UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

ANONYMITA V POČÍTAČOVÝCH SIEŤACH S VYUŽITÍM ONION ROUTINGU

BAKALÁRSKA PRÁCA

2012 **Peter CSIBA**

UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

ANONYMITA V POČÍTAČOVÝCH SIEŤACH S VYUŽITÍM ONION ROUTINGU

BAKALÁRSKA PRÁCA

Študijný program:	Informatika
Študijný odbor:	9.2.1 Informatika 2508
Školiace pracovisko:	Katedra informatiky
Školiteľ:	RNDr. Peter Gaži PhD.

Bratislava 2012 **Peter CSIBA**

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Peter Csiba
Študijný program: informatika (Jednoodborové štúdium, bakalársky I. st., denná forma)
Študijný odbor: 9.2.1. informatika
Typ záverečnej práce: bakalárska
Jazyk záverečnej práce: slovenský

Názov: Anonymita v počítačových sieťach s využitím onion routingu

Cieľ: Jedným z nástrojov na zabezpečenie anonymity pri komunikácii v počítačovej sieti je technika onion routing navrhnutá Reedom, Syversonom a Goldschlagom a patentovaná v roku 1998 americkou armádou. TOR (The Onion Router) je počítačovou aplikáciou umožňujúcou využívať protokol onion routing každému a má celosvetovú užívateľskú bázu. Cieľom práce je popísať anonymné siete, protokol Onion Routing a aplikáciu TOR, pričom sa zameria na bezpečnosť ich paradigiem a implementácií. Práca prinesie prehľad existujúcich analýz bezpečnosti TORu, jeho slabín a opíše podobné protokoly, ktoré tieto problémy nemajú. Práca by mala zahŕňať aj implementácie vybraných útokov, prípadne navrhovaných riešení.

Vedúci: Mgr. Peter Gaži, PhD.

Dátum zadania: 26.10.2011

Dátum schválenia: 26.10.2011

doc. RNDr. Daniel Olejár, PhD.
garant študijného programu

.....
študent

.....
vedúci

Poďakovanie

Chcem poďakovať svojmu vedúcemu RNDr. Petrovi Gažimu PhD. za cenné pripomienky a návrhy.

Rovnako chcem poďakovať svojim rodičom za ich odhodlanie, vytrvalosť a obetavosť pri zabezpečovaní kvality môjho vzdelania.

Abstrakt

Práca sa zaoberá anonymnými sieťami a poskytuje prehľad konkrétnych návrhov a implementácií. V prvej časti práca najprv zavádza terminológiu a ďalej sa snaží podrobne popísať vybrané siete a uviesť zoznam ďalších sietí a štandardných útokov s ich stručným opisom. V druhej časti sa snaží detailne analyzovať protokoly Onion routing a Tor, zdôvodniť použitie jednotlivých schém a vysvetliť použitie niektorých útokov.

Kľúčové slová: anonymná komunikácia, počítačové siete, analýza prenosu, Onion routing, Tor, mix

Abstract

This thesis presents an overview of anonymous networks. In the first part it introduces the used terminology and then aims to provide a comprehensive list of anonymous networks and describe in detail some of them. In the second part it aims to analyze the innards of Onion routing and Tor and reason about their design decisions. This paper also aims to give a list of standard attacks against anonymous networks and describe them briefly.

Keywords: anonymous communication, computer networks, traffic analysis, Onion routing, Tor, mix

Obsah

Úvod

V súčastnosti, keď je možné monitorovať prakticky každý náš krok, je anonymita cennou vlastnosťou. To platí aj pre počítačové siete, ako napríklad Internet. Pre bežných užívateľov sa môže prehliadanie webových stránok zdať anonymné a nevystopovateľné, ale v skutočnosti je možné užívateľovu komunikáciu ľahko zachytiť a jednoduchým spôsobom určiť s kým komunikuje a to aj v prípade, keď používa šifrované protokoly ako HTTPS alebo SSH.

V našej práci sa zaoberáme *anonymnými sieťami*, ktorých cieľom je skryť informáciu o tom, kto s kým v sieti komunikuje. Rozoberáme viacero prístupov a niekoľko z nich podrobne analyzujeme. Pritom sa zameriavame na anonymné siete založené na *Onion routingu*, ktoré viacnásobným preposielaním správ zabezpečujú anonymitu a pritom zachovávajú bežné odozvy a prenosové rýchlosti.

Veľkú časť práce venujeme populárnemu Toru, ktorý rozširuje a implementuje pôvodný Onion routing a zahŕňa celosvetovú sieť pozostávajúcu z niekoľko tisíc routrov. Tor je využívaný napríklad v spravodajstve, armáde, krajinách s cenzúrovaným Internetom alebo pri bežnom prehliadaní webových stránok.

Motivácie anonymity môžu byť rôzne a preto uvedieme niekoľko príkladov využitia:

- Pacient, ktorý zisťuje informácie o svojej chorobe.
- Obyvateľ, ktorý chce poukázať na korupciu.
- Užívateľ, ktorý nechce poskytnúť svoje dáta webovým serverom.
- Firma, ktorá získava údaje o konkurencií.
- Cenzúrovaný užívateľ, ktorý sa chce dostať k slobodným informáciám.

Anonymné siete využívajú vo svoj prospech aj útočníci, ktorý sa chcú skryť pred vyššími autoritami. Tomu sa ale v anonymných sieťach nedá úplne zabrániť. Niektoré systémy sa snažia také správy detekovať a nepustiť von zo siete.

Práca je členená do dvoch hlavných kapitol. V prvej kapitole definujeme terminológiu a poskytujeme široký prehľad anonymných sietí, používaných paradigiem a možných útokov na anonymné siete. Niektoré z nich podrobne rozoberáme. V druhej kapitole do detailov rozoberáme protokol Tor a pôvodný Onion routing.

1 Anonymné siete

1.1 Úvod

Kapitola sa snaží poskytnúť ucelený prehľad rôznych protokolov zabezpečujúcich anonymnú komunikáciu. Pritom sa naschvál vyhýba protokolom podobným Onion routingu a Toru, ktorým je určená nasledujúca kapitola 2.

Na rozdiel od prehľadov anonymných sietí [29, 11, 12] kapitola navyše detailne opisuje vybrané protokoly, vysvetľuje ich základné princípy a poukazuje na možné nedostatky.

V kapitole 1.2 zavádzame jednotnú terminológiu používanú v celej práci. V ďalších kapitolách 1.3, 1.4 a 1.5 popisujeme protokoly poskytujúce anonymitu pri rôzne silných predpokladoch. V kapitole 1.6 stručne spomíname ďalšie anonymné protokoly a v kapitole 1.7 uvádzame krátky zoznam štandardných útokov voči anonymným sieťam.

1.2 Terminológia

Postupne definujeme základné pojmy anonymných sietí používané v tejto práci podľa [59] a [39]. Definície sme upravili a doplnili podľa našich potrieb. Všetky množiny sú konečné.

1.2.1 Prostredie

Používame model, kde *odosielatelia* posielajú *správy* pre *príjemcov* využitím komunikačnej *siete*. Všetky objekty schopné vytvárať alebo spracovať správy vnútri siete nazývame *účastníci*, účastníka používajúceho sieť na svoje vlastné účely nazývame *klient* a účastníka, ktorý nie je klientom nazývame *zariadenie*. Množina zariadení je verejne známa a každé zariadenie má svoj verejný kľúč pre asymetrické šifrovanie. Zariadenia medzi sebou komunikujú po šifrovaných komunikačných kanáloch.

Všetky tvrdenia, ak nepovieme inak, sú formulované z pohľadu *útočníka*, ktorého záujmom môže byť sledovanie existujúcej komunikácie, priradenie správ k účastníkom alebo manipulácia samotnej komunikácie.

Nepredpokladáme iba *pasívneho* útočníka, ktorí odposlúcha komunikačné kanály, ale aj *aktívneho* útočníka vo vnútri systému, schopného účastniť sa na bežnej komunikácií

a kontrolujúceho aspoň časť systému. Útočníka ďalej delíme na *globálneho*, schopného pozorovať všetky správy v systéme, a na *vnútorného*, ktorý vie pozorovať len komunikáciu medzi zlomkom[1] všetkých účastníkov. Útočník v našom modeli nevie získať informácie o obsahu správ v sieti[2].

Predpokladáme, že útočník používa všetky jemu dostupné fakty na odvodenie jeho *objektov záujmu*. Príkladom objektu záujmu je väzba medzi správou a jej odosielateľom. Ďalej, útočník svoje *znalosti*[3] nezabúda a teda jeho znalosť sa v čase iba zväčšuje.

1.2.2 Anonymita

Aby bolo možné hovoriť o anonymite účastníka[4], musí existovať viacero účastníkov prejavujúcich rovnaké vlastnosti[5].

Anonymita je stav, keď účastník nie je identifikovateľný vo viacprvkovej množine účastníkov, nazývanej množina anonymity[6] [59].

Množina anonymity je množina všetkých potenciálnych účastníkov pre daný objekt záujmu. Odosielateľ je anonymný len vnútri množiny potenciálnych odosielateľov, jeho *odosielateľovej množiny anonymity*, čo môže byť teoreticky množina všetkých odosielateľov na svete. Podobne definujeme pre príjemcu jeho *príjemcovu množinu anonymity*. Tieto dve množiny môžu byť rôzne, môžu byť rovnaké, môžu mať spoločný neprázdny prienik. Množiny anonymity sa môžu meniť v čase, ale keďže útočník podľa našich predpokladov nezabúda, tak sa môžu len zmenšovať.

1.2.3 Nespojiteľnosť

Pre útočníka sú dva alebo viaceré objekty záujmu *nespojiteľné*, ak pravdepodobnosť, že tieto objekty vykonávajú spoločné operácie, sa jeho pozorovaniami nezvyšuje a zostáva

[1] Pod zlomkom účastníkov, resp. zariadení, sa v každom protokole myslí niečo iné. Najčastejšie sa predpokladá, že pasívny útočník nikdy nemá pod kontrolou dostatok účastníkov potrebných na to, aby vedel *jednoznačne* odhaliť identity klientov pre danú správu.

[2] Komunikačné kanály sú šifrované, ak sa nepovie inak.

[3] V literatúre sa zvyčajne *znalosť* útočníka definuje ako pravdepodobnosť jeho objektov záujmu. Viac znalosti potom znamená presnejšie pravdepodobnosti.

[4] Ako napríklad klienta komunikácie, právneho subjektu alebo počítača

[5] Za vlastnosť považujeme napríklad schopnosť prijímať, posielať alebo spracovávať správy.

[6] Návrh formalizovaného prístupu k anonymite sa uvádza v [39], v kapitole 2.5.

na zanedbateľnej úrovni.

Pomocou pojmu nespojiteľnosti vieme definovať pojmy anonymity. *Anonymita odosielateľa* je, keď daná správa je nespojiteľná s odosielateľom. Podobne, *anonymita príjemcu* je, keď daná správa je nespojiteľná s príjemcom. *Vzájomná anonymita* je, keď daný odosielateľ a daný príjemca sú nespojiteľní. Inými slovami, nedá sa zistiť, či daný odosielateľ komunikuje v systéme s daným príjemcom.

Ak je odhalená anonymita odosielateľa aj anonymita príjemcu, tak je odhalená aj vzájomná anonymita, lebo vieme priradiť k správam ich príjemcov aj odosielateľov. Ak je odhalená vzájomná anonymita, tak to nemusí odhaliť anonymitu príjemcu alebo anonymitu odosielateľa, ak nevieme určiť ktoré správy si medzi sebou posielajú. Najčastejšie nás zaujíma anonymita príjemcu a vzájomná anonymita.

1.2.4 Pseudonymita

Pseudonym je identifikátor užívateľov. Hovoríme, že užívateľ *drží* svoj pseudonym. Byť *pseudonymný* je stav užívateľa, keď užívateľ používa svoj pseudonym ako svoj identifikátor [59].

Definícia umožňuje viacerým užívateľom držať jeden pseudonym. Takýto pseudonym budeme nazývať *skupinový pseudonym* a v ďalšom budeme predpokladať, že pseudonym drží práve jeden účastník.

Treba poznamenať, že aj keď sa pojmy anonymity a pseudonymity slovne podobajú, vyjadrujú iný typ objektov. Anonymita je stav účastníka a pseudonym je atribút účastníka.

Príklad pseudonymu je napríklad emailová adresa.

1.2.5 Sila anonymity

Úvod. V práci Kellyho [39] sa spomína viacero druhov metrík, vhodných na meranie "sily" anonymity.

V tejto kapitole budeme modelovať anonymitu z pohľadu útočníka podľa pravdepodobnostného[7] modelu navrhnutého Kesdoganom, Egnerom a Büschkesom [41].

[7] V literatúre sa deterministické modely anonymity nepoužívajú, keďže môže byť veľmi ťažké presne vyjadriť počty objektov záujmov pre jednotlivé protokoly.

Vzhľadom na danú správu M definujeme Ψ ako množinu všetkých užívateľov a $r \in R$ ako rolu užívateľa ($R = \{odosielatel, prijemca\}$). Nech U je distribúcia pravdepodobnosti účastníkov $u \in \Psi$ majúcich rolu r vzhľadom na správu M, po pozorovaniach útočníka[8]. Za útočníkov experiment považujeme vykonanie algoritmického útoku, ktorý mu môže priniesť čiastočné alebo úplne informácie dopomáhajúce k odhaleniu anonymity účastníkov.

Náhodnú premennú[9] U označujeme ako *pravdepodobnostnú distribúciu r-anonymity*. Formálne $U : \Psi \times R \mapsto [0, 1]$ a $\sum_{u \in \Psi} U(u, r) = 1$. Ak to kontext nevyžaduje, vynechávame explicitné určenie správy M a roly r.

Niekoľko príkladov vypočítaných hodnôt mier anonymity uvádzame v kapitole 1.5.5.

Mohutnosť množiny anonymity. Z definície anonymity vieme priamym spôsobom definovať silu anonymity ako mohutnosť množiny anonymity. Inými slovami je to počet účastníkov, ktorým náhodná premenná U priraďuje nenulovú pravdepodobnosť. Formálne nás zaujíma stredná hodnota $E(S)$, kde $S = |\{u, u \in \Psi \wedge U(u) \neq 0\}|$.

Výhodou tejto miery je relatívna jednoduchosť a jej nevýhodou je, že nevieme z nej určiť pravdepodobnosť odhalenia jednotlivých účastníkov, respektíve distribúciu pravdepodobnosti vnútri množiny anonymity.

Táto miera nám poskytuje iba intuíciu o tom, že medzi koľkými účastníkmi je daný účastník potenciálne anonymný.

Entropia množiny anonymity. Entropia je pojem z teórie informácie zavedený Shannonom [69]. Definovanie miery anonymity na základe entropie sa snaží zahrnúť aj pravdepodobnosti jednotlivých prvkov množiny Ψ.

Pre pravdepodobnostnú distribúciu r-anonymity U definujeme *efektívnu veľkosť množiny anonymity* S, rovnú entropií tejto distribúcie [67]. Inými slovami:

$$S = -\sum_{u \in \Psi} p_u \lg(p_u),$$

kde $p_u = U(u, r)$.

[8] V odbornej literatúre sa pravdepodobnosť po experimente nazýva *posteriórna pravdepodobnosť*.

[9] Náhodná premenná priraďuje pravdepodobnosti jednotlivým prvkom množiny a v súčte sú tieto pravdepodobnosti rovné 1. Stredná hodnota je vážený priemer prvkov množiny.

Túto veličinu je možné interpretovať ako počet bitov informácie, ktoré útočník potrebuje na identifikovanie užívateľa u. Základné pozorovania sú:

- Vždy platí $0 \leq S \leq lg|\Psi|$. Anonymitu považujeme za *perfektnú*, ak nastáva $S = lg|\Psi|$.
- Ak $S = 0$, tak systém nezaručuje žiadnu anonymitu.

Iné. Zaujímavú mieru vzhľadom na konkrétneho účastníka systému, navrhli dizajnéri protokolu Crowds [62]. Zavádzajú pojmy ako *absolútne súkromie*, *za hranicou podozrenia*, *pravdepodobne nevinný*, *odhalený* a *dokázateľne odhalený* na základe distribúcie pravdepodobnosti vnútri množiny anonymity. Viac o Crowds v kapitole 1.6.1.

1.2.6 Reputácia

Reputácia je miera, ktorá sa snaží odzrkadľovať dôveryhodnosť entity v budúcnosti, na základe jej správania v minulosti [20]. Miera reputácie môže byť definovaná na prirodzených číslach, alebo ako normalizovaná pravdepodobnosť vyjadrujúca dôveryhodnosť.

Množina operácií verejne známych entít v sieti sa rozdelí na *dobré* a *zlé*. Za dobré možno považovať napríklad správne doručenie správy, za zlé napríklad nepreposlanie správy ďalej. Dobrými operáciami si entita zvyšuje *pozitívnu reputáciu* a zlými operáciami si zvyšuje *negatívnu reputáciu*.

V reálnom svete, kde sú transakcie krátke alebo verejne overiteľné, sú reputačné systémy úspešným nástrojom. Napríklad eBay, celosvetovo rozšírená online aukcia, má vytvorený aukčný systém na základe feedbacku z každej obchodnej transakcie. Spätnú väzbu od kupujúceho má vyjadriť spokojnosť s aktivitami predajcu [39].

V anonymných sieťach majú reputačné systémy problémy [20, 25]. Overiť správne doručenie vie len odosielateľ správy a vystopovať dôvod zlyhania môže byť ťažké. Na druhej strane, útočník môže použiť viacero identít[10], aby si úspešným posielaním správ medzi nimi zvyšoval pozitívnu reputáciu.

Tieto problémy skúmajú v [20] a ich záverom sú ďalšie otázky. Navrhujú použitie grafov reputácie, podobným grafu dôvery certifikačných autorít.

[10] Využívanie viacero identít jedným útočníkom sa v anglickej odbornej terminológií nazýva *pseudospoofing*.

1.3 Večerajúci kryptológovia

1.3.1 Pôvodný protokol

Úvod. Problém večerajúcich kryptológov[11] je základným problémom anonymných sietí. Uvádzame upravené pôvodné znenie D. Chauma [6]:

Traja kryptológovia sedia na večeri za okrúhlym stolom. Za večeru platí buď jeden z kryptológov, alebo NSA[12]. Traja kryptológovia rešpektujú právo vykonať anonymnú platbu, ale sú zvedaví, či zaplatila NSA, respektíve či zaplatil jeden z nich.

Protokol navrhnutý Chaumom umožňuje v jednom svojom kole broadcastovať[13] jeden bit informácie anonymne, za predpokladu, že každý klient protokolu je čestný. Tento protokol je považovaný za dôkaz toho, že nepodmienená anonymita je dosiahnuteľná, avšak protokol sa v praxi nepoužíva. Rozšírenie protokolu, *Večerajúci kryptológovia na diskotéke* [74], umožňuje detekovať nečestných užívateľov.

Priebeh. Popíšeme priebeh protokolu pre n účastníkov.

