Anonymizační síť Tor

Technická zpráva FIT VUT v Brně

Zdeněk Coufal,Libor Polčák

Technická zpráva č. FIT-TR-2014-02
Fakulta informačních technologií, Vysoké učení technické v Brně

Last modified: 27. května 2014

Anonymizační síť Tor

Zdeněk Coufal, Libor Polčák

Vysoké učení technické v Brně, email:
xcoufa06@stud.fit.vutbr.cz,ipolcak@fit.vutbr.cz

Abstrakt Metadata obsažená v hlavičkách síťových protokolů do značné míry prozrazují identitu komunikujících stran. Někteří uživatelé však potřebují, či chtějí komunikovat anonymně. Síť Tor si klade za cíl takovou komunikaci umožnit. Tato technická zpráva popisuje fungování sítě Tor po technické stránce.

1 Úvod

Běžná komunikace na Internetu není anonymní, protože data vyměňovaná v rámci Internetu mohou být spojena s identitou odesilatele, či příjemce za pomoci metadat. V rámci těchto metadat jsou dostupné síťové identifikátory, jako je IP adresa, které mohou být využity v procesu odkrytí identity některé z komunikujících stran.

V některých případech však může být vhodné komunikovat anonymně [24], či pod pseudonymem [24]. Např. pokud potřebujeme utajit svou identitu před administrátorem stránek, které se chystáme navštívit. Síť Tor [4] poskytuje anonymitu v prostředí Internetu. Bohužel je tato síť zneužívaná i pro páchání trestné činnosti [7].

Tato technická zpráva se zabývá především popisem sítě Tor a principy jejího fungování. Tato technická zpráva je tedy zaměřena na popis sítě Tor především z technické stránky.

Sekce 2 se zabývá teoretickou stránkou anonymity a vysvětluje důležité pojmy z oblasti ukrývání identity. Sekce 3 se zabývá předchůdci sítě Tor, protože značně ovlivnili její architekturu. Samotnou síti Tor a jejími základní principy se zabývá sekce 4. Na ní navazuje sekce 5, ve které je popsáno navazování spojení a správa virtuálních okruhů. Sekce 6 poskytuje seznam nástrojů, které jsou využívány ve spojitosti se síti Tor. Technická zpráva je shrnuta v sekci 7.

2 Anonymita ve veřejných sítích

Uživatelé na Internetu realizují denně nespočet transakcí různých forem. Některé z těchto transakcí jsou komunikativního charakteru např. zaslání emailu, přečtení novinek, chat. Další mohou být komerčního charakteru např. nákupy, prodeje. Obdobných tříd bychom mohli nalézt více včetně transakcí vojenských, politických, či ilegálních.

V každém případě účastníci mezi sebou vyměňují nějaký obsah, v případě komunikativních transakcí jsou to informace a v případě komerčních transakcí jsou to hodnoty. Jenže tyto transakce zahrnují mimo jiné výměnu (mezi uživateli) nebo odhalení (někomu zvenčí) meta-informací týkajících se účastníků nebo prováděných transakcí. Těmito meta-informacemi může být čas transakce, hodnoty vzájemně vyměněných položek, fyzická lokace nebo informace o identitách účastníků.

2.1 Definice problému anonymizace

Goldberg definoval [14] základní pojmy v oblasti anonymizace.

- **Soukromí**: Schopnost uživatele řídit šíření informací o něm samotném.
- **Anonymita**: Forma soukromí identity. Systém, který poskytuje anonymitu, je takový systém, který umožní uživateli určit, kdo se dozví jeho pravou identitu (pravé jméno). Znamená to, že systém automaticky nevkládá informace o identitě uživatele do hlaviček protokolů. Pro útočníka není jednoduché, nebo může být nemožné, aby prolomil takový systém a odhalil pravou identitu uživatele.
- **Pseudonymita**: Další forma soukromí identity. Uživatel udržuje jeden nebo více pseudonymů, což jsou zosobnění, která jsou spojena s jeho fyzickou identitou. Druhá strana komunikace si může být jista, že se pod pseudonymem skrývá jedna a ta samá osoba, ale skrývá fyzickou identitu. Tím se odlišuje od anonymity, kde žádný takový perzistentní identifikátor neexistuje a nelze říci, zda transakci provedla jedna a ta samá osoba.
- **Dopředné utajení** (*Forward Secrecy*): Neschopnost odhalit bezpečnostně kritické informace jako je identita odesílatele zprávy potom, co byla zpráva odeslána. Systém poskytující anonymitu se snaží této vlastnosti dosáhnout např. tím, že si neudržuje žádné záznamy o provedených transakcích.
- **Spojitelnost** (*Linkability*): Schopnost dát si do souvislosti transakce s identitou účastníků.

Pfitzmann a Hansen [24] pak definovali anonymitu, nespojitelnost, nedetekovatelnost, nepozorovatelnost, pseudonymitu a správu identit na základě dříve vydaných článku v této oblast. Výhodou této technické zprávy [24] je, že obsahuje překlad terminologie do českého jazyka.

2.2 Anonymizující techniky

Li, et al. definovali [20] popis a dělení anonymizujících technik. Protože účastníci využívají ke komunikaci na Internetu převážně pseudonymy, nezaručuje jim to zcela anonymitu, protože je lze stále dohledat prostřednictvím jejich IP adresy.

V sítích postavených na protokolu IP (Internet Protocol) obsahují pakety zdrojovou a cílovou IP adresu komunikujících uzlů, a proto na této úrovni nemohou být žádné datové pakety ve skutečnosti anonymní. Nicméně záměnou

původní zdrojové adresy za falešnou lze anonymní komunikace dosáhnout, protože není jednoduché zpětně vysledovat původní zdroj komunikace.

Anonymizující techniky poskytují Internetovým uživatelům určitou úroveň soukromí tím, že zabraňují sběru identifikačních informací jako je jejich IP adresa a další specifické položky protokolů (většinou aplikačních), které by mohly odhalit uživatelovu identitu nebo lokaci. Tyto techniky si kladou za cíl utajit identitu zdroje komunikace, případně i cíle komunikace před vnějším pozorovatelem nebo nebo i mezi účastníky komunikace.

Anonymizující systémy mohou být kategorizovány podle jejich latence, úrovně důvěry, typu sítě, anonymizujících vlastností a možností případného útočníka. Z hlediska zpoždění doručení zpráv jsou anonymizující systémy klasifikovány do dvou kategorií: *systémy s vysokým zpožděním doručení dat*, které poskytují vysokou úroveň anonymity a *systémy s nízkým zpožděním doručení dat*, kde je vysoká míra interakce uživatele s webovou aplikací a rychlá odezva je prioritou, většinou používané při anonymizovaném surfování na webu. Strategie přeposílání zpráv na uzlech je podmíněna nízkým zpožděním doručení dat, a proto tyto systémy volí kompromis mezi dosaženou úrovní anonymity a poskytovaným zpožděním doručení dat.

Z hlediska složitosti a úrovně poskytované anonymity se dělí anonymizující systémy na jednoduché anonymizující proxy servery a distribuované sítě. V případě anonymizujících proxy serverů se jeden konkrétní server stará o zajištění anonymity. V takovém případě uživatel musí plně důvěřovat jedné entitě. Prozrazení informací touto entitou pak potenciálně mohou odhalit identitu všech anonymních transakcí. U distribuovaných anonymizujících systémů na poskytnutí vysoké úrovně anonymity spolupracuje více uzlů. Důvěra uživatele je rozložena na všech těchto spolupracujících uzlech. Potenciální kompromitace jednoho z těchto uzlů neznamená deanonymizování všech transakcí, ale na druhou stranu může být složitější uhlídat chování správců všech uzlů.

3 Počátek anonymizujících technik

V této sekci jsou popsáni předchůdci sítě Tor, protože Tor přejímá významnou část jejich funkcionality.

3.1 Systémy s velkým zpožděním doručení dat

První metodou, která umožňovala skrytí identity komunikujících stran, byl MixNet [6]. Při e-mailové komunikaci umožňuje skrýt identitu obou nebo jedné komunikující strany.

Základní myšlenkou je využití asymetrické kryptografie ve spojení se sítí proxy serverů tzv. *mixů*, které slouží pro přeposílání zpráv mezi účastníky komunikace. Klient vybírá určitý počet mixů, které tvoří *cestu*, kterou jsou zprávy přeposílány. Každý mix vlastní pár soukromý a veřejný klíč.

Mix je *store-and-forward* zařízení, které přijímá určitý počet zpráv stejné velikosti od různých zdrojů, provede kryptografické transformace na přijatých

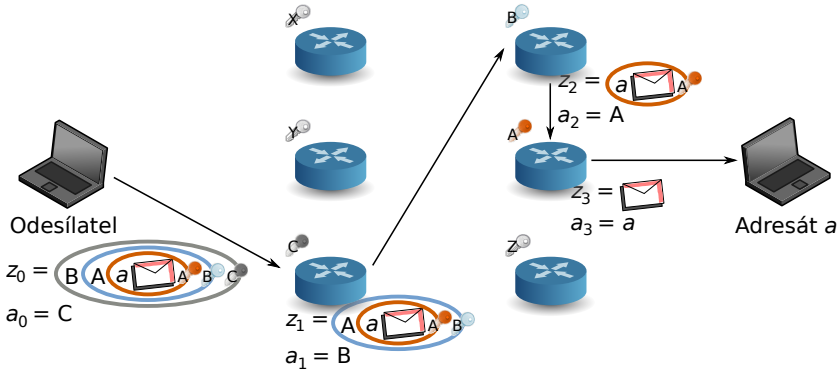
zprávách a následně zprávy přeposílá směrem k další destinaci v nepředpovídatelném pořadí a čase. Korelace zpráv na vstupu a výstupu jediného mixu je obtížná, protože mix zprávy přeposílá dávkově, v jiném pořadí než byly doručeny a s předem neurčeným zpožděním. Tím, že zpráva projde více mixy, se náročnost korelace ještě řádově zvyšuje. A není jednoduché určit, kdo s kým komunikuje.

