultatus pateiksime visų gautų failų. Iš gautų rezultatų atskirsime viešojo rakto dalis ir jas patalpinsime prie 2.1 skyrelyje aprašytų atitinkamų duomenų failų.

Taigi, pirmiausia komandų eilutėje rašome:

openssl genrsa –out raktaiRSA.key 2048

Ši komanda sugeneruoja 2048 bitų ilgio RSA raktų porą ir išsaugo ją faile *raktaiRSA.key*.

Programos rodomas rezultatas:

Loading 'screen' into random state - done

Generating RSA private key, 2048 bit long modulus

.....+++

.....+++

e is 65537 (0x10001)

Į failą *failasRSAraktai.txt* yra išsaugomi sugeneruoti raktai:

openssl rsa –noout –text –in raktaiRSA.key –out failasRSAraktai.txt

Atliekame automatinį parametrų patikrinimą:

openssl rsa –check –in raktaiRSA.key

Gautas programos rezultatas parodo, kad visi sugeneruotų raktų parametrai yra teisingi ir raktai yra tinkami naudoti. Taip pat, kaip patvirtinimas, yra parodoma privatus raktas *Base64* formatu:

RSA key ok

writing RSA key

-----BEGIN RSA PRIVATE KEY-----
MIIEpgIBAAKCAQEAzqqvOhVLZPBDWFNh2+zOQqwJscicNgxcy4slGaznv6xj7KQN
D35YPAx/PM1l0tAHAHvUlKWysSCxhN7g03LVABFwDK/1955l6EdG6Z0y5iQ4hDyk
Ca9YdcNh06whW7wjAJRDPynoGPgH4lwV5pOihJcvgxYkuVaCnH5uXMTdtCY8+jqi
…
HG58zU3pAoGBAOegDtEIkboyU7D5VQSot1kkMb5Q+WfEZAPaxx6gWrM5Gsb1F0JD
lLJNxuUYdcgjMEQgKq1pBBeohUgl5h5enCzVJ2nkxAZP5GSsTc7RpIO3Wdkz4gz0
YRP1OFvDHt6nEn8K+uwDLDFsy32fU+XqmDVP5heC6wk3aijd7elWSGXu
-----END RSA PRIVATE KEY-----

Taigi, buvo sugeneruoti 2048 bitų ilgio RSA raktai kiekvienam duomenų failui, išsaugoti atitinkamuose *.txt* failuose ir patikrintas parametrų korektiškumas automatine OpenSSL komanda. Žemiau pateikiamos viešojo ir privačiojo rakto dalys:

1) Pirmojo duomenų failo *rakete1.txt* viešas ir privatusis raktai:

$VR_{rakete1\_RSA} = (n, e) =$

(28456806715674589738760617871942209957630842060416971849718845650032022520100512885823586291152101754701907734635064913778278259751739321085902909054234677101480369892586322255379349128009659455953786014666465599519426089670451901405976030256363393203441232257779261511303268749857637112464624490186849668782833173355224767873321801121085858932102249361181696292472790586043476944269640941529051233729095485554375762739058531982622693264082179602022048150510719084909742910071213367441665191844

035935879561283382677603000481964129054255827006285603813172969717404057761366364104138127269628747768237482766609918985757, 65537)

$PR_{rakete1\_RSA} = d =$

181477979871159723229854894785819409673799692374555951364725140598911817939118503450416831566703097919917944637238893765561917674950630166895846938969092013588716611938392867946897156843487452425589893721559566310925799688386184478884106410815983035375105101273156268194290266170300737463640230795941129729277261890105919334120148463235240900028392847767338633429633604666854164974557577722181218105689114305380092780651217698836320749767577499305597321003696524989393950554203452483599221984587640544512242157478809039129438153571284198481965101094216828309279501835842134015209345601131306439606740328518492091921

2) Antrojo duomenų failo *rakete2.txt* viešas ir privatusis raktai:

$VR_{rakete2\_RSA} = (n, e) =$

(238030189428960920076964197591674905555825150834346886935941958435973942222485155467673944685062319536432039083685337627780296619547399733892816566633664228213722889375121928457747856567914022295886983030498608304582842646828391506932066998789280730386251261861746051190946341994925621829787706491699871063499724161210701531686902773271676924466225324267063700910909443640815545346807365819395837160413467093305905501352138548004436601327349636174978101421738686703752718887707297637668411672679215270445454481608168789441349944551084902964151214776175636474082753153605234811418196172016768140301575284952648852970, 65537)

$PR_{rakete2\_RSA} = d =$

145319833212416732123829508672055551004686209469964345227794584722855080370841716212476955014633389794653132059406656450937904512637304282356309926416826211682854975461006812632908883365714145384755696446790054730528771795203484635913437183401100314379423825294876501124570152578832858614700641232271900470292566238620032335343940990399433604288302689374502277769325047070561453976498728367086783900997324519485018941140141584719108266064864606699056914335489809513562145936329020138713907036519052513903621152636065633717462197510105162167000093941692305301514576685623811623623833586927679200585173048759769033164

3) Trečiojo duomenų failo *rakete3.txt* viešas ir privatusis raktai:

$VR_{rakete3\_RSA} = (n, e) =$

(26289379540748687835710034848033358487742825554755503581859787401172604742771564347142767318702868578658931890341226585602645208641875686191458959575045149123721732316412558123767323580998406671143534239321602103922045747782970225207836992071222816998197693559141101535869375566400369547899745431360948482773437515127221345931353399112252668336194993001272322216716074004836161112184107110753205540793785805427034154197175241687421064543508018922554854638408625410021221231203054033270712855514841169092439282487246733006183675859252824780282088024716139909098116949498566121913576586386257577628969335647594871321 0, 65537)

$PR_{rakete3\_RSA} = d =$

16654045354429761400051474995587239945917861491322218627768631361436791905422082747155443043938340094605839072452152282748142593148675601013332951228728038454148880183415477907509467523876754348930621038709659498428850470889811478095637096599187599269331425660235609082879730005801976631828642310051001696130297997703785436390712990639140903195393295370879389586243963141971983101424682411498931702667271319253685130318274110924191493200769498298301368137479125539045458469576354619509763980240817501329595490028765363740408544316968015460264269444934707750473045956597003271531513252284571041353677133087539165285 3

Gauti viešieji raktų parametrai gali būti žinomi visiems, todėl juos patalpiname prie suformuoto RSA sistemos duomenų failo (žr. 2.1 poskyris).