V prvej fáze si každá dvojica klientov i a j dohodne spoločný náhodný bit $M_{ij} = M_{ji}$, ktorý je ich spoločným tajomstvom. V druhej fáze každý klient i zverejní informáciu P_i, čím je XOR[14] jeho spoločných tajomstiev a informácie, či zaplatil:

$$P_i = B_i \oplus (\bigoplus_{j \neq i} M_{ij}),$$

kde $B_i = 1$ práve vtedy keď klient i za večeru zaplatil. Ak sa nepovie inak, tak indexujeme od 1 po n.

Označme $A = \bigoplus_i P_i$. Keďže všetky M_{ij} boli zarátané v A dvakrát, tak jednoduchými úpravami dostávame:

$$A = \bigoplus_i P_i = \bigoplus_i (B_i \oplus (\bigoplus_{j \neq i} M_{ij})) = \bigoplus_i B_i. \tag{1}$$

Podľa nášho predpokladu maximálne pre jedného klienta platí $B_i = 1$. A preto je $A = 1$ práve vtedy, keď niektorý z klientov zaplatil. Vo všeobecnosti A informuje o parite počtu platiacich klientov.

[11] Názov je podobný *Problému večerajúcich filozofov* z oblasti synchronizácie procesov.

[12] NSA - United States National Security Agency je hlavnou organizáciou pre bezpečnosť v USA. Slovenským ekvivalentom je NBÚ - národný bezpečnostný úrad.

[13] Poslať všetkým účastníkom.

[14] Slovensky *bitový súčet*.

Dôkaz anonymity. Predpoklady. Uvažujme graf spoločných tajomstiev G', kde vrcholy V sú klienti a hrany E sú ich spoločné tajomstvá. Nech útočník ovláda množinu klientov $C \subsetneq V$ a teda pozná všetky spoločné tajomstvá, ktoré vlastnia C. Označme graf G, ktorý má rovnaké vrcholy ako G' a ktorého hrany sú spoločné tajomstvá, ktoré útočník nepozná. Predpokladajme, že v G neexistuje izolovaný vrchol[15]. Ďalej uvažujme ľubovoľný komponent súvislosti S grafu G. Z predošlého predpokladu $|S| \geq 2$ a $|C| \leq |V| - 2$. V dôkaze nepredpokladáme, že $\sum_{v \in V} B_v \leq 1$. Chceme ukázať, že útočník má rovnakú znalosť pred aj po svojich pozorovaniach. Na presnú formuláciu dokazovaného tvrdenia je nutné zaviesť ďalšie označenia.

Označenia. Označme $m = |S|$, n počet hrán v S, M binárnu maticu susednosti s m riadkami zodpovedajúcim vrcholom S a n stĺpcami zodpovedajúcim hranám S. Označme náhodné premenné K, I a D, ktoré sú binárnymi stĺpcovými vektormi dĺžok n, m a m, pričom premenné K a I sú nezávislé. Vektor K reprezentuje spoločné tajomstvá, *informačný vektor* I reprezentuje hodnoty $B_v, v \in S$ a vektor D reprezentuje hodnoty $P_v, v \in S$. Aritmetické operácie protokolu vyjadruje vzťah $(MK) \oplus I = D$.

Chceme ukázať, že útočník má rovnakú znalosť pred aj po svojich pozorovaniach. Inými slovami, ak d je pozorovaný výsledok a i je hodnota informačného vektora, tak podmienená pravdepodobnosť $P(D = d | I = i) = P(I = i)$.

Dôkaz. Označme $Par(X)$ paritu binárneho vektora X, ktorou je XOR jeho bitov. Vieme[16], že $Par(d) = Par(i)$. Uvažujme systém m lineárnych rovníc o n neznámych $MK = i \oplus d$, kde neznáme sú binárny vektor k. Kvôli konštrukcií M pre každý jeho stĺpec t platí $Par(t) = 0$ a preto sú riadky matice lineárne závislé. Na druhej strane, kvôli súvislosti S je každá vlastná[17] podmnožina riadkov lineárne nezávislá.

To, že sú nezávislé, vieme dokázať sporom. Keby neboli lineárne nezávislé, tak by vybrané riadky museli mať v každom stĺpci paritu 0. Vráťme sa ku grafovej reprezentácií. Vybratý riadok znamená vybratý vrchol. To by znamenalo, že každá hrana prítomná vo vybratých riadkoch by musela byť započítaná v stĺpcoch párny počet krát, a teda práve dvakrát, lebo je tam maximálne dvakrát. A teda pre každú hranu musíme vybrať

[15] Ak by izolovaný vrchol v grafe G existoval, tak by útočník vedel jednoznačne určiť akú informáciu tento klient posiela.

[16] V prípade $Par(d) \neq Par(i)$ niektorý z klientov klamal.

[17] Vlastná podmnožina X množiny Y je podmnožina, ktorá nie je Y. Zapisuje sa $X \subsetneq Y$.

oba jej koncové vrcholy a pre každý riadok všetky jej prítomné hrany. Iteratívnym procesom, podobným prehľadávaniu do šírky, získame takým to spôsobom všetky riadky, čo je spor s predpokladom o vlastnej podmnožine riadkov.

Preto je hodnosť matice M práve $m-1$. A keďže všetky stĺpce M sú párne, tak výsledkami (MK) sú práve všetky párne vektory dĺžky m. A teda má systém 2^{n-m+1} riešení.

Podmienená pravdepodobnosť $P(D=d|I=i)$ je inými slovami pravdepodobnosť, že rovnosť $MK = i \oplus d$ je platná, keď určíme $I=i$. Keďže K je rovnomerne distribuovaná, tak počítaním možností pre všetky možné hodnoty k a vyhovujúce rovností dostávame $P(D=d|I=i) = \frac{2^{n-m+1}}{2^n} = 2^{1-m} = P(I=i)$.

Diskusia. Výsledkom je nepodmienená *perfektná* anonymita odosielateľa. Jedinými predpokladmi boli korektná dohoda spoločných tajomstiev, spoľahlivosť komunikačných liniek a čestnosť klientov. Preto sú Večerajúci kryptológovia veľmi silným teoretickým protokolom.

1.3.2 Nedostatky

Pôvodná verzia protokolu má v praxi nedostatky ako rýchlosť prenosu, potenciálne kolízie a kvadratický počet spoločných tajomstiev. V tejto sekcií ukážeme niektoré možné riešenia na protokole, kde sa klienti snažia odvysielať správy $1|dlzka\_dat(int_{64})|data$. Jednotka na začiatku označuje začiatok správy a nasledujúcich 64 bitov kóduje dĺžku dát. Stále predpokladáme, že účastníci sú čestný, ale už môžu vysielať naraz. Útočník je globálny pasívny a nepozná všetky spoločné tajomstvá.

- **Rýchlosť prenosu.** Po každom kole protokolu je nutné vygenerovať nové spoločné tajomstvá[18]. Ako navrhol už Chaum [6], tak namiesto jednobitových spoločných tajomstiev vygenerujeme dlhé spoločné tajomstvá, takže nemusíme pred každým kolom generovať nové. Alebo za spoločné tajomstvá použijeme inicializačné vektory pre pseudonáhodný generátor čísel[19]. Tieto prístupy nám umožňujú

[18] Keby nové spoločné tajomstvá neboli generované znova, tak by sa hodnoty P_i líšili od predošlého kola pre maximálne dvoch klientov.

[19] Pseudonáhodné generátory čísel sú deterministickými zobrazeniami množiny čísel na seba. Postupným aplikovaním zobrazenia sa generujú nové čísla. Pseudonáhodný generátor s vysokou entropiou je cenným zariadením.

vykonať viacero kôl naraz.

- **Kolízie**. Zo štruktúry správy je zrejmé, že každý klient vie, kedy sa vysiela nejaká správa. Pridáme dve pravidlá. Prvé je, že keď nejaký klient vysiela, iný klient nezačne vysielať. Druhé je, že keď vysielajúci klient uvidí iný výsledný bit A, ako posielal, tak prestane ďalej vysielať.

  Vysielanie správy sa začne len keď začne vysielať nepárny počet klientov. A za týchto predpokladov platí, že až do konca vysielania nejakej správy bude správne vysielať nepárny počet klientov. Lebo A nadobudne hodnotu, ktorú vysiela nepárny počet klientov. Obe hodnoty 0 a 1 nemôžu byť vysielané nepárnym počtom klientov, lebo by v tom prípade vysielalo párny počet klientov. Nevýhodou je, že každý vysielajúci klient musí po každom kole overiť korektnosť svojho prenosu a znovu máme problém s rýchlosťou prenosu.

  Tento problém vieme vyriešiť pravdepodobnostným prístupom. Budeme uvažovať iba platnosť prvého pravidla. Zmeníme štruktúru správy tak, že po prvom bite správy nasleduje n bitov určených na losovanie o právo vysielať. Po prvom kole vysielania správy si každý klient náhodne vygeneruje číslo k od 1 do n a vyšle n-bitový nulový vektor s jednotkou na k-tej pozícií. Právo vysielať má ten klient, ktorého jednotlivý bit je prvý. Kolízia nastane len v prípade, keď prvý jednotkový bit určilo nepárny počet klientov väčší ako dva. Pravdepodobnosť tejto udalosti vieme minimalizovať ľubovoľne. Pri tomto losovacom prístupe netreba čakať po každom kole a je možné celú správu poslať na trikrát. Čaká sa len vtedy, keď nikto nevysiela.

- **Kvadratický počet spoločných tajomstiev**. Od začiatku sme vyžadovali, aby každá dvojica klientov mala spoločné tajomstvo, ktorých je dokopy $\frac{n(n-1)}{2}$. Uvažujme graf spoločných tajomstiev, ktorého štruktúra je verejná. Doteraz sme predpokladali kompletný graf, ale protokol bude anonymný aj v prípade, keď každý vrchol je stupňa aspoň dva. Vo všeobecnosti požadujeme, aby útočník ovládajúci ľubovoľnú k-prvkovú množinu klientov, nevedel rozdeliť graf na časti a tým zmenšiť množinu anonymity [7]. Napríklad, ak útočník pozná všetkých hranových susedov klienta u, tak vie jednoznačne určiť všetky B_u. Riešením je

ľubovoľný $(k+1)$-súvislý graf. V špeciálnom prípade $k = n-1$ riešenie neexistuje.

1.4 Centralizované prístupy

1.4.1 Penet

Úvod. Webový server *anon.penet.fi* bol jeden z prvých verejne dostupných *remailerov*. Remailer je služba, proxy, ktorá preposiela klientské emaily pod iným menom. Remailery je možné rozdeliť na dva druhy, *stavové* a *bezstavové* [20] .

Stavové remailery majú tabuľku, ktorá mapuje každú klientskú emailovú adresu na jeho verejnú, pseudonymnú adresu. Ak klient správu posiela cez remailer, tak remailer zmení hlavičku emailu a správu pošle pod pseudonymnou adresou. Naopak, ak tretia strana chce poslať odpoveď, tak pošle email na pseudonymnú adresu a remailer ju preloží na adresu klienta.

Bezstavové remailery z hlavičky emailu odstránia adresu klienta a pošlú ho ďalej pod svojim menom. Keďže si nepamätajú priradenie medzi klientskou adresou a pseudonymnou adresou, tak nie je možné na tieto emaily odpovedať.

Julf Helsingius, tvorca stavového remaileru Penet, vytvoril základnú pracujúcu verziu v priebehu dvoch dní [36]. Komunikácia nebola šifrovaná a správanie služby bolo deterministické. V prípade pasívneho útočníka schopného zachytávať komunikáciu Penetu by bolo na základe postupnosti prichádzajúcich a odchádzajúcich správ triviálne priradiť pseudonymné adresy ku klientským adresám.

Útok. Penet bol vytvorený v roku 1993 a v časoch svojej najväčšej popularity mal 700,000 registrovaných klientov a preposielal denne viac ako 10,000 emailov [36].

Remailer nekontroloval svojich klientov ani obsah preposielaných správ. Niektorí klienti remailer zneužívali na distribúciu ilegálneho obsahu, na zverejňovanie privátneho a tajného obsahu alebo na vulgárnu kritiku verejných inštitúcií. Z toho dôvodu čelil Penet verejným aj právnym výzvam na odhalenie skutočných identít konkrétnych pseudonymov. Vyvrcholením bol článok na titulnej strane novín The Observer, ktorý obviňoval Helsingusa z distribúcie 90% detskej pornografie. Vyšetrovanie polície viedlo

[20] Podľa iného názvoslovia sa na rovnaké kategórie delia pod názvami *vystopovateľné* (stavové) a *nevystopovateľné* (bezstavové).

k vyvráteniu tohoto faktu, keďže Penet dlhý čas neumožňoval posielať multimediálny obsah v emailoch [36].

Kvôli stupňujúcemu sa tlaku sa Helsingus dňa 30.08.1996 rozhodol Penet vypnúť. Bolo možné, že štátu sa podarí získať mapovacie dáta a realizovať tak úspešný *právny útok*[21].

1.4.2 Anonymizer

Úvod. Špecifikácia HTTP protokolu núti webový prehliadač odkrývať mnohé dáta o klientovi. Napríklad premenné HTTP_USER_AGENT, REMOTE_HOST alebo HTTP_REFERER informujú o webovom prehliadači, operačnom systéme, procesore, IP adrese a predošlej navštívenej adrese klienta. Protokol HTTP cookie umožňuje webovému serveru ukladať informácie na klientskej strane a to umožňuje serveru stopovať akcie klienta na jeho stránkach.

Zariadenia, ktoré očisťujú HTTP protokol od dát umožňujúce identifikovať klienta, sa nazývajú *anonymizačné proxy*. Anonymizer je reálnou implementáciou anonymizačného proxy.

Implementácia. Ak klient chce využiť službu Anonymizera na čítanie URL adresy u, tak vedie svoju komunikáciu cez webové proxy vyžiadaním adresy `http://www.anonymizer.com:8080/u`. Webové proxy vo svojej prvej verzií [5] modifikovalo HTTP komunikáciu nasledovne:

- Namiesto IP adresy klienta používa svoju IP adresu.
- Odstráni informácie o konfigurácií klientského stroja z MIME hlavičky.
- Odstráni informácie o navštívených stránkach.
- Filtruje Java applety a Javascript aby nevytvárali paralelné HTTP spojenia.
- Filtruje cookies.
- Vracia pozitívnu spätnú väzbu pridaním reťazca "[Anonymizer]" do titulky.

[21] Viac o právnych útokoch v kapitole 1.7.

- Mení všetky odkazy tak, aby išli cez Anonymizer.

Odpoveď servera preposiela naspäť klientovi. Akonáhle je požiadavka klienta vykonaná, Anonymizer vymaže všetky dáta, ktoré využíval na vykonanie požiadavky. Týmto spôsobom sa vyhýba možnostiam priameho právneho útoku 1.7.

Anonymita. Nutným predpokladom je dôvera klienta v server.

Globálny pasívny útočník monitorujúci komunikáciu Anonymizera vie na základe postupností požiadaviek a odpovedí priradiť správy k odosielateľovi aj príjemcovi. Obranou sú systémy podobné mixovým sieťam popísané v kapitole 1.5.

Anonymizer sa snaží riešiť nedostatky anonymity HTTP protokolu. Už jeho tvorca Boyan tvrdil[5], že Anonymizer nevie zaručiť stopercentnú anonymitu, keďže potenciálny obsah HTML stránok[22] je stále vo vývoji. Vo všeobecnosti sa snaží odstrániť všetky stopy identity v prenášaných dátach a znemožniť komunikáciu mimo Anonymizer[23].

Ak klient používa štandardný protokol HTTP bez anonymizácie, tak je možné ho identifikovať na základe *odtlačkov prehliadača*. Monitorovaním čistého HTTP protokolu dosiahli [28] úspešnosť 99,1% identifikovania prehliadača v prípade opakovanej návštevy.

História. Anonymizer bol nasadený v roku 1995. Keďže všetky požiadavky klientov prechádzajú aj so všetkým dátami cez jeden server, Anonymizer časom podľahol neúnosnej záťaži, ktorá sa dala riešiť iba nakúpením nového hardwareu. Preto sa v roku 1997 stala komerčnou službou. Služby Anonymizera sú stále dostupné, v roku 2012 za 80 dolárov na rok [53]. Okrem základných služieb ponúka rovnomenná spoločnosť stavový remailer a anonymitu pre Wifi siete[24].

[22] HTML kód je jedným z najčastejšie prenášaných dát cez HTTP.

[23] Napríklad streamovanie multimédií sa realizuje vytvorením nového komunikačného spojenia.

[24] V našej práci sa anonymitou vo wifi sieťach nezaoberáme. Viac je možné nájsť v práci Kellyho [39].

1.5 Mixové siete

Úvod. Mixové siete poskytujú anonymitu odosielateľovi maskovaním jeho správ medzi ostatnými správami siete. Ako už názov naznačuje, zariadenie *mix* poradie preposielaných správ mieša a tým robí priradenie vstupných a výstupných správ ťažším.

Mixové siete sa kvôli ich vysokým odozvám používajú v anonymných remaileroch a v protokoloch elektronických volieb [66]. Snažia sa chrániť proti globálnemu pasívnemu útočníkovi. Môžu mať problémy s aktívnym útokom popísaným v kapitole 1.5.4.

Koncept a terminológia. Mix je zariadenie pracujúce v troch fázach:

1. *Zhromaždenie* správ. Mix čaká, kým dostane n správ tvoriacich jednu *dávku*[25].
2. *Premiešanie* dávky. Mix poradie správ v dávke spermutuje.
3. *Vypustenie* premiešanej dávky. V spermutovanom poradí mix pošle správy ďalej.

Anonymita odosielateľa sa dosahuje za predpokladu, že útočník nevie priradiť správy zo vstupu mixu správam vo výstupe mixu.

1.5.1 Chaumov mix

Úvod. Prvý protokol využívajúci myšlienku premiešavania správ bol navrhnutý Chaumom v roku 1981 [7].

Chaum uvažuje dva základné predpoklady pre svoj protokol [7]:

1. *Šifrovať a dešifrovať správy vie len účastník poznajúci kľúč.* Formálne, nikto nevie priradiť šifrované správy ku korešpondujúcej množine nešifrovaných správ, alebo vytvárať falzifikáty, bez znalosti príslušného náhodného reťazca alebo súkromného kľúča.
2. *Vonkajší globálny aktívny útočník.* Ktokoľvek vie čítať všetky správy na komunikačnej vrstve[26], každý môže také správy vytvárať, mazať a meniť.

[25] V kapitole o mixoch budeme číslom n označovať veľkosť jednej dávky mixu v aktuálnom kontexte.

[26] Chaum pod tým myslel napr. pakety. Predpokladal, že každý vidí ich hlavičku a teda aj zdroj a cieľ.

Protokol. Všetky správy M od odosielateľa B pre adresu prijímateľa A sa posielajú cez jeden daný mix. Nech R_1 a R_0 sú náhodnými reťazcami (*nonce*), K_1 a K_A sú verejné kľúče mixu a A. Nech $E_k(X)$ označuje doprednú asymetrickú šifrovú transformáciu. Potom mix transformuje každú správu nasledovne [7]:

$$E_{K_1}(R_1, E_{K_A}(R_0, M), A) \mapsto E_{K_A}(R_0, M), A.$$

Ľavú stranu budeme považovať za *prijatú správu* a pravú stranu za *odoslanú* správu.

