Darknets (Freenet, Tor, I2P)

Nils Schmidt
Philipps-Universität Marburg, Deutschland
Fachbereich Mathematik und Informatik, Verteilte Systeme
E-Mail: schmidt89 at informatik.uni-marburg.de
Februar 2013

***Zusammenfassung*—Das Internet hat in den vergangenen Jahren immer mehr an Bedeutung erlangt. Jedoch ist man im Internet nicht anonym, denn jeder Nutzer ist durch seine IP-Adresse identifizierbar. Des Weiteren ist die Tatsache, dass zwei Parteien miteinander kommunizieren auch kein Geheimnis. Außerdem ist es nicht für jeden frei zugänglich, weil zensierende Staaten wie China Teile des Internet blockieren.**
Für diese Zwecke u.a. existieren Systeme mit denen sowohl Zensur umgangen werden kann als auch anonyme Kommunikation ermöglicht wird. Mit Hilfe solcher Darknets kann die Identität, je nach System, entweder im regulären Internet oder in einem in sich geschlossen Netzwerk, verschleiert werden.
Meist wird dafür eine Kaskade von Knoten aufgebaut, über die die Kommunikation dann stattfindet. Der letzte Knoten nimmt dann Kontakt mit dem Ziel auf. Dabei kennt jeder Knoten nur seinen vorhergehenden und nachfolgenden Knoten.
Das Ziel dieser Ausarbeitung ist die Darknets Freenet, Tor und I2P vorzustellen. Auf die Sicherheit sowie mögliche Angriffsszenarien, die die Schwachstellen des jeweiligen Darknet ausnutzen, um die Identität eines Nutzers aufzudecken, wird nicht eingegangen.

I. Einleitung

Darknet ist die Kurzform für 'dark internet'. Gemeint ist damit ein Netzwerk in dem Informationen anonym von Teilnehmern des Darknet geteilt werden [1].
Solche Darknets können zum Beispiel dafür genutzt werden, um Zensur zu entgehen oder, um die Tatsache zu verschleiern, dass zwei Parteien überhaupt miteinander kommunizieren.
Laut einem Bericht aus dem Jahr 2011 der Organisation Reporter ohne Grenzen wird jedem Dritten der freie Zugang zum Internet verwehrt [2].
Eines der bekanntesten Beispiele für zensierende Staaten ist China. Mit der *Great Firewall of China* werden Seiten explizit gesperrt sowie TCP-Verbindungen zurückgesetzt, wenn IP-Pakete bestimmte Stichworte enthalten [3].
Weitere Beispiele für Staaten in denen Zensur herrscht sind der Iran, Kuba, Nord Korea, Saudi Arabien, Syrien, Vietnam etc. Sie werden von Reporter ohne Grenzen sogar als Feinde des Internet bezeichnet [2].
Allerdings darf man nicht vergessen, dass Darknets nicht nur dafür benutzt werden, um in zensierenden Staaten freien Internetzugang zu erhalten, sondern auch von Kriminellen für ihre Zwecke missbraucht werden kann, in dem sie anonym bleiben.
Ein weiteres sehr sensibles und bekanntes Thema ist Kinderpornographie. Mit den Mitteln anonymer Kommunikation können ungehindert solche Inhalte ausgetauscht werden ohne eine wirkliche Kontrolle oder Möglichkeiten zu haben, solche Inhalte zu unterbinden.
Im Folgenden werden drei Darknets im Detail beschrieben. Diese sind Freenet, Tor und I2P.

II. Freenet

Freenet ist ein verteiltes System zur Speicherung und zum Abrufen von Informationen. Die ursprüngliche Idee stammt von dem irischen Studenten Ian Clarke (Universität Edinburgh).
Freenet ist ein Peer-to-Peer-Netzwerk von Knoten, welche die Freenet Software installiert haben. Jeder Knoten hat einen Datenspeicher auf der eigenen Festplatte, der dem Netzwerk für Lese- sowie Schreibzugriffe zur Verfügung gestellt wird.
Auch wenn es für die Speicherung von Daten geschaffen wurde, garantiert es jedoch keine permanente Speicherung von Daten. Das System arbeitet auf der Anwendungsschicht und bietet nur Anonymität für Dateiaustausch innerhalb des Netzwerkes.
Jedem Knoten wird, wenn er dem Netzwerk beitritt, zufällig eine *Location* aus dem Intervall [0,1) zugewiesen. Daten werden ebenfalls mit einem Wert aus diesem Intervall verknüpft [4] und sind durch einen Schlüssel identifiziert (siehe Sektion II-F).
Des Weiteren verwaltet jeder Knoten eine dynamische Routingtabelle, die eine Zuordnung von Knoten zu Daten beinhaltet. Somit ergibt sich eine *Distributed Hash Table* und ein dezentrales Netzwerk.
Im Prinzip durchläuft eine Anfrage nach einer Datei eine Kette von Proxy-Anfragen, wobei jeder Knoten eine Entscheidung trifft, welcher Nachfolgeknoten der geeignetste Knoten auf dem Weg zum Zielknoten ist. Der Zielknoten ist Derjenige, der die Datei besitzt. Jeder Knoten in dieser Kette kennt nur seinen vorhergehenden und nachfolgenden Knoten. Analog zur TTL (time to live) des IP-Protokolls gibt es einen *HTL*-Wert (hops to live), der bei jedem Knoten dekrementiert wird. Somit können keine unendlich langen Ketten aufgebaut werden.
Um Schleifen innerhalb des Netzwerkes zu verhindern, wird jeder Anfrage eine ID zugewiesen. Dadurch können bereits weitergeleitete Anfragen verworfen werden [5].
Um eine Verbindung zum Freenet herzustellen, benötigt man

eine oder mehrere Adressen bereits aktiver Knoten (*seednodes*) im Freenet [6].

A. Opennet & Darknet

Mit der Version 0.7 wurde die Entwicklung des Projekts drastisch geändert. Es kam die Idee auf, ein *Friend-to-Friend*-Netzwerk zu schaffen, das von den Entwicklern als *Darknet* bezeichnet wird.
Das bedeutet, dass die Kommunikation innerhalb des Freenet auf Freunde (*trust peers*) beschränkt wird, aber Daten immer noch global verfügbar sind. Dieses Modell erhöht unter der Annahme, dass Freunde auch wirklich vertrauenswürdig sind, die Sicherheit enorm. Allerdings leidet die Benutzerfreundlichkeit hierunter, denn, um sich mit dem Netzwerk zu verbinden, muss ein Benutzer mindestens eine vertrauenswürdige Person kennen, die sich ebenfalls im Freenet befindet.
Es gibt allerdings auch ein Hybrid-Modell, in dem Knoten, die sich im Darknet befinden, ihre Identität auch an Unbekannte weitergeben. Das sogenannte *Opennet* [6].

