TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Viktoria Siigur 185405IADB

WireGuard tehnoloogial ning P2P ühendustel põhineva VPN infrastruktuuri arendus

Bakalaureusetöö

Juhendajad: Toomas Lepikult
Doktorikraad
Janno Stern
Bakalaureusekraad

Tallinn 2021

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: Viktoria Siigur

17.05.2021

Annotatsioon

Käesoleva töö eesmärgiks on luua VPN andmevahetus kahe seadme vahel, mille tavapärast ühendust on takistamas tulemüür. Töö teoreetiline osa koosneb virtuaalse privaatvõrgu ülevaatest ja selle kasutusvaldkondadest ning *site-to-site* ja *remote access* VPN ühenduste erinevustest. Samuti käsitletakse VPN protokollide eesmärke ning PPTP, OpenVPN ning WireGuard tehnoloogiate eeliseid ning kitsaskohti. Välja tuuakse lahenduses kasutatava tehnoloogia tavapärase ühenduse seadistus ning pakettide liikumine. Teoreetilise osa teises pooles kirjeldatakse P2P ühendusi üldiselt ning kuidas taolist andmevahetus soketite abil läbi võrguaadresside teisenduse realiseerida. Töö praktilises osas võetakse sammhaaval lahti lahenduse olemus ning kirjeldatakse detailselt koodi toimimist. Lõpetuseks tuuakse välja ühenduse tulemused ning andmete järgi selle kiirus ning töökindlus.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 24 leheküljel, 5 peatükki, 17 joonist.

Abstract

VPN Infrastructure Development Based on WireGuard Technology and P2P Connections

The aim of this work is to create a VPN data exchange between two devices, where the normal connection would be hindered by a firewall. The theoretical part of the work consists of an overview of the virtual private network and its areas of use, and the differences between site-to-site and remote access VPN connections. The advantages and disadvantages of PPTP, OpenVPN and WireGuard protocols are discussed. The configuration of the usual connection of the technology used in the solution and the movement of packets are presented. The second part of the theoretical work describes P2P connections in general and how to implement such data exchange by sockets through network address translation. In the practical part of the work, the essence of the solution is explained step by step, and the operation of the code is described in detail. The solution contains two steps of coding – dealing with the packets from WireGuard client-side and server-side applications and redirecting them through firewall with the help of UDP hole puncher. Finally, the results of the connection and, according to the data, its speed and reliability are displayed.

The thesis is in Estonian and contains 24 pages of text, 5 chapters, 17 figures.

Lühendite ja mõistete sõnastik

API	*Application Program Interface*, rakendusliides
IP	*Internet Protocol*, internetiprotokoll
NAT	*Network Access Translation*, võrguaadresside teisendus
NFC	*Near Field Communication*, lähiväljaside
P2P	*Peer-to-Peer*
PPP	*Point-to-Point Protocol*
PPTP	*Point-to-Point Tunneling Protocol*
TCP	*Transmission Control Protocol*, edastusohje protokoll
UDP	*User Datagram Protocol*, kasutajadatagrammi protokoll
VPN	*Virtual Private Network*, virtuaalne privaatvõrk

Sisukord

Jooniste loetelu

1 Sissejuhatus

Andmevahetus tänapäevases maailmas on muutumas aina enam internetipõhisemaks. Kui eelmise aastakümne lõpuni oli eelmainitud areng üsna stabiilne, siis 2020. aastal alanud globaalne pandeemia suunas paljude inimeste elud, tööd ja tegemised internetti ning lühikese perioodi jooksul muudeti pea kogu suhtlus võrgupõhiseks. Mida suurem on delikaatsete andmete hulk internetis, seda olulisemaks muutub sellest saadava informatsiooni kaitse. Üks meetod, mida internetikasutaja saab iseenda turvalisuse tagamiseks ära teha, on paigaldada enda seadmesse VPN, mis suunab võrguliikluse krüpteeritud kujul teise seadmesse. Nii võetakse pahatahtlikelt pealtkuulajatelt võimalus aru saada, missuguseid andmeid üle võrgu saadetakse.

Käesoleva lõputöö eesmärgiks on luua krüpteeritud ühendus kahe tulemüüriga piiratud seadme vahel. Andmeside loomiseks kasutatakse WireGuard tehnoloogiat, võrguaadresside teisenduse läbimist ning Berkeley soketeid. Lahenduse kasulikkus seisneb seadistamise mugavuses ning kiiruses, kuna tavapäraselt on niisuguse ühenduse loomist vaja internetipakkujaga kooskõlastada ning see asjaolu kulutab nii aega kui raha.

Töö sisaldab detailset ülevaadet virtuaalsete privaatvõrkude tööst, nende tüüpidest ning kasutusvaldkondadest. Lisaks on kirjeldatud VPN protokollide eesmärke ning populaarsemaid tehnoloogiaid, sealjuures on välja toodud detailsem ülevaade WireGuard'ist ja selle seadistamisest. Kirjeldatud on ka *peer-to-peer* ühendused ning nende loomine soketite abil üle võrgu. Lõpetuseks on ära märgitud implementeeritud lahenduse detailid, joonised ning tulem.

Lõputöö praktilise osa arendusfaas koosneb kaheosalisest programmikoodist, mille funktsionaalsuseks on transportida paketid läbi tulemüüride. Arendatud koodi käivitamisel luuakse sild, mille kaudu on võimalik krüpteeritud pakettidel üle võrgu seadmest seadmesse liikuda, eirates tulemüüri takistusi. Pärast silla loomist on koodide ülesandeks võtta vastu WireGuard'ist välja saadetud krüpteeritud UDP paketid ning edastada need teise seadmesse, kus asub koodi teine pool. Edastatud paketid suunab kood selles seadmes asuvasse VPN rakendusse. Kahepoolse koodi mõlemad osad täidavad

sarnast eesmärki, kuid olenevalt seadme ülesandest on need osad mõnevõrra erinevad. Erinevus on tingitud sellest, et üks pool koodist asub seadmes, kust päringuid tehakse, ning teine pool seadmes, kuhu päringud suunatakse. Ülekandekoode kirjeldades toimub ühes seadmes alati esmalt päringute edastamine ning teises vastuvõtmine, kuid põhifunktsionaalsus on mõlemas programmis sama – krüpteeritud pakettide lugemine, nende edastamine ning päringu vastuse tagastamine.

Töö tulemina valmib infrastruktuur, mille kaudu liiguvad määratud aadressile tehtud päringud läbi teise tulemüüriga piiratud seadme ning nende kahe seadme vaheliseks kommunikatsioonikanaliks on töö praktilises osas implementeeritud sild.

2 Virtuaalse privaatvõrgu kirjeldus

Virtual Private Network ehk eesti keeles virtuaalne privaatvõrk on tänapäeval enamasti tuntud kui vahend, mille abil on võimalik internetis suhelda läbi võõra IP-aadressi ning seeläbi varjata seadme originaalset võrku ning asukohta. Kirjeldatud kasutusviis on reaalsuses vaid üks eelistest antud ühenduse puhul. Definitsiooni järgi võimaldab VPN internetis olevatel seadmetel omada turvalist juurdepääsu teistele võrkudele olenemata ühenduse vahemaast [1]. Mõiste tähendab, et suvalises võrgus asuval seadmel on võimalik saata ning vastu võtta andmeid mujal võrgus olevatest seadmetest ilma vaheserverita mistahes kauguselt. Antud ühenduse teeb turvaliseks asjaolu, et kogu suhtlus toimub krüpteerituna ning seda pealt kuulates ei ole võimalik tuvastada andmete päritolu ning sisu. Juhul kui võrku saadetakse kogu liiklus, mis kaugühendatud seadmest läbi läheb, jääb avalikkusele mulje, et päringud tulevad teise võrgu avalikult IP aadressilt ning seetõttu saab seadme näiline asukoht erineda selle reaalsest viibimispaigast.