Hlavnou úlohou mixu je utajiť väzbu medzi prijatou a odoslanou správou. To sa snaží dosiahnuť tým, že správu neposiela ďalej ihneď. Namiesto toho mix najprv nazhromaždí n správ, preusporiada ich v lexikografickom poradí a až potom pošle celú dávku na určené adresy. Množina anonymity odosielateľa sú intuitívne práve klienti, ktorých správy boli vo vypustenej dávke.

Nedostatky. V tejto verzií je protokol centralizovaným systémom. Preto môže dojsť k jeho zahlteniu, výpadku alebo úspešnému útoku na samotný mix. Zahltenie a výpadok sa rieši behom viacerých mixov súčasne.

Jedným z hlavných útokov proti mixovým sieťam je $n-1$ útok popísaný v kapitole 1.5.4. Myšlienka útoku je taká, že sa útočníkovi podarí do jednej dávky mixu dostať jeho $n-1$ správ a jednu správu jeho záujmu m. Po vypustení dávky útočník vie triviálnym spôsobom odlíšiť správu m od svojich správ, a tým zistiť jej odosielateľa.

Vo všeobecnosti sa útoky na centralizované mixy riešia napríklad použitím viacerým mixov na smerovanie jednej správy, takzvanými *kaskádami*.

Kaskády. Chaum definuje *kaskádu* ako statickú postupnosť N mixov v rade, pričom k-ty mix vykonáva transformáciu rovnakú, ako pri centralizovanom prístupe:

$$\begin{aligned} E_{K_{N-k+1}}(R_{N-k+1}, E_{K_{N-k}}(R_{N-k}, \ldots, E_{K_1}(R_1, E_{K_A}(R_O, M), A)\ldots)) \mapsto \\ \mapsto E_{K_{N-k}}(R_{N-k}, \ldots, E_{K_1}(R_1, E_{K_A}(R_O, M), A)\ldots). \end{aligned} \tag{2}$$

Pôvodná správa od odosielateľa je zabalená do zašifrovaných *vrstiev*, pričom každý mix jednu z vrstiev *rozbalí*. V takejto sieti stačí, aby bol jeden z mixov čestný. Ďalšou výhodou je, kvôli statickej sieti, že v jednotlivých vrstvách nie je potrebné poskytnúť informáciu o nasledujúcom mixe.

Novodobé mixové siete, ako napríklad Mixminion[27], umožňujú odosielateľovi vytvoriť si pre svoje správy vlastnú cestu cez mixovú sieť. Koncept sa v literatúre nazýva *voľné smerovanie*. Nevýhodou môže byť zložitosť, keďže mix nevie dopredu kam môže správa smerovať, tak je nútený použiť asymetrické šifrovanie[28]. Navyše, ak vie útočník určiť, že správa je spätná, tak je protokol náchylný na útok opakovaním, ktorý môže viesť k odhaleniu pôvodného odosielateľa.

Ďalšie možné topológie siete skúmajú Dingledine, Shmatikov a Syverson [24].

Spätné správy. Protokol umožňuje posielať anonymné správy od účastníka X účastníkovi Y. Chaum uvažoval aj o schéme, ktorá umožňuje Y poslať *spätnú správu* X bez odhalenia identity X.

Odosielateľ X priloží anonymnú spätnú adresu $K_1(R_1, A_X)$ do svojej správy, kde K_1 je kľúč mixu, R_1 je nonce generovaný X a A_X je adresa X. Mix má špeciálnu transformačnú funkciu pre spätné správy:

$$E_{K_1}(R_1, A_X), E_{K_X}(R_0, M) \mapsto A_X, R_1(E_{K_X}(R_0, M)),$$

kde K_X je kľúč vygenerovaný pre túto príležitosť, R_0 je nonce a M je spätná správa. Pravá časť je šifrovaná R_1, aby útočník nemohol opakovať správu od Y. Reťazec R_1 poznajú iba X a mix. Táto konštrukcia sa dá rozšíriť na kaskády. Spätné správy navrhované Chaumom sú podobné spätným onionom popísaným v kapitole 2.2.3.

1.5.2 Remailery druhej generácie

Remailery druhej generácie sú mixovými sieťami implementujúce koncepty Chauma [7]. Medzi najznámejšie patria Babel [34] a Mixmaster [48], oba vytvorené v polovici 90tych rokov.

Mixmaster. Protokol Mixmaster bol navrhnutý v roku 1995 Ulfom Möllerom [48] a vyvíjal sa až do vydania verzie 3 v roku 2008. Mixmaster je jeden z najrozšírenejších remailerov súčasnosti [12] a používa na prenos protokol SMTP. Postupne vymenujeme hlavné vlastnosti Mixmastera podľa jeho špecifikácie [48].

[27] Viac o Mixminione v kapitole 1.5.3.

[28] Lebo si nevie dopredu dohodnúť kľúč pre symetrické šifrovanie.

Akumulácia správ[29]. Mixmaster správy zhromaždí a uloží si ich do *akumulátora*. Keď uplynie istý čas, alebo je v akumulátore dostatok správ, tak z nich vyberie časť, pomieša a pošle ďalej. Tento koncept je čiastočnou odpoveďou na $n-1$ útok, popísaný v kapitole 1.5.4.

Voľné smerovanie. Mixmaster necháva voľbu cesty na odosielateľovi. Zoznam dostupných mixov je verejnou informáciou.

Rozsekanie správ. Mixmaster umožňuje dlhé správy rozsekať na *kusy*. Tieto kusy môže užívateľ poslať rôznymi cestami, ak platí, že posledný mix majú všetky zvolené cesty spoločný. Posledný mix kusy správ zhromažďuje, kým neuplynie globálne daný časový limit. Ak má všetky, tak ich spojí dokopy a pošle príjemcovi. Okrem toho, Mixmaster umožňuje poslať viacero kópií rovnakého kusu správy. Posledný mix vyberie prvý z nich. To zvyšuje spoľahlivosť protokolu. Ide o dôležitú súčasť protokolu, keďže odosielateľ nevie zistiť, či príjemca správu dostal.

Ochrana pred útokom opakovaním. Mixmaster používa časové pečiatky, kontrolu unikátnosti identifikátora správy a rotáciu kľúčov. Ak vypršala časová pečiatka prijatej správy, tak ju mix zahodí. Mix si pamätá identifikátory všetkých správ, ktoré majú platnú časovú pečiatku. Ak príjme správu s rovnakým identifikátorom, tak ju zahodí. Identifikátory s neplatnou časovou pečiatkou mix maže, lebo by mohli byť v budúcnosti zneužité útočníkom[30]. Každý mix svoje verejné kľúče mení v čase, ak dostane správu šifrovanú neplatným kľúčom, tak ju nevie dešifrovať a preto ju zahodí.

Falošné správy. Každý mix vytvára falošné správy. Vypúšťajú sa do poolu podľa exponenciálneho rozdelenia. Na každú prijatú správu pripadá v priemere 0.1 falošnej správy.

Hybridné šifrovanie. Hlavička správy je šifrovaná asymetrickou schémou, a obsahuje kľúč pre symetrické šifrovanie. Tento kľúč je potom použitý na dešifrovanie tela správy.

Integrita správ. V každej vrstve správy sa nachádza hash správy, ktorý sa overuje pri každom prechode cez mix. Kvôli tomu je nemožné vytvárať *spätné správy*, lebo odosielateľ nevie dopredu vypočítať hash spätnej správy[31].

[29] Viac o tejto stratégií mixovania v kapitole 1.5.5.

[30] Napríklad v prípade, keď sa útočníkovi podarí získať kontrolu nad mixom.

[31] Keby sa nezachovávala integrita spätnej správy, tak by bolo zbytočné zachovať integritu doprednej správ. Aby sa zachovala integrita spätnej správy, musel by byť hash pripojený v každej vrstve spätnej

1.5.3 Mixminion

Úvod. Mixminion [14] je rozšírením pôvodného Chaumovho návrhu mixovej siete a jej používanej implementácie Mixmaster[32]. Tvorcovia návrhu Mixminionu poskytujú každému možnosť behu svojho vlastného mixu. Okrem toho existuje aplikačný klient určený na posielanie anonymných emailov cez sieť Mixminionu.

Mixminion, rovnako ako Mixmaster, používa na miešanie správ pool stratégiu, popísanú v kapitole 1.5.5. Hlavným rozdielom oproti Mixmasteru je možnosť posielania spätných správ.

SURB - Single Use Reply Block (Jednorazový spätný blok). Keďže odosielateľ správy nepozná kľúče v hlavičke spätnej správy, tak obsah správy posiela pre prvý mix nešifrovaný. Obsah správy sa pri prechode cez mixy postupne zaobaľuje do šifrovaných vrstiev. Príjemca spätnej správy si musí pamätať kľúče symetrického šifrovania kódované v SURB, aby mohol prijatú správu dešifrovať.

Filozofiou Mixminionu je, aby dopredné a spätné správy boli nerozlíšiteľné[33] pre každého účastníka. To znemožňuje použitie hashov na kontrolu integrity *obsahu* správ.

Mixminion preto zavádza hlavičky rozdelené na dve *nohy*. Hlavička je rozdelená na *hlavnú* časť a *vedľajšiu* časť, ktoré sú rovnako veľké a obe podliehajú kontrole integrity. Hlavná hlavička je anonymnou adresou pre cestu z A do M a vedľajšia hlavička je anonymnou cestou z M do B, A je odosielateľ, B je príjemca a M je mix, nazývaný *bod výmeny*. Vedľajšia hlavička je šifrovaná symetricky odosielateľom, kľúč je odvodený od hashu obsahu správy.

Ak príde správa do bodu výmeny, tak sa hlavná hlavička dešifruje súkromným kľúčom, a mix zistí, že je bodom výmeny. Preto dešifruje vedľajšiu hlavičku podľa hashu obsahu a potom podľa svojho súkromného kľúča a tieto hlavičky vymení. Ďalej posiela správu podľa novej hlavnej hlavičky.

Ak by sa obsah správy zmenil na ceste $A \rightarrow M$, tak by sa M nepodarilo dešifrovať

anonymnej adresy. A to je nemožné, lebo odosielateľ dopredu nevie hash spätnej správy príjemcu.

[32] O pôvodnom návrhu Chauma píšeme v kapitole 1.5.1 a o Mixmasterovi v kapitole 1.5.2.

[33] A teda obsahuje identifikátor správy, adresy mixov, kľúče pre symetrické šifrovanie. Tieto informácie sú takisto zabalené šifrovaním verejnými kľúčmi mixov. Preto sa odpoveď musí poslať skôr, ako príde k rotácií kľúčov na jednotlivých mixoch.

vedľajšiu hlavičku. Na cesta $M \to B$ môže prísť k zmene správy. Útoku sa tak bráni Mixminion iba čiastočne.

Implementácia. V tejto kapitole spomenieme niektoré odlišnosti od Mixmastera, popisovaného v kapitole 1.5.2.

Mixminion, narozdiel od Mixmastera, používa TLS[34] nad TCP na zabezpečenie linkovej vrstvy. Okrem vyššej bezpečnosti je tým umožnené posielať aj iné typy dát ako emaily.

Proti útoku opakovaním sa Mixminion chráni *jedine* rotáciou kľúčov a zapamätaním si všetkých identifikátorov správ pre aktuálne platný kľúč. Tým znemožňuje ľubovoľnému užívateľovi poslať rovnakú správu cez rovnaký mix viackrát.

Mixminion používa verejné priečinkové servre[35], ktoré navyše poskytujú výkonnostné charakteristiky jednotlivých mixov.

Každý mix má vlastné výstupné pravidlá, dostupné z priečinkových servrov, ktorými môže obmedziť dáta, ktoré vypúšťa mimo sieť mixov. Tento návrh je podobný priečinkovým servrom Toru popísaných v kapitole 2.3.2.

Mixminion nevytvára falošné správy, keďže podľa ich autorov neexistuje presvedčivá analýza, ktorá by preukázala ich výhody [14].

Všetky správy sú rovnako dlhé, aby boli nerozlíšiteľné. Správa sa pri prechode cez mix skracuje, lebo sa odstráni jedna vrstva hlavičky. Preto je nutné hlavičku doplniť vypchávkou (paddingom). V článku [50] navrhujú bezpečný padding pre mixové siete.

Presnejšie údaje o implementácií návrhu je možné nájsť v špecifikácií Mixminionu [15], od rovnakých autorov.

1.5.4 $n-1$ útok

Úvod. V sekcií prezentujeme $n-1$ útok[36]. Článok [68] rozoberá úspešnosť útoku proti štandardným druhom mixov, článok [56] sa zaoberá aj s mixami s pamäťou a článok [40] uvádza algoritmus so simuláciou.

V sekcií popíšeme útok a v krátkosti diskutujeme o možných rozšíreniach a obranách

[34] A používa jedine Diffie-Hellmanov protokol na dohodu kľúča.

[35] Viac o priečinkových servrov v kapitole 2.3.2.

[36] Ktorý je známy aj pod menami *disclosure*, *flood-trickle* alebo *blending*.

Model útočníka. Predpokladáme, že v sieti existuje jeden mix, ktorý je čestný, vie rozpoznať opakované správy a vykonáva štandardný trojfázový beh: zhromaždí správy, preusporiada správy a vypustí správy.

Predpokladáme globálneho aktívneho útočníka, ktorý vie pozorovať všetky správy, zachytiť ich, pozdržať ich, meniť ich alebo vytvárať nové ale nevie získať priamu kontrolu nad samotným mixom. Pripomeňme, že útočník kvôli šifrovým transformáciám vo vnútri mixu nevie priradiť ku vstupnej správe jej výstupný obraz. Cieľom útočníka je narušiť vzájomnú anonymitu odosielateľa a príjemcu, respektíve odhaliť s kým daný klient komunikuje.

$n-1$ útok. Predpokladajme, že mix vždy zhromaždí aj vypustí presne n správ. Uvažujme algoritmus na vykonanie $n-1$ útoku z článku [68]: Útočník zachytí a pozdrží správu svojho záujmu m. Následne posiela správy do mixu, kým mix dávku nevypustí. Akonáhle mix dávku vypustil, útočník pozdrží cudzie správy a pošle mixu $n-1$ svojich správ spolu s m. Potom, čo mix správy vypustil vie útočník jednoznačne určiť, ktorá z vypustených správ bola m.

Poznamenajme, že za našich predpokladov je $n-1$ útok *istým* útokom, ktorý je vždy úspešný.

Zovšeobecnený útok. Kvôli mixom s pamäťou nemusí stačiť jedno kolo na úspešný útok, keďže správa m môže v mixe po vypustení zostať (a podobne mix nie je úplne prázdny). V [56] formálne definujú viackolový pravdepodobnostný útok, ktorého podstatou je vo viacerých kolách vyprázdniť mix od cudzích správ a potom posielať vlastné správy, až kým m nie je z mixu vypustená. V článku odvodia vzťah počtu potrebných kôl v závislosti od požadovanej pravdepodobnosti a bližšie skúmajú úspešnosť $n-1$ útoku na rôzne typy mixov.

Napriek intuitívnej komplikácií v prípade pravdepodobnostných mixov s pamäťou nedochádza k reálnemu zvýšeniu nákladov útoku [68].

V článku [56] sa snažia presne vyčísliť počet kôl útoku potrebných na dosiahnutie dostatočnej pravdepodobnosti v prípade akumulačných mixov.

Možné obrany. Sieť viacerých mixov s možnosťou voľného smerovania správ je náročnejším prostredím na vykonanie $n - 1$ útokov ako siete s jedným mixom alebo kaskádovým zapojením viacerých mixov.

Pridaním aspoň jednej falošnej správy do každej vypustenej dávky znemožní útočníkovi jednoznačne identifikovať m. Uvažujú sa falošné správy vytvorené mixami určené pre mixy, ktoré vykonajú v sieti niekoľko skokov. Útočník je preto nútený sledovať všetky cudzie správy vypustené z mixu počas jeho útoku v sieti ďalej rovnako, ako správu m. Keďže jedine správa m sieť mixov opustí, je útočník schopný v konečnom čase správu identifikovať. Počet sledovaných správ sa ale exponenciálne zvyšuje od počtu skokov m, čo môže prakticky znemožniť úspešný útok.

Inú možnosť obrany poskytujú *RGB mixy* [16]. Každý mix M má množinu *kontrolných* mixov B_M. Medzi M a B_M sa pravidelne posielajú podpísané *heartbeat*[37] správy. RGB je skratka pre *red* (červená), *green* (zelená) a *black* (čierna)[38]. Farby označujú rôzne typy správ. Červené sú práve heartbeat správy. V prípade, že M prestane dostávať červené správy, tak predpokladá, že je zahltení čiernymi správami a preto do svojho výstupu pridáva falošné zelené správy.

Ďalšími možnosťami obrany sú napríklad:

- Kontrola doručenia správ v nasledujúcej dávke predchádzajúcim odosielateľom.
- Vytváranie podobvodov, cez ktoré sa správa pred vypustením preposiela medzi mixami. Túto schému implementuje systém Babel [34] a je podobný princípu uvoľneného smerovania popisovaného v kapitole 2.2.3.
- Zavedenie reputačných schém na základe záväzkov [68].

1.5.5 Stratégie mixovania

Uvádzame zoznam rôznych spôsobov, akými môže mix správy preusporiadať.

Prahový mix (Treshold mix). Štandardný mix sa nazýva aj *prahovým mix*om. Jediným jeho parametrom je prah n. Zhromaždí n správ, preusporiada ich podľa da-

[37] Heartbeat - *srdcový tep* - je pravidelným pingom štandardne určeným na monitorovanie stavu zariadení.

[38] RGB štandardne v informatike znamená red, green a *blue*.

ného porovnania a vypustí ich všetky v jednej dávke. Nie je odolný proti $n-1$ útoku popísaného v kapitole 1.5.4.

Časovaný mix (Timed mix). Jediným jeho parameterom je perióda t. Nezávisle na počte nazhromaždených správ vypustí každých t sekúnd všetky nazhromaždené správy. Výhodou je, že každá správa sa z neho dostane v konečnom čase. Nevýhodou je, že útočníkovi stačí v prípade $n-1$ útoku správu svojho záujmu m pozdržať a počkať, kým mix svoje správy vypustí. Potom vloží do mixu m a pozdrží všetky ostatné správy. Po t sekundách čakania vie jednoznačne určiť vypustenú správu m. Predpoklady na úspešný útok sú teda slabšie, ako v prípade štandardného $n-1$ útoku.

Akumulačný mix (Pool mix). Definujeme *akumulačný mix* podľa [68]. Akumulačný mix má dva parametre: prah n a veľkosť akumulátora f. Mix vypúšťa správy, keď sa v ňom naakumuluje $n+f$ správ. Vtedy z nich rovnomerne náhodne vyberie f, ktoré ponechá v akumulátore. Ostatných n preusporiada ako prahový mix a vypustí ich v jednej dávke.

V priemere sa správa zdrží v mixe $1+\frac{f}{n+f}$ kôl [68]. Efektívnu veľkosť množiny anonymity[39] je možné odhadnúť za predpokladov, že každú zo správ poslal iný účastník [67]:

$$lim_{k\to\infty}E = (1+\frac{f}{n})\lg(n+f) - \frac{f}{n}\lg n,$$

kde k je počet kôl behu. Napríklad, ak $f = 10$ a $n = 100$, tak $lim_{k\to\infty}E \approx 7.129$, čo intuitívne znamená, že každá zo 100 nazhromaždených správ je potenciálne medzi $2^{2.719} \approx 140$ vypustenými správami.