B. Daten herunterladen

Dateien können aus dem Freenet heruntergeladen werden, indem ein Request an den lokalen Knoten gestellt wird. Diese Forderung muss den Schlüssel, der die Datei identifiziert sowie einen HTL-Wert enthalten.
Jeder Knoten, der diesen Request empfängt, überprüft zunächst ob der eigene Datenspeicher die geforderte Datei bereits enthält und liefert sie ggf. zurück. Zusätzlich wird die Information weitergegeben, dass der Knoten die Quelle der Datei ist. Wenn die Datei jedoch nicht gefunden wurde, so wird in der Routingtabelle der Knoten herausgesucht, der einem ähnlichen[1] Schlüssel zugeordnet ist und der Request wird an diesen weitergeleitet.
Wird z.B. der Schlüssel mit der Location 0,4 gesucht und die Routingtabelle enthält nur die beiden Schlüssel 0,6 und 0,7, so wird der Knoten ausgewählt, dem der erste Schlüssel zugeordnet ist.
Kann dieser Knoten den Request erfolgreich beantworten, so wird die Datei an den vorhergehenden Knoten weitergegeben. Dieser speichert die Datei zwischen und erstellt einen Eintrag in seiner Routingtabelle, in der die Quelle der Datei dem gesuchten Schlüssel zugeordnet wird. Diesen Vorgang durchläuft jeder Knoten in der Kette, bis die Datei schließlich an den ursprünglichen Requester weitergeleitet wurde.
Wenn ein Knoten keinen Kandidaten mehr zum Weiterleiten hat oder eine Schleife entstanden ist, wird ein Misserfolg via Backtracking zurückgemeldet. Man beachte, dass diese Knoten nicht Teil des etablierten Pfades sind. Der vorherige Knoten leitet dann ebenfalls an den nächst näheren[1] Knoten weiter etc.
Wenn der HTL-Wert auf null gesunken ist, wird ebenfalls ein Scheitern zum ursprünglichen Requester zurückgemeldet. Die Suche ist beendet und es wird kein weiterer Versuch unternommen, einen anderen Knoten zu kontaktieren.

[1]Gemeint ist der Abstand von Location zu Location oder von Location zu Schlüssel, wobei der Abstand zwischen 0,1 und 0,9 0,2 ist.

Das Anlegen einer Routingtabelle und somit einer Tabelle von Knoten, die angibt auf welchem Knoten welche Datei gespeichert ist, stellt ein Sicherheitsrisiko dar. Damit jeder Teilnehmer des Freenet bestreiten kann, dass gewisse Inhalte auf dem eigenen Computer gespeichert sind, kann jeder Knoten behaupten, dass er die Quelle der Datei ist.

C. Daten hochladen

Das Hochladen von Dateien ist ähnlich dem Herunterladen von Dateien. Im Folgenden bezeichne *Inserter* den Knoten, der eine neue Datei in das Freenet einfügen möchte.
Beim Hochladen von Dateien werden wieder Schlüssel sowie ein HTL-Wert benötigt, der jetzt aber angibt, auf wie vielen Knoten die Datei maximal gespeichert werden soll.
Bekommt ein Knoten eine solche Anfrage, so wird wieder zuerst der eigene Datenspeicher daraufhin überprüft, ob die Datei bereits vorhanden ist. Ist dies der Fall wird die gefundene Datei zurückgeliefert und der Knoten, der die Datei hochladen wollte, bemerkt, dass eine Schlüsselkollision vorliegt. Es muss ein anderer Schlüssel gewählt werden. Zu einer Kollision kommt es allerdings nur, wenn der Schlüssel aus einer beschreibenden Zeichenkette (*Keyword Signed Key*) berechnet wird.
Andernfalls wird nach dem eben beschriebenen Routingalgorithmus fortgefahren als wäre ein Request nach der einzufügenden Datei gemacht worden. Wenn keine weitere Kollision auftrifft, wird die Kette solange weiter ausgebaut, bis der HTL-Wert auf null herabgesetzt wurde. Jetzt wird eine Nachricht zurück an den Inserter über die etablierte Kette geschickt, die besagt, dass die Datei hochgeladen werden kann. Entlang des Pfades wird auf jedem Knoten die Datei repliziert und ein Routingeintrag erzeugt, in dem der Inserter mit dem Schlüssel der Datei verknüpft wird. Auch hier kann jeder Knoten behaupten, dass er der Inserter der Datei ist.

D. Clustering

Der soeben beschriebene Mechanismus verbessert über die Zeit die Qualität des Routings. Denn einerseits spezialisieren sich Knoten auf das Auffinden ähnlicher[1] Schlüssel und andererseits speichern sie Dateien mit ähnlichen[1] Schlüsseln. Der Grund dafür ist, dass wenn ein Knoten für einen bestimmten Schlüssel in der Routingtabelle gelistet ist, er auch Anfragen mit ähnlichen[1] Schlüssel erhält und sowohl die Dateien, die über diese Kette transferiert werden speichert, als auch neue Einträge für sein Routing erstellt und somit besser Bescheid weiß, zu welchen Knoten er weiterleiten soll [5].

E. Speichersystem

Alle Dateien, die im Freenet gespeichert sind, liegen verteilt in den Datenspeichern der Knoten. Da Speicherplatz jedoch endlich ist, müssen alte Daten gelöscht werden, um Platz für Neue zu schaffen. Dies geschieht nach dem Prinzip LRU (Least Recently Used). Also werden die am längsten nicht

verwendeten Dateien verdrängt, bis genug Platz zur Verfügung steht.
Einträge in der Routingtabelle werden in ähnlicher Art und Weise gelöscht. Allerdings bleiben diese länger erhalten, damit die Möglichkeit besteht, erneut eine Kopie der gelöschten Dateien zu erstellen [5]. Dateien werden, je nachdem ob sie beliebt sind, behalten oder gelöscht. Als Benutzer hat man quasi nicht die Möglichkeit zu entscheiden, was in dem eigenen Datenspeicher abgelegt wird. Dies ist so gewollt, damit Freenet zensurresistent ist. Daten können nur aus dem Netzwerk entfernt werden, indem nicht nach ihnen gesucht wird.
Des Weiteren werden die Dateien mit einem 256-Bit langen symmetrischen Schlüssel [7] (Rijndael [8]) verschlüsselt. Betreiber eines Freenet Knoten können auf diese Weise bestreiten, dass sich eine gewisse Datei auf dem Rechner befindet [5].

