hyväksymispäivä arvosana

arvostelija

Jaettujen hajautustaulujen käyttö ja turvallisuusongelmat

Valtteri Vuorikoski

Helsinki 5.11.2012

Pro gradu -tutkielma

HELSINGIN YLIOPISTO
Tietojenkäsittelytieteen laitos

HELSINGIN YLIOPISTO — HELSINGFORS UNIVERSITET — UNIVERSITY OF HELSINKI

Tiedekunta — Fakultet — Faculty	Laitos — Institution — Department
Matemaattis-luonnontieteellinen	Tietojenkäsittelytieteen laitos

Tekijä — Författare — Author
Valtteri Vuorikoski

Työn nimi — Arbetets titel — Title

Jaettujen hajautustaulujen käyttö ja turvallisuusongelmat

Oppiaine — Läroämne — Subject
Tietojenkäsittelytiede

Työn laji — Arbetets art — Level	Aika — Datum — Month and year	Sivumäärä — Sidoantal — Number of pages
Pro gradu -tutkielma	5.11.2012	66 sivua

Tiivistelmä — Referat — Abstract

Jaetut hajautustaulut (*Distributed Hash Table*, DHT) ovat rakenteisten vertaisverkkojen tyyppi, jotka mahdollistavat tiedon hajauttamisen ennalta määrittelemättömälle joukolle solmuja ilman keskitettyä koordinaatiota. Kuten usein vertaisverkoissa, kaikki verkon solmut osallistuvat tiedon haku- ja talletusprosessiin. Tällaisessa järjestelmässä on keskeistä, että kaikki verkon solmut noudattavat verkon protokollaa.

Solmut voivat poiketa protokollasta joko ohjelmistovirheiden vuoksi tai koska ne pyrkivät häiritsemään verkon toimintaa. Toisaalta verkon sisällöstä tai liikenteestä kiinnostuneet tahot voivat myös liittää verkkoon solmuja, joiden tavoite on lokittaa verkon liikennettä ja liikennöijiä. Nämä tarkkailijat voivat toimia muutoin pääosin protokollan mukaisesti. Hyökkäyksen suorittavia solmuja tarvitaan yleensä suuri määrä normaalikäyttöön verrattuna. Mahdollisuus liittää suuri määrä tällaisia solmuja perustuu osallistujien *heikkoon identiteettiin* sekä hankaluuteen varmistaa ulkopuolelta solmujen *oikea toiminta*.

Tutkijat ovat kehittäneet sekä identiteetin vahvistamiseen, että verkon ja solmujen toiminnan tilastolliseen tarkasteluun perustuvia *puolustusmenetelmiä*. Nämä menetelmät pyrkivät tunnistamaan hyökkäävät toimenpiteet joko havaitsemalla verkkoon liittyneet poikkeavat solmuryppäät tai pyrkimällä havaitsemaan poikkeamat protokollien invarianteista.

Maailmalla eniten käytetyt DHT:t perustuvat Kademlia-algoritmiin. Tätä käyttävät muun muassa eMule-tiedostonjako-ohjelman käyttämä Kad-verkko sekä kaksi erillistä BitTorrent-asiakasohjelmien käyttämää DHT:a. Näitä verkkoja vastaan on toteutettu useita käytännön hyökkäys- ja tarkkailumekanismeja sekä esitetty useita puolustusmekanismeja.

Tämä tutkielma tarkastelee tutkimuskirjallisuutta DHT:jen toiminnan ja turvallisuuden teoreettisesta perustasta sekä periaatteellisista hyökkäys- ja puolustusmekanismeista esitetyille järjestelmille. Lisäksi tutkielma tarkastelee spesifisesti Kademlia-algoritmin käyttöä käytännön sovelluksissa ja niihin sovellettavia hyökkäys- ja puolustusmekanismeja.

ACM Computing Classification System (CCS):
C2.1 Network Architecture and Design—Distributed networks
C2.2 Network Protocols—Applications
C2.4 Distributed Systems—Distributed databases
E.1 Data Structures—Distributed data structures
H3.4 Information Storage and Retrieval—Systems and software—Distributed systems

Avainsanat — Nyckelord — Keywords
Tietoverkot, vertaisverkot, DHT, Kademlia, hajautetut järjestelmät, turvallisuus

Säilytyspaikka — Förvaringsställe — Where deposited

Muita tietoja — övriga uppgifter — Additional information

Sisältö

1 Johdanto

Jaettujen hajautustaulujen (*Distributed Hash Table*, DHT) tavoite on nimensä mukaisesti tarjota mekanismi, jolla käyttäjä voi hakea avaimen perusteella siihen liittyvän arvon tietämättä ennakolta paikkaa, jossa arvo sijaitsee. Toisin kuin perinteisessä hajautustaulussa, arvot jakautuvat yhden koneen muistin sijasta usealle tietoverkon kautta toisiinsa kytketylle koneelle.

Perinteinen hajautustaulu sijaitsee yhden koneen muistissa ja hakuoperaatiossa algoritmi muodostaa hakuavaimesta *hajautusfunktiolla* muistiosoitteen, jossa arvon pitäisi sijaita. Tämän operaation aikavaatimus on tyypillisesti $O(1)$. Perinteinen hajautustaulu olettaa kuitenkin, että taulu koostuu jatkuvasta muistialueesta, joka on kokonaisuudessaan sekä muuttumaton (mikäli käyttäjä ei muuta sitä eksplisiittisesti), että aina saatavilla.

Teoriassa olisi mahdollista toteuttaa suora hajautettu versio tästä ajatuksesta, mutta tämä edellyttäisi, että sen toimintaan osallistuvien koneiden joukko olisi ennalta tunnettu ja että kaikki osallistuvat koneet olisivat aina saavutettavissa. Koska tällainen lähestyminen ei kuitenkaan ole käytännössä kovin hyödyllinen, DHT-järjestelmien perusominaisuuksiin kuuluu, että osallistuvien koneiden joukko voi muuttua järjestelmän käytön aikana. Tämä tapahtuu ilman, että järjestelmän toiminta häiriintyy tai tietoa katoaa.

DHT-järjestelmien tutkimuksen historialliset lähtökohdat olivat usein varhaiset peer-to-peer (P2P) -sovellukset kuten Napster ja Gnutella[RD01, SMK+01]; toisaalta tavoite oli luoda järjestelmiä, joissa kaikki toimijat ovat keskenään samanarvoisia ja keskenään vaihdettavia ja toisaalta välttää aiempien P2P-sovellusten vaatimukset keskitetyistä palvelimista sekä skaalautumisongelmat. Näin ollen DHT-järjestelmien keskeisiä tavoitteita ovat desentralisaatio, skaalautuvuus suurelle käyttäjäjoukolle ja saavutettavuus osallistujien vaihtuessa. Lisäksi järjestelmien kehittäjät ovat halunneet välttää DNS:n kaltaista rajoitettua, hierarkkista hajautusta, joten DHT:jen hakuavaimet ovat vapaamuotoisia[SMK+01]. Toiveisiin käytännön järjestelmille liitetään yleensä myös *häirinnän sietokyky* sekä käyttäjien *anonymiteetti*.

Toisin kuin perinteisessä hajautustaulussa, P2P-perustaisissa DHT:issa avaimesta ei voi päätellä suoraan sitä vastaavan arvon fyysistä sijaintia. Tyypillisessä DHT-järjestelmässä *avainavaruus* on jaettu osallistuvien koneiden kesken. Yleensä avainta hakeva käyttäjä ei tunne kaikkien osallistujien osoitteita ja avainavaruuksia vaan vain muutamia naapureita. Käyttäjän kone päättelee avaimesta parhaan naapurin

ja kysyy tältä osoitteita sellaiselle osallistujalle, jolla pitäisi olla tarkempi tieto arvon sijainnista. Tämä prosessi toistuu, kunnes avainta hallussaan pitävä kone löytyy. Yleensä tämä hakuprosessi vaatii logaritmisen ajan[RD01, SMK+01, MM02].

DHT:jen ajatus on varsin elegantti, mutta käytännön järjestelmät joutuvat kohtaamaan paljon tilanteita, joihin alkuperäiset algoritmien kuvaukset eivät ota kantaa. Näitä ovat joko vahingossa tai tarkoituksella väärin käyttäytyvät osallistujat, yritykset manipuloida verkon toimintaa tiedon muuttamiseksi tai piilottamiseksi käyttäjiltä, palvelunestohyökkäykset osallistujia kohtaan sekä pyrkimykset selvittää osallistujien identiteetit.

Koska järjestelmien pyrkimys on ollut pyrkiä eroon kaikenlaisista keskitetyistä auktoriteeteista, niihin ei myöskään kuulu keskitettyä käyttäjäidentiteettien takaajaa kuten perinteisissä, X.509:n kaltaisissa keskitetyissä avaininfrastruktuureissa (*public key infrastructure*, PKI). Kuka tahansa voi siis muodostaa rajattoman määrän käyttäjäidentiteettejä ilman, että minkään muun tahon tarvitsee taata niitä tai että mikään muu taho voi rajoittaa niiden luomista. Tästä seuraa merkittäviä rajoituksia DHT-järjestelmien kyvylle puolustautua hyökkäyksiltä: vaikka yhden käyttäjän vahingollisen toiminnan voisi estää, mikään ei estä häntä luomasta uutta identiteettiä ja jatkamasta samaa toimintaa. Tällaisessa tilanteen täydellinen välttäminen on itse asiassa teoreettisesti mahdotonta, kuten J.R. Douceur osoitti perustavanlaatuisessa artikkelissaan “The Sybil Attack” vuonna 2002[Dou02].

Näennäisistä puutteista huolimatta muutamat DHT-sovellukset ovat saavuttaneet suhteellisen laajan käyttäjäkunnan. Näitä ovat useisiin BitTorrent-tiedostonjakojärjestelmän asiakasohjelmiin liittyvät kaksi erilaista DHT-verkkoa sekä eMule-tiedostonjako-ohjelman käyttämä Kad-DHT. Kaikki kolme perustuvat vuonna 2002 julkaistuun Kademlia-algoritmiin[MM02].

Vaikka tutkielmassa viitataan käytännön sovelluksina lähinnä tiedostonjakosovelluksiin, DHT:t ovat sovellettavissa myös muihin viestintäsovelluksiin. Piratismin torjuntaa tuotteena myyvät yritykset ovat kohdistaneet tarkkailutoimia Kad-verkkoon[SENB09], mutta toisaalta eräiden hallitusten väitetään tilanneen laajamittaisia Twitteriin ja vastaaviin keskitettyihin sosiaalisiin verkostoihin keskittyneitä propagandakampanjoita ja sensuuritoimia[Roc11, Elm12]. Näin ollen myöskään mahdollisilta tulevaisuuden DHT-pohjaisilta sosiaalisten verkostojen sovelluksilta tuskin tulee myöskään puuttumaan mittavillakin resursseilla varustettuja hyökkääjiä. DHT:jen turvallisuuden tarkastelun relevanssi ei siis rajoitu ainoastaan tiedostonjakosovelluksiin.

Tämä tutkielma tarkastelee, millaisia nämä käytännössä laajalle levinneet DHT-järjestelmät ovat, miten niitä on tutkittu ja mitattu, millaisia uhkia niihin käytännössä kohdistuu sekä miten uhkia on pyritty torjumaan. Tutkimus on olennaista sekä nykyisten toteutusten toiminnan parantamisen kannalta, että mahdollisten uusien, muuhun kuin tiedostonjakoon tarkoitettujen sovellusten tuottamisen kannalta. Erityisesti jälkimmäisessä tapauksessa on toivottavaa tehdä asiat heti aluksi turvallisesti, koska vertaisverkkojen hajautetun luonteen vuoksi suurten muutosten tekeminen protokolliin on hankalaa jälkikäteen.

Tutkielma keskittyy Kademlia-algoritmin sovelluksiin, mutta viittaa taustana myös vanhempaan Chord-algoritmiin[SMK+01] sekä turvallisuuskysymyksissä anonymiteettiin pyrkivään DHT:n kaltaiseen Freenet-verkkoon[CSTV10]. Tutkielma perustuu pääasiassa alan tieteellisissä lehdissä ja konferensseissa julkaistuihin tutkimusartikkeleihin, mutta viittaa taustatietoina myös muun muassa ohjelmistojen tekijöiden itsensä tuottamiin mittauksiin ja kuvauksiin heidän ohjelmistojensa toiminnasta.

Luku 2 esittelee DHT-järjestelmien teoriaa ja käytännön toteutuksia. Luku 3 tarkastelee yleisellä tasolla DHT:jen turvallisuutta, turvallisuusuhkia ja luottamussuhteiden muodostamista käyttäjien välille. Luku 4 siirtyy käytännön toteutuksiin ja esittelee em. Kademlia-pohjaisten järjestelmien käyttöä, liikennemääriä ja sisältöä. Luvussa 5 tarkastellaan menetelmiä, joilla DHT:jen liikennettä ja käyttäjämääriä voidaan tutkia. Luku 6 tarkastelee erityisesti Kademlia-pohjaisiin järjestelmiin kohdistuvia hyökkäyksiä sekä teoreettisia että käytännössä toteutettuja puolustusmekanismeja. Luku 7 tekee yhteenvedon tutkielman sisällöstä ja esittää johtopäätökset.

2 Distributed Hash Tablejen (DHT) teoria ja käytäntö

Tämä luku esittelee DHT-järjestelmien teoriaa, käsitteitä ja algoritmeja, niiden roolia tutkimusympäristöissä ja suurelle yleisölle sovellettuna sekä toteutuksien kohtaamia ongelmia.

2.1 Rakenteettomat ja rakenteiset verkot

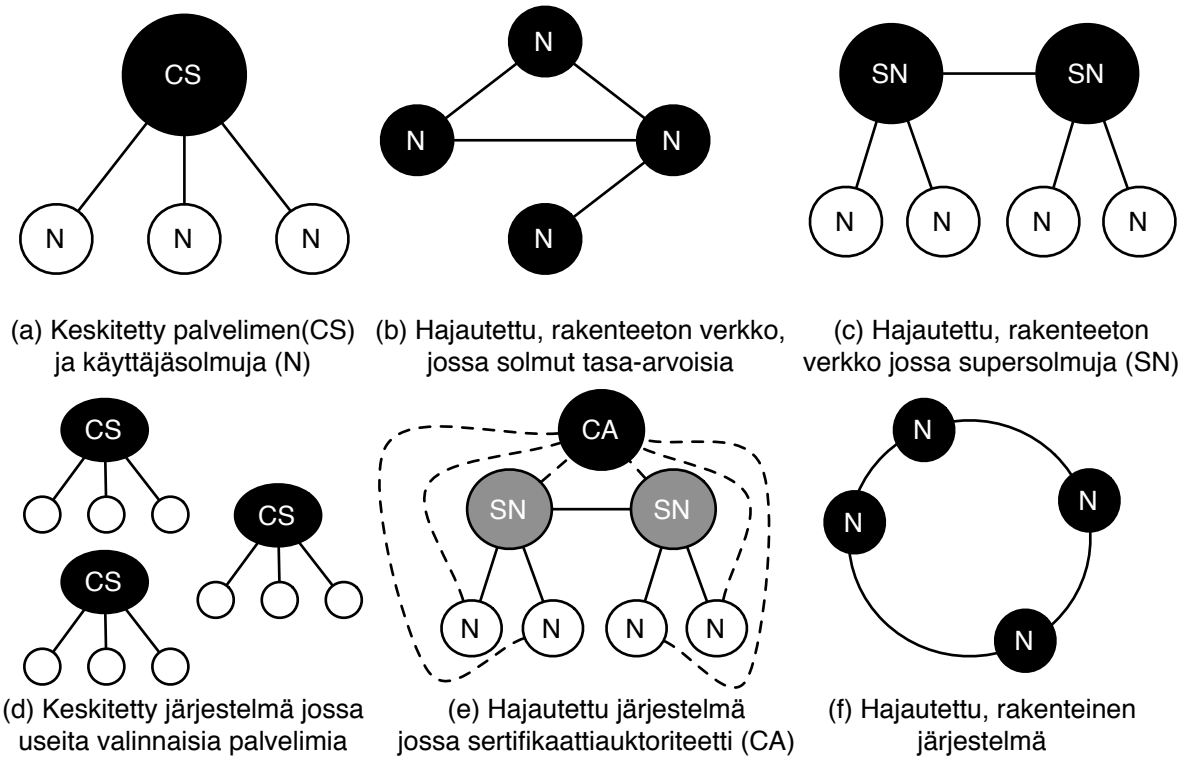
Tiedon etsimisen kannalta P2P-järjestelmät jaetaan *rakenteettomiin* (unstructured) ja *rakenteisiin* (structured) tyyppeihin. Rakenteettomassa verkossa tietyn *hakuavai-*

men sijaintia ei ole ennalta määrätty, vaan se voi sijaita millä tahansa verkon solmulla. Yksinkertaisessa rakenteettomassa järjestelmässä kysely välitetään solmulta toiselle yleislähetyksenä ja hakuavaimeen sopivaa tietoa omistavat solmut kertovat tästä kysyjälle[SG05]. Tällä menetelmällä ei ole taattua, että avain löytyy vaikka se olisi verkossa (kysely ei koskaan saavu avaimen haltijalle) ja kyselyviestien määrä kasvaa lineaarisesti verkon reunojen määrän suhteen. Esimerkiksi Gnutella-verkko perustui tähän menetelmään ja kasvaessaan muuttui lähes käyttökelvottomaksi suuren kyselyviestiliikenteen johdosta[RF02]. Rakenteettomassa verkossa kaikki solmut voivat olla tasapuolisessa asemassa, tai haku- ja reititystehtävät voivat olla kokonaain tai osittain rajoitettuja ainoastaan tietyille *supersolmuille*.

DHT:t edustavat rakenteista mallia, missä *avaimen sijainti on ennalta määrätty* (kuten hajautustaululta voi odottaa). Mikäli tietty avain on julkaistu verkossa, se on periaatteessa aina löydettävissä ja tämä prosessi vaatii vain *reitin* löytämisen kyseisen avaimen hallussapitäjälle. Verkon solmut voivat siis tehdä itse avaimen perusteella päätöksen mille naapurille (jos millekään) hakuviesti pitää lähettää seuraavaksi sen sijaan, että lähettäisivät sen kaikille.

Kuva 1 havainnollistaa eri tyyppisiä verkkorakenteita. Kohdat (a) ja (d) ovat edellisessä luvussa kuvattuja keskitettyjä järjestelmiä, joissa solmut löytävät toisensa keskitetyn palvelimen kautta (mikäli näitä on useita kuten kohdassa (d), käyttäjät valitsevat manuaalisesti käyttämänsä palvelimen). Kohdat (b) ja (c) edustavat rakenteettomia verkkoja, joissa tietoa välitetään joko kaikkien solmujen välillä tai vain jollakin kriteerillä valittujen “hyvien” solmujen välillä. Kohta (e) kuvaa järjestelmää, jossa keskitetty sertifikaattiauktoriteetti tarjoaa muutoin hajautetulle järjestelmälle allekirjoituspalveluita; tätä tapausta käsitellään tarkemmin luvussa 3.1. Kohta (f) kuvaa rakenteista järjestelmää, jossa solmut sijoittuvat protokollan määräämin asemiin verkossa.

Vuosina 2001–2002 tutkijat kehittivät useita erilaisia DHT-algoritmeja. Tämä luku keskittyy näistä paljon akateemisesti tukittuun Chordiin[SMK+01] sekä tiedostonjakoohjelmissa käytettyyn Kademliaan[MM02]. Toisaalta tutkimus on jatkunut myös rakenteettomien verkkojen suhteen, ja tutkijat ovat esittäneet myös näille verkoille tehokasta ja kattavaa läpikäyntimenetelmää[TKLB07]. Rakenteettomia verkkoja ei kuitenkaan käsitellä tässä tutkielmassa tarkemmin.



(a) Keskitetty palvelimen(CS) ja käyttäjäsolmuja (N)

(b) Hajautettu, rakenteeton verkko, jossa solmut tasa-arvoisia

(c) Hajautettu, rakenteeton verkko jossa supersolmuja (SN)

(d) Keskitetty järjestelmä jossa useita valinnaisia palvelimia

(e) Hajautettu järjestelmä jossa sertifikaattiauktoriteetti (CA)

(f) Hajautettu, rakenteinen järjestelmä

Kuva 1: Eri tyyppisiä P2P-verkkoja ja niiden luottamussuhteita. Mustien solmujen täytyy välittää luotettavasti tietoa ja toimia yhdessä muiden solmujen kanssa. Valkoiset solmut eivät välitä tietoja. Kohdassa (e) harmaat solmut välittävät tietoja, mutta tieto ja/tai solmujen identiteetit ovat kryptografisesti allekirjoitettuja. Katkoviivat ilmaisevat sertifiointisuhteita.

2.2 Verkon rakenne ja tiedon hakuprosessi

Seuraavassa kuvaillaan taustatietona avaimen tallentamisen ja haun perusteita Chordissa ja Kademliassa. Tarkoitus on selvittää toiminnan teoreettinen perusta. Protokollien teknisiä yksityiskohtia käsitellään tarkemmin tarpeen mukaan myöhemmissä luvuissa.

Kuvaukset esittelevät algoritmien perusajatukset, joista saatetaan joissain tapauksissa optimointina tai muista syistä poiketa käytännön toteutuksista. Näitä poikkeuksia ja niiden vaikutuksia käsitellään myöhemmissä luvuissa.

2.2.1 Chordin toiminta

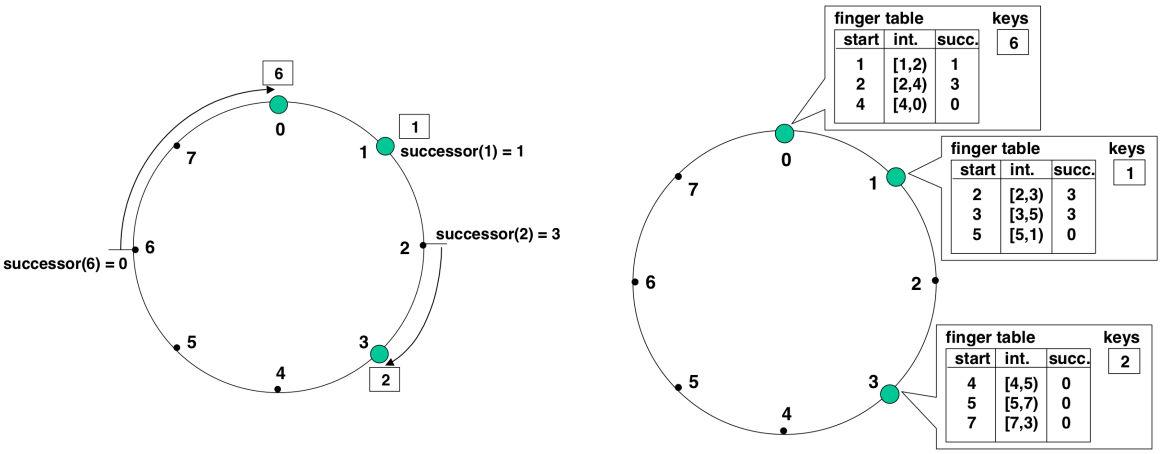
Chord-verkon rakenne on käsitteellisesti yksinkertainen, ja se soveltuu kuvaamaan DHT:jen perustoiminallisuutta, eli verkkoon liittymistä, avainten jakamista koneiden kesken sekä hakuprosessia.

Verkkoon liittyvä solmu valitsee itselleen m-bittisestä avaruudesta oletettavasti uniikin tunnisteen i[SMK+01]. Tunniste voi pohjautua koneen IP-osoitteeseen tai johonkin muuhun pysyvään tunnisteeseen tai se voi olla satunnainen. Tunnisteet on järjestetty *tunnisteympyräksi*, eli ympyräksi joka koostuu luvuista $0 \leq i \leq 2^m - 1$.

Kullakin talletettavalla arvolla on samoin m-bittinen avain k, jota pitää hallussaan avaimen *seuraajasolmu*. Tämän solmun vastuualueeksi määritellään kaikki avaimet $k \leq i$, eli avaimet, jotka ovat numeerisesti pienempiä tai yhtä suuria kuin solmun tunniste. Toisin sanottuna avaimen k seuraajasolmu on avaimen arvosta seuraava solmu kuljettaessa tunnisteympyrää myötäpäivään.

Jokainen solmu tietää seuraajansa IP-osoitteen, joten tässä vaiheessa olisi jo mahdollista hakea tietoa verkosta. Tämä kuitenkin vaatisi kyselyn jokaiselle solmulle lähtö- ja kohdesolmun välissä, joten menetelmä olisi liian tehoton. Tästä syystä jokainen solmu ylläpitää myös *sormitauluja*, jossa jokainen rivi ("sormi") indeksillä i viittaa seuraajaan, joka on vähintään 2^{i-1} tunnisteen päässä kyseisestä solmusta. Seuraamalla näitä viitteitä hakuprosessissa etäisyys lähtö- ja kohdesolun välillä puolittuu joka askeleella lineaarisen läpikäynnin sijaan, eli toisin sanoen haun vaativuus on $O(\log n)$.

Kuva 2 esittää Chordin tunnisteympyrän rakennetta yksinkertaisessa verkossa, jonka tunnisteavaruus on kolmebittinen. Kohta (a) esittää verkon, jossa on kolme solmua ja kolme tallennettua avainta. Avainta 1 kuuluu suoraan solmulle, jonka tun-



(a) Chordin osoiteavaruuden rakenne (b) Chordin sormitaulut

Kuva 2: (a) Chordin osoiteavaruuden rakenne, kun käytössä ovat kolmibittiset tunnisteet ja verkkoon liittynyt kolme solmua. (b) Kunkin solmun sormitaulujen sisältö. Kuvien lähde [SMK+01].

niste on 1. Avainta 2 pitää hallussaan sen seuraajasolmu 3. Avainta 6 pitää myös hallussaan sen seuraajasolmu, mutta tässä tapauksessa haku kiertyy tunnisteympyrän ympäri ja avain päätyy solmulle 0. Kohta (b) esittää solmujen sormitaulut.

Uuden solmun liittyessä verkkoon sen tulee selvittää vähintään oma edeltäjänsä ja seuraajansa tunnisteympyrällä. Lisäksi sekä seuraajan että edeltäjän pitää päivittää tietonsa siitä, kuka on niiden uusi seuraaja/edeltäjä (solmut ylläpitävät tietoa omasta edeltäjästään liittymis- ja poistumisprosessin helpottamiseksi, vaikka sitä ei normaalisti käytetä hakuprosessissa). Uuden solmun käsittelemään tunnistealueeseen viittaavien olemassa olevien solmujen sormitaulut pitää myös päivittää, joskaan virheelliset rivit sormitauluissa eivät estä algoritmin toimintaa vaan vain hidastavat sitä. Haku saattaa joutua esimerkiksi käymään tunnisteympyrää läpi lineaarisesti jonkin matkaa, jos sormitaulun mukaan ei päästä niin lähelle kohdetta kuin olisi täydellisessä verkossa mahdollista.

