Tellija

RIIGI INFOSÜSTEEMI AMET

Krüptograafiliste algoritmide elutsükkel

UURING

2017

CYBERNETICA

Krüptograafiliste algoritmide elutsükli uuring 2017

Tehniline dokument

Versioon 2.0

9. veebruar 2018. a.

28 lk

Dok A-101-9

Projektijuhid:	Kaur Virunurm (Riigi Infosüstemi Amet)
	Mari Seeba (Cybernetica)
Autorid:	Arne Ansper, MSc (Cybernetica)
	Ahto Buldas, PhD (Cybernetica)
	Jan Willemson, PhD (Cybernetica)

Riigi Infosüsteemi Amet, Pärnu maantee 139a, 15169 Tallinn, Eesti.
Email: ria@ria.ee, Web: `https://www.ria.ee`, Telefon: +372 663 0200.

Cybernetica AS, Mäealuse 2/1, 12618 Tallinn, Eesti.
E-mail: info@cyber.ee, Web: `https://www.cyber.ee`, Telefon: +372 639 7991.

Sisukord

1 Sissejuhatus

Käesolev aruanne on järjeks neljale varasemale uuringule, mis pärinevad aastatest 2011–2016 [3, 4, 5, 7]. Seekord katame kolme suuremat teemat.

2. peatükis vaatleme viimase aasta jooksul murdunud olulisemaid krüptoalgoritme (mida ei ole õnneks palju) ning anname soovitusi kasutatavate primitiivide ja võtmepikkuste jaoks. Suurimaks lahtiseks küsimuseks jääb selles vallas endiselt üldotstarbelise kvantarvuti loomise kiirus, mida on väga raske ennustada, kuid mis määrab otseselt praegu kasutatavate algoritmide eluea.

2017. aasta Eesti suurim krüptograafiline sündmus oli kahtlemata ID-kaardi turvanõrkuse avastamine tšehhi teadlasterühma poolt. 3. peatükis avame lühidalt selle probleemi sisu ning kirjeldame tema lahendamiseks tehtud samme koos kasutuselevõetud uute, elliptkõveratel põhinevate krüptoalgoritmidega.

Maailma mastaabis aga on ühed kiiremini arenevad krüptograafiarakendused plokiahelad. Käesoleva uuringu 4. peatükk annab plokiahelatest süstemaatilise ülevaate ning aitab lugejal otsustada, kas ja milline plokiaheltehnoloogia tema vajadusi võiks rahuldada. Täiendava panusena on selle ülevaate koostamise käigus loodud eestikeelne plokiahelate terminoloogia.

2 Krüptograafiliste primitiivide ja protokollide tugevushinnangud

2.1 Võtmepikkuse hinnangud

Krüptoanalüütilised edusammud

2017. aasta veebruaris leidis Google'i poolt toetatud teadlaste meeskond lõpuks ammuoodatud kollisiooni räsifunktsioonile SHA-1 [26]. Kuigi kollisiooni olemasolu ei tähenda automaatselt kõigi kasutusstsenaariumite ebaturvaliseks muutumist, kordame siinkohal juba kõigis eelmistes krüptograafiliste algoritmide elutsükli uuringutes antud soovitust SHA-1 käibelt kõrvaldada. Asenduseks sobivad kõik SHA-2 ja SHA-3 perekonna räsifunktsioonid (arvestades soovitavat turvataset, vt jaotis 2.2).

Siinses aruandes käsitletaval perioodil (2016–2017) pole tehtud märkimisväärseid läbimurdeid teiste põhiliste krüptograafiliste primitiivide murdmisel. Seega on ka kasutamiseks soovitatavate algoritmide loetelu ja võtmepikkused üldjoontes samad kui eelmises aruandes [7].

Kõige olulisemaks tundmatuks suuruseks võtmepikkuste eluea hindamisel on üldotstarbeliste kvantarvutite arenduskiirus. 2017. aasta oktoobris teatas Intel 17-bitise ülijuhtiva kvantkiibi väljalaskmisest[1].

Kui suur samm see on? Google'i teadurid on hinnanud, et tänaste klassikaliste arvutite jõudluse ületamiseks mõnes funktsioonis piisab kvantarvutist, kus on umbkaudu 50 ülijuhtivusel põhinevat kvantbitti [16].

Proos ja Zalka leidsid, et n-bitise RSA mooduli tegurdamiseks on vaja umbes $2n$ kvantbitti ning n-bitises elliptkõverarühmas diskreetse logaritmi leidmiseks on vaja umbes $6n$ kvantbitti [23]. Seega näiteks on 2048-bitise RSA murdmiseks vaja umbes 4096 ning kõveral P-256 diskreetse logaritmi arvutamiseks umbes 1500 kvantbitti. Proos ja Zalka märgivad ka, et füüsiliste kvantbittide ebastabiilsuse tõttu tuleb Shori algoritmis kasutatavatele bittidele lisada veaparandusliiasust, mistõttu praktikas võivad need arvud olla paar korda suuremad.

On näidatud [19], et üldotstarbelisel kvantarvutil saab Groveri algoritmi abil kiirendada sümmeetriliste šifrite (näiteks AES) k-bitise võtmeruumi täielikku läbivaatust 2^k sammult $2^{\frac{k}{2}}$ sammuni. Selleks vajalike kvantbittide arv on toodud tabelis 1. See tähendab, et piisavalt pika kvantregistriga kvantarvutite ilmudes tuleb plokkšifrite võtmepikkust sama ründekindluse saavutamiseks kahekordistada. Ka selles rakenduses tuleb lisaks arvestada veaparandusvajadusega.

[1] https://newsroom.intel.com/news/intel-delivers-17-qubit-superconducting-chip-advanced-packaging-

Tabel 1. AES-i murdmiseks vajalike kvantbittide arv

AESi võtmepikkus	Vajalike kvantbittide arv
128	2953
192	4449
256	6681

Aggarwal jt. ennustasid aastal 2017, et ühes arvutis kasutada olevate kvantbittide arv jõuab 10000-ni ajavahemikus 2025–2035 [12]. Kuna nende hinnangud tuginevad vaid väga piiratud arvule andmepunktidele, tuleb sellesse ennustusse suhtuda üsna kriitiliselt.

Samas nõuavad kvantkindlate algoritmide arendus, standardimine ja juurutus võrdlemisi palju aega. Seetõttu tuleb ettevalmistustega postkvant-algoritmidele üleminekuks alustada juba täna. Septembris 2017 tegi ITU (*International Telecommunication Union*) avalduse, milles teatas X.509 standardi plaanitavast uuendusest [11]. Uuenduse sisuks on mitme avaliku võtme algoritmi toe võimaldamine X.509 sertifikaatides. Selle toe abil saab võimalikuks postkvant-algoritmide järkjärguline kasutuselevõtt.

Juurutuseks sobivate postkvant-algoritmide väljavalimisega tegelevad ka teised standardiorganisatsioonid, näiteks NIST[2] ja ETSI[3]. Eestis peaks töö järgmise põlvkonna krüptolahenduste alustehnoloogiate valikul algama samuti juba lähiaastatel.

Eriti kriitilised on elektroonilise identiteedi lahendused. Hetkel välja antavate ID-kaardid kehtivad viis aastat, millele lisandub veel viieaastane ID-kaartide väljaandmise lepingu kehtivuse periood. Hankeprotsess ise nõuab samuti umbes kolm aastat. Kokkuvõtteks saame ID-kaardi elutsükli pikkuseks ligikaudu 13 aastat.

Kui realiseerub kõige optimistlikum kvantarvuti valmimise stsenaarium, jääb järgmise ID-kaardi põlvkonna eluiga sellesse perioodi juba osaliselt sisse. Seetõttu tuleks uues ID-kaardi (või üldisemalt igas eID platvormi) hankes juba arvestada vajadusega võtta kasutusele kvantarvutikindlaid algoritme. Välja tuleb selgitada, millised on tootjate plaanid ja võimalused, aga samuti olla kursis rahvusvaheliste jõupingutustega uute krüptomehhanismide standardimise vallas. Eesti jaoks on palju kaalul ning õigeaegne info kogumine ja tegevuste plaanimine vähendab tulevikus oluliselt riske ja kulusid.

2.2 Võtmepikkuste soovitused

Nagu ülal juba mainitud, pole viimase aasta jooksul peale SHA-1 kollisiooni leidmise märkimisväärseid krüptograafilisi läbimurdeid toimunud. Seega kehtivad üldjoontes eelmistes aruande versioonides [5, 7] antud soovitused.

Need soovitused tuginevad peamiselt ECRYPT II ja ENISA aruannetele [13, 25], mida uuendati viimati vastavalt aastatel 2012 ja 2014. Selles uuringus tugineme NIST-i (USA *National Institute of Standards and Technology*) poolt 2016. aastal antud hinnangutele [14] ning Saksamaa BSI (*Bundesamt für Sicherheit in der Informationstechnik*) 2017. aasta soovitustele [8].

[2] https://csrc.nist.gov/Projects/Post-Quantum-Cryptography
[3] https://portal.etsi.org/TBSiteMap/CYBER/CYBERQSCToR.aspx

Tabel 2 võtab kokku nende aruannete soovitused peamiste krüptograafiliste algoritmide perede jaoks. Algoritmide turvatase määratakse bittides, kus mingi algoritmi b-bitine turvatase tähendab, et efektiivseim teadaolev rünne kasutab arvutusressurssi, mis kulub ligikaudu 2^b ploki krüpteerimiseks plokkšifriga.

