Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

Cena obranných mechanizmov anonymizačných protokolov

Diplomová práca

2018
Bc. Tomáš Wiedermann

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

Cena obranných mechanizmov anonymizačných protokolov

Diplomová práca

Študijný program:	Informatika
Študijný odbor:	2508 Informatika
Školiace pracovisko:	Katedra informatiky
Školiteľ:	RNDr. Jaroslav Janáček, PhD.

Bratislava, 2018
Bc. Tomáš Wiedermann

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Bc. Tomáš Wiedermann
Študijný program: informatika (Jednoodborové štúdium, magisterský II. st., denná forma)
Študijný odbor: informatika
Typ záverečnej práce: diplomová
Jazyk záverečnej práce: slovenský
Sekundárny jazyk: anglický

Názov: Cena obranných mechanizmov anonymizačných protokolov
Defence Mechanizms Cost in Anonymization Protocols

Cieľ: Cieľom práce je preskúmať vybrané typy útokov zameraných na narušenie anonymity poskytovanej vybranými anonymizačnými sieťovými protokolmi, metódy obrany proti týmto útokom, ich účinnosť a vplyv na záťaž siete. Práca bude zameraná najmä na techniku onion routing použitú v anonymizačnej sieti Tor a jej obmenu garlic routing a na útoky využívajúce informácie o sieťovej prevádzke.

Vedúci: RNDr. Jaroslav Janáček, PhD.
Katedra: FMFI.KI - Katedra informatiky
Vedúci katedry: prof. RNDr. Martin Škoviera, PhD.

Spôsob sprístupnenia elektronickej verzie práce:
bez obmedzenia

Dátum zadania: 08.12.2016

Dátum schválenia: 08.12.2016 prof. RNDr. Rastislav Kráľovič, PhD.
garant študijného programu

.....
študent

.....
vedúci práce

Abstrakt

Diplomová práca je venovaná anonymnej komunikácii v počítačových sieťach. Práca obsahuje úvod do problematiky a stručný súhrn doterajších poznatkov. V práci sa pojednáva hlavne o dvoch známych typoch útokov na deanonymizáciu používateľa v systéme Tor a ich možné použite na menej známy a skúmaný systém I2P. Ďalším cieľom sú neoverené techniky, ktoré I2P využiva. Konkrétne sú to jednosmerné tunely a využitie P2P. V práci sú popísané postupy pri vytváraní potrebných dát, ich následné spracovanie a získanie výsledkov potrebných na vyhodnotenie. Hlavnou snahou bolo využitie existujúcich systémov na zabezpečenie čo najpresnejších výsledkov.

Kľúčové slová: anonymná komunikácia, Tor, I2P

Abstract

Master thesis is dedicated to anonymous communication in computer networks. Thesis contains introduction to topic and brief summary of previous results. Mostly two of the presented attacks to deanonymize user in system Tor are taken in account and their potential usage in less known and used system I2P is considered. Next goal is to look into untested techniques, which I2P uses. Namely one way tunnels and usage of P2P. Thesis contains description of approaches used in dataset generation, data processing and gaining results needed for evaluation. Main goal is to use existing systems to ensure as accurate results as possible.

Keywords: anonymous communication, Tor, I2P

Obsah

Úvod **1**

1 Anonymná komunikácia **2**
1.1 DC-net . . . 3
1.1.1 Využitie v počítačových sieťach . . . 4
1.2 MIX-net . . . 6
1.2.1 Mixovacie algoritmy . . . 9
1.2.2 Voľba cesty . . . 10
1.2.3 Útoky . . . 10
1.2.4 Obranné mechanizmy . . . 12
1.3 Náhodná cesta používateľmi . . . 13
1.4 Onion routing . . . 14
1.4.1 Tor . . . 14
1.5 Garlic routing . . . 18
1.5.1 I2P . . . 18

2 Ciele práce **22**

3 Útok koreláciou **24**
3.1 Vytvorenie dát . . . 24
3.2 Spracovanie dát . . . 26
3.3 Výsledky . . . 30
3.3.1 Jednosmerné tunely . . . 37
3.3.2 P2P . . . 38

4 Útok s preťažením **41**
4.1 Infraštruktúra a nástroje . . . 41
4.2 Spracovanie dát . . . 44
4.3 Výsledky . . . 45

Záver **47**

A Prílohy **56**

Zoznam obrázkov

1.1 Ukážka DC-net 4
1.2 Únik informácie v DC-net 5
1.3 Mix-net schéma s jedným serverom 8
1.4 Mix-net schéma s viacerými servermi 8
1.5 Mix-net schéma so spätnými adresami 8
1.6 Príklad stavby reťaze a komunikácie pre reťaz dĺžky dva 16

3.1 Upravený Pearsonov korelačný koeficient 29
3.2 Veľkosť prenesených dát 39
3.3 Úspešnosť určenia v závislosti od korelačného faktoru 40
3.4 Úspešnosť určenia v závislosti od korelačného faktoru - spätná komunikácia 40

4.1 Rýchlosti odpovedí monitorovaných uzlov pre parametre: 500kB, 100kB, 2s 46
4.2 Rýchlosti odpovedí monitorovaných uzlov pre parametre: 300kB, 25kB, 1s 46

Úvod

Diplomová práca je zameraná na vybrané časti z oblasti anonymnej komunikácie. S šírením cenzúry a sledovania používateľov na internete sa téma stala veľmi populárnou. V priebehu posledných dvoch desaťročí bolo prezentovaných mnoho techník a možností na anonymizáciu komunikácie, ale taktiež aj mnoho útokov, ktorých cieľom je odhaliť komunikujúcu dvojicu či skupinu. Vzniklo aj viacero implementácií mnohých z týchto techník, ale postupom času prevažná väčsina zanikla hlavne z dôvodu nedostatku používateľov. Jednou z mála, čo sa v praxi udržala a zároveň dnes najpoužívanejšou implementáciou je systém Tor. Vzhľadom na popularitu tohoto systému bol výskum najviac zameraný práve naň. Nám bude slúžiť ako základ na porovnanie s menej známym systémom I2P, ktorý používa viacero odlišných a často neoverených prístupov. Preto cieľom práce je vyhodnotiť útoky prezentované na systém Tor aj pre systém I2P a preskúmať neoverené techniky, ktoré používa. Keďže komunikácia na internete je veľmi rôznorodá, práca je zameraná na najbežnejšie využitie, čo je web.

Práca je rozdelená na štyri kapitoly. Prvá kapitola obsahuje popis problému anonymnej komunikácie, existujúce riešenia a prehľad doterajšieho výskumu v tejto téme. Druhá kapitola podrobnejšie popisuje ciele práce. Tretia kapitola popisuje postup získania dát, ich spracovanie a výsledky korelačných útokov na systémy Tor a I2P. Tiež uvádza výsledky využívania spätnej komunikácie pri týchto útokoch a využitie zapojenia sa do P2P na skrytie vlastnej komunikácie. Štvrtá a posledná kapitola pojednáva o útoku v minulosti prezentovanom na systém Tor a skúša jeho aplikovateľnosť na systém I2P.

Kapitola 1

Anonymná komunikácia

Diplomová práca pojednáva o probléme **anonymnej komunikácie** v počítačových sieťach. Je veľmi drahé až nemožné vytvárať bezpečné komunikačné kanály, ktoré by nebolo možné odpočúvať. V internete môže byť každá linka odpočúvaná a manipulovaná napríklad poskytovateľom pripojenia, štátnym orgánom alebo iným útočníkom. Klasické kryptografické prostriedky dokážu skryť prenášané dáta, zabezpečiť integritu dát a s pomocou dôveryhodnej autority zabezpečiť aj autentickosť. Nedokážu však skryť, že komunikácia medzi dvomi alebo viacerými konkrétnymi účastníkmi prebieha. Jedná sa o stále aktívny odbor, sú prezentované nové útoky a navrhované obrany voči nim. Uvažujú sa rôzne modely z pohľadu sledovania liniek, výpočtovej sily útočníka a ovládnutia častí anonymizačného systému. Spravidla sa však predpokladá, že útočník nevie efektívne prelomiť kryptografické prostriedky a ani v nich nepozná žiadne nepublikované zraniteľnosti.
Rôzne anonymizačné schémy môžu poskytovať anonymitu v inom zmysle. Vždy sa v prvom rade uvažuje **anonymita odosielateľa** a **nespojiteľnosť odosielateľa a prijímateľa**. Je veľmi časté aj poskytovanie anonymity pre prijímateľa no zriedka je snaha skryť aj samotnú aktivitu odosielateľa či prijímateľa. Aj keď používame pojem anonymita, správne sa má používať pojem **pseudonymita**, keďže používateľ je anonymný len v rámci systému a nie v celej sieti. Tieto pojmy sa pre jednoduchosť zjednocujú a rovnako tomu bude aj v tejto práci.

Možnosť anonymnej komunikácie prináša aj etické otázky, hlavne či je anonymná sieťová komunikácia potrebná a či nenapomáha kriminálnej činnosti. Naopak možnosť anonymnej komunikácie má prínos pre bežného používateľa, ktorý tak získa možnosť ochrany súkromia a obídenie cenzúry informácií, ktorú zavádzajú niektoré štáty.

V tejto kapitole ešte uvedieme stručný prehľad doterajších poznatkov. Časti, ktorých sa práca týka, popíšeme obšírnejšie, no veľa vecí len spomenieme a uvedieme refe-

renciu. Veľmi dobrým zdrojom je zbierka výskumných prác na Free Haven Project[**?**], z ktorých bola čerpaná väčšina informácií.

1.1 DC-net

DC-net je skratka z anglického `Dining Cryptographers`, voľne preložené ako večerajúci kryptológovia. Názov je odvodený z problému troch večerajúcich kryptológov, na ktorom je schéma ilustrovaná. Síce sa jedná o druhú schému publikovanú Davidom Chaumom [10], ale popíšeme ju ako prvú, lebo nasledujúce schémy budú mať viac spoločných znakov. Naviac táto schéma ponúka najlepšie záruky anonymity, zatiaľ čo v ďalších budú klesať. Na druhú stranu je táto schéma nepraktická a jej implementácia[33] sa v praxi neudržala.

Problém **troch večerajúcich kryptológov** vyzerá nasledovne:
Traja kryptológovia sú na večeri v reštaurácii. Keď si vypýtajú účet, čašníčka im povie že už bol zaplatený. Všetci rešpektujú možnosť anonymnej platby, ale chcú vedieť, či zaplatil niekto z nich alebo za nich zaplatila NSA. Preto vymyslia spôsob akým to overiť. Každý s každým si hodí mincou tak, aby to tretí nevidel. Potom každý vyhlási, či sa obe mince ktoré videl zhodujú alebo nie. V prípade ak platil, tak svoje tvrdenie zneguje. Nepárny počet nezhôd znamená, že platil niekto z nich a párny počet nezhôd znamená že platila NSA.
Ľahko nahliadnuť, že každá operácia v schéme je **XOR** (tiež zvané sčítanie modulo dva, označenie $\oplus$). Príklad tejto schémy môžeme vidieť na obrázku 1.1. V prípade keď nikto neplatil, je zápis nasledujúci.

$$(x_1 \oplus x_2) \oplus (x_2 \oplus x_3) \oplus (x_3 \oplus x_1)$$

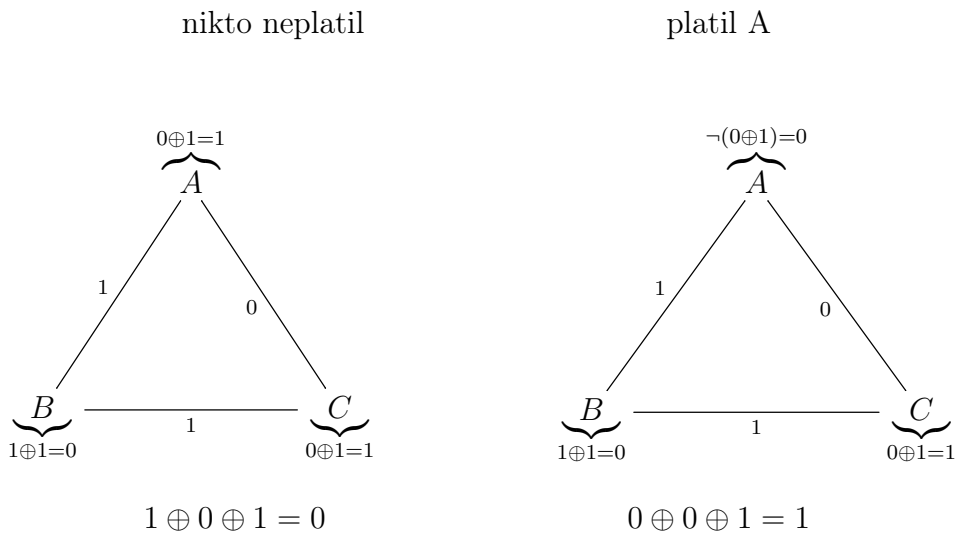
XOR je **komutatívna** ($A \oplus B \Leftrightarrow B \oplus A$) a **asociatívna** ($(A \oplus B) \oplus C \Leftrightarrow A \oplus (B \oplus C)$)) operácia. Pre ľubovoľné x_i platí $x_i \oplus x_i = 0$ a každé x_i je započítané práve dva krát. Tým pádom je výsledok 0. V prípade ak platil práve jeden

$$(x_1 \oplus x_2) \oplus \neg(x_2 \oplus x_3) \oplus (x_3 \oplus x_1) \Leftrightarrow \neg(x_2 \oplus x_3) \oplus (x_2 \oplus x_3) \Leftrightarrow 1$$

Ak by ale platili dvaja dostávame

$$\underbrace{\neg(x_1 \oplus x_2) \oplus \neg(x_2 \oplus x_3)}_{\Leftrightarrow (x_1 \oplus x_3)} \oplus (x_3 \oplus x_1) \Leftrightarrow 0$$

Ľahko nahliadnuť, že táto schéma sa dá rozšíriť na ľubovoľný počet a pravidlá zostanú zachované. Teda pri nepárnom počte negácií je výsledok 1 a pri párnom 0.



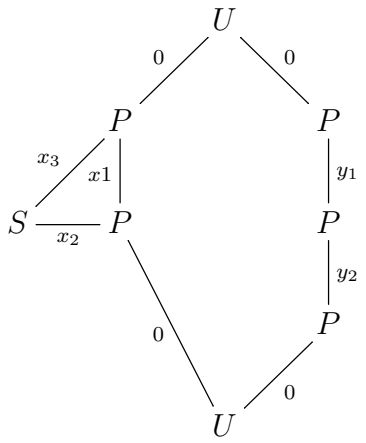
Obr. 1.1: Ukážka DC-net

1.1.1 Využitie v počítačových sieťach

Tento princíp je prenositeľný do počítačových sietí na anonymizáciu komunikácie. Každý používateľ s každým iným používateľom zdieľa dostatočne dlhý kľúč. Keď odosiela správu, tak spočíta XOR zo všetkých jeho kľúčov a v prípade, ak chce odoslať na danom mieste v správe jednotkový bit, tak výstup ešte zneguje. Ak je odosielaný bit v správe nulový, tak ho nemení. Ostatní používatelia len spočítajú XOR a publikujú svoju správu. Všetky publikované správy sú sčítané operáciou XOR a výsledkom je pôvodná správa odosielateľa. Tým pádom, každý používateľ musí s každým používateľom vopred zdieľať dostatočne dlhý kľúč, aby bolo možné odoslať správu. Toto nepredstavuje veľký problém vďaka kryptografickým generátorom pseudonáhodných postupností, kde je potrebná len dohoda medzi používateľmi pri inicializácii a ďalej nie je potrebné si vymieňať ďalšie správy. Dokonca nie je potrebné, aby každý používateľ zdieľal kľúč s každým. Ak si zdieľanie kľúčov predstavíme ako graf, kde používateľ predstavuje vrchol a zdieľaný kľúč hranu, tak požadujeme, aby medzi každým párom vrcholov existovali aspoň dve vrcholovo rozdielne cesty. Takýto graf nemá **artikuláciu** (vrchol, ktorého odobratím by graf prestal byť súvislý) a zachováva vlastnosť, že *žiadny používateľ nevie o ostatných získať žiadnu informáciu.* Problém nastáva, keď je v sieti viacero útočníkov a tvoria množinu, ktorej odobratím graf prestane byť súvislým. Z hľadiska anonymity je preto najlepší úplný graf, zatiaľ čo z hľadiska efektivity je najvýhodnejší kruhový graf.

Ako príklad uniknutia informácie z útočníkmi rozdeleného grafu uvedieme osem používateľov z ktorých dvaja sú útočníci a majú medzi sebou na oboch stranách troch rôzne zoskupených používateľov. 1.2 Pre jednoduchosť príkladu stanovíme útočníkom (označený U) zdieľané kľúče na 0 a kľúče medzi používateľmi (označený P) a odosielateľom (označený S) na neznáme x_i a y_i podľa strany. Je možné zhliadnuť, že pravá strana je rovná $y_1 \oplus (y_1 \oplus y_2) \oplus y_2 = 0$, teda pravá strana nevysiela, pričom ľavá strana $\neg(x_2 \oplus x_3) \oplus (x_3 \oplus x_1) \oplus (x_1 \oplus x_2) = 1$, teda vysielaná správa je z ľavej strany. Týmto sme potencionálnych odosielateľov rozdelili na dve skupiny. Je jednoduché nahliadnuť, že rovnaký princíp bude platiť pre ľubovoľný graf, ktorý rozbijeme útočníkmi na jednotlivé komponenty.

Obr. 1.2: Únik informácie v DC-net



V ľubovoľnej topológii zdieľania kľúčov stále platí, že na každú správu od jedného používateľa sa musí skladať každý iný používateľ v skupine. Preto sú potrebné kolá, v ktorých je nutné odoslať správu, či už s vlastnými dátami alebo len doplnenie pre korektný celkový výpočet. Problémom sú taktiež kolízie správ, keď chce viacero používateľov odosielať správu naraz. Je potrebné kolízie detegovať a zabrániť odoslaniu poškodenej správy. Vhodným postupom pri detekcii je použiť klasický prístup zo sietí a počkať náhodný počet kôl.

Odpovede na odoslanú správu nemajú presnú adresu a preto je potrebné ich doručiť všetkým používateľom v rámci skupiny. Takto je každý používateľ nútený zdieľať zaťaženie linky komunikáciami všetkých používateľov.

Doteraz sme uvažovali len útočníka, ktorý sa snažil útočiť proti anonymite a nie proti systému. Nech sa jedná o ľubovoľnú topológiu zdieľaných kľúčov a ľubovoľný počet používateľov. Vždy je aj jeden používateľ schopný narušiť celú komunikáciu tým, že nedodržiava protokol. Buď sa snaží v každom kole odoslať správu alebo zámerne robí chyby pri výpočte XOR hodnoty z kľúčov.
Pôvodne bol problém riešený za predpokladu, že v jednom odosielanom bloku je viacero správ a prvá správa v bloku slúži na rezerváciu správ v nasledujúcom bloku. Každý používateľ môže znegovať práve jeden bit na niektorej z pozícií a po výslednom zložení správ si vie overiť, ktoré časti sú v nasledujúcom bloku rezervované a dokáže si aj svoju rezerváciu overiť. Vzhľadom na narodeninový paradox, by mal byť počet rezervačných blokov kvadratický k počtu používateľov. V takom prípade by mal rezervačný blok mať približne rovnaký počet jednotiek ako používateľov, čo by mohlo viesť k detekcii narušenia.
Rezervovaná správa môže byť využitá na detekciu poškodenia tak, že je do nej uložená zašifrovaná náhodná správa a pozícia v bloku. Pri jej narušení je kľúč zverejnený a používatelia vyzvaný na poskytnutie kľúča, ktorým prispeli k danej správe. Týmto spôsobom je možné získať podozrenie o ktorého používateľa sa jedná. Táto technika odhalenia vyžaduje posielanie správ medzi všetkými účastníkmi a v neskorších prácach[75] [36] bola snaha o zjednodušenie odhalenia podvádzajúceho používateľa.

Schéma DC-net poskytuje veľmi dobré záruky anonymity. Keďže výsledná správa je vyskladaná z mnohých správ od jednotlivých používateľov, vonkajší útočník nemá k dispozícii žiaden možný útok. Jediná možnosť napadnutia schémy je zvnútra a pri dostatočne hustom grafe je potrebný vysoký počet útočníkov. Pri úplnom grafe až všetkých okrem sledovaného používateľa. Nevýhodou je veľké množstvo prenesených dát na odoslanie alebo prijatie správy a problémy s podvádzajúcimi používateľmi.