Viimeinen askel liittymisprosessissa on uuden solmun vastuualueeseen kuuluvien avainten siirto uudelle tulijalle. Siirron kohteena ovat avaimet tunnisteilla $i_{n-1} < k \leq i_n$, missä i_{n-1} on edeltäjän tunniste ja i_n liittyneen solmun tunniste. Siirron toteutus on yleensä Chord-verkkoa käyttävän sovelluksen vastuulla, eikä varsinaisesti liity itse Chord-protokollaan[SMK+01].

Chordin voi toteuttaa sekä *iteratiivisesti* että *rekursiivisesti*. Iteratiivisessa toteutuksessa alkuperäinen haun suorittaja pyytää seuraajasolmua kertomaan, mikä on

siitä seuraava solmu johon pitäisi ottaa yhteyttä ja suorittaa yhteydenoton kuhunkin seuraajaan itse kunnes saa yhteyden avaimen haltijaan. Rekursiivisessa toteutuksessa taas kukin seuraaja ottaa yhteyden omaan seuraajansa kunnes avaimen haltija löytyy, eli haun suorittaja kommunikoi vain yhden muun solmun, oman seuraajansa, kanssa[SMK+01]. Toteutustavalla on merkitystä mm. verkon anonymiteetin ja kuormituksenkeston suhteen. Iteratiivisessa mallissa kysyjän ja arvon haltijan välissä olevat solmut kommunikoivat suoraan kysyjän kanssa, eli saavat tietoonsa kysyjän IP-osoitteen. Toisaalta taas rekursiivisessa mallissa kyselyn kuljettaminen eteenpäin vaatii, että kaikki välillä olevat solmut kuljettavat kyselyä eteenpäin. Lisäksi niiden tulee myös pitää yhteys seuraajaansa auki vastauksen palauttamista varten.

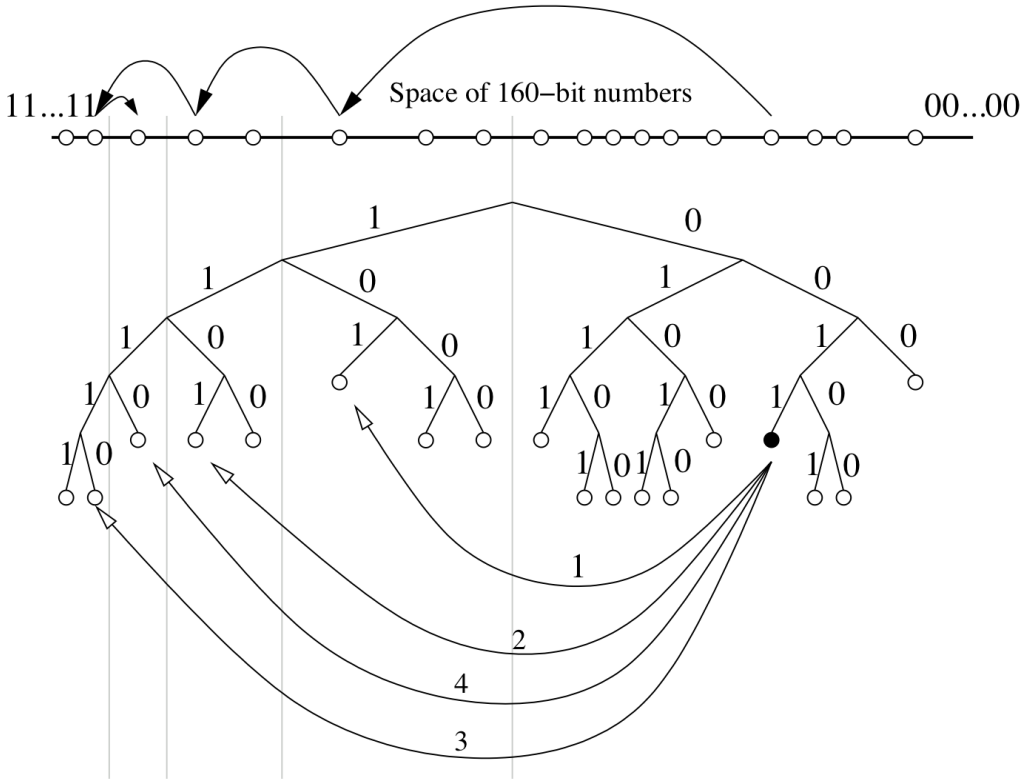
2.2.2 Kademlian toiminta

Kademlia seuraa DHT:jen yleisiä periaatteita eli verkko on rakenteinen, haut siitä vaativat $O(\log n)$ ajan ja se ei vaadi mitään keskitettyä komponenttia. Kuten Chordissa, avaimet ovat rakenteettomia ja avainten sijoittelu perustuu solmujen tunnisteisiin. Kademlian hakuprosessi on sekä alkuperäisen määritelmän[MM02] mukaan, että suurimmassa käytännön toteutuksessa Kadissa iteratiivinen[SBEN07] eli alkuperäinen kysyjä saa kyselyihinsä vastauksena lähempänä avainta olevien solmujen osoitteita (tai avaimen arvon jos se löytyy) ja lähettää näille itse kyselyt jatkoreitityksestä.

Rakenne ja hakuprosessi ovat hieman vähemmän intuitiivisia kuin Chordissa. Solmut muodostavat binääripuun, missä solmu tunnistetaan sen lyhimmän uniikin etuliitteen pohjalta, jonka sen tunnisteesta voi muodostaa. Esimerkiksi kuvassa 3 näkyy mustalla solmu, jonka tunnisteen lyhin uniikki etuliite binäärinä on 0011. Lisäksi protokolla takaa, että kaikki solmut tietävät ainakin yhden solmun IP-osoitteen jokaisessa epätyhjässä alipuussa, johon solmu itse ei kuulu.

Avaimet jaetaan solmujen kesken solmun tunnisteen ja avaimen etäisyyden perusteella. Etäisyyden mitta Kademliassa on *XOR-metriikka* $d(x, y) = x \oplus y$, eli tunniste ja avain tulkitaan kokonaisluvuiksi, niiden välillä lasketaan biteittäin exclusive-or. Tuloksena saatu kokonaisluku on haettu etäisyys. Täysin täytetyssä puussa tämä metriikka vastaa suoraan sen alipuun korkeutta, joka sisältää sekä avaimen että solmun. Lisäksi toisin kuin Chordin yksisuuntaisesti kuljettava ympyrä, XOR-metriikalla mitatut etäisyydet ovat symmetrisiä ($\forall x, y : d(x, y) = d(y, x)$) [MM02].

Käytettäessä m-bittisiä avaimia jokainen solmu ylläpitää listaa, jossa kaikille i välillä



Kuva 3: Kademlia-verkon rakenne ja hakuprosessin eteneminen. Kuvan lähde [MM02].

$0 \leq i < m$ on k alkion lista, joka sisältää naapurisolmuja, joiden etäisyys d listan ylläpitäjästä on välillä $2^i < d < 2^{i+1}$. Kussakin alilistassa on enintään k alkiota, jotka sisältävät naapurin IP-osoitteen, UDP-portin ja tunnisteen. Näitä alilistoja kutsutaan *k-ämpäreiksi*. Alkiot on järjestetty sen mukaan, milloin alkiota vastaavaan naapuriin viittaava viesti on viimeksi nähty.

Listan viimeinen alkio vastaa naapuria, johon liittyvän paketin solmu on viimeksi nähnyt. Kun ämpäri on täynnä (eli sisältää k alkiota) ja solmu saa viestin tuntemattomalta naapurilta, se tarkastaa ensin `PING`-viestillä listan ensimmäisen solmun (eli solmun, jonka viestistä on kulunut eniten aikaa) hengissäolon. Mikäli vanha solmu on hengissä, se pidetään listan alussa ja uusi tulokas jätetään huomiotta; muutoin vanha solmu poistetaan listalta ja uusi tulokas lisätään siihen.

Tämä järjestely perustuu Gnutellasta saatuihin tutkimustuloksiin, joiden perusteella annetun solmun todennäköisyys pysyä tavoitettavana jatkossa kasvaa käytännössä sen mukaan, kuinka pitkään se on ollut tavoitettavissa tähän mennessä[SGG02]. Koska algoritmi priorisoi pisimpään hengissä olleita naapureita, sillä on suurin todennäköisyys pitää kyseisessä ämpärissä jatkossakin hengissä pysyvien naapurien

yhteystietoja. Toinen etu on, että vanhojen naapurien suosiminen hankaloittaa verkon häiritsemistä tulvittamalla reititystauluja uusilla naapureilla, koska näitä uusia tulijoita ei oteta mukaan jo täynnä oleviin ämpäreihin[MM02].

Avaimen haussa solmu lähettää `FIND_VALUE`-kutsuja naapureille, jotka ovat XOR-metriikan perusteella avainta lähimmässä k-ämpärissä. Solmu voi lähettää useita kutsuja rinnakkain sen sijaan, että odottaisi aina edelliseen vastausta. Tällöin vältytään odottamasta turhaan verkosta poistuneiden naapurien vastauksia ja vastauksia, jotka eivät anna lisätietoa seuraavasta askelesta. Mikäli joku vastaaja pitää hallussaan avaimeen liittyvää tietoa, se palauttaa vastauksena sen. Vastaajat, jotka tietävät avainta lähempänä olevan solmun lähettävät sen osoitteen, eli käytännössä solmun, joka on yhtä tasoa alempana osoitteiden muodostamassa puussa. Koska jokainen reititysaskel etenee puussa yhden tason alaspäin, algoritmi toimii ajassa $O(\log n)$.

Avaimen talletus toimii pitkälti kuten sen nouto, mutta `FIND_VALUE`-kutsun sijaan solmu käyttää `FIND_NODE`-kutsua löytääkseen k solmua, jotka ovat lähinnä avaimen tunnistetta. Kun solmut on löydetty, arvo talletetaan niille erillisellä `STORE`-kutsulla.

Jotta avaimet tietoineen eivät katoaisi verkosta solmujen poistuessa tai kun uudet, lähempänä avainta sijaitsevat solmut liittyvät verkkoon, tiettyä avainta hallussapitävät solmut uudelleenjulkaisevat sen kerran tunnissa. Tämä tapahtuu normaalilla talletusmekanismilla, eli ensin hakemalla lähimmät k solmua `FIND_NODE`-kutsulla ja sen jälkeen tallettamalla `STORE`-kutsulla. Optimointina kukin solmu nollaa uudelleenjulkaisulaskurinsa annetulle avaimelle vastaanottaessaan `STORE`-kutsun, jolloin yleensä kukin avain uudelleenjulkaistaan vain kerran tunnissa koko verkon laajuisesti.

Liittyäkseen verkkoon uuden solmun tarvitsee tietää jonkin jo verkkoon kuuluvan solmun IP-osoite. Tällaisen osoitteen löytäminen on järjestettävä jollakin järjestelmän ulkoisella menetelmällä. Liittyessään solmu ensin suorittaa haun omalla tunnisteellaan ja selvittää omat lähimmät naapurinsa verkossa. Sen jälkeen se tekee haun yhdellä jokaista sitä itseään puussa ylempänä olevaan alipuuhun kuuluvalla tunnisteella täyttääkseen k-ämpärinsä. Tämän prosessin jälkeen uusi solmu tuntee ainakin yhden naapurin kaikista alipuista, joihin se ei itse kuulu[MM02].

Kuva 3 näyttää haun etenemisen tapauksessa, jossa solmu 0011 etsii solmun, joka käsittelee etuliitettä 1100. Askeleessa 1 kysyjä lähettää viestin solmulle 101, jonka se tuntee 1-alkuisen alipuun edustajana käynnistysvaiheen perusteella. Hauissa 2–3 kysyjä saa yhä läheisimpien alipuiden edustajilta uusia reititystietoja. Näiden pe-

rusteella se lopulta saa osoitteen solmulle, jolle etuliite 1100 kuuluu ja kommunikoi tämän kanssa vaiheessa 4.

2.3 DHT:t reaalimaailmassa

DHT:ihin liittyvän tutkimuksen suuresta määrästä huolimatta DHT-pohjaisia järjestelmiä on laajalti käytössä suhteellisen vähän.

Chord ei tiettävästi ole laajassa käytössä missään, joskin siitä on olemassa muutamia avoimen lähdekoodin toteutuksia. Hajautettu välimuistijärjestelmä Coral CDN käyttää sisäisesti DHT-pohjaista mekanismia välimuistiin tallennettujen tiedostojen löytämiseen. Joskin Coralilla on paljon käyttäjiä, järjestelmä näkyy heille HTTP-välityspalvelimena ja järjestelmän sisäinen DHT-mekanismi on vain palvelinympäristön sisäisessä käytössä[Fre10].

Tällä hetkellä kaikki laajasti tunnetut julkiset DHT-verkot käyttävät Kademliaa. Näitä verkkoja on kolme: eMule-tiedostonjako-ohjelman seuraajineen käyttämä *Kad*, Vuze-BitTorrent-asiakasohjelman oma DHT-toteutus sekä useiden BitTorrent-asiakasohjelmien (mukaan lukien Vuze johon on asennettu sopiva lisämoduuli) käyttämä *BitTorrent Mainline DHT*-verkko[Ste09, Loe08]. Näitä verkkoja käsitellään tarkemmin luvussa 4.

Vuonna 2000 aloitettu Freenet-projekti käyttää version 0.7 jälkeen projektia varten kehitettyä järjestelmää, joka perii ominaisuuksia DHT:sta, mutta esimerkiksi annetulla avaimella ei ole samassa merkityksessä kiinteä sijoituspaikkaa kuten varsinaisissa DHT:issa. Järjestelmä on suunniteltu erityisesti turvallisuutta ja anonymiteettia silmällä pitäen, ja uusimissa versioissa kehittäjät kehottavat käyttäjiä käyttämään ohjelmistoa ns. darknet-moodissa. Tässä tilassa yhteyksiä solmujen välillä ei luoda automaattisesti, vaan kunkin solmun ylläpitäjän tulee saada kutsu ainakin yhdeltä olemassa olevan solmun ylläpitäjältä. Toisin kuin useimmissa muissa verkoissa, tavoitteena ei siis ole yhden maailmanlaajuisen verkon vaan useiden pienempien puolisalaisten verkkojen luominen[CSTV10].

Freenet herätti alkuperäisen julkistuksensa aikaan paljon kiinnostusta, mutta sittemmin siihen liittyvä tutkimus on ollut vähäistä, eikä laajasta käytöstä ole merkkejä julkisuudessa. On epäselvää, johtuuko tämä käyttäjien vähäisyydestä vaiko siitä, että se on onnistunut tehtävässään pitää verkot salaisina.

3 Luottamus ja turvallisuus DHT:issa

Tämä luku käsittelee luottamuksen ja turvallisuuden käsitteitä tietoverkoissa, sekä teoreettisella tasolla DHT:ihin vaikuttavia turvallisuuden tekijöitä. Spesifisesti Kademlian ja sen sovellusten ongelmakohtia turvallisuuden suhteen käsitellään erikseen luvussa 6.

3.1 Luottamuksen ja turvallisuuden käsitteet

Perinteisesti "jonkun tunteminen" on tarkoittanut, että kaksi osapuolta ovat tavanneet keskenään tai ainakin olleet jonkinlaisessa ihmisten välisessä reaaliaikaisessa kanssakäymisessä (esimerkiksi puhelimen välityksellä) ja näin muodostaneet käsityksen toistensa identiteetistä "intuitiivisella" perustalla. Sen sijaan perinteisessäkin elektronisessa viestinvälityksessä, kuten sähköpostissa, on yleistä, että osapuolet eivät ole tavanneet toisiaan tai muuten olleet yhteydessä keskenään ja tuntevat toisensa vain *digitaalisten identiteettien* perusteella. Toisin sanottuna tunnistaminen tapahtuu sähköpostiosoitteen ja jossain määrin kirjoitustyylin perusteella, mutta koska osapuolilla on joka "vuorolla" aikaa vastata, on mahdollista asetella sanansa ja käyttää esimerkiksi ulkoisia tietolähteitä huomattavasti laajemmin kuin puheyhteydessä.

Siltikin henkilöiden välisessä kommunikaatiossa käsitys luottamuksesta ja identiteetistä on mahdollista muodostaa kohtuullisen hyvin pitkään jatkuneen kommunikaation harkintakyvylle antamien vihjeiden kautta. Vertaisverkoissa tilanne on toisenlainen. Käyttäjiä edustavat yleensä puoliautonomiset ohjelmistot, esimerkiksi taustalla ajautuvat tiedostonjakosovellukset, jotka usein suorittavat toimintoja, joista varsinainen käyttäjä ei ole edes tietoinen. Näihin saattaa kuulua osanotto DHT-järjestelmään, jonka toiminnan käyttäjä saattaa tuntea vain hyvin korkealla tasolla ja josta ohjelmisto ei välttämättä kerro juurikaan.

Tällaisessa automaattisessa toiminnassa käyttäjää edustavan ohjelmiston käyttämän identiteetin ei tarvitse olla sidottu käyttäjään itseensä eikä pitkäaikaisesti pysyvä. Anonymiteetin osalta tämä saattaa olla jopa käyttäjän kannalta toivottu ominaisuus. Lisäksi ohjelmiston käyttämien identiteettien määrää ei ole välttämättä järjestelmän puolesta rajoitettu, ja niitä voi olla käytössä useita samaan aikaan. Tästä syystä käyttäjien tunnistamiseen ei ehkä kuulu, eikä välttämättä voikaan kuulua, minkäänlaista käyttäjän itsensä vuorovaikutusta, vaan tunnistus ja luottamussuhteiden hallinta eri ohjelmistoagenttien käyttämien identiteettien välillä on täysin

automaattista.

Vertaisverkkoympäristössä solmujen vaihtuvuus on suuri, joten esimerkiksi IP-osoitteeseen perustuva tietyn solmun jatkuva tunnistaminen ei todennäköisesti ole luotettavaa. Lisäksi täysin hajautetuissa verkoissa, joita käytännössä DHT-toteutukset ovat, tietyn solmun löytäminen tunnisteen perusteella edellyttää, että kysyjän ja kohteen välisellä "matkalla" olevat solmut toimivat käytössä olevan protokollan mukaan, eli muuttamatta välittämiensä viestien sisältöä tai palauttamatta harhaanjohtavaa tietoa. Toisin sanoen sen lisäksi, että kommunikaation päätepisteiden keskinäinen tunnistaminen on hankalaa, myös kommunikaatioreitin luotettavuuden takaaminen on hankalaa.

Voidaan siis sanoa, että hajautetussa vertaisverkkoympäristössä identiteetin ja turvallisuuden käsitteet ovat kytkeytyneet kiinteästi toisiinsa. Mikäli identiteettejä voidaan monistaa ja väärentää lähes rajatta, on myös verkossa tapahtuvan viestinvälityksen turvallisuuden takaaminen lähes mahdotonta.

Perinteinen ratkaisu luottamusongelmaan ovat olleet *luotetut kolmannet osapuolet*, jollaisia ovat joko keskitetyt palvelimet, joiden läpi viestinvälitys ja käyttäjätunnistus tapahtuu, tai *sertifikaattiauktoriteetit*. Sertifikaattiauktoriteettien tapauksessa jollakin ulkoisella tavalla luottamuksen saavuttanut auktoriteetti myöntää digitaalisia sertifikaatteja, jotka todistavat käyttäjän identiteetin olevan (jollakin tavalla) varmennettu. Tyypillinen esimerkki edellisestä ovat ensimmäisen sukupolven vertaisverkot kuten Napster, jossa käyttäjätunnistuksen ja tiedostojen hakutietokantojen ylläpitoa hoiti keskitetty palvelin, ja ainoastaan tiedostojen siirto tapahtui suoraan verkon osanottajien välillä keskitetyn palvelimen toimittamien IP-osoitteiden perusteella [Bel01]. Jälkimmäistä edustaa esimerkiksi verkkosivustojen SSL-sertifikaateissa käytetty X.509-avaininfrastruktuuri, missä sertifikaattiauktoriteetit myöntävät vaihtelevan tasoisten identiteetin varmistusten perusteella sertifikaatteja käyttäjille ja verkko-osoitteille.

Vastaavia malleja ei ole haluttu ottaa käyttöön DHT-järjestelmissä, oletettavasti koska tarve keskitetylle palvelimelle tai auktoriteetille sotii puhtaasti käyttäjien keskeiseen viestinvälitykseen perustuvaa ideaalia vastaan. Lisäksi vaikka jokin keskitetty valvoja voisi ainakin rajoittaa uusien identiteettien luomista ja olemassaolevien väärinkäyttöä, senkin tarjoama suoja on vain niin hyvä kuin auktoriteetin käyttämä varmistusmenetelmä. Esimerkiksi SSL-sertifikaattien tapauksessa on yleistä, että sertifikaatin saaminen tietylle verkko-osoitteelle edellyttää vain, että sertifikaatin hakija pystyy lukemaan sähköpostin, jonka sertifikaattiauktoriteetin automaatti-

nen järjestelmä lähettää haettavaan verkko-osoitteeseen liittyvälle postipalvelimelle. Käytännössä tämä edellyttää kyseisen osoitteen nimipalvelimen ja postipalvelimen hallintaa, joka ei itsessäänkään ole luodinkestävä turva. Vertaisverkkoympäristössä ei tarjolla ole edes DNS-arkkitehtuurin tarjoamaa luottamusrakennetta, ja toisaalta käyttäjien yksilöiminen henkilöihin saattaa olla vastoin toiveita käyttäjien anonymiteetista.

Edellisen luvun kuvasta 1 (sivu 5) ilmenevät luottamusmallit eri tyyppisissä verkoissa. Kuvan keskitetyt järjestelmät (a) ja (d) luottavat keskitettyyn palvelimeen reititystiedon välityksessä. Kohdissa (b), (c) ja (f) joko verkon kaikki solmut tai valitut "hyvät" yksilöt ovat luotettuja välittämään tietoja ja toimimaan osana reititysprosessia. Kuten kuvasta näkyy, luottamuskysymykset koskevat myös rakenteettomia verkkoja. Kohdassa (e) näkyy luotettu kolmas osapuoli (sertifikaattiauktoriteetti), joka sertifioi solmujen identiteetit.

Identiteetin varmistamisen aiheuttamat vaikeudet voidaan jakaa kahteen ryhmään: järjestelmän *sisäisiin uhkiin*, eli uhkiin, jotka kohdistuvat järjestelmän viestinvälitys- ja reititysprosessiin, sekä *ulkoisiin uhkiin*, jotka liittyvät järjestelmän tietosisällön oikeellisuuteen sekä järjestelmään kuuluvien solmujen itsensä haavoittuvuuteen. Tietosisällön oikeellisuudesta tyypillinen esimerkki tiedostonjakojärjestelmissä on, että tiedosto `Avatar.mp4` ei sisälläkään uutuuselokuvaa, vaan mahdollisesti pelkkiä satunnaislukuja. Solmujen itsensä potentiaalinen haavoittuvuus taas on esimerkiksi, että vertaisverkon kautta löytyneeseen IP-osoitteeseen kohdistetaan palvelunestohyökkäys, joka estää sen toiminnan.

Tietosisältöjen oikeellisuudesta ja erityisesti vertaisverkkoihin kohdistuvista palvelunestohyökkäyksistä ei ole kovin laajasti systemaattista tutkimusta. Vuonna 2010 tehty, kahden tuhannen Kad-verkossa jaettuun tiedostoon kohdistunut tutkimus arvioi kuitenkin, että 50 prosenttia tiedostoista joko ei vastannut varmasti tai todennäköisesti kuvaustaan[CMD+11]. Joka tapauksessa ulkoiset uhat ovat pitkälti joko spesifisiä vertaisverkkoa käyttävän sovelluksen käyttötarkoitukseen tai yleisiä tietoverkkojen ja tietokoneiden tietoturvaan liittyviä ongelmia, joten tämä tutkielma ei käsittele niitä laajemmin. Tutkielma keskittyy turvallisuuden ja identiteettien tarkastelussa DHT:ihin kohdistuviin sisäisiin uhkiin.

3.2 Identiteettien monistaminen: Sybil-hyökkäykset

J.R. Douceur osoitti perustavanlaatuisessa artikkelissaan *The Sybil Attack*[Dou02] vuonna 2002, että kappaleessa 3.1 mainittu identiteettien rajaton luomismahdollisuus muodostaa perustavanlaatuisen rajoitteen luottamukselle hajautetuissa vertaisverkoissa.

Rakenteisessa vertaisverkossa on tärkeää, että vähintään suurin osa osallistuvista solmuista toimii protokollan mukaan, jotta hakumekanismit toimivat. Yksittäinen tarkoituksella tai bugien vuoksi väärin toimiva solmu voi aiheuttaa ongelmia mikäli ongelmiin ei ole varauduttu lainkaan, mutta yksittäisiä tapauksia voi kompensoida esimerkiksi lähettämällä saman kyselyn monelle vastaanottajalle, joilla pitäisi olla haettu tieto. Esimerkiksi alkuperäisen määritelmän mukainenkin Kademlia lähettää hakukyselyt rinnakkain usealle naapurille kohteena olevassa k-ämpärissä[MM02].

Tämä lähestyminen olettaa kuitenkin, että kullakin kohdekoneella on vain yksi identiteetti, eli että on olemassa vain yksi solmu käyttäjää kohden. Järjestelmässä, jossa identiteettien luomiselle ei ole rajoituksia, tämä ei kuitenkaan ole oikeellinen oletus. Solmujen, joiden identiteetti on luotu tätä oletusta noudattavien järjestelmien hämäämiseksi, kutsutaan Douceurin termin mukaan yleisesti *Sybil-solmuiksi*. Oletuksen rikkoutumisesta seuraa, että väärin toimivat solmut eivät ole enää pieni vähemmistö verkossa, vaan voivat halutessaan muodostaa merkittävät osan siitä, jolloin yksinkertaiset varmistustoimet eivät enää toimi. Tämän tilanteen torjumiseksi verkkoa käyttävän solmun pitäisi siis pystyä erottamaan ne etäsolmut, joihin oletus yhdestä identiteetistä kohdekonetta kohden pätee ja ne joihin se ei päde, eli Sybil-solmut. Tämän tekeminen täydellisesti on mahdoton tehtävä yleisessä tapauksessa, kun mitään tunnettua identiteettejä takaavaa auktoriteettia ei ole olemassa, eikä verkon rakenteesta voi tehdä oletuksia[Dou02].

Kun solmu joutuu tekemisiin ennalta tuntemattoman etäsolmun identiteetin kanssa, se voi joko hyväksyä identiteetin suoraan (*suora validaatio*) tai se voi hyväksyä identiteetit, joiden aitouden joku sen jo tuntema muu solmu takaa (*epäsuora validaatio*).

