Tartu Ülikool
Arvutiteaduse instituut
Informaatika õppekava

Richard Mario Raun

Raspberry Pi baasil turvalise ja hajusa andmehoidla loomine

Bakalaureusetöö (9 EAP)

Juhendaja:
Alo Peets

Tartu 2025

Raspberry Pi baasil turvalise ja hajusa andmehoidla loomine

Lühikokkuvõte:

Töö eesmärk on testida erinevate andmekandjate kiiruseid Raspberry Pi-ga kasutamisel ja parimast leitud salvestusseadmest ehitada Raspberry Pi 5 baasil turvaline ja hajus andmehoidla, millel on kaitse küberlunaraha rünnakute vastu läbi versioonihalduse. Andmekandjateks on erinevad populaarsed SD-kaardid ja läbi M.2 Hat+ mooduli Raspberry Pi pooljuhtkettad. Andmehoidla hõlmab endas vähemalt kahte Raspberry Pi 5, mis on füüsiliselt erinevates kohtades, et tagada andmete liiasus. Lahenduse loomine sisaldas endas nelja väiksema süsteemi loomist ja nende ühendamist. Võrgusalvesti loodi kasutades OpenMediaVault Linuxi distributsiooni. Andmete kaugligipääsetavus saavutati läbi virtuaalse privaatvõrgu loomise kasutades WireGuardi. Masinate vahel andmete sünkroniseerimine saavutati kasutades Syncthingi ning versioonihalduse jaoks kasutati Borgi. Töö tulemuseks saavutati kulutõhus, vastupidav ja turvaline andmehaldussüsteem, mis on sobilik kasutamiseks nii isiklikel eesmärkidel kui ka väiksemates organisatsioonides.

Võtmesõnad:

Raspberry Pi, Andmekandjad, Võrgusalvestid, Turvaline, Hajus, Virtuaalne privaatne võrk

CERCS:

P170. Arvutiteadus, arvutusmeetodid, süsteemid, juhtimine

Secure and Distributed Data Storage based on Raspberry Pi

Abstract:

The objective of the thesis is to evaluate the performance of various data mediums on the Raspberry Pi 5 and to design a secure and distributed data storage on the most effective medium. The system will include protection against cyber ransomware attacks through version control. The data mediums will consist of several popular SD cards and Raspberry Pi solid-state drives connected through the M.2 HAT+ module. The data storage solution includes at least two Raspberry Pi 5 devices located in physically separate locations to ensure data redundancy. The development of the solution consisted of creating four smaller subsystems and integrating them. Network attached storage was implemented using the OpenMediaVault Linux distribution. Remote data access was achieved by establishing a virtual private network using WireGuard. Data synchronization between devices was implemented with Syncthing, and version control was managed using Borg. As a result, a cost-effective, resilient and secure data management system was created, suitable for both personal use and small-scale organizational environments.

Keywords:

Raspberry Pi, Data mediums, Network attached storage, Security, Distributed, VPN

CERCS:

P170. Computer science, numerical analysis, systems, control

Sisukord

1. Sissejuhatus

Peatükk annab ülevaate lõputöös käsitletud probleemist. Sealhulgas tutvustab kasutatud riistvara, kontseptsioone, Raspberry Pi-d koos lisamoodulitega ning annab ülevaate lõpproduktist. Täiendavalt selgitatakse töö käigus seatud eesmärke ja nende saavutamiseks kasutatud lähenemisviise. Samuti antakse lühiülevaade süsteemi arhitektuurist ja selle põhikomponentidest, mis tagavad andmete turvalisuse ja kättesaadavuse.

1.1 Probleemi kirjeldus

Isiklike andmete, näiteks multimeedia failide, dokumentide, krüptograafiliste võtmete ja varukoopiate eksponentsiaalse kasvu tõttu vajavad inimesed üha enam usaldusväärseid, turvalisi, kulutõhusaid ja kergesti ligipääsetavaid salvestuslahendusi. Kuigi pilveteenused on mugavad, tekitavad need privaatsusprobleeme, tellimistasusid ja neid kasutades puudub kasutajal informatsioon enda andmete käsitlusviiside kohta. Lisaks on need platvormid koos kohalike salvestusseadmetega üha haavatavamad küberohtude, eelkõige lunavararünnakute vastu. Selliste rünnakute käigus krüpteeritakse kriitilisi kasutajaandmeid, sealhulgas tundlikke faile ja krüptograafilisi võtmeid, sundides ohvreid juurdepääsu taastamiseks lunaraha maksma.

Isikliku võrgusalvesti loomine maandab eelnimetatud riske, pakkudes kasutajale täielikku kontrolli enda andmete üle, rohkem privaatsust, versioonihaldust ja üleüldist paremat vastupidavust küberohtudele. Versioonihalduse abil saavad kasutajad krüpteeritud või kahjustunud failid taastada, tehes seeläbi rünnaku kahjutuks.

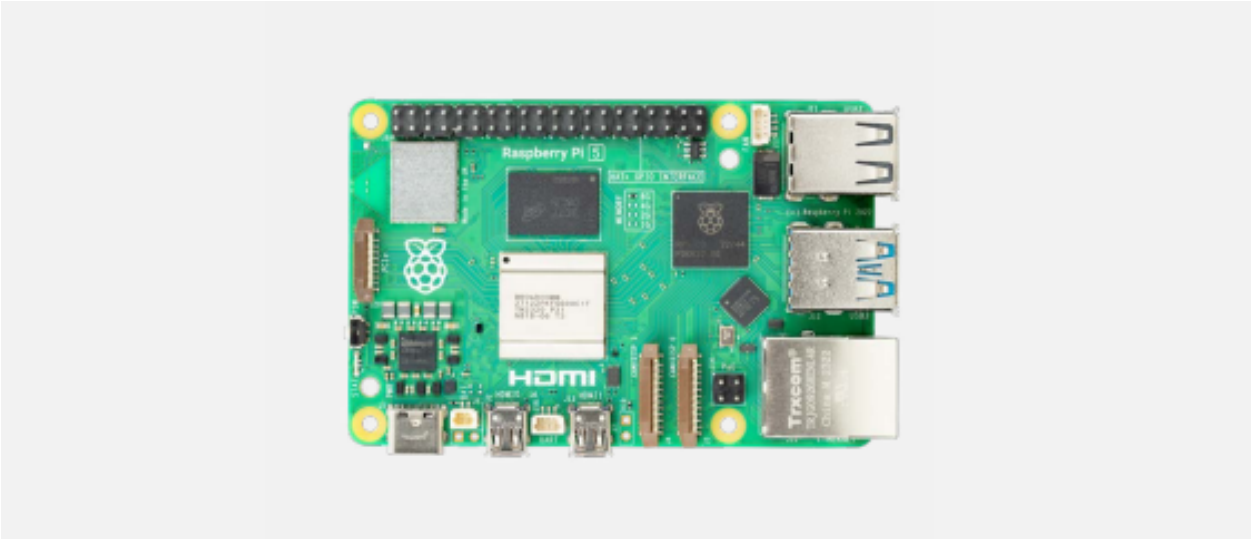
Versioonihaldusega, turvalise ja hajusa isikliku võrgusalvesti loomine tõstatab küsimusi andmekandja valiku kohta. On ebaselge, kuidas laialdaselt saadaval olevad andmekandjad nagu SD-kaardid ja pooljuhtkettad toimivad tüüpiliste NAS-i töökoormuste, versioonihaldus ja varundus toimingute korral, eriti kui neid kasutatakse piiratud jõudlusega keskkondades nagu Raspberry Pi.

Käesoleva töö eesmärk on leida erinevate andmekandjate mõju Raspberrry Pi 5 jõudlusele ning parimast leitud andmekandjast ehitada Raspberry Pi baasil turvaline ja hajus andmehoidla koos versioonihaldus süsteemiga.

1.2 Raspberry Pi 5 ülevaade

Raspberry Pi 5 on umbes pangakaardi suurune monoplaatarvuti (SoC - *single-board computer*) [1] [2]. Monoplaatarvuti [3] ehk plaatarvuti on arvuti, kus on kõik vajalikud osad ühe trükkplaadi peal. Plaatarvuti tähtsaim osa on ühe-kiibi-arvuti (SoC). Ühe-kiibi-arvuti[1] on kiip, millele on kas täienisti või suuremas osas arvuti või elektronsüsteem sisse integreeritud.

Arvuti peamisteks kasutusvaldkondadeks on prototüüpimine, professionaalsed lahendused ja hariduses kasutamine [2] . Erinevalt tavalisest arvutist on Raspberry Pidel GPIOd. GPIO[2] (*General Purpose Input/Output*) ehk universaalport on mikroprotsessori või mikrokontrolleri liides, mis on võimalik programmeerida nii signaali saajaks kui ka vastuvõtjaks. Tänu oma taskukohasele hinnale, mis jääb ligikaudu saja euro piiresse, on Raspberry Pi populaarne nii hobikasutajate kui ka haridusasutuste seas.



Joonis 1. Raspberry Pi 5 [1].

1.3 Miks kasutada Raspberry Pi'd personaalse võrgusalvestina

Andmehaldussüsteemide loomisel on Raspberry Pi energiatõhus ja odav lahendus. Raspberry Pi süsteemil on võimalik luua nii võrgusalvesteid[3] (NAS - *network-attached storage*) kui ka salvestusvõrke[4] (SAN - *storage area network*). Võrgusalvesti ehk NAS on arvuti, mis on

[1] ANDMEKAITSE JA INFOTURBE PORTAAL. (2025). https://akit.cyber.ee/term/14661
[2] ANDMEKAITSE JA INFOTURBE PORTAAL. (2025). https://akit.cyber.ee/term/10345-gpio
[3] ANDMEKAITSE JA INFOTURBE PORTAAL. (2024). https://akit.cyber.ee/term/10253-vorgusalvesti
[4] ANDMEKAITSE JA INFOTURBE PORTAAL. (2024). https://akit.cyber.ee/term/1070-salvestusvork

spetsialiseerunud andmesalvestusteenuste pakkumiseks teistele võrguseadmetele. Erinevalt serverist keskendub NAS ainult failide jagamisele ja varundamisele [10]. NAS-süsteemid võimaldavad erinevatel kasutajatel samaaegselt kasutada samu failipääsu teenuseid, mis lihtsustab andmete haldamist ja jagamist [4]. Lisaks on Raspberry Pi leidnud kasutust ka SAN-süsteemides, kus selle eelisteks on madal hind, kompaktsus ja skaleeritavus [4]. Nasser jt [5] demonstreerisid oma uurimistöös, kuidas Raspberry Pi koos NVMe[5] pooljuhtketastega[6] võib saavutada kuni 1 gigabitt sekundis andmeedastuskiirust. Sellest võib järeldada, et kuigi Raspberry Pi on odav ja tagasihoidliku välimusega, siis on ta võimeline pakkuma kõrget jõudlust.

1.4 Raspberry Pi 5 spetsifikatsioon

Raspberry Pi 5 tehnilised andmed tuginevad Raspberry Pi Foundationi ametlikule veebilehele [6]. Selle järgi on Raspberry Pi 5 üles ehitatud Broadcom BCM2712 SoC kiibi peale. Kiibi keskseks elemendiks on 2.4GHz neljatuumaline 64-bitine Arm Cortex-A76 protsessor koos krüptograafia laiendustega ja 512kB L2-vahemälu igale tuumale ning 2 MB jagatud L3-vahemälu. Graafikakaardiks on arvutil VideoCore VII, mis toetab OpenGL ES3.1-te ja Vulkan1.2-te. See võimaldab samaaegselt kahte 4Kp60 HDMI-väljundit koos HDR-toega ning toetab 4Kp60 HEVC-dekodeerimist. Vahemäluna on kasutusel LPDDR4X-4267 SDRAM, millest on kolm valikut: 2GB, 4GB ja 8GB. Andmesalvestuseks on mikro SD-kaardi pesa koos SDR104 toega. Traadita ühenduse loomiseks on olemas kahesageduslik 802.11ac Wi-Fi, Bluetooth 5.0 ja BLE. Seadmel on kaks USB 3.0 porti (5 Gbps) ja lisaks kaks USB 2.0 porti. Täiendavalt leiab kaks neljarealist MIPI kaamera-/ekraamiliidest, PCIe 2.0 x1 laienduspesa (vajab eraldi M.2 HAT-i või vastavat adapterit) ning 5V/5A toite USB-C kaudu (Power Delivery tugi). Raspberry Pi 5-l on standardne 40-viigine[7] GPIO päis, integreeritud reaalajakell (RTC), mida saab vooluga varustada välisest akust ja toitenupp.

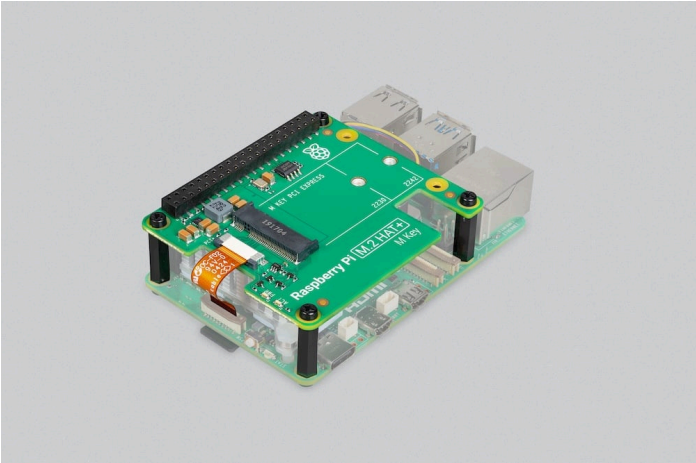
[5] NVM EXPRESS, “NVM Express” (2025). https://nvmexpress.org/
[6] ANDMEKAITSE JA INFOTURBE PORTAAL. (2024). https://akit.cyber.ee/term/7151
[7] ANDMEKAITSE JA INFOTURBE PORTAAL. (2024). https://akit.cyber.ee/term/16655-viik

1.5 M.2 Hat+

Raspberry Pi M.2 HAT+ võimaldab Raspberry Pi 5 külge ühendada erinevaid M.2 mooduleid. See adapterplaat muundab PCIe-ühenduse M.2 M Key ühenduseks, mis annab võimaluse arvutiga ühendada 2230 ja 2242 kujuteguriga[8] M.2-seadmeid nagu NVMe pooljuhtkettad [7].

NVMe (Non-Volatile Memory Express) [8] on mäluprotokoll, mis kasutab PCI Expressi liidest, et pakkuda SSD-dele senisest suuremat andmeedastus võimet. Erinevalt SATA'st on NVMe loodud just pooljuhtketaste jaoks, võimaldades sadu tuhandeid I/O-käske sekundis.

