

Deanonymisierung von Teilnehmern des I2P-basierten Handelsnetzwerk für digitale Werte („DIVA.EXCHANGE“)

Bachelorarbeit

Brian Boss & Marco Purtschert

| | |
|-------------------|-------------------|
| Auftraggeber: | DIVA.EXCHANGE |
| Betreuungsperson: | Dr. Dieter Arnold |
| Experte: | Jan Alsenz |

Bachelorarbeit an der Hochschule Luzern – Informatik

Titel: Deanonymisierung von Teilnehmern des I2P-basierten Handelsnetzwerk für digitale Werte („DIVA.EXCHANGE“)

Studenten: Brian Boss & Marco Purtschert

Studiengang: BSc Information and Cyber Security

Jahr: 2022

Betreuungsperson: Dr. Dieter Arnold

Experte: Jan Alsenz

Auftraggeber: DIVA.EXCHANGE, Konrad Bächler

Codierung / Klassifizierung der Arbeit:

☒ Öffentlich

☐ Vertraulich

Eidesstattliche Erklärung Wir erklären hiermit, dass wir die vorliegende Arbeit selbstständig und ohne unerlaubte fremde Hilfe angefertigt haben, alle verwendeten Quellen, Literatur und andere Hilfsmittel angegeben haben, wörtlich oder inhaltlich entnommene Stellen als solche kenntlich gemacht haben, das Vertraulichkeitsinteresse des Auftraggebers wahren und die Urheberrechtsbestimmungen der Hochschule Luzern respektieren werden.

Ort / Datum, Unterschrift _____ (Brian Boss)

Ort / Datum, Unterschrift _____ (Marco Purtschert)

Abgabe der Arbeit auf der Portfolio Datenbank:

Bestätigungsvisum Studentin/Student

Wir bestätigen, dass wir die Bachelorarbeit korrekt gemäss Merkblatt auf der Portfolio Datenbank abgelegt haben. Die Verantwortlichkeit sowie die Berechtigungen haben wir abgegeben, so dass wir keine Änderungen mehr vornehmen können oder weitere Dateien hochladen können.

Ort / Datum, Unterschrift _____ (Brian Boss)

Ort / Datum, Unterschrift _____ (Marco Purtschert)

Verdankung

Dr. Dieter Arnold – Für die gute Betreuung während der gesamten Dauer der Arbeit.

Konrad Bächler – Für die gute Zusammenarbeit während des Projektes

Jan Alsenz – Für die hilfreichen Inputs während der Zwischenpräsentation

Christoph Egger – Für das spannende Interview und Einblicke bezüglich seines Vorgehens bei seiner Bachelorarbeit

zzz & zlatinb – Für das aufschlussreiche Expertengespräch und Einblicke in die Mitigationsmassnahmen von I2P

Jan Bucher und Guido Purtschert – Für das Korrekturlesen der Arbeit

Ausschliesslich bei Abgabe in gedruckter Form:**Eingangsvisum durch das Sekretariat auszufüllen**

Rotkreuz, den _____ Visum: _____

Hinweis: Geistiges Eigentum gemäss der Studienordnung für die Ausbildung an der Hochschule Luzern, FH Zentralschweiz

Abstract

Die vorliegende Arbeit befasst sich mit der Frage, welchen Massnahmen und unter welchen Bedingungen Teilnehmer im I2P Netzwerk, spezifischer im DIVA.EXCHANGE I2P Netzwerk, deanonymisiert werden können. Ein Teilnehmer gilt als deanonymisiert, sobald seine Adresse im I2P-Netzwerk seiner öffentlichen IP-Adresse zugewiesen werden kann. Um dieses Ziel zu erreichen, wird nach Schwachstellen im Konzept und in der Architektur von I2P gesucht. Der Fokus liegt auf der konzeptuellen Umsetzung des Netzwerks und nicht auf der Implementation der dazugehörigen Applikationen. Daher wurde innerhalb dieser Arbeit von einer «perfekten Implementation» ausgegangen. Es wurde kein Penetrationstest der Applikationen oder der Infrastruktur durchgeführt.

Um die Möglichkeit einer Deanonymisierung im DIVA.EXCHANGE I2P Netzwerk nachzuweisen, wurden mehrere Angriffsideen erstellt. Die Ideen beruhten primär auf Basis von Recherchen bezüglich I2P und weiteren Overlay-Netzwerken, wie Tor. Auch wurden einige eigene Ideen aufgestellt und überprüft. Die Ideen wurden mittels Gedankenexperimenten, weiteren Recherchen und Codeanalyse der unterschiedlichen Implementierungen zu Angriffskonzepten ausgearbeitet. Mittels mathematischen Modellierungen wurde das erfolgversprechendste Konzept simuliert. Anhand dieser Simulation konnten mögliche Mitigationsmassnahmen und deren Wirksamkeit abgeleitet werden. Eine praktische Umsetzung des Angriffs wurde in Betracht gezogen, konnte aber aufgrund der Komplexität und der benötigten Ressourcen nicht durchgeführt werden.

Die erarbeiteten Konzepte und gewonnenen Erkenntnisse zeigen mögliche Massnahmen, welche die Resistenz des DIVA.EXCHANGE I2P Netzwerks gegenüber Deanonymisierungsangriffen verbessern. So konnte mit der vorliegenden Arbeit aufgezeigt werden, dass durch Verwendung variabler Tunnellängen eine verbesserte Resilienz gegen Deanonymisierungsangriffen erreicht werden kann.

Inhaltsverzeichnis

| | | |
|-----------|---|-----------|
| 1. | Einleitung | 1 |
| 1.1. | Ausgangslage | 1 |
| 1.2. | Problemstellung | 1 |
| 1.3. | Fragestellung | 2 |
| 1.4. | Ziel der Arbeit | 2 |
| 2. | Stand der Technik | 3 |
| 2.1. | Einleitung I2P | 3 |
| 2.2. | The I2P Network Protocol (I2NP) | 3 |
| 2.2.1. | I2NP Messages | 3 |
| 2.2.2. | Garlic Routing | 3 |
| 2.2.3. | Garlic Methoden | 5 |
| 2.3. | Tunnel | 5 |
| 2.3.1. | Client Tunnels | 6 |
| 2.3.2. | Exploratory Tunnels | 6 |
| 2.3.3. | 0-Hop-Tunnel | 7 |
| 2.3.4. | Tunnel Building | 7 |
| 2.3.5. | Tunnel-ID | 8 |
| 2.4. | The Network Database | 8 |
| 2.4.1. | RouterInfo | 9 |
| 2.4.2. | LeaseSet | 9 |
| 2.4.3. | NetDB Funktionen | 9 |
| 2.5. | Kademlia | 10 |
| 2.5.1. | Routing Key | 10 |
| 2.5.2. | Kademlia Closeness Metric | 11 |
| 2.6. | Verbindungsaufbau zu I2P | 11 |
| 2.7. | Verbindung zu bestehenden I2P Seiten | 12 |
| 2.8. | Bekannte Angriffe auf I2P | 13 |
| 2.8.1. | Sybil Attacke | 13 |
| 2.8.2. | Eclipse Attacke | 13 |
| 2.8.3. | Attacks on Bootstrapping | 14 |
| 2.8.4. | Traffic-Analyse | 14 |
| 3. | Methoden | 15 |
| 3.1. | Kanban | 15 |
| 3.2. | Aktionsforschung | 15 |
| 3.3. | Explorative Forschung | 16 |
| 3.3.1. | Literaturrecherche | 16 |
| 3.3.2. | Interview mit Christoph Egger | 16 |
| 3.3.3. | Expertengespräch mit I2P Entwicklern | 17 |
| 3.3.4. | Experimente und Beobachtungen am Forschungsobjekt | 18 |
| 3.4. | Begriffsdefinitionen | 18 |
| 3.4.1. | Identität | 18 |

| | |
|--|-----------|
| 3.4.2. Anonymisierung | 18 |
| 3.4.3. Deanonymisierung | 18 |
| 3.5. Annahme: Perfekte Implementation | 18 |
| 3.6. Anpassungen an der Zielsetzung | 19 |
| 3.6.1. Durchführung einer Deanonymisierung | 19 |
| 3.6.2. Testnetzwerk | 19 |
| 3.7. Verwendete Technologien | 19 |
| 3.8. Projektorganisation | 20 |
| 3.9. Umsetzung der Angriffe | 20 |
| 3.9.1. Abgrenzungen zu Tor | 20 |
| 3.9.5. Experiment: Testnetzwerk | 23 |
| 4. Ideen und Konzepte | 24 |
| 4.1. Tunnel-ID: Rückschluss auf Pos. in Tunnel | 24 |
| 4.2. Übernahme der NetDB durch Bootstrapping | 25 |
| 4.3. Tunnelkontrolle | 26 |
| 4.4. Zero Hop Tunnels: Reseed Server | 29 |
| 4.5. Mustererkennung Datenverkehr | 31 |
| 4.6. Mustererkennung: Nachricht/Tagging | 32 |
| 4.7. Peer Selection Ordering | 33 |
| 4.8. Testnetzwerk | 34 |
| 5. Realisierung | 35 |
| 5.1. Tunnelkontrolle Ansatz 1: Partielle Tunnelkontrolle | 35 |
| 5.1.1. Berechnung benötigter Router | 35 |
| 5.1.2. Erkenntnisse | 38 |
| 5.1.3. Modifikationen | 41 |
| 5.1.4. Variable Tunnellänge | 42 |
| 5.2. Tunnelkontrolle Ansatz 2: DoS auf Participants | 42 |
| 5.3. Testnetzwerk | 44 |
| 5.3.1. Entscheidung | 44 |
| 5.3.2. Probleme | 44 |
| 5.3.3. Anpassung der Konfiguration | 45 |
| 5.3.4. Funktionen | 46 |
| 6. Evaluation und Validation | 47 |
| 6.1. Deanonymisierung von Teilnehmern des I2P Netzwerks | 47 |
| 6.2. Informationsbeschaffung | 47 |
| 6.3. Methodik | 47 |
| 6.4. Verbesserungspotenzial beim Vorgehen | 48 |
| 6.5. Falsifikation | 48 |
| 6.6. Testnetzwerk | 49 |
| 7. Ausblick | 50 |
| 7.1. Fokus: Router Implementationen | 50 |
| 7.2. Fokus: DIVA.EXCHANGE | 50 |
| 7.3. Praktische Umsetzung: Tunnelkontrolle | 50 |
| 7.4. Mitigationsmassnahmen: Tunnelkontrolle | 50 |
| 7.5. Testnetzwerk | 51 |
| 7.6. Fazit | 51 |

| | |
|---------------------------------|-----------|
| Anhänge | 52 |
| A. Aufgabenstellung | 53 |
| Verzeichnisse | 55 |
| Abbildungsverzeichnis | 55 |
| Abkürzungsverzeichnis | 56 |
| Glossar | 57 |
| Literaturverzeichnis | 59 |

1. Einleitung

1.1. Ausgangslage

Das als Verein organisierte freie Software- und Netzwerkprojekt DIVA.EXCHANGE entwickelt den Softwareprototypen DIVA, mit welchem aufgezeigt werden soll, wie die Aufbewahrung, der Handel und der Zahlungsverkehr mit digitalen Werten ohne einen zentralen Dienstleister funktioniert. Ein wichtiger Fokus des Prototyps ist das Gewährleisten der Privatsphäre und der damit verbundenen Anonymität der Nutzer. Um dies zu erreichen, wird als Anonymisierungsschicht das [I2P](#) verwendet.

