TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond
Tarkvarateaduse instituut

Jelizaveta Vakarjuk 164041IABB

VÕREPÕHISTE VÕTMEKEHTESTUSPROTOKOLLIDE ÜLEVAADE JA ANALÜÜS

Bakalaurusetöö

Juhendaja: Ahto Buldas
Professor

Tallinn 2019

Autorideklaratsioon

Kinnitan, et olen koostanud antud lõputöö iseseisvalt ning seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on töös viidatud.

Autor: Jelizaveta Vakarjuk

[18.05.2019]

Annotatsioon

Viimaste aastate jooksul on kvantarvutite uurimise teema saanud palju tähelepanu nii suurimate arvutifirmade, kui ka valitsuste poolt ja kui piisavalt suur kvantarvuti on kunagi ehitatud, muutuvad ebaturvalisteks paljud avaliku võtme krüptograafilised algoritmid, mis kaitsevad kõike, internetipanga tehingutelt ja inimeste digitaalidentiteedilt privaatsete e-kirjadeni. Ohustatud on sellised krüptograafilised primitiivid nagu RSA ja Diffie ja Hellmani võtmekehtestus.

Käesoleva bakalaureusetöö eesmärk on analüüsida ja anda ülevaade võrepõhisest võtmekehtesusest, mis aitab kaitsta andmete turvalisust ja konfidentsiaalsust, siis kui tänapäeval kasutatavad algoritmid muutuvad ebaturvalisteks kvantarvutuse vastu. Töös kirjeldatakse viit võrepõhist võtmekehtestusalgoritmi: NewHope, Frodo, Crystals-Kyber, NTRU-HRSS ja Three Bears, mida peetakse kõige tugevamateks kandidaatideks post-kvant krüptograafia standardimisel.

Töö käigus selgus, et paljud vaadlevatest protokollidest tõeliselt võivad tulevikus täiendada või asendada Diffie ja Hellmani võtmekehtestust. Selgusid ka vaadlevate protokollide tugevused ja nõrkused, mis takistavad nende algoritmide kiiret praktilist kasutuselevõtu ning olulisemad aspektid, millega tuleb arvestada võrepõhiste võtmekehtestusprotokollide teostamisel.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 41 leheküljel, 5 peatükki, 5 joonist, 1 tabelit.

Abstract

Theoretical and Analytical Overview of Lattice-based Key Establishment Protocols

Over the last few years, the subject of quantum computer research has received much attention from major computer companies as well as from governments. If a large enough quantum computer is built, many public key cryptographic algorithms that protect everything from online banking and digital identity to private e-mails will become insecure. Under threat are protocols that rely on the difficulty of certain number theoretic problems such as integer factorization or discrete logarithm problems over various groups, these are RSA and Diffie-Hellman key exchange.

The purpose of this Bachelor's thesis is to analyze and provide an overview of lattice-based key establisment that helps to protect data security when nowaday's algorithms become insecure against quantum computing. The work describes five lattice-based key exchange algorithms: NewHope, Frodo, Crystals-Kyber, NTRU-HRSS, and Three Bears, which are considered to be the strongest candidates for standardizing post-quantum cryptography.

During the work, it was revealed that many of the protocols under review may indeed be used in the future instead of, or with, the elliptic-curve Diffie–Hellman key exchange. During the work, there are also described the weaknesses of the observing protocols, which prevent the rapid implementation of these algorithms, the strengths and the most important aspects that need to be considered in the implementation of these key protocols.

The thesis is in Estonian and contains 41 pages of text, 5 chapters, 5 figures, 1 table.

Lühendite ja mõistete sõnastik

AES	*Advanced Encryption Standard*, progressiivne krüpteerimisstandard
AVX	*Advanced Vector Extensions*, progressiivne vektorlaiend
CCA	*Chosen-Ciphertext Attack*, valitava krüptogrammiga rünne
CCA2	*Adaptive Chosen-Ciphertext Attack*, valitava krüptogrammiga adaptiivrünne
CPA	*Chosen-Plaintext Attack*, valitava avatekstiga rünne
ECB	*Electronic Code Book*, koodiraamat
ECC	*Elliptic-curve Cryptography*, elliptkrüptograafia
HTTPS	*HyperText Transfer Protocol Secure*, turvaline hüperteksti edastuse protokoll
IKE	*Internet Key Exchange*, võtmevahetus Internetis
KEM	*Key Encapsulation Mechanism*, võtmekapseldusmehhanism
LWE	*Learning with Errors*, vigadega õppimine
NIST	*National Institute of Standards and Technology*, Rahvuslik Standartide ja Tehnoloogia Instituut
NSA	*National Security Agency*, Riiklik Julgeolekuagentuur
NTT	*Number Theoretic Transform*, arvuteoreetiline teisendus
Ring-LWE	*Ring Learning with Errors*, ringidel põhinev vigadega õppimine
RSA	*Rivest-Shamir-Adleman*
SVP	*Shortest Vector Problem*, lühima vektori probleem
TLS	*Transport Layer Security*, transpordikihi turve
XOF	*Extendable-Output Function*, laienduv väljundfunktsioon

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

Tänapäeval olulisimad võrguprotokollid tuginevad peamiselt kolmele põhilisele krüptograafia funktsioonile: avaliku võtme krüptograafia, digitaalallkirjad ja võtmekehtesus. Need mängivad olulist rolli digitaalühiskonnas, näiteks võimaldades turvalise internetpanganduse, privaatsete e-kirjade vahetamise ning ka kõigi Eesti riigi e-teenuste turvalise kasutamise ja digiallkirjastamise. Need krüptograafia funktsioonid on peamiselt rakendatud kasutades Diffie ja Hellmani võtmekehtestust, RSA (*Rivest-Shamir-Adleman*) krüptosüsteemi ja elliptkõverate krüptosüsteeme. Nende turvalisus põhineb arvuteooria probleemidel, nagu täisarvu tegurdamine ja diskreetlogaritm. Peter Shor näitas 1994. aastal, et kvantarvutite abil saab kõik need probleemid tõhusalt lahendada, muutes seeläbi kõik nendel probleemidel põhinevad avaliku võtmega krüptosüsteemid ebaturvalisteks, mis ohustaks sealhulgas ka Eesti e-identiteeti ja e-teenuseid [28].

Viimastel aastatel on toimunud märkimisväärne hulk kvantarvutite uuringuid, sest kvantarvutitel on võime lahendada probleeme, mis klassikalistel arvutitel võtaksid kogu Universumi eluea. Näiteks optimeerimine ja keerukamate käitumismudelite koostamine finantssektoris. Mõned eksperdid ennustavad, et järgmise 12 aasta jooksul luuakse kvantarvutid, mis on piisavalt suured selleks, et murda paljud kasutuses olevad avaliku võtmega skeemid, sealhulgas ka need mida kasutab TLS (*Transport Layer Security*) ja HTTPS (*HyperText Transfer Protocol Secure*) turvaprotokoll [3].

Post-kvant krüptograafia eesmärk on arendada krüptograafilisi süsteeme, mis on turvalised nii kvantarvutite kui ka klassikaliste arvutite vastu ja mis on võimelised koos töötama olemasolevate võrguprotokollidega. Sujuva ja turvalise ülemineku tagamiseks praegu kasutatavatest krüptosüsteemidest nende kvantarvutuskindlatele analoogidele on vaja teha suurt uurimis- ja analüüsimistööd. Seetõttu on vaja hakata juba praegu süsteeme ette valmistama, et neid saaks muuta kvant-kindlaks.

Tänapäevaks on pakutud mitmeid võrepõhiseid võtmekehtestusalgoritme, mis on eraldi kirjeldatud teadustöödes, kuid puudub töö, mis annaks neist ülevaate, võrdleks nende

algoritmide põhiaspekte, oleks piisav selleks, et tutvuda võrepõhise võtmekehtestuse mõistetega ja pärast selle lugemist võimaldaks aru saada võrepõhise võtmekehtestuse teemalisi teadustöid. Käesoleva töö ülesanne on analüüsida võrepõhiseid võtmekehtestusalgoritme, koostada ülevaade, kus kirjeldatakse nende algoritmide tööpõhimõtet, teostusaspekte, tugevusi ja puudusi ning soovitatavaid parameetreid ja võrrelda algoritmide teostuse tulemusi ning uurida, kas on võimalik vaadlevate protokollide reaalelus kasutamine eliptkõverate Diffie ja Hellmani võtmekehtestuse asemel või koos sellega.

Kuna post-kvant krüptograafia uurimise teema pakub huvi mitte ainult suurfirmadele nagu IBM ja Microsoft, vaid ka mitmele Eestis tegutsevatele küberturvalisusega seotud firmadele, võib käesolev töö pakkuda olulist väärtust post-kvant krüptograafia uuringute läbiviimises.

1.1 Töö sisu ülevaade

Lõputöö esimeses peatükis kirjeldatakse töö tausta ja probleemi, püstitakse ülesanne ja antakse ülevaade tööst. Töö teises peatükis antakse ülevaade võtmekehtestusest ning post-kvant krüptograafiast ja post-kvant algoritmide peamistest peredest. Kolmandas peatükis koostakse viie võrepõhise võtmekehtestusprotokolli kirjeldus, mis sisaldab protokolli aluseks oleva matemaatilise probleemi definitsiooni, tööpõhimõtte sõnalist ja skemaatilist või algoritmilist kirjeldust, parameetrite valiku ja teastusaspekte kirjeldust. Lisaks, kahe protokolli jaoks on Google'i poolt tehtud katsed, mis on samuti kirjeldatud kolmandas peatükis. Lõputöö neljandas peatükis koostakse protokollide võrdlev tabel ja võrreldatakse omavahel eespool kirjeldatud protokolle. Töö lõpeb kokkuvõttega.

2 Ülevaade võtmekehtestusest ja post-kvant krüptograafiast

Käesolevas peatükis antakse teoreetiline ülevaade võtmekehtestusest ja selle rollist andmeside turvalisuse tagamises ning ülevaade post-kvant krüptograafia algoritmide peamistest peredest.

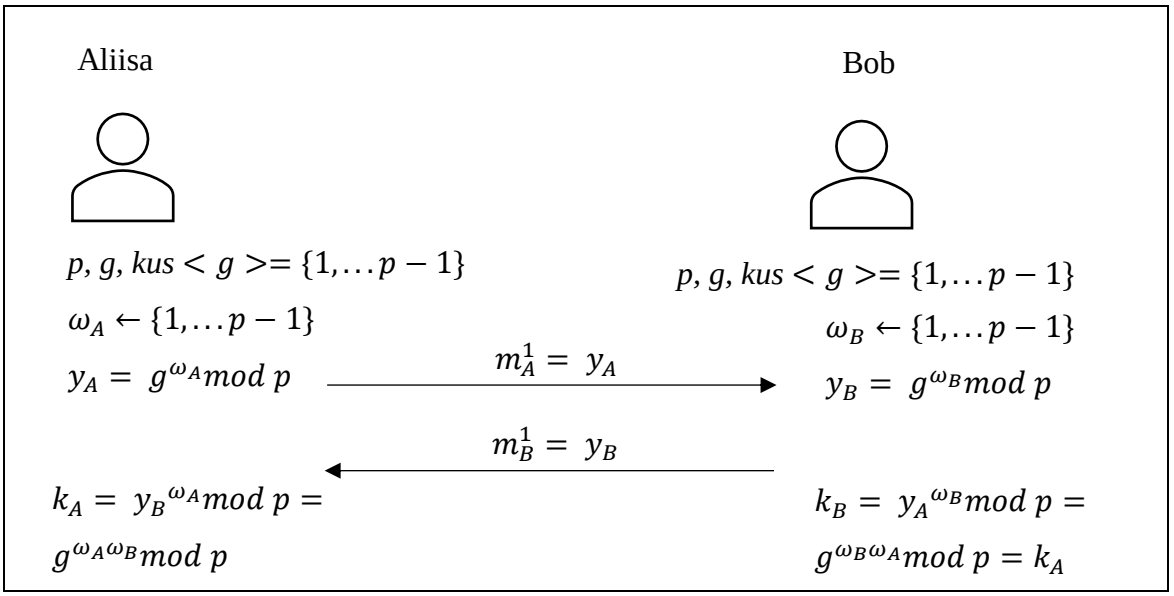
2.1 Võtmekehtestus

Krüpteerimine on üks põhivadenditest, mis aitavad tagada andmete turvalisust, kaitses konfidentsiaalseid andmeid rünnete vastu. Meetodid, mida peetakse kõige turvalisemaks ja kasutatakse kõige laialdasemalt andmete konfidentsiaalsuse ja tervikluse tagamiseks põhinevad sümmeetrilisel krüptograafial, ehk salajase võtmega krüptograafial, kus krüpteerimiseks ja dekrüpteerimiseks kasutatakse sama võtit [1]. Neid eelistatakse selle pärast, et sümmeetriliste krüpteerimisskeemide efektiivsus on palju parem kui avaliku võtme krüpteerimisskeemidel ja nende kasutamine reaalsetes rakendustes on odavam. Sümmeetrilise krüptograafia peamine praktiline probleem on turvaline võtmekehtestus suhtlevate osapoolte vahel. Seetõttu on turvalise sidekanali loomiseks äärmiselt oluline turvaline ja tõhus võtmekehtestusprotokoll.

Andmekaitse ja infoturbe leksikonis [30] on võtmekehtestus defineeritud järgnevalt: „Võtmekehtestus on võtme kokkuleppimist või võtme edastust sisaldav protokoll ühisvõtme andmiseks ühe või mitme olemi käsutusse“. Tänapäevased võtmekehtestusprotokollid kasutavad avaliku võtmega krüptograafia skeeme nagu, RSA, Diffie ja Hellmani ja ECC (*Elliptic-curve Cryptography*), mis põhinevad sellistel matemaatilistel arvutustel, mida on lihtne teostada, kuid mille pööramiseks on vaja teostamatut arvutusvõimsust. Need on arvuteooria probleemid - täisarvu tegurdamise ja diskreetlogaritmi probleem [1].

Jooninel 1 kirjeldatakse Diffie ja Hellmani võtmekehtestuse. Aliisa valib juhusliku astendaja ω_A hulgast $\{1, \ldots p-1\}$, arvutab astme $y_A = g^{\omega_A}$ tsüklilises rühmas moodustajaga g (p ja g on avalikud parameetrid) ja saadab Bobile. Bob valib oma

juhusliku astendaja ω_B hulgast $\{1, \ldots p-1\}$, arvutab astme $y_B = g^{\omega_B}$ samas tsüklilises rühmas ja saadab Aliisale. Aliisa ja Bob saavad nüüd arvutada salajase võtme $k = {y_B}^{\omega_A} = {y_A}^{\omega_B}$ [30].



Joonis 1. Võtmekehtestus prokoll Diffie ja Hellmani protokolli näitel.

2.2 Ülevaade post-kvant krüptograafiast

1994. aastal esitas Peter Shor algoritmi kvantarvuti jaoks, mis efektiivselt lahendab täisarvu tegurdamise probleemi, muutes seeläbi kõik sellel probleemil põhinevad avaliku võtmega krüptosüsteemid ebaturvalisteks [2]. Kuid sellel ajal ei olnud selge, kas kvantarvutamise tehnoloogia on kunagi teostatav või mitte. Vähem kui kümme aasta pärast, 2001. aastal tõendasin IBM teadlased Shori algoritmi korrektsuse, tegurdades $15 = 3 \cdot 5$ 7- kvantbitise kvantarvutiga [4]. Ka tänapäeval ei ole teada, millal täpselt kvantarvutid muutuvad praktiliseks, kuid sellel teemal toimub palju uuringuid ning, näiteks Michele Mosca, Waterloo ülikooli kvantarvutuste instituudi direktor, hindab, et ½ tõenäosusega on RSA-2048 2031. aastaks murtud [3].