F. Schlüssel

Um Dateien aus dem Freenet herunterzuladen, benötigt man deren Schlüssel. Die zwei wichtigsten Schlüssel sind hierbei *Content-Hash Keys* (CHK) und *Signed-Subspace Keys* (SSK). Diese werden ermittelt, indem ein Hash mittels SHA-256 [9] berechnet wird [10]. CHKs werden meist für statische Daten und SSKs für dynamische Inhalte wie z.B. Websites genutzt. [7].

1) CHK: Content-Hash Keys bestehen aus drei Teilen [7]. Einerseits aus einem Hash, der aus dem Inhalt der Datei kalkuliert wird [10], einem Schlüssel, mit dem man die Datei entschlüsseln kann und schließlich mit der Angabe, welche kryptographischen Einstellungen dafür nötig sind.

2) SSK: Signed-Subspace Keys werden hauptsächlich für eine indirekte Speicherung benutzt. Dabei funktionieren sie als eine Art Container zur Aufbewahrung von mehreren Dateien, indem auf deren CHKs verwiesen wird. Außerdem werden sie dafür genutzt, um Dateien, die unter einem SSK veröffentlicht wurden, zu aktualisieren [10].
Mit Hilfe der Public-Key Kryptographie [7] entsteht ein persönlicher Namensraum, den jeder lesen, aber nur der Besitzer beschreiben kann [10]. Hierfür wird zunächst ein Paar bestehend aus einem öffentlichen sowie einem privaten Schlüssel generiert (DSA 2048 Bit [8]).
Der Private Schlüssel wird zum Signieren der Datei genutzt und nicht weitergegeben [10]. Der öffentliche Schlüssel wird mit der Datei veröffentlicht [7] und dient zur Überprüfung der Integrität des Inhalts.
Zum Aktualisieren einer Datei in einem SSK, wird diese zunächst unter einem CHK veröffentlicht. Daraufhin wird der SSK so aktualisiert, dass er auf die neue Version zeigt. Dafür ist wieder der private Schlüssel notwendig, denn der Schlüssel muss erneut signiert werden.
Die aktualisierte Version ist also unter dem SSK und die alte Version unter dem CHK der alten Datei einsehbar.
Außerdem werden Signed-Subspace Keys dafür benutzt, um große Dateien hochzuladen, indem sie in mehrere Teile aufgespalten werden, die dann seperat under einem Content-Hash Key veröffentlicht werden [10].

G. Darknet

1) Small-World Network: Um ein effizientes Routing für das Darknet zu finden, betrachtet man zunächst den Graph, der durch die Verbindungen unter den trust peers entsteht. Dieser Graph kann nicht für das Routing verbessert werden, denn wenn jeder Knoten nur Verbindungen zu vertrauenswürdigen Parteien aufbaut, dann ist der Graph durch diese festgelegt.
Das heißt, dass die Effizienz des Routings allein von der Struktur des Graphen abhängt, der die sozialen Strukturen widerspiegelt.
Es wird angenommen, dass dieser ein sogenanntes *Small-World Network* darstellt [4]. Denn in einem solchen Netzwerk gibt es Routen zwischen zwei Knoten, die als klein angesehen werden. Dabei meint klein mindestens logarithmisch [4]. Außerdem ist es wahrscheinlicher, dass sich Knoten kennen, wenn sie einen gemeinsamen Freund haben [6]. Das beste Beispiel für ein Small-World Network sind soziale Netzwerke [4].

2) Location Switching: Ziel des Algorithmus ist es, die Zuweisung der Locations so zu verändern, dass trusted peers auch ähnliche[1] Locations bekommen. Um dies zu bewerkstelligen, wird ein zufälliger Pfad durch das Netzwerk gewählt, bei dem jedoch maximal sechs Knoten besucht werden [6].
Der Ausgangsknoten heiße Alice und der Endknoten Bob. Man berechnet das Produkt D_1 der Abstände von Alice zu seinen Nachbarn multipliziert mit dem Produkt der Abstände von Bob zu seinen Nachbarn. Das gleiche Produkt D_2 berechnet man für den Fall, dass die beiden Knoten ihre Location tauschen. Alice und Bob tauschen genau dann ihre Location, wenn $D_1 \geq D_2$. Folglich wird der durchschnittliche Abstand zu den Nachbarknoten verringert [4].

H. Evaluation

Bei Freenet ist es durch *Harvesting* trotz der dezentralen Struktur möglich die IP-Adressen der Teilnehmer herauszufinden. Dabei wird ausgenutzt, dass eine Anfrage nach einer Datei die mögliche Quelle verrät und jeder Knoten eine Routingtabelle verwaltet. Somit kann durch Einschleusen eines feindlichen Knoten die Adressen der Freenet Knoten gesammelt werden und folglich der Zugang unterbunden werden, so wie es China im Jahr 2005 gezeigt hat [11]. Harvesting in Freenet ist jedoch nur im Opennet möglich, da die Verbindungen im Darknet fest sind [12].
Freenet hat außerdem den Vorteil, dass Dateien, wenn sie einmal hochgeladen wurden, auch noch verfügbar sind, wenn der ursprüngliche Besitzer nicht mehr online ist. Zumindest, wenn die Datei populär genug ist. Denn Dateien, die nicht angefordert werden, verschwinden mit der Zeit aus dem Netzwerk. Folglich kann seine Anonymität nachdem er die Datei hochgeladen hat, nicht mehr enthüllt werden.

III. Tor

Mit Tor wird das 2^{nd} Generation Onion Routing bezeichnet. Seinen Ursprung hat es beim Naval Research Laboratory

(USA), wo es u.a. durch einen der Erfinder (Paul Syverson) des ursprünglichen Onion Routings entwickelt wurde [13].