1.2 MIX-net

Prvý princíp pre anonymnú komunikáciu vznikol už začiatkom 80-tych rokov[9] a bol skôr myslený pre mailovú komunikáciu, ale postupne sa rozšíril aj do ostatných častí. Napríklad jedna z prvých implementácií bola v telefónnych sieťach na poskytnutie anonymných hovorov[60]. Až neskôr vznikli implementácie pre pôvodne zamýšľanú mailovú [37][50][15] alebo všeobecnú sieťovú komunikáciu[63][5]. Okolo tejto témy bolo vedených veľa výskumov, hlavne v oblasti obrany, lebo všeobecne schéma uvažuje a snaží sa brániť proti veľmi silnému modelu aktívneho globálneho útočníka. Takýto útočník môže odpočúvať a manipulovať so všetkými linkami. Posledné roky sa už táto

oblasť neskúma a nerozvíja, lebo sa prešlo na novšie princípy. Tie síce neuvažujú tak silný model útočníka, ale sú jednoduchšie a teda aj praktickejšie.

Najprv popíšeme pôvodný koncept[9] a následne rôzne obmeny a vylepšenia. Pri posielaní správy odosielateľ správu najprv doplní o náhodnú časť, následne zašifruje verejným kľúčom príjemcu a k šifrovanej časti pridá adresu príjemcu. K tejto správe opäť pridá náhodnú časť, zašifruje ju verejným kľúčom mix servera a pridá adresu mix servera. Takáto správa je odoslaná mix serveru, kde sa odšifruje, náhodná časť sa zahodí a správa sa uloží. Keď nastane splnenie podmienky mixovacieho algoritmu, napríklad v pôvodnom koncepte počet uložených správ, tak sú všetky odoslané v náhodnom poradí príjemcom. Tí po odšifrovaní získajú pôvodnú správu bez znalosti o odosielateľovi 1.3. Mix server udržiava históriu prijatých správ, aby sa predišlo útoku zopakovaním správy, kde útočníkovi stačí odchytenú správu zopakovať aby videl, ktorý výstup sa zopakuje. Ideálne históriu udržiava pokým nezmení verejný kľúč.
Mix server nemusí byť na ceste od odosielateľa k príjemcovi iba jeden, Môžu tvoriť postupnú reťaz, kde každý server odšifruje jednu vrstvu, odstráni náhodnú časť a správu 'premieša' s ostatnými 1.4. Ďalej v podkapitole budeme pre jednoduchosť uvádzať len jeden server, aj keď sa tým môže myslieť celá reťaz.
Koncept umožňuje aj spätné adresy, teda príjemca vie odpovedať odosielateľovi správy bez toho, aby vedel jeho identitu 1.5. Keď odosielateľ chce umožniť príjemcovi odpovedať, pribalí k správe vytvorený kľúč, ktorým príjemca bude šifrovať správu a časť šifrovanú verejným kľúčom mix servera. Tá obsahuje odosielateľovu adresu a ďalší kľúč, ktorý použije mix server pri posielaní správy späť a zároveň slúži aj ako náhodný reťazec. V takomto prípade je spätná adresa šifrovaná kľúčmi mix servera, príjemcu a opäť mix servera. Takže musí prejsť určenú cestu a keďže je správa na spätnej ceste šifrovaná odosielateľom vytvorenými kľúčmi, iba on je schopný správu rozšifrovať. Pravdaže aj spätná reťaz môže mať ľubovoľnú dĺžku, ktorú určuje odosielateľ.
Toto predstavuje aj možnosť overenia doručenia správy, preto je koncept vhodný aj na elektronické voľby.

- MX - adresa mix servera
- B - adresa príjemcu
- A - adresa odosielateľa
- K_{MX} - verejný kľúč mix servera
- K_B - verejný kľúč príjemcu
- R - náhodná časť
- K_P - kľúč generovaný odosielateľom pre príjemcu
- K_R - kľúč generovaný odosielateľom pre mix server
- Spr - správa pre príjemcu

Obr. 1.3: Mix-net schéma s jedným serverom

$$A \rightarrow MX \rightarrow B$$
$$K_{MX}(R_1, B, K_B(R_2, Spr)) \rightarrow K_B(R_2, Spr)$$

Obr. 1.4: Mix-net schéma s viacerými servermi

$$A \rightarrow MX_1 \rightarrow \cdots \rightarrow MX_n \rightarrow B$$
$$K_{MX_1}(R_1, MX_2, K_{MX_2}(\ldots(R_n, MX_n, K_{MX_n}(R_n, B, K_B(R_{n+1}, Spr))\ldots))) \rightarrow$$
$$K_{MX_2}(\ldots(R_n, MX_n, K_{MX_n}(R_n, B, K_B(R_{n+1}, Spr))\ldots)) \rightarrow \ldots$$
$$\cdots \rightarrow K_{MX_n}(R_n, B, K_B(R_{n+1}, Spr)) \rightarrow K_B(R_{n+1}, Spr)$$

Obr. 1.5: Mix-net schéma so spätnými adresami

$$A \rightarrow MX \rightarrow B$$
$$K_{MX}(R_1, B, K_B(R_2, Spr, K_{MX}(A, K_R), K_P)) \rightarrow K_B(R_2, Spr, K_{MX}(A, K_R), K_P)$$
$$B \rightarrow MX \rightarrow A$$
$$K_{MX}(A, K_R), K_P(R1, Spr) \rightarrow K_R(K_P(R1, Spr))$$

1.2.1 Mixovacie algoritmy

Najpodstatnejšou súčasťou MIX-net systému je použitý mixovací algoritmus[19][20]. To je algoritmus, ktorý je použitý na odoslanie prijatých správ tak, aby čo najlepšie skryl súvislosti medzi prijatými a odoslanými správami. Mnohé mixovacie algoritmy sa môžu zdať až zbytočne zložité, ale práve tie lepšie odolávajú útoku `n-1`, ktorý spomenieme v nasledujúcej podkapitole.

Spoločným znakom všetkých mixovacích algoritmov je, že čakajú na splnenie podmienky. Pôvodný odosielací algoritmus ukladá prijaté správy a čaká na určený počet uložených správ. Po prijatí daného počtu všetky správy odošle v náhodnom poradí. Problémom tohoto algoritmu je, že nemá žiadne záruky na čas doručenia správy a môže sa stať, že dlhú dobu čaká aj na jedinú správu.
Ďalší jednoduchý odosielací algoritmus ako podmienku využíva čas. Po stanovenú dobu ukladá prijaté správy a po uplynutí časového limitu odošle všetky prijaté správy. Takýto algoritmus poskytuje záruku doručenia správy. Negatívnym následkom je prípad, keď je uložených málo správ, v okrajovom prípade iba jedna. Z tohoto dôvodu je zisk anonymity neurčitý.
Je možné aj oba prístupy skombinovať a čakať na splnenie jednej podmienky. Pri slabom zaťažení je správanie podobné časovanému algoritmu a pri vyššom zaťažení sa správa podobne algoritmu s hranicou počtu správ. Možnosťou je aj čakať na splnenie oboch požiadaviek tak, že sa server periodicky pokúša o odoslanie, ale odošle len ak má dostatočný počet.
Doteraz sme uvažovali, že správy sa posielajú v jednom kole všetky, preto prejdeme k algoritmom, ktoré si správy udržiavajú v pamäti aj medzi jednotlivými kolami. Opäť sa používajú predchádzajúce podmienky. Napríklad po dosiahnutí určitého počtu správ sa odošle presný počet náhodne zvolených správ z celého počtu. Pri podmienke času sa môže odosielať zlomok zo všetkých uložených správ. Možnosťou je aj každú správu odoslať s určitou pravdepodobnosťou. Takto sa môžu počty odoslaných správ medzi jednotlivými kolami líšiť, ale z dlhodobého hľadiska sa blížia danej pravdepodobnosti. Nevýhodou zostáva, že je ešte zložitejšie predpovedať čas prechodu správy anonymizačným systémom, keďže správy sú vyberané náhodne.
Alternatívou, ktorá poskytuje záruky ohľadom času doručenia, je každú správu spracovávať zvlášť a rôzne ju pozdržať. Pričom pozdržanie môže určiť používateľ. Buď relatívne, teda určí koľko má byť správa v danom bode zdržaná[23], alebo absolútne, čiže určí časové okno, v ktorom má správa doraziť a na konci ktorého má byť odoslaná. Ak príde mimo časového okna je zahodená.[43] Takýmto spôsobom má používateľ kontrolu nad pomerom anonymity a rýchlosti.

1.2.2 Voľba cesty

Pri stavbe kompletného systému využívajúceho Mix-net schému (alebo inú schému s reťazou preposielajúcich serverov) je dôležité, akým spôsobom si môže používateľ voliť cestu pre správu. Často pripadajú do úvahy dve alebo viac možností, z ktorých každá je lepšia v inom prípade. Úplným základom je poskytovanie serverov, ktoré môžu byť buď len priamo od poskytovateľov systému, alebo môže byť umožnené tretej strane prevádzkovať vlastné servery. Treťou možnosťou sú peer-to-peer siete (siete bez centrálnej autority, označovať budeme skrátene P2P), kde je každý používateľ zároveň aj server.
Ďalší základný atribút je určovanie cesty, kde si používateľ vyberá buď z pevne určených ciest alebo si z množiny serverov vytvorí ľubovoľnú cestu sám. Teda aj určenie dĺžky je na používateľovi. Obdive možnosti majú svoje výhody a nevýhody[6] a treba prihliadať aj na použitý odosielací algoritmus[24], spoľahlivosť jednotlivých serverov[25], počet používateľov v systéme a počet preposielacích serverov. Napríklad pri nízkom počte používateľov je lepšie použiť pevné cesty, aby boli komunikácie lepšie premiešané.[66]
Potom treba určiť, či používateľ vidí všetky možné preposielacie servery alebo len určitú časť[54]. Vyberá si z nich úplne náhodne alebo na základe ohodnotenia. Je ohodnotenie statické, teda server publikuje svoje parametre alebo sa dynamicky vyhodnocujú na základe používateľovej spätnej väzby[53]. Hodnotí sa kvalita pripojenia servera k sieti alebo kvalita linky medzi každou dvojicou serverov[67].
Možností prístupu k voľbe cesty je vskutku veľa a zlá kombinácia môže viesť k prípadným nedostatkom v anonymite systému.

1.2.3 Útoky

Najskôr spomenieme možné útoky v danom modeli, až potom sa budeme venovať dodatočným obranným mechanizmom. Útoky bočnými kanálmi, napríklad vloženie škodlivého kódu do webovej stránky, nebudeme uvažovať a zameriame sa skôr na všeobecné útoky na schému.

Disclosure attack

Na úvod len spomenieme anglicky nazvaný disclosure attack (neexistuje slovenský ekvivalent, budeme používať anglický názov)[42]. Útok je zameraný na jednu konkrétnu osobu a nezáleží aká anonymizačná schéma je medzi odosielateľom a príjemcom. Útočník v tomto útoku pozoruje celú sieť a vždy, keď sledovaný používateľ odošle správu, útočník zaznamenáva potencionálnych príjemcov. Útok taktiež predpokladá konkrétny

počet komunikačných partnerov so sledovanou osobou, počet označíme m. Útočník čaká, kým nebude mať m množín, ktoré medzi sebou nemajú žiaden prienik. Takže v každej množine musí byť len jeden komunikačný partner so sledovaným používateľom. Následne ďalším sledovaním redukuje množiny tak, že vždy keď nová množina má prienik s práve jednou z pôvodných, tak ju nahradí prienikom pôvodnej a novej. Takto pokračuje pokým nie je v každej z pôvodných množín práve jeden príjemca.
Tento útok je NP-úplný[1] a aj keď boli publikované vylepšenia [13][14][72][59][17], tak tento útok budeme naďalej považovať za príliš zložitý na reálne použitie.

Pasívne útoky s využitím informácií o prenose

Pasívne útoky proti anonymizačným schémam využívajú zaznamenané informácie o paketoch, ktoré sa s prechodom anonymizačným systémom nemenia alebo sa menia len málo. Hlavne sa jedná o systémy, ktoré sa snažia byť čo najefektívnejšie z hľadiska rýchlosti odozvy a rýchlosti prenosu dát. Jedná sa o počet prenesených paketov, ich veľkosti, časy medzi paketmi, čas prechodu časťou siete a pod. Užitočné môžu byť aj časy otvorenia a uzavretia spojenia.
Napríklad ak systém využíva pevné cesty, ktoré majú dostatočne rozdielne časy prechodu a útočník vidí v akých časoch správy do anonymizačného systému vchádzajú a vychádzajú, tak vie určiť ktorú cestu používateľ použil.[61] Ak systém nepoužíva žiadne obranné techniky meniace veľkosť prenesených dát, tak jednoduchým počítaním paketov útočník vie získať informáciu, hlavne v slabo zaťaženom systéme[66].
Často sa útoky snažia korelovať časy medzi paketmi pred vstupom a po výstupe buď z celej anonymizačnej siete alebo jedného MIX servera[45]. Keďže Mix-net schéma mieša správy a čaká na splnenie požiadavky, tento útok je sťažený. Ak vezmeme do úvahy aj veľkosti prenesených dát v časových intervaloch, dostávame známy útok využívajúci koreláciu toku dát. Pri Mix-net schéme nie je použitie úplne priamočiare a je najprv potrebné analyzovať, ako jednotlivé odosielacie algoritmy menia tok dát[79].
Ak budeme uvažovať aj aktívneho útočníka, tak je možné predchádzajúci útok zlepšiť generovaním výraznejšieho vzoru toku dát. Všeobecne dobrá stratégia je nejakú dobu pakety pozdržať a následne v rýchlom slede odoslať.

Ovládnutie reťaze

Ak systém umožňuje používateľovi byť súčasťou preposielania, môže sa útočník pokúsiť zahltiť anonymizačnú sieť svojimi preposielacími servermi a dostať tak pod kontrolu celú reťaz obete[77]. V P2P sieťach môže jeden používateľ vystupovať pod viacerými pseudonymami a zvyšovať tak svoje šance na ovládnutie[26]. Napríklad v anonymizačnom systéme I2P, o ktorom budeme hovoriť neskôr, bol prezentovaný útok, ktorý

využíval, že používateľ vidí len časť preposielacích serverov a snažil sa získať rovnaký obraz o sieti ako obeť. Následne preťažil tie, ktoré nemal pod kontrolou, aby sa používateľovi nepodarilo s nimi dohodnúť a zvyšoval tým svoje šance[38].
Princíp, keď používateľ má len čiastočný prehľad o anonymizačnej sieti sa viackrát ukázal byť problematický[34][16], ale v P2P sieťach je zložité získať celkový prehľad, keďže neexistuje centrálna autorita.

n-1

Útok zvaný `n-1` alebo aj flush či flood je aktívny útok na deanonymizovanie jednej správy v jednom mix serveri. Princíp je veľmi jednoduchý, keď chce útočník sledovať jednu správu, tak v prvom kroku zahltí mix server svojimi správami, aby ho vyprázdnil. Práca potrebná na vyprázdnenie závisí od odosielacieho algoritmu. V tomto čase pravdaže všetky ostatné blokuje. Následne do servera nechá prísť práve jednu správu, ktorú chce sledovať a opäť svoje, ktoré vie rozoznať, keďže ich vytvoril. Svoje správy posiela kým neuvidí jednu, ktorú chcel sledovať.
Tento útok je dôvodom, prečo boli vytvorené komplikovanejšie odosielacie algoritmy. Napriek tomu je útok použiteľný voči všetkým spomenutým odosielacím algoritmom, akurát útok zťažuje. S použitím rôznych foriem obrán sa útok ešte viac komplikuje alebo úplne stráca účinok.

1.2.4 Obranné mechanizmy

Na zvýšenie odolnosti anonymizačného systému je možné použiť dodatočné obranné mechanizmy. Ich úlohou je zakryť informácie využiteľné pri útokoch spomenutých v predchádzajúcej kapitole. Najčastejšie uvažované je využitie falošných správ, ktoré sa dajú rozoznať od naozajstných až pri odšifrovaní príjemcom. Opäť existuje viacero možných prístupov. Takéto správy môžu byť generované používateľmi alebo preposielajúcimi servermi. Taktiež končiť môžu u používateľa, ktorý ich obsah jednoducho zahodí alebo v inom preposielacom serveri.

Napríklad vytvorenie a odoslanie niekoľkých správ v každom odosielacom kole zabráni útoku `n-1`. Keby bol počet falošných správ v každom serveri päť a dĺžka cesty by bola tri, tak aj keby útočník sledoval jednu vstupnú správu a postupne by sledoval všetky možné výstupy použitím `n-1`, tak by v poslednom kole sledoval viac ako 5^3 serverov. Problémom je, že server nevie o prijatej správe povedať, či je falošná a tak preposiela falošné čo prijal a tie, ktoré sám vytvoril. Čím by boli v takomto systéme dlhšie cesty, tým by bol väčší počet vygenerovaných falošných správ.
Jedným z prvých konceptov bolo nastavenie každej komunikujúcej linky na dohodnutú

konštantnú rýchlosť a neustále prenášanie dát[12][32], či už ide o reálne alebo len vytvorené na zaplnenie. Takýto spôsob je ale neefektívny[64], keďže linka musí byť rovnako zaťažená po celú dobu a nemôže v prípade potreby rýchlosť prenosu prekročiť.
Na obranu útoku `n-1` je možné použiť falošné správy so začiatkom a koncom v rovnakom preposielacom serveri, ktorý si takto overuje, či nie je s niektorým spojením manipulované. Až keď zdeteguje zreteľnú zmenu v čase prechodu jeho kontrolnej správy, tak začne používať dodatočné obrany.[19]

Proti útokoch koreláciou, ktoré sme tiež uviedli v predchádzajúcej podkapitole, sú obrany konštruované inak. Cieľom je čo najviac zmeniť vzor toku dát alebo zabezpečiť aby boli vzory čo najviac uniformné. Tento problém sa netýka len Mix-net schémy, ale veľkej časti vysokorýchlostných anonymizačných systémov a preto bude možné využiť rovnaké obrany aj v nich. Jednou z možností je, že používateľ vytvára pakety, v ktorých je skrytá informácie pre preposielací server, či má paket zahodiť. Takto dokáže používateľ ovplyvniť svoj vzor toku dát a dostatočne ho pozmeniť[45].
Ďalšou z možností je nastaviť každý preposielací server aby generoval rovnaký vzor na každej svojej linke tým, že posiela po všetkých linkách súčasne a vždy len keď má pre každú paket.[79] Pre vyhnutie sa problémom s málo aktívnou linkou, ktorá by brzdila ostatné, je možné vyžadovať pakety len pre určitý počet liniek a na ostatných odoslať predpripravený falošný paket.[64]
Ešte jednou možnosťou je prispôsobovať vzor inému získanému zo štatistiky, kde v prípade, že paket príde skôr ako je očakávané, tak je pozdržaný. Ak je očakávané odoslanie, tak sa pošle falošný.[68]

Nájsť optimum medzi praktickosťou a bezpečnosťou pri falošných správach je problematické a vždy je potrebné prihliadať aj na použitý mixovací algoritmus. [18][19][69]

1.3 Náhodná cesta používateľmi

Jeden z konceptov využívajúcich používateľov na anonymizáciu je koncept náhodnej cesty medzi používateľmi. V rámci skupiny každý používateľ s každým zdieľa symetrický kľúč. Keď chce používateľ odoslať správu, náhodne zvolí iného a šifrovane mu ju prepošle. Používateľ, ktorý správu prijme si 'hodí mincou' a správu buď prepošle ďalšiemu náhodne vybratému používateľovi alebo pošle príjemcovi. Takto žiaden používateľ nevie o prijatej správe povedať koľkokrát už bola preposlaná.
Spätné adresy sú buď riešené rovnakou cestou späť[62] alebo multicast na skupinu používateľov[46].
Malou nevýhodou je, že každý používateľ je nútený byť súčasťou. Tento spôsob sa už

ale približuje dnes najpoužívanejšej technike o ktorej hovorí nasledujúca podkapitola.

1.4 Onion routing

V súčasnosti najpopulárnejšou schémou je Onion routing (routing je anglický pojem pre smerovanie paketov). Schéma sa v základnej myšlienke podobá Mix-net schéme, keď na anonymizáciu využíva postupnosť preposielajúcich serverov (nazývané Onion Router, skrátene OR), ale nerobí mixovanie paketov ani žiadne iné obfuskácie pri ich preposielaní. Schéma uvažuje slabší a reálnejší model lokálneho útočníka, ktorý má pod kontrolou len časť siete, na ktorej môže vykonávať útoky. Ostatné podstatné informácie ako výber OR-ov alebo dodatočné obrany sú už vecou jednotlivých systémov. Keďže jeden popularitou, či už z hľadiska výskumu alebo počtu používateľov, vysoko prevyšuje všetky ostatné, tak uvedieme iba ten. Pravdaže boli implementované aj ďalšie[11][58].

