KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS

Julius Kriukas

Laisvai parenkamo mazgo identifikatoriaus įtakos DHT tinklo saugumui analizė

Magistro darbas

Darbo vadovas:
doc. dr. Rimantas Kavaliūnas

Kaunas
2012

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS

Julius Kriukas

Laisvai parenkamo mazgo identifikatoriaus įtakos DHT tinklo saugumui analizė

Magistro darbas

Vadovas
doc. dr. Rimantas Kavaliūnas
2012-05-28

Recenzentas:
Dr. Tomas Blažauskas
2012-05-28

Atliko:
Julius Kriukas
2012-05-28

Kaunas
2012

TURINYS

1 ĮVADAS

Paskirstytos sistemos yra tokios sistemos, kurios sudarytos iš daug nedidelį resursų kiekį turinčių įrenginių koordinuotai dirbančių vienam bendram tikslui pasiekti. Paskirstytos sistemos leidžia efektyviai panaudoti nedidelius įrenginių resursus tam kad būtu atliekamas darbas reikalaujantis daugiau resursų nei atskirai turi kuris nors vienas įrenginys. Speciali paskirstytų sistemų klasė yra lygiarangės (toliau žymimos P2P) sistemos. P2P sistemose visi sistemos nariai turi vienodas teises atlikti bet kurią sistemos funkcija. Ne P2P sistemos vadinamos centralizuotomis (arba kliento-serverio) sistemomis.

P2P sistemos yra lengviau plečiamos nei centralizuotos sistemos, kiekvienas naujas mazgas vienodai gali prisidėti prie bendros tinklo veiklos. Taip pat P2P sistemos yra atsparesnės trikdžiams, jose nėra specializuotas funkcijas atliekančių mazgų, kurių negalėtų pakeisti kitas mazgas. Internet tinklą taip pat galima laikyti P2P sistema, kurioje vienas mazgas gali perduoti informacijos žinutę bet kuriam kitam mazgui. Dėl savitos problematikos P2P sistemos yra aktyviai nagrinėjamos akademinėje visuomenėje.

Paskirstyta maišos lentelė (toliau žymima DHT), tai paskirstyta sistema, kurios pagrindinis tikslas yra suformuoti paskirstytą duomenų bazę, kurioje kiekvienas mazgas gali išsaugoti duomenis arba atlikti duomenų paiešką. DHT sistemoje saugojamiems duomenims suteikiamas identifikatorius, kuris vėliau gali būti naudojamas duomenų radimui. DHT sistemoje saugojami duomenys yra paskirstomi mazgams, taip kad mazgui išėjus arba atėjus naujam mazgui duomenų paskirstymas mazgams kiek įmanoma išliktų pastovus. Greitai duomenų paieškai ir veiksmų koordinavimui DHT sistemos mazgai suformuoja tinklą užmegzdami ryšius su kitais strategiškai patogiais mazgais, dėl to DHT sistemos dažnai vadinamos DHT tinklais. Šiame darbe nagrinėjami DHT panaudojimo P2P sistemose atvejai.

DHT tinklai yra pakankamai universali paskirstyta duomenų struktūra, todėl jie praktikoje dažnai naudojami, kaip sudėtingesnės sistemos komponentai. DHT tinklai naudojami paskirstytose duomenų bazėse, turinio paskirstymo tinkluose (angl. CDN, Content Distribution Network), web paieškos sistemose, failų apsikeitimo programose bei anoniminiam informacijos platinimui. Dėl plataus DHT tinklų panaudojimo, saugumo problemos juose yra aktyviai nagrinėjama sritis. Didelė dalis literatūros šaltinių nagrinėja saugumo problemas remiantis teoriniais DHT modeliais, tačiau nerasta šaltinių kuriuose saugumas būtų nagrinėjamas tais atvejais, kai praktinėse realizacijose nėra tenkinamos visos teorinio modelio keliamos sąlygos.

Šiame darbe keliama hipotezė, kad praktinės DHT tinklų realizacijos neužtikrina teoriniuose modeliuose priimtos sąlygos, kad mazgų prisijungiančių prie tinklo identifikatoriai bus generuojami atsitiktinai. Randamas atakų sudėtingumo įvertis, kai atsitiktinių identifikatorių generavimas yra privalomas ir kai identifikatorius galima pasirinkti laisvai. Hipotezės patvirtinimui atliekamas eksperimentas. Surenkami ir analizuojami vieno didžiausių DHT tinklų (BitTorrent DHT) duomenys. Aprašomas literatūroje siūlomas problemos sprendimo būdas ir praktinės problemos kylančios jį realizuojant. Pasiūlomas naujas praktiškai pritaikomas ir našus identifikatorių generavimo patikrinimo metodas bei metodas naujų identifikatorių generavimo greičiui DHT tinkle valdyti.

Pagrindinė darbo medžiaga pateikiama nuo antro iki septinto skyriuose, kurie suskirstyti tokiu būdu:

Antrajame skyriuje apžvelgiami keturi klasikiniai DHT modeliai ir pateikiama DHT atakų klasifikacija.

Trečiajame skyriuje suformuluojamas darbo tikslas ir uždaviniai.

Ketvirtajame skyriuje įvertinamas teorinis atakų prieš DHT tinklus sudėtingumas. Įvertinamas Eclipse atakai atlikti reikalingas mazgų skaičius ir identifikatorių generavimo operacijų skaičius, kai tinkle identifikatorių laisvai pasirinkti negalima.

Penktajame skyriuje aprašomi BitTorrent DHT tinklo eksperimentiniai rezultatai. Parodoma, kad didelė tinklo dalis naudoja ne atsitiktinai generuotus identifikatorius, tinkle pasitaiko identifikatorių kopijavimo atvejai, tinke aktyviai vykdomos Eclipse atakos.

Šeštajame skyriuje apžvelgiamos praktinės mazgų identifikatorių generavimo įrodymo ir apsaugos nuo kopijavimo problemos. Pateikiamas naujas identifikatorių apsaugos metodas, kuris pranašesnis už klasikinius savo našumu ir galimybe jį lengvai pritaikyti prie žinutėmis grįstų DHT protokolų. Pateikiami eksperimentai įvertinantis siūlomo metodo našumą.

Septintajame skyriuje aprašoma problema naujų mazgų generavimo greičio problema. Pateikiamas metodas leidžiantis reguliuoti naujų mazgų prisijungimo prie tinklo greitį, pateikiama jo integracijos su identifikatorių apsaugos sistema schema. Atliekamas eksperimentas įvertinantis siūlomo metodo efektyvumą.

2 PASKIRSTYTOS MAIŠOS LENTELĖS (DHT)

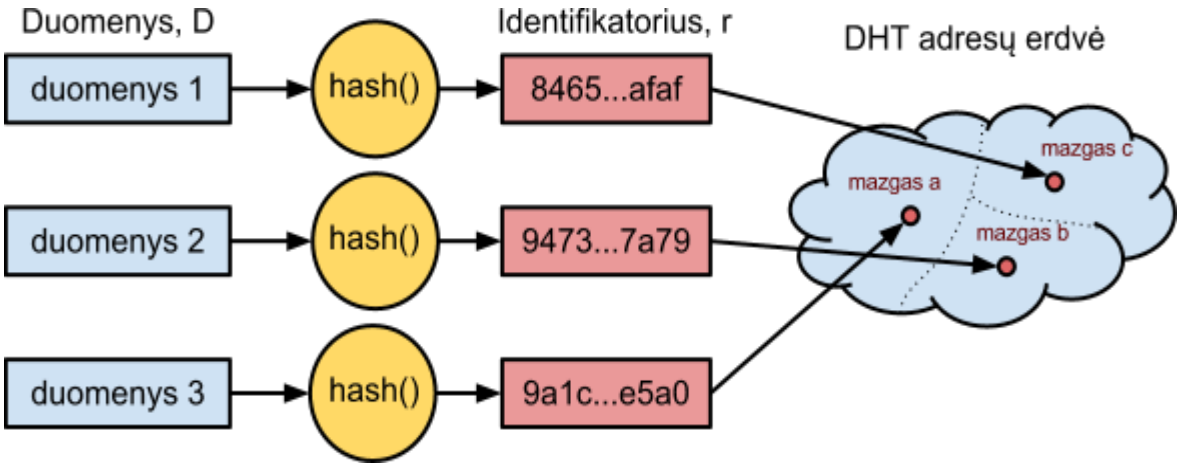
2.1 DHT veikimo principas

DHT funkcionalumas yra panašus kaip įprastos maišos lentelės duomenų struktūros. Įprastos maišos lentelės yra skirtos greitam duomenų pasiekimui. Jos realizuojamos naudojant

maišos funkciją ir rodyklių masyvą. Ieškomi duomenys (arba ieškomų duomenų indeksas) hešuojami, o gautas rezultatas yra masyvo indeksas, kuriame saugoma rodyklė į ieškomus duomenis.

DHT yra panaši duomenų struktūra, tačiau ji realizuojama ne vienos programos viduje, o kaip paskirstyta sistema vienu metu veikianti daugelyje kompiuterių. DHT tinkle veikiantys mazgai suformuoja bendrą paskirstytą duomenų saugyklą. Pagrindinis DHT uždavinys kaip ir įprastų maišos lentelių - žinant duomenų identifikatorių (pačių duomenų arba jų indekso maišos funkcijos reikšmę) lengvai įterpti arba paimti duomenis iš DHT tinklo.

DHT tinklo modelis nusako adresų erdvę, iš kurios duomenims ir mazgams priskiriami identifikatoriai (adresai). Visa identifikatorių erdvė padalinama mazgams, taip kad kiekvienas mazgas saugo tuos duomenis, kurių identifikatoriai patenka į tam mazgui priskirtą erdvės segmentą. Adresų erdvės padalinimas organizuojamas taip, kad nepriklausytų nuo tinkle veikiančių mazgų skaičiaus, todėl visa duomenų adresų erdvė įrenginiams yra dalinama dinamiškai priklausomai nuo tuo metu tinkle dalyvaujančių įrenginių skaičiaus. Principinė DHT veikimo schema pateikta 1 pav.



1 pav. Principinė DHT schema

Kadangi prie DHT tinklo galima lengvai prijungti praktiškai neribotą įrenginių skaičių šios sistemos yra labai lanksčios plėtimui. Kitos naudingos DHT savybės yra tinklo dinamiškumas ir patikimumas. DHT tinklai formuojami taip, kad bet kuriuo metu prie tinklo įrenginiai galėtų prisijungti ir jį palikti pačiame tinkle neprarandant duomenų.

Konkretaus DHT veikimą galima išskirti į dvi dalis - adresų paskirstymą ir tinklo suformavimą. Adresų paskirstymo uždavinys sprendžia kokia yra raktų identifikuojančių duomenis erdvė ir kaip ji išdalinama DHT tinklo mazgams. Tinklo suformavimo uždavinys sprendžia kaip mazgai komunikuoja ir susijungia į vientisą tinklą.

2.2 DHT taikymo sritys

DHT kaip ir įprasta maišos lentelės duomenų struktūra yra pakankamai abstraktus ir nuo konkrečios taikymo srities nepriklausantis komponentas. DHT taikymo sritis galima išskirti į dvi grupes pasinaudojančias atskiromis DHT savybėmis. Pirmoji grupė, tai taikymai naudojantys DHT dėl lengvo išplečiamumo ir galimybės galingus serverius pakeisti dideliu kiekiu silpnesnių įrenginių. Šioje kategorijoje yra paskirstytos duomenų bazės, web paieškos varikliai, turinio paskirstymo tinklai. Šio tipo DHT taikymų pavyzdžiai:

- Apache Cassandra - Paskirstytos duomenų bazės valdymo sistema skirta saugoti didelius duomenų kiekius nedidelių pajėgumų kompiuteriuose [1].
- CloudSNAP - P2P debesų skaičiavimų infrastruktūra leidžianti paleisti įprastas web aplikacijas P2P debesies platformoje [2].
- Coral CDN - web kešas ir turinio paskirstymo tinklas (angl. Content Distribution Network). P2P tinkle saugant populiarius web serverio failus web serverio apkrovimas paskirstomas daugeliui silpnesnių mazgų [3].
- FAROO - paskirstytas web puslapių paieškos variklis. Su papildoma programine įranga vartotojai savo kompiuterius sujungia į DHT tinklą kuriame indeksuojami ir saugomi visi aplankomi tinklapiai ir jų lankomumo statistika. Prieinantis prie šios duomenų bazės vartotojai gali atlikti web puslapių paiešką.
- JXTA - atviro kodo protokolas skirtas sujungti mazgus (negalinčius tiesiogiai bendrauti dėl ugniasienių arba NAT) į bendrą tinklą. Dalis šio protokolo aprašo informacijos talpinimą į paskirstytą DHT principu veikiančią duomenų bazę [4].
- Oracle Coherence - komercinis Oracle produktas skirtas duomenų kešavimo replikavimo ir paskirstymo paslaugoms.
- YaCy - paskirstytas web paieškos variklis. Paskirstyta web puslapių paieškos duomenų bazė yra atspari tinklo rezultatų filtravimui, bei leidžia užtikrinti ieškančiojo privatumą [5].

Antroji taikymo grupė naudojasi tuo, kad duomenys DHT tinkle yra decentralizuoti ir dėl to sunkiai paveikiami arba pašalinami iš tinklo. Ši savybė naudinga failų apsikeitimo tinklams ir tinklams skirtiems apsisaugoti nuo interneto turinio filtravimo ar kontrolės. Taikymų pavyzdžiai:

- GnuNET - atviro kodo failų dalinimosi tinklas skirtas anonimiškai įkelti ir paimti duomenis iš tinklo [6].
- FreeNet - anoniminio informacijos talpinimo tinklas [7].
- eMule Kad network - failų apsikeitimo programos eMule praplėtimas, panaudojantis DHT failų paieškai.

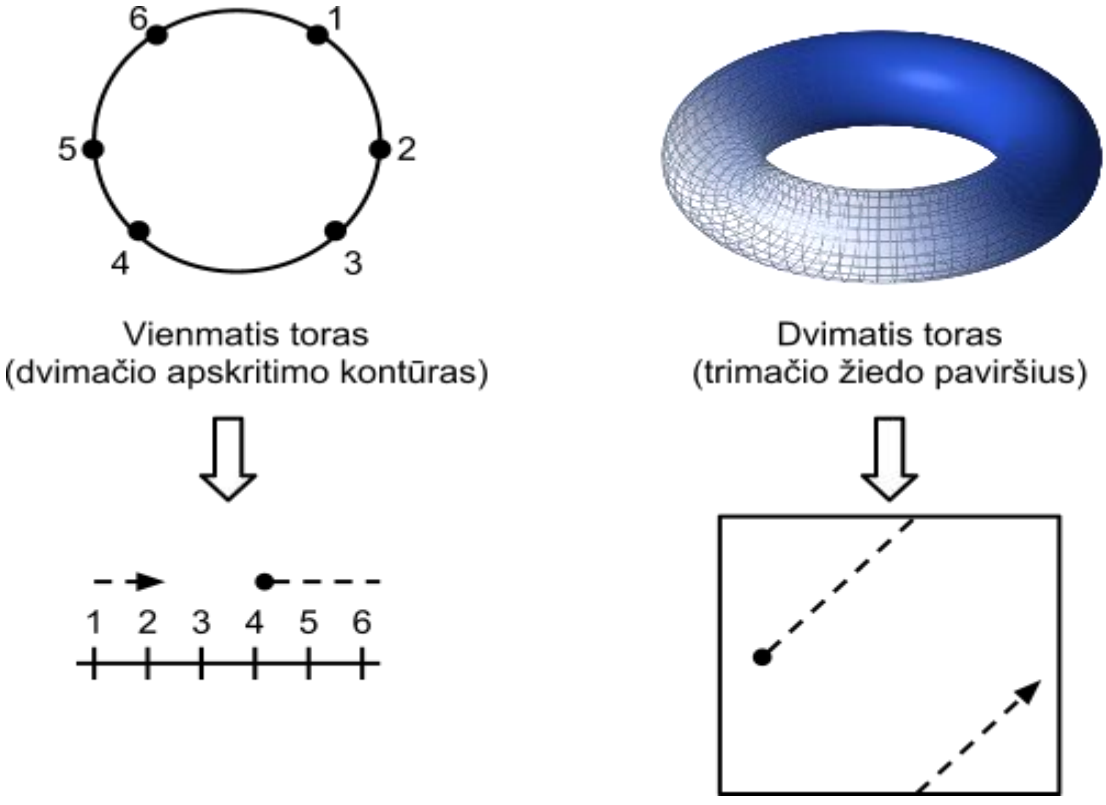
- BitTorrent DHT - failų siuntimosi protokolo BitTorrent praplėtimas leidžiantis tinklui veikti be siuntimą koordinuojančių serverių (angl. trackers).

2.3 DHT tinklo modeliai

Šiame skyriuje aprašomi penki populiariausi ir dažniausiai mokslinėje literatūroje sutinkami DHT modeliai - CAN [8], Chord [9], Pastry [10], Tapestry [11] ir Kademlia [12].

2.3.1 CAN modelis

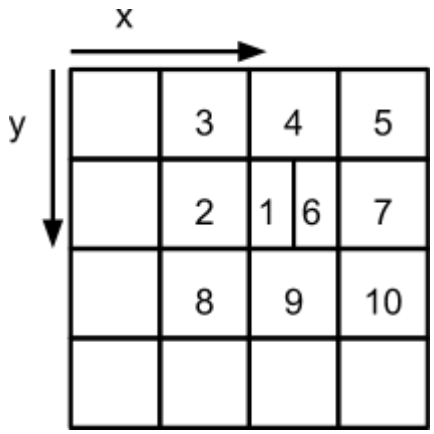
CAN (angl. Content-Addressable Network) - tai vienas iš pirmųjų žymių DHT modelių. CAN tinkle duomenų adresų erdvė yra traktuojama kaip *d* matavimų (d-matis) toras. Toras tai toks matematinis koordinačių modelis, kuriame kiekviena ašis yra sujungta į ciklą. (alternatyvi formuluotė) Toras, tai toks matematinis modelis, kuriame kiekvienos matavimų ašies paskutinė reikšmė yra gretima šios matavimų ašies pradiniam taškui. Grafiškai matematinį toro modelį galima lengvai pavaizduoti *d*+1 matmenų erdvėje. Vienmačio ir dvimačio toro pavyzdžiai pateikti 2 pav.



2 pav. Vienmatis ir dvimatis toras

CAN tinkle visą galimų adresų d-matė erdvė yra dinamiškai padalinama į tiek dalių kiek DHT tinkle yra dalyvaujančių mazgų. Adresų erdvės dalys yra d-mačiai kubai. Dėl tokio padalinimo kiekvienas mazgas turi po $2d$ kaimynus - kiekviena ašimi po vieną kaimyną esantį prieš mazgą ir po vieną esantį už mazgo. Mazgai tarpusavyje suformuoja tinklą

sudarydami ryšius su kaimyniniais mazgais. Dvimačio toro erdvės sudalinimo pavyzdys pateikiamas 3 pav. Mazgo nr. 1 kaimynai x ašimi yra mazgai 2 ir 6, o y ašimi 4 ir 9.