Tabeli 2 veerg "DSA, DH" viitab DSA signatuurialgoritmile ja Diffie-Hellmani võtmevahetusele üle jäägiklassiringi, kus avaliku ja salajase võtme soovitatavad pikkused on vastavalt L ja N. Veerg "RSA" annab soovituse RSA algoritmi mooduli pikkusele, veerg "ECC" aga elliptkõveratele tuginevate krüptoalgoritmide võtmepikkustele. Veerud "Plokkšifrid" ja "SHA-2, SHA-3" soovitavad vastavalt sümmeetriliste algoritmide võtmepikkust ning räsifunktsioonide väljundi pikkust. Peamiseks soovitatavaks sümmeetriliseks krüptoalgoritmiks on jätkuvalt AES.

Tabel 2. Krüptograafiliste algoritmide soovitatavad võtmepikkused

Turvatase	DSA, DH	RSA	ECC	Plokkšifrid	SHA-2, SHA-3
128	$L = 3072$, $N = 256$	3072	256 . . . 383	128	256
192	$L = 7680$, $N = 384$	7680	384 . . . 511	192	384
256	$L = 15360$, $N = 512$	15360	512+	256	512

NIST-i aruandes [14] leitakse, et kõik tabelis 2 toodud võtmepikkused sobivad kasutamiseks nii lühikeses kui keskpikas perspektiivis (kuni 2030. aastani ning edaspidi). Algoritmid, mis pakuvad ainult 112-bitist turvataset (näiteks SHA2-224 ja 3TDEA, aga ka 2048-bitine RSA), on kasutatavad ainult lühikeses perspektiivis ja pärandsüsteemides. BSI 2017. aasta aruanne [8] täpsustab viimast soovitust nõudes, et alla 128-bitise turvatasemega algoritmide kasutamine tuleb lõpetada 2022. aastal.

Rõhutame veelkord, et need hinnangud ei võta arvesse võimalikke ründeid kvantarvutiga. Nii ei soovita teine USA agentuur NSA (*National Security Agency*) kvantarvutite peatse ilmumise kartuses uutes rakendustes kasutada elliptkõveraid lühema kui 384-bitise võtmega ega AES-i lühema kui 256-bitise võtmega [6]. See soovitus oli ka üheks põhjuseks, miks Eesti ID-kaardi uute krüptoalgoritmide aluseks valiti elliptkõver P-384.

2.3 TLS

TLS (*Transport Layer Security*) on valdava osa Interneti-liikluse turvamiseks kasutatav protokollistik. Hetkel (2018. aasta alguses) on värskeimaks standardi versiooniks endiselt 1.2, kuigi töö uuema versiooni 1.3 kallal peaks lähiajal lõpusirgele jõudma.

Võrreldes versiooniga 1.2 toob 1.3 endaga kaasa mitmeid olulisi uuendusi, sealhulgas:

- välja on jäetud hulk nõrgaks muutunud šifreid ja töörežiime (nt SHA-1, RC4, DES, 3DES, AES-CBC, MD5 jt),
- sümmeetrilised algoritmid töötavad ainult autenditud režiimis,
- kätlusprotokoll (*handshake*) nõuab vähem sõnumeid ja on seetõttu kiirem,
- lisatud on uusi elliptkõveratele tuginevaid krüptosüsteeme (nt X25519).

Kuigi osade brauserite ja teekide lähtekoodis on TLS 1.3 tugi juba olemas, pole see reeglina vaikimisi kasutusele võetud, sest ülejäänud võrguseadmed ja -teenused pole juurutuses

järele jõudnud[4]. Ilmselt kestab täielik üleminek uuele standardile pärast tema ametlikku vastuvõtmist veel mõnda aega. Senikaua soovitame ühilduvuse tagamiseks kasutada TLS versiooni 1.2 šifrikomplekte võimalikult tugevate krüptograafiliste primitiividega. Näiteks sobivad

- `TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384`,
- `TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384`,
- `TLS_DHE_RSA_WITH_AES_256_GCM_SHA384`.

Igal juhul tuleb keelata teadaolevalt nõrkade primitiivide (nt DSA, MD5, SHA-1) kasutamine.

[4] `https://blog.cloudflare.com/why-tls-1-3-isnt-in-browsers-yet/`

3 Eesti ID-kaardi nõrkus

3.1 Probleemi olemus

2017. aastal avaldasid Nemec, Sys jt oma uurimuse RSA võtmete genereerimisest mitmetel laialt levinud riistvaraplatvormidel [21]. Oluline nõrkus leiti ka Infineoni kiibil, mis on Eesti ID-kaardi aluseks. Järgnevalt anname lühikese ülevaate probleemi krüptograafilisest sisust.

RSA võtmepaari loomine algab kahe enamvähem võrdse bitipikkusega algarvu p ja q genereerimisest, misjärel avalik moodul arvutatakse kui nende korrutis $N = pq$. Näiteks kui moodul N peab sisaldama 2048 bitti, valitakse nii p kui ka q umbes 1024-bitised.

Kahjuks pole suurte algarvude genereerimine lihtne ülesanne ning selle lahendamiseks kasutatakse piiratud arvutusvõimsusega platvormidel (nagu kiipkaardid) mitmeid optimeerimisvõtteid. Just Infineoni kiibil kasutatud optimeerimismeetodit Nemec, Sys jt ründasidki.

Nende uuringud algasid tähelepanekust, et Infineoni kiibi poolt genereeritud avalike moodulite jäägid jagamisel väikeste algarvudega ei ole ühtlaselt jaotatud. Edasine analüüs näitas, et algarve p (ja q) genereeritakse vastavalt valemile

$$p = k \cdot M + (65537^a \bmod M) \ , \tag{1}$$

kus M on n esimese algarvu korrutis. Seejuures on n valitud nii, et arvu M bitipikkus oleks umbes sama, mis p ja q oma. Näiteks 1024-bitiste algarvude genereerimiseks leitakse M esimese 126 algarvu korrutisena; niisiis on arv $M = 2 \cdot 3 \cdot 5 \cdot \ldots \cdot 701$ ise 961-bitine.

Arvu $N = pq$ naiivne tegurdamismeetod oleks valemis (1) erinevate a väärtuste läbiproovimine. Vastava suuruse k saab leida Coppersmithi algoritmiga [18]. Paraku nõuab see lähenemine arvu a jaoks $\mathrm{ord}_M(65537) \approx 2^{254}$ kandidaadi läbivaatamist (kus $\mathrm{ord}_M(65537)$ tähistab arvu 65537 *järku* modulo M).

Samas on Coppersmithi algoritmi rakendamisel valikuruumi. Nimelt läheb selle algoritmi edukaks rakendamiseks vaja infot ainult $\frac{1}{4}$ osa kohta mooduli N bittidest. Seega saab tegurdamiseks asendada arvu M tema mingi jagajaga M', millel oleks vähemalt $\frac{\log_2 N}{4}$ bitti, kuid mille jaoks $\mathrm{ord}_{M'}(65537)$-elemendiline hulk oleks mõistliku ajaga läbi vaadatav. Paneme tähele, et ka niisuguse M' korral saab arvu p esitada kujul

$$p = k' \cdot M' + (65537^{a'} \bmod M') \ .$$

Arvu M' täpset väärtust pole artiklis [21] antud, kuid selle leidmise meetodit kirjeldatakse külladase üksikasjalikkusega. Samuti märgime, et hea aeg-edukus suhte saamiseks ei pea M' olema optimaalne.

Artikli [21] autorid väidavad, et nende poolt 2048-bitiste moodulite tegurdamiseks leitud M' väärtuse korral on arvu a' otsinguruumi suurus umbes 2^{34}. Arvestades Coppersmithi algoritmi rakendamisele kuluvat aega (umbes $0,2$ sekundit), on ühe mooduli keskmine oodatav tegurdamisaeg umbes 140 protsessoriaastat. Kui maksta tuleks ainult arvutamisele kulutatud elektri eest, tähendaks see umbes 1000-dollarilist kulu. Dan Bernstein ja Tanja Lange on väitnud, et seda rünnet saab veel $5 \ldots 25\%$ efektiivsemaks muuta[5].

3.2 Probleemi lahendamine ja õppetunnid

ID-kaart on Eestis laialdaselt kasutusel. Paljud protsessid sõltuvad riikliku eID toimimisest ning on mitmeid süsteeme, mis toetavad vaid ID-kaarti (ja mitte mobiil-ID-d). ID-kaardi "väljalülitamine" oleks tõsiselt kahjustanud riigi ja ettevõtete toimimist. Ilmselt oleks tulnud kasutusele võtta ebaturvalisemaid asendusmeetmeid, mitmed toimingud oleks läinud hinnanguliselt palju aeglasemaks ja kulukamaks.

Kaartide väljavahetamine turvaveata kaartide vastu oleks võtnud väga kaua aega. Kõigepealt oleks tulnud uued kaardid luua: valida kiip, programmeerida ja testida rakendus, kõik sertifitseerida ning tellida uue kiibiga toorikud. Seejärel oleks tulnud käivitada kaartide väljavahetamine, mille läbilaskevõime oleks olnud piiratud PPA klienditeenindajate arvuga. Tõenäoliselt oleks see protsess kestnud aasta kui mitte rohkem. Lisaks tuleb arvestada, et leping senise teenusepakkujaga oli lõppemas ning see oleks selle pikaajalise protsessi veel keerukamaks muutnud.