Algupäraselt oli VPN ühendus mõeldud ettevõtete jaoks, kus delikaatsete ärisaladuste jagamine võib turvamata kujul jõuda valedesse kätesse ning põhjustada tõsiseid probleeme äri toimimises. Turvalise ühenduse abil saab firma töötaja kodukontoris või komandeeringul olles ligi ettevõtte võrgule ning andmeid vahendades jääb nende lekkerisk minimaalseks. Samuti kasutatakse turvaühendust juhul, kui ettevõttel on kontorid erinevates linnades või ärilistel põhjustel on tarvilik jagada andmeid vahetult teiste ettevõtetega.

Aja jooksul on muutunud VPN kasulikuks vahendiks interneti tavakasutaja jaoks. Mitmed riigid on kehtestanud tsensuuri internetis, et säilitada traditsionaalseid väärtusi, hoida poliitilist stabiilsust ning tagada rahvusliku julgeolekut [2]. Samuti on paljude tuntud rakenduste sisud piirkonniti erinevad. Voogedastusplatform Netflix on teenuse jagamisel lähtunud regionaalse blokeeringu süsteemist, et vältida autoriõiguste ning litsentside rikkumist [3]. Kuna süsteem kontrollib kasutajate asukohti IP-aadressi põhiselt, on kasutajatel võimalik seda virtuaalse privaatvõrgu abil ära petta. Nõudlus vaba interneti järele on tekitanud pinnase VPN teenusepakkujatele, kes püstitavad servereid

erinevatesse riikidesse, kuhu kliendid saavad enda seadmeid vastavalt vajadusele turvaliselt kaugühendada.

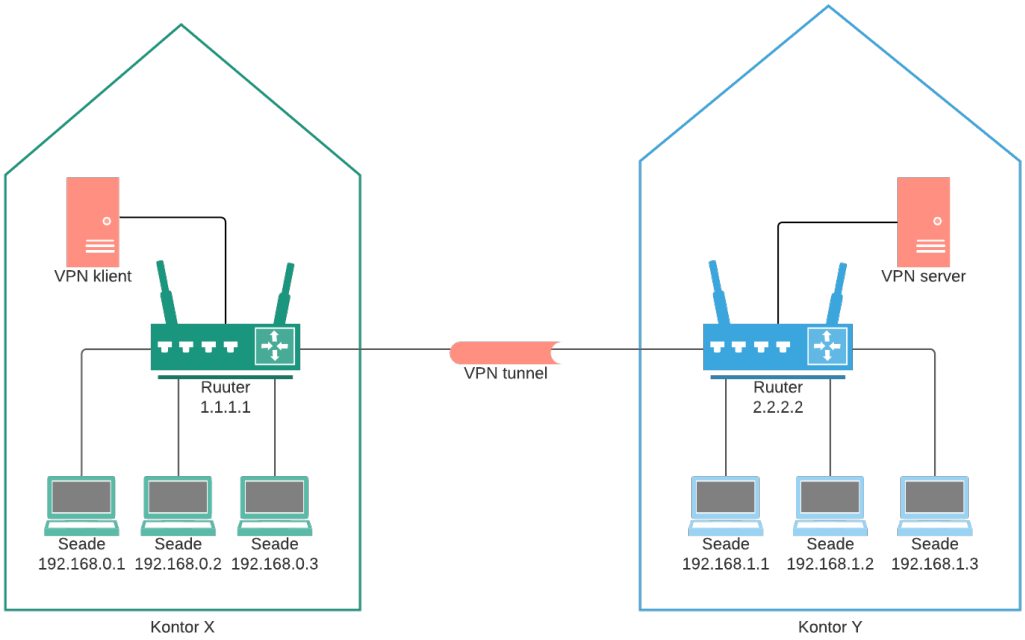
Internetivabaduse suurenemine loob kasutajatele võrdsemaid võimalusi olenemata nende asukohast. Kuna süsteem tugineb faktile, et kogu suhtlus on krüpteeritud ning tegevuste kaudu on kasutajaid pea võimatu tuvastada, on see loonud võimalusi ka kuritegelikule maailmale. Mõjuka näitena saab tuua 2011. aastal loodud populaarseima illegaalse kaubandusplatvormi Silk Roadi [4], mille eksisteerimine poleks VPN ühenduste ning raskendatud isikutuvastuseta olnud võimalik. Kuid kuritegevust on võimalik leida igast olemasolevast valdkonnast ning virtuaalse privaatvõrgu tehnoloogia peamiseks eesmärgiks on siiski tagada inimeste ning nende andmete turvalisus ja kaitse.

2.1 VPN tüübid

Virtuaalsed privaatvõrgud on algupäraselt ning ka tänapäeval kasutuses ettevõtete töö turvaliseks toimimiseks. Nende andmevahetus ei pruugi alati olla vaid põhivõrgu ning ühendatava seadme vahel, vaid vaja võib minna ka ühendust teise võrguga. Kuna mainitud ühendused erinevad üksteisest tehniliselt, jaotatakse need kaheks erinevaks VPN tüübiks.

2.1.1 Site-to-Site

Kahe erineva võrgu vahelist VPN tunnelit nimetatakse *site-to-site* või teise nimega *router-to-router* tüüpi turvaühenduseks. Peamine omapära antud ühenduse juures on see, et ühendatud pole ainult üksikud seadmed, vaid kõik kohaliku võrgu liikmed korraga. Tunnelit kasutatakse suurtes ettevõtetes või organisatsioonides, kus kontorite asukohad võivad olla üksteisest kaugel. Ettevõttesisest ühendust nimetatakse Intranetile baseerutud virtuaalseks privaatvõrguks. Juhul, kui ühendus luuakse kontoriga, mis kuulub teisele ettevõttele, on VPN Extranetile põhinev. [5] Joonisel 1 on visualiseeritud *site-to-site* ühendus kahe kontori vahel.

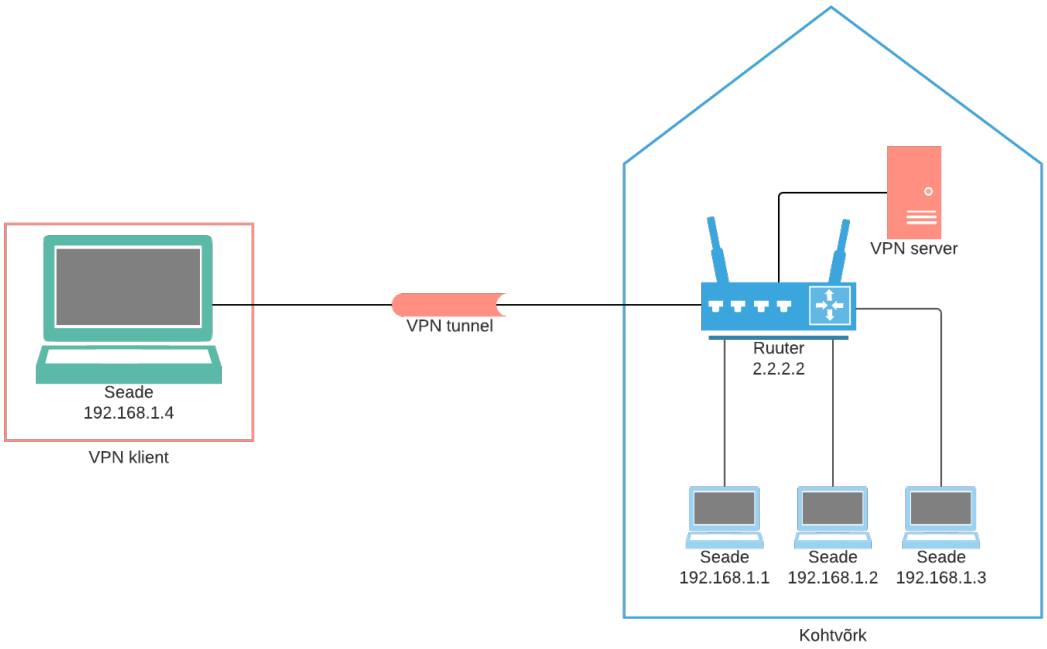


Joonis 1. Site-to-Site VPN ühendus.

