Tor és Torpark: az új generációs anonim böngészők funkcionális és teljesítményelemzése

Gyöngyösi László*

Budapesti Műszaki és Gazdaságtudományi Egyetem
Gazdaság- és Társadalomtudományi Kar
Információ- és Tudásmenedzsment Tanszék
1111 Budapest, Sztoczek u. 2. St. ép. I. em. 117.
Telefon: (36 1) 463-1832, Fax: (36 1) 463-4035
e-mail: gyongyosi.laszlo@gmail.com

Absztrakt

Az onion routing technológia alkalmazása az internetes kommunikáció anonimitásának, lehallgathatatlanságának korszerű eszköze. A Tor rendszerek onion routing alapon működő felhasználói rendszerek, amelyek a felhasználók egyéni alkalmazásait és a hálózat működését egyaránt támogatják. A Tor hálózatokon működő ingyenes alkalmazások új generációját képviseli a 2006-ban közzétett Torpark böngésző, amely Firefox alapon anonim böngészést tesz lehetővé és akár egy flashdrive-ról is futtatható. A szerző áttekinti az onion routing technológia alapjait, bemutatja a Tor rendszer működését, majd elemzi a Tor működését kliens, illetve szerver szerepben, végül a Torpark böngésző használatát teszteli. A tesztelés központi eleme a hálózati sebesség mérése különböző mérési elrendezésekben. Megállapítható, hogy mind a Tor, mind a Torpark a megadott funkcionalitásnak megfelelően működik, hozzáférésük ingyenes, kezelhetőségük egyszerű, azonban használatukkal a weblapok illetve fájlok letöltési ideje számottevően megnő.

Kulcsszavak: onion routing, Tor, Torpark, anonim kommunikáció.

1. Bevezető

Napjainkban a magánjellegű, privát kommunikáció egyre nagyobb jelentőségűvé válik, az elküldött üzeneteink védelme során alkalmazott biztonsági megoldások önmagukban azonban csupán azok illetéktelen módon történő felhasználását képesek megakadályozni. A privát kommunikáció lehallgathatatlanságának és sérthetetlenségének biztosítása mára már bárki számára elérhető a rendelkezésünkre álló technikai eszközökkel, azonban ez nem jelenti azt, hogy a titkos kommunikáció sebezhetetlen is. Az elküldött információ ugyanis

* Gyöngyösi László, IV. évfolyamos műszaki informatika szakos hallgató, BME Híradástechnikai Tanszék

nyomonkövethető, analizálható, legyen az bármilyen erejű titkosító kóddal is védve. Mi jelenthet valódi megoldást adataink védelmére? Létezik egyáltalán olyan technika, amely teljes anonimitást biztosít számunkra az interneten való böngészés folyamán?

Az egyik lehetséges alternatívát az onion routing hálózat jelentheti. Az onion routing technológiát az USA haditengerészeti kutatólaboratóriumában fejlesztették ki. E technológia szerint speciális routerek, – ún. onion routerek – adják át egymásnak a csomagokat a hálózaton belül. A csomagok többszörösen titkosítva közlekednek, és – mivel a titkosított rétegek hagymahéjszerűen egymásba ágyazódnak – az egyes hálózati útvonalakon elhelyezkedő routerek a csomagok tartalmát sem ismerik. Az onion struktúrából adódóan, a csomópontoknak arról sincs információjuk, hogy egy-egy adott csomag kitől jött, illetve az hová tart.

Kik alkotják a Tor (*The Onion Router*) rendszert? A Tor rendszere olyan felhasználókból áll, akik egymást segítve, saját erőforrásaik teljesítményének egy részét feláldozva, közösen hajtanak végre egy-egy adott részfeladatot az onion-hálózaton belül. Az onion routing technológia egy nagy hatásfokú módszernek tekinthető, azonban megvalósítása – egyelőre legalábbis – önmagában még drága a speciális dedikált hálózat miatt. A hálózat működéséhez egy igen jelentős méretű, előre kiépített kiszolgáló rendszer szükséges. A megvalósítás költségeinek ellenére, napjainkra a Tor rendszerek elterjedése immár lehetővé teszi, hogy az átlagos felhasználó is az onion routing által nyújtott szolgáltatást vehessen igénybe.

A Tor-t már számos közelmúltban megjelent ingyenes alkalmazás támogatja, így például a TorPark is, amely egy – a Firefox alapjaira épülő – anonim böngésző program. [5] Jelen tanulmány elsődleges célja a Tor kliens, illetve a Torpark működési sebességén keresztül az onion-hálózat teljesítményének, hatékonyságának összehasonlítása egy hagyományos kapcsolat teljesítményével. A mérések során különböző weboldalakat látogattam meg, illetve több, különböző méretű fájlt töltöttem le. A hagyományos és az onion-hálózat teljesítménybeli különbségének megállapításához több, különféle típusú mérést végeztem el. A hagyományos kapcsolat sebességének megállapítása után a Tor kliens sebességét, majd a Torpark teljesítményét elemeztem. A Torpark böngésző programot mind merevlemezről, mind pedig usb-ről futtatva egyaránt teszteltem. A kapott eredményeket táblázatba foglaltam a könnyebb átláthatóság kedvéért, valamint több összehasonlító értékelést is készítettem.

2. Az Onion Routing technológia

2.1. Általános ismertető

Maga az Onion Routing technológia egy olyan nyilvános hálózatot használó infrastruktúra, amelynek célja az anonim kapcsolatok kiépítése, létrehozása és működtetése. Az onion („hagyma”) technológia lényege egy olyan réteges felépítésű adatstruktúra, amely a kapcsolat útvonalát határozza meg, illetve a kapcsolat útvonalát definiálja. Amikor az onion routing rendszerben felépítünk egy kapcsolatot, az adott út mentén lévő csomópontok mindegyike lebont egy réteget a hagyma-struktúráról, aminek következtében a lebontó router hozzájut az útvonal következő routerének címéhez. Ezen megoldás segítségével az adatokat továbbító csomópontok nem ismerik magát a teljes útvonalat, csupán azon két szomszédjukat, akitől kapták, illetve akinek továbbküldik az adatokat. Az egyes routerek esetében azonban az információk más és más formában jelenhetnek meg, mivel a hagymahéj struktúra egyes rétegeiben, valamint a titkos csatornákon is eltérő lehet az alkalmazott rejtjelezési módszer. Annak ellenére, hogy az egyes routerekhez eltérő formátumú információk kerülnek, az onion routing esetében a csomópontok között találhatunk egy előre rögzített struktúrájú elemet is – a csomópontok között áramló cellák méretét. Annak ellenére, hogy az onion routing eltérő rejtjelezési módszereket alkalmaz az egyes rétegekben, a cellahosszak minden esetben megegyeznek.[2] A rögzített cellaméretből adódóan azonban, egy onion-hálózaton belüli csomag útjának nyomonkövethetősége és felderítése kivitelezhetetlenül nehéz feladatnak bizonyul. A gyakorlatban így, egy adott fél anonimitása abban az esetben is biztosított, ha a teljes útvonalon mindössze egyetlen onion router található.

Az onion routing technológiában a kapcsolatfelépítés során az egyes rétegek nyilvános kulcsú kriptográfia alkalmazásával kerülnek kialakításra, a küldött adatokat azonban már nem publikus kulcsú aszimmetrikus rejtjelezéssel, hanem szimmetrikus kulcsú kódolással védjük. A kapcsolat létrejötte után, a küldésre szánt adatok méretéből adódóan, a szimmetrikus kulcsú rejtjelezés használatával jelentős mennyiségű erőforrást spórolhatunk meg a hálózaton belül. A nyilvános kulcsú algoritmusokat jellemzően nem használjuk a küldött adataink titkosítására, mivel az aszimmetrikus kulcsú rejtjelező algoritmusok számításigénye számottevően jelentősebb. Amíg a kapcsolat kialakítása során továbbítandó adatok esetén a publikus kulcsú rejtjelezés hatékonyan megvalósítható, a kapcsolat létrejötte után küldött üzenetek aszimmetrikus rejtjelezése esetében már jelentős teljesítménycsökkenéssel kellene számolnunk az onion-hálózaton belül. Az egyes csomópontok között a rejtjelezés nyilvánvalóan okoz valamiféle többletterhet, amely közvetlenül is megjelenik a hálózaton belül. A kriptográfiai algoritmusok okozta számítási többletet azonban az egyes csomópontok egymás között egyenletesen elosztják. A

gyakorlatban ez azt jelenti, hogy minden router körülbelül ugyanakkora erőforrás-igényű transzformációt végez a rajta keresztülhaladó csomagon, mint egy bármelyik másik router a struktúrán belül.

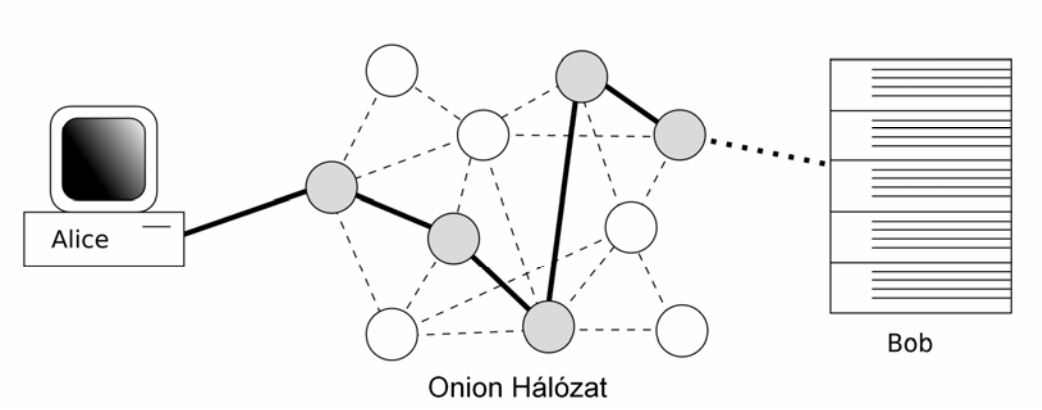
Az onion routing technológiát az egyes alkalmazások, így a mindennap használt böngésző programok, illetve levelezőprogramok, ugyanúgy képesek használni, mint egy hagyományos TCP/IP rendszert. Az alkalmazások egy proxy segítségével vehetik igénybe az onion router hálózatot, így magán a futtatandó alkalmazáson gyakorlatilag semmit sem kell változtatnunk. Az egymással kommunikáló, másik félre utaló információk egyszerűen eltávolíthatóak a küldött adatokból, amivel magát a kapcsolatot is anonimmá változtatjuk, így a kommunikáció teljes folyamatát is. A szükséges információ kiszűrése pedig könnyedén és egyszerűen elvégezhető proxy-k segítségével. A kommunikáló felek anonimizálására azonban nem csupán egy külső fél számára kerül sor, hanem az egymással kommunikáló felek számára is. Így elérhető, hogy maguk a résztvevő felek se rendelkezzenek információval egymás kilétét illetően, amennyiben ez az üzenet tartalmából explicit módon egyébként sem derülhetne ki. Az adott szakaszra érvényes kapcsolaton túl így maga a kommunikáció teljes folyamata anonimmá tehető.

2.2. A onion-hálózat működési elve

Az onion-hálózatban az Alice és Bob közti, azaz a kapcsolatot kezdeményező fél és a célgép közti kapcsolat nem közvetlenül, hanem az onion routereken keresztül épül fel. Magának az onion-hálózatnak elsődleges feladata és célja annak biztosítása az Alice és Bob közti kapcsolat teljesen anonimitása. A hálózaton belül külön kezeljük a kezdeményező és a cél között felépült kapcsolatot, valamint az egyes proxy-k között lévő összeköttetéseket. A kapcsolat felépítésének legelején a kezdeményező fél, Alice egy alkalmazás proxy-val veszi fel a kapcsolatot, amely a kérést továbbítja az onion-hálózatba. [1] Az onion-hálózaton keresztül felépített útvonalat az alkalmazás proxyt követő, onion-hálózaton belüli proxy határozza meg. Az anonim kapcsolat felépülése után az útvonalon keresztül Alice már adatot is tud küldeni, amelyet az összes, útvonalon belüli onion-router számára külön-külön rejtjelez. A közbenső onion-routerek számára ez egy explicit módon megkülönböztethető „hagymahéjat” jelent, amelyet az éppen aktuális router „leránt” az onionról. A héjak lerántása után Bobhoz az Alice által eredetileg küldött üzenet érkezik meg.