Lahendus

Alternatiivina tuli välja töötada lahendus, mis võimaldaks vigasest kohast mööda minna ning olemasolevad kaardid uuendada. Vigane koht ID-kaardi kiibis oli RSA võtmete genereerimine. ID-kaardi kasutamine kvalifitseeritud digitaalallkirja andmise vahendina nõuab, et salajane võti genereeritaks kiibis (vt ID-kaardi sertifitseerimispoliitika [10] jaotis 6.1.1). Seega ei tulnud kõne alla ka võimalus genereerida võtmed kaardist väljas ning need siis kaarti importida. Pikemate (näiteks 3072-bitiste) RSA võtmetega jätkamine ei olnud võimalik, sest kaart ei olnud suuteline niisuguseid võtmeid genereerima. Lisaks tuli otsused vastu võtta ajal, mil turvanõrkust kirjeldav artikkel ise polnud veel avalikult kättesaadav ning kõik ründe nüansid polnud teada ning puudus kindlus pikemate võtmete turvalisuse osas. RSA teostuse kasutamisest selles konkreetses kiibis tuli loobuda. Õnneks pakub ID-kaardi riistvara võimalust kasutada elliptkõveratele tuginevaid krüptosüsteeme ning nende teostusel teadaolevaid turvanõrkusi ei ole. Lahenduseks oli üleminek elliptkõveratele.

ID-kaardi riistvara poolt toetatud elliptkõverate seast valiti kasutamiseks NIST-i kõver P-384. Valiku peamiseks kriteeriumiks oli lahenduse toetatus operatsioonisüsteemides ja rakendustes, aga samuti NSA soovitused (vt jaotis 2.2). Mitte-NIST-i kõverate tugi jõuab teostustesse kahjuks väga aeglaselt. Märgime siinkohal, et tulevikku silmas pidades on oluline nõuda uute ja paremate krüptograafiliste primitiivide kasutuselevõttu tootjate poolt. Tulevikus tekkivate võimalike probleemide lahendamine võib nõuda praegu kasutatavate algoritmide täielikku väljavahetamist ning selleks tuleb valmis olla.

Kuna mobiil-ID oli juba mõned aastad varem elliptkõveratele üle läinud, oli suur osa ökosüsteemist (sealhulgas digitaalallkirja rakendused) üleminekuks valmis. Isegi isikut tõendavate

[5] `http://blog.cr.yp.to/20171105-infineon4.txt`

dokumentide sertifikaatide profiili tuli muuta vaid minimaalselt – sertifikaadi omaniku avaliku võtme kirjelduse väärtusena lubati lisaks elliptkõver P-384 (tänu mobiil-ID-le oli P-256 juba varem lubatud) [9]. Samuti lisati sertifikaadi laienduste kohta täpsustus, mis ei võimalda elliptkõverate põhiseid autentimissertifikaate otse krüpteerimiseks kasutada.

Olemas oli ka kaartide kauguuendamise süsteem, mida oli varem kasutatud vigaselt kodeeritud sertifikaatide väljavahetamiseks. Probleemiks kujunes läbilaskevõime – kauguuendamise süsteem ei olnud mõeldud nii suure arvu ID-kaartide üheaegseks uuendamiseks.

Lisaks kaardirakendusele oli vaja muuta ka ID-kaardi draivereid operatsioonisüsteemide jaoks, töötada välja uus lahendus failide krüpteerimiseks-dekrüpteerimiseks ning uuendada teenusepakkujate veebirakendusi. Töö käigus tuli välja ID-kaardi mitmeid huvitavaid kasutusviise, mis olid ühel või teisel moel RSA-spetsiifilised ning vajasid elliptkõverate toetamiseks järeleaitamist.

Üldiselt toetasid uuemad operatsioonisüsteemid ja veebibrauserid elliptkõveraid hästi. Probleeme esines mõnel juhul riistvaralistel krüptoseadmetel olevate elliptkõverate kasutamisega.

Põhimõttelise lahenduse jaoks sobivaid kiiresti teostatavaid variante ei olnud eriti palju, mistõttu nende seast valiku tegemine oli lihtne. Põhiaeg kulus ID-kaardi baastarkvara loomisele ja testimisele koos kaardirakendusega, uuendussüsteemi ümberhäälestamisele ning teenusepakkujate süsteemide uuendamisele ja elliptkõverate toega varustamisele.

Kriisi praktilised tagajärjed

Eestis on juriidilist jõudu omavad digitaalallkirjad juba digitaalallkirjade seadustamisest alates olnud varustatud ajatempliga, mis aitab tagada allkirjastatud dokumentide autentsust. Alati on võimalik tõestada, et dokument allkirjastati enne teatud ajahetke, näiteks sertifikaadi tühistamise või nõrkuse avalikustamise hetke. Kui tulevikus tekib vaidlusi mõne konkreetse digitaalallkirja kehtivuse üle, tuleb nende juhtumitega eraldi tegeleda. Nendes vaidluses võib ajatemplitel olla otsustav tähendus, mis on hea näide sellest, kuidas põhjalik töö digitaalallkirja seaduse loomisel ja rakendamisel võib end positiivselt ära tasuda.

ID-kaardi tänast autentimisfunktsiooni kriis enam ei mõjuta – kui teenusepakkujad kontrollivad korrektselt sertifikaatide kehtivust, ei pääse ebaturvaliseks muutunud ja nüüdseks peatatud või tühistatud sertifikaatide alusel keegi e-teenustele ligi. Kui tulevikus tekib põhjendatud kahtlusi autentimisfunktsiooni kuritarvitusest ajavahemikus 2014–2017 selle konkreetse nõrkuse tõttu, siis tuleb ka nende juhtumitega eraldi tegelda.

Kõige suuremaks probleemiks jäävad krüpteeritud dokumendid, mida on aastate jooksul ID-kaardi omanikele saadetud. Nende konfidentsiaalsus on jätkuvalt ohus. Tänu kõnealusele nõrkusele on ründajal võimalik enda valdusesse saadud CDOC-faile dekrüpteerida ilma ID-kaardile juurdepääsu omamata. See eeldab hetkel veel olulist investeeringut, kuid aja jooksul muutub rünne tõenäoliselt odavamaks. See näide illustreerib selgelt fakti, et laialt kasutatava krüptorakenduse projekteerimisele ja loomisele tuleks rohkem vaeva kulutada ning pakkuda kaitset ka esmapilgul võimatuna näivate ohtude vastu.

Edaspidi tuleb kindlasti vähendada taristu sõltuvust ühest elektroonilise identiteedi platvormist, alternatiivide kasutuselevõttu tuleb laialdaselt soodustada ja propageerida. Samuti tuleb parandada eID massilise uuendamise ja asendamise protsesside läbilaskevõimet –

seda nii elektrooniliste kui füüsiliste kanalite (klienditeenindajad) osas. Riik peab partneritelt (eID teenuse osutajad) ostma ka võimekuse ja valmisoleku toimida *force majeure* tingimustes, mida kaartide kiirkorras vahetamine kahtlemata on. Uute eID platvormide valikul tuleb arvestada vajadusega kasutatavad krüptograafilised primitiivid välja vahetada, võimalusel ka niisuguste algoritmide vastu, mida platvormi loomise ajal veel olemaski ei ole.

Eraldi vajab läbimõtlemist krüpteerimine info salastamise eesmärgil. Piisavalt võimsate kvantarvutite ilmumisel tekib oht, et sideseansi või dokumendi salvestanud osapool saab andmed dekrüpteerida. Seega ei tohiks tänaste vahenditega (nt ID-kaardiga avamise jaoks krüpteeritult) edastada materjale, mille salastustähtaeg on pikem kui 10-15 aastat (vt jaotis 2.1).

Lahendust pakkuvad hübriidkrüptosüsteemid on juba täna olemas, kuid nende kasutuselevõtt eeldab integratsiooniprobleemide lahendamist. Ka nende probleemide lahendamisel tuleb arvestada võimalusega, et kasutatavad primitiivid murduvad. Salastuse tagamise mehhanismid tuleb disainida nii, et mõne kasutatava algoritmi kompromiteerumine ei seaks ohtu kogu krüpteeritud teabe konfidentsiaalsust.

3.3 ID-kaardi puhul kasutusele võetud uued krüptograafilised protokollid

Nagu ülalpool selgitatud, tuli ID-kaartidel RSA asemel kasutusele võtta elliptkõveratel põhinevad krüptosüsteemid. Üldise sissejuhatuse elliptkõverate matemaatikasse leiab huvitatud lugeja 2015. aasta aruandest [5]. Siinkohal kirjeldame täpsemalt kahte põhilist protokolli – elliptkõverate digitaalsignatuurialgoritmi ECDSA ning integreeritud krüpteerimisskeemi ECIES.

ECDSA

Elliptkõverate digitaalsignatuurialgoritmi (*Elliptic Curve Digital Signature Algorithm*, ECDSA) ülesseadmine algab parameetrite valikust. Avalikud parameetrid on määratud kasutatava elliptkõveraga P-384 (vt nt standard FIPS 186-4 [1]). Need sisaldavad endas

- suurt algarvu p ja tema poolt määratud lõplikku korpust $GF(p)$,
- kõverat defineeriva võrrandi

  $$y^2 = x^3 - 3x + b \bmod p$$

  vabaliiget b,
- baaspunkti G esitatuna oma koordinaatidega (G_x, G_y) ning
- moodustatava rühma järku n.