Kui kontoris X asuvast seadmest, mis on internetiga ühendatud läbi ruuteri IP-aadressiga 1.1.1.1, saadetakse andmepakett kontorisse Y, läheb pakett kõigepealt läbi VPN serveri, kus ta krüpteeritakse. Kontoris Y asuv ruuter IP-aadressiga 2.2.2.2 võtab paketi vastu ning suunab VPN serverile, mis selle omakorda dekrüpteerib ning edastab lõppsihtmärgile ehk seadmele, kellele antud pakett oli mõeldud.

2.1.2 Remote Access

Kaugjuurdepääsu VPN toimub *site-to-site* ühendusest mõnevõrra teisiti. Kui eelneva puhul on ühendatud kogu võrk, siis *remote access* VPN-iga saab ühendada konkreetse seadme, millesse on seadistatud VPN klient, eemal asuva võrguga, kus asub turvaühenduse server. Joonisel 2 on kujutatud seade, millele on võimaldatud kaugligipääs võrgule avaliku IP-aadressiga 2.2.2.2.



Joonis 2. Remote Access tüüpi VPN ühendus.

Kui ühendatud seadmest soovitakse saata andmeid võrku, krüpteeritakse need seadmes olevas VPN kliendis, edastatakse läbi avaliku interneti, dekrüpteeritakse võrgus olevas VPN serveris ning suunatakse lõppseadmele.

Kuna antud ühenduse puhul on asub seade virtuaalselt uues kohtvõrgus, omistatakse sellele uus avalik IP-aadress, läbi mille on võimalik suunata kõik seadmes tehtavad päringud. Teisisõnu, kui seadmest tehakse päring suvalisse veebiserverisse, siis see liigub esmajoones võrku, kuhu seade on ühendatud, ning seejärel serverisse, kuhu päring suunati. Sellisel juhul saab päringut sisaldav pakett endale külge seadme originaalse avaliku IP-aadressi asemel kaugühendatud võrgu avaliku aadressi ning veebiserveril ei ole võimalik tuvastada päringu algset asukohta. Antud asjaolu seletab ka seadme geolokatsiooni muutumise nähtust, sest seadme asukoha IP põhist tuvastust sooritatakse avaliku aadressi järgi, kuid VPN-i puhul võivad seadme reaalne asukoht ning kaugühendatud ruuteri avalik IP olla üksteisest mistahes kaugusel.

2.2 VPN protokollid

Virtuaalse privaatvõrgu protokoll on reeglistik, mille põhjal rakendatakse juhised sõnumite turvaliseks saatmiseks avalikus internetis [6]. Erinevate VPN protokollide

tööpõhimõtted varieeruvad nende kasutajate soovitud eesmärkidest. Nende turvalisus ja kiirus sõltuvad üksteisest vastupidiselt – mida tugevam krüpteering, seda kauem kestab andmevahetus. Virtuaalse privaatvõrgu töös hoidmiseks kasutatakse protokolle, kus on kirjeldatud nii andmete saatmise kui krüpteerimise reeglistik, kuid samuti on VPN süsteeme, kus andmete liigutamiseks kasutatakse ühte protokolli ning krüpteerimiseks teist. [7]

2.2.1 Point-to-Point Tunneling Protocol

Esimest VPN protokolli kirjeldati 1999. aastal ning selleks oli PPTP ehk *Point-to-Point Tunneling Protocol.* Protokoll põhineb 1996. aastal loodud *Point-to-Point Protocol*'ile, tuntud ka lühendina PPP. PPTP kirjeldab meetodit, kuidas transportida PPP läbi tunneli ehk muuta andmed pealtkuulajatele arusaamatuks. [8] Protokolli krüpteering on üsna primaarne ning seda on võimalik kiirelt lahti murda, kuid ühenduse kiirus on see-eest parem, kui mõne uuema VPN protokolli puhul [7]. PPTP on siiani kasutusel, kuid selle kõrvale on aegade jooksul ilmunud teisi, mille turva ning andmevahetuskiiruse parameetrid on üksteisest mõnevõrra erinevad.

2.2.2 OpenVPN

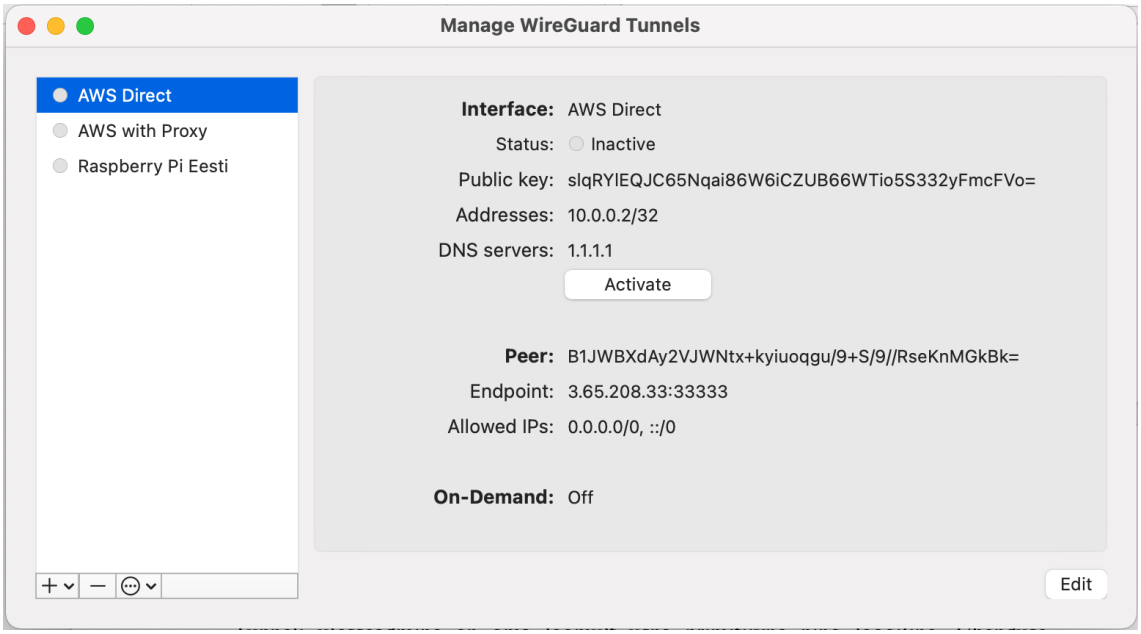
Teine läbimurdeline VPN protokoll, mis erineb eelnevatest oma läbipaistvuse ning integreerimisvõimaluste tõttu, on 2001. aastal loodud OpenVPN. Protokoll on avatud lähtekoodiga ning seda on võimalik kasutada nii *site-to-site* ühendusel kui ka kaugjuurdepääsu loomisel. Ühendus on võimeline transportima UDP ja TCP pakette ning OpenVPN proksiserverit on võimalik paigaldada nii klient- kui ka serverarvutisse ühtmoodi. [9]

2.2.3 WireGuard

2016. aastal välja antud WireGuard on üks uuemaid avatud lähtekoodiga protokolle, mille arenduseks on kasutatud modernseid lahendusi ning võtmesõnaks on lihtsus. Protokolli ühenduse kiirus on testide tulemuste järgi hinnatud 4 korda kiiremaks kui OpenVPN jõudlus seda võimaldab [10]. Samuti on WireGuardi koodibaas tunduvalt väiksem võrreldes teiste VPN protokollidega, mis muudab tehnoloogia analüüsimise võimetekohaseks ka üksikule indiviidile. [11]

2.3 WireGuard tunneli seadistamine

Tunneli ülesseadmine on oma loomult üsna primitiivne ning loogiline. Ühenduse loomiseks on kasutatud kahe erineva avaliku IP taga olevat seadet. Kliendiks on seade, mille operatsioonisüsteemiks on MacOS ning server töötab Ubuntul. MacOS'is on võimalik App Store'ist alla laadida WireGuardi graafiline liides, mis genereerib kasutajale automaatselt ühenduse privaatsed ning avalikud võtmed, mille järgi hakatakse andmeid krüpteerima. Joonisel 3 on näha MacOS operatsioonisüsteemile loodud WireGuardi graafilist liidest.



Joonis 3. Kuvatõmmis WireGuardi graafilisest liidesest.