1.4.1 Tor

Implementácia onion routingu Tor (skratka pre The Onion Router) vznikla priamo s výskumom schémy[35], ktorá bola neskôr podstatne zmenená po skúsenostiach s prvými implementáciami[22]. Dnes je najrozšírenejším anonymizačným systémom. Aktuálne sa už počet pripojení rôznych používateľov blíži k trom miliónom za jeden deň. Takýto počet používateľov systém získal hlavne vďaka dobrej prenosovej rýchlosti, jednoduchému použitiu a podpore širokej škály iných programov. Čo sa týka OR-ov, aktuálne je funkčných vyše šesť tisíc, aj keď optimálny by bol vyšší počet, o čom svedčia viaceré snahy motivovať používateľov, aby prevádzkovali vlastný OR.[55][40][3]
Všetky dáta, ktoré sme práve uviedli sú zverejnené projektom Tor na ich stránke[**?**].

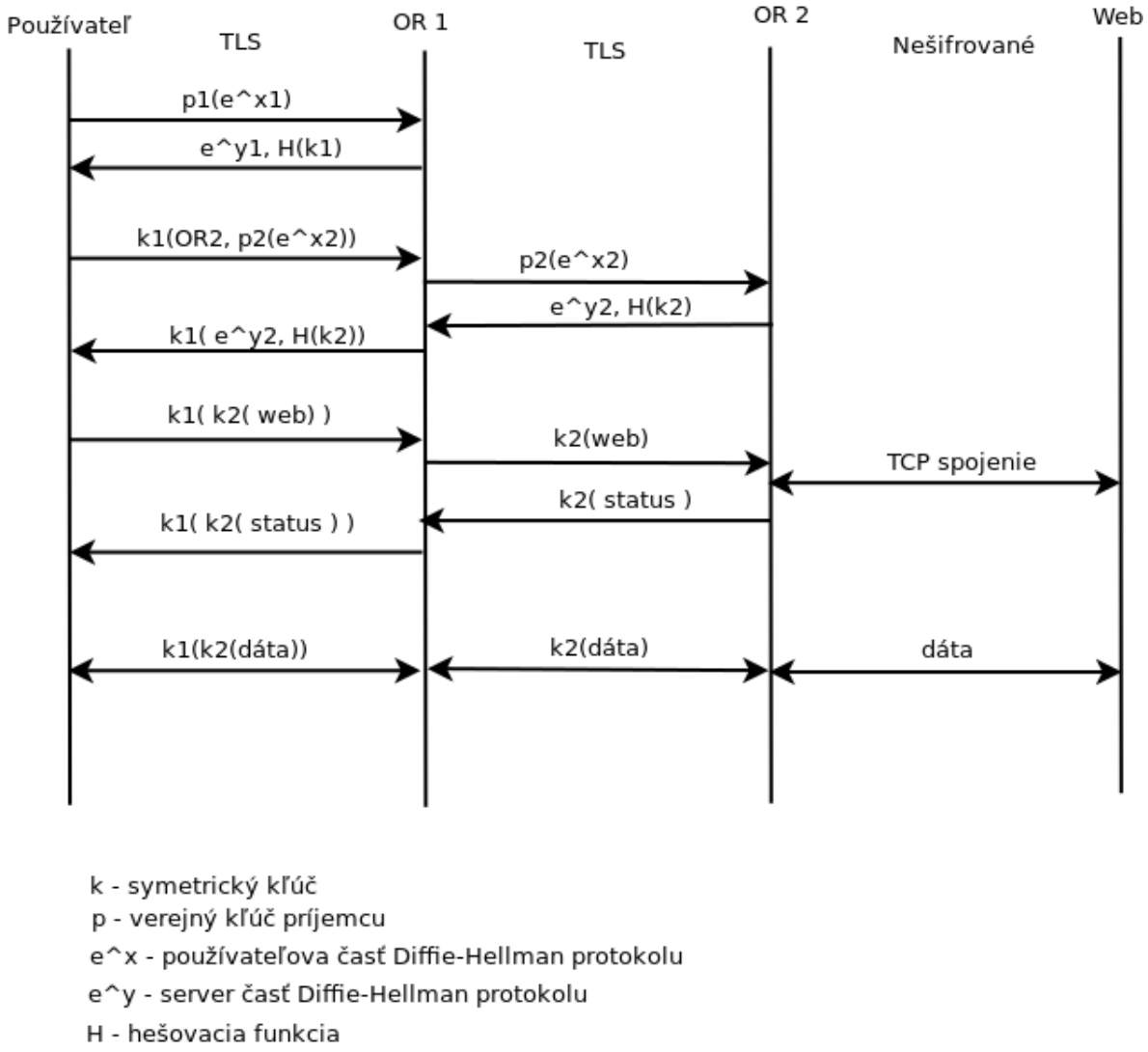
Voľba cesty a komunikácia

Pri popise systému najprv začneme pri voľbe cesty a autoritatívnych serveroch v systéme. Uvažovať budeme najnovšiu verziu (verzia 3), ktorá je popísaná v špecifikácii[**?**]. V systéme je niekoľko (približne medzi päť až desať) autoritatívnych serverov, z ktorých každý má svoj súkromný podpisový kľúč, ideálne uchovávaný šifrovane alebo offline. Verejné kľúče k nim sú priamo v inštalačnom balíku. Tieto súkromné kľúče sú určené výhradne na podpisovanie certifikátov ku kľúčom, ktoré si server generuje. Až týmto vygenerovaným kľúčom bude podpisovať jednotlivé záznamy pre používateľov. Tento spôsob dodatočnej vrstvy je častý vo viacerých systémoch a umožňuje jednoduchšiu zmenu kľúčov, keďže nie je potrebné s novým kľúčom meniť inštalačný balík.
Všetky OR-y posielajú sebou podpísaný súhrn svojich informácií ako rýchlosť pripojenia, identita, verejné kľúče, atď. autoritatívnym serverom. Autoritatívne servery sa

periodicky volebným protokolom dohodnú na celkovom stave systému. Ostatné OR-y, ktoré chcú poskytovať informácie o stave OR-ov v systéme, si tento celkový stav stiahnu a poskytujú ho používateľom. Väčšina OR-ov túto možnosť poskytuje.
Používateľ si pri stavbe reťaze stiahne kompletný podpísaný súhrn a vyberie si práve tri OR-y, ktoré si vyberá váženým algoritmom podľa rýchlosti pripojenia daného OR-u. Takže OR-y s lepším pripojením sú preferovanejšie. Reťaze sú vytvárané v predstihu, takže pri probléme na jednej systém automaticky prejde na druhú, ktorú už mal pripravenú. Systém používa reťaz po dobu desiatich minút alebo pokiaľ nie je spojenie ukončené, ako napríklad pri dlhom pripojení cez `ssh`. Veľa reťazí zdieľa jedno prepojenie medzi OR-mi a viacero pripojení jedného používateľa zdieľa jednu reťaz.
Viackrát bolo uvažované upraviť algoritmus tak, aby nenastal prípad, keď je celá reťaz zložená z OR-ov v jednom autonómnom systéme (skrátene AS)[31][27] [2][4]. V takomto prípade by sa z lokálneho útočníka stal pre daného používateľa a reťaz globálny útočník.
Ešte jedna vec, ktorú sme zatiaľ zamlčali je, že systém neposkytuje informácie o všetkých OR-och. Dôvodom je, že by bolo potom veľmi jednoduché pre autority blokovať prístup, keby stačilo iba zakázať známy zoznam adries. Tor preto poskytuje aj skryté pripájacie body, ktoré sa získavajú inými spôsobmi, napríklad si ich môžu zdieľať používatelia alebo je možné sa na niektorý spýtať priamo správcov projektu.

Teraz uvedieme ako sa z vybraných OR-ov vyskladá tunel. Všetky spojenia, či už medzi klientmi a OR-om alebo jednotlivými OR-mi sú `TCP` spojenia s šifrovaním `TLS`. Nadviazanie spojenia prebieha upraveným Diffie-Hellman protokolom a následné symetrické šifrovanie je preferovane `AES` v `CBC` (Cipher block chain) móde dĺžky 256 bitov, alternatívne 128 bitov alebo je použitý `3DES`. Vďaka použitiu týchto protokolov je automaticky zabránené útoku preposlaním odpočutej správy. Pri stavbe tunela najprv používateľ nadviaže spojenie s prvým zvoleným OR-om, keď pošle svoju časť Diffie-Hellman protokolu šifrovanú verejným kľúčom OR-u. následne dostane v odpovedi druhú časť a heš dohodnutého kľúča. Takáto zmeno oproti štandardnému protokolu je z dôvodu veľkosti bunky, do ktorej by sa podpis nevošiel. Následne naviaže spojenie s druhým OR-om rovnako, až na to, že už komunikáciu preposiela šifrovane cez prvý OR na ceste, s ktorým má zdieľaný kľúč. Keďže používateľova časť Diffie-Hellman protokolu je šifrovaná verejným kľúčom druhého OR-u, tak prvý nemôže použiť tzv. man-in-the-middle útok, kde by prvý OR vystupoval aj ako druhý. Rovnakým spôsobom sa dohodne na symetrickom kľúči aj s tretím. 1.6 Takto, keď používateľ posiela správu, ktorá má byť odoslaná z tretieho uzla, tak ju zašifruje tromi vrstvami, ktoré sa postupne cestou k cieľu odstraňujú. Odpoveď postupne prechádza od posledného uzla k používateľovi a vždy je pridaná jedna vrstva šifrovania. Všetky tri kľúče potrebné na odšifrovanie má len používateľ, ktorý sa na nich dohadoval.

Ako sme už spomenuli, veľa reťazí zdieľa jedno `TCP` prepojenie medzi OR-mi, preto má každá správa v sebe identifikátor špecifický pre danú dvojicu bezprostredne komunikujúcich. Všetky správy, ktoré sú vymieňané na úrovni systému, majú rovnakú fixnú veľkosť 512 bajtov, bez ohľadu na počet vrstiev šifrovania. Toto je jediná pridaná obrana a systém nevyužíva žiadne falošné správy ani oneskorenia.
Takto je to uvádzané v pôvodnom návrhu[22], aj často kladených otázkach[?], ale v manuáli[?] je už možné nájsť položku `ConnectionPadding`, ktorú je možné nastaviť. Pri použití prepínača `-list-torrc-options` na program nainštalovaný z repozitára, ktorý je verzie 0.2.9.11, ešte táto možnosť nie je, ale pri stiahnutom a skompilovanom programe verzie 0.3.1.8 už je táto možnosť poskytovaná. Špecifikácia[?] je ešte nedokončená, ale dá sa z nej zistiť, že táto obrana je mierená proti útokom využívajúcim časy otvorenia a uzavretia spojenia.



Obr. 1.6: Príklad stavby reťaze a komunikácie pre reťaz dĺžky dva

Anonymita prijímateľa

Okrem anonymity odosielateľa poskytuje Tor aj anonymitu prijímateľa, teda anonymné poskytovanie služieb cez `.onion` adresy. Tieto adresy nie sú uložené v bežných `DNS` záznamoch, ale sú poskytované iba v systéme Tor. Poskytovateľ si pri vytváraní anonymnej služby zvolí niekoľko OR-ov, do ktorých si vytvorí tunel a na ktorých bude prijímať žiadosti o spojenie. Informácie o `.onion` adrese, množine takýchto bodov a verejný kľúč odošle autoritatívnemu serveru. Keď sa používateľ chce napojiť na takúto službu, najprv si zvolí OR, do ktorého si vystavia tunel a odošle túto informáciu aj so svojou časťou Diffie-Helman do OR-u na ktorom príma služba žiadosti. Následne si aj server poskytujúci službu vystavia tunel do bodu, ktorý mu používateľ oznámil a sú spojení. Anonymné služby sú častým terčom útokov[57][51][78][7][44][56], ale jednotlivé útoky popisovať nebudeme, keďže anonymné služby budú mimo zamerania tejto práce.

Útoky

Namiesto toho popíšeme útoky na používateľov a na všeobecné sledovanie siete. Útoky, voči ktorým sa oficiálne Tor nebráni a spadajú do modelu, sú útoky s korelovaním komunikácie. Tento útok spadá do modelu len v prípade, že útočník vie sledovať oba konce komunikácie, teda medzi používateľom a prvým OR-om na ceste a medzi cieľovou službou a posledným OR-om na ceste. Treba aj uvážiť, že ak je útočníkom štátna inštitúcia, môže si vyžiadať potrebnú časť logu od iného štátu, s ktorým je v spolupráci. Silnejší predpoklad pri tomto útoku je, že používateľ už je podozrivý z prístupu k danej službe a už sa len predpoklad overuje, lebo sledovanie všetkých používateľov a služieb je výpočtovo príliš náročné.
Ešte spomenieme najznámejšie útoky, ktoré sa vyskytli v minulosti a ako sú účinné dnes. Najznámejší útok bol prezentovaný v roku 2005[52]. Útočník využíval vlastný OR, ktorý mal aktívne spojenie s inými a podľa rýchlosti ich odpovede vedel odhadovať ich aktuálne zaťaženie. Naviac útok uvažoval, že odpovedajúci server schválne odpovedal nárazovo, teda striedal odoslanie veľkého množstva dát za krátku dobu nasledované pauzou. Tým, že útočník vedel aké sú rozostupy a časy nárazov a dokázal merať aktuálne zaťaženie, tak vedel určiť ktoré OR-y sú v reťazi používateľa, čo sa pripájal. Nevedel však určiť priamo používateľa, ale znalosť celej reťaze je už narušenie anonymity. Tento útok sa dá preniesť aj na iné anonymizačné systémy, ktoré sa snažia o čo najlepšiu rýchlosť. Pri P2P schémach by mohol viesť priamo k odhaleniu používateľa[48], keďže každý používateľ aj preposiela komunikáciu iných. Tento útok sa časom stal nepraktický[30], keďže počet OR-ov násobne vzrástol od pôvodných 50.

Bolo prezentované aj vylepšenie, ktoré umožnilo útok naďalej používať[30], ale vyžadovalo konštrukciu dlhých ciest, čomu bolo v neskorších verziách zabránené.
Iný útok spočíval v rôznych priepustnostiach jednotlivých OR-ov a fakte, že jeden OR je veľmi pravdepodobne hrdlo spojenia[49]. Následne z korelácií priepustnosti je možné určiť či dve spojenia zdieľajú rovnakú reťaz alebo nie. Taktiež útočník vie určiť vytváraním reťaze dĺžky jedna a koreláciou, či daný OR je na ceste používateľa, hrdlo komunikácie. Tento útok taktiež neútočil priamo na odhalenie používateľa, ale na zistenie cesty.

1.5 Garlic routing

Garlic routing je nadstavba onion routingu. Táto schéma bola už dávnejšie spomenutá a špecifikovaná v jednej práci[21], ale aspoň miernu pozornosť získala až neskôr. Oproti onion routingu je podstatne menej skúmaná a aj implementácií je minimum. Okrem jednej zameranej čisto na zdieľanie súborov, existuje len jeden všeobecný anonymizačný systém a to I2P[41]. Vznikol približne v časoch rozvoja systému Tor a je popísaný v nasledujúcej podkapitole.

Podstata schémy je rovnako ako pri onion routingu, vrstvené šifrovanie a využívanie reťaze serverov na odšifrovanie vrstvy a preposlanie. Rozdielom je, že zašifrované správy sú spojené do väčších celkov, ktoré sú tiež prekryté vrstvou šifrovania. Preto aj názov vyplýva z anglického slova garlic, slovensky cesnak. Server po prijatí takéhoto balíku správ odšifruje jednu vrstvu a môže vidieť viacero ďalších skupín správ s rôznym smerovaním. Taktiež je možné viacero správ s rovnakým smerovaním zoskupiť, zašifrovať príslušným kľúčom a odoslať. V takomto prípade je možné vidieť paralelu s schémou Mix-net.

1.5.1 I2P

Ako sme už spomenuli, I2P je aktuálne jedinou nám známou implementáciou Garlic routingu. I2P je ideológiou v mnohých otázkach úplne opačný od systému Tor. Zatiaľ čo Tor využíva na preposielanie komunikácie servery spravované dobrovoľníkmi, tak v I2P je každý používateľ nútený preposielať komunikáciu iných. Pravdaže existujú vysokorýchlostné uzly, ktoré slúžia ako preposielacie servery, ale z pohľadu systému aj samotného bežiaceho programu sú totožné s používateľom. Preto budeme tieto pojmy často zamieňať. Poskytovanie jednotlivých serverov na konštrukciu reťaze je v Tor-e zabezpečené cez autoritatívne servery a v I2P je medzi používateľmi distribuovaná da-

tabáza na základe algoritmu Kademlia[47]. V Tore pri voľbe cesty sa vyžaduje, aby používateľ poznal všetky možné OR-y, zatiaľ čo v I2P pozná len časť. Obidva systémy používajú vážený algoritmus na volenie jednotlivých uzlov, ale Tor používa statické záznamy a I2P dynamicky vyhodnocuje. Rozdielny je tiež prístup k službám. Tor sa snaží byť hlavne proxy, ktorá poskytuje anonymný prístup do bežných sietí, ale vie poskytnúť aj anonymné poskytovanie služby v rámci systému. Naopak I2P sa snaží všetky služby držať v rámci systému a existuje len veľmi málo výstupných bodov. Spomínaný bol dokonca len jeden [70](taktiež existujú aj vstupné body pre prístup z normálnych sietí k službám v anonymnom systéme). Aj v otázke programovacieho jazyka sú systémy odlišné. Tor je v programovacom jazyku `C` a I2P je v jazyku `Java`, čo je neobvyklé pre sieťový protokol. Ale existuje aj neoficiálna otvorená implementácia I2P v jazyku `C++`.

Distribuovaná databáza

Najprv uvedieme stručný popis distribuovanej databázy a jej záznamov. Databáza ukladá len dva typy záznamov. Informácie o preposielacích serveroch, ako adresa, kľúče, časová známka a sieťové parametre. Tieto záznamy publikuje server priamo do distribuovanej databázy. Druhý typ záznamov v databáze sa týka cieľových služieb v systéme. Tiež sú uvedené kryptografické kľúče, ale na poskytnutie anonymity službám sú uvedené len kontaktné body, identifikátor pre smerovanie a čas expirácie kontaktného bodu. Tieto záznamy nie sú publikované priamo, ale cez tunel, ktorý si služba vytvorí. Stavba a použitie tunelov je popísané neskôr.
Na distribuovanej databáze sa nepodieľajú všetci, ale len časť vybraných serverov, ktorá sa časom mení. Tieto servery sú buď manuálne nastavené alebo volené automaticky pri nedostatočnom počte a naopak pri prevyšujúcom sa na databáze prestanú podieľať. Na automatické zaradenie medzi servery podieľajúce sa na databáze, musí server poskytovať vysokú rýchlosť a prejsť viacerými testami. Aj keď sa server verejne nepodiela na distribuovanej databáze, tak svoju vlastnú si udržiava aby nebol závislý a mohol hned po štarte skúšať pripojenia.
Pri prijatí *požiadavky na uloženie*, ju databázový server vyzdieľa na základe spomenutého algoritmu Kademlia, ktorý pojem vzdialenosti definuje ako XOR hešov kľúčov. Pri prijatí *požiadavky na vyhľadanie* priamo odpovedá. Požiadavky na vyhľadanie používateľ posiela databáze najbližšej k vlastnému kľúču, čo je v prvom kroku jeho vlastná. Vzdialenosť je takýmto spôsobom definovaná na zabránenie útoku, v ktorom by sa chcel útočník stať zvolenou databázou pre používateľa. Naviac každé vyhľadanie sa overuje z dvoch najbližších inštancií databáz a databáza neposkytuje všetky servery v systéme, ale náhodne zvolenú podmnožinu.