Az anonim kapcsolat felépítéséhez Alice, azaz a küldő fél kiválasztja a küldéshez használni kívánt proxy-k halmazát. [1] A következőkben az aktuálisan rendelkezésre álló proxy-t P_i-vel jelöljük, a teljes proxy listát pedig $\left(P_{\phi(1)}, P_{\phi(2)}, \ldots P_{\phi(n)}\right)$-vel, amely n darab proxy-t tartalmaz. A kiválasztott proxy-k alkotják a titkos kommunikáció útvonalát. Az útvonal

minden elemének, azaz minden láncszemének megbízhatónak és sérthetetlennek kell lenni, ugyanis a titkos kommunikáció e csomópontokon keresztül kerül felépítésre és megvalósításra. A láncolat felépítéséhez Alice elküldi az onion-ját, amely onion az egyes routerek publikus kulcsaiból áll. A kapcsolatfelépítés során egy *virtuális áramkört* hozunk létre, amelyben az egyes proxy-k között szimmetrikus kulcsú titkosítással biztosítjuk a kommunikáció védelmét. [9] A láncolat legutolsó eleme, vagyis az utolsó proxy jelenti az egyetlen kapcsolódási pontot a külvilággal, esetünkben az Internettel. A láncolat utolsó láncszemén keresztül kommunikálunk a vevő oldallal, azaz Bobbal. Az utolsó proxy, valamit a vevő oldal közötti kommunikáció már egy átlagos tulajdonságokkal rendelkező csatornán keresztül valósul meg, így ezen útvonal már nem eleme az anonim hálózatnak.



1. ábra: Alice meghatározza a kommunikációban résztvevő proxy-kat

A fenti ábrán a szürke csomópontok jelentik a kiválasztott proxy-kat. A proxy-k közötti megvastagított útvonalakon a kommunikáció védett, titkosított. Azon proxy-k között, amelyek nem vesznek részt az anonim kommunikációban, a kommunikáció a hagyományos módon zajlik. [6]

A virtuális áramkörön keresztüli kommunikáció – miután a kapcsolatot sikerült felépíteni –, meglehetősen stabilnak, illetve gyorsnak mondható. A kapcsolat sebességének megítélésekor azonban figyelembe kell vennünk azt, hogy a proxy-k közötti útvonalak, valamint az Alice és az első $P_{\phi(1)}$ proxy között található útvonal is, külön-külön titkosítási eljárásokat, kriptográfiai primitíveket használ. Az onion routing technológia legutolsó $P_{\phi(n)}$-el jelölt proxy-ja, és a vevő oldal közötti útvonal azonban már *nem védett*, azaz a protokollban már nem vonatkozik kötelező védelem ezen útvonalra.

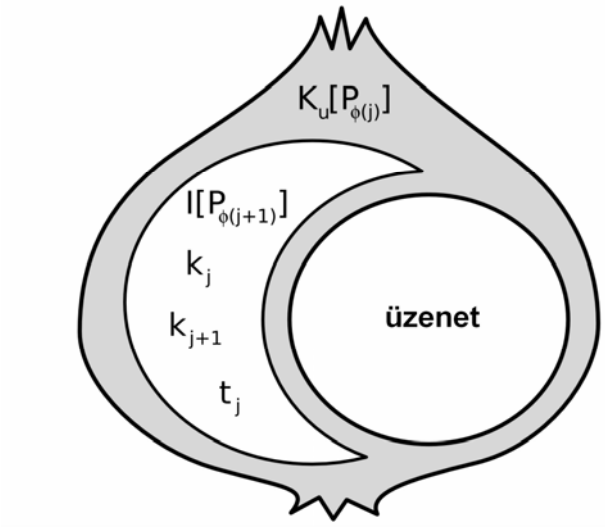
A tanulmány elkövetkező részére vonatkozóan az onion jelölésére bevezetem az O_j jelölést, amely egy proxy szerver által is feldolgozható datagram-ot jelent. Az adott datagram aktuális állapotának feldolgozására az adott hálózaton belül és az adott

időpontban azonban csak egyetlen egy proxy szerver lehet képes. Azaz, általánosítva, az O_j onion-t csak és kizárólag a $P_{\phi(j)}$ proxy szerver olvashatja, amely a kiolvasott adatok alapján felépíti a kapcsolatot az őt megelőző $P_{\phi(j-1)}$ -ik proxyval, valamit az őt követő $P_{\phi(j+1)}$ proxyval. Mivel a $P_{\phi(j)}$ -ik proxy már ismeri a $P_{\phi(j-1)}$ -ik proxy-t – mivel tőle kapta az O_j onion-t –, így a datagramnak ezek után már csak a következő csomópont IP címére lesz szüksége, azaz $I\left[P_{\phi(j+1)}\right]$ -re.

Az onion aktuális rétege tartalmaz továbbá egy-egy szimmetrikus kulcsot is, ezt jelöljük k_j , illetve k_{j+1} – el. A k_j kulcsot az előző csomóponthoz, a k_{j+1} kulcsot pedig a következő csomóponttal történő kommunikáció során használhatjuk majd fel. Az Onion része továbbá egy időzítő, amelyet t_j -vel jelöltem. A t_j időtartam adja meg a kapcsolatfelépítésre rendelkezésre álló maximális időmennyiséget a kijelölt útvonalon.

Miután a $P_{\phi(j)}$ proxy feldolgozta az O_j onion-t, azt továbbküldi a láncolat következő elemének, azaz a $P_{\phi(j+1)}$ proxynak. A datagram tartalmát azonban a $P_{\phi(j)}$ proxy publikus kulcsával titkosítottuk, így azt csak és kizárólag a $P_{\phi(j)}$ proxy képes visszafejteni, a saját privát kulcspárájával. [9] Ezen kódolt adat a $P_{\phi(j)}$ proxyhoz kerülő O_j onion legfelső rétegében található, ezt $K_u\left[P_{\phi(j)}\right]$ -vel jelöltem.

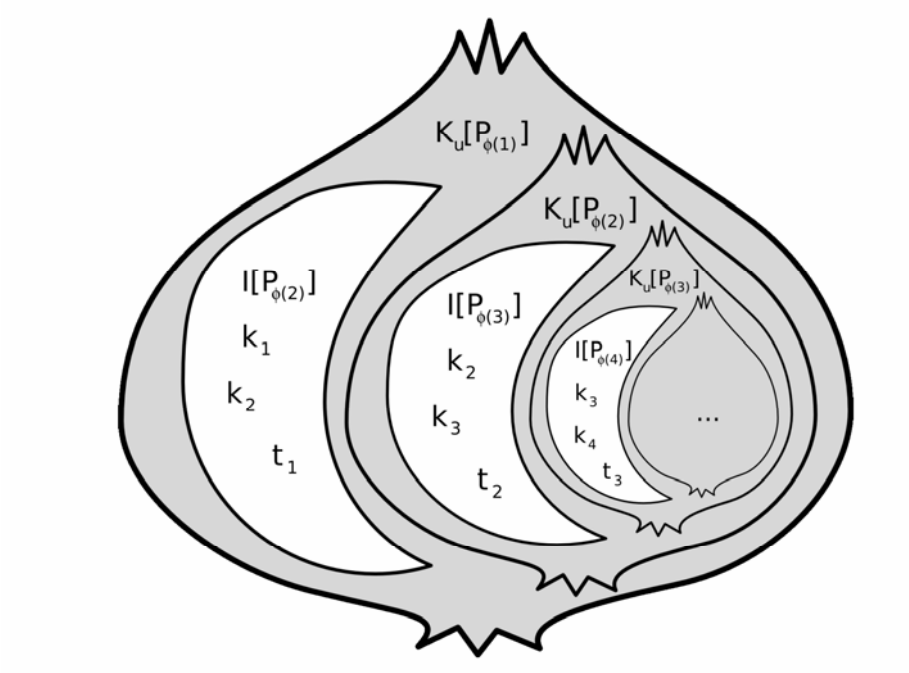
A következő ábrán láthatjuk az előzőeknek megfelelően kialakított O_j onion felépítését, amelyet csak és kizárólag a $P_{\phi(j)}$ proxy képes dekódolni. Az onion routing igazi ereje tehát ezen réteges felépítésben, illetve a többszörös, többrétegű információelosztásban rejlik elsősorban. [1][6]



2. ábra: Egy onion általános felépítése

Minden egyes onion üzenete tehát egy olyan újabb onion, amely a következő csomópontnak szól, és ez rekurzívan ismétlődik, egészen O_n -ig. Az O_n onion azonban már nem tartalmaz tényleges üzenetet, így az vagy üres, vagy kitöltő adatokat, azaz paddinget tartalmaz. Az O_1 onionon belüli üzenet gyakorlatilag egy újabb onion, amelyen belül egy újabb onion-t találhatunk, az O_2 -t.

A kezdeti O_1 onion kialakításához Alice-nek fordított sorrendben kell felépítenie az onion-t, így első lépésként egy M padding üzenetet küld, amelyet a $P_{\phi(n)}$ proxy kap meg. Az M padding üzenethez mellékeli a k_n szimmetrikus kulcsot is, amelyre a $P_{\phi(n)}$ és a $P_{\phi(n-1)}$ proxy közti kommunikáció során lesz szükség. A kulcs mellé Alice elhelyezi a t_n időzítőt is, majd a teljes réteget titkosítja a $P_{\phi(n)}$ proxy K_u nyilvános kulcsával, amelynek eredménye $K_u\left[P_{\phi(j)}\right]$ lesz. Ezen titkosított datagram, vagy másképpen az O_n onion lesz az O_{n-1} -ik onion üzenet részében, amely a $P_{\phi(n-2)}$ proxy $I\left[P_{\phi(n-2)}\right]$ IP címe mellett, tartalmazza a k_{n-1}, k_n szimmetrikus kulcsokat, valamint a t_{n-1} időzítőt is. Az onion struktúra felépítésére rétegről-rétegre, fordított sorrendben kerül sor. Egy-egy csomópontnak szóló üzenet, egy olyan újabb onion formájában kerül továbbításra az adott proxy nyilvános kulcsával titkosítva, amelyet csak és kizárólag maga a címzett lehet képes dekódolni. [6]



3. ábra: Az egymásba ágyazott onion struktúra

2.2.1. Kapcsolatfelvétel az onion-hálózaton keresztül

Ahhoz, hogy Bob képes legyen a titkos útvonalat alkotó proxy-láncolaton keresztül kapcsolatba kerülni Alice-szel, Alice kialakítj egy ún. *kezdeményező*, vagy más néven *előre irányuló* oniont. Alice, az onion felépítése során egy *create* üzenetbe illeszti az elkészült onion réteget, majd az üzenethez csatol egy speciális fejlécet, amelyben a virtuális áramkör felépítését kezdeményezi az fogadó alkalmazás proxy-tól. Az alkalmazás proxy a $P_{\phi(1)}$ proxy-nak továbbítja az O_1 onion-t, amelynek dekódolását így egyedül a $P_{\phi(1)}$ proxy képes elvégezni, a saját privát kulcsával. A dekódolás után a $P_{\phi(1)}$ proxy feldolgozza a személyesen neki szóló útvonalinformációkat. Ezt követően egy egyedi azonosítószámot rendel a kapcsolathoz, amely – a jelenlegi helyzetben – Alice-től indul, és egészen a $P_{\phi(2)}$ -es proxy-ig tart. A csomópont a vett O_1 onion-t a dekódolás és feldolgozás után nem dobja el, a későbbiekben előforduló, esetleges onion-spoofing támadások kivédése céljából.

A következőkben, az összes *előre irányuló* kapcsolat során küldött adat dekódolása a k_1 szimmetrikus kulccsal történik, amik azután a $P_{\phi(2)}$ proxy-hoz kerülnek tovább. Hasonlóképpen, az Alice felé irányuló kommunikáció esetén, a $P_{\phi(2)}$ proxy-tól érkező csomagok a k_1 szimmetrikus kulccsal kerülnek kódolásra. Az O_2 onion tehát az O_1 onion részeként kerül továbbításra a hálózaton belül, amelyet – a jelen példában –, a $P_{\phi(1)}$ proxy küld a $P_{\phi(2)}$ proxynak, egy *create* üzeneten belül.

Az *előre irányuló* kommunikáció során tehát minden egyes proxy „leránt” egy-egy réteget az onionról, amely réteg tartalmaz:

- egy időzítőt;
- a következő node azonosítóját;
- egy szimmetrikus kulcsot a megelőző, illetve a következő node-dal történő kommunikációhoz;
- illetve egy újabb onion-t.

Az adott proxy-nak a lerántott rétegben található szimmetrikus kulcs segítségével kell kialakítania a titkos kapcsolatot az őt megelőző, illetve az őt követő node-dal, mindezt az időzítőben megadott maximális időtartamon belül. Amint az egyes proxy-k végrehajtották a rájuk kiszabott feladatokat, Alice immár készen áll a Bobbal történő kommunikációra.