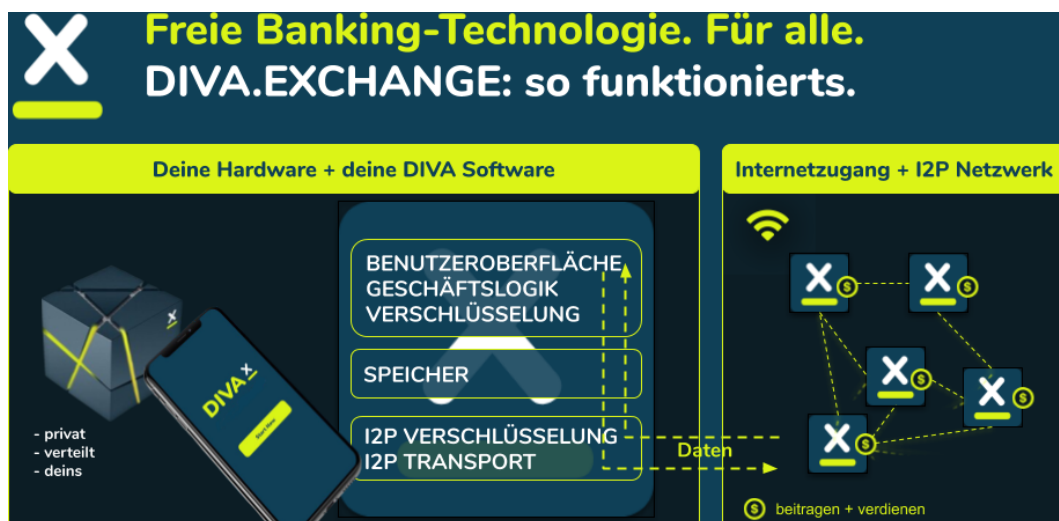


Abbildung 1.1.: DIVA.EXCHANGE: Funktionsweise [9]

Bei DIVA handelt es sich um ein langfristiges Forschungsprojekt, wobei diese Arbeit Teil des Forschungsprojektes ist. Das übergeordnete Ziel dieser Arbeit ist aufzuzeigen, mit welchen Massnahmen und unter welchen Bedingungen Teilnehmer im I2P Netzwerk, spezifischer im DIVA.EXCHANGE I2P Netzwerk, de-anonymisiert werden können.

1.2. Problemstellung

Die Software „DIVA“ basiert auf dem Anonymisierungsnetzwerk [I2P](#). Es soll aufgezeigt werden, wie die Teilnehmer des Netzwerkes deanonymisiert werden können. Deanonymisierung ist definiert als „Offenlegung der IP-Adresse“ eines am DIVA.EXCHANGE Netzwerk teilnehmenden Knotens. Im [I2P](#) Netzwerk gilt zu beachten, dass nicht verschleiert wird, dass die jeweiligen Nutzer Teil des Netzwerkes sind. Unter Berücksichtigung dieser Erkenntnisse ist De-Anonymisierung in dieser Arbeit definiert als Zuweisung der IP-Adresse eines am DIVA.EXCHANGE Netzwerk teilnehmenden Knotens mit dessen B32-Adresse.

1.3. Fragestellung

Basierend auf der Ausgangslage und der Problemstellung wurde die folgende Fragestellung mit den entsprechenden Folgefragen abgeleitet.

Können Teilnehmer des I2P Netzwerk deanonymisiert werden?

Falls diese Frage mit **Ja** beantwortet werden kann, ergeben sich die folgenden Folgefragen.

- Welche Mittel und Voraussetzungen sind für eine Deanonymisierung notwendig?
- Wie schwierig ist eine Deanonymisierung zu erreichen?
- Was kann gegen diese Deanonymisierung gemacht werden?

Falls diese mit **Nein** beantwortet werden kann, ergeben sich zusätzlich die folgenden Folgefragen.

- Wieso sind diese Angriffe nicht erfolgreich?
- Welche Massnahmen sind notwendig, um nicht erfolgreiche Angriffe umzusetzen?

1.4. Ziel der Arbeit

Ziel der Arbeit ist aufzuzeigen, mit welchen Massnahmen und unter welchen Bedingungen Teilnehmer im I2P Netzwerk, spezifischer im DIVA.EXCHANGE I2P Netzwerk, deanonymisiert werden können. Ein Teilnehmer gilt als deanonymisiert, sobald seine IP-Adresse offengelegt wurde, respektive, wenn einem LeaseSet das entsprechende Router-Info zugewiesen werden kann.

Zum Hauptziel der Deanonymisierung sind folgenden Zusatzziele definiert. Diese sollen aus den Erkenntnissen zu der Deanonymisierung beantwortet und festgehalten werden.

1. Aufbauen des Know-hows bezüglich I2P
2. Aufbauen des Know-hows bezüglich bekannter Angriffe in I2P
3. Erarbeiten von neuen Angriffen
4. Analyse von existierenden Angriffen
5. Beschreiben von funktionierenden Angriffen
6. Beschreiben von nicht funktionierenden Angriffen
7. Erstellen eines „Grundlagen-Dokuments“, auf welchem weitere Arbeiten aufbauen können

2. Stand der Technik

2.1. Einleitung I2P

Das [Invisible Internet Project \(I2P\)](#) ist eine vollständig verschlüsselte Netzwerkschicht, welche in Form eines sogenannten Overlay-Netzwerks auf der bestehenden, [IP](#) basierten Internetinfrastruktur aufgebaut ist. Das Ziel von [I2P](#) ist es, allen Teilnehmern komplette Anonymität zu gewähren, egal ob Nutzer oder Anbieter eines Services. Dies wird erreicht, indem die Teilnehmer untereinander über sogenannte Tunnels verbunden sind. Was ein Tunnel genau ist und wie diese funktionieren, wird in Kapitel [2.3 Tunnel](#) erläutert. Durch diese Tunnels wird erreicht, dass die Clients nie direkt mit der Gegenseite kommunizieren. Durch die bereits auf der Kommunikationsschicht eingeführte Verschlüsselung, wird sichergestellt, dass eine sichere und anonyme Kommunikation gewährleistet werden kann. Darüber liegende Applikationen müssen sich somit nicht mehr darum kümmern, ihre Daten während des Transports zu verschlüsseln.

2.2. The I2P Network Protocol (I2NP)

Das [I2NP](#) ist ein Peer zu Peer Protokoll und kommt in [I2P](#) zur Anwendung. Im [I2NP](#) werden insbesondere [I2NP Messages](#), [Garlic Routing](#) und die [Tunnel](#) definiert.

2.2.1. I2NP Messages

Die [I2NP](#) Messages bilden das Grundgerüst der Kommunikationsebene von [I2P](#). In ihnen wird definiert, welche Nachrichten und Informationen wie ausgetauscht werden. Für die verschiedenen Anwendungsfälle wurden unterschiedliche [I2NP](#) Messages definiert, welche sich primär durch die im Header enthaltenen Informationen unterscheiden. Bei der Implementation wurde ein grosser Fokus auf das Need-to-know-Prinzip gelegt, sodass es für einen Angreifer nicht möglich sein sollte, anhand einer Nachricht Rückschlüsse auf den Sender oder Empfänger zu schliessen. [I2P](#) kennt viele unterschiedliche Arten von Messages. Eine vollständige Auflistung ist in der Dokumentation der [I2P](#) Entwickler zu finden. [17]

2.2.2. Garlic Routing

Garlic Routing ist eine Technik zum Aufbau von Tunnels durch eine Reihe von Peers und anschliessender Nutzung dieser Tunnels. Das *Garlic Routing* basiert stark auf dem *Onion Routing* Protokoll, welches innerhalb des TOR Netzwerks verwendet wird. Der Begriff *garlic* hat in [I2P](#) keine präzise Definition. Wie auf der Webseite der [I2P](#) Entwickler [15] beschrieben, wird dieser Begriff für eine der folgenden Definitionen oder deren Kombination verwendet.

Layered Encryption

Unter dem Begriff Layered Encryption wird das mehrfache Verschlüsseln einer Nachricht bezeichnet. Der Name entstand dadurch, dass mehrere Schichten oder Layers von Verschlüsselungen über die gleiche Nachricht gelegt werden. Vor dem Versenden einer Nachricht innerhalb des I2P Netzwerks wird die Nachricht für jeden Teilnehmer des eigenen Outbound Tunnel verschlüsselt. Durch diesen Vorgang können die beinhalteten Informationen immer nur von dem Router gelesen werden, für welchen die jeweiligen Informationen bestimmt sind. Beim Inbound Tunnel wird ebenfalls eine Art von Layered Encryption verwendet. Jeder Router auf dem eingehenden Weg verschlüsselt die Nachricht mit seinem Schlüssel. Dadurch wird verhindert, dass ein Teilnehmer in der Tunnelkette Informationen der vorgehenden Teilnehmer lesen kann. Die mehrfache Verschlüsselung hat nicht das Ziel, dass die Nachricht besser geschützt ist, sondern dass gezielt gesteuert werden kann, wer die Nachricht lesen kann. Dabei ist ebenfalls die Reihenfolge wichtig, da jeder Teilnehmer seinen Verschlüsselungslayer entfernt und dadurch den nächsten freigibt.

Bundling multiple messages together

Mit dem Begriff Garlic kann auch die Erweiterung des Onion Routings zu Garlic Routing gemeint sein. Die Idee dabei ist, dass mehrere einzelnen Nachrichten, auch Garlic Clove genannt, zu einem Garlic kombiniert werden können. Eine Garlic Clove beinhaltet eine vollständige I2NP Message. Eine Garlic Clove kann zum Beispiel eine Database Store Message beinhalten, welche verwendet wird, um eine LeaseSet an einen Floodfill zu schicken.