Voľba cesty a komunikácia

Keď už používateľ pozná jednotlivé preposielacie servery, tak si z nich zvolí množinu na základe váženého algoritmu. Z tejto množiny vytvára dva jednosmerné tunely. Jeden pre komunikáciu vedúcu od používateľa a druhý pre komunikáciu smerom k používateľovi. Ich dĺžka nie je pevne daná a najčastejšie sa používa dĺžka dva alebo tri. Síce stavba tunelu je postupná, ale z dôvodu jednosmernosti tunelov sa používateľ dozvie o úspechu či neúspechu až na koniec. Najprv sa stavia tunel smerujúci od používateľa. Používateľ posiela vrstvovo šifrovanú správu, kde každá vrstva obsahuje informácie pre server, nasledujúce smerovanie a šifrovanú časť pre nasledujúci server. Tieto vrstvy sú šifrované asymetricky šifrou `ElGamal`. Informácie vo vrstve obsahujú identifikátor tunela, časovú známku, `AES` kľúč pre šifrovanie v tuneli a `AES` kľúč pre zašifrovanie odpovede na požiadavky na stavbu tunela. Takto server pri prijatí požiadavky najprv odšifruje asymetrickú časť, svoju odpoveď zreťazenú so zašifrovanou odpoveďou od predchádzajúceho serveru zašifruje získaným kľúčom a pošle ďalej. Posledný uzol tunela posiela celkovú odpoveď používateľovi. Za predpokladu, že používateľ už mal vystavaný nejaký k nemu smerujúci tunel, čo sa deje keď používateľ pridáva dodatočný alebo mení tunel, tak je odpoveď poslaná týmto tunelom. Ak ešte nemal vystavaný žiadny tunel, tak si vytvorí dovnútra smerujúci tunel dĺžky nula, cez ktorý prijme odpoveď. Dnu smerujúce tunely sa budujú od konca, kde sa správa dostane von smerujúcim tunelom. Tým, že server pridáva len svoju odpoveď na stavbu tunela a nevie odpovede ostatných, ktoré mohli stavbu zamietnuť, tak udržiava tunel, aj keď je nepoužívaný. Na ochranu voči útoku zopakovaním odpočutej správy, kde by sa zopakovala správa na stavbu tunela sa používajú časové známky a história. Žiadna požiadavka nemôže mať staršiu časovú známku ako hodinu, po ktorú je udržiavaná história požiadaviek. Požiadavky so staršou časovou známkou sú ignorované.

Teraz prejdeme k posielaniu správ medzi dvomi účastníkmi, napríklad keď chce používateľ systému kontaktovať službu. Najprv si používateľ získa z databázy záznam o vstupných tuneloch služby. Je zjavné, že po odchádzajúcom tuneli sa vrstvy šifry odoberajú a po vstupnom sa pridávajú. Na šifrovanie týchto vrstiev sa používa šifra `AES` a kľúče získané pri stavbe tunela. Bez dodatočnej ochrany by správy šli medzi výstupným tunelom odosielateľa a vstupným prijímateľa bez ochrany. Preto sa používa ešte vrstva šifrovania medzi oboma koncami. Používateľ použije verejný kľúč získaný z databázy, ktorým zašifruje `AES` kľúč a pridá správu zašifrovanú týmto kľúčom a dodatočné informácie ako kontrolný heš, dĺžka a niekoľko jednorazovo použiteľných 32 bajtových identifikátorov. Tie slúžia aj ako inicializačné vektory pre šifru `AES`. Tieto identifikátory sú umiestnené na začiatku každej správy šifrovanej algoritmom `AES`. Ak sa tam identifikátor na začiatku správy nezhoduje s nejakým predtým prijatým, tak je

správa šifrovaná výpočtovo drahou asymetrickou šifrou `ElGamal`. Okrem iného používateľ posiela aj jeho kontaktný bod, teda koniec a identifikátor jeho vstupného tunela. Je potrebné si uvedomiť, že systém I2P poskytuje nespoľahlivú, nespájanú službu, teda neposkytuje záruky doručenia ani poradia. Na zabezpečenie týchto záruk sú použité ďalšie vrstvy, napríklad zabudovaný `I2PTunnel`, ktorý umožňuje prenášať štandardné protokoly cez I2P.

Všetky dátové správy medzi používateľmi sú veľkosti 1024 bajtov. Na transportnej vrstve sa v minulosti používal len `UDP` protokol, ale neskôr bol implementovaný aj `TCP` protokol. Nad týmito protokolmi nie je štandardná bezpečnostná vrstva, ale protokoly špecifické pre systém, čo nie je optimálne a existujú návrhy prejsť na štandardizované `SSL`[73].

Útoky

Útoky prezentované na tento systém sa zaoberali hlavne útokmi na distribuovanú databázu a ovplyvňovanie voľby tunelov[39][28][71]. Garlic routing, teda balenie správ do väčších celkov sa používa len občas pri správach medzi dvomi používateľmi. V pravidelných intervaloch sa pribalia dve správy, jedna pre status a druhá pre informácie o prístupových bodoch, aby sa predišlo pristupovaniu k databáze pri prípadnej zmene tunela. Preto je veľmi pravdepodobné, že veľa útokov na Tor sa dá použiť aj na I2P.

Kapitola 2

Ciele práce

V tejto kapitole uvedieme ciele práce a otvorené otázky, ktorým sa budeme venovať. Budeme uvažovať útoky s analýzou prenosu dát. Najprv sa budeme venovať útoku s koreláciou dát a jeho účinnosťou v schéme **Garlic routing**. Keďže vlastná implementácia by bola pracná a jediná existujúca implementácia tejto schémy je I2P, budeme uvažovať práve túto, aj keď skladanie správ skoro nevyužíva. Je vysoko pravdepodobné, že tento útok bude stále použiteľný, tak ako pri všetkých systémoch, ktoré sú orientované na výkonnosť. Skôr je otázne, o koľko je útok nepresnejší a výpočtovo a časovo náročnejší. Pri porovnaní budeme uvažovať systém Tor, ktorého ideológia je vykonávať čo najmenej potrebnej obfuskácie a poskytovať čo najlepší výkon. Ideálne budeme uvažovať koreláciu pri pasívnom odpočúvaní aj pri aktívnej modulácii prenosu.
Okrem tohoto základného útoku budeme uvažovať najznámejší útok na systém Tor, ktorý sme uviedli v predchádzajúcej kapitole. Síce útok uvažuje, že všetky preposielacie servery sú používateľovi známe, čo v systéme I2P nemusí platiť, ale útočníkovi stačí mať pod kontrolou server s distribuovanou databázou. Skutočnosť, že nie je problém poznať skoro všetky servery v sieti bola ukázaná aj vo výskumných prácach [29]. Ak sa ukáže, že niektorý útok je podstatne sťažený alebo nemožný, tak bude nasledovať analýza faktorov, ktoré sú pre obranu najpodstatnejšie a ich prenositeľnosť na iné systémy. Ak nastane opak, tak bude snaha odstrániť zbytočné obfuskácie zo systému, ktoré iba znižujú výkonnosť. Alternatívne sa nepokúšať zvyšovať rýchlosť, keďže ideológia I2P je viac zameraná na záruky anonymity, ale skúsiť využiť niektoré princípy z Mix-net schémy. Túto alternatívu uvádza aj stránka projektu a najviac sa uvažuje nad Stop-and-Go verziou. V nej by aj používateľ vedel ovplyvniť pomer anonymity a rýchlosti.

Najpodstatnejšie rozdiely, ktoré sú otázne, sú hlavne jednosmerné tunely a participácia používateľov na preposielaní komunikácie. Jednosmerné tunely sú zjavne lepšie, keďže poskytujú menej informácie. Ale korelácia prenosu uvažuje jeden smer a najčastejšie je prenos dát jednosmerný, napríklad pripájanie na stránku alebo stiahnutie

súboru. Smerom späť idú v takomto prípade len potvrdenia. Vynútenie preposielania pomáha, keď sa zmieša komunikácia používateľov, ale je otázne koľko cudzej komunikácie jednotlivý používateľ prenáša a či účasť na preposielaní neumožňuje použiť útoky na tunel k zisteniu používateľa[76]. Alternatívou by mohlo byť, že len náhodne vybraná podmnožina používateľov by sa účastnila preposielania a menila by sa dynamicky ako pri distribuovanej databáze.

Kapitola 3

Útok koreláciou

Ako sme už spomenuli v predchádzajúcej kapitole, základným útokom je korelácia toku dát. Tomuto útoku je náročné brániť a je základom mnohých ďalších. Preto sa najprv budeme zaoberať porovnaním korelácií pre systém Tor a I2P. Následne sa budeme zaoberať účinnosťou P2P v I2P pri anonymizácii.

3.1 Vytvorenie dát

Pri prvých dvoch experimentoch stačí vedieť sledovať komunikáciu servera a pripájajúceho sa klienta. Preto môžeme využiť reálnu sieť Tor a I2P. Prvým krokom je simulácia komunikácie, lenže sieťová komunikácia môže byť veľmi rôznorodá. Vzdialené pripojenie, zdieľanie súborov a web sú bežné typy komunikácie, ktoré sú navzájom veľmi odlišné. Takýchto typov by sme mohli nájsť veľa, namiesto toho sme sa zaoberali najbežnejším, ktorý je aj hlavným cieľom, teda web.
Prvým krokom je vybratie web servera a web stránok, ktoré poskytuje. Ako web server sme zvolili `Nginx`, jednak pre rýchlosť a efektivitu, ale hlavne z dôvodu poskytovania tzv. `CGI` rozhrania (skratka z anglického Common Gateway Interface). To umožňuje nechať spracovanie `HTTP` požiadavky inému programu. Túto možnosť využívame, aby sme dokázali stránky náhodne generovať pri každej novej požiadavke. Trochu predbehneme a spomenieme, že takýchto programov je zároveň spustených niekoľko, aby bolo možné odpovedať na viaceré požiadavky zároveň. Naviac tým, že máme pod kontrolou výstup a ak pre `Ngninx` vypneme vyrovnávaciu pamäť, tak vieme vytvárať nárazové odpovede alebo simulovať moduláciu prenosu. V našom prípade sú generujúce programy napísané v jazyku C s využitím príslušnej knižnice `libfcgi`. K nej len spomenieme, že je vhodné si ju pri použití prezrieť pretože direktívou `#undef` redefinuje niektoré štandardné knižničné funkcie, napr. `printf`.
Úvodný `HTML` dokument pre experiment je korektný a obsahuje náhodný počet odka-

zov na ďalšie súbory a náhodne veľký blok textu. Ďalšie súbory, konkrétne sa jedná o skripty (HTML tag `<script>`) a obrázky (HTML tag `<img>`), už nie sú formátom korektné, ale jedná sa o náhodné dáta. Keďže tieto dáta sú len prenesené a nie sú zobrazované či kontrolované, tak to nepredstavuje problém.

Čo sa týka náhodnosti počtu a veľkostí súborov, tak spočiatku bola snaha použiť distribúcie zo štatistík uverejnených na HTTP Archive **??**. Časom sa vyskytlo viacero problémov. To, že v niektorých prípadoch súčet percent nedával sumu sto, ani keby boli všetky hodnoty zaokrúhlené smerom nadol ešte nepredstavuje problém (stav z Novembra 2017, aktuálne prešla stránka mnohými zmenami). Omnoho horšie bolo, že stránky generované podľa takýchto štatistík násobne prekračovali veľkosť bežnej stránky. To nemusí nutne znamenať že štatistika je chybná, skôr sú v nej rôzne skryté závislosti. Intuitívne, ak má stránka vysoký počet obrazových súborov, tak sú najskôr malej veľkosti a naopak pri nízkom počte sa pravdepodobne jedná o veľké súbory tvoriace grafiku stránky. Rovnaká úvaha sa dá použiť aj pre skripty. Napriek tomu sme sa rozhodli štatistík v čo najväčšej miere držať, iba veľkosti boli predelené konštantou, po ktorej sa veľkosti priblížili realite. Rozhodne vhodnejšie by bolo využiť možnosť písania vlastných dotazov do poskytovanej databázy **??**. Alebo spraviť vlastnú vzorku webových dát, ale to už je príliš vzdialené od témy a zadania práce.

Následne bolo potrebné simulovať správanie klienta, teda webový prehliadač. Webový prehliadač najprv získa hlavný `HTML` dokument a následne vytvára viacero paralelných spojení, najprv pre textové súbory (`CSS`, `JavaScript`, `PHP`, ...) a potom pre grafické. Práve v počte paralelných spojení sa jednotlivé prehliadače líšia, preto je pre každé spojenie tento počet generovaný náhodne z odpozorovaného rozsahu. Na jednotlivé `HTTP` požiadavky sme použili program `curl`. Taktiež sme využili aj možnosť aktívneho útoku s ovplyvnením toku dát na zvýšenie korelačného koeficientu. Dáta sa odosielali nárazovo v dávkach, po ktorých nasledovala krátka pauza.

Pre klient Tor bol použitý štandardný konfiguračný súbor s tromi zmenami. Prvou bolo nastavenie logovania na `debug` úroveň, ktorá poskytuje najviac informácií o behu programu. Druhou zmenou bolo nastavenie portov, na ktoré sa môže klient pripájať pre komunikáciu. Špecifické pre Tor sú porty 9001 a 9030, ale často na vyhnutie sa blokovaniu firewallom sú používané porty pre najrozšírenejšie služby napr. 80 (`HTTP`), 443 (`HTTPS`), 53 (`DNS`), 25 (`SMTP`), ... Posledným nastavením bolo zníženie času zmeny okruhu. Keďže rýchlosť prenášaných dát je ovplyvnená zvolenou cestou, na vytvorenie čo najlepšej reprezentácie je vhodné pre každé načítanie stránky použiť iný okruh. Predvolené nastavenie je desať minút. Úpravou tohoto parametru bolo možné urýchliť zber dát. Samotné automatizovanie zberu dát využívalo viaceré skripty na spustenia servera a vytváranie pripojení v pravidelných intervaloch.

Pre I2P bolo potrebné zmeniť viacero nastavení. Prvým bolo vypnutie poskytovania pripojenia pre ostatných používateľov a vypnutie možnosti podieľania sa na distribu-

ovanej databáze za cieľom jednoduchšieho oddelenia komunikácie používateľa. Zdieľanie dát pri P2P je merané samostatne (neskôr v kapitole). Keďže tunely sú jednosmerné, je potrebné otvoriť určitý port (zvolený náhodne pri prvom štarte) aspoň pre UDP komunikáciu. Keďže podstatná väčšina používateľov využíva UDP aj TCP, tak aj v našom prípade bol port otvorený pre oba protokoly. Keďže ideológia I2P je držať služby v rámci systému, tak bolo potrebné nakonfigurovať prístup k serveru. Potrebné je zmeniť nastavenie servera určujúce maximálny počet pripojení od jedného klienta. Táto hodnota je prednastavená na nízky počet a po čase by server odmietal všetky požiadavky nášho klienta. Textový názov web adresy nebol vypublikovaný do systému na zabránenie pripojenia iného používateľa. Keďže niektoré iné programy bežiace v pozadí zvyknú overiť dostupné aktualizácie, tak boli jednotlivé DNS požiadavky zachytené a komunikácia s danými adresami odstránená. Samotné merania tiež neboli bez problémov. Prvým problémom bola nízka rýchlosť I2P. Stránky vytvárané pre systém I2P sú v priemere podstatne menšie ako bežné stránky. Preto pre merania v I2P boli stránky zmenšené aby lepšie reprezentovali skutočnosť, aj za cenu nezhody štatistiky pre Tor a I2P. Avšak je potrebné si overiť nakoľko tento prístup ovplyvňuje výsledky a preto sme odskúšali ako veľkosť dát ovplyvňuje výsledky. Preto sme pre Tor spravili aj dodatočné merania s rovnakou distribúciou veľkostí prenášaných dát ako pre I2P a porovnali ich s pôvodnou.
Druhým problémom bolo, že I2P okruhy sú nespoľahlivé, keďže sa jedná o P2P systém a ľubovoľný používateľ sa môže kedykoľvek odpojiť. Preto bolo potrebné uvažovať aj spadnutie spojenia. Drobným nedostatkom je, že nie je možné nastaviť čas zmeny okruhu, čo predlžuje zbieranie dát.

Pre meranie P2P sme zvolili tri hodnoty poskytovanej rýchlosti. Konkrétne 1, 5 a 10 Mbps pre prichádzajúcu aj odchádzajúcu komunikáciu po dobu 24 hodín. Takáto vzorka rýchlostí zachytáva bežné možnosti používateľa. Už pri prvých pozorovaniach bolo viditeľné, že počet tunelov, v ktorých participujeme s časom stúpal, keďže pri volení cesty sa okrem poskytnutej rýchlosti berie do úvahy aj spoľahlivosť. Mnoho tunelov (približne dve tretiny) však boli nevyužité a pravdepodobne sa jednalo o záložné alebo zlyhané.

3.2 Spracovanie dát

V prvom rade je potrebné odfiltrovať dáta, ktoré sa netýkajú našich meraní. V prípade systému Tor sme mali štandardný server vypublikovaný do internetu. Je známe, že porty verejných IP adries sú často skenované je potrebné prípadné pripojenie mimo

experimentu odstrániť. Vďaka nastaveniu logovania na úroveň `debug`, je možné zistiť priebehy výstavby každého tunelu. Postačujúce je zistiť `IP` adresy koncových uzlov tunelov a porovnať ich s `IP` adresami pripájajúcimi sa na náš server. Treba však vziať do úvahy aj fakt, že niektoré uzly majú viac `IP` adries. Pre takéto uzly sa na jednej prenášané dáta prijímajú a z druhej sa odosielajú. Stalo sa tak aj pri našom meraní, ale bolo to jednoducho rozlíšiteľné, keďže `IP` adresa bola často len zvýšená o jedna alebo čas pripojenia na serveri sa zhodoval s naším klientom. Tým, že publikovaný port mal vysoké a neobvyklé číslo, tak nebol skenovaný často. Aj tak sa objavilo niekoľko paketov s príznakom `SYN` za ktorými okamžite nasledovali pakety s príznakom `RST`, čo je bežný spôsob skenovania. V jednom prípade došlo aj o pokus o pripojenie, ale skenujúci predpokladal službu vzdialeného pripojenia pre operačný systém Windows a nie webserver.
Pre klienta žiadne filtrovanie potrebné nebolo, keďže využíval len určené porty. Naopak pre I2P nebola adresa publikovaná a len klient k nej mal prístup. Ale vzhľadom na to, že I2P využíva širokú škálu portov pre `TCP` komunikáciu je potrebné odstrániť komunikácie mimo systému, napr. spomínané overenie dostupnosti aktualizácií operačných systémov. Keďže sme zachytili celú `DNS` komunikáciu klienta aj servera, bolo jednoduché získať odpovede na dotazy a komunikáciu s týmito `IP` adresami odstrániť.

Následne z logov extrahujeme len potrebné informácie, ktoré uložíme vo forme lepšej pre ďalšie spracovanie. V oboch prípadoch rozdeľujeme prichádzajúcu a odchádzajúcu komunikáciu do separátnych súborov. Ukladáme časové pečiatky z pohľadu systému (tzv. kernel timestamp), veľkosti dát, `IP` adresy, porty a pre `TCP` komunikáciu aj sekvenčné čísla, potvrdzujúce čísla a príznakové bity (tzv. flag). Pre Tor môže byť klient aj server na jednom stroji, čo neplatí pre I2P, ktoré by `HTTP` dotaz k lokálnej službe poslalo v rámci stroja a teda by nedošlo k prechodu dát anonymizačnou sieťou. Tým, že sú použité viaceré stroje, či už virtuálne alebo fyzické, je potrebné časové pečiatky zosúladiť. Keďže logovanie paketov zachytáva aj reálny čas, je možné určiť čas štartu operačného systému a časy zosúladiť. Pravdaže existuje aj možnosť využiť len jeden stroj, keď sa vytvoria virtuálne sieťové rozhrania, spustia sa dve inštancie programu a obom sa nastaví prístup k rozdielnym rozhraniam. Avšak táto možnosť nám príde pracnejšia ako napísanie skriptu na zosúladenie času.

Potom komunikáciu rozdelíme na jednotlivé pripojenia. Práve v tomto nám výrazne pomôže zachytávanie `TCP` príznakov, ktoré určujú začiatok a koniec spojenia. Tor slúži ako proxy, takže komunikácia medzi posledným uzlom a serverom je štandardná. Či už sa jedná o `HTTPS`, kde vidíme len začiatok, šifrované dáta a koniec alebo `HTTP`, kde vidíme aj jednotlivé dotazy a dáta v otvorenom tvare. V oboch prípadoch nie je problém určiť začiatok a koniec komunikácie. Toto však neplatí pre I2P, v ktorom sa k službám

pristupuje cez poskytnutý tunel. Preto útočník nevie jednoducho určiť začiatok a koniec komunikácie alebo `IP` adresu servera. Síce Tor poskytuje aj skryté služby a I2P možnosť prístupu do bežného internetu cez tzv. outproxy, ale aktuálne je aktívna len jedna a nie je úplne spoľahlivá. Preto je potrebné buď stanoviť dva rozdielne modely útočníka na čo najlepšie priblíženie sa realite, alebo určiť rovnaké podmienky aj keď sa jedná skôr o špecifické prípady. Keďže našim cieľom je určiť korelácie komunikácie a útoky s ich využitím, tak sme zvolili prvú možnosť. Preto v I2P uvažujeme, že útočník buď vie `IP` adresu servera, alebo ho priamo ovláda.
Následne po rozdelení dát na jednotlivé pripojenia, každému pripojeniu určíme začiatočný čas na nulu. Takže stratíme informáciu o reálnom čase pripojenia a uvažujeme len samotný priebeh komunikácie. Naviac uvažujeme len načítanie práve jednej webovej stránky, pričom útočník bežne využíva aj navštívené podstránky a kompletnú aktivitu na zlepšenie presnosti.
Tým, že v danom čase sa mohla vyskytnúť aj komunikácia so systémom, napríklad výstavba záložného okruhu, tak v pripojení necháme len jednu `IP` adresu a to tú s najvyšším počtom prenesených dát v danom čase.
Pre I2P postupujeme podobne, kde rozdelíme komunikáciu s jednotlivými adresami do zvláštnych súborov (ďalej nazývané prúdy). Pre oba smery komunikácie zvolíme práve jeden. Pre server vieme správny prúd určiť jednoducho, keďže vieme veľkosti odoslaných dát. Pre prijaté dáta servera uvažujeme interné správy systému a nie potvrdzujúce pakety `TCP` spojenia s prvým uzlom v odosielacom tuneli. Ich rozlíšenie je jednoduché, keďže interné pakety majú podstatne väčšiu veľkosť.
Pre klienta určujeme prúd buď na základe najnižšej odchýlky vo veľkosti prijatých (v prípade spätnej komunikácie odoslaných) dát alebo podľa korelácie s opačným koncom komunikácie. Práve určenie správneho prúdu sme neskôr využili ako metriku účinnosti P2P.