Perusmenetelmä, jolla solmu voi selvittää, vastaako jokin joukko identiteettejä oletusta yhdestä identiteetistä kohdekonetta kohden on pyytää kunkin identiteetin haltijaa suorittamaan jokin tehtävä, josta vain oletuksen täyttävä haltija voi selvitä. Tällainen tehtävä on esimerkiksi jokin aikaa vievää laskentatehtävä, tai että kohde pystyy vastaamaan tiettyyn minimimäärään verkkoviestejä sekunnissa.

Tällainen tehtävä pitää mitoittaa verkon pienimmillä resursseilla r_M varustetun koneen mukaan. Tästä seuraa menetelmän ensimmäinen rajoitus: Jos f on hyökkäävä kone, jonka resurssien r_f ja pienimmillä mahdollisilla resursseilla varustetun koneen resurssien suhde on $\rho = r_f/r_M$, niin f voi hyväksyttävästi esittää $g = \lfloor \rho \rfloor$ eri Sybil-identiteettiä. Tämä asettaa alarajan vahingolle, jonka f voi aiheuttaa verkolle.

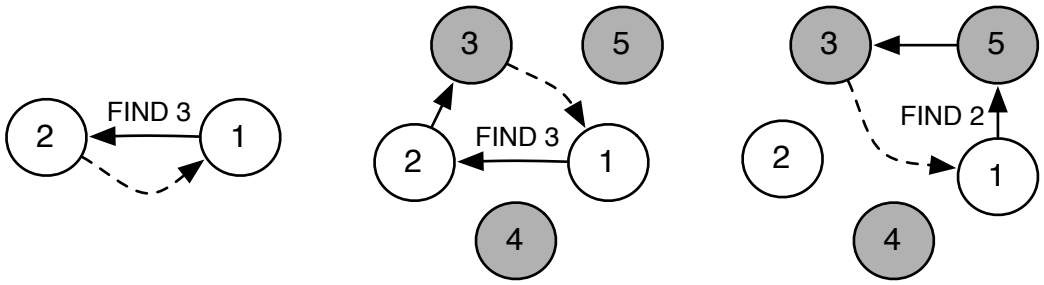
Vastaukset tarkistavan solmun l tarvitsee lisäksi sekä lähettää haasteet tarkastettaville identiteeteille että vastaanottaa ne *samanaikaisesti*. Mikäli l tarkastaa validointitehtäviä peräkkäin, f voi myös laskea vastaukset peräkkäin ja hyväksyttää siten rajoittamattoman määrän Sybil-identiteettejä. Koska tietoliikenteen nopeus on rajallinen, myös kerralla tarkastettavan identiteettijoukon koko on rajallinen.

Epäsuoran validaation tapauksessa paikallinen solmu l lähettää suoran validaation sijaan aiemmin hyväksymilleen naapureille kyselyn, ovatko nämä hyväksyneet sille esitetyn uuden identiteetin. Olkoon F kaikkien hyökkäävien koneiden joukko ja q se määrä naapureita, joilta l tarkastaa sille tarjotun uuden identiteetin hyväksynnän. Tällöin epäsuoran validaation ongelma on, että hyökkääjän tarvitsee saada $|F| \geq q$ identiteettiä hyväksytettyä l:llä, jonka jälkeen hyökkääjä voi hyväksyttää l:llä rajattoman määrän uusia identiteettejä.

Liittyessään verkkoon l:n tarvitsee löytää ensin ne q naapuria, joilta pyytää tarkastuksia. Koska tämä alkuvaihe edellyttää suoraa validaatiota, q naapurin joukon muodostamista koskevat suoran validaation ongelmat. Tällöin F voi saada hyväksytettyä itse luomansa q identiteettiä l:llä, mikäli sen resurssit vastaavat vähintään $q + |F|$ minimaalisilla resursseilla (r_M) varustettua konetta.

Lisäksi hyökkääjä voi myös kiertää tilanteen, missä solmu l vaatii epäsuoran hyväksynnän, mikäli verkon muut solmut hyväksyvät suoran validaation, mutta eivät vaadi, että uudet identiteetit hyväksytetään kaikilla olemassa olevilla solmuilla samanaikaisesti. Olkoon joukko C kaikkien normaalien solmujen joukko, jonka hyökkääjä jakaa alijoukkoihin C_k, joiden koko on vähintään q. Sen jälkeen se hyväksyttää peräkkäin identiteetin i_k jokaisella C_k jäsenellä, jonka jälkeen l:lle on tarjolla tarvittavat q naapuria, jotka ovat hyväksyneet identiteetin i_k. Tällöin hyökkääjä voi hyväksyttää $\lfloor \frac{C}{q} \rfloor$ Sybil-solmua solmulle l[Dou02].

Tämä hyökkäys asettaa teoreettisen rajan sille, miten identiteettien luontia voi rajoittaa verkossa, missä niille ei ole keskitettyä takaajaa, eikä verkon rakenteesta voi tehdä mitään oletuksia. Useiden identiteettien luonti mahdollistaa hyökkäyksen verkon reititysmekanismeja vastaan, jota käsitellään seuraavassa kappaleessa. Käytännön järjestelmissä on kuitenkin mahdollista tehdä lisäoletuksia verkon ra-



(a) Kysely reititetään normaalisti (tuloksena on avaimen lähin solmu)

(b) Normaali reititys hyökkääjäsolmujen läsnäollessa

(c) Virheellinen reititys hyökkääjäsolmujen läsnäollessa

Kuva 4: Reititys verkossa, jossa on mukana hyökkääjäsolmuja. Normaalisolmut on merkitty valkoisella, hyökkääjäsolmut harmaalla.

kenteesta ja solmujen normaalista käyttäytymisestä, mikä mahdollistaa laajempien tarkastusmekanismien rakentamisen. Näitä mekanismeja käsitellään luvuissa 3.4 ja 6.4.

3.3 Hyökkäykset DHT:ja vastaan

Rakenteisessa verkossa solmujen pitää noudattaa reititysalgoritmin sääntöjä, jotta reititys toimii. Algoritmin pitäisi määritellä reitityksen säännöt invariantteina, jotta ainakin yksinkertaiset virheet olisi mahdollista tunnistaa ja mahdollisesti välttää. Lisäksi solmujen pitää myös noudattaa oletuksia niille annettujen tietojen säilyttämisestä.

Hyökkäykset reititysmekanismeja vastaan voi jakaa neljään kategoriaan. Ensimmäinen näistä on väärät vastaukset reitityskyselyihin. Tässä tilanteessa hyökkäävä solmu vastaa sille lähetettyihin viesteihin ja näin ollen sen naapurit pitävät sen mukana omissa naapurilistoissaan. Yksinkertaisimmillaan hyökkääjä voi lähettää kysyjän kauemmaksi haetusta avaimesta sen sijaan, että kertoisi sille lähempien solmujen osoitteita. Lisäksi se voi myös kertoa vastauksessa talletusviestiin, että jokin etäinen solmu on vastuussa jostakin avaimesta ja näin aiheuttaa tiedon tallentumisen väärälle solmulle.

Kuva 4 havainnollistaa hyökkäävien solmujen vaikutusta verkon toimintaan. Kohdassa (a) verkossa on vain normaalisolmuja ja solmu 1 löytää avaimen 3 normaalisti. Kohdassa (b) reititys päätyy hyökkääjäsolmulle, joka saattaa palauttaa tai olla palauttamatta oikean arvon avaimelle. Kohdassa (c) solmu 5 ohjaa kysyjän väärälle solmulle (verkossa olisi myös tarkalleen haettua avainta 2 vastaava solmu.

Yksittäisten hyökkääjien tapauksessa selkeästi vääränlaiseen reititykseen on melko helppo varautua; mikäli kysyjä voi seurata kyselyn etenemistä, se voi helposti todeta tilanteen, missä se päätyy kauemmas haetusta avaimesta. Väärään paikkaan talletuksen tapauksessa talletuksen kohde voi myös vastata virheellä, mikäli avain ei kuulu sille[SM02]. Chord ja Kademlia käyttävät iteratiivista reititystä, joten alkuperäinen kysyjä näkee jatkuvasti kyselyn etenemisen.

Toinen kategoria ovat virheelliset reititystietojen päivitykset, joiden tavoite on korruptoida muiden solmujen reititystaulut, esimerkiksi niin, että solmut yrittävät ottaa yhteyttä vääriin tai olemattomiin solmuihin. Tämä on yksinkertaisimmassa tapauksessa vältettävissä tarkastamalla tuntemattomat osoitteet ennen niiden lisäämistä reititystauluun[SM02], tosin suuri määrä väärennettyjä osoitteita voi aiheuttaa palvelunestohyökkäyksen kolmatta osapuolta kohtaan, kun solmut yrittävät ottaa siihen toistuvasti yhteyttä.

Sekä Chord että Kademlia muodostavat reititystaulunsa pääosin samoilla operaatioilla, joita ne käyttävät viestinvälitykseen. Kademlia myös pyrkii pitämään reititystauluissa vanhoja ja toimiviksi tunnettuja osoitteita sen sijaan, että ottaisi aina uusia vanhojen tilalle[MM02]. Näissä kahdessa algoritmissa tämä kategoria on käytännössä yhtenevä ensimmäisen kanssa: virheelliset vastaukset reitityskyselyihin voivat aiheuttaa reititystaulujen korruptiota, mutta ei ole erillistä luokkaa reititystaulun päivitysviestejä, jotka voisivat aiheuttaa sen (tosin toteutukset saattavat käyttää erillisiä reititystietojen päivitysviestejä, kuten luvussa 6 käy ilmi).

Kolmas kategoria on hyökkäys uusia liittyjiä vastaan, eli pyrkimys hajottaa verkko osiin. Uuden solmun liittyessä sen tarvitsee ottaa yhteys johonkin olemassa olevaan solmuun, joka sen täytyy saada tietoonsa jollakin varsinaisen algoritmin ulkopuolisella tavalla. Mikäli hyökkääjä voi vaikuttaa tähän mekanismiin, se voi antaa uudelle solmulle osoitteen, joka viittaa hyökkääjän kontrolloimaan, käyttäjän haluamasta verkosta erilliseen verkkoon. Tämä verkko saattaa käyttäytyä normaalin verkon tavoin ja jopa sisältää osan oikean verkon tietosisällöstä, mutta on kokonaan tai osillaan erossa siitä verkosta, johon käyttäjä halusi liittyä[SM02]. Tämä hyökkäys on erillään itse DHT-reititysalgoritmin huijaamisesta ja riippuu vain liittymismekanismista sekä siitä, onko olemassa jokin mekanismi (esimerkiksi keskitetty tunnistuspalvelin), jolla liittyjä voi tunnistaa “aidot” solmut.

Neljäs kategoria ovat itse avainten viittaaman tiedon säilytykseen liittyvä virhekäyttäytyminen. Hyökkääjä voi kieltäytyä palauttamasta tietoa, jonka avaimesta se on vastuussa tai väittää tallettaneensa uuden tiedon joka sille kuuluu, mutta jättää

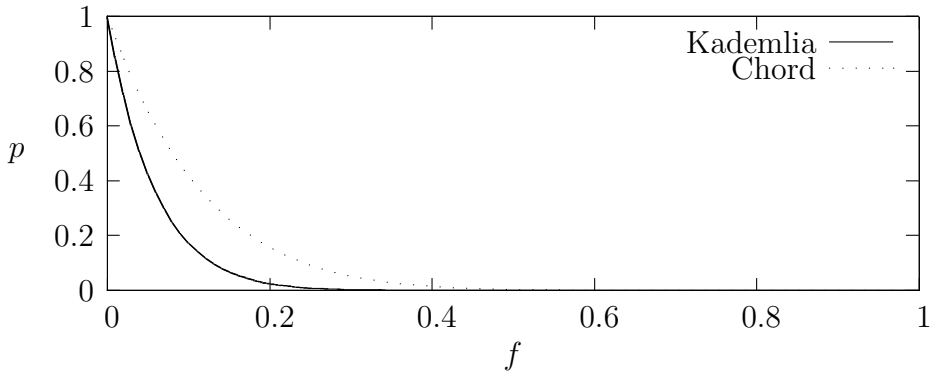
sen tallentamatta. Tämän tilanteen välttäminen yksinkertaisissa tapauksissa edellyttää, että tieto on replikoitu usealle solmulle ja lisäksi nuo kopiot on mahdollista löytää ilman, että jokin yksittäinen solmu on vastuussa sijaintitietojen ylläpidosta tai replikoinnin suorittamisesta[SM02]. Chordin tapauksessa replikaatiomekanismia ei ole määritelty kiinteästi, mutta kehittäjät ehdottavat, että tiedon voisi tallettaa myös yhdelle tai useammalle avaimen varsinaisen haltijan seuraajalle[SMK+01]. Kademliassa on määritelty avain-arvo-parien uudelleenjulkaisu jo alkuperäisessä määrittelyssä. Arvot replikoidaan toteutuskohtaiselle vakiomäärälle avainta läheisimpiä solmuja ja ne voi hakea miltä tahansa niistä[MM02].

Hyökkääjä voi pyrkiä välttämään tunnistetuksi tulemista käyttäytymällä epäjohdonmukaisesti. Se voi esimerkiksi avaimen hallussapitäjänä palauttaa oikean avaimeen liittyvän tiedon osalle kysyjistä, mutta jättää vastaamatta lopuille[SM02].

Näiden hyökkäysten tavoite voi olla joko hallita kaikkea liikennettä verkossa, eristää yksittäisiä solmuja muusta verkosta tai estää yhden tai useamman avaimen löytäminen verkosta[CDG+02]. Tämäntyyppiset hyökkäykset tunnetaan kollektiivisesti nimellä "pimennyshyökkäykset" (*Eclipse attack*), koska kontrolloimalla reititystä tai arvojen talletusta hyökkääjä voi "pimentää" haluamiensa avainten tai solmujen näkyvyyden verkossa[SCDR04]. Tämän voi myös yhdistää Sybil-hyökkäykseen niin, että oikeita solmuja eristetään muusta verkosta, jolloin hyökkääjä tarvitsee vähemmän Sybil-solmuja jäljelle jäävän osan tarkkailuun tai manipulointiin[UPS11].

Triviaalimmat hyökkäykset voi torjua edellä mainituilla yksinkertaisilla tarkastuksilla. Pienissä verkoissa hyökkääjä voi pystyttää useita koneita, jota käyttävät valittuja tunnisteita pimentääkseen tietyn kohteen. Useiden identiteettien luominen kappaleessa 3.2 esitellyn Sybil-hyökkäyksen tavoin on kuitenkin yleensä taloudellisin ja tehokkain keino, mikäli tätä vastaan ei ole tarkastuksia tai hyökkääjä onnistuu kiertämään ne.

Mikäli osuus f verkon solmuista (mukaanlukien mahdollisesti Sybil-solmut) on hyökkääjän hallussa, todennäköisyys saada viesti perille halutulle vastaanottajalle on $(1-f)^{h-1}$, missä h on algoritmista riippuva keskimääräinen hakijan ja kohteen välissä olevien solmujen määrä. Tosin koska myös avainta hallussaan pitävä solmu itse voi olla hyökkääjän hallussa, lopullinen todennäköisyys saavuttaa oikean tiedon luovuttava solmu on vain $(1-f)^h$. Chordin tapauksessa $h = \frac{\log_2(N)}{2}$, missä N on odotettu verkon solmujen määrä[CDG+02]. Kademlialla tämä on alkuperäisen artikkelin määrittämässä muodossa $h = \log_2(N)$[SR06]. Kuva 5 esittää todennäköisyyden, että haettu avain löytyy 100000 solmun Kademlia- ja Chord-verkoissa, kun osuus f kas-



Kuva 5: Kyselyn onnistumistodennäköisyys p Chord- ja Kademlia-verkoissa, kun viallisten tai hyökkäävien solmujen osuus f kasvaa ($N = 100000$).

vaa. Mikäli hyökkääjä korruptoi tai pudottaa kaikki viestit, todennäköisyydet saada viesti perille laskevat nopeasti. Tämä tarkastelu ei kuitenkaan ota huomioon, että toteutukset voivat pyrkiä löytämään kohdeavaimen suorittamalla hakuja monen eri reitin kautta.

Käytännössä suositut hyökkäystyypit ja niiden toivotut vaikutukset riippuvat DHT:a käyttävistä sovelluksista ja hyökkääjien tavoitteista. Kademlia-verkkojen häirintään pyrkiviä käytännön hyökkäyksiä käsitellään tarkemmin luvussa 6. Hyökkääjän tavoitteena ei kuitenkaan välttämättä ole aina häiritä verkon toimintaa tai estää tiedon löytämistä, vaan tarkkailla verkon liikennettä. Tässä tilanteessa hyökkääjä voi luoda suuren määrän Sybil-identiteettejä käyttäviä solmuja, jotka käyttäytyvät korrektisti verkon sääntöjen mukaan, mutta tallettavat näkemänsä avaimet ja arvot ja mahdollisesti näiden hakijoiden ja tallettajien osoitteet. Seuraavassa kappaleessa on esimerkki tällaisesti hyökkäyksestä, joka kohdistui Vuze-DHT:a vastaan, mutta on periaatteessa sovellettavissa myös muihin DHT-järjestelmiin. Lisäksi verkon osanottajien ja verkon sisältämän tiedon tehokasta keräämistä käsitellään tarkemmin luvussa 5.

3.3.1 Esimerkkitapaus: Hyökkäys Vanishia vastaan

Washingtonin yliopiston tutkijat kehittivät järjestelmän nimeltä *Vanish*[GKLL09], jonka tavoite oli mahdollistaa "itsestään tuhoutuvien" dokumenttien julkaiseminen. Julkaisija salaa dokumentin satunnaisella avaimella K, joka pilkotaan n palaksi, joista lukija tarvitsee k kappaletta (esimerkiksi 50 prosenttia alkuperäisistä) dokumentin avaamiseksi. Lisäksi julkaisija valitsee "pääsyavaimen" L, jonka pohjalta

järjestelmä generoi n DHT-avainta. Lopuksi järjestelmä tallentaa avaimet vapaavalintaiseen DHT:een. Dokumentin avaamiseksi käyttäjä hakee pääsyavaimen pohjalta tarvittavan määrän avaimen K paloja DHT:sta. Dokumentin "tuhoutuminen" seuraa K:n katoamisesta DHT:sta kun julkaisija ei enää uudellenjulkaise avaimen palasia järjestelmään ja DHT:n solmut siivoavat vanhat avain-arvo-parit pois tietovarastoistaan. Prototyypissä käytettiin Kademlia-pohjaista Vuze-DHT:a ja sittemmin suljettua OpenDHT-tutkimusalustaa[Rhe].

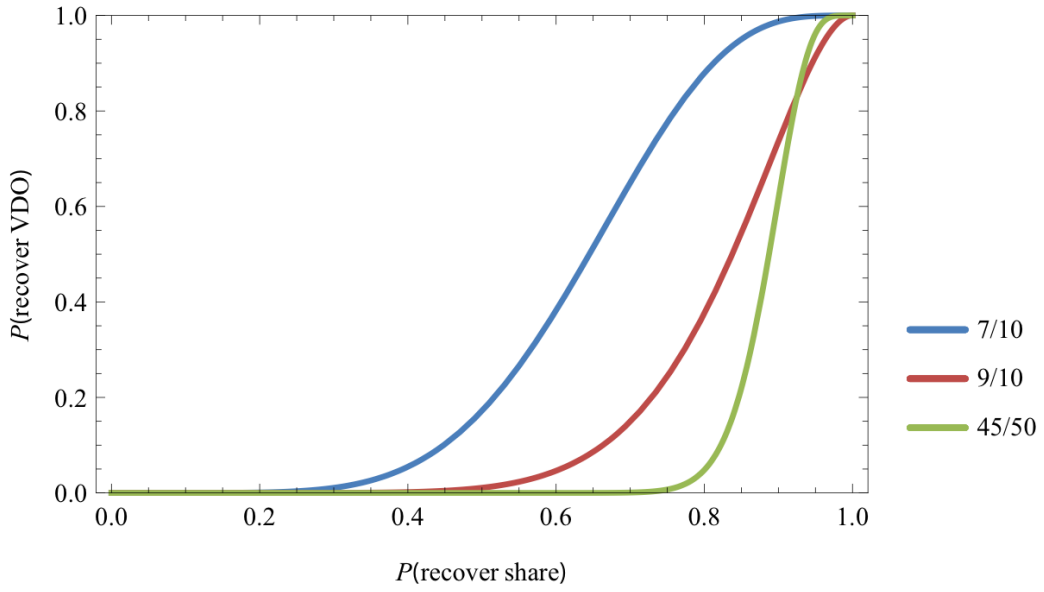
Tämän järjestelmän turvallisuus nojaa oletukseen, että hyökkääjä ei pysty saamaan käsiinsä merkittävää osaa DHT:n tietosisällöstä. Hyökkäys tätä oletusta vastaan on Sybil-hyökkäys, eli liittää suuri määrä solmuja avaimia varastoivaan DHT:een, tallettaa niiden näkemät viestit ja pyrkiä löytämään niiden joukosta Vanish-dokumenttien avainten palat.

Suunnittelijat arvioivat itse, että hyökkäys, jolla saa avattua 25 prosenttia talletetuista dokumenteista miljoonan käyttäjän DHT:ssa, vaatisi noin 87000 jatkuvasti toiminnassa olevaa Sybil-solmua kun $n = 50$, joka tulisi maksamaan Amazon EC2 -hosting-palvelun hinnoilla noin 860 tuhatta dollaria vuodessa[GKLL09].

Tämä arvio on kuitenkin hyvin optimistinen, koska osoittautuu, että suurimittaisen Sybil-hyökkäyksen toteutus on huomattavasti alkuperäistä arviota halvempaa ja helpompaa. Merkittävin ongelma alkuperäisen tutkimuksen turvallisuusanalyysissa oli, että siinä oletettiin, että seurannan pitäisi olla jatkuvaa. Vuze-DHT:ssa solmut virkistävät avaimia ajoittain tallentamalla ne uudelleen, joten Sybil-solmut voivat joko olla mukana verkossa vain osan aikaa tai vaihtaa lyhyin väliajoin tunnistettaan ja siten tarkkailla suurempaa osaa osoiteavaruudesta. Käytännössä kokeiden perusteella kolme minuuttia oli riittävän pitkä aika selvittää lähes kaikki tietyn tunnisteen lähellä sijaitsevat tietoalkiot.

Toinen ongelma alkuperäisessä analyysissa olivat ylimitoitetut odotukset Sybil-solmujen resurssitarpeista. Alkuperäinen oletus oli, että yhdellä EC2-virtuaalikoneella voisi ajaa vain muutamaa solmua yhtä aikaa. Alkuperäinen analyysi ei määrittele pohjaoletuksia kovin tarkasti, mutta ilmeisesti analyysissä oletettiin, että Sybil-solmu olisi täysimittainen Vuze-asiakasohjelma.

Käytännössä muutaman sadan rivin muutoksella alkuperäiseen Vuze-asiakasohjelmaan sillä pystyy kuitenkin ajamaan noin 50 Sybil-solmua yhdellä virtuaalikoneella. Lisäksi minimaalinen oma toteutus Vuze-DHT:n algoritmista ei ollut myöskään erityisen vaativa. Tällä toteutuksella hyökkäysmahdollisuuksien tutkijat pystyivät ajamaan 200 yhtäaikaista Sybil-solmua yhdellä virtuaalikoneella. Yhdistettynä jat-



Kuva 6: P(recover VDO) kertoo todennäköisyyden, jolla hyökkääjä voi avata Vanish-dokumentin kun P(recover share) eli todennäköisyys löytää mikä tahansa avaimen pala kasvaa. Arvio eri k/n arvoille. Kuvan lähde [WHH+09].

kuvaan tunnisteen vaihtamiseen tämä vastasi kahdeksan tunnin aikana noin 320000 Sybil-solmua, joiden avulla tutkijat löysivät avaimet 99.4 prosentille 1650 testidokumentistaan. Riippuen miten moneen palaan käyttäjä hajottaa Vanish-avaimet, tutkijat arvioivat, että vuoden mittainen tarkkailu maksaisi noin 6–12 tuhatta dollaria mikäli tavoitteena on löytää 99 prosenttia kaikista avaimista. Tämä on huomattavasti vähemmän kuin alkuperäinen 860000 dollarin arvio.

Kuva 6 näyttää todennäköisyyden, että hyökkääjä pystyy löytämään Vanishilla salatun dokumentin avaimen, kun todennäköisyys löytää mikä tahansa avaimen pala kasvaa. Viivat kuvaavat todennäköisyyttä kolmelle eri avaimen palojen tarvittavan määrän ja kokonaismäärän suhteelle k/n.

Ongelma seuraa osittain siitä, että Vuze-DHT ei ole alunperin suunnattu tämäntyyppiseen käyttöön ja sallii tietosisällön keräämisen melko helposti. Periaatteessa tilannetta auttaisivat muutokset Vuze-DHT:n toimintaan tai jonkin muun DHT:n käyttö, mutta toisaalta Vanishin perusajatus nojaa siihen, että se ei ole verkon “erityinen” käyttäjä vaan katoaa muun liikenteen sekaan. Mikäli siitä tulisi DHT:n pääasiallinen käyttökohde, voisivat sen tarkkailuun erikoistuneet tahot taas sijoittaa suurempia resursseja kyseisen DHT:n tarkkailuun[WHH+09].

Tämä esimerkki näyttää, että ainakin yhdessä laajalti käytetyssä DHT:ssa Sybil-

hyökkäys on sekä tehokas että toteutettavissa kohtuullisilla resursseilla. Tässä esitelty hyökkäys on melko suoraviivainen Sybil-hyökkäys, joka on periaatteessa sovellettavissa monenlaisiin DHT:ihin, jotka eivät ole varautuneet järjestelmälliseen sisällön keruuseen. Tämä on luultavasti seurausta siitä, että sekä tutkimuksessa että sovelluksissa on varauduttu enemmänkin palvelunestohyökkäysten ja pimennyshyökkäysten torjuntaan. Näitä käsitellään Kadelmia-pohjaisten DHT:jen osalta luvussa 6.

3.4 Puolustusmekanismit

Puolustuskohteet voidaan jakaa kolmeen ryhmään sen perusteella, mitä pyritään torjumaan[UPS11]:

1. Puolustus Sybil-identiteettejä vastaan eli identiteettien varmennusmekanismit.
2. Puolustus pimennyshyökkäyksiä vastaan eli mekanismit, joilla asetetaan vaatimuksia naapurisolmujen toiminnalle.
3. Puolustus reititys- ja tietosisältöhyökkäyksiä vastaan, joissa solmut reitittävät tietoa väärin tai palauttavat väärää tietoa.

Tämä luku esittelee yleisiä mekanismeja kahdesta ensimmäisestä kategoriasta. Kolmannessa kategoriassa sekä hyökkäys- että puolustusmekanismit liittyvät DHT:n spesifiseen toteutukseen, ja niitä tarkastellaan Kademlian osalta luvussa 6, kuten myös käytännön sovelluksia kahden ensimmäisen kategorian teoreettisista ratkaisuista.