3.2.2 ELIPSINIŲ KREIVIŲ RAKTŲ POROS GENERAVIMAS

Elipsinių kreivių kriptosistemos (EKK) raktų kūrimas naudojant OpenSSL paketą yra labai panašus į RSA kriptosistemos. Paprasčiausiai dirbant su kreivės parametrais naudojamas komandos eilutės trumpinys *ecparam*, o dirbant su raktais – *ec*.

Prieš sukuriant raktą susirandame visų kreivių sąrašą:

openssl ecparam -list_curves

Įvykdžius programą gauname kreivių sąrašą:

secp112r1 : SECG/WTLS curve over a 112 bit prime field
secp112r2 : SECG curve over a 112 bit prime field
secp128r1 : SECG curve over a 128 bit prime field
…
secp224k1 : SECG curve over a 224 bit prime field
…
prime256v1: X9.62/SECG curve over a 256 bit prime field
…
sect571r1 : NIST/SECG curve over a 571 bit binary field
c2pnb163v1: X9.62 curve over a 163 bit binary field
…
c2pnb368w1: X9.62 curve over a 368 bit binary field
c2tnb431r1: X9.62 curve over a 431 bit binary field
wap-wsg-idm-ecid-wtls1: WTLS curve over a 113 bit binary field
…
wap-wsg-idm-ecid-wtls12: WTLS curvs over a 224 bit prime field
…

Iš pateikto sąrašo išsirenkame vieną kreivę, kurios pagrindu atliksime tolesnius skaičiavimus. Kadangi nustatytas saugus naudojamas elipsinių kreivių raktų dydis yra 224 bitai, todėl tolimesniems skaičiavimams pasirinksime *secp224k1* kreivę.

Naudojant pasirinktą kreivę, susikuriame kreivės parametrų failą *secp224k1.pem*:

openssl ecparam -name secp224k1 -out secp224k1.pem

Tuomet iš gauto parametrų failo *secp224k1.pem* generuojame viešojo ir privačiojo raktų poros failą *secp224k1ECDSA1-key.pem*:

openssl ecparam -in secp224k1.pem -genkey -noout -out secp224k1ECDSA1-key.pem

Peržiūrime gautus raktus:

openssl ec -noout -text -in secp224k1ECDSA1-key.pem -out failasECraktai1.txt

Taigi, buvo sugeneruoti 224 bitų ilgio EKK raktai kiekvienam duomenų failui ir išsaugoti atitinkamuose *.txt* failuose. Žemiau pateikiamos viešojo ir privačiojo rakto dalys:

1) Pirmojo duomenų failo *rakete1.txt* EKK viešas ir privatusis raktai:

$VR_{rakete1\_EKK}$ =
34776816508812742001885402842146525632965384113816372712661636774725963027105599306424985257914339559261785591221061882380370115866700 35

$PR_{rakete1\_EKK}$ =
639955914570566169973328267818513589824206129968164222353865946870 8

2) Antrojo duomenų failo *rakete2.txt* viešas ir privatusis raktai:

$VR_{rakete2\_EKK}$ =
31268355583390573886192992520877750462574003624236127332236536085110497912049295531010517275707939277174960589286593016306162389613121 33

$PR_{rakete2\_EKK}$ =
22983457051877394044831482351181461763177388392208942999208786808986

3) Trečiojo duomenų failo *rakete3.txt* viešas ir privatusis raktai:

$VR_{rakete3\_EKK}$ =
34409356947963054713893685610640558123160936442774562028289368395959678519805011019874911856739452651701240092334474739276872666459807 72

$PR_{rakete3\_EKK}$ =
18244143097613047733579674834020723256054683336978806601863769675096

Atlikus ECDSA raktų generavimo žingsnius galime akivaizdžiai matyti, jog sukurti raktai yra daug trumpesni nei RSA sistemos (žr. 3.2.1 poskyrį). Gautus viešus parametrus (*VR*) pridedame prie suformuoto ECDSA duomenų failo (žr. 2.1 poskyrį).

3.3. E. PARAŠO FORMAVIMAS IR TIKRINIMAS

3.3.1 RSA E. PARAŠO FORMAVIMAS IR TIKRINIMAS

Šiame skyrelyje formuosime RSA e. parašą ir patikrinsime jo korektiškumą, naudodami OpenSSL paketo komandą *dgst*. Veiksmai bus atliekami remiantis 2.2.1 poskyryje aprašytais algoritmais. Gautas korektiškas parašas bus pridedamas prie 2.1 poskyryje suformuoto RSA duomenų failo.

E. parašo formavimas:

Pasirinkto failo e. parašo formavimui su SHA1 santraukos funkcija vykdome komandą:

openssl dgst –sha1 –sign raktaiRSA.key –out signRSA.bin rakete1.txt

kur *rakete1.txt* – failas, kuriam formuojamas e. parašas,

raktaiRSA.key – pasirašančiojo asmens RSA privataus rakto komplektas,

signRSA.bin – e. parašo failas dvejetainiu formatu.

Dvejetainio e. parašo formato *signRSA.bin* konvertavimui į Base64 formatą *signRSA.txt* naudosime OpenSSL komandą *base64:*

openssl base64 –e –in signRSA.bin –out signRSA.txt

1) Gautas RSA e. parašas Base64 formatu failui *rakete1.txt*:

   HercAGHrOG4NNIvVATHAQiAob4LzKoBKgmxwhHlV0mBYwUGcsU9M3QRVmJGtEbd3NgZTuvO7Xie
   mhcg/eG8xUE2dsxNw37Uwqeq+N6tE932jO96vRzCxbNoll3aL66l9F6ORXhGp/wTtfaSMytPpKDs5iKrI/OjC
   yAJlsPufs3Gg4kB1EiwuMTZ6gGHN603+/O9Ns9Fmx/w1flwkjP4tuXBaWphNBQAkqSc46AbPzQXILl2qs6gj
   lTk/Ro+L+mbfE8eNOipQk0EVpiTQtYjhBnGozIFFobKkU4HWfh57cbGxbGiq5fuFGogxUlXrSsFHGnQ9l8U
   RgWG+vs7vpwMIEA==