Akumulačné mixy je možné kombinovať s časovanými [68].

Napriek nedeterministickému správaniu vedie opakovanie kôl $n-1$ útoku k ľubovoľne veľkej pravdepodobnosti odhalenia.

Akumulačný mix s dynamickou pamäťou (Dynamic pool mix). Namiesto statického parametru f štandardných akumulačných mixov, používajú *dynamické mixy* parameter f ako náhodnú premennú. Štandardne sa volí f podľa exponenciálneho alebo

[39] Definujeme v kapitole 1.2.5.

binomického rozdelenia. Aj keď sa úspešnosť $n-1$ útoku ďalej znižuje, stále zostáva opakovaním kôl úspešný [68].

V článku [27] vypočítavajú veľkosti množín anonymity pre akumulačné mixy. Rozoberajú prípady statickej aj dynamickej pamäte a nepoužitie a použitie falošných správ. Najviac (v zmysle entropie) anonymity z nich zabezpečujú dynamické akumulačné mixy, ktoré pridávajú náhodný počet falošných správ do vypustenej dávky.

Stop-and-Go (SG) mix. SG mix rozširuje myšlienku časovaných mixov a má tri parametre: T_{min}, T_{max} a μ. Pre každú správu individuálne vypočíta náhodné časové *omeškanie* v rozpätí (T_{min}, T_{max}) podľa exponenciálnej distribúcie pravdepodobnosti s parametrom μ.

V článku [10] ukázali, že zo všetkých spojitých funkcií práve exponenciálna dosahuje najväčšiu efektívnu veľkosť množiny anonymity.

V článku [41] analyzovali SG protokol, ktorý do správ navyše pridáva časové pečiatky. Exponenciálna distribúcia omeškania dáva klientom dobré časové odhady o trvaní doručenia. Navyše vypočítali strednú hodnotu množiny anonymity:

$$E(S) = \frac{\lambda}{\mu} + \frac{e^{\lambda/\mu} + 1}{2},$$

kde λ je parameter exponenciálnej distribúcie prijatých správ a μ je parameter exponenciálnej distribúcie omeškania SG mixu.

Dummy messages (Falošné správy). Pridávanie nerozpoznateľných *falošných správ* do výstupných dávok je výhodnou stratégiou. Ako sme naznačili v kapitole 1.5.4, tak úspešnosť $n-1$ útoku klesá pri pridávaní konštantného počtu falošných správ exponenciálne od počtu skokov správy.

Presnejšie odhady poskytuje článok [27]. Detaily neuvádzame, keďže prekračujú rozsah našej práce.

Krehké mixovanie (Fragile mixing). Zaujímavým teoretickým modelom sú *krehké* mixy [63], ktoré rozširujú štandardné prahové mixy. Ich účelom je motivovať administrátorov mixov, aby dodržovali protokol a žiadnym spôsobom neuprednostňovali správy (napríklad vlastné). Myšlienkou je skryť vykonanú permutáciu Φ_D na dávke D tak, aby

odhalenie jednej inverzie viedlo k odhaleniu celej permutácie Φ_D a tým odhaleniu všetkých vypustených správ v D.

Krehké mixy sa zakladajú na beznalostných dôkazoch (zero-knowledge proofs) [3]. Článok [63] navrhuje, dokazuje korektnosť a implementuje protokol krehkého mixovania.

1.6 Ďalšie anonymné siete

V kapitole v krátkosti uvádzame ďalšie rozšírené anonymné siete, ktoré kategorizujeme podľa ich využitia na *anonymné posielanie správ* a *anonymné zdieľanie súborov*.

Používame pojmy z iných protokolov popísaných v tejto práci.

1.6.1 Anonymné posielanie správ

Crowds. Crowds [62] je pomerne unikátnym protokolom, ktorý sa snaží chrániť iba anonymitu odosielateľa. Všetci klienti sú aktívnymi užívateľmi siete, ktorý sú schopný preposielať správy. Klienti sa delia do skupín - *crowd*ov, ktoré sú geograficky rozptýlené. Pre klienta A je *jondo*m každý iný klient v crowde klienta A.

Ak klient J_0 chce poslať správu m pre B, tak si hodí váženou mincou. V jednom prípade (hlava) pošle správu priamo B a v druhom prípade (znak) si vyberie náhodné jondo J_1. V druhom prípade vykoná J_1 rovnaký postup ako J_0. Toto sa opakuje, kým na váženej minci nepadne hlava.

Anonymita odosielateľa je zabezpečená tým, že útočník nevie, či je posledný z postupnosti jondov tvorcom, alebo preposielateľom správy.

Freedom. Freedom [1] je protokol a aplikačný klient vytvorený spoločnosťou Zero Knowledge v roku 2001. Umožňuje posielať ľubovoľné správy a svojich klientov adresuje podľa pseudonymov uložených v centrálnej autorite. Každý pseudonym obsahuje anonymnú adresu, konštrukciou podobnú reply onionu popísaného v kapitole 2.2.3 a údaje, ktoré umožňujú autentifikovať pseudonymného klienta. Vlastnosti ako šifrovanie po vrstvách, zabezpečenie linky, centrálne repozitáre, miesta stretnutia a podpora mnohých transportných protokolov pripomínajú Tor. Konceptuálnym rozdielom je, že Freedom poskytuje iba pseudonymitu. Jeden klient môže vlastniť viacero pseudonymov.

Tarzan. Tarzan [31] je peer-to-peer protokolom bez centrálnej autority pripomínajúcim Onion routing a bol navrhnutý v roku 2002. Rozdielom je, že Tarzan vyberá routre pre svoj obvod iba zo svojho okolia. Pomocou falošných správ sa snaží chrániť aj proti globálnemu pasívnemu útočníkovi.

Drac. Drac [13] je protokol určený na zabezpečenie anonymity v prostredí IM a VoIP využívajúci sociálne siete navrhnutý v roku 2010. Keďže IM aj VoIP sú protokoly s nestálym prenosom dát, tak Drac používa veľké množstvo falošných správ, aby sa chránil pred globálnym pasívnym útočníkom.

Telex. Telex [77] je protokol navrhnutý v roku 2011 určený na obchádzanie cenzúrových systémov[40]. Jeho myšlienkou je pridávať *značky* reprezentujúce požadovanú blokovanú URL do bežnej webovej komunikácie, ktoré sú cenzúrovými systémami nerozpoznateľné. Využitím ISP[41] na vyhľadávanie týchto značiek by presmerovali komunikáciu na požadovanú URL. Telex chce ISP motivovať k poskytnutiu svojich zdrojov pomocou štátov, ktoré chcú zabezpečiť slobodný obsah cenzúrovaným krajinám.

I2P. I2P [78] je protokolom a aplikáciou podobnou Toru navrhnutým v roku 2003. Na rozdiel od Toru nepredpokladá existenciu centrálnych autorít a na adresovanie používa DHT - distribuované hashové tabuľky. Ďalším rozdielom je, že používa navyše schému *Garlic routingu* (cesnakového smerovania). Každý router okrem toho, že zaobaľuje a rozbaľuje oniony, čaká na oniony s rovnakou nasledujúcou adresou. Keď ich je viac, tak ich zabalí do jedného garlicu, v ktorom sú tieto správy nerozlíšiteľné.

1.6.2 Anonymné zdieľanie súborov

Pod pojmom *anonymné zdieľanie súborov* myslíme systémy, ktoré anonymizujú vytváranie, zmenenie, mazanie, a prístup k súborom v distribuovanom súborovom systéme.

Freenet. Freenet [9] je peer-to-peer aplikácia bez centrálnej autority implementovaná v roku 2000. Každý klient poskytuje časť svojej pamäte, v ktorej sú uložené klientom

[40] Viac o cenzúrových systémoch v kapitole 2.3.3.

[41] ISP - Internet service providers sú *poskytovateľmi internetového pripojenia* pre právnické a fyzické osoby, zvyčajne pokrývajúce pripojením regióny a štáty.

nerozpoznateľné dáta. Každý súbor je v sieti uložený viacnásobne.

Pri hľadaní súboru v sieti sa súbor premiestňuje bližšie ku klientom, ktorý ho často vyžadujú a naopak sa vymazáva z miest, z ktorých je žiadaný málokedy.

V rámci dátového priestoru Freenetu sa vytvoril anonymný statický Internet, ktorý obsahuje aj tajné informácie a dokumenty.

Podobným systémom je Publius[75].

Iné. Ďalšími protokolmi určených na zdieľanie súborov sú FreeHaven [21], ktorý využíva reputačný systém, Mute[8], ktorý na vyhľadávanie používa mravcový algoritmus [26] a OneSwarm [60], ktorý sa snaží zabezpečiť anonymitu podobnú Onion routingu a výkon podobný protokolu BitTorrent.

1.7 Štandardné útoky

V kapitole uvádzame zoznam štandardných útokov na anonymné siete. Nerozoberáme ich do hĺbky, keďže ich realizácia často závisí na konkrétnom protokole. Špeciálne v prípade Toru v kapitole 2.4 popisujeme potenciálne útoky a v kapitole 1.5.4 popisujeme $n-1$ útok určený hlavne pre mixové siete.

Útoky rozdeľujeme podľa ich účelu na dva typy:

- *potvrdenie prenosu* (traffic confirmation), ktorého účelom je odhaliť vzájomnú anonymitu dvoch daných klientov.
- *analýza prenosu* (traffic analysis), ktorého účelom je odhaliť anonymitu odosielateľa, alebo príjemcu. Inými slovami je cieľom k danému klientovi nájsť klienta ktorý s ním komunikuje.

Vo väčšine prípadov nás zaujíma hlavne analýza prenosu. V prípade sietí s nízkou latenciou spojenia, ako napríklad Tor, je zbytočné uvažovať o obranách proti potvrdeniu prenosu, keďže časové korelácie na vstupoch a výstupoch siete odhaľujú vzájomnú anonymitu [23].

Využitie prienikov (Intersection attack). Popisuje základnú techniku, keď útočník vytvára prieniky množín anonymity jeho objektu záujmu. Útočník môže získať

viacero množín anonymity pre daný objekt záujmu vykonaním viacerých útokov. Ide o prostriedok analýzy prenosu.

Počítanie paketov (Packet counting). Ak sa v sieti zariadení neposielajú falošné správy, tak platí, že počet prijatých a odoslaných správ daného zariadenia je rovnaký. Špeciálne, ak je vytvorená cesta $C = Z_1, Z_2, \ldots, Z_n$ cez sieť zariadení, po ktorej si koncový klienti posielajú viacero správ, tak platí, že počet správ poslaných po linke $Z_{i-1} \rightarrow Z_i$ je rovnaký, ako počet správ poslaných po linke $Z_i \rightarrow Z_{i+1}$.

Napríklad v prípade mixovej siete, ktorá posiela viacero správ po rovnakom obvode, je možné počítaním správ na vstupných a výstupných linkách daného mixu získať postupnosti vstupných počtov $(a_1, a_2, \ldots, a_k)$ a výstupných počtov $(b_1, b_2, \ldots, b_m)$ správ. Pre každé b_i platí, že je súčtom niektorých a_j, pričom každé a_j prispieva k práve jednému b_i. To je dostatočnou informáciou na odhaleniu časti obvodu. V prípade, keď je v sieti iba jeden mix to vedie rovno k odhaleniu anonymity.

Útok sa môže využiť pri analýze aj potvrdení prenosu.

Označovací útok (Tagging attack). Myšlienkou útoku je *označiť* danú správu m tak, aby ju bolo možné v budúcnosti odhaliť. Používa sa hlavne v protokoloch, v ktorých sa integrita správ neoveruje a teda je možné správy meniť.

Koncept SURB v protokole Mixminion[14] sa snaží brániť proti označovaciemu útoku rozdelením hlavičky. Viac v kapitole 1.5.3.

O možnostiach označovacích útokov voči Toru píšme v kapitole 2.4.4.

Útok sa využíva pri potvrdení prenosu.

Útoky využívajúce latenciu spojení. V reálnych sieťach je čas doručenia správ medzi rôznymi účastníkmi často rôzny. Pasívny útočník vie tieto *latencie* (omeškania) namerať medzi zariadeniami. Ak sa v protokole budujú pomerne krátke spojenia, tak sledovaním vstupných a výstupných časov správ je možné odhadnúť, cez ktoré zariadenia správa prechádzala a vybrať najpravdepodobnejšiu cestu.

Ak je útočník aktívny, tak vie latencie umelo zvýšiť *zahltením* vybraných liniek alebo zariadení posielaním veľkého množstva správ. Viac o *útokoch zahltením* píšeme v kapitole 2.4.3.

Právne útoky a útoky na reputáciu siete. Od čias úspešného *útoku na reputáciu* siete Penet v roku 1996, o ktorom píšeme v kapitole 1.4.1, uvažujú anonymné protokoly aj o obranách proti útočníkom, ktorí chcú znížiť reputáciu siete.

Reputáciu anonymnej siete je možné znižovať verejným šírením dezinformácií, využívaním siete na distribúciu nelegálneho materiálu alebo využitím siete na vykonanie útokov proti iným sieťam.

Zníženie reputácie siete vedie k zníženiu počtu užívateľov a potenciálne aj štátnemu zásahu - *právnemu útoku*. Proti právnym útokom sa anonymné siete chránia rozložením zariadení do viacerých jurisdikcií, čo znižuje pravdepodobnosť globálneho zákazu. Aby nebolo možné súdnym spôsobom prinútiť zariadenia odhaliť routovacie dáta, tak si pamätajú len minimum informácií.

2 Tor

2.1 Úvod

Tor je populárna sieť zabezpečujúca anonymitu proti lokálnemu aktívnemu účastníkovi. Keďže zabezpečuje anonymné obojsmerné spojenia nad TCP a pritom zachováva relatívne nízku odozvu a umožňuje rýchly prenos dát, tak je ideálny pre interaktívne aplikácie ako prehliadanie webových stránok, zdieľanie súborov alebo odosielanie okamžitých správ [46].

Názov protokolu *Tor* - The Onion routing je odvodený od protokolu Onion routing, ktorý Tor rozširuje a implementuje. Tor, na rozdiel od Onion routingu, zabezpečuje perfektnú anonymitu odosielateľa.

Pod názvom Tor sa skrýva aj aplikačný klient komunikujúci so *sieťou Toru*. Sieť Toru je podsieťou Internetu a pozostáva z niekoľko tisíc *Tor relayov* preposielajúcich správy, *mostov* umožňujúcich pripojenie cenzúrovaných klientov a *skrytých služieb* poskytujúcich prístup k anonymných servrom.

Kapitola je rozdelená na tri hlavné sekcie. V prvej sekcií popisujeme pôvodný protokol Onion routing. V druhej sekcií podrobne popisujeme Tor a jeho rozšírenia. V tretej sekcií uvádzame zoznam možných útokov voči Toru.

2.1.1 Základné vlastnosti protokolu

Za základné vlastnosti a ciele Toru jeho autori považujú [23]:

- **Nasaditeľnosť**. Tor sa spolieha na veľkú množinu dobrovoľníkov, ochotných obetovať svoje zdroje. Preto inštalácia klienta, routru ani mostu nemôže vyžadovať administrátorské práva a nemala by byť komplikovaná. Tor by nemal mať požiadavky voči iným aplikáciám a preto sa zaoberá výhradne prenosom na úrovni TCP.

- **Použiteľnosť**. Spokojnosť klientov so službami Toru zvyšuje ich počet. Čím viac užívateľov, tým je množina anonymity väčšia. Preto je použiteľnosť atribútom bezpečnosti. Použiteľnosť je možné zvýšiť rýchlosťou pripojenia, minimalizáciou konfigurácie a podporou pre viaceré platformy. Tor aj preto zverejňuje zdrojové kódy aplikačného klienta a zbiera spätnú väzbu priamo od klientov [23].

- **Flexibilita**. Známe nedostatky Toru je nutné opraviť čo najskôr, keďže nedôveryhodnosť vedie k zníženiu užívateľskej základne. Preto Tor reprezentuje aktívna komunita https://www.torproject.org/.

- **Nezabezpečovať proti potvrdeniu prenosu**. Hlavnou vlastnosťou Toru je nízka latencia prenosu. Útočník monitorujúci dvoch koncových klientov vie na základe časových korelácií správ jednoducho potvrdiť, či daní klienti spolu komunikujú. Tor sa kvôli praktickej nemožnosti obrany proti potvrdeniu prenosu zameriava výhradne na obranu proti *analýze prenosu*, ktorá sa snaží odhaliť odosielateľa alebo príjemcu danej správy. Viac o analýze a potvrdení prenosu v kapitole 1.7.

- **Nezaoberať sa prenášanými dátami**. Tor ponecháva na odosielateľovi, aby jeho dáta poskytnuté koncovému klientovi neodhaľovali jeho identitu. V prípade prehliadania webových stránok poskytujú aktuálne verzie Toru [55] užívateľom webový prehliadač s integrovanou anonymizačnou proxy Privoxy [54]. Viac o anonymizačných proxy v kapitole 1.4.2.

2.2 Pôvodný protokol Onion routing

2.2.1 Úvod

Protokol Onion routing [73] bol navrhnutý v roku 1997. Jeho cieľom je zabezpečiť anonymnú komunikáciu s nízkou odozvou. V sieti verejne známych routrov vytvára dlhotrvajúce spojenia cez niekoľko z nich, v ktorých každý router pozná iba svojho predchodcu a nasledovníka.

Názov *Onion routing* je odvodený z názvu špeciálnej správy *onion* (cibuľa), ktorá svojimi šifrovými vrstvami pripomína cibuľu. Návrh protokolu umožňuje postaviť sieť prakticky na akomkoľvek transportnom protokole. Protokol bol implementovaný iba na malej lokálnej sieti piatich onion routrov. Napriek tomu sieť spracovávala viac ako 50 tisíc správ denne [23].

2.2.2 Pôvodný návrh

V kapitole popisujeme pôvodnú verziu protokolu navrhnutú Goldschlagom, Reedom a Syversonom v článkoch [73, 61, 33, 71].

Predpoklady. Predpokladáme aktívneho lokálneho útočníka v prostredí definovanom v kapitole 1.2. Ďalej predpokladáme statickú sieť v ktorej má každé zariadenie svoj verejný kľúč a verejne známu adresu. Medzi každou dvojicou zariadení je na *linkovej vrstve* vytvorené priame *šifrované spojenie*. Poznamenajme, že kvôli šifrovaniu na linkovej vrstve vyzerá rovnaká správa prechádzajúca medzi rôznymi linkami inak.

Prehľad. Hlavným rozdielom návrhu Onionu Routingu oproti mixovým sieťam je, že prenos dát prebieha dvojfázovo. V prvej fáze klient pošle *onion*, anonymnú adresu pozostávajúcu z viacerých vrstiev. Zariadenia, *onion routre*, cez ktoré onion postupne prechádza, vytvárajú *virtuálny obvod*, ktorý si onion routre pamätajú. V druhej fáze sa obojsmerne posielajú dáta cez vytvorený virtuálny obvod.