Võtmepaari moodustamiseks valitakse salajane kordaja $d \in [1, n-1]$ ja arvutatakse avalik punkt

$$Q = dG \ ,$$

kus dG tähistab punkti G korrutamist täisarvuga d (vt [5]). Võtmepaari salajaseks komponendiks saabki täisarv d ning avalikuks komponendiks punkt Q.

Digitaalsignatuuri moodustamine on kirjeldatud standardis SEC1 [2]. Sõnumi M signeerimiseks tehakse järgmised sammud.

1. Valitakse juhuarv $k \in [1, n-1]$ ning leitakse punkt $kG = R = (x_R, y_R)$.
2. Arvutatakse $r = x_R \bmod n$. Kui $r = 0$, minnakse tagasi sammule 1 ning valitakse uus k.
3. Leitakse sõnumi M räsi $e = H(M)$.
4. Arvutatakse
$$s = k^{-1}(e + rd) \bmod n \ .$$
Kui $s = 0$, tuleb sammust 1 uuesti alustada.
5. Signatuuriks on paar $S = (r, s)$.

Sõnumi M signatuuri (r, s) kontrollimiseks tuleb teha järgmised sammud.

1. Kontrollitakse, kas arvud r ja s kuuluvad vahemikku $[1, n-1]$. Kui see nii ei ole, väljastatakse veateade.
2. Arvutatakse sõnumi M räsi $e = H(M)$.
3. Arvutatakse
$$u_1 = es^{-1} \bmod n \quad \text{ja} \quad u_2 = rs^{-1} \bmod n \ .$$
4. Leitakse
$$R = (x_R, y_R) = u_1 G + u_2 Q \ .$$
Kui R on rühma neutraalne element, antakse veateade.
5. Kontrollitakse, kas
$$r = x_R \bmod n \ .$$
Kui jah, on signatuuri kontroll edukas; kui ei, väljastatakse veateade.

ECDSA protokollile tuginevad nii ID-kaardi signeerimis- kui ka autentimisfunktsionaalsus. Teise osapoole autentimiseks genereerib autentija värske juhuarvu (nonsi) ning laseb autenditaval selle autentimisvõtme abil signeerida. Kui signatuuri kontroll õnnestub, loetakse teine osapool autendituks.

ECIES

Erinevalt RSA krüptosüsteemist ei toeta elliptkõverate teostus ID-kaardil otseselt dekrüpteerimisoperatsiooni. Seetõttu tuleb krüpteerimis-dekrüpteerimisfunktsionaalsus ehitada võtmevahetuskihi peale. Tulemusena saadav konstruktsioon kannab nime integreeritud krüpteerimisskeem (*Elliptic Curve Integrated Encryption Scheme*, ECIES) ning see on Eesti ID-kaardi uue CDOC-vormingu krüptograafiliseks aluseks. Seda skeemi kirjeldab detailsemalt standard SEC1 [2].

ECIES protokoll koosneb sisuliselt kahest osast. Kõigepealt tehakse Diffie-Hellmani võtmevahetus, kus sõnumi saaja võtmepool on staatiline kõvera punkt $Q = dG$, mis pärineb tema avaliku võtme sertifikaadist. Sõnumi saatja valib dünaamilise (iga krüpteerimise jaoks uue) salajase väärtuse $c \in [1, n-1]$ ning arvutab jagatud saladuse

$$cQ = cdG \ .$$

Sellest saladusest tuletatakse deterministliku võtmetuletusfunktsiooniga (*Key Derivation Function*, KDF) sümmeetrilise krüptoalgoritmi võti K, mida kasutatakse sõnumi krüpteerimiseks.

Koos krüpteeritud sõnumiga saadetakse saajale ka kõvera punkt cG (tuletame meelde, et baaspunkt G on avalik, kuid kordaja c on saatja saladus). Saaja kasutab oma salajast kordajat d ning leiab

$$d(cG) = cdG \ ,$$

millest ta saab KDF-i abil dekrüpteerimiseks vajaliku salajase võtme tuletada.

ECIES protokolli teiseks osaks ongi (de)krüpteerimine. Põhimõtteliselt saaks võtit K otse sõnumi krüpteerimiseks kasutada, kuid Eesti ID-kaardi puhul tuleb toetada ka stsenaariumit, kus üks fail krüpteeritakse mitmele saajale. Seda on mõistlikum korraldada nii, et fail krüpteeritakse üks kord universaalse värske transpordivõtmega ning seejärel krüpteeritakse transpordivõti iga saaja B_i jaoks eraldi moodustatud võtmega K_i.

SEC1 standard [2] määrab krüpteerimisel ka sõnumiautentimiskoodi kasutamise. Eesti ID-kaardi puhul on see uues CDOC-vormingus asendatud autenditud krüpteerimisrežiimiga AES-GCM.

4 Plokiahelad

Maailma mastaabis aga on ühed kiiremini arenevad krüptograafiarakendused plokiahelad. Käesolev peatükk annab plokiahelatest süstemaatilise ülevaate ning aitab lugejal otsustada, kas ja milline plokiaheltehnoloogia tema vajadusi võiks rahuldada. Täiendava panusena on ülevaate koostamise käigus loodud eestikeelne plokiahelate terminoloogia (Tabel 3).

4.1 Mõiste

Plokiahel (*block-chain*) on järjestikustest andmeplokkidest koosnev andmestruktuur. Plokiahela iga järgmine plokk luuakse kas iga fikseeritud ajavahemiku möödudes või mingi muu sündmuse, näiteks eduka kaeve (*mining*) toimudes.

Plokkide sisu kui andmestruktuur esitab mingil kokkulepitud viisil kodeeritud arvestusraamatut (*ledger*), kus registreeritakse sündmusi, millel võib olla informatiivne, äriline või õiguslik tähendus. Plokiahela vormingureeglid on plokiahela haldajatele ja kasutajatele teada.

Plokiahela terviklust kaitstakse iteratiivse räsimisega. Igale plokile arvutatakse räsi, rakendades ploki andmetele ja eelmise ploki räsile mingit räsifunktsiooni. Räside endi tervikluse kaitseks kasutatakse kas

- digitaalsignatuure, st mingil viisil (avalike võtmete kaudu) volitatud isikute digitaalallkirju
- räsitud ajatembeldust, mis kasutab publitseerimismehhanismi, või
- interaktsioonita ajatembeldust (*non-interactive time-stamping*), mis seisneb plokkide erilistes vormingureeglites, mis muudavad korrektse ploki loomise arvutuslikult keerukaks ülesandeks

Plokiahelat säilitatakse ja hallatakse tavaliselt hajusraamatu (*distributed ledger*) kujul, kus mitu osapoolt hoiab arvestusraamatu koopiat. See eeldab komponentide vahelise konsensusprotokolli kasutamist.

Plokiahelate üks populaarsemaid rakendusi on krüptoraha (*cryptocurrency*), kus arvestusraamat esitab kasutajate kontoseise ja omavahelisi makseid rahaühikutes, mis on defineeritud arvestusraamatu enda sees ja mis ei tarvitse olla kuidagi seotud traditsioonilise rahaga nagu dollar, euro jne. Sellest hoolimata võib krüptoraha olla börsil kaubeldav ja vahetatav traditsiooniliseks rahaks.

Tuntuim krüptoraha süsteem on *Bitcoin*, milles kasutatakse rahaühikutena bittmünti (*bitcoin*) ja satoshit (sajamiljondik bittmünti).

Plokiahelasüsteemid võivad olla kas:

- **tsentraalsed teenused**, kus arvestusraamatut hallatakse tsentraalse teenusena ühe juriidilise isiku poolt. Näiteks Guardtime'i süsteem.
- **loalised hajusraamatud** (*permissioned ledgers*), kus teenuse osutamine on hajutatud mitme fikseeritud osapoole vahel, kes tegutsevad lepingu või muu õigusakti alusel. Näiteks pankadevahelisi tehinguid võimaldav Ripple, mida haldavad finantsasutused, või globaalset identiteeti loov Sovrin.
- **loatud hajusraamatud** (*permissionless ledgers*), kus teenuse osutajad ei ole fikseeritud ja põhimõtteliselt võib igaüks hakata teenuse halduriks. Näiteks Bitcoin, Ethereum, jt.

Enamik loatuid hajusraamatul põhinevaid plokiahelasüsteeme sisaldab sõltumatut krüptoraha. Põhjus on selles, et halduritevahelise lepingu puudumise tõttu ei ole enamasti olemas muid motivaatoreid tagamaks plokiahela vabatahtlikku haldust.

4.2 Ajalooline taust

Nüüdisaegsed plokiaheltehnoloogiad põhinevad mitmele ammutuntud krüptograafiaga seotud kontseptsioonile:

Räsifunktsioonid tekkisid 1970-ndatel aastatel seoses Merkle'i [20], Rabini [24], Yuvali [27] jt. töödega.

Räsitud ajatemplid. Tekkisid 1990-ndate aastate alguses Haberi, Bayeri ja Stornetta töödega [15].