3.4.1 Identiteettien varmennus

Identiteettien varmennuksen tavoite on estää rajoittamaton identiteettien luominen, eli Sybil-hyökkäykset. Suoraviivaisin lähestyminen tähän on jo aiemminkin mainittu sertifikaattien myöntäminen keskitetyltä sertifikaattiauktoriteetilta. Sen lisäksi, että tämä malli olettaa keskitetyn varmennusauktoriteetin olemassaolon, se myös vaatii, että on jokin kiinteä tunniste johon sertifikaatit sidotaan. Tämä voi olla joko solmun tunniste DHT:n sisällä, joka edellyttää, että tunniste pysyy aina samana tai solmun IP-osoite, joka edellyttää, että IP-osoite pysyy samana ja että yhdellä osoitteella ei ole montaa käyttäjää[CDG+02].

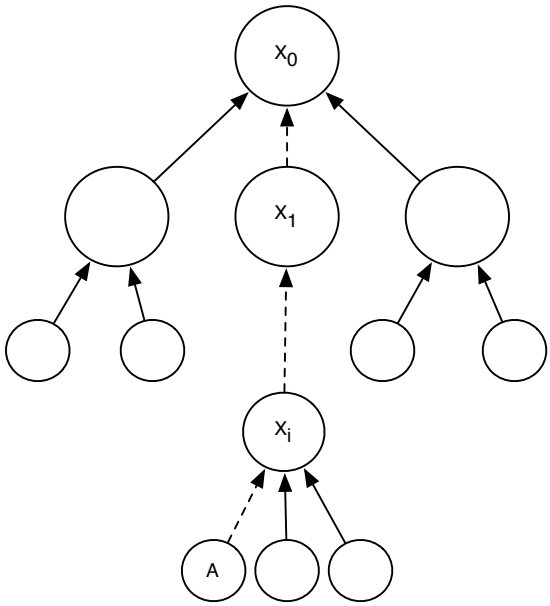
Koska useimmat kotikäyttäjät saavat IP-osoitteensa DHCP-palvelimilta ja ovat usein NAT-yhdyskäytävien takana, sertifikaatin sitominen IP-osoitteeseen ei ole käytännössä realistinen malli. Kiinteänä pysyvä solmun tunniste tekee käyttäjästä pysyvästi tunnistettavan, joka saattaa olla ongelma joissain sovelluksissa. Lisäksi joissakin DHT:issa solmun tunniste muuttuu sen elinaikana, mutta tämä ei ole ongelma Chordissa tai Kademliassa. Mikäli tunnistettavuus, keskitetty auktoriteetti ja sen vaatima hallinnollinen työ ovat hyväksyttäviä, tämä on todennäköisesti sekä teknisesti helpoin että deterministisin tapa hoitaa identiteettien varmennus.

Muut lähestymiset ovat vaihtelevissa määrin todennäköisyyksiin perustuvia. Niiden pyrkimys on hidastaa tai tunnistaa lisäidentiteettien luominen, ei estää sitä täysin.

Useat tutkijat ovat ehdottaneet menetelmiä, jotka yhdistävät solmun IP-osoitteen tai muun ulkoisen tunnisteen sekä verkkolatenssimittauksia tunnettuihin "majakoihin", joista muodostetaan varmistettavissa oleva tunniste ilman keskitettyä sertifioijaa. Näiden menetelmien ongelmia ovat IP-osoitteiden muuttuminen, NAT-yhdyskäytävät, muuttuvat etäisyydet majakoista ja joissakin menetelmissä hankaluus määritellä ero "läheisten" ja "kaukaisten" solmujen välillä[UPS11]. Menetelmissä, jotka perustuvat latenssimittauksiin toimivuus saattaa olla rajoitettu erityisesti mobiiliympäristöissä, joissa latenssit voivat olla hyvinkin epävakaita. Osoitepohjaisten menetelmien kannalta taas useiden IP-osoitteiden hankkiminen ei edellytä käytännössä hyökkääjältä nykyään erityisen suuria resursseja.

Toinen lähestyminen ovat laskentatehtävät, joilla solmut todistavat uniikkiutensa. Nämä järjestelmät vähintään hidastavat uusien identiteettien luontia, mutta kärsivät Douceurin alkuperäisessä Sybil-artikkelissa määrittelemistä huijausmahdollisuuksista ja vaativat jatkuvaa todistusten laskentaa verkon osanottajilta.

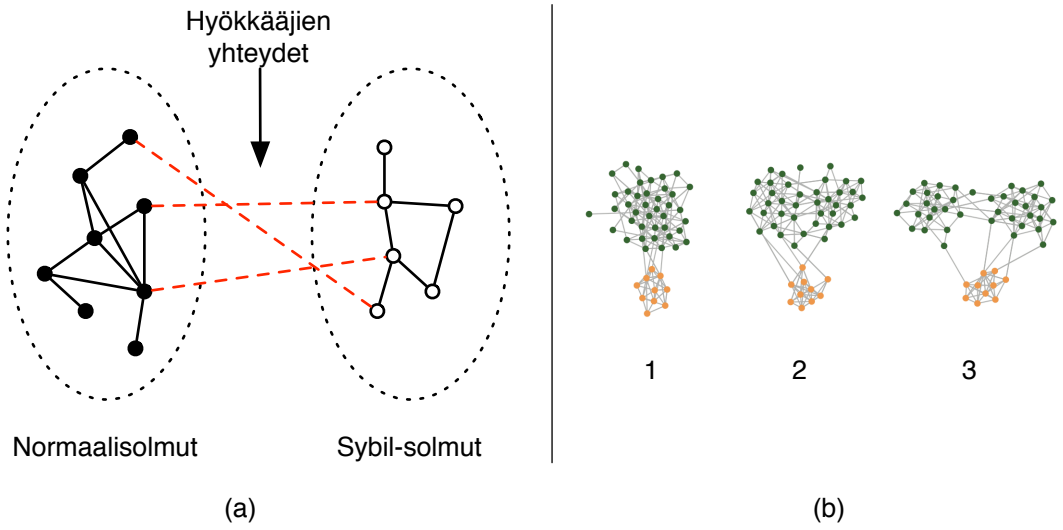
Esimerkki tällaisesta menetelmästä on Rowaihy et al.[REMLP07] esittämä "pääsynhallintajärjestelmä", jossa verkon solmut muodostavat dynaamisesti puun, joka liittyjän on käytävä läpi ratkaisten laskentatehtävä jokaisella tasolla juureen asti. Tällaisen puun rakenne, johon on liittymässä solmu A näkyy kuvassa 7. Uusi liittyjä A valitsee itselleen avainparin ja lähettää viestin jollekin olemassa olevalle lehtisolmulle X_i. Lehti lähettää liittyjälle tehtävän, jossa liittyjän tulee "kääntää" kryptografisella hajautusfunktiolla laskettu hajautusarvo. Hajautusfunktioiden luonteesta johtuen tämä vaatii suurella todennäköisyydellä noin puolen lähdearvoavaruuden läpikäymistä. Kun liittyjä on ratkaissut tämän tehtävän, se saa uuden, solmun X_i allekirjoittaman tehtävän, jolla sen pitää todistaa aitoutensa solmulle X_{i-1}. Tämä prosessi toistetaan, kunnes liittyjä saavuttaa juurisolmun X_0. Tämä juurisolmu an-



Kuva 7: Rowaihy et al. esittämä puumainen pääsynhallintajärjestelmä. Kuvassa solmu A on liittymässä verkkoon. Sen tulee suorittaa laskentatehtävät katkoviivalla merkittyä polkua edeten. Kuva [REMLP07] mukaan.

taa sille lopullisen allekirjoitetun identiteetin, joka perustuu liittyjän julkiseen avaimeen. Muut solmut voivat tämän jälkeen varmistaa, että liittyjän identiteetti on sertifioitu (järjestelmä olettaa, että X_0 julkinen avain on kaikkien osallisten tiedossa). Lisäksi sertifiointi pitää suorittaa ajoittain uusiksi, jotta hyökkääjä ei voi pitkällä aikavälillä kerätä suurta määrää sertifioituja identiteettejä, jotka se ottaisi käyttöön suurena massana myöhemmin. Mikäli hyökkääjä yrittää käyttää asemaansa puussa väärin myöntämällä liian helposti uusia sertifikaatteja, sen yläpuolella oleva solmu voi tunnistaa sen toimivan väärin. Tällöin oletetun hyökkääjän sertifikaatti perutaan ja koko sen alla olevan puun on liityttävä verkkoon uudelleen[REMLP07]. Tämä tosin tekee korkeaan asemaan verkossa nousseiden hyökkääjäsolmujen poistamisesta potentiaalisesti verkkoa laajalti häiritsevän operaation.

Simulaatiotesteissä kahdeksan hyökkääjän saavuttama osuus verkosta pysyi noin 6 prosentin osuudessa solmuista, kun sertifikaattien elinikä oli neljä tuntia[REMLP07]. Järjestelmän yhdistää keskitetyn sertifioinnin ja desentralisoidun varmistuksen piirteitä. Puun huipulla olevien solmujen tulee olla luotettavia, koska erityisesti juurisolmua ei voi valvoa kukaan. Toisaalta liittyjiä ei tarvitse arvioida minkään ulkoisen tai subjektiivisen kriteerin perusteella. Lisäksi hyvänä puolena on, että liittyjät eivät pääse valitsemaan omia tunnisteitaan, mikä heikentää mahdollisuuksia kohdennettuihin hyökkäyksiin. Varjopuolena kuitenkin on edellä mainittu tarve jatkuvalle



Kuva 8: (a) Löyhästi kytkeytyneet normaali- ja Sybil-solmujen joukot. Kuva [YKGF06] mukaan. (b) Normaalisolmujen verkon (ylhäällä) jakautuminen kahteen yhteisöön, joka muistuttaa Sybil-solmujen kaltaista erillistä joukkoa (alhaalla). Kuvan lähde [VPGM10].

todistusten laskennalle, jonka vaativuus riippuu sertifikaattien eliniästä. Tiettävästi järjestelmästä ei ole olemassa laajalle levinneitä käytännön toteutuksia.

Kolmas lähestyminen nojaa tilastollisiin menetelmiin. Tässä lähestymisessä voidaan tarkastella solmujen tunnisteiden jakautumista kohdistettujen hyökkäysten tunnistamiseksi [CCFD12], olettaa että verkon sisäiset suhteet vastaavat sosiaalisten verkostojen rakennetta[DM09] tai käyttää hyväksi käyttäjän DHT:n ulkopuolella muodostaminen Facebook-kaveruuksien kaltaisia luottamussuhteita[YKGF06, YGKX08]. Käytännössä DHT-järjestelmien naapurisuhteet muodostetaan automaattisesti, joten ne eivät itsessään muistuta sosiaalista verkostoa. Tästä syystä sosiaalisiin verkostoihin perustuvissa mekanismeissa nojataan yleensä ulkoisesti muodostettuihin verkostoihin. Toisin kuin aiemmassa sertifikaatiomenetelmässä, nämä menetelmät eivät vaadi edes dynaamisesti muodostettua sertifiointihierarkiaa DHT:n sisällä.

Tunnisteiden jakautumiin perustuvia menetelmiä sovellettuna Kademliaan tarkastellaan tarkemmin kappaleessa 6.4. Sosiaalisiin verkostoihin perustuvat menetelmät ovat periaatteessa lupaavia, mutta niistäkään ei ole tiettävästi olemassa laajasti käytössä olevia käytännön toteutuksia. Yu et al. esittivät vuonna 2006 artikkelissaan menetelmää nimeltä SybilGuard[YKGF06], jonka perusajatus on, että sosiaalisessa verkostossa aidot käyttäjät muodostavat keskenään tiivisti kytkeytyneen verkon, kun taas Sybil-solmut muodostavat keskenään verkon, joka on kytkeytynyt löyhästi aitojen käyttäjien verkkoon. Aito solmu voi tarkastaa tuntemattoman solmun kysy-

mällä siltä joukon satunnaisia reittejä sen tuntemassa sosiaalisessa verkossa ja toteamalla, onko tarkastettavilla reiteillä yhteisiä solmuja sen omien reittien kanssa. Sybil-solmujen tapauksessa tämä on epätodennäköistä, koska ne eivät oletettavasti ole kytkeytyneet aitojen käyttäjien verkkoon kuin muutama reittiä (tämä voi olla seurausta joko hyökkääjän osallistumisesta manuaalisesti aitoon verkkoon tai aitojen käyttäjien hyväksyessä vahingossa kaveripyyntöjä hyökkääjän verkosta). Kuvan 8 kohta (a) näyttää verkon, jossa erillinen Sybil-solmujen joukko on kytkeytynyt löyhästi normaalisolmujen joukkoon.

Koska koko vertaisverkon rakenne ei ole tiedossa, kysymys on, miten tunnistaa luotettavasti tilanne, missä siirrytään aitojen solmujen verkosta Sybil-solmujen verkkoon. Alkuperäisestä menetelmästä on kehitetty useita versioita, joista uusin on Danezis et al. vuonna 2009 julkaisema SybilInfer[DM09]. Menetelmä perustuu kolmeen oletukseen: verkossa tunnetaan ainakin yksi aito solmu (joka voi olla käyttäjän oma solmu), sosiaaliset verkot ovat nopeasti sekoittuvia (satunnaiskulku verkossa muodostaa nopeasti rakenteen, joka muistuttaa koko verkon rakennetta) ja että solmu tuntee koko sosiaalisen verkon G rakenteen (joka on suhteellisen staattinen ja oletettavasti saatavissa ulkoisista sosiaalisista verkkopalveluista kuten esimerkiksi Facebookista). Vastaavasti siirtymä aitojen ja Sybil-solmujen välillä johtaa hitaaseen sekoittumiseen, koska verkot ovat löyhästi kytkeytyneitä. Muodostamalla joukko satunnaiskävelyitä T verkossa G ja käyttämällä oletustodennäköisyyttä että jokin tämän osajoukko X sisältää vain aitoja solmua (eli $P(X = \text{Aitojensolmujenjoukko})$), voidaan arvioida kaikille eri X todennäköisyys $P(X = \text{Aitojensolmujenjoukko}|T)$.

Sybil-solmujen määrän kasvaessa siirtymä verkon osien välillä muuttuu selkeämmäksi eli jälkimmäinen todennäköisyys muuttuu suuremmaksi. Kun todennäköisyyden raja-arvoksi asetetaan 0.7 ja oletetaan, että 1000 solmun verkosta 100 solmua on hyökkääjän hallinnassa, synteettisellä aineistolla menetelmä ei tunnista alle 100 Sybil-solmun lisäämistä, mutta tunnistussuhde paranee nopeasti tämän rajan yli mentäessä. Tällöin menetelmä tunnistaa sekä Sybil-solmut että alkuperäiset hyökkääjän haltuun ottamat solmut[DM09]. Alkuperäiset tutkijat testasivat sekä SybilInferiä että SybilGuardia ja SybilLimittiä myös aidoilla aineistolla, mutta koska näiden aineistojen tapauksessa tutkijoilla ei ollut luotettavaa tapaa todeta, mitkä solmut olivat todella hyökkääjiä ja mitkä eivät, tulosten tulkinta jää avoimeksi. Kaikki menetelmät ovat sovellettavissa yleisesti sosiaalisiin verkostoihin. Vain alkuperäinen SybilLimit-esitys määrittelee spesifisen toteutuksen, jolla menetelmää voi soveltaa DHT-solmujen kesken.

Viswanath et al. vertailivat näitä menetelmiä keskenään ja käyttivät lisäksi yleisesti verkkoihin suunnattua yhteisötunnistusalgoritmia. Synteettisessä, mittakaavamattomassa verkossa sekä SybilGuard että SybilLimit olivat huomattavasti jäljessä SybilInferiä sekä yhteisötunnistusalgoritmia. Sovellettuna Facebookista kerättyyn verkkoon, johon lisättiin Sybil-solmuja, kaikki Sybil-spesifiset menetelmät eivät selvinneet juuri arvaamista paremmin ja yleinen yhteisötunnistus ohitti ne kaikki. Tämä johtuu todennäköisesti siitä, että kun verkko sisältää useita löyhästi keskenään kytkeytyneitä yhteisöjä, SybilGuard ja vastaavat menetelmät eivät enää erota naapuriyhteisöjä joukosta Sybil-solmuja[VPGM10]. Kuvan 8 kohta (b) näyttää tällaisen tilanteen synnyn. Tilanteessa 1 näkyy selkeä normaalien solmujen joukko. Tilanteessa 3 normaalikäyttäjien verkko on erkaantunut kahteen löyhästi kytkeytyneeseen ryhmään, joka muistuttaa Sybil-solmujen ryhmää.

Toisaalta tulos viittaa myös siihen, että yleisiä algoritmeja voisi soveltaa myös Sybiltunnistukseen. Tutkijat selvittivät kuitenkin yleistä tunnistustehokkuuden laskua ja totesivat, että kun verkon sisällä on enemmän yhteisöjä (eli vähemmän satunnaisuutta), kaikkien algoritmien tarkkuus heikkenee[VPGM10].

Käytännön ongelma näissä menetelmissä DHT:iden suhteen on, että ne tarvitsevat taustalleen olemassa olevan sosiaalisen verkoston ennen DHT:n muodostamista. Lisäksi merkittävä ongelma on DHT-identiteettien liittäminen olemassa olevan sosiaalisen verkoston identiteettiin sekä menetelmien helppokäyttöinen käytännön toteutus DHT-asiakasohjelmissa[UPS11].

3.4.2 Naapurisolmujen varmennus

Naapurisolmujen varmennus pyrkii estämään pimennyshyökkäykset tunnistamalla väärin käyttäytyvät naapurisolmut. Mahdollisuus luoda Sybil-solmuja helpottaa potentiaalisesti pimennyshyökkäysten toteuttamista, mutta se ei ole välttämätön, koska myös joukko yhdessä hyökkäävästi toimivia ei-Sybil solmuja voi toteuttaa pimennyshyökkäyksen, mikäli niihin ei ole erikseen varauduttu.

Suoraviivaisin tapa pimennyshyökkäysten torjuntaan on mahdollistaa avaimen haltijan löytäminen montaa eri reittiä. Tämä voi tarkoittaa joko reititysjärjestelmää, missä yhteen kohteeseen on mahdollista löytää monta reittiä tai saman avaimen tietojen sijoittamista monelle eri solmulle.

Laajempia puolustusmekanismeja pimennyshyökkäyksille on kehitelty vähemmän kuin identiteetin varmistukseen. Lisäksi monet näistä menetelmistä olettavat, et-

tä hyökkääjän solmut ovat jakautuneet laajalti tunnisteavaruuteen ja pyrkivät häiritsemään koko verkon toimintaan. Tämä ei kuitenkaan ole välttämättä paikkansapitävä oletus erityisesti avaimen ja solmutunnisteen etäisyyteen perustuvissa järjestelmissä, kuten Kademliassa. Tämäntyyppisissä verkoissa hyökkääjä voi pyrkiä pimittämään vain tiettyjä avaimia eli keskittää häiritsevät solmut kohdeavaimen lähistölle[UPS11]. Jälkimmäisen tyyppisiin hyökkäyksiin auttaa tosin myös sertifioitujen identiteettien jakaminen, olettaen että järjestelmässä solmut eivät voi itse päättää tunnistettaan. Tällöin hyökkääjä joutuu investoimaan suuremman määrän resursseja hyökkäykseen.

Castro et al. esittävät mallia, jossa solmut ylläpitävät kahta reititystaulua. Näistä ensimmäinen kuvaa järjestelmän normaalia, mahdollisimman tehokasta reititystä ja toinen *varmistettua* reititystä. Jälkimmäistä taulua käytetään, mikäli solmu epäilee, että edellisen palauttamat tulokset eivät ole luotettavia. Varmistetun reititystaulun reitteihin liittyy jokin tarkastettavissa oleva invariantti, mutta sen tarjoama reitti ei välttämättä ole verkkoteknisesti paras. Tarkka toteutus on kiinni suojattavasta järjestelmästä, mutta kriteeri voi olla esimerkiksi, että reitin kohdesolmun tunniste eroaa reitin tarjoajan tunnisteesta vain yhdellä tavulla[CDG+02]. Tämä menetelmää tukee turvattujen identiteettien jakaminen, jotta hyökkääjä ei voi vain ottaa haltuunsa myös varmistetun reititystaulun määrittämiä kohteita.

Tämän järjestelmän ongelma on lisäksi, että hyökkäyksen jatkuessa normaalin reititystaulun laatu laskee jatkuvasti ja reitityksessä joudutaan käyttämään hitaampaa varmistettua reititystä. Condie et al. esittävät tähän menetelmää, jossa osa verkon solmuista korvaa satunnaisin väliajoin normaalin reititystaulun sisällön varmistetulla reititystaululla ja antaa taulun optimoitua uudelleen muiden solmujen poistuessa ja saapuessa. Menetelmä tosin edellyttää myös, että jokaisella uusintakerralla solmu vaihtaa tunnistettaan. Tämä lisää huomattavan määrän poistumis- ja liittymistrafiikkia verkkoon[UPS11].

Singh et al.[SNDW06] esittävät tilastollisista menetelmää, joka perustuu havaintoon, että kohdennetussa hyökkäyksessä paljon reititystietoja osoittaa hyökkääjäsolmuihin, eli niiden indegree on suuri. Lisäksi mikäli hyökkääjät ovat saaneet aidon solmun reititystauluun lisättyä suuren määrän omia tietojaan väärän reititystiedon levittämiseksi, tämän solmun outdegree saattaa olla suuri. Tällöin tavoitteena on tunnistaa solmut, joiden aste ylittää rajan. Mikäli rajaksi asetetaan tarkalleen järjestelmän “tyypillinen” aste, menetelmä rajoittaa hyökkääjien osuudeksi annetun solmun naapurijoukosta korkeintaan $\frac{f}{1-f}$, missä f on hyökkääjien osuus koko verkon

solmujen määrästä.

Menetelmä edellyttää, että normaalisolmu x tietää, kenellä kaikilla on viittaus siihen reititystauluissaan eli selvittää itseensä kohdistuvan *takaisinosoitusjoukon*. x käyttää kolmatta, satunnaisesti valittua *anonymisoijasolmua* y ja selvittää ajoittain tämän kautta kultakin itseensä viittaavalta solmulta z niiden reititystaulun. Mikäli x itse ei ilmene palautetussa reititystaulussa, reititystaulun koko on odotettua isompi tai vastausta ei tule, x voi päätellä, että kyseinen z saattaa olla hyökkääjä. Anonymisoijan rooli on varmistaa, että z ei tiedä kuka on kysyjä. Tällöin se ei voi väärentää vastausta jossa tarkastaja x olisi aina mukana ja jättää muita naapureita pois.

Lisäksi x kyselee omilta naapureiltaan niiden takaisinosoitusjoukot ja varmistaa, että on itse mukana niissä ja että niiden koot ovat hyväksyttävissä rajoissa. Menetelmä toimii, mikäli y on normaalisolmu. Mikäli myös y on hyökkääjä, se voi joko johtaa x:n uskomaan, että hyökkääjä z on normaalisolmu tai että normaalisolmu z on hyökkääjä. Menetelmä pyrkii välttämään tätä tilannetta valitsemalla y:n täysin tai osittain satunnaisesti.

Yksinkertaisessa simulaatiotestissä menetelmä vähensi 10000 solmun verkossa hyökkääjien osuuden testatun protokollan reititystaulun useimmin käytetystä osasta 78 prosentista 41 prosenttiin kymmenen tunnin sisällä. Paremmin todellisen Internetin olosuhteita vastaavassa simulaatiossa vastaavat tulokset saavutettiin kuitenkin vain alle 2000 solmun testiverkolla. Lisäksi menetelmän ongelmia ovat, että se lisää verkon latenssia, koska se rajoittaa reititystaulujen kokoja, eikä tarjoa apua kohdennettuihin hyökkäyksiin[SNDW06].

Tiettävästi mitään näistä monimutkaisemmista menetelmistä ei ole sovellettu käytännön DHT-toteutuksissa. Tässä esitellyt menetelmät auttavat jossakin määrin pimennyshyökkäyksen tunnistamisessa, mutta toisaalta ne asettavat uusia rajoituksia järjestelmille ja lisäävät verkon yleiskustannuksia.

Yleisemmin ongelma, joka pimennyshyökkäysten suhteen pitäisi ratkaista, on ennustamattomalla tavalla väärin käyttäytyvien solmujen tunnistaminen eli niin sanotun Bysantin kenraaleiden ongelman ratkaiseminen. Tällaiset järjestelmät perustuvat siihen, että verkon solmuja käsitellään *tilakoneina*, joiden tila muuttuu verkko-operaatioiden seurauksena. Perinteisissä menetelmissä tämän tilakoneen tilaa replikoidaan usealle tarkastajalle, jotka varmistavat tilasiirtymien oikeellisuuden[CL99]. Perinteisen näkemyksen mukaan tämäntyyppisten menetelmien toteutus vertaisverkkoympäristössä ei kuitenkaan ole realistista solmujen suuren määrän ja suu-

ren kommunikaatiotarpeen vuoksi. Vuonna 2012 Young et al. esittivät kuitenkin useita uusia menetelmiä, jotka parantaisivat olemassa olevia menetelmiä tämän ongelman ratkaisemiseen yhdellä tai kahdella kertaluokalla[YKGK12]. Toinen mahdollisuus on tarkkailla *naapureiden* käyttäytymistä ja pyrkiä tilakonemallin perusteella toteamaan virheet niiden käytöksessä[HKD07]. Menetelmät, jotka pyrkivät ratkaisemaan tämän yleisen ongelman ovat hyvin laaja aihe. Niitä ei käsitellä tarkemmin tässä tutkielmassa.

4 Kademlia-pohjaisten DHT:jen käyttö

Tämä luku esittelee lyhyesti julkisten DHT:jen käyttäjämääriä, käyttöaktiivisuutta sekä tietosisältöä. Tämä tieto toimii taustana myöhempään järjestelmien turvallisuuden tarkasteluun ja turvallisuusmekanismien skaalautuvuuteen liittyviin haasteisiin.

4.1 Ohjelmistot ja käyttäjämäärät

Huolimatta niiden suosiosta tutkimuksessa, DHT:jen käyttö laajalle levinneissä sovelluksissa on ollut vähäistä. Vuonna 2008 tehdyn tutkimuksen mukaan ainoat suurimittaiset verkot pohjautuvat Kademlia-algoritmiin[Ste09]. Näitä olivat silloin eMule-tiedostonjako-ohjelman käyttämä Kad-verkko sekä Vuze-nimisen BitTorrent-asiakasohjelman Kademlia-pohjainen verkko (jatkossa *Vuze-DHT*; tämä verkko tunnetaan myös nimellä *Azureus-DHT* ohjelman entisen nimen mukaan). Näiden lisäksi uudempi tulokas on alkuperäisen BitTorrent-asiakasohjelman kyljessä kehitetty *päälinja-DHT* (*BitTorrent Mainline DHT*)[Loe08] (Vuze-ohjelman käyttäjät voivat käyttää kumpaakin asentamalla sopivan lisäohjelmiston).