Garlic Encryption

Als weitere Möglichkeit kann Garlic auch im Zusammenhang mit einer einfachen Verschlüsselung verwendet werden. Dafür wird häufig der Begriff Garlic Encryption verwendet. Damit ist die End-to-End Verschlüsselung zwischen zwei Clients gemeint.

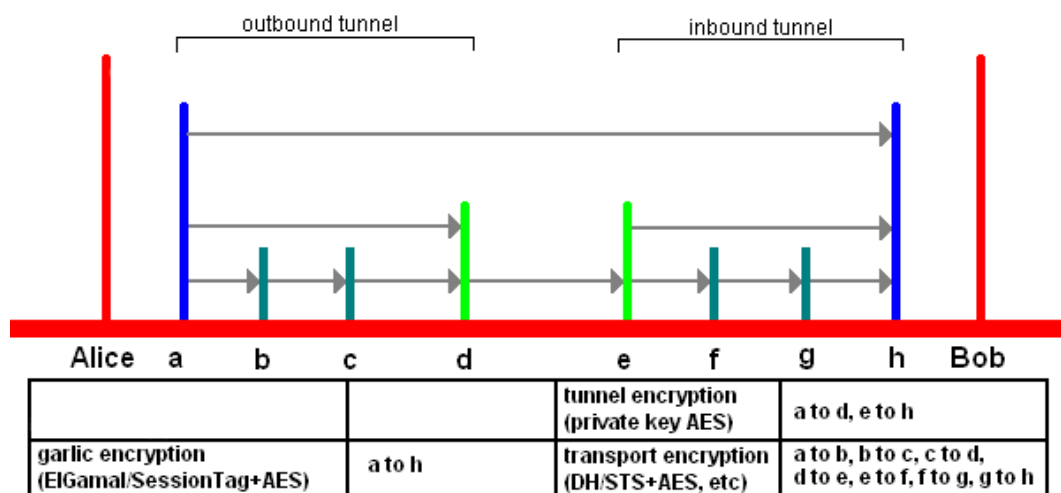


Abbildung 2.1.: Unterschied zwischen Tunnel Encryption und Transport (Garlic) Encryption [1]

Im Gegensatz zur Layered Encryption wird dabei nur die eigentliche Nachricht einmalig verschlüsselt. Dies wird im I2P Netzwerk zusätzlich zur Layered Encryption verwendet. Die Layered Encryption stellt dabei die Transport Encryption dar und die Garlic Encryption die Verschlüsselung zwischen zwei Endpunkten.

2.2.3. Garlic Methoden

Die drei in [Unterabschnitt 2.2.2](#) beschriebenen Definition werden für die drei Garlic Methoden verwendet.

Tunnel Building and Routing

Die Layered Encryption wird für das Erstellen und die Verwendung von Tunnels benötigt. Dies wird im [Unterabschnitt 2.3.4](#) genauer beschrieben. Alle Nachrichten, welche durch einen Tunnel verschickt werden, sind durch das Layered Encryption geschützt. Wie bereits im vorherigen Kapitel beschrieben, soll damit erreicht werden, dass jeder Tunnel-Teilnehmer die Informationen lesen kann, welche er für seine Aufgabe innerhalb des Tunnels benötigt.

End-to-End Message Bundling

Ein [Router](#) kann mehrere Nachrichten zusammen in einer Garlic Message verschicken. Ein Beispiel dafür ist das Anfordern einer Empfangsbestätigung durch eine *Delivery Status Message*. Diese wird zusammen mit einer *Database Store Message* verschickt. Beide Nachrichten sind jeweils eine *Clove Message*. In [Abbildung 2.2](#) der I2P Dokumentation wird als Beispiel noch eine weitere Nachricht in die Garlic Message verpackt. Im erwähnten Beispiel wird ein HTTP Request an den Floodfill geschickt. Zusätzlich wird diesem Floodfill noch das LeaseSet des Absenders zugestellt und eine Empfangsbestätigung angefordert.

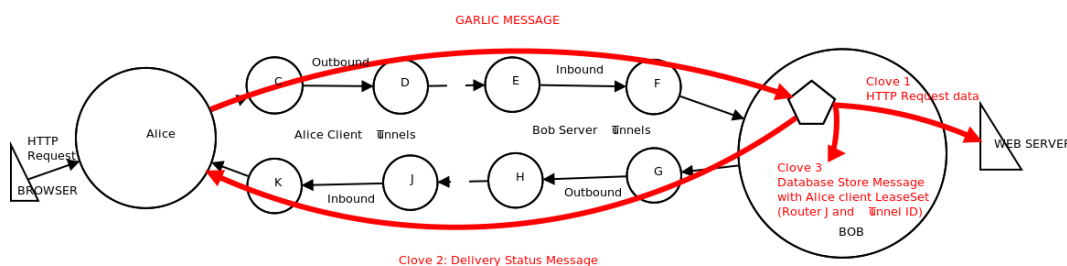


Abbildung 2.2.: End-to-End Message Bundling [15]

2.3. Tunnel

Innerhalb von I2P sind die Tunnels eine essenzielle Komponente. Um zu verstehen, wie in [Invisible Internet Project \(I2P\)](#) Daten anonym übertragen werden, muss das Konzept und die Funktionsweise der unterschiedlichen Tunnels verstanden werden. Ein Tunnel besteht aus mehreren Teilnehmern, welche die empfangenen Datenpakete entgegennehmen und weiterleiten. Anzumerken ist, dass jeder Teilnehmer (mit Ausnahmen des Tunnelerstellers) einzig seine eigene Rolle kennen darf. Dies ist notwendig, damit keine Rückschlüsse

auf die Identität des Empfängers, des Senders, oder auch die weiteren, nicht benachbarten Tunnelteilnehmer gezogen werden können.

I2P kennt mehrere Arten von Tunnel-Typen, welche unterschiedliche Funktionen übernehmen. Wichtig ist, dass I2P nur unidirektionale Tunnels kennt. Das bedeutet, dass für das Senden und das Empfangen unterschiedliche Tunnels verwendet werden.

2.3.1. Client Tunnels

Sogenannte Client Tunnels werden für das Verschicken und Empfangen von Nachrichten innerhalb des I2P Netzwerks verwendet. Bei der Verbindung zu einem Webserver wird die Anfrage über Client Tunnels verschickt. Wie erwähnt sind Tunnels in I2P immer unidirektional. Daher benötigt I2P Inbound und Outbound Tunnels.

Outbound Tunnels

Der Outbound-Tunnel übernimmt die Funktion des Versendens von Nachrichten. Er besteht aus einem [Outbound Gateway](#), aus keinem oder mehreren [Outbound Participants](#) und einem [Outbound Endpoint](#). Der [Outbound Gateway](#) ist Sender der Nachricht. Seine Aufgabe ist es, die Nachricht zu verschlüsseln und an den ersten [Outbound Participants](#) weiterzuschicken. Der [Outbound Endpoint](#) ist der letzte Teilnehmer, welcher dem Absender bekannt ist. Er entschlüsselt die letzte Schicht, welche bei der [Layered Encryption](#) erstellt wurde. Darin enthalten ist die Information für den Inbound Tunnel Gateway und die Garlic verschlüsselte Nachricht. Mit Ausnahme des Gateways und des Endpoints innerhalb des Tunnels, haben alle Teilnehmer die gleiche Funktion. Sie entschlüsseln ihr Layer der Nachricht und leiten die Nachricht an den nächsten Teilnehmer weiter.

Inbound Tunnels

Ein Inbound Tunnel besteht aus einem [Inbound Gateway](#), aus keinem oder mehreren [Inbound Participants](#) und einem [Inbound Endpoint](#). Der Gateway ist der erste Node im Tunnel und nimmt die Daten initial entgegen. Ähnlich wie bei einem Inbound Tunnel leiteten die Teilnehmer die Nachrichten an den nächsten Teilnehmer weiter. Der Unterschied dabei ist, dass die Nachricht von jedem Teilnehmer erneut verschlüsselt und nicht entschlüsselt wird. Erst der Empfänger, der [Inbound Endpoint](#) ist in der Lage, alle Schichten wieder zu entschlüsseln.

2.3.2. Exploratory Tunnels

Exploratory Tunnels sind technisch gleich aufgebaut wie die im [Unterabschnitt 2.3.1](#) beschriebenen Inbound und Outbound Tunnel. Sie übernehmen administrative Aufgaben, wie das Prüfen, ob bekannte [Router](#) noch erreichbar sind [32]. Weiter werden die Exploratory Tunnels verwendet, um die eigene [Network Database \(NetDB\)](#) zu erweitern. Dies geschieht, indem zufällig ein [Routingkey](#) generiert wird. Anschliessend wird der [Floodfill](#) aus der lokalen [NetDB](#) ausgewählt, welcher gemäss [Kademlia](#) am nächsten zu dem [Routingkey](#) ist. Diesem [Floodfill](#) wird eine sogenannte *DatabaseLookup* Message gesendet, bei welcher das „exploration lookup“ Flag gesetzt ist. Als Key für den Lookup wird der generierte [Routingkey](#) verwendet. Der [Floodfill](#) sendet als Antwort eine *DatabaseSearchReply* Message in der [Router](#) enthalten sind, welche am nächsten zum generierten [Routingkey](#) sind.

2.3.3. 0-Hop-Tunnel

Ein **0-Hop-Tunnel** ist ein Spezialfall. Hierbei handelt es sich um keinen eigentlichen Tunnel, da er nur einen Absender besitzt. Ein **0-Hop-Tunnel** wird nur benötigt, wenn keine andere Inbound oder Outbound Tunnels vorhanden sind. Das Verhalten des **Router** ist identisch wie bei einem Client oder Exploratory Tunnel. Somit soll verhindert werden, dass die Gegenseite feststellen kann, dass ein **0-Hop-Tunnel** verwendet wurde.

0-Hop-Tunnel kommen insbesondere während dem Bootstrapping Prozess zum Einsatz, da zu diesem Zeitpunkt noch keine Tunnel existieren, die für die Kommunikation innerhalb des Netzwerks verwendet werden können. Weiter kann ein I2P-Teilnehmer in seiner Konfiguration **0-Hop-Tunnel** konfigurieren. Dies bewirkt primär eine bessere Performance, auf Kosten der Anonymität.