Hlavnou motiváciou rozdelenia vytvárania virtuálneho obvodu a posielania dát je, že počas posielania dát sa používajú už len symetrické šifrovacie transformácie, ktoré sú výrazne rýchlejšie ako asymetrické. To je kľúčovou vlastnosťou pre obvody s nízkou latenciou.

Sieť a prístup do siete Onion routingu sú podobné ako v prípade Mixminionu[42]. Klient si určí náhodnú cestu, cez ktoré budú jeho správy posielané. Spôsob, keď si klient určí vlastnú cestu pre svoje správy, sa nazýva *voľné smerovanie* (free routing).

Pojmy a označenia. Všetky správy v protokole sú prenášané v *cell*och (bunkách). Cell pozostáva z *hlavičky* (header) a *nákladu* (payload). Hlavička má jednu z hodnôt *padding* (*vypchávka*), *create* (*vytvoriť*), *data* (*dáta*) a *destroy* (*zničiť*). V prípadoch *create* a *destroy* je nákladom onion, v prípade *data* prenášané dáta a v prípade *padding* náhodné dáta používané ako *falošné správy* (dummy messages)[43].

Dopredné a spätné šifrovacie transformácie s kľúčom k označujeme štandardne $E_k(X)$ a $D_k(X)$. Pričom ich považujeme za symetrické, ak je kľúč určený pre symetrické

[42] Viac o Mixminione v kapitole 1.5.3.

[43] Viac o falošných správach v kapitolách 1.5.5 a 2.3.2.

šifrovanie a za asymetrické, ak je kľúč určený pre asymetrické šifrovanie.

Vytvorenie virtuálneho spojenia. Ak chce klient A vytvoriť anonymný komunikačný kanál s užívateľom B, tak si najprv zvolí náhodnú[44] cestu C cez onion routre R_1 až R_n, ktorej dĺžka je maximálne 11 [71].

Následne A vytvorí *onion*, pozostávajúci dokopy z n *vrstiev*, jednej vrstvy pre každý onion router na C. Každá vrstva V_i obsahuje kľúč KS_i pre symetrické šifrovanie, časovú pečiatku T a adresu A_i nasledujúceho routru. Špeciálne pre koncové adresy určíme $A_0 = A$ a $A_{n+1} = B$. To, či adresa A_{n+1} je adresou onion routra alebo adresou mimo sieť onion routrov, vie R_n zistiť zo zoznamu adries všetkých onion routrov (predpoklad statickej siete).

Klient onion zašifruje verejnými kľúčmi onion routrov tak, aby každý onion router vedel prečítať iba svoju vrstvu a pred zabalený onion pripojí náhodný identifikátor spojenia IS. Formálne, postupne od $i = n$ do $i = 1$ odosielateľ vykonáva asymetrické šifrovacie transformácie a vytvorí tak onion:

$$O_0 = IS||E_{K_1}(V_1, E_{K_2}(V_2, \ldots, EK_n(V_n)\ldots)),$$

kde K_i je verejný kľúč R_i. Onion bude nákladom cellu a *create* bude hlavičkou cellu. Ďalej sa zaoberáme už len samotným nákladom správy.

Klient vytvorený onion pošle R_1, ten onion *rozbalí* - dešifruje svojim súkromným kľúčom, zistí A_2 a pošle onion ďalej po ceste. Rovnakým spôsobom onion prepošlú aj zvyšné R_i až kým onion neprijme R_n.

Presnejšie, každý z $R_1, R_2, \ldots, R_n$ vygeneruje vzhľadom na seba unikátny *identifikátor anonymnej komunikácie* IAK_i (špeciálne $IAK_0 = IS$) a pošle onion pre R_{i+1} v tvare:

$$O_i = IAK_i||EK_{i+1}(V_{i+1}, E_{K_{i+2}}(V_{i+2}, \ldots, E_{K_n}(V_n)\ldots)).$$

Navyše si R_i zapamätá[45] zobrazenie $(IAK_{i-1}, A_{i-1}) \mapsto (IAK_i, KS_i, A_{i+1}, T)$.

Keď R_n onion spracuje, tak je vytvorený *virtuálny obvod* z routrov $R_1, R_2, \ldots, R_n$, po ktorom je možné obojsmerne posielať správy.

[44] Nasledujúci router na ceste je vybraný rovnomerne náhodne. Iné prístupy poskytujú útočníkovi viac informácie.

[45] Keďže IAK_{i-1} je pre R_{i-1} unikátny, tak je zobrazenie dobre definované.

Prenos dát. Počas existencie virtuálneho obvodu môže A posielať dáta d užívateľovi B. Podobným spôsobom môže B cez vytvorený obvod posielať anonymné odpovede pre A. Správy od A pre B označujeme ako *dopredné správy* a správy od B pre A označujeme ako *spätné správy*. Iný spôsob posielania anonymných odpovedí využitím reply onionov popisujeme v kapitole 2.2.3.

Na všetky správy prechádzajúce po virtuálnom obvode aplikujú onion routre symetrické šifrové transformácie podľa kľúčov KS_i. Na dopredné správy aplikujú D_{KS_i} a na spätné správy E_{KS_i}. Poznamenajme, že dopredná aj spätná správa je odhalená pre B, R_n a A.

Podrobnejšie popíšeme dopredný prenos dát od A pre B. Predtým, ako A pošle dáta prvému onion routru tak ich zašifruje v opačnom poradí, ako sú onion routre na ceste a na začiatok pripojí identifikátor spojenia. Formálne:

$$d_1 = IS||KS_1(KS_2(\ldots KS_n(d)\ldots)).$$

Každý router podľa IAK_{i-1} a A_{i-1} získa štvoricu $(IAK_i, KS_i, A_{i+1}, T)$. Použitím kľúča KS_i dešifruje jednu vrstvu, pripojí k dátam identifikátor spojenia a pošle ich ďalej. Formálne:

$$d_{i+1} = IAK_i||E_{KS_{i+1}}(E_{KS_{i+2}}(\ldots E_{KS_n}(d)\ldots)).$$

Posledný router posiela B nešifrované dáta d. Tieto dáta môže A po vytvorení obvodu zabezpečiť kľúčom získaným DH-protokolom na výmenu kľúča, alebo pomocou PGP[46].

Pri spätných správach sú správy postupne obaľované do šifrových vrstiev a nakoniec A všetky vrstvy rozbalí naraz. Formálne onion routre vykonávajú na spätných správach transformáciu:

$$IAK_i||E_{KS_{i+1}}(E_{KS_{i+2}}(\ldots E_{KS_n}(d)\ldots)) \mapsto IAK_{i-1}||E_{KS_i}(E_{KS_{i+1}}(\ldots E_{KS_n}(d)\ldots)).$$

Vyčistenie. Keď už obvod nie je potrebný, tak A pošle *nový* onion spolu s príkazom *destroy* - zničiť. Ak onion router dostane príkaz *destroy* s platným onionom, tak je povinný vymazať všetky informácie prislúchajúce k danému virtuálnemu spojeniu.

Ak je spojenie pristaré, môže sa onion router rozhodnúť vymazať svoje dáta o spojení.

[46]PGP vyžaduje od A znalosť verejného kľúča B. Kľúč musí A získať anonymným spôsobom, inak by odhalil svoju identitu ešte pred prenosom.

Vstupný a výstupný kanál. *Vstupný kanál* (entry funnel) a *výstupný kanál* (exit funnel) sú abstrakciami nad pripojeniami klientov ku sieti onion routrov. Klienti komunikujú cez kanále priamo s onion routrom. Ide o dôležité objekty protokolu, keďže útočník odpočúvaním oboch kanálov vie na základe časových korelácií správ zistiť, či klienti na konci kanálov spolu komunikujú.

Špeciálnu pozornosť si vyžaduje výstupný kanál. Dáta prenášané cez výstupný kanál sú protokolom Onion routing nešifrované a môžu odhaliť identitu A. V prípade prehliadania webových stránok sa preto odporúča použiť anonymizačné proxy[47], ktoré opisujeme v kapitole 1.4.2.

Proxy. Pôvodný návrh predpokladá dve proxy zariadenia na strane klienta. *Aplikačné proxy*, ktoré očisťuje dáta z aplikácií aby neobsahovali informácie o identite užívateľa. *Onion proxy*, ktoré komunikuje s aplikačným proxy a kanálmi a je zodpovedné za vytváranie a spracovanie onionov.

2.2.3 Rozšírenia a ďalšie špecifikácie

Spätný onion (Reply onion). Účelom *spätného onionu* je umožniť akémukoľvek účastníkovi B vytvoriť anonymné spojenie s neznámym tvorcom spätného onionu A [61]. Spätný onion vytvára A a môže ho poslať napríklad v dátach pôvodného spojenia alebo ho anonymne uverejniť. Jeden spätný onion sa môže použiť iba raz, kvôli útoku opakovaním.

Jediným rozdielom medzi spätným onionom a štandardný onionom je, že posledná vrstva je určená pre A. Presnejšie, A si zvolí ľubovoľných n onion routrov R_i, pričom cesta k nemu vedie od R_n po R_1, a identifikátor spojenia IS. Následne A vytvorí spätný onion, ktorý je rovnaký ako keby bol vytvorený štandardným spôsobom, ale z druhej strany cesty. Všetky použité informácie si A zapamätá pod identifikátorom IS. Formálne:

$$s\_onion = IS||A_n||E_{K_n}(V_n, EK_{n-1}(V_{n-1}, \ldots, E_{K_1}(V_1, E_{K_A}(IS))\ldots)),$$

kde $V_i = (A_{n-1}, KS_{n-1}, T)$, A_i je adresa R_i, KS_i je kľúč pre symetrické šifrovanie a T je časová pečiatka.

[47] Protokol HTTPS nestačí, keďže B vie napríklad z HTTP hlavičky odhaliť identitu A.

B pošle *s_onion* bez adresy A_n onion routru R_n a ten buduje obvod, ako keby bol B tvorcom. Keď A dostane správu $E_{KA}(IS)$, tak ju dešifruje svojim súkromným kľúčom, podľa IS priradí k novému spojeniu kľúče KS_i a vytvorí tak *spätný virtuálny obvod*. Všetky R_i sa ďalej správajú, ako keby to bol štandardný obvod vytvorený B.

Keďže B posiela nezabalené správy, tak doručený dátový cell pre A má tvar:

$$DATA||IAK_1||D_{KS_{n+1}}(D_{KS_n}(\ldots D_{KS_1}(m)\ldots)).$$

Uvoľnené smerovanie (Loose routing). V článku [33] sa opisuje technika *uvoľneného smerovania*, ktorá umožňuje vytvoriť ľubovoľne dlhé virtuálne obvody.

Myšlienkou je, že namiesto toho, aby onion router R_k pri vytváraní obvodu poslal onion O_k priamo R_{k+1}, tak O_k zabalí do nového onionu O'_k určeného pre routrom R_k zvolenú cestu $R_{j_1}, R_{j_2}, \ldots, R_{j_m} = R_{k+1}$. Onion O'_k má na nových vrstvách rovnaké informácie akoby ju vytváral odosielateľ. Formálne:

$$\begin{aligned} O'_k &= K_{j_1}(V_{j_1}, K_{j_2}(V_{j_2}, \ldots, K_{j_{m-1}}(V_{j_{m-1}}, O)\ldots)) \\ &= K_{j_1}(V_{j_1}, \ldots, K_{j_{m-1}}(V_{j_{m-1}}, K_{k+1}(V_{k+1}, \ldots, K_n(V_n)\ldots))\ldots), \end{aligned}$$

kde $K(X)$ je skrátený zápis pre $E_K(X)$.

Vytvorí sa tak virtuálny obvod:

$$R_1 \to R_2 \to \ldots \to R_k \to R_{j_1} \to R_{j_2} \to \ldots \to R_{j_{m-1}} \to R_{k+1} \to R_{k+2} \to \ldots \to R_n.$$

Router R_k si navyše pamätá $V_{j_1}, V_{j_2}, \ldots, V_{j_m}$, ktoré používa na ďalšie zabalenie dopredných správ a na rozbalenie spätných správ v podobvode $R_{j_1} \to R_{j_2} \to \ldots \to R_{j_m}$. Je možné sa na predĺžený obvod pozerať tak, že R_k vytvoril nový virtuálny obvod s R_{k+1} po ktorom preposiela komunikáciu medzi A a B.

Technika uvoľneného smerovania nie je v Onion routingu ani Tore implementovaná, lebo je považovaná za zbytočnú [23] a umožňuje vytvárať ľubovolne dlhé obvody, ktoré môžu byť využité útočníkom [58]. Viac v kapitole 2.4.3.

Časové pečiatky. Onion routing sa proti útoku opakovaním[48] bráni podobne ako Mixmaster, časovými pečiatkami. Každý onion router si pamätá každý prijatý onion,

[48] Viac o útokoch opakovaním v kapitole 2.4.4.

kým má v platnosti svoju časovú pečiatku. Každý ďalší prijatý onion sa porovnáva s onionmi v pamäti a v prípade, že je rovnaký, tak je zahodený.

Tým sa znemožní vytvorenie rovnakého obvodu útočníkom po existencií pôvodného virtuálneho obvodu. Útočník má preto obmedzený čas na to, aby využil existujúce virtuálne spojenie na svoje útoky. Poznamenajme, že takéto použitie časových pečiatok nebráni útočníkovi posielať rovnaké dátové správy.

V prípade spätných onionov sa používajú časové pečiatky s výrazne vyššou hodnotou, keďže nevieme dopredu kedy sa spätné virtuálne spojenie vytvorí [33].

Falošné správy (Dummy messages). Väčší počet správ v sieti zväčšuje množinu anonymity. Intuitívne sa preto snažíme zakaždým zabezpečiť dostatočný prenos správ v celej sieti. Jednou z možností je posielať falošné správy, ktoré sú na linkovej vrstve nerozlíšiteľné od skutočnej komunikácie.

Onion routing ich špecifikuje hlavičkou *padding* v celloch prenášaných správ. Falošné správy vytvárajú a posielajú medzi sebou onion routre, ich obsahom je náhodný šum a sú ihneď po prijatí zahodené. Aby neprichádzalo k zbytočnému zahlteniu siete, sú falošné správy vytvárané len v prípade, keď je prenos dát malý alebo žiadny. Pripomeňme, že komunikácia na linkovej vrstve medzi onion routrami je šifrovaná a preto stačí informáciu o falošnej správe ponechať v hlavičke.

Falošné správy[49] považuje Tor za nevýhodné [23] a podľa [14] nie sú ich výhody a nevýhody dostatočne pochopené.

Vypchávka (Padding). Veľkosť onionu môže byť užitočnou informáciou pre útočníka, keďže pri prechode onionu cez onion routre sa jeho dĺžka skracuje.

Onion routing preto navrhuje rovnakú dĺžku pre všetky oniony v sieti. Po rozbalení onionu v onion routri sa pridá náhodný padding za koniec onionu a pred začiatok onionu sa pridá informácia o jeho skutočnej dĺžke. Pripomeňme, že komunikácia na linkovej vrstve medzi onion routrami je šifrovaná. Poznamenajme, že pred nečestnými onion routrami je z princípu nemožné utajiť informáciu o dĺžke spracovaného onionu.

Podľa tvorcov Freedomu [1], siete využívajúcej onion router, je vypchávanie na rovnakú dĺžku neekonomické.

[49] Viac o falošných správach v Tore v kapitole 2.3.2.

Mixovanie správ. Zväčšovanie množiny anonymity preusporiadaním prijatých správ je prvotnou myšlienkou na nadobudnutie anonymity [7], popísanej v kapitole 1.5.

V prípade Onion routingu je poradie správ miešané pseudonáhodne, pričom sa dodržuje poradie správ v rovnakých virtuálnych spojeniach [73]. Časový interval na zhromaždenie správ je výrazne nižší ako v prípade mixových sietí. Musí byť na úrovni latencie sieťového spojenia, aby sa neporušila vlastnosť nízkej latencie Onion routingu.

V Tore [23] sa mixovanie nevyužíva. Viac v kapitole 2.3.2.

Šírenie informácií o onion routroch. V praxi nepredpokladáme statickú sieť. Preto je nutné informovať každého účastníka o zmenách verejných informácií onion routrov, ako napríklad verejné kľúče, adresy alebo prenosové rýchlosti. Tvorcovia Onion Routingu uvažujú peer-to-peer prístup. Informácia o zmene je preposielaná medzi onion routrami, ktoré navzájom práve komunikujú. Tento prístup zbytočne zahlcuje sieť a preto Tor používa na distribúciu verejných informácií centrálne autority, nazývané priečinkové servre. Viac o priečinkových servroch v kapitole 2.3.2.

2.3 Špecifiká Toru

V kapitole porovnávame Tor s pôvodným návrhom Onion routingu.

Zachovávame významy pojmov *onion router* a *onion proxy*. Pojmy ako virtuálny obvod, aplikačná proxy alebo funnel nepoužívame, alebo definujeme inak. V prípade, že je onion router členom reálnej *siete Toru*, tak sa nazýva *Tor relay*. Pojmy Tor relayu a onion routru aj onion proxy a klienta často zamieňame podľa kontextu.

Predpoklady. Každý onion router má jeden asymetrický *kľúč identity* s dlhou životnosťou a jeden asymetrický *onion kľúč* s krátkou životnosťou používaný pri vytváraní obvodov. Predpokladáme dynamickú verejnú sieť, v ktorej sú medzi každými dvoma Tor relaymi vytvorené TLS [18] spojenia.

2.3.1 Virtuálne obvody a prúdy

Tor je určený na anonymizáciu komunikácie v protokole SOCKS [42], ktorý zahŕňa väčšinu štandardnej komunikácie na Internete. Anonymné dáta sú prenášané cez *prúdy*

(streams), ktoré sú súčasťou obojsmerných *obvodov* (circuits). Obvody umožňujú jeho *tvorcovi* (initiator) obojsmernú *anonymnú* komunikáciu s každým onion routrom na obvode okrem prvého z nich.

Predpokladáme, že čitateľ je oboznámený s metódou budovania virtuálnych obvodov v protokole Onion routing popísanej v kapitole 2.2.2.

Cell. Základnou komunikačnou jednotkou v sieti Tor, je ako v prípade Onion routingu, *cell*. Každý cell má presne 512 bajtov a pozostáva z *hlavičky*, *identifikátoru obvodu* a *nákladu* [23].

Cell môže byť dvoch typov, *control cell* (umožňujúci kontrolu) a *relay cell* (zabezpečujúci prenos). Control cell je určený na vytváranie a mazanie obvodu a relay cell na prenos dát a špecializovanej informácie. Detaily budú poskytnuté nižšie.

Hlavička **control cellu** prenáša jednu z hodnôt:

- *create* - *vytvoriť*, slúži na vytvorenie nového obvodu.
- *created* - *vytvorené*, informuje o úspešnom vytvorení obvodu.
- *destroy* - *zničiť*, prikazuje zničiť obvod.
- *padding* - *vypchávka*, prenos falošnej správy.