Ubuntu puhul käib kogu seadistus terminali akna kaudu. Esmalt laetakse alla WireGuardi tööriistad ning genereeritakse käsuga „wg genkey | tee privatekey | wg pubkey > publickey" avalik ja privaatne võti. Seejärel luuakse konfiguratsioonifail nimega wg0.conf kausta „/etc/wireguard/" koos vajalike seadistustega. Joonistel 4 ja 5 on näha eduka VPN ühenduse konfiguratsioone.

```
[Interface]
PrivateKey = iBA8A34kfYs/1TVgJu5OwYV7vHT3PjN5qn2+6tpmVXY=
Address = 10.0.0.2/32
DNS = 1.1.1.1

[Peer]
PublicKey = B1JWBXdAy2VJWNtx+kyiuoqgu/9+S/9//RseKnMGkBk=
AllowedIPs = 0.0.0.0/0, ::/0
Endpoint = 3.65.208.33:33333
```

Joonis 4. Kliendipoolne WireGuard tunneli seadistus.

Konfigureerimisel luuakse kliendipoolsele liidesele uus kohalik IP aadress ning lisatakse pakettide dekrüpteerimiseks eelnevalt genereeritud privaatne võti. Samuti lisatakse seadistustesse lõpp-punkti aadress, et klientrakendus teaks, kuhu pakette saata, ning selle avalik võti, millega paketid enne serverisse saatmist krüpteeritakse. Kliendi puhul on oluline ka ära määrata, millistele IP-aadressidele saadetud päringud lubatakse ümber suunata serverisse. Joonisel 4 on lubatud IP-aadressideks määratud kõikide olemasolevate IP-de spekter.

```
[Interface]
ListenPort = 33333
Address = 10.0.0.1/32
PrivateKey = KB54GYpQQodFD0AV5G2YW7fzIov2gutE09sflADLIWs=
PostUp = iptables -A FORWARD -i %i -j ACCEPT; iptables -A
FORWARD -o %i -j ACCEPT; iptables -t nat -A POSTROUTING -
o eth0 -j MASQUERADE;
PostDown = iptables -D FORWARD -i %i -j ACCEPT; iptables
-D FORWARD -o %i -j ACCEPT; iptables -t nat -D
POSTROUTING -o eth0 -j MASQUERADE;

[Peer]
PublicKey = fNDdeQGbp5bC3DzvPeWWaae8+5ED2zsbUZArXC3wpDY=
AllowedIPs = 10.0.0.2/32
```

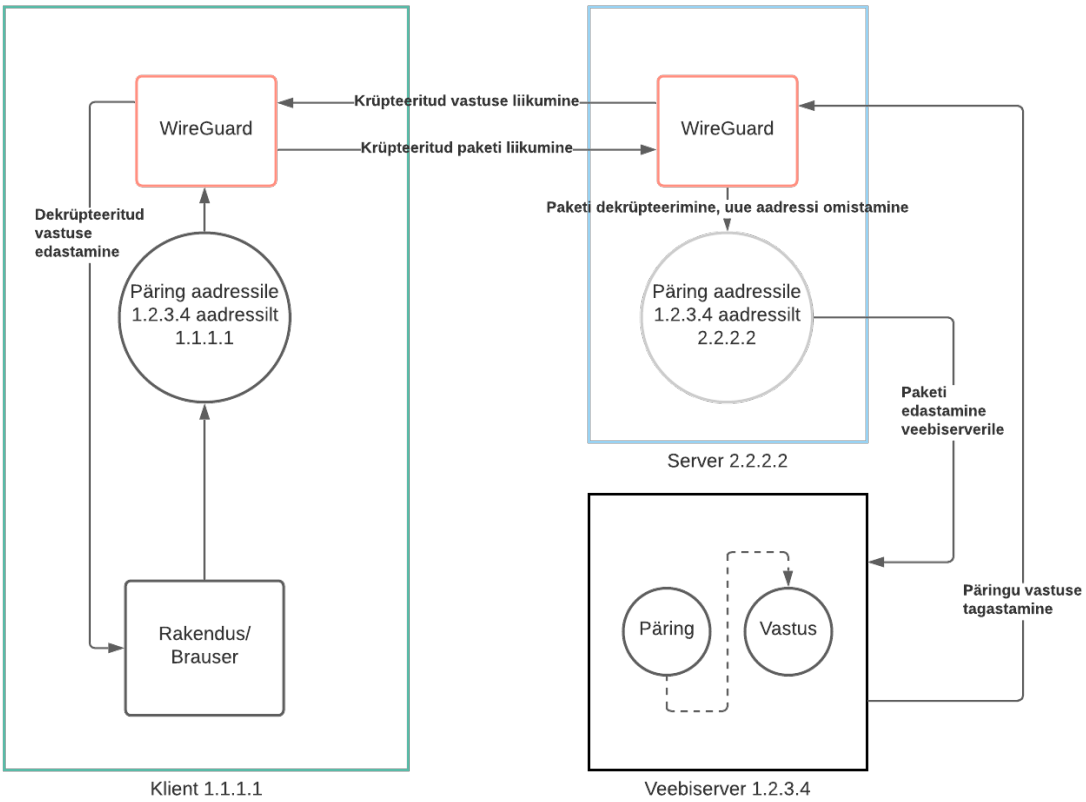
Joonis 5. Serveripoolne WireGuard tunneli seadistus.

Serveri liidese puhul määratakse samuti uue ühenduse kohalik aadress ning port, mille peal rakendus pakette kuulama hakkab. Lisaks on konfiguratsioonis serveri privaatne võti, mis suudab sisse saadetud andmed dekrüpteerida. Päringu vastuse krüpteerimiseks

on lisatud seadistustesse kliendi avalik võti ning uue VPN võrgu kohalik aadress, et server saadaks kliendile ainult need vastused, mis on temale suunatud.

2.3.1 Pakettide liikumine

Kui klient teeb päringu IP-le, mis on konfiguratsioonis lubatud aadresside all olemas, suunatakse see määratud sihtkoha asemel WireGuardi rakendusele. Rakendus muudab serveri avalikku võtit kasutades päringus olevad andmed loetamatuks. Andmed saadetakse WireGuard serverile kasutades UDP pakette. Server võtab paketid koos andmetega vastu, muudab uuesti loetavaks ja edastab need serveri aadressi kasutades esialgselt määratud sihtkohale. Sihtkoht saadab päringu vastused tagasi serverile, mis omakorda krüpteerib andmed kliendi avalikku võtit kasutades ja edastab lugematuks muudetud vastused kliendile tagasi. WireGuard rakendus edastab vastused pordile, kust on pärit algne päring. [11] Joonisel 6 on visualiseeritud pakettide liikumine päringu algpunktist veebiserverini ning sealt saadud vastuse teekond tagasi pärijani.



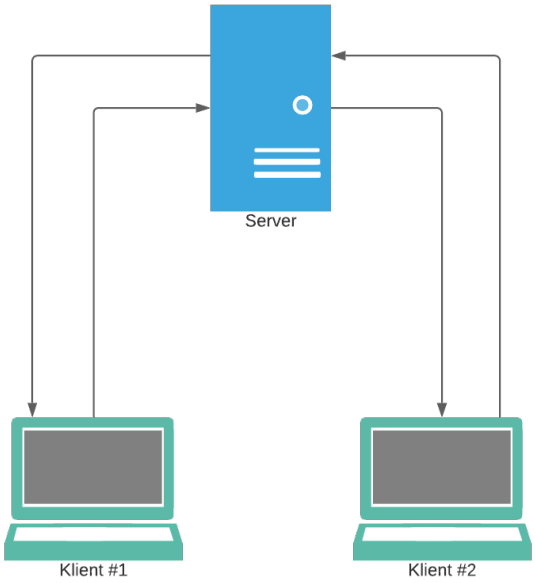
Joonis 6. Paketi liikumine läbi tavapärase WireGuard ühenduse.