Lisaks kasutab M.2 HAT+ Raspberry Pi HAT+ spetsifikatsiooni, mis võimaldab Raspberry Pi OS-il tuvastada sellele ühendatud seadmed automaatselt [7].



Joonis 2. Raspberry Pi M.2 HAT+ [9].

1.6 Küberlunaraha rünnakud ja ennetus Raspberry Pi NAS-is

Küberlunaraha rünnakud (*ransomware*) on pahavara (*malware*) rünnakud, mille käigus krüpteeritakse ohvri andmed ning nõutakse lunaraha nende uuesti kättesaadavaks tegemise eest [12][14]. Sellised rünnakud võivad ulatuda üksikisikute arvutitest kuni suurte ettevõtete keerukate serveriteni. Viimastel aastatel on tähelepanu all olnud ka võrgu külge ühendatud salvestusseadmed ehk NAS-id, mis võivad jääda ründajatele kergesti kättesaadavaks, kui neid pole õigesti konfigureeritud või kui turvameetmed on puudulikud [13].

[8] ANDMEKAITSE JA INFOTURBE PORTAAL. (2024). https://akit.cyber.ee/term/10253-vorgusalvesti

Raspberry Pi 5 baasil ehitatud NAS ei ole erand. Kuigi tegemist on väikese ja energiasäästliku seadmega, mis pole peamiseks sihtmärgiks suurte operatsioonide jaoks, võib ka see sattuda asjakohaste kaitsemehhanismide puudumisel küberlunaraha rünnaku ohvriks. Üheks oluliseks turvameetmeks on regulaarne süsteemivärskenduste tegemine. Lisaks on tähtis konfigureerida turvaline autentimine (näiteks SSH võtme põhine ligipääs või kahefaktoriline autentimine) ning tulemüür, mis piiraks juurdepääsu süsteemile ainult vajalikest võrgusegmentidest [11].

Tõhus kaitse küberlunaraha rünnakute vastu nõuab korrapäraseid varukoopiaid väljaspool NAS-i, näiteks välisel kettal või eraldiseisvas asukohas. Töös planeeriti salvestada andmed kolmele erinevale Raspberry Pi masinale, millest kahel on versioonihaldus süsteem nii, et viimase 30 päeva failide versioonid jäävad alles ja kõiki andmeid on võimalik 30 päeva jooksul taastada. Ehk, kui võtmed krüpteeritakse lunaraharünnaku käigus ära, saab need ilma suurema vaevata taastada. Andmete geograafilise liiasuse[9] saavutamiseks võib kasutada mitut füüsiliselt erinevas asukohas paiknevat Raspberry Pi süsteemi. Seeläbi on andmed kaitstud mitte ainult pahavara, vaid ka riistvara rikete ja muude ootamatute õnnetuste eest. Süsteemivärskenduste rakendamine, tulemüüri ja õiguste korrektne haldamine ning regulaarsed varukoopiad tagavad seadmele efektiivse kaitse ka küberlunaraha rünnakute vastu.

1.7 Töö struktuur ja kasutatavad tehnoloogiad

Töö struktuur on üles ehitatud mitmest koostööd tegevast komponendist, mille keskseks haldustööriistaks on OpenMediaVault. OpenMediaVault on tasuta kättesaadav Linuxi distributsioon, mis on disainitud haldama võrgusalvesti süsteeme ja seda peamiselt läbi graafilise kasutajaliidese. OpenMediaVault on paigaldatud Raspberry Pi seadmetele ja vastutab võrgusalvesti süsteemide haldamise eest, pakkudes kasutajale ligipääsu failisüsteemidele.

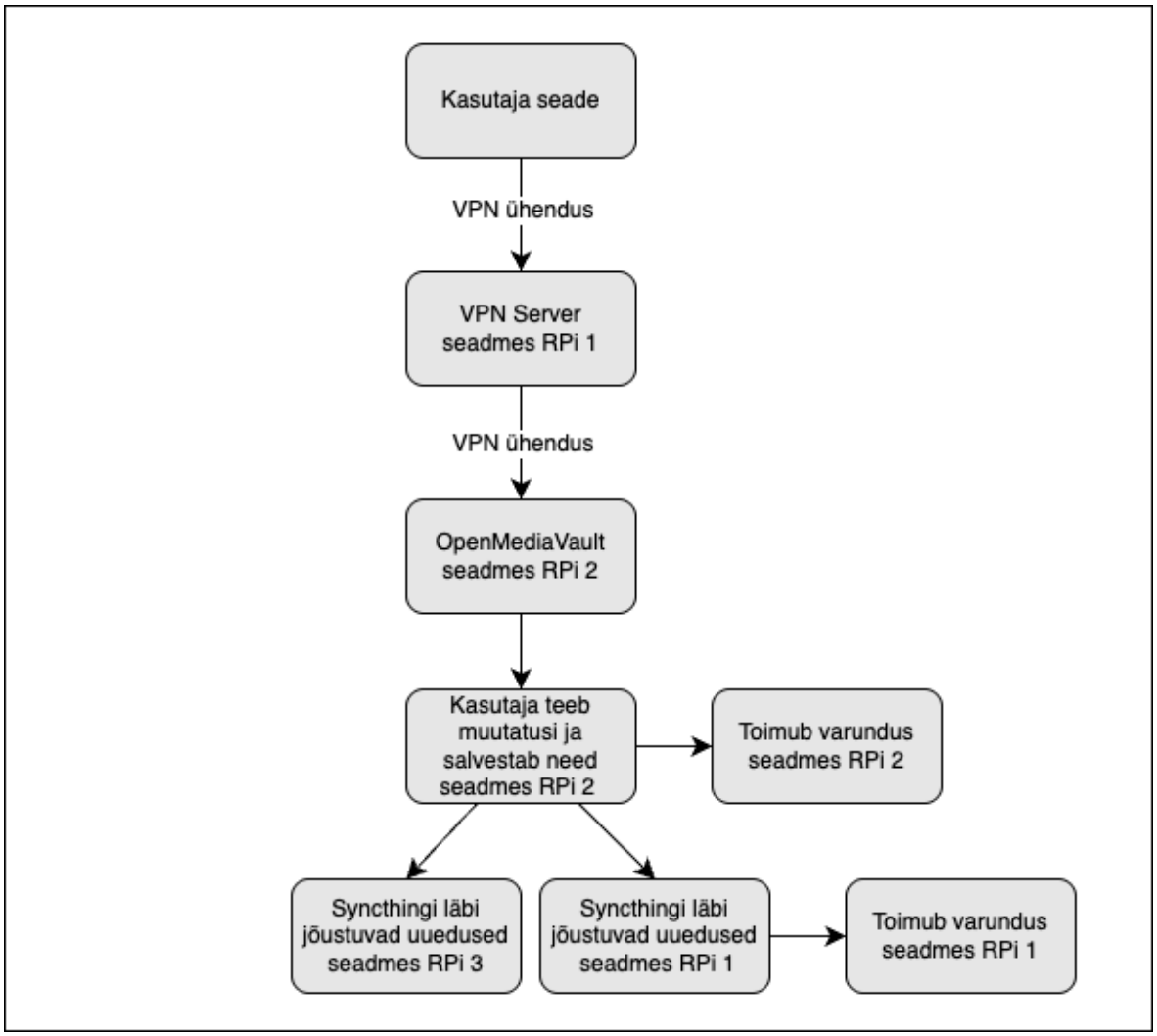
Andmete turvaliseks kaugjuurdepääsuks loodi WireGuard VPN, mis võimaldab kasutajatel krüpteeritud ühenduse kaudu liituda sisevõrguga ja pääseda võrgusalvestile. Pärast VPN-ühenduse loomist saavad kasutajad kasutada oma seadmete failihaldusrakendusi, et suhelda OpenMediaVaulti poolt hallatavate failiserveritega.

[9]ANDMEKAITSE JA INFOTURBE PORTAAL. (2025). https://akit.cyber.ee/term/49-liiasus

Failide ja muudatuste haldamine erinevate Raspberry Pi seadmete vahel toimub läbi Syncthingi, mis sünkroniseerib andmed reaalajas ja seda võrdvõrgu põhimõttel ilma keskse serverita. Nii liiguvad muudatused automaatselt kõigi süsteemi kuuluvate seadmete vahel.

Andmete kaitsmiseks ja varunduste tagamiseks kasutatakse Borgi ja Borgmaticu lahendust, mis teevad regulaarselt varukoopiaid. Enne igat muudatust salvestatakse andmete eelnev olek, et vajadusel oleks võimalik faile varasemale versioonile taastada.

Sellise ülesehitusega struktuur tagab, et kõik failihalduse ja varundamisega seotud tegevused toimuvad koordineeritult. Joonis 3 illustreerib seda protsessi kasutaja poolt tehtavate muudatuste näitel.



Joonis 3. Näide süsteemi struktuurist.

Seega moodustavad kirjeldatud tehnoloogiad ühtse ja turvalise süsteemi, mille toimimiseks on oluline ka piisava jõudlusega riistvara valik.

2. Raspberry Pi andmekandjate kiiruste võrdlus

Iga andmehoidla üheks olulisemaks komponendiks on salvestusseadmed. Otsuse tegemiseks võrreldatakse käesolevas peatükis erinevate andmekandjate kiirust ja jõudlust Raspberry Pi-l, et selgitada välja lahenduse jaoks kõige sobivam andmekandja.

2.1 Võrreldavad andmekandjad

2.1.1 Andmekandjad

Võrdluses kasutati erinevaid andmekandjaid. Võrreldi erinevaid mikro SD-kaarte. SD-kaartide osas huvituti kahte kategooriasse kuuluvatest SD-kaartidest. Raspberry Pi Foundationi poolt müüdavad spetsiaalsed Raspberry Pi mikro SD-kaardid ja jaekaubanduses müüdavad mikro SD-kaardid. Kokku võrreldi kümmet erinevat mikro SD-kaarti. Lisaks katsetati välist pooljuhtketast ja NVMe pooljuhtkettaid, et näha, kui suur on nende jõudluse erinevus võrreldes varem testitud SD-kaartidega.

2.1.2 Andmekandjate klassifikatsioonid

SD-kaartidel on standardiseeritud kiiruste klassifikatsioonid, mis väljendavad kaardi minimaalset kirjutamis- või lugemiskiirust ning tootja poolt soovitatud kaardi kasutusalasid. Speed class (class 2,4,6,8,10) väljendab minimaalset järjestikuse kirjutamise kiirust. Class 2 tagab vähemalt 2 MB/s kirjutamiskiiruse, Class 4 vähemalt 4 MB/s ning analoogne loogika kehtib ka kõrgemate klasside puhul. UHS Speed Class (U1, U3) väljendab samuti minimaalset püsivat kirjutamiskiirust, kuid lisaks näitab see, et kaart kasutab UHS (Ultra High Speed) andmesiini. U1 tähistab vähemalt 10 MB/s püsivat kirjutamiskiirust, U3 vähemalt 30 MB/s. UHS Bus Interface (UHS-I, UHS-II) väljendab andmesiini tehnoloogiat, mis määrab kaardi maksimaalse andmeedastuskiiruse. UHS-I väljendab kuni 104MB/s teoreetiliseks siini kiiruseks ja UHS-II kuni 312MB/s teoreetiliseks siini kiiruseks. Video Speed Class (V6, V10, V30, V60, V90) on disainitud spetsiaalselt video filmimise jaoks. Ka siin V6 tähistab minimaalselt vähemalt 6 MB/s stabiilselt kirjutamis kiirust ning analoogne loogika kehtib ka kõrgemate klasside puhul.Application Performance Class (A1, A2) väljendab kaardi sobivust programmide jooksutamisel otse kaardilt. A1 väljendab lugemisel minimaalselt 1500 suvalist IOPSi[10]

[10] ANDMEKAITSE JA INFOTURBE PORTAAL. (2025). https://akit.cyber.ee/term/9126-iops

(sisend-väljundoperatsiooni sekundis) ja kirjutamisel minimaalselt 500 suvalist IOPSi. A2 on kiirem ja väljendab lugemisel minimaalselt 4000 suvalist IOPSi ja kirjutamisel 2000.

2.2 Võrdlusmetoodika

2.2.1 Struktuur

Võrdlemisel peeti oluliseks, et kõigil kaartidel oleksid võrdsed tegurid ja võimalused. Operatsioonisüsteem käivitati mälupulgal, mikro SD-kaardi kiirusi mõõdeti Raspberry Pi mikro SD-kaardi pesas, välis pooljuhtketast mõõdeti ühendades USB 3.1 gen 2 kaabel läbi Raspberry Pi USB 3.0 SuperSpeed pesa ja NVMe ketas ühendati M.2 HAT+i abil, et pudelikaelasid ei tekiks. Et kõigi andmekandjate vaheline võrdlus oleks võrdne, laeti Raspberry Pi operatsioonisüsteemi mälupulga peale ja võrreldavaid andmekandjaid kasutati ainult testimiseks.