2) Gautas RSA e. parašas Base64 formatu failui *rakete2.txt*:

   G6H1Xep7sugDXog9sF1SspLF95C3pCNenoAW7v8X6xQbLQpXBgmtqfcSls/abqVKd2GZBzN3/OvsPLEWe
   ej4+O41Y2lfUAnn/+RImR8TYZOZIzISb1sripnVJGKnEch+PraJZ76VZ7+0DVO+ZX5XGqzZquZ43/cg8FmIa
   inimukFTKqTUgUsWzzrnNTTNpw50hQfn4v+frAPv8jC7FNP9GTFiV9IgIOpGnWjZYuwjIjNxU03mTSQl5p
   xR1iWToY2LBDFniSZjSGsN/F2ZRWL3JT/3x/7E5UWxB3GY9cy/GMtGr9wPY13+XQ3Kz6jqoaHJWe4vb1
   D9q0V7l/TkId7yQ==

3) Gautas RSA e. parašas Base64 formatu failui *rakete3.txt*:

   M4xRMKWxRHWtNlkJ44DVYjFmy7Z9QHTjRMkZAsksoJRfGefY19gLtiHIUfnBgpsjtC4E3JPONWuEYvp
   UbbbP9P2jtkyYdw08++CUUnoZoTFwkls91+kDQejVHXlbnL54og5noer8MechJ1DTnrWblUK3bqADoh0T+y
   +x50AeZcGKJAIxEkxp1AX1kMkts1k2ruohkcbQJr+xWIYDWJQv1l+0nuue+PFG0hGarvBkezUQvmzVdFr0+
   3veXurOHPeCAs7b/vCeuwr5BsmXkdCy1iZaCgkYsAXVaJjCHkkZCwVyb5VXhyLOX9x6hqi7YgeI2F9eNS3
   N5F1UBzaQe6H5pQ==

E. parašo tikrinimas:

Pirmiausia, norint patikrinti e. parašo tinkamumą, mums reikalingas pasirašančiojo asmens viešasis raktas (žr. (2.6) formulę). Todėl turime atskirti viešojo rakto dalį:

openssl rsa –in raktaiRSA.key –pubout –out VR_raktaiRSA.pem

kur *raktaiRSA.key* – 3.3.1 skyrelyje sugeneruotas 2048 bitų ilgio raktas,

VR_raktaiRSA.pem – viešojo rakto dalis.

Dabar, kai turime viešąjį raktą, galime patikrinti pasirinkto failo parašo tinkamumą, vykdant komandą:

openssl dgst –sha1 – verify VR_raktaiRSA.pem –signature signRSA.bin rakete1.txt

kur *VR_raktaiRSA.pem* – pasirašančiojo asmens RSA viešasis raktas,

signRSA.bin – e. parašo failas,

rakete1.txt – failas, kuriam tikrinamas e. parašas.

Patikrinus visų failų (*rakete1.txt, rakete2.txt, rakete3.txt*) parašą OpenSSL komandos lange gauname pranešimą:

Verified OK

Taigi, atlikus automatinį OpenSSL patikrinimą, galime teigti, kad sugeneruoti RSA e. parašai yra tinkami. Todėl saugiai pridedame gautus e. parašo prie 2.1 poskyryje suformuoto RSA sistemos duomenų failo.

3.3.2 ELIPSINIŲ KREIVIŲ E. PARAŠO FORMAVIMAS IR TIKRINIMAS

Šiame skyrelyje formuosime EKK e. parašą ir patikrinsime jo korektiškumą, naudodami OpenSSL paketo komandą *dgst*. Veiksmai bus atliekami remiantis 2.2.2 poskyryje aprašytais algoritmais. Gautas korektiškas parašas bus pridedamas prie 2.1 poskyryje suformuoto ECDSA duomenų failo.

E. parašo formavimas:

Pasirinkto failo e. parašo formavimui su SHA1 santraukos funkcija vykdome komandą:

openssl dgst -ecdsa-with-SHA1 -sign secp224k1ECDSA1-key.pem -out signEC.bin rakete1.txt

kur *rakete1.txt* – failas, kuriam formuojamas e. parašas,

secp224k1ECDSA1-key.pem – pasirašančiojo asmens EKK privataus rakto komplektas,

signEC.bin – e. parašo failas dvejetainiu formatu.

Dvejetainio e. parašo formato *signEC.bin* konvertavimui į Base64 formatą *signEC.txt* naudosime OpenSSL komandą *base64:*

openssl base64 -e -in signEC.bin -out signEC.txt

1) Gautas EKK e. parašas Base64 formatu failui *rakete1.txt*:

MD4CHQCwppKFRrmHI9/4sbDcJGvFWnmwhlXamtPFuEqOAh0A1JUAatvOUoNDVDSv1ioH53Sh5Jyq7StLi/uMLg==

2) Gautas EKK e. parašas Base64 formatu failui *rakete2.txt*:
MD0CHG9sY7qAI9+qWLYai15BusCiEjBAWaIcgXdRy2gCHQDLCEXrFxkcRReOAMVazndXTx0bpskrkZxm1V/M

3) Gautas EKK e. parašas Base64 formatu failui *rakete3.txt*:
MD0CHQDy42Yv9L+4fz2tXzWv5Y0OvI+kURHpRYFtblucAhxOd2xTArAp5rerChbKnCjJxyYeIA95gize3ETm

E. parašo tikrinimas:

Pirmiausia, norint patikrinti e. parašo tinkamumą, mums reikalingas pasirašančiojo asmens viešasis raktas (žr. (2.17) formulę). Todėl turime atskirti viešojo rakto dalį:

openssl ec –in secp256k1ECDSA1-key.pem –pubout –out VR_raktaiEC.pem

kur *secp256k1ECDSA1-key.pem* – 3.3.2 skyrelyje sugeneruotas 256 bitų ilgio raktas,
VR_raktaiEC.pem – viešojo rakto dalis.