Takový princip přeposílání má vysoké zpoždění doručení zpráv, které může dosahovat několika hodin, nebo dokonce i dnů [29].

Doručení zprávy Z skrze anonymizující síť probíhá následovně:

1. Odesílatel vytvoří cestu $c = (m_1, m_2 \dots m_k, a)$, kde k je zvolená délka cesty anonymizující sítí, $m_1 \dots m_k$ jsou náhodně zvolené *mixy* a a je adresát zprávy.
2. Odesílatel ze zprávy Z postupně vytváří zprávu Z_0 určenou k odeslání tak, že inicializuje zprávu Z_k, kterou bude odesílat *mix* m_k na $Z_k = Z$ a adresáta a_k zprávy Z_k nastaví na $a_k = a$. Odesílatel pro $i \in \{0, \dots, k-1\}$ vytvoří zprávu Z_i (postupuje postupně od $i = k-1$ po $i = 0$) tak, že zašifruje Z_{i+1} a a_{i+1} veřejným klíčem *mixu* m_{i+1} a nastaví $a_i = m_{i+1}$. Výsledná odesílaná zpráva Z_0 je tedy nakonec tvořena několika vrstvami šifrovaných zpráv.
3. Odesílatel odešle a_0 zprávu Z_0.
4. Zpráva je doručena přes specifikované *mixy* m_1 až m_k tak, že každý *mix* m_i na cestě od odesílatele k příjemci z obdržené zprávy Z_{i-1} rozšifruje Z_i a a_i. Zprávu Z_i ale nepřepošle a_i okamžitě, ale s odesláním počká předem neznámou dobu. Během této doby jsou *mixu* m_i doručeny další zprávy posílané skrze anonymizující síť. Pro zvýšení anonymity každý *mix* mění pořadí odesílaných zpráv oproti pořadí, ve kterém mu byly zprávy doručeny.

Tím, že každý z *mixů* odstraňuje jednu vrstvu šifrování a zároveň odesílá zprávy v jiném pořadí, než ve kterém mu byly doručeny, se znemožňuje možná korelace zpráv při odposlechu komunikace v rámci anonymizující sítě a na jejich okrajích. Postup zasílání e-mailové zprávy anonymizující sítí je zachycen na obrázku 1.



Obrázek 1. Zasílání e-mailových zpráv anonymizující síti Mix-Net

Mixy přijímají zprávy od více klientů, zamíchají je a v určitých časech takto vytvořenou dávku společně s vycpávkovými zprávami přeposílají dále. Délka každého kola je určena algoritmem, který bere v úvahu prahovou hodnotu založenou na počtu přijatých zpráv ve vyrovnávací paměti nebo na časovači, či jejich kombinaci.

Popsaný způsob doručení zpráv v systému Mix-Net zaručuje, že kromě odesílatele a adresáta zná každý uzel na cestě jen předchozí a následující uzel v anonymizující síti. Přeposílající uzly tedy nemají možnost zjistit identitu komunikujících stran. Poslední uzel v anonymizující síti, ale může odhadnout, že je poslední a pozorovat obsah zprávy Z.

Předpokladem výše navrženého způsobu přeposílání je, že nesmí být možné určit vztah mezi množinou zašifrovaných zpráv a odpovídající množinou nezašifrovaných zpráv vstupujících respektive vystupujících z mixu. Z toho důvodu mixy odstraňují redundantní zprávy. Stejně tak nesmí být možné vytvářet padělky zpráv bez znalosti náhodného řetězce a soukromého klíče. Požadavek vychází z vlastností asymetrické kryptografie [1].

Pro anonymní doručení odpovědi systém Mix-Net umožňuje, aby odesílatel vytvořil strukturu popisující zpáteční cestu. Za využití asymetrické kryptografie je možné zajistit, že i na zpáteční cestě zná každý z přeposílajících uzlů pouze identitu předchozího a následujícího uzlu v anonymizující síti, ale nezná adresy ostatních uzlů ani obsah přenášené zprávy.

Použitím asymetrické kryptografie je zajištěno, že i na cestě zpět zná každý z přeposílajících uzlů pouze identitu svých sousedů v rámci cesty. Aby nedocházelo na mixu k redundancím (příležitost pro útočníka dát si do souvislosti vstup a výstup mixu), nesmí mixem projít dvakrát stejná zpáteční adresa [6]. Z toho důvodu odesílatel musí pro každou zprávu, na kterou očekává odpověď, vytvořit novou zpáteční adresu.

Systémy *Mixmaster* [22], *Babel* [16] a *Miximinion* [10] se snažily vylepšit řešení anonymní komunikace navržené pro Mix-Net. Goldberg et al. [13] shrnují stav anonymizujících technik pro e-mailovou komunikaci na konci 90. let 20. století. Mazières a Kaashoek [21] popsali zkušenosti s provozem anonymizující e-mailové služby včetně několika případů, kdy došlo ke zneužití pro kriminální účely. V literatuře se objevují i pokusy zabývající se možnostmi získání identity komunikujících stran používající anonymizující techniky. Pfitzmann a Pfitzmann [25] nalezli možnost prozrazení identity v řešení založeném na Mix-Netu. Další studii možných hrozeb zpracoval Cottrell [9].

3.2 Systémy s nízkým zpožděním doručení dat

Hlavní nevýhodou Mix-Netu a odvozených technik a služeb je velká latence. V některých případech (např. *anon.penet.fi*) trvalo několik dnů, než se zpráva

[1] Zná-li útočník kryptogram $K(X)$, což je zpráva X zašifrovaná veřejným klíčem K, pak si může dovolit odhadnout, zda $Y = X$ tím, že vyzkouší zda $K(Y) = K(X)$. Tuto hrozbu lze jednoduše eliminovat tak, že před vlastní zprávu X je přidán náhodný řetězec bitů R a kryptogram pak vypadá následovně $K(R, X)$.

dostala k adresátovi. Proto není možné uplatnit stejný postup na protokoly, u kterých záleží na poměrně rychlém doručení, např. při prohlížení WWW stránek. Postupně se začala objevovat nová řešení anonymizace, která berou v úvahu rychlost doručování.

Onion Routing [15,26,28,29] představuje obecné řešení pro anonymizaci spojení řady aplikačních protokolů včetně HTTP, FTP, SMTP, telnet aj. Onion Routing předpokládá existenci známé sítě anonymizujících stanic (směrovačů) s dostupnými veřejnými kryptografickými klíči. Zdroj komunikace nejdříve vytvoří virtuální okruh pomocí struktury *onion* (cibule), která specifikuje ve vrstvách cestu k cíli.

Směrovače Onion Routingu si mezi sebou udržují TCP/IP spojení a tím tvoří topologii *overlay* Onion Routing sítě. Informace o aktuálním stavu sítě (topologii) je propagována přes všechny aktivní směrovače a uživatelské proxy.

U Mix-Netu musel uživatel pro každou zasílanou zprávu vytvářet zvlášť cibuli a používat často operace asymetrické kryptografie. To bylo sice vhodné pro případ komunikace elektronickou poštou, ale u interaktivní aplikace by to znamenalo spoustu výpočetně náročných operací s veřejným klíčem.

V síti Onion Routing disponuje každý směrovač párem veřejný/soukromý klíč. Asymetrickou kryptografii (konkrétně algoritmus RSA) používá tato síť pouze pro zašifrování sdíleného klíče, kterým je zašifrován zbytek zprávy. Využitím symetrické kryptografie pro vlastní předávání zpráv snižuje nároky mechanismu na výpočetní zdroje.

Princip rezervace virtuálního okruhu vychází z postupu zasílání zpráv Mix-Netem. Jednotlivé směrovače na cestě vytvořenou cibuli loupou a rezervují dopřednou a zpětnou cestu pro vytvářený virtuální okruh. Kromě počátečního směrovače zná každý směrovač pouze identitu svých sousedů. Po vytvoření virtuálního okruhu může zdroj komunikace komunikovat s cílem pomocí brány proxy, kterou realizuje první směrovač v anonymizující síti. V rámci anonymizující sítě se posílají pakety pevné velikosti a veškerá komunikace je šifrovaná. Aby nebylo možné odhadnout, že v rámci sítě aktuálně proudí data, posílají jednotlivé směrovače vycpávkový provoz.

Autoři uznávají, že v případě monitorování vstupních i výstupních bran by bylo možné odhadnout, které stanice mezi sebou komunikují, ale pokud je anonymizující síť dostatečně velká, je těžké monitorovat všechny vstupní a výstupní brány.

V případě, kdy není potřeba přeposílat skutečná data uživatelů sítě, vyměňují si mezi sebou směrovače vycpávkový provoz, aby nemohl útočník pozorovat, zda sítí proudí data, či nikoliv. Nicméně protože nejsou ukládány zaslané zprávy jako u Mix-Netu, hrozí zde útok opakováním přenosu rezervační cibule[2].

[2] Útok typu *onion replay* mohl útočník využít k opětovnému vytvoření dříve existujícího virtuálního okruhu pomocí rezervační cibule, kterou dříve zachytil. Tím pádem získal přístup k použitým šifrovacím klíčům a algoritmům a mohl nechat dešifrovat dříve zachycenou komunikaci [30].