Selleks, et tagada andmete konfidentsiaalsust ka siis kui kõik tänapäeval kasutatavad krüptograafilised protokollid muutuvad ebaturvalisteks, on vaja pöörduda post-kvant krüptograafia poole. Post-kvant krüptograafia põhineb matemaatilistel probleemidel, peale täisarvu tegurduse ja diskreetlogaritmi probleemi, mida peetakse turvaliseks nii klassikalise kui ka kvantarvuti rünnete vastu [2].

Post-kvant algoritmide suur eelis on võime töötada klassikalistes arvutites ning koos olemasolevate võrguprotokollidega. Kuid selle lahendusega säilib olukord, kus algoritmide turvalisus põhineb matemaatiliste probleemide hüpoteetilisel keerukusel ehk need algoritmid on turvalised seni kuni keegi leiab algoritmi, mis lahendab need probleemid efektiivselt kas klassikalise või kvantarvuti peal [3]. Üks ületamist vajav lisaprobleem on paljude post-kvant algoritmide oluliselt suurenenud võtmed, võrreldes tänapäeval kasutatavate algoritmidega. Selle tulemusena võib olla vajalik kohandada võrguprotokolle, nagu TLS protokoll või IKE (*Internet Key Exchange*) [2].

Post-kvant krüptograafia standardite väljatöötamine nõuab märkimisväärseid ressursse, kandidaatskeemide analüüsimiseks ja nõuab ka teadlaskonna kaasamist. 2016. aastal kuulutas NIST (*National Institute of Standards and Technology*) välja projekti „*Post-Quantum Cryptography Standardization*“, mille eesmärk on koguda, hinnata ja standardiseerida ühte või mitut post-kvant avaliku võtmega krüptograafilist algoritmi, kuhu oli esitatud 23 digitaalallkirjaskeemi ja 59 krüpteerimis/võtmekehtestus skeemi. 30. jaanuaril 2019 kuulutati välja projekti teise vooru kandidaadid, kuhu pääses kokku 26 skeemi, mille seas on mitu võrepõhist võtmekehtestusalgoritmi [2].

On olemas mitu post-kvant algoritmide perekonda: võrepõhine krüptograafia, koodipõhine krüptograafia, räsipõhine krüptograafia, mitmemuutujaline krüptograafia ning järgnevalt antakse lühiülevaade nendest perekondadest [5]. Lisaks kirjeldatakse ka selliseid algoritme, mis ei kuulu nendesse perekondadesse, näiteks supersingulaarsetel isogeensustel põhinev võtmekehtestus.

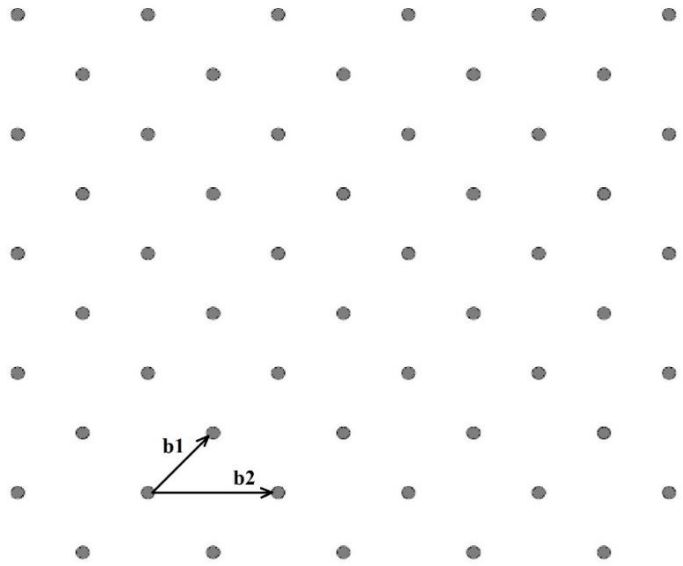
2.2.1 Võrepõhine krüptograafia

Viimaste aastate jooksul on võrepõhised probleemid saanud kõige rohkem tähelepanu kõigist arvutusprobleemidest, mida peetakse ohutuks kvantarvutuse vastu ja seda mitmel põhjusel. Esiteks, võrepõhised algoritmid on suhteliselt lihtsad, efektiivsed ja kiired [5]. Teiseks, kõiki võtmeid on sama keeruline murda nii kõige lihtsamal kui ka kõige halvimal juhul võrepõhise krüptosüsteemi parameetrite seadistamisel. RSA-sarnastel krüptosüsteemidel on oht genereerida nõrku võtmeid, mille tulemuseks on nõrk turvatase. Võrepõhist krüptograafiat kasutades on kõikvõimalikud võtmevalikud aga sama tugevad ja raskesti lahendatavad [2].

Definitsioon 2.2.1.1 [27] Võreks L nimetatakse eukleidilise ruumi $\mathbb{R}^n$ vektorite (punktide) diskreetset alamhulka, mis on kinnine vektorite liitmise ja lahutamise suhtes.

Definitsioon 2.2.1.1 [27] Võre L baasiks nimetatakse vektorite hulka B, nii et võre L iga punkt (vektor) avaldub ühesel viisil hulga B elementide täisarvulise lineaarkombinatsioonina.

Ühe võre jaoks, kui selle dimensioon on vähemalt 2, leidub lõpmatu arv baase. Joonisel 2 on esitatud võre baasiga $B = \{b_1, b_2\}$.



Joonis 2. Võre tasandil.

Võrede olulisim probleem on lühima vektori probleem (ing. *Shortest Vector Problem*, SVP), mis on NP-keeruline ja seisneb selles, et baasiga B esitatud võres leida kõige lühem mittenulline vektor [5]. Selle probleemi jaoks ei ole veel leitud efektiivset kvantalgoritmi. Praktikas ei põhine võrepõhised krüptosüsteemid SVP probleemil, vaid selle probleemi variatsioonidel, mida on samuti raske lahendada.

Käesolevas töös vaadeldakse põhjalikumalt võtmekehtestusprotokolle, mis kuuluvad võrepõhise krüptograafia perekonda.

2.2.2 Koodipõhine krüptograafia

Veaparanduskoodidel on pika aja jooksul olnud oluline roll sidetehnoloogias ning juba 1978. aastal pakuti esmakordselt välja koodidel põhinev McEliece krüptosüsteem ja

tänase päevani ei ole seda süsteemi murtud [2]. Sellest ajast alates on välja pakutud ka muid veaparanduskoodidel põhinevaid krüptosüsteeme.

Nii McEliece kui ka teiste veaparanduskoodidel põhinevate krüptosüsteemide eelis on krüpteerimisekiirus ja suhteliselt kiire dekrüpteerimine. Suurim puudus, mis takistab nende krüptosüsteemide praktilist kasutamist, on äärmiselt suured võtmed [5].

2.2.3 Mitmemuutujaline krüptograafia

Mitmemuutujalise krüptograafia algoritmid põhinevad mitmemuutujaliste polünoomide süsteemide lahendamise keerukusel üle lõplike korpuste. Viimaste aastate jooksul on välja pakutud mitmeid selliseid süsteeme, kuid paljud neist on juba murtud [2].

Kõige lootusrikkamaks mitmemuutujaliseks krüpteerimisskeemiks peetakse praegu lihtsa maatriksi (või ABC) krüpteerimisskeemi, kus kõik arvutused tehakse üle lõpliku korpuse ja dekrüpteerimisprotsess koosneb ainult lineaarsete süsteemide lahendamisest, mis teeb selle skeemi väga efektiivseks [5].

2.2.4 Räsipõhine krüptograafia

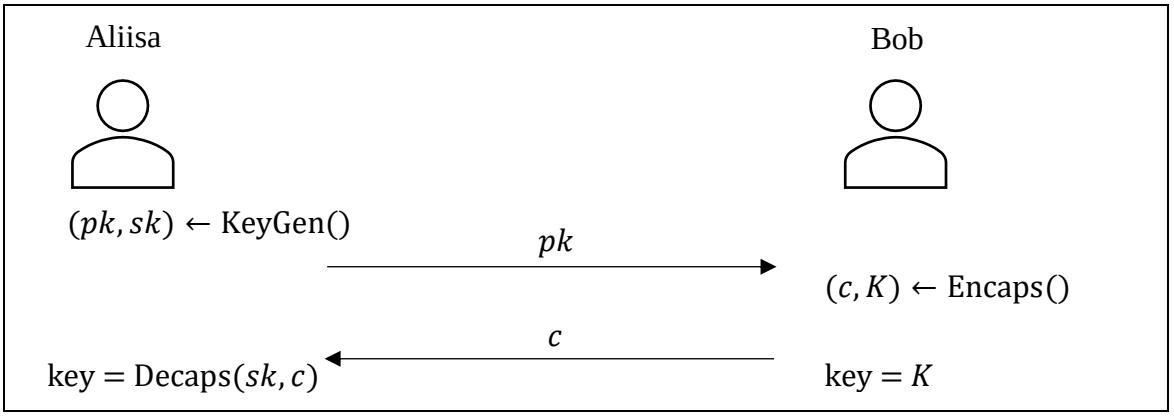
Räsipõhine krüptograafia pakub ühekordseid digitaalallkirja skeeme, mis põhinevad räsifunktsioonidel. Selliste skeemide turvalisus sõltub ainult kasutatava krüptograafilise räsifunktsiooni kollisioonivabadusest [5].

Räsipõhise krüptograafia suur eelis on krüptoskeemide paindlikus, sest neid saab kasutada erinevate räsifunktsioonidega [5]. See tähendab, et kui hetkel kasutatavas räsifunktsioonis leitakse nõrkus, siis digitaalallkirja skeemis tuleb ainult räsifunktsioon vahetada uue ja turvalisema vastu. Kuid räsipõhistel süsteemidel on ka mõned puudused, näiteks allkirjastaja peab salvestama eelnevalt allkirjastatud sõnumite täpse arvu ning kui selles tekkib viga, võib see põhjustada ebaturvalisust. Teine puudus on see, et need skeemid võivad toota ainult piiratud arvu allkirju [2].

3 Post-kvant võtmekehtestus

Selleks, et koostada põhjalik ülevaade olemasolevatest võtmekehtestusprotokollidest ja analüüsida nende teostuse parameetreid, on välja valitud need protokollid, mis pääsesid NIST projekti teise vooru. Valitud protokollide seas on nii võtmekehtestusprotokollid kui ka võtmekapseldusmehhanismid (ing. *Key Encapsulation Mechanism,* KEM), sest nad täidavad samu eesmärke, ehk võimaldavad suhtlevatel osapooltel sümmeetrilise võtme kokkuleppimist, ja neid saab ka omavahel võrrelda. Võtmekapseldusmehhanismi erinevus võtmekehtestusest seisneb selles, et ainult üks osapool genereerib salajase võtme, krüpteerib selle teise osapoole avaliku võtmega ja seejärel teine osapool dekrüpteerib võtme, kasutades oma salajast võtit. Võtmekehtestusalgoritmide kaks osapoolt osalevad mõlemad sümmeetrilise võtme loomises ja ühel osapoolel ei ole võimalik iseseisvalt võtit genereerida.

Võtmekapseldusmehhanism KEM = (KeyGen, Encaps, Decaps) on tõenäosuslike algoritmide kolmik koos võtmeruumiga K [12]. Võtme genereerimise algoritm KeyGen tagastab paari (*pk, sk*), mis koosneb avalikust võtmest *pk* ja salajasest võtmest *sk*. Kapseldusalgoritm Encaps võtab avaliku võtme *pk*, et genereerida sümmeetriline võti $K \in \mathrm{K}$ ja selle võtme krüptogramm *c*. Lõpuks, deterministlik lahtikapseldusalgoritm Decaps võtab salajase võtme *sk*, krüptogrammi *c* ja väljastab kas võtme $K \in \mathrm{K}$ või erilise sümboli $\perp$, mis näitab tagasilükkamist [12]. Võtmekapseldusmehhanismist saab otse luua võtmekehtestusprotokolli, mida kirjeldab joonis 3 [12].



Joonis 3. Võtmekapseldusmehhanismist saadud võtmekehtestusprotokoll.

Järgnevalt antakse ülevaade valitud protokollidest ja kirjeldatakse nende tööpõhimõtet, parameetrite valikut ja teostusaspekte. Need protokollid erinevad turvalisuse poolest (nt passiivne ja aktiivne turvalisus), raskete arvutusprobleemide poolest, millisel need põhinevad, aluseks olevate võrede struktuuri ning jõudluse, kiiruse ja võtmete suuruse poolest.

3.1 NewHope

NewHope on passiivselt turvaline võtmekehtestusprotokoll, mille aluseks on ringidel põhinev vigadega õppimise probleem (ing. *Ring Learning with Errors,* Ring-LWE) ja mille esitasid Erdem Alkim, Léo Ducas, Thomas Pöppelmann ja Peter Schwabe. Alljärgnev protokolli kirjeldus põhineb teadustööl [7].

Tähistus. Olgu $\mathbb{Z}$ on täisarvude ring. $R_q = \mathbb{Z}_q[X]/(X^n + 1)$ on täisarvuliste polünoomide ring mooduli $X^n + 1$ järgi, kus iga polünoomi kordaja on arvutatud mooduli q järgi. Polünoom **a** ringis Rq on esitatud kujul $\mathbf{a} = a_1 + a_2x + \cdots + a_nx^{n-1}$, polünoomi kordajad võivad olla esitatud ka vektorkujul $\mathrm{a} = (a_1, \cdots, a_n)$. Kui χ on tõenäosusjaotus üle R_q siis $x \xleftarrow{\$} \chi$ tähendab valiku $x \in R_q$ tegemist χ järgi. $U(R_q)$ tähistab ühtlast jaotust R_q üle ja $x \xleftarrow{\$} U(R_q)$ tähendab ühtlase juhusliku valiku x tegemist ringist R_q. Polünoomid on tähistatud rasvaste väiketähtedega.

3.1.1 Ring-LWE probleem

Olgu n ja q positiivsed täisarvud. Olgu χ_s ja χ_e tõenäosusjaotused üle R_q. Olgu $\mathbf{s} \xleftarrow{\$} \chi_s$, $\mathbf{e} \xleftarrow{\$} \chi_e$, $\mathbf{a} \xleftarrow{\$} U(R_q)$ ja $\mathbf{b} \leftarrow \mathbf{as} + \mathbf{e}$.

Definitsioon 3.1.1 [6] Ring-LWE otsinguprobleem (ing. *search problem*) on (n, q, χ_s, χ_e) jaoks leida **s**, kui on antud (**a**,**b**).