Alice küldendő üzenetét jelöljük *M*-el, amelynek továbbítására hagyományos eszközöket használ, így a küldő és a vevő közti kommunikáció alapvetően egy védelem nélküli csatornán keresztül kerül továbbításra. Alice ekkor az *M* üzenetet az összes, birtokában lévő szimmetrikus kulcs felhasználásával titkosítja, az egyes kulcsokat pedig eltérő sorrendben alkalmazza az onion-hálózat felépítése miatt. Az üzenetküldés során alkalmazott kódolási sorrend ennek következtében $k_n \ldots k_1$ lesz. A titkosított *M* üzenetet egy *data* cellában küldi tovább az első proxynak, azaz $P_{\phi(1)}$-nek. A $P_{\phi(1)}$ proxy lerántja az első réteget a kódolt üzenetről, majd a maradékot továbbítja a $P_{\phi(2)}$ proxynak, mindezt egy újabb *data* cellába ágyazottan. A felépült virtuális áramkörön belül minden egyes proxy lerántja a saját rétegét az üzenetről, így maga a titkosítás nélküli M üzenet érkezik meg a $P_{\phi(n)}$ proxyhoz. [1] A $P_{\phi(n)}$ proxy ekkor felveszi a kapcsolatot Bobbal, majd továbbítja az immáron titkosítást nem tartalmazó M üzenetet. Bob, a $P_{\phi(n)}$ proxy által küldött M üzenet, és az Alice által eredetileg küldött üzenet között nem képes különbséget tenni, így Bob tulajdonaképpen azt hiszi, hogy éppen Alice-szel vette fel a kapcsolatot.

Mivel a kommunikáció során szimmetrikus kulcsokat használunk, így az üzenetek kódolásához, illetve dekódolásához szükséges számítási kapacitás mindvégig alacsony szinten marad, amely egy aszimmetrikus kulcsú módszerrel összehasonlítva jelentős erőforrás megtakarítást jelent a kommunikáció során. Az üzenetek küldésekor Alice az egyetlen olyan tagja a kommunikációnak, aki a teljes, globális útvonalra vonatkozóan rendelkezik információkkal a résztvevő csomópontokkal kapcsolatban.

A Bob-tól Alice felé tartó, azaz a virtuális áramkörön belüli *visszafelé irányuló* kommunikáció során a Bob által küldött üzenetet ismét kódolnunk kell, amely ezúttal a közbenső csomópontok segítségével történik. A Bobtól érkező válasz üzenetet jelöljük *R*-rel. A válaszüzenetet elsőként a $P_{\phi(n)}$ proxy kapja meg, ahol a proxy a k_n szimmetrikus kulcsával lekódolja a kapott R-t, majd azt továbbküldi (azaz a láncolatban visszafelé küldi) a $P_{\phi(n-1)}$ proxynak. A folyamat egészen a vételi oldalig ismétlődik, Alice végül a $k_n \ldots k_1$ szimmetrikus kulcssorozattal tikosított R választ kapja meg. Alice a dekódolást a legkülső réteggel kezdi, azaz kívülről halad befelé. A szimmetrikus kulcsokat így az eredeti $k_1 \ldots k_n$ sorrendben alkalmazva, dekódolhatja a kapott üzenetet.

Összefoglalva, Alice és Bob bármiféle előzetes kulcsegyeztetés, illetve saját kilétük felfedése nélkül képes kapcsolatba lépni egymással. A kommunikáció tehát teljesen titkos és anonim.

2.2.2. Válasz-onionok

Mivel az onion routing által használt és felépített virtuális áramkör gyorsan bontásra kerül – az egyes időzítők rövid élettartamúak –, így a technológiának egy külön eljárásra lesz szüksége annak érdekében, hogy a vételi oldal válaszolhasson a kezdeményező fél kérésére.[1] Vagyis egy olyan megoldásra lesz szükségünk, amely Bobtól indulva képes újra felépíteni a virtuális áramkört, úgy, hogy Bob eközben nem szerezhet tudomást Alice személyazonossága felől. A *válasz-onionok* ebben nyújtanak segítséget Alice és Bob számára.

Egy válasz-onion ugyanazokat az útvonal-információkat tartalmazza, mint az eredeti virtuális áramkör onion-jai, az egyetlen különbség a *rétegek sorrendjében* van. A Bobtól Alice felé irányuló kommunikáció során ugyanis az egyes onion rétegek sorrendje fordított. Az eredetileg az onion közepében elhelyezett kitöltő padding információ pedig ezúttal a $K_u\left[P_{\phi(1)}\right]$ -ben kerül elküldésre. A válasz onion ezen kívül tartalmazza a szükséges útvonal-információkat, valamint azon szimmetrikus kulcsot is, amelyet a $P_{\phi(1)}$ proxy használhat az Alice-hez való kapcsolódáshoz. Az onion-t egy *create* cellában küldi el Bob, ami után már képes lesz arra, hogy visszaállítsa, vagyis újból felépítse a közte és Alice közt húzódó virtuális áramkört. A Bob által felhasznált onion-t pedig Alice küldi el Bobnak, a Bobbal folytatott kommunikáció legvégén.

Összefoglalva: A routerek közötti kérések onion formában továbbítódnak, az onion-on belül pedig minden réteg az adott onion router nyilvános kulcsával kódolt. A hálózaton belül Alice, azaz a kezdeti router állítja elő a teljes onion üzenetet, amelyben a rétegkulcsok a routerek nyilvános kulcsait jelenti. A kérés továbbítása után minden egyes router lebont egy-egy réteget az onion-ról, amely csak és kizárólag a router saját rétege lehet. Fordított irányban pedig a routerek hozzáadnak egy-egy réteget a küldött datagramhoz. Alice legelső kérése Bobhoz véletlenszerűen jut el az onion-hálózaton keresztül. Amennyiben az onion-hálózaton belüli, véletlenszerűen választott kilépési pont képes felvenni a kapcsolatot Bobbal, akkor kialakul egy virtuális áramkör, amely egy logikai útvonalat jelent Alice és Bob között, az onion-hálózaton keresztül. A virtuális áramkör felépülése után, Alice és Bob között a teljes adatforgalom, illetve kommunikáció ezen a logikai útvonalon keresztül zajlik. A felépített kapcsolat bontását a kommunikáció két végpontja, Alice vagy Bob kezdeményezheti, amelyet a kommunikációban részt vevő közbenső routerek továbbítanak egymásnak. A hálózatban a szomszédos routerek között, az onion-hálózat felépítésekor ún. hosszú távú kapcsolatok alakulnak ki, így magának az onion-hálózatnak a hálózati topológiája előre rögzített. Ezáltal válik megvalósíthatóvá például az, hogy az egyes közbenső routerek ismerjék saját szomszédjaikat.

3. A technológia gyakorlati megvalósítása

3.1. A Tor kapcsolatfelépítése

Az onion routing modellben a kezdeti onion minden egyes rétege speciális információkat tartalmaz a virtuális áramkörben résztvevő proxy-k számára. Az onion előrehaladtával annak mérete folyamatosan csökken, az egyes csomópontok eltávolítják az onionról a saját rétegüket. Ennek következtében, ha ismert a kezdeti onion mérete, akkor abból képesek lehetünk megállapítani adott proxy elhelyezkedését a virtuális áramkörön belül. [1] Az onion-hálózaton belüli proxy-k elhelyezkedése azonban már egy olyan extra információnak számít, aminek nyilvánosságra kerülését a rendszer tervezői mindenféleképpen szerették volna elkerülni. Az onion méretének megfigyeléséből így hozzájuthatunk egy titkos információhoz, és mindezt úgy tehetjük meg, hogy bármiféle feltörési kísérlet hajtottunk volna végre a kommunikációs rendszeren. A megfigyelés alapján történő bepozícionálás elkerülése végett a protokoll tervezői azt javasolták, hogy a lerántott réteg után az adott proxy, – még a továbbküldés előtt –, helyezzen el értéktelen kitöltő információt az értékes információ helyébe. Így ugyanis az onion mérete változatlan marad, azaz a *méretfigyeléses* támadás kivédhetővé válik. A paddingelés azonban a gyakorlatban igen leterhelte a rendszert, aminek jelentős sebességcsökkenés lett az eredménye. Ahhoz ugyanis, hogy az adott proxy képes legyen a paddingelés végrehajtására, az onion rétegén belül további információkat is el kell helyezni, így például az alkalmazott kódolási eljárást, valamint azt is biztosítani kell, hogy a véletlenszerű padding nem változtatja meg a korábbi adatok dekódolásának sikerességét. [1]

A Tor az onion méretváltozásának elkerülése, és így az információszivárgás megelőzése érdekében a virtuális áramkör méretét növeli meg, mégpedig minden egyes lépésben egy újabb útvonallal. A növelés érdekében egy előre rögzített méretű *extended* cellát küld a virtuális áramkör legutolsó node-jának.[2] Az *extended* cella a *Diffie-Hellmann* kulcscserével kapcsolatban tartalmaz információkat, amelynek elküldésére az adott csomópont publikus kulcsát használja. Ezt követően, Alice és az új proxy anonim üzenetek felhasználásával kialakít egy szimmetrikus kulcsot, amely üzeneteket az útvonal egyes közbenső csomópontjai közvetítik.

A hagyományos onion routing technológia a virtuális áramkörön belüli útvonalak titkosítására vonatkozóan nem köt ki speciális feltételeket. A Tor ezzel szemben a node-ok közötti útvonalak titkosítására a *TLS*-t használja, ami szintén egy handshake alapú, szimmetrikus kulcsú protokoll. A node-ok közötti kommunikáció biztonsága a TLS használatával garantált, hiszen ha egy támadó kompromittálja valamelyik útvonal kulcsát –

akár többet is egyszerre –, a megtámadott útvonalon elküldött üzeneteket akkor sem lesz képes dekódolni a támadó.

A *truncate* cellák az *extended* cellákkal ellentétes szerepet töltenek be, ugyanis a *truncate* egy olyan csomópontot jelöl ki a virtuális áramkörön belül, amelytől számítva – egy későbbi link meghibásodása esetén – az összes node automatikusan kikerül a virtuális áramkörből, azaz a rendszer kizárja őket. Például, tegyük fel, hogy Alice a $P_{\phi(j)}$ proxynak küldött egy üzenetet, amely következőképpen néz ki: $M \cdot k_j \cdot k_{j-1} \cdot k_{j-2} \ldots k_1$. Amikor az adott router veszi az üzenetet, a saját rétegét eltávolítja, majd ellenőrzi, hogy az üzenet értelmezhető-e számára. Amennyiben a router nem képes értelmezni az információt, az üzenetet titkosítva továbbküldi a következő node-nak.

A küldő fél, Alice a *truncate* cellával tehát képes kijavítani a virtuális áramkörben bekövetkezett, esetleges csomópontsérülést is. Ezt a *truncate* üzenet eljuttatásával teheti meg, amelyet közvetlenül a sérült node előtti lévő csomópontnak kell eljuttatnia, amely ezt követően kizárja a virtuális áramkörből az őt követő csomópontokat, és másfelé tereli a forgalmat. Az új csomópontok felvételére pedig a már ismertetett *extended* cellát használja.[2] A módszer így lehetővé teszi azt is, hogy egy teljesen új útvonalat alakítsunk ki anélkül, hogy ehhez magát a teljes felépített virtuális áramkört bontanunk kellene. Ennek következtében elkerülhetjük a teljes, immár megváltoztatott logikai útvonal újbóli felépítését. Az útvonalak változtatása tehát dinamikusan kezelhető az onion-hálózaton belül, továbbá így bizonyos támadásokat is kivédhetünk a rendszeren belül. A gyakran változtatott útvonalakkal ugyanis elkerülhetjük azt, hogy egy támadó az üzenetek analizálásából bármiféle következtetést tudjon levonni a küldő személyazonosságát illetően. A Tor kliens szoftvere éppen ezért periodikusan változtatja a kommunikáció során alkalmazott útvonalakat, szisztematikusan újabb és újabb útvonalakat generálva. A Tor kliensén belül ezt egy háttérben futó processz végzi, amely az előzőekben vázolt módszerrel képes megváltoztatni a kommunikációban résztvevő csomópontokat, így magát a kommunikáció útvonalát is. [2]