Onion Routing nebyl v praxi příliš úspěšný, reálně byl nasazen pouze jako *proof-of-concept*. Mnoho problémů ať již v návrhu nebo zjištěných při nasazení nebylo vyřešeno a samotný návrh nebyl již léta aktualizován.

4 Principy sítě Tor

Síť Tor vychází z původního konceptu sítě *Onion Routing*, avšak s řadou vylepšení a je označována jako druhá generace Onion Routing. Byla spuštěna v roce 2003 (skryté servery v roce 2004). Na rozdíl od Onion Routing není Tor zatížen žádnými patenty, jedná se o *open-source* projekt s rozsáhlou komunitou [3].

Návrh sítě popsali Dingledine et al. [12,11], z tohoto návrhu také budeme vycházet v dalším textu. Od uvedení sítě do praxe uběhla již řada let a proto tato práce upozorňuje i na další významné změny od nasazení zasahující do původního návrhu.

Detaily lze nalézt přímo ve specifikačních dokumentech protokolu Toru [5,2,1]. Projekt je neustále aktivně vyvíjen a v každé nové verzi se objeví řada více či méně důležitých změn, z toho důvodu nerozebírám v dalších pasážích vyloženě nízkoúrovňové postupy, pokud to pro práci není relevantní.

4.1 Architektura a komponenty

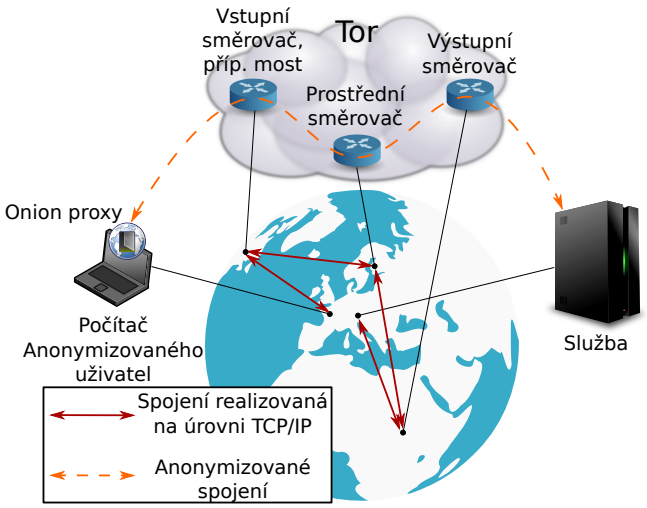
Z pohledu návrhu je síť Tor koncipována tak, že využívá již existující Internetovou TCP/IP infrastrukturu, jedná se tedy o tzv. *overlay* síť, která je pouze další logickou vrstvou nad již existující vrstvou.

Síť Tor je tvořena následujícími základními komponentami, jejich popis bude dále rozveden dále v textu.

- **Směrovače** (*onion routers* - OR): Tyto směrovače jsou základním stavebním kamenem celé sítě Tor, vytváří její topologii a především se jedná o uzly, kterými prochází virtuální okruhy. Podle pozice ve virtuálním okruhu se dělí na vstupní směrovače tzv. strážce (*guards*), prostřední směrovače (*middle*) a výstupní směrovače (*exits*).
- **Uživatelská proxy** (*onion proxy* - OP): Získávají aktuální data o topologii a stavu sítě z adresářových serverů, vytváří virtuální okruhy a řídí přenos datových toků mezi uživatelskými aplikacemi a sítí Tor. Typicky je spouštěna na počítači koncového uživatele.
- **Adresářové servery** (*directory servers*): Udržují globální pohled na topologii a stav jednotlivých uzlů sítě Tor.
- **Skryté servery** (*hidden services* - HS): Tor poskytuje tzv. skryté služby, což je integrovaný mechanismus, který umožňuje serverům anonymně poskytovat služby, bez toho aniž by byla prozrazena identita těchto serverů. K těmto serverů je přistupováno přes upravené doménové jméno v rámci sítě Tor.
- **Mosty** (*bridges*): Tyto uzly byly představeny až později [11] kvůli boji proti cenzuře. Jedná se o speciální vstupní směrovače, informace o nich není veřejně dostupná na adresářových serverech, tudíž je nelze hromadně blokovat. Uživatel si může vyžádat IP adresu těchto uzlů, ale přidělený počet je vázán na jeho vlastní IP adresu.

– **Uživatelé**: Využívají služeb sítě Tor, z hlediska přístupu k síti se dělí na přímé uživatele (přistupují přes veřejně dostupné OR) a cenzurované uživatele (přistupují přes mosty).
– **Maskování provozu** (*pluggable transports*): Transformují charakteristiky síťového provozu Toru [17], typicky mezi klientem a mosty. Mezi transformace se mohou řadit např. změna rozložení délek paketů, odlišná délka mezi paketových mezer, apod. Touto cestou mohou uživatelé obelstít pozorovatele provozu, kteří by se snažili detekovat provoz sítě Tor na základě jejich obvyklých charakteristik [17].

Obrázek 2 zachycuje typické použití sítě Tor. Uživatel sítě Tor nekontaktuje službu přímo. Namísto toho využije služeb sítě Tor pro vytvoření několika spojení napříč celým světem pro ukrytí skutečného cíle komunikace.



Obrázek 2. Základní funcionalita Toru. Data jsou zasilaná přes OR rozmístěné různě ve světě.

4.2 Významné inovace

– V síti síti Onion Routing první generace mohl jediný kompromitovaný uzel na cestě zaznamenávat komunikaci a později kompromitovat ostatní uzly okruhu a donutit je zaznamenanou komunikaci dešifrovat. Místo jedné mnohonásobně zašifrované datové struktury (cibule) příslušející každému okruhu využívá Tor **inkrementální (teleskopický)** přístup k budování cesty (virtuálního okruhu). Iniciátor komunikace se dohodne na klíčích sezení s každým následujícím uzlem okruhu, jakmile jsou tyto klíče jednou smazány,

následně kompromitované uzly již nemohou dešifrovat zaznamenanou komunikaci. Díky tomu již není dále potřeba detekovat útoky typu *onion replay*. Proces budování okruhů a komunikace jsou více spolehlivé, protože OP uživatele vyjednává postupně s každým uzlem okruhu zvlášť, klíče nejsou po zrušení okruhu dále dostupné a tím je zajištěno dopředné utajení (*forward secrecy*). Navíc v případě selhání libovolného uzlu, OP ví, který uzel selhal a může rozšířit okruh pomocí jiného uzlu.

- *Onion Routing* vyžadoval vlastní aplikační proxy pro každý podporovaný aplikační protokol. Tor používá standard *SOCKS*[18,19], což je proxy rozhraní, které podporuje většinu programů založených na TCP a to bez nutnosti dalších modifikací. Pro případy kdy není Tor schopen zaručit únik identifikujících informací např. DNS dotaz přes UDP, doporučují autoři používat Tor ve spojení s lokální proxy (např. *Privoxy*[3]).
- Tor **nevyužívá** přeuspořádání paketů, vycpávkový provoz ani rozprostření síťového provozu (*traffic shaping*). Ukázalo se, že to není ekonomicky ani prakticky příliš dobrý přístup, protože i plně zatížená linka pomocí vycpávkového provozu je stále zranitelná. Kvůli tomu se prozatím upustilo od těchto principů alespoň do té doby, než se objeví vhodný návrh pro *traffic shaping* nebo *nízkolatentní "mixování"*, které by mohly zvýšit bezpečnost a úroveň anonymity vůči realistickému útočníkovi.
- Více TCP spojení může sdílet jediný okruh. *Onion Routing* vytvářel separátní okruh pro každé nové aplikační spojení, mnohonásobné operace s veřejným klíčem pro každý požadavek ohrožovaly anonymitu při budování tak velkého počtu okruhů. Tor využívá **multiplexování** a umožňuje tak využít jeden virtuální okruh pro více TCP spojení. Uživatel má kontrolu nad tím, které datové toky mohou virtuální okruh sdílet.
- Za využití principu *leaky pipe* může OP uživatele využít jako výstupní OR jiný OR, než je poslední v okruhu. Toho je možné využít např. z důvodů odlišných výchozích politik, nebo pro zmatení útočníka, který provádí korelaci na začátku a konci okruhu.
- Dřívější anonymizační návrhy se nezabývaly nedostatečnou průchodností provozu (zahlcením sítě). Typické přístupy jako *load balancing* a řízení toku dat u *overlay* sítí vyžadují vzájemnou komunikaci mezi uzly takové sítě a globální pohled na síťový provoz. Tor má **decentralizované řízení zahlcení**, které využívá potvrzování mezi koncovými uzly, což zajišťuje zachování anonymity zatímco umožňuje uzlům na krajích sítě detekovat zahlcení a posílat méně dat, než zahlcení odezní.
- *Onion Routing* šířil informaci o stavu jednotlivých uzlů sítě a topologii sítě přeposíláním skrze celou síť. Tor využívá **důvěryhodné adresářové autoritativní servery**, které spolupracují při vytváření podepsaných adresářů obsahujících známé OR a jejich aktuální stav. OP uživatelů si adresáře periodicky stahují přímo z adresářových serverů pomocí protokolu HTTP tunelovaného přes virtuální okruh sítě Tor.