Hlavička **relay cellu** umožňuje po vytvorenom spojení posielať príkazy, ktoré v ďalších odstavcoch vysvetlíme. Štruktúra hlavičky je:

1. *stream_id* - *identifikátor prúdu*, identifikuje spojenie využívajúce obvod, napríklad spojenie TCP.
2. *payload_length* - *dĺžka nákladu*, počet zvyšných bajtov tvoriacich informácie a nie šum.
3. *relay_command* - *príkaz*, má jednu z nasledujúcich hodnôt:
   - *relay_data* - *dáta*, v náklade sú dáta.
   - *relay_begin* - *začiatok*, začiatok prúdenia dát na novom prúde.
   - *relay_end* - *koniec*, koniec prúdenia dát.

- *relay_teardown* - *strhnúť*, bezdôvodné zničenie obvodu.
- *relay_connected* - *pripojený*, druhá strana potvrdzuje prijatie *relay_begin*.
- *relay_extend* - *predĺžiť*, pridanie ďalšieho onion routra na koniec obvodu.
- *relay_extended* - *predĺžené*, potvrdenie pridania.
- *relay_truncate* - *odseknúť*, odobratie posledného onion routra z obvodu.
- *relay_truncated* - *odseknutý*, potvrdenie odobratia.
- *relay_sendme* - *pošli*, používa sa na kontrolu preplnenia siete.
- *relay_drop* - *zahoď*, falošná správa.

Relay cell je posielaný jedine cez vytvorený obvod a je vždy šifrovaný 128-bitovým AESom [52] v CTR móde[50] pri každom prechode cez onion router na obvode. Pripomeňme, že v tomto prípade sú šifrovanie a dešifrovanie rovnakými transformáciami a všetci vlastníci kľúča si musia pamätať aktuálny stav inicializačného vektora.

Postupné budovanie obvodu (Incremental path building). Tor nepoužíva onions na vytváranie obvodov. Obvody sa vytvárajú predlžovaním už existujúcich obvodov pod kontrolou tvorcu.

Označme A tvorcu obvodu, C identifikátor obvodu a $R_0, R_1, \ldots, R_n$ onion routre vybraté pre obvod, špeciálne $R_0 = A$. Tvorca A nemusí byť onion router, volíme tak pre vhodnosť zápisu. Ďalej, nech $K_0, K_1, \ldots, K_n$ sú verejné RSA [65] kľúče R_i, g je generátor grupy pre Diffie-Hellmanov protokol[51] na výmenu kľúča a nech H je jednosmerná hashovacia funkcia. Pripomeňme, že všetky *priame spojenia* medzi onion routrami *sú šifrované TLS*.

Na začiatku obvod pozostáva iba z tvorcu A. Popíšeme vytváranie obvodu pre k od 0 až po $n-1$. Nech doteraz vytvorený obvod je $R_0 \to R_1 \to \ldots \to R_k$. Potom A pošle[52]

[50] CTR mód vytvára z blokovej šifry prúdovú šifru, inkrementovaním dohodnutého náhodného inicializačného vektora, ktorý sa pripája ku dohodnutému kľúču. Kvôli optimalizácií veľkosti hlavičky je inicializačný vektor odvodený z kľúča.

[51] Diffie-Hellmanov (DH) protokol [19] na výmenu kľúča je určený na bezpečné dohodnutie spoločného kľúča v nešifrovanom spojení. Na vytváraní kľúča sa podieľajú obe strany náhodnými číslami x a y a k nim prislúchajúcim g^x a g^y.

[52] V prípade $k = 0$ posielanie tejto správy v skutočnosti vynecháva, keďže R_0 = A.

control cell create c_k pre R_k, obsahujúci náklad $(C, E_{K_{k+1}}(g^{x_{k+1}}))$, ktorý je zabalený v k vrstvách určených pre routre $R_1, R_2, \ldots, R_k$. Formálne:

$$c_k = create||C||X_{KS_1}(X_{KS_2}(\ldots X_{KS_k}(EK_{k+1}(g^{x_{k+1}}))\ldots)),$$

kde C je identifikátor obvodu, X_d je šifrovanie AES v counter móde s kľúčom d a KS_i sú kľúče dohodnuté medzi A a R_i DH protokolom. Autenticita R_k, ako zabezpečenie pred man-in-the-middle[53] útokom, je dosiahnutá schopnosťou dešifrovať $EK_{k+1}(g^{x_{k+1}})$ V prípade $k = 0$ sa všetky transformácie X_{KS_i} vynechajú.

Onion router R_k dostane po obvode správu $create||C||E_{K_{k+1}}(g^{x_{k+1}})$ a prepošle ju ďalej R_{k+1}. Onion router R_{k+1} správu dešifruje, vygeneruje podľa DH protokolu $g^{y_{k+1}}$, získa tak kľúč KS_{k+1} a pošle späť po obvode správu pre A obsahujúcu:

$$created||C||g^{y_{k+1}}||H(KS_{k+1}),$$

ktorú routre postupne zaobaľujú do šifrových vrstiev. Špeciálne R_k si zapamätá predĺženie obvodu s adresou R_{k+1} a C.

Formálne, A dostane správu:

$$created||C||X_{KS_1}(X_{KS_2}(\ldots X_{KS_k}(g^{y_{k+1}}, H(KS_{k+1}))\ldots)).$$

Tvorca obvodu A postupným dešifrovaním vrstiev získa $(g^{y_{k+1}}, H(K_{k+1}))$, podľa DH protokolu odvodí kľúč KS_{k+1} a overí jeho správnosť pomocou $H(KS_{k+1})$. Všimnime si, že A použil rádovo rovnako veľa šifrových transformácií X, D a E ako sieť.

Postupným budovaním vytvorí A obvod až po R_n. Po vytvorení obvodu je možné posielať celly oboma smermi. Označme *hore* smer od A, *dole* smer k A a samotný virtuálny obvod ako $A \Rightarrow R_n$, kde $\Rightarrow$ znázorňuje smer budovania. Náklady cellov sú v oboch smeroch v R_i šifrované X_{KS_i}, v prípade relay cellov sa šifrujú aj hlavičky.

Onion router R_1 budeme nazývať *vstupným bodom* (guard node) siete Toru a onion router R_n budeme nazývať *výstupným bodom* (exit node) siete Toru.

[53] Útok útočníka uprostred spočíva v tom, že útočník C je sprostredkovateľom komunikácie medzi A a B. Úspešný útok presvedčí A aj B, že sa dohodli navzájom, ale v skutočnosti sa obaja dohodli s C. V prípade DH protokolu bez autentifikácie by C vedel dohodnúť kľúče K_A a K_B s A a B a čítať tak ich správy.

Poznamenajme, že doprednú správu vidia v otvorenom tvare iba A a R_{k+1} a spätnú správu vidia v otvorenom tvare iba A, R_k a R_{k+1}. Keďže obe správy obsahujú informácie iba podľa DH protokolu, tak je možné [23] podľa metódy [47] dokázať perfektnú doprednú anonymitu A pre $R_2, R_3, \ldots, R_n$ za predpokladu čestných účastníkov.

Ďalej poznamenajme, že hodnota C je rôzna pre všetky dvojice onion routrov na obvode. Zakaždým R_k volí hodnotu C_k tak, aby bola unikátna pre dvojicu (R_{k-1}, C_{k-1}). Všetky onion routre si pamätajú mapovanie $(R_{k-1}, C_{k-1}) \mapsto (R_{k+1}, C_k)$.

Hlavnou výhodou postupného budovania obvodov oproti onionom je bezpečnosť. Po prvé, nie je nutné sa brániť proti útokom opakovaním. Útočník nevie po vytvorení obvodu simulovať proces budovania, keďže nepozná dohodnuté kľúče KS_i. Po druhé, v prípade onionov bolo možné, aj po zničení obvodu, postupným získaním kontroly nad onion routrami, zistiť celú cestu a odhaliť tak A a aj všetky kľúče pre symetrické šifrovanie. V prípade Toru to nie je prakticky možné, keďže je nutné poznať kľúče KS_i, ktoré sú priamo závislé na zmazaných x_i a y_i. Tým je zabezpečená odosielateľova anonymita za predpokladu, že je R_1 čestný. Poznamenajme, že opakovanie správ počas existencie obvodu je z pohľadu útočníka ekvivalentné, ako posielanie *relay_data* správ s rovnakým identifikátorom obvodu.

Vytváranie prúdov. Ak chce A vytvoriť nové SOCKS spojenie s B, tak si onion proxy vyberie najnovšie vytvorený obvod a pošle cez neho správu *relay_begin* obsahujúcu nový náhodný *stream_id* a C. Posledný onion router na obvode R_n vytvorí priame SOCKS spojenie s B, zapamätá si mapovanie $(C, stream\_id) \mapsto B$ a pošle *relay_connected* spolu s odpoveďou od B dole po obvode. Akonáhle A dostane správu *relay_connected*, je úspešne vytvorený nový *prúd*. Ďalej R_n preposiela komunikáciu určenú *stream_id* medzi B a obvodom (prúdom). Poznamenajme, že daný *stream_id* poznajú iba A a R_n. Ako už bolo spomenuté, cestou dole po obvode sa správy rozbaľujú a cestou hore sa správy zaobaľujú, preto ich obsah poznajú len A, R_n a B.

Ak chce A spojenie ukončiť, tak pošle cell *relay_end*, R_n ukončí spojenie s B a vymaže asociáciu $(C, stream\_id) \mapsto B$.

Problémom niektorých protokolov môže byť interpretovanie URL adresy DNS servrom pred samotným vytvorením SOCKS spojenia. Útočník vie v takom prípade ľahko zistiť, kam sa A pripája. V prípade webových prehliadačov je DNS požiadavka presme-

rovaná cez prúd.

Kontrola prúdov a obvodu. Priamym spôsobom, ako môže ktorýkoľvek onion router *zničiť* obvod, je poslať control cell *destroy* hore a dole po obvode. Pripomeňme, že hlavičky control cellov nie sú šifrované. Druhým spôsobom je postupné mazanie obvodu kontrolované A. Podobne, ako A obvod budoval, môže ho aj mazať. V každom kroku $k = 0, 1, \ldots, n-1$ pošle poslednému R_{n-k} cell *relay_truncate* a R_{n-k} mu odpovie *relay_truncated*.

V prípade, že A chce existujúci obvod predĺžiť, môže tak urobiť alternatívnym spôsobom pomocou príkazov *relay_extend* a odpoveďou je *relay_extended*.

Klient A buduje nový obvod rádovo každú minútu a môže mať naraz vytvorených viacero obvodov. Obvody vytvára vopred, keďže ide o pomerne dlhotrvajúcu operáciu. Nový prúd vytvára na najnovšom obvode. Staré obvody, ktoré už nemajú otvorený prúd ihneď ničí kvôli bezpečnostným dôvodom.

2.3.2 Ďalšie rozdiely oproti Onion routingu

Overovanie integrity správy na koncoch prúdu. Overovanie integrity správ je obranou proti zmene dát útočníkom[54] a označovaciemu útoku popísaného v kapitole 1.7. Z dôvodov uvádzaných v kapitole 1.5.3 o SURBoch nie je možné overovať integritu spätnej správy medzi každými dvoma onion routrami pri zachovaní symetrie. Aj preto Tor overuje integritu správ iba na koncoch prúdov [23] a predpokladá čestnosť výstupného bodu.

Označme A tvorcu prúdu a R výstupný bod prúdu. Obaja majú na začiatku inicializované rovnaké hodnoty H_A a H_R odvodené z ich spoločného kľúča KS_n. Pred zabalením správy A vypočíta SHA-1 [51] *odtlačok* správy, prvých 32 bitov z neho prixoruje k H_A a túto hodnotu uloží do poľa *checksum* v relay celle. Keď R rozbalenú správu dostane, rovnako vypočíta SHA-1 odtlačok správy a prvých 32-bitov z neho prixoruje k H_R. Ak H_R nekorešponduje s hodnotou *checksum*, tak R obvod zničí.

Pravdepodobnosť, že sa útočníkovi podarí naslepo zmeniť správu a aj jej *checksum* je zanedbateľná v porovnaní s tým, že neúspešný útok vedie k zničeniu obvodu. Navyše,

[54] Zmena dát bez spozorovania je v prípade prúdových šifier jednoduchá. V prípade, že by vedel útočník uhádnuť časť správy, vie túto časť zmeniť ľubovoľne (pri zachovaní počtu znakov).

zapamätaním si hodnôt H_A a H_R je vynútené, aby správy boli odoslané a prijaté v rovnakom poradí. Špeciálne v prípade útoku opakovaním vedie preposlanie správy (v prípade nenulového odtlačku) k nekorešpondujúcim H_A a H_R a k zničeniu obvodu.

Výpočtovo ide o efektívnu schému, keďže zložitosť popísaného spôsobu overovania integrity je zanedbateľná v porovnaní so šifrovaním AES na každom R_i.

Priečinkové servre (Directory servers). Pôvodný Onion routing predpokladal, že informácie o stave sieti sa budú šíriť sieťou. To je náročná operácia, ktorá môže byť ľahko zneužitá [23]. Tor preto implementuje centrálne *priečinkové servre*, ktoré majú aktuálne informácie jednotlivých onion routrov o ich adresách, verejných kľúčov, rýchlosti pripojení a ďalšie potrebné informácie. Klienti si informácie z priečinkových servrov pravidelne obnovujú.

Priečinkové servre sú často práve dôveryhodné onion routre. Každý server svoj obsah podpisuje, konzultuje s ostatnými servrami a overuje jeho aktualitu. Aby útočník nemohol behom vlastného servra poskytnúť nepravdivé informácie, tak je odporúčané klientom žiadať rovnakú informáciu od viacerých servrov. Viac o nebezpečenstvách priečinkových servrov v kapitole 2.4.3.

Výstupné pravidlá (Exit policies). Návrh Toru umožňuje každému onion routru určiť verejne známe *výstupné pravidlá*, podľa ktorých sa riadi pri komunikácií mimo sieť Toru. Tieto pravidlá sú uložené v priečinkových servroch. Motiváciou ich zavedenia je znížiť potenciálne zneužitie siete vandalmi, čo znižuje dôveryhodnosť a tým aj počet užívateľov siete [23].

Najjednoduchším pravidlom je nevytvárať žiadne externé spojenia. V opačnom prípade sa onion router nazýva *výstupným bodom*. Výstupný bod môže napríklad obmedziť podporované protokoly ako HTTP, SSH alebo AIM, alebo aj podporované porty ako 80 alebo 443.

Náročnejším pravidlom môže byť filtrovanie štandardných útokov útočníkov využívajúcich anonymné vlastnosti siete na skrytie svojej identity. Napríklad rozpoznávanie útokov voči databázam.

Podľa [12] bolo zavedenie výstupných pravidiel hlavným dôvodom, ktorý umožnil Toru rozšíriť sa v širokom rozsahu.

Žiadne mixovanie, vypchávky ani falošné správy. Navrhovatelia Toru nepovažujú mixovanie, vypchávky (padding) ani falošné správy (dummy messages) za prínosné [23]. Všetky tri techniky síce môžu priniesť zvýšenie anonymity, ale za cenu pomalšieho behu protokolu. V prípade paddingu argumentujú skúsenosťami zo siete Freedom [1] a teoretickými výsledkami o nebezpečenstvách paddingu [43]. O možných spôsoboch implementácie falošných správ teoretizujú[72]. Pripúšťajú, že sa ich rozhodnutie v budúcnosti zmení.

Deravé potrubie (Leaky pipe). Doteraz sme predpokladali, že práve posledný onion router na obvode je aj posledným onion routrom na prúde. Návrh Toru umožňuje vytvoriť prúdy, ktoré nekončia v R_n. Schéma, ktorá to dosahuje, sa nazýva *deravé potrubie* a môže skomplikovať analýzu prenosu [23].

Pripomeňme, že overovanie integrity medzi A a R_n bolo implementované pomocou poľa *checksum* a zapamätaných hodnôt H_A a H_R, pričom platilo $H_A \oplus SHA(m) = checksum = SHA(m) \oplus H_R$. Ak hodnoty H_A a H_R nekorešpondovali, tak sa obvod zničil.

Ak chce A poslať správu ľubovoľnému routru R_k na obvode, tak ju zabalí iba do k vrstiev. Každý z routrov R_i na obvode kontroluje, či *checksum* v hlavičke korešponduje so správou a zapamätanou hodnotou H_{R_k}. Ak áno, tak je určená pre R_k, nastaví $H_{R_k} = checksum$ a ďalej správu nepreposiela. Ak *checksum* nekorešponduje, tak správu pošle ďalej po obvode. Ak *checksum* nekorešponduje ani pri R_n, tak obvod zničí. Podobne R_k môže poslať dole po obvode správu, ktorá sa postupne zaobaľuje do k vrstiev. A ju skúsi n-krát dešifrovať a pri každom dešifrovaní overuje *checksum*. Implementačný detail je, že pre každú dvojicu (A, R_i) je nutné si pamätať hodnoty H_{A_i} a H_{R_i}.

2.3.3 Ďalšie schémy

Body stretnutia a skryté služby (Hidden services and rendezvous points). Predpokladajme, že A chce komunikovať so *skrytou službou* B, napríklad webserverom, ktorý chce svoju identitu ponechať anonymnou. Tor implementuje *schému bodov stretnutia* [23] na základe [32], ktorá umožňuje zachovať anonymitu A aj B. Koncept schémy je, že sa A a B dohodnú na onion routri R, ktorý bude ich *bodom stretnutia*. Štandardným postupným budovaním sa vytvoria virtuálne obvody $A \Rightarrow R$ a $B \Rightarrow R$,

medzi ktorými bude R komunikáciu preposielať.

Keď chce B sprístupniť svoju skrytú službu, tak vytvorí dlhodobý verejný kľúč, vyberie si onion routre, svoje *body uvedenia* (introduction point), a vytvorí k nim obvody. Mimo sieť Tor svoju skrytú službu popíše, uverejní svoje body uvedenia a tieto informácie podpíše. Tor navrhuje udržiavať centrálny repozitár URL adries v tvare $x.y.onion$, kde y je hash verejného kľúča B a x je autorizačné cookie [23]. Tieto adresy potom preloží onion proxy na množinu bodov uvedenia.

Klient A sa o službe dozvie, vyberie si bod stretnutia R a bod uvedenia I, vytvorí prúd $A \rightarrow R$ a cez neho požiada I o služby B. Ďalej označme zjednodušený prúd $A \Rightarrow R \rightarrow I \Rightarrow B$ ako prúd $A \longrightarrow B$ v ktorom R a I správy medzi prúdmi $A \Rightarrow R$ a $B \Rightarrow I$ preposielajú. Poznamenajme, že dáta sú odhalené len pre A, R, I a B a A ani B svoje identity neodhalili. Dáta je možné chrániť kľúčom, ktorý si môžu dohodnúť A a B pomocou DH protokolu.

Služba B môže čeliť podobným útokom ako štandardné webové servre, ako napríklad zahltenie služby.

Mosty (Bridges). Jedným zo základných cieľov Toru je poskytnúť prístup k sieti Toru užívateľom za firewallom alebo cenzúrovým systémom [23]. Zablokovať priamy prístup k sieti Toru je možné blokovaním IP adries všetkých Tor relayov. Preto Tor umožňuje každému klientovi byť *bridge relay*om - *mostom* M, cez ktorý sa cenzúrovaný klient C môže pripojiť k sieti Tor. C vytvorí priame spojenie s M a k sieti Toru sa pripája v mene M. Tým sa pripojenie k sieti Toru skryje za bežné spojenie, ktoré je z pohľadu cenzúrového systému podobné bežnému šifrovanému spojeniu. Množina mostov sa mení často a je potenciálne veľká.