3.1.1. Címtár szerverek

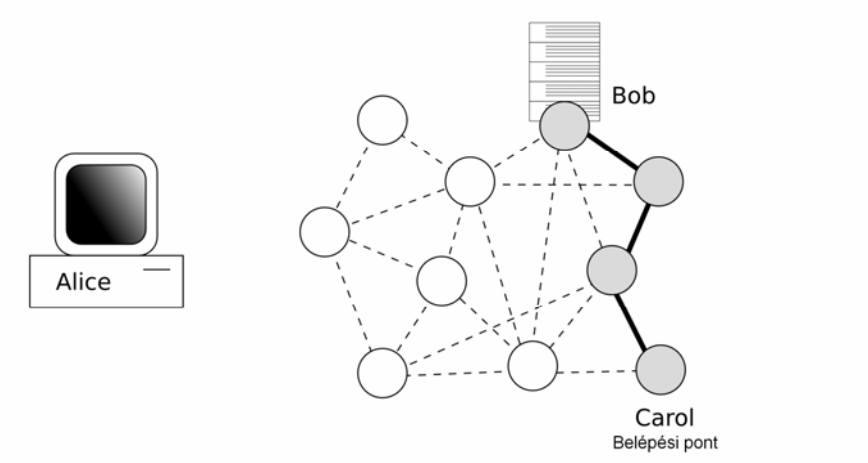
Ahhoz, hogy Alice képes legyen meghatározni a virtuális áramkörben résztvevő csomópontokat, ismernie kell az adott hálózaton belüli proxy-kat, illetve a proxy-khoz tartozó publikus kulcsokat. A kapcsolatfelépítéshez ezen kívül szüksége lehet az egyes csomópontok késleltetési idejére is. [2]

A hagyományos onion routing-ban ezeket az üzeneteket Alice broadcast-olva kapja meg, azaz minden egyes node egy bizonyos időközönként elárasztja a hálózatot a helyzetjelentő információival, amely az aktuális node-ról tartalmaz információkat. A broadcast-olt üzeneteket mind a kliensek, mind pedig a routerek képesek olvasni. Így az egyes csomópontok képesek megállapítani a teljes hálózat az aktuális állapotát, illetve pontosan feltérképezhetik azt. A broadcasting-gal történő elárasztásos módszer azonban jelentős hálózati forgalomnövekedéssel jár együtt, ami nagymértékben leterheli a hálózatot, illetve nehézkessé válik a hálózat konzisztenciájának megállapítása is. A Tor esetében ezért az aktuálisan elérhető proxy-k egy külön adatbázisba kerülnek, azaz egy dedikált szerver szolgáltat információt az éppen aktív és eltérhető proxy-król. A dedikált szerverek pedig csak olyan proxy-k lehetnek, amelyeknek hitelességéről a routerek már közvetlenül is meggyőződtek. Az útvonal szereplőinek meghatározásakor a kiválasztott szerverek összehasonlítják a hálózatról kialakított „térképüket", majd közösen döntenek az útvonal egyes routereiről. Ezt követően mindegyik kiválasztott proxy megkapja azt az aláírást, amellyel jogosulttá válik az útvonalban való részvételre. A kliensek és a routerek az útvonalak változtatására a dedikált szervereket egy újabb onion-on keresztül kérhetik meg. A Tor-on belül általában *15 percenként* kerül sor az útvonalak megváltoztatására. [2]

3.1.2. Rejtett szerverek, találkozási pontok

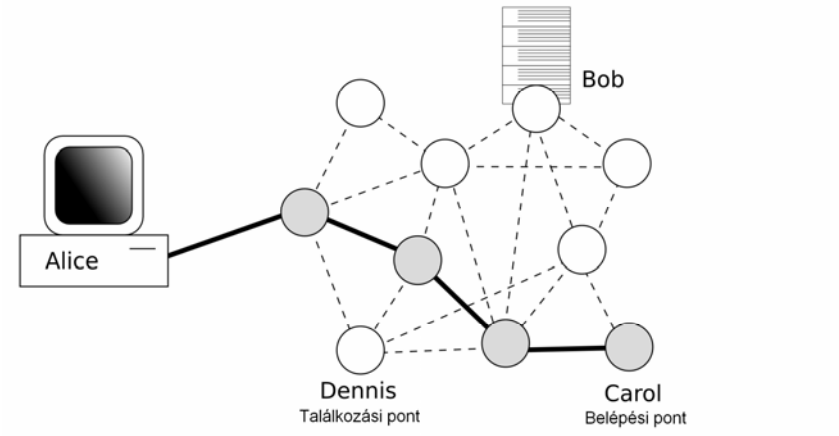
A Tor egyik legfőbb újdonsága a hagyományos onion routing technológiához képest a *rejtett szerverek* alkalmazása. A rejtett szerver egy olyan gépen található, amely egyébként egy aktív onion routerként is funkcionál a rendszerben, ami a gyakorlatban általában egy klienst jelent. Az onion struktúra többszörös rétegezettsége lehetővé teszi, hogy a kliensek anonimitásukat megőrizve kapcsolódhassak egy-egy rejtett szerverhez, anélkül, hogy pontosan tudnák, hogy kihez is csatlakoztak.[2]

A *találkozási pont* alkalmazása pedig lehetővé teszi azt, hogy az útvonalválasztás teljes mértékben anonim maradjon, azaz sem az adatfolyam tartalmáról, sem pedig magáról a forrásról vagy a célról nem kerülhet nyilvánosságra semmilyen információ. Tegyük fel, hogy Bob egy rejtett szerver, amit a felhasználók egy független routeren keresztül érhetnek el, azaz egy ún. *találkozási ponton* keresztül. Annak érdekében, hogy a résztvevő felelek teljes anonimitásuk megőrzése mellett legyenek képesek kialakítani egy találkozási pontot, Bobnak néhány közbenső csomópont segítségét is igénybe kell vennie. A közbenső proxy-k használatára tehát a találkozási pont egyeztetése miatt lesz szükség. Első lépésként, a Bob-hoz csatlakozni kívánó felhasználók kerülnek regisztrálásra, azaz egyenként kapnak egy-egy egyedi azonosító kulcsot, amely a rejtett weboldal eléréséhez használható majd fel a későbbiekben. [6]



4. ábra: Bob a találkozási pontra vonatkozó kérések visszaküldését kéri a közbenső csomópontoktól

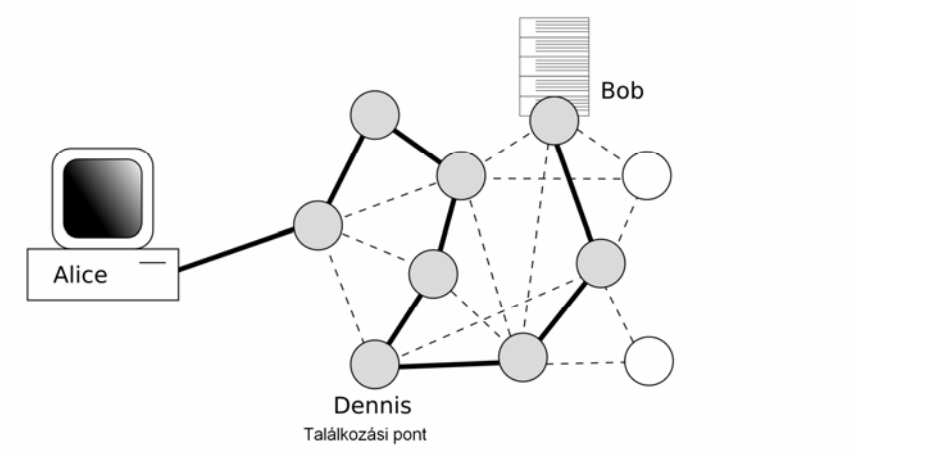
Bob ezt követően kiválaszt néhány routert, amelyet *belépési csomópontok* szerepét töltik majd be az elkövetkezőkben. A kiválasztott routereket tartalmazó listát pedig elküldi a címtár szervereknek, amely alapján a proxy-k immár elérhetik a találkozási pont kialakításához szükséges információkat. Bob, az információ hitelességének biztosítása érdekében az elküldött listát ellátta aláírásával is, amelyet a saját titkos weboldalához tartozó kulccsal generált. Miután Bob meghatározta a routereket, egy üzenetben megkéri őket, hogy küldjék el neki a találkozási pont kialakítására vonatkozó kéréseiket. Abban az esetben, ha Alice szeretne kapcsolódni Bob rejtett weboldalára, akkor Alice választ egy proxy-t, – a kiválasztott proxy legyen *Dennis* – , amely proxy a találkozási pont szerepét fogja betölteni a kommunikációban. Dennis, a címtár szerver alapján kiválaszt egyet a lehetséges belépési pontok közül, amellyel ezt követően kapcsolatba lép. A listából kiválasztott belépési pont pedig legyen *Carol*. [6]



5. ábra: Alice, Carol segítségével közli Bobbal, hogy a találkozási pont Dennis lesz.

Alice ezután közli Carollal, hogy közte és Bob weboldala között valahol található egy találkozási pont, amely jelen példában Dennis routerét jelenti. Alice egy datagramban elküld Carolnak egy random *találkozási kulcsot*, valamit a Diffie-Hellmann kulcscsere *első felét,* mindezt Bob rejtett weboldalához tartozó nyilvános kulccsal titkosítva. A hitelesség biztosítása érdekében Alice a datagramban egy aláírást is elhelyezhet, miáltal Bob egyértelműen megbizonyosodhat arról, hogy az üzenet Alice-től érkezett. Az így küldött üzenetet Carol nem képes megfejteni, egyedül csak Bob képes kinyitni a saját weboldalához tartozó privát kulcspárjával. Ezután Carol válasz-onion formában továbbítja a kéréseket Bobnak. Alice ezt követően kialakít egy virtuális áramkört közte és Dennis között, valamint arra kéri, hogy a felépült logikai kapcsolatba vegye fel a Bob oldaláról induló virtuális áramkört is, amelyet a találkozási kulccsal tud majd beazonosítani.

Ha Bob engedélyezi Alice hozzáférését a rejtett weboldalához, akkor Bob is kialakít egy virtuális áramkört közte és Dennis között, valamint elküldi a találkozási kulcsot, illetve a Diffie-Helmann kulcscsere hiányzó, második felét. A Diffie-Helmann paraméterek kicserélésének eredményeképpen pedig Alice és Bob között kialakult a titkos, szimmetrikus kulcs. A kulcsot Dennis nem képes megfejteni, azt csak és kizárólag Alice és Bob számára ismert [6]



6. ábra: Dennis továbbítja az Alice és Bob között folyó kommunikáció csomagjait.

A kulcskialakítási folyamat során Alice és Bob is csupán a saját virtuális áramkörén belüli csomópontokat ismeri, hasonlóképpen, mint Dennis és Carol. A kommunikációban résztvevő felek így csupán a velük közvetlenül kapcsolatban álló csomópontokat képesek beazonosítani. *Azaz, Carol és Dennis nem rendelkezik semmiféle információval sem egymás kilétét, sem pedig Alice vagy Bob személyét illetően.* A kommunikációban résztvevő közbenső csomópontok pedig még arról sem kapnak információt, hogy ők éppenséggel egy titkos kommunikáció részesei. A találkozási ponton keresztüli csatlakozás során a hálózat

megfigyelhető jellemzői, viselkedése, mind a tényleges résztvevők, mind pedig egy külső lehallgató számára is megkülönböztethetetlen a hagyományos értelemben vett onion-hálózatot használó kommunikációtól.