[3] Privoxy: `http://www.privoxy.org`

– Variabilní **politiky pro odchozí provoz** umožňují každému výstupnímu uzlu specifikovat, ke kterým uzlům a portům je ochoten se připojit. Tyto politiky jsou vítanou změnou pro dobrovolníky provozující výstupní uzly sítě, protože jim to umožňuje přesně definovat síťový provoz, který jsou ochotni přeposílat.
– *Onion Routing* neprováděl kontrolu integrity dat, uzel na cestě mohl pozměnit obsah zprávy (např. nastavit požadavek na spojení tak, že připojení proběhlo k jinému webovému serveru, nebo si označkoval zašifrovanou komunikaci a tu pak sledoval na okrajích sítě). Tomu Tor zabránil tím, že **verifikuje integritu dat**.
– Onion Routing k realizaci skrytých služeb používal odpovědní cibule, ale ty neumožňovaly dopředné utajení a stávaly se zbytečnými, pokud kterýkoliv uzel na cestě spadl nebo změnil klíč. Tor využívá tzv. *rendezvous* uzlů, na kterých se klienti dohodnou jako na bodu setkání, aby se dostali k serverům poskytujícím skryté služby.
– Tor je **odolný vůči cenzuře** a pokusům zablokovat uživatelům přístup do jeho sítě. Zavedl nový uzel typu most *bridge*, který není nabízen adresářovými servery a jeho IP adresu zná jen malá množina individuálních osob. Navíc byl Tor navržen tak, aby se jevil na síti podobně jako protokol HTTPS (včetně využití typického portu 443, nicméně může využívat i jiný) a blokování Toru znamená zablokovat i HTTPS. Protože některé země blokují protokol HTTPS nebo mohou cenzoři provádět hlubokou inspekci paketů (DPI), zavedl Tor obfuskační rozšíření **pluggable transports**.
– Tor zavedl tzv. *Tor control* protokol [1], který poskytuje programátorské API dovolující spolupráci OP s dalšími uživatelskými či systémovými komponentami a umožňuje tak realizovat širokou škálu požadavků uživatelů na různé aplikace. Např. grafické uživatelské rozhraní *Vidalia* je samostatná aplikace, která komunikuje s OP přes výše uvedený řídicí protokol.

4.3 Adresářové servery

V první generaci Onion Routing se aktualizace o stavu sítě šířila pomocí šíření zpráv napříč celou sítí. Anonymizující sítě mají nicméně jiné bezpečnostní cíle než typické *link-state* směrovací protokoly. Velký vliv zde má zpoždění, které by mohlo zapříčinit různé pohledy klientů na topologii sítě a taková situace by byla prospěšná pro útočníka.

Tor používá malé skupiny redundantních dobře známých OR, které sleduji změny v síťové topologii a stavy dalších OR, včetně klíčů, odchozích politik a další. Tyto vybrané OR se nazývají adresářové servery.

Adresářové servery se chovají jako HTTP servery, umožňují klientům stáhnout aktuální stav sítě a seznam OR. Malá množina důvěryhodných adresářových serverů jsou tzv. adresářové autority. OR periodicky zasílají podepsané prohlášení o svém stavu tzv. *deskriptory* všem těmto adresářovým autoritám. Všechny adresářové autority pak tyto informace kombinují s jejich vlastním pohledem na síť a periodicky generují podepsaný popis o stavu celé sítě, tzv. adresáře (*konsensus*). Tyto adresáře vznikají na základě hlasování adresářových

autorit a jsou jimi taktéž podepsány. Čím více autorit adresář podepíše, tím je pohled na síť věrohodnější.

Když obdrží adresářová autorita prohlášení daného OR, nenabízí ho okamžitě jako aktivní OR, ale nejdříve ho otestuje, zda správně reaguje na vytvoření přímého a anonymizovaného spojení. Počet OR, které mohou být identifikované jedinou IP adresou je omezen.

Adresářové servery nejsou spravovány centrálně, ale existuje více distribuovaných autorit. Koncový uživatelé pak použijí takový pohled na síť Tor, na kterém se shodne nejméně polovina z celkového počtu všech autorit, které uživatelova OP zná. Navíc adresářový dokument platí pouze po omezenou dobu.

Uživatelova OP je vybavena seznamem těchto adresářových autorit a jejich klíči. OP kontaktuje adresářovou autoritu pouze když potřebuje získat prvopočáteční pohled na síť. Aby se předešlo velkému zatížení těchto autorit, mohou být podepsané adresáře ukládány ve vyrovnávací paměti přímo na OR. OR si je pak ve vyrovnávací paměti periodicky aktualizují.

Později byly zavedeny [11] tzv. *mikro-deskriptory*, OP nepotřebuje znát tolik informací o OR jako adresářové servery, stačí mu znát pouze *onion* klíč a odchozí politiky pro každé OR. OP pak stahuje pouze adresář s těmito mikro-deskriptory.

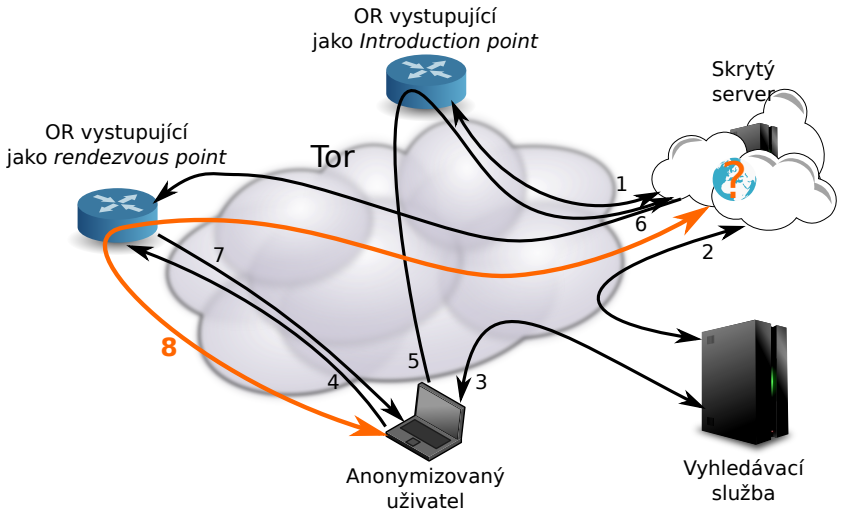
4.4 Správa klíčů a použité šifrování

Síť Tor musí zajistit 1) důvěrnost dat ve vlastní síti, 2) autentizaci OR vůči klientům a 3) koordinaci při distribuci informací o síti. K dosažení těchto cílů je potřeba kryptografických technik.

- **Šifrování**: Všechna spojení v síti Tor mezi jednotlivými OR používají protokol TLS. To znemožňuje pozorovatelům zjistit, ke kterému okruhu daná buňka patří. Dále klient zřizuje dočasný sdílený šifrovací klíč postupně s každým OR v okruhu a toto vrstvené zašifrování zajišťuje, že pouze výstupní OR může číst obsah buňky. Po zániku okruhu klient i OR tyto klíče zahazují. Zaznamenávání provozu a následné kompromitování OR pak ztrácí smysl.
- **Autentizace**: Ke každému OR je k dispozici jeho veřejný klíč zvaný *onion* klíč. OR rotují tyto klíče přibližně jednou za týden. Když klient zřizuje nový okruh, v každém kroku vyžaduje po OR znalost párového soukromého klíče a tím se OR klientovi autentizuje.
- **Koordinace**: Aby měli klienti jistotu, že komunikují se správnými OR pomocí správných klíčů, vlastní každý OR dlouhodobý veřejný klíč zvaný *identifikační klíč*. Tímto klíčem podepisuje certifikát, který specifikuje jeho další klíče, lokaci, typ uzlu a další informace o daném typu uzlu. Adresářové autority disponují adresářovým klíčem. Tyto adresářové autority poskytují podepsaný seznam všech známých OR. V tomto seznamu se nachází množina certifikátů příslušejících jednotlivým OR, každý podepsaný příslušným identifikačním klíčem.

4.5 Skryté služby

Skryté služby umožňují provozovat TCP službu jako je např. webový server tak, aniž by byla prozrazena IP adresa této služby. Dále v zavedené terminologii spolu komunikují OP **Alice** a **Boba**. Na obrázku 3 je znázorněn postup poskytnutí skryté služby.



Obrázek 3. Princip poskytnutí skrytých služeb. Všechna uvedená spojení na obrázku jsou anonymizovaná spojení Toru.

Poskytnutí skryté služby :

1. Skrytý server (HS) vygeneruje dlouhodobý pár soukromý a veřejný klíč za účelem identifikace jeho služby. Pak se připojí k vybranému OR a dotáže se ho, zda by mohl jménem HS vystupovat jako jeho *introduction point* (InP). Pokud OR souhlasí, HS s ním nadále udržuje otevřený okruh (dokud se daný uzel nerestartuje nebo se nerozhodne okruh ukončit). Pokud nesouhlasí, zkusí se dotázat jiného OR. Takto se dotáže více OR, aby získal požadovaný počet InP.
2. HS nabídne vybrané InP skrze vyhledávací službu, kterou poskytují adresářové servery. Předem nabídku podepíše svým soukromým klíčem. Od této chvíle může HS očekávat požadavky klientů.
3. Anonymizovaný uživatel (klient) se dotáže vyhledávací služby na adresářovém serveru na adresy InP pro jím identifikovanou službu a pokud služba existuje, tak je obdrží.