2.3.2 Tulemüüri probleemi lahendamine ning ülesande püstitus

Loodud ühendus töötab edukalt, sest näites kasutatud serverile on olemas ligipääs kõikidelt IP-aadressidelt. Juhul kui tegemist on seadmega, mis asub tulemüüri taga, blokeeritakse suvalistelt IP-aadressidelt tulnud paketid ära. Üldjuhul lahendatakse antud probleem manuaalselt läbi internetipakkuja, kellega kooskõlastades saab avada tulemüüris pordid, läbi mille oleks võimalik ühendus tööle saada, kuid see tähendaks lisa aja- ning rahakulu. Eesmärk on luua lahendus, mille abil on võimalik transportida WireGuardis genereeritud krüpteeritud andmetega UDP paketid porte manuaalselt avamata läbi blokeeriva tulemüüri ning seeläbi saavutada toimiv andmevahetus kahe seadme vahel.

3 P2P ühendus

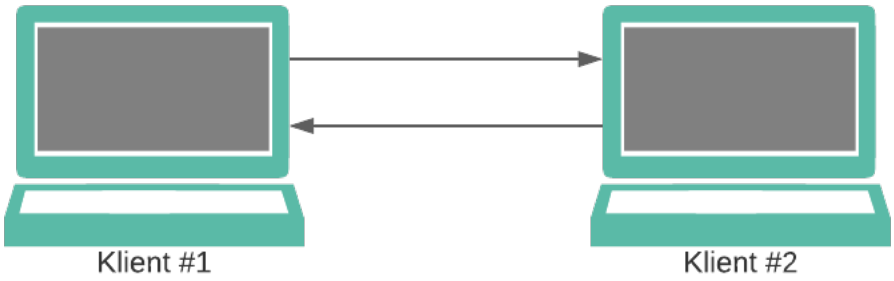
Kahe seadme vahelist otseühendust nimetatakse P2P ehk *peer-to-peer* ühenduseks. Kui klient/server andmevahetuse puhul saadetakse andmed kesksesse serverisse, kust teine kasutaja neile vajadusel ligi pääseb, siis P2P ühendusel suunatakse andmed vahetult teise seadmesse. Joonistel 7 ning 8 on vastavalt visualiseeritud klient/server kui ka P2P ühendused.



Joonis 7. Klient/server ühendus.

Bluetooth, NFC ning sajandivahetusel aktiivselt kasutuses olnud infrapunaliides võimaldavad luua seadmete vahelist otseühendust. Tänapäeval on üha enam hakatud kasutama internetivälise ühenduse loomiseks tehnoloogiat nimega Wi-Fi Direct. Kõik eelmainitud tehnoloogiad on loodud ühendusteks seadmete vahel, mis asuvad üksteisele lähedal. Andmevahetust, mis toimuks olenemata seadmete omavahelisest kaugusest, luuakse üldjuhul läbi internetiühenduse.

WireGuardi puhul liiguvad andmed samuti *peer*'ist *peer*'i. Protokoll kasutab suhtluseks UDP ehk kasutajadatagrammi protokolli järgi struktureeritud pakette, mis üle võrgu kliendist serverini ning vastupidi liiguvad.



Joonis 8. P2P ühendus.

3.1 Berkeley Socket'ite rakendusliides

Teek, mida nimetatakse Berkeley *Socket*'ite API-ks, on funktsioonide, makrode ja andmestruktuuride kogu, mis lubab programmil luua ning hallata suhtlusvooge kahe või enama protsessi vahel samas seadmes või seadmete võrgus [12]. Soketite abil on võimalik saata UDP pakette ühelt aadressilt teisele paari koodireaga. Joonisel 9 on kuvatud lihtne näidis sõnumi saatmisest kliendilt serverile.

```
const dgram = require('dgram');

// ====== Klient ======
const client = dgram.createSocket('udp4');

const serverAddress = {
  port: 3333,
  host: '127.0.0.1'
}

client.send("Tere, server!", serverAddress.port, serverAddress.host);

// ====== Server ======
const server = dgram.createSocket('udp4');

server.on('message', (message, rinfo) => {
  console.log(`${rinfo.address}:${rinfo.port} - ${message}`);
});

server.bind(3333);
```

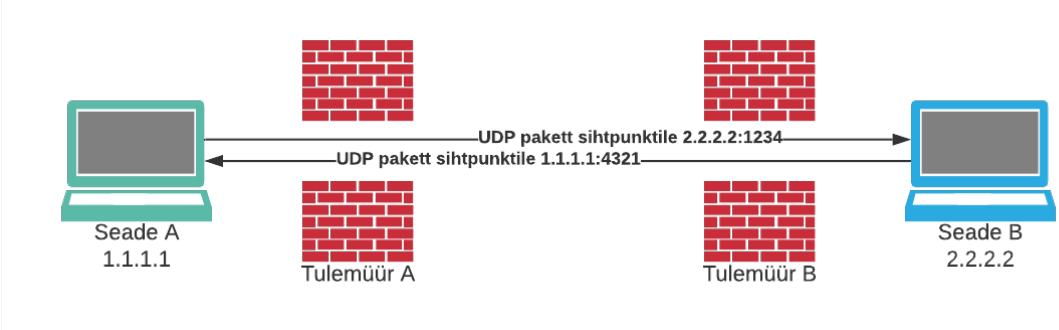
Joonis 9. Berkeley Socket'ite kasutusnäide JavaScriptis.

Programmikoodis luuakse esimesena datagrammi instants, mille kaudu on võimalik genereerida UDP pakettide jaoks loodud soketeid. Eelmainitud koodis on kasutusel kliendi soket, mis on mõeldud sõnumite saatmiseks, ning serveri soket, mis need teisel pordil vastu võtab. Määratud on konstandina serveri aadress, mis koosneb IP-st ning pordist. Kuna server asub kliendiga samas võrgus, on serveri IP-na märgitud lokaalse hosti aadress. Funktsiooniga „server.bind(3333)" luuakse side kliendiga, mis määratud pordile saadetud paketid suudab sisse võtta. Sõnumite saatmiseks kasutatakse funktsiooni „client.send()" millele antakse parameetritena kaasa sõnumi sisu ja saaja IP ning port. Serverist sõnumite lugemiseks on soket programmi jooksmise ajal konstantselt ootel ning kui server tuvastab paketi, käivitatakse funktsioon sõnumi ning saatja andmete konsooli logimiseks.

3.2 Võrguaadresside teisendus

P2P ühenduste on peamiseks takistuseks on NAT ehk võrguaadresside teisendus, mis võimaldab ühe avaliku IP-aadressi taga hoida suurel hulgal seadmeid. Teisendus tehakse ruuteri/tulemüüri sees ning see on peatükis 2.3.2 kirjeldatud probleemi põhjuseks.

Suhtluse loomist tulemüüri taga olevate seadmete vahel nimetatakse NAT *traversal*'iks ning selle realiseerimiseks UDP pakettide transpordil on olemas meetod, mida nimetatakse UDP *hole punching*'uks. Meetod seisneb asjaolul, et tulemüür lubab UDP paketil võrku siseneda juhul, kui sealt on eelnevalt läbi läinud saatja aadressile suunatud pakett. Joonisel 10 on visualiseeritud juhtum, kus seadmel A töötavast rakendusest, mis asub pordil 4321, saadetakse pakett seadmel B olevale rakendusele pordinumbril 1234.

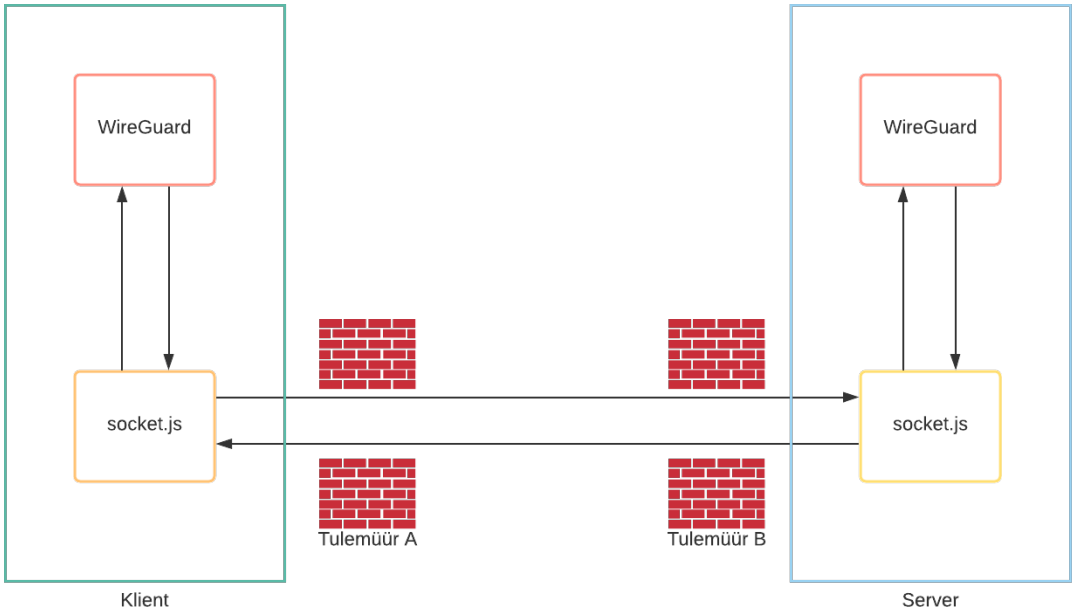


Joonis 10. UDP pakettide läbimine tulemüürist.