Po upravení dát je jednoduché použiť existujúce techniky na výpočet korelačného koeficientu. Zvolili sme viacero možných metód na výpočet, konkrétne

- Pearsonov
- Spearmanov
- Kendallov
- normalizovaný výpočet vzájomnej informácie
- upravený Pearsonov[68] 3.1

Pri poslednej metóde sa uvažuje aj oneskorenie(budeme označovat ako Pearson+). Všetky tieto metódy, s výnimkou Pearson+, sú už naprogramované v balíku `scikit` pre jazyk `Python`. Parametrom pre všetky tieto metódy je veľkosť časového okna v rámci ktorého sa uvažuje suma prijatých či odoslaných dát. Pre vzorec Pearson+ sa ako parameter určuje aj oneskorenie v počte časových okien. V spomenutej práci[68]

sa uvažovali systémy s fixnými veľkosťami paketov. Preto sme odskúšali aj predelenie veľkostí jednotlivých paketov veľkosťou dátovej bunky v systéme (Tor 512, I2P 1024) a zaokrúhlenie nahor alebo úplnú filtráciu paketov s veľkosťou nepresahujúcou veľkosť bunky. Pre systém Tor sme uvažovali aj fakt, že systém svojou dodatočnou vrstvou medzi klientom a prvým uzlom, ktorá ale nie je použitá medzi posledným uzlom a serverom, môže ovplyvniť veľkosť prenesených dát. Pre systém I2P môžu zmenu spôsobit v úvode spomenuté pakety o stave. Preto sme odskúšali aj prenásobenie veľkostí dát vypočítaným podielom.
Keďže všetky metódy uvažujú dve postupnosti, tak primárne počítame korelačný koeficient pre dáta smerujúce od serveru ku klientovi. Tie sú spravidla rádovo väčšie ako dáta idúce opačným smerom. Koeficient pre dáta prenášané od klienta k serveru môže byť počítaný buď zvlášť alebo sa dá uvažovať súčet oboch smerov. Z týchto výsledkov vieme určiť účinnosť jednosmerných tunelov, aj keď len pre webovú komunikáciu. Pre systém Tor máme spätný smer dát priamo daný a môžeme ho využiť, zatiaľ čo pre systém I2P ho musíme nájsť, ako sme už v predchádzajúcej časti spomenuli.
Ako dodatočnú optimalizáciu môžeme použiť prenásobenie korelačného koeficientu podielom dát odoslaných zo servera a prijatých u klienta. Korelačný koeficient berie do úvahy len podobnosť postupností, ale veľkosť odoslaných a prijatých dát je rovnaká a líši sa len minimálne kvôli sieťovým vrstvám, ktoré dáta obaľujú. Týmto dokážeme odfiltrovať podobnosť komunikácií, ktoré ale prenášajú príliš rôzne veľkosti dát.

Obr. 3.1: Upravený Pearsonov korelačný koeficient

$$r = \frac{\sum_{i=1}^{n}(x_i - \overline{x})(y_{i+d} - \overline{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \overline{x})^2(y_{i+d} - \overline{y})^2}}$$

Následne vieme spočítať pre každú dvojicu korelačný koeficient a určiť tak najpravdepodobnejšiu komunikujúcu dvojicu. Keďže spájame každú dvojicu, tak je počet výpočtov korelácií kvadratický k počtu vzoriek. To je dôvod, prečo je tento útok príliš náročný na použitie pre väčšie systémy. Následne už len vypočítame úspešnosť uhádnutia spojenia.

Otázne je, aké parametre použiť pre dosiahnutie najlepšej úspešnosti. Tým, že máme viaceré korelačné metódy, zlučovacie metódy a rôzne časové okná, tak je kombinácií veľa. Preto tento proces automatizujeme na menšej trénovacej vzorke a nájdené optimálne parametre používame na hlavné dáta.

Okrem webových dát sme pre I2P separátne merali aj P2P záťaž, čo nám poslúži na vypočítanie účinnosti P2P pri maskovaní vlastnej komunikácie.
V predchádzajúcej časti sme pre klienta určovali prúd nesúci dáta na základe veľkosti dát alebo korelačného faktoru. To bolo jednoduché, keďže iba jeden prúd niesol používateľove dáta a ostatné boli vnútorné procesy I2P ako stavba záložných tunelov alebo získanie informácií pre databázu. Takéto procesy totiž prenášajú menej dát a naviac je nepravdepodobné, že by mali lepšiu koreláciu, či menej sa odlišujúcu veľkosť prenesených dát v danom časovom úseku ako reálny prenos. Ale pre prenosy dát ostatných používateľov je už otázne, nakoľko sú podobné a aká je pravdepodobnosť, že sa vyskytnú v čase vhodnom na zamaskovanie komunikácie.
Okrem nameraných dát pri participácii v P2P použijeme aj dáta z obyčajných pripojení cez I2P. Pre každé pripojenie niekoľkokrát zvolíme náhodný čas, z P2P dát získame jednotlivé prúdy v danom čase a rovnakým algoritmom ako v pôvodnom určovaní prúdu zvolíme jedného kandidáta. Následne zistíme, či je vhodnejší ako reálny a teda či sa pomocou participácie v systéme podarilo používateľovi skryť jeho komunikáciu.

3.3 Výsledky

Najprv sa budeme zaoberať presnosťou pre systém Tor. Pod presnosťou myslíme ku koľkým percentám pripojení na server správne určíme komunikujúceho klienta. Na začiatok budeme uvažovať len komunikáciu smerujúcu zo servera ku klientovi. Využitiu opačného smeru komunikácie sa budeme venovať zvlášť v nasledujúcej podkapitole. Prvým vecou, ktorý sme vyhodnocovali bolo spracovanie radu veľkostí dát do jednotlivých časových okien. Ako sme už spomínali, môžeme uvažovať nespracované veľkosti, predeliť veľkosťou dátovej bunky v systéme a zaokrúhliť nahor, prenásobiť priemernou zmenou veľkostí medzi odoslanými a prijatými dátami alebo oba prístupy skombinovať. Prenásobením zmenou veľkostí sa myslí priemerná zmena medzi veľkosťou dát servera s posledným uzlom na ceste a prvým uzlom klienta.
Každá metóda na spracovanie bola vyhodnotená pre každý korelačný koeficient zvlášť na základe najlepšieho priemerného korelačného koeficientu. Pre Tor mal najlepšie výsledky prístup s predelením veľkosťou dátovej bunky. Pre I2P najlešie výsledky dosahoval prístup s prenásobením priemernou zmenou, keďže I2P kvôli spomínaným správam o stave je viac dát odoslaných ako prijatých. Je vhodné podotknúť, že samotný korelačný koeficient toho veľa nevraví, keďže je možné ho podstatne zvýšiť napríklad zväčšením časového okna. To ale spôsobí zníženie presnosti správneho určenia komunikácie. Preto je korelačný koeficient až druhoradý. Prístup s filtrovaním paketov menšej veľkosti ako je dátová bunka neposkytol žiadne zlepšenie, práve naopak, o pár percent

Tabuľka 3.1: Tor - trénovacia množina

časové okno / metóda	Pearson	Spearman	Kendall	vzájomná informácia
0.05s	88%	63%	63%	69%
0.1s	87%	59%	62%	59%
0.25s	92%	69%	71%	54%
0.5s	90%	79%	80%	55%
0.75s	90%	87%	88%	50%
1s	91%	90%	92%	50%
1.25s	93%	91%	94%	47%
1.5s	91%	94%	96%	37%
1.75s	95%	93%	95%	33%

Tabuľka 3.2: Tor - trénovacia množina, metóda Pearson+

časové okno / posun	0	1	2	3	4	5
0.05s	-	-	93%	95%	95%	98%
0.1s	-	94%	98%	96%	95%	-
0.25s	92%	96%	95%	92%	-	-
0.5s	90%	96%	95%	-	-	-
0.75s	90%	98%	-	-	-	-
1s	91%	97%	-	-	-	-
1.25s	-	-	-	-	-	-
1.5s	91%	-	-	-	-	-

zhoršoval presnosť určenia. Tento prístup má väčšiu šancu byť užitočný skôr v opačnom smere komunikácie, kde sa odstránia potvrdzujúce pakety `TCP` spojenia medzi klientom a prvým uzlom a zostanú len tie určené pre posledný uzol na ceste k serveru.

Štandardné korelačné metódy 3.1 dosahujú na trénovacej množine úspešnosť 90 až 95 percent pre časové okná s dĺžkou väčšou ako sekunda s výnimkou výpočtu metódou vzájomnej informácie. Pri nej vychádzajú najlepšie výsledky pre časové okno dlhé 0.05 sekundy, ale úspešnosť je podstatne nižšia ako pre ostatné metódy. Metóda Pearson+, ktorá uvažuje aj oneskorenie 3.2 vykazuje ešte lepšie výsledky, kde presnosť určenia komunikujúcej dvojice dosahuje až 98 percent pre viaceré kombinácie dĺžky časového okna a posunu.

Po odskúšaní na trénovacej množine sme použili získané parametre na hlavnú množinu. Opäť sme uvažovali všetky metódy ale už len tri najlepšie získané parametre, okrem metódy Pearson+ kde sme vyhodnotili sedem najlepších.

- Pearson
  - 1.25s - 92.0%
  - 1.5s - 98.8%
  - 1.75s - 99.2%
- Spearman
  - 1.25s - 98.6%
  - 1.5s - 97.8%
  - 1.75s - 97.2%
- Kendall
  - 1.25s - 99.0%
  - 1.5s - 98.8%
  - 1.75s - 97.8%
- vzájomná informácia
  - 0.05s - 90.4%
  - 0.1s - 75.6%
  - 0.25s - 57.7%
- Pearson+
  - 0.25s, oneskorenie 1 - 99.6%
  - 0.50s, oneskorenie 1 - 89.0%
  - 0.75s, oneskorenie 1 - 72.8%
  - 1.00s, oneskorenie 1 - 58.2%
  - 0.1s, oneskorenie 2 - 99.8%
  - 0.1s, oneskorenie 3 - 96.6%
  - 0.05s, oneskorenie 5 - 98.6%

Mierne zvláštne je, že skoro všetky úspešnosti sa pri väčšej vzorke ešte zvýšili. Naopak pre metódu Pearson+ s niektorými parametrami výrazne klesla úspešnosť.
Po vyhodnotení presností metód a parametrov je vhodné pozrieť sa aj na samotný korelačný koeficient. Uvažujme trénovaciu množinu a dve najpresnejšie vyhodnotenia, čo je v Pearson+ pre časové okno 0.75 sekundy a oneskorenie 1 a časové okno 0.1 sekundy a oneskorenie 2. Pre obe uvedené kombinácie parametrov nastali dve chyby. Pre prvú kombináciu parametrov mali chybne určené spojenia korelačný faktor 0.370 a 0.396, čo je druhý a piaty najnižší korelačný koeficient. Priemerný korelačný koeficient je 0.588. Pre druhú kombináciu parametrov boli chybné korelačné koeficienty 0.261 a 0.257 čo sú ôsme a jedenáste najhoršie korelačné koeficenty. Priemerný je 0.418.
Pre hlavnú množinu a najlepšie parametre, čo opäť bola metóda Pearson+ s časovým oknom dĺžky 0.1s a oneskorením 2, nastala len jedna chyba s korelačným koeficientom 0.2. čo bola zároveň aj najnižšia hodnota. Priemerná hodnota bola 0.702.
Tieto poznatky je možné využiť na nastavenie hranice, od ktorej budeme považovať

Tabuľka 3.3: Tor - znížené veľkosti

časové okno / metóda	Pearson	Spearman	Kendall	vzájomná informácia
0.05s	80%	63.6%	60.1%	80%
0.1s	82.7%	60.1%	55.5%	67.3%
0.25s	88.2%	68.2%	67.3%	60.1%
0.5s	89.1%	85.5%	80.1%	63.6%
0.75s	92.8%	89.1%	88.2%	62.7%
1s	94.6%	88.2%	86.4%	60.1%
1.25s	92.8%	90%	90%	60.1%
1.5s	92.8%	90%	91.8%	58.2%
1.75s	95.5%	87.3%	90%	49.1%

Tabuľka 3.4: Tor - znížené veľkosti, metóda Pearson+

časové okno / posun	0	1	2	3	4	5
0.05s	-	-	90.1%	90%	94.5%	90%
0.1s	-	93.6%	95.4%	87.3%	76.4%	-
0.25s	88.2%	95.4%	62.8%	37.3%	-	-
0.5s	89.1%	78.2%	28.2%	-	-	-
0.75s	92.8%	59.1%	-	-	-	-
1s	94.6%	44.5%	-	-	-	-
1.25s	-	-	-	-	-	-
1.5s	92.8%	-	-	-	-	-

určenie spojenia za presné, keďže chybné výsledky sa nachádzajú medzi najnižšími hodnotami.

Predtým ako prejdeme k výsledkom pre I2P, overíme ako zníženie objemu prenesených dát, ovplyvní presnosť určenia spojenia. Ak sa budú výsledky výrazne líšiť, tak to budeme musieť zohľadniť pri vyvodzovaní záverov.

Z výsledkov pre trénovaciu 3.1 3.2 množinu a pre množinu so zníženým objemom prenášaných dát 3.3 3.4 na úroveň akú použijeme v I2P je viditeľné len mierne zhoršenie, okrem metódy vzájomnej informácie, kde nastalo zlepšenie. Ale keďže tento prístup má najhoršie výsledky, tak jeho zlepšenie nie je až tak podstatné ako zhoršenie v presnejších metódach.
Ak porovnáme úspešnosti z množiny so sníženými dátami s úspešnosťami z hlavnej množiny je vidieť podstatnejšie zhoršenie. Teda platí intuícia, čím viac dát na spracovanie a väčšie rozdiely, tým presnejšie vieme určiť komunikujúcu dvojicu. Takže cieľom

Tabuľka 3.5: I2P - trénovacia množina

časové okno / metóda	Pearson	Spearman	Kendall	vzájomná informácia
0.05s	42.3%	33.0%	33.0%	43.3%
0.1s	41.2%	38.1%	38.1%	42.3%
0.25s	47.4%	47.4%	44.3%	55.7%
0.5s	60.8%	54.6%	48.4%	56.7%
0.75s	58.8%	51.5%	48.4%	54.6%
1s	63.9%	59.8%	57.7%	61.9%
1.25s	63.9%	59.8%	56.7%	54.6%
1.5s	70.1%	58.8%	56.7%	56.7%
1.75s	64.9%	58.8%	56.7%	53.6%
2s	67.0%	62.9%	62.9%	52.6%
2.25s	71.1%	56.7%	56.7%	49.5%

pre určovanie spojení v I2P bude dosiahnuť rovnaké úspešnosti ako pre túto množinu. Opäť je vhodné sa pozrieť na korelačné faktory a zistiť, či sa chybné určenia vyskytujú medzi najnižšími hodnotami. Znovu uvažujeme metódu Pearson+ a parametre 0.1 sekundy pre dĺžku časového okna a dve oneskorenia. Tentokrát je chybných určení päť z čoho štyri sa nachádzajú medzi poslednými dvanástimi, kde najvyššia hodnota z týchto štyroch bola 0.451. Piaty mal dvadsiatu najvyššiu hodnotu 0.771. Priemerná hodnota korelačného koeficientu bola 0.648. Pri použití parametrov, časové okno 0.25 sekundy a oneskorenie jedna sa síce najvyššia chybová hodnota posunula o deväť pozícií nižšie s hodnotou 0.834, ale stále prevyšovala priemer 0.745. Z toho je možné usúdiť, že s nižším objemom dát neklesá len úspešnosť ale aj istota, že pri nadpriemernej hodnote je vyhodnotenie správne.

Teraz pristúpime k výsledkom pre I2P. Doteraz sme sa venovali systému Tor a výsledky nám poslúžia ako základ pre porovnanie. Na vyhodnotenie budeme používať rovnaké metódy, iba zmeníme veľkosti časových okien, keďže I2P je výrazne pomalší systém.

Pre systém I2P sme zaznamenali podstatné zhoršenie, keď najlepší výsledok dosahoval len 70 percent 3.5 3.6. Najlepšie výsledky dosahovala Pearsonova metóda. Taktiež si môžeme všimnúť, že optimálne parametre pre metódu s výpočtom vzájomnej informácie sa posunuli do okolia jednej sekundy. Metóda Pearson+, ktorá doteraz vykazovala najlepšie výsledky stratila na účinnosti. Pravdepodobne z dôvodu často sa meniacej záťaže a teda aj oneskorenia I2P siete.
Filtrácia paketov s veľkosťou nepresahujúcou veľkosť dátovej bunky systému opäť ne-

Tabuľka 3.6: I2P - trénovacia množina, metóda Pearson+

časové okno / posun	0	1	2	3	4	5
0.1s	-	49.5%	44.3%	41.2%	40.2%	39.2%
0.25s	47.4%	50.5%	46.4%	38.1%	43.3%	-
0.5s	60.8%	52.6%	53.6%	47.4%	-	-
0.75s	58.8%	59.8%	-	-	-	-
1s	63.9%	60.8%	48.4%	-	-	-
1.25s	-	55.7%	45.3%	-	-	-
1.5s	70.1%	-	-	-	-	-

zlepšila výsledky a preto ich neuvádzame.
Tiež sa môžeme pozrieť na hodnoty korelácie a ako ich hodnota ovplyvní istotu pri určení spojenia. Uvažovať budeme najlepšie parametre, teda Pearsonovu metódu a časové okná dĺžky 2.25 a 1.5 sekundy. Keďže chybných určení je podstatne viac, budeme uvažovať koľko je chybných určení s hodnotou korelačného koeficientu vyššou ako priemer. V prvom z uvedených je priemerná hodnota korelačného koeficeintu 0.565 a približne 63% spojení má hodnotu vyššiu. Z nich je len 10% chybných. Preto ak by sme chceli dosiahnuť takúto úspešnosť určenia, museli by sme viac ako tretinu spojení neuvažovať. Pre druhý uvedený parameter je počet nadpriemerných spojení skoro rovnaký, ale chybných je 12%.

Teraz prejdeme k výsledkom z hlavnej množiny. Opäť uvažujeme niekoľko najlepších parametrov získaných z predchádzajúcich meraní.

- Pearson
  - 1.5s - 34.9%
  - 2s - 37.6%
  - 2.25s - 38.2%
- Spearman
  - 1.00s - 28.3%
  - 1.25s - 28.7%
  - 2.0s - 31.8%
- Kendall
  - 1.75s - 28.3%
  - 2.0s - 27.9%
  - 2.25s - 28.1%
- vzájomná informácia
  - 1.0s - 23.6%

  - 1.5s - 26.9%
  - 0.5s - 25.0%
- Pearson+
  - 0.75s, oneskorenie 1 - 38.0%
  - 1.0s, oneskorenie 1 - 38.6%
  - 1.25s, oneskorenie 1 - 40.7%
  - 0.5s, oneskorenie 2- 34.1%
  - 1.0s, oneskorenie 2 - 33.9%

Zväčšenie množiny pre I2P podstatne znížilo úspešnosť správneho určenia spojenia, ktorá už len slabo presahuje 40 percent. V tomto prípade už ani nemá zmysel hovoriť o hodnotách korelačného koeficientu a určovať hranicu, lebo už v horných 10 percentách z pohľadu hodnoty korelačného koeficeintu je chyba 20 percent.