4. Klient zvolí nějaké OR jako prostředníka pro komunikaci s HS tzv. *rendezvous point* (RP). Ke zvolenému RP klient založí okruh a předá mu náhodně zvolené *rendezvous cookie* (RC).
5. Klient otevře okruh s jedním z InP dané HS a předá mu zprávu zašifrovanou veřejným klíčem HS, ve které se mu představí (zvolené RP, RC a první část dohody algoritmu *Diffie-Hellman* (D-H) [27] na klíči). InP přepošle zprávu HS skrze okruh otevřený v bodě 1.
6. Pokud má HS zájem s klientem komunikovat, založí okruh k jeho zvolenému RP a zpětně odešle (RC, druhou část dohody D-H na klíči a otisk klíče sezení, který nyní HS s klientem sdílí[4]). RP spojí okruh, který sdílí s HS a ten, který sdílí s klientem.
7. RP pošle klientovi potvrzení o úspěšném navázání spojení s HS.
8. Nyní může začít klient komunikovat s HS přes nově vytvořené anonymní spojení.

Z výše uvedeného postupu lze vyvodit :

- Klient nezná lokaci (IP adresu) HS, ale zná lokaci RP.
- HS nezná lokaci klienta, ale zná lokaci RP.
- RP nezná lokaci klienta ani HS, nezná ani poskytovanou službu a obsah přenášených dat.

Integrace HS v uživatelských aplikacích :

Uživatel nakonfiguruje jeho OP, aby znala lokální IP adresu a port jím nabízené služby, strategii pro autorizaci klientů a jeho veřejný klíč. OP anonymně publikuje podepsané údaje o jeho veřejném klíči, expiračním čase a aktuálních InP pro jeho službu ve vyhledávací službě (indexováno otiskem jeho veřejného klíče). Uživatelova nabízená služba zůstává nezměněna a ani neví, že je ukryta za sítí Tor.

Aplikace na straně klienta zůstává taktéž nezměněna (rozhraním zůstává SOCKS proxy). Všechny nezbytné informace jsou zakódovány do doménového jména, které klient využije při navazování spojení se službou. Skryté služby využívají virtuální nejvyšší úroveň doménového jména nazývaný **.onion**, doménové jméno má pak podobu *x.y.onion*, kde x je autorizační cookie a y zakódovaný otisk veřejného klíče. OP klienta prohlédne adresy a pokud jsou mířené na skrytý server, dekóduje klíč a začne proces navazování spojení se skrytým serverem.

4.6 Odchozí politiky

Provozovatelé OR mohou omezit provoz odesílaný jimi spravovanými OR pomocí tzv. odchozích politik. Tedy zda má OR přeposílat data jen mezi ostatními OR, nebo zda může být výstupním OR. Pro výstupní OR je pak možné určit, ke kterým síťovým uzlům a portům se mohou uživatelé sítě Tor přímo připojit.

Provozovatelé výstupních OR uzlů mohou být označeni za původce nelegálních aktivit. Je to z toho důvodu, že uživatel komunikující přes síť Tor, vystupuje

[4] Klient ví, že sdílí tento klíč pouze s HS.

pod IP adresou výstupního OR. Zavedením odchozích politik se takovému scénáři dá částečně zabránit.

- Otevřené výstupní OR (*open exits*) se připojí všude.
- Zprostředkovatelské OR pouze přeposílají provoz mezi jinými OR.
- Privátní (*private exits*) OR se připojí pouze k lokálnímu uzlu nebo dané síti. Umožňují bezpečnější připojení v tom smyslu, že externí útočník nemůže odposlouchávat provoz mezi privátním OR a jeho cílovou destinací.
- Nejčastější typ jsou omezené (*restricted exits*) OR, umožňují se připojit kdekoliv po světě, ale zabraňují přístupu k některým zneužití náchylným adresám a službám jako je např. SMTP.

Pro provozování výstupního uzlu je však doporučováno postupovat podle speciálního návodu [23]. Např. je doporučováno nemíchat provoz vystupující ze sítě Tor s vlastním provozem, provozovat webový server vysvětlující princip sítě Tor apod.

5 Správa virtuálních okruhů

Původní *Onion Routing* vytvářel zvlášť okruh pro každé spojení TCP, což bylo časově i výpočetně náročné (latence sítě, použití asymetrické kryptografie), a proto nepoužitelné pro běžné prohlížení webu (otevírá příliš mnoho TCP spojení). Tor sdílí okruh pro více spojení TCP a navíc, aby se předešlo prodlevám, uživatelské OP konstruují okruhy preemptivně (v předstihu).

Uživatel má možnost označení datových toků jako izolované. Do okruhu takového toku nepřidá OP nový datový tok. Novému spojení nebude přidělen ani okruh, který byl vytvořen déle než před 10-ti minutami (takový okruh bude zrušen ihned, jakmile budou uzavřeny všechny datové toky, které přenáší). OP periodicky (1 minuta) zvažuje rotaci na nový okruh (i nároční uživatelé tak stráví pouze zanedbatelný čas vytvářením okruhů).

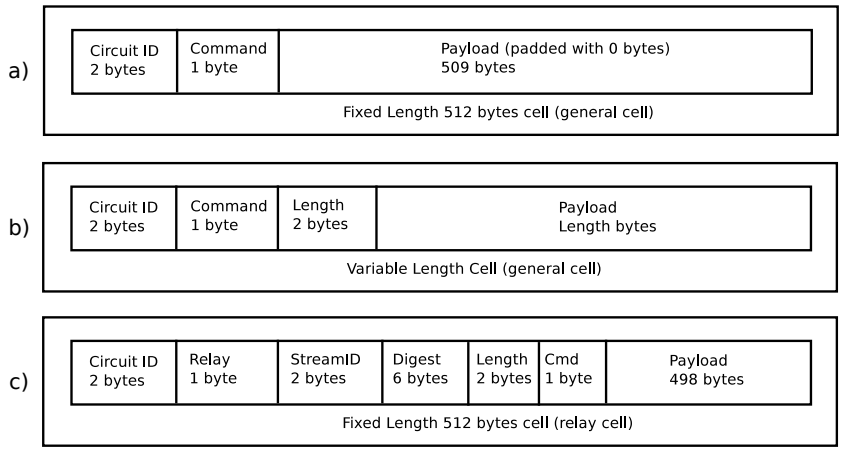
Separována (označena jako izolovaná) jsou ta spojení TCP, která byla iniciována různými uživateli nebo přišla ze spojení SOCKS [18,19] s různými autentizačními údaji, případně z jiného portu SOCKS.

Veškerá správa okruhů je v režii OP, pokud se okruh stane nepoužitelným, uživatel to nemusí vůbec zaznamenat.

5.1 Komunikace a typy zpráv

Provoz je přenášen v buňkách [12,17] o pevné velikosti 512B[5]. Každá buňka se skládá z hlavičky a užitečných dat, viz obrázek 4.

[5] Pevná velikost buněk je to neefektivní [11]. Proto byly později [11,8] představeny některé řídicí buňky, které mohou mít proměnlivou velikost. Navíc použití pevných velikostí u buněk vytváří unikátní rozložení výskytu velikosti paketů [17], což je přímo v rozporu s požadavkem na odolnost vůči klasifikaci síťového provozu.



Obrázek 4. Struktura buněk přenášených sítí Tor.

Hlavička obsahuje [11,8] unikátní identifikátor okruhu, ke kterému buňka náleží *CircID*. Pro každou část okruhu mezi OR je hodnota *CircID* jiná[6]. Dále obsahuje hlavička příkaz (*commands*), který říká, co se má provést s užitečnými daty. V závislosti na tomto poli jsou buňky děleny na řídicí (*control*) a přenosové (*relay* a *relay_ early*)[7].

Přenosové buňky mají v hlavičce navíc *StreamID* určující přenášený datový tok TCP, kontrolní součet (*Digest*) pro kontrolu integrity a délku užitečných dat (*Len*). Tyto položky hlavičky jsou spolu s užitečnými daty společně šifrovány i dešifrovány (po celou dobu, co se pohybují po okruhu) pomocí algoritmu AES-128 v módu čítače.

V síti Tor rozlišujeme následující typy řídících buněk:

1. **Řídicí buňky** pevné délky (*general cells*, obrázek 4a) jsou:
   - *padding* – aktuálně posílaná pro udržování spojení (*keep-alive*),
   - *create, created* – založení nového okruhu a potvrzení,
   - *create_fast, created_fast* – rychlé založení okruhu k prvnímu OR bez operací s veřejným klíčem,
   - *netinfo* – využíváno OR ke zjištění přesného času a adresy, pod kterou je OR dostupné v rámci sítě,
   - *destroy* – zrušení okruhu.

[6] Není nutné, aby byla tato hodnota sdílená pro celý okruh, protože OR si může asociovat dvě hodnoty *CircID*, a tak ví, které OR je další na cestě.

[7] Takové dělení přenosových buněk vzniklo za účelem omezení maximální délky [11] okruhu kvůli DoS útoku. Útočník mohl vytvořit libovolně dlouhý okruh přes jediný OR a ten tím pádem přetížit. Buňky *relay_ early* mohou přenášet vše co buňky *relay* s tím rozdílem, že jich může být po okruhu zasláno pouze 8 a tím je omezena maximální délka okruhu.

2. **Řídicí buňky** proměnlivé délky (*general cells*, obrázek 4b) jsou:
   - *versions* – určeno k vyjednávání o verzi protokolu Toru,
   - *vpadding* – zarovnání proměnlivé délky (nevyužito),
   - *certs, auth_challenge, authenticate, authorize* – autentizace mezi OR a OR nebo mezi OP a OR.
3. **Přenosové buňky** pevné délky (*relay cells*, obrázek 4c) jsou:
   - *data* – přenos dat,
   - *begin, end* – otevření/uzavření spojení,
   - *begin dir* – otevření lokálního spojení pro adresářové informace,
   - *teardown* – uzavření poškozeného spojení,
   - *connected* – notifikace OP, že spojení bylo úspěšně navázáno,
   - *extend, extended* – rozšíření okruhu o jeden hop a potvrzení,
   - *truncate, truncated* – zrušení části okruhu a potvrzení,
   - *resolve, resolved* – pro anonymní DNS,
   - *sendme* – řízení zahlcení.