Dabar, kai turime viešąjį raktą, galime patikrinti pasirinkto failo parašo tinkamumą, vykdant komandą:

openssl dgst –ecdsa-with-SHA1 – verify VR_raktaiEC.pem –signature signEC.bin rakete1.txt

kur *VR_raktaiEC.pem* – pasirašančiojo asmens EKK viešasis raktas,
signEC.bin – e. parašo failas,
rakete1.txt – failas, kuriam tikrinamas e. parašas.

Patikrinus visų failų (*rakete1.txt, rakete2.txt, rakete3.txt*) parašą OpenSSL komandos lange gauname pranešimą:

Verified OK

Taigi, atlikus automatinį OpenSSL patikrinimą, galime teigti, kad sugeneruoti EKK e. parašai yra tinkami. Todėl saugiai pridedame gautus e. parašo prie 2.1 poskyryje suformuoto ECDSA sistemos duomenų failo.

3.4. SERTIFIKATO GENERAVIMAS IR TIKRINIMAS

Viešasis raktas turi būti patvirtintas sertifikavimo centro. Kaip buvo minėta 1.4 poskyryje, tam yra reikalinga sugeneruoti užklausą su nurodytu viešuoju raktu ir identifikacine informacija. Naudojant OpenSSL paketą, tai galima atlikti vykdant *req* komandą. Sertifikato patikrinimui atliekama viešojo rakto sertifikato parametrų peržiūra. Kaip ir buvo minėta 2.3 poskyryje, dėl QR kodo ribotos informacijos talpos, sertifikatai bus talpinami serveryje.

3.4.1 RSA SERTIFIKATO GENERAVIMAS IR TIKRINIMAS

Pirmiausia, kuriant *CA* sertifikatą, reikia sugeneruoti saugaus dydžio, t. y. 2048 bitų ilgio, RSA raktų porą:

openssl genrsa -out ca.pem 2048

kur *ca.pem* – gautas *CA* privataus rakto komplektas.

Pasinaudojant sugeneruotu privačiuoju *CA* raktų failu *ca.pem*, nustačius sertifikato galiojimo laiką lygų 5 metams (*-days 1826*) yra sugeneruojamas, taip vadinamas *self-sign CA* sertifikatas ir išsaugomas *ca.crt* faile:

openssl req -new -x509 -days 1826 -key ca.pem -out ca.crt

Atsiradusiame programos lange reikia teisingai užpildyti visus identifikacinius laukus, nes ši informacija bus matoma visiems prieinamame viešojo rakto sertifikate:

You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.

Country Name (2 letter code) [AU]:**LT**
State or Province Name (full name) [Some-State]:**Kaunas**
Locality Name (eg, city) []:Kaunas
Organization Name (eg, company) [Internet Widgits Pty Ltd]:**Kaunas University of Technology**
Organizational Unit Name (eg, section) []:**KTU**
Common Name (e.g. server FQDN or YOUR name) []:**Indre**
Email Address []:**indre.stareviciute@ktu.edu**

Toliau turi būti sukurtas sugeneruoto sertifikato prašymas serveriui, panaudojant CA privataus rakto komplektą *ca.pem* ir išsaugomas faile *RSA.csr*:

openssl req -new -key ca.pem -out RSA.csr

Vėlgi, atsiradusiame programos lange matomi laukai, kurie turi būti užpildyti:

You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.

Country Name (2 letter code) [AU]:**LT**
State or Province Name (full name) [Some-State]:**Kaunas**
Locality Name (eg, city) []:**Kaunas**
Organization Name (eg, company) [Internet Widgits Pty Ltd]:**RSA certificates**
Organizational Unit Name (eg, section) []:**RSA**
Common Name (e.g. server FQDN or YOUR name) []:**Indre Star**
Email Address []:**indre.stareviciute@ktu.lt**

Pasirašomas sukurtas serverio prašymo failas *RSA.csr* panaudojant *CA* sertifikato *ca.crt* privatųjį raktą *ca.pem* ir išsaugomas saugiai sugeneruotas RSA sertifikatas *RSA.crt*, kuris galioja 5 metus:

openssl x509 -req -days 1826 -in RSA.csr -CA ca.crt -CAkey ca.pem -set_serial 01 -out RSA.crt

Signature ok
subject=/C=LT/ST=Kaunas/L=Kaunas/O=RSA certificates/OU=RSA/CN=Indre Star/emailAd
dress=indre.stareviciute@ktu.lt
Getting CA Private Key

Sertifikato patikrinimui naudojame OpenSSL *verify* komandą, kuri patikrina, kad sertifikatas *RSA.crt* pasirašytas su tinkamu *CA* sertifikatu *ca.crt*:

openssl verify -CAfile ca.crt RSA.crt

Rodomas rezultatas įvykdžius komandą:

RSA.crt: OK

Taigi, matome, kad RSA sertifikato generavimo ir tikrinimo procesai veikia korektiškai.

3.4.2 ECDSA SERTIFIKATO GENERAVIMAS IR TIKRINIMAS

Analogiškas sertifikato kūrimo procesas (kaip ir 3.4.1 poskyryje) vykdomas ir norint sugeneruoti *CA* patvirtintą ECDSA sertifikatą. Šiame poskyryje pus pateiktos tik konkrečios vykdytos komandos. Norint matyti detalesnį sertifikatų kūrimo procesą žiūrėti 3.2.1 poskyrį.

Pirmiausia sukuriame 224 bitų ilgio ECDSA privataus rakto komplektą *PRecdsa.pem*:

openssl ecparam -genkey -name secp224r1 -noout -out PRecdsa.pem

Toliau sukuriame sugeneruoto sertifikato prašymą serveriui, panaudodami sugeneruotą ECDSA privataus rakto komplektą *PRecdsa.pem*. Užpildome visus identifikacinius laukus ir gautą prašymą išsaugome faile *ECDSA.csr*:

openssl req -new -key PRecdsa.pem -out ECDSA.csr

Pasirašomas sukurtas serverio prašymo failas *ECDSA.csr* panaudojant *CA* sertifikato *ca.crt* privatųjį raktą *ca.pem* ir išsaugomas saugiai sugeneruotas ECDSA sertifikatas *ECDSA.crt*, kuris galioja 5 metus:

openssl x509 -req -days 1826 -in ECDSA.csr -CA ca.crt -CAkey ca.pem -set_serial 02 -out ECDSA.crt

Gauto sertifikato patikrinimui naudojame OpenSSL *verify* komandą, kuri patikrina, kad sertifikatas *ECDSA.crt* patvirtintas tinkamu *CA* sertifikatu *ca.crt*:

openssl verify -CAfile ca.crt ECDSA.crt

Patikrinus sertifikatą darome išvadą, kad ECDSA sertifikato generavimo ir tikrinimo procesai veikia korektiškai.