Teraz sa budeme venovať aj prípadu, keď má útočník pod kontrolou server alebo linku medzi serverom a posledným uzlom v reťazi klienta. Túto možnosť využíva na upravovanie toku dát, aby zvýšil korelačný koeficient a jednoduchšie našiel ovplyvnenú komunikáciu. V tomto prípade nebudeme uvádzať úspešnosti určenia komunikujúcej dvojice, keďže ovplyvnenie všetkých komunikácií rovnakým spôsobom úspešnosť znižuje. Ovplyvnenie náhodným volením pauzy medzi odoslaniami časti dát tiež nezlepší úspešnosť, keďže veľkosť časového okna pre vzorec sa volí pevne. Význam aktívneho ovplyvnenia je zvýšenie korelačného koeficientu.
Ako ovplyvnenie sme zvolili odoslanie 250kB s nasledujúcou pauzou 250ms a parametre sme zvolili podľa najúspešnejších metód pri určovaní. Z výsledkov pre Tor 3.7 môžeme vidieť, že aktívne ovplyvnenie má v priemere vyššiu hodnotu korelačného koeficientu oproti trénovacej množine, ale nie oproti hlavnej. Tá vykazuje vysoké hodnoty, čo je s veľkou pravdepodobnosťou dôvod, prečo sa úspešnosť v tejto množine zvýšila. Pre I2P 3.8 rovnako vidíme zvýšený korelačný koeficient oproti trénovacej množine a aj hlavnej. Pri hlavnej si tentoraz môžeme všimnúť nižšie hodnoty ako pri trénovacej, čo spôsobilo zníženie presnosti určenia spojenia. Čo spôsobuje viditeľné zmeny korelácií v neovplyvnených množinách, je však otázne. Na určenie dôvodu by bolo potrebné sledovať záťaž celej siete.
Aktívne ovplyvnenie zjavne podstatne zlepšuje korelačný koeficient pre rovnaké parametre. Útočník by tým získal väčšiu istotu pri vyhľadávaní komunikácií, ktoré ovplyvnil.

Z výsledkov je jasné, že I2P oveľa lepšie odoláva základným útokom s koreláciou dát. Tento výsledok je mierne prekvapujúci a je otázne, z akého dôvodu o toľko lepšie odoláva systém I2P bežným technikám. Vezmime do úvahy, že systém I2P je rádovo pomalší ako Tor. Ak by sme uvažovali celkové dáta odoslané za celú dobu načítania stránky a nie jednotlivé `TCP` spojenia, tak je Tor viac ako 20 krát rýchlejší. Jedna práca

Tabuľka 3.7: Tor - korelačný koeficient s ovplyvnením

metóda, čas, oneskorenie / množina	trénovacia	hlavná	aktívne ovplyvnenie
Pearson+, 0.75, 1	0.585	0.557	0.701
Pearson+, 0.1s, 2	0.415	0.7	0.604
Pearson+, 0.25s, 1	0.512	0.776	0.758
Pearson, 1.5s	0.64	0.892	0.856
Spearman, 1.5s	0.703	0.884	0.82
Kendall, 1.5s	0.568	0.781	0.752
vz. info, 0.1s	0.284	0.488	0.703

Tabuľka 3.8: I2P - korelačný koeficient s ovplyvnením

metóda, čas, oneskorenie / množina	trénovacia	hlavná	aktívne ovplyvnenie
Pearson, 1.5s	0.512	0.372	0.53
Pearson, 2.25s	0.561	0.433	0.596
Spearman, 2s	0.491	0.404	0.534
Kendall, 2s	0.405	0.324	0.438
vz. info, 1s	0.517	0.455	0.601
Pearson+, 1s, 1	0.377	0.352	0.423

naznačovala, že so zrýchlením sietí a znížením časov odoziev sa takéto útoky stávajú efektívnejšími[8]. Aj medzi trénovacou a hlavnou množinou pre Tor bol podstatný rozdiel v priemerných rýchlostiach, čo mohlo spôsobiť zvýšenie presnosti. Preto jediným vysvetlením prichádzajúcim do úvahy je práve rýchlosť a aktuálne zaťaženie systému. Znovu podotkneme, že sme uvažovali krátke spojenia, teda iba načítanie jednej stránky. Pre dlhodobé spojenia sú útoky efektívnejšie.

3.3.1 Jednosmerné tunely

Doteraz sme uvažovali len dáta, ktoré server posielal klientovi. Teraz však budeme uvažovať aj dáta, ktoré posiela klient serveru. Pri webovej komunikácií sa jedná hlavne o potvrdenia prijatia dát. Aby sa ukázalo, že je vôbec užitočné skrývať spätnú komunikáciu zvlášť do druhého tunela, musí tento smer dát poskytovať lepšie výsledky pri určovaní spojenia ako smer ku klientovi.
Opäť použijeme trénovaciu množinu, na ktorej zistíme, aké parametre sú najvhodnejšie pre určovanie spojenia. Ak sa ukáže, že aj spätný smer poskytuje dostatočnú informáciu, použijeme najlepšie parametre na hlavnú množinu.

Pre Tor bolo najúspešnejšie použiť Pearsonovu metódu a časové okno dĺžky 1.25

sekundy. Dosiahla sa tak úspešnosť 46%, čo je ale podstatne nižšie ako úspešnosti pri komunikácií smerom ku klientovi. Keďže hlavná množina vykazovala lepšie výsledky, odskúšali sme, či aj v uvažovaní opačného smeru sa výsledky zlepšia, ale práve naopak úspešnosť klesla na 20%.

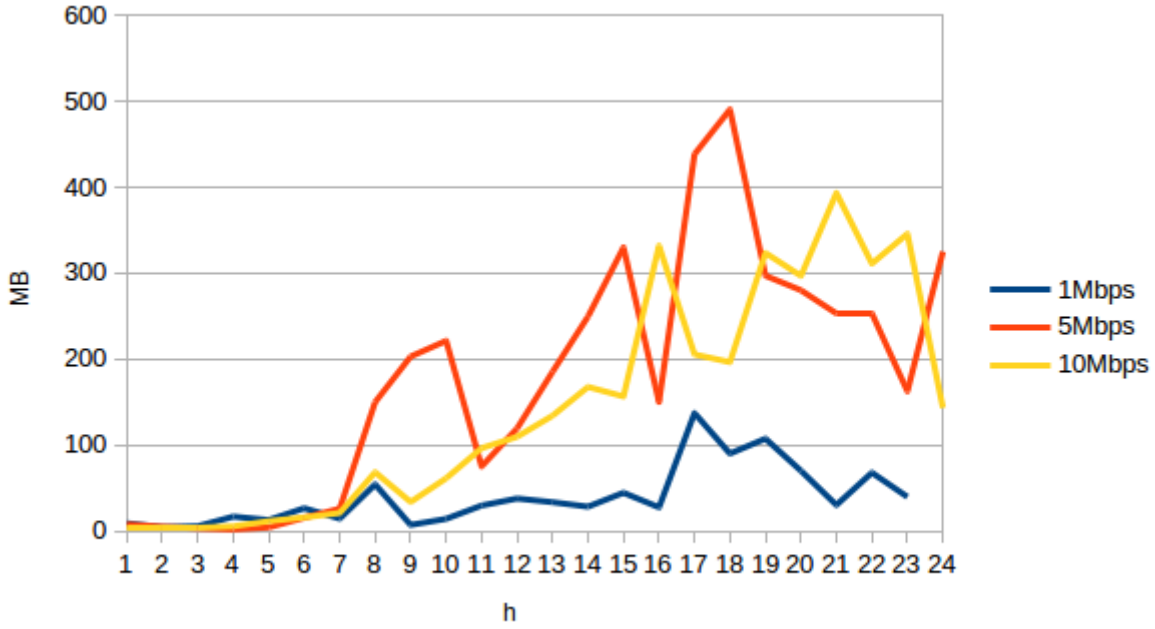
Pre I2P bol najlepší prístup Pearsonova metóda s časovým oknom dĺžky 2.25 sekundy, čo dosiahlo tesne cez 50%. To je síce stále podstatné zhoršenie oproti výsledkom pre trénovaciu množinu, ale táto úspešnosť by bola vyššia ako úspešnosť pri štandardnom prístupe pre hlavnú množinu. Preto sme spustili nájdenie parametrov pre hlavnú množinu, aj za cenu dlhých výpočtov. Opäť najlepšie výsledky dosiahol rovnaký prístup a úspešnosť klesla na 43%, čo je stále najlepšia úspešnosť pre hlavnú množinu. Z toho vyplýva, že pre I2P je výhodné oddeliť spätnú komunikáciu do samostatného tunelu, keďže poskytuje viac informácií útočníkovi. Na druhú stranu, rozdiel troch precent pri takej nízkej úspešnosti je skoro zanedbateľný a z praktického hľadiska by bolo lepšie mať obojsmerné tunely.

3.3.2 P2P

Na začiatok uvedieme ako sa mení veľkosť prenesených dát s časom, po ktorý sme aktívny. Z grafu 3.2 je viditeľné, že veľkosť prenesených dát s časom stúpa. Najväčší nárast nastáva približne po 8 hodinách. Medzi poskytnutím 5Mbps a 10Mbps boli minimálne rozdiely. Intuitívne je jasné, že čím viac prenesených dát, tým lepšie dokáže používateľ skryť vlastnú komunikáciu.

Ako metriku ohodnocujúcu skrytia vlastnej komunikácie, použijeme úspešnosť zvolenia správneho prúdu spomedzi všetkých videných v náhodne zvolenom čase. Teda spomedzi všetkých `IP` adries, s ktorými klient v náhodne vybranom čase komunikoval vyberieme tú, s ktorou sa komunikácia najviac veľkosťou alebo koreláciou približuje komunikácii servera. Ak je zvolená komunikácia lepšia v ohodnotení ako bola pôvodná, tak sa klientovi podarilo skryť jeho vlastnú.
Aj keď sme v predchádzajúcej podkapitole označili spätnú komunikáciu ako nedostatočnú pre útok, tak budeme uvažovať aj jej skrytie. Ak uvažujeme zvolenie toku dát na základe najbližšej veľkosti dát 3.9, tak je skrytie komunikácie dobré aj pri 1Mbps. Medzi 5Mbps a 10Mbps sú minimálne rozdiely, keďže ako sme videli, vytvárajú podobnú záťaž. Určenie spätnej komunikácie, je viditeľne ešte zložitejšie.
Ak uvažuejeme určenie na základe najlepšej korelácie 3.10 tak dostaneme podstatne

Obr. 3.2: Veľkosť prenesených dát



Tabuľka 3.9: P2P úspešnosť na základe veľkosti

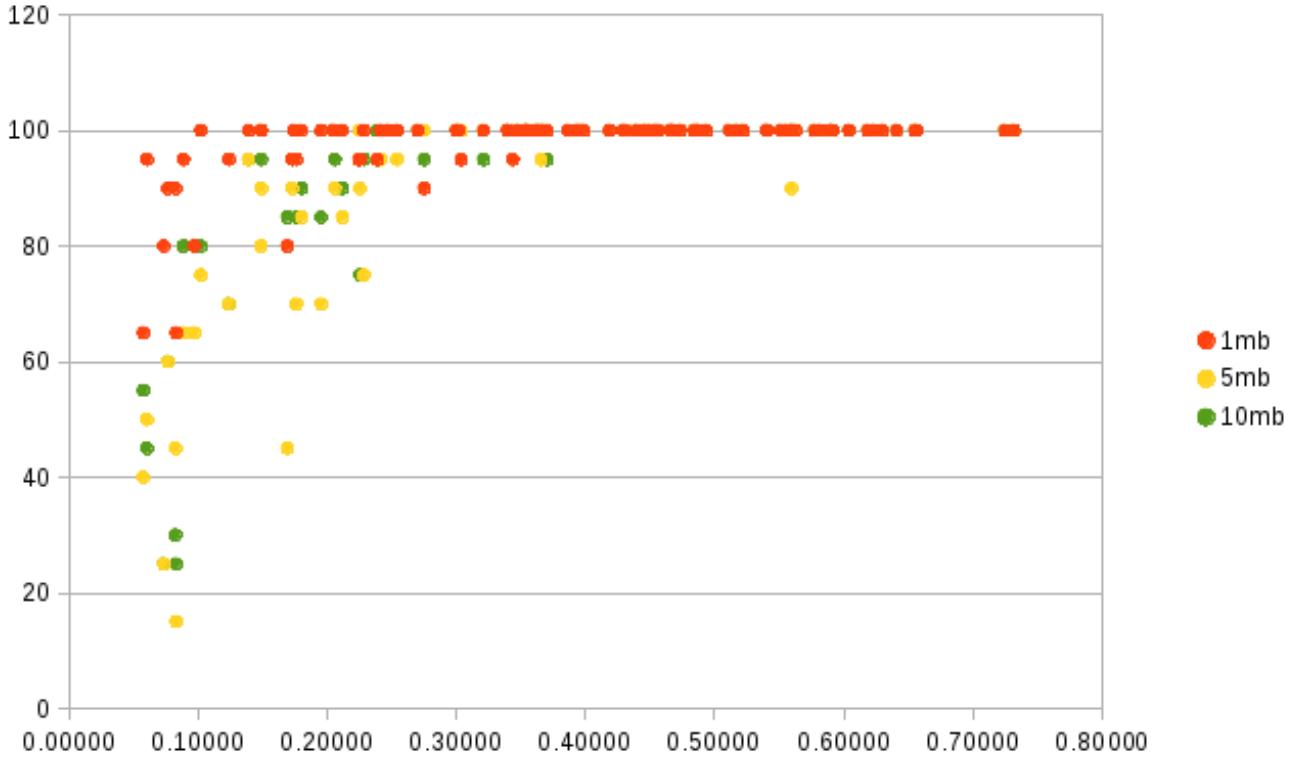
	1Mbps	5Mbps	10Mbps
dopredná komunikácia	81.82%	62.84%	65.36%
spätná komunikácia	59.79%	34.07%	44.48%

lepšie výsledky, ktoré ale sú výpočtovo náročnejšie.

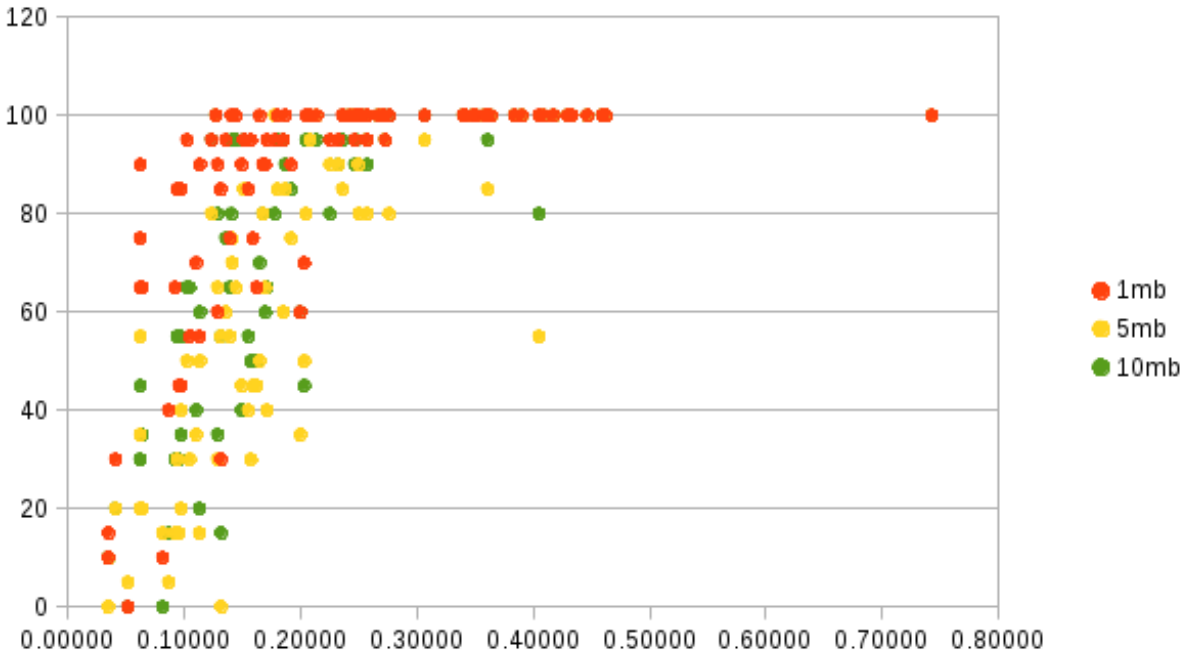
Na záver ešte uvedieme ako ovplyvňuje veľkosť dát a korelačný koeficient úspešnosť určenia správnej komunikácie. Pre korelačný koeficient platí 3.3, že čím vyšší, tým nižšia pravdepodobnosť chyby. Rovnako pre spätnú komunikáciu 3.4. Takéto výsledky sú očakávané a taktiež boli očakávané aj pre veľkosti dát, ale nepotvrdili sa a z grafov nie sú viditeľné žiadne výsledky. Z tohoto dôvodu ich neuvádzame.

Tabuľka 3.10: P2P úspešnosť na základe korelačného faktoru

	1Mbps	5Mbps	10Mbps
dopredná komunikácia	97.67%	91.56%	93.50%
spätná komunikácia	85.39%	68.33%	74.28%



Obr. 3.3: Úspešnosť určenia v závislosti od korelačného faktoru



Obr. 3.4: Úspešnosť určenia v závislosti od korelačného faktoru - spätná komunikácia

Kapitola 4

Útok s preťažením

V tejto kapitole sa budeme zaoberať útokom, ktorý sme tiež popísali v úvodnej kapitole. Útočník v ňom mal pod kontrolou server a odpovedal nárazovo na krátke preťaženie uzlov na ceste medzi ním a klientom. Pôvodne bol tento útok mierený na Tor a preto budeme skúmať jeho prenositeľnosť na systém I2P. V systéme Tor dokázal útok odhaliť okruh, ale nie koncového používateľa. V pôvodnej publikácii[52] bolo naznačené, že v P2P systémoch by mohlo dôjsť k odhaleniu používateľa. Preto naším cieľom bude overenie tohoto predpokladu.

4.1 Infraštruktúra a nástroje

Prvým nástrojom, ktorý pripadal do úvahy bol Shadow. Jedná sa o sieťový emulátor, ktorý umožňuje vytvárať rôznu topológiu, nastavovať geolokácie, rýchlosti prepojení, stratu paketov, atď. Pôvodne vznikol práve na testovanie systému Tor bez potreby reálnej infraštruktúry. Shadow využíva priamo kód testovaného programu a umožňuje odchytávať systémové volania, teda aj prácu so sieťou a simulovať prechod dát interne. Testovaný program je nahraný v pamäti len raz a využíva sa zmena kontextu, keď sa zmenia len dáta pre špecifickú inštanciu. Naviac odchytením výpočtovo drahých kryptografických volaní je možné simulovať aj vysoký počet inštancií na jednom počítači. Každý testovaný program musí byť pripravený ako doplnok pre Shadow.
Samotná inštalácia nebola problémová, ale pri spustení poskytnutých testov boli dva problémové. Konkrétne sa jednalo o testy číslo 41 (názov sleep) a 43 (názov sockbuf), ktorý bol spomenutý aj inými používateľmi ako chybný. Mätúce na týchto testoch bolo, že sa správali rôzne na rôznych operačných systémoch. Na čerstvo nainštalovanom virtuálnom stroji s `Ubuntu 16` neprechádzali oba testy, ale nie konzistente, takže občas aj obidva skončili úspechom. Rovnaký prípad bol operačný systém `Debian 9`. Pre operačný systém `Fedora` neprešiel len test č. 41, ale vždy. Každopádne vo všetkých prípadoch skončil ukážkový pokus s poskytnutou konfiguráciou úspešne.

Napriek týmto problémom sme chceli tento simulátor využiť a vytvoriť doplnok pre program I2P. Konkrétne sme chceli použiť otvorenú implementáciu tohoto protokolu v jazyku `C++` kvôli lepšiemu narábaniu s programom Shadow, ktorý je v jazyku `C`. Úplné znemožnenie však je, že program pre Shadow musí byť práve jeden proces s práve jedným vláknom, čo ani pre túto implementáciu I2P neplatí.

Ďalšou skúmanou alternatívou bol starší projekt ExperimenTor, ktorý využíval virtuálne stroje a sieťový simulátor ModelNet. Tento projekt je už ale zanechaný a odkazy na stiahnutie sú nefunkčné. Simulátor ModelNet síce je možné ešte nájsť, ale je už zastaralý v zmysle, že je určený pre operačný systém `FreeBSD` 6.3, ktorý je zhruba desať rokov starý.