5.2 Výběr uzlů při vytváření okruhů

Před vlastní konstrukcí okruhu OP volí ve výchozím nastavení konfigurace 3 OR, které budou okruh tvořit. O výběr vhodných OR se stará integrovaný algoritmus výběru cesty na straně OP.

Tor měří reálnou přenosovou kapacitu, kterou každý OR nabízí a podle toho přiděluje těmto OR váhy. Tyto váhy jsou používány k ovlivnění výběru OR při vytváření okruhů za účelem distribuce zátěže mezi OR.

Adresářové autority také přiřazují všem OR jeden či více statusů, které OP berou v potaz při výběru OR při vytváření okruhů.

- Strážce (*guard*) status získá OR, jehož doba běhu je alespoň mediánem doby pro všechny známé směrovače a jeho přenosová kapacita je minimum z 250 KiB/s a mediánu přenosové kapacity všech ostatních OR. OP zvolí 3 strážce a ty používá jako vstupní OR pro všechny vytvářené okruhy. Tyto 3 OR jsou obměňovány každých 30 až 60 dní. Zavedení OR typu strážce nesnižuje šanci, že by mohl být při vytváření prvního okruhu vybrán kompromitovaný OR, ale zajišťuje, že pokud se tak nestane, budou všechny další okruhy využívající tento OR bezpečné.
- Výstupní (*exit*) status získá OR, pokud povolují přímé spojení s cíli v Internetu. Takové OR navíc specifikují individuální odchozí politiky specifikující rozsah IP adres a portů, ke kterým jsou ochotny se připojit. Uživatelé se na základě těchto politik rozhodují, který směrovač zvolit pro koncovou pozici v každém z jejich okruhů. Strážcům a výstupním OR jsou přidělovány větší váhy pro vstupní respektive výstupní pozice v okruzích.
- Stabilní (*stable*) status získávají OR, jejichž střední doba mezi výpadky je alespoň mediánem pro všechny ostatní známé aktivní OR. Uživatelé, kteří pracují s datovými toky, které vyžadují dlouhodobé spojení, musí vybírat stabilní OR.

– Rychlý (*fast*) status získá OR, pokud je vhodné pro okruhy s vysokými nároky na přenosovou kapacitu.

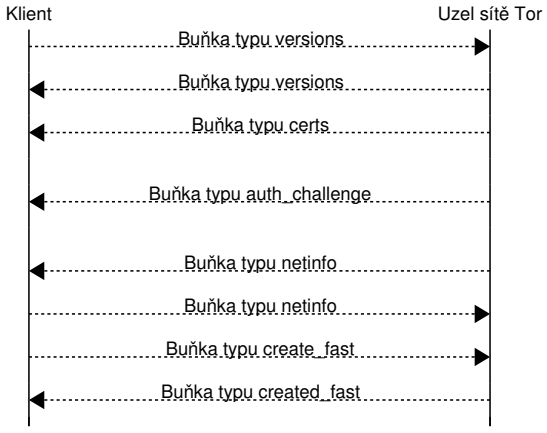
Tento sofistikovaný přístup, kdy jsou OR vybírány proporcionálně ku jejich přenosové kapacitě s využitím algoritmu vážení pro optimalizaci distribuce zátěže mezi OR s rozdílnými možnostmi (*guard*, *middle*, *exit* atd.), zajišťuje rozumné využití prostředků sítě. Neuvažuje ale aktuální zatížení uzlů a to může nepříznivě ovlivnit zpoždění daného okruhu.

Aby se zmírnil dopad korelačního útoku, OP při výběru OR při vytváření nového okruhu nikdy nevolí dva OR ze stejné podsítě (/16) nebo ze stejné rodiny. Rodina je množina OR, které vzájemně prohlašují, že tvoří skupinu. Vytváření těchto rodin vzniklo kvůli tomu, aby mohli běžní provozovatelé používat více OR než jeden a zároveň si vědomě nezvyšovaly šance, že získají více informací o síťovém provozu, než správci dvou izolovaných uzlů.

5.3 Konstrukce a zrušení okruhu

OP konstruuje okruhy inkrementálně, postupně vyjednává symetrický klíč s každým OR, jenž je součástí okruhu (po 1 hopu). Postup vyjednávání při vytváření okruhu je znázorněn na obrázku 5.

Před vyjednáním klíče probíhá s každým OR inicializační rutina, viz obrázek 5. V té se OP s OR dohodne na konkrétní verzi protokolu Toru, v dalším kroce zasílá OR autentizační výzvu, tu OP využije k autentizaci daného OR. Dále si zasílají informaci o síťových adresách, na kterých se mohou vzájemne zastihnout a v posledním kroce již dochází k vlastnímu postupu vyjednání sdíleného symetrického klíče. Ten je popsán podrobněji dále.



Obrázek 5. Zprávy zasílané [8] při inicializaci okruhu.

Pro názornost uvádím entity, které spolu komunikují jako **Alici** (uzel vytvářející okruh), **Boba** (první uzel v okruhu) a **Carol** (další uzel v okruhu).

1. K sestavení první části okruhu OP (Alice) zašle buňku typu *create* prvnímu OR na zvolené cestě (Bob). Tato buňka obsahuje nově zvolené *CircID* a veřejným klíčem (*onion* klíčem) Bobai.
2. Bob odpovídá buňkou typu *created*.
3. Jakmile je okruh mezi Alicí a Bobem sestaven, mohou si vzájemně vyměňovat další přenosové buňky zašifrované vyjednaným klíčem (ten je použit k vytvoření dalších dvou symetrických klíčů používaných algoritmem AES a při kontrole integrity pro oba směry komunikace).
4. K rozšíření okruhu Alice zašle Bobovi buňku typu *extend*, ta obsahuje adresu dalšího OR (Carol) a veřejným klíčem Carol.
5. Bob zkopíruje první část D-H od Alice do buňky typu *create* a přepošle jej Carol, zároveň si Bob zvolí nové *CircID*, které Alice již znát nemusí, Bob si pouze asociuje *CircID*, které sdílí s Alicí a nově zvolené jako součást okruhu.
6. Carol odpovídá buňkou typu *created*. Bob zabalí užitečná data této buňky do buňky typu *extended* a pošle ji Alici. Nyní je okruh rozšířen až ke Carol a Alice s Carol sdílí společný klíč.
7. Pro rozšíření okruhu o další OR a dále Alice vždy postupuje postupem uvedeným výše, poslednímu OR v okruhu vždy zasílá buňku *extend*, aby rozšířil okruh o jeden hop dále.

Tento postup dohody na klíči (*handshake*) umožňuje jednostrannou autentizaci entit, Alice ví, že provádí *handshake* s OR, ale OR se nestará o to, kdo se snaží okruh vytvořit, Alice nepoužívá veřejný klíč a zůstává tak anonymní. A jednostrannou autentizaci klíčů, Alice a OR se dohodnou na klíči, ale Alice ví pouze to, že se ho OR naučil také. Tímto způsobem vytváření okruhů je zajištěno dopředné utajení a čerstvost klíčů.

Jako optimalizaci může Alice při vytváření okruhu zaslat buňku typu *create_fast*, ve které posílá náhodné x. Bob odpovídá buňkou typu *created_fast* obsahující náhodnou hodnotu y. Společné sdílené klíče pak vytváří z otisku konkatenace těchto dvou hodnot $H(x|y)$. Tento způsob dohody na klíči je výpočetně nenáročný, ale neposkytuje autentizaci, integritu, důvěrnost ani dopředné utajení. Spoléhá se na to, že tyto vlastnosti zajistí protokol TLS.

V posledních verzích Toru se vyskytl nový způsob D-H dohody na klíči založený na eliptických křivkách, který je podle vývojářů Toru více bezpečný a podstatně rychlejší. Každopádně principiálně funguje podobně jako výše popsaný algoritmus.

Pro zrušení okruhu Alice zašle buňku typu *destroy*, tu obdrží každý OR v okruhu a uzavře všechny spojení okruhu, následně přepošle novou buňku *destroy* dále po směru okruhu. Alice může zrušit pouze část okruhu, a to tak, že zašle vybranému OR buňku typu *truncate*, ten následně pošle buňku *destroy* (tím zruší část okruhu za ním) a zrušení potvrdí Alici zpětně buňkou *truncated*. Ta může následně rozšířit okruh přes odlišné OR bez toho, aniž by dávala signalizaci dřívějším OR či externímu pozorovateli, že změnila okruh. Obdobně

pokud spadne uzel v okruhu, sousední uzel zašle zpět Alici buňku *truncated*. To je zároveň obranou proti útoku, kdy útočník vyřadil OR a sledoval, které okruhy to ovlivní.

5.4 Zasílání dat po okruhu

Okruh je sestaven, Alice sdílí klíče s každým OR (Bob, Carol) po cestě k výstupnímu uzlu, lze začít posílat přenosové buňky.

Když OR obdrží přenosovou buňku, vyhledá odpovídající okruh, dešifruje přenosovou hlavičku a užitečná data klíčem sezení pro daný okruh. Pokud buňka putuje směrem od Alice, OR zkontroluje, zda má dešifrovaná buňka platný otisk. Pokud je otisk platný, OR vyhledá *CircID* a další OR na cestě v okruhu, zamění *CircID* a dešifrovanou zprávu pošle dalšímu OR.