2.3.4. Tunnel Building

Der Tunnel Building Prozess wird von jedem **Router** regelmässig durchgeführt. Da ein Tunnel eine Laufzeit von maximal 10 Minuten hat, müssen diese regelmässig wieder neu generiert werden. Für das Erstellen eines Tunnels verfasst der **Router** eine *TunnelBuild* Message, welche mehrere *BuildRequestRecords* beinhaltet. Laut Dokumentation [17] wird die *TunnelBuild* durch die flexiblere *VariableTunnelBuild* Message abgelöst. Vom Aufbau her sind beide identisch. Allerdings beinhaltet die *TunnelBuild* Message eine fixe Anzahl Records, wobei die variable Version eine unterschiedliche Anzahl an Records beinhalten kann. Die **i2p.i2p** Implementation verwendet für Tunnels mit der Länge fünf oder kleiner, fünf Records. Für längere Tunnels werden acht Records verwendet. Längere Tunnels mit noch mehr Records werden aktuell nicht unterstützt, um das Netzwerk nicht unnötig zu belasten.

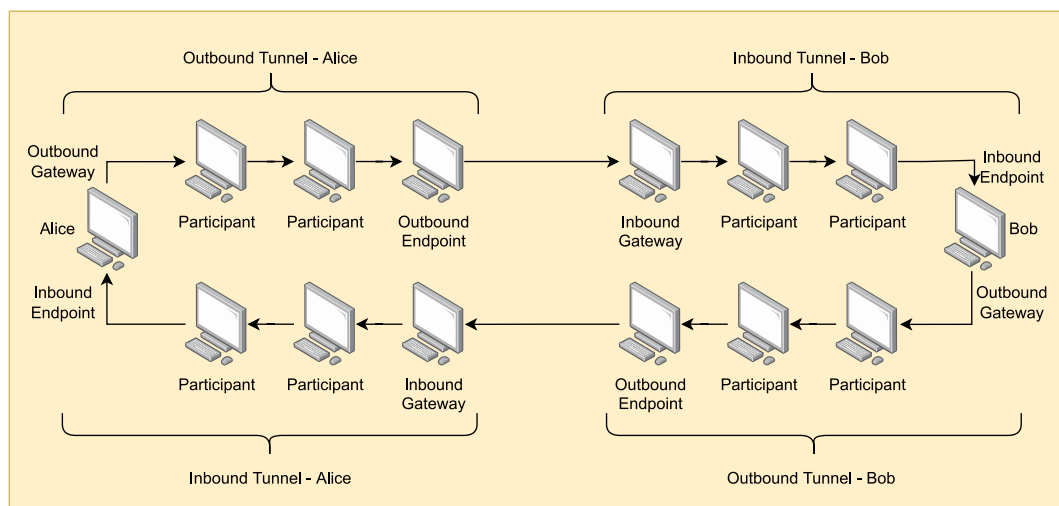


Abbildung 2.3.: Inbound/Outbound Tunnels

Jeder dieser Records wird mit dem Public Key des Empfängers asymmetrisch verschlüsselt. Dieser beinhaltet eine eindeutige und Teilnehmer spezifische Tunnel-ID. Ebenfalls sind die Kontaktinformationen, Routeridentität und Tunnel-ID, des nächsten Teilnehmers enthalten. Zusätzlich werden Informationen für das Erstellen eines Layerkeys mitgeliefert. Diesen verwendet der **Router** für das Ver- oder Entschlüsseln der weiterzuleitenden Nachrichten. Wenn der **Router** seinen Record gelesen hat, ersetzt er diesen durch einen

verschlüsselten *BuildResponseRecord* [17]. Darin teilt er mit, ob er seine Position im Tunnel annehmen möchte. Diese Antwort wird einem mitgeschickten Reply Key symmetrisch verschlüsselt und dann weiter an den nächsten potenziellen Teilnehmer geschickt. Wichtig dabei ist, dass erst der initiale Sender, also der Tunnel Ersteller, die Antwort aller Teilnehmer lesen kann. Es ist also möglich, dass ein Tunnel bereits nach dem ersten Teilnehmer nicht zustande kommt, trotzdem aber bei allen anderen Teilnehmer eingetragen wird. Erst der letzte [Router](#) schickt die Nachricht an den Absender zurück. Der vollständige Prozess ist auf der Seite der [I2P Entwickler](#) [38] im Detail beschrieben. In [Abbildung 2.3](#) sieht man eine Übersicht, wie die Kommunikation innerhalb des Netzwerks funktioniert. [I2P](#) verwendet vier Tunnels, um die Kommunikation zwischen zwei Kommunikationspartner zu gewähren. Dabei steht jeweils ein Inbound und ein Outbound Tunnel unter Kontrolle der jeweiligen Parteien. So kann die Anonymität zwischen den einzelnen Teilnehmern auch gewährleistet werden, wenn Angreifer eine Kommunikation mittels [0-Hop-Tunnel](#) erzwingen möchte.

Unterschied Inbound und Outbound Tunnels

Zwischen dem Aufbau eines Inbound und Outbound Tunnels gibt es nur einen Unterschied. Die initiale Nachricht für einen Inbound Tunnel wird über einen Outbound Tunnel verschickt. Die Teilnehmer des Tunnels leiteten die Build Message so lange weiter, bis diese beim Tunnel Ersteller ankommt. Für den Outbound Tunnel wird die *TunnelBuild* Message direkt vom Ersteller verschickt. Der letzte Teilnehmer des [Outbound Gateway](#) erhält eine zusätzliche Information zu seiner Rolle in seinem Record. Der letzte Teilnehmer hat ebenfalls die Aufgabe, alle Records zurückzuschicken. Dazu wird die vollständige *TunnelBuildMessage* in eine *TunnelBuildReply* Message verpackt. Diese Message wird danach an die im Record definierte Reply Adresse geschickt. Der Ersteller des Tunnels fügt dazu die Informationen eines Inbound Tunnels zur *TunnelBuild* Message hinzu.

2.3.5. Tunnel-ID

Die Tunnel-ID ist eine ID, welche beim Erstellen des Tunnels generiert wird. Diese identifiziert einen Tunnel und wird von einem [Router](#) verwendet, um Pakete dem korrekten Tunnel zuzuweisen. Speziell an dieser ID ist, dass sie für jeden Teilnehmer eines Tunnels unterschiedlich ist. Beim Tunnel Building wird für jeden Teilnehmer eine eigene ID generiert, welche für ihn mitgeschickt wird. Jeder Teilnehmer des Tunnels speichert sich seine ID und den dazugehörigen, nächsten Teilnehmer des Tunnels. Sobald er ein Paket erhält, sucht er die im Paket enthaltene ID in seinem lokalen Speicher. Falls er diese findet, leitet er das Paket an die dazugehörigen Teilnehmer des Tunnels weiter. Falls die ID unbekannt ist, verwirft der [Router](#) die Nachricht.

2.4. The Network Database

Die [\(\)](#) ist eine verteilte (distributed) Datenbank, welche das [I2P](#) Netzwerk verwendet, um Informationen über das Netzwerk zu teilen. Sie beinhaltet die zwei Informationen, welche Teilnehmer benötigen, um miteinander zu kommunizieren: die [RouterInfo](#) und die [LeaseSet](#). Es ist anzumerken, dass kein Teilnehmer eine vollständige Kopie der Datenbank besitzt, sondern nur Teile der Datenbank. Auch gibt es keine zentrale Autorität,

welche die Datenbank verwaltet. Der im Kapitel [Kademlia](#) beschriebene Algorithmus stellt sicher, dass eine Kommunikation trotz Teilwissens des Netzwerks möglich ist.

2.4.1. RouterInfo

Eine [RouterInfo](#) beschreibt, wie ein anderer [Router](#) erreichbar ist. Sie beinhaltet eine öffentliche IP-Adresse, einen Port und die Identität des [Router](#). Zusätzlich sind noch „Capabilities flags“ und weitere Informationen über die laufende Router Software enthalten. Das Erstere wird verwendet, um anzugeben, wie hoch die Bandbreite eines [Routers](#) ist. Diese Bandbreite kann verwendet werden, um zu entscheiden, ob eine Verbindung zu einem [Router](#) aufgebaut werden soll. Die vollständige Spezifikation kann der Dokumentation [7] entnommen werden.

2.4.2. LeaseSet

Ein [LeaseSet](#) ist im übertragenen Sinn eine Liste von mögliche Kontaktinformationen ([Lease](#)) einer Destination. Ein [Lease](#) innerhalb des [LeaseSets](#) beinhaltet die Router-identity des [Inbound Gateways](#), mit welcher die entsprechende [RouterInfo](#) in der [NetDB](#) gefunden werden kann. Weiter enthalten ist die entsprechende Tunnel-ID und das Ablaufdatum des Tunnels. Wie genau ein [LeaseSet](#) und die dazugehörigen [Leases](#) funktionieren, ist in der Dokumentation der [I2P](#) Entwickler beschrieben [7].

2.4.3. NetDB Funktionen

Für die Verteilung von [NetDB](#) Informationen kommen im [I2P](#) die Store und die Lookup Funktion zur Anwendung. Eine Funktion zum Löschen von Datenbankeinträgen gibt es nicht. Datenbankeinträgen werden von den einzelnen [Routern](#) selbstständig aus ihrer eigenen [NetDB](#) gelöscht, sobald die Einträge als veraltet erkannt werden.

Store

Damit ein [LeaseSet](#) gefunden werden kann, muss dieses erst einmal dem Netzwerk bekannt geben werden. Um dies zu erreichen, wird das entsprechende [LeaseSet](#) einer Anzahl von Floodfills zugestellt. Ein [Router](#) versendet dazu die *DatabaseStore* [17] Message an eine bestimmte Anzahl Floodfills, welche eine möglichst kleine Distanz zu seiner eigenen Router Identität haben. Diese Floodfills speichern das [LeaseSet](#) in ihrer [NetDB](#) und kann so von weiteren Teilnehmern gefunden werden.

Lookup

Um ein Ziel zu finden, muss der [Router](#), welcher eine Verbindung aufbauen möchte, im Besitz der Kontaktinformationen sein. Falls er diese nicht in seiner [NetDB](#) besitzt, kann er die Kontaktinformationen mittels *DatabaseLookup* Message bei einem Floodfill erfragen. Ein Floodfill, welcher die *DatabaseLookup* Message erhält, prüft sein eigene [NetDB](#) ob er die erfragte Identität kennt. Falls nicht, antwortet der Floodfill mit einer *DatabaseSearchReplyMessage*, welche eine Liste mit weiteren Floodfills in der Nähe des gesuchten Ziels enthält. Die Dokumentation [37] der [I2P](#) Entwickler enthält weiter, dass bis zu zwei Floodfills gleichzeitig angefragt werden können. Dies ist jedoch je nach Implementation unterschiedlich.