3 pav. Dvimačio toro adresų erdvės padalinimo pavyzdys

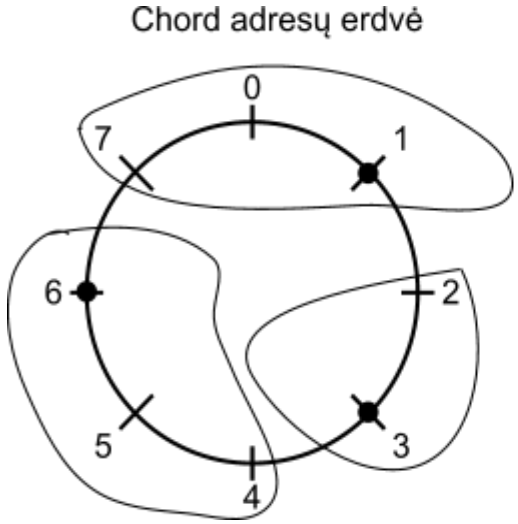
Duomenų paieška tokiame tinkle vykdoma apskaičiuojant ieškomo duomenų bloko padėtį adresų erdvėje ir einant per kaimyninius mazgus link mazgo saugančio duomenų bloką trumpiausiu keliu. Tinkle kuriame veikia M tolygiai adresų erdvėje išsidėsčiusių mazgų adresų erdvė yra padalinta į M panašaus dydžio segmentų, tuomet vidutinis kelias tarp bet kurių mazgų yra $\frac{d}{4}M^{-d}$ [8]. Didinant adresų erdvės matavimų skaičių d gaunami trumpesni vidutiniai keliai tarp mazgų. Prie konkrečios d reikšmės kiekvienam mazgui tenka fiksuotas kaimynų skaičius nepriklausomas nuo bendro mazgų skaičiaus. Kelio radimo operacijos sudėtingumas CAN tinkle yra $O(\sqrt[d]{M})$.

Mazgams prisijungiant prie CAN tinklo vykdomos operacijos - mazgo pozicijos radimas, adresų erdvės padalinimas ir informacijos iš mazgų kurio adresų erdvė buvo perskirta paėmimas. Mazgui atsijungiant nuo tinklo vykdoma adresų erdvės suliejimo operacija, detaliau šios operacijos aprašytos [8]. Bazinis tinklo modelis nenumato duomenų replikavimo mechanizmo, todėl sugedus vienam tinklo mazgui prarandami visi mazge saugoti duomenys. Papildomai duomenų apsaugai ir spartesnei duomenų paieškai tinkle, kiekvienas tinkle dalyvaujantis mazgas gali prisijungi prie k virtualių CAN tinklų, kuriuose mazgas turėtų skirtingą identifikatorių. Kiekviename iš virtualių tinklų būtų saugomi tie patys duomenys, todėl atlikdamas duomenų paiešką mazgas galėtų pasirinkti tą virtualų tinklą, kuriame atstumas nuo mazgo iki duomenų yra mažiausias.

2.3.2 Chord modelis

Kitas plačiai žinomas DHT tinklo modelis yra Chord. Šiame modelyje raktų erdvė yra vaizduojama, kaip vienmatis toras arba tiesiog uždaras galimų raktų reikšmių ciklas. Mazgai Chord tinkle identifikuojami tokio pati ilgio adresais, kaip ir duomenys. Tam, kad duomenys identifikatorių erdvėje pasiskirstytų tolygiai duomenų identifikatorius apskaičiuojamas kaip

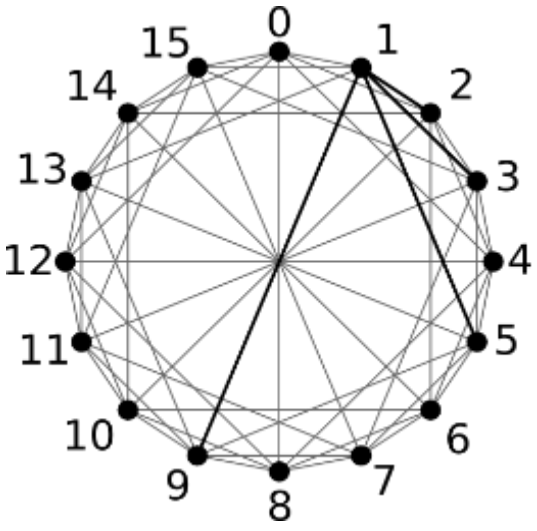
duomenų kriptografinės maišos funkcijos SHA-1 [13] reikšmė. Duomenų erdvė mazgams dalinama tokiu būdu: tarkime tinkle, kurio adresų erdvė sudaryta iš *N* skirtingų identifikatorių yra *M* mazgų, kurių identifikatoriai atitinkamai yra $\{x_1, x_2, \dots, x_M\}$, tuomet mazgas i, kur $i \in [1, M]$, kurio adresas $x_i \in [0, N-1]$ saugo tuos duomenis, kurių identifikatoriai y priklauso intervalui reikšmių esančių tarp mazgo identifikatoriaus ir prieš jį esančio mazgo identifikatoriaus $y \in (x_{n-1}, x_n)$. Grafiškai adresų erdvės padalinimas pavaizduotas 4 pav.



Mazgo ID, x	Duomenų identifikatoriai, y
1	7, 0, 1
3	2, 3
6	4, 5, 6

4 pav. Adresų erdvės padalinimo Chord modelyje pavyzdys

Tinklo suformavimui ir palaikymui kiekvienas mazgas sudaro maršrutų lentelę, kurioje yra *n* įrašų, čia *n* yra adresų erdvės ilgis bitais, $n = \log_2(N)$. Įrašai numeruojami $i \in [0, n-1]$, tuomet mazgo *x* *i*-tojoje maršrutų lentelės eilutėje saugomas IP adresas ir identifikatorius mazgo, kuriam priklauso $y = x + 2^i$ identifikatorius. Jei tokį identifikatorių turinčio mazgo nėra, randamas pirmas mazgas kurio identifikatorius didesnis už y. Pavyzdžiui tinkle, kurio identifikatorių ilgis *n*=4 bitai ir kuriame yra visi 16 skirtingus identifikatorius turintys mazgai, mazgas *x*=1 maršrutų lentelėje saugos informaciją apie mazgus 2, 3, 5 ir 9. Grafiškai tokio tinklo pavyzdys pateiktas 5 pav.



Mazgo x=1 maršrutų lentelė

i	kaimyno ID, x+2^i
0	2
1	3
2	5
3	9

Mazgo x=1 ryšiai
Kitų mazgų ryšiai

5 pav. Mazgo x=1 maršrutų lentelė

Maršrutų lentelė naudojama duomenų paieškai. Vidutinis paieškos operacijų skaičius Chord DHT tinkle yra $\frac{1}{2}\log_2(M)$ [9], tai neformaliai galima įrodyti pastebėjimu, kad kiekviename maršrutų parinkimo žingsnyje atstumas iki duomenų sumažėja ne mažiau kaip pusiau. Bendra paieškos sudėtingumas yra $O(\log_2(M))$ operacijų.

2.3.3 Pastry ir Tapestry modeliai

Pastry [10] ir Tapestry [11] yra dar du gerai žinomi DHT modeliai. Kadangi abiejų modelių veikimo principas labai panašus detaliau bus aprašytas tik Pastry modelis. Pastry DHT modelyje adresų erdvė panašiai, kaip ir Chord tinkle yra vienmatė, tačiau nusakoma, ne tik adresų erdvės skaitmenų ilgiu *n* bet ir adreso kodavimo baze *b*. Jei adresų erdvės taškų skaičių pažymime *N*, tuomet gauname $N = b^m$. Pavyzdžiui pasirinkus šešioliktainę kodavimo bazę b=16 ir adresų erdvės skaitmenų kiekį n=6, gausime adresų erdvę, kurioje galimos adresų reikšmės yra nuo `0x000000` iki `0xFFFFFF`.

Duomenų paieškos tinklas Pastry modelyje sudaromas tokiu būdų. Kiekvienas mazgas turi *n* maršrutų lentelių. Šiose lentelėse saugomi kitų tinklo mazgų adresai tokie, kad *i*-tojoje maršrutų lentelėje yra tik tie mazgai kurių pirmi *i*-1 adreso skaitmenys sutampa su lentelę saugančio mazgo adresu. Pavyzdžiui, jei tinkle b=16; m=6, mazgo adresas yra `0x37A59C` tuomet pirmojoje maršrutų lentelėje gali būti saugomi `0x******` mazgų adresai, kur * reiškia bet kokį skaitmenį, antrojoje lentelėje `0x3*****`, trečiojoje `0x37****`, šeštojoje `0x37A59*`. Kiekviena maršrutų lentelė turi po *b* eilučių, kur *j*-ojoje eilutėje saugomas mazgo adresas pirmoje bet kokią reikšmę galinčioje įgyti pozicijoje privalo būti lygus *j*. Maršrutų lentelių pavyzdys pateikiamas 6 pav.

i=1

j	Adreso šablonas
0	0x0*****
1	0x1*****
2	0x2*****
...	...
14	0xE*****
15	0xF*****

i=2

j	Adreso šablonas
0	0x30****
1	0x31****
2	0x32****
...	...
14	0x3E****
15	0x3F****

...

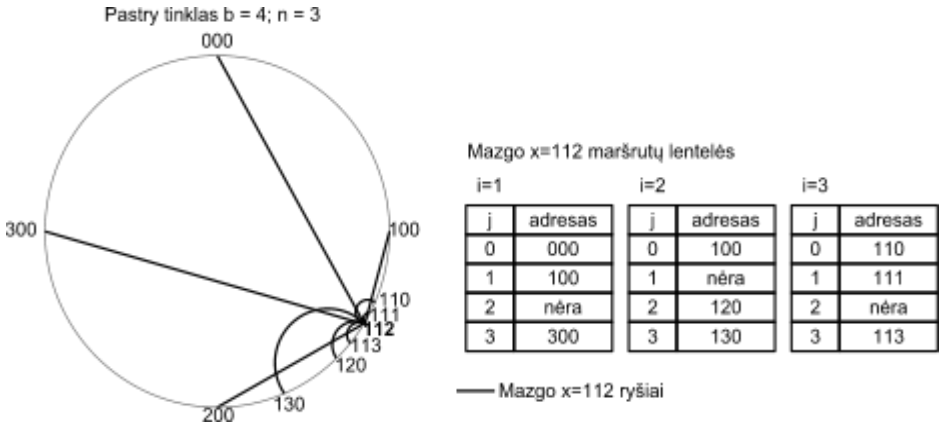
i=6

j	Adreso šablonas
0	0x34A590
1	0x34A591
2	0x34A592
...	...
14	0x34A59E
15	0x34A59F

6 pav. Mazgo x = 0x37A59C; m=6; b=16 Pastry tinkle maršrutų lentelės pavyzdys

Maršrutų lentelės nebūtinai turi būti užpildytos pilnai. Iš pavyzdžio galima pastebėti, kad paskutinėje lentelėje saugomi gretimi mazgai ir ji gali būti užpildyta tiktai tuomet kai tinkle

yra maksimalus leidžiamas mazgų skaičius. Maršrutų parinkimas Pastry tinkle atliekamas panašiai kaip IP paketų - iš maršrutų lentelių sekantis mazgas parenkamas taip, kad kuo ilgesnė ieškomo adreso dalis sutaptu su mazgo identifikatoriumi. Tokiame tinkle kelio ilgis iki laisvai pasirinkto adreso visada bus trumpesnis arba lygus $\log_b N = n$, nes kiekviename žingsnyje žinutė bus siunčiama mazgui, kurio adresas su paieškos adresu sutaps vienu skaitmeniu daugiau. Kitas kelio ilgio formulės paaiškinimas būtų toks, kad kiekviename žingsnyje bendra adresų erdvė, kurioje gali būti ieškomas adresas dalinama į b dalių ir paieška toliau tęsiama vienoje iš tų dalių. Grafiškai maršrutų lentelėje saugomų mazgu išsidėstymas pateiktas 7 pav. Šiame pavyzdyje parodytas tinklas, kur b=4; n=3 ir pateiktos mazgo x=112 maršrutų lentelės.



j	adresas
0	000
1	100
2	nėra
3	300

j	adresas
0	100
1	nėra
2	120
3	130

j	adresas
0	110
1	111
2	nėra
3	113

7 pav. Grafinis mazgo x=112 kaimynų pavyzdys

Matome, kad Pastry tinklas, kuriame b=2 iš esmės labai panašus į Chord tinklą. Esminis skirtumas tarp Pastry ir Chord yra, tame, kad Pastry modelis maršrutų lentelėje leidžia laisvai pasirinkti mazgą iš leidžiamų mazgų intervalo, o Chord modelyje ieškomo mazgo numeris išskaičiuojamos nuo nagrinėjamo mazgo numerio. Tačiau lyginant paieškos kelio ilgį galime laikyti, kad Pastry tinklas yra apibendrintas $\log_b(N)$ paieškos algoritmų modelis leidžiantis laisvai pasirinkti logaritmo bazę b. Logaritmo bazės parinkimas laidžia pasirinkti santyki tarp saugomų briaunų skaičiaus ir paieškos kelio. Pavyzdžiui jei turime $b = 2$; $n = 16$; $K = b^n = 65536$ adresų erdvę gauname, kad kiekvienas mazgas turės $b \cdot n = 32$ įrašus maršrutų lentelėje, o paieškos kelio ilgis bus lygus n=16 žingsnių. Adresų erdvę padalinus žymint šešioliktaine kodavimo sistema $b = 16$; $n = 4$; $K = b^n = 65536$, gauname $b \cdot n = 64$ maršrutų lentelės dydi, tačiau paieškos kelio ilgis n=4. Tokiu būdu nekeičiant adresų erdvės dydžio, o keičiant tik adresų kodavimo sistemą naudojant Pastry DHT modelį galime keisti mazgų apkrovimo ir kelio paieškos ilgio santyki.

2.3.4 Kademlia modelis

Kademlia DHT modelis [12] yra vienas plačiausiai naudojamų DHT modelių P2P failų apsikeitimo sistemose. Jis naudojamas eMule, BitTorrent, Overnet, Gnutella failų apsikeitimo programose, taip pat paskirstytų web svetainių talpinimo projekte Osiris. Kademlia modelyje duomenų ir mazgų erdvė yra vienmatė ir uždara dvejetainių reikšmių aibė. Modelio autoriai siūlo naudoti 160 bitų ilgio adresų erdvę, nes būtent tokio ilgio yra SHA-1 kriptografinės maišos funkcijos rezultato reikšmė. Esminis Kademlia adresų erdvės skirtumas lyginant su kitais modeliais yra XOR adresų atstumo metrika. Pastry it Tapestry modeliuose atstumas *d(a,b)* tarp dviejų adresų erdvės taškų a ir b buvo laikomas šių reikšmių absoliutiniu skirtumu $d_P(a,b) = |b - a|$, Chord modelyje atstumas buvo matuojamas tik einant pagal laikrodžio rodyklę $d_C(a,b) = (b - a + N)\ mod\ N$. Kademlia tinklo modelyje atstumas tarp erdvės taškų yra šių reikšmių XOR operacijos rezultatas $d_K(a,b) = a \oplus b$. XOR operacija tenkina visas tris atstumo metrikos sąlygas:

1. $d_K(x,x) = 0;\ d_K(x,y) > 0, kai\ x \neq y$
2. $d_K(x,y) = d_K(y,x)$
3. $d_K(x,y) + d_K(y,z) \geq d_K(x,z)$

Kademlia tinklo briaunos ir maršrutų lentelė suformuojama panašiai, kaip ir Pastry modelyje. Suformuojama tiek maršrutų lentelių kiek adrese yra bitų ir kiekvienoje lentelėje saugomi mazgų adresai, kurių *i*=0,1, ...,*n*-1 bitų sutampa su lentelę saugančio mazgo adresu, o i+1 bitas nesutampa, kur *n* yra adreso ilgis bitais. Nuo Pastry šis modelis skiriasi tuo, kad lentelės viduje adresų skaičius yra nurodomas, kaip bendras tinklo parametras *k* (autorių rekomenduojama *k*=20). Pavyzdžiui, jei analizuojamo mazgo adresas yra `1100101`, tuomet pirmoji maršrutų lentelė saugos adresus `0******` (mazgai su adresais `1******` nesaugomi, nes jie papultų į antrąją arba tolimesnes lenteles), antroji `10*****` trečioji `111****` Adresų lentelės pavyzdys pateikiamas 8 pav.

i=0

Nr.	Adreso šablonas
1	0******
2	0******
...	0******
20	0******

i=1

Nr.	Adreso šablonas
1	10*****
2	10*****
...	10*****
20	10*****

i=2

Nr.	Adreso šablonas
1	111****
2	111****
...	111****
16	111****

8 pav. Mazgo x=1100101; k=20 Kademlia tinkle maršrutų lentelės pavyzdys

Iš pateikto pavyzdžio matome, kad paskutinės lentelės *i*=2,3,4,5 negalės turėti po 20 skirtingų įrašų, tuomet lentelės paliekamos trumpesnės. Aukščiau pateiktas pavyzdys leidžia

pasiekti $O(\log_2 M)$ paieškos efektyvumą, kur M yra dalyvaujančių tinkle mazgų skaičius. Kademlia maršrutų lentelių sudarymą galima apibendrinti ir $O(\log_b M)$, kur $b = 2^a$ atvejui visai kaip Pastry modelyje. Tam reikia maršrutų lenteles sudarinėti adresą dalinant ne po vieną bitą, o po a bitų.

Kademlia tinklo modelyje XOR atstumo metrika suteikia unikalią savybę - mazgai ieškantys to paties duomenų adreso paiešką tikėtina atliks per tuos pačius tarpinius mazgus. Panaudojus šią savybę ir tarpiniai mazgai gali kešuoti duomenis ir mazgų adresus taip paspartinti populiarių duomenų paiešką ir greičiau užpildyti maršrutų lenteles. Pats adresų palyginimas panaudojant XOR metriką atliekamas tik tuomet kai pasirenkama, kuriam iš k mazgų vienoje maršrutų lentelėje toliau siųsti užklausą. Todėl tinkamos maršrutų lentelės parinkimas garantuotai sumažina ieškomą adresų erdvę pusiau arba daugiau. Naudojant įprastą atstumo metrika toliau būtų parenkamas mazgas kurio identifikatorius su ieškomu adresu turi mažiausia skirtumą, naudojant XOR metriką būtų parenkamas toks mazgas, kurio kuo didesnis bitų skaičius (esantis už nesutampančio bito) sutampa su ieškomu adresu. Toks parinkimas praverčia, nes tikėtina, kad į šį mazgą kreipsis ir kiti to paties adreso ieškantys mazgai.