A. Architetkur

Tor ist ein Overlay-Netzwerk, das TCP-Verbindungen anonymisiert. Dafür wird das SOCKS Proxy Interface benutzt, um möglichst viele Applikationen ohne Modifikation unterstützen zu können [14].
Eine Verbindung durch das Tor-Netzwerk wird über mindestens drei Knoten abgewickelt [15].
Will Alice nun eine anonyme Verbindung zu Bob aufbauen - Bob befindet sich nicht im Tor-Netzwerk - so wählt Alice Router aus, die sich bereits im Tor-Netzwerk befinden. Alice etabliert nur einen *Circuit* durch diese Knoten. Nachrichten die an Bob geschickt werden sind schichtweise mit Session Keys verschlüsselt, die Alice im Vorhinein mit den Knoten ausgehandelt hat. Jeder dieser Knoten kann mit seinem Schlüssel eine Schicht der Nachricht abtragen. Daher kommt auch der Begriff des *Onion Routing*.
Hat der letzte Knoten in dem Circuit die Nachricht entschlüsselt, so sieht er diese in seiner ursprünglichen Form. Die Nachricht ist also für diesen lesbar, falls die Kommunikation von Alice nicht verschlüsselt wurde.
An dieser Stelle verlässt die Nachricht das Tor-Netzwerk, denn sie wurde von dem letzten Knoten an Bob weitergeleitet. Antworten von Bob werden ebenso schichtweise verschlüsselt und durch den gleichen Circuit geleitet.
Die Adressen für die Router zum Aufbau des Pfades werden von einem *Directory Server* heruntergeladen. Die Directory Server sind dem eigenen Knoten bekannt, da diese bei der Installation der Tor-Software mitgeliefert werden.
Nicht jeder Knoten erlaubt Verbindungen zu allen Adressen und Ports. Diese *Exit Policies* wurden eingeführt, um möglichst viele Leute dazu zu bewegen einen Exit Node zu betreiben und Ihnen gleichzeitig die Möglichkeit zu bieten, bestimme Protokolle oder an gezielte Adressen nicht weiterzuleiten, um den eigenen Exit Node vor Missbrauch zu schützen. Des Weiteren bietet Tor mit *Hidden Services* die Möglichkeit, Dienste anonym innerhalb des Netzwerkes anzubieten [14].

B. Onion Cells

Knoten kommunizieren miteinander indem sie Zellen austauschen. Jede dieser Zellen ist 512 Byte groß und besteht aus einem Header sowie einer Payload. Es gibt zwei Sorten von Zellen. Zum einen gibt es *Control Cells* und zum anderen *Relay Cells* (siehe Abbildung 1). Letztere transportieren die Nutzdaten des TCP-Streams. Jede Zelle ist einem Circuit zu-

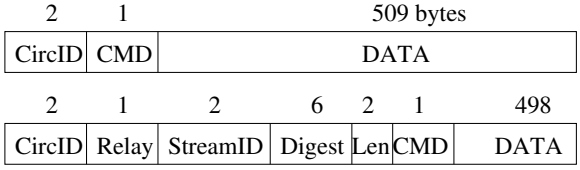

Abbildung 1. Tor Control und Relay Cell [14]

geordnet (CirdID). Dabei hat ein Circuit auf jeder Verbindung zwischen zwei Knoten eine andere CircID. Außerdem können mehrere Circuits über eine TLS-Verbindung gebündelt werden (Multiplexing).
Eine Control Cell wird zum Auf- und Abbau sowie zum Aufrechterhalten eines Circuit genutzt. Für den Aufbau werden die Kommandos *create* und *created* benutzt. Letzteres dient als Bestätigung für das erfolgreiche Etablieren eines Circuit. Das Aufrechterhalten geschieht mit dem Befehl *padding*. Eine Verbindung wird durch *destroy* abgebaut. Diese Kommandos werden unter CMD in der Zelle hinterlegt.
Eine Relay Cell hingegen kontrolliert den eigentlichen Stream. Zunächst kann ein Stream geöffnet (*relay begin)* und wieder geschlossen werden (*relay end*). Die Bestätigung für das erfolgreiche Öffnen eines Stream geschieht mit *relay connected.* Um einen Circuit komplett aufzubauen, muss dieser iterativ um Hops erweitert werden. Denn nach dem Erstellen eines Circuits ist dieser nur eine Verbindung zu einem einzigen Router. Dafür gibt es das *relay extend* sowie *relay extended* Kommando. Genauso kann mit *relay truncate* und r*elay truncated* der Circuit um einen Onion Router reduziert werden.
Relay Cells haben zusätzlich noch einen Relay Header vor den Nutzdaten. Da mehr als ein Stream über einen Circuit transportiert werden kann, wird in diesem Header die zugehörige StreamID gespeichert. Hinzu kommt eine Prüfsumme, um die Integrität der Daten von einem Endpunkt zum anderen zu gewährleisten.
Sowohl der Relay Header als auch die Nutzdaten werden verschlüsselt über die TLS-Verbindung zwischen den einzelnen Knoten übertragen. Daher kann nur der letzte Knoten die Integrität der Daten überprüfen [14].

C. Onion Routing

1) Aufbau eines Circuit: Ein Circuit wird iterativ aufgebaut. Die Knoten, die für den Aufbau des Pfades benötigt werden sowie deren öffentlicher Schlüssel sind durch den Directory Server (siehe Sektion III-D) bekannt [13].
Um einen Circuit von Alice über Bob zu Carol aufzubauen wird zunächst mittels einer Create Cell ein Circuit zu Bob erzeugt (siehe Abbildung 2). Hierfür wird das Public-Key Verfahren benötigt. Jeder Onion Router besitzt also ein Schlüsselpaar bestehend aus einem öffentlichen (Onion Key) sowie einem privaten Schlüssel. Mit diesem Onion Key werden

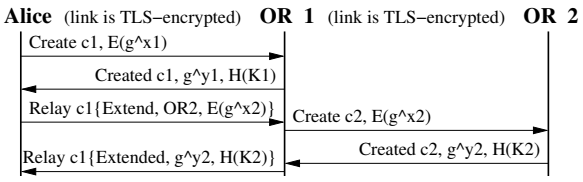

Abbildung 2. Aufbau eines Circuit [14]

sowohl Relay Header als auch Payload verschlüsselt[2]. Die erste Hälfte des *Diffie-Hellman Handshakes* (g^{x_1}) wird in der Payload der Create Cell mitgeführt. Bob entschlüsselt diese

[2]In Abbildung 2 ist dies als E(x) mit RSA-Verschlüsselung dargestellt.

mit seinem privaten Schlüssel und antwortet mit einer Created Cell, die die zweite Hälfte des Handshakes (g^{y_1}) sowie den Hash des ausgehandelten Schlüssels (H(K1)) enthält.
Aus diesem Schlüssel wird jeweils ein symmetrischer Schlüssel für jede Richtung abgeleitet und somit ist die Kommunikation zwischen Alice und Bob (TLS-Verbindung)[3] gesichert.
Um den Circuit zu erweitern, sendet Alice eine Relay Extend Cell (verschlüsselt mit dem ausgehandeltem Session Key), die die Adresse des Onion Router sowie wie eben die Hälfte eines Diffie-Hellman Handshakes (verschlüsselt mit dem öffentlichen Schlüssel von Carol) enthält. Anschließend packt Bob diesen in eine Create Cell, die er an Carol weiterleitet.
Als Antwort bekommt Bob eine Created Cell, die er via Relay Extended an Alice weiterleitet (verschlüsselt mit dem eben ausgehandeltem Session Key).
Alice teilt nun sowohl mit Bob als auch mit Carol symmetrische Schlüssel. Die Schlüssel, die Alice mit Carol ausgehandelt hat, kennt Bob nicht. Alice bleibt außerdem anonym, da sie keinen öffentlichen Schlüssel benutzt.
Für weitere Knoten wird die Route mit dem beschriebenen Verfahren erweitert.
Circuits werden vorbeugend erzeugt, um später keine Verzögerungen zu haben. Des Weiteren werden alte Circuits, die keine offenen Verbindungen mehr halten geschlossen [14] und bestehende jede zehn Minuten [16] ersetzt.