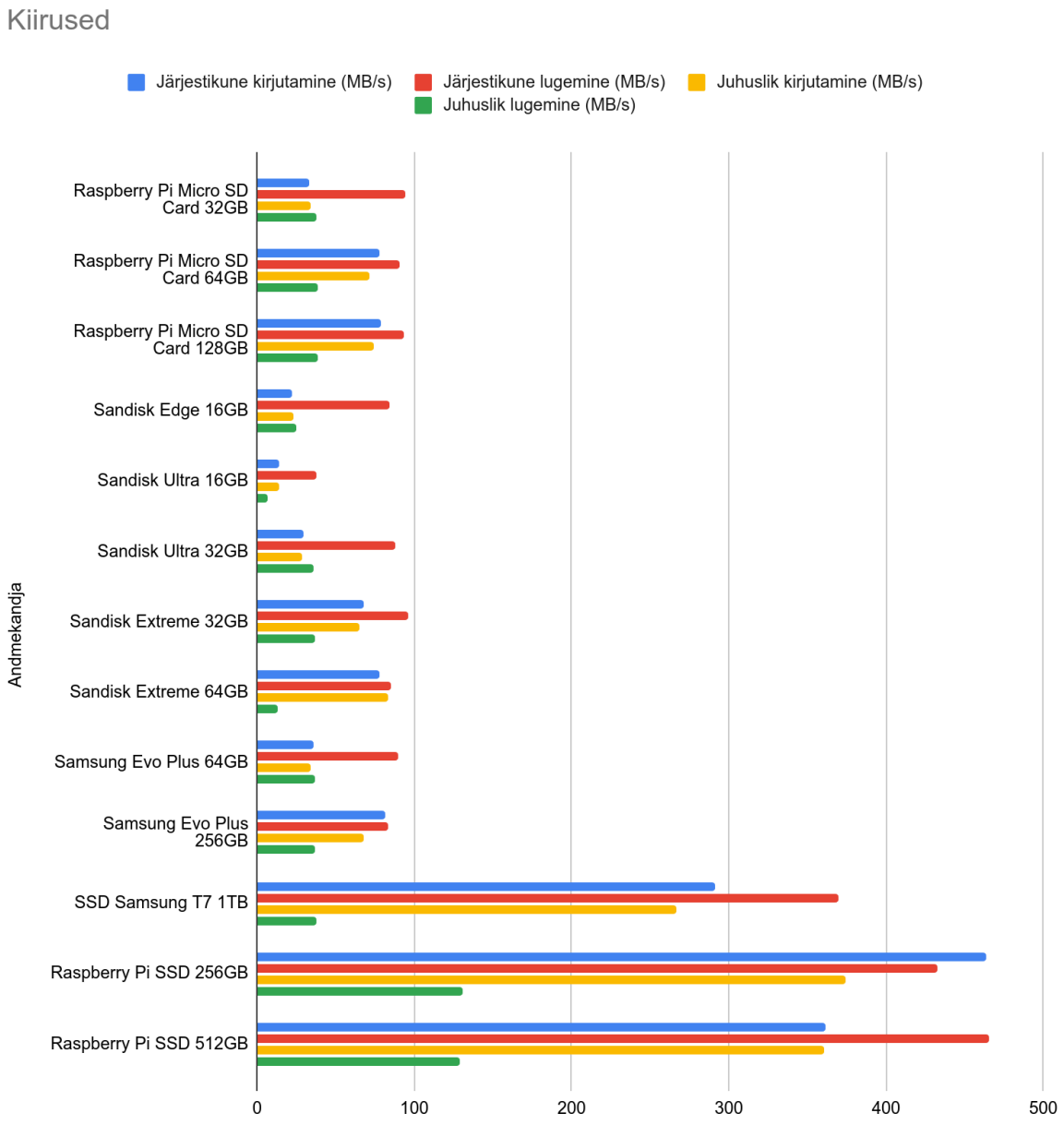
2.2.2 Võrdluseks valitud meetrikud

Mõõtmised viidi läbi automatiseeritud skripti, et tagada võrdsed tingimused kõikidele andmekandjatele. Testimisskript koosneb seitsmest erinevast testist. Järjestikune lugemine ja kirjutamine, juhuslik lugemine ja kirjutamine, madala tasemeline (dd) lugemine ja kirjutamine ning HDparm lugemine. Mõõtmiseks kasutati fio tööriista, mis on laialdaselt tunnustatud täpsete I/O mõõtmistulemuste poolest. Järjestikustes testides kasutati 1 MB ploki suurust ning juhuslikes testides kasutati 4 KB suuruseid plokke. Iga testi kestvuseks fikseeriti 60 sekundit, mille jooksul tööriist tegi pidevalt lugemis- või kirjutamistoiminguid 1 GB testfaili piires. Tulemused salvestati csv faili kasutades ühikuks megabait sekundis. Lisa II sisaldab jõudluse mõõtmiseks kasutatud skripti.

2.3 Tulemused

Tabel 1. Algandmetest kohandatud tabel. Algandmete tabel lisati töö lisade sektsiooni. Vaata lisa I.

Andmekandja	Eeldatav kirjuta-mise vahemik (MB/s)	Järjestikune kirjutamine (MB/s)	Järjestikune lugemine (MB/s)	Juhuslik kirjutamine (MB/s)	Juhuslik lugemine (MB/s)
Raspberry Pi Micro SD Card 32GB	30-90	33.24	94.06	34.29	37.33
Raspberry Pi Micro SD Card 64GB	30-90	78.12	90.81	71.3	38.48
Raspberry Pi Micro SD Card 128GB	30-90	78.85	93.11	74.24	38.27
Sandisk Edge 16GB	10-90	22.23	84.41	22.75	24.96
Sandisk Ultra 16GB	10-45	14.16	37.54	14.37	6.76
Sandisk Ultra 32GB	10-90	29.05	87.98	28.21	36.1
Sandisk Extreme 32GB	30-90	68.05	95.73	65.4	37.22
Sandisk Extreme 64GB	30-90	77.38	84.83	82.94	13.32
Samsung Evo Plus 64GB	10-90	35.55	89.97	34.29	36.6
Samsung Evo Plus 256GB	30-90	81.6	82.94	67.74	36.7
SSD Samsung T7 1TB		291.5	370.1	266.34	37.54
Raspberry Pi SSD 256GB		463.47	433.06	374.34	131.07
Raspberry Pi SSD 512GB		361.76	465.57	360.71	128.97



Joonis 4. Graafik tabeli andmete põhjal.

Tulemustest on näha, et juhuslik kirjutamine on vahel kiirem, kui juhuslik lugemine. See anomaalia tuleneb peamiselt kahest asjaolust. Esiteks uuemad andmekandjad kasutavad sageli pseudo-SLC (pSLC) vahemälu või DRAM-vahemälu, mis võimaldab andmed edastada esmalt vahemälusse ja hiljem salvestada lõplikult kaardile, kuid testimis skripti jaoks on kirjutamine tehtud siis, kui andmed jõuavad vahemällu, mille tulemusena mõõdetav kirjutamiskiirus on

tegelikkusest kõrgem. Teiseks võivad fio testi seadistused ja andmekandjate sisemised optimeerimise mehhanismid põhjustada olukorra, kus väiksemad 4 KB plokid koondatakse suurematekskse plokkideks, muutes tulemuse sarnasemaks järjestikulise kirjutamisega. Kuna andmekandjate testimises on vaja teha lihtsustusi võrreldes reaalse olukorraga ning kuna kõigil andmekandjatel on võrdsed tingimused, siis ei mõjuta see oluliselt lõpptulemust.

2.4 Järeldus

Andmekandjate testi tulemustest on näha, et NVMe SSD-d on kiiremad, kui teised andmekandjad nii järjestikuste kui ka juhuslike toimingute puhul. NVMe pooljuhtkettad saavutasid üle 450 MB/s järjestikuste lugemis/kirjutamis operatsioonidega ja juhuslik lugemis/kirjutamis kiirus oli vastavalt 370 MB/s ja 130 MB/s. Kõige paremaks osutus Raspberry Pi ametlik SD-kaart, mis saavutas järjestikusel kirjutamisel 79 MB/s ja juhuslikul kirjutamisel 74 MB/s. Suures pildis saavutasid kallimad tuntud kaubamärkide mikro SD-kaardid samaväärilisi tulemusi Raspberry Pi ametlike mikro SD-kaartidega.

Selleks, et võrrelda andmekandjate tulemusi ka hinnaga loodi andmekandjatele kulutõhusus skoor. Skoor loodi valemiga $\frac{juhusliku\ lugemise\ kiirus \times andmekandja\ maht}{andmekandja\ hind}$. Loodud skoori järgi kõige kulutõhusamaks andmekandjaks oli Raspberry Pi pooljuhtketas suurusega 512GB. Skooridest loodud graafik asub töö lisade sektsioonis. Kokkuvõttes oli mõistlikum töö järgmises etapis kasutada NVMe SSD-sid, sest isegi parimad mikro SD-kaardid olid 3-5 korda aeglasemad, kui NVMe kettad ning NVMe kettad osutusid ka kõige kulutõhusamaks lahenduseks.

3.Tehnilise lahenduse kirjeldus

Andmekandjate analüüsi põhjal tehtud järelduste tulemusena liiguti edasi süsteemi riist- ja tarkvaralise lahenduse realiseerimisega.

3.1 Tehnilise töö sissejuhatus

Töö ehitati üles toimima 2+n masinal ehk projekt vajab vähemalt kahte masinat, et tagada andmete ohutu talletamine. Praktikas ehitati süsteem üles kolme erineva masinaga, et tõestada lahenduse töötamist ka rohkema, kui minimaalse vajaliku kahe masinaga. Töö esimese sammuna paigaldati NVMe ketastele operatsioonisüsteem, tehti vajalikud seadistused ning loodi salvestus ja varundus sektsioonid. Järgmiseks krüpteeriti turvalisuse tagamiseks kõik loodud sektsioonid ning mugava, kuid siiski turvalise ligipääsu saavutamiseks loodi iseseisvale andmekandjale dekrüpteerimisvõti[11]. Krüpteerimise eesmärk seisneb eelkõige selles, et füüsilise juurdepääsu korral ei oleks kõrvalistel isikutel võimalik andmekandjatelt ilma võtmeta tundlikele andmetele ligi pääseda. Sellepärast võivad kõik krüpteeritud sektsioonid turvalisust kaotamata kasutada ühte ja sama dekrüpteerimisvõtit, sest võtit läheb vaja ainult süsteemi käivitamise hetkel. Kui süsteem on juba sisse lülitatud ja sektsioonid on dekrüpteeritud võib võtme eemaldada. Viimaseks konfigureeriti virtuaalne privaatne võrk ning seati üles võrgusalvesti koos andmete varundus süsteemiga.

3.2 Töös kasutatud riistvara ja selle ettevalmistamine

Töös kasutati kolme Raspberry Pi 5te koos 8GB muutmäluga, kahte Active Cooler moodulit, kolme Raspberry Pi M.2 Hat+i, kahte 512GB ja ühte 256GB NVMe pooljuhtketast, kolme Raspberry Pi 27W USB-C toiteplokki, seadistamiseks ühte Raspberry Pi 32GB mikro SD-kaarti ja süsteemi mugavaks kasutamiseks veel ühte andmekandjat. Töö riistvaraline ettevalmistus seisnes lisaks Raspberry Pi tavapärasele käivitamisele ka Raspberry Pi Foundationi juhendi [12] järgi kõigile kolmele seadmele NVMe HAT+ paigaldamisest.

3.3 Operatsioonisüsteemi algne seadistamine ja krüpteerimine

Järgnevad alampeatükid sisaldavad detailselt tehnilisi käske ja seadistusi, mis võivad tavalugejale jääda raskesti mõistetavaks, kui puudub soov süsteemi ise üles seada. Süsteemi

[11] https://akit.cyber.ee/term/2831-dekrupteerimisvoti

lõplik tulemus ja kasutajavaade on asub neljandas peatükis. Tehnilise ülesseadmise läbimiseks on vajalik Raspberry Pi 5 koos M.2 HAT+ lisamooduliga ning Raspberry Pi operatsioonisüsteemiga SD-kaart.

3.3.1 Operatsioonisüsteemi algne seadistamine

Masinasse sisestati graafilise Raspberry Pi operatsioonisüsteemiga SD-kaart ja masin käivitati. Seejärel värskendati ja paigaldati vajalikud programmid.

```
sudo apt update && sudo apt upgrade
sudo apt install cryptsetup cryptsetup-initramfs rsync parted
```

Pooljuhtkettale laeti läbi Raspberry Pi Imageri Raspberry Pi OS Lite versioon, mis hõlmab endas ainult käsurida ja milles ei ole graafilist kasutajaliidest. Suurendati operatsioonisüsteemi sektsiooni ning loodi ext4 formaadis salvestus ja varundus sektsioonid. Operatsioonisüsteemi sektsioon suurendati 40960 mibibaidini ning salvestus ja varundus sektsioonid loodi suurustega 204800 MiB.

3.3.2 Operatsioonisüsteemi krüpteerimine

NVMe kettal olev operatsioonisüsteem kopeeriti hetkel kasutatava SD-kaardi peale, krüpteeriti NVMe operatsioonisüsteemi sektsioon, valiti krüpteeringule parool ja vormindati krüpteeritud sektsioon. Seejärel kopeeriti SD-kaardil olev operatsioonisüsteem tagasi NVMe pooljuhtkettale.

```
sudo mkdir -p /mnt/nvme-os
sudo mount /dev/nvme0n1p2 /mnt/nvme-os
mkdir -p ~/nvme_os_backup
sudo rsync -aAXv /mnt/nvme-os/ ~/nvme_os_backup/
sudo umount /mnt/nvme-os

sudo cryptsetup luksFormat /dev/nvme0n1p2
sudo cryptsetup luksOpen /dev/nvme0n1p2 cryptroot
sudo mkfs.ext4 /dev/mapper/cryptroot

sudo mkdir -p /mnt/oldroot /mnt/newroot
```

```
sudo mount /dev/mapper/cryptroot /mnt/newroot
sudo rsync -aAXv ~/nvme_os_backup/ /mnt/newroot/
```

Valmistati ette chroot-keskkond ühendades uus krüpteeritud operatsioonisüsteem.

```
for dir in proc sys dev run; do
  sudo mount --bind /$dir /mnt/newroot/$dir
done

sudo mkdir -p /mnt/newroot/boot/firmware
sudo mount /dev/nvme0n1p1 /mnt/newroot/boot/firmware
sudo chroot /mnt/newroot
```

Värskendati operatsioonisüsteem ja paigaldati sinna vajalikud programmid.

```
sudo apt update && sudo apt upgrade
sudo apt install cryptsetup cryptsetup-initramfs rsync parted
```

Järgmiseks tehti muudatusi erinevates failides.

```
# Leia järgmise käsuga UUID ja asenda see käsus <...> osaga:
blkid /dev/nvme0n1p2

echo "cryptroot UUID=<nvme0n1p2-uuid> none luks" > /etc/crypttab
```

```
# Leia järgmise käsuga PARTUUID ja asenda see all olevas tekstis <>
osaga ning muuda faili fstab sisu all olevaks tekstiks
blkid /dev/nvme0n1p1

nano /etc/fstab
```

```
proc /proc proc defaults 0 0
```

```
/dev/mapper/cryptroot / ext4 defaults,noatime 0 1
PARTUUID=<nvme0n1p1-partuuid> /boot/firmware vfat defaults 0 2
```

```
# Järgmist käsku on vaja käivitada
sed -i 's/^MODULES=.*/MODULES=most/'
/etc/initramfs-tools/initramfs.conf
```

Loodi initramfsi jaoks haak tehes uus fail ja muudeti fail täitmisfailiks.

```
nano /etc/initramfs-tools/hooks/force-cryptsetup-include
```

```
#!/bin/sh
set -ex
LOGFILE="/root/force-cryptsetup.log"
echo "DEBUG: Starting force-cryptsetup-include hook at $(date)" >>
"$LOGFILE"
echo "DEBUG: DESTDIR is: $DESTDIR" >> "$LOGFILE"

mkdir -p "${DESTDIR}/sbin" "${DESTDIR}/usr/sbin"

if [ -f /sbin/cryptsetup ]; then
  install -D -m 0755 /sbin/cryptsetup "${DESTDIR}/sbin/cryptsetup"
  echo "DEBUG: Installed from /sbin" >> "$LOGFILE"
elif [ -f /usr/sbin/cryptsetup ]; then
  install -D -m 0755 /usr/sbin/cryptsetup
"${DESTDIR}/usr/sbin/cryptsetup"
  echo "DEBUG: Installed from /usr/sbin" >> "$LOGFILE"
else
  echo "ERROR: cryptsetup not found!" >> "$LOGFILE"
  exit 1
fi

echo "DEBUG: Finished copying cryptsetup at $(date)" >> "$LOGFILE"
```

```
chmod +x /etc/initramfs-tools/hooks/force-cryptsetup-include
```

Loodi juurkasutaja parool, uuendati initramfs ja väljuti chroot-keskkonnast.

```
passwd root

update-initramfs -u -k all

exit
```

Tehti muudatused *cmdline.txt* failile.

```
# Leia järgmise käsuga UUID ja asenda see all olevas tekstis <> osaga ning
muuda faili cmdline.txt sisu all olevaks tekstiks
sudo blkid /dev/nvme0n1p2
sudo nano /mnt/newroot/boot/firmware/cmdline.txt
```

```
console=serial0,115200 console=tty1 root=/dev/mapper/cryptroot rw rootwait
cryptdevice=UUID=<nvme0n1p2-UUID>:cryptroot
```

Uus operatsioonisüsteem ühendati lahti, Raspberry Pi masin lülitati välja, eemaldati SD-kaart ja taaskäivitati Raspberry Pi.

```
sudo umount /mnt/newroot/boot/firmware
sudo umount /mnt/newroot/{dev,sys,proc,run}
sudo umount /mnt/newroot
sudo cryptsetup luksClose cryptroot
poweroff
```

Peale seda küsis Raspberry Pi krüpteeritud sektsiooni parooli, mille sisestamisel käivitus Pi uude operatsioonisüsteemi. Raspberry Pi küsis standardseid esimese käivituse küsimusi ja selle käigus loodi operatsioonisüsteemi kasutaja. Peale kasutajaga sisselogimist tehti järgmised seadistused: käsuga *sudo raspi-config*, kus *System Options* valiku alt muudeti hostname, seadistati Pi

automaatselt kasutajasse sisse logima, WiFiga ühendatud Pi puhul ühendati Raspberry Pi ka võrku. Lisaks lubati *Interface Options* valiku alt SSH ühendused.