Konsensusprotokollid. Hakati uurima juba 1970-ndatel aastatel. Bütsantsileppe mõiste tekkis 1980-ndate aastate algusest seoses Pease'i, Shostaki, Lamport'i jt. töödega [22].

Elektrooniline sularaha. Tekkis juba 1980-ndate aastate alguses seoses Chaumi jt. töödega [17].

4.3 Plokiaheltehnoloogia kihid

Plokiahela võib jagada järgmisteks kihtideks:

- **Võrgukiht**. Programne ja riistvaraline reeglistik, mis teostab uute komponentide liitumist, liitunud komponendi ühendumist võrguga, sõnumite saatmist teistele komponentidele ja sõnumite vastuvõtmist teistelt komponentidelt.
- **Plokilevituskiht**. Programselt teostatud reeglistik plokkide edastuseks võrgukomponentide vahel ja arvestusraamatule uute plokkide lisamiseks.
- **Semantikakiht**. Reeglid, mis kirjeldavad ploki kui arvestusraamatu osa vormingut ja naaberplokkide omavahelisi seoseid, ning samuti ka algoritme vormingu ja seoste korrektsuse kindlakstegemiseks.
- **Rakenduskiht**. Soovitud funktsionaalsust teostav kood, mis esitatakse semantikakihi reeglitega lubatud programmeerimiskeeles, nagu näiteks keel *Solidity* plokiahelasüsteemis *Ethereum*.

4.4 Arvestusraamat

Arvestusraamat (*ledger*) on pidevalt täienev elektrooniline dokument, milles registreeritakse informatiivse, ärilise või õigusliku tähendusega sündmusi. Arvestusraamatu vormingureeglid sõltuvad kasutusvaldkonnast. Organisatsioonilises plaanis on arvestusraamatuga seotud

- **Kasutajad**: isikud või mis tahes olemid, kes/mis lisavad arvestusraamatusse kirjeid vastavalt vajadusele. Kasutajaid identifitseerib arvestusraamatus tavaliselt nende avalik võti, mis enamasti on seotud traditsiooniliste digitaalsignatuuridega nagu RSA, DSA, ECDSA, vms. Avalikku võtit või selle räsi käsitletakse enamasti ka kui kontonumbrit.
- **Haldurid**: isikud, kes tagavad arvestusraamatu korrektsuse, säilimise ja toimimise. Haldurid aktsepteerivad ja lisavad arvestusraamatusse vaid vormingureeglitele vastavaid kirjeid. Võib juhtuda, et haldurid peavad ise vastavalt vajadusele lisama arvestusraamatusse andmeid, kui vormingureeglid seda nõuavad.

Tehnilises mõttes vaadeldakse haldureid kui automaatselt toimivaid olemeid. Õiguslikus mõttes võivad haldurid olla nii juriidilised kui füüsilised isikud, kusjuures nende osalemist arvestusraamatu haldamisel ei tarvitse reguleerida ükski õigusakt. Näiteks Bitcoini süsteemi haldurid tegutsevad vabatahtlikuse alusel ja enamasti anonüümselt.

Arvestusraamatu moodustamine toimub plokiaheltehnoloogias plokkide kaupa järgmiste sammudena:

1. Kasutajad saadavad haldurile kirjeid.
2. Haldur kontrollib saabuvate kirjete korrektset vormingut ja salvestab korrektsed kirjed.
3. Haldur valib mingi hulga salvestatud kirjetest (näiteks kõik) ja moodustab neist ploki ja lisab ploki arvestusraamatusse.

Kasutajaid võib vastavalt nende tegevusele jagada omakorda kahte liiki:

1. **Passiivsed kasutajad**, kes vaid vaatlevad arvestusraamatu sisu ja võivad selle põhjal teha äri- ja haldusotsuseid.
2. **Aktiivsed kasutajad**, kes lisaks vaatlemisele salvestavad arvestusraamatusse ka enda poolt loodud kirjeid.

Oluline on siin märkida, et ehkki haldurite tegutsemist arvestusraamatu pidamisel ei tarvitse reguleerida ükski õigusakt, võib arvestusraamatu sisul endal olla õiguslik tähendus, juhul kui kasutajad (näiteks Eesti riik) on nii määratlenud.

Näiteks, kui arvestusraamat esitab kasutajate vahelisi elektroonilisi makseid ja kontoseise, siis kirjed võivad koosneda komponentidest nagu makse teostaja, saaja, makse suurus, jne. Arvestusraamatu vormingureeglid võivad nõuda, et:

- makset esitavale kirjele peab olema lisatud makse teostaja digitaalallkiri (näiteks traditsioonilise signeerimisalgoritmiga, nagu RSA, ECDSA, vms.),
- makse suurus ei saa ületada teostaja kontol oleva raha hulka,
- vms.

Kui kontoseise esitatakse eraldi kirjetena, siis lisades arvestusraamatule vormingukohase maksekirje, peab haldur muutma vastavalt ka kontoseise kirjeldavaid kirjeid.

Formaal-loogiliselt saab vormingureegleid esitada tingimusena $\mathcal{P}$, nii et $\mathcal{P}(L)$ on tõene parajasti siis, kui arvestusraamatu sisu L vastab vormingureeglitele. Vormingureegleid esitava tingimuse $\mathcal{P}$ kontroll võib olla keerukas arvutusülesanne, seda eriti juhul, kui kasutajad ise saavad luua keerulisi tingimusi sisaldavaid kirjeid, näiteks kui vorming lubab nn. lepingumonitore (vt. 4.5).

Tingimuse $\mathcal{P}$ kontroll võib olla küll keerukas arvutusülesanne, kuid $\mathcal{P}(L)$ tõesus sõltub ainult arvestusraamatu L sisust (st tema bittesitusest) ja mitte ainsastki välisest tingimusest, näiteks kasutajate kui isikutevahelistest õiguslikest jm. suhetest.

4.5 Lepingumonitorid

Lepingumonitor (või 'nutileping", ingl. *smart contract*) on arvestusraamatu kirje, mis esitab mingit kasutajatevahelist lepingut. Lepingumonitor koosneb predikaadina esitatud tingimustest $\mathcal{P}_i$ ja programsetest juhisest $\mathcal{A}_i$ arvestusraamatu L sisu muutmiseks.

Näiteks võivad kasutajad A ja B kihla vedada, kas pärast ajahetke t moodustatava ploki räsi on paaris või paaritu arv, st kui paaris, siis A maksab B-le S ühikut ja kui paaritu, siis B maksab A-le S ühikut.

Selleks sisestatakse lepingu kirjeldus arvestusraamatusse ja mõlemad osapooled lisavad lepingule oma digitaalallkirjad, mis samuti salvestatakse arvestusraamatusse. Halduri ülesanne on lepingumonitori jälgida ja kui on möödunud ajahetk t ning järgmise ploki räsi on arvutatud, muuta vastaval viisil kasutajate A ja B kontoseise. Siin tuleb eeldada, et arvestusraamatu plokid sisaldavad ühe kirjena nende moodustamise aega, sest vastasel korral sõltuks arvestusraamatu L korrektsus ajast kui välisest parameetrist ja ei oleks defineeritav ainult L bittesitusest sõltuva predikaadina $\mathcal{P}$.

Näiteks kui sooviksime luua plokiaheltehnoloogial põhinevat sporditulemuste totalisaatorit, siis tuleks ka sporditulemused ise arvestusraamatusse salvestada. Korrektsuspredikaat ei saa muidugi sisaldada salvestatud tulemuste tegeliku tõesuse kontrolli, kuid võib nõuda, et salvestatud tulemused oleksid signeeritud ja verifitseeritavad mingi hulga fikseeritud "usaldusväärsete" avalike võtmetega.

Lepingumonitorid võivad sisaldada keerukaid arvutuseeskirju. Näiteks plokiahelasüsteemis Ethereum lubab kasutada eeskirju täisväärtuslikus (nn. Turingi täielikus) programmeerimiskeeles. Lõpmatute tsükilte vältimiseks piiratakse täidetavate käskude arvu, andes lepingumonitoris ette vastava ülempiiri. Arvestusraamatu vormingureeglid võivad ette näha ka käskude arvust tuleneva tasu, mille lepingumonitori signeerinud kasutajad loovutavad arvestusraamatu haldurile.

4.6 Hajusraamat

Kui arvestusraamatu haldureid on mitu, siis nimetatakse arvestusraamatu teostust hajusraamatuks (*distributed ledger*). Kasutajate saadetud kirjed jõuavad sel juhul kas otse või kaudselt kõigi halduriteni, kes sõltumatult haldavad arvestusraamatut.

Hajususe otstarve

Hajusust võib vaja olla peamiselt kahel põhjusel:

- **Usaldus**. Üksikut haldurit ei loeta piisavalt usaldusväärseks lahenduseks, arvestades võimalikku korruptsiooni.
- **Töökindlus**. Üksiku halduriga lahendust ei loeta piisavalt töökindlaks, arvestades võimalusega, et üks haldur võib muutuda side- ja muude probleemide tõttu kasutajaile kättesaamatuks.

Loalised ja loatud hajusraamatud

Kui haldurite arv ja nende identiteet on üksteisele teada (näiteks avalike võtmete kaudu), siis nimetatakse arvestusraamatu teostust loaliseks hajusraamatuks (*permissioned ledger*). Eeldatakse, et halduriks saamiseks peavad teised haldurid uut haldurit aktsepteerima, st talle selleks loa andma.