3.5. TYRIMO METU GAUTA PREKĖS KILMĖS QR KODO INFORMACINĖ STRUKTŪRA

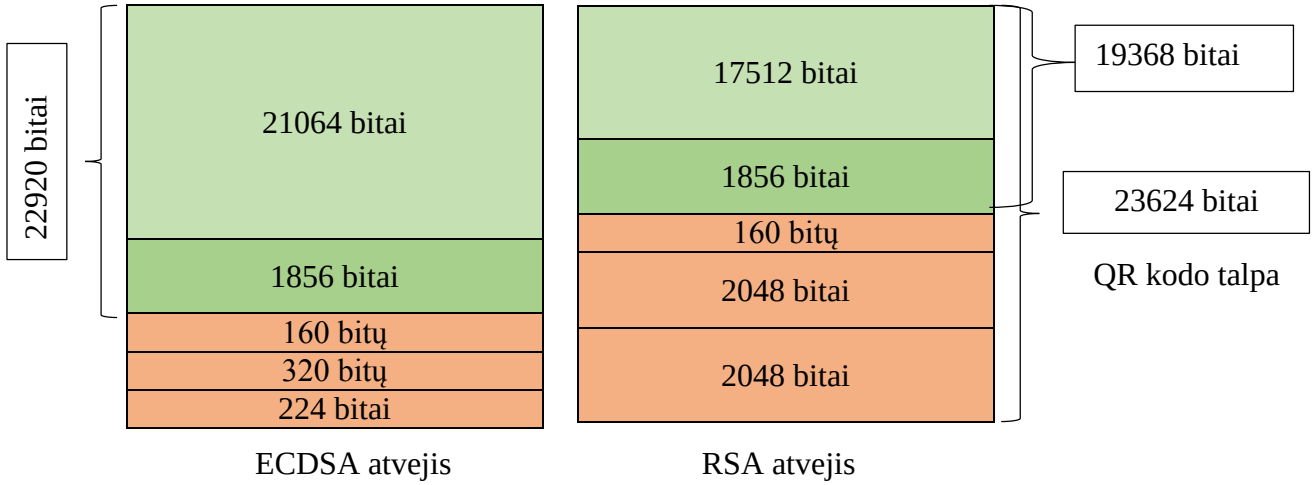
Darbo metu buvo nustatyta, kad QR kodo maksimali talpa, kai duomenys yra koduojami dvejetainiu būdu, gali siekti 23624 bitus, t. y. 2,9 KB (žr. 2.3 poskyrį). Tai reiškia, kad į QR kodą negalima talpinti didesnio tūrio informacijos nei 2,9 KB. Taip pat nustatėme, kad QR kodo informacija yra padalinta į dvi dalis: analitinę, kitaip vadinamą kriptografine, kuri talpina fiksuoto dydžio kriptografinius duomenis, ir naudingą – informacija, kurią norime pateikti vartotojui. Pagal 2.3 poskyryje pasirinktą QR kodo infrastruktūros vaizdavimo būdą šiame poskyryje pavaizduosime atitinkamas RSA ir ESDA atvejais gautas QR kodo infrastruktūras ir jas palyginsime pagal išnaudojamus informacijos kiekius.

Darbe nagrinėjamo pavyzdžio atveju prekės kilmės duomenų failas yra 1856 bitų dydžio. Tai yra visa naudinga QR kodo informacija (žr. 2.1 poskyrį). Prie šios naudingos informacijos, duomenų failo autentiškumui, vientisumui ir neišsiginamumui užtikrinti buvo pridėti kriptografiniai duomenys: atvirojo kodo OpenSSL paketu sugeneruota ir patikrinta santraukos funkcijos reikšmė, e. parašas ir viešasis raktas. Visi duomenys buvo pridėti į sukurtą duomenų failą Base64 formatu. Kadangi analizuojame ir lyginame dvi kriptografines sistemas, tai buvo kuriami du atskiri duomenų failai:

- *PrekiuDuomenys_RSA.csv* – failas, kuris talpina 2.1 poskyryje aprašytus duomenis ir RSA kriptosistemos OpenSSL bibliotekos sugeneruotus kriptografinius parametrus (santraukos funkcijos reikšmę (žr.3.1 poskyrį), RSA viešojo rakto reikšmę (žr. 3.2.1 poskyrį), RSA e. parašo reikšmę (žr. 3.3.1 poskyrį)).
- *PrekiuDuomenys_EKK.csv* – failas, kuris talpina 2.1 poskyryje aprašytus duomenis ir EKK kriptosistemos OpenSSL bibliotekos sugeneruotus kriptografinius parametrus (santraukos funkcijos reikšmę (žr.3.1 poskyrį), EKK viešojo rakto reikšmę (žr. 3.2.2 poskyrį), EKK e. parašo reikšmę (žr. 3.3.2 poskyrį)).

RSA atveju, analitinė informacija, t. y. kriptografiniai duomenys užima 4256 bitus, iš kurių santraukos funkcijos reikšmė užima 160 bitų, e. parašas – 2048 bitus ir viešasis raktas – taip pat 2048 bitus (žr. 3.1 pav. (RSA atvejis) oranžinė spalva). Atsižvelgiant į anksčiau minėtą QR kodo maksimalią informacijos talpą, tai reiškia, kad naudingiems duomenims (prekės informacijai) lieka 19368 bitai (žr. 3.1 pav.). Kadangi darbe nagrinėjamiems duomenims jau turime išnaudotus 1856 bitus (žr. 3.1 pav. (RSA atvejis) tamsesnė žalia spalva)., tai informaciją, kurią dar galime patalpinti į šį QR kodą gali siekti 17512 bitus (žr. 3.1 pav. (RSA atvejis) šviesiai žalia spalva).