2) Kommunikation: Um eine TCP-Verbindung aufzubauen, wird diese via SOCKS an den lokalen Onion Proxy signalisiert. Es wird ein Circuit sowie ein passender Exit Node (bezüglich dessen Exit Policies) ausgewählt, an den eine Relay Begin Cell gesendet wird. Wenn der Exit Node mit dem Zielhost die Verbindung aufgenommen hat, antwortet er mit einer Relay Connected Cell. Der Onion Proxy akzeptiert fortan die Daten des Streams und packt diese in Relay Data Cells.
Dabei verschlüsselt der OP nacheinander die Zelle mit den symmetrischen Schlüsseln, die er mit den Knoten ausgehandelt hat. Der Stream durch einen Circuit muss nicht immer an dem gleichen Exit Node enden. Er kann an verschiedenen Onion Routern enden (*Leaky-Pipe Circuit Topologie*).
Antworten des Webservers werden von jedem Knoten in dem Circuit verschlüsselt und von Alice schichtweise entschlüsselt [14] .

D. Directory Servers & Bridge Relays

Jeder Client benötigt eine Liste von Onion Router, um einen Circuit aufzubauen. Für diesen Zweck gibt es Directory Server, die mit der Tor-Software mitgeliefert werden [14]. Via HTTP werden bekannte Onion Router sowie deren öffentlicher Schlüssel und ihre Exit Policies [13] als *Signed Directory* [14] heruntergeladen. Onion Router veröffentlichen die eben erwähnten Informationen an einen Directory Server und signieren sie mit ihrem Identity Key.
Die Directory Server synchronisieren sich untereinander. Dabei werden nur Router übernommen, die von mehr als der Hälfte der Anderen gespeichert wurden [13].
Um überhaupt in einem Directory Server aufgenommen zu werden, muss zunächst dessen Administrator dies genehmigen. Unbekannte Identity Keys werden nicht in die Liste von Routern aufgenommen. Denn sonst könnten Angreifer beliebig viele Onion Router einschleusen und das Netzwerk übernehmen [14].
Um in zensierenden Staaten ebenfalls Zugriff auf Tor zu erhalten, gibt es Router, die nicht öffentlich sind. Sogenannte *Bridge Relays* werden über drei Pools vergeben, damit, falls alle Adressen aus einem Pool blockiert wurden, noch andere Bridge Relays bekannt sind. Zum einen gibt es die Möglichkeit diese über ein Formular der Tor Website oder via E-Mail zu bekommen. Und als letzte Möglichkeit bekommt man eine Relay Bridge Adresse über den Direktkontakt zu einem Entwickler des Projekts [16].

E. Entry Guards

Tor kann keine Anonymität gewährleisten, wenn sowohl der Eingangs- als auch der Ausgangsknoten überwacht werden. Denn dann kann durch Analyse des Datenverkehrs basierend auf Mustern wie Größe und Zeit festgestellt werden, dass zwei Parteien miteinander kommunizieren. Tor Circuits sind kurzlebig (zehn Minuten) und aus diesem Grund wächst die Wahrscheinlichkeit für dieses Szenario. Die Chance, dass beide Endpunkte unter der Kontrolle des Angreifers sind, kann durch Einsatz von Entry Guards gemildert werden. Entry Guards werden nicht bei jedem Etablieren eines Circuit neu gewählt, sondern werden über mehrere Wochen benutzt. Sie werden zu solchen durch sehr gute Übertragungskapazitäten und eine hohe Verfügbarkeit über bereits längere Laufzeit [16].

F. Hidden Services

Hidden Services sind eine Möglichkeit innerhalb des Tor-Netzwerkes Dienste anonym anzubieten. Angenommen Bob betreibe einen anonymen Webdienst und Alice wolle darauf zugreifen.
Alice lädt dafür den *Service Descriptor* aus einer Distributed-Hash-Table herunter. Dieser beinhaltet den öffentlichen Schlüssel, der Bobs Dienst identifiziert sowie die Introduction Points, die er ausgewählt hat [16].
Bobs Webdienst ist unter einem Fully Qualified Domain Name (FQDN) mit einer Adresse der Form XYZ.onion erreichbar [17] und weiß nicht einmal davon, dass er durch Tor erreichbar ist. Alice baut einen Circuit zu einem Rendezvous Point auf, bei dem sie einen zufällig gewählten *Rendezvous Cookie* (one-time secret [17]) hinterlegt. Der RP ist ein normaler Onion Router, den sie ausgewählt hat.
Danach baut sie eine Verbindung zu einem der Introduction Points auf und fordert diesen auf, eine Nachricht an Bob zu

[3]Verschlüsselung via AES mit einem symmetrischen Schlüssel zwischen zwei Knoten wird in Abbildung 2 als {X} aufgezeigt.

senden. Diese Nachricht ist mit Bobs öffentlichen Schlüssel verschlüsselt und beinhaltet den Rendezvous Cookie [14], die Hälfte eines Diffie-Hellman Handshakes und ihren Rendezvous Point.
Bob hat nun bedingt durch den indirekten Kontaktversuch die Möglichkeit auf Alice Anfrage nicht zu reagieren. Somit kann er einer DoS Attacke entgegnen, indem er möglichst viele Introduction Points wählt, denn ein Angreifer ist gezwungen das Tor-Netzwerk als solches anzugreifen.
Wenn er allerdings mit Alice kommunizieren will, so antwortet er mit der zweiten Hälfte des Handshakes und einem Hash des zwischen Alice und Bob ausgehandelten Schlüssel, indem er einen Circuit zu dem Rendezvous Point aufbaut. Es wurden also die beiden Circuits (Alice $\leftrightarrow$ Rendezvous Point und Introduction Point $\leftrightarrow$ Bob) miteinander verbunden, ohne dass Alice die Identität von Bob und umgekehrt kennt [14].