Kui haldurite arv ja identiteedid ei ole teada ja põhimõtteliselt igaüks võib halduriks hakata, siis nimetatakse arvestusraamatu teostust loatuks hajusraamatuks (*permissionless ledger*).

4.7 Konsensusprotokollid

Hajusraamatus ei saa välistada, et sideprobleemide tõttu ei jõua kõik kirjed kõigi halduriteni ja seetõttu võib vastuvõetud kirjete arv olla halduriti erinev. Tagamaks kõigi arvestusraamatu koopiate ühtsust, on vaja haldurite vahelist konsensusprotokolli.

Olgu $(B_1, B_2, \ldots, B_t)$ arvestusraamatu ajahetke t sisu esitus plokkidena. Siis võib konsensuse saavutamise strateegiad jagada kahte klassi:

- **Leppeprotokollid** (*agreement protocols*), kus järgmine plokk B_{t+1} lepitakse haldurite vahel kokku vaid eeldusel, et kõik eelnenud plokid $B_1, B_2, \ldots, B_t$ on kokku lepitud ja kõigil samad.
- **Nakamoto konsensus** (*Nakamoto consensus*), mis kasutab ühtset ahelate võrdlemise kriteeriumi, st kui on kaks ahelat $(B_1, B_2, \ldots, B_m)$ ja $(B'_1, B'_2, \ldots, B'_{m'})$, siis üks neist on "tugevam" ja haldurid eelistavad alati tugevaimat ahelat.

Leppeprotokollid

Leppeprotokollis osalevate haldurite $P_1, P_2, \ldots, P_n$ versioonid plokist B_{t+1} võivad olla erinevad. Olgu need vastavalt $B^1_{t+1}, B^2_{t+1}, \ldots, B^n_{t+1}$. Leppeprotokoll peab olema loodud selliselt, et piisavalt suur arv selles protokollis osalevaid korrektselt funktsioneerivaid haldureid jõuab kokkuleppele ühes ja samas B_{t+1} versioonis.

Leppeprotokollid on enamasti aja ja sõnumimahukad. Kui haldurite arv n on suur, siis muutub leppeprotokolli kasutamine kõigi n haldurite vahel ebapraktiliseks või isegi teostamatuks.

Mõnedes plokiaheltehnoloogiates (näiteks Algorand) kasutatakse seetõttu loteriimehhanismi abil moodustatavat valikkoalitsiooni $P_{i_1}, P_{i_2}, \ldots, P_{i_{n'}}$, kus $n' \ll n$.

Leppeprotokollidel põhinevate plokiaheltehnoloogiate põhipuudus on see, et leppeprotokoll peab tingimata õnnestuma enne järgmise ploki loomist. Kui see mingil põhjusel takerdub, lakkab ka kogu hajusraamatu haldus.

Nakamoto konsensus

Nakamoto konsensuse põhieesmärk on luua plokiahel $(B_1, B_2, \ldots, B_t, \ldots)$ ja saavutada võimalikult kiiresti olukord, et võimalikult suure $t' \leq t$ korral on korrektselt funktsioneerivate haldurite ahelatel üks ja seesama algus $(B_1, \ldots, B_{t'})$.

Haldur, kel õnnestus moodustada plokk B_{t+1}, saadab selle teistele halduritele, kes püüavad seda ühendada neile teadaolevate "eelmiste" plokkidega $B_t, B'_t, \ldots$, et neid ahelaisse ühendada. Õige ahela valikul rakendub ahelate võrdlemise kriteerium.

Siin on oluline aru saada, et ehkki uus plokk B_{t+1} viitab üheselt (räsiväärtuse kaudu) mingile eelmisele plokile B_t, ei tarvitse eelmise ploki B_t osas konsensust veel olla ja seetõttu on võimalik, et hiljem osutub õigeks hoopis ahel $B_0, \ldots, B_{t-1}, B'_t, B'_{t+1}$, kus plokid B'_t ja B'_{t+1} ei tarvitse kokku langeda plokkidega B_t ja B_{t+1}.

Võib juhtuda, et plokk B_t sisaldab mingit kirjet r, kuid hiljem "võitnud" plokid B'_t ja B'_{t+1} kirjet r ei sisalda. Seetõttu võib tekkida olukord, kus mingi äsja arvestusraamatusse lisatud kirje r eemaldatakse arvestusraamatust. Teatud loomulikel eeldustel saab siiski tõestada, et mida "vanem" on kirje r, seda väiksem on tõenäosus, et ta eemaldatakse.

Seda võimalust silmas pidades on halduritel tavaliselt eraldi arvepidamine saabunud kirjetest. Kui mingi kirje r eemaldatakse arvestusraamatust, siis haldur lisab selle esimesel võimalusel uuesti arvestusraamatusse.

Et Nakamoto konsensus põhineb vaid ahelate võrdlusel ja ei eelda, et kogu eelnev ahel $(B_1, \ldots, B_t)$ on sünkroniseeritud, siis on Nakamoto konsensust kasutavad konsensusprotokollid leppeprotokollidega võrreldes palju töökindlamad halbade sidetingimiste korral. Samas tuleb arvestada vastlisatud kirjete eemaldamise võimalusega ja tuleb oodata hajusraamatu "stabiliseerumist".

Bitcoini ploki tugevuskriteerium

Bitcoini süsteemis kasutatakse ploki B_i tugevuse kriteeriumina s_i ploki positiivse 256-bitise täisarvuna esitatud SHA-256 räsiväärtuse kõrgeimate nullbittide arvu, st. eelistatud on alati väiksema räsiväärtusega plokid. Kogu ahela $(B_1, \ldots, B_m)$ tugevus s on summa

$$s = 2^{s_1} + 2^{s_2} + \ldots + 2^{s_m} \ .$$

Lisaks sellele on Bitcoini süsteemis olemas kokkuleppeline tugevuse alampiir, mis muutub aja jooksul ja mis on reguleeritud selliseks, et iga kümne minuti jooksul saaks kogu halduri kasutada oleva arvutusvõimsusega moodustada keskmiselt ühe minimaaltugevust ületava ploki. Et kahe sellise ploki tekkimise tõenäosus on oluliselt väiksem, siis tavaliselt ei teki plokist B_{t+1} mitut versiooni.

Kaeve ja töötõendus

Ettenähtud tugevusega ploki leidmist nimetatakse kaeveks (*mining*). Näiteks Bitcoini süsteemis seisneb kaeve plokis B sisalduva 64-bitise parameetri ν varieerimist nii, et räsiväärtus $h(B_\nu)$ oleks võimalikult väike.

Sobiliku parameetri ν leidmist ja esitamist võib nimetada ka töötõenduseks (*proof of work*), sest kui räsiväärtuse $h(B_\nu)$ kõrgeimad k bitti on nullid, tõestab see kaudselt umbes 2^{k-1} räsimisele vastava arvutusressursi kulutamist.

Töötõendus aitab kaitsta hajusraamatu terviklust muutes raskeks "vanade" kirjete modifitseerimise, sest muutes plokki $B_{t'}$, kus $t' \ll t$ ajahetkel t, tuleks muuta ka kõiki järgnevaid plokke $B_{t'+1}, \ldots, B_t$, mis aga nõuab arvutusressurssi, et läbi teha

$$\frac{1}{2}\left(2^{s_{t'}} + 2^{s_{t'+1}} + \ldots + 2^{s_t}\right)$$

räsimist.

Majanduslikud aspektid

Loatute hajusraamatute haldurid peavad mingil viisil olema süsteemi haldamisest (näiteks kaevandamine) huvitatud. Ainuke teadaolev viis neid motiveerida on süsteemipõhise krüptoraha olemasolu. Iga uue ploki eest saab selle loonud haldur preemiaks teatud hulga krüptoraha. Selline loogika on arvestusraamatu vormingureeglitesse sisse ehitatud. Teine motiveerimisviis on haldurite võimalus võtta plokis osalenud kirjete loojailt ploki loomise maksu.

Kaeve on seotud suure ressursikuluga, mis muudab ühe ploki loomise omahinna väga kõrgeks. Näiteks Bitcoini on sisse ehitatud mehhanism, mille järgi ploki loomise preemia suurus regulaarselt väheneb. Kui ka krüptoraha enda vahetuskurss ajutiselt langeb, võib tekkida olukord, kus kaeve ei tasu end enam majanduslikult ära.

Panusetõendus

Panusetõendus (*proof of stake*) on alternatiiv töötõendusele, mis aitab parandada haldurite majanduslikku motiveeritust.

Panusetõendus eeldab, et halduritel endil on süsteemis piisaval hulgal krüptoraha ja nad on majanduslikult huvitatud süsteemi säilimisest. Ploki loomise ja selle tugevuse määramise eeskirjad luuakse nii, et haldurid, kellel on rohkem krüptoraha, saavad eelisõiguse ploki moodustamisel. Näiteks võib ploki moodustamise õiguse otsustada loterii abil, mille võitmise tõenäosus on proportsionaalne omatava krüptoraha hulgaga.