Kad-verkon kautta käyttäjä voi etsiä avainsanoja, jotka osoittavat *lähdeavaimiin*, jotka ovat jaettavista tiedostoista laskettuja hajautusarvoja. Lähdeavaimet osoittavat tiedostoja jakavien käyttäjien IP-osoitteisiin. Vuze-DHT:ssa ei ole avainsana-avaimia, vaan DHT:a käytetään vain muualta hankitun `.torrent`-tiedoston pohjalta saadun avaimen kautta tiedostoa jakavien käyttäjien löytämiseen[Ste09]. Päälinja-DHT:n toiminallisuus on vastaava kuin Vuze-DHT:n, mutta toteutukset eivät ole yhteensopivat[Loe08]. BitTorrent-asiakasohjelmien tapauksessa DHT-toiminnallisuus on suhteellisen myöhään saapunut lisäominaisuus, jota käyttäjä voi käyttää rinnakkain perinteisten tracker-palvelimien kanssa tai niiden sijaan. eMule ja muut

Kad-verkkoa käyttävät ohjelmat voivat käyttää joko Kad-DHT-pohjaista hakua tai eDonkey2000-protokollaa käyttäviä hakupalvelimia jaettujen tiedostojen löytämiseen.

Mahdollisesti johtuen hakupalvelinten ylläpitäjiin kohdistuneista tekijänoikeusloukkauksiin liittyvistä oikeusjutuista, eMulen DHT-toiminnallisuus on ollut enemmän esillä julkisuudessa ja kehityksen ja tutkimuksen kohteena kuin BitTorrent-DHT:t. Myös BitTorrent-tracker-palvelimien ylläpitäjiin on kohdistunut oikeusjuttuja ja vuonna 2009 The Pirate Bay, eräs suurimmista BitTorrent-trackerien ylläpitäjistä, ilmoitti sulkevansa tracker-palvelimet ja suositteli DHT-käyttöön siirtymistä[Pir09].

Kad-verkossa oli tutkijoiden mukaan vuonna 2004 kellonajasta riippuen noin 3–4,3 miljoonaa solmua, joista tosin mittauspalvelin sai suoran yhteyden vain noin 1,5–2 miljoonaan[Ste09]. Tämä johtunee siitä, että osa solmuista on palomuurien ja NAT-käytävien takana, jotka estävät sisääntulevat yhteydet. Azureus-DHT:ssa oli samassa tutkimuksesta kellonajasta riippuen noin 1–1,4 miljoonaa solmua, joista ajankohdasta riippuen vain noin 100.000–400.000 vastasivat niille lähetettyihin paketteihin.

JMule-asiakasohjelman kehittäjien ylläpitämän tilaston mukaan Kad-verkossa vieraili tammikuussa 2011 noin 2,2 miljoonaa uniikkia tunnistetta käyttävää asiakasta[JMu11]. Vuonna 2011 julkaistun verkon läpikäyntiin perustuvan tutkimukseen mukaan päälinja-DHT:lla oli tyypillisesti 6–7 miljoonaa käyttäjää[YXLZ11]. Vuze-ohjelma arvioi sekä oman DHT:nsa että päälinja-DHT:n käyttäjien määrää algoritmilla, joka on tyypillisesti paikkansapitävä noin 5 prosentin tarkkuudella[Ste09]. Ohjelman arvioiden mukaan helmikuussa 2011 Vuze-DHT:lla oli tyypillisesti noin 2,3-2,9 miljoonaa käyttäjää ja päälinja-DHT:lla noin 5,6-7,6 miljoonaa käyttäjää. Käyttäjämääristä ei valitettavasti ole olemassa julkisia pitkän aikavälin tilastoja, joten esimerkiksi oikeusjuttujen, asiakasohjelmien muutosten ja muiden ulkoisten tekijöiden vaikutusta käyttäjämääriin ei voi tutkia helposti jälkikäteen.

Joskin käyttäjämäärät ovat absoluuttisesti melko suuria, ne eivät vastaa tiedostonjako-ohjelmien käyttäjämääriä. Esimerkiksi vuonna 2008 µTorrent-BitTorrent-asiakasohjelman kehittäjät ilmoittivat, että ohjelmalla oli tällöin 28 miljoonaa kuukausittaista käyttäjää[Tor08] ja useista korkean profiilin oikeusjutuista sekä uudesta lainsäädännöstä huolimatta tiettävästi käyttäjämäärät eivät ole tiettävästi olennaisesti ainakaan vähentyneet. Todennäköisesti tämä johtuu pääasiassa siitä, että useimmat käyttäjät pitävät ohjelmia käynnissä vain kun tarvitsevat niitä. Käyttäjät voivat myös poistaa asiakasohjelmissa DHT-toiminallisuuden käytöstä.

“Virallisten” käyttötarkoitusten lisäksi verkkoja voivat käyttää myös muut kuin

ohjelmat, joille ne on alunperin suunniteltu. Yksi tällainen on aiemmin mainittu Vanish-ohjelma[GKLL09]. Käyttäjät eivät välttämättä ole kuitenkaan tervetulleita vieraita. Vuonna 2007 laajalle levinnyt Storm-mato käytti Kad-verkkoa edeltänyttä Overnet-DHT:a (alunperin kummankin protokolla oli sama) komentokanava, jolla madon tekijä kontrolloi kaappaamiaan koneita. Mato etsii verkosta avainta, jonka se muodostaa päivämäärän ja satunnaisluvun perusteella. Mikäli madon tekijä haluaa antaa tartunnan saaneille koneille käskyjä tiettynä päivänä, hän tallentaa näiden avainten tietosisällöksi oman hallintapalvelimensa osoitteen, johon tartunnan saanut kone sitten ottaa yhteyden hakeakseen toimintakäskyjä.

Uudemmissa versioissa mato siirtyi yleisestä Overnetista omaan verkkoonsa. Tämä verkko noudattaa lähes täysin samaa protokollaa, mutta sen "käyttäjinä" ovat ainoastaan madon saastuttamat koneet[HSD+08].

4.2 Verkkojen tietosisältö ja käyttöaktiivisuus

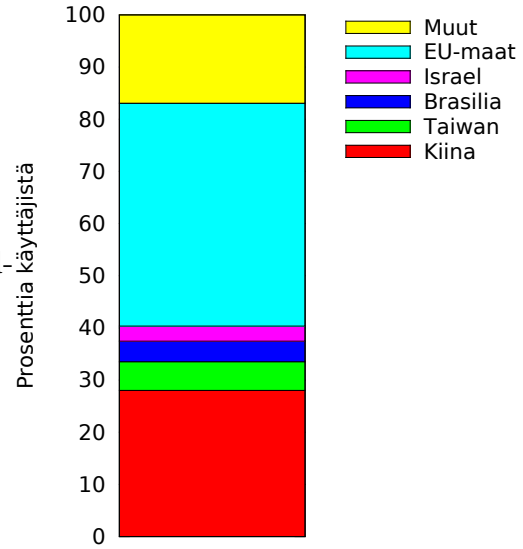
Siltä osin kuin suuria julkisia DHT:ja käyttävät ohjelmat ovat tiedostonjako-ohjelmia joille ne on alunperin perustettu, eivätkä myöhemmin tulleita sovelluksia, verkkojen tietosisältö muodostuu tiedostoja jakavien asiakasohjelmien osoitteista sekä Kadin tapauksessa myös viittauksista avainsanoista tiedostoihin ja niiden metadataan. "Aitojen" käyttäjien osuus verkossa verrattuna uusiin käyttäjiin, hyökkääjiin ja vanhentuneisiin reititystaulujen alkioihin ei kuitenkaan ole helposti mitattavissa.

Joillakin Internet-operaattoreilla asiakkaiden IP-osoitteet vaihtuvat esimerkiksi kerran vuorokaudessa, josta seuraa, että reititystauluissa näkyy paljon solmuja, jotka eivät vastaa pyyntöihin. Toisaalta esimerkiksi Kadin tapauksessa vuonna 2009 tehdyssä tutkimuksessa tutkijat löysivät muun muassa kymmeniä tuhansia solmuja, jotka käyttivät kaikki samaa tunnistetta sekä useita muita artifakteja. Monet näistä samaa tunnistetta käyttävistä solmuista vastasivat ainakin osaan protokollan viesteistä. Suurin osa näistä epätavallisista solmuista sijaitsi IP-osoitteen perusteella Kiinassa[YFX+09]. Sekä tästä että aiemmista tutkimuksista Storm-madon DHT-käytön suhteen voidaan päätellä, että todennäköisesti sekalaiset tuntemattomat käyttäjät muodostavat vähintäänkin merkittävän osan Kadin käyttäjistä. Potentiaalisesti BitTorrent-DHT:illa on samankaltaisia käyttäjiä, mutta näistä ei toistaiseksi ole olemassa selkeitä tutkimustuloksia.

Kadin tapauksessa suuri osa käyttäjistä oli vuonna 2006 IP-osoitteiden perusteella Kiinasta (noin 24 prosenttia), mutta EU-maat kuten Espanja, Ranska, Italia

ja Saksa muodostavat yhdessä prosentuaalisesti suurimman käyttäjäblokin. Maakohdistuksen luotettavuutta tukee käyttäjien aktiivisuusjakaumaa, jossa esimerkiksi kiinalaisiksi oletettujen käyttäjien määrä kasvaa selvästi iltaisin ja iltapäivisin. Sessioiden pituudet vaihtelevat huomattavasti, mutta ovat keskimäärin noin neljän tunnin luokkaa. Vain 20 prosenttia verkossa kaksi tuntia olleista solmuista on paikalla vielä 24 tunnin päästä. Tästä seuraa, että avaimet on syytä tallettaa monelle solmulle, jotta ne pysyisivät saavutettavina[SENB09]. JMule-projektin keräämien tilastojen perusteella IP-osoitteista johdettu käyttäjien maajakauma oli tammikuussa 2011 vastaavaa suuruusluokkaa Kiinan ja EU:n välillä, joskin yksittäisten EU-maiden osuudet olivat muuttuneet[JMu11]. Tämä vuoden 2011 käyttäjäkuntien jakauma Kad-verkossa näkyy kokonaisuudessaan kuvassa 9. Merkittävästi amerikkalaiset eivät erotu kummassakaan tutkimuksessa merkittävänä joukkona. Tämä kertonee siitä, että tiedostonjako-ohjelmien käyttäjät eivät ole maailmanlaajuinen, homogeeninen joukko, vaan käyttötavat vaihtelevat alueittain.

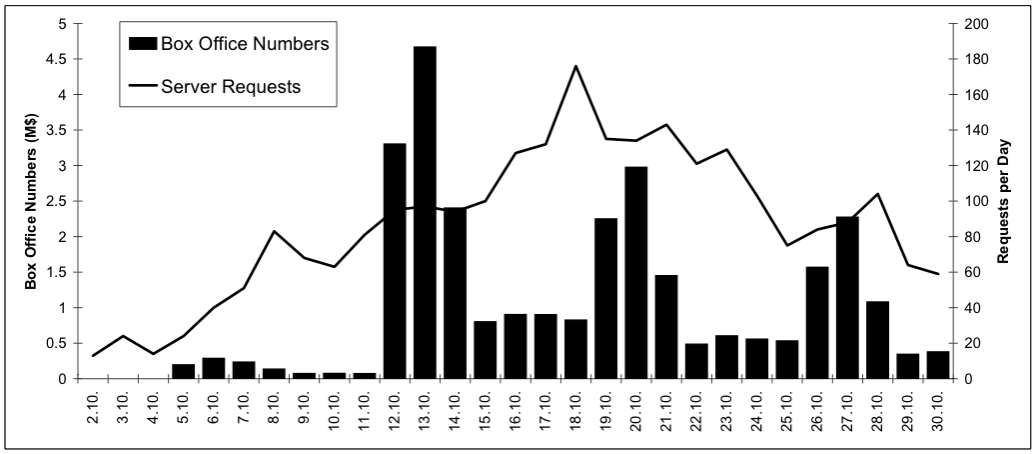
Yksittäisten identiteettien käyttöajat eroavat huomattavasti maittain. Kiinalaiset käyttäjät käyttävät verkkoa samalla identiteetillä ainakin 150 päivää vain 10 prosentin todennäköisyydellä, kun eurooppalaisilla tämä todennäköisyys on noin 40 prosenttia. Liikennemäärien osalta Kad-verkossa avainten julkaisusta aiheutuva liikenne on noin sata kertaa suurempi kuin hauista aiheutuva liikenne, mikä on melko huono hyötysuhde. Tätä tilannetta voisi mahdollisesti parantaa pyrkimällä valitsemaan tallettaviksi solmuiksi sellaiset solmut, jotka ovat tiettävästi olleet jo aiemmin pitkään saavutettavissa [SENB09].



Kuva 9: Käyttäjien jakautuminen maittain Kad-verkossa tammikuussa 2011. Tietojen lähde [JMu11].

Vuze-DHT:n käyttökuvio vastaa Kadia, mutta käyttäjän maantieteellinen jakautuma eroaa huomattavasti. Selvästi suurin käyttäjämaa on Yhdysvallat ja kuten Kadissa, useat EU-maat yhdessä muodostavat toisen suuren blokin. Kiinalaiset sen sijaan eivät edes mahdu 10 suurimman käyttäjämaan joukkoon[Ste09].

Koska minkään järjestelmän käyttäjäidentiteetit eivät ole kiinteitä, ei myöskään



Kuva 10: Elokuvateattereissa pyörivän elokuvan ja sen suosion suhde Kad-verkossa. Kuvan lähde [LMSW09].

ole olemassa kattavaa tietoa verkkojen kokonaiskäyttäjämääristä, pois lukien yksittäisten ohjelmistovalmistajien markkinointimateriaaleissa ajoittain julkaistut luvut. Näin ollen järjestelmien kokonaissuosiota väestön keskuudessa on hankala arvioida.

Kad-verkon tapauksessa on mahdollista tarkastella tiedostonjako-ohjelmissa DHT:ssa julkaisemia avainsanoja ja selvittää jaettujen tiedostojen luonnetta. eMule-ohjelma ja muut vastaavan toiminallisuuden tarjoavat Kad-verkkoa käyttävät ohjelmat jakavat tiedostojen nimet paloiksi ja julkaisevat nämä palaset avainsanoina. Tästä seuraa, että suurin osa verkon avainsanoista on tiedostopäätteitä tai tiedostojen yleisluontoisia kuvauksia, esimerkiksi `dvd` tai `music`. Näistä voi päätellä, että suurin osa verkossa jaetusta sisällöstä on elokuva- ja musiikkitiedostoja. Suurta osaa sisällöstä ei myöskään koskaan haeta. Vuonna 2007 verkossa oli julkaistu noin 42000 avainsanaa, mutta tarkkailun aikana vain 1100 erilaista hakusanaa haettiin[Ste09]. Spesifisten tiedostojen hakujen tarkastelu näyttää, että verkossa suosittu sisältö vastaa sen ulkopuolella suosittua sisältöä. Esimerkiksi elokuvan suosio teattereissa korreloi positiivisesti kyseisen elokuvan tiedoston hakujen määrään[LMSW09]. Kuvassa 10 näkyy elokuvan tuotto elokuvateattereissa ja siihen kohdistuvien hakujen määrä Kad-verkossa esimerkkielokuvan osalta.

BitTorrent-asiakasohjelmien DHT:t eivät sisällä avainsanoja, joten puhdas DHT:n sisällön tarkastelu ei anna suoraa tietoa jaetuista tiedoista. Avainten arvoina toimivat torrenttien *infohash*-hajautusluvut ovat kuitenkin monessa tapauksessa samoja, kuin perinteisillä seurantapalvelimilla levitettyjen `.torrent`-tiedostojen luvut. Näin ollen on mahdollista hakea suuria määriä nimen sisältäviä `.torrent`-tiedostoja ja yhdistää DHT:n sisältämät infohash-arvot näihin tiedostoihin, jolloin vastaavan tie-

doston nimi käy selville. Vuze-DHT:n sisällöstä vuonna 2010 tehdyssä selvityksessä suuri osa sisällöstä (noin 54%) oli videotiedostoja, jota seurasivat äänitiedostot (noin 23%). Kuten Kadin tapauksessa, yksittäisten tiedostojen suosio korreloi pitkälti niiden sisällön suosioon esimerkiksi elokuvateattereissa ja televisiossa[WH10].

Päälinja-DHT:n suhteen ei ole olemassa vastaavaa tutkimustietoa käyttäjien sijainnin, käyttötapojen tai tietosisällön osalta, mutta Kadista ja Vuze-DHT:sta saadun tiedon perusteella on todennäköistä, että ainakin käyttötavat ja tietosisältö ovat samankaltaisia. Koska DHT:ja mahdollisesti käyttävät muut sovellukset eivät noudata samaa alkioiden formaattia kuin tiedostonjako-ohjelmat ja tieto saattaa olla salattua tai infohash-arvojen tapaan läpinäkymätön tunniste, tästä tietosisällöstä on vaikea tuottaa yleisluontoista yhteenvetoa.

5 DHT:jen mittausmenetelmät

5.1 Mittauksen tyypit ja tavoitteet

DHT:jen tehokkuuden, käytön ja turvallisuuden tutkimus vaatii sekä algoritmien että kentällä olemassaolevien verkkojen mittausta. Itse algoritmeja on mahdollista testata verkkosimulaattoreissa sekä testauksen suorittajan omista palvelimista muodostetuissa suljetuissa testiverkoissa. Simulaatioilla ja suljetuilla verkoilla on mahdollista testata algoritmien ja toteutusten tehokkuutta ja skaalautuvuutta tiedonsiirto- ja laskentamäärien kannalta, näiden vakautta sekä vastustuskykyä vikatilanteille[KGK+08]. Simulaatiotestaus ei kuitenkaan ole aina riittävä menetelmä, koska todellisuudessa toteutusten kohtaamat verkkotopologiat, alla olevan tietoverkon viiveet, vihamielisesti tai muutoin väärin toimivat asiakasohjelmat ja muut erityistilanteet eivät ole helposti kattavasti mallinnettavissa tai ennustettavissa. Lisäksi tuotantokäytössä olevien verkkojen todellista liikennettä ei luonnollisesti voi mitata simulaatioissa. Koska tämä tutkielma keskittyy pääasiassa tuotantoverkkoihin, tämä luku käsittelee vain tuotantoverkkojen mittausmenetelmiä.

Käytännössä DHT:jen tapauksessa “mittaus” merkitsee yleensä jonkin taulukossa 1 mainituista tutkimuskohteista mittausta.

Merkitykset 1 ja 2 ovat menetelmiltään samankaltaisia, koska niihin soveltuvat aktiiviset menetelmät, joissa verkon solmuilta kysellään niiden tietosisällöstä tai muutoin pyritään onkimaan niiltä tietoja. Näihin voi soveltaa myös passiivisia menetelmiä. Tämä ei välttämättä ole tehokasta kattavan kuvan saamiseksi, mutta toisaalta

Taulukko 1: Mittauksen tai tutkimuksen kohteet DHT:issa.

1. *Kytkeytyneiden toteutusten tutkimus*; millaisilla ohjelmistoilla käyttäjät ovat kytkeytyneet verkkoon, miten suuri osa reititystauluissa olevista solmuista vastaa kyselyihin.

2. *Tietosisällön tutkimus*; miten paljon ja minkä tyyppistä tai sisältöistä tietoa verkon solmut varastoivat.

3. *Verkon rakenteen tutkimus*; verkon topologian ja osanottajamäärän selvittäminen.

4. *Liikenteen tutkimus*; miten paljon ja minkä tyyppistä liikennettä verkossa tapahtuu.

passiivinen liikenteen tarkkailu voi paljastaa enemmän siitä, mikä verkon sisällä on suosittua.

Mittaus merkityksessä 3 edellyttää joko reititystaulujen passiivista keräämistä omilta solmuilta tai niiden aktiivista kyselemistä muilta solmuilta (tämän mahdollistavilla protokollilla). Merkityksessä 4 mittaus on periaatteessa passiivista (esimerkiksi solmun välittämän tiedon keräystä), mutta siihen saattaa menetelmästä riippuen liittyä myös aktiivisia komponentteja mittaussolmujen sijoittamiseksi sopiviin paikkoihin tai lisäliikenteen puoleensavetämiseksi. Tämäntyyppisessä mittauksessa tavoitteet ovat enemmänkin kvantitatiivisia (liikenteen määrä, mahdollisesti kategorioihin rajattuna), kun taas tyypeissä 1 ja 2 liikenteen analyysin tavoitteet ovat enemmän kvalitatiivisia (verkon sisällön laatu).

Näiden menetelmien suhteen on kuitenkin syytä huomata, että yleensä “passiivinen” ei tarkoita täysin sivullista tarkkailua esimerkiksi pakettikaappauksella. Tällaisen aineiston saatavuus merkittäviä määriä välittävistä verkon pisteistä on heikko ja tämäntyyppisiin pakettikaappauksiin liittyy merkittäviä yksityisyydensuojaongelmia. Tässä “passiivisella” tarkoitetaankin menetelmiä, jossa tutkijan hallinnoima solmu liittyy DHT-verkkoon ja toimii protokollan näkökulmasta normaalisti verkon osana, mutta lisäksi tallettaa normaalisti väliaikaiseksi tarkoitettua tietoa, kuten reititystaulujen tai välitetyn liikenteen sisältöä. Näihin menetelmiin voi liittyä myös selkeämmin aktiivisia komponentteja, kuten esimerkiksi aiemmin mainittu tutkimussolmun sijoittelun optimointi.

Joskin peer-to-peer-verkkojen tutkimukseen sisältyy myös tyypillisesti tiedostonjakokäytössä olevien verkkojen tietosisällön tutkimus, spesifisesti DHT-algoritmeihin ja -toteutuksiin kohdistuva tutkimus keskittyy yleensä taulukossa 1 lueteltuihin tutkimustyyppeihin 3 ja 4. Joissakin tutkimuksissa tuloksiin on kerätty lisäksi myös tyyppeihin 1 tai 2 liittyviä tietoja, mutta enemmän yleiskuvana verkon luonteesta kuin spesifisiin tietotyyppeihin keskittyvänä selvityksenä.

Seuraavat aliluvut esittelevät eräitä menetelmiä DHT:jen mittaamiseen ja tarkkailuun. Nämä menetelmät ovat olennaisia tutkimukselle tosimaailman verkoista, mahdollisten hyökkäysten ja poikkeavan käytön tunnistamiselle sekä puolustusmekanismien kehitykselle. Toisaalta jotkin niistä liittävät verkkoon solmuja, jotka toimivat poikkeavilla tavoilla normaalikäyttäjien asiakasohjelmista mittaustiedon keräämiseksi. Tällaiset solmut edustavat myös eräänlaista hyökkäystä DHT:jen “toivottua” rakennetta vastaan, vaikka välttävätkin vaikuttamasta verkon toimintaan lopputulosten osalta.

Luku 5.2 käsittelee aktiivisia mittausmenetelmiä ja luku 5.3 yllä kuvatun kaltaisia “passiivisia” menetelmiä.

5.2 Aktiiviset mittausmenetelmät

Aktiivinen mittaus merkitsee verkkoon liittyneiden solmujen läpikäyntiä (*crawling*). Kademliassa solmujen tunnisteet kuuluvat samaan avaruuteen kuin avaimet, joten käytännössä tämä merkitsee tunnisteavaruuden alijoukon läpikäyntiä verkon reititystaulujen perusteella.

Steiner et al. kehittämällä *Blizzard*-nimisellä rinnakkaistoteutuksella 1.5–2 miljoonan solmun verkko oli mahdollista käydä läpi hieman yli 10 minuutissa, kun tavoitteena oli todeta vain vastaako kohdesolmu vai ei. Periaatteessa solmut valitsevat tunnisteensa satunnaisesti, mikä myös toteutui pääosin vuonna 2006 tehdyissä mittauksissa; solmut olivat jakautuneet tasaisesti tunnisteavaruuteen, lukuun ottamatta muutamia ilmeisesti käsin valittuja erityisillä tavoilla levittäytyneitä tunnisteita. Tästä tasaisesta jakaumasta seuraa, että on mahdollista ja tutkimustehokkuuden kannalta hyödyllistä valita jokin tunnisteavaruuden aliavaruus tutkimuskohteeksi. Esimerkiksi tunnisteet, joilla oli tietty yhteinen 8-bittinen tunnisteen etuliite, oli mahdollista läpikäydä noin 2,5 sekunnissa. Tämä vastaa yhtä 256-osaa verkon tunnisteavaruudesta[SBEN07].

Kademlian tunnisteavaruutta voi käydä läpi leveyssuunnassa (*breadth-first search*)

tai syvyyssuunnassa (*depth-first search*). Ensimmäisessä läpikäyjä lähettää hakuja avaimille, jotka valitaan tasaisesti tarkasteltavassa tunnisteavaruudessa. Tällöin vastaukset sisältävät myös solmuja, jotka ovat tarkasteltavan tunnisteavaruuden osan ulkopuolella. Jälkimmäisessä avaimet valitaan suoraan tunnettujen solmujen tunnisteista ja kerätään uudet tarkastettavat avaimet näiltä saaduista reititysvastauksista. Tällöin vastaukset ovat pääosin samassa osassa tunnisteavaruutta[YXLZ11].

Blizzard käytti ainoastaan leveyssuuntaista läpikäyntiä, mutta myöhempi Yu et al. kehittämä toteutus *Husky* [YXLZ11] yhdistää nämä strategiat. Leveyssuuntainen läpikäynti kerää aluksi nopeasti paljon solmuja, mutta sen keräysnopeus hidastuu, koska se alkaa myöhemmin kohdata paljon jo-nähtyjen solmujen osoitteita. Tässä vaiheessa on kannattavaa vaihtaa syvyyssuuntaiseen läpikäyntiin, jolloin kerääjä saa kerättyä jo-tunnettujen solmujen lähellä sijaitsevien solmujen osoitteita tehokkaasti. Tutkijoiden toteutus tästä yhdistelmästrategiasta oli noin kolme kertaa tehokkaampi kuin Blizzardin käyttämä suoraviivaisempi strategia[YXLZ11].

Läpikäyntitoteutuksia on mahdollista siirtää kohtuullisen helposti eri Kademliatoteutusten välillä. Esimerkiksi edellä mainittu Blizzard-toteutus oli tehty Kadverkkoa varten, mutta tutkijat tekivät siitä sittemmin version, joka toimi Vuze-DHT:n kanssa[SB08]. Huskyn kohde taas oli päälinja-DHT[YXLZ11].