Beispiel Search und Store

In [Abbildung 2.4](#) sieht man an einem vereinfachten Beispiel, wie der Ablauf funktioniert. Bob sucht dabei das LeaseSet von Alice und fragt dafür diverse Floodfills an, die er kennt und gemäss Kademlia Algorithmus am logisch nächsten zu Bob sind. Im beschriebenen Beispiel hat Bob Glück und einer seiner bekannten Floodfills kennt das LeaseSet von Alice. Dies liegt daran, dass Alice ihr LeaseSet diesem Floodfill direkt mitgeteilt hat. Es ist aber auch möglich, dass der von Bob angefragte Floodfill das LeaseSet über einen dritten Floodfill erfahren hat, da diese sich ebenfalls austauschen.

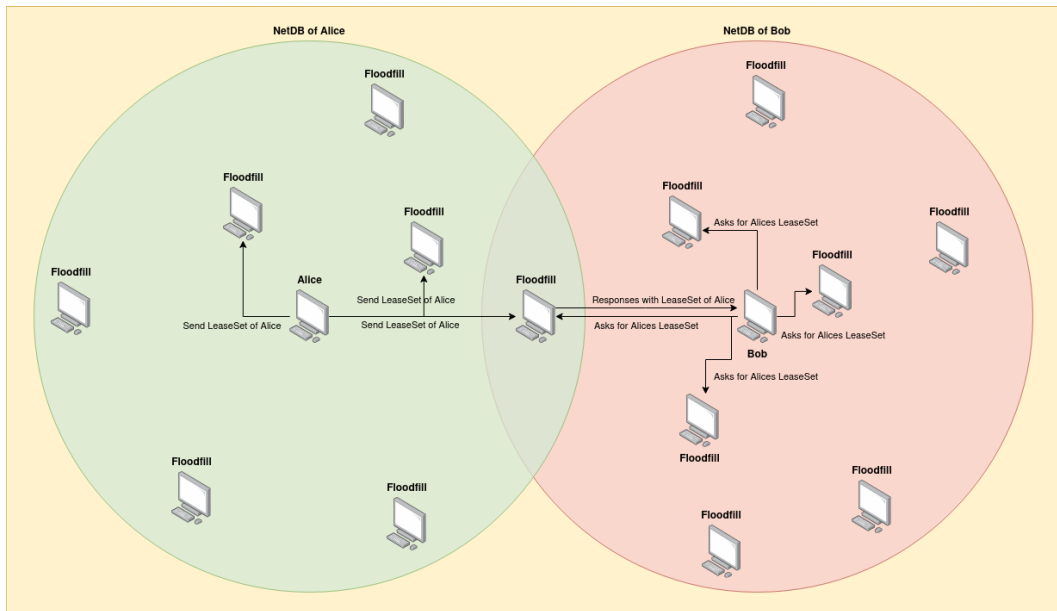


Abbildung 2.4.: Beispiel Search und Store

Der Kademlia Algorithmus hilft dabei, dass Router mit einem Teilwissen des Netzwerks einen Weg durchs Netzwerk findet. Es besteht aber immer die Möglichkeit, dass keiner der angefragten Teilnehmer die Information kennt und eine Anfrage im Leeren landet. Um dieses Problem zu verringern, wird das LeaseSet der Router über mehrere Floodfills verteilt, in der Hoffnung, dass beide Router mindestens einen Floodfill gemeinsamen haben. [30].

2.5. Kademlia

Kademlia ist die von I2P verwendete Distributed Hash Table (DHT). Sie wird für die Abfrage von LeaseSets verwendet. Dafür wird in der Datenbank der Hashwert der Router Identität als Key verwendet und das dazugehörige LeaseSet gespeichert. Jeder Router führt eine eigene NetDB und tätigt die notwendigen Berechnungen für den Kademlia Algorithmus selbst.

2.5.1. Routing Key

Für die Erstellung des Hashes oder auch Routing Key wird, wie bereits erwähnt, die Identität des Routers verwendet. Zur Berechnung des Routing Key in I2P wird das

aktuelle Datum angehängt. Wie in der Dokumentation der Entwickler beschrieben [37], wird dies gemacht, um den Aufwand für eine Sybil Attacke zu erhöhen. Der Routing Key wird nie übertragen, sondern jeder [Router](#) berechnet diesen selbst.

2.5.2. Kademlia Closeness Metric

Für eine möglichst effiziente Verteilung der Informationen innerhalb der [NetDB](#) wird eine Distanz oder Closeness Metric berechnet. Dazu werden von beiden [Routern](#) die Routing Keys erstellt. Auf diese beiden Routing Keys wird dann ein XOR angewendet. Das Resultat dieser Berechnung ergibt die Distanz zwischen den beiden Identitäten. Beispiel:

$$\begin{aligned} 1001 \oplus 1010 &= 0011 \\ 0011_2 &= 3_{10} \end{aligned}$$

Wie am obigen Beispiel ersichtlich ergibt dies eine Distanz von drei.

2.6. Verbindungsaufbau zu I2P

Damit ein Teilnehmer im [I2P](#) Netzwerks kommunizieren kann, muss er mit den weiteren Netzwerkteilnehmern in Verbindung treten. Beim erstmaligen Verbinden zum Netzwerk sind jedoch noch keine weiteren Teilnehmer bekannt und es ist daher nicht möglich eine Verbindung zum Netzwerk herzustellen. Dies ist ein bekanntes Problem bei Peer-to-Peer-Netzwerken, welches auch als Bootstrapping Problem bekannt ist. Um dieses Problem zu lösen, verwendet [I2P](#) sogenannte Reseed Server. Diese sind über das Internet erreichbar und geben bei einer Anfrage eine Liste von existierenden [Router](#) zurück. Damit die [Router](#) während dem Bootstrapping die Reseed Server kennen, werden sie mit der [Router](#) Implementation ausgeliefert. Weiter wäre es möglich, mittels Config-File weitere Reseed Server anzugeben.

Sobald der [Router](#) seine ersten Nodes kennt, beginnt er mit dem Aufbau von Inbound und Outbound Tunnel. Die Tunnel Build Messages für das Erstellen der Inbound Tunnel werden dabei über die Outbound Tunnel versendet. Die Antworten über das Erstellen der Outbound Tunnel werden wiederum über die Inbound Tunnel gesendet. Damit dies auch beim Erstellen der ersten Tunnels funktioniert, wird ein [0-Hop-Tunnel](#) erstellt. So kann die Anonymität selbst beim Erstellen der Tunnels gewährt werden.

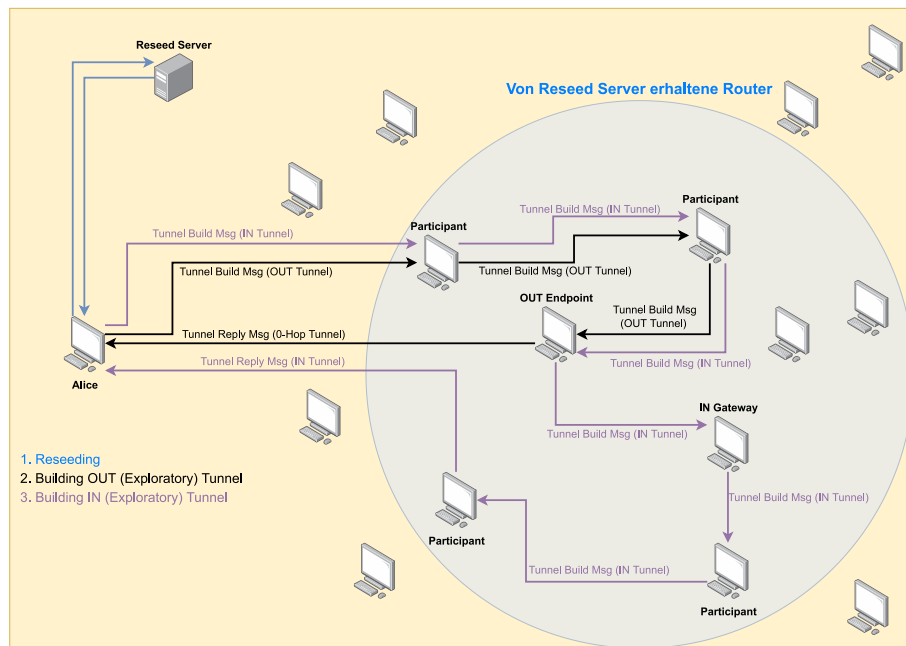


Abbildung 2.5.: Vereinfachte Visualisierung eines Bootstrapping-Vorgangs

Dieser Vorgang wird [Abbildung 2.5](#) nochmals grafisch dargestellt. Wie aus der Abbildung zu entnehmen ist, fragt Alice als Erstes beim Reseed Server nach einer Liste von existierenden [Router](#) (Pfeile in Blau) und erhält vom Reseed Server eine gewisse Anzahl Router (Computer innerhalb des blauen Kreises). Als Nächstes wird der Exploratory Outbound Tunnel erstellt (schwarze Pfeile). Da Alice noch keine Tunnels besitzt, an welche die *TunnelBuildReply* Message gesendet werden könnte, stellt Alice für diesen Zweck einen [0-Hop-Tunnel](#) zur Verfügung. Nachdem der Exploratory Outbound Tunnel erstellt wurde, wird der Exploratory Inbound Tunnel erstellt (violette Pfeile). Da bereits ein [Outbound Tunnel](#) existiert, wird die *TunnelBuild* Message über diesen versendet, um die Anonymität von Alice zu wahren. Nach dem Erstellen des Exploratory Inbound Tunnels besitzt Alice je einen Inbound und einen Outbound Tunnel, mit welchen sie ihre [Inbound Tunnels](#) und [Outbound Tunnels](#) anonym erstellen kann. Anschliessend ist das Bootstrapping abgeschlossen und Alice kann Verbindungen zu bestehenden I2P Seiten aufbauen.

2.7. Verbindung zu bestehenden I2P Seiten

Sobald ein [Router](#) seine Tunnels aufgebaut hat, kann er diese für die weitere Kommunikation verwenden. Um einen Dienst innerhalb des [I2P](#) Netzwerks zu erreichen, benötigt er die entsprechende B32 Adresse. Diese ist im übertragenen Sinn vergleichbar mit einem DNS Eintrag im Internet. Der Sender prüft in seiner lokalen [NetDB](#) ob er bereits einen [LeaseSet](#) des gesuchten Services kennt. Falls er diese Information noch nicht hat, fragt der Sender den [Floodfill](#) an, welcher gemäss [Kademlia](#) eine möglichst kleine Distanz zu dem gesuchten Services hat. Dieser Vorgang wird näher in [Abschnitt 2.5](#) beschrieben. Sobald der [Router](#) ein [Lease](#) und somit einen Inbound Tunnel gefunden hat, kann er seine Anfrage durch einen eigenen Outbound Tunnel an den Inbound Tunnel des gewünschten Ziels schicken. In [Abbildung 2.3](#) erkennt man gut, wie beide Seiten solche Tunnels besitzen und die jeweiligen Anfragen und Antworten durch diese verschicken.