4.8 Plokiahela terviklusmehhanismide võrdlus

Analüüsime plokiaheltehnoloogiate olulisimate terviklusmehhanismide põhiomadusi, sh garantiisid, mida need mehhanismid annavad arvestusraamatu sisu volitamata modifitseerimise vastu. Vaatleme abstraktset stsenaariumi, kus plokiahel $(B_1, B_2, \ldots, B_{t'}, B_{t'+1}, \ldots, B_t)$ asendatakse plokiahelaga $(B_1, B_2, \ldots, B_{t'}, B'_{t'+1}, \ldots, B'_t)$.

Räsitud ajatembeldus

Räsitud ajatembeldusega lahenduses publitseeritakse regulaarselt plokiahela plokkide räsiväärtusi mingil arvestusraamatu enda haldusest sõltumatul viisil, näiteks avaldamine ajalehes. Kui näiteks ploki B_t räsi on publitseeritud mingil ajahetkel $T \geq t$, siis pärast seda on ahelat $(B_1, B_2, \ldots, B_t)$ võimalik muuta vaid siis, kui kas

a) muudetakse publitseeritud räsi või

b) kasutatavale räsifunktsioonile teostatakse edukas kollisioonrünne.

Töötõendus

Et iga järgmise ploki moodustamiseks on vajalik suhteliselt suur arvutusressurss C, siis uue ahela tegemiseks vajalik ressurss on $C \cdot (t - t')$. Piisavalt suure vahe $t - t'$ korral võib pidada vastavat arvutust ründajale teostamatuks.

See teostamatuse eeldus on nõrgematel alustel võrreldes räsitud ajatembeldusega seotud eeldustega, sest plokkide moodustamine ei saa olla kogu haldurite hulgale liiga keerukas (muidu ei saaks plokiahelat üldse moodustada), samas aga ei tohi see olla jõukohane ründajale. Seega peab eeldama, et ründaja ressursid on oluliselt väiksemad võrreldes haldurite koguressursiga.

Digitaalallkiri ja panusetõendus

Kui plokkide terviklus on kaitstud digitaalallkirjadega, siis uue plokiahela moodustamiseks peab kas

a) mõjutama või kontrollima märkimisväärset osa haldureist või

b) edukalt võltsima kasutatavaid digitaalsignatuuri skeeme.

Eeldatakse, et haldurid, kel on piisavalt plokiahelaga seotud krüptoraha, ei ole majanduslikult huvitatud hajusraamatu terviklust kahjustama. Seega põhinevad panusetõestusega plokiahelate terviklusmehhanismid suures osas sotsiaalteadusel.

4.9 Plokiahela vajalikkuse kriteeriumid

Esitame kriteeriumid, mis aitavad välja selgitada, kas mingi kasutajate-vahelise andmevahetusega seotud ülesande lahendamiseks

a) piisab **hajusandmevahetusest** kasutajalt kasutajale (*P2P*) ilma keskse andmebaasita,

b) piisab **tsentraalsest teenusest**,

c) piisab **loalisest hajusraamatust**, või

d) on vaja **loatut hajusraamatut**.

Üldine andmevahetusülesanne

Kasutajad $U_1, U_2, \ldots, U_n$ saadavad üksteisele sõnumeid. Igal kasutajal U_i on oma arvestusraamat L_i, milles oleva info põhjal tehakse mingeid olulisi otsuseid. Otsuseid võivad teha haldurid ja ka kasutajad (nii passiivsed kui aktiivsed). Näiteks riigiasutus määrab saabunud digitaalallkirjastatud avalduste põhjal kodanikele sotsiaaltoetusi vms.

Formaal-loogilises plaanis saab vaadelda ka koond-arvestusraamatut $L = (L_1, L_2, \ldots, L_n)$, milles kajastub kogu info osapoolte vahelise andmevahetuse kohta.

Küsimustik

Plokiahela kasutamise vajadusi selgitav küsimustik-skeem on esitatud Joonisel 1 ja koosneb järgmistest küsimustest:

1. *Kas iga kasutaja U_i saab oma otsused teha ainuüksi omaenese arvestusraamatu L_i põhjal?* Kui jah, siis ei ole süsteemi teostamiseks vajalik keskse teenuse kasutamine ja piisab hajusandmevahetusest.
2. *Kas üks juriidiline/füüsiline isik on piisavalt usaldusväärne tsentraalse teenuse osutaja?* Usaldusväärsuse kriteerium võib tuleneda kontekstist, rakendusest ja vaidluste lahendamise õiguslikust raamistikust. Samuti võib usaldusväärsus tuleneda sellest, kas toimitakse riigi õiguse järgi ja seega ka konkreetse riigi kehtivast õigusest. Kui vastus küsimusele on jah, siis piisab tsentraalsest teenusest.
3. *Kas teenuse osutamise võib usaldada fikseeritud juriidiliste/füüsiliste isikute grupile (näiteks mitu riigiasutust, pangad, jms)?* Riigi kontekstis tähendaks see sisuliselt küsimust, kas riiki kui sellist saab üldse usaldada? Kui vastus küsimusele on jah, siis piisab loalisest hajusraamatust.

Loata hajusraamatut on vaja vaid selliste ülesannete lahendamisel, kus ei riiki ega mitte ühtki eraõiguslike isikute konsortsiumit, ega ka riigiasutuste ja eraõiguslike isikute konsortsiumit ei loeta piisavalt usaldusväärseks, või siis soovitakse end kaitsta suurvõimude vastu, kes võivad mõjutada mis tahes konsortsiumi.

Mõned näidissüsteemid ja nende plokiahelavajaduste analüüs

Selles alapunktis vaatleme mõningaid lihtsustatud süsteeme ja analüüsime neid lühidalt plokiaheltehnoloogia vajaduste kontekstis. Uuritavad süsteemid on lihtsalt näited illustreerimaks lühiküsimustiku kasutamist.

Asutustevaheline andmevahetus. Riigiasutused vahetavad omavahel andmeid, mis on vajalikud nende kehtivast õigusest tulenevate ülesannete täitmiseks.

Siin piisab hajusandmevahetusest, sest iga asutus saab otsuseid teha omaenese arvestusraamatu põhjal.

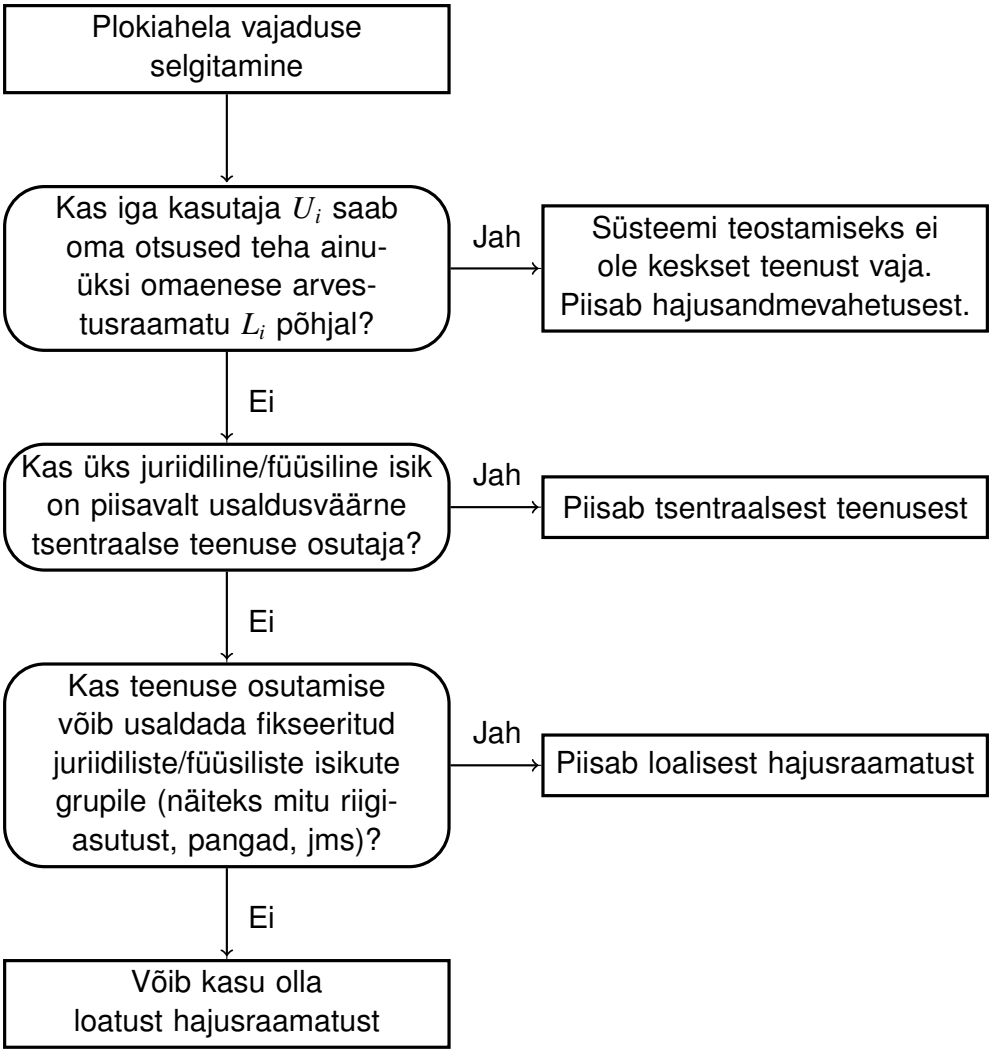
Sotsiaaltoetuste süsteem. Kasutajad (Eesti elanikud) saadavad asutustele avaldusi toetuse saamiseks, kusjuures tingimus on, et asutus A saab toetustaotluse rahuldada vaid siis, kui teised asutused seda ei tee.