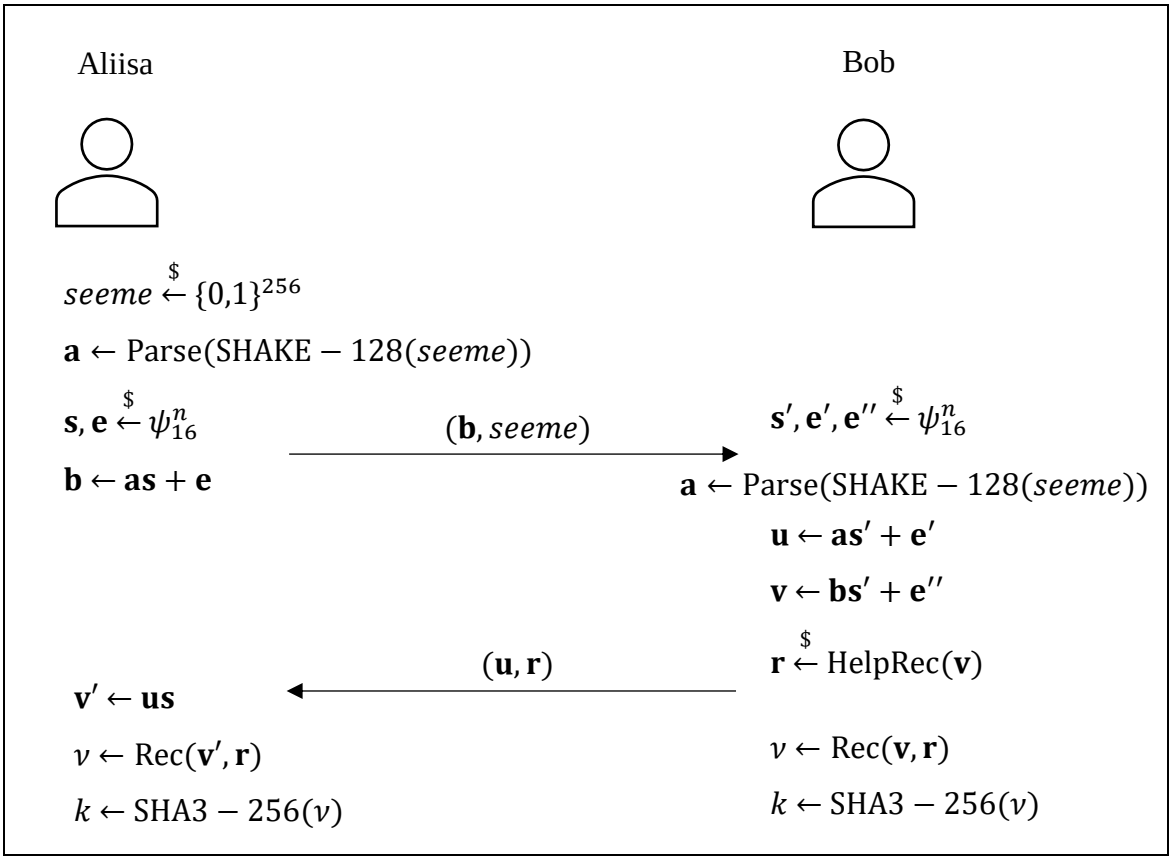
Defineerime kaks oraaklit:

- $O_{\chi_e,s}$: $\mathbf{a} \xleftarrow{\$} U(R_q)$, $\mathbf{e} \xleftarrow{\$} \chi_e$; tagasta (**a**, **as** + **e**)
- U: $\mathbf{a}, \mathbf{u} \xleftarrow{\$} U(R_q)$; tagasta (**a**, **u**)

Definitsioon 3.1.2 [6] Ring-LWE otsustusprobleem (ing. *decision problem*) on (n, q, χ_s, χ_e) jaoks eristada $O_{\chi_{e,s}}$ ja U.

3.1.2 Protokolli tööpõhimõtte

Protokolli kirjeldatakse joonisel 4.



Joonis 4. NewHope võtmekehtestus.

Aliisa alustab polünoomi **a** genereerimisega juhuslikust seemnest, genereerimiseks kasutatakse SHAKE-128, mis pakub 128-bitist post-kvant turvalisust kollisioonrünnete vastu, mille eesmärk on leida kaks erinevat originaali, mille räsid ühtivad ja originaalirünnete vastu, mille eesmärk on leida etteantud räsile vastav originaal [30].

Pärast **a** genereerimist, valib Aliisa salajase komponendi **s** ja mürapolünoomi **e**. Need on polünoomid, kus iga kordaja on võetud juhuslikust jaotusest, mis on antud juhul tsentreeritud binoomjaotus. Pärast arvutab Aliisa sõnumi **b** = **as** + **e**, mille ta saadab koos seemnega Bobile.

Bob valib oma salajased komponendid **s′**, **e′** ja **e′′** ja pärast seemne saamist genereerib polünoomi **a** ning selle abil arvutab oma osa salajasest võtmest **u** = **as′** + **e′** ja saadab selle

tagasi Aliisale. Nüüd võib Aliisa arvutada $\mathbf{v'} = \mathbf{us} = \mathbf{(as'} + \mathbf{e')s} = \mathbf{as's} + \mathbf{e's}$ ja Bob võib arvutada $\mathbf{v} = \mathbf{bs'} + \mathbf{e''} = \mathbf{(as} + \mathbf{e)s'} + \mathbf{e''} = \mathbf{ass'} + \mathbf{es'} + \mathbf{e''}$. Kuid need saadud tulemused ei ole omavahel võrdsed.

Lisatud müra on vajalik turvalisuse tagamiseks, kuid selle lisamine tähendab omakorda, et protokolli kasutatavad osapooled arvutavad erinevad väärtused. Kuigi saadud väärtused on erinevad, on need omavahel väga sarnased, sest lisatud mürapolünoomide norm on väike. See tähendab, et selleks, et protokolli kasutatavad osapooled saaksid tulemuseks ühe ja sama jagatud võtme, on vaja leida lepitusmehhanism (ing. *reconciliation mechanism*), mis aitab leida jagatud võtme, kui on antud kaks sarnast väärtust [8].

NewHope lepitusmehhanism koosneb kahest funktsioonist: HelpRec ja Rec. HelpRec funktsioon on vajalik selleks, et arvutada lepitusvektor (ing. *reconciliation vector*) **r**, mille Bob saadab oma **u** sõnumiga Aliisale. Lepitusvektori arvutamiseks teisendatakse polünoomi **v** kordajad vektoriteks. Vektor **r** on vajalik selleks, et anda teisele osapoolele lisainformatsiooni **v**-st saadud vektorite võrel paiknemise kohta, kuid pealtkuulaja ei saa sellest lisainformatsioonist midagi teada võtme kohta, sest see ütleb ainult, milline on vektori ja võrepunkti vahe, kuid mitte seda kas see punkt bittideks teisendatuna annab nulli või ühte. Rec funktsioon, kasutades lepitusvektorit **r** ning Aliisa ja Bobi poolt saadud arvutuste tulemusi **v** ja **v′**, aitab osapooltel polünoomist arvutada kindla suurusega, näiteks 256-bitise, sümmeetrilise võtme, leides vektori bitile vastandi.

Lõpuks lepitusmehhanismi väljundid räsitakse kasutades räsifunktsiooni SHA3-256. Selline lähenemine aitab vältida eristusrünnet, mis seisneb krüpteeritud andmete eristamises juhuarvudest.

Kuigi antud protokollil on olemas lepitusmehhanism, võib juhtuda et suhtlevad osapooled saavad tulemuseks erinevad väärtused. Selle sündmuse tõenäosus ei ületa 2^{-60}.

3.1.3 Parameetrite valik

Võtmekehtestusprotokoll koosneb järgmistest parameetritest:

- Moodul q
- R_q aste n

- Juhuslik jaotus, et valida **s** ja **e**

Kõik protokollis kasutatavad polünoomid, välja arvarud $\mathbf{r} \in R_4$, on defineeritud ringis $R_q = \mathbb{Z}_q[X]/(X^n + 1)$, kus $n = 1024$ ja $q = 12289$. Arvu n valik on põhjustatud vajadusest saavutada sobivat pikaajalise turvalisuse taset. Moodul $q = 12289$ on vähim algarv, mille jaoks kehtib $q \equiv 1 \bmod 2n$. See tähendab, et arvutuste lihtsustamiseks ja kiirendamiseks võib kasutada Fourier' teisenduse variandi, mida nimetatakse arvuteoreetiliseks teisenduseks (ing. *number theoretic transform*, NTT).

Juhuslikeks jaotuseks on valitud tsentreeritud binoomjaotus ψ_k parameetriga $k = 16$, sest sellest valimine on lihtne ega nõua suure täpsusega arvutusi või suuri eelarvutatud tabeleid. Lisaks, selle jaotuse teostamist on lihtne kaitsta ajastusrünnete vastu. Jaotuse dispersioon on $k/2$ ja $k = 16$ jaoks standardhälve on $\varsigma = \sqrt{16/2}$.

Selles protokollis on vaja tulemusena saada ainult 256-bitine sümmeetriline võti, kuid on olemas $n = 1024$ kordajat, kuhu saab andmeid kodeerida. Seega üks võtmebitt kodeeritakse nelja kordajasse, mis võimaldab kasutada suuremat müra parema turvalisuse saamiseks.

Antud parameetrid pakuvad vähemalt 128-bitist post-kvant turvalisust.

3.1.4 Teostusaspektid

Protokoll oli teostatud kahel erineval viisil. Esimene teostus on C keeles, mille eelis on lihtsus ja porditavus ja teine on optimeeritud teostus Intel Haswell protsessorite jaoks, kus, kasutades AVX2 (*Advanced Vector Extensions*) laiendust, on kiirendatud võtmekehtestuse mõningaid faase. Mõlemad teostused on testitud kasutades Intel Core i7-4770K (Haswell) protsessorit.

Kõige olulisem aspekt, mis aitab parandada protokolli jõudlust on NTT rakendamine, et teisendada polünoomid piirkonda, kus korrutamine on palju kiirem. Lisaks jõudluse parendamisele aitab NTT rakendamine kaitsta protokolli selliste külgrünnete vastu nagu horisontaalne võimsustarbe diferentsiaalanalüüs, mis seisneb krüptomooduli tarbitava toitevõimsuse võnkumise analüüsis, saamaks informatsiooni kasutatavate võtmete kohta, mis tavalise polünoomide korrutamise korral teeb protokolli ebaturvaliseks [9].

Protokoll ei tugine globaalselt valitud avalikule komponendile **a**, sest selle kaudu efektiivsuse suurendamine ei ole põhjendatud arvestades meetmeid, mida tuleb võtta, et võimaldada avaliku komponendi usaldusväärset genereerimist ja tagauste eest kaitsmist. Lisaks aitab **a** uuesti genereerimine iga uue võtmekehtestuse käituse jaoks vältida olukorda, kus kõik ühendused põhinevad ühel ja samal võreprobleemil.

Diffie ja Hellmani efemeerkehtestuse korral TLS'is on tihti kasutusel võtmepaari lühiajaline vahemällu salvestus, et parandada jõudlust. Kuid NewHope protokolli jaoks ohustab selline vahemällu salvestus protokolli turvalisust. Seetõttu on väga oluline, et osapooled genereeriksid uue võtme iga käituse korral uuesti.

Tsentreeritud binoomjaotusest mürapolünoomide valimine põhineb kiirel pseudojuhuarvude generaatoril. Konkreetse pseudojuhuarvude generaatori valik on puhtalt lokaalne valik. Iga kasutaja saab selle sihtotstarbelise riistvaraarhitektuuri alusel sõltumatult valida. Protokolli C-keelne teostus kasutab ChaCha20, mis on kiire, triviaalselt kaitstud ajastusrünnete eest ja mida kasutavad juba paljud TLS-i kliendid ja serverid.

Tähtis aspekt on ka sõnumite formaat, mida protokolli töötamise käigus edastatakse, need on (**b**, *seeme*) ja (**u**, **r**) sõnumid. Mõlemal juhtumil edastatakse polünoome **b** ja **u** NTT piirkonnas, need on kodeeritud nagu massiiv, mille suurus on 1792 baiti, pakitud pöördjärjestusega formaadis. Seeme on kodeeritud 32-baidise massiivina ja on konkateneeritud **b** kodeerimisega: (**b**, *seeme*) = 1792 + 32 = 1824 baiti. **r** kodeerimiseks pakitakse neli kahebitist kordajat ühte baiti ja tulemuseks on 256-baidine suurus: (**u**, **r**) = 1792 + 256 = 2048 baiti.

3.1.5 Protokolli katsetamine

7. juulil 2016. aastal tegi Matt Braithwaite postituse Google turvalisuse blogis, kus kirjeldas väikest katset post-kvant krüptograafia teostamisega TLS 1.2 protokollis, mis korraldati Google'i poolt. Katses kasutati NewHope algoritmi, mis tundus eksperimendi läbiviijaile kõige lootustandvam [23].

Lühikese aja jooksul, osa töölaua Chrome'i ja Google'i serverite vahelistest ühendustest kasutas post-kvant võtmekehtestusalgoritmi lisaks olemasolevale elliptkõverate võtmekehtestusalgoritmile. Kuna post-kvant algoritmid ei olnud veel piisavalt uuritud ja

võis juhtuda, et neid võib murda isegi klassikaliste arvutite abil, võis nende puhas kasutamine halvasti mõjutada kasutaja turvalisusele. Seega post-kvant algoritm oli lisatud olemasolevale, mis võimaldas katsetada uut algoritmi ilma kasutaja turvalisust mõjutamata, sest elliptkõverate algoritm tagas endiselt parima turvalisuse, mida tänapäeva tehnoloogia pakub. Nende algoritmide kombinatsioon oli nimetatud CECPQ1 [23].

Selle eksperimendi eesmärk oli esiteks suunata krüptoanalüütilist tähelepanu Ring-LWE probleemide perekonnale, mida Google peab oluliseks teadusuuringute valdkonnaks. Teine eesmärk oli mõõta post-kvant võtmekehtestuse TLS-is kasutamise võimalikkust, ühendades valitud post-kvant algoritmi olemasoleva elliptkõverate võtmekehtestusega: X25519, mis pakub 128-bitist turvalisust [24].

Kuna TLS-is võtmekehtestus ei olnud kunagi nii suure andmemahuga, oli täiendava võrguliikluse tõttu oodata väikest ajalist viivitust. Mediaanviivitus suurenes aga ainult millisekundi võrra, aeglase 5% viivitus suurenes 20 ms võrra ja kõige aeglasema 1% viivitus suurenes 150 ms võrra. Kuna NewHope algoritm on arvutuslikult odav, võis see viivitus olla tingitud suurenenud sõnumitest [24].

Katse tulemusena ei leitud ühtegi NewHope'i paigaldamisega seotud ootamatut takistust ning ei leitud probleeme ka selle töötamise käigus. Katse näitas, et kui tekib nõudlus, võib seda kiiresti paigaldada ja kasutusele võtta [24]. NewHope teostust testiti TLS 1.2 protokollis, TLS 1.3 teostamine on samuti võimalik, kuid keerukuse poolest erineb natuke eelmisest variandist, sest TLS 1.3 sisaldab kätlust (ing. *handshake*) puudutavaid muudatusi.

3.2 Frodo

Frodo on passiivselt turvaline võtmekehtestusprotokoll, mis põhineb vigadega õppimise probleemil (ing. *Learning with Errors*, LWE) ja pakuti välja Joppe Bos, Craig Costello, Léo Ducas, Ilya Mironov, Michael Naehrig, Valeria Nikolaenko, Ananth Raghunathan ja Douglas Stebila poolt. Alljärgnev protokolli kirjeldus põhineb teadustööl [10].

Tähistus. $\mathbb{Z}$ on täisarvude ring ja $\mathbb{Z}_q = \mathbb{Z}/q\mathbb{Z}$ on täisarvude ring mooduli q järgi. Kui χ on tõenäosusjaotus üle hulga *S*, siis $\mathbf{X} \xleftarrow{\$} \chi(S^{n\times m})$ tähendab $n \times m$ maatriksi **X** genereerimist, valides iga elemendi sõltumatult χ järgi. $\lfloor a \rceil$ on täisosa funktsioon (ing. *floor function)*, mis väljastab suurima täisarvu, mis on väiksem või võrdne *a*-ga, näiteks $\lfloor 2.4 \rceil = 2$. Lähima täisarvu reaalarvule *a* tähistatakse $\lfloor a \rceil = \lfloor a + 1/2 \rfloor$. Kahe *n*-mõõtmelise vektori **a**, **b** skalaarkorrutis üle ringi *R* on sisekorrutis $\langle \mathbf{a}, \mathbf{b} \rangle = \sum_{i=0}^{n-1} \mathbf{a_i} \mathbf{b_i} \in R$. Maatriksid on tähistatud rasvaste suurtähtedega ja vektorid rasvaste väiketähtedega.