G. Evaluation

Beim Surfen mit Tor ist z.B. zu beachten, dass Browser-Plugins wie Flash nicht den Proxy-Einstellungen entsprechend über das Tor-Netzwerk kommunizieren und die IP-Adresse enthüllen können. Für diesen Zweck gibt es jedoch das Tor Browser Bundle, das als Applikationen Tor sowie einen Webbrowser mitliefert, der speziell für die Benutzung von Tor angepasst wurde und u.a. solche Plugins deaktiviert hat [18]. Ein weiterer kritischer Punkt ist, wenn eine Applikation DNS nicht über Tor auflöst [14].
Außerdem sieht der Exit Node die Nachricht in *plaintext*. Deshalb sollten sensible Daten entweder gar nicht über Tor gesendet werden oder wo möglich Verschlüsselung eingesetzt werden, um Missbrauch vorzubeugen.
Durch die zentralen Directory Server, ist es leicht, den Zugang zu Tor zu verhindern, indem Verbindungen zu IP-Adressen der Tor Nodes blockiert werden. Dafür bietet es jedoch Bridge Relays, die nicht in der öffentlichen Liste der Tor Nodes eingetragen sind. Jeder kann ein Bridge Relay betreiben und somit helfen in zensierenden Ländern den Zugang zu Tor zu ermöglichen. Jedoch ist diese Lösung nicht optimal, denn China hat sowohl die Vergabe via Website als auch den E-Mail Pool angegriffen und alle gefundenen Bridge Relays daraufhin gesperrt [16].

IV. I2P

I2P (Internet Invisible Project) ist wie Tor ein low latency anonymes Overlay-Netzwerk [19]. Es wird aktiv seit 2003 entwickelt [20]. Interessant ist, dass die Entwickler nicht bekannt sind und seither anonym unterwegs sind. Der Hauptentwickler ist unter seinem Nickname jrandom bekannt [19].

A. Architektur

I2P ist komplett dezentralisiert und basiert nicht auf zentralen Strukturen wie Directory Servern. Es ist ein eigenständiges Netzwerk [21] in dem anonyme Kommunikation nur innerhalb dessen[4] möglich ist.
Genauer gesagt ist es ein Anonymous Network Layer [20] auf dem Applikationen aufbauen können. Auch wenn es möglich ist bestehende Anwendung durch das SOCKS Proxy Interface in das Netzwerk einzugliedern, wird von dessen Nutzung abgeraten. Denn der einzig sichere Weg für Anonymität, ist eine Anwendung speziell für das I2P Netzwerk zu erstellen, indem es auf deren APIs aufbaut [20] und somit keine sensitiven Informationen hinterlässt, mit denen auf die Identität eines Benutzer geschlossen werden kann [22].
Für diesen Zweck bietet es zwei verbindungsorientierte Protokolle an. Diese sind *NIO-based TCP* (NTCP) für TCP sowie *Secure Semireliable UDP* (SSU) für UDP-Verbindungen [21]. Es wird bereits eine Vielzahl von Diensten wie z.B HTTP, Chat, Filesharing, E-Mail, Newsgroups etc. bereitgestellt [20]. Die Kommunikation zwischen zwei Parteien erfolgt durch mehrere Knoten über sogenannte *Tunnel* [19].
Zunächst erfolgt eine begriffliche Trennung. *Destinations* sind die Endpunkte einer Kommunikation und *Router* die Knoten, über die diese stattfindet.
Ein Tunnel ist also ein Pfad durch solche Router. Es gibt drei verschiedene Arten von Tunneln. Zum einen gibt es ausgehende *(Outbound Tunnel*) und eingehende Tunnel (*Inbound Tunnel*). Diese werden für die eigentliche Kommunikation verwendet. Wenn Alice nun eine Nachricht an Bob schicken will, so geht diese durch ihren Outbound Tunnel und Bob empfängt sie durch seinen Inbound Tunnel (zu sehen an Abbildung 3). Tunnel sind also nur in eine Richtung benutzbar. Für eine vollständige Kommunikation zwischen zwei Parteien werden also mindestens vier Tunnel benötigt (zwei für jede Richtung). Zum anderen gibt es noch

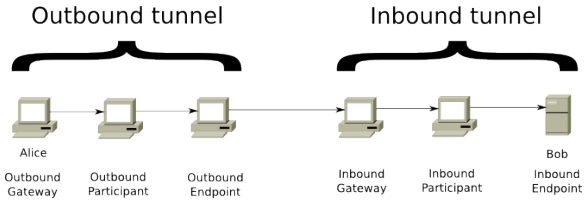

Abbildung 3. Tunnel in I2P [20]

Exploratory Tunnel (ebenfalls für jede Richtung einer) zum Erforschen des Netzwerk. Sie werden dafür genutzt, um die *NetDb* abzufragen. Diese enthält Informationen wie Router und Destinations zu kontaktieren sind.
Daten zwischen zwei Endpunkten einer Kommunikation sind verschlüsselt und werden mit der gleichen atomaren Übetragungseinheit wie bei Tor ausgetauscht. Jedoch mit dem Unterschied, dass mehrere Onion Cells (maximal zwei) zu einem *Garlic Clove* gebündelt werden. Diese können mit einem extra Padding und einer Instruktion für eine zeitliche Verzögerung versehen werden. Jeder Knoten, der

[4]Es gibt auch Gateways für die Kommunikation zwischen I2P und dem Internet [19] sowie Bridges zu Freenet [20], jedoch wurde es dafür nicht geschaffen [19].

diese empfängt entschlüsselt sie zunächst, führt ggf. die Verzögerung aus und packt die Zellen in neue Garlic Cloves. Die Größe der Zellen als auch die Anzahl der Onion Cells variiert von Knoten zu Knoten [19], um eine Analyse des Datenverkehrs basierend auf der Korrelation von Mustern bezüglich Zeit und Größe zu erschweren.