Keďže spomenuté existujúce nástroje nebolo možné použiť, treba vytvoriť vlastnú infraštruktúru na vykonanie útoku. V našej schéme uvažujeme jedného klienta, jeden kompromitovaný server, jedného útočníka, množinu používateľov sprostredkujúcich komunikáciu, sieťový simulátor a celú existujúcu sieť I2P. Tá slúži na generovanie reálnej záťaže.
Server poskytuje webové stránky a odpovedá nárazovo, teda dáta odosiela po častiach medzi ktorými sú rovnako dlhé pauzy. Táto webová služba má nastavené dĺžky tunelov na nulu. Keďže uvažujeme, že server je pod kontrolou útočníka, takýto prístup je možný a uľahčuje mu prácu.
Množina uzlov, ktoré sprostredkujú komunikáciu a zapájajú sa do reálnej anonymizačnej siete. Okrem štandardných nastavení obsahujú tunely na pripojenie sa na službu `ssh`. Pre vstupný aj výstupný tunel je dĺžka nastavená na nulu. Význam týchto tunelov ešte podrobne popíšeme neskôr.

Klient sa taktiež zapája do existujúcej anonymizačnej siete, ale pre tvorbu tunelov využíva prednostne uzly z našej množiny. Predpoklad, že útočník pozná množinu, z ktorej si používateľ vyberá je síce silný, ale vychádza z predchádzajúcich prác ohľadom systému I2P[28][74]. V najhoršom prípade by útočník musel uvažovať všetkých aktuálnych účastníkov systému, čo by ale veľmi predražilo útok z pohľadu výpočtovej sily. Klient taktiež poskytuje službu `ssh` s nulovými dĺžkami tunelov.
Možností ako dosiahnuť prednostnú voľbu z našej množiny je viac, ale každá má rôzne výhody a nevýhody.
V každom prípade je najprv potrebné nášho klienta oboznámiť s existenciou všetkých uzlov z našej množiny. Každý používateľ si udržiava informácie o iných uzloch v sieti sám a získava nové od určených uzlov. Teda jednou z možností by bolo mať zapnutého klienta aj všetky uzly po dlhú dobu a veľmi pravdepodobne by ich záznamy získal. Jednoduchšie však je záznamy ručne pridať, keďže každý uzol obsahuje aj svoj záznam

a každý záznam je zvlášť oddelený súbor. Súbory síce obsahujú binárne dáta, ale `IP` adresy sú v štandardnom tvare, takže nie je problém potrebné súbory nájsť, napr. príkazom `grep -R`. Vymazanie ostatných súborov u klienta, by síce na začiatku spôsobilo že vyberá výhradne z našej množiny, ale časom by získal ďaľšie záznamy. Mazať tieto záznamy za behu programu by môhlo spôsobiť chyby až spadnutie programu keďže so súbormi program pracuje.

Okrem manipulácie s databázou, je možné využiť tzv. blacklist, teda zoznam zablokovaných `IP` adries. Keďže zoznam sa načítava pri spustení, nie je možné doň pridávať záznamy dynamicky a teda dostaneme rovnaký problém ako s vymazaním s databázy. Zablokovať všetky adresy je naopak nemožné na stroji s normálnymi zdrojmi, keďže možných adries sú približne štyri miliardy. Grafické rozhranie I2P zobrazuje aj blokovanie časti siete, kde sa určí adresa a maska podsiete. Napríklad sa jedná o lokálne siete 192.168.0.0/26 a 10.0.0.0/8. Ale takéto blokovanie sa nedá nastaviť v konfiguračnom súbore a je už priamo zadefinované v súbore `jar`, teda by bolo potrebné rekompilovať zdrojový kód. Pravdepodobne by takýto prístup bol najlepší, otázne ale je, či zablokované `IP` adresy môžu vytvárať tunely cez naše uzly, pretože participácia v sieti je jeden z predpokladov útoku. Ideálne by bolo, ak by I2P obsahoval tzv. whitelist pre tunely, teda zoznam cez ktoré má svoju komunikáciu posielať.

Z týchto dôvodov sme zvolili iný prístup. Priamo v grafickom rozhraní je možné pridať kladný alebo záporný bonus uzlu v sieti. To ovplyvňuje pravdepodobnosť jeho výberu. Rozhranie obsahuje aj možnosť zakázať určený uzol do reštartu, ale opäť je otázne či s ním potom povolí spoluprácu. Kombinácia pridania kladných bonusov našej množine a záporných iným vysokokapacitným uzlom v sieti sa ukázala byť celkom účinná. Hlavné zameranie bolo na vytvorenie dovnútra smerujúceho tunelu cez našu množinu, keďže práve dáta smerujúce zo servera vytvárajú krátkodobé preťaženie. Nevýhodou takéhoto prístupu je, že je potrebné pridávať záporné bonusy manuálne, keďže uzly v sieti sú dynamicky vyhodnocované a neustále objavované. Tým, že grafické rozhranie je vo webovom prehliadači by síce bolo možné vytvoriť špecifický program, ktorý by tento proces automatizoval. Požiadavky na server by sa vytvárali len v prípade keď celý dovnútra smerujúci tunel patrí do našej množiny. Vzhľadom na pracnosť takéhoto riešenia a fakt, že nám stačí len malý počet vzoriek na overenie útoku sme dáta vygenerovali manuálne pre štyri rôzne kombinácie parametrov. Pre každú kombináciu sme vygenerovali aspoň desať spojení.

Ako prvú kombináciu sme zvolili 500kB odosielaných po 100kB a pauzou 1 sekunda. V druhej kombinácii sme pauzu predĺžili na 2 sekundy. Tretia sada bola odosielala po 50kB a pauza bola znovu 1 sekunda. Posledná kombinácia bola odoslanie 300kB dát v dávkach po 25kB s pauzou 1 sekunda.

Je vhodné podotknúť, že za akceptovateľnú sme považovali aj cestu kde dva z troch uzlov patrili do našej množiny.

V našej schéme sa útočník sa do anonymizačnej siete nezapája a na monitorovanie času odozvy jednotlivých uzlov z našej množiny a tiež klienta využíva práve službu `ssh`, ktorú sám poskytuje. Dĺžky oboch tunelov sú opäť nastavené na nulu, čím sa dosiahne priama komunikácia. Týmto spôsobom simulujeme útočníkov uzol z pôvodnej práce[52], ktorý vlastnou komunikáciou monitoroval čas odozvy. Pravdaže by bolo vhodnejšie naprogramovať takéto správanie, ale jednalo by sa o omnoho pracnejšie riešenie. Cez službu `ssh` sa napojí každý uzol z množiny a spustí skript, ktorý každých 100 milisekúnd vygeneruje jedno písmeno na výstup. Tým spôsobí pravidelné odoslanie jedného paketu od útočníka na ktorý uzol odpovie potvrdením. Potrebné je si uvedomiť, že potvrdzujúce pakety sú len na úrovni `TCP` spojenia, zatiaľ čo samotné I2P pakety sú na aplikačnej vrstve. Preto je potrebné nastaviť rýchlosť linky rovnakú ako rýchlosť pre I2P.
Lenže všetky naše stroje sú na lokálnej sieti, ktorá je omnoho rýchlejšia a neobsahuje bežné javy ako oneskorenie, či strata paketov, tak využijeme sieťový simulátor aby sa všetky linky medzi našimi strojmi čo najviac blížili skutočnosti. Na oneskorenia opäť využijeme verejne dostupné dáta, tentokrát z webstránky Wondernetwork,

ktorá uvádza časy prechodu `ICMP ping` medzi mnohými mestami vo svete. Niekoľko miest vyberieme a podľa nich nastavíme spomalenie na linkách. Ostatné parametre ako strata paketov, počet duplikátov, atď. nastavíme len na minimálne až nulové hodnoty. Sieťový simulátor sme volili spomedzi

- Riverbed Modeler,
- NetSim Emulator a
- CORE.

Zvolili sme **CORE**, keďže sa jedná o open-source riešenie a obsahuje všetko čo potrebujeme.

4.2 Spracovanie dát

Prvým krokom je odstránenie `IP` adries, ktoré nepatrili do našej sledovanej množiny. Keďže uvažujeme log vytvorený firewallom a časové pečiatky z pohľadu systému, tak najprv upravíme pečiatky aby prislúchali rovnakým reálnym časom. Potom logy upravíme do podoby jednoduchšej na ďaľšie spracovanie a ponecháme len užitočné informácie. Opäť nechávame rovnaká informácie ako v predchádzajúcej kapitole. Tiež znova rozdelíme log na jednotlivé časti podľa času spojenia, ktoré vieme získať z logu servera. Následne už len rozdelíme na jednotlivé súbory podľa `IP` adries.

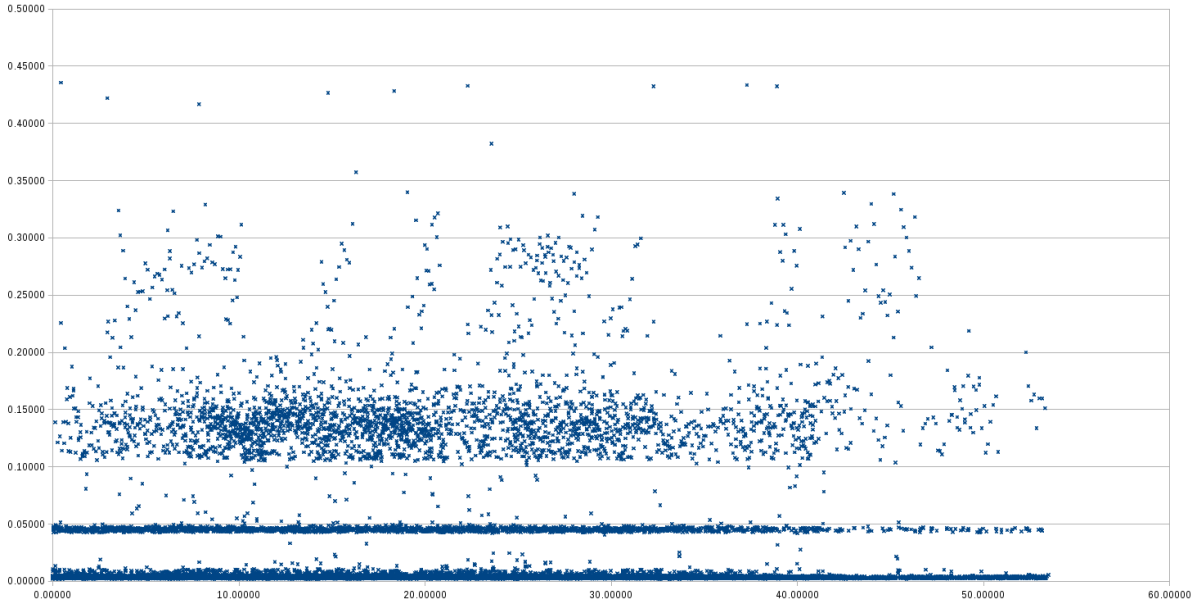
Následne pre každú IP adresu a každý odoslaný paket získame čas odozvy. Na tento účel nám poslúžia práve zachytené sekvenčné a potvrdzovacie čísla TCP spojenia. Pre UDP spojenie takúto informáciu nemáme a môžeme záťaž odhadovať len z rozdielu času medzi jednotlivými paketmi, keďže zo strany útočníka je komunikácia pravidelná.

4.3 Výsledky

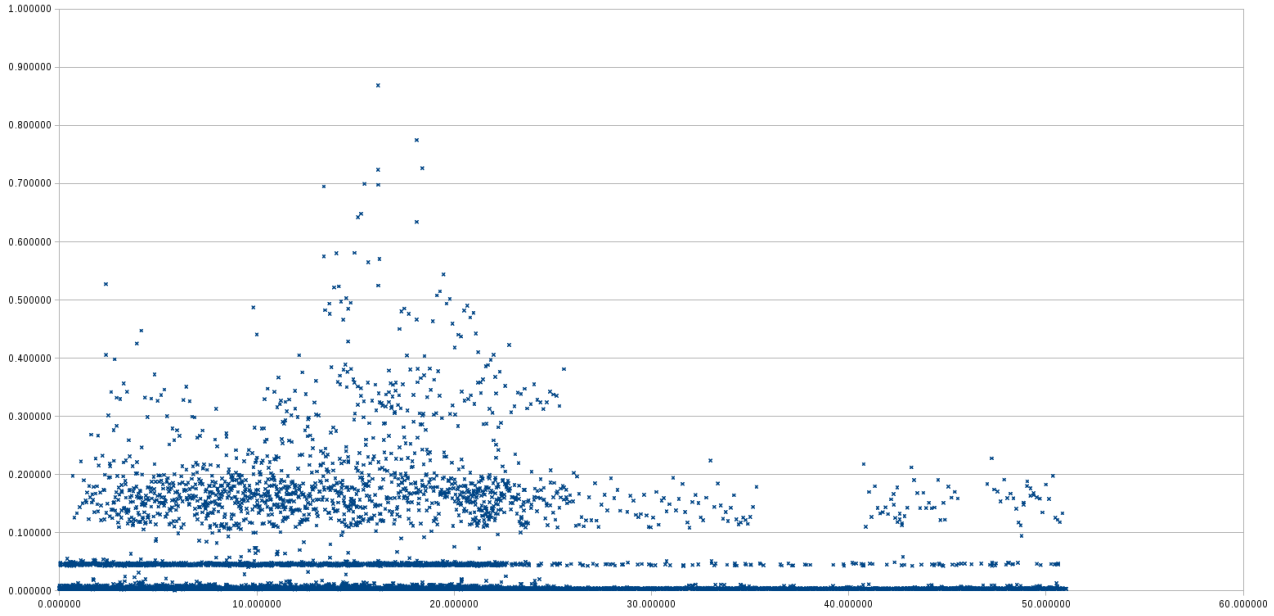
Keďže grafy pre všetky štyri kombinácie parametrov vyzerali veľmi podobne, tak uvedieme len dva. V každom meraní sme navyše zaradili aj päť sekúnd pred a po, nech je viditeľný rozdiel. Ale z grafu 4.1 žiaden rozdiel viditeľný nie je. Konkrétne tento graf bol pre druhú spomínanú kombináciu parametrov, takže v ideálnom prípade by sme mali vidieť päť náhlych spomalení s odstupom dve sekundy na rozmedzí približne 5 až 15 sekúnd. Z grafu sa dá pozorovať mierne zlé rozloženie oneskorení, čo spôsobilo vrstvy a dĺžka spojení. Napriek tomu, že sa odosielalo len 500kB po 100kb za dve sekundy, teda očakávaná dĺžka by mala byť niečo nad desať sekúnd, tak dĺžka spojenia dosahovala aj 40 sekúnd. Dve najdlhšie spojenia pre tieto parametre nastali práve keď bola celá cesta v nami ovládanej množine uzlov, čo je tiež prekvapujúce vzhľadom na to, že stroje boli v sieti zapojené len krátko (cca 2h). Pritom sme v minulej kapitole videli 3.2, že záťaž začne zreteľne narastať až po približne ôsmich hodinách.
Uvedieme ešte graf pre štvrtú sadu parametrov 4.2, kde počas skoro všetkých spojení viedla cesta cez naše uzly. Pre túto sadu boli skoro všetky spojenia veľmi rýchle a väčšina skončila do dvanástich sekúnd, ale napriek tomu tiež nie je viditeľná náhla nárazovosť. Skôr je možné sledovať, že oneskorenia v monitorovacej komunikácií vyzerajú náhodne.

Útok sa teda v z výsledkov potvrdiť nepodaril. Ale nie je možné ho s istotou vyvrátiť. Bolo by potrebné odskúšať viacero parametrov, vziať do úvahy čas behu pre množinu sprostredkujúcich uzlov a tiež naprogramovať monitorovací program, ktorý by nevyužíval rôzne zabudované funkcie, ale priamo využíval vrstvu I2P. Zvýšila by sa tým presnosť a rozšírili možnosti pre útok.
Ak by sa ani po takýchto vylepšeniach útok nepodarilo úspešne potvrdiť, tak by sa môhol vyvodiť rovnaký záver ako pre systém Tor, teda že daná anonymizačná sieť je príliš zaťažená a rôzne náhodné udalosti v nej bránia použitiu takéhoto typu útoku.

Obr. 4.1: Rýchlosti odpovedí monitorovaných uzlov pre parametre: 500kB, 100kB, 2s



Obr. 4.2: Rýchlosti odpovedí monitorovaných uzlov pre parametre: 300kB, 25kB, 1s



Záver

Stanovené ciele sa podarilo v značnej miere dosiahnuť. Z výsledkov vyplýva, že I2P lepšie odoláva korelačným útokom za cenu násobnej straty rýchlosti. Pre jednosmerné tunely boli výsledky rôzne. Zatiaľ čo pre Tor nemá zmysel schovávať spätnú komunikáciu ak sa jedná o bežnú webovu komunikáciu, tak pre I2P spätná komunikácia poskytuje rovnakú, až o trochu lepšiu informáciu na tento útok. Pre zapojenia sa do P2P sa ukázalo, že až po niekoľkých hodinách začne množstvo komunikácie prechádzajúcej cez používateľa výrazne stúpať. Teda je lepšie mať zapnutý systém dlhodobo na lepšie skrytie vlastnej komunikácie. Určovanie toku nesúceho komunikáciu zo servera na základe veľkosti prenesených dát sa pri P2P ukázalo byť slabo efektívne. Podstatne lepšie výsledky vykazovalo určovanie na základe korelačného faktoru, čo je ale výpočtovo náročnejšie.
Útok s preťažením a monitorovaním uzlov na odhalenie cesty, sa potvrdiť nepodarilo napriek odskúšaniu rôznych parametrov. Vyvrátiť sa zo získaných dát tiež s určitosťou nedá. Je potrebné odskúšať omnoho viac parametrov, keďže I2P je násobne pomalší a nie je jednoduché určiť aké parametre spôsobia krátkodobé preťaženie. Tiež je potrebné uvažovať čas behu pre uzly na ceste, keďže určuje ich integráciu v systéme.

Okrem lepšieho preskúmania druhého útoku je možné pokračovať vo viacerých veciach. Prvou je zistenie úspešnosti pri prenášaní rovnakých dát rôznymi cestami a určiť tým, nakoľko voľba cesty ovplyvňuje silu útoku. V tejto oblasti sa dá zaoberať aj dĺžkou cesty, kde by výsledkom bol pomer zisku rýchlosti k zlepšeniu presnosti útoku. Na Tor boli prezentované aj viaceré útoky s pozmenením časov medzi jednotlivými paketmi a tým bola do komunikácie vnesená informácia, ktorú sa útočník pokúšal detegovať. Bolo by vhodné overiť, či tieto schémy sú využiteľné na I2P alebo opäť výrazné spomalenie zmení časovania natoľko, že už sa nebude dať získať žiadna informácia.

Literatúra

[1] Dakshi Agrawal, Dogan Kesdogan, and Stefan Penz. Probabilistic Treatment of MIXes to Hamper Traffic Analysis. In *Proceedings of the 2003 IEEE Symposium on Security and Privacy*, pages 16–27, May 2003.

[2] Masoud Akhoondi, Curtis Yu, and Harsha V. Madhyastha. LASTor: A Low-Latency AS-Aware Tor Client. In *Proceedings of the 2012 IEEE Symposium on Security and Privacy*, May 2012.

[3] Elli Androulaki, Mariana Raykova, Shreyas Srivatsan, Angelos Stavrou, and Steven M. Bellovin. Par: Payment for anonymous routing. In Nikita Borisov and Ian Goldberg, editors, *Proceedings of the Eighth International Symposium on Privacy Enhancing Technologies (PETS 2008)*, pages 219–236. Springer, July 2008.

[4] Armon Barton and Matthew Wright. Denasa: Destination-naive as-awareness in anonymous communications. *Proceedings on Privacy Enhancing Technologies*, 2016(4), October 2016.

[5] Oliver Berthold, Hannes Federrath, and Stefan Köpsell. Web MIXes: A system for anonymous and unobservable Internet access. In H. Federrath, editor, *Proceedings of Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability*, pages 115–129. Springer-Verlag, LNCS 2009, July 2000.

[6] Oliver Berthold, Andreas Pfitzmann, and Ronny Standtke. The disadvantages of free MIX routes and how to overcome them. In H. Federrath, editor, *Proceedings of Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability*, pages 30–45. Springer-Verlag, LNCS 2009, July 2000.

[7] Alex Biryukov, Ivan Pustogarov, and Ralf-Philipp Weinmann. Trawling for tor hidden services: Detection, measurement, deanonymization. In *Proceedings of the 2013 IEEE Symposium on Security and Privacy*, May 2013.

[8] Sambuddho Chakravarty, Angelos Stavrou, and Angelos D. Keromytis. Traffic analysis against low-latency anonymity networks using available bandwidth estimation. In *Proceedings of the European Symposium Research Computer Security - ESORICS'10*. Springer, September 2010.