Tuo tarpu, ECDSA analitinės informacijos tūris siekia tik 736 bitus, kur santraukos funkcija, e. parašas ir viešasis raktas užima atitinkamai 160, 320 ir 224 bitus (žr. 3.1 pav. (ECDSA atvejis) oranžinė spalva). Tuo tarpu, naudingiems duomenims lieka net 22920 bitai. Kadangi darbe nagrinėjamiems duomenims jau turime išnaudotus 1856 bitus (žr. 3.1 pav. (ECDSA atvejis) tamsesnė žalia spalva)., tai informaciją, kurią dar galime patalpinti į šį QR kodą gali siekti 21064 bitus (žr. 3.1 pav. (ECDSA atvejis) šviesiai žalia spalva).



3.1 pav. ECDSA ir RSA QR kodo talpinamos informacijos paskirstymas

Taigi, nusistačius maksimalią QR kodo talpą, atskirai išanalizavus RSA ir ECDSA talpinamos informacijos QR kode atvejus, galime daryti išvadą, kad ECDSA talpinamos naudingos informacijos kiekis QR kode yra 3552 bitų (444 baitų) didesnis nei RSA.

3.6. PREKĖS KILMĖS OR KODO NUSKAITYMAS

Apskaičiavus visus kriptografinius duomenis (santraukos funkciją (žr. 3.1 poskyrį), raktų poras (žr. 3.2 poskyrį) ir e. parašus (žr. 3.3 poskyrį)) abiem (RSA ir EKK) tyrinėjamoms kriptosistemoms, turime pilnai paruoštus CSV failus: *PrekiuDuomenys_RSA.csv* ir *PrekiuDuomenys_EKK.csv*. Failai talpina rakečių ir kriptografinius duomenis, atitinkamai šifruojamus panaudojant RSA ir EKK kriptografines sistemas. Turint šiuos duomenų failus, jiems galime generuoti QR kodus, panaudojant automatinį QR kodo generavimo algoritmą. Tai reiškia, kad QR kodas bus sugeneruotas internetiniame puslapyje, pasinaudojant nemokama QR kodo generavimo programėle (žr. 1.5.4 poskyrį).

Kadangi duomenys išsaugoti CSV formatu, tai QR kodo generavimui buvo pasirinktas būtent CSV formato failus nuskaitantis ir iš jų QR kodus sugeneruojantis puslapis [30]. Įkėlus failą buvo sukuriami atskiri QR kodai kiekvienai duomenų failo eilutei. Parsisiuntus ZIP formatu sutrauktus visus sugeneruotus QR kodus, jie buvo išsiskleisti ir naudojami tolimesnei analizei.

Pasinaudojus nemokama *Windows* operacinės sistemos telefonams skirta *Microsoft* aplikacija *QR Code and Barcode Reader* buvo nuskaityti sugeneruoti QR kodai.

3.2 ir 3.3 paveikslėliuose galime matyti iš gautų CSV failų sugeneruotus atitinkamus QR kodus.

QR_RSA_Rakete1 QR_RSA_Rakete2 QR_RSA_Rakete3

3.2 pav. RSA sistemos duomenų failo *PrekiųDuomenys_RSA.csv* sugeneruoti QR kodai

QR_ECDSA_Rakete1 QR_ECDSA_Rakete2 QR_ECDSA_Rakete3

3.3 pav. ECDSA sistemos duomenų failo *PrekiųDuomenys_EC.csv* sugeneruoti QR kodai

ECDSA atveju QR kodo informacija užima 1,09 KB. Tuo tarpu, RSA atveju QR kodo informacija užima 3,26 KB. Akivaizdžiai matomas ECDSA sistemos pranašumas, kur suformuoto duomenų failo dydis yra beveik 3 kartus mažesnis nei RSA.

QR kodas yra sudarytas iš modulių, kurių skaičius priklauso nuo talpinamos informacijos kiekio (žr. 1.5.1 ir 2.3 poskyrius). 3.2 paveikslėlyje galime matyti, kad gautas QR kodas yra gana smulkus, o tuo tarpu 3.3 paveikslėlyje sugeneruotas QR kodas yra kur kas didesnis. Todėl aiškiai matome, kad ECDSA atveju QR kodo nuskaitymas bus aiškesnis (netgi jį mažinant) nei RSA panaudojimo atveju.

Taip pat, modulių kiekis QR kode lemia ir QR kodo nuskaitymo galimybes. Todėl galime teigti, kad, dėl mažesnio talpinamų duomenų kiekio, mažinant ECDSA QR kodą jis bus aiškiau nuskaitomas nei RSA. Taigi, ECDSA QR kodas bus plačiau panaudojamas įvairiems marketingo sprendimams, kai reikalingas mažas QR kodo dydis. Kitaip tariant, ECDSA QR kodo panaudojimas bus praktiškesnis, t. y. galimas ant mažesnių prekių pakuočių nei RSA.

IŠVADOS

Darbo metu, išanalizavus RSA ir ECDSA kriptografinių sistemų saugumą, tiriant badmintono rakečių QR kodo kilmės patvirtinimo atvejį, buvo gautos atitinkamos išvados:

- Apžvelgus pagrindinių santraukos funkcijų panaudojimo galimybes ir gyvavimo ciklo laikotarpius, sukurtiems duomenims pasirinkta riboto, tačiau leistino galiojimo SHA-1 santraukos funkcija, kaip tinkama naudoti, norint užtikrinti badmintono rakečių duomenų failo autentiškumą ir vientisumą.
- Apžvelgus e. parašo sistemos pagrindinius veikimo principus, buvo nustatyta, kad e. parašo pridėjimas prie duomenų failo užtikrina duomenų vientisumą, autentiškumą ir neišsiginamumą.
- Analizuojant QR kodo duomenų talpos galimybes bei informacijos kodavimo būdus, darbo metu efektyviam ir kompaktiškam duomenų saugojimui QR kode pritaikytas dvejetainis kodavimo būdas.
- Išanalizavus darbui pasirinktų kriptografinių sistemų galimus raktų ilgius, buvo nustatyta, kad patikimas ir saugus ECDSA raktų ilgis yra daugiau nei 8 kartus trumpesnis nei RSA.
- Išanalizavus darbui pasirinktoms kriptografinėms sistemoms naudojamus e. parašo ilgius, buvo nustatyta, kad ECDSA e. parašas yra 6,4 kartus trumpesnis nei RSA.
- Apžvelgus QR kodo talpos galimybes ir sertifikato kūrimo principus, buvo prieita išvados, kad dėl QR kodo riboto informacijos tūrio, QR kodo e. parašo sertifikatas turi būti patalpintas serveryje.
- Tyrimo metu, sugeneravus ir nuskaičius darbo metu sukurtus QR kodus, buvo nustatyta, kad ECDSA talpinamos naudingos informacijos kiekis QR kode yra 3552 bitų (444 baitų) didesnis nei RSA.
- Tyrimas parodė, kad ECDSA suformuoto duomenų failo dydis yra beveik 3 kartus mažesnis nei RSA.
- Tyrimo metu buvo gauta, kad ECDSA QR kodą sudaro mažiau modulių nei RSA pagrindu sugeneruotą QR kodą, todėl ECDSA QR kodo panaudojimas galimas ant mažesnių prekių pakuočių nei RSA.

Taigi, gautos darbo išvados parodo, kad QR kodo informacijos autentiškumo ir vientisumo užtikrinimui tikslinga naudoti ECDSA kriptografinę sistemą.

LITERATŪRA

[1] QRMarket, „QR Market,“ 2016. [Tinkle]. Available: http://www.qrmarket.lt/. [Kreiptasi 20 04 2016].

[2] J. H. Chang, „An introduction to using QR codes in scholarly journals,“ *Training Material,* t. 1, nr. 2, pp. 113-114, 2014.

[3] S. Majumdar, A. Maiti, B. Bhattacharyya ir A. Nath, „A new encrypted Data hiding algorithm inside a QR Code implemented for an Android Smartphone system: S_QR algorithm,“ *International Journal of Innovative Research in Advanced Engineering (IJIRAE),* t. 2, nr. 4, pp. 40-46, 2015.

[4] A. Singhal ir R. Pavithr, „Degree Certificate Authentication using QR Code and Smartphone,“ *International Journal of Computer Applications,* t. 120, nr. 16, pp. 38-43, 2015.

[5] E. Sakalauskas, T. Blažauskas ir K. Lukšys, Elektroninių dokumentų ir duomenų sauga, Kaunas: Vitae Litera, 2008.

[6] V. Aurora, „The code monkey's guide to cryptographic hashes for content-based addressing,“ 12 11 2007. [Tinkle]. Available: http://valerieaurora.org/monkey.html. [Kreiptasi 30 12 2015].

[7] I. Mironov, „Hash functions: Theory, attacks, and applications,“ 14 11 2005.

[8] VU.mif, „Kodavimas ir kriptografija,“ [Tinkle]. Available: http://mif.vu.lt/lt2/dlsm/studentams/studiju-programos/failai/bakalaurai/7semestras/kriptografija-ir-kodavimas_konspektas. [Kreiptasi 20 05 2014].

[9] S. Minkevičius, „Kriptografija, duomenų saugumas ir jų pritaikymas elektroninėje komercijoje,“ įtraukta *Elektroninės komercijos technologijų pagrindai*, Vilnius, Vilniaus universitetas, 2007, pp. 28-37.

[10] E. Sakalauskas, N. Listobadskis, G. S. Dosinas, K. Lukšys ir A. Katvickis, „Slaptojo rakto (simetrinė) šifravimo sistema,“ įtraukta *Kriptografinės sistemos*, Kaunas, Kauno technologijos universitetas, 2008, pp. 53-70.

[11] E. Sakalauskas, N. Listobadskis, G. S. Dosinas, K. Lukšys ir A. Katvickis, „Kriptografinės sistemos,“ įtraukta *Viešojo rakto (asimetrinė) kriptosistema*, Kaunas, Kauno technologijos universitetas, 2008, pp. 71-75.

[12] E. Sakalauskas, T. Blažauskas ir K. Lukšys, „Simetrinė šifravimo sistema,“ įtraukta *Elektroninių dokumentų ir duomenų sauga*, Kaunas, Kauno technologijos universitetas, 2008, p. 23.

[13] E. Sakalauskas, T. Blažauskas ir K. Lukšys, „Asimetrinė šifravimo sistema,“ įtraukta *Elektroninių dokumentų ir duomenų sauga*, Kaunas, Kauno technologijos universitetas, 2008, p. 28.

[14] S. Minkevičius, „Simetrinių ir asimetrinių kriptosistemų lyginimas,“ įtraukta *Elektroninės komercijos technologijų pagrindai*, Vilnius, Vilniaus universitetas, 2007, pp. 38-39.

[15] W. Diffie ir M. E. Hellman, „New Directions in Cryptography,“ *IEEE,* t. 22, nr. 6, pp. 77-78, 2008.

[16] B. S. Kaliski, „Dr.Dobb's: The World of Software Development: RSA Digital Signatures,“ 01 05 2001. [Tinkle]. Available: http://www.drdobbs.com/rsa-digital-signatures/184404605. [Kreiptasi 10 05 2016].

[17] A. I. Ali, „Comparison And Evaluation Of Digital Signature Schemes Employed In Ndn Network,“ *International Journal of Embedded systems and Applications(IJESA),* t. 5, nr. 2, pp. 15-29, 2015.

[18] E. Sakalauskas, G. Dosinas, N. Listopadskis, K. Lukšys ir A. Katvickis, Kriptografinės sistemos, Kaunas: Vitae Litera, 2008.

[19] H. H. Harralson, „Forensic Analysis of Electronic Signatures,“ įtraukta

Developments in Handwriting and Signature Identification in the Digital Age, ISBN, 2014, pp. 71-111.

[20] E. Sakalauskas, T. Blažauskas ir K. Lukšys, Elektroninių dokumentų ir duomenų sauga, Kaunas: Vitae Litera, 2008.

[21] E. Sakalauskas, N. Listopadskis ir G. S. Dosinas, Kriptografijos Teorija, Kaunas: Vitae Litera, 2008.