OP zpracovává příchozí přenosové buňky obdobným způsobem, iterativně odstraňuje (dešifruje) přenosovou hlavičku a užitečná data pomocí klíčů, které sdílí s každým OR v okruhu (od nejbližšího k nejvzdálenějšímu). Pokud je v jakékoliv fázi otisk buňky validní, buňka musí pocházet od OR, jehož zašifrování bylo právě odstraněno (dešifrováno).

Konstrukce přenosové buňky adresované danému OR probíhá tak, že Alice přidá k buňce otisk užitečných dat, iterativně šifruje obsah buňky (přenosovou hlavičku i užitečná data) symetrickými klíči pro každý uzel až k danému OR. Otisk je zašifrován v každém kroku iterace na jinou hodnotu, až u cílového OR bude mít smysluplnou hodnotu.

Když OR odpovídá Alici pomocí přenosové buňky, zašifruje přenosovou hlavičku a užitečná data klíčem, který sdílí s Alicí a pošle zpět směrem k Alici zpět po okruhu. Jednotlivá OR po cestě zpět přidávají další vrstvy zašifrování a přeposílají buňku zpět k Alici.

5.5 Otevírání a uzavírání spojení TCP

Vyžaduje-li uživatelská aplikace spojení TCP na danou adresu a port požádá OP skrze univerzální proxy rozhraní SOCKS, aby spojení vytvořila.

1. OP vybere nejnovější otevřený okruh nebo jej vytvoří, zvolí vhodné OR daného okruhu jako výstupní uzel (obvykle poslední uzel, ale může to být i jiný vzhledem k odchozím politikám). Zašle buňku *begin* výstupnímu uzlu společně s náhodně zvoleným *StreamID*.
2. Výstupní OR se spojí se vzdáleným uzlem a odpovídá buňkou *connected*.
3. OP zašle pomocí SOCKS odpověď, aby informovala uživatelskou aplikaci o úspěchu. Následně OP přijímá data aplikačního spojení a rozděluje je do buněk *data*, které pak putují po okruhu až ke zvolenému OR.

Uzavření Tor spojení je analogické s uzavíráním TCP spojení. Běžně se používá dvoucestný *handshake*, při normálním stavu spojení. Jednocestný *handshake* pak při výskytu chyb. Při abnormálním ukončení spojení, OR, který toto ukončení detekoval, zasílá buňku *teardown* směrem k funkčnímu konci okruhu. Při

normálním ukončení OR zasílá buňku *end* směrem po okruhu a druhá strana odpovídá též buňkou *end*. Protože všechny přenosové buňky používají vrstvené zašifrování, pouze cílový OR ví, že je to požadavek na uzavření spojení. Dvoucestný *handshake* umožňuje Toru podporovat aplikace založené na TCP, které používají polouzavřené spojení.

5.6 Kontrola integrity

Onion Routing neprováděl kontrolu integrity a síť tak byla více náchylná k útokům, kdy útočník sice neviděl obsah přenášených dat, protože byla zašifrovaná proudovou šifrou, ale mohl data pozměnit a tyto změny pozorovat dále na cestě.

Tor používá TLS, tudíž externí útočník nemůže modifikovat přenášená data. Kontrola integrity pak probíhá na hranicích okruhu.

Při vyjednávání klíčů s OR po cestě si vždy Alice a dané OR inicializují otisk SHA-1 derivátem ze sdíleného klíče (náhodnost, kterou znají pouze oni dva). Inkrementálně přidávají k otisku obsah všech přenosových buněk, které vytvoří a ke každé takto vytvořené buňce přidávají 4B aktuálního otisku, zároveň oba dva udržují otisk SHA-1 přijatých dat, aby mohly směrovače verifikovat, že otisky přijatých dat jsou korektní.

Aby mohl útočník odstranit nebo modifikovat buňky, musel by vydedukovat aktuální stav otisku (jenž závisí na veškerém datovém provozu mezi Alicí a Bobem), což nelze, protože otisky jsou po cestě okruhem zašifrované. Nároky pro výpočet otisku jsou minimální ve srovnání se šifrováním pomocí algoritmu AES, které je prováděno na každém OR při průchodu buňky okruhem.

5.7 Řízení zahlcení a omezení přenosové kapacity

K omezení přenosové kapacity v rámci okruhu byl implementován přístup *token bucket*, který vynucuje dlouhodobý průměrný datový tok, ale dovoluje i krátkodobý nárazový datový tok přesahující povolenou přenosovou kapacitu. OR umožňuje i nastavení, kdy po vyčerpání dostupné přenosové kapacity, přejde OR do režimu spánku a tím se stává dočasně nedostupným.

K zajištění dobrého zpoždění interaktivních služeb OR udržuje pro každý aktivní okruh vlastní frontu. Pořadí, které určuje, ze které fronty se bude vybírat příští buňka, je určeno podle využití přenosové kapacity (okruhy, které zaslaly málo dat, dostávají přednost). Okruhy s malým, občasným objemem dat mají lepší zpoždění než okruhy, které jsou využívány pro přenos velkých dávkových dat např. přenosy velkých souborů.

Zahlcení může být buď náhodné nebo úmyslné. V prvním případě stačí, když si více uživatelů vybere stejné spojení mezi dvěma OR a dojde k saturaci přenosové kapacity linky mezi nimi. V druhém případě stačí, když útočník pošle velký soubor skrze síť směrem k webovému serverů, který provozuje a následně útočník odmítne číst odpovědi.

Na úrovni okruhů probíhá řízení zahlcení tak, že si OR udržuje dvě okna *packaging window* a *delivery window*. První okno sleduje, kolik přenosových buněk

může vyslat OR/OP daným okruhem. Druhé sleduje, kolik přenosových buněk je OR/OP ochotno přijmout z daného okruhu. Tyto limity se neuplatňují na buňkách, které OR pouze přeposílají.

Okna jsou ve výchozím stavu inicializována na hodnotu 1000, při přijetí nebo odeslání buňky je tato hodnota patřičně dekrementována. Když OR obdrží dostatečný počet datových buněk pošle směrem k OP buňku *sendme* se *StreamID* 0. Když OR přijme tuto buňku, inkrementuje *packaging window* o 100. Dosáhne-li hodnota *packaging window* 0, OR přestává číst přenosové buňky z TCP spojení pro všechna spojení na daném okruhu a neposílá žádné další přenosové buňky do té doby než obdrží buňku *sendme*.

OP se chová identicky s výjimkou, že musí sledovat tato dvě okna pro každé OR na okruhu. Když *packaging window* dosáhne hodnoty 0, OP přestává číst ze spojení pocházejících od daného OR.

Na úrovni spojení je to podobné. OR i OP využívá buňky *sendme* k zajištění koncového řízení toku pro individuální spojení napříč okruhem. Každé spojení začíná s *packaging window* 500 a inkrementuje vždy o fixní hodnotu 50 po obdržení buňky *sendme*. Spíše než zaslání buňky *sendme* poté, co dorazil dostatečný počet buněk, řízení na této úrovni kontroluje, zda byla data úspěšně zaslána daným TCP spojením. Buňku *sendme* zasílá pouze tehdy, když počet bajtů čekajících na odeslání je pod nějakou hranicí (momentálně 10 buněk). Takto zvolené parametry dávají tolerované, ale rozhodně ne výborné hodnoty propustnosti a zpoždění.

6 Použití sítě Tor

Tato sekce popisuje jakým způsobem mohou uživatelé využívat funkcionality sítě Tor. Následuje výčet režimů, ve kterých lze vlastní instanci Toru provozovat, včetně výchozí konfigurace, se kterou Tor pracuje. Závěr této sekce uvádí přehled užitečných nástrojů, které mohou už uživatelé nebo vývojáři využít.

6.1 Přístup k síti Tor

Rozsáhlá komunita kolem Toru vytvořila celou škálu různých projektů zacílených na všechny známé platformy (Windows, Linux, FreeBSD a další), které se snaží uživatelům zjednodušit a zpříjemnit využívání funkcionality sítě Tor.

- Tor Standalone – Služba na pozadí, která zpřístupní síť Tor jako SOCKS proxy server. Je základem pro další uvedené možnosti.
- Tor Browser Bundle – Balíček obsahující předem nakonfigurovanou instanci Toru a upravenou verzi Firefoxu pro přímo surfování na webu. Výhodou tohoto balíčku je, že je co do bezpečnosti optimalizován pro protokol HTTP. Součástí je i HTTP Proxy Polipo[8].
- Vidalia – Grafické uživatelské rozhraní pro manipulaci s lokální instancí Toru. S klientskou OP komunikuje pomocí řídicího protokolu Toru.

[8] http://www.pps.univ-paris-diderot.fr/~jch/software/polipo/

– Tails[9] – Modifikovaný OS založený na Debianu, který obsahuje výše uvedenou konfiguraci Tor Browser Bundle pro rychlý anonymní přístup k surfování na webu. Jedná se o „live“ OS, který lze spouštět přímo z CD nebo USB klíčenky. Výhodou je, že nejsou ukládány žádné dočasné informace, celý OS běží přímo v RAM a po ukončení práce jsou veškerá uživatelská data smazána.
– Orbot [10] – Klientská aplikace pro použití sítě Tor na mobilní platformě Android.

6.2 Režimy činnosti

Instance Toru může běžet ve čtyřech různých režimech, uživatel může dle potřeby používat jednu či více instancí v jednom z dále uvedených režimů činnosti.