Seadmest A välja läinud paketi lõppaadress märgitakse tulemüüris A üles ning selle järgi lubatakse seadmest B saadetud paketid läbi, kuna nende päritolu on tuttav. Samamoodi toimib lahendus ka tulemüüri B puhul. [13]

4 WireGuard suhtlus läbi tulemüüri aukude

Krüpteeritud andmete kohale jõudmiseks tulemüüri taga asuvasse seadmesse on vaja peatükis 3.2 kirjeldatud meetodi abil suunata need läbi UDP pakettidele mõeldud aukude. Selleks on tarvis suunata esmalt kõik paketid pordile, mille peal pakettide suunamiseks loodud programm neid kuulamas on. Kui programmis oleva soketi aadressile tuleb sõnum, edastatakse see otsejoones edasi läbi UDP *hole puncher*'i teegiga loodud ühenduse serverseadmes jooksvasse programmi. Vastuvõetud pakett edastatakse pordile, mida on kuulamas WireGuard serveripoolne rakendus ning seal dekrüpteeritakse sõnum. Kui eelmainitud sõnumist välja loetud päring saab vastuse, tagastatakse see WireGuard rakendusest esmajoones serveris käimasolevasse programmi, kus on muutujasse salvestatud paketi algasukoha port. Selle järgi oskab rakendus suunata krüpteeritud andmed läbi aukude kliendipoolsele programmile tagasi. Klientprogrammis kuulab teine soket serverist tulnud pakette ning edastab need kliendipoolsesse WireGuard rakendusse. Joonisel 11 on näha lahenduse visualiseeritud struktuur.



Joonis 11. Lahenduse üldine struktuur.

4.1 WireGuardi klient

WireGuardi konfigureerimine toimub uudse infrastruktuuri puhul sarnaselt peatükis 2.3 kirjeldatud seadistustele. Liidese privaatvõtme ning ühendatud seadme avaliku võtme genereerimine ning määramine toimub samamoodi, nagu tavalise ühenduse puhul, sest krüpteering toimub ka uue ühenduse puhul WireGuardi rakenduses. Peamine muudatus konfiguratsioonis on ühenduse lõpp-punkti aadressi vahetus, sest rakendus ei saada UDP pakette nüüdsest mitte otse server-rakendusse, vaid klientseadmes jooksvasse rakendusse. Joonisel 12 on määratud lõpp-punkti IP-aadressiks 127.0.0.1, mis on *localhost*'i ehk kohaliku hosti aadress, koos pordinumbriga 11111. Samuti on muudetud edastamiseks lubatud IP-aadressiks määratud vaid üks konkreetne aadress, et infrastruktuuri testimise käigus saaks päringuid paremini hallata ning analüüsida.

```
[Interface]
PrivateKey = oA8990fl1xcIw/rHOSYuIzCoyDaQUApGJJ2y7biDa0k=
Address = 10.0.0.2/32

[Peer]
PublicKey = J6/U0f4iKVFSO4euUg7EVQl5L3TkgRRAElYIXDFgeFA=
AllowedIPs = 10.0.0.1/32
Endpoint = 127.0.0.1:11111
```

Joonis 12. Kliendipoolne WireGuardi seadistus edastuslahendusel.

4.2 Pakettide edastamine ja vastuvõtt

Krüpteeritud sõnumite lugemiseks ja edastamiseks loodud kaheosaline programmikood koosneb kolmest põhilülist ning kahest etapist. Esimeseks etapiks on luua kliendi ning serveri vaheline ühendus läbi UDP *hole puncher*'i. Ühendus luuakse joonistel 13 ja 14, vastavalt kliendis ja serveris määratud konstandi „puncher“ kaudu kasutades „dgram“ teegi funktsiooniga genereeritud soketit, mille tüübina on määratud UDP4 kasutajadatagrammi protokolliga struktureeritud pakettide saatmiseks ja vastuvõtuks. Teise sammuna suunatakse klient- ja serverrakendustes sissetulevad ja väljaminevad paketid eelmises etapis loodud ühendusest läbi. Eelpool mainitud kolmeks põhilüliks on

pakettide vastuvõtt WireGuard rakendusest, nende edastamine teisele programmile ning suunamine WireGuard rakendusse.

```
const dgram = require('dgram');
const socketA = dgram.createSocket('udp4');
const socketB = dgram.createSocket('udp4');
const UDPHolePuncher = require('udp-hole-puncher');

const serverIp = '178.23.118.195';
const clientIp = '127.0.0.1';

let localPort = 0;

socketB.on('listening', () => {
  const puncher = new UDPHolePuncher(socketB);
  puncher.on('connected', () => {
    socketA.on('message', (message, sender) => {
      socketB.send(message, 44444, serverIp);
      localPort = sender.port;
    });
  });
  puncher.connect(serverIp, 44444);
});
socketB.bind(55555);

socketB.on('message', (message) => {
  socketA.send(message, localPort, clientIp);
});
socketA.bind(11111);
```

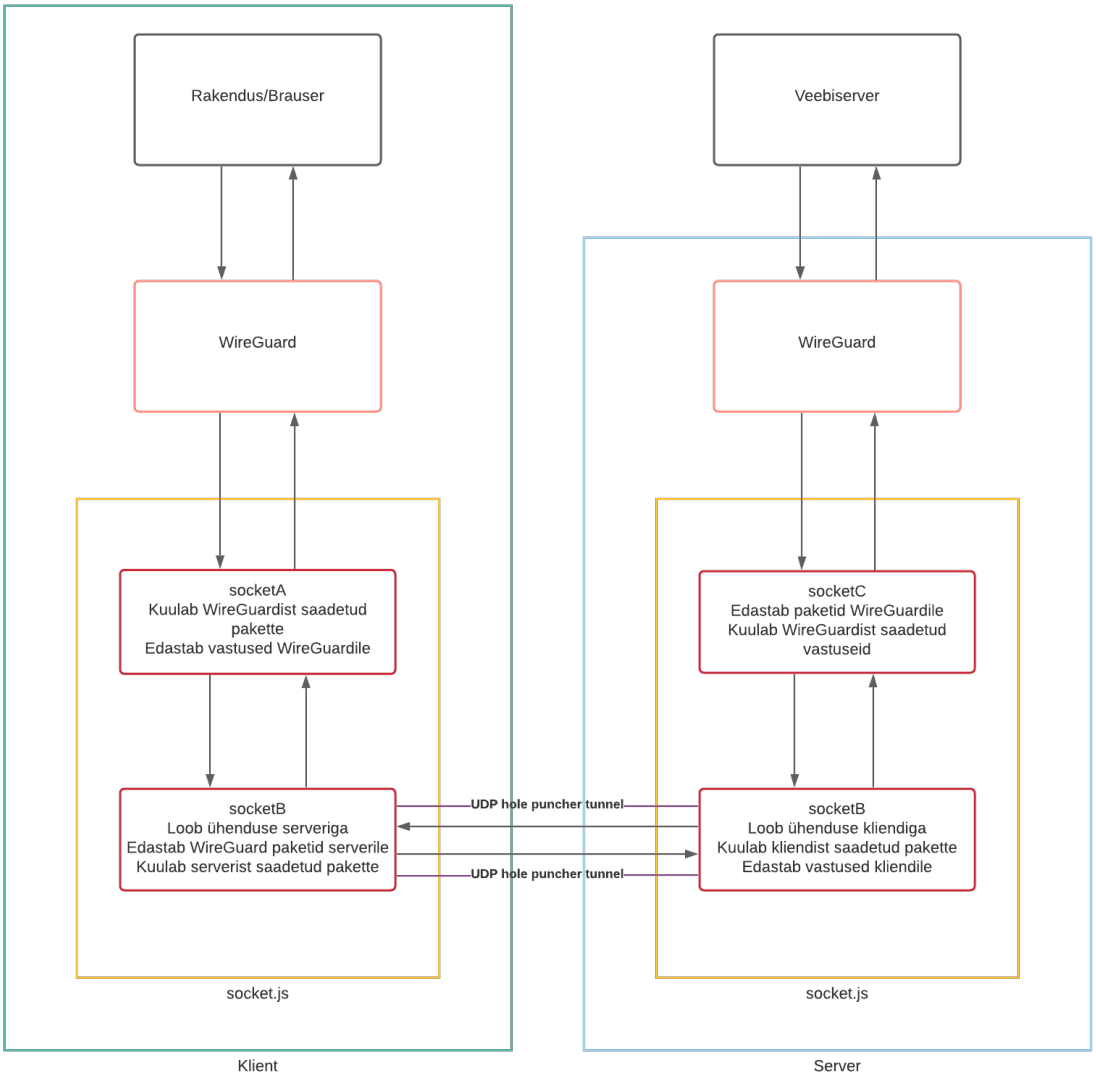
Joonis 13. Kliendipoolne edastusprogramm UDP pakettide ülekandeks.