Nopeasta läpikäynnistä on etua, koska solmujen liittyminen ja eroaminen verkosta muuttavat rakennetta jatkuvasti. Tästä syystä mitä nopeampi läpikäynti on, sen johdonmukaisempi saatu kuva verkosta on[SBEN07]. Mikäli kohteena ei ole ainoastaan solmujen saavutettavuus vaan niiden tietosisältö, läpikäynti vaatii enemmän aikaa, mutta toisaalta rakenteellinen johdonmukaisuus ei ole yhtä kriittistä.

Verkon tietosisällön selvityksen menetelmät riippuvat verkkoa käyttävästä sovelluksesta. Esimerkiksi BitTorrent-DHT:jen tapauksessa asiakasohjelma käyttää DHT:a tiettyjä tiedostoja jakavien muiden käyttäjien löytämiseen, mikäli keskitettyjä trackerpalvelimia ei ole käytettävissä. Avaimia toimivat `.torrent`-metatietotiedostojen niin sanotut *infohash*-arvot. Koska DHT itsessään ei sisällä jaettujen tiedostojen sisältöä tai niiden metatietoa, BitTorrent-jakojen tutkimuksessa on tarvetta kerätä DHT:n sisältämien tietojen lisäksi myös `.torrent`-metatietotiedostoja. Nämä tiedostot ovat tehokkaimmin saatavilla BitTorrent-hakusivustoilta, joista tunnetuin on luultavasti ruotsalainen Pirate Bay.

Liittämällä metatietojen sisältämä tiedoston nimi DHT:n sisältämiin tiedostoa jakavien BitTorrent-solmujen IP-osoitteisiin, on mahdollista selvittää ketkä jakavat mitä tiedostoja[WH10]. Toisin sanoen DHT:jen mittaus saattaa vaatia lisätietojen

keräystä ulkoisista lähteistä merkityksellisten tulosten saavuttamiseksi, mikäli tavoitteena on tutkia muutakin kuin itse verkon ominaisuuksia.

5.3 Passiiviset mittausmenetelmät

Passiivisissa mittausmenetelmissä yleensä yksi tai useampi normaalin DHT-asiakasohjelman tavoin toimiva asiakas liittyy olemassa olevaan verkkoon ja tallentaa saamansa pyynnöt. Periaatteessa termi voi viitata myös täysin sivulliseen pakettikaappaukseen pohjautuvaan tarkkailuun, mutta luvussa 5.1 esitetyistä syistä johtuen merkittävän kokoisten aineistojen kerääminen tällä menetelmällä on hankalaa. Mikäli tällainen aineisto on saatavissa, siihen voidaan periaatteessa soveltaa samantyyppisiä analyysimenetelmiä kuin verkkoon osallistuvien asiakkaiden keräämästä seurantatiedosta. Vaikka pakettikaappaus saattaa olla hyödyllinen menetelmä yksittäisen tietoverkkojen käytön valvonnassa, sen rooli DHT:jen itsensä tutkimuksessa on rajallinen, joten sitä ei spesifisesti käsitellä enempää tässä tutkielmassa.

Suoraviivainen instrumentoitujen asiakasohjelmien käyttö mittauksessa on kuitenkin ongelmallista. Pieni määriä mitattavia solmuja ei näe kuin pienen osan liikenteestä, kun taas suuri joukko saattaa itsessään vaikuttaa verkon rakenteeseen ja toimintaan. Kummassakaan tapauksessa kerätty aineisto ei välttämättä anna kokonaiskuvaa verkon “luonnollisesta” toiminnasta[MRGS09]. Esimerkiksi mikäli mittaussolmut pysyvät vakaasti verkossa mukana pidempään kuin tyypillinen käyttäjä, niiden olemassaolo pidentää niille kuuluvien avainten elinikää verkossa sekä vähentää reititystaulujen muutosta, mikäli asiakasohjelmien toteutus suosii reititystauluissa pitkään saavutettavissa olleita naapureita.

Memon et al. [MRGS12] ovat esittäneet yleisesti DHT:iden toimintaperiaatteeseen soveltuvan menetelmän, jossa tarkkailijat ovat erityisiä, rajoitettuja toteutuksia DHT:n protokollasta. Nämä solmut pyrkivät näkymään verkossa niin vähän kuin mahdollista asettumalla olemassaolevien solmujen viereen tunnisteavaruudessa, jolloin ne saavat kopion kohdesolmun liikenteestä. Tätä menetelmää ja sen sovellusta Kademlia-pohjaisiin verkkoihin kuvaillaan tarkemmin luvussa 6.3.

6 Hyökkäykset Kademlia-pohjaisia DHT:ja vastaan

Aiemmat luvut ovat käsitelleet vertaisverkkojen ja DHT:jen turvallisuutta yleisellä tasolla, sekä Kademlia-pohjaisten verkkojen käyttöä ja mittaus- ja tarkkailumenetelmiä. Tämä luku tarkastelee erityisesti Kademlia-pohjaisiin verkkoihin liittyviä tarkkailu- ja häirintäuhkia, sekä näitä vastaan esitettyjä puolustusmekanismeja.

6.1 Hyökkäysmahdollisuudet

Kademlia-pohjaiset verkot ovat luonnollisesti alttiita samoille uhkille kuin muutkin avointen tietoverkkojen palvelut, kuten palvelunestohyökkäykset, joiden tarkoitus on estää yksittäisten tai tietyn yhteyden takana olevien solmujen liikennöinti. Erityisenä hyökkäysten luokkana ovat kuitenkin luvussa 3 yleisesti kuvatut *varmistetun identiteetin* puutteeseen perustuvat hyökkäykset, joissa hyökkääjä joko pyrkii muuttamaan verkon reititystä itselleen edulliseen suuntaan tai suoraan muuttamaan verkon solmuihin varastoituja tietoalkioita.

Alttius reitityksen tai tietosisällön manipulaatiolle on käytännössä kiinni toteutuksesta. Mikäli verkkoa käyttävä "rehellinen" toteutus uskoo kaiken mitä esimerkiksi Kademlia-protokollan `FIND_NODE`-kutsujen vastauksissa muut sille kertovat tai antaa kenen tahansa korvata olemassa olevan varastoidun tietoalkion `STORE`-kutsulla, hyökkäys on suoraviivainen. Yksinkertaisimmassa tapauksessa esimerkiksi tietyn tietoalkion saatavuuden häiritseminen vaatii vain, että hyökkääjä lähettää alkion avaimen omistaville solmuille `STORE`-kutsuja useammin kuin avaimen alkuperäinen julkaisija. Tällöin avaimen hakijat saavat todennäköisimmin vastauksena hyökkääjän haluaman arvon, riippuen väärien ja oikeiden arvojen julkaisijoiden määrän suhteista ja julkaisijoiden aktiivisuudesta.

Mikäli olemassaolevien arvojen suoraviivainen korvaaminen, hyökkääjä voi pyrkiä vaikuttamaan siihen, keneltä arvoja haetaan antamalla viittauksia itseensä reititysprosessin `FIND_NODE`-kutsujen vastauksissa. Mikäli toteutus hyväksyy kaikki saamansa vastaukset, esimerkiksi luvussa 3.3 kuvatun pimennyshyökkäyksen toteuttaminen on suoraviivaista.

Tilanne muuttuu mielenkiintoisemmaksi, mikäli toteutus pyrkii varautumaan näihin hyökkäyksiin esimerkiksi toteuttamalla jonkin luvussa 3.4 kuvatun mekanismin. Nämä puolustusmekanismit asettavat *alarajan* hyökkäyksen kompleksisuudelle. Hyök-

kääjän tarvitsee siis käyttää enemmän resursseja (joko suunnittelutyötä tai raakaa laskentaa tai tiedonsiirtoa) päästäkseen samoihin tuloksiin kuin suoraviivaisia toteutuksia huijatessa.

Puolustusmekanismeilla itsellään on tietty *yläraja*, mihin asti ne pystyvät toimimaan. Mikäli mekanismi perustuu johonkin ulkoiseen varmenteeseen, raja liittyy ulkoisen varmentajan toimintaan. Esimerkki tällaisesta rajasta on, miten paljon hyökkääjältä vaatii työtä saada sertifikaattiauktoriteetti myöntämään hänelle sertifikaatti, joka mahdollistaa hänen tavoittelemansa toimenpiteet.

Sisäisissä varmennusmekanismeissa mekanismi taistelee luvussa 3.2 kuvattua Sybil-hyökkäystä eli identiteettien monistamista vastaan. Raja on tällöin mekanismin kyky erottaa "jyvät akanoista" tai suurimittaisen hyökkäyksen tapauksessa kyky "löytää heinä neulasuovasta", eli erottaa hyökkääjäsolmut muista viittaamatta mihinkään ulkoiseen auktoriteettiin.

Joissakin sovelluksissa myös käyttäjien identiteetin tai hänen hakemansa tietosisällön suojaaminen saattaa olla olennaista. Kuten luvussa 5 kävi ilmi, mittaus- ja tarkkailumenetelmät ovat läheistä sukua verkon toiminnan häirintämenetelmille. Toteutetut puolustusmekanismit saattavat vaikuttaa myös tarkkailumenetelmien sovellettavuuteen.

Kuten yllä kävi ilmi, käyttäjän kannalta epätoivottujen vastausten tuottaminen on mahdollista toteuttaa sekä korvaamalla toivotut arvot käyttäen verkon normaalia arvojen talletusmekanismia tai ottamalla haltuun avainten hakuprosessi. Periaatteessa kummassakin on kyse samasta asiasta: DHT:n osanottaja toimii tavalla, joka on sääntöjen mukaan väärä. Arvojen sisällön suhteen nämä säännöt käsittelevät verkon ulkoisia ominaisuuksia (kuten mitä sen tietosisällön pitäisi olla), reititysprosessin suhteen sen sisäisiä (miten reitityskyselyihin tulee vastata). Kummassakin tapauksessa puolustusmekanismien pitäisi tunnistaa, mitkä toimivat ovat "hyvien puolella". Mekanismien toiminta-alue kuitenkin eroaa, koska arvojen oikeellisuuden varmistamisessa kyse on oikeellisuudesta tietoa käyttävän sovelluksen kannalta, kun taas hakuprosessin toiminnan oikeellisuus on olennaista verkon rakenteen oikeellisuuden kannalta (joka saattaa vaikuttaa moneen sovellukseen, mikäli useat sovellukset käyttävät samaa verkkoa eri tarkoituksiin).

Samoja periaatteellisia mekanismeja voisi ainakin osittain soveltaa tiedon oikeellisuuden tarkastamiseen esimerkiksi käyttämällä sertifikaattiauktoriteetin myöntämää sertifikaattia luotettavien lähteiden tunnistamiseen. Koska verkon rakenteen suojaaminen on kuitenkin laaja-alaisempi tehtävä, tämä luku ei käsittele tietoal-

kioiden sisällön oikeellisuuteen liittyviä kysymyksiä tämän enempää.

Seuraava aliluku käsittelee häirinnän tarkempaa toteutusta Kademlia-pohjaisessa verkossa sekä häirinnän estämistä. Luku 6.3 tarkastelee vastaavasti käytön ja käyttäjien seurannan käytännön toteutusta ja luku 6.4 vastatoimia häirinnälle ja tarkkailulle.

6.2 Verkon toiminnan häirintä

6.2.1 Yksinkertainen Sybil-hyökkäys ja reititystaulujen myrkytys

Kad-verkon asiakasohjelmien Kademlia-toteutusta on mahdollista hyökätä suoraviivaisella Sybil-hyökkäyksellä[SENB07]. Tämä ei kuitenkaan ole erityisen tehokas tapa[WTCT+08], ja hakukyselyjen vastauksia ja reititystaulujen päivityskomentoja manipuloivilla hyökkääjäsolmuilla on mahdollista toteuttaa sama hyökkäys nopeammin ja kontrolloitavimmin. Wang et al. [WTCT+08] kokosivat tällaisen käytännön pimennyshyökkäyksen Kad-verkossa vuonna 2008 käytössä ollutta toteutusta vastaan. Tämä hyökkäys hyödyntää Sybil-periaatetta kaapatakseen liikennettä hyökkäyssolmuille, mutta pelkän solmujen verkkoon liittämisen lisäksi hyökkääjä myös myrkyttää aktiivisesti olemassaolevien solmujen reititystauluja.

Valmisteluvaiheessa tutkijoiden n solmusta kukin $0 \leq i \leq n-1$ valitsi itselleen tunnisteen $M_i = \frac{2^{128} \times i}{n}$, eli käytännössä jakoi 128-bitin tunnisteavaruuden n palaan. Kukin solmu liittyi verkkoon ja pyrki löytämään sillä hetkellä verkkoon liittyneet solmut Q tunnisteavaruuden välillä $M_i \leq A < M_{i+1}$. Solmut lähettivät aiemmin löydetyille käyttäjien solmuille `KADEMLIA_REQ`-avaimenhakupyyntöjä, joihin kohdesolmut vastasivat kertomalla reititystauluistaan lisää solmuja kysytyn avaimen lähiavaruudesta.

Koska tutkimuksen aikaan käytössä olleet asiakasohjelmat eivät tarkastaneet tunnisteita millään tavalla, tutkijat pystyivät lisäksi kaappaamaan solmun A reititystaulusta haluamansa osan. Koska solmutunnisteet voivat säilyä IP-osoitteen vaihdosten yli, hyökkääjäsolmu M pystyi lähettämään `HELLO_REQ`-viestin olemassa olevan solmun B tunnisteella ja omalla IP-osoitteella. Solmu A korvasi tällöin tunnisteen B osoitteen hyökkääjään osoitteella, jolloin myös kyselyt solmulle B lähtivät hyökkääjäsolmun M IP-osoitteeseen.

Valmisteluvaiheen jälkeen tutkijoiden hyökkääjäsolmut näkivät tietyn osan kyselyliikenteestä (reititystaulujen kaapatusta osasta riippuen) ja pystyivät estämään tai

hidastamaan haluamiensa avainten hakuja. Kumpikaan ei kuitenkaan ole täydellisesti mahdollista tällä menetelmällä, koska hyökkääjäsolmut saavat haltuunsa vain osan liikenteestä.

Tämäntyyppisessä hyökkäyksessä hyökkääjä ei voi pudottaa kyselyitä kokonaan, koska käyttäjäsolmut tulkitsevat hiljaisuuden merkiksi siitä, että vastaaja on poistunut verkosta ja poistavat tämän reititystauluistaan ja kokeilevat seuraavaa kandidaattia. Vanhemmat käyttäjän ohjelmistoversiot lopettivat tulosten hakemisen kun ne olivat saaneet 300 vastausta kyselyynsä, joten niitä oli mahdollista estää löytämästä oikeita osumia antamalla niille heti 300 vastausta. Uudemmat versiot kuitenkin vaativat, että vastausten sisältö sopii alkuperäiseen hakuun. Koska Kad-verkon kyselyt tehdään hajautusalgoritmin läpi ajetuilla hakusanoilla, hyökkääjän tulisi voida kääntää hajautustulokset takaisin alkuperäisiksi hakusanoiksi. Tämä on mahdollista yleisten hakusanojen sanakirjalla, mutta hankaloittaa hyökkäystä.

Toimivin tutkijoiden löytämä menetelmä oli vastata `KADEMLIA_REQ`-hakupyyntöihin IP-osoitteilla, joissa ei oletettavasti ollut ketään vastaamassa. Tämä aiheutti asiakasohjelmassa haun aikakatkaisun, ja oli tehokas menetelmä kaikkia testattuja asiakasohjelmia vastaan. Tällä menetelmällä ei kuitenkaan ollut mahdollista väärentää haun tuloksia, vaan ainoastaan estää tulosten löytyminen.

Hyökkäysten toteutusta haittasi asiakasohjelmien käyttämä *rinnakkaiskyselymenetelmä* eli asiakkaat lähettivät pyyntöjä monelle solmulle rinnakkain. Koska hyökkäys otti haltuun vain osan reititystauluista, käyttäjien asiakasohjelmat löysivät myös normaaleja solmuja ja saivat hakuja toteutettuja. Vaikka osuutta on mahdollista nostaa, tämä nostaisi entisestään jo valmiiksi suurehkoa kaistavaatimusta (noin 100Mbps 50 hyökkääjän kaapatessa 40 prosenttia 11000-16000 uhrin reititystauluista) [WTCT$^+$08].

Tämä melko suoraviivainen hyökkäys Kad-verkon reititysmekanismia vastaan käytti siis seuraavia haavoittuvuuksia järjestelmässä:

1. *Solmut saavat valita tunnisteensa vapaasti.* Tästä seurasi, että hyökkääjä saattoi asettaa hyökkääjäsolmut vapaasti parhaisiin paikkoihin tunnisteavaruudessa.

2. *Käyttäjäsolmut eivät kyseenalaistaneet mitään saamiaan tietoja.* Hyökkääjä pystyi ohjaamaan haluamiensa solmutunnisteiden liikenteen itselleen tai eimihinkään.

Vaikka hyökkääjä sai siis tehdä verkossa käytännössä mitä tahansa, rinnakkaiskyselymenetelmä heikensi kuitenkin toiminnan tehokkuutta. Käytettyjen verkkoresurssien valossa Kad-verkon manipulaatio tällä menetelmällä vaikuttaa kalliilta.

Vuoden 2009 jälkeen Kad-verkon asiakasohjelmien tekijät lisäsivät asteittain ohjelmiin useita suojausmekanismeja jotka pyrkivät rajoittamaan edellä kuvattuja hyökkäyksiä[CCF09]:

1. *Tulvitusrajoitus.* Asiakasohjelmat pitävät kirjaa pakettien lähettäjien IP-osoitteista, ja estävät liikenteen mikäli sitä on liikaa. Tämä rajoittaa nopeuttaa, jolla hyökkääjä voi manipuloida reititystauluja `HELLO_REQ`-viesteillä.

2. *IP-osoiterajoitukset.* Yksi asiakasohjelma hyväksyy yhdeltä IP-osoitteelta vain yhden solmutunnisteen ja yhdestä /24 aliverkosta korkeintaan 10 tunnistetta tai myöhemmissä versioissa 2 tunnistetta. Tämä rajoittaa useiden hyökkääjäsolmujen sijoittamista yhden kohdeavaimen ympärille, olettaen että hyökkääjällä ei ole suurta määrää IP-osoitteita käytettävissään.

3. *Identiteettien varmistus.* Asiakasohjelmat tarkastavat, että `HELLO_REQ`-viestin lähettäjä on olemassa vaatimalla, että alkuperäinen lähettäjä vastaa kolmannella viestillä asiakkaan `HELLO_RES`-viestiin. Lisäksi päivityksiin sisältyy kryptografinen allekirjoitus. Asiakasohjelma ei päivitä olemassa olevan tunnisteen osoitetta, mikäli kyseistä tunnistetta koskeva `HELLO_REQ` ei sisällä allekirjoitusta samalla avaimella kuin alkuperäinen reititystaulualkio.

Nämä suojaukset ovat yleisluontoisia, eivätkä varsinaisesti spesifisiä Kademliaan. Ne ovat varsin yksinkertaisia verrattuna luvun 3.4 esittelemiin monimutkaisempiin periaatteellisiin mekanismeihin. Cholez et al. [CCF09] tarkastelivat näiden mekanismien käytännön toteutuksia ja vaikutusta yleisesti, ja totesivat niiden kuitenkin vähintään vaikeuttavan ja hidastavan Sybil-hyökkäyksiä huomattavasti.

Rajoitus 1 yksinään hidastaisi Wang et al. määrittelemää hyökkäystä, koska se hidastaa reititystaulujen manipulaatiota. Rajoitus 2 vaikuttaa enemmänkin yksittäisten tunnisteiden kaappaukseen, mutta sillä saattaa olla jonkin asteinen vaikutus, mikäli hyökkääjäsolmujen määrä on suuri (yhden asiakassolmun näköpiiriin tulee monta hyökkääjää samasta osoitteesta). Rajoitus 3 estäisi hyökkäyksen aktiivisen kaappaustoiminnan, olettaen että asiakasohjelmat hylkäävät allekirjoittamattomat päivitykset täysin (jotkin versiot hyväksyivät ne silti väliaikaisesti reititystauluun[CCF09].

Muutosten yleisluontoisuudesta ja yksinkertaisuudesta huolimatta Cholez et al. tekemien mittausten perusteella voi siis olettaa näiden toteutus- ja protokollamuutosten olevan tehokkaita Wang et al. määrittelemää hyökkäystä vastaan.

Hyökkääjät, joilla on käytettävissään jonkin verran IP-osoitteita voivat kuitenkin tehdä kohdennetumpia pimennyshyökkäyksiä valittuja hakuavaimia vastaan. Tämä vaatii sekä vähemmän tietoliikenne- että osoiteresursseja kuin koko verkkoa koskeva hyökkäys, joka on potentiaalisesti liiankin laaja moneen käyttöön [CCF09, CCF10]. Tätä käsitellään seuraavassa luvussa.

6.2.2 Valittujen kohdeavainten tarkkailu ja pimennys

Edellisessä luvussa kuvattu hyökkäys pyrkii ottamaan haltuun merkittävän määrän koko DHT:n liikennettä manipuloidakseen tai tarkkaillakseen sitä. Tämä on kuitenkin verkkoresurssien kannalta suhteellisen vaativa operaatio, ja lisäksi suuri määrä solmuja samoista IP-osoitteista saattaa tehdä sen helposti havaittavaksi ulkopuoliselle tarkkailijalle. Yksittäisiin avaimiin kohdistuvien hakujen kaappaaminen saattaa olla tavoitteesta riippuen tehokkaampi ja helpommin naamioitava hyökkäystapa. Esimerkkejä tällaisen hyökkäyksen sovelluskohteista voisi olla tietyn elokuvan levityksen estäminen tai tarkkailu tai jonkin jaettavan ohjelmiston puhtaan lähteen korvaaminen viruksen sisältävällä kappaleella.

Cholez et al. [CCF10] ja Kohnen et al. [KLR09] esittävät kumpikin samankaltaista menetelmää yksittäisiin avaimiin kohdistuvien hakujen kaappaamiseksi. Edellisen menetelmä toteuttaa hyökkäyksen perusperiaatteeltaan yksinkertaisemman mutta hitaamman hyökkäyksen tiettyjä kohdetunnistetta vastaan. Periaatteellisen tarkastelun yksinkertaistamiseksi tämä luku käsittelee ensin hyökkäystä edellisen tutkimuksen näkökulmasta.

Koska Kad-verkon tunnisteavaruus on hyvin laaja (2^{128} mahdollista tunnistetta), on epätodennäköistä, että satunnaisesti minkään käyttäjän asiakasohjelman satunnaisesti valitsema tunniste on lähellä minkään sisältöavaimen tunnistetta. Kohnenin ja Cholezin menetelmien tavoite on asettaa niin sanotut “hunajapurkkisolmut” lähelle valittua avainta valitsemalla näiden solmujen tunnisteet kyseisen avaimen läheltä. Tutkimusten aikaan Kad-verkossa oli noin viisi miljoonaa käyttäjää, ja Cholezin käyttämässä menetelmässä hyökkääjäsolmujen tunnisteiksi valittiin 96 ensimmäistä bittiä kohdeavaimesta ja loput 32 bittiä satunnaisesti. Viiden miljoonaan käyttäjän verkossa todennäköisyys, että jokin käyttäjäsolmu on valinnut tunnisteen samalta

etäisyydeltä on häviävän pieni (noin $\frac{2^{32}}{2^{128}} \times 5 \times 10^6 \approx 6.31 \times 10^{-23}$).

Kun asiakasohjelmat etsivät tietystä avaimesta vastaavia solmuja, ne lähettävät kolme `KADEMLIA_REQ`-viestiä rinnakkain, valitsevat joka vastauksesta kolme uutta kohdetta jotka ovat lähempänä haluttua avainta ja toistavat prosessin. Edellisen luvun IP-osoiterajoitusten mukaan versiota riippuen vain 2–10 näistä kohteista saa olla saman /24 aliverkon sisällä. Kun läheisempiä solmuja ei enää löydy, asiakasohjelma lähettää varsinaiset tietoalkion haku- tai talletuspyynnöt siihen mennessä löydetyille lähimmille solmuille. Asiakasohjelmat pyrkivät tallentamaan uudet tietoalkiot ainakin kymmenelle kohdesolmulle.

Koska hyökkääjäsolmut ovat todennäköisesti kohdeavaimen lähimmät solmut, ne todennäköisesti saavat käsiteltäväkseen myös kaikki siihen liittyvät haku- ja tallennuspyynnöt. Hyökkääjä voi tällöin joko vastata normaalisti ja tallentaa kaikki saamansa pyynnöt tai toteuttaa pimennyshyökkäyksen eli kertoa hakijoille, että haettua avainta ei ole olemassa. Cholez et al. mittausten mukaan 20 “hunajapurkkia” riitti kaappaamaan pääosan tiettyyn hakusanaan kohdistuvan liikenteen sillä perusteella, että hyökkääjäsolmuverkko näki tyypillisesti kymmenen tai useampia julkaisupyyntöjä kaapatulle avaimelle[CCF10]. Olettaen, että asiakasohjelmat hyväksyvät yhdestä /24 aliverkosta vain kaksi solmutunnistetta ja yhdeltä IP-osoitteelta yhden, hyökkäys vaatisi siis 20 IP-osoitetta kymmenestä aliverkosta. Tämä määrä on todennäköisesti pienimuotoisemmankin hyökkääjän hankittavissa sekä luvallisin että luvattomin menetelmin (luvattomista menetelmistä esimerkkinä botnet, jonka solmuilla hyökkääjä voi ajaa haluamiaan ohjelmia). Lisäksi samoja IP-osoitteita voi käyttää hyökkäyksissä monia eri kohdetunnisteita vastaan, koska kohdetunnisteet ovat todennäköisesti kaukana toisistaan tunnisteavaruudessa[CCF10]. Toisin sanottuna asiakasohjelmien reititystauluihin ei todennäköisesti päädy kuin yhden kohteen lähialueen reititystietoja. Todennäköisesti yhtäläisyyksiä muiden kohteiden kanssa on vain reititystaulun hyvin korkeilla tasoilla.

Kohnen et al. hyökkäyksessä hyökkääjä valitsee samalla tavalla hyökkäyskohteeksi tietyn kohdetunnisteen, mutta lisäksi hyökkäyksessä käydään läpi läheinen tunnisteavaruus. Läpikäyntiohjelma myrkyttää löytyneiden asiakasohjelma-solmujen reititystauluja `HELLO_REQ`-pyynnöillä niin, että kohdeavaimen lähellä odottavat hunajapurkkisolmut saavat todennäköisemmin kohdeavaimeen liittyvää liikennettä. Myrkyttämällä reititystaulut sellaisilla solmuilla, joiden tunnisteen etuliitteestä 13–19 bittiä oli yhteisiä kohdeavaimen kanssa, hyökkäys onnistui kaappaamaan kohdeavaimen haut hunajapurkkisolmuille välittömästi. Hyökkäyksen tietoliikennekustannus

oli noin 4.8kbit/s, eli hyvin pieni[KLR09]. Tätä hyökkäysmallia ei haittaa edellisessä luvussa mainittu reititystietojen ylikirjoitussuojaus, koska reititystauluihin lisätään uusia kohdesolmuja olemassaolevien solmutunnisteiden kaappaamisen sijaan.