3.2. A Tor felhasználói kliense

A fentebb vázolt technikai jellegű finomításokon kívül, a Tor projekt további célja az anonimitás fokozása, illetve annak maximális szintre hozása. A hálózati kommunikáción belüli anonimitás magasabb szintre emelése az onion routerek hálózatba illesztésével technikailag biztosított. Azonban az így elérhető magas fokú anonimitás sokkalta inkább vet fel társadalmi jellegű kérdéseket, mintsem újabb technikai jellegűeket. A Tor adminisztrátorai nyomatékosan hangsúlyozzák, hogy a hálózat ilyetén módon történő anonimizálását csak a törvényes, legitim adatforgalom számára preferálják, azaz az illegális tartalmú weboldalak megtekintését, valamit az illegális P2P forgalom elrejtését, illetve az egyéb bűncselekmények elkövetését nem támogatják. [2]

Természetesen senkinek sem lehet joga arra, hogy a Tor felhasználói köréről bármiféle listát állítson össze, azonban az egyéb informális felmérésekből már jó közelítéssel bemérhető a felhasználók száma, illetve azok köre. A felmérésekből kiderül, hogy a Tor-t alkalmazók szignifikáns hányada már nem csupán kliensként, hanem proxy-ként is megjelenik az onion-hálózaton belül. [2] Vagyis a Tor felhasználók jelentős hányada már nem csupán az anonim böngészést szeretné igénybe venni az onion-hálózaton keresztül, hanem egyúttal proxy-ként funkcionálva is szeretne bekapcsolódni az anonim hálózati kommunikációba. [10]

Az onion-hálózat a legitim forgalmának növelésére talán a hétköznapi felhasználók inspirálása lehetne az egyik legkézenfekvőbb megoldás. A Tor ezt könnyedén el is érheti, hiszen egy olyan felhasználóbarát grafikus felülettel rendelkezik, amelynek kezelését könnyedén elsajátíthatja bárki, mindemellett a program nyílt forráskódú. A kliens számítógépére feltelepítve egy tűzfalként funkcionál, amely a kimenő TCP/IP kapcsolatot folyamatosan figyeli és ellenőrzi, ill. továbbítja azt az onion-hálózatba. A kliens szoftver egy szerver szoftver is egyben, így a felhasználók proxy-ként is funkcionálhatnak az onion-hálózatban. [1]

4. Valós implementáció tesztelése

4.1. A Tor tesztelése kliens és szerver szerepben

A Tor népszerűségének egyik legfontosabb tényezője annak egyszerű kezelhetősége, valamint egyszerű feltelepítése és konfigurálhatósága. A Tor felépítésből adódóan, minél több ember használja az onion-hálózatot, annak teljesítménye annál nagyobb lesz, így egyre nagyobb hatékonysággal biztosítható az anonimitás a kommunikáció során. A hatékonyság növekedése mellett ugyanakkor az egyre több proxy megjelenése további biztonságot injektál a rendszerbe. A következőkben arról szeretnék meggyőződni, hogy a Tor használata valóban olyan egyszerű-e és gördülékeny-e, mint azt ahogyan a készítői állítják.[2]

Kliens szerepben a kapcsolatfelépítéshez szükséges időt mértem, összevetve a hagyományos értékekkel, valamint megvizsgáltam a kommunikációs overhead és a letöltött fájlméretek közötti kapcsolatot. Az elvégzett mérésekből megállapítottam, hogy mennyire erős a korreláció a letöltött fájlméret illetve a kommunikáció késleltetése között. Az eredmények egyértelműen rávilágítanak a rendszer működésének sajátosságaira, annak jellegzetességeire. A szerver szerepben a szerver anonimitásának megnövelése volt a célom, valamint a többi szerver illetve kapcsolódó proxy-k anonimitásának növelése.

4.1.1. A kliens

A Tor kliensét a http://tor.eff.org/ [3] weboldalról tölthetjük le. A weboldal jól felépített, könnyen kezelhető és egyszerűen megtalálhatóak a legfontosabb dokumentumok, programok. A következő lépésben kiválasztottam a letöltendő Tor klienst, a Windows-hoz két package áll rendelkezésünkre. A legegyszerűbb választás esetében a Tor-t tölthetjük le egymagában, a Tor Control Panel-lel (TorCP), valamint a Privoxy-val együtt, amely a fejlécek kezelését végzi. Az „expert” csomag csak a Tor-t tartalmazza, így lehetővé teszi a letöltő számára azt, hogy saját vezérlőpanelt, illetve proxy-t használjon. A letöltések közül a javasolt Windows-os alapcsomagot töltöttem le és telepítettem fel. A letöltés után a szoftvert az alapbeállításoknak megfelelően, automatikus módon telepítettem. A installálás után megjelent a program ikonja a tálcán is. Következő lépésben a web böngészőt kellett beállítanom úgy, hogy az képes legyen a Privoxy kezelésére. A proxy szerver a Tor csomagban került feltelepítésre. A Firefox böngészőt használva egy rendkívül egyszerű plug-in segítségével, egy kattintással ki-be kapcsolhatjuk a Tor szolgáltatásait. Internet Explorer esetében ilyen plugin nem áll rendelkezésünkre, így ebben az esetben a proxy-t manuálisan kell feltelepítenünk. A FireFoxhoz a *Torbutton* [4] elnevezésű plugint használtam. A plugin telepítése után a böngésző ablakában megjelent a Torbutton ikonja.

A Tor helyes működésének leellenőrzéséhez a számítógép IP címét vizsgáltam meg, egy erre specializálódott oldal segítségével: http://www.showmyip.com/. [7] A hagyományos és az onion-hálózat működési sebességének különbsége az IP-cím lekérdezésekor már egyértelműen észlelhetővé vált. A Tor kliens nélküli, hagyományos kapcsolat esetén az IP-cím lekérdezése 1.8 másodpercet vett igénybe, míg a Tor bekapcsolása után ez mintegy 13 másodpercig tartott. A Tor engedélyezése után a számítógép eredeti IP-címe maszkolva lett, és egy „ál”-IP címet kapott a számítógép.

4.2. Teljesítményelemzés

4.2.1. A tesztelési módszer

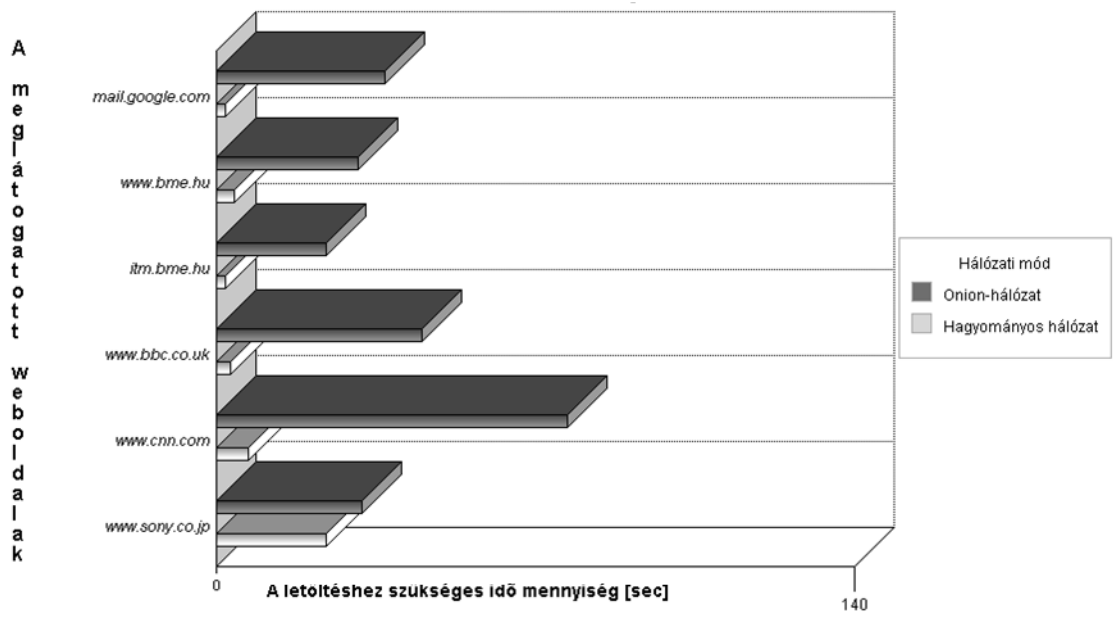
A következő tesztsorozatban a Tor használatával járó sebességkülönbséget, illetve a kommunikációs overhead mértékének megváltozását vizsgáltam. A meglátogatott weboldalakat igyekeztem úgy kiválasztani, hogy az egyes szerverek földrajzi elhelyezkedése is eltérő legyen, felmérve így az onion-hálózat, illetve a végpontok közötti eszközök működésének sebességét. A tesztek során a különböző weboldalak, fájlok letöltési idejét mértem, Tor nélküli hagyományos kapcsolat esetén, illetve aktivizált Tor esetén. A kapott eredmények alapján elmondható, hogy a Tor aktivizálása jelentős sebességcsökkenést idézett elő a weboldalak letöltési idejében. Az elvégzett mérések eredménye alapján, a Tor használatával átlagban több mint tízszeres sebességcsökkenést tapasztaltam a letöltések között.

1. Táblázat: A tesztelt weboldalakhoz tartozó letöltési idők

Meglátogatott oldal	Tor KI	Tor BE	Különbség %
http://mail.google.com	1.8 sec	37.2 sec	2055.55
http://www.bme.hu	4.5 sec	31.6 sec	688.88
http://itm.bme.hu/	2.3 sec	23.5 sec	1021.73
http://www.bbc.co.uk	3.5 sec	45.3 sec	1285.71
http://www.cnn.com/	7.2 sec	77.8 sec	1069.44
http://www.sony.co.jp	23.3 sec	32.1 sec	137.33

A meglátogatott weboldalakhoz tartozó mérési eredményeket a következő grafikon foglalja össze, amely a hagyományos Tor nélküli, illetve a Tor használatával kapott eredményeket mutatja. A weboldalak letöltési sebességei alapján a legnagyobb kommunikációs overhead azon oldalaknál jelentkezett, amelyeknek földrajzi elhelyezkedése közelebbi, a legjelentősebb lassulást így azon oldalak esetében tapasztaltam, amelyek egy hagyományos

csatlakozás esetén szinte azonnal letöltődtek a böngészőn keresztül. A leghosszabb letöltési időt a www.cnn.com weboldal letöltése alatt mértem, azonban Tor nélküli és a Tor-ral mért eredmények közti különbség „mindössze" tízszeres, azaz a leghosszabb mért letöltési idő nem azonos a legnagyobb mértékű sebességváltozással. A legnagyobb különbséget ugyanis a postafiókomként használt levelezőrendszer, azaz a http://mail.google.com betöltése alatt mértem. Ennek elsősorban az lehetett az oka, hogy a Google levelezőrendszerének használatához speciálisan titkosított kapcsolat szükséges, amelynek megvalósítása az onion-hálózaton belül jelentős számítástöbbletet igényelt.



7. ábra: A letöltési idők alakulása

A www.sony.co.jp japán weboldal betöltése a hagyományos, Tor nélküli kapcsolat esetén is nagyságrendekkel tovább tartott, ez azonban betudható a weboldal által használt grafikus elemeknek is. A Tor-ral mért eredmények közel azonosnak mondhatóak a Tor nélküli kapcsolat esetén kapott mérési eredményekkel. Az itthoni illetőségű weboldalakhoz viszonyítva, a mért időtöbblet csupán azok töredékére adódott. Az eltérések okát az egy-egy földrajzi ponton elhelyezkedő szerverek, illetve az addig felépült onion-hálózat alkotóelemeinek számában kereshetjük elsősorban.

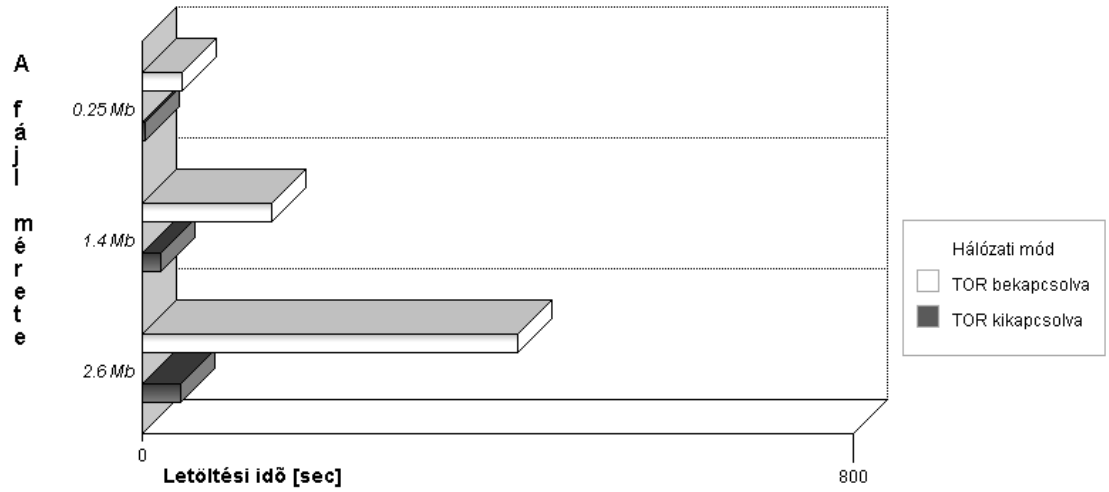
A következő mérés során különböző fájlokat töltöttem le hagyományos, illetve az onion-hálózatot használva. A fájlméreteket 0.26 Mb és 2.65 Mb közötti intervallumból választottam, amellyel sikerült szignifikáns eredményeket kapnom a hálózati sebességek kommunikációs overhead-jének megállapításához. A letöltött fájlok a BME oldalaihoz,

illetve egy, az aktuális félévben hallgatott tárgyhoz (Számítógépes grafika és képfeldolgozás) kapcsolódnak.