Kui ühenduse loomine UDP augulööja abil läbi tulemüüri on õnnestunud, hakkab joonisel 13 olevas programmikoodis määratud konstant nimega „socketA“ vastu võtma WireGuard kliendist tulevaid sõnumeid, sest antud soket on seotud sama pordiga, mis on VPN klientrakenduses määratud kui pakettide saatmise lõpp-punkt. Kui „socketA“ saab sõnumi, käivitatakse funktsioon, millele antakse kaasa sõnumi sisu ning saatja aadress ja port. Funktsioon edastab „socketB“ kaudu saadud sõnumi „puncher“-iga juba ühendatud pordile ja aadressile. Lisaks salvestatakse muutujasse saatja port, et hiljem oskaks programm päringu vastused tagasi saata.

```
const dgram = require('dgram');
const socketC = dgram.createSocket('udp4');
const socketB = dgram.createSocket('udp4');
const UDPHolePuncher = require('udp-hole-puncher');

const serverIp = '127.0.0.1';
const serverPort = 33333;

const clientIp = '78.104.100.127';

socketB.on('message', (message) => {
  socketC.send(message);
});

socketB.on('listening', () => {
  const puncher = new UDPHolePuncher(socketB);
  puncher.on('connected', () => {
    socketC.on('message', (message) => {
      socketB.send(message, 55555, clientIp);
    });
  });
  puncher.connect(clientIp, 55555);
});
socketB.bind(44444);

socketC.connect(serverPort, serverIp);
```

Joonis 14. Serveripoolne edastusprogramm UDP pakettide ülekandeks.

Kui programm on kliendipoolsest seadmest UDP paketi teele pannud, on serverprogrammis defineeritud soket nimetusega „socketB“ juba ootamas, sest tegemist on läbi UDP *Hole Puncher*’i rajatud ühenduse loomiseks kasutatud objektiga. Sõnumi kättesaamisel edastatakse see „socketC“ kaudu WireGuardi serverile, kuhu see on eelnevalt kohaliku hosti aadressiga ning WireGuard konfiguratsioonis määratud kuulamispordiga ühendatud. Paketi saamisel ei salvestata saatja aadressi muutujasse, nagu seda tehti klientprogrammis, sest serveri puhul on saatja port ning aadress eelnevalt loodud ühenduses juba määratud ning programm teab, kuhu päringu vastus tagasi saata. Joonisel 15 on kuvatud koodi toimimismehhanismi visualiseeritud skeem.



Joonis 15. Koodi visualisatsioon.

4.3 WireGuardi server

Serveris asuva WireGuard rakenduse konfiguratsioonis pakettide ülekandeks loodud programmi asukohta määrama ei pea, kuna serverist andmete välja saatmiseks peavad päringud olema eelnevalt kliendist või antud juhul ülekandeks loodud programmist läbi tulnud. Paketi serverisse jõudmisel salvestatakse automaatselt selle saatja aadress, et hiljem päringu vastused saatjale ehk käesoleval juhul programmile tagastada. Eelkirjeldatud põhjusel ei muutu ülekandemeetodil toimiva VPN-i serveri konfiguratsioon võrreldes tavapärase ühenduse seadistustega, välja arvatud turvavõtmete väärtused, mida on näha jooniste 4 ning 16 võrdlemisel. Konfiguratsioonis märgitud port

peab olema sama väärtusega, millega joonisel 14 kuvatud ülekandeprogrammis loodud soket nimega „socketC“ ühendatakse.

```
[Interface]
PrivateKey = OOznzhtgZ0wjJAl/IYRIFZo6Rx5AnBO26UcNiFJDYnE=
Address = 10.0.0.1/32
ListenPort = 33333
PostUp = iptables -A FORWARD -i %i -j ACCEPT; iptables -A
FORWARD -o %i -j ACCEPT; iptables -t nat -A POSTROUTING -o
eth0 -j MASQUERADE;
PostDown = iptables -D FORWARD -i %i -j ACCEPT; iptables -D
FORWARD -o %i -j ACCEPT; iptables -t nat -D POSTROUTING -o
eth0 -j MASQUERADE;

[Peer]
PublicKey = +HsYsIpCptA8ViR5vCP9JJxe80Dgi0otxvhI9H4ckmg=
AllowedIPs = 10.0.0.2/32
```

Joonis 16. Serveripoolne WireGuardi seadistus edastuslahendusel.

Prototüübi rakendamiseks teistes seadmetes on vaja klient- ja serverprogrammides määrata vastavalt serveri ja kliendi avalikud IP-aadressid P2P ühenduse loomiseks ning genereerida uued avalikud ning privaatsed võtmed.

4.4 Tulemused

Ühenduse toimimise kontrollimiseks saadetakse klientseadmest ping aadressile 10.0.0.1, mis on kliendi konfiguratsioonis märgitud ainukeseks aadressiks, millele päringuid tehes suunatakse andmed läbi VPN-i. Joonisel 17 on kuvatud testpäringute tulemused.

```
64 bytes from 10.0.0.1: icmp_seq=101 ttl=64 time=80.496 ms
64 bytes from 10.0.0.1: icmp_seq=102 ttl=64 time=85.766 ms
64 bytes from 10.0.0.1: icmp_seq=103 ttl=64 time=113.864 ms
64 bytes from 10.0.0.1: icmp_seq=104 ttl=64 time=117.225 ms
64 bytes from 10.0.0.1: icmp_seq=105 ttl=64 time=114.439 ms
64 bytes from 10.0.0.1: icmp_seq=106 ttl=64 time=102.474 ms
64 bytes from 10.0.0.1: icmp_seq=107 ttl=64 time=100.194 ms
64 bytes from 10.0.0.1: icmp_seq=108 ttl=64 time=104.922 ms
64 bytes from 10.0.0.1: icmp_seq=109 ttl=64 time=110.740 ms
Request timeout for icmp_seq 110
64 bytes from 10.0.0.1: icmp_seq=111 ttl=64 time=96.226 ms
64 bytes from 10.0.0.1: icmp_seq=112 ttl=64 time=92.674 ms
64 bytes from 10.0.0.1: icmp_seq=113 ttl=64 time=93.161 ms
64 bytes from 10.0.0.1: icmp_seq=114 ttl=64 time=132.361 ms
Request timeout for icmp_seq 115
64 bytes from 10.0.0.1: icmp_seq=116 ttl=64 time=100.093 ms
64 bytes from 10.0.0.1: icmp_seq=117 ttl=64 time=127.624 ms
64 bytes from 10.0.0.1: icmp_seq=118 ttl=64 time=87.512 ms
64 bytes from 10.0.0.1: icmp_seq=119 ttl=64 time=122.694 ms
```

Joonis 17. Testpäringute saatmistulemused.