2.4 DHT tinklo atakos

DHT tinklas yra P2P sistema, nes jame visi mazgai yra lygiaverčiai ir atlieka tokias pačias funkcijas bei turi vienodas teises į resursus. Todėl DHT tinkle visi mazgai turi vienodą įtaką tinklo veikimui. Konkretūs DHT taikymai gali turėti ir kliento - serverio architektūros elementų leidžiančių įvesti centralizuotą kontrolę ir apskaitą. Pavyzdžiui paskirstyta duomenų bazė gali naudoti DHT tinklą duomenų paskirstymui, tačiau turėti centrinį autentifikavimo ir valdymo serverį. Tokia aplikacija turi ir P2P ir kliento-serverio komponentus. Šiame darbe toliau bus nagrinėjamos tik P2P sistemos, kuriuose visi mazgai yra lygiaverčiai.

Remiantis [14] DHT tinklo atakas galima suskirstyti į tris bendrines kategorijas:

1. Atakos prieš DHT tinklo naudotojų privatumą
2. Atakos pasinaudojančios DHT tinklu
3. Atakos prieš DHT tinklą

2.4.1 DHT tinklo atakų klasifikacija

Atakos prieš DHT tinklo naudotojų privatumą. Tai atakų grupė, kurios tikslas panaudojant DHT tinklą išgauti papildomą informaciją apie tinklo naudotojus, kurios tinklo naudotojai nenorėtų atskleisti. Atakų prieš naudotojų privatumą pavyzdžiai:

- Pilno mazgų priklausančių DHT tinklui sąrašo gavimas.

- Mazgų ieškančių, turinčių ar platinančių tam tikrą informaciją radimas.
- Konkretaus mazgo saugomos informacijos išgavimas.
- Tam tikros informacijos į tinklą įkėlusio mazgo radimas.
- Vartotojų susiejimo su DHT tinklo mazgais išgavimas.

Taip pat atakoms prieš DHT tinklo naudotojus priskiriamos ir socialinės inžinerijos atakos.

Atakos pasinaudojančios DHT tinklu. Tai atakų grupė, kurių atlikimui reikalingas veikiantis DHT tinklas, tačiau atakų tikslas nėra DHT tinklo darbo sutrikdymas. Tokių atakų pavyzdžiai:

- Virusų platinimas pasinaudojant DHT tinklu.
- DHT tinklo panaudojimas netiesioginio komunikacijų kanalo sudarymui (pvz. apkrėstų kompiuterių valdymui).
- DHT tinklo mazgų resursų panaudojimas siekiant sutrikdyti kitų serverių darbą.

Atakos prieš DHT tinklą. Atakos prieš DHT tinklą, tai atakos kurių tikslas sutrikdyti DHT tinklo darbą. Šios grupės atakas galima apibendrintai vadinti DoS (angl. Denial-of-Service) atakomis. DoS atakų pavyzdžiai:

- DoS ataka prieš DHT tinklo mazgą. Mazgas nebegali pasinaudoti DHT tinklu arba išnaudojami mazgo resursai taip kad mazgas nebegali atlikti ir kitų su DHT tinklu nesusiejusiu darbų. Ši ataka vadinamos Eclipse ataka [15].
- DoS ataka prieš DHT tinkle esančius duomenis. Duomenys priverstinai pašalinami iš DHT tinklo ir arba sudaroma situacija kai tam tikrų duomenų negalima įkelti į DHT tinklą.
- DoS ataka prieš visą DHT tinklą. Sutrikdomas tinklo darbas, niekas nebegali pasinaudoti tinklu.

2.4.2 Sybil atakų grupė

Literatūroje taip pat dažnai sutinkamas terminas Sybil ataka. Sybil atakos, tai atakų grupė, kuri apibūdina ne atakos tikslą ar rezultatą, bet atlikimo pobūdį. Pagrindinis Sybil atakų bruožas, tai dirbtinis didelio valdomų tinklo mazgų skaičiaus sukūrimas. Terminas šiai atakų klasei pasiūlytas [16]. Straipsnio autoriai teigia, kad bet kuri P2P sistema kurioje nauji vartotojai nėra sertifikuojami daugiau ar mažiau gali būti paveikiama Sybil atakos.

Kadangi DHT tinkle kiekvienas mazgas bendrai tinklo veiklai turi vienodą įtaką gauname, kad tinkle susidedančiame iš N mazgų vieno mazgo įtaka ir potencialiai galima padaryti žala tinklui yra 1/N. Idealiu atveju kritiniai sprendimai tinkle galėtų būti priimami daugumos balsavimu, tačiau net ir tokiu atveju jei puolėjas turės pakankamus resursus sukurti

dirbtiniems (N/2+1) mazgui jis galės kontroliuoti tinklą. Net ir pilnai neanalizuojant P2P sistemos veikimo matome, kad paskirstytuose tinkluose bet koks didesnės įtakos gavimas remiasi dirbtinų mazgų į tinklą atvedimu. Praktinėse DHT realizacijose dažnai pakanka žymiai mažiau resursu įprastoms atakoms atlikti. Šiame darbe siekiama nustatyti, kaip DHT tinklo atsparumą Sybil atakoms įtakoja galimybė specialiu būdu parinkti mazgų identifikatorius.

2.4.3 Laisvai pasirenkamų identifikatorių problemos hipotezė

Visi aprašyti teoriniai DHT tinklo modeliai nurodo, kad mazgų ir duomenų identifikatoriai turi būti tolygiai pasiskirstę adresų erdvėje.

Atsitiktiniai duomenų identifikatoriai paprastai gaunami pridedant reikalavimą, kad duomenų identifikatorius turi pačių duomenų kriptografinės maišos funkcijos reikšmė. Tokį reikalavimą gali patikrinti mazgas kurio segmente turi būti išsaugoti duomenys prieš juos išsaugodamas ir atsisakyti juos saugoti, jei reikalavimas bus netenkinamas.

Užtikrinti, kad mazgų identifikatoriais bus generuojami atsitiktinai nėra paprasta. Kademlia ir Chord DHT teoriniuose modeliuose siūloma naudoti mazgo IP adreso maišos funkcijos reikšmę, kiti modeliai nurodo, kad identifikatorių turi atsitiktinai parinkti klientinė programinė įranga. DHT tinklas įgavus didelį populiarumą buvo atrastą nemažai saugumo spragų, atsitiktinių identifikatorių užtikrinimo problema aprašyta [17]. Praktiškesnis mazgų identifikatorių generavimo užtikrinimo metodas ir kiti saugumo pataisymai pasiūlyti [18] ir [19], straipsniuose siūloma mazgo identifikatorius susieti su privačiais ir viešais raktais, tačiau tai smarkiai mažina bendrą tinklo našumą.

Keliama hipotezė, kad praktinės DHT tinklų realizacijos neužtikrina, kad mazgų identifikatoriai tikrai bus generuojami atsitiktinai ir leidžia juos laisvai pasirinkti.

3 TIKSLAI IR UŽDAVINIAI

Atlikus DHT taikymo sričių, tinklo modelių atakų apžvalgą sudarytas darbo tikslas - ištirti dėl galimybės laisvai pasirinkti mazgo identifikatorių DHT tinkluose kylančias problemas ir pasiūlyti galimus problemos sprendimo būdus.

Šiam tikslui pasiekti bus atliekami šie uždaviniai

1. Įvertinti DHT tinklo atsparumo Eclipse atakai skirtumą kai mazgai gali arba negali laisvai pasirinkti identifikatorių.
2. Išanalizuoti BitTorrent DHT tinklą, nustatyti galimas problemas ir įvertinti jų apimtį.

3. Išanalizuoti galimas problemas praktiškai pritaikant privalomą mazgų identifikatorių generavimą.
4. Pasiūlyti metodą, kaip efektyviai realizuoti identifikatorių generavimo įrodymą ir apsaugoti identifikatorius nuo kopijavimo
5. Pasiūlyti metodą, kaip apsaugoti mažus DHT tinklus, kuriuose net atsitiktinai generuojant identifikatorius atakai atlikti nereikia didelių resursų.

4 DHT TINKLO MODELIŲ ATSPARUMO ECLIPSE ATAKAI ANALIZĖ

Eclipse ataka aprašyta [15] yra taikoma tokiems DHT tinklams, kuriuose mazgai identifikatorius privalomai generuoja atsitiktinai ir nenagrinėjamas atvejis, kai konkreti DHT tinklo realizacija leidžia mazgams laisvai pasirinkti savo identifikatorių. Tačiau praktikoje naudojamuose eMule ir BitTorrent DHT tinkluose mazgai savo identifikatorių gali pasirinkti laisvai. Šiame skyriuje įvertinamas Eclipse atakos sudėtingumo skirtumas, kai mazgai gali ir negali laisvai pasirinkti identifikatorių Chord ir Kademlia tinklo modeliuose. CAN ir Tapestry DHT modeliai nenagrinėjami, nes jie rečiau naudojami praktikoje.

Eclipse ataka, tai ataka prieš vieną DHT tinklo mazgą siekiant atskirti mazgą nuo likusio DHT tinklo. Kiekvienas mazgas turi maršrutų lentelę ar lenteles, kuriose saugojama informacija apie kitus tinklo mazgus. Kai mazgas ieško informacijos paieška pradedama komunikuojant su maršrutų lentelėje esančiais mazgais, todėl kenkėjiškas vartotojas gali atlikti Eclipse ataką prieš mazgą į tinklą įvesdamas kenkėjiškus mazgus kurie užimtų visus maršrutų lentelės įrašus. Pilnai užėmus puolamo mazgo maršrutų lenteles kenkėjiški mazgai gali nustoti atsakinėti į mazgo užklausas ir taip puolamas mazgas liks atskirtas nuo DHT tinklo.

Eclipse ataką taip pat galima atlikti ir prieš DHT tinkle esančius duomenis arba adresų erdvės segmentą. Tokios atakos atlikimui reikia įvesti tiek kenkėjiškų mazgų, kad visos duomenų bloko kopijos arba adresų erdvės segmentas priklausytų tik kenkėjiškiems mazgams. Įvedamų kenkėjiškų mazgų skaičius randamas iš duomenų replikavimo skaičiaus. Tuomet vienu metu išjungus visus kenkėjiškus mazgus puolami duomenys būtų pašalinami iš tinklo.

Abi atakas galima išskaidyti į du etapus:

1. Atakai atlikti reikiamo kenkėjiškų mazgų skaičiaus įvertinimo.
2. Atakai tinkamų kenkėjiškų mazgų identifikatorių parinkimo.

Kai DHT tinklas leidžia laisvai pasirinkti mazgo identifikatorių, tuomet Eclipse atakai atlikti reikalingas resursų kiekis yra tiktai pirmame žingsnyje rasto mazgų skaičiaus

sukūrimas ir įvedimas į tinklą. Kai identifikatoriaus privaloma generuoti atsitiktinai dalis resursų sunaudojama tinkamų identifikatorių generavimui.

4.1 Vieno identifikatoriaus parinkimo sudėtingumas

P2P sistemose, kuriose yra mechanizmas, leidžiantis patikrinti ar mazgo identifikatorius tikrai yra sugeneruotas atsitiktinai, kenkėjiškas vartotojas, norintis gauti vieną konkretų identifikatorių, turi generuoti naujus identifikatorius tol kol bus gautas ieškomas identifikatorius. Tinkle, kurio adresų erdvė sudaryta iš n bitų ilgio identifikatorių, tikimybė gauti norimą identifikatorių (įvykis A) apskaičiuojama pagal formulę (1).

$$P(A) = \frac{1}{2^n} \qquad (1)$$

Chord [9] ir Kademlia [12] DHT modeliuose naudojami n=160 bitų ilgio identifikatoriai. Tikimybė atsitiktinai sugeneruoti norima identifikatorių šio modelio tinkluose yra $2^{-160} \approx 6.84 \cdot 10^{-49}$. Tačiau DHT modeliuose mazgo saugomi duomenų blokai ir ryšiai su kitais mazgais nepriklauso nuo absoliučios mazgo identifikatoriaus reikšmės, o priklauso nuo mazgo santykinės pozicijos tinke (kas yra mazgo kaimynai ir kokiame adresų erdvės segmente yra mazgas). Todėl kenkėjiškam vartotojui nebūtina turėti konkretų identifikatorių o pakanka patekti į norimą adresų erdvės segmentą. Atsitiktinius identifikatorius turintys mazgai tolygiai pasiskirsto visoje DHT adresų erdvėje, todėl tinklas, kuriame yra M mazgų, bendrą adresų erdvę padaliną į M panašaus dydžio segmentų. Tikimybė naujam mazgui patekti į norimą tinklo adresų segmentą (įvykis B) yra apskaičiuojama pagal formulę (2).

$$P(B) \approx \frac{1}{M} \qquad (2)$$

Tikimybė patekti į norimą adresų erdvės segmentą priklauso tiktai nuo mazgų skaičiaus tinkle. Pavyzdžiui Kademlia tinkle užfiksuoti 3.5 milijonai vartotojų [20]. Tikimybė tokiame tinkle sugeneruoti identifikatorių norimame adresų segmente yra $3500000^{-1} \approx 2.86 \cdot 10^{-7}$.

4.2 Chord modelio analizė, kenkėjiškų mazgų skaičius

Chord DHT modelyje kiekvienas mazgas turi maršrutų lentelę sudarytą iš n įrašų. Mazgo su identifikatoriumi x maršrutų lentelės i-tojoje eilutėje saugomas mazgo atsakingo už adresų erdvės tašką randamą pagal $\left(x + 2^i\right) mod\ N; i = 0,1, \dots, n-1$ formulę IP adresas. Eclipse atakos atlikimui prieš mazgą x, į tinklą pakanka įvesti n kenkėjiškų mazgų su atitinkamais identifikatoriais.

Ataką galima supaprastinti atsižvelgiant į tai, kad adresų erdvė tinkle niekada nebūna pilnai užpildyta mazgų identifikatoriais ir dėl to maršrutų lentelėse nebūna užpildytos visos n

eilutės. Tinklo kuriame yra *M* mazgų adresų erdvė yra padalina į *M* panašaus dydžio dalių ir vidutinis atstumas tarp dviejų mazgų yra $\approx \frac{N}{M}$. Mazgo *x* maršrutų lentelės *i*-tojoje eilutėje saugoma mazgo nutolusio nuo *x* per 2^i vienetus informacija. Gauname, maršrutų lentelės įrašai i tenkinantys nelygybę $x + 2^i < x + \frac{N}{M}$ rodys į tuščią erdvę esančią tarp mazgo *x* jo kaimyno. Pertvarkius nelygybę gauname $i < n - \log_2(M)$, todėl iš visų *n* maršrutų lentelės pirmi $n - \log_2(M)$ įrašai bus tušti, o likę $\log_2(M)$ užpildyti.

Eclipse atakai Chord tinkle atlikti reikia vidutiniškai (3) formule paskaičiuojamo kenkėjiškų mazgų skaičiaus.

$$m = \log_2(M) \qquad (3)$$

Reikia atkreipti dėmesį, kad *M* savo skaitine reikšme visada yra mažesnis arba lygus *N*, nes tinkle negali būti daugiau mazgų nei egzistuoja skirtingų identifikatorių. Kadangi $\log_2(N) = n$, o *n* yra konstanta (adresų erdvės ilgis bitais), galima teigti kad atakai pakanka fiksuoto kenkėjiškų mazgų skaičiaus ir atakos sudėtingumas yra *O*(1).

4.3 Chord modelio analizė, identifikatorių generavimo operacijų skaičius

Neturint galimybės laisvai pasirinkti mazgo identifikatorių, ataka tampa sudėtingesnė, nes reikia generuoti kenkėjiškų mazgų identifikatorius, tol kol bus rasti atakai tinkantys identifikatoriai. Tarkime ataka atliekama prieš mazgą su identifikatoriumi *x*, esantį tinkle, kuriame yra *M* mazgų. Tuomet visa adresų erdvė yra sudalinta į *M* segmentų. Iš visų *M* segmentų *m* segmentuose yra y_i identifikatoriai tinkantys atakai ir rasti pagal formulę $y_i = \left(x + 2^i\right) mod\ N; i = 0,1, \dots, m - 1$. Kenkėjiškų mazgų identifikatoriai generuojami tol, kol sugeneruotas identifikatorius patenka į vieną iš reikalingų segmentų *i* ir skaitine reikšme yra didesnis arba lygus y_i. Toks identifikatorius yra tinkamas atakai. Teisingai veikiančių mazgų identifikatoriai adresų erdvėje yra pasiskirstę tolygiai, todėl identifikatorius y_i segmento *i* viduje su vienoda tikimybe gali būti bet kurioje pozicijoje. Tikimybė, kad segmente *i* rastas identifikatorius yra tinkamas (didesnis arba lygus y_i) yra tokia pati kaip ir tikimybė, kad jis netinkamas ir lygi $\frac{1}{2}$. Remiantis (2) formule gauname, kad vienam tinkamam kenkėjiškam identifikatoriui rasti reikės vidutiniškai perrinkti 2*M* skirtingų identifikatorių. Pilnai atakai reikės *m* tokių identifikatorių, ir jiems rasti reikės (4) formule randamo identifikatorių generavimo operacijų.

$$\sum_{i=0}^{\log_2(M)-1} \frac{2M}{\log_2(M) - i} \qquad (4)$$

Gaunama, kad negalint laisvai pasirinkti identifikatoriaus reikia atlikti vidutiniškai *O(M)* identifikatoriaus generavimo operacijų.

4.4 Kademlia modelio analizė, kenkėjiškų mazgų skaičius

Kademlia tinkle kiekvienas mazgas turi *n* (adresų erdvės ilgis bitais) maršrutų lentelių. Mazgo su identifikatoriumi *x* maršrutų lentelėse $RT_i; i = 0,1, \dots, n-1$ saugomi IP adresai ir identifikatoriai tų mazgų, kurių identifikatorių pirmi *i* bitai sutampa su mazgo identifikatoriumi *x*, tačiau *i*+1 bitas nesutampa. Kiekvienoje lentelėje saugoma *k* (bendras tinklo duomenų replikavimo parametras) įrašų. Pilnai užpildytos mazgo maršrutų lentelės turi $k \cdot n$ įrašų. Visa galimų adresų erdvė yra pakankamai didelė ir nebūna pilnai užpildyta, todėl tolygiai atsitiktinai pasiskirstę mazgų identifikatoriai taip pat kaip Chord modelyje vidutiniškai užpildo tik pirmas $\log_2(M)$ mazgų maršrutų lenteles, kur *M* yra bendras tinkle veikiančių mazgų skaičius. Eclipse atakai atlikti Kademlia tinkle reikia pagal (5) formulę randamo kenkėjiškų mazgų skaičiaus.

$$k \cdot \log_2(M) \qquad (5)$$

Kadangi $\log_2(M) \leq n$, o *n* yra konstanta, tai atakai pakanka fiksuoto kenkėjiškų mazgų skaičiaus ir atakos sudėtingumas yra O(1).