B. NetDB

1) RouterInfo & LeaseSet: Die NetDb ist eine verteilte Datenbank (DHT), die von Routern, die eine große Bandbreite aufweisen (*Floodfill Router*), verwaltet wird. Für diese wird angenommen, dass sie kontinuierlich erreichbar sind und eine kleinere Latenz haben. Momentan ist die minimale Bandbreite dafür 128 KBytes/s. Außerdem müssen sie weitere Tests bestehen [23]. Manche sind manuell dazu konfiguriert als Floodfill Router zu fungieren. Andere melden sich freiwillig dazu bereit, wenn sie die eben erwähnten Kriterien erfüllen [20] und die Anzahl an Floodfill Router unter eine Grenze von 250 fällt [24].
Die Datenbank enthält Informationen darüber wie Router und Destinations zu erreichen sind. Für diesen Zweck beinhaltet sie zwei Formen von Einträgen.
Zum einen ist das die *RouterInfo*, die die Identität[5] eines Router sowie die Adresse, unter der er erreichbar ist (IP und Port) und das Datum wann der Eintrag publiziert wurde, enthält. Diese Informationen werden durch den DSA Schlüssel signiert und unter dem Hash (SHA 256) der Identität gespeichert.
Zum anderen hält sie ähnliche Informationen für Destinations bereit (*LeaseSet*). Dieses Set beschreibt die Eintrittspunkte der Inbound Tunnel. Dafür speichert sie die Identität[5] des Tunnel Gateways, die Tunnel-ID und wann dieser verfällt. Außerdem enthält es die Identität[5] der Destination sowie einen zusätzlichen Schlüssel, der für die Ende-zu-Ende-Veschlüsselung von Garlic Cloves genutzt wird. Auch diese Einträge werden unter dem Hash der Identität in der NetDb eingeordnet [23].
Da Knoten sowohl Router als auch Destinations sein können, haben diese eine unterschiedliche Identität [19].
Da Tunnel jede zehn Minute neu aufgebaut werden, verfallen auch LeaseSets nach gleicher Zeit. RouterInfos hingegen verweilen wesentlich länger und werden kontinuierlich auf der Festplatte gespeichert, damit sie nach einem Neustart der I2P Software wieder verfügbar sind [20].

2) Anfragen und Speichern von Einträgen der NetDb: Um Einträge aus der NetDb zu erfragen, wird eine I2NP (*I2P Network Protocol*) *DatabaseLookupMessage* an die zwei Floodfill Router geschickt, die dem gesuchten Schlüssel am ähnlichsten[6] sind.
Die Anfragen erfolgen durch einen Outbound Exloratory Tunnel und sind nicht veschlüsselt, damit der öffentliche Schlüssel des Anfrager nicht bekannt wird.
Nun wird im Erfolgsfall der Eintrag mit einer *DatabaseStoreMessage* durch einen Inbound Exploratory Tunnel zurückgeliefert. Andernfalls antwortet der Router mit einer *DatabaseSearchReplyMessage*, die eine Liste von anderen Floodfill Routern beinhaltet, die dem gesuchten Eintrag nahe sind. Mit dem beschriebenen Verfahren wird nun solange fortgefahren bis der Eintrag gefunden wurde oder entweder ein Timeout oder die maximale Anzahl zu befragender Router überschritten wurde.
Das Speichern von RouterInfos erfolgt durch direkte Kommunikation mit einem Floodfill Router, indem eine *DatabaseStoreMessage* gesendet wird. LeaseSets hingegen werden durch einen Outbound Tunnel gesendet und sind im Gegensatz zur RouterInfo von Endpunkt zu Endpunkt verschlüsselt, damit sie nicht von dem Endpunkt des Outbound Tunnel gelesen werden können.
Nachdem eine solche Nachricht von einem Floodfill Router gesehen wird, wird diese zunächst überprüft (RouterInfo und LeaseSet sind signiert). Wenn er sie akzeptiert, wird sie an die sieben Floodfill Router gesendet, die der Identität am nähesten[6] sind.
Dies geschieht durch eine DatabaseStoreMessage mit einem Reply Token, der null ist. Damit wird signalisiert, dass der Eintrag nicht erneut verbreitet werden soll.
Um zu überprüfen, dass der Eintrag auch tatsächlich gespeichert wurde, wird nach zehn Sekunden eine Anfrage nach diesem Eintrag gesendet. Wurde der Eintrag nicht gefunden, so wird das Speichern erneut versucht. Der Algorithmus zum Speichern nennt sich *Floodfill* [23].

3) Bootstrapping & Exploration: Das Herstellen von Tunneln benötigt eine Liste von Peers aus dem I2P Netzwerk. Sollte noch keiner bekannt sein, so wird eine Liste bekannter Knoten aus dem Internet heruntergeladen. Die URLs dafür sind in der I2P Software hinterlegt.
Damit das Netzwerk weiter erforscht werden kann, werden Anfragen an die NetDb für zufällige Schlüssel gesendet. Da der Schlüssel nicht gefunden wird, antwortet der Router mit einer Liste von Routern, die nicht als floodfill eingestuft werden (mit entsprechendem Feld in DataBaseLookupMessage gesetzt, denn sonst würden Floodfill Router zurückgeliefert werden) [23].

C. Tunnels

Kommunikation in I2P findet über Tunnel statt. Wie bereits erwähnt gibt es eingehende und ausgehende Tunnel (Client Tunnel) sowie Exploratory Tunnel.
Um überhaupt erst Tunnel zu erstellen wird via NetDb eine Liste von Routern erfragt, die dann als Hops in dem Tunnel agieren. Nun wird dem ersten Hop eine Nachricht zum Aufbau des Tunnels geschickt, mit Hilfe dessen die Nachricht an die anderen Hops weitergeleitet wird, bis der Tunnel aufgebaut ist [20].
Nachrichten, die für einen Outbound Tunnel bestimmt sind,

[5] 2048 Bit ElGamal encryption key, 1024 Bit DSA signing key und certificate [20].

[6] I2P benutzt eine XOR Metrik für den Abstand. Dabei ist diese das XOR der Hashs der Identität konkateniert mit dem Datum [23].

werden schichtweise vom Ersteller des Tunnels verschlüsselt und von jedem Hop um eine Schicht abgetragen [21]. Entschlüsselt sieht der Hop so die IP des nächsten Router sowie die verschlüsselten Informationen [19], denn Nachrichten sind zusätzlich von Endpunkt zu Endpunkt verschlüsselt (*Garlic Encryption*) [20].
Bei einem Inbound Tunnel hingegen kann das Gateway nicht iterativ verschlüsseln, denn es kennt nicht die Schlüssel aller Router des Tunnels. Deshalb fügt jeder Hop eine Schicht Verschlüsselung hinzu. Der Endpunkt des Inbound Tunnel kann diese nun nacheinander mit den Schlüsseln der Hops entschlüsseln [21]. Router sind in vier Kategorien eingeteilt. Dabei spielen Informationen wie Latenz, Performanz und Verfügbarkeit eine Rolle [24], die I2P sammelt, wenn es mit Routern interagiert (*Peer Profiling*).
Somit ergibt sich folgende Einteilung: *Fast and High Capacity, High Capacity, Not Failing und Failing*.
Router werden aus der ersten Kategorie gewählt, um Client Tunnel aufzubauen. Für Exploratory Tunnel werden Router aus der Not Failing Kategorie verwendet [20].
Will Alice nun mit Bob in Verbindung treten, so sucht sie sich zunächst die notwendigen Informationen wie das Gateway von Bobs Inbound Tunnel sowie Bobs öffentlicher Schlüssel (mit dem Garlic Cloves verschlüsselt werden) heraus [20]. Sie sendet ihre Nachricht durch einen ihrer Outbound Tunnel mit Instruktionen für den Endpunkt des Tunnels die Nachricht weiter an Bobs Inbound Tunnel Gateway zu leiten. Damit Bob ihr antworten kann muss Alice ihre Destination der Nachricht beifügen [20].