3.3.3 Krüpteeritud operatsioonisüsteemi automaatne avamine esimesel masinal

Loodi võtme fail ja see lisati krüpteeritud sektsiooni võtmeks.

```
sudo dd if=/dev/urandom of=/root/keyfile.bin bs=512 count=4
sudo chmod 400 /root/keyfile.bin

sudo cryptsetup luksAddKey /dev/nvme0n1p2 /root/keyfile.bin
# Pärast eelmise käsu edastamist küsitakse ka krüpteeritud osa parooli
```

Andmekandja ühendati Raspberry Pi-ga, vormindati, nimetati ümber ning sellele laeti just loodud võti.

```
sudo dd if=/dev/urandom of=/root/keyfile.bin bs=512 count=4
sudo chmod 400 /root/keyfile.bin

# Uue andmekandja tunnuse saab leida järgmise käsuga
lsblk
# Nüüd tuleb andmekandja tunnus asendada <sda1> osaga
sudo mkfs.ext4 /dev/<sda1>
sudo e2label /dev/<sda1>

# Võtme kopeerimine
sudo mkdir /mnt/usb
sudo mount /dev/sda1 /mnt/usb
sudo cp /root/keyfile.bin /mnt/usb/
sudo umount /mnt/usb
```

Loodi skript andmekandja leidmiseks ja muudeti ta täitmisfailiks.

```
sudo nano /lib/cryptsetup/scripts/usb-keyfile.sh
# Kopeeri järgnev osa uude faili
```

```
#!/bin/sh
set -e
KEYFILE_NAME="keyfile.bin"
MOUNTPOINT="/mnt/usbkey"

for device in /dev/disk/by-label/*; do
    LABEL=$(basename "$device")
    if [ "$LABEL" = "RPIKEY" ]; then
        mkdir -p "$MOUNTPOINT"
        if mount "$device" "$MOUNTPOINT" 2>/dev/null; then
            if [ -f "$MOUNTPOINT/$KEYFILE_NAME" ]; then
                cat "$MOUNTPOINT/$KEYFILE_NAME"
                umount "$MOUNTPOINT"
                exit 0
            fi
            umount "$MOUNTPOINT"
        fi
    fi
done
exit 1
```

```
sudo chmod +x /lib/cryptsetup/scripts/usb-keyfile.sh
```

Uuendati */etc/crypttab* fail.

```
# Leia järgmise käsuga krüpteeritud sektsiooni UUID ja asenda see all olevas
tekstis <> osaga ning muuda faili /etc/crypttab sisu all olevaks tekstiks
sudo blkid /dev/nvme0n1p2
```

```
cryptroot UUID=<your-uuid> none
luks,keyscript=/lib/cryptsetup/scripts/usb-keyfile.sh
```

```
sudo nano /etc/crypttab
```

Seejärel värskendati *initramfs*.

```
sudo update-initramfs -u
```

3.3.4 Krüpteeritud operatsioonisüsteemi automaatne avamine teisel masinal

Teise masina korral binaarset võtit enam ei loodud, vaid kopeeriti olemasolev võti esmalt Raspberry Pi-le ja lisati see võtmeks.

```
sudo mkdir -p /mnt/sdkey
sudo mount /dev/sda1 /mnt/sdkey
sudo cp /mnt/sdkey/keyfile.bin ~/
sudo umount /mnt/sdkey
sudo mv keyfile.bin /root/keyfile.bin
sudo chmod 400 /root/keyfile.bin
sudo cryptsetup luksAddKey /dev/nvme0n1p2 /root/keyfile.bin
```

Tehti andmekandja leidmis skript.

```
sudo nano /lib/cryptsetup/scripts/usb-keyfile.sh
# Kopeeri järgnev osa uude faili
```

```
#!/bin/sh
set -e
KEYFILE_NAME="keyfile.bin"
MOUNTPOINT="/mnt/usbkey"

for device in /dev/disk/by-label/*; do
    LABEL=$(basename "$device")
    if [ "$LABEL" = "RPIKEY" ]; then
        mkdir -p "$MOUNTPOINT"
        if mount "$device" "$MOUNTPOINT" 2>/dev/null; then
            if [ -f "$MOUNTPOINT/$KEYFILE_NAME" ]; then
                cat "$MOUNTPOINT/$KEYFILE_NAME"
                umount "$MOUNTPOINT"
                exit 0
            fi
            umount "$MOUNTPOINT"
        fi
    fi
done
exit 1
```

Skripti fail muudeti täitmisfailiks.

```
sudo chmod +x /lib/cryptsetup/scripts/usb-keyfile.sh
```

Uuendati *crypttab* faili, lisades sinna "keyscript=/lib/cryptsetup/scripts/usb-keyfile.sh" nii, et kõik jäi ühele reale.

```
sudo nano /etc/crypttab
```

```
cryptroot UUID=<your-uuid> none
luks,keyscript=/lib/cryptsetup/scripts/usb-keyfile.sh
```

Uuendati *initramfs* ja taaskäivitati masin, kuid nüüd automaatselt krüpteeritud operatsioonisüsteemi avamisega.

```
sudo update-initramfs -u
sudo reboot
```

Töö käigus avastati, et kui mingi etapp vahele jätta ja hiljem parandada, kuid *initramfs* oli juba uuendatud, siis tuleb *initramfs* täielikult uuesti genereerida, et avamine töötaks.

```
sudo update-initramfs -c -k "$(uname -r)"
sudo cp /boot/initrd.img-$(uname -r) /boot/firmware/initramfs8
sudo reboot
```

3.3.5 Salvestus ja varundamise sektsiooni krüpteerimine ja selle automaatne avamine

Krüpteeriti mõlemad sektsioonid.

```
# Salvestus sektsioon
sudo cryptsetup luksFormat /dev/nvme0n1p3
sudo cryptsetup luksOpen /dev/nvme0n1p3 storage_crypt
sudo mkfs.ext4 /dev/mapper/storage_crypt

# Varundamise sektsioon
```

```
sudo cryptsetup luksFormat /dev/nvme0n1p4
sudo cryptsetup luksOpen /dev/nvme0n1p4 backup_crypt
sudo mkfs.ext4 /dev/mapper/backup_crypt
```

Loodi uus avamis skript, mis käivitub pärast operatsioonisüsteemi tööle hakkamist.

```
sudo nano /usr/local/sbin/unlock-volumes.sh
# Muuda järgmises skriptis <Salvestus-sektsiooni-UUID> ja
<Varundus-sektsiooni-UUID> sektsioonide päris UUIDdeks
```

```
#!/bin/bash

# Võtmefaili nimi
KEYFILE_NAME="keyfile.bin"

# Kuhu andmekandja ühendatud on
MOUNT_DIR="/mnt/keysrc"
KEYFILE_PATH="$MOUNT_DIR/$KEYFILE_NAME"

# Sektsioonide UUIDd
STORAGE_UUID="<Salvestus-sektsiooni-UUID>"
BACKUP_UUID="<Varundus-sektsiooni-UUID>"

# Vastendite nimed
STORAGE_NAME="crypt-storage"
BACKUP_NAME="crypt-backup"

mkdir -p "$MOUNT_DIR"

# Ürita ühendada andmekandija(nii sd kaart kui ka mälupulk)
echo "[INFO] Trying to remount key device..."
for DEV in /dev/mmcblk0 /dev/mmcblk0p1 /dev/sd*1 /dev/mmcblk*p1; do
    mount "$DEV" "$MOUNT_DIR" 2>/dev/null && break
done

# Kontrolli, kas võtmefail on olemas
if [ ! -f "$KEYFILE_PATH" ]; then
    echo "[ERROR] Keyfile not found at $KEYFILE_PATH. Skipping unlock."
    exit 1
else
    echo "[INFO] Keyfile found: $KEYFILE_PATH"
```

```
fi

# Avamis funktsioon
unlock_partition() {
    local UUID=$1
    local NAME=$2

    if [ -e /dev/mapper/$NAME ]; then
        echo "[INFO] $NAME already unlocked, skipping."
        return
    fi

    echo "[INFO] Unlocking $NAME using keyfile..."
    cryptsetup luksOpen UUID=$UUID $NAME --key-file "$KEYFILE_PATH"
}

# Ava krüpteeritud sektsioonid
unlock_partition "$STORAGE_UUID" "$STORAGE_NAME"
unlock_partition "$BACKUP_UUID" "$BACKUP_NAME"

# Ühenda avatud sektsioonid
[ -e /dev/mapper/crypt-storage ] && mount -t auto /dev/mapper/crypt-storage
/srv/storage
[ -e /dev/mapper/crypt-backup ] && mount -t auto /dev/mapper/crypt-backup
/srv/backup #

# Ühenda lahti võtmefail
echo "[INFO] Unmounting key source..."
umount "$MOUNT_DIR" 2>/dev/null
```

```
# Muuda loodud skript täitimisvalmiks
sudo chmod +x /usr/local/sbin/unlock-volumes.sh
```

Loodi *systemd* teenus, et skript käivituks pärast operatsioonisüsteemi käivitumist.

```
sudo nano /etc/systemd/system/unlock-volumes.service
# Muuda faili unlock-volumes sisuks järgmine tekst
```

```
[Unit]
Description=Post-boot unlock for encrypted NAS partitions
```

```
After=multi-user.target

[Service]
ExecStart=/usr/local/sbin/unlock-volumes.sh
Type=oneshot
RemainAfterExit=yes

[Install]
WantedBy=multi-user.target
```

Käivitati loodud teenus.

```
sudo systemctl daemon-reexec
sudo systemctl enable unlock-volumes.service
```

Järgmiseks lisati sama võtme fail, millega avati operatsioonisüsteem ka salvestus ja varundus sektsioonide võtmeteks.

```
sudo cryptsetup luksAddKey /dev/nvme0n1p3 /root/keyfile.bin
sudo cryptsetup luksAddKey /dev/nvme0n1p4 /root/keyfile.bin
```

Scp käsuga lisati personaalne id_ed25519 võti ja tehti seadistusi, et SSHga ühendamine oleks turvalisem.

```
chmod 700 ~/.ssh
chmod 600 ~/.ssh/id_ed25519
chmod 644 ~/.ssh/id_ed25519.pub
sudo cat ~/.ssh/id_ed25519.pub | sudo tee -a ~/.ssh/authorized_keys > /dev/null
sudo chmod 600 ~/.ssh/authorized_keys
```

```
sudo nano /etc/ssh/sshd_config
# Muuda faili sshd_config sisu nii, et muutujad oleksid samade väärtustega,
nagu ette näidatud tekstis
```

```
PasswordAuthentication no
PermitRootLogin no
ChallengeResponseAuthentication no
UsePAM no
PubkeyAuthentication yes
PermitEmptyPasswords no
AuthenticationMethods publickey
```

```
sudo systemctl restart ssh
```

3.4 Virtuaalne privaatne võrk

3.4.1 Ruuterit hõlmavad seadistused

Virtuaalse privaatvõrgu ehk VPNi ülesseadmiseks seadistati esmalt VPN serveri seade. Selle jaoks oli vaja seadistada ruuter, mille külge Raspberry Pi läbi ethernet kaabli ühendatud on. Esmalt võeti ühendust interneti teenusepakkujaga, et saada ruuterile staatiline IP-aadress. Süsteemi on võimalik üles seada ka ilma staatilise IP-aadressita kasutades dünaamilise DNSi teenust. Raspberry Pi lokaalset IP-aadressi üritati teha staatiliseks läbi ruuteri seadete, kuid kasutatud ruuteril sellist võimalust ei olnud ning seda teostati hiljem Raspberry Pi enda seadetes. Järgmiseks tehti ruuteris IPv4 pordi edastus reegel. Kasutati UDP protokolli WAN ja LAN pordist 5180 Raspberry Pi IP-aadressile.

3.4.2 Open Media Vault paigaldamine

Enne virtuaalse privaatvõrgu loomist paigaldati OpenMediaVault. Kuigi OpenMediaVault on eraldiseisev Linuxi distributsioon, saab seda paigaldada ka juba olemasolevale Debiani põhisele operatsioonisüsteemile. Kuna OpenMediaVaulti paigaldamine teeb olemasolevas süsteemis palju muudatusi, tasub distributsioon paigaldada enne järgmiste seadistuse elluviimist. Põhjalikum ülevaade Open Media Vaultist tuleb peatükis 3.5 NAS süsteem. OpenMediaVault paigaldati järgmise käsuga.

```
wget -4 -O -
https://github.com/OpenMediaVault-Plugin-Developers/installScript/raw/master/install | sudo bash -s -- -4
sudo apt install openmediavault
```

3.4.3 Wi-Fi taasseadistamine

Osa masinatest oli planeeritud läbi Wi-Fi liidese võrku saama. Pärast OpenMediaVaulti paigaldamist kaob Wi-F liides ära ja see on vaja uuesti paigaldada.