Siin võib tsentraalsest teenusest kasu olla, sest otsuste tegemisel on vaja ka teiste asutuste andmeid. Võib kasu olla ka loalisest hajusraamatust, sest see aitaks leevendada ühe asutuse usaldamisest tulenevat korruptsiooniohtu.

Elektroonilise sularaha süsteem keskpangaga. Keskpank väljastab digitaalseid münte kasutajatele, kes ostavad nende abil kaupu kaupmeestelt. Tähtis on, et ühte ja sama münti ei kasutataks mitmekordselt.

Siin piisab lahendusest, kus keskpank haldab tsentraalset teenust, mis lisaks müntide väljastamisele peab arvet nende kasutuse üle, et vältida topeltkasutust.

Elektroonilise sularaha süsteem keskpangata. Nagu eelmine, kuid ei usaldata keskpanka, st mündid ringlevad vabalt kasutajate vahel. Peab olema tagatud reegel, et kasutaja U_i



Joonis 1. Plokiahela kasutusvajaduse selgitamise skeem.

saab maksta kasutajale U_j mündiga m, kui m on olnud mingil ajahetkel t kasutaja U_i valduses ja vahepeal (st ajahetkest t kuni praeguseni) ei ole U_i münti m juba maksmiseks kasutanud.

Topeltkasutuse vältimise kontroll sõltub kõikide kasutajate andmetest ja seega on hajusraamat vajalik. Kas siin on vaja ka loatut hajusraamatut, sõltub sellest, kas usaldatakse näiteks pankade koalitsiooni või mitte.

Tabel 3. Plokiahelatega seotud terminid eesti vastetega

termin	**ingliskeelne vaste**	**lk**
arvestusraamat	ledger	15, 17
bittmünt	bitcoin	15
hajusraamat	distributed ledger	15, 18
kaeve	mining	20
krüptoraha	cryptocurrency	15
lepingumonitor	smart contract	18, 18
leppeprotokoll	agreement protocol	19, 19
loaline hajusraamat	permissioned ledger	16, 19
loatu hajusraamat	permissionless ledger	16, 19
Nakamoto konsensus	Nakamoto consensus	19, 20
panusetõendus	proof of stake	21
plokiahel	block-chain	15
töötõendus	proof of work	20

Kirjandus

[1] Digital Signature Standard (DSS). FIPS PUB 186-4. http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf.

[2] SEC 1: Elliptic Curve Cryptography, 2009. http://www.secg.org/sec1-v2.pdf.

[3] Krüptograafiliste algoritmide kasutusvaldkondade ja elutsükli uuring (Report on the life cycle of cryptographic algorithms). http://www.riso.ee/sites/default/files/elfinder/article_files/kryptoalgoritmide_elutsykli_uuring.pdf, 2011. Cybernetica report no. A-60-1 (in Estonian).

[4] Krüptograafiliste algoritmide kasutusvaldkondade ja elutsükli uuring (Report on the life cycle of cryptographic algorithms). https://www.ria.ee/public/PKI/kruptograafiliste_algoritmide_elutsukli_uuring_II.pdf, 2013. Cybernetica report no. A-77-5 (in Estonian).

[5] Krüptograafiliste algoritmide kasutusvaldkondade ja elutsükli uuring (Report on the life cycle of cryptographic algorithms). https://www.ria.ee/public/RIA/Kruptograafiliste_algoritmide_uuring_2015.pdf, 2015. Cybernetica report no. A-101-1 (in Estonian).

[6] CNSA Suite and Quantum Computing FAQ, January 2016. https://www.iad.gov/iad/library/ia-guidance/ia-solutions-for-classified/algorithm-guidance/cnsa-suite-and-quantum-computing-faq.cfm.

[7] Cryptographic algorithms lifecycle report 2016). https://www.ria.ee/public/RIA/Cryptographic_Algorithms_Lifecycle_Report_2016.pdf, 2016. Cybernetica report no. A-101-3.

[8] Kryptographische Verfahren: Empfehlungen und Schlüssellängen, 2017. BSI – Technische Richtlinie, https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/TechnischeRichtlinien/TR02102/BSI-TR-02102.pdf.

[9] Sertifikaadi, CRL-i ja OCSP profiil Eesti Vabariigi isikut tõendavatel dokumentidel). https://www.sk.ee/upload/files/SK-CPR-ESTEID-ET_v8_1_20171104.pdf, 2017. Versioon 8.1.

[10] SK ID Solutions AS – Certificate Policy for ID card). https://sk.ee/upload/files/SK-CP-ID%20CARD-EN-v7_0-20171101.pdf, 2017.

[11] ITU-T Study Group 17. LS on ITU-T SG17 work on quantum-safe PKI. https://www.ietf.org/lib/dt/documents/LIAISON/liaison-2017-09-13-itu-t-sg-17-ipsecme-lamps-ls-on-itu-t-sg17-work-on-quantum-pdf, 2017.

[12] Divesh Aggarwal, Gavin K Brennen, Troy Lee, Miklos Santha, and Marco Tomamichel. Quantum attacks on Bitcoin, and how to protect against them, 2017. arXiv preprint arXiv:1710.10377, https://arxiv.org/pdf/1710.10377.pdf.

[13] Steve Babbage, Dario Catalano, Carlos Cid, Benne de Weger, Orr Dunkelman, Christian Gehrmann, Louis Granboulan, Tim Güneysu, Jens Hermans, Tanja Lange, Arjen Lenstra, Chris Mitchell, Mats Näslund, Phong Nguyen, Christof Paar, Kenny Paterson, Jan Pelzl, Thomas Pornin, Bart Preneel, Christian Rechberger, Vincent Rijmen, Matt Robshaw, Andy Rupp, Martin Schläffer, Nigel Smart, Serge Vaudenay, Fré Vercauteren, and Michael Ward. ECRYPT II Yearly Report on Algorithms and Keysizes (2011-2012). Technical report, European Network of Excellence in Cryptology II, September 2012. http://www.ecrypt.eu.org/ecrypt2/documents/D.SPA.20.pdf.

[14] Elaine Barker. Recommendation for Key Management. Part 1: General, 2016. NIST Special Publication 800-57 Part 1, Revision 4, http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-57pt1r4.pdf.

[15] Dave Bayer, Stuart Haber, and W.Ščott Stornetta. *Improving the Efficiency and Reliability of Digital Time-Stamping*, pages 329–334. Springer New York, New York, NY, 1993.

[16] Sergio Boixo, Sergei V Isakov, Vadim N Smelyanskiy, Ryan Babbush, Nan Ding, Zhang Jiang, John M Martinis, and Hartmut Neven. Characterizing quantum supremacy in near-term devices, 2016. arXiv preprint arXiv:1608.00263, https://arxiv.org/abs/1608.00263.

[17] David Chaum. *Blind Signatures for Untraceable Payments*, pages 199–203. Springer US, Boston, MA, 1983.

[18] Don Coppersmith. Finding a Small Root of a Bivariate Integer Equation; Factoring with High Bits Known. In *Advances in Cryptology – EUROCRYPT '96, International Conference on the Theory and Application of Cryptographic Techniques, Saragossa, Spain, May 12-16, 1996, Proceeding*, volume 1070 of *Lecture Notes in Computer Science*, pages 178–189. Springer, 1996.

[19] Markus Grassl, Brandon Langenberg, Martin Roetteler, and Rainer Steinwandt. Applying Grover's Algorithm to AES: Quantum Resource Estimates. In *Post-Quantum Cryptography*, volume 9606 of *LNCS*, pages 29–43. Springer, 2016.

[20] Ralph Charles Merkle. *Secrecy, Authentication, and Public Key Systems.* PhD thesis, Stanford, CA, USA, 1979. AAI8001972.

[21] Matus Nemec, Marek Sys, Petr Svenda, Dusan Klinec, and Vashek Matyas. The Return of Coppersmith's Attack: Practical Factorization of Widely Used RSA Moduli. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS '17, pages 1631–1648. ACM, 2017.

[22] M. Pease, R.Šhostak, and L. Lamport. Reaching agreement in the presence of faults. *J. ACM*, 27(2):228–234, April 1980.

[23] John Proos and Christof Zalka. Shor's discrete logarithm quantum algorithm for elliptic curves, 2003. arXiv preprint quant-ph/0301141, https://arxiv.org/abs/quant-ph/0301141.

[24] Michael Rabin. *Digitalized signatures and public-key functions as intractable as factorization*. MIT Laboratory for Computer Science, 1979.

[25] Nigel P. Smart, Vincent Rijmen, Benedikt Gierlichs, Kenneth G. Paterson, Martijn Stam, Bogdan Warinschi, and Gaven Watson. Algorithms, key size and parameters report – 2014. Technical report, ENISA, 2014. http://www.enisa.europa.eu/activities/identity-and-trust/library/deliverables/algorithms-key-size-and-parameters-report-2014/.

[26] Marc Stevens, Elie Bursztein, Pierre Karpman, Ange Albertini, and Yarik Markov. The first collision for full SHA-1, 2017. https://shattered.io/static/shattered.pdf.

[27] Gideon Yuval. How to Swindle Rabin. *Cryptologia*, 3(3):187–191, 1979.