2. Táblázat: A különböző fájlméretek esetén mért letöltési idők

Letöltött fájl	Méret (Mb)	Tor KI	Tor BE
00000034.pdf	0.26	3.1 sec	45 sec
bmegame.pdf	1.40	21 sec	145 sec
memodhw.pdf	2.60	43 sec	423 sec

A kapott eredmények alapján elmondható, hogy Tor bekapcsolásával a sebességek ugyanolyan arányban csökkentek, mint a weboldal-látogatások alatt. A 0.26 Mb méretű fájltól egészen a 2.6 Mb-os fájlig, a Tor esetén mért sebesség a hagyományos kapcsolat esetén mért sebesség mintegy tizedére adódott. Az tesztelt fájlméretek alapján, egy hagyományos kapcsolat esetén a letöltési sebességek átlaga 71.67 Kbit/sec-re adódott, míg ugyanezen fájlok letöltése Tor használatával már csupán 7.33 Kbit/sec volt.

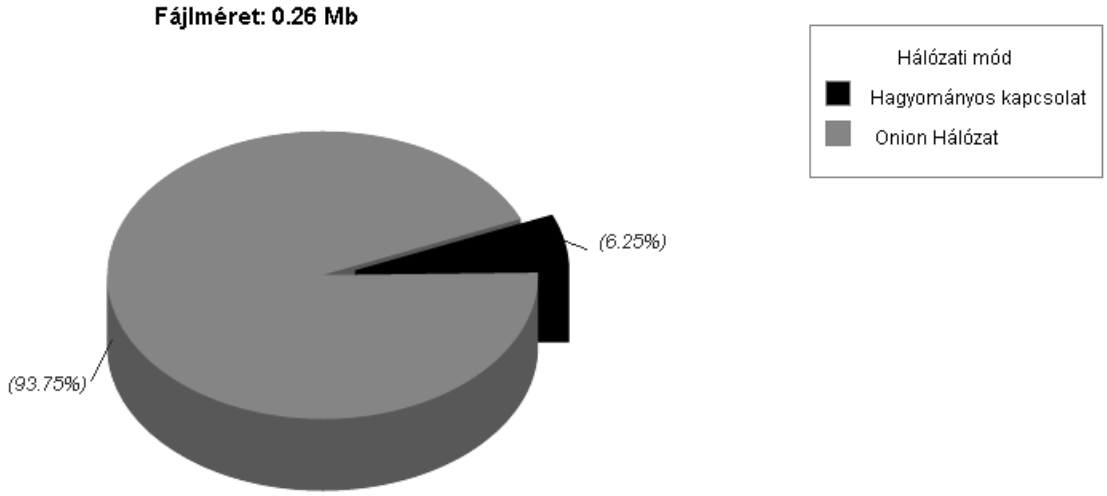


8. ábra: A fájlok letöltési ideje

A letöltési idők arányát külön-külön fájlméretekre lebontva is láthatjuk a következőkben.[1] Elsőként a 0.26 Mb-os fájl esetében mért, hagyományos és onion-hálózaton keresztül mért letöltési idők arányát hasonlítottam össze A 0.26 Mb-os fájl letöltése esetében az onion-

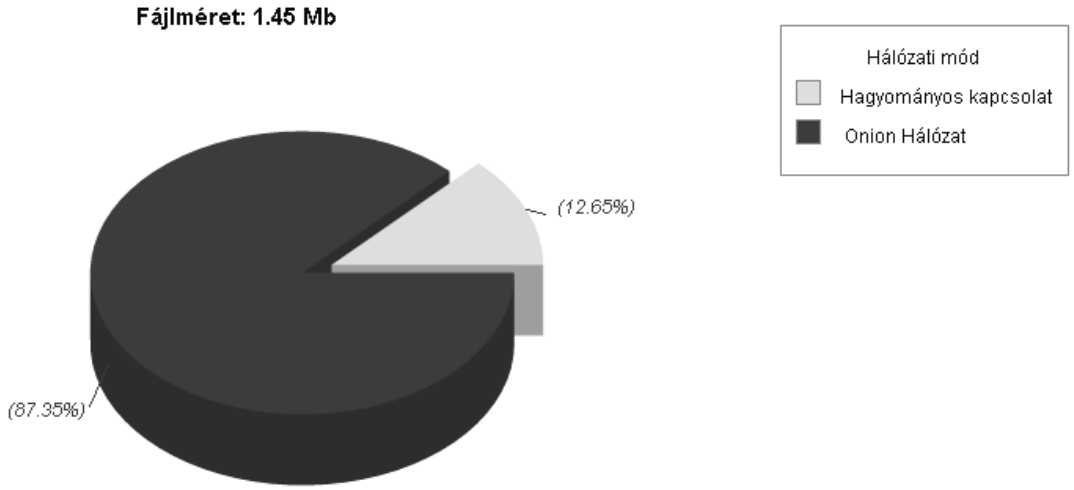
[1] A körgrafikonok az arányok vizuális érzékeltetésére szolgálnak, a 100%-os érték a két összehasonlított érték összegének megfelelője.

hálózaton keresztül mintegy tizenötször hosszabb ideig tartott, mint egy hagyományos kapcsolat esetében.



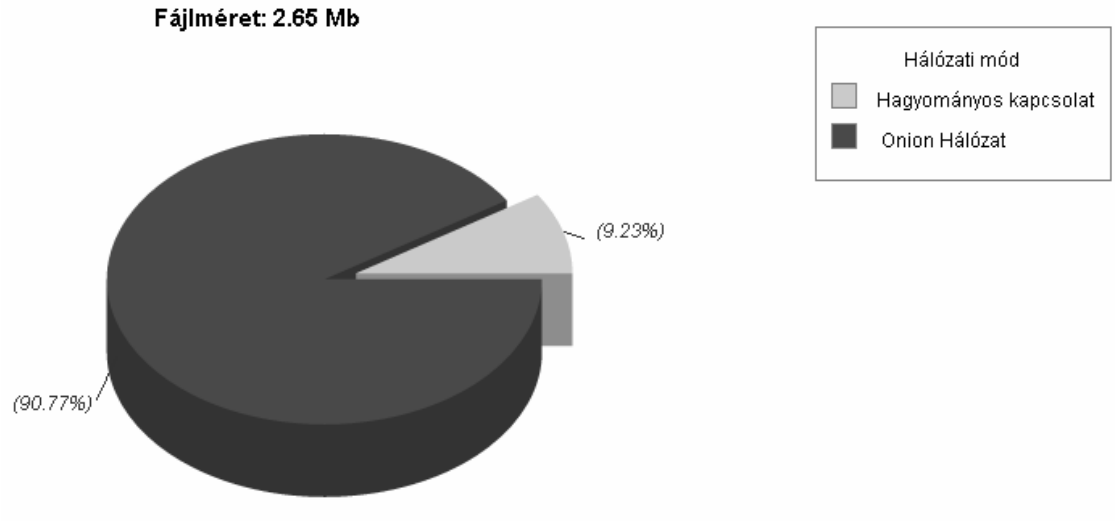
9. ábra: A letöltési idők aránya, hagyományos és anonim kapcsolaton keresztül

A fájlméretet növelve, az 1.45 Mb-os fájl letöltése esetén az eredmények a következőképpen alakultak:



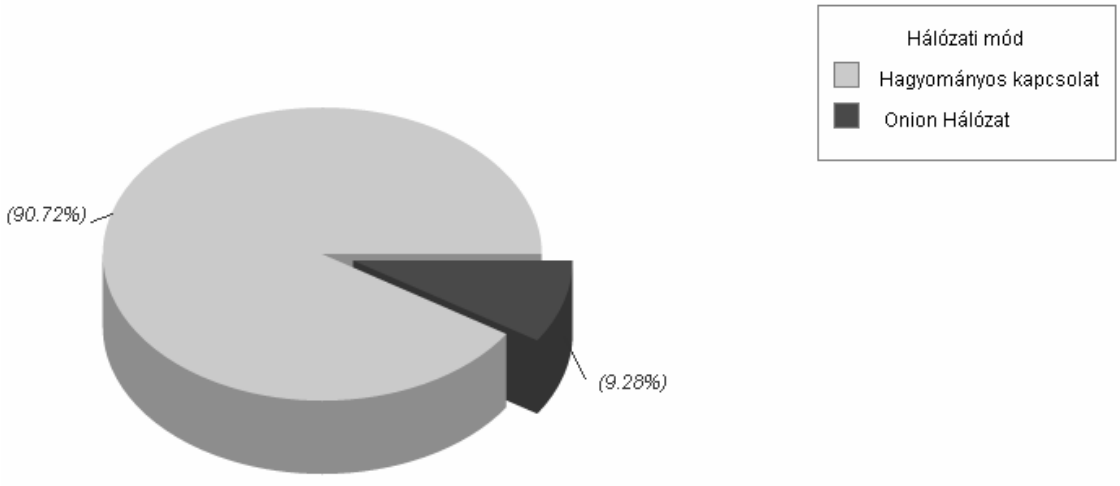
10. ábra: A letöltési idők aránya, hagyományos és anonim kapcsolaton keresztül

A fájlméret növelésével a kommunikációs overhead valamelyest lecsökkent, hiszen az mindösszesen nagyjából hatszoros növekedésről beszélhetünk. A legutolsó tesztelt fájlméret a 2.65 Mb volt. Ebben az esetben a hagyományos, illetve a Tor-ral kapott eredmények közti eltérés nagyságrendileg tízszeres volt, amely megfeleltethető az egyes mért értékek átlagának is.



11. ábra: A letöltési idők aránya, hagyományos és anonim kapcsolaton keresztül

A fájlméretenként kapott letöltési idők kiértékelése után, a hagyományos, illetve az onion-hálózat esetében mért letöltési sebességek átlagát is összehasonlítottam egymással. A kiátlagolt értékek alapján látható, hogy az onion-hálózat esetében mintegy tizedére esett vissza a letöltési sebesség, a hagyományos kapcsolatban mért letöltési sebességekhez viszonyítva.



12. ábra: A letöltési sebességek átlagainak aránya, hagyományos és anonim kapcsolaton keresztül

4.2.2 A teszteredmények értékelése

Amikor az onion routing technológián belüli kommunikációs késleltetési időről beszélünk, a legfontosabb talán mindenekelőtt azt kiemelni, hogy a kommunikáció késleltetési ideje

nem a kommunikáció során alkalmazott titkosítások végrehajtásának idejéből adódik. Az onion-hálózaton belüli kommunikáció késleltetésének okát tehát *nem a kriptográfiai primitívek számításigényében kereshetjük*. A hálózaton belül alkalmazott rejtjelezési módszerek ugyanis javarészt pillanatok alatt végrehajthatóak, a háttér-processzek ezt előlünk, felhasználók elől eltakarják, mi ebből jóformán semmit sem észlelhetünk.

A virtuális áramkörön keresztül felépített kapcsolatban a proxy-k közötti kommunikációt szimmetrikus kulcsú rejtjelezéssel védjük, amely még az aszimmetrikus kulcsú rejtjelezésnél is gyorsabban elvégezhető, mind a kódolási, mind pedig a dekódolási oldalon. Vagyis az onion-hálózatban használt titkosítási eljárások közül a publikus kulcsú, aszimmetrikus kriptográfiai primitívek igényelnék a több számítást, így a titkosítási eljárások közül egyedül a publikus kulcs használata jelenthetne némi időbeli késleltetést, azonban a protokollon belül ez is hatékonyan megvalósítható. Kijelenthető tehát, hogy *az onion-hálózaton belüli titkosítási eljárások egyike sem okozhatja a kommunikáció késleltetési idejének megnövekedését.*

A jelentős hálózati késleltetés okát másban kell keresnünk. *Az onion-hálózaton belüli legjelentősebb késleltetést talán a multi-hop technika használata jelenti. A multi-hop proxy átirányítás során ugyanis a teljes kapcsolatot maszkolni kell két router között, azon közbenső routerek, csomópontok számára, amelyek nem részesei a kommunikáció dedikált útvonalának.*

A Tor tervezői a biztonságos kommunikációt tartották elsődlegesen szem előtt a tervezés során, így a sebesség csupán másodlagos kérdéssé redukálódott a technológia kidolgozása során. A biztonság növelése érdekében például a Tor egy olyan forgalomirányító algoritmust használ, amely a torlódások elhárításakor nem a minél hamarabbi, hanem a lehető *legbiztonságosabb* feldolgozást tartja szem előtt. A gyakorlatban ez azt jelenti, hogy például egy hálózati torlódás esetén az elsődleges cél a DoS elárasztásos támadások megakadályozása, nem pedig a torlódás mihamarabbi megszüntetése. [2] Elmondható, hogy a *rendszer alkotói egyértelműen a biztonságra helyezték a hangsúlyt, így a kommunikáció hatékonysága csupán másodlagos szempont volt a hálózati architektúra megtervezésekor.* Az elvégzett tesztek alapján, egy anonim kapcsolat felépítésének ideje nagyjából 8–10-szer több időbe telt, mint egy hagyományos kapcsolatfelvétel. A működés lassulását a Tor fejlesztői is elismerik, akik a *www.cnn.com* weblapra történő közvetlen csatlakozás idejét tesztelték. Esetükben, a hagyományos, direkt csatlakozás esetében mért 0.3, illetve 0.4 másodperccel szemben, az anonim kapcsolatfelvétel ideje 5.3 másodpercre emelkedett. [2] A hivatkozott tanulmány rámutat arra, hogy a nagyon fájlmérethez, - arányaiban tekintve, kisebb kommunikációs overhead társul, ez azonban még mindig jelentős mértékű. Az előzőekben bemutatott tesztek is alátámasztják ezen megállapítást. A többszörös

átirányítással együttjáró kommunikációs overhead mellett azonban mindenféleképpen „enyhítő körülményként” tekinthetünk arra a tényre, hogy a Tor hálózata folyamatosan és nagy sebességgel növekszik. A hálózat bővülésével pedig párhuzamosan nő annak teljesítménye, a felhasználók által elérhető anonimitási szinttel együtt.