[9] David Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2), February 1981.

[10] David Chaum. The dining cryptographers problem: Unconditional sender and recipient untraceability. *Journal of Cryptology*, 1:65–75, 1988.

[11] Chen Chen, Daniele E. Asoni, David Barrera, George Danezis, and Adrian Perrig. HORNET: High-speed onion routing at the network layer. In *Proceedings of the 22nd ACM Conference on Computer and Communications Security (CCS '15)*, October 2015.

[12] Wei Dai. Pipenet 1.0. Post to Cypherpunks mailing list, January 1998.

[13] George Danezis. Statistical disclosure attacks: Traffic confirmation in open environments. In Gritzalis, Vimercati, Samarati, and Katsikas, editors, *Proceedings of Security and Privacy in the Age of Uncertainty, (SEC2003)*, pages 421–426. IFIP TC11, Kluwer, May 2003.

[14] George Danezis, Claudia Diaz, and Carmela Troncoso. Two-sided statistical disclosure attack. In Nikita Borisov and Philippe Golle, editors, *Proceedings of the Seventh Workshop on Privacy Enhancing Technologies (PET 2007)*. Springer, June 2007.

[15] George Danezis, Roger Dingledine, and Nick Mathewson. Mixminion: Design of a Type III Anonymous Remailer Protocol. In *Proceedings of the 2003 IEEE Symposium on Security and Privacy*, pages 2–15, May 2003.

[16] George Danezis and Paul Syverson. Bridging and fingerprinting: Epistemic attacks on route selection. In Nikita Borisov and Ian Goldberg, editors, *Proceedings of the Eighth International Symposium on Privacy Enhancing Technologies (PETS 2008)*, pages 133–150. Springer, July 2008.

[17] George Danezis and Carmela Troncoso. Vida: How to use bayesian inference to de-anonymize persistent communications. In Ian Goldberg and Mikhail J. Atallah, editors, *Proceedings of Privacy Enhancing Technologies, 9th International Symposium (PETS 2009)*, volume 5672 of *Lecture Notes in Computer Science*, pages 56–72. Springer, August 2009.

[18] Claudia Diaz and Bart Preneel. Reasoning about the anonymity provided by pool mixes that generate dummy traffic. In *Proceedings of 6th Information Hiding Workshop (IH 2004)*, LNCS, May 2004.

[19] Claudia Diaz and Bart Preneel. Taxonomy of mixes and dummy traffic. In *Proceedings of I-NetSec04: 3rd Working Conference on Privacy and Anonymity in Networked and Distributed Systems*, August 2004.

[20] Claudia Diaz and Andrei Serjantov. Generalising mixes. In Roger Dingledine, editor, *Proceedings of Privacy Enhancing Technologies workshop (PET 2003)*, pages 18–31. Springer-Verlag, LNCS 2760, March 2003.

[21] Roger Dingledine, Michael J. Freedman, and David Molnar. The free haven project: Distributed anonymous storage service. In *International Workshop on Designing Privacy Enhancing Technologies: Design Issues in Anonymity and Unobservability*, pages 67–95, New York, NY, USA, 2001. Springer-Verlag New York, Inc.

[22] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The second-generation onion router. In *Proceedings of the 13th USENIX Security Symposium*, August 2004.

[23] Roger Dingledine, Andrei Serjantov, and Paul Syverson. Blending different latency traffic with alpha-mixing. In George Danezis and Philippe Golle, editors, *Proceedings of the Sixth Workshop on Privacy Enhancing Technologies (PET 2006)*, pages 245–257. Springer, June 2006.

[24] Roger Dingledine, Vitaly Shmatikov, and Paul Syverson. Synchronous batching: From cascades to free routes. In *Proceedings of Privacy Enhancing Technologies workshop (PET 2004)*, volume 3424 of *LNCS*, pages 186–206, May 2004.

[25] Roger Dingledine and Paul Syverson. Reliable MIX Cascade Networks through Reputation. In Matt Blaze, editor, *Proceedings of Financial Cryptography (FC '02)*. Springer-Verlag, LNCS 2357, March 2002.

[26] John Douceur. The Sybil Attack. In *Proceedings of the 1st International Peer To Peer Systems Workshop (IPTPS 2002)*, March 2002.

[27] Matthew Edman and Paul F. Syverson. AS-awareness in Tor path selection. In Ehab Al-Shaer, Somesh Jha, and Angelos D. Keromytis, editors, *Proceedings of the 2009 ACM Conference on Computer and Communications Security, CCS 2009*, pages 380–389. ACM, November 2009.

[28] Christoph Egger, Johannes Schlumberger, Christopher Kruegel, and Giovanni Vigna. Practical attacks against the i2p network. In *Proceedings of the 16th International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2013)*, October 2013.

[29] Peipeng Liu et al. Empirical measurement and analysis of i2p routers. *Journal of Networks*, 9(9):2269–2278, September 2014.

[30] Nathan Evans, Roger Dingledine, and Christian Grothoff. A practical congestion attack on Tor using long paths. In *Proceedings of the 18th USENIX Security Symposium*, August 2009.

[31] Nick Feamster and Roger Dingledine. Location diversity in anonymity networks. In *Proceedings of the Workshop on Privacy in the Electronic Society (WPES 2004)*, October 2004.

[32] Xinwen Fu, Bryan Graham, Riccardo Bettati, and Wei Zhao. Analytical and empirical analysis of countermeasures to traffic analysis attacks. In *Proceedings of the 2003 International Conference on Parallel Processing*, pages 483–492, 2003.

[33] Sharad Goel, Mark Robson, Milo Polte, and Emin Gun Sirer. Herbivore: A Scalable and Efficient Protocol for Anonymous Communication. Technical Report 2003-1890, Cornell University, Ithaca, NY, February 2003.

[34] Marcin Gogolewski, Marek Klonowski, and Miroslaw Kutylowski. Local view attack on anonymous communication. In *Proceedings of ESORICS 2005*, September 2005.

[35] David M. Goldschlag, Michael G. Reed, and Paul F. Syverson. Hiding Routing Information. In R. Anderson, editor, *Proceedings of Information Hiding: First International Workshop*, pages 137–150. Springer-Verlag, LNCS 1174, May 1996.

[36] Philippe Golle and Ari Juels. Dining cryptographers revisited. In *Proceedings of Eurocrypt 2004*, May 2004.

[37] Ceki Gülcü and Gene Tsudik. Mixing E-mail with Babel. In *Proceedings of the Network and Distributed Security Symposium - NDSS '96*, pages 2–16. IEEE, February 1996.

[38] Michael Herrmann and Christian Grothoff. Privacy implications of performance-based peer selection by onion routers: A real-world case study using i2p. In *Proceedings of the 11th Privacy Enhancing Technologies Symposium (PETS 2011)*, July 2011.

[39] Michael Herrmann and Christian Grothoff. Privacy implications of performance-based peer selection by onion routers: A real-world case study using i2p. In *Proceedings of the 11th Privacy Enhancing Technologies Symposium (PETS 2011)*, July 2011.

[40] Rob Jansen, Aaron Johnson, and Paul Syverson. LIRA: Lightweight Incentivized Routing for Anonymity. In *Proceedings of the Network and Distributed System Security Symposium - NDSS'13*. Internet Society, February 2013.

[41] jrandom (Pseudonym). Invisible internet project (i2p) project overview. Design document, August 2003.

[42] Dogan Kesdogan, Dakshi Agrawal, and Stefan Penz. Limits of anonymity in open environments. In Fabien Petitcolas, editor, *Proceedings of Information Hiding Workshop (IH 2002)*. Springer-Verlag, LNCS 2578, October 2002.

[43] Dogan Kesdogan, Jan Egner, and Roland Büschkes. Stop-and-go MIXes: Providing probabilistic anonymity in an open system. In *Proceedings of Information Hiding Workshop (IH 1998)*. Springer-Verlag, LNCS 1525, 1998.

[44] Albert Kwon, Mashael AlSabah, David Lazar, Marc Dacier, and Srinivas Devadas. Circuit fingerprinting attacks: Passive deanonymization of tor hidden services. In *Proceedings of the 24th Usenix Security Symposium*, August 2015.

[45] Brian N. Levine, Michael K. Reiter, Chenxi Wang, and Matthew K. Wright. Timing attacks in low-latency mix-based systems. In Ari Juels, editor, *Proceedings of Financial Cryptography (FC '04)*, pages 251–265. Springer-Verlag, LNCS 3110, February 2004.

[46] Brian Neil Levine and Clay Shields. Hordes — A Multicast Based Protocol for Anonymity. *Journal of Computer Security*, 10(3):213–240, 2002.

[47] Petar Maymounkov and David Mazières. Kademlia: A peer-to-peer information system based on the xor metric. In *Revised Papers from the First International Workshop on Peer-to-Peer Systems*, IPTPS '01, pages 53–65, London, UK, UK, 2002. Springer-Verlag.

[48] Jon McLachlan and Nicholas Hopper. Don't clog the queue: Circuit clogging and mitigation in P2P anonymity schemes. In *Proceedings of Financial Cryptography (FC '08)*, January 2008.

[49] Prateek Mittal, Ahmed Khurshid, Joshua Juen, Matthew Caesar, and Nikita Borisov. Stealthy traffic analysis of low-latency anonymous communication using

throughput fingerprinting. In *Proceedings of the 18th ACM conference on Computer and Communications Security (CCS 2011)*, October 2011.

[50] Ulf Möller, Lance Cottrell, Peter Palfrader, and Len Sassaman. Mixmaster Protocol — Version 2. IETF Internet Draft, July 2003.

[51] Steven J. Murdoch. Hot or not: Revealing hidden services by their clock skew. In *Proceedings of CCS 2006*, November 2006.

[52] Steven J. Murdoch and George Danezis. Low-cost traffic analysis of Tor. In *Proceedings of the 2005 IEEE Symposium on Security and Privacy*. IEEE CS, May 2005.

[53] Steven J. Murdoch and Robert N. M. Watson. Metrics for security and performance in low-latency anonymity networks. In Nikita Borisov and Ian Goldberg, editors, *Proceedings of the Eighth International Symposium on Privacy Enhancing Technologies (PETS 2008)*, pages 115–132. Springer, July 2008.

[54] Arjun Nambiar and Matthew Wright. Salsa: A structured approach to large-scale anonymity. In *Proceedings of CCS 2006*, November 2006.

[55] Tsuen-Wan "Johnny" Ngan, Roger Dingledine, and Dan S. Wallach. Building Incentives into Tor. In Radu Sion, editor, *Proceedings of Financial Cryptography (FC '10)*, January 2010.

[56] Rebekah Overdorf, Marc Juarez, Gunes Acar, Rachel Greenstadt, and Claudia Diaz. How unique is your .onion? an analysis of the fingerprintability of tor onion services. In *Proceedings of the 23rd ACM Conference on Computer and Communications Security (CCS '17)*, November 2017.

[57] Lasse Øverlier and Paul Syverson. Locating hidden servers. In *Proceedings of the 2006 IEEE Symposium on Security and Privacy*. IEEE CS, May 2006.

[58] Andriy Panchenko, Benedikt Westermann, Lexi Pimenidis, and Christer Andersson. SHALON: Lightweight anonymization based on open standards. In *Proceedings of 18th International Conference on Computer Communications and Networks*, August 2009.

[59] Fernando Perez-Gonzalez and Carmela Troncoso. Understanding statistical disclosure: A least squares approach. In *Proceedings of the 12th Privacy Enhancing Technologies Symposium (PETS 2012)*. Springer, July 2012.

[60] Andreas Pfitzmann, Birgit Pfitzmann, and Michael Waidner. ISDN-mixes: Untraceable communication with very small bandwidth overhead. In *Proceedings of the*

GI/ITG Conference on Communication in Distributed Systems, pages 451–463, February 1991.

[61] Jean-François Raymond. Traffic Analysis: Protocols, Attacks, Design Issues, and Open Problems. In H. Federrath, editor, *Proceedings of Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability*, pages 10–29. Springer-Verlag, LNCS 2009, July 2000.

[62] Michael Reiter and Aviel Rubin. Crowds: Anonymity for web transactions. *ACM Transactions on Information and System Security*, 1(1), June 1998.

[63] Marc Rennhard and Bernhard Plattner. Introducing MorphMix: Peer-to-Peer based Anonymous Internet Usage with Collusion Detection. In *Proceedings of the Workshop on Privacy in the Electronic Society (WPES 2002)*, November 2002.

[64] Marc Rennhard, Sandro Rafaeli, Laurent Mathy, Bernhard Plattner, and David Hutchison. Analysis of an Anonymity Network for Web Browsing. In *Proceedings of the IEEE 7th Intl. Workshop on Enterprise Security (WET ICE 2002)*, pages 49–54, June 2002.

[65] Andrei Serjantov, Roger Dingledine, and Paul Syverson. From a trickle to a flood: Active attacks on several mix types. In Fabien Petitcolas, editor, *Proceedings of Information Hiding Workshop (IH 2002)*. Springer-Verlag, LNCS 2578, October 2002.

[66] Andrei Serjantov and Peter Sewell. Passive attack analysis for connection-based anonymity systems. In *Proceedings of ESORICS 2003*, October 2003.

[67] Micah Sherr, Matt Blaze, and Boon Thau Loo. Scalable link-based relay selection for anonymous routing. In Ian Goldberg and Mikhail J. Atallah, editors, *Proceedings of Privacy Enhancing Technologies, 9th International Symposium (PETS 2009)*, volume 5672 of *Lecture Notes in Computer Science*, pages 73–93. Springer, August 2009.

[68] Vitaly Shmatikov and Ming-Hsiu Wang. Timing analysis in low-latency mix networks: Attacks and defenses. In *Proceedings of ESORICS 2006*, September 2006.

[69] Carmela Troncoso Simon Oya and Fernando Pérez-González. Do dummies pay off? limits of dummy traffic protection in anonymous communications. In *Proceedings of the 14th Privacy Enhancing Technologies Symposium (PETS 2014)*, July 2014.

[70] Gildas Nya Tchabe and Yinhua Xu. Anonymous communications: A survey on i2p. CDC Publication, 2014.

[71] Juan Pablo Timpanaro, Thibault Cholez, Isabelle Chrisment, and Olivier Festor. Evaluation of the Anonymous I2P Network's Design Choices Against Performance and Security. In *Proceedings of the 1st International Conference on Information Systems Security and Privacy (ICISSP 2015)*, pages 46–55, Angers, France, February 2015. SciTePress.

[72] Carmela Troncoso, Benedikt Gierlichs, Bart Preneel, and Ingrid Verbauwhede. Perfect matching statistical disclosure attacks. In Nikita Borisov and Ian Goldberg, editors, *Proceedings of the Eighth International Symposium on Privacy Enhancing Technologies (PETS 2008)*, pages 2–23. Springer, July 2008.

[73] Dipal Vashi and Girish Khilari. Performance improvement in i2p using ssl. *International Journal of Science, Engineering and Technology Research*, 4(5):1454–1456, May 2015.

[74] Hasib Vhora and Girish Khilari. Defending eclipse attack in i2p using structured overlay network. *International Journal of Science, Engineering and Technology Research*, 4(5):1515–1518, May 2015.

[75] Michael Waidner and Birgit Pfitzmann. The dining cryptographers in the disco: Unconditional sender and recipient untraceability. In *Proceedings of EUROCRYPT 1989*. Springer-Verlag, LNCS 434, 1990.

[76] Rungrat Wiangsripanawan, Willy Susilo, and Rei Safavi-Naini. Design principles for low latency anonymous network systems secure against timing attacks. In *Proceedings of the fifth Australasian symposium on ACSW frontiers (ACSW '07)*, pages 183–191, Darlinghurst, Australia, Australia, 2007. Australian Computer Society, Inc.

[77] Matthew Wright, Micah Adler, Brian Neil Levine, and Clay Shields. The Predecessor Attack: An Analysis of a Threat to Anonymous Communications Systems. *ACM Transactions on Information and System Security (TISSEC)*, 4(7):489–522, November 2004.

[78] Sebastian Zander and Steven J. Murdoch. An improved clock-skew measurement technique for revealing hidden services. In *Proceedings of the 17th USENIX Security Symposium*, July 2008.

[79] Ye Zhu, Xinwen Fu, Bryan Graham, Riccardo Bettati, and Wei Zhao. On flow correlation attacks and countermeasures in mix networks. In *Proceedings of Privacy Enhancing Technologies workshop (PET 2004)*, volume 3424 of *LNCS*, pages 207–225, May 2004.

A Prílohy

K diplomovej práci je priložené DVD so skomprimovanými dátami, konfiguračnými súbormi a zdrojovými kódmi k diplomovej práci.

Hlavný adresár obsahuje dva podadresáre `codes` a `data`. Prvý obsahuje konfigururačné súbory a zdrojové kódy použité pri diplomovej práci. Druhý obsahuje všetky namerané dáta, s výnimkou logov pre `localhost`. Tie neboli pridané z jednak dôvodu veľkosti súborov a z dôvodu, že vo výsledkoch sa nepoužívajú.

Štruktúra adresára `codes`.

- configs - konfiguračné súbory pre rsyslog, Tor, Nginx
- correlation - skripty na výpočer korelačného koeficientu
- graphs - .ods súbory s tabulkami a grafmi
- i2p_codes - skripty na spracovanie I2P logov
- lcta - skripty na spracovanie dát pre útok s preťažením
- rand_site_gen - program na náhodnú generáciu web stránok
- setups - skripty na spúšťanie a zastavovanie servera
- tor_codes - skripty na spracovanie Tor logov

Štruktúra adresára `data`.

- i2p_active - I2P, aktívne ovplyvnenie
- i2p_train_part1 - I2P, trénovacia množina, prvá časť
- i2p_train_part2 - I2P, trénovacia množina, druhá časť
- i2p_train_merged - I2P, trénovacia množina, spojené obe časti
- i2p_result_set - I2P, hlavná množina
- p2p_10mb - P2P, 10Mbps
- p2p_5mb - P2P, 5Mbps
- p2p_1mb - P2P, 1Mbps
- tor_active - Tor, aktívne ovplyvnenie
- tor_passive_low_data_1 - Tor, znížená veľkos prenášaných dát, prvá časť
- tor_passive_low_data_2 - Tor, znížená veľkos prenášaných dát, druhá časť

- tor_passive_low_data_merged - Tor, znížená veľkos prenášaných dát, spojené obe časti
- tor_train - Tor, trénovacia množina,
- tor_result_set - Tor, hlavná množina
- lcta - dáta pre útok s preťažením

Postupy pre získanie výsledkov. Budeme uvádzať trojicu adresár z ktorého sa skript spúšťa, názov skriptu a poznámku o úlohe skriptu. Na oddelenie používame bodkočiarku. Bodkou označujeme začiatočný adresár.

Tor

1. . ; get_ips.sh ; získa koncové `IP` adresy reťazí
2. . ; split.sh ; rozdelí súbory na jednotlivé pripojenia
3. . ; check.sh ; skontroluje či sa pripojili len IP adresy získané z get_ips.sh
4. . ; error_rate.sh / params_find.py ; výsledky

I2P

1. ./client ; rm_suc.sh ; odstránenie sudo príkazov z logu
2. ./client ; filt_dns.sh ; odstráni komunikáciu s IP adresami z DNS logu
3. ./server; filt_dns.sh ; odstráni komunikáciu s IP adresami z DNS logu
4. . ; unify_timestamp.pl ; zjednotenie časových pečiatok
5. ./server ; split.sh ; rozdelenie na pripojenia
6. ./client ; split.sh ; rozdelenie na pripojenia
7. . ; stream_choose.sh ; označí súbory s dátami smerujúcimi od serveru ku klientovi
8. . ; bck_stream_choose.sh ; označí súbory s dátami smerujúcimi od klienta k serveru
9. . ; clean_non_cells.sh ; vytvorí súbory s vyfiltrovanými paketmi nedostatočnej veľkosti
10. . ; i2p_error_rate.sh / i2p_params_find.py ; výsledky

P2P

1. ./p2p ; filt_dns.sh ; odstráni komunikáciu s IP adresami z DNS logu
2. ./p2p ; parse ; spracovanie
3. . ; results.sh ; výsledky

Útok s preťažením

1. ./attacker ; filter.sh ; odstráni komunikáciu s IP adresami mimo experimentu
2. . ; unify_timestamp.pl ; zjednotí časové pečiatky
3. ./attacker ; parse.sh ; spracovanie
4. ./server ; server_parse.sh ; spracovanie
5. ./attacker ; split.pl ; rozdelí podľa pripojení klienta na server
6. ./attacker ; ip_split.pl ; rozdelí súbory v pripojeniach podľa IP adresy
7. ./attacker ; lsplit.sh ; odstráni nepotrebné pakety
8. ./attacker ; latencies.pl ; vytvorí rad časov odpovedí