4.5 Kademlia modelio analizė, identifikatorių generavimo operacijų skaičius

Neturint galimybės laisvai pasirinkti identifikatorių, tam, kad kenkėjiškas mazgas *y* patektų į mazgo *x* maršrutų lentelę RT_i, reikia, kad pirmieji *y* identifikatoriaus *i* bitai sutaptu su *x* ir *i*+1 bitas nesutaptų. Remiantis (1) formule gauname, kad tokį identifikatorių galima vidutiniškai sugeneruoti atliekant 2^{i+1} generavimo operacijų. Visoms *n* maršrutų lentelėms tinkamus identifikatorius galima rasti atlikus (6) formulėje pateiktą identifikatorių generavimo operacijų skaičių. Formulėje (6) neįvertinama, kad paskutinėse maršrutų lentelėse negali būti *k* įrašų, nes nėra tiek galimų identifikatorių kombinacijų, tačiau tai sudaro nežymią paklaidą, nes tokių lentelių gali būti ne daugiau kaip $\log_2(k)$, kai $k = 10;\ \log_2(k) < 4$.

$$k \sum_{i=0}^{n-1} 2^{i+1} = 2k \sum_{i=0}^{n-1} 2^i = 2k(2^n - 1) \qquad (6)$$

Įvertinus, kad mazguose vidutiniškai būna užpildytos tiktai $\log_2(M)$ maršrutų lentelės gauname formulę (7). Šioje formulėje nebelieka paklaidos jei $n - \log_2(M) > \log_2(k)$.

$$2k\left(2^{\log_2(M)} - 1\right) = 2k(M - 1) \qquad (7)$$

Kademlia tinklo modeliui atliekamos Eclipse atakos identifikatorių generavimo sudėtingumas taip pat kaip ir Chord modelyje yra *O(M)*.

Pilnam Eclipse atakos realizavimui Kademlia tinkle reikiamų identifikatorių sugeneruoti nepakanka. Kademlia modelyje maršrutų lentelės pildomos atliekant bet kokį veiksmą su kitu mazgu (mazgo paiešką, duomenų paiešką, mazgo pasiekiamumo tikrinimą), taip pat mazgai maršrutų lenteles rikiuoja pagal tai, kiek laiko be pertraukimo mazgai buvo pasiekiami. Todėl lentelių viršuje visada būna tie mazgai, kurie ilgiausiai dirbo be pertraukos. Sėkmingai atakai reikia, kad kenkėjiški mazgai veiktų ilgiau ir patikimiau nei visi kiti teisingai veikiantys mazgai pretenduojantys į mazgo x maršrutų lenteles. Tai sunkiausia pasiekti mazgams, kurie turi patekti į RT_0 maršrutų lentelę, nes į šią lentelę patekti pretenduoja pusė visų tinkle veikiančių mazgų.

4.6 Išvados

Apibendrinus gautus rezultatus, galima teigti, kad turint galimybę laisvai pasirinkti mazgo identifikatorių Eclipse ataką prieš abu DHT modelius galima atlikti su fiksuotu kenkėjiškų mazgų skaičiumi, randamu pagal formules (3) ir (5). Kai identifikatorius privaloma generuoti atsitiktinai kenkėjiškų mazgų skaičius atakai atlikti yra toks pat, tačiau kenkėjiškų mazgų identifikatoriams rasti reikia atlikti nuo tinkle esančių vartotojų skaičiaus *M* priklausantį identifikatorių generavimo operacijų skaičių, kuris randamas pagal formules (4) ir (7).

Atitinkamai atakų sudėtingumą galima pažymėti *O*(1), kai identifikatoriai pasirenkami laisvai ir *O(M)*, kai identifikatorius privaloma generuoti atsitiktinai.

5 BITTRORRENT DHT TINKLO ANALIZĖ

BitTorrent DHT tinklas yra vienas plačiausiai žinomų konkrečių Kademlia DHT modelio pritaikymų. Atliktos analizės tikslas yra apytiksliai nustatyti BitTorrent DHT tinklo dydi (prie tinklo prisijungusių mazgų skaičių), išanalizuoti rastas tinklo anomalijas, patikrinti ar galimybė laisvai pasirinkti mazgo identifikatorių yra naudojama kenkėjiškiems tikslams.

Įprastu rėžimu veikiantis BitTorrent klientas duomenų siuntimą koordinuoja su specialiu tracker serveriu (nuo angliško termino “to track” - sekti, stebėti), kurio pagrindinis tikslas yra platinti informaciją kas šiuo metu siunčiasi prižiūrimą duomenų bloką. BitTorrent DHT tinklas yra naudojamas failų apsikeitimui be dedikuoto tracker serverio. Kiekvienas mazgas ir duomenų blokas turi 160 bitų ilgio identifikatorių. Duomenų blokams identifikatorius randamas apskaičiuojant duomenų meta informacijos SHA-1 kriptografinę maišos funkcijos reikšmę, mazgų identifikatoriai pasirenkami laisvai. Bet kuris BitTorrent DHT tinkle

dalyvaujantis mazgas gali atlikti tracker serverio funkciją, o DHT tinklas suformuojamas tam, kad mazgai pasiskirstytų kuriems duomenų blokams jie turi veikti kaip tracker serveris.

Tinklo analizė atlikta remiantis [21] sukūrus specializuotą klientinę programinę įrangą, kuri ieško ir registruoja į vidinę duomenų bazę BitTorrent DHT tinklui priklausančius mazgus. Informacija apie tinklą buvo surinkta pasinaudojant Kademlia protokolo RPC (angl. Remote Procedure Call) užklausą FIND_NODE. FIND_NODE užklausoje pateikiamas ieškomas identifikatorius, o mazgas atsakantis į užklausą iš maršrutų lentelių išrenka k arčiausiai ieškomo identifikatoriaus esančius įrašus ir išsiunčia užklausėjui. Mazgui x pateikus FIND_NODE užklausą, kurios argumentas būtų x mazgas atsako k arčiausiai jo esančių jam žinomų mazgų informaciją. Iteratyviai siunčiant tokias užklausas kiekvienam naujai pamatytam mazgui ratu apeinama ir visa DHT adresų erdvė.

Atliekant eksperimentą buvo pastebėta, kad BitTorrent DHT tinklas yra labai dinamiškas ir tinklo mazgų kaita viršija tinklo apėjimo greitį. Paieška buvo stabdoma atlikus 10 tinklo apėjimo iteracijų. Tokia riba buvo parinkta atsižvelgiant į eksperimentui skirtą darbinės stoties operatyvinės atminties resursų kiekį.

Surinktos informacijos apimtis pateikiama 1 lentelėje.

1 lentelė. BitTorrent DHT tinklo mazgų apėjimo rezultatai

Įrašų skaičius, (id, IP, port) unikalios poros	28 144 165
Unikalių identifikatorių skaičius	18 969 645
Unikalių IP adresų skaičius	17 368 323
Į užklausą atsakiusių mazgų skaičius	5 586 466
Tinkle veikiančių mazgų įvertis (remiantis [22])	~16 500 000

Gauta, kad skenavimo metu užmegzti abipusį ryšį pavyko su daugiau nei 5,6 milijono mazgų. Norint įvertinti kiek iš viso skenavimo metu DHT tinkle buvo veikiančių mazgų remiantis [22] gautą aktyvių mazgų skaičių reikėtų dauginti iš 3. Straipsnio autorių atliktas tyrimas parodė, kad dėl ugniasienių ir NAT įrenginių tiktai $\frac{1}{3}$ visų tinklo mazgų yra pasiekiami ir atsako į užklausas.

Surinktuose duomenyse buvo rastos tokios tinklo anomalijos:

1. Ne atsitiktinai parinkti identifikatoriai.
2. Ne unikalūs identifikatoriai (identifikatoriai, kuriuos naudoja daugiau nei vienas mazgas).
3. Atliekamų Eclipse atakų požymiai.

Gauti rezultatai įrodo, kad iškelta hipotezė buvo teisinga ir galimybė laisvai pasirinkti identifikatorių yra naudojama atakoms atlikti.

5.1 Ne atsitiktinai parinktų identifikatorių dalies tinke įvertinimas

Surinktuose duomenyse buvo atlikta akivaizdžiai specialiai parinktų identifikatorių paieška. Kadangi identifikatoriai yra 160 bitų, 20 baitų ilgio juose ASCII (angl. American Standard Code for Information Interchange) koduote galima saugoti 20 simbolių eilutę. Atlikus paiešką identifikatorių, kurių visi 20 baitų yra atvaizduojami ASCII simboliai (simbolių kodai nuo 0x20 iki 0x7e) rasti 138 skirtingi identifikatoriai. Rastų identifikatorių pavyzdžiai pateikti 2 lentelėje.

2 lentelė. Rastų specialiai pagal ASCII kodus parinktų identifikatorių pavyzdžiai

Nr.	Identifikatorius šešioliktainiu kodu	ASCII atitikmuo
1	3031323334353637383930313233343536373839 5758595a636465666768696a737475767778797a 676f75676f7520646874206e6176696761746f72 686967687363686f6f6c656e67696e6563726131	01234567890123456789 WXYZcdefghijstuvwxyz gougou dht navigator highschoolenginecra1
2	454d42303030353945384535373433585858797 1 4a4353333538452d312e3032702d303038616630 535452383138332d312e3632612d303033353165 2d5753303130302d313233343536373839303132	EMB00059E8E5743XXXyq JCS358E-1.02p-008af0 STR8183-1.62a-00351e -WS0100-123456789012
3	3c3f786d6c2076657273696f6e3d22312e302220 3c21444f43545950452068746d6c205055424c49 3b673d682e676574456c656d656e747342795461 617574686f7269643d36373937223e323c2f613e	<?xml version="1.0" <!DOCTYPE html PUBLI ;g=h.getElementsByTa authorid=6797">2

Kadangi skirtingų atvaizduojamų ASCII simbolių yra 138, tai atsitiktinai generuojant vieno baito reikšmę tikimybė sugeneruoti ASCII simbolį yra $\frac{94}{256} = 0.3719$, tikimybė atsitiktinai sugeneruoti 20 simbolių ASCII eilutę yra $\left(\frac{94}{256}\right)^{20} = 1.985 \cdot 10^{-9}$, apytiksliai vienas iš 500 milijonų identifikatorių. Iš pavyzdžių matoma, kad identifikatoriai tikrai nebuvo generuoti, nes rastos simbolių eilutės turi aiškiai matoma struktūrą. Eilutėje 1 pateikti identifikatorių pavyzdžiai tikriausiai naudojami panašioje tinklo DHT tinklo analizės programinėje įrangoje, identifikatoriai eilutėje 2 galėjo būti parinkti formuojant identifikatorių pagal programinės įrangos versiją arba įrenginio serijinį numerį. Identifikatorių pavyzdžiai 3 eilutėje panašus į programinės įrangos klaidą, nes susideda iš kodo nurodančio XML arba HTML dokumento ištrauką.

Kitas ne specialiai parinktų identifikatorių radimo požymis buvo vienodų simbolių sekos paieška identifikatoriuje. Tikimybė atsitiktinai sugeneruoti vieną konkrečią *i* bitų seka

pasikartojančią k kartų yra $\left(\frac{1}{2^i}\right)^k = 2^{-i \cdot k}$, tokia seka n bitų ilgio identifikatoriuje gali būti skirtingose $\frac{n}{i} - k + 1$ pozicijose (stumiant ieškomą seką ne po bitą bet po i bitų), iš viso skirtingų i bitų sekų gali būti 2^i. Gauname, n=160 bitų ilgio identifikatoriuje i=8 bitų ilgio seka k=5 kartus pasikartoti gali su tikimybe $2^{-i \cdot k} \cdot \left(\frac{n}{i} - k + 1\right) \cdot 2^i = 3.7253 \cdot 10^{-9}$, apytiksliai vienas iš 250 milijonų identifikatorių turėtų tenkinti tokias sąlygas. Atlikus vienodų 5 baitų sekų paiešką surinktuose duomenyse buvo rasti 341 173 skirtingi identifikatoriai, tai yra sudaro apie $\frac{341173}{18969645} \approx 1.8\%$ visų užfiksuotų identifikatorių.

Iš rastų 341 173 identifikatorių tik 27 buvo turintys ne `0000000000` seką.

Didelį specialiai pasirinktų identifikatorių kiekį duomenyse galima vaizdžiai įvertinti analizuojant rastų identifikatorių pasiskirstymą. Atsitiktinai generuoti identifikatoriai adresų erdvėje pasiskirsto tolygiai, nes kiekvieno atsitiktinai generuoto baito reikšmė nepriklauso nuo jo pozicijos identifikatoriuje ir su vienoda tikimybe gali įgyti visas reikšmes. 9 pav. pateikti grafikai kuriuose atvaizduojamas rastų identifikatorių pasiskirstymas fiksuojant pirmą, antrą, trečią arba ketvirtą identifikatoriaus baito reikšmę. Pirmajame grafike x ašyje atidėtos visos galimos pirmo identifikatoriaus baito reikšmės (nuo `00` iki `ff`), o y ašyje atidėtas identifikatorių skaičius turintis tokią pirmo baito reikšmę. Likusiuose grafikuose atitinkamai atidedamos visas galimos antro, trečio ir ketvirto baito reikšmės.

9 pav. Identifikatorių pasiskirstymas fiksuojant pirmą, antrą, trečia ir ketvirtą baitus

Pirmame grafike matoma, kad identifikatoriai adresų erdvėje nėra pasiskirstę tolygiai. Tai gali rodyti, kad tinkle dažnai identifikatoriais nėra parenkami atsitiktinai. Netolygus pirmo

baito reikšmės pasiskirstymas taip pat galėjo būti gautas ir dėl duomenis rinkusios programos savybių, nes duomenis rinkusi programa buvo sustabdyta anksčiau nei buvo rasti visi tinkle egzistuojantys identifikatoriai.

Gautuose grafikuose matomi pikai prie reikšmės `90` pirmame baite, `69` antrame baite, `ca` trečiame baite ir `78` ketvirtame baite. Patikrinus duomenis gauta, kad visus šiuos pikus sudaro tie patys 34 398 identifikatoriai prasidedantys seka `9069ca78`.

Ketvirtajame grafike matomas dar vienas didelis pikas ties reikšme `00`, su tokia ketvirto baito reikšme rasti 413 503 identifikatoriai. Ketvirtojo grafiko viršus nukirptas, nes reikšmė ties `00` daugiau nei 3 kartus viršija bendrą grafiko skalę. Iš viso 340 691 identifikatorių atitinka šabloną `******00000000000000000000000000000000` todėl panašaus dydžio pikas kartojasi per visas likusių nuo 5 iki 20-to identifikatorių baitų reikšmes. Taip pat tokio šablono identifikatoriai sudaro didžiąją dalį identifikatorių rastų ieškant pasikartojančios 5 baitų sekos.

Apibendrinus gautus rezultatus matyti, kad iš 18 969 645 unikalių identifikatorių mažiausiai 138 + 27 + 340 691 + 34 398 = 375 254 identifikatoriai buvo parinkti ne atsitiktinai. Tai sudaro ≈ 2% visų tinkle esančių identifikatorių. Platesnė tinklo analizė galėtų padėti tiksliau įvertinti ne atsitiktinai parinktų identifikatorių dalį tinkle. Netolygus identifikatorių pasiskirstymas adresų erdvėje nėra naudingas teisingai veikiantiems mazgams, todėl tokios anomalijos gali atsirasti arba dėl programinės įrangos realizacijos klaidų arba dėl specializuotos atakos prieš DHT tinklą.

5.2 Ne unikalūs identifikatoriai

Surinktoje IP adresų ir identifikatorių duomenų bazėje buvo atlikta paieška tokių identifikatorių, kuriuos naudoja daugiau nei vienas mazgas (mazgas identifikuojamas pagal IP adresą). Paieška buvo atlikta du kartus, vieną kartą ieškant tiktai tarp aktyvių mazgų (tų kurie buvo patikrinti ir atsakė į užklausas), antrą kartą tarp visų užregistruotų mazgų. Penki dažniausiai naudojami identifikatoriai ir juos naudojančių mazgų skaičius pateiktas 3 lentelėje.

3 lentelė. Ne unikalūs identifikatoriai ir juos naudojančių mazgų skaičius

Nr.	Identifikatorius	Aktyvūs mazgai	Rasti mazgai
1	`0000000000000000000000000000000000000000`	`122`	`341`
2	`6e15bedc1676aa10b3db85cbfcdb2a3c05d793aa`	`97`	`160`
3	`190ead05be6bc2e45d19329aa617283442489fc2`	`19`	`39`
4	`8493951e8a9b0fd85ca6c9b513a03a5b77c83cec`	`19`	`29`
5	`c7a5d7c5d14704462beb025f4035acb1fbe85cb0`	`18`	`21`

Skirtingais IP adresas tie patys mazgai gali būti užregistruoti dėl kelių priežasčių - neteisingai veikiančios programinės įrangos, tinklo įrenginiuose atliekamos adresų transliacijos, sąmoningai siekiant pasinaudoti svetimo mazgo identifikatorių.

Dėl programinės įrangos kaltės paskelbti identifikatorių dublikatai, gali būti tiek pavojingi, tiek ir nepavojingi. Jeigu sukurta nauja identifikatoriaus ir IP adreso pora iš tikro yra veikiantis tačiau tokio identifikatoriaus nenaudojantis DHT mazgas, tokia klaida yra pavojinga, nes gali klaidinti kitus paiešką atliekančius mazgus. Jei naujai sukurta identifikatoriaus ir IP pora ištiko neatsako į DHT užklausas dublikatas papildomų problemų tinklui nekelia.

Sąmoningu identifikatoriaus kopijavimo atveju, identifikatorių kopijuojantis mazgas atsako į užklausas ir visais kitais aspektais elgiasi kaip tokį identifikatorių turintis mazgas. Identifikatorių kopijavimo galimybė gali būti panaudota DoS atakoms prieš mazgą arba duomenis atlikti.

Identifikatorių kopijos paskelbtos dėl tinklo įrenginiuose atliekamos adreso transliacijos sukeliamos tuomet, kai užklausos paketui keliaujant tinklu nuo siuntėjo iki gavėjo tarpinis mazgas pakeičia siuntėjo IP adresą. Toks pakeitimas atliekamas DNAT (angl. Destination Network Address Translation) įrenginiuose, proxy serveriuose ir apkrovos išlyginimo (angl. load-balancers) įrenginiuose. Už tokių įrenginių esantis mazgas vietoje tikro siuntėjo adreso mato tarpinio tinklo įrenginio IP adresą ir gautą IP adreso ir identifikatoriaus porą toliau skelbia kitiems DHT tinklo mazgams. Tokie identifikatorių dubliavimo atvejai nėra pavojingi nes tarpiniam tinklo įrenginiui adresuojami tarnybiniai DHT paketai ignoruojami ir tokia IP adreso ir identifikatoriaus pora neatsako į DHT užklausas.

Atliekant BitTorrent DHT tinklo skenavimą buvo gauti rezultatai rodantys tarpinės tinklo įrangos kaltės sukeltą identifikatoriaus dubliavimą. Skenavimą atliekančiam mazgui buvo suteiktas unikalus identifikatorius `6620626974746f7272656e7420637261776c6572`, tačiau skenavimo eigoje buvo rasti dar 9 mazgai turintys tokį patį identifikatorių, rasti mazgai neatsakė į jiems siųstas užklausas. Vieno iš 9 identifikatoriaus dublikatų IP adresas buvo iš privataus IP adresų ruožo 192.168.0.0/16, kitų du buvo globaliai maršrutizuojami IP adresai.