Keskmine paketi liikumise aeg päringu alguspunktist WireGuard süsteemi, sealt omakorda ülekandeks loodud programmi, läbi tulemüüri aukude serverprogrammist edasi VPN serverrakendusse ning sama teed mööda tagasi on keskmiselt 100 millisekundit. Arvestades asjaolu, et testseadmete füüsiliseks vahekauguseks on umbes 1500 kilomeetrit, on saavutatud tulemus rahuldav. Joonisel 10 on võimalik tuvastada ka mõningaid kaduma läinud pakette, kuid UDP puhul on taoline nähtus normaalne, sest protokoll pakettide kohale jõudmist ei kontrolli. See asjaolu ühenduse kvaliteeti olulisel määral ei muuda, sest suurem osa pakette jõuavad sihtmärgini igal juhul.

5 Kokkuvõte

Käesoleva töö eesmärgiks oli luua WireGuard VPN ühendus kahe tulemüüri taga oleva ning teisendatud võrguaadressidega seadmete vahel. Tulemustest selgus, et samadel protokollidel töötavate tehnoloogiate kombineerimine on võimalik ning resultaat on silmaga nähtav. Töös läbivaks protokolliks, mida iga lahendusel kasutatud tehnoloogia toetas, oli interneti transpordikihi kasutajadatagrammi protokoll ehk UDP. Kuna WireGuard saadab andmeid eranditult ainult UDP pakettidega, oli võimalik kasutada soketeid, mille kaudu antud pakette transportida ning spetsiaalse JavaScripti teegi abil läbi tulemüüri suunata.

Loodud lahendust on võimalik edasi arendada nii töökindluse kui ka kasutajamugavuse aspektist. Hetkel lubab süsteem ainult konkreetsele IP-aadressile päringuid saates pakettide ümbersuunamist. Järgmise sammuna oleks tarvilik võimaldada kogu liikluse süsteemile edasi saatmine, et lahendust arvestatava VPN ühendusena otstarbekalt kasutada saaks. Samuti oleks võimalik edasiarendusena implementeerida ülekandeprogrammide automatiseeritud käivitamine või kasutajaliides, mis võimaldab ilma spetsiaalsete oskuste ning teadmisteta loodud süsteemi kasutada. Infrastruktuuri universaalsemaks muutmiseks on võimalus luua vaheserver, kuhu salvestatakse osapoolte aadressid ning mille abil saaks kasutada samasugust koodi erinevate ühenduste loomisel. Niisugusel juhul sisestatakse vaheserverisse salvestatud aadressid seadmetes asuvatesse koodidesse automaatselt ning identset implementatsiooni on võimalik kasutada kõikides seadmetes.

Täiendades töös kirjeldatud lahendust eelmainitud viisidel, saab kirjeldatud ideest luua asjakohase toote, millega küberturveteenuste turule siseneda. Arvestades kasutatud tehnoloogiate uudsust ning asjaolu, et produkt võimaldab lahendada aktuaalset probleemi, võib prognoositav kliendibaas olla märkimisväärne.

Töö autori teadmistepagas suurenes märkimisväärselt tehtud praktilise töö käigus. Lisaks kinnistatud ning uutele oskustele tarkvaraarenduse vallas omandati arusaam arvutivõrkude detailsemast toimimisest, sealhulgas IP-aadresside jagunemisest ning

portide toimimismehhanismist, ning küberturvalisusest. Eelmainitud teadmiste olemasolu on süsteemide arenduse puhul oluline, kuna suure pildi hoomamine võimaldab luua kvaliteetsemaid ning mitmeid aspekte arvestavaid lahendusi.

Kasutatud kirjandus

[1] J. M. Stewart ja D. Kinsey, *Network security, firewalls, and VPNs*, Third edition. Burlington, MA: Jones & Bartlett Learning, 2021.

[2] V. Mujović, „The History of VPN creation | Purpose of VPN“, *Le VPN*, aug 17, 2018. https://www.le-vpn.com/history-of-vpn/ (vaadatud apr 18, 2021).

[3] „Geo-blocking: how and why Netflix does it? – Skope Entertainment Inc“. https://skopemag.com/2019/03/06/geo-blocking-how-and-why-netflix-does-it (vaadatud apr 19, 2021).

[4] J. Frankenfield, „Silk Road Definition“, *Investopedia*. https://www.investopedia.com/terms/s/silk-road.asp (vaadatud apr 19, 2021).

[5] Pankaj, „Types of Virtual Private Network (VPN) and its Protocols“, *GeeksforGeeks*, apr 10, 2019. https://www.geeksforgeeks.org/types-of-virtual-private-network-vpn-and-its-protocols/ (vaadatud apr 21, 2021).

[6] „Types of VPN and VPN Protocols Explained - PrivacyEnd“. https://www.privacyend.com/vpn/vpn-protocols-explained/ (vaadatud apr 23, 2021).

[7] A. Harkness, „5 Common VPN protocols explained | NetMotion Software“, mai 15, 2019. https://www.netmotionsoftware.com/blog/connectivity/vpn-protocols (vaadatud apr 24, 2021).

[8] K. Hamzeh, G. Pall, W. Verthein, J. Taarud, W. A. Little, ja G. Zorn, „RFC 2637: Point-to-Point Tunneling Protocol (PPTP)“. https://www.hjp.at/doc/rfc/rfc2637.html (vaadatud apr 24, 2021).

[9] M. Feilner, *OpenVPN: building and integrating virtual private networks ; learn how to build secure VPNs using this powerful open source application*, 1. publ. Birmingham: Packt Publ, 2006.

[10] J. A. Donenfeld, „Performance - WireGuard“. https://www.wireguard.com/performance/ (vaadatud apr 24, 2021).

[11] J. A. Donenfeld, „WireGuard: fast, modern, secure VPN tunnel“. https://www.wireguard.com/ (vaadatud apr 24, 2021).

[12] C. Brown, „Visualising the Berkeley Sockets API“, lk 72.

[13] D. Anderson, „How NAT traversal works · Tailscale“, aug 21, 2020. https://tailscale.com/blog/how-nat-traversal-works/ (vaadatud apr 29, 2021).

Lisa 1 – Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks[1]

Mina, Viktoria Siigur[1]

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose „WireGuard tehnoloogial ning P2P ühendustel põhineva VPN infrastruktuuri arendus“, mille juhendajateks on Toomas Lepikult ning Janno Stern.
   1.1. reprodutseerimiseks lõputöö säilitamise ja elektroonse avaldamise eesmärgil, sh Tallinna Tehnikaülikooli raamatukogu digikogusse lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;
   1.2. üldsusele kättesaadavaks tegemiseks Tallinna Tehnikaülikooli veebikeskkonna kaudu, sealhulgas Tallinna Tehnikaülikooli raamatukogu digikogu kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.
2. Olen teadlik, et käesoleva lihtlitsentsi punktis 1 nimetatud õigused jäävad alles ka autorile.
3. Kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest ning muudest õigusaktidest tulenevaid õigusi.

17.05.2021

1 Lihtlitsents ei kehti juurdepääsupiirangu kehtivuse ajal vastavalt üliõpilase taotlusele lõputööle juurdepääsupiirangu kehtestamiseks, mis on allkirjastatud teaduskonna dekaani poolt, välja arvatud ülikooli õigus lõputööd reprodutseerida üksnes säilitamise eesmärgil. Kui lõputöö on loonud kaks või enam isikut oma ühise loomingulise tegevusega ning lõputöö kaas- või ühisautor(id) ei ole andnud lõputööd kaitsvale üliõpilasele kindlaksmääratud tähtajaks nõusolekut lõputöö reprodutseerimiseks ja avalikustamiseks vastavalt lihtlitsentsi punktidele 1.1. ja 1.2, siis lihtlitsents nimetatud tähtaja jooksul ei kehti.