Loodi WPA_supplicant konfiguratsioonifail.

```
sudo nano /etc/wpa_supplicant/wpa_supplicant-wlan0.conf
```

```
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1
country=EE # (your country code)

network={
    ssid="YourWiFiName"
    psk="YourWiFiPassword"  #kui wifi võrgul pole parooli, siis tuleks psk=...
asendada key_mgmt=NONE reaga
}
```

Loodi võrgu konfiguratsiooni fail.

```
sudo mkdir -p /etc/systemd/network
sudo nano /etc/systemd/network/25-wlan.network
```

```
[Match]
Name=wlan0

[Network]
DHCP=yes
```

Loodi lüli WPA konfiguratsiooni faili jaoks.

```
sudo ln -sf /etc/wpa_supplicant/wpa_supplicant-wlan0.conf
/etc/wpa_supplicant/wpa_supplicant.conf
```

Viimaks teenus aktiveeriti.

```
sudo systemctl enable wpa_supplicant@wlan0
sudo systemctl start wpa_supplicant@wlan0
sudo systemctl restart systemd-networkd
```

3.4.4 VPN serveri seadmel

Tehti värskendus ja paigaldati vajalikud programmid.

```
sudo apt update
sudo apt install wireguard -y
sudo apt install resolvconf -y
sudo apt install iptables -y
```

Loodi võtmepaarid kõikide virtuaalsesse võrku ühenduvate seadmete jaoks ja lisaks veel üks. Võtmepaare loodi järgmise käsuga.

```
wg genkey | tee privatekey | wg pubkey > publickey
sudo nano /etc/systemd/system/unlock-volumes.service
# Loodud võtmete väärtusi saab näha järgmiste käskudega
sudo cat pubkey
sudo cat privatekey
```

Peale seda valiti VPN liidese nimi ja loodi VPN konfiguratsiooni fail järgmise malli alusel, kus network-interface-name on asendatud interneti liidese nimega ja võtme muutujad on asendatud eelnevalt genereeritud võtmete väärtustega. Eeljagatud võtme kasutamine pole kohustuslik, kuid see lisab turvalisust. Kui Raspberry Pi on ühendatud läbi ethernet kaabli, siis on see nimi tavapäraselt, kas eth0-või end0. Faili loomine tehti järgmise käsuga.

```
sudo nano /etc/wireguard/<interface-name>.conf
```

Mall:

```
[Interface]
PrivateKey = <serveri-salajane-alusvõti>
Address = 10.46.17.1/24
```

```
ListenPort = 51820
PostUp = iptables -A FORWARD -i <network-interface-name> -j ACCEPT; iptables -t
nat -A POSTROUTING -o <network-interface-name> -j MASQUERADE
PostDown = iptables -D FORWARD -i <network-interface-name> -j ACCEPT; iptables
-t nat -D POSTROUTING -o <network-interface-name> -j MASQUERADE

#peer1
[Peer]
PublicKey = <klient1-avalik-alusvõti>
PresharedKey = <eeljagatud-võti>
AllowedIPs = 10.46.17.2/32

#peer2
[Peer]
PublicKey = <klient2-avalik-alusvõti>
PresharedKey = <eeljagatud-võti>
AllowedIPs = 10.46.17.3/32

#peer3
[Peer]
PublicKey = <klient3-avalik-alusvõti>
PresharedKey = <eeljagatud-võti>
AllowedIPs = 10.46.17.4/32
```

Määrati failiõigused:

```
sudo chown root:root /etc/wireguard/<interface-name>.conf
sudo chmod 600 /etc/wireguard/<interface-name>.conf
```

VPN teenus seati käivituma automaatselt pärast Raspberry Pi internetiga ühenduse saamist.

```
sudo systemctl start wg-quick@<interface-name>
sudo systemctl enable wg-quick@<interface-name>
```

```
sudo systemctl edit wg-quick@<interface-name>
# Sisesta järgnev konfiguratsioon
```

```
[Unit]
After=network-online.target
Wants=network-online.target
```

```
sudo systemctl daemon-reexec
sudo systemctl restart wg-quick@<interface-name>
```

Seadistuse korrektset käitumist kontrolliti järgneva käsuga.

```
sudo wg show <interface-name>
```

3.4.5 VPN kliendi seadmel

Tehti värskendus ja paigaldati vajalikud programmid.

```
sudo apt update
sudo apt install wireguard -y
sudo apt install resolvconf -y
sudo apt install iptables -y
```

VPN liidesele valiti nimi ja loodi VPN konfiguratsiooni fail järgmise malli alusel, kus võtme muutujad on asendatud eelnevalt genereeritud võtmete väärtustega. Faili loomine tehti järgmise käsuga.

```
sudo nano /etc/wireguard/<interface-name>.conf
```

Mall:

```
[Interface]
PrivateKey = <kliendi-salajane-alusvõti>
Address = <kliendile-määratud-IP-aadress>/24
ListenPort = 51820

[Peer]
PublicKey = <serveri-avalik-alusvõti>
```

```
PresharedKey = <eeljagatud-võti>
Endpoint = <serveri-globaalne-IP-aadress>:51820
AllowedIPs = 0.0.0.0/0, ::/0
PersistentKeepalive = 25
```

Seejärel määrati failiõigused järgmiselt:

```
sudo chown root:root /etc/wireguard/<interface-name>.conf
sudo chmod 600 /etc/wireguard/<interface-name>.conf
```

Viimaks seati VPN teenus käivituma automaatselt pärast Raspberry Pi internetiga ühenduse saamist.

```
sudo systemctl start wg-quick@<interface-name>
sudo systemctl enable wg-quick@<interface-name>
```

```
sudo systemctl edit wg-quick@<interface-name>
# Sisesta järgnev konfiguratsioon
```

```
[Unit]
After=network-online.target
Wants=network-online.target
```

```
sudo systemctl daemon-reexec
sudo systemctl restart wg-quick@<interface-name>
```

Seadistuse korrektset käitumist kontrolliti järgneva käsuga, kus <interface-name> asendati päris liidese nimega.

```
sudo wg show <interface-name>
```

3.5 NAS süsteem

NAS süsteemi seadistamine tehti läbi graafilise liidese. Graafilisse liidesesse saab läbi veebibrauseri. Töös ühendati teine arvuti virtuaalsesse võrku ning mindi Raspberry Pi IP-aadressile. Esimese vaatena on ees sisselogimis aken. Vaikimisi on kasutajatunnus admin ja parool openmediavault. Pärast sisselogimist vahetati vaikimisi parool ära. Peale seda uuendati kogu süsteem järgmiselt. *System -> Update Management -> Updates -> Install Updates -> Confirm -> Install updates.*
Seejärel paigaldati süsteemi OpenMediaVault Extrad järgmise käsuga.

```
wget -O -
https://github.com/OpenMediaVault-Plugin-Developers/packages/raw/master/install
| sudo bash
```

Kliendi masinatel on läbi virtuaalse privaatvõrgu ligipääs ainult iPv4 protokollile ja aja kokkuhoidmise eesmärkidel kasutati klient seadmetel lisaks ka *--inet4-only* lippu.

```
wget --inet4-only -O -
https://github.com/OpenMediaVault-Plugin-Developers/packages/raw/master/install
| sudo bash
```

Peale seda paigaldati OMV süsteemi jaoks vajalikud pluginad, milleks olid *openmediavault-luksencryption, openmediavault-compose* ja *omv extras* sektsiooni alt võimaldati *Docker repo*.

OpenMediaVault graafilises liideses seadistuste tegemise järel avanes hüpikaken, kus tuli kinnitada, et valitud seadistused jõustuksid. Seejärel ühendati krüpteeritud sektsioonid järgmiselt. *Storage -> File Systems -> Mount existing file system ->* valiti õige sektsioon. Seda tehti nii salvestus sektsioonile kui ka varundus sektsioonile. Järgmiseks seadistati kaustad andmete hoiustamiseks. *Storage -> Shared Folders -> Create*. Salvestus sektsiooni jaoks tehti järgmised valikud: *Name* - storage*, File system* - /dev/dm-1*, Permissions* - Administrator:read/write, Users: read/write, Others: no access. Varundus sektsiooni jaoks tehti järgmised valikud: *Name* - backup, *File system* - /dev/dm-2, *Permissions* - Administrator:read/write, Users: read, Others: no access.

Seejärel seadistati salvestus sektsiooni ja varundus sektsiooni kaustad läbi võrgu ligipääsevateks. Salvestus sektsiooni jaoks tehtud valikud: *Services -> SMB/CIFS -> Settings -> Enabled*. *Services -> SMB/CIFS -> Shares -> Create* ja järgmised valikud: *Shared Folder* - storage, *Public* - no, *Save*. Varundus kasuta jaoks: *Services -> SMB/CIFS -> Shares -> Create* ja järgmised valikud: *Shared Folder* - backup, *Public* - no, *Read-only*, *Save*. Et andmetele ligipääs oleks tagatud seadistati kasutaja sätteid. *Users* -> "kasutaja" -> Edit: valiti parool ja *Save*. *Users* -> "kasutaja" -> *Shared folder permissions* ja järgmised valikud: *backup* - Read-only, *storage* - Read/Write ja *Save*. Sellega saavutati andmete hajus talletamine läbi võrgu. Seadistused tehti kolme masina peal ja arvutid jagasid andmeid enese ja teiste masinate vahel edukalt.

3.6 Sünkroniseerimine

Sünkroniseerimiseks kasutati OpenMediaVaultis Dockeri pluginat ja Syncthingi. Vajalikud pluginad paigaldati NAS süsteemi seadistamisel. Dockeri jaoks loodi jagatud kaust. *Storage -> Shared Folders -> Create* ja valiti järgmised valikud: *Name* - compose, *File system* - "operatsiooni süsteemi sektsioon", *Permissions* - Administrator: read/write, Users: no access, Others: no access ja *Save*. *Service -> Compose -> Settings* ja järgmised valikud: *Shared folder* - compose, *Enable Docker repo* ja *Save*.

Seejärel loodi Dockeri fail järgmiselt: *Services -> Compose -> Files -> Add*. Uue faili nimeks pandi Syncthing ja faili sisuks kopeeriti järgmine lõik, kus failitee asendati kasutusel oleva salvestus sektsiooni failiteega. Failitee leiti käsuga *ls /srv/* ja kausta sisu, ehk valiti kaustatee, milles olev kaust sisaldas storage kausta.

```
version: "2.1"
services:
  syncthing:
    image: lscr.io/linuxserver/syncthing:latest
    container_name: syncthing
    environment:
      - PUID=1000
      - PGID=100
    volumes:
      -<failitee, mis algab /srv/dev-disk-by-uuid->/storage:/var/syncthing
      - /home/riki/.config/syncthing:/config
```

```
    ports:
      - 8384:8384        # Web UI
      - 22000:22000      # Sync port
      - 21027:21027/udp  # Discovery
    restart: unless-stopped
```

Peale faili loomist käivitati protsess järgmiselt: *Services -> Compose -> Files -> syncthing -> Up*. Korrektset töötamist kontrolliti minnes veebibrauseris lehele http://<lokaalne-IP-aadress>:8384/. Seadistusi tehti kõigi kolme masina peal.

3.6.1 Syncthing

Lehele jõudes avanes Syncthingi hüpikaken, mis soovis luba andmete jagamiseks Syncthingiga, millest keelduti. Peale seda loodi graafilises liideses seadistamiseks kasutajatunnus ja parool. Seejärel lisati VPN server masinasse teise kahe masina ID. ID oli leitav minnes rippmenüüsse ja valides *Show ID*. Läbi *Add Remote Devices* valiku sisestades masinate ID ja masinate nimi. Teiste masinate GUI lehelt valiti add. Viimaks ühendati teised kaks kliendi masinat ka omavahel kasutades eelkirjeldatud meetodit.

Seejärel tehti server masinas uus kaust tehes järgmised valikuid:
Add Folder -> Folder Label - storage*, Folder Id* - storage-sync*, Folder Path* - /var/syncthing*, -> Sharing* ja valiti mõlemad ühendatud masinad ning *Save*. Peale seda tulid teiste masinate GUI liidestes hüpikaknad, mis küsisid luba kausta jagamiseks. Seal valiti jah ning muudeti *Folder Pathi* väärtuseks var/syncthing.

3.7 Varundamine

Varundamiseks kasutati Borg ja Borgmatic programme. Need paigaldati järgmise käsuga.

```
sudo apt install -y borgbackup borgmatic
```

Loodi Borgi jaoks kaust ja algatati Borgi hoidla järgmiste käskudega vahetades <> osa päris faili tee vastu.

```
sudo mkdir -p /srv/<dev-disk-by-uuid-XXXX>/backup/borg-repo
sudo borg init --encryption=none /srv/<dev-disk-by-uuid-XXXX>/backup/borg-repo
```

Loodi Borgi konfiguratsiooni kaust ja fail nii, et faili sisus <> vahelised osad vahetati päris failiteede vastu.

```
sudo mkdir -p /etc/borgmatic
sudo nano /etc/borgmatic/config.yaml
```

```
location:
  source_directories:
    - /srv/<dev-disk-by-uuid-XXXX>/storage
  repositories:
    - /srv/<dev-disk-by-uuid-YYYY>/backup/borg-repo

storage:
  compression: auto,zstd

retention:
  keep_daily: 30
  keep_hourly: 0
  keep_weekly: 0
  keep_monthly: 0
  keep_yearly: 0

consistency:
  checks:
    - repository
    - archives

hooks:
#  before_backup:
#    - |
#      echo "Checking if backup is needed..."
repo="/srv/dev-disk-by-uuid-92e62e31-3303-41d9-a36f-e05f9f6d25e8/backup/borg-re
po"
#
source="/srv/dev-disk-by-uuid-42ca0e84-7bac-45ef-9415-2e2fbf4eeb7f/Storage"
#      borg create --dry-run "$repo"::check-$(date +%s) "$source"
#      if [ $? -eq 0 ]; then
```

```
#         echo "Changes detected -- backup will proceed."
#       else
#         echo "No changes -- skipping backup."
#         exit 75
#       fi
```