5.3 DoS atakų prieš adresų erdvės segmentą požymiai

Gautuose duomenyse buvo rastos identifikatorių sekos kurios rodo atliekamą DoS ataką prieš DHT tinklo adresų erdvės segmentą. Adresų erdvės segmento užėmimo atakos atlikimui kenkėjiškas vartotojas turi sugeneruoti ne mažiau kaip k=10 (tinklo replikavimo parametras)

kenkėjiškų mazgų tarp kurių identifikatorių ir užimamo adresų erdvės segmento atstumas būtų mažesnis nei bet kurio kito atakoje nedalyvaujančio mazgo.

Atakas atliekantys mazgai ir tinklo segmentai rasti tokiu būdu. Galima daryti prielaidą, kad ataka bus atliekama iš vieno įrenginio, todėl atakos pėdsakams rasti galima analizuoti IP adreso ir jam priklausančių identifikatorių skaičiaus santyki. Iš sukauptos duomenų bazės atrinkti tie IP adresai, kuriems priklauso ne mažiau kaip 10 skirtingų identifikatorių. Į tokio filtro rezultatus taip pat patenka IP adresai, už kurių yra daugiau nei 10 teisingai veikiančių mazgų. Kadangi ataką atliekantys identifikatoriai privalo būti arti vienas kito, galima daryti prielaidą, kad radus daugiau arba lygiai 10 unikalių identifikatorių kurių pirmi 4 baitai sutampa priklausančių vienam IP adresui yra rastas ataką atliekantis vartotojas Tokią prielaidą galime daryti dėl to, kad tikimybė dviem atsitiktinai identifikatorių generuojantiems mazgams pasirinkti identifikatorius, kurių pirmi 4 baitai sutampa, yra nykstamai maža ir lygi $\left(\frac{1}{2^{32}}\right)^2 = 2^{-33} = 1.1642 \cdot 10^{-10}$.

Atlikus paiešką pagal aprašytas atakos atpažinimo taisykles rastas 301 adresų erdvės segmentas, kuriame matomi atakos pėdsakai. Visos atakos buvo atliekamos tik iš 7 skirtingų IP adresų. Iš rastų 301 atakuojamo adresų erdvės segmentų 168 segmentai turėjo lygiai 20 identifikatorių. Galime daryti prielaidą, kad norėdami padidinti sėkmingos atakos tikimybę kenkėjiški vartotojai į tinklą įvedė lygiai 2*k*=20 kenkėjiškų mazgų. Konkrečios atakos prieš mazgą su identifikatoriumi `41558bceeab445f5e5731d90237c73663554375e` perimti pateiktas 4 lentelėje.

4 lentelė. Eclipse atakos prieš identifikatorių `41558bceeab445f5e5731d90237c73663554375e` pavyzdys

Identifikatorius	Mazgo IP adresas ir portas
41558bceeab445f5e5731d90237c73663550**0feb**	74.172.163.108:6882
41558bceeab445f5e5731d90237c73663554**2015**	74.172.163.108:6882
41558bceeab445f5e5731d90237c7366355**4202f**	74.172.163.108:6882
41558bceeab445f5e5731d90237c73663554**2113**	74.172.163.108:6882
41558bceeab445f5e5731d90237c73663554**2205**	74.172.163.108:6882
41558bceeab445f5e5731d90237c73663554**2a9f**	74.172.163.108:6882
41558bceeab445f5e5731d90237c73663554**2b19**	74.172.163.108:6882
41558bceeab445f5e5731d90237c73663554**2fee**	74.172.163.108:6882
41558bceeab445f5e5731d90237c7366355<u>4375e</u>	**71.196.137.156:7881**
41558bceeab445f5e5731d90237c7366355**440c2**	74.172.163.108:6882
41558bceeab445f5e5731d90237c73663554**4407**	74.172.163.108:6882
41558bceeab445f5e5731d90237c7366355**457db**	74.172.163.108:6882
41558bceeab445f5e5731d90237c7366355**4692c**	74.172.163.108:6882
41558bceeab445f5e5731d90237c73663554**7e98**	74.172.163.108:6882
41558bceeab445f5e5731d90237c73663554**1115**	74.172.163.108:6882
41558bceeab445f5e5731d90237c73663554**64d9f**	74.172.163.108:6882
41558bceeab445f5e5731d90237c7366355**699db**	74.172.163.108:6882
41558bceeab445f5e5731d90237c7366355**6a0b1**	74.172.163.108:6882
41558bceeab445f5e5731d90237c7366355**6bf1c**	74.172.163.108:6882
41558bceeab445f5e5731d90237c73663554**78762**	74.172.163.108:6882
41558bceeab445f5e5731d90237c7366355**7ef54**	74.172.163.108:6882

Gauti rezultatai neužtikrina visų atakų prieš adresų erdvės segmentus radimo, tačiau rastas atakų rodo, kad tokios atakos yra aktyviai atliekamos BitTorrent DHT tinkle.

5.4 Išvados

Tyrimo metu BitTorrent DHT tinkle užfiksuoti ~16 milijonų mazgų. Iš visų rastų identifikatorių ne mažiau 2% buvo parinkti specialiai, o ne generuoti atsitiktinai. Rasti identifikatorių kopijavimo atvejai ir atliekamos DoS atakos. Rezultatai patvirtina iškeltą hipotezę, kad praktikoje nesilaikoma nurodymo mazgų identifikatorius parinkti atsitiktinai. Kadangi negalima tikėtis, jog visi mazgai laikysis tinklo specifikacijos nurodymų ir identifikatorius generuos atsitiktinai reikalingi mechanizmai, kurie leistų patikrinti ar identifikatorius tikrai buvo generuotas atsitiktinai.

6 IDENTIFIKATORIAUS GENERAVIMO ĮRODYMAS

Remiantis ketvirto skyriaus rezultatais gauta, kad jei mazgų identifikatoriai privalomai būtų generuojami, o ne pasirenkami atsitiktinai Eclipse atakos atlikimui reikalingas resursų kiekis smarkiai išaugtų ir taptų priklausomas nuo tinkle veikiančių mazgų skaičiaus M. Penktame skyriuje atliktas eksperimentas parodė, kad Eclipse atakos ir specialiai parinkti identifikatoriai yra praktiškai naudojami BitTorrent DHT tinkle. Tam, kad identifikatorių generavimas būtų privalomas kiekvienas mazgas turi turėti galimybę patikrinti kito mazgo identifikatorių ir atitinkamai pats pateikti įrodymą, kad jo identifikatorius yra generuotas atsitiktinai. Šiame skyriuje aprašoma mazgo identifikatorių įrodymo ir apsaugojimo nuo kopijavimo problema.

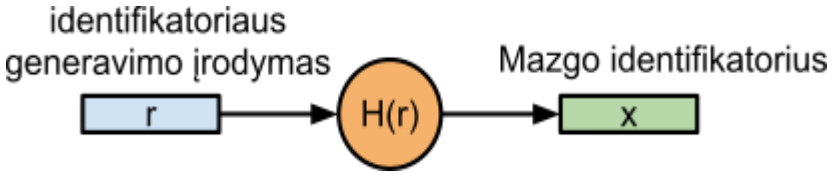
Duomenų identifikatoriaus generavimui Chord [9], Pastry [10], Tapestry [11], Kademlia [12] DHT modeliai rekomenduoja naudoti kriptografines maišos funkcijas. Kriptografinės maišos funkcijos pasižymi tokiomis savybėmis:

- jų reikšmės tolygiai atsitiktinai pasiskirsčiusios reikšmių srityje.
- jos yra vienkryptės - turint duomenis galima lengvai suskaičiuoti maišos funkcijos reikšmę, tačiau turint funkcijos reikšmę rasti pradinius duomenis galima tik perrinkimo būdu.

Dėl pirmosios savybės duomenų identifikatoriai nepriklausomai nuo duomenų turinio pasiskirsto tolygiai visoje adresų erdvėje. Taip pat DHT tinkle didėjant duomenų kiekiui mazgai apkraunami tolygiai. Antroji kriptografinių maišos funkcijų savybė naudinga tuo, kad duomenis įkeliantis mazgas negali pasirinkti kur bus saugomi duomenys. Įkeliant duomenis į tinklą mazgas atsakingas už duomenų saugojimą visada gali suskaičiuoti saugojamų duomenų

kriptografinės maišos funkcijos reikšmę ir atsisakyti saugoti duomenis, jei gautas rezultatas nepatenka į adresų erdvės segmentą už kurį atsakingas mazgas.

Atsitiktinių mazgų identifikatorių generavimui taip pat galima naudoti kriptografines maišos funkcijas. Naujas mazgas norintis gauti identifikatorių sugeneruoja atsitiktinę reikšmę r ir randa šios reikšmės maišos funkcijos reikšmę $x=H(r)$. Gauta reikšmė x naudojama kaip mazgo identifikatorius. Prireikus įrodyti, kad identifikatorius buvo generuotas atsitiktinai mazgas gali pateikti r reikšmę ir savo identifikatorių x, o tikrinantis mazgas gali paskaičiuoti $H(r)$ ir palyginti su x. Kadangi žinant tiktai x rasti r yra sunku, r reikšmės žinojimas įrodo, kad mazgo identifikatorius x išskaičiuotas pagal $H(r)$, o ne pasirinktas laisvai. Identifikatoriaus generavimo įrodymo iliustracija pateikta 10 pav.



10 pav. Mazgo identifikatoriaus generavimo įrodymo schema

Toks identifikatorių generavimo metodas užtikrina kad adresai bus tolygiai pasiskirstę adresų erdvėje ir nebus galima laisvai pasirinkti norimo adreso, tačiau šis metodas neapsaugo nuo mazgo identifikatoriaus kopijavimo. Kiekvienas mazgas, kuriam buvo pateiktas įrodymas, kad $x=H(r)$ žino x ir r todėl gali taip pat pradėti naudoti šį adresą. Eclipse atakos atveju kenkėjiški mazgai galėtų negeneruoti naujų adresų, o prisijungus prie tinklo sužinoti r reikšmes tų identifikatorių, kurie tinkami atakai.

6.1 Identifikatoriaus kopijavimo apsaugos metodai

Apsisaugoti nuo mazgo identifikatoriaus kopijavimo galima pridėjus papildomus apribojimus, kurie nurodytų kokias r reikšmes mazgas gali naudoti. Kademlia [12] ir Chord [9] DHT modeliuose siūloma naudoti mazgo IP adreso maišos funkcijos reikšmę. Tada identifikatorių kenkėjiškas mazgas galėtų kopijuoti tik tuos identifikatorius, kurie sugeneruoti iš jam priklausančių adresų ruožo. Toks apribojimas yra efektyvus tačiau nepraktiškas. Dėl IP adresų trūkumo dažnai naudojamas NAT (angl. Network Address Translation) mechanizmas, leidžiantis vieną globaliai maršrutizuojamą IP adresą naudoti daugiau nei vienam įrenginiui. Taikant IP adreso hešavimo metodą iš tinklo kuriame naudojamas NAT prie DHT tinklo galėtų prisijungti tik vienas įrenginys. Taip pat atsiranda problema mobiliems įrenginiams, kurių IP adresas dažnai keičiasi. Tokie įrenginiai būtų privesti keisti DHT mazgo identifikatorių kiekvieną kartą, kai keičiasi jų IP adresas. Keičiant mazgo identifikatorių reikia

naujai užpildyti maršrutų lenteles ir gauti naujam identifikatoriui priklausančius saugoti duomenis.

Identifikatorių generavimo problemą galima spręsti turinti centralizuota sertifikavimo servisą [23, 24]. Tačiau toks sprendimas negali būti taikomas P2P sistemose. Alternatyvus mazgo identifikatoriaus generavimo įrodymo metodai pasiūlyti [18] ir [19] naudoja viešo rakto kriptografiją. Kiekvienas mazgas prisijungdamas prie tinklo sugeneruoja privataus ir viešo rakto porą (PK_x, SK_x). Mazgo identifikatorius *x* randamas apskaičiuojant $x = H(SK_x)$. Mazgo identifikatoriaus generavimo įrodymui pateikiamas mazgo *x* viešas raktas SK_x, o mazgo tapatybės įrodymui panaudojant privatų raktą PK_x generuojamas skaitmeninis parašas. Taikant tokią sistemą neatskleidus mazgo privataus rakto PK_x pateikiamas įrodymas, kad mazgas turi viešą raktą SK_x, atitinkantį privatų raktą PK_x.

Tokia sistema nėra naudojama didžiuosiuose eMule ir BitTorrent DHT tinkluose dėl jos realizacijos sudėtingumo ir reikalavimų skaičiavimo resursams. Skaitmeninio parašo generavimas yra daug resursų reikalaujanti operacija nes generuojant skaitmeninį parašą atliekama didelių skaičių (priklausomai nuo privataus rakto dydžio, dažniausiai apie 2048 bitų ilgio) kėlimo laipsniu operacija. Naudojamų resursų kiekiui sumažinti į pastovų ryšį orientuoti protokolai, tokie kaip SSL ar TLS privačius raktus naudoja tiktai bendro simetrinio rakto parinkimui, kurį toliau naudoja atidaryto ryšio kanalo žinučių šifravimui ir patikrinimui. DHT tinklo komunikacijų protokolai nėra orientuoti į ilgalaikius ryšius ir duomenų perdavimą ir susideda iš paprastų užklausų-atsakymų žinučių, kur gavėjas gali būti betkuris DHT tinklo mazgas. Dėl šios priežastys eMule ir BitTorrent DHT protokolai žinučių perdavimui naudoja ne TCP, o UDP transporto sluoksnio protokolą. UDP ryšiai nepalaiko ryšio būsenos, todėl norint pritaikyti skaitmeninius parašus ir viešo rakto kriptografiją reikėtų skaitmeninė parašą generuoti kiekvienam duomenų paketui.

Resursų reikalingų skaitmeninių parašų generavimui įvertinimui atliktas eksperimentas. Naudojant 2048 bitų ilgio RSA raktų porą (PK_x, SK_x) generuojami fiksuoto ilgio 512 baitų žinutės skaitmeniniai parašai. Žinutės santrumpos generavimui naudojama SHA-256 kriptografinė maišos funkcija. Skaitmeninio parašo generavimui naudojama openssl biblioteka, skaičiavimai atliekami tik ant vieno procesoriaus branduolio. 5 lentelėje pateikiami 1000 skaitmeninių parašų generavimo laikai milisekundėmis skirtingo galingumo asmeniniuose kompiuteriuose, gautas skaitmeninių parašų skaičius per sekundę ir maksimalus galimas pasirašomų duomenų srautas per sekundę (parašų skaičius padaugintas ir 512 baitų paketo dydžio).

5 lentelė. RSA skaitmeninių parašų generavimo laikas

Procesorius modelis, taktinis dažnis	parašai, vnt.	laikas, msec	parašai per sekundę	duomenų srautas, KB/s
AMD Sempron(tm) @ 1.8GHz	1000	24730	40	20
Intel Core2 6600 @ 2.40GHz	1000	2732	366	183
Intel Core i5-2500K @ 3.30GHz	1000	1358	736	368

Gauti rezultatai rodo, kad net moderniuose kompiuteriuose pilnai apkrovus vieną procesoriaus branduolį pasirašinėjamų žinučių srautas nesiekia 1 MB/s. Praktiškai skaitmeninį RSA parašą galima naudoti tiktai sesijos sudarymui, o ne kiekvienai žinutei. DHT protokolus galima pritaikyti į sesijas orientuotų ryšių sudarymu, tačiau tai protokolą padaro papildomai sudėtingesniu.

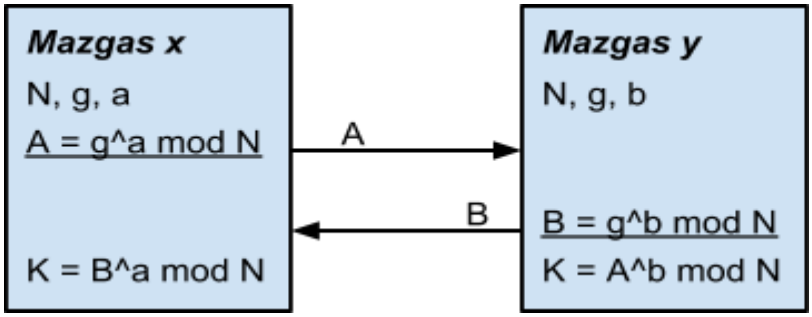
6.2 Siūloma kriptografinė identifikatorių apsaugos sistema

Atsižvelgiant į problemas taikant klasikinį viešo rakto kriptografijos metodą darbe aprašoma nauja sistema. Nauja sistema turi viešo rakto kriptografijos savybių leidžiančių apsaugoti identifikatorius nuo kopijavimo, tačiau mazgų autentifikacijai naudoja simetrinius raktus. Siūloma sistema remiasi Diffie-Hellman raktų apsikeitimo protokolu [25] ir ElGamal viešo rakto kriptografijos sistema [26].

Klasikinis Diffie-Hellman raktų apsikeitimo protokolas. Diffie-Hellman raktų apsikeitimo tarp mazgų *x* ir *y* atliekamas tokiu būdu:

1. Tinkle viešai žinomos dvi konstantos reikalingos raktų apsikeitimui, N (modulis) - skaičius kurio moduliu atliekamos operacijos, g (generatorius) - skaičius, kuris yra sąlyginai pirminis skaičiui N ir aibė $\{g^0, g^1, g^2, \dots, g^{N-1}\}\ mod\ N = \{0,1,2,3, \dots, N-1\}$.
2. Mazgas *x* atsitiktinai parenka skaičių a ir siunčia $A = g^a\ mod\ N$ mazgui *y*.
3. Mazgas *y* atsitiktinai parenka skaičių b ir siunčia $B = g^b\ mod\ N$ mazgui *b*.
4. Bendrą simetrinį raktą K_{xy} mazgas *x* randa apskaičiuodamas $K_{xy} = B^a\ mod\ N = g^{ba}\ mod\ N$, mazgas *y* raktą K_{xy} apskaičiuoja $K_{xy} = A^b\ mod\ N = g^{ab}\ mod\ N$.

Gauname, kad mazgai *x* ir *y* žino bendrą simetrinį raktą K_{xy} ir *g,N,A,B* reikšmes, o tinklą stebintys tarpiniai mazgai žino reikšmes *g,N,A,B*. Norint rasti rasti K_{xy} reikia rasti a arba b reikšmę skaičiuojant diskretų logaritmą $a = \log_g(A)\ mod\ N$ arba $b = \log_g(B)\ mod\ N$. Iki šiol nėra žinomas efektyvus algoritmas diskretaus logaritmo radimui. Principinė Diffie-Hellman raktų apsikeitimo schema pateikta 11 pav.



11 pav. Diffie-Hellman raktų apsikeitimo schema

Diffie-Hellman raktų apsikeitimo protokolo rezultatas yra bendras simetrinis raktas žinomas tiktai dviems protokolą atliekantiems mazgams, tačiau jį galimą panaudoti viešo rakto kriptografinės sistemos sukūrimui.