Egy másik érdekessége a hálózatnak, hogy egy onion-hálózaton belüli virtuális áramkörben a kilépési csomópont bárhol elhelyezkedhet. Ezen tény számos, előre nem várt esemény bekövetkezését vonhatja maga után. Ilyen „furcsa” esemény például az, amikor egy globális weboldalt szeretnénk elérni, ehelyett azonban egy lokális weboldalt kapunk. (Például, amikor a lokális google.hu oldalt kapjuk a google.com kérésünkre.) A felmérések alapján azonban a legtöbb felhasználó – főként az USA-ban élőket kiemelve – általában a saját hazájukon belüli weblapokat tekintik meg elsődlegesen, így a *cache szerverek* is földrajzilag nagyságrendekkel közelebb találhatóak ezen felhasználókhoz. A cache szerverek elhelyezkedési jellemzőiből adódóan így már nem meglepő az, hogy jelentős lesz azon weboldal késleltetési ideje, amely például egy Magyarországon található kilépési csomópontból egy amerikai weboldalra irányul.

4.3. A Tor szerver

A Tor kliensként történő használatán túl módunkban áll szerverként is funkcionálni az onion-hálózatban. [3][6] A szerverként való bekapcsolódással egyrészt növeljük az onion-hálózat teljesítményét, másrészt pedig az elérhető anonimitást fokozzuk, vagyis növeljük a rendszer már meglévő biztonsági szintjét. Ahhoz, hogy az onion-hálózat valóban megbízhatóan és eredményesen működhessen, mindenféleképpen szükséges az, hogy számos, egymástól földrajzilag távol elhelyezkedő proxy-t tartalmazzon a hálózat. A szerverként való bekapcsolódással így nem csupán mások anonimitását, hanem saját magunk anonimitását is megnöveljük, összegezve: *a teljes rendszert erősítjük akkor, ha szerverként is bekapcsolódunk az onion-hálózatba.*[1]

A szerver szerep nyilván felveti azt a kérdést, hogy mennyivel lassítja majd le a saját hálózati sebességünket, azaz mennyit veszítünk a sávszélességből azáltal, hogy szerverként is funkcionálunk az onion routing hálózatban. A Tor szerver üzemmódjában már jelenleg is számos olyan funkció áll a rendelkezésünkre, amelynek segítségével jelentősen lecsökkenthető a szükséges sávszélesség. A gyakorlatilag minimális szintre leszorítható követelményeknek, valamint a Tor-hoz kidolgozott hálózatmenedzsmentnek köszönhetően, optimálisnak mondható „áldozatvállalás” mellett csatlakozhatunk be szerverként az onion-hálózatba.

A szerver konfigurációhoz első lépésben a Tor konfigurációs fájlon kell módosítanunk, amely egy egyszerű text fájl. A konfigurálás a Tor weboldalán található instrukciók alapján egyértelműen elvégezhető. A szerver bekonfigurálásához a tűzfal beállításain is módosítani kellett, amely bizonyos esetekben akár nagyon hosszadalmas is lehet, illetve más esetekben akár néhány kattintással is elvégezhetőek a szükséges beállítások. [6] A konfiguráció során a tűzfalban egy „lyukat" kell nyitni, amely azon két portot jelenti, amely a Tor kommunikációjához szükséges. A legtöbb tűzfalnak egyszerű a kezelőfelülete, így a szükséges változtatások könnyedén elvégezhetőek. Miután a konfigurációs fájlt beállítottuk, valamint a tűzfalon is elvégeztük a szükséges beállításokat, a Tor-t újra kell indítanunk. Az újraindítást követően körülbelül 1 percet kell várnunk a szükséges log fájlra, amelyben visszaigazolást kapunk arról, hogy immáron szerverként is funkcionálunk. Ezt követően a *http://belegost.mit.edu/tor/status/authority* weboldalon keresztül leellenőrizhetjük, hogy az aktuális szerverlistában szerepel-e az azonosítónk. A szerverként való üzemelés utolsó lépéseként elektronikus formában is értesítenünk kell a Tor operátorait, miután regisztrálják a saját nick-nevünket, amelyen keresztül esetlegesen kapcsolatba léphetnek velünk a hálózat adminisztrátorai.

4.3.1. A feltelepítés tanulságai

A Tor weboldalán található utasítások követésével egyszerűen, néhány percen belül válhatunk magunk is Tor klienssé, vagy pedig Tor szerverré – ekkor pedig kevesebb, mint egy órán belül. A Tor kliens feltelepítés meglehetősen egyszerű mechanizmusa után a szerver feltelepítése következett. A szerver feltelepítse már valamennyivel bonyolultabbnak mondható. A teszt végeredményeképpen elmondható, hogy a Tor használata valóban olyan egyszerű, mint ahogyan azt a fejlesztők hirdetik, azaz nem csupán egy marketingfogásról van szó.

4.4. Torpark

A *Torpark* a *http://www.torrify.*com weboldalon keresztül tölthető le, egy 9.3 Mb-os csomagon belül. A letöltés után a böngészőt telepíthetjük a számítógépünk merevlemezére, de akár egy hordozható usb tárolóra (pendrive-ra) is. [8] Elsőként a winchesterre telepítettem a böngészőt, majd a böngésző működési sebességét letesztelteM az usb-n keresztül futtatva is.

4.4.1. A Torpark működése

A Torpark kifejlesztése a *Hacktivismo* fejlesztőcsoport nevéhez fűződik [5], e csoport tagjai főként biztonsági szakemberek. A Torpark gyakorlatilag egy olyan speciális Firefox

kiegészítés, amely beépített Tor támogatást tartalmaz. A Torpark célja a kommunikáló felek anonim adatainak megőrzése, valamint a nyomkövetési eljárások kivédése. [5][10] A fejlesztőcsoport a Torparkot elsődlegesen hordozható tárolókra szánta, így elsősorban usb-re. A böngésző azonban a számítógépünk merevlemezéről is futtatható. A Torpark tehát szintén az onion routing technológiát használja. Azaz a Torpark használatakor is, a routerek mindössze annyi információval rendelkeznek a kommunikációval kapcsolatban, hogy melyik másik onion routertől kapták a csomagot, valamint hogy melyik másik routernek kell továbbadni azt. Az elérhető biztonsági szintet így ugyanúgy a hálózati útvonalon lévő router láncolat legerősebb láncszemével jellemezhetjük.

Egy hagyományos hálózaton belül a felhasználó azonosításának leginkább alkalmazott módszere az IP-cím regisztrációja, tárolása. A Torpark az onion-hálózat segítségével néhány percenként megváltoztatja a számítógépünk IP-címét, így lenyomozhatatlan és nyomonkövethetetlen az, hogy éppen hol tartózkodik az adott felhasználó. [8] A letöltött adat a Tor-szerverig titkosítva kerül átvitelre, a Tor-hálózatról a meglátogatott weboldal felé ez azonban már nem igaz. A Torpark ugyanúgy, mint a Firefox-hoz felinstallált Tor kliens illetve Tor szerver konfiguráció, természetesen számos etikai kérdést is felvet. Ezzel azonban sajnos számos egyéb program, illetve alkalmazás esetén is számolnunk kell, ami ellen a védekezés gyakorlatilag lehetetlen. A két szélsőséges álláspont között kellene egy arany középutat találnunk, miszerint vagy legyen elérhető mindenki számára, vagy pedig kerüljön a program teljes mértékben tiltólistára, minek következtében mindenki elől elzárnánk a program használatának lehetőségét.

4.4.1.1. A Torpark indítása

A Torpark az indítást követően több percig kereste az anonimitást biztosító Tor-hálózatot, és a beállított nyitóoldal megjelenítése is körülbelül 2.5 percig tartott. Vagyis már a merevlemezről történő futtatás legelején megállapítható, hogy a böngésző indulási sebessége jelentősen eltér a Firefox sebességétől. Mivel a kezdőoldalként beállított *mail.google.com* titkosított kapcsolatot használ, a Torpark erre egy üzenetben figyelmeztet már az induláskor. A Torpark az SSL 2.0 illetve 3.0-ás verzióját támogatja, valamint az újabb fejlesztésű TLS 1.0-ás verzióját. [5] A böngészőprogramban alapbeállításként mindhárom titkosítási eljárást igénybe vehetjük. A Torparkon belüli *IP-spoofing*, azaz az IP-cím hamisító funkció az indításkor azonnal működésbe lép. A Torpark ugyanis informál minket arról is, hogy éppen milyen IP címen keresztül „lát” bennünket az adott weboldal, amely természetesen nem azonos az eredeti IP-címünkkel. [5][8]

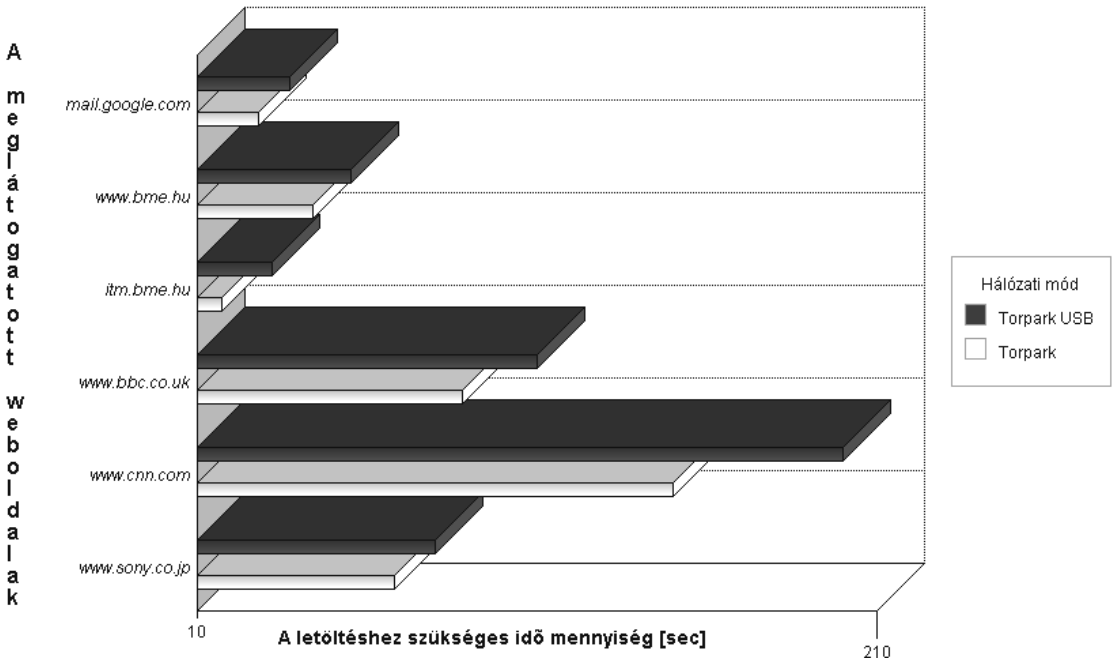
4.4.2. Futtatás USB tárolóról

Az usb-ről történő futtatás során elsőként a böngésző kezdeti betöltődésének idejét, majd az egyes weboldalak betöltési sebességeit vizsgáltam, illetve hasonlítottam össze. Az usb meghajtón keresztül nyilvánvalóan csak alacsonyabb sebességgel tudjuk elérni a kívánt tartalmakat – ez az egyik ára az anonimitásunk teljes megőrzésének. A Torparkkal sikerült megtekintem minden lekért weboldalt, azaz a Torparkot böngészőként használva, a meglátogatható weboldalak köre nem szűkült. A következő táblázatban a meglátogatott weboldalak során mért letöltési időket foglaltam össze.