Seejärel tehti andmetest esimene varundus.

```
sudo borgmatic --verbosity 1
```

Peale seda muudeti konfiguratsiooni faili sisestades välja kommenteeritud koodi tagasi sisse.

```
location:
  source_directories:
    - /srv/<dev-disk-by-uuid-XXXX>/storage
  repositories:
    - /srv/<dev-disk-by-uuid-YYYY>/backup/borg-repo

storage:
  compression: auto,zstd

retention:
  keep_daily: 30
  keep_hourly: 0
  keep_weekly: 0
  keep_monthly: 0
  keep_yearly: 0

consistency:
  checks:
    - repository
    - archives

hooks:
 before_backup:
    - |
      echo "Checking if backup is needed..."
repo="/srv/dev-disk-by-uuid-92e62e31-3303-41d9-a36f-e05f9f6d25e8/backup/borg-re
po"
```

```
source="/srv/dev-disk-by-uuid-42ca0e84-7bac-45ef-9415-2e2fbf4eeb7f/Storage"
      borg create --dry-run "$repo"::check-$(date +%s) "$source"
      if [ $? -eq 0 ]; then
        echo "Changes detected -- backup will proceed."
      else
        echo "No changes -- skipping backup."
        exit 75
      fi
```

Seejärel loodi teenus ja taimer, et varundusi tehtaks iga minuti järel, kui andmetesse on tulnud muudatusi.

```
sudo nano /etc/systemd/system/borgmatic-minutely.service
```

```
[Unit]
Description=Run Borgmatic backup

[Service]
Type=oneshot
ExecStart=/usr/bin/borgmatic --verbosity 1
```

```
sudo nano /etc/systemd/system/borgmatic-minutely.timer
```

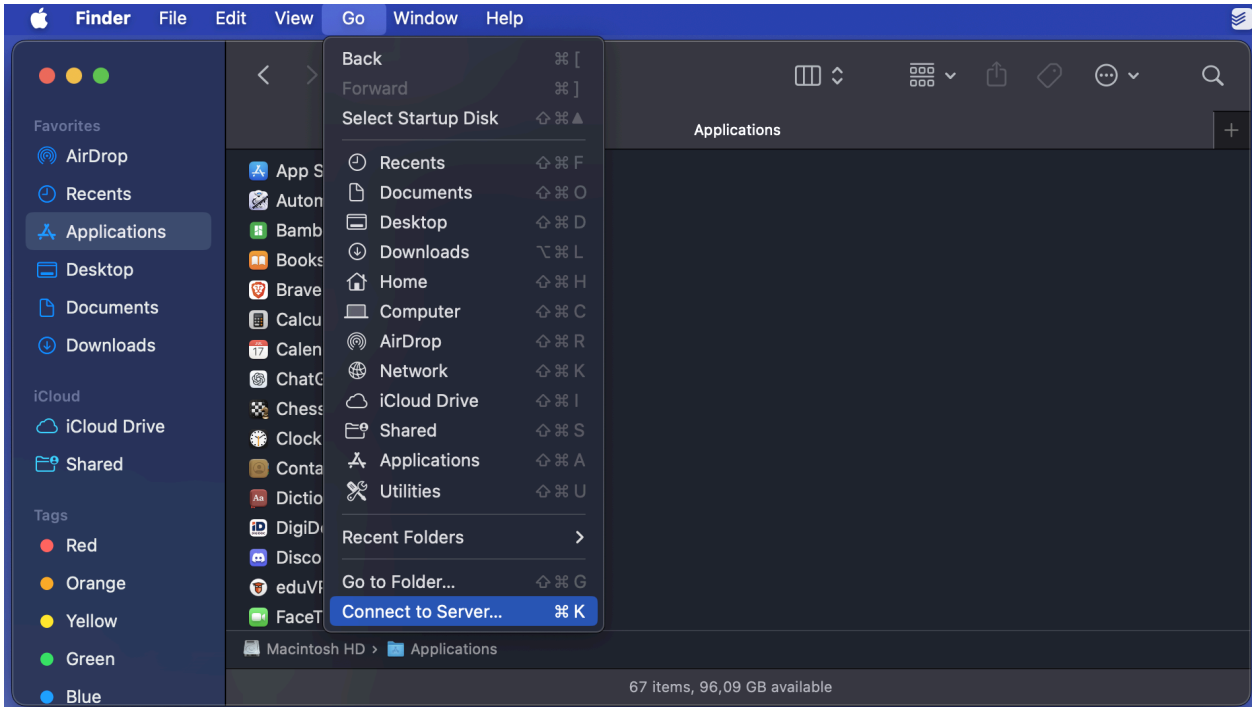
```
[Unit]
Description=Run Borgmatic every 1 minute

[Timer]
OnBootSec=1min
OnUnitActiveSec=1min
Persistent=true

[Install]
WantedBy=timers.target
```

```
sudo systemctl daemon-reexec
sudo systemctl daemon-reload
sudo systemctl enable --now borgmatic-minutely.timer
```

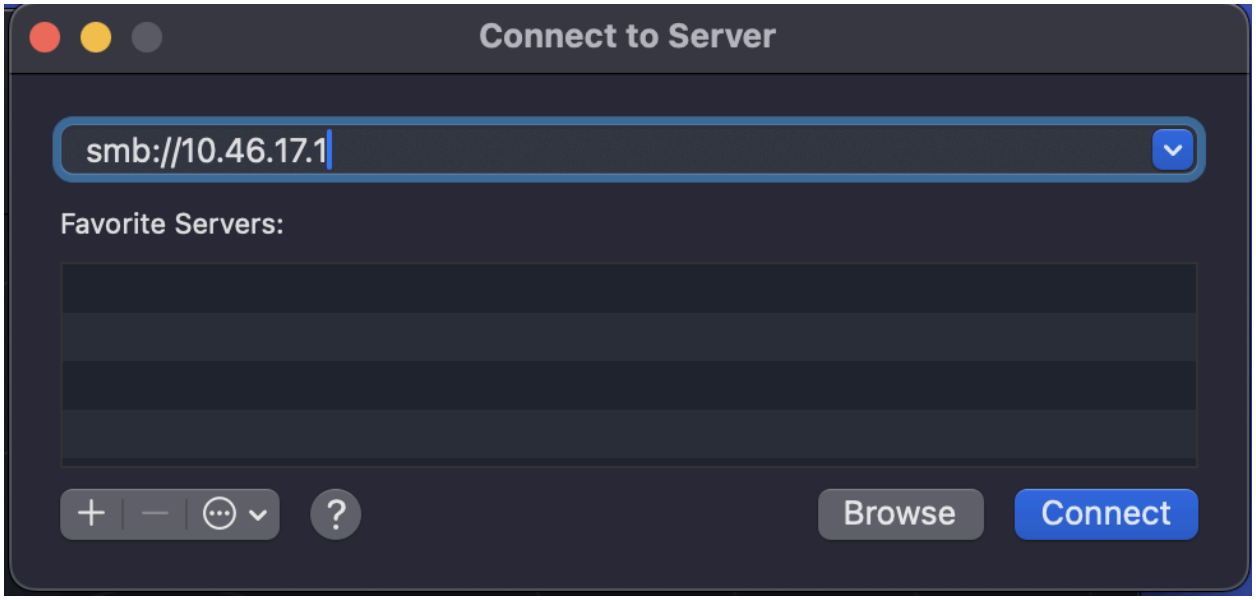
4.Tulemus ja testimine

4.1 Tulemus

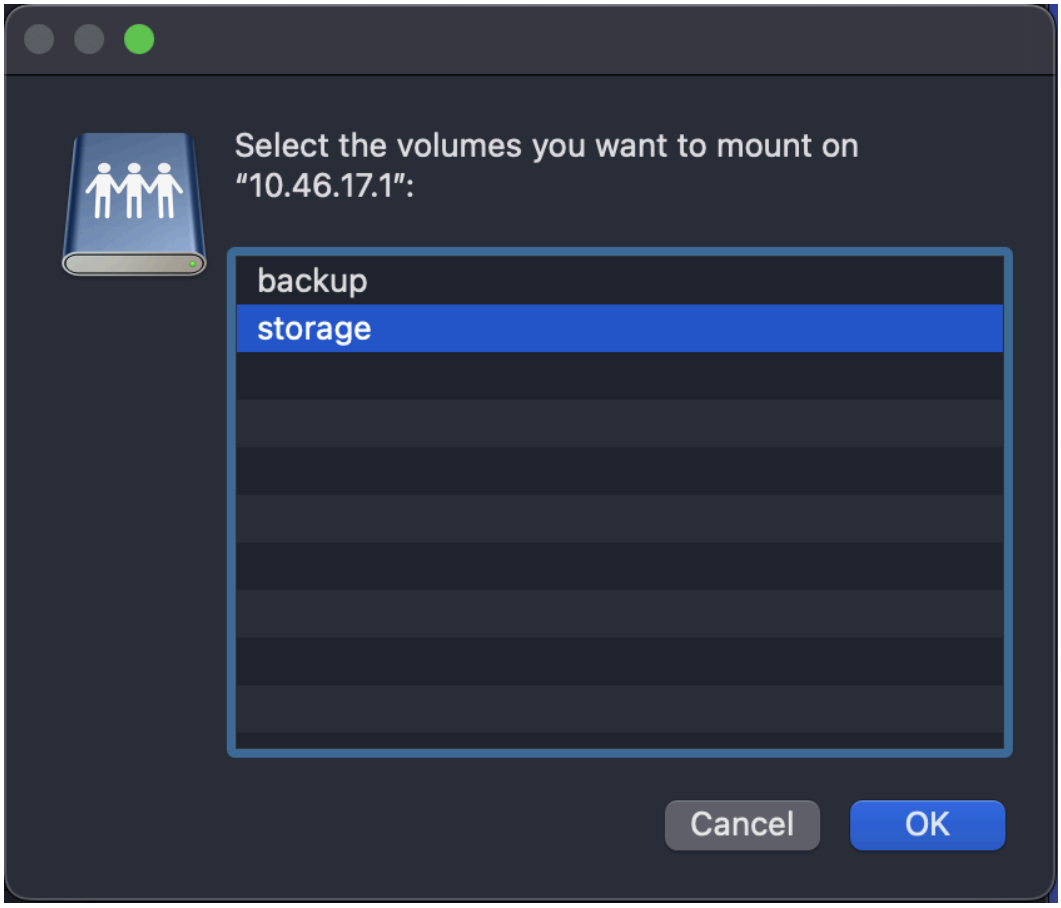
Tulemuseks saavutati andmehoidla, mille turvalisus on tagatud läbi krüpteeritud operatsioonisüsteemi, krüpteeritud salvestus ja varundus sektsioonide ning virtuaalse privaatvõrgu. Pärast süsteemi loomist sisaldab kasutaja kogemus endast peamiselt graafilise failihaldus programmi käsitlemist, kuid andmete taastamiseks ja versiooni kontrolli kasutamiseks on vaja kasutada ka käsuviipa. Failidele pääseti ligi läbi SMB võrguprotokolli nii Windowsis, Linuxis, MacOSis, iOSis kui ka Androidis. Selleks ühendati virtuaalses võrgus seadme failihaldus keskkonnas sisestada mõne Raspberry Pi lokaalne võrguaadress. Näide keskkonda saamisest on järgmistel joonistel kasutades MacOSi, mis on virtuaalsesse privaatvõrku ühendatud.



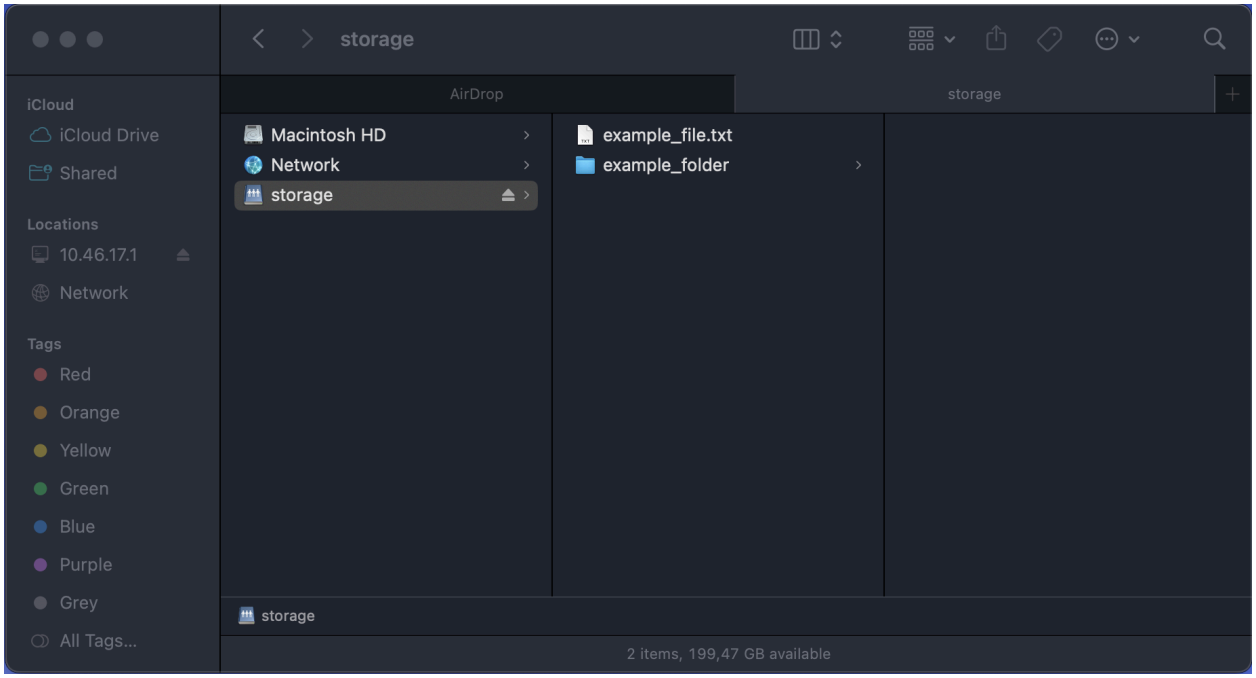
Joonis 5. Pilt failihaldurist salvestusvõrgu süsteemi ühedamisel.



Joonis 6. Pilt failihaldurist salvestusvõrgu süsteemi ühedamisel.



Joonis 7. Pilt failihaldurist salvestusvõrgu süsteemi ühedamisel.



Joonis 8. Pilt failihaldurist, kus on süsteemile ligi pääsetud.

Varundatud failide taastamiseks peab kasutama käsuviipa, kuid selle lihtsustamiseks loodi kaks skripti. Mõlemas skriptis asendati nurksulgude vahelised osad päris failiteedega. Skriptid lisati töö lisade sektsiooni. Vaata lisa IV.

4.2 Hind

Kõik seadmed soetati internetipoest ThePiHut. ThePiHut on Raspberry Pi Foundationi poolt heakskiidetud edasimüüja. Tabel 2 toob välja süsteemi komponentide kogused ja hinnad, mille alusel tuli lõpphinnaks 501 eurot ja 37 senti.

Tabel 2. Süsteemi komponentide hinna näidistabel.