ElGamal viešo rakto kriptografijos schema. Viešo rakto kriptografinę sistemą panaudojant Diffie-Hellman protokolą pasiūlė Taher ElGamal [26]. Pagrindinė šios sistemos idėja yra pastebėjimas, kad raktų apsikeitimo protokolo žingsniai 2 ir 3 yra vienas nuo kito nepriklausomi ir jų atlikimo eilės tvarka ir laikas nėra svarbus. Mazgas *x* gali sugeneruoti skaičių a, rasti reikšmę $A = g^a \ mod \ N$ vieną kartą. Reikšmę A galime laikyti mazgo *x* viešu raktu $SK_x = A$ o a privačiu raktu $PK_x = a$. Viešai paskelbus SK_x reikšmę, bet kuris įrenginys ar tinklo mazgas panaudodamas SK_x gali nusiųsti žinutę, kuri bus perskaitoma tiktai mazgui *x*. Žinutės užšifravimui atsitiktinai parenkamas skaičius b, randamas žinutės raktas $K = (SK_x)^b \ mod \ N$, žinutė užšifruojama raktu K ir kartu su šifruota žinute nusiunčiama reikšmė $B = g^b$. Mazgas *x* žinute atšifruoja rasdamas raktą K pagal formulę $K = B^{PK_x} \ mod \ N$. Taher ElGamal taip pat pateikia ir metodą skaitmeninių parašų generavimui [26], kuriuos galima patikrinti žinant tiktai SK_x.

Siūlomos kriptografinės sistemos aprašas. Žinučių šifravimas viešu raktu ir skaitmeninių parašų tikrinimas tiek RSA tiek ElGamal metodu yra daug resursų reikalaujanti operacija. Abiem atvejais kiekvienai žinutei šifruoti ar parašui generuoti atliekamos atsitiktinio skaičiaus parinkimo ir laipsniu kėlimo operacijos. ElGamal šifravimo metodu siųsti žinutę kitam mazgui gali mazgas neturintis savo privataus ir viešo rakto poros, nes kiekvienai žinutei sukuriamas naujas žinutės raktas. Taip pat ElGamal metodu sukurtą skaitmeninį parašą gali patikrinti mazgas neturintis privataus ir viešo rakto poros, nes pačiame paraše pridedamas papildomas parašo tikrinimo raktas. Tokie reikalavimai yra būtini įprastoms viešo rakto kriptografinėms sistemoms, tačiau konkrečių DHT identifikatorių apsaugos atveju galime daryti prie laidą, kad kiekvienas DHT mazgas privalo turėti privataus ir viešo rakto porą. Panaudojus klasikinį Diffie-Hellman raktų apsikeitimo protokolą ir Taher ElGamal įžvalgą apie Diffie-Hellman protokolo privataus rakto savybes galima sukonstruoti naują kriptografinę schemą našiam DHT tinklo mazgų identifikatorių tikrinimui ir apsaugai.

Tarkime turime viešai žinomas g ir N reikšmes, o kiekvienas naujai prie DHT tinklo prisijungęs mazgas x sugeneruoja raktų porą (PK_x, SK_x), kur $SK_x = g^{PK_x} \, mod \, N$. Mazgo identifikatorius x randams $x = H(SK_x)$. Tuomet tarp visų DHT tinklo mazgų porų (x,y) egzistuoja vienas tik tai porai žinomas unikalus simetrinis raktas K_{xy}. Simetrinį raktą K_{xy} mazgas x gali apskaičiuoti kai sužino mazgo y viešą raktą SK_y. Vieši raktais sugeneruojami vieną kartą mazgui prisijungiant ir toliau darbo eigoje nekinta, todėl kitų mazgų viešus raktus galima kešuoti. Vieši raktai ir mazgų identifikatoriai yra vienas nuo kito priklausomi, todėl mazgas negali pasikeisi savo viešo rakto nepakeisdamas identifikatoriaus. Kiekvienas naujas prie tinklo prisijungęs mazgas turintis viešą raktą iškarto turi tiek statinių raktų kiek tinkle yra mazgų, šių raktų radimui tereikia apsikeisti viešais raktais.

Statinis raktas gali būti naudojamas skaitmeniniams parašas generuoti, tokį parašą gali sugeneruoti ir patikrinti tiktai simetrinį raktą žinantys mazgai. Tokie parašai vadinami MAC (angl. Message Authentication Code) - žinutės autentifikacijos kodais. Standartizuotas MAC algoritmas HMAC aprašytas RFC 2104 [27]. MAC apskaičiavimas reikalauja žymiai mažiau resursų nei asimetrinio skaitmeninio parašo generavimas. HMAC algoritme intensyviausia skaičiavimams operacija yra kriptografinės maišos funkcijos radimas. HMAC parašai pranašesni prieš įprastus viešo rakto kriptografijos parašus dėl jų ilgio. RSA parašo ilgis visa būna toks pat kaip ir privataus viešo rakto poros ilgis (šiuo metu dažniausiai naudojamas 2048 bitų). HMAC parašo ilgis priklauso tik nuo parašo generavimui naudojamos kriptografinės maišos funkcijos ilgio (dažniausiai naudojama SHA-1 160 bitų ilgio arba SHA-256 atitinkamai 256 bitų ilgio).

6.3 Eksperimentas

HMAC algoritmo efektyvumo palyginimui su RSA skaitmeniniais parašais atliktas eksperimentas. Eksperimento metu skaičiuojami 1000000 skirtingų 512 baitų ilgio žinučių HMAC parašai, panaudojant iš anksto paruoštą statinį 256 bitų raktą. HMAC skaičiavimui naudojama SHA-256 kriptografinė funkcija, openssl biblioteka, skaičiavimai atliekami tik ant vieno procesoriaus branduolio. Rezultatai pateikti 6 lentelėje. Be skaičiavimo trukmės taip pat pateiktas vidutinis parašų generavimo operacijų per sekundę skaičius ir įvertinamas galimas pasirašomų duomenų srautas (parašų skaičius dauginamas iš pasirašomos žinutės dydžio 512 baitų).

6 lentelė. HMAC parašų skaičiavimo trukmė

Procesorius modelis, taktinis dažnis	parašai, vnt.	laikas, msec	parašai per sekundę	duomenų srautas, MB/s
AMD Sempron @ 1.8GHz	1000000	14385	69516	33.943
Intel Core2 6600 @ 2.40GHz	1000000	6198	161342	78.780
Intel Core i5-2500K @ 3.30GHz	1000000	4812	207813	101.470

Gauti rezultatai patvirtino, kad HMAC parašų skaičiavimas yra žymiai mažiau (daugiau nei trimis eilėmis) resursų reikalaujanti operacija nei RSA skaitmeniniai parašai. DHT tinklo mazgai turintys 100 mb/s = 12.5MB/s tinklo liniją net ir su vidutinio galingumo procesoriumi spėtų generuoti reikiamą HMAC parašų skaičių.

Aprašytą mazgų identifikatorių apsaugos sistemą galima lengvai pritaikyti prie bet kuriam DHT tarnybinės informacijos apsikeitimo protokolui. Pakanka pridėti vieną papildomą žinutės ir atsakymo tipą viešo rakto užklausimui ir atsakymo atsiuntimui. Visos kitos DHT tarnybinės žinutės ir atsakymai modifikuojami taip, kad jų gale prirašomas žinutės turinio HMAC parašas.

6.4 Siūlomo metodo saugumo analizė

Siūlomos modifikuotos Diffie-Hellman raktų apsikeitimo schemos saugumas pagrindžiamas tokiu būdu. Kiekvienas mazgas viešai skelbdamas savo viešą raktą *SK* neišduoda daugiau informacijos nei klasikinį Diffie-Hellman raktų apsikeitimo protokolą vykdantys du mazgai. Iš viešai skelbiamo *SK* rasti privatų raktą *PK* galima rasti tiktai atliekant sveikaskaitinio logaritmo skaičiavimo operaciją. Bet kuris mazgas, net ir nedalyvaujantis DHT tinkle, žinodamas mazgo *x* viešą raktą SK_x gali sugeneruoti norimą skaičių raktų K_{x*} generuodamas PK_* reikšmes ir skaičiuodamas $K_{x*} = (SK_x)^{PK_*} \bmod N$, todėl papildomų kenkėjiškų mazgų atvedimas į tinklą nesuteikia privatų raktą norinčiam išgauti vartotojui daugiau informacijos nei jis gali gauti atvesdamas tiktai vieną mazgą.

Kenkėjiškas vartotojas gali nesilaikyti reikalavimų ir vietoje viešo rakto apskaičiavimo pagal formulę $SK_* = g^{PK_*} \bmod N$ viešą raktą pasirinkti laisvai, pavyzdžiui $SK_* = 1$. Teisingai veikiantis mazgas *x* apskaičiuos bendro rakto reikšmę pagal formulę $K_{x*} = 1^{PK_x} \bmod N = 1$. Tokiu būdu ataką atliekantis mazgas gali priversti visus kitus su juo komunikuojančius mazgus naudoti lengvai apskaičiuojamą raktą. Tačiau tokios atakos atlikimas turi įtakos tik žinutėms siunčiamoms ataką atliekančiam mazgui. Todėl ataką

atliekantis mazgas gali tiktai susilpninti savo identifikatoriaus apsaugą nepaveikdamas kitų mazgų identifikatorių stiprumo.

7 IDENTIFIKATORIŲ GENERAVIMO GREIČIO RIBOJIMO PROBLEMA

Panaudojus šeštame skyriuje aprašytą identifikatorių apsaugos metodą, Eclipse atakai atlikti reikia $O(M)$ identifikatorių generavimo operacijų. Identifikatoriaus generavimo operacijų skaičius randamas pagal formules (4) ir (7). Dideliuose DHT tinkluose (pavyzdžiui eMule arba BitTorrent tinkluose) atakos atlikimas reikalauja daug resursų, tačiau mažuose tinkluose kuriuose bendras mazgų skaičius tinkle nėra didelis Eclipse atakas galima atlikti su nedideliu resursų kiekiu. Eclipse atakas mažuose DHT tinkluose sulėtinti galima pakeičiant mazgo prisijungimo prie tinklo reikalavimus taip, kad naujo mazgo prijungimui prie tinklo (naujo identifikatoriaus generavimui) būtų sunaudojama daugiau kompiuterio resursų. Naujų identifikatoriių generavimo greičio ribojimas taip pat naudingas tuo, kad apsunkina bet kokią Sybil principu veikiančią ataką.

Papildomai išnaudojami kompiuterio resursai prijungiant naują mazgą prie tinklo turi būti įrodomi kitiems mazgams. Jei papildomų resursų išnaudojimas būtų tik DHT programinės įrangos reikalavimas, kenkėjiški mazgai galėtų modifikuoti programinę įrangą taip kad būtų išvengiamas nereikalingas resursų naudojimas. Tokį funkcionalumą galima realizuoti suformuluojant kaip matematinį galvosūkį, kurio sprendimą galima pateikti kaip įrodymą, kad galvosūkis tikrai buvo išspręstas. Sprendimo patikrinimas turėtų naudoti žymiai mažiau resursų nei uždavinio sprendimas. Galvosūkis turi būti valdomo sudėtingumo, tam kad jį būtų galima pritaikyti skirtingo dydžio DHT tinklams arba priderinti prie didėjančių kompiuterių skaičiavimo pajėgumų.

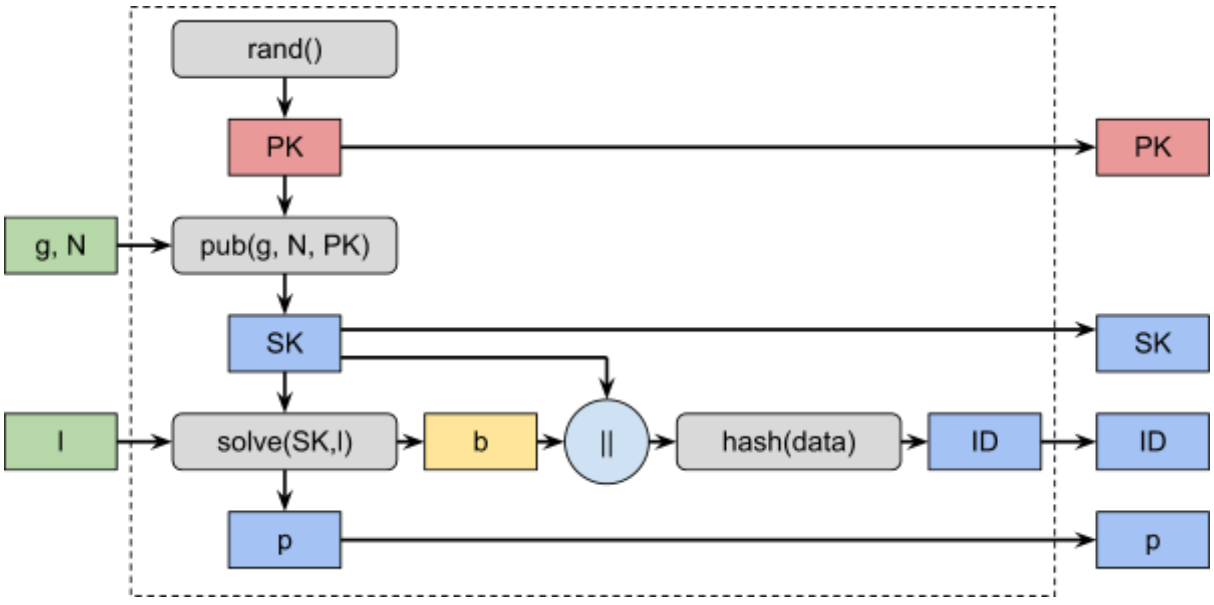
Tokie galvosūkiai vadinami vadinami POW (angl. Proof-of-work) [28]. POW galvosūkiai yra plačiai naudojami praktikoje, pavyzdžiui kriptografiškai generuojamų IPv6 adresų apskaičiavimui [29], kompiuterio atlikto darbo įrodymui generuojant elektroninius pinigus [30]. Maišos funkcijomis paremtas matematinis galvosūkis yra radimas tokios reikšmės a, kurios kriptografinės maišos funkcijos reikšmė $b=H(a)$ tenkina tam tikrus apribojimus. Pavyzdžiui rasti tokį a, kad pirmi l reikšmės b bitai būtų lygus 0. Žinant norimą maišos funkcijos rezultatą rasti funkcijos argumentą galima tiktai perrinkimo būdu. Kadangi kriptografinės maišos funkcijos rezultatas turi tokias pačias savybes, kaip pseudo atsitiktinių skaičių generatorius, tikimybė, kad bet kuris maišos funkcijos rezultato bitas bus 0 yra lygi $\frac{1}{2}$. Norint rasti maišos funkcijos reikšmę, kurios pirmi l bitai būtų lygūs 0 reikia vidutiniškai

apskaičiuoti 2^l maišos funkcijų reikšmių. Išspręstą galvosūkį galima lengvai patikrinti apskaičiuojant $b=H(a)$ ir patikrinant b reikšmės pirmus l-bitus.

Maišos funkcijų galvosūkio sprendinys b prie tam tikros sudėtingumo reikšmės i taip pat kaip ir identifikatoriaus generavimo įrodymas gali būti lengvai kopijuojamas. Todėl kiekviena kriptografinį galvosūkį reikia susieti su konkrečiu galvosūkį sprendžiančius mazgu, taip kad vieno mazgo x galvosūkio sprendinys būtų tinkamas tiktai mazgo x resursų išnaudojimo patikrinimui. Tai galima atlikti galvosūkį modifikuojant tokiu būdu. Mazgo x prisijungimo prie tinklo galvosūkio sprendinys p_x prie sudėtingumo lygio l yra tokia reikšmė su kuria suskaičiavus $b = H(SK_x||p_x)$ pirmieji l rezultato bitai yra lygus 0. Čia simboliu || žymima eilučių sujungimo operacija.

Matematinio galvosūkio reikšmę taip pat reikia susieti su mazgo identifikatoriumi. Jei mazgo identifikatorius ir galvosūkio sprendinys gali būti apskaičiuojamas nepriklausomai, tuomet kenkėjiškas vartotojas norintis gauti konkretų mazgo identifikatorių gali generuoti naujus identifikatorius nespręsdamas matematinio galvosūkio. Tuomet galvosūkio sprendimas būtų atliekamas tik vieną kartą radus tinkamą identifikatorių. Mazgo identifikatoriaus generavimo operaciją pakeitus taip, kad identifikatorius būtų randamas hešuojant ne tik mazgo viešą raktą, bet kartu pridedant ir galvosūkio sprendinį $x = H(SK_x||b) = H(SK_x||H(SK_x||p_x))$ gauname, kad konkreti mazgo identifikatoriaus reikšmė gali būti randama tiktai po matematinio galvosūkio sprendimo. Tai priverčia kenkėjišką mazgą spręsti matematinį galvosūkį kiekvieno mazgo identifikatoriaus generavimo metu.

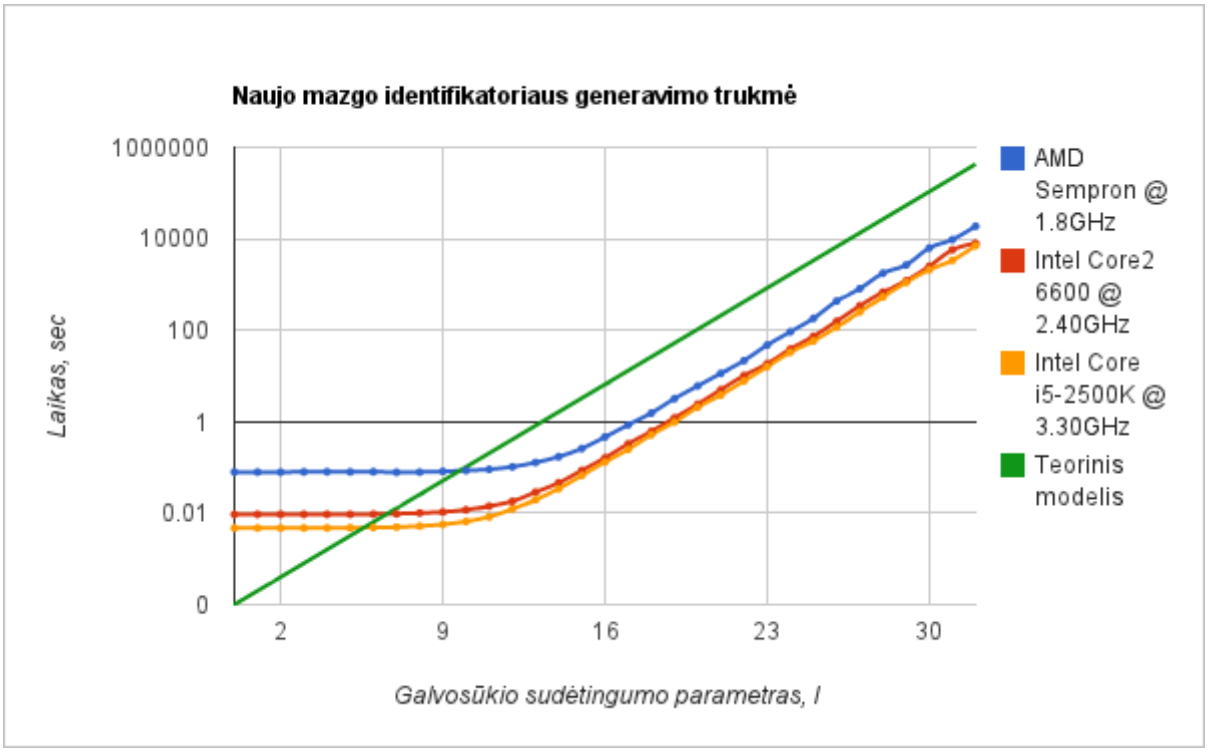
Apjungta identifikatorių generavimo įrodymo ir naujų mazgų sukūrimo greičio valdymo schema pateikta 12 pav.