Toisin kuin Cholez et al. tutkimuksessa, Kohnen et al. eivät testanneet hyökkäystä olemassa olevaan DHT:n sisältöön, vaan ainoastaan itse generoimiinsa avainsanoihin ja tiedostoihin. Tutkimuksesta ei käy ilmi, miten kilpailu suuren määrän olemassa olevia julkaisijoita kanssa vaikuttaisi lopputulokseen tai resurssivaatimuksiin. Koska perusperiaate "hunajapurkkien" asettelusta tunnisteavaruudessa on sama, on kuitenkin oletettavaa, että lopputulos olisi sama. Mahdollisen tehokkuuden heikkenemisen pitäisi näkyä ainoastaan kaappausprosessin kestossa.

Tämä hyökkäys Kad-verkon tunnisteiden valintamekanismia vastaan käytti siis seuraavia haavoittuvuuksia järjestelmässä:

1. *Solmut saavat valita tunnisteensa vapaasti.* Hyökkääjät saivat sijoittaa omat solmunsa lähelle haluttua hakusanaa.

2. *Käyttäjäsolmut eivät kyseenalaistaneet saamiaan tietoja juurikaan.* Hyökkääjä pystyi ohjaamaan haluamiensa solmutunnisteiden liikenteen itselleen vain pienimuotoisella puolustustoimenpiteiden kiertämisellä.

Käytetyt haavoittuvuudet ovat siis samat kuin edellisen luvun hyökkäyksessä, mutta niiden käyttötarkoitus on erilainen. Edellisen luvun puolustusmekanismit olivat hampaattomia näiden muuttuneiden käyttötarkoitusten suhteen. Kumpikaan tämän luvun hyökkäysten kehittäjistä ei myöskään esitä puolustusmekanismeja tämän tyyppisille hyökkäyksille. Eräitä mahdollisia mekanismeja käsitellään luvussa 6.4.

6.2.3 BitTorrent-DHT:jen heikkoudet

BitTorrent-ohjelmistoihin liittyvät kaksi DHT-toteutusta perustuvat kumpikin Kademliaan (toteutukset on esitelty luvussa 4.1). Näin ollen on oletettavaa, että niihin on mahdollista kohdistaa samat hyökkäykset kuin tämän luvun kuvaamat Kad-verkkoon kohdistuvat hyökkäykset.

Timpanaro et al. [TCCF11] toteavat, että tämä todellakin on mahdollista päälinja-DHT-toteutuksien tapauksessa. Vuonna 2011 käytössä olleet päälinja-DHT-toteutukset eivät toteuttaneet edes Kad-asiakasohjelmien toteuttamia perustason suo-

jausmekanismeja, jotka on esitelty luvussa 6.2.1. Päälinja-asiakasohjelmien Kademlia-toteutus on siis täysin haavoittuva tämän luvun aiemmin kuvaamille hyökkäyksille.

Vuze-DHT:n suhteen ei ole olemassa vastaavaa kattavaa tutkimusta, mutta Wolchok et al. tutkimus hyökkäyksestä Vuze-DHT:a käyttävää Vanish-järjestelmää vastaan sivuaa aihetta. Tutkimuksen yhteydessä käy ilmi, että Vuze-DHT:een liittyvän solmun tunnisteen tulee olla yhtä kuin hajautusalgoritmin tulos liittyvän solmun IP-osoitteesta ja DHT-liikenteeseen käyttämästä porttinumerosta [WHH+09]. Vuzen protokolladokumentaatio ilmoittaa saman suojausmenetelmän kuuluvan toteutukseen [Vuz12].

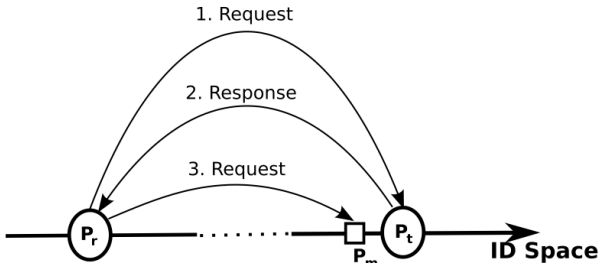
Tämä rajoitus solmutunnisteiden valinnassa on johdannainen luvussa 3.4.1 kuvatusta tunnisteiden sitomisesta. Tässä tapauksessa siis IP-osoite toimii luotetun auktoriteetin myöntämänä tunnisteena, johon DHT-liikenteessä käytettävä solmutunniste pohjautuu. Asiakasohjelma voi kuitenkin vaikuttaa tunnisteeseen liikennöintiportin valinnalla. Hyökkääjä voi siis edelleen perustaa noin 65000 Sybil-solmua yhtä IP-osoitetta kohden, mutta ei voi valita näiden tarkkaa sijoittumista tunnisteavaruudessa.

Oikein toteutettuna rajoitus vaikeuttaa tässä luvussa kuvattujen nopeiden hyökkäysten soveltamista Vuze-DHT:a vastaan, koska ne nojaavat juuri solmutunnisteen vapaaseen valintaan ja mahdollisuuteen ohjata minkä tahansa tunnisteen liikenne mihin tahansa osoitteeseen. Hyökkääjä voi silti pyrkiä vaikuttamaan verkon toimintaan suurella määrällä Sybil-solmuja, mutta tämä vaatii enemmän verkkoresursseja ja on todennäköisesti hitaampaa kuin hyökkäykset suojaamattomiin verkkoihin. Mekanismin varjopuolena on myöskin, että asiakassolmun täytyy tietää IP-osoite, jolla se näkyy muille käyttäjille, ja että DHT-liittymisprosessi täytyy aloittaa alusta aina IP-osoitteen muuttuessa.

Useiden porttinumeroiden ja IP-osoitteiden valinnalla hyökkääjä voi myös pyrkiä saamaan itselleen solmutunnisteita, jotka olisivat edes kohtuullisen lähellä haluttuja tunnisteita. Memon et al. testasivat tätä ottamalla käyttöön 115 IP-osoitetta ja kustakin portit 1025–65535, jolloin he saivat käyttöönsä noin seitsemän miljoonaa solmutunnistetta. Valitsemalla näistä haluttuja kohdeavaimia lähimmät tunnisteet, he onnistuivat tarkkailemaan kaikkea valitsemiensa kohteiden liikennettä[MRGS12]. Tämä tutkimus ei ottanut kantaa siihen, miten tehokkaasti valikoimasta poimituilla tunnisteilla voisi toteuttaa esimerkiksi pimennyshyökkäyksiä, jotka vaativat useamman kuin yhden solmun kohteen läheisyyteen.

6.3 Käyttäjien seuranta

Luku 6.2 esitteli kaksi menetelmää manipuloida Kademlia-pohjaisia DHT:ja haku- ja julkaisupyyntöjen kaappaamiseksi. Näissä menetelmissä hyökkäyksen tavoite on tarkkailun lisäksi myös vaikuttaa hakujen tuloksiin, joten hyökkääjä joutuu liittymään verkkoon usealla solmulla ja pyrkii sijoittamaan itsensä mahdollisimman monen muun solmun reititystauluun. Mutta mikäli tavoite on ainoastaan tarkkailla liikennettä, riittää, että edes yksi hyökkääjäsolmu saa kohdeavaimen haku- tai julkaisupyynnön. Tässä tapauksessa on perusteltua pitää tarkkailijat mahdollisimman näkymättöminä, jotta verkkoresurssien tarve ja havaituksi tulemisen todennäköisyys olisivat pienempiä.



Kuva 11: Liikenteen kaappaus Memon et al. menetelmässä. Kuvan lähde [MRGS12].

Memon et al. [MRGS12] esittävät menetelmää, jossa potentiaalisesti suuri määrä "minimaalisesti näkyviä tarkkailijoita" liitetään DHT:een tarkkailemaan kohdeavainten hakua ja julkaisuja. Toisin kuin aiemmin kuvatuissa menetelmissä, tarkkailijasolmuja P_m ei sijoiteta mahdollisimman lähellä itse avainta, vaan mahdollisimman lähelle avaimesta sillä hetkellä vastaavaa normaalisolmua P_t. Koska käytännön Kademlia-toteutukset lähettävät haku- ja julkaisupyyntöjä useille avaimen läheisille solmuille, tunnisteavaruudessa P_t vierelle sijoitettu tarkkailija P_m saa suurella todennäköisyydellä samat pyynnöt kuin P_t. Näkyvyyden minimoimiseksi P_m vaihtaa normaaleja `HELLO`-viestejä vain naapurinsa P_t kanssa. Se kirjaa ja pudottaa kaikilta muilta solmuilta saadut viestit. Näin ollen P_m on kaikkien muiden verkon osanottajien näkökulmasta verkosta poistunut solmu, mutta P_t pitää sitä elossa olevana ja jakaa sen osoitteen reititystaulunsa mukana. Tarkkailijan P_m sijoittelu ja pyyntöjen kulku näkyy kuvassa 11.

Menetelmä soveltuu myös kokonaisten tunnisteavaruuden siivujen tarkkailuun ennalta määriteltyjen kohteiden sijaan. Memon et al. totesivat, että yhdellä palvelimella on mahdollista tarkkailla kuuden bitin etuliitettä vastaavaa osuutta tunnisteavaruudesta, ennen kuin olemassaolevien solmujen löytämiseen tarvittava läpikäynti käy liian hitaaksi[MRGS12]. Samoin menetelmä käytännössä toimi myös Vuze-DHT:n kanssa, koska tarkkailijat sai sijoitettua riittävän lähelle kohteita generoimalla suuren määrän valideja solmutunnisteita luvussa 6.2.3 kuvatulla tavalla.

Olemassa olevia solmujen osoitteita voi etsiä esimerkiksi jollakin luvussa 5.2 kuvatulla menetelmällä.

Kad-verkon osalta luvussa 6.2.1 kuvatut suojausmekanismit eivät myöskään juuri vaikuta menetelmän toimintaa. Tulvitusrajoitus hidastaa olemassaolevien solmujen hakua tämän tutkimuksen käyttämässä menetelmässä, mutta yksi tunniste IP-osoite kohden -rajoituksella ei ole merkitystä, koska yhtä kohdetta varten on yksi tarkkailija. Samoin identiteettitarkistus on merkityksetön, koska tarkkailija käyttää `HELLO`-viestejä kuten normaali solmu[MRGS12].

Tämä puhtaasti tarkkailuun tähtäävä menetelmä on siis sovellettavissa useaan DHT-toteutukseen, eikä nykyisissä toteutuksissa ole tiettävästi mekanismeja sen estämiseksi. Menetelmän toimivuus perustuu Kademlian normaaleihin toimintamekanismeihin (kuolleiden solmujen pudotus reititystauluista, pyyntöjen lähetys monelle kohteelle haun tehostamiseksi), joten suoraviivainen puolustusmekanismi vaikuttaa saavuttamattomalta. Koska tarkkailijasolmujen näkyvyys rajoittuu eri tavalla kuin normaalisolmujen, lienee mahdollista, että kattavan verkon läpikäynnin tuloksista olisi tilastollisesti mahdollisesti erottaa näitä tarkkailijasolmuja. Jotkin seuraavan luvun käsittelemät edistyneet mekanismit saattavat myös vaikeuttaa tätä tarkkailumenetelmää esimerkiksi rajoittamalla tunnisteiden valintaa.

6.4 Edistyneet puolustusmekanismit

Tämä luku esittelee muutamia puolustusmekanismeja, joita niiden kehittäjät ovat soveltaneet erityisesti Kademliaan. Tässä esiteltävät mekanismit edustajat kolmea kategoriaa: kryptografiset ja reititystä monipuolistavat perusparannukset tunnistevalintaan ja reititykseen; solmuille annettaviin luottamusarvoihin perustuva solmujen luokittelu; ja tilastollinen solmujen luotettavuuden arviointi.

6.4.1 Verifioidut tunnisteet ja monipolkureititys

Baumgart ja Mies [BM07] esittävät sekalaista joukkoa kryptografiaan ja reitityksen monipuolistamiseen perustuvia parannuksia Kademliaan. Nämä menetelmät eivät edusta perinpohjaista turvallisuus- ja luottamusmallin uusimista, vaan ovat pikemminkin toteutus perusperiaatteista, joita muun muassa Castro et al. [CDG+02] esittivät jo viisi vuotta aiemmin. Keskeiset esitetyt menetelmät ovat:

1. *Viestien allekirjoitus.* Allekirjoitukset todentavat, että tietyn tunnisteen hal-

tija pysyy samana.

2. *Rajoitettu solmutunnisteiden luonti.* Solmutunnisteilla on joko ulkoinen varmentaja tai niiden luonti vaatii jonkin verifioitavan laskentaongelman ratkaisemisen.

3. *Rajoitettu lisäys reititystauluun.* Solmu lisää uusia naapureita reititystauluun eri luotettavuusasteilla sen mukaan, miten niistä kertovat viestit on allekirjoitettu. Lisäksi tunnisteet, jotka ovat liian läheisiä jonkin olemassa olevan tunnisteen kanssa tulkitaan epäluotettaviksi.

4. *Monipolkuiset avainhaut.* Sen sijaan, että kyselijäsolmu hakisi kohdetunnisteen käyttäen vain reititystaulun parhaita vaihtoehtoja, se lähettää `FIND_NODE`-kyselyitä kullakin askeleella monelle eri vaihtoehdolle ja seuraa rinnakkain eriäviä polkuja.

Simulaatiotestauksessa kahdeksan vaihtoehtoisen polun käyttäminen mahdollisti 90 prosentin hauista onnistumisen, kun 40 prosenttia verkon solmuista käyttäytyi virheellisesti ja solmut toteuttivat edellä esitetyt allekirjoitus- ja tunnisteidenluontimenetelmät. Yli kahdeksan vaihtoehdon käyttäminen ei parantanut tulosta enää merkittävästi. Tutkijat olettivat, että menetelmien 2 ja 3 seurauksena hyökkääjäsolmut ovat jakautuneet tasaisesti tunnisteavaruuteen ja rehellisten solmujen palauttamiin reititystauluihin. Samassa tilanteessa ilman monipolkuista hakua noin 40 prosenttia hauista onnistui ja kahdella vaihtoehtoisella reitillä noin 65 prosenttia onnistui.[BM07].

Nämä menetelmät edustavat suurin piirtein seuraavaa vaikeustasoa Kad-toteutusten käyttämien tunniste- ja IP-rajoitusten jälkeen. Toteutus ei todennäköisesti sisällä suuria hankaluuksia, mutta toisaalta menetelmät eivät välttämättä myöskään estä asialleen omistautuneita hyökkääjiä. Kevyillä identiteettitarkastuksilla varustettu keskushallinnoitu tunnusten luonti tai laskentaohjelmapohjaiset tunnisteet vain rajoittavat Sybil-solmujen luontinopeutta[Dou02] ja allekirjoitukset ilman sertifiointia eivät sinänsä rajoita identiteettien luontia, vaan ainoastaan olemassaolevien kaappausta.

Se, miten hyväksyttävää monipolkuisten hakujen vaikutus hakujen nopeuteen ja verkon liikennemäärään lopulta on, on myös aihe, joka vaatisi käytännön kokeita. Menetelmien 1–3 toteutus sen sijaan ei välttämättä toisi ainakaan suuria haittoja ja olisi luultavasti testaamisen arvoista. Menetelmä 2 rajoittanee jonkin verran luvun

6.3 kuvaamaa tarkkailumenetelmää, mutta kuten Vuze-DHT-esimerkin tapauksessa, omistautunut hyökkääjä voi päästä riittävän lähelle vain generoimalla suuren joukon tunnisteita joista valita sopivat.

6.4.2 Luottamusarvopohjainen reititys

M. Kohnen [Koh12a] on esittänyt menetelmän, jossa verkon solmuille lasketaan *luottamusarvot*. Jokaisella solmulla on itse avainpari sekä itse-allekirjoitettu sertifikaatti, jonka perusteella se tunnistetaan sen osoitteesta tai DHT:n sisäisestä tunnisteesta riippumatta. Se myös generoi DHT-tunnisteensa varmistettavalla tavalla tämän sertifikaatin hajautusarvosta. Jotta yksi solmu ei voi pitää hallussaan yhtä osaa avainavaruudesta pitkään, sertifikaattien voimassaoloajan tulee olla 24 tuntia.

Kun solmu ("asiakas") pyytää toista solmua ("kohde") tekemään jotain, asiakas saa oikeuden antaa erilliselle *luottamuksenhallintasolmulle* arvion kohteen toiminnasta. Tämä arvio voi olla joko positiivinen (pyyntö tuotti odotetun tuloksen) tai negatiivinen. Tiedon haku- ja talletuspyyntöjen saamat arviot muodostavat yhdessä *tietosisältöarvion* ja reititykseen liittyvien pyyntöjen saamat arviot *reititysarvion*. Tietosisällön arviointi riippuu käytössä olevasta sovelluksesta, mutta reititysarvio on sovelluksesta riippumaton; se kertoo vain, onko solmu vastannut reitityspyyntöihin tavalla, jolla asiakas on saavuttanut haluamansa kohteen. Luottamuksenhallintasolmut laskevat solmujen lopulliset luottamusarvot saamiensa positiivisten ja negatiivisen arvioiden erotuksesta. Reititysvaiheessa asiakas huomioi vain valitsemaansa rajaa paremman arvion saaneet solmut reititysprosessissa.

Simulaatiokokeissa menetelmä pystyi parhaimmillaan erottamaan kaikki hyökkääjäsolmut oikein toimivista, vaikka verkon solmuista 20 prosenttia oli hyökkääjiä. Toisaalta simulaatiossa tehtiin yksinkertaistavia oletuksia, kuten että luotettavaksi todetut solmut ovat aina luotettavia, eivätkä "vaihda puolia"[Koh12a].

Vaikka Kohnen käyttää lähtökohtana Kademlian toimintaa, kuvattu menetelmä on yleisellä tasolla sovellettavissa periaatteessa myös muihin DHT-järjestelmiin[Koh12a]. Jatkotutkimuksessa Kohnen on kehittänyt myös tarkemmat tavat laskea luottamusarvot spesifisesti Kademlia-pohjaisessa järjestelmässä[Koh12b].

Kohnenin menetelmä näyttää paperilla lupaavalta, mutta siltä puuttuu täysin testaus todellisissa ympäristöissä, joten sen käytännön toimivuutta on mahdoton arvioida. Periaatteessa menetelmä sisältää konsepteja sosiaalisiin verkkoihin perustuvista suojausmekanismeista ilman tarvetta muodostaa verkkoja manuaalisesti, ja si-

sältää myös mekanismeja yhteistyötä tekevien hyökkääjien torjumiseksi. Tästä huolimatta simulaatioissa tehty oletus, että “hyvät pysyvät hyvinä” on käytännön kannalta hyvin kyseenalainen. Mikäli menetelmästä olisi jalostettavissa toimiva versio, se saattaisi kuitenkin olla tehokas väärin toimivien solmujen karsimisessa reitityksestä. Myös epätavallisilla tavoilla toimivat tarkkailijasolmut saattaisivat saada riittävästi negatiivista palautetta, jotta ne pudotettaisiin verkosta, joskin tämä luultavasti riippuisi menetelmän käytännön toteutuksesta.

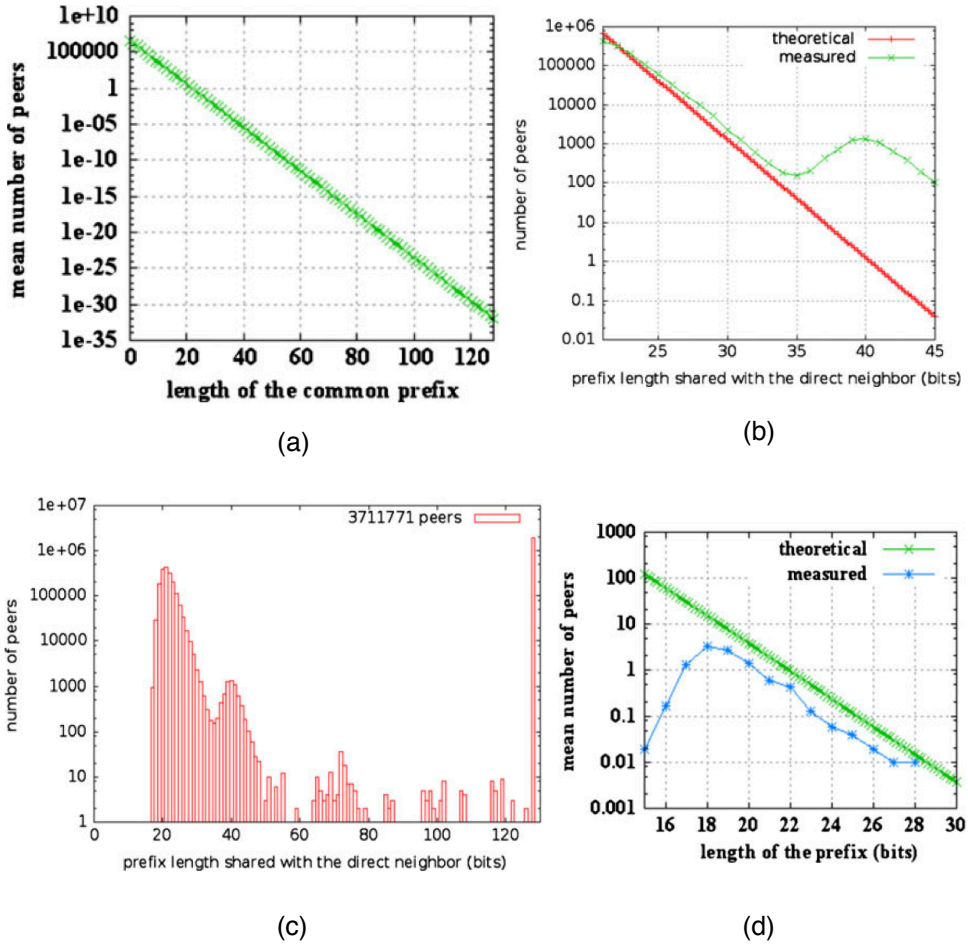
Koska Kohnenin esittämä menetelmä on hyvin uusi, se ei ole ehtinyt myöskään saada vielä akateemista kritiikkiä tai ulkopuolista jatkotutkimusta.

6.4.3 Hyökkäysten tunnistus tunnistejakaumien perusteella

Edellä kuvatut menetelmät vaativat muutoksia olemassa oleviin ohjelmistoihin, jotta menetelmät olisivat mahdollisimman hyödyllisiä. Heterogeenista käyttäjäkuntaa saattaa kuitenkin olla vaikea saada päivittämään ohjelmistojaan aktiivisesti. Cholez et al. [CCFD12] esittävät menetelmää, jossa haun suorittaja pyrkii erottamaan “epäilyttävät” solmut “normaaleista” vertailemalla solmutunnisteiden jakaumaa niiden ideaaliseen jakaumaan.

Tällä hetkellä laajasti käytössä olevat Kademlia-toteutukset valitsevat solmutunnisteet satunnaisesti, eli tunnisteiden pitäisi olla tasaisesti jakautunut tunnisteavaruuteen. Tästä seuraa, että keskenään d-bittisen etuliitteen jakavien naapuriparien määrän pitäisi vähentyä tasaisesti, ja tutkimusajankohdan N solmun Kad-verkon tapauksessa odotusarvo keskimääräiselle yhteiselle etuliitteelle olisi $\log_2(N) = 21.93\text{bit}$. Tämä jakauma näkyy kuvan 12 kohdassa (a). Käytännössä tämä oletus ei kuitenkaan toteudu, vaan 35–45 bittiä jakavat naapuriparit muodostavat huomattavan kuplan jakaumassa ,kuten edellä mainitun kuvan kohdasta (b) käy ilmi. Lisäksi kuten saman kohdan (c) oikeasta laidasta näkyy, verkossa on myös huomattava määrä solmuja, jotka jakavat hyvin pitkiä etuliitteitä. Verkon läpikäynnin perusteella tutkijat arvioivat, että käynnissä olisi ainakin noin kaksi tuhatta toisistaan erottuvaa hyökkäystä.

Tutkijat julkaisivat omia satunnaisesti generoituja tiedostoja ja avainsanoja Kad-verkossa ja tarkastelivat niiden solmujen tunnisteita, jotka vastasivat testisisältöjä koskeviin kyselyihin. Näiden vastaajien jakauman perusteella yli 17-bittisten etuliitteiden osalta saman etuliitteen jakavien parien määrä väheni lähes täysin teoreettisen jakaumaoletuksen mukaisesti. Odotettu ja mitattu jakauma näkyvät kuvan



Kuva 12: (a) Odotettu keskimääräinen määrä solmuja verkossa ($N = 4 \times 10^6$), jolla on y-akselin osoittama määrä yhteisiä bittejä satunnaisesti valitun avaimen kanssa. (b) Teoreettisesti odotettu ja Kad-DHT:ssa havaittu solmujen määrä, jotka jakavat y-akselin osoittaman pituisen etuliitteen naapurinsa kanssa. (c) Naapurisolmujen yhteisen etuliitteen pituuksien jakauma Kad-DHT:ssa. (d) Rehellisiksi oletetuilta solmuilta saaduista reititystauluista mitattu yhteisen etuliitteen muutos verrattuna teoreettisesti odotettuun muutoksen. Kuvien lähde [CCFD12].

12 kohdassa (d). Näin ollen oli oletettavissa, että normaalisolmut on mahdollista erottaa hyökkääjistä tunnistejakaumia tarkastelemalla.

Koska saatavilla oleva näytekoko on suhteellisen pieni, tutkijat päätyivät käyttämään havaitun ja odotetun jakauman eron mittarina Kullbackin-Leiblerin divergenssinä. K-L-jakauma todettiin herkemmäksi kuin perinteiset tilastolliset menetelmät kuten χ^2. Tarkastelun kohteena olivat solmujoukot, joilla oli 18–28-bittiset yhteiset etuliitteet. Etukäteen todettuja hyökkääjäsolmujakaumia vastaan testattuna menetelmä tuotti noin 7,95 prosenttia vääriä negatiivisia tuloksia ja 8,65 prosenttia vääriä positiivisia tuloksia. Kymmenen hyökkääjäsolmun joukkojen tapauksessa vääriä negatiivisia tuloksia oli 1,56 prosenttia.