Seade	**Kogus**	**Hind (tk)**
Raspberry Pi 5 8GB	3	€89,97
Raspberry Pi SSD Kit for Raspberry Pi 5 512GB	2	€62,10
Raspberry Pi SSD Kit for Raspberry Pi 5 256GB	1	€45,16
Raspberry Pi 27W USB-C Power Supply	3	€13,41

Active Cooler for Raspberry Pi 5	2	€5,29
Raspberry Pi Micro SD Card with RPi OS Pre-Installed	1	€11,29

Lisaks süsteemi soetamise ühekordsele kulule kaasnevad süsteemi kasutamisega ka püsikulud. Püsikuludeks on internetiühendus ja seadmete elektritarbimine. Kuna internetiühendus on tänapäeval laialt levinud ja selle maksumus sõltub individuaalsetest lepingutest, keskendutakse siin elektrikuludele.

Raspberry Pi 5 koos M.2 HAT+ lisamooduliga ja NVMe SSD-ga võib maksimaalse koormuse korral tarbida kuni 12 vatti elektrienergiat [17]. Eeldades, et seade töötab pidevalt ööpäevaringselt täisvõimsusel kujuneb ühe seadme kuine elektritarbimine järgmiselt:
$12W \times 24\,h \times 30\,päeva = 8.64\,kWh$ kuus. Kuna süsteem hõlmab kolme seadet, siis tuleb arvestada kõigi kolme seadme elektritarbimist, mis on kokku $8.64\,kWh \times 3 = 25.92\,kWh$ kuus.

Aastal 2024 keskmiseks elektri hinnaks Eestis oli 8.73 senti kilovatt-tunni kohta [16], millele lisandub võrgutasu hetkeseisuga keskmiselt 4.4 senti kilovatt-tunni kohta [17], ehk kokku on eeldatav igakuine kulu $25.92\,kWh \times (0.0873\,€/kWh + 0.044\,€/kWh) = 3.40\,€$. Võrdluseks, Google Drive pilveteenuse hind 200 gigabaiti kohta on 2.99 eurot kuus [18], iCloudi pilveteenuse hind 200 gigabaidi kohta 3.59 USA dollarit kuus [19], ning Microsofti pilveteenuse hind, mis hõlmab endast 100 gigabaiti pilve salvestusruumi ja 100 gigabaiti postkasti salvestusruumi on 1.99 USA dollarit kuus [20].

4.2 Lõplik testimine

Pärast süsteemi lõplikku ülesseadmist viidi läbi katsed, mille eesmärgiks oli hinnata lahenduse töökindlust, rikete taluvust ning taastumisvõimet erinevates ootamatutes olukordades. Katsetes simuleeriti erinevaid tõrkeolukordi, sealhulgas toitekatkestusi, salvestusruumi täitumist, ühenduste katkemist ja kasutajapoolseid tegevusi nagu failide kustutamine.

Toitekatkestuse korral lakkas serverseadmes kogu süsteem ajutiselt töötamast. Kui seadmega oli ühendatud krüpteeringut avav andmekandja, taastati töö normaalne kulg automaatselt pärast

seadme taaskäivitust. Kui katkestus tekitati kliendiseadmes ning samal ajal tehti muudatusi teistes seadmetes, siis kliendi toite ja krüptovõtmega andmekandja ühendamisel taastati seadme töö ja muudatused sünkroniseeriti edukalt ilma tõrgeteta.

Failide tahtlik kustutamine ja nende taastamine versioonihalduse (Borg/Borgmatic) abil toimis korrektselt nii server- kui ka kliendiseadmetes. Katsed kustutada varukoopiad ebaõnnestusid, kuna vastavad juurdepääsuõigused puudusid.

Süsteemi täielikul seiskumisel, kui kõik seadmed olid korraga välja lülitatud, jäi andmesalvesti kättesaamatuks seni, kuni serveri seade oli taas võrgus, kuna ilma serveri seadmeta pole ühelgi seadmel ligipääsu virtuaalsesse privaatvõrku. Samuti katsetati olukorda, kus Raspberry Pi küljest eemaldati töötava süsteemi ajal NVMe-ketas ning seejärel ühendati see uuesti. Seade ei suutnud enam käivituda ning logidest ilmnes, et seade ei osanud enam krüpteeritud operatsioonisüsteemi avada.

Lisaks katsetati olukordi, kus salvestus- või varundussektsiooni mälu sai täis. Kui salvestussektsioon täitus, ei olnud võimalik andmeid enam salvestada ning süsteem väljastas veateate nii käsurea liidese kaudu kui ka võrgusalvesti graafilistes kasutajaliidestes. Seevastu varundus sektsiooni täitumisel katkes varundamise protsess ilma igasuguse hoiatuse või teavituseta.

Kokkuvõttes näitasid testid, et süsteem suudab edukalt toime tulla tavapäraste rikete ja katkestustega, näiteks seadmete taaskäivituse või failide kustutamisega. Siiski ilmnes kitsaskohti, eriti automaatsete hoiatussüsteemide puudumine varundusruumi täitumise korral, mis võib mõjutada süsteemi töökindlust pikemaajalisel kasutamisel.

4.3 Järeldus

Kuigi loodud süsteemi soetamise summa (501,37 €) on märkimisväärne ja süsteemi jooksvad kulud on võrreldavad populaarsete pilveteenustega, pakub lahendus väärtust teistes aspektides. Elektrikulu jääb küll minimaalseks (halvimal juhul keskmiselt 3,40 € kuus), kuid arvestades suurt alginvesteeringut, ei tasuks süsteem end tõenäoliselt rahaliselt kunagi ära. Siiski ei ole see

süsteem loodud kulude optimeerimiseks, vaid ennekõike andmete turvalisuse, privaatsuse ja kontrolli tagamiseks. Kasutaja ei pea usaldama oma isiklikke ja tundlikke andmeid kolmandatele osapooltele, vaid saab omada täielikku kontrolli oma andmete säilitamise ja haldamise üle. Läbi viidud test stsenaariumid näitasid, et süsteem suudab edukalt toime tulla erinevate võimalike rikete ja katkestustega, kuid ilmnes ka vajadus hetkel puuduva teavitussüsteemide järele, eelkõige varundusruumi täitumise olukordades.

Loodud lahendus on suunatud eelkõige neile kasutajatele, kes peavad oluliseks andmete turvalisust ja privaatsust ning on valmis investeerima suurema iseseisvuse ja kontrolli saavutamiseks. Majanduslikult ei pruugi süsteem investeeringut tasa teenida, kuid saavutatud andmete kaitse, hajutatus ja taastamisvõimalused on väärtused, mida tavalised kommertsteenused sageli tagada ei suuda.

5. Kokkuvõte

Käesolevas töös loodi turvaline ja hajus andmehaldussüsteem, mis põhineb Raspberry Pi 5 seadmel ja mille eesmärk on pakkuda kasutajatele suuremat kontrolli andmete salvestamise, turvalisuse ja privaatsuse üle ning pakkuda kaitsed küberlunavara rünnakute eest. Süsteem tugineb mitmele füüsiliselt eraldatud seadmele, tagedes andmekaitse krüpteeritud operatsioonisüsteemide, salvestus ja varundus sektsioonide, versioonihalduse ja turvalise kaugjuurdepääsu virtuaalse privaatvõrgu kaudu.

Süsteemi realiseerimisel kasutati mitmeid tehnoloogiaid. Võrgusalvesti loodi kasutades OpenMediaVault Linuxi distributsiooni. Andmete kaugligipääsetavus saavutati läbi virtuaalse privaatvõrgu loomise, kasutades WireGuardi. Seadmete vaheline andmete sünkroniseerimine toimus võrdvõrgu põhimõttel kasutades Syncthingi ning versioonihalduse jaoks kasutati Borgi.

Andmekandjate kiiruste võrdlus näitas, et Raspberry Pi NVMe pooljuhtkettad edestavad mikro SD-kaarte oluliselt nii kiiruse kui ka kulutõhususe poolest, muutes need kõige optimaalsemaks valikuks piiratud jõudlusega hajutatud salvestuslahenduse loomiseks. Süsteemis läbi viidud test stsenaariumid andsid kinnitust, et lahendus saab edukalt hakkama levinud tõrgete ja katkestustega ilma, et oleks andmekadu.

Kuigi süsteemi esialgne maksumus on kõrge ja võrreldes kommertspilveteenustega ei saavutatud suurt kokkuhoidu, pakub lahendus eeliseid, mida tüüpilised pilveteenused sageli pakkuda ei suuda. Kasutaja säilitab täieliku kontrolli enda andmete üle, andmeid hoitakse kolmes eri asukohas, mis lisaks küberohtudele pakub kaitset ka füüsiliste ohtude nagu näiteks üleujutuste või tulekahjude eest. Versioonihaldus võimaldab taastada andmeid 30 päeva ulatuses, pakkudes kaitset ka lunavara rünnakute ja kasutajavigade eest.

Välja töötatud lahendus on sobilik kasutajatele, kes väärtustavad andmete turvalisust, privaatsust ja iseseisvust enam kui otsest rahalist kokkuhoidu. Edasisteks arendusteks võiks kaaluda teavitussüsteemide ja automaatse järelevalve lisamist, et suurendada süsteemi kasutusmugavust ja töökindlust veelgi.

6. Viited

[1] - Raspberry Pi Foundation, “Raspberry Pi Documentation” (2025) https://www.raspberrypi.com/documentation/, (09.05.2025)

[2] - Metshein, “Mis on Raspberry Pi” (2025) https://www.metshein.com/unit/mis-on-raspberry-pi/, (28.01.2025)

[3] - Rowse, “What Are Single Board Computers?” (2013), https://www.rowse.co.uk/blog/post/what-are-single-board-computers, (09.05.2025)

[4] - Shrivastava, A. R., & Gadge, J. “Home server and NAS using Raspberry Pi. 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)" , 2270–2275 (2017) https://doi.org/10.1109/ICACCI.2017.8126184

[5] - Nasser, M. S. bin A., Attarbashi, Z. S., Aman, A. H. M., & Abuzaraida, M. A. , “Building an affordable portable Storage Area Network (SAN) with Raspberry Pi: Design, implementation, and performance evaluation. 2024 IEEE 4th International Maghreb Meeting of the Conference on Sciences and Techniques of Automatic Control and Computer Engineering (MI-STA) 787–792” (2024) https://doi.org/10.1109/MI-STA61267.2024.10599641

[6] - Raspberry Pi Foundation, “Raspberry Pi 5” (2025) https://www.raspberrypi.com/products/raspberry-pi-5/, (09.05.2025)

[7] - Raspberry Pi Foundation, “Raspberry Pi M.2 Hat Plus – Product Brief” (2025) https://datasheets.raspberrypi.com/m2-hat-plus/raspberry-pi-m2-hat-plus-product-brief.pdf, (09.05.2025)

[8] - NVM EXPRESS, “NVM Express”, https://nvmexpress.org/, (13.05.2025)

[9] - Raspberry Pi Foundation, “Raspberry Pi Documentation M.2 HAT+” , https://www.raspberrypi.com/documentation/accessories/m2-hat-plus.html, (25.04.2025)

[10] - IBM, “What is network attached storage (NAS)?” (2024), https://www.ibm.com/think/topics/network-attached-storage#:~:text=NAS%20devices%20automatically%20back%20up,than%20external%20hard%20disk%20drives.&text=Today's%20NAS%20systems%20incorporate%20data,optimize%20and%20automate%20storage%20functions.&text=Only%201%20in%204%20enterprises,ROI%20from%20cloud%20transformation%20efforts, (08.05.2025)

[11] - RIA, “ISKE meetmed” (2022) https://www.ria.ee/sites/default/files/documents/2022-11/ISKE-meetmed-8-06_0.pdf, (09.05.2025)
[12] - William C. Barker, Karen Scarfone, William Fisher, Murugiah Souppaya, ”Cybersecurity Framework Profile for Ransomware Risk Management”, 2021, https://csrc.nist.gov/CSRC/media/Publications/nistir/draft/documents/NIST.IR.8374-preliminary-draft.pdf , (09.05.2025)
[13] - US-CERT, “Ransomware Guidance”, 2021, https://www.cisa.gov/stopransomware/ransomware-guide, (03.04.2025)
[14] - Jack Schofield, “How can I remove a ransomware infection?”, 2016, https://www.theguardian.com/technology/askjack/2016/jul/28/how-can-i-remove-ransomware-infection (08.02.2025)
[15] - Ernestas Naprys, “Raspberry Pi 5 brings extra kick, creeps in price and power usage” 2023, https://cybernews.com/tech/new-raspberry-pi5-more-performance-higher-price/#:~:text=While%20more%20efficient%2C%20Pi5%20has,8W%20for%20Raspberry%20Pi%204, (12.05.2025)
[16] Estonian Competition Authority, 2024, https://aastaraamat.konkurentsiamet.ee/en/aastaraamat-2024-trends-and-overviews/2024-electricity-and-gas-market-summary, (12.05.2025)
[17] Elektrilevi, “Võrgutasu muudatus alates 01.10.2024”, https://elektrilevi.ee/et/vorguleping/vorgutasu-muudatus , (14.05.2025)
[18] “Plans & Pricing to Upgrade Your Cloud Storage - Google One”, https://one.google.com/about/plans?g1_landing_page=0, (12.05.2025)
[19] “iCloud+ plans and pricing - Apple Support”, https://support.apple.com/en-us/108047, (12.05.2025)
[20] “Cloud Storage Pricing and Plans – Microsoft OneDrive”, https://www.microsoft.com/en-us/microsoft-365/onedrive/compare-onedrive-plans-b, (12.05.2025)

7. Lisad

Lisa I Andmekandjate kiiruste tabelid

Tabel 3. Andmekandjate kiirusklasside tähised ja nendest loodud eeldatav stabiilse kirjutamise kiirusvahemik.

Seadme nimetus	Tüüp	Speed Class	UHS Speed Class	Video Speed Class	Application Performance Class	UHS Bus Speed	Eeldatav kiiruse vahemik (MB/s)
Raspberry Pi Micro SD Card 32GB	micro sd HC	10	U3	V30	A2	UHS-I	30-90
Raspberry Pi Micro SD Card 64GB	micro sd XC	10	U3	V30	A2	UHS-I	30-90
Raspberry Pi Micro SD Card 128GB	micro sd XC	10	U3	V30	A2	UHS-I	30-90
Sandisk Edge 16GB	micro sd HC	10	U1	-	A1	UHS-I	10-90
Sandisk Ultra 16GB	micro sd HC	-	U1	-	-	UHS-I	10-45
Sandisk Ultra 32GB	micro sd HC	10	U1	-	A1	UHS-I	10-90
Sandisk Extreme 32GB	micro sd HC	-	U3	V30	A1	UHS-I	30-90
Sandisk Extreme 64GB	micro sd XC	-	U3	V30	A2	UHS-I	30-90
Samsung Evo Plus 64GB	micro sd XC	-	U1	V10	A1	UHS-I	10-90
Samsung Evo Plus 256GB	micro sd XC	-	U3	V30	A2	UHS-I	30-90

Tabel 4. Mõõtmistulemustest loodud algandmete tabel.