2.8. Bekannte Angriffe auf I2P

Während der Recherche konnten mehrere Angriffe identifiziert werden. Bei den identifizierten Angriffen handelt es sich um Angriffe, welche in Peer-to-Peer-Netzwerken üblich sind und lassen sich in den folgend beschriebenen Angriffskategorien zusammen fassen.

2.8.1. Sybil Attacke

Bei einer Sybil Attacke versucht ein Angreifer ein Netzwerk zu übernehmen, indem er mehrere eigene Teilnehmer ins Netzwerk einbringt. Das Ziel dabei kann sein, durch das Kontrollieren von mehreren Nodes zusätzliche Informationen zu erhalten oder auch Schaden am Netzwerk anzurichten. Ein bekannteres Beispiel ist das Beeinflussen oder die komplette Übernahme des [Konsensus](#) einer Blockchain. Ein Angreifer versucht durch das Einbringen vieler eigener Sybil-Nodes eine Mehrheit im Netzwerk zu gewinnen, um dadurch Einfluss zu nehmen, welche Transaktionen geschrieben werden und welche nicht. Sybil Angriffe sind im Bereich von Peer-to-Peer-Netzwerken äusserst bekannt. John R. Douceur beschreibt in seinem Paper über die Sybil Attacke [12] dass ein Sybil Angriff, ohne zentrale Stelle fast unmöglich zu verhindern ist. Dies gilt zum Beispiel für das [I2P](#) Netzwerk, da diese ohne zentrale Leitstelle auskommen soll.

2.8.2. Eclipse Attacke

Eine Eclipse Attacke ist der beschriebenen Sybil Attacke ähnlich. Der Hauptunterschied liegt darin, dass bei einer Eclipse Attacke gezielt ein bestimmtes Ziel attackiert wird [43]. In Paper von Yuval Marco et al [29] wird eine Eclipse Attacke als eine vollständige Übernahme und Kontrolle der Kommunikation eines Netzwerkteilnehmers beschrieben, welcher auf dem von [I2P](#) verwendeten Kademlia Protokoll basiert.

Eclipse Attacken werden oft in Kombination mit einer Sybil Attacke durchgeführt. Im Gegensatz zu einer Sybil Attacke wird bei einer Eclipse Attacke jedoch nicht versucht ein komplettes Netzwerk zu beeinflussen, sondern nur ein kleiner Teil, oftmals nur ein einzelner Node. Dies wird erreicht, indem modifizierte Nodes im Netzwerk so platziert werden, dass das Opfer gemäss [Kademlia](#) nur über die Nodes des Angreifers mit dem restlichen Netz in Verbindung treten kann. Um ein einzelnes, spezifisches Ziel anzugreifen, wird bei einer Eclipse Attacke eine vergleichsweise kleine Anzahl Nodes benötigt. Gemäss aktuellem Wissensstand ist es ausreichend, modifizierte Nodes im unteren zweistelligen, je nach Angriff gar im einstelligen Bereich zu platzieren, um ein Node mittels Eclipse basiertem DoS Angriff kurzzeitig vom Netz zu nehmen. Bezogen auf [I2P](#) würden die Nodes den verschiedenen [Router](#) entsprechen.

Im I2P-Netzwerk können Eclipse Attacken durchgeführt werden, indem der Kademlia Routing Key des Opfers berechnet wird und anschliessend mehrere, manipulierte Floodfills nahe dem Routing Key des Opfers positioniert werden. Die manipulierten Floodfills sind so konfiguriert, dass sie keine [LeaseSets](#) verteilen. Dies führt dazu, dass der Teil des Netzwerks, welcher im Einflussbereich der manipulierten Floodfills steht, nicht mehr vom restlichen I2P-Netz erreichbar ist. Der Kademlia Algorithmus wird näher im [Abschnitt 2.5](#) erläutert.

2.8.3. Attacks on Bootstrapping

Das Bootstrapping ist ein weitverbreitetes Problem in Peer-to-Peer-Netzwerken. Wie der Name deuteten lässt, kommunizieren in einem Peer-to-Peer-Netzwerk die Peers direkt miteinander. Dazu müssen diese sich aber erst einmal kennen. Für I2P gibt es, wie auf der Webseite [34] der Entwickler beschrieben, mehrere Möglichkeiten, wie dieser initiale Schritt durchgeführt werden kann. Der allgemeine Begriff dafür ist Reseed. I2P bietet dafür automatisches reseed über Reseed Server oder auf dateibasiertes Reseeding an. Eine Reseed Server stellt RouterInformationen zur Verfügung, die von den bootstrappenden Routern heruntergeladen werden können. Dies passiert über das Internet mittels HTTP oder HTTPS Verbindung. Ein Reseed Server kann von jeder Person betrieben und verwaltet werden. Eine von den Entwicklern der Router Implementationen getroffene Auswahl dieser Reseed Server sind in den beiden Router Implementationen fest hinterlegt.

2.8.4. Traffic-Analyse

Bei der Traffic-Analyse wird versucht, ein Muster innerhalb des Netzwerkverkehrs zu finden. Dies können sowohl unterschiedliche Datenmengen als auch unterschiedlich grosse Datenpakete sein. Anhand dieser Informationen wird versucht zu erkennen, was ein Client macht oder mit wem er kommuniziert.

3. Methoden

3.1. Kanban

Zu Beginn der Arbeit wurden die gewonnenen Erkenntnisse mit Kanban festgehalten und organisiert. Mit Ansätzen aus der Aktionsforschung wurde nach neuen Informationen, Angriffsvektoren und weiteren Erkenntnissen bezüglich [I2P](#) geforscht.

Kanban wurde zu Beginn für die Projektplanung, wie auch für das Dokumentieren der Erkenntnisse über [I2P](#) und dessen Angriffsvektoren verwendet. Für jede einzelne Idee und jedes Erkenntnis wurde eine neue Karte erstellt, in welcher alle gewonnenen Informationen mittels Kommentaren und Kategorisierungen festgehalten wurden.

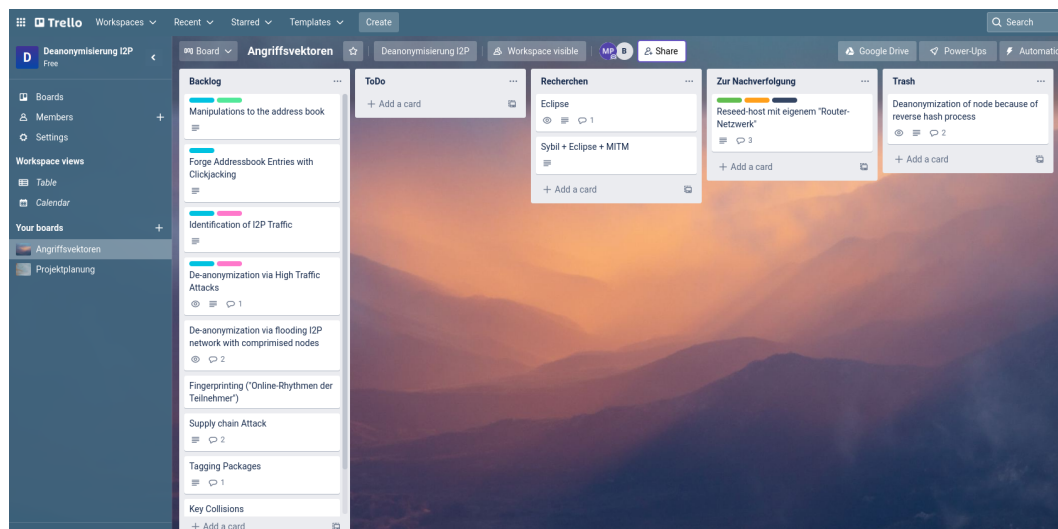


Abbildung 3.1.: Trello: Kanban-Board für die Projektplanung und Ideen-Datenbank

3.2. Aktionsforschung

Die Aktionsforschung als Forschungsmethodik wurde aus mehreren Gründen gewählt. Wie auf der Webseite der [Zürcher Hochschule der Künste \(ZHdK\)](#) [25] beschrieben, ist eines der Kernelemente der Aktionsforschung, die Aufhebung der strikten Trennung zwischen Forschern und Forschungsobjekt. So setzt die Aktionsforschung voraus, dass die Forschenden aktiv mit dem Forschungsobjekt interagieren, und selbst Akteur innerhalb des Forschungsobjektes sind. Weiter werden gewonnenen Informationen aus der Recherche ins Forschungsobjekt eingeführt. Dadurch können gewonnenen Erkenntnisse direkt in Forschungsprojekt integriert werden und es kann beobachtet werden, wie sich die neu gewonnenen Erkenntnisse auf das Forschungsobjekt auswirken. Ursprünglich wurde davon ausgegangen, dass mit diesem Vorgehen effizienter ermittelt werden kann, welche Angriffe einfach durchzuführen und zu mitigieren sind und welche nur unter hohem Aufwand oder gar nicht.

3.3. Explorative Forschung

Innerhalb der ersten Wochen der Arbeit zeigte sich jedoch, dass die geplante Methodik nicht zielführend ist. Einerseits konnten die gewonnenen Erkenntnisse aufgrund ihrer Komplexität nicht komplett mittels Kanban-Karten abgebildet werden. Es war deshalb nötig, zusätzliche Dokumente zu führen, in welchen die weiteren Informationen festgehalten wurden. Andererseits zeigte sich auch, dass zu wenig weitere Informationen (wie Forschungsarbeiten) vorlagen, um eine sinnvolle Aktionsforschung durchführen zu können. Aus diesen Gründen wurde auf die explorative Forschung gewechselt. Basierend auf der Beschreibung des [BWL-Lexikon](#) [4] und den gewonnenen Erfahrungen, eignet sich dies Vorgehen besser für die geplanten Arbeiten. Die explorative Forschung ist darauf ausgelegt, unbekannte und wenig erforschte Forschungsgebiete zu erschliessen. Dies wird erreicht, indem keine standardisierten Methoden zum Einsatz kommen, sondern je nach aktueller Situation unterschiedlich agiert wird. Bei dieser Arbeit kamen insbesondere die folgenden Methoden zum Einsatz:

- Literaturrecherche
- Interview
- Expertengespräch
- Experimente und Beobachtungen am Forschungsobjekt

Die Kanban Karten wurden durch steckbriefartige Dokumente ersetzt. Für jeden Angriff wurde ein Dokument erstellt, welches mit den wichtigsten Informationen und Quellen abgefüllt wurde.