D. Evaluation

I2P ist ein in sich geschlossenes Netzwerk, das versucht, mit speziell auf die I2P APIs angepassten Anwendungen, bekannte Dienste des Internet nachzubauen. Die APIs sind im Gegensatz zum SOCKS Interface speziell auf Anonymität ausgelegt [25] und somit wird verhindert, dass Applikationen Informationen preisgeben, die die Identität enthüllen könnte.
Es werden konsequent zentrale Ansätze vermieden und Nachrichten sind von einem Ende zum Anderen verschlüsselt. Somit kann bis auf den Empfänger keiner der an der Kommunikation beteiligten Knoten die Nachricht lesen.
Auch wenn Tor mit den Hidden Services eine ähnliche Funktionalität hat, ist I2P speziell für diesen Zweck konzipiert und auch schneller als es Hidden Services in Tor sind. Durch die unidirektionalen Tunnel, müssen, um die komplette Kommunikation abzuhören, doppelt so viele Knoten kompromittiert werden [25].

V. Fazit

Es wurden drei unterschiedliche Projekte vorgestellt.
Freenet ist ein verteilter Datenspeicher und im Gegensatz zu Tor und I2P nicht für Echtzeichtkommunikation geeignet. Das Netzkwerk ist wie I2P dezentral organisiert. In Freenet und I2P ist der Datenverkehr von Endpunkt zu Endpunkt verschlüsselt. Die größte Nutzerbasis sowie Performanz leistet Tor [25]. Im Gegensatz zu Tor unterstützt I2P sowohl TCP als auch UDP-Verbindungen.
Freenet und I2P sind ein in sich geschlossenes Netzwerk, wohingegen Tor zur Kommunikation innerhalb des Internet genutzt wird. Jedoch bietet es mit Hidden Services eine ähnliche Funktionalität. Zusammengefasst kann man also sagen, dass jedes der vorgestellten Darknets seine Daseinsberechtigung hat.

Literatur

[1] Definition darknet. [Online]. Available: http://www.webopedia.com/TERM/D/darknet.html
[2] Internet enemies. Reporter ohne Grenzen. [Online]. Available: http://www.reporter-ohne-grenzen.de/fileadmin/rte/docs/2011/110311_Internetbericht_engl.pdf
[3] Great firewall of china. [Online]. Available: https://de.wikipedia.org/wiki/Internetzensur_in_der_Volksrepublik_China
[4] N. Evans, C. GauthierDickey, and C. Grothoff, "Routing in the dark: Pitch black," in *Computer Security Applications Conference, 2007. ACSAC 2007. Twenty-Third Annual*, dec. 2007, pp. 305 –314.
[5] I. Clarke, O. Sandberg, B. Wiley, and T. Hong, "Freenet: A distributed anonymous information storage and retrieval system," in *Designing Privacy Enhancing Technologies*, ser. Lecture Notes in Computer Science, H. Federrath, Ed. Springer Berlin Heidelberg, 2001, vol. 2009, pp. 46–66.
[6] I. Clarke, O. Sandberg, B. Wiley, and V. Verend. (2010) Private communication through a network of trusted connections: The dark freenet. [Online]. Available: https://freenetproject.org/papers/freenet-0.7.5-paper.pdf
[7] Freenet understand. [Online]. Available: https://freenetproject.org/understand.html
[8] Signed-subspace keys. [Online]. Available: https://wiki.freenetproject.org/SSK
[9] Content-hash keys. [Online]. Available: https://wiki.freenetproject.org/CHK
[10] I. Clarke, S. Miller, T. Hong, O. Sandberg, and B. Wiley, "Protecting free expression online with freenet," *Internet Computing, IEEE*, vol. 6, no. 1, pp. 40 –49, jan/feb 2002.
[11] Freenet faq. [Online]. Available: https://freenetproject.org/faq.html
[12] Freenet node harvesting. [Online]. Available: https://wiki.freenetproject.org/Node_harvesting
[13] K. Michaelis, L. für Netz, and D. Schwenk, "Eine analyse des tor-protokolls," 2012.
[14] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The second-generation onion router." Proceedings of the 13th USENIX Security Symposium Security 2004, 2004.
[15] Tor faq. [Online]. Available: https://www.torproject.org/docs/faq
[16] M. Gaida. (2012, Oktober) Tor the onion router. [Online]. Available: http://www.fh-wedel.de/~gb/seminare/ss2012/gaida_tor.pdf
[17] Tor hidden services. [Online]. Available: https://www.torproject.org/docs/hidden-services.html.en
[18] Tor download. [Online]. Available: https://www.torproject.org/download/download-easy.html.en
[19] B. Zantout and R. Haraty, "I2p data communication system," in *ICN 2011, The Tenth International Conference on Networks*, 2011, pp. 401–409.
[20] I2p techintro. [Online]. Available: http://www.i2p2.de/techintro.html
[21] M. Herrmann and C. Grothoff, "Privacy-implications of performance-based peer selection by onion-routers: A real-world case study using i2p," in *Privacy Enhancing Technologies*, ser. Lecture Notes in Computer Science, S. Fischer-Hübner and N. Hopper, Eds. Springer Berlin Heidelberg, 2011, vol. 6794, pp. 155–174.
[22] I2p socks proxy. [Online]. Available: http://www.i2p2.de/socks.html
[23] I2p netdb. [Online]. Available: http://www.i2p2.de/how_networkdatabase.html
[24] J. Timpanaro, I. Chrisment, and O. Festor, "A bird's eye view on the i2p anonymous file-sharing environment," in *Network and System Security*, ser. Lecture Notes in Computer Science, L. Xu, E. Bertino, and Y. Mu, Eds. Springer Berlin Heidelberg, 2012, vol. 7645, pp. 135–148.
[25] I2p compared to tor and freenet. [Online]. Available: http://www.i2p2.de/how_networkcomparisons.html