Andmekandja	Järjestikune kirjutamine (MiB/s)	Järjestikune lugemine (MiB/s)	Suvaline kirjutamine (MiB/s)	Suvaline lugemine (MiB/s)	dd kirjutamine (MB/s)	dd lugemine (MB/s)	hdparm lugemine (MB/s)
Raspberry Pi Micro SD Card 32GB	31.7	89.7	32.7	35.6	32.9	94.1	89.81
Raspberry Pi Micro SD Card 64GB	74.5	86.6	68	36.7	72.8	91.6	90.3
Raspberry Pi Micro SD Card 128GB	75.2	88.8	70.8	36.5	73.9	90.9	89.48
Sandisk Edge 16GB	21.2	80.5	21.7	23.8	21.2	92.1	87.17
Sandisk Ultra 16GB	13.5	35.8	13.7	6.45	13.7	40.7	43.47
Sandisk Ultra 32GB	27.7	83.9	26.9	34.5	27.1	93.1	87.73
Sandisk Extreme 32GB	64.9	91.3	62.4	35.5	64.3	93.8	87.9
Sandisk Extreme 64GB	73.8	80.9	79.1	12.7	74.2	85	90.14
Samsung Evo Plus 64GB	33.9	85.8	32.7	34.9	32.7	94.2	91.16
Samsung Evo Plus 256GB	77.9	79.1	64.6	35	67.1	93.8	90.68
SSD Samsung T7 1TB	278	353	254	35.8	285	381	331.75
Raspberry Pi SSD 256GB	442	413	357	125	337	437	415.56
Raspberry Pi SSD 512GB	345	444	344	123	349	470	438.32

Lisa II Kiiruste mõõtmise skript

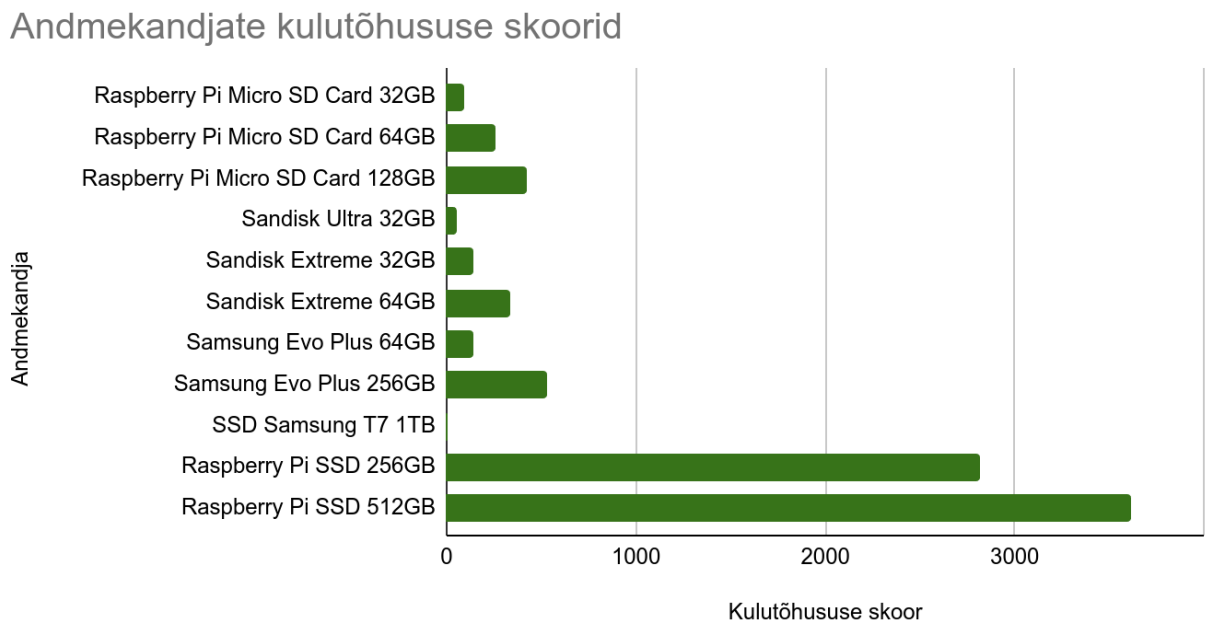
```
#!/bin/bash

# Paigalda vajalikud programmid
sudo apt update && sudo apt install -y fio hdparm dosfstools

# Saadaval olevate seadmete loetelu funktsioon
list_devices() {
    echo "🔍 Available Storage Devices:"
    lsblk -o NAME,SIZE,MOUNTPOINT,TYPE | grep -E "disk|part"
    echo ""
}

list_devices
read -p "Enter the device to test (e.g., nvme0n1, mmcblk0): " DEVICE

# Kontroll, kas valitud seade eksisteerib
if [ ! -b "/dev/$DEVICE" ]; then
    echo "Error: Device /dev/$DEVICE not found!"
    exit 1
fi

# Küsib, kas kasutaja tahab vormindada valitud seadme, kui jah, siis vormindab
read -p "Do you want to format /dev/$DEVICE before testing? (y/n): "
format_choice
if [[ "$format_choice" =~ ^[Yy]$ ]]; then
    echo "WARNING: This will erase all data on /dev/$DEVICE!"
    read -p "Choose format type - (1) ext4 [Recommended], (2) FAT32: "
format_type

    sudo umount "/dev/$DEVICE"* 2>/dev/null

    if [[ "$format_type" == "1" ]]; then
        echo "🛠️ Formatting /dev/$DEVICE as ext4..."
        sudo mkfs.ext4 -F "/dev/$DEVICE"
    elif [[ "$format_type" == "2" ]]; then
        echo "🛠️ Formatting /dev/$DEVICE as FAT32..."
        sudo mkfs.vfat -F 32 "/dev/$DEVICE"
    else
        echo "Invalid selection! Skipping format..."
    fi
fi

# Küsib väljundfaili nime
```

```
read -p "Enter the output file name (e.g., my_benchmark_results.csv): "
RESULTS_FILE

# Kui fail ei eksisteeri siis teeb faili
if [ ! -f "$RESULTS_FILE" ]; then
    echo "Device, Test, Speed (MB/s)" > $RESULTS_FILE
fi

# Küsib kinnituspunkti
read -p "Enter the mount point (e.g., /mnt/test): " MOUNT_POINT
sudo mkdir -p $MOUNT_POINT
sudo mount "/dev/$DEVICE" $MOUNT_POINT || { echo "Failed to mount
/dev/$DEVICE"; exit 1; }

TEST_FILE="$MOUNT_POINT/testfile"

echo "Running benchmarks for /dev/$DEVICE mounted at $MOUNT_POINT..."
echo "Results will be saved in $RESULTS_FILE"

# Käivitab kõik testid automaatselt

## Järjestikuse Kirjutamise Test (fio)
echo "Testing Sequential Write..."
SEQ_WRITE=$(fio --name=seq_write --ioengine=libaio --rw=write --bs=1M
--numjobs=1 --size=1G --runtime=60 --time_based --group_reporting
--filename=$TEST_FILE | grep "WRITE:" | awk '{print $2}')
echo "/dev/$DEVICE, Sequential Write, $SEQ_WRITE" >> $RESULTS_FILE

## Järjestikuse Lugemise Test (fio)
echo "Testing Sequential Read..."
SEQ_READ=$(fio --name=seq_read --ioengine=libaio --rw=read --bs=1M --numjobs=1
--size=1G --runtime=60 --time_based --group_reporting --filename=$TEST_FILE |
grep "READ:" | awk '{print $2}')
echo "/dev/$DEVICE, Sequential Read, $SEQ_READ" >> $RESULTS_FILE

##  Juhusliku Kirjutamise Test (fio)
echo "Testing Random Write..."
RAND_WRITE=$(fio --name=rand_write --ioengine=libaio --rw=randwrite --bs=4k
--numjobs=1 --size=1G --runtime=60 --time_based --group_reporting
--filename=$TEST_FILE | grep "WRITE:" | awk '{print $2}')
echo "/dev/$DEVICE, Random Write, $RAND_WRITE" >> $RESULTS_FILE

##  Juhusliku Lugemise Test (fio)
echo "Testing Random Read..."
```

```
RAND_READ=$(fio --name=rand_read --ioengine=libaio --rw=randread --bs=4k
--numjobs=1 --size=1G --runtime=60 --time_based --group_reporting
--filename=$TEST_FILE | grep "READ:" | awk '{print $2}')
echo "/dev/$DEVICE, Random Read, $RAND_READ" >> $RESULTS_FILE

##  dd Kirjutamise Test
echo "Running dd Write Test..."
DD_WRITE=$(dd if=/dev/zero of=$TEST_FILE bs=1M count=1000 conv=fdatasync 2>&1 |
tail -1 | awk '{print $(NF-1), $NF}')
echo "/dev/$DEVICE, dd Write, $DD_WRITE" >> $RESULTS_FILE

##  dd Lugemise Test
echo "Running dd Read Test..."
sudo sync && sudo echo 3 | sudo tee /proc/sys/vm/drop_caches
DD_READ=$(dd if=$TEST_FILE of=/dev/null bs=1M 2>&1 | tail -1 | awk '{print
$(NF-1), $NF}')
echo "/dev/$DEVICE, dd Read, $DD_READ" >> $RESULTS_FILE

##  hdparm Lugemise Test
echo "Running hdparm Read Test..."
HDPARM_READ=$(sudo hdparm -tT "/dev/$DEVICE" | grep "Timing buffered disk
reads" | awk '{print $11, $12}')
echo "/dev/$DEVICE, hdparm Read, $HDPARM_READ" >> $RESULTS_FILE

# Ühendab lahti ja puhastab
rm -f $TEST_FILE
sudo umount $MOUNT_POINT

echo "Completed tests for /dev/$DEVICE!"
echo "Results saved in $RESULTS_FILE"

# Küsib, kas lõpetada tegevus, või testida järgmist seadet
read -p "Do you want to test another device? (y/n): " choice
if [[ "$choice" =~ ^[Yy]$ ]]; then
    exec "$0"  # Taaskäivitab skripti
fi
```

Lisa III Kulutõhususe skooride põhjal loodud graafik

Joonis 9. Andmekandjate kulutõhususe skooride graafik.



Lisa IV Varundust lihtsustavad skriptid

Skript, mis leiab kindla faili kõik versioonid, mida on võimalik taastada.

```
#!/bin/bash

REPO="<borg-repo-fullpath>"

echo "🔍 Finding latest archive..."
LATEST_ARCHIVE=$(sudo borg list "$REPO" --short | tail -n 1)

if [ -z "$LATEST_ARCHIVE" ]; then
    echo " No archives found."
    exit 1
fi

echo "Latest archive: $LATEST_ARCHIVE"
echo "Listing files with last modification time:"
sudo borg list "$REPO::$LATEST_ARCHIVE" --format="{mtime} {path}{NL}"
```

Skript, mis taastab soovitud faili soovitud versiooni.

```
#!/bin/bash

REPO="<borg-repo-fullpath>"
DEVICE_UUID="<storage-device-UUID>"
DEVICE_PREFIX="srv/dev-disk-by-uuid-$DEVICE_UUID/Storage"
RESTORE_TARGET="/srv/dev-disk-by-uuid-$DEVICE_UUID/Storage"

read -p "Enter the file path starting from /Storage/ (e.g., test.txt): "
REL_PATH
ARCHIVE_PATH="$DEVICE_PREFIX/$REL_PATH"
ABSOLUTE_PATH="$RESTORE_TARGET/$REL_PATH"

echo -e "\n🔍 Searching for: $REL_PATH..."

ARCHIVES=$(sudo borg list --short "$REPO")

FOUND=0
declare -A ARCHIVE_MAP

for archive in $ARCHIVES; do
    line=$(sudo borg list "$REPO::$archive" --format="{mtime} {path}{NL}"
```

```
2>/dev/null | grep "$ARCHIVE_PATH")
    if [[ -n "$line" ]]; then
        FOUND=1
        mod_time=$(echo "$line" | cut -d' ' -f1-2)
        echo "📦 [$archive] -- modified: $mod_time"
        ARCHIVE_MAP["$archive"]="$mod_time"
    fi
done

if [[ $FOUND -eq 0 ]]; then
    echo "File not found in any archive."
    exit 1
fi


echo ""
read -p "Enter the archive name to restore from (copy from above): "
CHOSEN_ARCHIVE

echo -e "\n  Previewing file content from archive '$CHOSEN_ARCHIVE':"
echo "----------------------------------------"
sudo borg extract --stdout "$REPO::$CHOSEN_ARCHIVE" "$ARCHIVE_PATH" 2>/dev/null
echo "----------------------------------------"

read -p "  Do you want to restore this version to the original file location?
[y/N]: " confirm
if [[ "$confirm" =~ ^[Yy]$ ]]; then
    cd /  # aitab õigesse kausta salvestada
    sudo borg extract "$REPO::$CHOSEN_ARCHIVE" "$ARCHIVE_PATH"
    echo "File restored to: $ABSOLUTE_PATH"
else
    echo "Restore cancelled."
fi
```

Lisa V Litsents

Lihtlitsents lõputöö reprodutseerimiseks ja üldsusele kättesaadavaks tegemiseks

Mina, Richard Mario Raun,

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) minu loodud teose **Raspberry Pi baasil turvalise ja hajusa andmehoidla loomine**, mille juhendaja on **Alo Peets**, reprodutseerimiseks eesmärgiga seda säilitada, sealhulgas lisada digitaalarhiivi DSpace kuni autoriõiguse kehtivuse lõppemiseni.
2. Annan Tartu Ülikoolile loa teha punktis 1 nimetatud teos üldsusele kättesaadavaks Tartu Ülikooli veebikeskkonna, sealhulgas digitaalarhiivi DSpace kaudu Creative Commonsi litsentsiga CC BY NC ND 4.0, mis lubab autorile viidates teost reprodutseerida, levitada ja üldsusele suunata ning keelab luua tuletatud teost ja kasutada teost ärieesmärgil, kuni autoriõiguse kehtivuse lõppemiseni.
3. Olen teadlik, et punktides 1 ja 2 nimetatud õigused jäävad alles ka autorile.
4. Kinnitan, et lihtlitsentsi andmisega ei riku ma teiste isikute intellektuaalomandi ega isikuandmete kaitse õigusaktidest tulenevaid õigusi.

Richard Mario Raun
15.05.2025