12 pav. Naujo mazgo identifikatoriaus generavimo schema

Šioje schemoje funkcija *rand()* parenka atsitiktinį privatų raktą, rakto ilgis priklauso nuo bendro tinklo parametro *N*. Funkcija *pub()* iš privataus rakto ir bendrų tinklo parametrų *g* ir *N* pagal formulę $g^{PK} \ mod \ N$ randa viešą raktą *SK*. Funkcija *solve()* išsprendžia *l* sudėtingumo matematinį galvosūkį ir randa reikšmę *p*, bei *b*=*H*(*SK*||*p*).

Pateiktos schemos ir matematinio galvosūkio praktiniam sudėtingumo lygiui įvertinti buvo atliktas eksperimentas. Eksperimento metu buvo generuojami nauji identifikatoriai pagal pav. 12 schemą prie skirtingų galvosūkio sudėtingumo laipsnių l ir matuojama vidutinė tokios operacijos trukmė. Skaičiavimai buvo atlikti ant trijų skirtingo galingumo asmeninių kompiuterių. Mazgų privatus, viešas raktai ir *N* imami 2048 bitų ilgio, maišos funkcija SHA-256. Kriptografinės funkcijos naudojamos iš openssl bibliotekos. Tikrintos *l*=0,1,2,...,32 reikšmės. Rezultatai pateikiami 13 pav.



13 pav. Identifikatoriaus generavimo trukmės bandymas

Grafike pateikiamas vidutinis naujo identifikatoriaus generavimo laikas, su reikšmėmis *l*=0,1,2,...,24 pateiktas 100 generavimo operacijų vidurkis. Su didesnėmis *l*=25,26,...,32 skaičiuotas 10 generavimo operacijų vidurkis. Mažesnis generavimo operacijų skaičius parinktas todėl, kad prie didelių *l* eksperimentas užtrunka pakankamai ilgai, dėl mažesnio bandymų skaičiaus grafike matomi didesni kreivių svyravimai. Papildomai grafike pridėta teorinio modelio kreivė. Teoriniame modelyje laikoma, kad raktų generavimas papildomai laiko neužtrunka, o viena maišos funkcijos skaičiavimo operacija trunka 0.0001 sekundės dalį, todėl brėžiamas $f(l) = 0.0001 \cdot 2^l$. Gauti rezultatai, kad l reikšmės nuo 0 iki 10 identifikatoriaus generavimo laiko beveik neįtakoja, nes didesnė laiko dalis sugaištama mazgo

privataus ir viešo rakto generavimo procedūrai, nei matematinio galvosūkio sprendimui. Teorinio modelio ir praktinių rezultatų gautos tiesės prie didesnių l reikšmių yra lygiagrečios, tai rodo praktinė realizacija atitinka teorinį modelį ir didinant l vienu vienetu naujo mazgo identifikatoriaus generavimui reikalingi resursai išauga lygiai du kartus.

Iš atlikto eksperimento rezultatų nustatyta, kad praktinėse realizacijose reikėtų rinktis galvosūkio sudėtingumo parametrą t iš intervalo nuo 24 iki 32. Kai t=24 priklausomai nuo procesoriaus galingumo naujas mazgas prie tinklo galės prisijungti po 0,5 - 1,5 minutės skaičiavimo. Kai t=32 nauji mazgai vidutiniškai galės prisijungti po 1,5 - 3 valandų skaičiavimo. Renkantis t parametrą reikėtų atsižvelgti į tinklo dydi nes atakoms reikalingas identifikatorių generavimo skaičius tiesiogiai priklauso nuo tinkle esančių mazgų skaičiaus.

8 IŠVADOS

Atlikus darbą gautos tokios išvados:

1. Nustatyta, kad tinkluose, kuriuose mazgo identifikatorių galima pasirinkti laisvai Eclipse ataką galima atlikti su fiksuotu resursų kiekiu $O(1)$ nepriklausomai nuo tinklo dydžio.
2. Nustatyta, kad tinkluose, kuriuose mazgų identifikatoriai privalomai generuojami atsitiktinai Eclipse atakai reikalingas resursų kiekis priklauso nuo tinkle veikiančių mazgų skaičiaus $O(M)$.
3. Atlikus BitTorrent tinklo veikimo eksperimentinį tyrimą nustatyta - tinklo dydis, apie 16 milijonų mazgų, tinkle $\approx 2\%$ visų naudojamų identifikatorių nėra generuoti atsitiktinai, tinkle yra atliekamos Eclipse atakos.
4. Išnagrinėjus klasikinius identifikatorių apsaugos metodus, nustatyta, kad klasikiniai metodai reikalauja daug resursų ir dėl to nėra naudojami praktiškai.
5. Pasiūlytas naujas identifikatorių generavimo ir apsaugos nuo kopijavimo metodas, kuriame naudojami simetriniai skaitmeniniai parašai, dėl to pasiekiamas didesnis našumas ir supaprastinama metodo realizacija. Atliktas bandymas patvirtinantis metodo efektyvumą.
6. Pasiūlytas mechanizmas leidžiantis apsunkinti Sybil atakas ribojant naujų identifikatorių generavimo greitį. Atliktas bandymas patvirtinantis metodo efektyvumą.

9 LITERATŪRA

[1] **Lakshman A., Malik P.** Cassandra - A Decentralized Structured Storage System// Proceedings of the Workshop on Large-Scale Distribued Systems and Middleware (LADIS '09), Big Sky MT, 2009.

[2] **Mondéjar R., García-López P., Pairot C., Pamies-Juarez L.** CloudSNAP : A transparent infrastructure for decentralized web deployment using distributed interception// Future Generation Computer Systems. ISSN 0167-739X. 2011.

[3] **Freedman M. J., Freudenthal E., Mazi` D.** Democratizing content publication with Coral// Networked Systems Design and Implementation. San Francisco CA, 2004.

[4] **Antoniu G., Hatcher P., Jan M., Noblet D.** Performance Evaluation of JXTA Communication Layers// Proceedings of the Fifth International Workshop on Global and Peer-to-Peer Computing. 2005.

[5] YaCy - The Peer to Peer Search Engine: Technology [interaktyvus]. 2011 m. gruodis. - [žiūrėta 2012-02-15]. Prieiga per internetą: http://yacy.net/en/Technology.html

[6] **Kügler D.** An Analysis of GNUnet and the Implications for Anonymous, Censorship-Resistant Networks// Privacy Enhancing Technologies workshop. Volume 2760 of LNCS. Dresden, Germany, 2003. p. 161–176.

[7] **Clarke I., Sandberg O., Wiley B., Hong T. W.** Freenet: A distributed anonymous information storage and retrieval system// Proceedings of the ICSI Workshop on Design Issues in Anonymity and Unobservability. Berkeley, California. 2000.

[8] **Ratnasamy S., Francis P., Handley M., Karp R., Henker S.** A scalable content-addressable network// Proceedings of ACM SIGCOMM'01. San Diego CA. 2001.

[9] **Stoica I., Morris R., Karger D., Kaashoek M. F., Balakrishnan H.** Chord: A scalable peer-to-peer lookup service for Internet applications// Proceedings of ACM SIGCOMM'01, San Diego CA, 2001.

[10] **Rowstron A., Druschel P.** Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems// IFIP/ACM International Conference on Distributed Systems Platforms (Middleware). Heidelberg, Germany. 2001. p. 329–350.

[11] **Zhao B. Y., Kubiatowicz J. D., Joseph A. D.** Tapestry: An infrastructure for fault resilient wide-area location and routing// Technical Report UCB//CSD-01-1141, U. C. Berkeley. 2001.

[12] **Maymounkov P., Mazieres D.** Kademlia: A peer-to-peer information system based on the XOR metric// Proceedings of IPTPS. Cambridge MA. 2002. p. 53–65.

[13] FIPS 180-1. Secure Hash Standard. U.S. Department of Commerce/NIST, National Technical Information Service. Springfield, VA. 1995.

[14] **Schäfer J., Malinka K., Hanáček P.** Peer-to-Peer Networks: Security Analysis// Intl. Journal on Advances in Security, volume 2, 2009. p. 53-61.

[15] **Singh A., Castro M., Druschel P., Rowstron A.** Defending Against Eclipse Attacks on Overlay Networks// Proceedings of SIGOPS European Workshop. Leuven, Belgium. 2004.

[16] **Douceur J. R.** The Sybil attack// 1st International Workshop on Peer-to-Peer Systems (IPTPS '02). 2002.

[17] **Wallach D.** A survey of peer-to-peer security issues// In International Symposium on Software Security. Tokyo, Japan. 2002.

[18] **Sit E., Morris R.** Security Considerations for Peer-to-Peer Distributed Hash Tables// Proceedings of 1st Iinternational Workshop on Peer-to-Peer Systems (IPTPS). Cambridge MA. 2002.

[19] **Baumgart I., Mies S.** S/Kademlia: A Practicable Approach Towards Secure Key-Based Routing// IEEE International Conference on Parallel and Distributed Systems. 2007.

[20] **Steiner M., En-Najjary T., Biersack E.** A Global View of KAD// Proceedings of the 7th ACM SIGCOMM conference on Internet measurement. ISBN: 978-1-59593-908-1. 2007.

[21] **Loewenstern A.** BitTorrent Enhancement Proposal 5 (BEP5): DHT Protocol. 2008.

[22] **Jimenez R., Osmani F., Knutsson B.** Connectivity properties of Mainline BitTorrent DHT nodes// 9th International Conference on Peer-to-Peer Computing. Seattle, Washington, USA. 2009.

[23] **Maccari L., Rosi M., Fantacci R., Chisci L., Aiello L. M., Milanesio M.** Avoiding Eclipse attacks on Kad/Kademlia: an identity based approach// ICC 2009 Communication and Information Systems Security Symposium. 2009.

[24] **Steiner M., En-Najjary T., Biersack E.** Exploiting KAD: Possible Uses and Misuses// ACM SIGCOMM Computer Communication Review, Volume 37 Issue 5. New York NY. 2007.

[25] **Diffie W., Hellman M. E.** New directions in cryptography// IEEE Trans. Info. Theory IT-22 No.6. 1976. p. 644-654.

[26] **ElGamal T.** A public key cryptosystem and a signature scheme based on discrete logarithms// IEEE Trans. Info. Theory IT-31 No.4. 1985. p. 469-472.

[27] **Krawczyk H., Bellare M., Canetti R.** HMAC: Keyed-Hashing for Message Authentication// RFC 2104. 1997.

[28] **Jakobsson M., Juels A.** Proofs of work and bread pudding protocols// In Proceedings of the IFIP TC6 and TC11 Joint Working Conference on Communications and Multimedia Security (CMS ’99). Leuven, Belgium. 1999.

[29] **Aura T.** Cryptographically Generated Addresses (CGA)// 6th Information Security Conference (ISC'03). Bristol, UK. 2003.

[30] **Nakamoto S.** Bitcoin: A Peer-to-Peer Electronic Cash System. 2008.

ANALYSIS OF SECURITY IMPLICATIONS IN DHT NETWORK IF NODE ID CAN BE SELECTED ARBITRARILY

Summary

Distributed Hash Table models and its security implications, has long been a subject of interest. This thesis is based on the assumption that practical implementations do not enforce random node id generation regardless of the fact that theoretical models require node ids to be chosen by random and distributed in the address space uniformly. To measure the impact on the DHT network security if the assumption holds an analysis of attack complexity in both cases is performed. Results indicate that the complexity grows from $O(1)$ to $O(M)$ if the node id cannot be selected arbitrarily (M is the number of nodes in DHT network).

Stated assumption is confirmed by analysing classic node id protection methods and performing analysis of BitTorrent DHT network. The reason for the lack of node id protection in practice is considered to be the complexity and performance penalty of the classic methods.

To facilitate the implementations of DHT networks a new method to ensure random node id generation and copy protection is provided. Proposed method utilizes MACs based on shared keys to provide a proof of the ownership of the node id while still providing means to protect it from being copied. Efficiency of the proposed method is evaluated by conducting an experiment.

In order to protect small DHT networks against a Sybil attack a method to control the speed of node id generation is also proposed.

10 SANTUMPŲ IR TERMINŲ ŽODYNAS

P2P - (angl. Peer to Peer) lygiarangė paskirstyta sistema. Tai tokia paskirstyta sistema, kurioje visi mazgai turi vienodą įtaką tinklo darbui ir gali atlikti tokias pačias funkcijas.

DHT - (angl. Distributed Hahs Table) paskirstyta maišos lentelė. Tai paskirstyta sistema atliekanti duomenų saugyklos funkciją

DoS - (angl. Denial of Service) paslaugos pasiekiamumo sutrikdymo ataka.

POW - (angl. Proof of Work) matematinis galvosūkis kurio sprendimas įrodo, kad jį sprendęs įrenginys išleido tam tikrą resursų kiekį.

RPC - (angl. Remote Procedure Call) komunikavimo tarp procesų arba įrenginių mechanizmas imituojantis programos funkcijos iškvietimą. Užklausoje pateikiamas metodo vardas ir argumentų sarašas, atsakyme pateikiamas metodo gražinamas rezultatas.

BitTorrent Tracker - Serveris atsakingas už informacijos mainų koordinavimą tarp BitTorrent klientų.

ASCII - (angl. American Standard Code for Information Interchange) standartinė simbolių kodavimo sistema, kurioje vienas simbolis koduojamas 8 bitais.

XML - (angl. Extensible Markup Language) dokumentų struktūrų ir jų turinio aprašomoji kalba.

CDN - (angl. Content Distribution Network) Duomenų paskirstymo tinklas. Tai paskirstyta sistema arba infrastruktūra, dažniausiai veikianti per kelis duomenų centrus, skirta sparčiai ir patikimai pateikti informaciją klientams.

Eclipse ataka - tai ataka prieš DHT tinklą, kurios tikslas pasirinktą tinklo mazgą izoliuoti nuo likusio tinklo užpildant jo maršrutų lenteles vien tik kenkėjiškais mazgais.

Sybil ataka - tai ataka prieš P2P sistemą, kurios atlikimui į tinklą atvedamas didelis kiekis bendram tikslui dirbančių kenkėjiškų mazgų.

11 PRIEDAI

11.1 Tekste naudojami žymėjimai

x, y - mazgo arba duomenų adresas

d - adresų erdvės matavimų skaičius, $x \in N^d$

b - adreso kodavimo bazė

n - adresų erdvės ilgis skaitmenimis bazėje

N - adresų erdvės dydis (unikalių adresų skaičius), $N = b^n$

M - mazgų skaičius tinkle

k - KAD tinklo parametras

r - atsitiktinai parinkta reikšmė

H(*data*) - duomenų *data* kriptografinės maišos funkcijos reikšmė

SK_x - mazgo *x* viešas raktas

PK_x - mazgo *x* privatus raktas

K_{xy} - simetrinis raktas komunikacijai tarp mazgų *x* ir *y*

p_x - mazgo *x* matematinio galvosūkio sprendinys

l - mazgo prisijungimo prie tinklo uždavinio sudėtingumo laipsnis

11.2 Konferencijų medžiaga

LAISVAI PASIRENKAMO MAZGO IDENTIFIKATORIAUS ĮTAKOS DHT TINKLO SAUGUMUI ANALIZĖ

Julius Kriukas[1], Rimantas Kavaliūnas[2]

[1] *KTU, Kompiuterių tinklų katedra, Studentų g. 50, Kaunas, Lietuva, julius.kriukas@ktu.lt*
[2] *KTU, Kompiuterių tinklų katedra, Studentų g. 50, Kaunas, Lietuva, rimantas.kavaliunas@ktu.lt*

Santrauka (abstract). Ne visos paskirstytos lygiarangės (angl. Peer-to-peer, P2P) sistemos, naudojančios paskirstytas maišos lenteles (angl. Distributed Hash Tables, DHT) užtikrina, kad tinkle veikiantys mazgai privalo naudoti atsitiktiniu būdu sugeneruotus identifikatorius. Šiame straipsnyje nagrinėjamas DHT tinklo modelių Kademlia ir Chord atsparumas DoS (angl. Denial-of-Service) atakoms, kai DHT modelis naudojamas konkrečioje lygiarangėje sistemoje leidžia mazgams laisvai pasirinkti mazgo identifikatorių.

Raktiniai žodžiai: DHT, P2P, Kademlia, Chord, paskirstyti tinklai, DoS, Eclipse ataka, Sybil ataka.

1 Įžanga

Paskirstytų maišos lentelių (angl. Distributed Hash Tables, DHT) tinklo modeliai naudojami daugelyje lygiarangių (angl. Peer-to-peer, P2P) sistemų. Toliau P2P sistema bus vadinamas konkretus DHT tinklas, veikiantis pagal kurį nors DHT modelį. DHT modelis nusako kaip suformuoti paskirstytą duomenų saugyklą, bei mazgus jungiantį tinklą. Duomenims ir mazgams priskiriami identifikatoriai arba adresai. Šie terminai straipsnyje naudojami kaip sinonimai. DHT tinklo modelis nusako adresų erdvės struktūrą – kokias reikšmes gali įgyti identifikatoriai ir kokie santykiai gali būti tarp jų, pvz. atstumo metriką. DHT modeliuose duomenų ir mazgų identifikatoriai generuojami atsitiktinai, su vienoda tikimybe parinkti identifikatorių iš visos galimų adresų erdvės. Tada duomenys DHT tinkle pasiskirstyto tolygiai ir maršrutų parinkimo algoritmai veikia sparčiai. Kadangi nuo mazgo identifikatoriaus priklauso kokiame adresų erdvės segmente esančius duomenis jis turės saugoti ir su kuriais kitais mazgais turės užmegzti ryšį, galimybė laisvai pasirinkti mazgo identifikatorių leidžia kenkėjiškam vartotojui įterpti kenkėjiškus mazgus specialiai parinktose vietose siekiant sutrikdyti darbą tinkle. Nuo mazgo identifikatoriaus parinkimo taisyklių priklauso P2P sistemos atsparumas DoS (angl. Denial-of-Service) atakoms.

P2P sistemų programinėje įrangoje realizuojami atsitiktinių identifikatorių generavimo algoritmai. Tačiau jeigu sistema yra atvira, kenkėjiškas vartotojas gali modifikuoti programinę įrangą ir laisvai pasirinkti tinklo mazgo identifikatorių. Pilnai paskirstytose P2P sistemose (be centralizuoto valdymo) patikrinti, ar naujai prisijungęs mazgas identifikatorių sugeneravo atsitiktinai, ar pasirinko specialų, nėra paprasta. Todėl praktinės P2P sistemose neatlieka specialiai parinktų identifikatorių aptikimo.