[22] EDS, „Valstybinė mokesčių inspekcija prie Lietuvos Respublikos finansų ministerijos," 2010. [Tinkle]. Available: https://deklaravimas.vmi.lt/lt/Apie/Bendroji_informacija/Elektroninis_parasas.aspx. [Kreiptasi 05 04 2016].

[23] G. Repečka, „Elektroninis parašas," 05 03 2012. [Tinkle]. Available: https://repecka.net/2012/03/05/elektroninis-parasas/comment-page-1/. [Kreiptasi 10 04 2016].

[24] ISO, Information technology — Automatic identification and data capture techniques — QR Code 2005 bar code symbology specification, Šveicarija: ISO/IEC, 2006.

[25] „QR Code," 30 05 2012. [Tinkle]. Available: http://archive.is/20120530074133/http:/code.google.com/p/zxing/wiki/BarcodeContents. [Kreiptasi 28 06 2015].

[26] „goQR.me: QR Code Generator," [Tinkle]. Available: http://goqr.me/. [Kreiptasi 10 02 2015].

[27] „QRStuff.com: get your QR codes out here," [Tinkle]. Available: http://www.qrstuff.com/. [Kreiptasi 15 02 2015].

[28] Google, „QR Droid Code Scanner," Google Play, 07 04 2016. [Tinkle]. Available: https://play.google.com/store/apps/details?id=la.droid.qr. [Kreiptasi 20 04 2016].

[29] Google, „BeautyQR - QR code generator," Google Play, 07 09 2013. [Tinkle]. Available: https://play.google.com/store/apps/details?id=de.wendytech.beautyqr&feature=more_from_developer#?t=W251bGwsMSwxLDEwMiwiZGUud2VuZHl0ZWNoLmJlYXV0eXFyIl0. [Kreiptasi 21 04 2016].

[30] „Barcode," Eged Soft, 2011. [Tinkle]. Available: http://barcode.egedsoft.com/csv2qrcode.php. [Kreiptasi 10 05 2016].

[31] D. Treigytė ir I. Pikturnienė, „Prekės kilmės šalies ir prekės ženklo kilmės šalies įtaka vartotojo požiūriui į prekę," *VERSLAS: teorija ir praktika*, t. 4, nr. 1, pp. 38-46, 2009.

[32] E. Sakalauskas, N. Listobadskis, G. S. Dosinas, K. Lukšys ir A. Katvickis, „RSA sistema," įtraukta *Kriptografinės sistemos*, Kaunas, Kauno technologijos universitetas, 2008, pp. 88-89.

[33] D. Johnson, A. Menezes ir S. Vanstone, „The Elliptic Curve Digital Signature Algorithm (ECDSA)," *International Journal of Information Security*, t. 1, nr. 1, pp. 36-63, 2014.

[34] S. Majumdar, A. Maiti, B. Bhattacharyya ir A. Nath, „Advanced Security Algorithm Using QRCode Implemented for an Android Smartphone System: A_QR," *International Journal of Advance Research in Computer Science and Management Studies*, t. 3, nr. 5, pp. 21-31, 2015.

[35] W. Dai, „cryptopp.com," 31 03 2009. [Tinkle]. Available: http://www.cryptopp.com/benchmarks.html. [Kreiptasi 15 02 2016].

[36] L. E. B. III, „The Elliptic Curve Digital Signature Algorithm Validation System (ECDSAVS)," National Institute of Standards and Technology, 2004.

[37] „Certicom: Explaining Implicit Certificates," 2015. [Tinkle]. Available: https://www.certicom.com/index.php/explaining-implicit-certificate. [Kreiptasi 01 05 2016].

[38] web4it, „web4it: QR kodas," [Tinkle]. Available: http://www.web4it.lt/qr-kodas/. [Kreiptasi 12 06 2015].

[39] OpenSSL, „OpenSSL: The Open Source Toolkit for SSL/TSL,“ 2015. [Tinkle]. Available: https://www.openssl.org/. [Kreiptasi 01 05 2015].

[40] K. Lukšys, „P170M100 Kriptografinės sistemos,“ 2014. [Tinkle]. Available: https://moodle.ktu.edu/. [Kreiptasi 29 10 2014].

[41] „QR Code,“ 15 09 2012. [Tinkle]. Available: http://archive.is/20120915040142/http:/www.qrcode.com/en/vertable1.html. [Kreiptasi 25 06 2015].

1 Priedas. QR KODO MAKSIMALŪS DUOMENŲ KIEKIAI PAGAL VERSIJĄ

1 lentelė. Kiekvienos QR kodo versijos maksimalus duomenų kiekis [41]

Versija	Moduliai	Korekcijos lygmenys	Duomenų kiekis (bitais)	Skaitinis	Raidinis - Skaitinis	Dvejetainis	Kanji
1	21x21	L	152	41	25	17	10
		M	128	34	20	14	8
		Q	104	27	16	11	7
		H	72	17	10	7	4
2	25x25	L	272	77	47	32	20
		M	224	63	38	26	16
		Q	176	48	29	20	12
		H	128	34	20	14	8
3	29x29	L	440	127	77	53	32
		M	352	101	61	42	26
		Q	272	77	47	32	20
		H	208	58	35	24	15
4	33x33	L	640	187	114	78	48
		M	512	149	90	62	38
		Q	384	111	67	46	28
		H	288	82	50	34	21
5	37x37	L	864	255	154	106	65
		M	688	202	122	84	52
		Q	496	144	87	60	37
		H	368	106	64	44	27
6	41x41	L	1088	322	195	134	82
		M	864	255	154	106	65
		Q	608	178	108	74	45
		H	480	139	84	58	36
7	45x45	L	1248	370	224	154	95
		M	992	293	178	122	75
		Q	704	207	125	86	53
		H	528	154	93	64	39
8	49x49	L	1552	461	279	192	118
		M	1232	365	221	152	93
		Q	880	259	157	108	66
		H	688	202	122	84	52
9	53x53	L	1856	552	335	230	141
		M	1456	432	262	180	111
		Q	1056	312	189	130	80
		H	800	235	143	98	60
10	57x57	L	2192	652	395	271	167
		M	1728	513	311	213	131
		Q	1232	364	221	151	93
		H	976	288	174	119	74