3.3.1. Literaturrecherche

Die primäre Quelle der Literaturrecherche ist die Website geti2p.net, auf welcher die Kernentwickler von [I2P](#) eine Wissensdatenbank erstellt haben. In dieser Wissensdatenbank sind einerseits die grundlegenden Konzepte von [I2P](#) erläutert [24]. Weiter sind die Spezifikationen der unterschiedlichen Datentypen und [I2NP](#)-Paketen dokumentiert [17]. Ebenfalls konnte auf einige weitere wissenschaftliche Arbeiten zurückgegriffen werden. So konnte während des Aufbaus des Wissens bezüglich [I2P](#) auf die Bachelorarbeit von Jens Müller [31] zurückgegriffen werden, in welcher er Möglichkeiten zum Information Gathering und Attack Evaluation bezüglich [I2P](#) beschrieb. Weitere nützliche Informationen konnten aus dem Paper von Evers et al. [14] entnommen werden, in welchem die bekanntesten Angriffe auf das Tor-Netzwerk zusammengetragen wurden. Ebenfalls hilfreich war die Bachelorarbeit von Christoph Egger [13], in welcher er 2013 beschrieb, wie zum damaligen Zeitpunkt ein erfolgreicher Deanonymisierungsangriff im [I2P](#) Netzwerk durchgeführt werden konnte.

3.3.2. Interview mit Christoph Egger

Da die vorliegende Arbeit mit der Bachelorarbeit von Christoph Egger thematische Ähnlichkeiten hat, wurde ein Interview mit Christoph Egger durchgeführt. Als Vorbereitung zum Interview wurden in einem Dokument mehrere Fragen zusammengestellt. Die Fragen bezogen sich insbesondere auf das Vorgehen von Herrn Egger bei seiner Bachelor Arbeit, wie auch zum Angriff [Übernahme der NetDB durch Bootstrapping](#), beschrieben in [Abschnitt 4.2](#), welcher zum damaligen Zeitpunkt der vielversprechendste Angriff zur De-Anonymisierung war.

Herr Egger wollte nicht, dass das Interview veröffentlicht wird. Die Aufzeichnungen wurden daher nur für interne Zwecke verwendet und sind dem Anhang nicht beigelegt. Die wichtigsten Erkenntnisse wurden folgendermassen zusammengefasst:

- Kompromittieren eines Reseed Servers hat eine kleinere Wirkung als ursprünglich angenommen.
- Die grösste Schwierigkeit beim Erstellen eines Testnetzwerkes ist das Bootstrapping. So setzt die Implementierung der [I2P](#) Router voraus, dass sie sich in ein funktionierendes Netz einklinken können. Beim Hochfahren eines Testnetzes ist jedoch noch kein funktionierendes Netz vorhanden.
- Die [RouterInfos](#) können manuell unter den Routern hin und her kopiert werden. So wissen die Router voneinander Bescheid und das Netzwerk kann gestartet werden.

3.3.3. Expertengespräch mit I2P Entwicklern

Es wurde ein Expertengespräch mit den beiden [I2P](#) Entwicklern zzz und zlatinb geführt. zzz ist gemäss [geti2p.net](#) [8] Projektmanager und Kern-Leitentwickler vom [I2P](#) Router. zlatinb ist Hauptentwickler von [MuWire](#) [2], einem Filesharingclient für [I2P](#). Weiter ist zlatinb auch Mitentwickler des [i2p.i2p](#) Router. Das Expertengespräch fand nach dem Interview mit Christoph Egger statt, wodurch Erkenntnisse aus dem Interview im Expertengespräch angesprochen werden konnten.

Als Vorbereitung zum Expertengespräch wurden mehrere Unklarheiten bezüglich [I2P](#) in einem Dokument zusammengefasst und mit möglichen Angriffen ergänzt. Vor dem Gespräch wurde es den Experten zugestellt. Das Dokument wurde während dem Gespräch als Leitfaden verwendet. Aufgrund des Umfanges konnten nicht alle Positionen besprochen werden.

Auch die beiden Experten wollten nicht, dass das Expertengespräch veröffentlicht wird. Die Aufzeichnungen wurden daher nur für interne Zwecke verwendet und sind dem Anhang nicht beigelegt.

Nachfolgend sind die wichtigsten Erkenntnisse des Gesprächs aufgelistet:

- Informationen über die Funktionsweise des Sybil Analyse Tools
- Das Erstellen eines Testnetzwerkes ist nicht einfach und stellt auch bei den [I2P](#) Entwicklern eine Herausforderung dar,
 - "The easiest way to run a test network is to use somebody else's test network"
 - zzz
- Der Angriff „Partial Tunnel Control“ ist eine Möglichkeit um Teilnehmer zu deanonymisieren
 - "More or less, we say that if you get two colluding routers in the same tunnel, it's game over [...]" - zzz
- Seitens [I2P](#) wurden bereits einiges an Ressourcen investiert, um die Sicherheit von [I2P](#) zu gewährleisten.
 - ... You know, you're going up against some people who have been master students, PhD students and paid researchers. And you're trying to come up with some novel attack, who no one ever thought of, and documented, in such a short time, well, we're not saying it can't be done, but yea, you are "just" undergraduates, right? So don't kill yourselves. [...]

3.3.4. Experimente und Beobachtungen am Forschungsobjekt

Zu Beginn wurde der von DIVA.EXCHANGE zur Verfügung gestellte Dockercontainer [10] in Betrieb genommen, um den ersten Kontakt mit I2P herzustellen. Um besser zu verstehen, wie I2P funktioniert und sich verhält, wurden im späteren Verlauf die beiden Router Implementationen i2p.i2p [18] und die Invisible Internet Protocol (daemon) [22] in Betrieb genommen. Für die Kommunikation mit den I2P Entwickler wurden auch Services wie das im Abschnitt 3.7.2 beschriebene Susimail verwendet. Um bewusst Datenverkehr zu verschicken und diese zu analysieren wurden ein HTTP Tunnel nach der Beschreibung der I2PD Dokumentation [20] erstellt. Dieser erlaubte es, eigene Webseiten im I2P Netzwerk zu Verfügung zu stellen und auf diese zuzugreifen.

3.4. Begriffsdefinitionen

Für diese Arbeit werden Begriffe verwendet, welche je nach Kontext unterschiedlich definiert werden. Nachfolgend werden diese Begriffe im Kontext dieser Arbeit erklärt und festgelegt, wie sie innerhalb der vorliegenden Arbeit zu verstehen sind.

3.4.1. Identität

Die Identität eines Teilnehmers ist seine öffentliche IP-Adresse im Internet.

3.4.2. Anonymisierung

Eine Identität gilt so lange anonym, wie kein Rückschluss von der öffentlichen IP-Adresse auf die B32-Adresse aus dem I2P Netzwerk gezogen werden kann. Ebenfalls darf kein Rückschluss von der B32-Adresse auf die öffentliche IP-Adresse gezogen werden können.

3.4.3. Deanonymisierung

Die Identität eines Teilnehmers ist deanonymisiert, sobald die Anonymisierung nicht mehr gewährleistet ist. Dies bedeutete, sobald ein Rückschluss von der öffentlichen B32-Adresse auf die öffentliche IP-Adresse möglich ist, wurde eine Identität deanonymisiert. Ebenso wenn ein Rückschluss von der B32-Adresse auf die öffentliche IP-Adresse gezogen werden kann.

3.5. Annahme: Perfekte Implementation

Gemäss dem Auftraggeber soll davon ausgegangen werden, dass die Implementation des I2P Protokolls perfekt ist. Dies bedeutet, dass nicht geprüft werden soll, ob die Implementation der Router Software fehlerfrei ist, sondern ob die eigentliche Logik Probleme aufweist. Begründet wurde diese Annahme, da ein Fehler in der Architektur, oder gar in der zugrundeliegenden Logik, wesentlich schwerer zu beheben ist, als ein Fehler in der Softwareimplementation. Dies gilt ebenfalls für alle weiteren Applikationen, welche im Zusammenhang mit dem I2P Netzwerk verwendet werden. Aufgrund dieser Annahme wäre ein Penetrationstest chancenlos und ist daher Out of Scope.

3.6. Anpassungen an der Zielsetzung

Aufgrund der gewonnenen Erkenntnisse während der Arbeit hat sich die Zielsetzung bezüglich der Durchführung einer Deanonymisierung geändert. Weiter wurde die Erstellung eines Testnetzwerks geprüft.

3.6.1. Durchführung einer Deanonymisierung

Um die Frage aus [Abschnitt 1.3](#) „Können Teilnehmer des I2P Netzwerk deanonymisiert werden?“ definitiv beantworten zu können, ist aufgrund der Komplexität des Forschungsobjekts eine praktische Durchführung nötig. Dies ist insbesondere der Tatsache geschuldet, dass bei einem theoretischen Nachweis mehrere Annahmen zu unbekannten Faktoren getroffen werden müssen. Aus diesem Grund können mittels theoretischen Nachweisen nur Thesen und Hypothesen aufgestellt, jedoch keine Nachweise erbracht werden. Bei den Recherchen stellte sich heraus, dass vonseiten der [I2P](#) Entwicklern bereits einiges an Ressourcen bezüglich Mitigationen von bekannten und Angriffen investiert wurden. Diese Erkenntnis wurde insbesondere im Expertengespräch durch die Aussage von zzz bestätigt: *«[...] You know, you're going up against some people who have been master students, PhD students and paid researchers. And you're trying to come up with some novel attack, who no one ever thought of, and documented, in such a short time, well, we're not saying it can't be done, but yea, you are "just" undergraduates, right? So don't kill yourselves. [...]»* Aufgrund dieser Erkenntnisse wurde unter Absprache mit dem Auftraggeber und der Betreuungsperson davon abgesehen, einen Nachweis einer Deanonymisierung zu erbringen. Anstelle eines Nachweises sollten Hypothesen bezüglich möglichen Deanonymisierungen von Angriffen aufgestellt werden. Mit diesen Hypothesen sollte aufgezeigt werden, mit welchen Massnahmen DIVA sicherer gegenüber Deanonymisierungsangriffen gemacht werden kann.