Prvým krokom pre C je získanie adresy nejakého mostu. V sieti Tor sa okrem priečinkových servrov pre Tor relaye nachádzajú aj priečinkové servre pre mosty. Tie ale neposkytujú zoznam všetkých IP adries, ale vždy len malú podmnožinu z nich, aby nebolo možné blokovať všetky IP adresy mostov ako v prípade Tor relayov.

Podobne, každý most si môže pamätať krátky zoznam iných mostov. Prvý kontakt C s mostom je ale na iniciatíve C. Adresu mostu sa môže napríklad dozvedieť od kamaráta, z emailu alebo dierou vo firewalle [22]. Článok [70] popisuje možnosti sociálnych sietí zdielať cenzúrované informácie.

Most M vie čítať komunikáciu C so sieťou Toru. Nevie sa ale dozvedieť, s kým chce C komunikovať, z rovnakého dôvodu, ako keby bol aktívnym útočníkom na linke medzi A a onion routrami pri postupnom budovaní.

Komunikácia so sieťou Toru je špecifická[55] a môže byť ľahko odhaliteľná [17]. V prípade blokovania na základe vzorov komunikácie nastáva štandardná hra na "mačku a myš", keď sa útočník snaží využiť nedostatky obrany a naopak [22].

Kontrola upchania (Congestion control). Náhodný výber Tor relayov pre obvody môže viesť k *upchatiu* niektorých z nich, keď nestíhajú spracovávať správy. Upchatie môže byť bezdôvodné alebo byť dôsledkom útokov, popísaných v kapitole 2.4.3. *Kontrolou upchania* je možné predísť upchatiu a navyše umožní Tor relayom obmedziť ich prenosové rýchlosti.

Štandardné systémy kontroly upchania vyžadujú komunikáciu medzi prvkami siete a globálny pohľad na sieť. To je v prípade Toru neprípustné a preto Tor navrhuje [23] riešiť upchatie na troch úrovniach. Na úrovni Tor relayov prezentovaním poskytovanej rýchlosti prenosu[56] a na úrovniach obvodov a prúdov *oknami*. Okná sú počítadlá, ktorých aktuálny stav reprezentuje počet správ, ktoré je onion router ochotný spracovať.

V prípade prúdov majú A a výstupný bod R *dopredné* a *spätné* okná. Na začiatku sú obe okná inicializované na hodnotu 500. Každou správou sa príslušné okno dekrementuje. Keď R odoslal 100 správ, tak pošle dole po prúde *relay_sendme*. Naopak, keď A prijal 100 správ, tak pošle hore po prúde *relay_sendme*. Keď užívateľ dostane *relay_sendme*, tak inkrementuje príslušné okno o 100. Konštanty môžu byť rôzne v rôznych implementáciách Toru.

V prípade obvodov je protokol podobný.

V článku [76] navrhujú algoritmus, ktorý vyberá potenciálne menej Upchané Tor relaye na vytvorenie nových obvodov. Pred vytvorením obvodu námatkovo zmapujú upchatie množiny Tor relayov a potom z nej vyberú najvhodnejšiu.

[55] Viac o možnostiach útokov využívajúcich špecializovanú komunikáciu Toru v kapitole 2.4.2.

[56] Presnejšie triedami prenosových rýchlosti, ako napríklad DSL, ISDN alebo modem. Tento prístup používa napríklad Morphmix [64].

2.4 Útoky na protokol

2.4.1 Model útočníka

Predpokladáme aktívneho lokálneho útočníka, ktorého hlavným cieľom je na základe analýzy prenosu narušiť vzájomnú anonymitu. Ďalším cieľom útočníka môže byť zistiť s kým odosielateľ komunikuje alebo vytvoriť profil správania odosielateľa [23].

Útočník môže použiť *pasívne útoky* na zistenie korelácií medzi vstupným a výstupným prenosom na základe časovania, množstva alebo inej rozpoznateľnej vlastnosti [23].

Útočník môže použiť *aktívne útoky* a podieľať sa na behu siete, nabúravať onion routre za účelom získania ich kľúčov, opakovať správy, zamedzovať prístup k službám siete, presúvať užívateľov z dôveryhodných onion routrov na routre pod svojou kontrolou alebo vytvárať rozpoznateľné vzory v komunikácií. Môže sa pokúsiť nabúrať priečinkové servre za účelom poskytnutia nepravdivých informácií klientom. Navyše môže útočiť na spoľahlivosť siete priamymi útokmi na Tor relaye alebo znížením reputácie dôveryhodných užívateľov vykonávaním protizákonných, nečestných alebo vulgárnych aktivít. Zníženie celkovej reputácie siete vedie k zníženiu počtu užívateľov a zjednodušeniu ďalších útokov [23].

Keďže je Tor sieť s *nízkou latenciou*, je prakticky nemožné brániť sa proti útokom *potvrdzujúcim* komunikáciu medzi danými dvoma klientmi. Časové korelácie vstupných a výstupných správ vedia útočníkovi potvrdiť takúto komunikáciu. Tvorcovia Toru sa preto zameriavajú proti útokom, ktoré *analyzujú* s kým daný klient komunikuje [23].

2.4.2 Pasívne útoky

Odtlačky webových servrov (Website fingerprinting). V tomto odstavci predpokladáme, že klient komunikuje výhradne s webovými servrami a žiada od nich HTML stránky a ich ostatný obsah protokolom HTTP. *Odtlačok* webovej stránky je množina veľkostí (v bajtoch) preberaných súborov pre danú stránku. Poznamenajme, že štandardné správanie webových servrov a prehliadačov je jedným TCP spojením prebrať HTML stránku a následne postupne prebrať štýlopisy, skripty, multimédiá a iné externé súbory. Štandardne prehliadač v poradí výskytu odkazov v HTML iniciuje TCP spojenia, ktoré môžu prebiehať paralelne. Predpokladáme štandardné nastavenie pre-

hliadača, keď nie sú zablokované skriptovacie jazyky ani multimédiá a nepoužíva sa cache.

Hintz [37] využil fakt, že počet potenciálnych odtlačkov webových stránok je rádovo vyšší ako počet webových stránok na Internete, na ich identifikáciu. Poznamenajme, že zobrazenie URL stránok na ich odtlačky nie je bijektívne[57].

Hintz vytvoril útok na anonymizačnú proxy SafeWeb[58] založený na odtlačkoch webových stránok. Jeho implementácia bola minimalistická, ale potvrdila možnosť útoku založeného na odtlačkoch.

V prípade Toru je situácia komplikovanejšia. Útočník vie monitorovať len obvod, cez ktorý môže paralelne prechádzať viacero TCP spojení na viacerých prúdoch. Dáta sú navyše tvorené cellmi rovnakej veľkosti, čo znemožňuje presne identifikovať odtlačok webovej stránky.

Kvôli jednoduchosti predpokladáme, že útočník vie získať odtlačok s presnosťou na veľkosť cellov. V článku [57] použili Support Vector Machines [35] a implementovali útok v simulovanom prostredí tisícok najnavštevovanejších webových stránok (podľa poradia `alexa.com`), ktorý klasifikoval navštívené stránky s úspešnosťou 73%. Komunikácia bola zjednodušená na veľkosť premávky v čase a jej smer. Bralo sa v úvahu aj správanie browsera a vlastnosti TCP protokolu, ako napríklad ACK pakety. V prípade použitia falošných správ na úrovni dvojnásobku prenosu bola ich metóda neúspešná.

V článku [44] sledujú odtlačky v HTTPS protokole. Z webovej komunikácie University of Massachusetts vyfiltrovali HTTPS requesty voči ich 2000 najnavštevovanejším stránkam a dostali tak dáta s 480000 pripojeniami. Pomocou Bayesovského klasifikátora dosiahli úspešnosť 86,2% správnej klasifikácie na testovacích dátach. V prípade pridania falošných správ úspešnosť správnej klasifikácie neklesla pod 35,9%.

Využívanie odtlačkov webových stránok prináša pomerne dobré výsledky. Problémom môžu byť dynamické webové stránky, ktoré svoj obsah často menia. V budúcnosti sa uvažuje odhadnúť štandardné správanie užívateľa.

2.4.3 Aktívne útoky

[57] Stačí uvažovať čisté HTML stránky bez externého obsahu veľkosti maximálne 10KB, ktorých je určite viac ako 10240.

[58] Viac o anonymizačných proxy v kapitole 1.4.2 a o SafeWebe v článku [45].

Útok zahltením (Congestion attack). V čase, keď mala sieť Toru rádovo desiatky Tor relayov, bol uskutočniteľný útok [49], ktorý postupne *zahlcoval* veľkým množstvom správ každý Tor relay. Útočníkovým cieľom bolo zistiť, aký obvod klient vytvoril.

Útočník vytváraním vlastných krátkych obvodov a prenášaním veľkého množstva správ zahltí vybrané Tor relaye a spomalí tak všetky správy, ktoré cez ne prechádzajú. Ideálne by obvody mali prechádzať iba cez jeden Tor relay, ale keďže vybraný Tor relay vie zistiť, že je prvým aj posledným v obvode, tak sa vytvárajú obvody prechádzajúce cez dva Tor relaye. Poznamenajme, že Tor relay preposiela správy čakajúce na prúdoch algoritmom *Round Robin* - dookola prechádza prúdy a v jednom kole spracuje maximálne jednu správu pre každý z prúdov.

Počas existencie obvodu vie útočník postupným zahltením všetkých Tor relayov zistiť omeškanie, ktoré zahltenie pridá na Tor relay a korelovať ho s pridaným omeškaním, ktoré pozoruje na obvode svojho záujmu. Týmto spôsobom je možné zistiť, cez ktoré Tor relaye R klient svoj obvod vytváral. Ak má útočník pod kontrolou prvý Tor relay v obvode, tak vytvorením paralelného obvodu cez všetky R a ich zahltením vie identifikovať klienta [38].

Experiment na sieti Toru s 13timi Tor relaymi v novembri roku 2004 vedel identifikovať Tor relaye na obvode [49]. Podobným spôsobom je možné zistiť, či sú dva rôzne prúdy používané rovnakým klientom.

V súčasnej sieti tisícok Tor relayov je táto verzia útoku nepoužiteľná, keďže upchatie všetkých Tor relayov vyžaduje veľkú prenosovú rýchlosť a často upchatie obvodu nemusí korelovať s upchaním vytvoreným útočníkom [30].

Útok zahltením využívajúci heartbeat (Congestion attack). Problémy so zahltením značnej časti siete riešia v článku [30] z roku 2009. Cieľom je identifikovať klientov, ktorý komunikujú cez útočníkov výstupný bod s webový servrami pomocou identifikácie Tor relayov na klientovom obvode. Predpokladajú, že je možné vytvoriť pravidelný jednosmerný *heartbeat*[59] od klienta k webovému servru a využívajú možnosť vytvárať dlhé obvody. V článku uvažujú vloženie JavaScriptového kódu do klientovej komunikácie, ktorý pravidelne posiela požiadavky z klientskej strany. To je možné naprí-

[59]Heartbeat - *srdcový tep* - je pravidelným pingom štandardne určeným na monitorovanie stavu zariadení v distribuovaných systémoch.

klad v prípadoch, keď klient nepoužíva protokol HTTPS, koncový server nepodporuje HTTPS alebo klient neoveruje certifikát webového servra.

Kvôli prístupu Round Robin sa efekt zahltenia zvyšuje počtom vytvorených prúdov a ich rovnomerným zahltením. Aj preto pri zahlcovaní Tor relayu R vytvárajú *cyklické obvody*:

$$R \to R_1 \to R_2 \to R \to R_1 \to \ldots \to R_2 \to R \to R_1 \to R_2,$$

kde R_1 a R_2 sú Tor relaye s vysokou priepustnosťou. Existenciu cyklických obvodov návrh Toru nezakazuje, keďže nikto okrem tvorcu takého obvodu nevie zistiť, že je cyklický[60]. Uvažujú sa obvody dĺžky 24, keďže rádovo dlhšie obvody môžu mať problémy so stabilitou.

Pred samotným útokom sa vytvoria takéto obvody pre väčšinu Tor relayov. Keď už prebieha heartbeat od klienta k webovému servru, tak sa najprv na základe časových pečiatok obsiahnutých v heartbeatoch odhadne základná latencia spojenia. Potom sa postupným zahltením Tor relayov identifikujú relaye, cez ktoré obvod prechádza.

Celý útok musí prebehnúť v intervale, kým má klient webovú stránku otvorenú. Článok odhaduje, že tento prístup vie výrazne pomôcť všeobecnej analýze prenosu.

Kontrolou upchania sa zaoberáme v kapitole 2.3.3 - Kontrola upchania.

Útok na vytváranie obvodu (Path building attack). V prípade, že má útočník pod kontrolou vstupný aj výstupný bod obvodu, vie na základe časových korelácií vstupov a výstupov ľahko prelomiť vzájomnú anonymitu väčšieho počtu účastníkov [30]. Ak má útočník pod kontrolou viacero Tor relayov U, môže zamerať svoje úsilie na odhalenie klientov, ktorí taký obvod vytvoria. Poznamenajme, že Tor relay vie pomocou priečinkových servrov zistiť, či je prvý alebo posledný.

Pravdepodobnosť odhalenia sa dá zvýšiť tým, že $u \in U$ *odmietne* budovať obvody v ktorých nie vstupným alebo výstupným bodom obvodu [4]. V článku [2] začiatky R_1 a konce R_n obvodov korelovali ešte predtým ako boli prenášané dáta. Využili fakt, že Tor buduje obvody a prúdy deterministickým spôsobom a posledný $relay\_extend$ musí prejsť postupne $R_1 \to R_n \to R_1$, pričom dopredný aj spätný prenos trvajú pravdepodobne rovnako dlho a navyše mu predchádzalo postupné budovanie zachytené R_1.

[60] Viac o vytváraní obvodov v kapitole 2.3.1.

Týmto spôsobom je možné ušetriť zdroje útočníka, ktorý nemusí vyhodnocovať časové korelácie jednotlivých správ.

Využitím techník odmietania a korelovania pred budovaním, je možné v sieti 44 Tor relayov, z ktorých sú 2 pod kontrolou útočníka, odhaliť 9,1% všetkých spojení v experimentálnom prostredí [2].

Ďalším zlepšením môže byť vytváranie dlhých obvodov mimo U a ich upchatie, čím sa zvýši pravdepodobnosť vytvorenia požadovaného obvodu [58]. Vybudovanie obvodu dĺžky n stojí útočníka síce $O(n^2)$ šifrových operácií, ale po vytvorení vie útočník posielať celly pomocou $O(1)$ operácií, ktorých spracovanie stojí sieť $O(n)$ operácií. Poznamenajme, že $O(1)$ operácií stačí, keďže správy odoslané klientom sa môžu opakovať.

Využitie neaktuálnych priečinkových servrov (Directory server abuse). Ak sa podarí útočníkovi poskytnúť nepravdivé informácie o sieti Toru, tak vie uprednostniť svoje Tor relaye oproti ostatným.

Klient získava informácie o sieti Toru naraz od viacerých redundantných priečinkových servrov, ktoré svoj obsah podpisujú. Ak chce útočník nanútiť svoje informácie o sieti, tak musí získať dostatočnú reputáciu v sieti priečinkových servrov voči niektorým klientom. Nebezpečenstvo hrozí najmä novým klientom, ak im útočník zamedzí prístup k dôveryhodným priečinkom.

Viac o zneužití reputačných systémov v kapitole 1.7.

2.4.4 Nepravdepodobné útoky

V sekcií uvádzame zoznam niektorých štandardných útokov na anonymné siete, ktorým sa Tor bráni.

Označovanie správ (Tagging attack). Myšlienkou *označovania správ* je zmeniť správu tak, aby bola pri prenose alebo výstupe rozpoznateľná. Viac v kapitole 1.7.

Zmeny správ Tor kontroluje pri výstupe zo siete a preto musí mať útočník pod kontrolou výstupný bod R_n. Predpokladajme, že útočník má pod kontrolou aj iný Tor relay R_1, ktorý je vstupným bodom (scenár je podobný ako pri útoku na vytváranie obvodu). R_1 ku vstupnej správe m prixoruje bitovú masku b a pošle ďalej správu $l_1 = m \oplus b$. Tá je postupne dešifrovaná pri prechode onion routrami prixorovaním prúdovej

šifry: $l_{i+1} = l_i \oplus c_{i+1}$. Ak R_n dostane[61] l_n, tak zistí, že *checksum* nekorešponduje. Keďže $l_n = b \oplus m \oplus c_2 \oplus c_3 \oplus \ldots \oplus c_n$, tak prixorovaním $l'_n = l_n \oplus b$ získa R_n originálnu správu, ktorej *checksum* už korešponduje. Týmto spôsobom vie jednoznačne priradiť vstupné správy k výstupným.

Problémom útoku je zrušenie obvodu v prípade, že správa l_n neprechádza cez R_n (ako posledný onion router). To vedie k vysokej nespoľahlivosti R_1, ktorá môže byť nahlásená a alebo spozorovaná priečinkovými servrami.

Zopakovanie správ (Replay attack). Pripomeňme, že *checksum* je hodnotou, ktorá je spoločná pre klienta aj výstupný bod. Preto zopakovanie správ iným účastníkom ako tvorcom vedie k zničeniu obvodu. Aby bolo možné úspešne správu zopakovať, musel by byť výstupný bod pod kontrolou útočníka. Možnosti útočníka sú podobné, ako v prípade útoku označovania správ.

Zopakovaním správ môže útočník iniciovať DoS útok[62], keď zopakovaním každej správy klienta znemožní vybudovanie obvodu.

Postupné nabúranie (Iterated compromise). Predpokladajme, že útočník vie v reálnom čase získať kontrolu nad ľubovoľným onion routrom R - *nabúrať* R. Nech má zachytenú celú komunikáciu po obvode $R_1, R_2, \ldots, R_n$ klienta A a obvod už neexistuje. Potom vie nabúraním odhaliť dohodu kľúča DH protokolom, ale keďže obvod neexistuje, tak samotné kľúče boli zmazané R_i a preto sa ich útočník v reálnom čase nedozvie. Keby sa vedel nabúrať v čase existencie obvodu, tak vie ľahko zrekonštruovať celý obvod a aj komunikáciu.

Zdroje, ktoré umožňujú nabúranie, majú väčšinou len štáty využitím zákonov. Tie ale v čase existencie obvodu nevedia svoju silu realizovať. Viac o právnych útokoch v kapitole 1.7.

[61] Ak R_n správu nedostane, tak sa príslušný obvod zničí. To vedie k výraznej nespoľahlivosti R_1.
[62] DoS - Denial of Service - Odmietnutie služby.