3. Táblázat: Letöltési idők alakulása

Meglátogatott oldal	Torpark [sec]	Torpark usb [sec]
http://mail.google.com	28	37
www.bme.hu	43	55
http://itm.bme.hu/	17	32
http://www.bbc.co.uk	88	112
http://www.cnn.com/	154	201
www.sony.co.jp	68	81

A mért eredményeket grafikus formában is megtekinthetjük a következő ábrán.

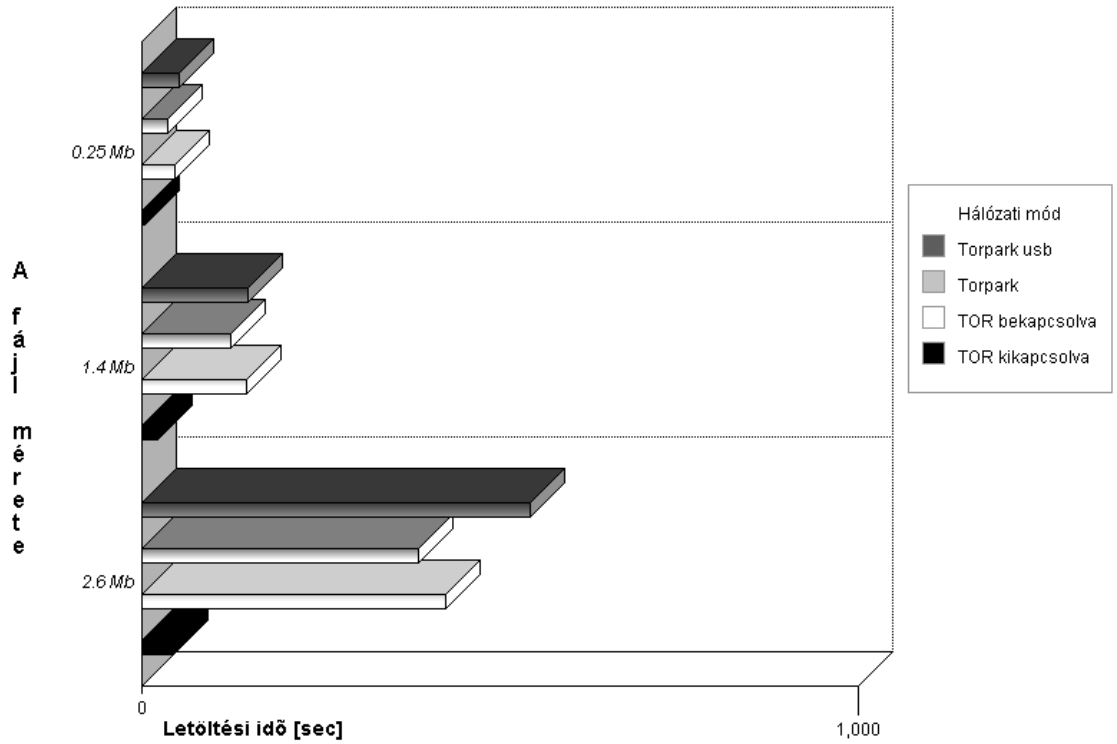


13. ábra: Az egyes weboldalak letöltési ideje

Az eredmények alapján elmondható, hogy a Torpark usb-ről futtatott változata körülbelül 30–35%-kal lassabb a merevlemezről futatott változatánál. A megtekintett weboldalak során a letöltési idők alakulása meglehetősen sztochasztikusnak nevezhető, így a közölt eredmények a hálózat egy adott időpillanatban értelmezendőek, amelyekhez képest természetesen lehetnek kisebb-nagyobb eltérések. Így például egy későbbi időpontban elvégzett mérés eredménye már nagy valószínűséggel nem egyezik pontosan ezen értékekkel. A mérések eredménye elsősorban ugyanis a kiszolgáló hálózat aktuális terheltségétől függ.

5. A teszteredmények összegzése

A vizsgálatok során összesen négy különböző módon teszteltem a hálózati kapcsolat sebességét. Elsőként egy „hagyományos", anonimitást kevésbé biztosító kapcsolat esetén mértem a különböző fájlméretek letöltéséhez szükséges időt. A második mérési sorozatban a Mozilla FireFox kiegészítő plugin-ként feltelepített Tor klienssel, a harmadik és negyedik körben pedig a Torpark nevű böngészővel teszteltem ugyanezen fájlok letöltési idejét, illetve a weboldalak letöltési sebességeit. A mérések során kapott eredményeket a következő grafikonon foglaltam össze:



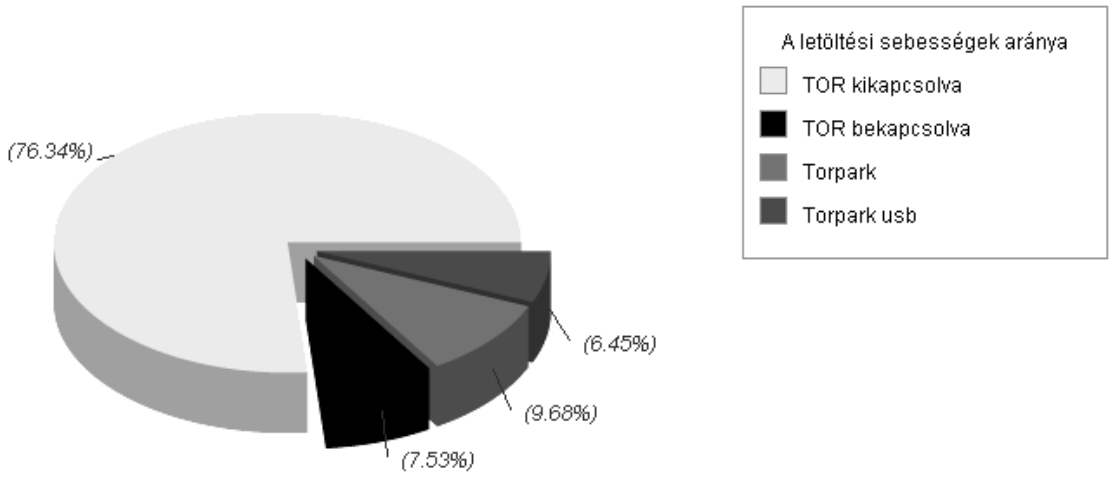
14. ábra: Eredmények összegzése

A kapott eredmények alapján megállapítható, hogy az egyes fájlokhoz szükséges letöltési idők a fájlmérettel egyenes arányban, lineárisan növekedtek. A leggyorsabb letöltést a hagyományos kapcsolat során mértem, amely természetesen megfelel előzetes várakozásainknak. A hagyományos kapcsolatot a merevlemezről futtatott Torpark böngésző követi, amelye a többi tesztelt programhoz képest a leggyorsabbnak bizonyult. A merevlemezről futtatott Torparkot a Firefox alá telepített Tor kliens követi, amelynek teljesítménye mindössze 10–12%-al maradt el a Torpark mögött. A teljesítményanalízis negyedik helyezettje pedig a Torpark usb-ről futtatható verziója lett, amely a merevlemezes változathoz viszonyítva kb. 35%-kal bizonyult lassabbnak, ami a technikai paraméterek figyelembevételével magyarázható. (Az usb-drive sebessége nagyságrendekkel elmarad a winchester sebessége mögött.) A különböző fájlméretek során mért sebességeket foglaltam össze a következő táblázatban, a négy tesztelt böngészési módra vonatkozóan:

4. Táblázat: Átlagos letöltési sebességek

Fájl mérete (Mb)	Hagyományos kapcsolat (Kb/sec)	Tor (Kb/sec)	Torpark (Kb/sec)	Torpark usb (Kb/sec)
0.26	83.00	5.70	7.72	4.60
1.40	70.00	10.00	11.70	8.60
2.60	62.00	6.30	7.00	4.88
Átlagsebesség (kbyte/sec)	*71.67*	*7.33*	*8.81*	*6.03*

A mért sebességek átlagainak egymáshoz viszonyított arányát a következő diagrammon láthatjuk:



15. ábra: A letöltési sebességek átlagainak aránya

A tesztelt, onion-hálózatot használó kliens, illetve böngésző programok közül a hagyományos hálózat sebességét leginkább megközelítő teljesítményt a merevlemezről futtatott Torpark esetén mértem, a leglassabb letöltést, illetve böngészést pedig az usb-ről futtatott változattal. Az elvégzett mérések eredményei alapján, a Firefox alá telepített Tor kliens teljesítménye pedig a Torpark merevlemezről, illetve usb-ről futtatott változatainak teljesítménye között helyezkedik el.

6. Összefoglalás

A titkos és anonim kommunikáció mellett a legtöbb felhasználót talán az elérhető sebesség érdekli leginkább, hiszen egy átlagos felhasználó amellett, hogy szeretne meggyőződni kommunikáció biztonságáról, egy elfogadható sebességet is elvár. Az átlag felhasználókat azonban kevésbé foglalkoztatja az, hogy az abszolút anonimitás, illetve a biztonsági szint növelése milyen módszerekkel kerül megvalósításra. Egy átlag felhasználó így elsődlegesen azt a kérdést teszi majd fel, hogy a Tor mennyire lassítja le az eddigi hagyományos, (viszont kevésbé biztonságos és anonim) kapcsolatom sebességét? Mennyit kell majd fizetnem a sebességben, a megnövekedett biztonságért? Megéri-e ez nekem? Egy hétköznapi felhasználóban ezek a kérdések fogalmazódnak meg elsőként, s e kérdésekkel természetesen számolniuk kellett a Tor fejlesztőinek is.

A tanulmányban kapott mérési eredmények alapján kijelenthető, hogy a fokozott biztonságért és anonimitásért az elérhető teljesítménnyel kell fizetnünk, méghozzá igen jelentős mértékben. Egy átlagos weboldal, amelynek elérési ideje „hagyományos úton”, azaz a Tor kliens bekapcsolása nélkül 1–2 másodperc, a Tor kliens bekapcsolásával 5–10 másodpercre növekedett.[2] Ez az eredmény elsőre igen lesújtónak tűnik, azonban a Tor dinamikus kezelhetősége valamelyest enyhít ezen, hiszen egyetlen kattintással ki-be kapcsolhatjuk, így a teljes kommunikáció alatt nem kell Tor-t használnunk. A kapott eredmények tükrében, az anonimitásunkat biztosító Tor-t, illetve a kiegészítő plugin-eket, akkor érdemes igazán használnunk, ha arra valóban nagy szükségünk van, és a kapcsolatunk sebessége másodlagos szempont csupán.

A Tor, s ezáltal az onion-hálózatot használó anonim kommunikáció elterjedésének egyik legfontosabb tényezője a használatához szükséges felhasználói programok egyszerű kezelhetősége, valamint azok egyszerű konfigurálhatósága. A Tor felépítésből adódóan, az onion-hálózat felhasználói körének növekedésével együtt, annak teljesítménye és hatékonysága is párhuzamosan emelkedik. A hatékonyság növekedése mellett ugyanakkor az egyre több proxy megjelenése a hálózaton belül további biztonságot injektál az onion-hálózatba. A mérések során kapott eredmények ellenére az onion technológia fejlődése

rendkívül bíztató, hiszen egy átlagos felhasználó számára – a napjainkban rohamos sebességgel növekvő és egyre szélesebb körben elérhető szélessávú internet kapcsolatnak köszönhetően – az onion routing rendszerek által nyújtott hatékonyság hamarosan teljes mértékben megegyezik egy hagyományos, ugyanakkor anonimitásunkat egyáltalán nem biztosító hálózat teljesítményével.

Irodalomjegyzék

[1] D. Goldschlag, M. Reed, and P. Syverson: Hiding routing information. Workshop on Information Hiding, 1996.
http://www.onion-router.net/Publications/IH-1996.pdf

[2] R. Dingledine, N. Mathewson, and P. Syverson: Tor: The second-generation onion router.
USENIX Security Symposium, 2004.
http://citeseer.ist.psu.edu/dingledine04tor.html

[3] http://tor.eff.org/

[4] https://addons.mozilla.org/firefox/2275/

[5] http://www.torrify.com/

[6] http://www.onion-router.net/

[7] http://www.showmyip.com/

[8] [CMU Usable Privacy and Security Laboratory: Foxtor: A tor design proposal
http://cups.cs.cmu.edu/pubs/TorGUIContest113005.pdf.

[9] Paul Syverson: Making anonymous communication.
http://www.onion-router.net/Publications/Briefing-2004.pdf ; 2004.

[10] Paul Syverson, Gene Tsudik, Michael Reed, and Carl Landwehr: Towards an analysis of onion routing security. Workshop on Design Issues in Anonymity and Unobservability, 2000.