3.2.1 LWE probleem

Olgu *n*, *m* ja *q* positiivsed täisarvud. Olgu χ_s ja χ_e tõenäosusjaotused üle $\mathbb{Z}$. Olgu $\mathbf{s} \xleftarrow{\$} \chi_s^n$, $\mathrm{e}_i \xleftarrow{\$} \chi_e$, $\mathbf{a}_i \xleftarrow{\$} U(\mathbb{Z}_q^n)$ ja $\mathrm{b}_i \leftarrow \langle \mathbf{a}_i, \mathbf{s} \rangle + \mathrm{e}_i \bmod q$, i = 1,...,m jaoks.

Definitsioon 3.2.1 [6] LWE otsinguprobleem on (n, m, q, χ_s, χ_e) jaoks leida **s**, kui on antud $(\mathbf{a}_i, \mathrm{b}_i)_{i=1}^m$.

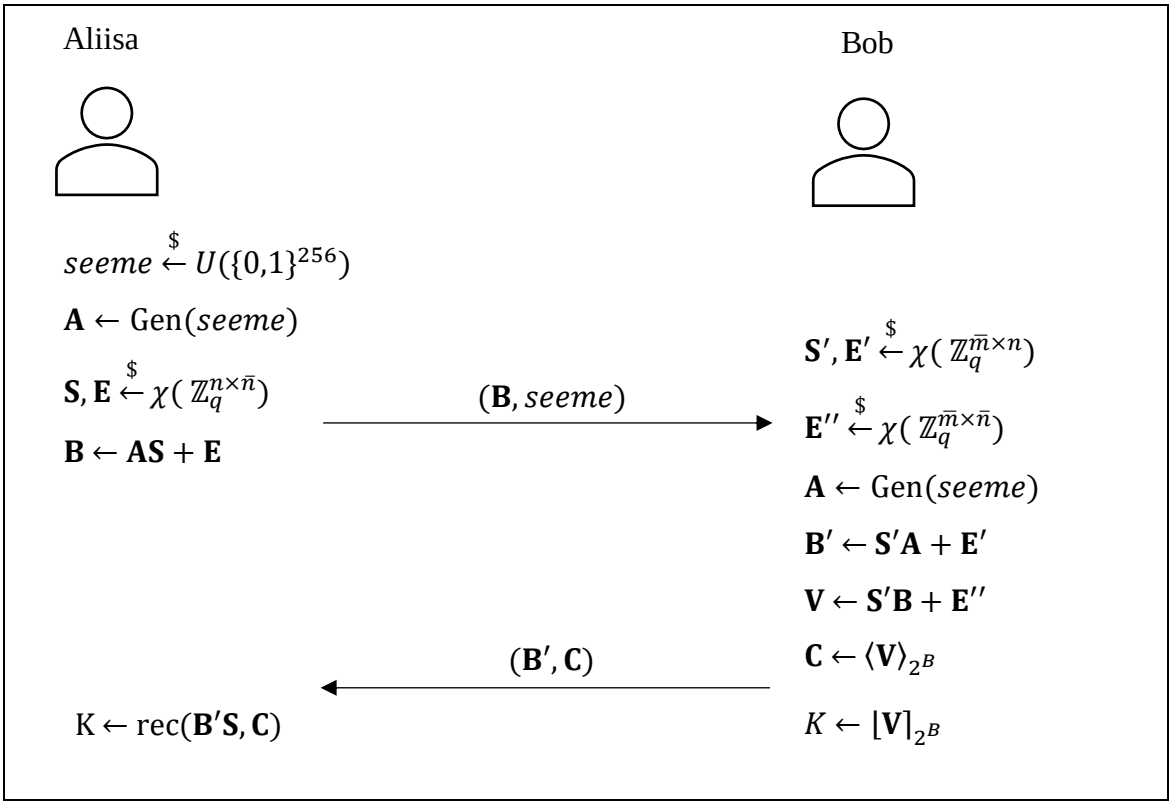
Defineerime kaks oraaklit:

- $O_{\chi_e,s}$: $\mathbf{a} \xleftarrow{\$} U(\mathbb{Z}_q^n)$, $\mathrm{e} \xleftarrow{\$} \chi_e$; tagasta $(\mathbf{a}, \langle \mathbf{a}, \mathbf{s} \rangle + \mathrm{e} \bmod q)$
- U: $\mathbf{a} \xleftarrow{\$} U(\mathbb{Z}_q^n)$, $\mathrm{u} \xleftarrow{\$} U(\mathbb{Z}_q)$; tagasta (**a**, u)

Definitsioon 3.2.2 [6] LWE otsustusprobleem on (n, q, χ_s, χ_e) jaoks eristada $O_{\chi_e,s}$ ja U.

3.2.2 Protokolli tööpõhimõtte

Protokolli kirjeldatakse joonisel 5.



Joonis 5. Frodo võtmekehtestus.

Aliisa alustab maatriksi **A** genereerimisega juhuslikust seemnest, milleks kasutatakse pseudojuhuarvude generaatorit. Pärast **A** genereerimist, valib Aliisa salajase komponendi **S** ja müramaatriksi **E**. Nende maatriksite elemendid on valitud juhuslikult, rakendades pöördvalimi meetodit, mis kasutab eelarvutatud tabeleid, mis vastavad diskreetsele jaotusfunktsioonile üle väikse intervalli. Pärast arvutab Aliisa sõnumi **B** = **AS** + **E**, mille ta saadab koos seemnega Bobile.

Bob valib oma salajased komponendid **S′**, **E′** ja **E′′** ja pärast seemne saamist genereerib maatriksi **A** ning selle abil arvutab oma osa salajasest võtmest **B′** = **S′A** + **E′** ja saadab selle tagasi Aliisale. Nüüd võib Aliisa arvutada **B′S = (S′A** + **E′)S = S′AS** + **E′S** ja Bob võib arvutada **V = S′B** + **E′′ = S′(AS** + **E)** + **E′′ = S′AS** + **S′E** + **E′′**. Saadud tulemused ei ole omavahel võrdsed, sest maatriksi $\mathbf{V} \in \mathbb{Z}_q^{\bar{m} \times \bar{n}}$ iga element on ainult ligikaudu võrdne maatriksi $\mathbf{B}'\mathbf{S} \in \mathbb{Z}_q^{\bar{m} \times \bar{n}}$ vastava elemendiga ja selleks, et osapooled saaksid tulemuseks ühe ja sama jagatud võtme, on vaja kasutada lepitusmehhanismi ja saata Aliisale lisainformatsioon $\mathbf{C} = \langle \mathbf{V} \rangle_{2^B} \in \mathbb{Z}_2^{\bar{m} \times \bar{n}}$.

Frodo lepitusmehhanism koosneb kolmest funktsioonist: ümardamise funktsioonist $\lfloor \cdot \rceil_{2^B}$, rist-ümardamise funktsioonist $\langle \cdot \rangle_{2^B}$ ja lepitusfunktsioonist rec. Ümardamise funktsioon jagab hulga $\mathbb{Z}_q$ 2^B täisarvude vahemikeks, millel on sama *B* kõrgeim bitt ja tagastab ühe maatriksi elemendi kohta *B* bitti. Rist-ümardamise funktsioon jagab $\mathbb{Z}_q$ kaheks alamhulgaks vastavalt nende (*B* + 1) kõrgeimale bitile ja tagastab ühe maatriksi elemendi kohta ühe biti, sõltuvalt sellest, millises intervallis element asub. Funktsioon rec, kasutades sisendiks maatriksite **C** ning **BS′** elemente, aitab taastada $\lfloor \mathbf{V} \rceil_{2^B} = \mathrm{rec}(\mathbf{B'S}, \mathbf{C})$ ehk lõpptulemusena arvutada maatriksist kindla suurusega sümmeetrilise võtme.

Protokolli spetsiifilisi parameetreid $\bar{n}$, $\bar{m}$ ja *B* kasutatakse selleks, et arvutada salajane maatriks $\mathbf{E''} \in \mathbb{Z}_q^{\bar{m} \times \bar{n}}$ ja lepitusmehanismis, selleks et välja võtta *B* bitti igast maatriksi elemendist, sümmeetrilise võtit *K* moodustamiseks. $\bar{n}$ ja $\bar{m}$ mõõdud peavad olema valitud sellisel viisil, et võtme *K* bittide arv oleks piisav sihitud turvaastme saavutamiseks. Näiteks kui on sihitud 128-bitine post-kvant turvaaste, siis $\bar{n} \cdot \bar{m} \cdot B \geq 256$. Seega, mida suurem *B* valitakse (bittide arv, mis võetakse välja ühest elemendist), seda väiksema $\bar{n}$ ja $\bar{m}$ võib valida, mis tähendab väiksema suurusega maatrikseid ja väiksemaid sidekulusid.

3.2.3 Parameetrite valik

Võtmekehtestusprotokoll koosneb järgmistest LWE parameetritest:

- Moodul q
- Maatriksi järk n
- Juhuslik jaotus, selleks et valida **S** ja **E**

Lisaks on olemas ka protokolli spetsiifilised parameetrid B, $\bar{m}, \bar{n} \in \mathbb{Z}$.

Oli pakutud mitu parameetrite valikut: „väljakutse parameetrid“; „klassikalised parameetrid“, mis tagavad 128-bitise turvalisuse klassikaliste rünnete vastu, kuid neid ei ole soovitav kasutada, sest need ei taga piisavat turvalisust kvantrünnete vastu; „soovitatud parameetrid“, mis tagavad liigikaudu 128-bitise turvalisuse kvantrünnete vastu ja nn „paranoilised parameetrid“, mis tagavad kõrgeima post-kvant turvalisuse.

Järgnevalt on esitatud „soovitatud parameetrid“, sest just nendega testiti protokolli teostust ja seega saab protokolli teostust teiste protokollidega võrrelda.

Maatriksi järk n = 752, moodul $q = 2^{15}$. Salajaste komponendite ja müramaatriksite genereerimisel kasutatakse pöördavalimi meetodit. Pakutud parameetritega osapooled genereerivad 256–bitise võtme ja läbikukkumise tõenäosus on $2^{-38.9}$. See on suurem kui New Hope protokollis ja sõltub sellest, et võti genereerimiseks saadakse ühest kordajast rohkem kui üks jagatud bitt, ja samas on piisavalt väike, et protokolli saaks kasutada praktikas. Kordaja kohta kokkulepitud bittide arv B on 4 ja kordajate arv $\bar{m} \cdot \bar{n} = 8^2$.

3.2.4 Teostusaspektid

Protokolli teostus on C keeles ja pakub kaitset lihtsate ajastusrünnete ja *cache-timing* rünnete vastu. Lisaks on teostus integreeritud OpenSSL-i, võimaldades selle kasutamist Apache abil toimivuse testimiseks reaalsetes seadetes.

Protokolli teostuses kasutatakse tavalist maatriksite korrutamist, sest maatriksite järgud ei ole piisavalt suured kasutamaks kiiremat korrutamismeetodit. Artiklis [9] näidati, et selline korrutamine on kaitsetu võimsustarbe diferentsiaalanalüüsi vastu, mille abil ründaja võib 99% tõenäosusega taastuda salajase võtme. Antud ründes sihitud tehteks on maatriksi **A** korrutamine salajase maatriksiga **S** või **S′** ja selleks, et kaitsta võtmekehtestusprotokolli, pakutakse arvutustele juhuslikkuse lisamist. See on saavutatav arvutuste järjekorra randomiseerimisega, sest maatriksite korrutamise tulemus ei sõltu osakorrutiste leidmise järjekorrast. Lisaks võib paralleelne andmetöötlus vähendada selle ründe tõhusust, sest arvutustes tekkib rohkem algoritmilist müra [9].

Maatriks **A** genereeritakse iga uue võtmekehtesuse käituse jaoks uuesti seemnest pseudojuhuarvude generaatori abil ning teisele osapoolele saadakse ainult seeme, mitte terve maatriks. See lähenemine aitab vältida tagauksi ja *„all-for-the-price-of-one“* ettearvutusega rünnet, mis aja võitmiseks sisaldab mahukate arvutuste sooritamist enne ründevõimaluse tekkimist, ning lisaks aitab vähendada vajalikku läbilaskevõimet. Selleks, et lihtsustada maatriksi genereerimist, eriti mälupiirangutega seadmete jaoks, pakuti maatriksi osade moodustamist, kasutamist ja kustutamist *on-the-fly* ehk automaatselt. Maatriksi tuletamine toimub komponentide kaupa, rakendades AES128-ECB (*Advanced Encryption Standard with Electronic Code Book Mode*) eeltäidetud maatriksiridadele garanteeritud unikaalsusega ja edasi saab maatriksit arvutada kas ridade - üks rida korraga või veergude - 8 veeru korraga kaupa, sõltuvalt sellest, kas **A** korrutatakse **S**-ga paremalt (kliendipoolne) või vasakult (serveripoolne).

Selleks, et integreerida pakutud võtmekehtestusprotokolli TLS 1.2 protokolli, mängib Aliisa rolli server ja Bob rolli klient. Andmeid, mida Aliisa saadab Bobile, saadakse *ServerKeyExchange* sõnumiga ja Bobi vastus saadakse *ClientKeyExchange* sõnumiga. Aliisa ja Bob kasutavad võtit *K* nagu eelsaladust (ing. *premaster secret*), mis on vajalik põhisaladuse (ing. *master secret*) tuletamiseks, nagu kirjeldatud TLS 1.2 spetsifikatsioonis [11].

3.3 Crystals-Kyber KEM

Crystals-Kyber on võtmekapseldusmehhanism, mis põhineb Module-LWE probleemil ja pakuti välja Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanckk, Peter Schwabe, Gregor Seiler, Damien Stehlé poolt. Crystals-Kyber võtmekapseldusmehhanism oli saadud Fujisaki-Okamoto transformatsiooni KEM variandi rakendamisel Kyber CPA-turvalise (*secure against Chosen-Plaintext Attack*) krüpteerimisskeemi peale. See KEM on CCA2-turvaline (*secure against adaptive Chosen-Ciphertext Attacks*). Alljärgnev protokolli kirjeldus põhineb teadustöödel [12]-[14].

Tähistus. *R* on ring $\mathbb{Z}[X]/(X^n + 1)$ ja R_q on ring $\mathbb{Z}_q[X]/(X^n + 1)$, kus $n = 2^{n'-1}$, nii et $X^n + 1$ on $2^{n'}$-ndat järku ringpolünoom. Vektori **v** (või maatriksi **A**) jaoks, $\mathbf{v}^{\mathrm{T}}$ (või $\mathbf{A}^{\mathrm{T}}$) tähistab selle transponeerimist. $y \sim S := \mathrm{Sam}(x)$ tähendab, et funktsioon Sam võtab sisendiks *x* ja tagastab väärtuse *y*, mis on jaotatud vastavalt jaotusele *S*. Regulaarsed tähed tähistavad *R* või R_q elemente, rasvased väiketähed on vektorid kordajatega *R*-s või R_q-s, rasvased suurtähed on maatriksid.

3.3.1 Module-LWE probleem

Olgu *n, k* ja *q* positiivsed täisarvud. Olgu χ_s ja χ_e tõenäosusjaotused üle R_q. Olgu $\mathbf{s} \xleftarrow{\$} \chi_n^k$, $\mathrm{e}_i \xleftarrow{\$} \chi_e$, $\mathbf{a}_i \xleftarrow{\$} U(R_q^{\mathrm{k}})$ ja $\mathrm{b}_i \leftarrow \mathbf{a}_i^{\boldsymbol{T}}\mathbf{s} + \mathrm{e}_i$.

Definitsioon 3.3.1 [12] Module-LWE otsinguprobleem on (n, k, q, χ_s, χ_e) jaoks leida **s**, kui on antud ($\mathbf{a}_i$,b_i).