Šiame straipsnyje nagrinėjama kaip pasikeičia P2P sistemos atsparumas DoS atakoms, jei ji neturi mechanizmo, leidžiančio aptikti ne atsitiktinai generuotus identifikatorius.

DoS atakas prieš DHT tinklą galima suklasifikuoti į tokias grupes:

1. DoS ataka prieš DHT tinklo mazgą. Sėkmingai atlikus ataką mazgas atskiriamas nuo DHT tinklo. Ši ataka vadinama Eclipse ataka [1].
2. DoS ataka prieš DHT tinkle esančius duomenis. Sėkmingai atlikta ataka izoliuoja pasirinktą duomenų bloką nuo likusio DHT tinklo ir duomenys tampa nebepasiekiami.
3. DoS ataka prieš visą DHT tinklą. Sėkmingai atlikta ataka pilnai sustabdo DHT tinklo veikimą.
4. DoS ataka, panaudojant DHT tinklą, prieš įrenginius neprisijungusius prie DHT tinklo. Sėkmingai atlikta ataka panaudoja DHT tinkle esančių mazgų resursus, pasirinkto įrenginio darbo sutrikdymui.

Šiame straipsnyje bus nagrinėjama Eclipse ataka prieš Chord [2] ir Kademlia [3] DHT tinklo modelius. Pirmiausia bus nustatomas kenkėjiškų mazgų skaičius reikalingas atakai atlikti. Toliau, remiantis rastu kenkėjiškų mazgų skaičiumi, įvertinamas identifikatorių generevimo operacijų skaičius, kai tinkle negalima laisvai pasirinkti identifikatoriaus.

2 Identifikatoriaus parinkimo sudėtingumas

Pilnai paskirstytose P2P sistemose visi mazgai yra lygiaverčiai ir gali atlikti vienodas funkcijas. Jei paskirstyta sistema yra atvira, prie jos nevaržomai gali prisijungti nauji mazgai. Į tokią sistemą kenkėjiškas vartotojas gali įvesti tiek mazgų, kiek turi laisvų resursų. Kuo didesnė kenkėjiškų mazgų dalis tinkle, tuo didesnė jų bendra įtaka tinklo darbui. Kenkėjiškų mazgų su skirtingomis tapatybėmis (DHT atveju identifikatoriais)

įvedimas į tinklą vienam bendram tikslui vadinamas Sybil ataka [4]. Sybil ataką galima atlikti visoms atviroms P2P sistemoms, skiriasi tik atakai atlikti reikiamas resursų kiekis [4].

Jeigu P2P sistemoje yra mechanizmas, leidžiantis patikrinti ar mazgo identifikatorius tikrai yra sugeneruotas atsitiktinai, kenkėjiškas vartotojas, norintis gauti vieną konkretų identifikatorių, turi generuoti naujus identifikatorius tol kol bus gautas ieškomas identifikatorius. Tinkle, kurio adresų erdvė sudaryta iš n bitų ilgio identifikatorių, tikimybė gauti norimą identifikatorių (įvykis A) apskaičiuojama pagal formulę (1).

$$P(A)=\frac{1}{2^n} \qquad (1)$$

Chord [2] ir Kademlia [3] DHT modeliuose naudojami n=160 bitų ilgio identifikatoriai. Tikimybė atsitiktinai sugeneruoti norima identifikatorių šio modelio tinkluose yra $2^{-160}\approx 6.84\cdot 10^{-49}$. Tačiau DHT modeliuose mazgo saugomi duomenų blokai ir ryšiai su kitais mazgais niekada nepriklauso nuo absoliučios mazgo identifikatoriaus reikšmės, o priklauso nuo mazgo santykinės pozicijos tinke (kas yra mazgo kaimynai ir kokiame adresų erdvės segmente yra mazgas). Todėl kenkėjiškam vartotojui nebūtina turėti konkretų identifikatorių o pakanka patekti į norimą adresų erdvės segmentą. Atsitiktinius identifikatorius turintys mazgai tolygiai pasiskirsto visoje DHT adresų erdvėje, todėl tinklas, kuriame yra m mazgų, bendrą adresų erdvę padaliną į m panašaus dydžio segmentų. Tikimybė naujam mazgui patekti į norimą tinklo adresų segmentą (įvykis B) yra apskaičiuojama pagal formulę (2).

$$P(B)\approx\frac{1}{m} \qquad (2)$$

Tikimybė patekti į norimą adresų erdės segmentą priklauso tiktai nuo mazgų skaičiaus tinkle. Pavyzdžiui Kademlia tinkle [5] užfiksuoti 3.5 milijonai vartotojų, todėl tikimybė pateikti į norimą segmentą yra $3500000^{-1}\approx 2.86\cdot 10^{-7}$.

3 Chord modelio analizė

Chord DHT modelyje [2] adresų erdvėje telpa $N=2^n$ galimų identifikatorių. Adresų erdvė dalinama tokiu būdu: mazgas, kurio identifikatorius yra x, o jam artimiausias kaimynas su mažesniu identifikatoriumi y (moduliu N) yra atsakingas už adresų erdvę $(y+1, x]$. Kitaip tariant mazgas yra atsakingas už adresų erdvės segmentą nuo artimiausio mazgo su mažesniu identifikatoriumi $+1$ iki savo identifikatoriaus numerio imtinai. Atstumas d tarp dviejų identifikatorių x ir y Chord modelyje yra identifikatorių reikšmių skirtumas adresų erdvės dydžio moduliu: $(x-y) mod N$.

Greitai paieškai Chord tinkle kiekvienas mazgas turi maršrutų lentelę sudarytą iš n-1 įrašų. Mazgo su identifikatoriumi x maršrutų lentelės i-tojoje eilutėje saugomas mazgo atsakingo už adresų erdvės tašką randamą pagal (3) formulę IP adresas.

$$(x+2^i) mod N;\ \ i=0,1,\ldots,n-1 \qquad (3)$$

3.1 Kenkėjiškų mazgų skaičius Eclipse atakai

Eclipse atakos atlikimui reikia užpildyti mazgo maršrutų lentelę vien tik kenkėjiškų mazgų identifikatoriais, kurie neperduotų informacijos tarp puolamo mazgo ir likusio tinklo. Iš (3) formulės matome, kad mazgo x maršrutų lentelė pildoma ieškant duomenų identifikatorių, kurie vienareikšmiškai nusakomi žinant x. Todėl garantuotos DoS atakos atlikimui prieš mazgą x, į tinklą reikia įvesti n kenkėjiškų mazgų su identifikatoriais z_i, kai $i = 0,1,\ldots,n-1$, kurie išsidėstytų pozicijose pagal (3) formulę. Ataką galima supaprastinti įvertinus tai, kad adresų erdvė niekada nebūna pilnai išnaudojama ir tinkle esant m mazgų kiekvieno mazgo maršrutų lentelėje vidutiniškai užpildoma tiktai (4) formule apskaičiuojamas įrašų kiekis.

$$\log_2(m) \qquad (4)$$

Turint galimybę laisvai pasirinkti mazgo identifikatorių, mazgą x galima pilnai atskirti nuo DHT tinklo į tinklą įvedant z_i, $i=0,1,\ldots,\log_2(m)-1$ kenkėjiškus mazgus su identifikatoriais randamais pagal formulę (3).

3.2 Identifikatorių generavimo operacijų skaičius Eclipse atakai

Neturint galimybės laisvai pasirinkti mazgo identifikatorių, ataka tampa sudėtingesnė. Tarkime ataka atliekama prieš mazgą su identifikatoriumi x, esantį tinkle, kuriame yra m mazgų. Tuomet visa adresų erdvė yra sudalinta į m segmentų. Iš visų m segmentų atrenkami n segmentai, į kuriuos patenka z_i identifikatoriai. Tada kenkėjiškų mazgų identifikatoriai generuojami tol, kol sugeneruotas identifikatorius patenka į vieną iš reikalingų segmentų i ir skaitine reikšme yra didesnis arba lygus z_i. Toks identifikatorius yra tinkamas atakai. Teisingai veikiančių mazgų identifikatoriai adresų erdvėje yra pasiskirstę tolygiai atsitiktinai, todėl identifikatorius z_i

segmento i viduje su vienoda tikimybe gali būti bet kurioje pozicijoje. Tikimybė, kad segmente i rastas identifikatorius yra tinkamas (didesnis arba lygus z_i) yra tokia pati kaip ir tikimybė, kad jis netinkamas ir lygi $\frac{1}{2}$. Remiantis (2) formule gauname, kad vienam tinkamam kenkėjiškam identifikatoriui rasti reikės vidutiniškai perrinkti $2 \cdot m$ skirtingų identifikatorių. Garantuotai atakai reikia n tinkamų kenkėjiškų identifikatorių, tačiau praktiškai pakanka pagal (4) randamo identifikatorių skaičiaus. Visiems jiems rasti vidutiniškai reikės perrinkti (5) formule randamą skirtingų identifikatorių kiekį.

$$\sum_{i=0}^{\log_2(m)-1} \frac{2 \cdot m}{\log_2(m)-i} \qquad (5)$$

4 Kademlia modelio analizė

Kademlia DHT modelis [3] yra dažniausiai naudojamas praktikoje: eMule, BitTorrent, Overnet, Gnutella failų apsikeitimo programose, taip pat paskirstytų web svetainių talpinimo projekte Osiris. Esminis skirtumas nuo Chord DHT modelio yra atstumo tarp mazgų skaičiavime. Kademlia modelyje atstumas skaičiuojamas panaudojant XOR funkciją tarp dviejų identifikatorių reikšmių. Kiekvienas Kademlia tinklas turi visiems mazgams bendrą tinklo parametrą k, kuris nurodo kiek kartų duomenys replikuojami įkeliant juos į tinklą, ir kiek įrašų saugoma kiekvienoje maršrutų lentelėje. Kiekvienas mazgas talpinantis duomenis į DHT tinklą turi periodiškai atlikti pakartotinę duomenų įkėlimo operaciją į k mazgų, kurių identifikatoriai yra arčiausiai duomenų identifikatoriaus (pagal XOR metriką).

Maršrutų lentelės sudaromos tokiu būdu: mazgas su identifikatoriumi x turi n maršrutų lentelių M_i: $i=0,1,\ldots,n-1$, kur lentelėje M_i saugomi IP adresai tų mazgų, kurių identifikatorių pirmi i bitai sutampa su mazgo identifikatoriumi x, tačiau $i+1$ bitas nesutampa. Kiekvienoje lentelėje saugoma k įrašų.

4.1 Kenkėjiškų mazgų skaičius Eclipse atakai

Eclipse atakos atlikimui Kademlia tinkle reikia pakeisti puolamo mazgo maršrutų lentelę. Pilnai užpildytos mazgo maršrutų lentelės turi $k \cdot n$ įrašų. Tačiau visa galima adresų erdvė yra pakankamai didelė ir niekada nebūna pilnai užpildyta, todėl tolygiai atsitiktinai pasiskirstę mazgų identifikatoriai vidutiniškai užpildo tik pirmas $log_2(m)$ mazgų maršrutų lenteles, kur m yra bendras tinkle veikiančių mazgų skaičius. Tai supaprastina Eclipse ataką, kuriai atlikti reikia pagal (6) formulę randamo kenkėjiškų mazgų skaičiaus.

$$k \cdot \log_2(m) \qquad (6)$$

4.2 Identifikatorių generavimo operacijų skaičius Eclipse atakai

Neturint galimybės laisvai pasirinkti identifikatorių, tam, kad kenkėjiškas mazgas y patektų į mazgo x maršrutų lentelę M_i, reikia, kad pirmieji y identifikatoriaus i bitai sutaptų su x ir $i+1$ bitas nesutaptų. Remiantis (1) formule gauname, kad tokį identifikatorių galima vidutiniškai sugeneruoti atliekant 2^{i+1} generavimo operacijų. Nepriklausomai nuo mazgų skaičiaus tinkle, pilnai užpildyti visas mazgo maršrutų lenteles galima atlikus (7) formulėje pateiktą identifikatorių generavimo operacijų skaičių. Formulėje (7) neįvertinama, kad paskutinėse maršrutų lentelėse negali būti k įrašų, nes nėra tiek galimų identifikatorių kombinacijų, tačiau tai sudaro nežymią paklaidą, nes tokių lentelių gali būti ne daugiau kaip $\log_2(n)=\log_2(160)\approx 7$.

$$k \cdot \sum_{i=1}^{n} 2^i = k \cdot \left(2^{n+1}-2\right) \qquad (7)$$

Kadangi mazgai niekada pilnai neužpildo visos DHT adresų erdvės galime patikslinti formulę įvertinant, vidutinį mazgo užpildytų maršrutų lentelių skaičių $log_2(m)$. Gauname formulę (8).

$$k \cdot \left(2^{\log_2(m)+1}-2\right) = 2 \cdot k \cdot (m-1) \qquad (8)$$

Tinkamų atakai identifikatorių sugeneruoti nepakanka, juos reikia įkelti į mazgo x maršrutų lentelę. Maršrutų lentelės pildomos atliekant bet kokį veiksmą su kitu mazgu (mazgo paiešką, duomenų paiešką, mazgo pasiekiamumo tikrinimą). Taip pat mazgai maršrutų lenteles rikiuoja pagal tai, kiek laiko be pertraukimo mazgai buvo pasiekiami. Todėl lentelių viršuje visada būna tie mazgai, kurie ilgiausiai dirbo be pertraukos. Sėkmingai atakai reikia, kad kenkėjiški mazgai veiktų ilgiau ir patikimiau nei visi kiti teisingai veikiantys mazgai pretenduojantys į mazgo x maršrutų lenteles. Tai sunkiausia pasiekti mazgams, kurie turi patekti į M_0 maršrutų lentelę, nes į šią lentelę patekti pretenduoja pusė visų tinkle veikiančių mazgų.

5 Išvados

Jei yra galimybė mazgų identifikatorius pasirinkti laisvai Eclipse ataką Chord tinkle įmanoma atlikti įvedant fiksuotą mazgų skaičių apskaičiuojamą pagal formulę (4). Kademlia tinkle įvedus mazgų skaičių randamą pagal (6). Jei identifikatorius privaloma generuoti abiejuose DHT modeliuose identifikatorių generavimo iteracijų skaičius tiesiogiai priklauso tik nuo tinkle esančių vartotojų skaičiaus ir apskaičiuojamas pagal formules (5), (8).

Lentelė Nr.1 Identifikatorių generavimo operacijų skaičius k=8, m=3000000 dydžio tinkle

	Laisvai parenkamas identifikatorius		**Generuojamas identifikatorius**	
	Kenkėjiškų mazgų sk.	Atakos sudėtingumas	Identifikatorių generavimo operacijų sk.	Atakos sudėtingumas
Chord	21	O(1)	21872152	O(m)
Kademlia	168	$O(\log_2(m))$	47999984	O(m)

Lentelėje Nr. 1 pateikiamas atakoms prieš konkretų tinklą atlikti reikalingas kenkėjiškų mazgų skaičius ir identifikatorių generavimo operacijų skaičius, kai tinkle yra $m=3000000$ mazgų ir Kademlia tinklo parametras $k=8$. Taip pat nurodomas atakos sudėtingumas įprastu O sudėtingumo žymėjimu. Be papildomų priemonių ar mechanizmų nuo šių atakų apsisaugoti negalima, tačiau galimybė laisvai pasirinkti mazgo identifikatorių Eclipse ataką leidžia atlikti su fiksuotu ir nuo tinklo dydžio nepriklausomu resursų kiekiu. Priklausomai nuo modelio atakų sudėtingumas keičiasi nežymiai. Prieš Kademlia modelį sunkiau atlikti atakas dėl modelyje įvesto bendro tinklo replikavimo parametro k ir modelio savybės, kad maršrutų lentelėse paliekami tik ilgiausiai tinkle dirbantys mazgai.

Mazgų identifikatorių generavimo patikrinimas nėra lengvai įgyvendinamas, tačiau įmanomas. Galimi problemos sprendimai nagrinėjami [6], [7], [8].

6 Ateities darbai

Išanalizuoti galimybes atlikti DoS atakas prieš duomenis. Išanalizuoti galimybes aptikti ir apsisaugoti nuo analizuotų DoS atakų. Išanalizuoti galimybes sulėtinti naujo mazgo prisijungimo prie tinklo arba identifikatoriaus keitimo operacijas tam kad DoS atakas būtų atlikti sunkiau.

Literatūros sąrašas

[1] **Singh A., Castro M., Druschel P.** Defending against Eclipse attacks on overlay networks. *In IEEE INFOCOM*, 2006.

[2] **Stoica I., Morris R., Karger D., Kaashoek M. F., Balakrishnan H.** Chord: A scalable peer-to-peer lookup service for Internet applications. *Proceedings of the ACM SIGCOMM'01, San Diego, CA*, 2001.

[3] **Maymounkov P., Mazieres D.** Kademlia: A peer-to-peer information system based on the XOR metric. *Proceedings of IPTPS, Cambridge, MA.* 2002, pp. 53–65.

[4] **Douceur R. J.**. The Sybil Attack. *International workshop on Peer-To-Peer Systems*. 2002.

[5] **Steiner M., En-Najjary T., Biersack E.** A Global View of KAD. *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, 2007, ISBN: 978-1-59593-908-1.

[6] **Steiner M., En-Najjary T., Biersack E.** Exploiting KAD: Possible Uses and Misuses. *ACM SIGCOMM Computer Communication Review, Volume 37 Issue 5*, 2007.

[7] **Maccari L., Rosi M., Fantacci R., Chisci L., Aiello L. M., Milanesio M.** Avoiding Eclipse attacks on Kad/Kademlia: an identity based approach. *ICC 2009 Communication and Information Systems Security Symposium*, 2009.

[8] **Baumgart I., Mies S.**. S/Kademlia: A Practicable Approach Towards Secure Key-Based Routing. *IEEE International Conference on Parallel and Distributed Systems*, 2007.

Analysis of security implications in DHT networks when choosing node id number is permitted

Not all P2P systems that implement some DHT model also implement proper mechanisms to force all nodes to generate random node ids thus allowing node id to be chosen without restrictions. In this paper we analyse the resistance of Chord and Kademlia DHT models to DoS attacks, when nodes participating in the DHT network are allowed to choose node id without any restrictions.

INFORMACINĖS TECHNOLOGIJOS 2012

17-OJI TARPUNIVERSITETINĖ MAGISTRANTŲ IR DOKTORANTŲ KONFERENCIJA

SERTIFIKATAS

Šis sertifikatas liudija, kad XVII Tarpuniversitetinės magistrantų ir doktorantų konferencijos

INFORMACINĖS TECHNOLOGIJOS 2012

Sekcijoje *Kompiuterių sistemos* išrinktas

GERIAUSIU

Autorių Julių Kuiką ir Rimantą Kavaliūną

Straipsnis „Laisvai pasirenkamo maršo identifikatoriaus įtakos DHT tinklo saugumui analizė"

Konferencijos pirmininkas doc.dr. A.Lopata

Vilniaus universiteto Kauno humanitarinis fakultetas
Kaunas
2012