– **Klientský režim** – Uživatel využívá sítě Tor pouze jako klient.
– **OR** – Uživatel přispívá svou vlastní přenosovou kapacitou k celkové přenosové kapacitě sítě Tor, je součástí vytvářených virtuálních okruhů ve prospěch jiných uživatelů. Informace o jeho uzlu je dostupná ve veřejném adresáři Toru.
– **Most** – Podobné jako režim OR s tím rozdílem, že informace o uzlu uživatele není veřejně dostupná v adresáři Toru. Adresa jeho uzlu pak mohou ostatní uživatelé získat přes webovou stránku `přes https://bridges.torproject.org/` nebo elektronickou poštou. V tomto režimu lze navíc povolit rozšíření *pluggable transports*, které zajišťuje maskování charakteristik síťového provozu Toru.
– **Skrytá služba** – Uživatel může anonymně provozovat libovolnou službu např. webový server a k té mohou ostatní přistupovat přes síť Tor skrze adresu, kterou je pro tyto účely speciálně modifikované doménové jméno.

6.3 Výchozí konfigurace

Konfigurace je uložena v souboru `torrc` a ve výchozím stavu je Tor nakonfigurován následovně.

– Běží v klientském režimu.
– Virtuální okruhy mají délku 3 uzly.
– Není využíván mechanismus *leaky-pipe*.
– OP udržuje aktivní 4 okruhy, kterým přiděluje příchozí klientské spojení TCP.
– Životnost okruhu je omezena na 10 minut.
– Každých 30 sekund zvažuje OP konstrukci nového okruhu.

[9] https://tails.boum.org/
[10] `https://guardianproject.info/apps/orbot/`

6.4 Další užitečné nástroje

Uvedené nástroje mohou shledat užitečnými nejen běžní uživatelé, ale i vývojáři a další, kteří provádí nad sítí Tor výzkumnou činnost.

- Atlas[11] – Stavová informace o uzlech sítě Tor.
- Globe[12] – Stavová informace o uzlech sítě Tor.
- TorStatus[13] – Stavová informace o uzlech sítě Tor.
- Compass[14] –Stavová informace o uzlech sítě Tor.
- Stem[15] – Knihovna implementující řídicí protokol Toru, která je určena pro programovací jazyk Python.
- Arm[16] – Monitor pro příkazovou řádku pro instanci Toru, který umožňuje přímo nebo vzdáleně (ssh) zobrazit aktuální statistiky dané instance (přenosová kapacita, využití CPU, paměti, konfigurace, události, informace o připojení a další).

7 Závěr

Tato technická zpráva se zabývala anonymitou [24] na Internetu. Po stručném představení anonymizačních technik v e-mailových sítích a počátku anonymizace s nízkým zpožděním byla představena síť Tor [5,2,1].

Síť Tor je jednoduše dostupná [4] všem uživatelům Internetu. Tor funguje [11] jako tzv. *overlay* síť nad samotným Internetem. Namísto přímého posílání dat mezi oběma stranami komunikace však zajišťuje sestavování okruhů s několika prostředníky. V rámci sítě Tor se promíchávají toky všech uživatelů a protože je vnitřní komunikace šifrována, je docíleno zajištění anonymity na úrovní síťové vrstvy.

Reference

1. Tor Control Protocol Specification. [online], [cit. 2014-04-05].
   URL https://gitweb.torproject.org/torspec.git?a=blob_plain;hb=HEAD;f=control-spec.txt
2. Tor Directory Protocol Specification. [online], [cit. 2014-04-05].
   URL https://gitweb.torproject.org/torspec.git/blob_plain/HEAD:/dir-spec.txt
3. Tor Project : Anonymity Online. [online], [cit. 2013-10-27].
   URL https://www.torproject.org/

[11] https://atlas.torproject.org
[12] https://globe.torproject.org
[13] http://torstatus.blutmagie.de
[14] https://compass.torproject.org
[15] https://stem.torproject.org
[16] https://www.atagar.com/arm

4. Tor Project: Anonymity Online. [[online]], citováno 2014-05-13.
URL https://www.torproject.org/
5. Tor Protocol Specification. [online], [cit. 2014-04-05].
URL https://gitweb.torproject.org/torspec.git?a=blob_plain;hb=HEAD;f=tor-spec.txt
6. Chaum, D. L.: Untraceable electronic mail, return addresses, and digital pseudonyms. *Commun. ACM*, ročník 24, Únor 1981: s. 84–90, ISSN 0001-0782.
7. Christin, N.: Traveling the Silk Road: A measurement analysis of a large anonymous online marketplace. Technická Zpráva CMU-CyLab-12-018, CyLab, Carnegie Mellon University, Pittsburgh, USA, Červenec 2012.
8. Comaneci, D.: Protecting Internet Anonymity with TOR - The Onion Router Protocol - Ixia Connect. [[online]], citováno 2014-05-14.
URL http://blogs.ixiacom.com/ixia-blog/protecting-internet-anonymity-with-tor-the-onion-router-protocol/
9. Cottrell, L.: *Mixmaster & Remailer Attacks*. Citováno 2011-11-22.
URL http://www.dia.unisa.it/~ads/corso-security/www/NEW/remailer-essay
10. Danezis, G.; Dingledine, R.; Mathewson, N.: Mixminion: design of a type III anonymous remailer protocol. In *2003 Symposium on Security and Privacy.*, Květen 2003, ISSN 1081-6011, s. 2–15.
11. Dingledine, R.; Mathewson, N.; Murdoch, S.; aj.: Tor: The Second-Generation Onion Router (2014 DRAFT v1), 2014.
URL http://www.cl.cam.ac.uk/~sjm217/papers/tor14design.pdf
12. Dingledine, R.; Mathewson, N.; Syverson, P.: Tor: the second-generation onion router. In *Proceedings of the 13th conference on USENIX Security Symposium - Volume 13*, SSYM'04, Berkeley, CA, USA: USENIX Association, 2004.
13. Goldberg, I.; Wagner, D.; Brewer, E.: Privacy-enhancing technologies for the Internet. In *Compcon '97. Proceedings, IEEE*, Únor 1997, s. 103–109.
14. Goldberg, I. A.: *A pseudonymous communications infrastructure for the internet*. Dizertační práce, University of California, 2000.
15. Goldschlag, D.; Reed, M.; Syverson, P.: Hiding Routing information. In *Information Hiding*, editace R. Anderson, Lecture Notes in Computer Science 1174, Springer Berlin / Heidelberg, 1996, ISBN 978-3-540-61996-3, s. 137–150.
16. Gulcu, C.; Tsudik, G.: Mixing E-mail with Babel. In *Proceedings of the Symposium on Network and Distributed System Security*, Únor 1996, s. 2–16.
17. Kadianakis, G.: Packet Size Pluggable Transport and Traffic Morphing. Technická Zpráva 2012-03-004, The Tor Project, 2012.
URL https://research.torproject.org/techreports/morpher-2012-03-13.pdf
18. Koblas, D.; Koblas, M. R.: Socks. In *Proceedings of the UNIX Security III Symposium*, Září 1992, s. 77–83.
19. Leech, M.; Ganis, M.; Lee, Y.; aj.: RFC 1928 SOCKS Protocol Version 5. Březen 1996.
20. Li, B.; Erdin, E.; Gunes, M. H.; aj.: An Overview of Anonymity Technology Usage. *Computer Communications*, 2013.

21. Mazières, D.; Kaashoek, M. F.: The design, implementation and operation of an email pseudonym server. In *Proceedings of the 5th ACM conference on Computer and communications security*, CCS '98, New York, NY, USA: ACM, 1998, ISBN 1-58113-007-4, s. 27–36.
22. Moeller, U.: *Mixmaster Protocol Version 3*. Citováno 2011-11-22.
URL http://www.eskimo.com/~rowdenw/crypt/Mix/draft-moeller-v3-01.txt
23. Perry, M.: Tips for Running an Exit Node with Minimal Harassment. [[online]], citováno 2014-05-13.
URL https://blog.torproject.org/blog/tips-running-exit-node-minimal-harassment
24. Pfitzmann, A.; Hansen, M.: A terminology for talking about privacy by data minimization: Anonymity, unlinkability, undetectability, unobservability, pseudonymity, and identity management. Technická zpráva, Srpen 2010, verze 0.34.
URL
https://dud.inf.tu-dresden.de/literatur/Anon_Terminology_v0.34.pdf
25. Pfitzmann, B.; Pfitzmann, A.: How to Break the Direct RSA-Implementation of Mixes. In *Advances in Cryptology — EUROCRYPT '89*, editace J.-J. Quisquater; J. Vandewalle, Lecture Notes in Computer Science 434, Springer Berlin / Heidelberg, 1990, ISBN 978-3-540-53433-4, s. 373–381.
26. Reed, M.; Syverson, P.; Goldschlag, D.: Anonymous connections and onion routing. *Selected Areas in Communications, IEEE Journal on*, ročník 16, č. 4, Květen 1998: s. 482 –494, ISSN 0733-8716.
27. Rescorla, E.: *RFC 2631 Diffie-Hellman Key Agreement Method*. Červen 1999.
28. Syverson, P.; Reed, M.; Goldschlag, D.: Onion routing access configurations. In *DARPA Information Survivability Conference and Exposition, 2000. DISCEX '00. Proceedings*, ročník 1, 2000, s. 34 –40 vol.1.
29. Syverson, P.; Tsudik, G.; Reed, M.; aj.: Towards an Analysis of Onion Routing Security. In *Designing Privacy Enhancing Technologies*, editace H. Federrath, Lecture Notes in Computer Science 2009, Springer Berlin / Heidelberg, 2001, ISBN 978-3-540-41724-8, s. 96–114.
30. Syverson, P.; Tsudik, G.; Reed, M.; aj.: Towards an analysis of onion routing security. In *Designing Privacy Enhancing Technologies*, Springer, 2001, s. 96–114.