Defineerime kaks oraaklit:

- $O_{\chi_{e,s}}$: $\mathbf{a_i} \xleftarrow{\$} U(R_q^k)$, $e_i \xleftarrow{\$} \chi_e$; tagasta $(\mathbf{a_i}, \mathbf{a_i^T s} + e_i)$
- U: $\mathbf{a_i}, u_i \xleftarrow{\$} U(R_q^k \times R_q)$; tagasta $(\mathbf{a_i}, u_i)$

Definitsioon 3.3.2 [12] Module-LWE otsustusprobleem on $(n, k, q, \chi_s, \chi_e)$ jaoks eristada $O_{\chi_{e,s}}$ ja U.

3.3.2 Protokolli tööpõhimõtte

Algoritm KeyGen(): võtme genereerimine

1: $\rho, \sigma, z \leftarrow \{0,1\}^{256}$

2: $\mathbf{A} \sim R_q^{k \times k} := \mathrm{Sam}(\rho)$

3: $(\mathbf{s}, \mathbf{e}) \sim \beta_\eta^k \times \beta_\eta^k := \mathrm{Sam}(\sigma)$

4: $\mathbf{t} := \mathrm{Compress}_q(\mathbf{As} + \mathbf{e}, d_t)$

5: tagasta $(pk := (\mathbf{t}, \rho), sk := (\mathbf{s}, z, \mathbf{t}, \rho))$

Alguses valitakse juhuslikud seemned ρ, σ, z. Sam on laienduv väljundfunktsioon (ing. *Extendable-Output Function*, XOF), ehk räsifunktsioon mis võtab sisendiks seeme ja tulemuseks annab suvaliselt suure hulga juhuslikena näivaid bitte. Seemnest ρ genereeritakse maatriks $\mathbf{A}$ valides selle elemendid ühtlaselt ja juhuslikult ringist R_q. Selleks kasutatakse räsifunktsiooni SHAKE-128. Seemnest σ genereeritakse salajased vektorid $\mathbf{s}$ ja $\mathbf{e}$, valides nende kordajad tsentreeritud binoomjaotuse järgi, selleks kasutatakse räsifunktsiooni SHAKE-256. Funktsioon Compress võtab sisendiks väärtust $\mathbf{As} + \mathbf{e}$ ja iga selle kordaja kohta väljastab arvu, hulgast $\{0, \ldots, 2^{d_t} - 1\}$. Compress ja Decompress funktsioonide kasutamise põhjus on see, et need võimaldavad kõrvale jätta mõned avaliku võtme ja krüptogrammi madalaimad bitid, millel ei ole palju mõju õige dekrüpteerimise tõenäosusele, muutes seeläbi parameetrid väiksemateks. Compress funktsioon mängib dekrüpteerimisalgoritmis ka lepitusfunktsiooni rolli, väljastades 0 või 1 sõltuvalt sellest, millisesse intervalli kuulub sisendiks antud väärtus. Võtme genereerimise algoritm väljastab tulemuseks avaliku võtme $pk = (\mathbf{t}, \rho)$ ja salajase võtme $sk = (\mathbf{s}, z, \mathbf{t}, \rho)$**.** Avaliku võtme *pk* saadab Aliisa Bobile.

Algoritm Encaps($pk = (\mathbf{t}, \rho)$): võtme kapseldamine

1: $m \leftarrow \{0,1\}^{256}$

2: $(\widehat{K}, r) := G(H(pk), m)$

3: $(\mathbf{u}, v) :=$ Kyber. CPA. Enc$((\mathbf{t}, \rho), m; r)$

4: $\mathbf{c} := (\mathbf{u}, v)$

5: $K := H\left(\widehat{\mathrm{K}}, H(c)\right)$

6: tagasta (c, K)

Alguses valitakse juhuslik bitijada *m*. Avaliku võtme *pk*, mille Bob sai Aliisalt, räsitakse, et saada „eelvõti“ $\widehat{K}$, rakendades räsifunktsiooni G: $\{0,1\}^* \rightarrow \{0,1\}^{2\times 256}$, mis on antud juhul SHA3-512 räsifunktsioon, väärusele $H(pk)$, kus H on ka räsifunktsioon H: $\{0,1\}^* \rightarrow \{0,1\}^{256}$, täpsemalt SHA3-256 räsifunktsioon. Eelvõtme leidmise vajadus on tingitud kahest põhjusest. Esiteks, see aitab parandada olukorda, et sümmeetriline võti sõltub ainult ühe osapoole sisendandmetest. Teine põhjus on kaitse ettearvutusega ründe eest, ehk see aitab suurendada protokolli turvalisust. Juhuslik bitijada *m* krüpteeritakse, kasutades avalikku võtit $pk = (\mathbf{t}, \rho)$ ja juhuslikku bitijada *r*, mis oli saadud *m*'ile räsifunktsiooni rakendades. Krüpteeritud väärtust ja „eelvõtit“ kasutades arvutatakse sümmeetriline võti *K*, räsifunktsiooni *H* abil. Algoritm väljastab tulemuseks krüptogrammi *c* ja sümmeetrilise võtme *K*. Krüptogrammi *c* saadab Bob Aliisale.

Algoritm Decaps($sk = (\mathbf{s}, z, \mathbf{t}, \rho), c = (\mathbf{u}, v)$): võtme lahtikapseldamine

1: $m' :=$ Kyber. CPA. Dec(s, $(\mathbf{u}, v)$)

2: $(\widehat{K}', r') := G(H(pk), m')$

3: $(\mathbf{u}', v') :=$ Kyber. CPA. Enc$((\mathbf{t}, \rho), m'; r')$

4: **if** $(\mathbf{u}', v') = (\mathbf{u}, v)$ **then** tagasta $K := H\left(\widehat{\mathrm{K}}', H(c)\right)$

5: **else** tagasta $K := H(z, H(c))$

Esiteks, väärtus $c = (\mathbf{u}, v)$ dekrüpteeritakse, kasutades salajast vektorit **s**, selleks, et teada saada bitijada m' väärtust. Seejärel räsitakse väärtus m' ja avalik võti, et leida „eelvõti“ $\widehat{K}'$ ja juhuslik bitijada r'. Kasutades saadud väärtusi ja krüpteerimisalgoritmi arvutatakse teine versioon krüptogrammist $(\mathbf{u}', v')$ ning võrreldatakse sellega, mis on Bobi käest saanud. Kui väärtused langevad kokku, siis arvutatakse sümmeetriline võti $K = H\left(\widehat{\mathrm{K}}', H(c)\right)$. Kui väärtused ei lange kokku siis ei väljasta algoritm tagasilükkamist $\perp$, vaid tagastab pseudojuhusliku võtme $K = H(z, H(c))$, mille sisendid on Bobilt saadud krüptogramm ja Aliisa poolt alguses leitud juhuslik seeme *z*.

3.3.3 Parameetrite valik

Võtmekapseldusmehhanism koosneb järgmistest Module-LWE parameetritest:

- Moodul q
- R_q aste n
- Mooduli järk k
- Juhuslik jaotus, selleks et valida **s** ja e

Lisaks on olemas ka protokolli spetsiifilised parameetrid η, mis on binoomjaotuse parameeter ning kolmik (d_u, d_v, d_t), mille väärtuseid kasutatakse Compress ja Decompress funktsioonides, mis esinevad võtme genereerimise, krüpteerimise ja dekrüpteerimise algoritmides.

Selleks, et tagada vähemalt 128-bitist turvalisust kvantrünnete vastu suure turvavaruga, et võtta arvesse krüptoanalüüsi edasist arengut, on pakutud järgmised parameetrid: R_q aste n = 256, mis on valitud selleks et kodeerida iga bitti ühe kordajasse. Mooduli q = 7681 valik on põhjustatud sellest, et 7681 on väikseim algarv, mille jaoks kehtib $q \equiv 1 \, mod \, 2n$. See tähendab, et korrutamise lihtsustamiseks ja kiirendamiseks võib kasutada NTT. Fikseeritud parameeter k = 3 reguleerib võre mõõtmeid ja seeläbi aitab reguleerida protokolli turvaastet, mille suurendamine või vähendamine võib olla realiseeritud parameetri *k* suuruse vahetamisega.

Juhuslikeks jaotuseks on valitud tsentreeritud binoomjaotus B_η parameetriga η = 4, jaotuse dispersioon on η /2 ja η = 4 jaoks standardhälve on $\varsigma = \sqrt{4/2}$. Parameetrid

$\eta = 4$, $d_u = 11$, $d_v = 3$ ja $d_t = 11$ olid valitud sellisel moel, et tasakaalustada protokolli turvalisust, läbikukkumise tõenäosust, avaliku võtme suurust ja krpüptogrammi suurust.

Läbikukkumise tõenäosus antud võtmekapseldusmehhanismi jaoks on 2^{-142}.

3.3.4 Teostusaspektid

Protokolli teostati kahel erineval viisil. Esimene teostus on C keeles, mille eelis on lihtsus ja porditavus. Teine on AVX2 optimeeritud teostus Intel Haswell protsessoritele. Mõlemad teostused pakuvad täieliku kaitset ajastusrünnete vastu ja on testitud kasutades Intel Core i7-4770K (Haswell) protsessorit.

Kyber protokollis on polünoomide korrutamise optimeerimiseks pakutud NTT rakendamist, selleks, et teisendada polünoomid piirkonda, kus korrutamine on palju kiirem ja odavam. NTT rakendamise eelised on suur kiirus ja täiendava mälunõude puudumine (nagu näiteks Karatsuba või Toom korrutamise puhul).

Maatriksi **A** genereerimiseks valiti NewHope'i lähenemisviis. See tähendab, et maatriks **A** ei ole süsteemi parameeter, vaid selle asemel genereeritakse maatriks **A** iga kord uuesti. Selle lähenemisviisi eelis on see, et kõigepealt välditakse arutelusid, kuidas genereerida ühtlaselt juhuslik süsteemi parameeter. Teiseks, see kaitseb *„all-for-the-price-of-one“* ettearvutusega rünnete eest. Selle otsuse maksumus on maatriksi **A** laiendamine juhuslikust seemnest võtme genereerimise ja kapseldamise jooksul.

Selle protokolli jaoks lubati lahtikapseldamise läbikukkumist ebaolulise (ing. *negligible*) tõenäosusega. Õigesti valitud parameetrite korral oli võimalik täielikult kõrvaldada läbikukkumise tõenäosus, kuid selle lähenemise puudused on kas turvalisuse vähenemine nende rünnete eest, mis sihivad protokolli aluseks olevat võreprobleemi, müra vähendamise kaudu või jõudluse vähenemine, kompenseerides turvalisuse kadu võre mõõtmete suurendamisega.

Sõnumite formaadi osas on vaja vaadelda kahte suurust, mis protokolli töötamise käigus edastatakse, need on $pk = (\mathbf{t}, \rho)$ ja $c = (\mathbf{u}, v)$ sõnumid. Mõlemal juhtumil edastatakse polünoome tavalises piirkonnas, mitte NTT piirkonnas nagu oli tehtud NewHope protokolli puhul. **t** on vektor, mis koosneb kolmest polünoomist, igas polünoomis on 256 11-bitist kordajat ja ρ on 32 baidine seeme. Polünoomid on esitatud pakitud

pöördjärjestusega formaadis, mis annab iga polünoomi kohta (256*11)/8 = 352 baiti. Sõnumi $pk = (\mathbf{t}, \rho)$ pikkus on 3*352 + 32 = 1088 baiti. Krüptogramm koosneb vektorist **u**, mis on sama suurusega, nagu vektor **t** ja polünoomist v, mis koosneb 256 3-bitist kordajast. Seega on krüptogrammi suurus 3*352 + (3*256)/8 = 1152 baiti.

3.4 NTRU-HRSS KEM

NTRU-HRSS KEM on võtmekapseldusmehhanism, mis põhineb NTRU probleemil ja mille pakkusid välja Andreas Hülsing, Joost Rijneveld, John Schanck ja Peter Schwabe. NTRU võtmekapseldusmehhanism oli saadud Fujisaki-Okamoto teisenduse KEM variandi rakendamisel NTRU CPA-turvalisele krüpteerimisskeemile. Selle võtmekapseldusmehhanismi karakteristikud näitavad, et NTRU krüptosüsteem, kõige vanem võrepõhine krüptosüsteem, mis on üle elanud 20 aastat krüptoanalüüsi, on võimeline konkureerima uuemate võrepõhiste skeemidega kiiruse, võtme suuruse ja krüptogrammi suuruse poolest. Üheks põhjuseks, miks NTRU ei ole veel laialdasemalt kasutusel, on kasutust piiravad patendid. Patentide kestvus lõppes märtsil 2017, võimaldades NTRU avaliku kasutamist. Lisaks, NTRU-HRSS KEM on turvaline valitava krüptogrammiga adaptiivründe vastu ehk see on CCA2-turvaline. Alljärgnev protokolli kirjeldus põhineb teadustöödel [15]-[18].

Tähistus. $(\mathbb{Z}/n)^{\times}$ on täisarvude multiplikatiivne rühm mooduli *n* järgi. Φ_n on polünoom $(x^n - 1)/(x - 1) = x^{n-1} + x^{n-2} + \cdots + 1$. R on faktorring $\mathbb{Z}[x]/(x^n - 1)$. S on faktorring $\mathbb{Z}[x]/(\Phi_n)$. Hulgad T ja T_+ on defineeritud järgmiselt: $T = \{v \in \{-1, 0, 1\}^n: v_{n-1} = 0\}$, $T_+ = \{v \in T : \langle x \cdot v, v \rangle \geq 0 \}$. $\lceil a \rceil$ on täisarvuks ümmardamise funktsioon (ing. *ceiling function*), mis väljastab vähima täisarvu, mis on suurem või võrdne *a*-ga, näiteks $\lceil 2.4 \rceil = 3$.

3.4.1 NTRU probleem

Pööratava $s \in R_q^*$ ja jaotuse χ R üle jaoks, defineerime $N_{s,\chi}$ olla jaotus, mis väljastab $e/s \in R_q$, kus $e \xleftarrow{\$} \chi$.

Definitsioon 3.4.1 [6] NTRU õppeprobleem (ing. *learning problem*): Antud näidiselemente $a_i \in R_q$ jaoks, kus iga näidiselement on jaotatud

1) kas $N_{s,\chi}$ järgi, juhuslikult valitud $s \in R_q$ jaoks
2) või ühtlase jaotuse järgi

eristada, millise juhtumiga (kas esimese või teisega) on tegu.

3.4.2 Protokolli tööpõhimõtte

Algoritm KeyGen(*coins*): võtme genereerimine

1: $g = \text{Sample}T_+(\text{XOF}(coins, \mu, \text{randg}))$

2: $f = \text{Sample}T_+(\text{XOF}(coins, \mu, \text{randf}))$

3: Arvuta f_q, nii et $(f \cdot f_q)\ mod\ q \equiv 1$ ringis S

4: Arvuta f_p, nii et $(f \cdot f_p)\ mod\ p \equiv 1$ ringis S

5: $h = (p \cdot (x-1) \cdot g \cdot f_q)\ mod\ q$

6: tagasta $(pk = h, sk = (f, f_p))$

Sample algoritmi kasutades, valitakse parameetrid g ja f hulgast T_+. Sample algoritmi kasutatakse ühtlase jaotuse asemel, sest sellega hulgast $T_+ ja\ T$ valiku tegemine konstantse ajaga võib olla keeruline ja aeglane. Algoritmi sisendiks on väärtus, mis on arvutatud XOF(*X*, *L*, *S*) funktsiooniga, kus *X* on sisend bitijada, *L* on soovitud väljundi pikkus bittides ja *S* on määramispiirkonna eraldaja (ing. *domain separator*), mis on vajalik selleks, et kahe erineva $S_1 \neq S_2$ jaoks XOF funktsioon annaks sõltumatud *X* sisendist saadut väärtust. Antud protokolli jaoks pakutakse XOF funktsiooni teostada räsifunktsioon SHAKE-128 abil. Seejärel arvutatakse teatud tingimustele vastavad f_q ja f_p ning pärast arvutatakse avalik võti $h \in R$. Võtme genereerimise algoritm väljastab tulemuseks avatud võtme $pk = h$ ja salajase võtme $sk = (f, f_p)$. Avaliku võtme *pk* saadab Aliisa Bobile.