Käytännön sovelluksena tutkijat esittävät menetelmää, jossa reititysprosessin tuloksena saaduista kandidaattisolmuista karsitaan epäilyttävät ja kandidaattien haku toistetaan mikäli tarpeellista. Lisäksi esikäsittelynä kandidaatit, jotka ovat peräisin samasta /24 aliverkosta tai joilla on pidempi yhteinen etuliite kuin 28 bittiä poistetaan suoraan (näin pitkät yhteiset etuliitteet ovat normaalisti hyvin epätodennäköisiä). Tavoitteena on, että lopputuloksena on lista seuraavan reititysaskelen kohteista, johon ei sisälly lainkaan epäilyttäväksi määriteltyjä solmuja.

Käytännön testaus Kad-verkossa tunnisti onnistuneesti neljä viidestä tutkijoiden lisäämästä Sybil-solmusta, jotka jatkoivat 20-bittisen etuliitteen. Lisäksi järjestelmästä on toteutus, joka sisältyy vähemmän tunnettua Gnutella-verkon DHT-versiota tukevaan `gtk-gnutella`-ohjelmistoon. Tämän toteutuksen menestyksestä ei ole tarkempaa tietoa[CCFD12].

Tässä esitetty menetelmä ei vaadi muutoksia muiden kuin sen käyttäjän itsensä asiakasohjelmiin, ja esitettyjen testien valossa tunnistaa ainakin varautumattomat hyökkääjät tehokkaasti. On epäselvää, miten helposti hyökkääjä pystyisi piiloutumaan tältä menetelmältä, mikäli hän on tietoinen menetelmän käytöstä ja toteutuksiin liittyvistä vakioista. Menetelmän tilastollinen osuus ei ole hyödyllinen luvun 6.3 tarkkailumenetelmän suhteen, koska se perustuu yksittäisiin solmuihin. Käytännön toteutukseen esitetty raja yli 28-bittisten etuliitteiden käyttäjien etukäteissuodattamisesta saattaa kuitenkin vähentää yhteydenottoja tarkkailijoihin.

Periaatteessa menetelmän suhteellinen yksinkertaisuus puoltaisi joka tapauksessa sen laajempaa testausta. Sen varjopuoli on kuitenkin, että se kodifioi odotuksen satunnaisesti jakautuneista avaimista verkossa. Vaikka avainten satunnainen jakauma on tällä hetkellä implisiittinen oletus, tarkoituksellinen avainten sijoittelu voi olla perusteltua myös ei-hyökkäystarkoituksessa. Esimerkiksi suositun avaimen kuor-

manjako sijoittamalla sen lähistölle useita solmuja olisi sovellus, jonka tässä esitetty menetelmä tulkitsisi hyökkäykseksi.

7 Johtopäätökset

Tutkielman luku 2 esitteli vertaisverkkojen ja erityisesti DHT-järjestelmien yleisiä toimintaperiaatteita. Tähän perustuen luku 3 käsitteli luottamuksen ja turvallisuuden käsitteitä DHT-järjestelmissä sekä hyökkäys- ja puolustusmekanismeja yleisellä tasolla. Kuten J.R. Douceur osoitti, ilman identiteettien hallintaa hyökkääjä voi välttämättä tuottaa suuria määriä verkossa toimivia identiteettejä. Tämä identiteettien tehtailumahdollisuus on useimpien verkon rakenteeseen kohdistuvien hyökkäyksien taustalla, mutta myös palvelunestohyökkäykset sekä roskatiedon lisääminen verkkoon ovat mahdollisia hyökkäysmekanismeja. Puolustusmekanismit pyrkivät rajoittamaan joko identiteettien muodostamista tai verkkoon liittymistä massatuotetuilla identiteeteillä tai tunnistamaan virheellisesti toimivat solmut.

Luku 4 esitteli Kademlia-algoritmin toteutuksia ja käyttöä nykypäivän Internetissä. Vaikka Kademlia-pohjaisilla järjestelmillä on useita miljoonia käyttäjiä maailmanlaajuisesti, sen sovellukset rajoittuvat kuitenkin yhä tiedostonjakosovellusten tukitoimintoihin, kuten avainsanojen ja jakajasolmujen hakuun. Verrattuna esimerkiksi nykypäivän keskitettyihin viestintäpalveluihin kuten Twitter tai Facebook, Kademlia-pohjaisten järjestelmien käyttö on rajoittuneiden sovellusten lisäksi käyttäjämäärältään yhä suhteellisen pientä.

Luku 5 käsitteli menetelmiä, joilla olemassa olevien järjestelmien käyttöä voi tarkkailla ja mitata. Suoranaisen sisällön valvonnan lisäksi mittaus on olennaista myös toteutusten ja algoritmien kehittämiseksi.

Viimeisenä luku 6 käsitteli spesifisesti Kademlia-pohjaisiin järjestelmiin kohdistuvia hyökkäys- ja tarkkailumenetelmiä sekä näiden vastatoimia. Laajalti käytössä olevat Kademlia-toteutukset eivät aluksi sisältäneet minkäänlaista suojausta hyökkääviä tai muuten virheellisesti toimivia solmuja vastaan. Sittemmin toteutetut suojaustoimet ovat yksinkertaisia ja vain rajoitetusti tehokkaita. Teorialuvussa ja tämän luvun loppupuolella esitettyjä edistyneempiä menetelmiä ei ole käytännössä toteutettu juuri missään normaalikäyttäjien asiakasohjelmissa, ja arviot niiden tehokkuudesta perustuvat pääasiassa simulaatioihin. Näin ollen on yhä epäselvää, kuinka tehokkaasti Kademlia-pohjaisia järjestelmiä voi todellisissa sovelluksissa suojata omistautuneen toimijan suorittamalta häirinnältä ja suurimittaiselta tarkkailulta.

Tutkielmassa esitellyt puolustusmekanismit perustuvat paljolti Sybil-solmujen tunnistamiseen. Kuten esitin luvussa 6.4.3, on mahdollista, että tulevissa sovelluksissa olisi perusteltua käyttää solmuilla muita kuin satunnaisesti valittuja tunnisteita tai muutoin toimia protokollan ideaaleista poikkeavasti. Tästä syystä mukailisin Viswanath et al. [VPGM10] johtopäätöstä, että jatkotutkimuksessa saattaisi olla perusteltua keskittyä Sybil-solmujen etsimisen ja välttelemisen sijaan niiden aiheuttamien virhetilanteiden sietämiseen.

Alkaessani tekemään tätä tutkielmaa mielessäni oli kysymys, ovatko DHT:t tai yleisemmin vertaisverkot käyttökelpoisia juuri muuhun kuin kuormanjakoon. Yleisesti ottaen DHT:ihin ja Kademliaan on kohdistunut paljon tutkimusta ja kehitystä niiden yli kymmenen vuoden olemassaolon aikana, ja toisaalta samana aikana Internet-palveluiden kirjo ja niihin kohdistuvat uhat lisääntyneet.

Periaatteessa esimerkiksi Kademlia olisi nähdäkseni tutkielmassa käsiteltyjen menetelmien ja parannusten pohjalta teknisesti valmis käytettäväksi laaja-alaisemmissa sovelluksissa, kuten vaikkapa Twitter-tyyppisissä sosiaalisissa viestintäjärjestelmissä. Täysin avoimessa verkossa on kuitenkin vaikea toteuttaa cypherpunk-ideaalien mukaista anonymiteettia seurantamahdollisuuksien vuoksi, ja lisäksi hitaampi suorituskyky ja vähemmät ominaisuudet keskitettyihin järjestelmiin verrattuna vaikuttavat kaikkien parannustenkin kanssa väistämättömältä. Toisaalta esimerkiksi Usenetin tai Internet Relay Chatin kaltaiset hajautetut järjestelmät, jotka eivät ole yhden organisaation kontrollissa, ovat nähdäkseni tasapuolisen ja vähemmän kaupallisille intresseille alistetun tiedonvälityksen kannalta toivottavia.

Vaikka siis teknologisesti DHT:t olisivatkin valmiita ainakin laajaan kokeelliseen käyttöön, nähdäkseni niiden käyttöönotto on enemmänkin arvokysymys. Koska Twitterin tai Facebookin omistamia hyvin suuria tietoliikenne- ja laskentaresursseja tuskin on mahdollista helposti koota maailmalta tehokkaasti käytettävään muotoon, näkisin, että vertaisverkkoja taustamekanismina käyttävät puolikeskitetyt järjestelmät olisivat potentiaalinen käytännön tie kohti enemmän hajautettuja palveluita. Osittainen keskitys mahdollistaisi toisaalta resurssien kokoamisen tehokkaammiksi ryppäiksi, mutta toisaalta järjestelmä voisi silti sisältää useita erillisiä ja pääosin toisistaan riippumattomia toimijoita. Puolihajautettu lähestyminen mahdollistaisi myös kevyemmät asiakasohjelmat tai selainvälitteiset käyttöliittymät, joiden keskeisyys korostuu esimerkiksi Applen iOS:n kaltaisten ohjelmien toiminnallisuutta rajoittavien ohjelmistoalustojen yleistyessä.

Potentiaalisia DHT:ihin ja Kademliaan liittyviä jatkotutkimuksen kohteita olisivat-

kin juuri sovellutukset puolikeskitettyihin järjestelmiin tai Freenetin kaltaisiin verkkoihin, joissa osanottoa rajoittaa jokin ulkoinen kriteeri. Vaikka toisaalta tähän asti toteutetut verkot ovat toimineet ilmeisen hyväksyttävästi hyvin täysin avoimina ja heikosti suojattuina, tällä tiellä jatkaminen on periaatteellisesta ihailtavuudesta huolimatta nähdäkseni riskialtista. Suojaamattomien verkkojen kohtaamat uhat tuskin vähenevät, ja yhä suurempien verkkojen hallinnan ja puolustusmekanismien monimutkaistumisen aiheuttama toteutusten kompleksisuuden kasvu saattaa tuoda omat hankaluutensa.

Lähteet

Bel01 Bellovin, S. M., Security aspects of Napster and Gnutella, presented at Usenix, kesäkuu 2001. URL `http://www.usenix.org/events/usenix01/invitedtalks/bellovin.pdf`.

BM07 Baumgart, I. ja Mies, S., S/Kademlia: A practicable approach towards secure key-based routing. *Proceedings of the 13th International Conference on Parallel and Distributed Systems - Volume 02*, ICPADS '07, Washington, DC, USA, 2007, IEEE Computer Society, sivut 1–8, URL `http://dx.doi.org/10.1109/ICPADS.2007.4447808`.

CCF09 Cholez, T., Chrisment, I. ja Festor, O., Evaluation of Sybil attacks protection schemes in KAD. *3rd International Conference on Autonomous Infrastructure, Management and Security - AIMS 2009*, Sadre, R. ja Pras, A., toimittajat, osa 5637 sarjasta *Lecture Notes in Computer Science*, Enschede, Netherlands, 2009, University of Twente, Springer, sivut 70–82, URL `http://hal.inria.fr/inria-00405381`.

CCF10 Cholez, T., Chrisment, I. ja Festor, O., Monitoring and Controlling Content Access in KAD. *International Conference on Communications - ICC 2010*, Capetown, South Africa, toukokuu 2010, IEEE, URL `http://hal.inria.fr/inria-00490347`.

CCFD12 Cholez, T., Chrisment, I., Festor, O. ja Doyen, G., Detection and mitigation of localized attacks in a widely deployed P2P network. *Peer-to-Peer Networking and Applications*, sivut 1–20. URL `http://dx.doi.org/10.1007/s12083-012-0137-7`.

CDG+02 Castro, M., Druschel, P., Ganesh, A., Rowstron, A. ja Wallach, D. S., Secure routing for structured peer-to-peer overlay networks. *SIGOPS Oper. Syst. Rev.*, 36,SI(2002), sivut 299–314. URL http://doi.acm.org/10.1145/844128.844156.

CL99 Castro, M. ja Liskov, B., Practical Byzantine fault tolerance. *Proceedings of the third symposium on Operating systems design and implementation*, OSDI '99, Berkeley, CA, USA, 1999, USENIX Association, sivut 173–186, URL http://dx.doi.org/10.1145/571637.571640.

CMD+11 Cholez, T., Montassier, G., Doyen, G., Khatoun, R., Chrisment, I. ja Festor, O., Détection et quantification de la pollution dans le réseau P2P KAD. Tekninen raportti, syyskuu 2011. URL http://hal.inria.fr/hal-00644174.

CSTV10 Clarke, I., Sandberg, O., Toseland, M. ja Verendel, V., Private communication through a network of trusted connections: The dark Freenet, 2010. URL http://amphibian.dyndns.org/freenet-paper-2010-preprint.pdf. Julkaisematon.

DM09 Danezis, G. ja Mittal, P., SybilInfer: Detecting Sybil nodes using social networks. *Proceedings of the Network and Distributed System Security Symposium, NDSS 2009*, Berkeley, CA, USA, helmikuu 2009, URL http://www.isoc.org/isoc/conferences/ndss/09/pdf/06.pdf.

Dou02 Douceur, J. R., The Sybil attack. *Electronic Proceedings for the 1st International Workshop on Peer-to-Peer Systems*, maaliskuu 2002, URL http://www.cs.rice.edu/Conferences/IPTPS02/101.pdf.

Elm12 Elmer, K., Battle lines in the Chinese blogosphere: Keyword control as a tactic in managing mass incidents. Working paper, The Finnish Institute of International Affairs, lokakuu 2012. URL http://www.fiia.fi/fi/publication/293/battle_lines_in_the_chinese_blogosphere/.

Fre10 Freedman, M. J., Experiences with CoralCDN: a five-year operational view. *Proceedings of the 7th USENIX conference on Networked systems design and implementation*, NSDI'10, Berkeley, CA, USA, 2010, USENIX Association, sivut 7–7, URL http://portal.acm.org/citation.cfm?id=1855711.1855718.

GKLL09 Geambasu, R., Kohno, T., Levy, A. ja Levy, H. M., Vanish: Increasing data privacy with self-destructing data. *Proc. of the 18th USENIX Security Symposium*, 2009, URL `http://www.usenix.org/events/sec09/tech/full_papers/geambasu.pdf`.

HKD07 Haeberlen, A., Kuznetsov, P. ja Druschel, P., PeerReview: Practical accountability for distributed systems. *Proceedings of the 21st ACM Symposium on Operating Systems Principles (SOSP'07)*, lokakuu 2007, URL `http://dx.doi.org/10.1145/1294261.1294279`.

HSD+08 Holz, T., Steiner, M., Dahl, F., Biersack, E. ja Freiling, F., Measurements and mitigation of peer-to-peer-based botnets: a case study on storm worm. *Proceedings of the 1st Usenix Workshop on Large-Scale Exploits and Emergent Threats*, Berkeley, CA, USA, 2008, USENIX Association, sivut 9:1–9:9, URL `http://portal.acm.org/citation.cfm?id=1387709.1387718`.

JMu11 JMule project contributors, Kad activity report, julkaistu JMule-projektin kotisivuilla, tammikuu 2011. URL `http://jmule.org/files/kad-activities/january2011/january2011-kadactivity-report.pdf`.

KGK+08 Kovacevic, A., Graffi, K., Kaune, S., Leng, C. ja Steinmetz, R., Towards benchmarking of structured peer-to-peer overlays for network virtual environments. *Proceedings of the 2nd International Workshop on Peer-to-Peer Network Virtual Environments (P2P-NVE 2008)*, Melbourne, Australia, joulukuu 2008, URL `http://dx.doi.org/10.1109/ICPADS.2008.68`.

KLR09 Kohnen, M., Leske, M. ja Rathgeb, E., Conducting and optimizing Eclipse attacks in the Kad peer-to-peer network. Teoksessa *NETWORKING 2009*, Fratta, L., Schulzrinne, H., Takahashi, Y. ja Spaniol, O., toimittajat, osa 5550 sarjasta *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, 2009, sivut 104–116, URL `http://dx.doi.org/10.1007/978-3-642-01399-7_9`.

Koh12a Kohnen, M., Analysis and optimization of routing trust values in a Kademlia-based Distributed Hash Table in a malicious environment. *Future Internet Communications (BCFIC), 2012 2nd Baltic Congress*

on, huhtikuu 2012, sivut 252–259, URL http://dx.doi.org/10.1109/BCFIC.2012.6217954.

Koh12b Kohnen, M., Applying trust and reputation mechanisms to a Kademlia-based Distributed Hash Table. *Proceedings of the IEEE International Conference on Communications (ICC 2012)*, Ottawa, Kanada, kesäkuu 2012.

LMSW09 Locher, T., Mysicka, D., Schmid, S. ja Wattenhofer, R., A peer activity study in eDonkey and Kad. *International Workshop on Dynamic Networks: Algorithms and Security (DYNAS), Wroclaw, Poland*, syyskuu 2009, URL http://www.dcg.ethz.ch/publications/dynas09.pdf.

Loe08 Loewenstern, A., DHT protocol, helmikuu 2008. URL http://www.bittorrent.org/beps/bep_0005.html.

MM02 Maymounkov, P. ja Mazières, D., Kademlia: A peer-to-peer information system based on the XOR metric. *Peer-to-Peer Systems, First International Workshop, IPTPS 2002, Cambridge, MA, USA, March 7-8, 2002, Revised Papers*, 2002, sivut 53–65, URL http://www.springerlink.com/content/2ekx2a76ptwd24qt/.

MRGS09 Memon, G., Rejaie, R., Guo, Y. ja Stutzbach, D., Large-scale monitoring of DHT traffic. *Proceedings of the 8th international conference on Peer-to-peer systems*, Proceedings of the 8th International Workshop on Peer-to-Peer Systems (IPTPS), Berkeley, CA, USA, 2009, USENIX Association, URL http://www.usenix.org/event/iptps09/tech/full_papers/memon/memon.pdf.

MRGS12 Memon, G., Rejaie, R., Guo, Y. ja Stutzbach, D., Montra: A large-scale DHT traffic monitor. *Computer Networks*, 56,3(2012), sivut 1080–1091. URL http://dx.doi.org/10.1016/j.comnet.2011.11.010.

Pir09 Pirate Bay, The, Worlds most resiliant tracking, marraskuu 2009. URL http://thepiratebay.org/blog/175.

RD01 Rowstron, A. I. T. ja Druschel, P., Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. *Middleware 2001, IFIP/ACM International Conference on Distributed Sys-*

tems Platforms Proceedings. Springer, 2001, sivut 329–350, URL http://www.springerlink.com/content/404522p56nm85503/.

REMLP07 Rowaihy, H., Enck, W., McDaniel, P. ja La Porta, T., Limiting Sybil attacks in structured P2P networks. *INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE*, toukokuu 2007, sivut 2596–2600, URL http://dx.doi.org/10.1109/INFCOM.2007.328.

RF02 Ripeanu, M. ja Foster, I., Mapping the Gnutella network: Macroscopic properties of large-scale peer-to-peer systems. Teoksessa *Peer-to-Peer Systems*, Druschel, P., Kaashoek, F. ja Rowstron, A., toimittajat, osa 2429 sarjasta *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, 2002, sivut 85–93, URL http://dx.doi.org/10.1007/3-540-45748-8_8.

Rhe Rhea, S., OpenDHT: A publicly accessible DHT service. URL http://www.opendht.org/.

Roc11 Rockefeller, H., The HB Gary email that should concern us all, in a blog entry at Daily Kos, helmikuu 2011. URL http://www.dailykos.com/story/2011/02/16/945768/-UPDATED:-The-HB-Gary-Email-That-Should-Concern-Us-All.

SB08 Steiner, M. ja Biersack, E. W., Crawling Azureus. Tekninen raportti EURECOM+2495, Institut Eurecom, France, kesäkuu 2008. URL http://www.eurecom.fr/~btroup/BPublished/RR-08-223.pdf.

SBEN07 Steiner, M., Biersack, E. W. ja En-Najjary, T., Actively monitoring peers in Kad. *Proceedings of the 6th International Workshop on Peer-to-Peer Systems (IPTPS), Bellevue, Washington, USA*, 2007, URL http://www.eurecom.fr/~btroup/BPublished/StBE07MonitoringKad.pdf.

SCDR04 Singh, A., Castro, M., Druschel, P. ja Rowstron, A., Defending against eclipse attacks on overlay networks. *Proceedings of the 11th workshop on ACM SIGOPS European workshop*, EW 11, New York, NY, USA, 2004, ACM, URL http://doi.acm.org/10.1145/1133572.1133613.

SENB07 Steiner, M., En-Najjary, T. ja Biersack, E. W., Exploiting KAD: possible uses and misuses. *SIGCOMM Comput. Commun. Rev.*, 37,5(2007), sivut 65–70. URL http://doi.acm.org/10.1145/1290168.1290176.

SENB09 Steiner, M., En-Najjary, T. ja Biersack, E. W., Long term study of peer behavior in the Kad DHT. *IEEE/ACM Trans. Netw.*, 17, sivut 1371–1384. URL http://dx.doi.org/10.1109/TNET.2008.2009053.

SG05 Subramanian, R. ja Goodman, B. D. *Peer to Peer Computing: The Evolution of a Disruptive Technology*, sivut 84–85. Idea Group Publishing, 2005.

SGG02 Saroiu, S., Gummadi, P. K. ja Gribble, S. D., A measurement study of peer-to-peer file sharing systems. Tekninen raportti UW-CSE-01-06-02, University of Washington, Department of Computer Science and Engineering, 2002.

SM02 Sit, E. ja Morris, R., Security considerations for peer-to-peer distributed hash tables. *Revised Papers from the First International Workshop on Peer-to-Peer Systems*, IPTPS '01, London, UK, 2002, Springer-Verlag, sivut 261–269, URL http://portal.acm.org/citation.cfm?id=646334.687810.

SMK+01 Stoica, I., Morris, R., Karger, D., Kaashoek, M. F. ja Balakrishnan, H., Chord: A scalable peer-to-peer lookup service for Internet applications. *SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, New York, NY, USA, 2001, ACM, sivut 149–160, URL http://doi.acm.org/10.1145/383059.383071.

SNDW06 Singh, A., Ngan, T.-W., Druschel, P. ja Wallach, D. S., Eclipse attacks on overlay networks: Threats and defenses. *INFOCOM 2006. 25th IEEE International Conference on Computer Communications, Joint Conference of the IEEE Computer and Communications Societies*, huhtikuu 2006, URL http://dx.doi.org/10.1109/INFOCOM.2006.231.

SR06 Stutzbach, D. ja Rejaie, R., Improving lookup performance over a widely-deployed DHT. *INFOCOM 2006. 25th IEEE International Con-*

ference on Computer Communications. Proceedings, huhtikuu 2006, sivut 1–12, URL http://dx.doi.org/10.1109/INFOCOM.2006.329.

Ste09 Steiner, M., *Structures and Algorithms for Peer-to-Peer Cooperation*. Väitöskirja, Universität Mannheim, kesäkuu 2009. URL http://madoc.bib.uni-mannheim.de/madoc/volltexte/2009/2149/.

TCCF11 Timpanaro, J. P., Cholez, T., Chrisment, I. ja Festor, O., BitTorrent's Mainline DHT Security Assessment. *4th IFIP International Conference on New Technologies, Mobility and Security - NTMS 2011*, Pariisi, Ranska, helmikuu 2011, IEEE, URL http://hal.inria.fr/inria-00577043. Projet GIS 3SGS ACDAP2P (Approche collaborative pour la détection d'attaques dans les réseaux pair à pair).

TKLB07 Terpstra, W. W., Kangasharju, J., Leng, C. ja Buchmann, A. P., Bubblestorm: resilient, probabilistic, and exhaustive peer-to-peer search. *SIGCOMM Comput. Commun. Rev.*, 37,4(2007), sivut 49–60. URL http://doi.acm.org/10.1145/1282427.1282387.

Tor08 TorrentFreak, uTorrent grows to 28 million monthly users, joulukuu 2008. URL http://torrentfreak.com/utorrent-grows-to-28-million-monthly-users-081225/.

UPS11 Urdaneta, G., Pierre, G. ja Steen, M. V., A survey of DHT security techniques. *ACM Comput. Surv.*, 43,2(2011), sivut 8:1–8:49. URL http://doi.acm.org/10.1145/1883612.1883615.

VPGM10 Viswanath, B., Post, A., Gummadi, K. P. ja Mislove, A., An analysis of social network-based Sybil defenses. *SIGCOMM Comput. Commun. Rev.*, 40,4(2010), sivut 363–374. URL http://doi.acm.org/10.1145/1851275.1851226.

Vuz12 Vuze, Distributed hash table, lokakuu 2012. URL http://wiki.vuze.com/w/Distributed_hash_table.

WH10 Wolchok, S. ja Halderman, J. A., Crawling BitTorrent DHTs for fun and profit. *Proceedings of the 4th USENIX conference on Offensive technologies*, WOOT'10, Berkeley, CA, USA, 2010, USENIX Association, sivut 1–8, URL http://www.usenix.org/event/woot10/tech/full_papers/Wolchok.pdf.

WHH+09 Wolchok, S., Hofmann, O. S., Heninger, N., Felten, E. W., Halderman, J. A., Rossbach, C. J., Waters, B. ja Witchel, E., Defeating Vanish with low-cost Sybil attacks against large DHTs, syyskuu 2009. URL http://z.cs.utexas.edu/users/osa/unvanish/papers/vanish-broken.pdf. Julkaisematon.

WTCT+08 Wang, P., Tyra, J., Chan-Tin, E., Malchow, T., Kune, D. F., Hopper, N. ja Kim, Y., Attacking the Kad network. *SecureComm '08: Proceedings of the 4th international conference on Security and privacy in communication netowrks*, New York, NY, USA, 2008, ACM, sivut 1–10, URL http://doi.acm.org/10.1145/1460877.1460907.

YFX+09 Yu, J., Fang, C., Xu, J., Chang, E.-C. ja Li, Z., ID repetition in Kad. *Proceedings P2P 2009, Ninth International Conference on Peer-to-Peer Computing*, Seattle, Washington, USA, syyskuu 2009, sivut 111–120, URL http://dx.doi.org/10.1109/P2P.2009.5284551.

YGKX08 Yu, H., Gibbons, P., Kaminsky, M. ja Xiao, F., SybilLimit: A near-optimal social network defense against sybil attacks. *Security and Privacy, 2008. SP 2008. IEEE Symposium on*, sivut 3–17. URL http://dx.doi.org/10.1109/SP.2008.13.

YKGF06 Yu, H., Kaminsky, M., Gibbons, P. B. ja Flaxman, A., SybilGuard: defending against sybil attacks via social networks. *SIGCOMM '06: Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications*, New York, NY, USA, 2006, ACM, sivut 267–278, URL http://doi.acm.org/10.1145/1159913.1159945.

YKGK12 Young, M., Kate, A., Goldberg, I. ja Karsten, M., Towards practical communication in Byzantine-resistant DHTs. *Networking, IEEE/ACM Transactions on*, PP,99(2012). URL http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp=&arnumber=6196191.

YXLZ11 Yu, J., Xiao, P., Li, Z. ja Zhou, Y., Toward an accurate snapshot of DHT networks. *Communications Letters, IEEE*, 15,1(2011), sivut 97–99. URL http://dx.doi.org/10.1109/LCOMM.2010.110310.101027.