Záver

V našej práci sme poskytli aktuálny prehľad anonymných sietí. Podľa zložitosti ich postupne popisujeme a hovoríme o ich predpokladoch a nedostatkoch. V prvej časti práce najprv zavádzame terminológiu, ktorá formalizmom podkladá naše intuitívne pochopenie. Ďalej detailne popisujeme protokol Večerajúcich kryptológov, ktorý zabezpečuje perfektnú anonymitu za predpokladu čestných účastníkov. Pokračujeme centralizovanými sieťami, ktoré sa snažia pomocou jedného zariadenia anonymizovať odosielateľa. Hlavnou témou prvej časti sú mixové siete, ktoré sú rozšírené pri komunikáciách s vysokou odozvou. Prvú časť ukončujeme zoznamom ďalších anonymných sietí a zoznamom štandardných útokov, ktoré stručne popisujeme. V druhej časti práce sa zaoberáme Onion routingom a jeho súčasnou implementáciou Tor. V tejto časti sa snažíme detailne popísať všetky podstatné súčasti Toru. Na konci druhej časti uvádzame zoznam potenciálnych útokov voči Toru a stručne uvádzame výsledky prác, ktoré sa nimi zaoberali.

Hlavným prínosom práce je prehľad jednotlivých anonymných sietí a detailné popísanie niektorých z nich. Práca je prvým prehľadom anonymných sietí v slovenčine, ktorá poskytuje čitateľovi postupný úvod do problematiky a zavádza potrebné pojmy v slovenčine. Práca sa snažila zvoliť vhodnú mieru medzi čitateľnosťou a formalizmom a to hlavne pri opisovaní Toru.

Zoznam použitej literatúry

[1] A. Back, I. Goldberg, and A. Shostack. Freedom Systems 2.1 Security Issues and Analysis. White paper, Zero Knowledge Systems, Inc., May 2001.

[2] K. Bauer, D. McCoy, D. Grunwald, T. Kohno, and D. Sicker. Low-resource routing attacks against Tor. In *Proceedings of the 2007 ACM workshop on Privacy in electronic society*, WPES '07, pages 11–20, New York, NY, USA, 2007. ACM.

[3] M. Blum, P. Feldman, and S. Micali. Non-interactive zero-knowledge and its applications. In *Proceedings of the twentieth annual ACM symposium on Theory of computing*, STOC '88, pages 103–112, New York, NY, USA, 1988. ACM.

[4] N. Borisov, G. Danezis, P. Mittal, and P. Tabriz. Denial of service or denial of security? How attacks on reliability can compromise anonymity. In R. Wright and P. Syverson, editors, *ACM Conference on Computer and Communications Security*, pages 92–102, New York, NY, USA, 2007. ACM.

[5] J. Boyan. The Anonymizer: Protecting User Privacy on the Web. *Computer-Mediated Communication Magazine*, 4(9), September 1997.

[6] D. Chaum. The dining cryptographers problem: unconditional sender and recipient untraceability. *J. Cryptol.*, 1(1):65–75, March 1988.

[7] D. L. Chaum. Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms. *Communications of the ACM*, 24(2):84–90, February 1981.

[8] T. Chothia. Analysing the MUTE Anonymous File-Sharing System Using the Pi-Calculus. In E. Najm, J.-F. Pradat-Peyre, and V. Donzeau-Gouge, editors, *Formal Techniques for Networked and Distributed Systems - FORTE 2006*, volume 4229 of *Lecture Notes in Computer Science*, pages 115–130. Springer Berlin / Heidelberg, 2006. 10.1007/11888116_9.

[9] I. Clarke, O. Sandberg, B. Wiley, and T. Hong. Freenet: A Distributed Anonymous Information Storage and Retrieval System. In H. Federrath, editor, *Designing Privacy Enhancing Technologies*, volume 2009 of *Lecture Notes in Computer Science*, pages 46–66. Springer Berlin / Heidelberg, 2001. 10.1007/3-540-44702-4_4.

[10] G. Danezis. The Traffic Analysis of Continuous-Time Mixes. In D. Martin and A. Serjantov, editors, *Privacy Enhancing Technologies*, volume 3424 of *Lecture Notes in Computer Science*, pages 742–746. Springer Berlin / Heidelberg, 2005. 10.1007/11423409_3.

[11] G. Danezis and C. Diaz. A Survey of Anonymous Communication Channels. Technical Report MSR-TR-2008-35, Microsoft Research, January 2008.

[12] G. Danezis, C. Diaz, and P. F. Syverson. Systems for Anonymous Communication. In B. Rosenberg and D. Stinson, editors, *CRC Handbook of Financial Cryptography and Security*, CRC Cryptography and Network Security Series, pages 341–390. Chapman & Hall, 2010.

[13] G. Danezis, C. Diaz, C. Troncoso, and B. Laurie. An Architecture for Anonymous Low-Volume Communications. In M. Atallah and N. Hopper, editors, *Privacy Enhancing Technologies*, volume 6205 of *Lecture Notes in Computer Science*, pages 202–219. Springer Berlin / Heidelberg, 2010. 10.1007/978-3-642-14527-8_12.

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

[15] G. Danezis, R. Dingledine, and N. Mathewson. Type III (Mixminion) Mix Protocol Specification, April 2007.

[16] G. Danezis and L. Sassaman. Heartbeat traffic to counter (n-1) attacks: red-green-black mixes. In *Proceedings of the 2003 ACM workshop on Privacy in the electronic society*, WPES '03, pages 89–93, New York, NY, USA, 2003. ACM.

[17] G. Danezis and P. Syverson. Bridging and Fingerprinting: Epistemic Attacks on Route Selection. In N. Borisov and I. Goldberg, editors, *Privacy Enhancing Technologies*, volume 5134 of *Lecture Notes in Computer Science*, pages 151–166. Springer Berlin / Heidelberg, 2008. 10.1007/978-3-540-70630-4_10.

[18] T. Dierks and C. Allen. The TLS Protocol Version 1.0 (RFC 2246). IETF Request For Comments, January 1999.

[19] W. Diffie and M. E. Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, November 1976.

[20] R. Dingledine, M. Freedman, D. Hopwood, and D. Molnar. A Reputation System to Increase MIX-Net Reliability. In I. Moskowitz, editor, *Information Hiding*, volume 2137 of *Lecture Notes in Computer Science*, pages 126–141. Springer Berlin / Heidelberg, 2001. 10.1007/3-540-45496-9_10.

[21] R. Dingledine, M. J. Freedman, and D. Molnar. The Free Haven Project: Distributed Anonymous Storage Service. In H. Federrath, editor, *Proceedings of Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability*, Lecture Notes in Computer Science. Springer Berlin / Heidelberg, July 2000.

[22] R. Dingledine and N. Mathewson. Design of a blocking-resistant anonymity system. Technical Report 2006-1, The Tor Project, November 2006.

[23] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The Second-Generation Onion Router. In *Proceedings of the 13th USENIX Security Symposium*, pages 303–320, San Diego, CA, USA, August 2004.

[24] R. Dingledine, V. Shmatikov, and P. Syverson. Synchronous Batching: From Cascades to Free Routes. In D. Martin and A. Serjantov, editors, *Privacy Enhancing Technologies*, volume 3424 of *Lecture Notes in Computer Science*, pages 771–780. Springer Berlin / Heidelberg, 2005. 10.1007/11423409_12.

[25] R. Dingledine and P. Syverson. Reliable MIX cascade networks through reputation. In *Proceedings of the 6th international conference on Financial cryptography*, FC'02, pages 253–268, Berlin, Heidelberg, 2003. Springer-Verlag.

[26] M. Dorigo, A. Colorni, and V. Maniezzo. Positive feedback as a search strategy. Technical Report 91-016, Dipartimento di Elettronica, Politecnico di Milano, Milan, Italy, 1991.

[27] C. Díaz and B. Preneel. Reasoning about the anonymity provided by pool mixes that generate dummy traffic. In J. Fridrich, editor, *Information Hiding*, volume

3200 of *Lecture Notes in Computer Science*, pages 309–325. Springer Berlin Heidelberg, 2004.

[28] P. Eckersley. How unique is your web browser? In *Proceedings of the 10th Privacy Enhancing Technologies Symposium*, pages 1–18, Berlin, Germany, July 2010.

[29] M. Edman and B. Yener. On anonymity in an electronic society: A survey of anonymous communication systems. *ACM Comput. Surv.*, 42(1):5:1–5:35, December 2009.

[30] N. S. Evans, R. Dingledine, and C. Grothoff. A practical congestion attack on tor using long paths. In *Proceedings of the 18th conference on USENIX security symposium*, SSYM'09, pages 33–50, Berkeley, CA, USA, 2009. USENIX Association.

[31] M. J. Freedman and R. Morris. Tarzan: a peer-to-peer anonymizing network layer. In *Proceedings of the 9th ACM conference on Computer and communications security*, CCS '02, pages 193–206, New York, NY, USA, 2002. ACM.

[32] I. A. Goldberg. *A pseudonymous communications infrastructure for the internet.* PhD thesis, University of California, Berkeley, 2000. AAI3001848.

[33] D. Goldschlag, M. Reed, and P. Syverson. Hiding Routing information. In R. Anderson, editor, *Information Hiding*, volume 1174 of *Lecture Notes in Computer Science*, pages 137–150. Springer Berlin / Heidelberg, 1996. 10.1007/3-540-61996-8_37.

[34] C. Gulcu and G. Tsudik. Mixing Email with Babel. In *Proceedings of the 1996 Symposium on Network and Distributed System Security (SNDSS '96)*, SNDSS '96, pages 2–, Washington, DC, USA, 1996. IEEE Computer Society.

[35] M. Hearst, S. Dumais, E. Osman, J. Platt, and B. Scholkopf. Support vector machines. *Intelligent Systems and their Applications, IEEE*, 13(4):18 –28, August 1998.

[36] S. Helmers. A Brief History of anon.penet.fi - The Legendary Anonymous Remailer. *Computer-Mediated Communication Magazine*, September 1997.

[37] A. Hintz. Fingerprinting Websites Using Traffic Analysis. In R. Dingledine and P. Syverson, editors, *Privacy Enhancing Technologies*, volume 2482 of *Lecture Notes in Computer Science*, pages 229–233. Springer Berlin / Heidelberg, 2003. 10.1007/3-540-36467-6_13.

[38] N. Hopper, E. Y. Vasserman, and E. Chan-Tin. How Much Anonymity does Network Latency Leak? *ACM Transactions on Information and System Security*, 13(2), February 2010.

[39] D. Kelly. *A taxonomy for and analysis of anonymous communications networks*. PhD thesis, Air Force Institute of Technology, March 2009.

[40] D. Kesdogan, D. Agrawal, and S. Penz. Limits of Anonymity in Open Environments. In F. Petitcolas, editor, *Information Hiding*, volume 2578 of *Lecture Notes in Computer Science*, pages 53–69. Springer Berlin / Heidelberg, 2003. 10.1007/3-540-36415-3_4.

[41] D. Kesdogan, J. Egner, and R. Büschkes. Stop- and- Go-MIXes Providing Probabilistic Anonymity in an Open System. In *Information Hiding*, volume 1525 of *Lecture Notes in Computer Science*, pages 83–98. Springer Berlin / Heidelberg, 1998. 10.1007/3-540-49380-8_7.

[42] M. Leech, M. Ganis, Y. Lee, R. Kuris, D. Koblas, and L. Jones. SOCKS Protocol Version 5. RFC 1928 (Proposed Standard), March 1996.

[43] B. Levine, M. Reiter, C. Wang, and M. Wright. Timing Attacks in Low-Latency Mix Systems. In A. Juels, editor, *Financial Cryptography*, volume 3110 of *Lecture Notes in Computer Science*, pages 251–265. Springer Berlin / Heidelberg, 2004. 10.1007/978-3-540-27809-2_25.

[44] M. Liberatore and B. N. Levine. Inferring the Source of Encrypted HTTP Connections. In *Proceedings of the 13th ACM conference on Computer and Communications Security (CCS 2006)*, pages 255–263, October 2006.

[45] D. Martin and A. Schulman. Deanonymizing Users of the SafeWeb Anonymizing Service. In *Proceedings of the 11th USENIX Security Symposium*, pages 123–137, Berkeley, CA, USA, 2002. USENIX Association.

[46] D. McCoy, K. Bauer, D. Grunwald, T. Kohno, and D. Sicker. Shining Light in Dark Places: Understanding the Tor Network. In N. Borisov and I. Goldberg, editors, *Proceedings of the Eighth International Symposium on Privacy Enhancing Technologies (PETS 2008)*, pages 63–76, Leuven, Belgium, July 2008. Springer.

[47] C. Meadows. The NRL Protocol Analyzer: An Overview. *The Journal of Logic Programming*, 26(2):113 – 131, 1996.

[48] U. Möller, L. Cottrell, P. Palfrader, and L. Sassaman. Mixmaster Protocol—Version 2. IETF Internet Draft, July 2003.

[49] S. J. Murdoch and G. Danezis. Low-Cost Traffic Analysis of Tor. In *SP '05: Proceedings of the 2005 IEEE Symposium on Security and Privacy*, pages 183–195, Washington, DC, USA, 2005. IEEE Computer Society.

[50] B. Möller. Provably Secure Public-Key Encryption for Length-Preserving Chaumian Mixes. In M. Joye, editor, *Topics in Cryptology — CT-RSA 2003*, volume 2612 of *Lecture Notes in Computer Science*, pages 244–262. Springer Berlin / Heidelberg, 2003. 10.1007/3-540-36563-X_17.

[51] National. FIPS 180-1. Secure Hash Standard. Technical report, US Department of Commerce, April 1995.

[52] N. I. of Standards and Technology. Advanced Encryption Standard. *NIST FIPS PUB 197*, 2001.

[53] Anonymizer. http://www.anonymizer.com/.

[54] Privoxy. http://www.privoxy.org/.

[55] Tor project. https://www.torproject.org/.

[56] L. O'Connor. On Blending Attacks for Mixes with Memory. In M. Barni, J. Herrera-Joancomartí, S. Katzenbeisser, and F. Pérez-González, editors, *Information Hiding*, volume 3727 of *Lecture Notes in Computer Science*, pages 39–52. Springer Berlin / Heidelberg, 2005. 10.1007/11558859_4.

[57] A. Panchenko, L. Niessen, A. Zinnen, and T. Engel. Website Fingerprinting in Onion Routing Based Anonymization Networks. In *Proceedings of the Workshop on Privacy in the Electronic Society (WPES 2011)*. ACM, October 2011.

[58] V. Pappas, E. Athanasopoulos, S. Ioannidis, and E. Markatos. Compromising Anonymity Using Packet Spinning. In T.-C. Wu, C.-L. Lei, V. Rijmen, and D.-T. Lee, editors, *Information Security*, volume 5222 of *Lecture Notes in Computer Science*, pages 161–174. Springer Berlin / Heidelberg, 2008. 10.1007/978-3-540-85886-7_11.

[59] A. Pfitzmann and M. Köhntopp. Anonymity, Unobservability, and Pseudonymity — A Proposal for Terminology. In H. Federrath, editor, *Designing Privacy Enhancing Technologies*, volume 2009 of *Lecture Notes in Computer Science*, pages 1–9. Springer Berlin / Heidelberg, 2001. 10.1007/3-540-44702-4_1.

[60] S. Prusty, M. Liberatore, and B. N. Levine. Forensic Investigation of the OneSwarm Anonymous Filesharing System. In *Proceedings of the 18th ACM conference on Computer and Communications Security (CCS 2011)*, October 2011.

[61] M. Reed, P. Syverson, and D. Goldschlag. Proxies for anonymous routing. In *Computer Security Applications Conference, 1996., 12th Annual*, pages 95 –104, December 1996.

[62] M. K. Reiter and A. D. Rubin. Crowds: anonymity for Web transactions. *ACM Trans. Inf. Syst. Secur.*, 1(1):66–92, November 1998.

[63] M. K. Reiter and X. Wang. Fragile mixing. In *Proceedings of the 11th ACM conference on Computer and communications security*, CCS '04, pages 227–235, New York, NY, USA, 2004. ACM.

[64] M. Rennhard and B. Plattner. Introducing MorphMix: peer-to-peer based anonymous Internet usage with collusion detection. In *Proceedings of the 2002 ACM workshop on Privacy in the Electronic Society*, WPES '02, pages 91–102, New York, NY, USA, 2002. ACM.

[65] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, February 1978.

[66] K. Sampigethaya and R. Poovendran. A Survey on Mix Networks and Their Secure Applications. *Proceedings of the IEEE*, 94(12), December 2006.

[67] A. Serjantov and G. Danezis. Towards an information theoretic metric for anonymity. In *Proceedings of the 2nd international conference on Privacy enhancing technologies*, PET'02, pages 41–53, Berlin, Heidelberg, 2003. Springer-Verlag.

[68] A. Serjantov, R. Dingledine, and P. Syverson. From a Trickle to a Flood: Active Attacks on Several Mix Types. In F. Petitcolas, editor, *Information Hiding*, volume 2578 of *Lecture Notes in Computer Science*, pages 36–52. Springer Berlin / Heidelberg, 2003. 10.1007/3-540-36415-3_3.

[69] C. Shannon. A mathematical theory of communication. Technical Report 27 :379, Bell System technical journal, 1948.

[70] Y. Sovran, A. Libonati, and J. Li. Pass it on: social networks stymie censors. In *Proceedings of the 7th international conference on Peer-to-peer systems*, IPTPS'08, pages 3–3, Berkeley, CA, USA, 2008. USENIX Association.

[71] P. Syverson, M. Reed, and D. Goldschlag. Onion routing access configurations. In *DARPA Information Survivability Conference and Exposition, 2000. DISCEX '00. Proceedings*, volume 1, pages 34 –40 vol.1, 2000.

[72] P. Syverson, G. Tsudik, M. Reed, and C. Landwehr. Towards an Analysis of Onion Routing Security. In H. Federrath, editor, *Designing Privacy Enhancing Technologies*, volume 2009 of *Lecture Notes in Computer Science*, pages 96–114. Springer Berlin / Heidelberg, 2001. 10.1007/3-540-44702-4_6.

[73] P. F. Syverson, D. M. Goldschlag, and M. G. Reed. Anonymous Connections and Onion Routing. In *Proceedings of the 1997 IEEE Symposium on Security and Privacy*, SP '97, pages 44–, Washington, DC, USA, 1997. IEEE Computer Society.

[74] M. Waidner and B. Pfitzmann. The Dining Cryptographers in the Disco: Unconditional Sender and Recipient Untraceability with Computationally Secure Serviceability. In J.-J. Quisquater and J. Vandewalle, editors, *Advances in Cryptology — EUROCRYPT '89*, volume 434 of *Lecture Notes in Computer Science*, pages 690–690. Springer Berlin Heidelberg, 1990.

[75] M. Waldman, A. D. Rubin, and L. F. Cranor. Publius: a robust, tamper-evident, censorship-resistant web publishing system. In *Proceedings of the 9th conference on USENIX Security Symposium - Volume 9*, SSYM'00, pages 5–5, Berkeley, CA, USA, 2000. USENIX Association.

[76] T. Wang, K. Bauer, C. Forero, and I. Goldberg. Congestion-aware Path Selection for Tor. In *Proceedings of Financial Cryptography and Data Security (FC'12)*, February 2012.

[77] E. Wustrow, S. Wolchok, I. Goldberg, and J. A. Halderman. Telex: Anticensorship in the Network Infrastructure. In *Proceedings of the 20th USENIX Security Symposium*, August 2011.

[78] B. Zantout and R. A. Haraty. I2P Data Communication System. In *ICN 2011 : The Tenth International Conference on Networks*, 2011.