Algoritm Encaps($pk = h$) võtme kapseldamine

1: $c_0 \leftarrow \{0,1\}^{\mu}$

2: $m = \text{Sample}T(c_0)$

3: $c_1 = \text{XOF}(m, \mu, coins)$

4: $k = \text{XOF}(m, \mu, key)$

5: $e_1 = \text{Enc}(m, c_1, h)$

6: $e_2 = \text{XOF}(m, len(m), \text{qrom})$

7: tagasta $(c = (e_1, e_2), K = k)$

Esiteks valitakse juhuslik seeme c_0, mida kasutatakse juhusliku bitijada m genereerimiseks Sample algoritmi abil hulgast T. Räsifunktsiooni XOF kasutades, tuletatakse bitijada m alusel deterministiliselt juhuslik münt c_1. Sümmeetriline võti k tuletatakse m-st, rakendades räsifunktsiooni teise määramispiirkonna eraldaja parameetriga. Bitijada m krüpteeritakse juhusliku mündi c_1 ja avaliku võtme h abil. Teine osa krüptogrammist e_2 saadakse m-le räsifunktsiooni rakendades, parameeter *len(m)* arvutatakse valemi $8 \cdot \lceil (n-1)/5 \rceil$ järgi. Lõpuks, väljastab algoritm krüptogrammi c ja sümmeetrilise võtme K, krüptogrammi c saadab Bob Aliisale.

Algoritm Decaps($sk = h, pk = f, c = (e_1, e_2)$)*:* võtme lahtikapseldamine

1: $m = \text{Dec}(c, f)$

2: $c_1 = \text{XOF}(m, \mu, \text{coins})$

3: $k = \text{XOF}(m, \mu, key)$

4: $e_1' = \text{Enc}(m, c_1, h)$

5: $e_2' = \text{XOF}(m, len(m), \text{qrom})$

6: **if** $(e_1', e_2') \neq (e_1, e_2)$ **then** tagasta $k = \perp$

7: **else** tagasta $K = k$

Lahtikapseldamisalgoritm dekrüpteerib krüptogrammi *c* salajase võtme *f* abil, et saada bitijada *m*. Pärast, tuletatakse räsifunktsiooni abil *m*-st juhuslik münt c_1 ja sümmeetriline võti *k*. Seejärel krüpteeritakse *m*, kasutades juhuslikku münti c_1 ja avalikku võtit *h* ja arvutatakse e_2' rakendades räsifunktsiooni *m*-le. Kui arvutuste tulemusena saadi krüptogramm (e_1', e_2'), mis vastab vastuvõetud krüptogrammile (e_1, e_2), siis algoritm väljastab sümmeetrilise võtme $K = k$. Kui krüptogrammid ei lange kokku, siis väljastab algoritm sümboli $\perp$, mis näitab tagasilükkamist.

3.4.3 Parameetrite valik

Võtmekapseldusmehhanism koosneb järgmistest NTRU parameetritest:

- Paaritu alarv *n*, mis indekseerib R_n ja S_n
- Ühistegurita positiivsed täisarvud *p* ja *q*

NTRU-HRSS parameetreid olid hoolikalt valitud ja optimeeritud, et saavutada liigikaudu 128-bitist post-kvant turvalisust, täielikult kõrvaldades samal ajal dekrüpteerimise ebaõnnestumise tõenäosuse. Protokolli autorid tõestasid, et p = 3 ja q = $q(n)$, kus $\log_2 q(n) = \lceil 7/2 + \log_2(n) \rceil$, puhul dekrüpteerimise ebaõnnestumine ei ole võimalik, seega p = 3 ja q, mis on vähim 2 aste, mis vastab tingimustele, *n* on ainus vaba parameeter. Viimane parameetrite piirang on see, et polünoom Φ_n peab olema taandumatu nii mooduli *p*, kui ka mooduli *q* järgi. Pakutud parameetrid: *n* = 701, *p* = 3, *q* = 8192.

Protokolli spetsiifiliseks parameetriks on $\mu = 256$, mis määrab ära XOF räsifunktsiooni soovitud väljundi pikkuse bittides.

3.4.4 Teostusaspektid

NTRU-HRSS põhise KEM toimimise näitamiseks esitati hästi optimeeritud teostus Intel protsessoritele koos AVX2 vektori laiendusega. Samale arhitektuurile, millele oli suunatud optimeeritud NewHope'i võtmekehtestus. Saadud tarkvara on esimene NTRU tarkvara, millel on täielik kaitse ajastusrünnete eest.

Selle protokolli peamised arvutusfunktsioonid on korrutamine, mida kasutatakse võtme genereerimise, krüpteerimise ja dekrüpteerimise algoritmides ja pööramine, mis mängib väga suurt rolli võtme genereerimise algoritmis. Korrutamise protseduur koosneb mitmest kombineeritud korrutusalgoritmist, mille seas on sellised algoritmid nagu Toom-

Cook ja Karatsuba. Pööramise protseduuri lihtsustamiseks ja optimeerimiseks on pakutud mitmeid teisendusi, et pööramise asemel leida teatud polünoomide korrutised ja ruudud.

Sõnumite formaadi osas on vaja vaadelda kahte suurust, mida protokolli töötamise käigus edastatakse, need on $pk = h$ ja $c = (e_1, e_2)$ sõnumid. Avaliku võtme h kodeerimiseks kasutatakse algoritmi Rq_to_bits, mis võtab sisendiks polünoomi h, mis vastab tingimustele $h(1) \equiv 0 \; mod \; q$, ja tagastab bitijada pikkusega (701-1)*13 = 9100 bitti = 1138 baiti. Krüptogramm koosneb kahest kontateneeritud väärtusest $e_1 \; ja \; e_2$. e_1 kodeeritakse nagu avalik võti, kasutades Rq_to_bits, seega selle pikkus on 9100 bitti, e_2 kodeeritakse kasutades teist algoritmi S3_to_bits, mille sisend on polünoom ja väljund on bitijada pikkusega $8 \cdot \lceil (701 - 1)/5 \rceil = 1120$ bitti. Kogu krüptogrammi suurus on seega 9100 + 1120 = 10220 bitti = 1278 baiti. Kodeerimisalgoritmid olid kirjeldatud bittide tasemel ja oktettideks konverteerimiseks on bitijada täidetud nullidega, seni kuni selle pikkus on $8 \cdot l$, mingi l korral. Seejärel kodeerimine on indeksite järjekorra säilitamisega, kus kaugus on vähemalt 8, kuid järjekorra muutmisega oktettides. Näiteks $b_1, ..., b_7, b_8, b_9, ..., b_{15}, b_{16}$ on kodeeritud kui $b_8 b_7 ... b_1, b_{16} b_{15} ... b_9$.

3.4.5 Protokolli katsetamine

Pärast punktis 3.1.5 kirjeldatud katse tegemist toimus kaks olulits muutust: esiteks, pakuti uued post-kvant võtmekehtestusprotokollid, mis esitati NIST projektile ja teiseks, mindi üle TLS 1.2-lt TLS 1.3 protokollidele. Viimane muutus mõjutas TLS'i kätluses ilmuvad suuremaid sõnumeid. TLS 1.2-s algvoolus teatab kasutaja, milliseid algoritme ta oskab kasutada, server valib ühe ja saadab oma vastuses avaliku võtme, seejärel lõpetab klient võtmekehtestuse oma teises voos. TLS 1.3 puhul pakub klient oma algvoolus mitmeid võimalikke avalikke võtmeid ja server lõpetab võtmekehtestuse ühega neist oma vastuses. Seega saadetakse TLS 1.3 kasutades suuremad post-kvant võtmed igale TLS-serverile, olenemata sellest, kas need seda kasutavad või mitte [25].

Selleks, et hinnata milline ajaline viivitus võib tekkida post-kvant võtmekehtestuse TLS 1.3-s kasutamisega, tegid Google'i teadlased katse, et simuleerida olukorda, kus klient ja server saadavad suuremaid post-kvant sõnumeid, Chrome-l võimaldati lisada fiktiivne, kindla suurusega laiendus TLS sõnumisse ehk see oli lihtsalt kindla suurusega mürabaitide arv, mida kasutati, et hinnata, millist mõju erineva suurusega laiendused avaldavad läbilaskevõimele [25]. Laiendusi pakuti 4 erinevat tüüpi, sõltuvalt

võtmekehtestusprotokollide perekondadest ja sellest, millise suurusega sõnumeid on vaja vahetada. Selle, millise suurusega laiendus sõnumisse lisada, valis Chrome juhuslikult [25].

- Kontrollrühm: laiendit pole saadetud
- Supersingulaarsed isogeensused: 400 baiti
- Struktureeritud võred: 1 100 baiti
- Struktureerimata võred: 3 300 baiti

Katse tulemused olid punktis 3.1.5 kirjeldatud katse tulemustest erinevad struktureeritud võrede korral (sellele perekonnale kuulub NewHope protokoll): mediaanühenduse viivitus suurenes 5.5 ms võrra (CECPQ1: 1 ms), aeglase 5% viivitus suurenes 136.9 ms võrra (CECPQ1: 20 ms), need erinevused on tingitud erinevustega TLS 1.2 ja TLS 1.3 struktuuris. Supersingulaarsete isogeensuste tulemused olid järgmised: mediaan ühenduse viivitus suurenes 2.6 ms võrra, aeglase 5% viivitus suurenes 19.2 ms võrra, kuid supersingulaarsete isogeensuste arvutused on aeglasemad kui struktureeritud võrede korral. Struktureerimata võrede tulemused olid liiga suured selleks, et neid saaks kasutada reaalaja TLS ühendustes. Supersingulaarsete isogeensuste ja struktureeritud võrede omaduste võrdlemise katse järeldus oli see, et post-kvant turvalisus TLS-is peaks tõenäoliselt põhinema protokollidel, mille baasiks on struktureeritud võred [25].

Lähitulevikus planeerib Google alustada uue katsega, mis hindaks struktureeritud võrede kasutamist TLS 1.3 korral [26]. Katse nimi on CECPQ2 ja see kombineerib eliptkõvera Diffie ja Hellmani protokolli koos NTRU-HRSS võtmekapseldusmehhanismi versiooniga, mis modifitseeriti Saito, Xagawa, ja Yamakawa poolt [19]. CECPQ2 on vajalik mitmel põhjustel: esiteks, NSA (*National Security Agency*) plaanides on avaldada post-kvant standardid 2024. aastaks NIST projekti tulemuste baasil ja kui arvestada Michele Mosca hinnangut, et 1/7 tõenäosusega leitakse piisavalt suur kvantarvuti, mis saab murda RSA-2048 on saadaval 2026. aastaks selgub, et on ohtlik oodata 5 aastat kuni standartide avaldamiseni [3]. Teiseks on kogemus TLS 1.3 kasutusele võtmisega, ehk seda pidanuks olema lihtne paigaldada, kuid praktikas osutus paigaldus aeglaseks protsessiks, kuna oli vaja parandada teostuses leitvaid vigu [26]. Kuna Internet muutub iga aastaga üha keerulisemaks on ohtlik oodata standardite avalikustamist selleks, et hakata tegelema post-kvant protokollide rakendamisega. Juba täna on vaja hakata vaikselt

liikuma post-kvant võtmekehtestuse rakendamise poole, et õigeaegselt avastada tekkivaid vigu ja parandada neid, et ka tulevikus garanteerida üle võrgu suhtlemise turvalisust [26].

3.5 Three Bears

Three Bears on võtmekapseldusmehhanism, mis põhineb täisarvu Module-LWE probleemil ja pakuti välja Mike Hamburg poolt. Three Bears on turvaline valitava krüptogrammiga rünne vastu ehk see on CCA-turvaline (*secure against Chosen-Ciphertext Attack*). Alljärgnev protokolli kirjeldus põhineb teadustöödel [20], [21].

Tähistus. Olgu $\mathbb{Z}$ on täisarvud ja $\mathbb{Z}/N\mathbb{Z}$ on täisarvude ring mooduli *N* järgi. Tähistagu T^n kõigi *T*-tüüpi elementide *n*-jadade hulka, neid tähistatakse $[\![a, b, \cdots, z]\!]$ või $[\![S_i]\!]_{i=0}^{n-1}$. $\lfloor a \rceil$ on täisosa funktsioon (ing. *floor function)*, mis väljastab suurima täisarvu, mis on väiksem või võrdne *a*-ga. Lähimat täisarvu reaalarvule *a* tähistatakse $\lfloor a \rceil = \lfloor a + 1/2 \rfloor$. Maatriksid on tähistatud rasvaste suurtähtedega ja vektorid on tähistatud rasvaste väiketähtedega. Kõik arvutused protokollis on tehtud mooduli *N* järgi, mis on suur Mersenne'i algarv.

3.5.1 Protokolli tööpõhimõtte

Algoritm KeyGen(): võtme genereerimine

1: $sk \leftarrow \text{randomBytes}(\text{privateKeyBytes})$

2: **for** *i = 0* **to** *d* - 1 **do** $\mathbf{a_i} \leftarrow \text{noise}_1(sk, i)$

3: $\text{martixSeed} \leftarrow H_1(sk, \text{matrixSeedLen})$

4: **for** *i*, *j* = 0 **to** *d* - 1 **do** $\mathbf{M_{i,j}} \leftarrow U(\text{matrixSeed}, i, j)$

5: **for** *i* = 0 **to** *d* - 1 **do** $A_i \leftarrow \text{noise}_1(sk, d + i) + \sum_{j=0}^{d-1} \mathbf{M_{i,j}} \cdot \mathbf{a_j} \cdot \text{clar}$

6: tagasta $(pk = (\text{matrixSeed}, [\![A_i]\!]_{i=0}^{d-1})$

Esiteks, algoritmis genereeritakse salajane võti, mis on ühtlaselt ja juhuslikult valitud baitide jada. Salajane võti on noise algoritmi sisend, et genereerida salajane vektor **a** mooduli *N* järgi. Algoritm toimib sisendi laiendamisega ühele baidile numbri kohta ja

seejärel toimub numbri teisendamine täisarvuks õige variatsiooniga. Räsifunktsiooni H_1 abil salajasest võtmest genereeritakse seeme maatriksi **M** loomiseks. Saadud seemnest genereeritakse ühtlaselt juhuslik $d \times d$ maatriks **M**, valides iga elemendi eraldi, ja seejärel arvutatakse $A = \mathbf{Ma} + \boldsymbol{\epsilon}_\mathbf{a}$, kus $\boldsymbol{\epsilon}_\mathbf{a}$ on müravektor, mille saadab Aliisa koos maatriksi seemnega Bobile.

Protokollis kasutatud räsifunktsioon *H* on saadud cSHAKE(*X*, *L*, *N*, *S*) räsifunktsioonist, kus *X* on on bitijada ehk funktsiooni peamine sisend, *L* on täisarv, mis näitab taotletud väljundi pikkust bittides, *N* on funktsiooni nime string, mida NIST kasutab cSHAKE-l põhinevate funktsioonide määratlemiseks. Kui ühtegi muud funktsiooni peale cSHAKE ei soovita, on *N* tühi string, *S* on kohandamisstring. Kasutaja muudab selle stringi funktsiooni variandi määramiseks [22]. Räsifunktsioon *H* on defineeritud järgnevalt: $H_p(data, L) := cSHAKE256(pblock \,||\, [\![0, p]\!] \,||\, data, 8 \cdot L, "", "ThreeBears")$, kus parameeter pblock on konkateneeritud protokolli versiooni (BabyBear, MamaBear, PapaBear) parameetrid. Kuna räsifunktsiooni kasutatakse protokollis erinevate eesmärkidega on lisatud ka 1-bitine parameeter *p* (ing. *purpose*), mis näitab räsifunktsiooni kasutamise eesmärki.

Algoritm Encaps($pk = (\text{matrixSeed}, [\![A_i]\!]_{i=0}^{d-1})$ võtme kapseldamine

---

1: $\text{seed} \leftarrow \text{randomBytes}(\text{encSeedBytes})$

2: **for** *i* = 0 **to** *d* - 1 **do** $\mathbf{b}_i \leftarrow \text{noise}_2(\text{matrixSeed}||\text{seed}, i)$

3: **for** *i*, *j* = 0 **to** *d* - 1 **do** $\mathbf{M}_{\mathbf{i,j}} \leftarrow U(\text{matrixSeed}, i, j)$

4: **for** *i* = 0 **to** *d* - 1 **do** $\text{B}_\text{i} \leftarrow \text{noise}_2(\text{seed}, d + i) + \sum_{j=0}^{d-1} \mathbf{M}_{i,j} \cdot \mathbf{b}_j \cdot \text{clar}$

5: $C_\text{b} \leftarrow \text{noise}_2(\text{seed}, 2 \cdot d) + \sum_{j=0}^{d-1} A_j \cdot \mathbf{b}_j \cdot \text{clar}$

6: $pt \leftarrow \text{seed}$

7: $\text{encpt} \leftarrow \text{FecEncode}(pt)$

8: **for** *i* = 0 **to** len(encpt) - 1 **do** $encr_i \leftarrow \text{extract}_4(\text{C}_b, i) + 8 \cdot \text{encoded\_seed}_i \bmod 16$

9: $\text{shared\_secret} \leftarrow H_2(\text{matrixSeed}||pt, \text{sharedSecretBytes})$

10: $\text{capsule} \leftarrow ([\![B_i]\!]_{i=0}^{d-1}, \text{nibbles}\ [\![encr_i]\!]_{i=0}^{len(pt)-1})$

11: tagasta $(\text{shared\_secret}, \text{capsule})$

Kapseldamise algoritm alustab seemne genereerimisega, mille pikus on määratud encSeedBytes suurusega. Saadud seeme koos Aliisalt saadud maatriksi seemnega on sisend noise algoritmisse, et genereerida salajane vektor **b** mooduli *N* järgi. Maatriksi seemnest genereeritakse maatriksi **M**. Maatriksi **M** ja salajast vektorit **b** kasutades arvutatakse $B = \mathbf{b}^{\mathrm{T}}\mathbf{M} + \boldsymbol{\epsilon}_{\mathbf{b}}{}^{\mathrm{T}}$, kus $\boldsymbol{\epsilon}_{\mathbf{a}}$ on müravektor, seejärel saab arvutada Bobi ligikaudse jagatud saladuse $\mathrm{C_b} = \mathbf{b}^{\mathrm{T}}\mathrm{A} + \boldsymbol{\epsilon}_{\mathbf{b}}{}'^{\mathrm{T}} = \mathbf{b}^{\mathrm{T}}(\mathbf{M}\mathbf{a} + \boldsymbol{\epsilon}_{\mathbf{a}}) + \boldsymbol{\epsilon}_{\mathbf{b}}{}'^{\mathrm{T}} = \mathbf{b}^{\mathrm{T}}\mathbf{M}\mathbf{a} + \mathbf{b}^{\mathrm{T}}\boldsymbol{\epsilon}_{\mathrm{a}} + \boldsymbol{\epsilon}_{\mathbf{b}}{}'^{\mathrm{T}}$. Seemnest genereeritakse avatekst *pt* ja kodeeritakse see algoritmi FecEncode kasutades, mis on ennetava veaparanduse algoritm, mis lisab 18 bitti ja parandab kuni 2 viga. Kapsli koostamiseks arvutatakse väärtust *encr*. Selleks on vaja välja võtta teatud arv bitte ligikaudsest jagatud saladusest C_b ja liita sellele kodeeritud seeme. Bittide väljavõtmiseks kasutatakse funktsiooni $\mathrm{extract_b}(\mathrm{C}, i)$, mis tagastab b kõrgemat bitti kordajast, millel on *i*-vähim müra. shared_secret väärtuse saamiseks, kasutatakse räsifunktsiooni, mille sisendiks on maatriksi seeme ja avatekst. Kapsel sisaldab Bobi arvutatud väärtust *B* ja *encr*, mis on esitatud 4-bitiste näkside (ing. *nibble*) järjestusega.

Algoritm Decaps$(sk, capsule)$: võtme lahtikapseldamine

1: **for** *i = 0* **to** *d* - 1 **do** $\mathbf{a}_i \leftarrow \mathrm{noise}_1(sk, i)$

2: $C_{\mathrm{a}} \leftarrow \mathrm{noise}_2(\mathrm{seed}, 2 \cdot d) + \sum_{j=0}^{d-1} B_j \cdot \mathbf{a}_{\boldsymbol{j}} \cdot \mathrm{clar}$

3: **for** *i* = 0 **to** len(encr) **do** $\mathrm{encoded\_seed}_i \leftarrow \left\lfloor \frac{2 \cdot \mathrm{encr}_i - \mathrm{extract}_5(\mathrm{C_a}, i)}{2^l} \right\rceil$

4: $\mathrm{seed} \leftarrow \mathrm{FecDecode}(\mathrm{encoded\_seed})$

5: $(\mathrm{shared\_secret}, \mathrm{capsule}') \leftarrow \mathrm{Encaps}(pk, \mathrm{seed})$

6: **if** capsule′ ≠ capsule **then** tagasta ⊥

7: **else** tagasta shared_secret

Bobilt saadud väärtuse *B* abil arvutab Aliisa ligikaudse jagatud saladuse $C_{\mathrm{a}} = \mathrm{B}\mathbf{a} + \boldsymbol{\epsilon}_{\mathbf{a}}{}' = (\mathbf{b}^{\mathrm{T}}\mathbf{M} + \boldsymbol{\epsilon}_{\mathbf{b}}{}^{\mathrm{T}})\mathbf{a} + \boldsymbol{\epsilon}_{\mathbf{a}}{}' = \mathbf{b}^{\mathrm{T}}\mathbf{M}\mathbf{a} + \boldsymbol{\epsilon}_{\mathbf{b}}{}^{\mathrm{T}}\mathbf{a} + \boldsymbol{\epsilon}_{\mathbf{a}}{}'$, mis on ligikaudselt võrdne Bobi väärtusega C_b, lahutab C_a hinnangu Bobilt saadud *encr* väärtusest (mis koosneb C_b hinnangust ja kodeeritud seemnest) ja ümardab väikse vea, et saada kodeeritud seeme. Pärast seemne saamist, dekodeeritakse see algoritmi FecDecode kasutades. Selleks, et kontrollida, kas

Bobilt saadud kapsel oli aus, kapseldatakse arvutatud seeme uuesti. Kui arvutuste tulemusena saadakse sama kapsel, siis algoritm väljastab sümmeetrilise võtme shared_secret. Kui saadud väärtused on erinevad, väljastab algoritm tagasilükkamise sümboli $\perp$.

3.5.2 Parameetrite valik

On pakutud 3 erinevat parameetrite valikut: BabyBear pakub kõige madalamat turvaastet ja seda võib kasutada lihtseadmetes, MamaBear pakub keskmist turvaastet ja on esmane soovitus kasutamiseks, PapaBear pakub kõige suuremat turvaastet ja seda võib kasutada rakendustes, mis nõuvad väga suurt turvalisust.

Järgnevalt on esitatud soovitatud parameetrid (MamaBear), sest nendega saab paremini antud protokolli teostust võrrelda teiste protokollidega.

Mooduli järk $d = 3$, müra dispersioon, mida kasutatakse juhuslike vektorite genereerimise algoritmis, $\sigma^2 = 1/2$. Nende parameetritega protokolli läbikukkumise tõenäosus on 2^{-147} ja protokoll pakub 219-bitist turvalisust.

Protokollis on kasutusel ka teised spetsiifilised parameetrid, suurim osa nendest on seotud seemnete suurustega. ThreeBears seemnete suurused on mõeldud 2^{256} või suuremaks otsinguruumiks. Avaliku võtme suurus oli valitud selleks, et rohkem kaitsta võtme genereerimise algoritmi võtmetuletusrünnete eest.

- Avaliku võtme suurus $\text{privateKeyBytes} = 40$
- Sümmeetrilise võtme suurus $\text{sharedSecretBytes} = 32$
- Maatriksi seemne suurus $\text{matrixSeedLen} = 24$
- Krüpteerimise seeme suurus $\text{encSeedBytes} = 32$

Moodul N pidi olema algarv ja kuna ei olnud sobiva suurusega Fermat algarvu, oli valitud Mersenne'i algarv $2^p - 1$. Parimaks algarvude kujuks on $x^D - x^{D/2} - 1$, selleks et saavutada rahuldavat protokolli läbikukkumise tõenäosust ja transporteerida 256-bitist võtit, peavad kehtima tingimused $x \geq 2^{10}$ ja $D \geq 256$. Algarvuks valiti $2^{3120} - 2^{1560} - 1$, kus ringi dimensioon on $D = 312$.

Krüpteerimise ümardamise täpsus $l = 4$, seda väärtust kasutab lahtikapseldamisalgoritmis Aliisa, et arvutada seeme ja selgitaja clar = $x^{D/2} - 1 = 2^{1560} - 1$, selle kasutatakse väärtuste A, B ja C arvutamisel.

3.5.3 Teostusaspektid

Salajaseks võtmeks on otsustatud jätta ainult ühtlaselt juhuslik baitide jada. Võtmekehtestuse reaalelus rakendamisel võib vahemällu salvestada sellised vahepealsed väärtused, nagu salajane vektor **a** või maatriks **M**, kuid ThreeBears pakub piisavalt kiireid algoritme. Seega nende värtuste salvestamine on protokolli teostamise otsus ja sõltub konkreetse teostamise nõuetest ja omadustest.

On olemas kaks võimalust, kuidas tegeleda lahtikapseldamise läbikukkumisega kui ($capsule'$) $\neq$ ($capsule$). Otsene tagasilükkamine (ing. *explicit rejection*), mille tulemusena tagastatakse spetsiaalne tagasilükkamise sümbol „ $\perp$ “ või kaudne tagasilükkamine (ing. *implicit rejection*), mille tulemusena tagastatakse pseudojuhuslik võti. Antud protokolli jaoks oli valitud otsene tagasilükkamine, sest see on lihtsam ja kiirem.

Oli otsustatud mitte räsida kogu avalikku võtit või kogu krüptogrammi, sest see raskendaks ja aeglustaks protokolli teostust, nõuaks rohkem mälu ja takistaks võtmete genereerimist ja dekrüpteerimist.

Võtme genereerimise ja kapseldamise algoritmid nõuavad $d \times d$ maatriksi korrutamist vektoriga ja selle korrutamise optimeerimiseks oli valitud Karatsuba algoritm. Korrutamiseks kuuluv aeg on võrreldav RSA krüpteerimisega väikese astendajaga.

Avalikke võtmeid, salajasi võtmeid ja krüptogramme säilitatakse ja edastatakse baidijadadena. ThreeBearsis kasutatavad kodeeringud on pöördjärjestusega ja kindla pikkusega. Kuna elementide arv järjestuses on alati fikseeritud selle tüübi ja parameetrite komplekti järgi, ei tekki vajadust pikkuse täidistamiseks. ThreeBears'i kapslid sisaldavad 4-bitiste näkside järjestust, need on elemendid, mis kuuluvad $\{0, \cdots, 15\}$. Neid kodeeritakse, pakkides kaks näksi ühe baidi sisse pöördjärjestusega. Avalik võti koosneb maatriksi seemnest, mille pikkus on 24 baidi ja väärtusest *A*, milles on 3*312 = 936 elemendi ja kuna ühe elemendi kohta on 10 bitti, *A* = 936*10/8 = 1170 baiti, seega avaliku võtme suurus on kokku 24 + 1170 = 1194 baiti. Krüptogramm koosneb väärtusest *B*, mis

on sama pikkusega, kui A ja krüpteeritud ligikaudsest jagatud saladusest C_b, seega krüptogrammi pikkus on 1170 + 137 = 1194 baiti.

4 Võrdlemine

Uuritud artiklite põhjal koostati tabel (Tabel 1), mis võrdleb kõiki vaadeldud protokolle.

NewHope [7], Kyber [13], NTRU-HRSS [16] tsüklite arv oli saavutatud 3.50 GHz Intel Core i7-4770K (Haswell) protsessori peal testimisega, Frodo oli testitud 2.6 GHz Intel Xeon E5 (Sandy Bridge) protsessoril [10] ja Three Bears oli testitud 2.3GHz Intel NUC6i5SYH Core i3-6100U (Skylake) protsessoril [20].

Tulbas „Võreprobleem“ on märgitud protokolli aluseks olev võreprobleem. Tulbas „Turvalisus“ on märgitud, kas tegemist on passiivse või aktiivse turvalisusega protokolliga. Tulbas „Turvaaste“ olevad väärtused näitavad, milline on selle protokolli post-kvant turvaaste, lähtudes „parim tuntud kvantründe“ hinnangust, mida kirjeldatakse NewHope protokolli leiutajate poolt. Tulbas „δ“ on esitatud protokolli läbikukkumise tõenäosus, ehk kui suure tõenäosusega protokolli täitmise lõpus saavad osapooled üksteisest erinevad võtmed. Tsüklite arv on näidatud võtme genereerimise (V) jaoks, mida teostab server; võtme kapseldamise (E) jaoks, mis on NewHope ja Frodo puhul võtme genereerimine ja sümmeetrilise võtme arvutamine kliendi poolt; võtme lahtikapseldamise (D) jaoks, mis on NewHope ja Frodo puhul sümmeetrilise võtme arvutamine serveri poolt. Viimases tulbas on näidatud salajase võtme (*sk*), avaliku võtme (*pk*) ja krüptogrammi (*c*) suurused baitides. Avalik võti on see, mille Aliisa (server) saadab Bobile (klient) ja NewHope, Frodo ja Kyber puhul sisaldab ka seeme ja krüptogrammi saadab Bob (klient) Aliisale (server). Kuna mälu kasutamise küsimust ei loeta nende protokollide puhul tavaliselt suureks probleemiks, konkreetsed mälu kasutamise arvud ei olnud kättesaadavad suurest osast uuritud artiklitest ja seetõttu töös seda aspekti ei võrrelda.

Tabel 1. Võtmekehtestusprotokollide teostuse võrdlemine.

Nimetus	**Võreprobleem**	**Turvalisus**	**Turvaaste**	***δ***	**Tsüklite arv**	**Baidid**
NewHope (AVX2 optimeeritud)	Ring-LWE	Passiivne	255	2^{-60}	**V:** 88 920	**sk:** 1 792
					E: 110 986	**pk:** 1 824
					D: 19 422	**k:** 2 048
Frodo (soovitatud parametrid)	LWE	Passiivne	130	$2^{-38.9}$	**V:** 2 938 000	**sk:** 11 280
					E: 3 484 000	**pk:** 11 296
					D: 338 000	**k:** 11 288
Kyber (AVX2 optimeeritud)	Module-LWE	CCA	161	2^{-142}	**V:** 85 472	**sk:** 2 400
					E: 112 660	**pk:** 1 088
					D: 108 904	**k:** 1 152
NTRU-HRSS KEM	NTRU	CCA2	123	0	**V:** 307 914	**sk:** 1 418
					E: 48 646	**pk:** 1 138
					D: 67 338	**k:** 1 278
Three Bears (Mama Bear)	Täisarvu Module-LWE	CCA	219	2^{-147}	**V:** 79 000	**sk:** 40
					E: 97 000	**pk:** 1 194
					D: 152 000	**k:** 1 307

Kui analüüsida avaliku võtme ja krüptogrammi suurust, ehk neid komponente, mida tuleb üle võrku edastada, on näha, et Frodo väärtused on teistest märkimisväärselt suuremad. See on tingitud maatriksite kasutamisest seal, kus teistes protokollides kasutatakse polünoome. Praegu TLS'is kasutatavas eliptkõvera Diffie ja Hellmani protokollis X25519 on need suurused ainult 32 baidised. Seega mida suuremad on uute protokollide poolt pakutavad avaliku võtme ja krüptogrammi suurused, seda keerulisem on nende kasutuselevõtt TLS protokolli sisse rakendades, sest suurenenud sõnumite maht võib negatiivselt mõjutada protokolli jõudlusele ja kiirusele, mis on põhjustatud tekkinud ajaliste viivitustega.

Kui analüüsida tsüklite arvu, mis näitab kui palju on vaja aega ühe lihtsa protsessori operatsiooni teostamiseks, on näha, et kõige suuremad väärtused on Frodo protokollil ja on põhjustatud maatriksarvutustest, kuid nende väärtuste vähendamiseks on võimalik Frodo protokolli optimeerida näiteks, rakendades vektoriseerimist, ehk maatriksite veeruvektoriteks teisendamist.

Võrreldes Frodo näitajatega nii võtme ja krüptogrammi suuruses kui ka tsüklite arvus ja protokolli läbikukkumise tõenäosuses, pakub NewHope palju paremat teostust ja suuremat turvaastet. Teisest küljest aga, kuna on ta passiivse turvalisusega protokoll, see on vähem paindlik võrreldes teiste CCA turvaliste skeemidega, sest NewHope puhul rikub efemeerväärtuste vahemällu salvestamine protokolli turvalisust.

Kyber protokolli teostus pakub konkurentsvõimelisi näitajaid võrreldes teiste protokollidega. Kyber võtme genereerimine on palju kiirem ja seega odavam, kui NTRU-HRSS võtme genereerimine, sest NTRU protokollis kasutatav ring ei toeta NTT rakendamist ja teine põhjus on see, et NTRU võtme genereerimine nõuab ka jagamist, samal ajal kui Kyber nõuab ainult korrutamist. Turvalisuse poolt vaadates, Kyber protokollil on ka head näitajad, Ring-LWE probleemil põhinevatel skeemidel, nagu NewHope, skeemi turvalisuse muutmine eeldab protokolli baasiks oleva ringi mõõtmete muutmist, mis omakorda tähendab ka kõikide ringis tehtavate operatsioonide teostuse muutmist. Samas Kyber protokollis ring on alati sama ja turvaaste muutmiseks on vaja muuta ainult mooduli järku (ing. module dimension), mis nõuab väga väikest koodimuutust.

Väga sarnased Kyber teostuse näitajatega on ThreeBears teostuse näitajad. Kuigi protokolli parameetrid ei toeta NTT rakendamist, pakub ta kõige kiiremat võtme genereerimist ning kapseldamiseks kuuluvate tsüklite arv on Kyber protokolliga võrreldes väiksem, kuid lahtikapseldamise aeg on palju suurem, mis on tingitud sellest, et lahtikapseldamise protsessi jooksul on vaja korrata kogu kapseldamise protsessi, et kontrollida, kas teiselt osapoolelt saadud krüptogramm oli aus. Protokolli puuduseks on ka see, et täisarvude Module-LWE probleem ei olnud sama põhjalikult uuritud nagu LWE või Ring-LWE, mis võib viia probleemi nõrkade kohtade leidmiseni tulevikus, mis potentsiaalselt võib pakutud skeemi muuta ebaturvaliseks.

Kõigi tabelis toodud tulemusi arvestades võib kõige efektiivsemaks ja soodsamaks nimetada NTRU-HRSS, mille eelis on kiirte arvutuste ja suhteliselt väiksematele võtmetele lisaks on kõrvaldatud läbikukkumise tõenäosus. NTRU-HRSS puhul on võtme genereerimiseks vaja palju rohkem tsükleid võrreldes teiste protokollidega, kuid kuna tegemist on CCA2 turvalise võtmekehtestusega, on serveritel lubatud salvestada vahemällu ja taaskasutada efemeervõtmeid, mis aitab vähendada tsüklite arvu, mida vajatakse võtme genereerimiseks. Võrreldes teiste protokollidega, NTRU-HRSS suur eelis on ka see, et protokolli baasiks olev NTRU probleem põhjalikumalt läbi uuritud, kui teised probleemid ja on üle elanud liigikaudu 20 aastat krüptoanalüüsi. NTRU-HRSS miinus on see, et kõrvaldatud läbikukkumise tõenäosuse tulemuseks on madalam turvatase kui võiks olla sama dimensiooni ja mooduliga, kuid suurema müraga.

5 Kokkuvõtte

Viimastel aastatel on suurenedud huvi kvantarvutite ehitamise vastu nii suurte ettevõtete, teadusasutuste (Microsoft, Google, IBM, Intel) kui ka valitsuste poolt. Seoses sellega on välja pakutud mitmeid kvantarvutuskindlaid krüptograafilisi primitiive ja alanud NIST poolt korraldatav post-kvant algoritmide standartiseerimise protsess. Pakutud lahenduste seas on võrepõhised krüptograafilised algoritmid. Neid peetakse kvantarvutuskindlateks ja kõige lootustandvamaks krüpteerimise ja võtmekehtestuse jaoks. Võrreldes teiste post-kvant algoritmide peredega, on võrepõhised algoritmid suhteliselt lihtsad, efektiivsed ja kiired. Kuid võrepõhistel algoritmidel on ka puudusi, mis takistavad nende algoritmide kiiret praktilist kasutuselevõtu. Esiteks, on suurenenud võtmed, võrreldes tänapäeval kasutavatega ja teiseks, nõudlus effektiivse lepitusalgoritmi järele, mis aitaks vähendada protokolli läbikukkumise tõenäosust.

Käesolev töö annab ülevaade võtmekehtesuse rollist andmeside turvalisuse tagamises ja post-kvant krüptograafiast, kirjeldab võrepõhise võtmekehtestuse seost peamiste võrepõhiste arvtusprobleemidega, analüüsib viit erinevat võrepõhist võtmekehtestusprotokolli (NewHope, Frodo, Crystals-Kyber KEM, NTRU-HRSS KEM, Three Bears), kirjeldab nende protokollide tööpõhimõttet nii skeemiliselt, kui ka sügavama ja parema ülevaade andmiseks sõnaliselt, vaatab läbi parameetrite valikuid ja kõige olulisemaid teostusaspekte. Lõpuks, töö analüüsib ja võrdleb omavahel protokollide jõudlust ning reaalmaimas teostamise võimalikkust, kirjeldades ka protokollide teostamisega seotud elliseid ja puudusi.

Protokollide jõudluse võrdlemise tulemusena osutus kõige efektiivsemaks ja soodsamaks NTRU-HRSS protokoll, millel on mitu eelist võrreldes teiste võrepõhiste protokollidega. Esiteks, kiired arvutused, suhteliselt väiksed võtmed ja krüptogrammid ning kõrvaldatud protokolli läbikukkumise tõenäosus, mis võimaldab NTRU-HRSS efektiivset kasutamist TLS turvaprotokollis. Teiseks on see, et protokolli baasiks olev arvutusprobleem on põhjalikumalt uuritud, võrreldes teiste võrepõhiste arvutusprobleemidega.

Lõputöö eesmärgid saavutati, sest antud töös on analüüsitud, kirjeldatud ja võrreldud peamisi võrepõhiseid võtmekehtestusprotokolle, mida NIST'i matemaatikud ja arvutiteadlased peavad kõige tugevamateks kandidaatideks post-kvant krüptograafia standardiseerimisel. Töö käigus selgus, et paljud vaadlevatest protokollidest võivad tõeliselt olla tulevikus kasutatud eliptkõverate Diffie ja Hellmani võtmekehtestuse asemel või ka koos sellega. Käesolev lõputöö võib olla abimaterjaliks sügavamate post-kvant krüptograafia uuringutele ja tulevikus on võimalik seda tööd jätkata analüüsides põhjalikumalt nende võtmekehtestusprotokollide teostust ja jõudlust, uurides kuidas võib protokollide teostust rohkem optimeerida, mitte ainult suurte arvutite ja nutitelefonide jaoks, vaid ka piiratud protsessorivõimsustega sedmete jaoks, sest kiipkaardid ja erinevad asjade interneti väiksed seadmed peavad samuti olema võimelised töötama post-kvant krüptograafia primitiividega.

Kasutatud kirjandus

[1] What is Quantum Key Distribution? [WWW] https://www.quintessencelabs.com/wp-content/uploads/2015/08/CSA-What-is-Quantum-Key-Distribution-QKD-1.pdf (18.02.2019)

[2] Chen L., Jordan S., Liu Y., Moody D., Peralta R., Perlner R., Smith-Tone D. (2016). Report on Post-Quantum Cryptography. [WWW] https://nvlpubs.nist.gov/nistpubs/ir/2016/NIST.IR.8105.pdf (17.02.2019)

[3] Mosca M. (2015). Cybersecurity in an era with quantum computers: will we be ready? [WWW] https://eprint.iacr.org/2015/1075.pdf (17.02.2019)

[4] Vandersypen L. M. K., Steffen M., Breyta G., Yannoni C. S., Sherwood M., Chuang I. L. (2001). Experimental realization of Shor's quantum factoring algorithm using nuclear magnetic resonance. [WWW] http://cryptome.org/shor-nature.pdf (18.02.2019)

[5] Quantum Safe Cryptography and Security; An introduction, benefits, enablers and challenges [WWW] https://portal.etsi.org/Portals/0/TBpages/QSC/Docs/Quantum_Safe_Whitepaper_1_0_0.pdf (18.02.2019)

[6] Stebila D. (2018). Introduction to post-quantum cryptography and learning with errors. [WWW] https://summerschool-croatia.cs.ru.nl/2018/slides/Introduction%20to%20post-quantum%20cryptography%20and%20learning%20with%20errors.pdf (18.02.2019)

[7] Alkim E., Ducas L., Pöppelmann T., Schwabe P. (2015). Post-quantum key exchange – a new hope. [WWW] https://eprint.iacr.org/2015/1092.pdf (18.02.2019)

[8] Langley A. (2015). Post-quantum key agreement. [WWW] https://www.imperialviolet.org/2015/12/24/rlwe.html (28.02.2019)

[9] Aysu A., Tobah Y., Tiwari M., Gerstlauer A., Orshansky M. (2018). Horizontal Side-Channel Vulnerabilities of Post-Quantum Key Exchange Protocols. [WWW] http://spark.ece.utexas.edu/pubs/HOST-18-pqdpa.pdf (18.02.2019)

[10] Bos J., Costello C., Ducas L., Mironov I., Naehrig M., Nikolaenko V., Raghunathan A., Stebila D. (2016). Frodo: Take off the ring! Practical, Quantum-Secure Key Exchange from LWE. [WWW] https://eprint.iacr.org/2016/659.pdf (18.02.2019)

[11] The Transport Layer Security (TLS) Protocol. Version 1.2 [WWW] https://tools.ietf.org/html/rfc5246 (08.03.2019)

[12] Bos J., Ducas L., Kiltz E., Lepoint T., Lyubashevsky V., Schanckk J. M., Schwabe P., Seiler G., Stehlé D. (2017). CRYSTALS -- Kyber: a CCA-secure module-lattice-based KEM. [WWW] https://eprint.iacr.org/2017/634.pdf (18.02.2019)

[13] Avanzi R., Bos J., Ducas L., Kiltz E., Lepoint T., Lyubashevsky V, Schanck J. M., Schwabe P., Seiler G., Stehlé D. (2017). CRYSTALS-Kyber. Algorithm Specifications And Supporting Documentation. [WWW] https://pq-crystals.org/kyber/data/kyber-specification.pdf (18.02.2019)

[14] Avanzi R., Bos J., Ducas L., Kiltz E., Lepoint T., Lyubashevsky V, Schanck J. M., Schwabe P., Seiler G., Stehlé D. (2018). CRYSTALS–Kyber presentation for First PQC Standardization Conference. [WWW] https://csrc.nist.gov/CSRC/media/Presentations/Crystals-Kyber/images-media/Crystals-Kyber-April2018.pdf (18.02.2019)

[15] Hülsing A., Rijneveld J., Schanck J., Schwabe P. (2017). High-speed key encapsulation from NTRU. [WWW] https://cryptojedi.org/papers/ntrukem-20170828.pdf (10.03.2019)

[16] Hülsing A., Rijneveld J., Schanck J., Schwabe P. (2017). NTRU-HRSS-KEM. Algorithm Specifications And Supporting Documentation [WWW] https://cryptojedi.org/papers/ntrukemnist-20171130.pdf (21.03.2019)

[17] Hülsing A., Rijneveld J., Schanck J., Schwabe P. (2018). NTRU-HRSS-KEM presentation for First PQC Standardization Conference. [WWW] https://csrc.nist.gov/CSRC/media/Presentations/NTRU-HRSS-KEM/images-media/ntru-hrss-kem-April2018.pdf (21.03.2019)

[18] Hülsing A., Rijneveld J., Schanck J., Schwabe P. (2017). High-speed key encapsulation from NTRU presentation. [WWW] https://joostrijneveld.nl/talks/20170926_ches_ntrukem.pdf (21.03.2019)

[19] Saito T., Xagawa K., Yamakawa T. (2017). Tightly-Secure Key-Encapsulation Mechanism in the Quantum Random Oracle Model. [WWW] https://eprint.iacr.org/2017/1005.pdf (18.02.2019)

[20] Hamburg M. (2017). Post-quantum cryptography proposal: ThreeBears [WWW] https://www.shiftleft.org/papers/threebears/nist-submission.pdf (18.02.2019)

[21] Hamburg M. (2018). ThreeBears postquantum KEM presentation for First PQC Standardization Conference. [WWW] https://csrc.nist.gov/CSRC/media/Presentations/Three-Bears/images-media/ThreeBears-April2018.pdf (18.02.2019)

[22] Kelsey J., Chang S., Perlner R. (2016). SHA-3 Derived Functions: cSHAKE, KMAC, TupleHash and ParallelHash. [WWW] https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-185.pdf (21.03.2019)

[23] Braithwaite M. (2016). Experimenting with Post-Quantum Cryptography. [WWW] https://security.googleblog.com/2016/07/experimenting-with-post-quantum.html (28.02.2019)

[24] Langley A. (2016). CECPQ1 results. [WWW] https://www.imperialviolet.org/2016/11/28/cecpq1.html (28.02.2019)

[25] Langley A. (2018). Post-quantum confidentiality for TLS. [WWW] https://www.imperialviolet.org/2018/04/11/pqconftls.html (28.02.2019)

[26] Langley A. (2018). CECPQ2. [WWW] https://www.imperialviolet.org/2018/12/12/cecpq2.html (28.02.2019)

[27] Pankova A., Willemson J., Buldas A. (2018). Postkvant-krüptograafia ülevaade. [WWW] https://www.ria.ee/sites/default/files/content-editors/publikatsioonid/postkvant-kruptograafia-ulevaade-2018.pdf (24.04.2019)

[28] Pau A. Kvantarvutid ohustavad Eesti ID-kaarti. [WWW] https://tehnika.postimees.ee/6544199/kvantarvutid-ohustavad-eesti-id-kaarti (24.04.2019)

[29] Abel M., Kaasik Ü. Matemaatikasõnastik. [WWW] http://www.keeleveeb.ee/dict/speciality/mathematics/ (18.02.2019)

[30] Hanson V., Buldas A., Veldre A., Laur M., Krasnosjolov J. Andmekaitse ja infoturbe leksikon [WWW] https://akit.cyber.ee/#showlist (18.02.2019)