3.6.2. Testnetzwerk

Da eine Umsetzung von gewissen Angriffen im aktiven [I2P](#) Netzwerk sehr aufwendig ist, wurde geprüft, ob ein Testnetzwerk umgesetzt werden kann.

3.7. Verwendete Technologien

3.7.1 I2P Router Software

Als Software für die [I2P](#) Router wurden der [i2p.i2p](#) Router, [I2PD](#) und der [I2P](#) Docker-container [10] verwendet, welcher von DIVA.EXCHANGE zur freien Verfügung gestellt wurde.

3.7.2 Susimail

Die Kontaktaufnahme für das Expertengespräch mit den [I2P](#) Entwicklern geschah per E-Mail über das [I2P](#) Netzwerk. Um E-Mails über das [I2P](#) Netzwerk zu versenden, müsste beim E-Mail-Client der [I2P](#) Proxy hinterlegt werden. Dies bringt einen gewissen Konfigurationsaufwand mit sich. Im [i2p.i2p](#) Router ist in der Routerkonsole jedoch bereits der Mail Client Susimail mitgeliefert und vorkonfiguriert. Deswegen wurde Susimail verwendet.

3.8. Projektorganisation

Die Projektorganisation besteht aus den folgenden drei Parteien. Die Hochschule Luzern übernimmt in diesem Projekt die Betreuung und die Bewertung der Arbeit. Die DIVA.EXCHANGE ist der Auftraggeber, welcher die Arbeit ausgeschrieben hat und technisch Betreuung leistet. Das Projektteam ist die ausführende Partei und Ersteller dieses Dokuments.

Hochschule Luzern

- Dieter Arnold (Betreuungsperson)
- Jan Alsenz (Experte)

DIVA.EXCHANGE

- Konrad Bächler (Auftraggeber)

Projektteam

- Brian Boss (Student)
- Marco Purtschert (Student)

3.9. Umsetzung der Angriffe

Wie in den vorgängigen Kapiteln beschrieben, wurde nach Möglichkeiten zur Deanonymisierung gesucht, unter der Annahme, dass die Implementation perfekt ist, und ein Penetrationstest zu keinem Erfolg führen würde. Um einen Angriff dennoch umsetzen zu können, wurden folgende Ansätze verfolgt.

3.9.1. Abgrenzungen zu Tor

Das Tor-Netzwerk ist analog zu [I2P](#) ein Overlay-Netzwerk, welches jedoch eine grössere Bekanntheit hat. Weiter wurden bezüglich des Tor-Netzwerks mehr Papers veröffentlicht, als bei [I2P](#). Auch konnten bereits mehrere Deanonymisierungs-Angriffe im Tor-Netzwerk erfolgreich nachgewiesen werden. Deswegen wurde bei der Literaturrecherche auch Tor berücksichtigt. Es konnten jedoch nicht alle Erkenntnisse von Tor auf diese Arbeit übertragen werden, da es mehrere Unterschiede zwischen den Implementationen gibt.

Unidirektionale Tunnels

In I2P werden unidirektionale Tunnels verwendet. Das Senden und Empfangen von Daten wird mittels Inbound und Outbound Tunnels realisiert. Im Tor-Netzwerk geschieht die Kommunikation mittels bidirektionalen Tunnels. Das Senden und Empfangen von Daten geschieht über dieselben Tunnels.

(Nahezu) keine zentralen Autoritäten

Die Implementation von I2P sieht vor, dass möglichst alles dezentral umgesetzt wird. So sind die Reseed Server die einzigen zentralen Instanzen, welche für den Betrieb von I2P zwingend nötig sind. Tor hingegen kennt zentrale Autoritäten. Dadurch ist es einerseits einfacher, gewisse Angriffe zu verhindern, z. B. Sybil Angriffe. Andererseits bieten sich zentrale Autoritäten als Ziel für weitere Angriffe an.

Keine Exit Nodes

Eine Kernfunktion von Tor sind die Exitnodes, welche es den einzelnen Teilnehmern ermöglichen, anonym auf Ressourcen des Clearnets zugreifen zu können. Mehrere Angriffe des Tornetzwerks basieren auf diesen Exitnodes. In I2P existieren jedoch keine Exitnodes. In I2P gibt es jedoch sogenannte Outproxies, welche von einzelnen Teilnehmern betrieben werden und eine ähnliche Funktion ermöglichen. Da in DIVA keine solche Outproxies verwendet werden, wurden auch keine Angriffe betreffend Exitnodes analysiert.

3.9.2 Angriffsideen

Durch eine umfassende Literaturrecherche wurde das Wissen zu I2P fortlaufend erweitert. Neue Angriffsmöglichkeiten wurden sogleich steckbriefartig festgehalten. Zu diesem Zeitpunkt wurde noch nicht überprüft, ob der Angriff überhaupt möglich ist.

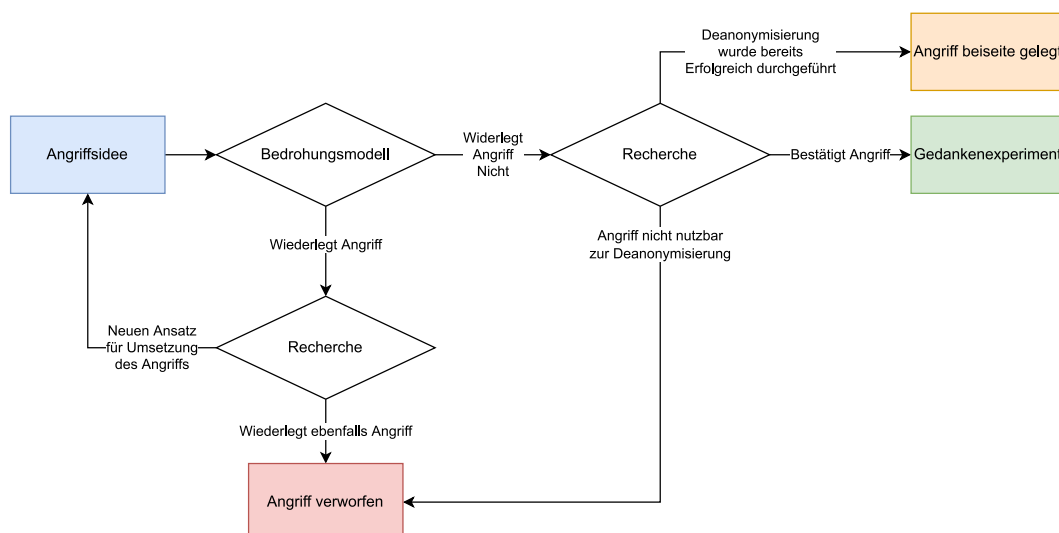


Abbildung 3.2.: Abarbeitung von Angriffsideen

Wie in Abbildung 3.2 ersichtlich, wurde die Angriffsidee zuerst mit dem Bedrohungsmodell [23] der I2P Entwicklern abgeglichen. Sollte das Bedrohungsmodell den Angriff

widerlegt haben, wurde mittels Recherche nach weiteren Möglichkeiten gesucht, ob die grundlegende Idee des Angriffs anderweitig umgesetzt werden könnte. Konnte der Angriff nicht direkt widerlegt werden, wurde weiter recherchiert. Falls während der Recherche festgestellt wurde, dass die Angriffsidee bereits erfolgreich in einem Deanonymisierungsangriff belegt wurde, wurde sie beiseite gelegt. Verworfen wurde die Angriffsidee, falls bei der Recherche festgestellt wurde, dass der Angriff entweder nicht durchführbar ist, oder nicht für eine Deanonymisierung verwendet werden kann. Falls die Idee weder beiseite gelegt, noch verworfen wurde, wurde mittels Gedankenexperiment versucht, den Angriff konzeptionell durchzuführen.

Die relevantesten Angriffsideen sind in [Kapitel 4](#) dokumentiert.

3.9.3 Angriffskonzepte

Mittels Brainstorming an Whiteboards, Interviews, Expertengesprächen und weiteren Recherchen (wie Sourcecodeanalyse) wurden aus den Ideen Gedankenexperimente erstellt, mit welchen wiederum die Konzepte für einen Deanonymisierungsangriff aufgestellt und ausgearbeitet werden konnten. Während dem Ausarbeiten der Gedankenexperimente wurden die meisten Angriffsideen verworfen, da sich oft herausstellte, dass die Entwickler von [I2P](#) bereits Mitigationen bezüglich den Angriffen implementiert haben. So wurde ebenfalls die Erkenntnis gewonnen, dass viele der einfach durchzuführenden Angriffe mitigiert wurden, was die praktische Durchführung einer Deanonymisierung im Rahmen dieser Arbeit verunmöglichte. Während dem Expertengespräch erbrachte „zlatinb“ den Hinweis, dass anstelle eines praktischen Nachweises auch eine mathematische Modellierung als Nachweise einer Deanonymisierung dienen könnte.

Die relevantesten Angriffskonzepte sind in [Kapitel 4](#) dokumentiert.

Nach der Ausarbeitung der Angriffskonzepte stellte sich heraus, dass der Angriff der Tunnelkontrolle, beschrieben in [Abschnitt 4.3](#), und die Mustererkennung von Datenverkehr, beschrieben in [Abschnitt 4.5](#), die vielversprechendsten Angriffsvarianten darstellten. Von diesen beiden Angriffsmöglichkeiten wurde die Tunnelkontrolle als wahrscheinlicheren Deanonymisierungsangriff betrachtet, da dieser durch eine einzelne Zelle durchgeführt werden könnte. Für eine Mustererkennung von Datenverkehr ist nach ersten Erkenntnissen eine internationale Zusammenarbeit von Internet Providern nötig. Weiter wurde eine Form des Angriffs im Paper von Müller [31], im Kapitel „6.1.5. Deanonymization via High Traffic Attacks“ behandelt. Angesichts dessen wurde der Fokus in der Realisierung auf den Angriff der Tunnelkontrolle gelegt.

3.9.4 Mathematische Modellierung

Zur mathematischen Modellierung wurde mehrere Papers beigezogen, welche wichtige Erkenntnisse lieferten. Die wichtigsten Papers sind einerseits jenes von Tan et al. [36], in welchem mittels mathematischen Modellierungen Eclipsattacks im Tor Netzwerk simuliert wurden. Weiter von Bedeutung war das Paper von Gui et al. [14], in welchem Sybill Attacks in Ad-Hoc Routing Protokollen mathematisch analysiert und modelliert wurden. Auch wurde das Paper von Casey et al. [5] berücksichtigt, in welchem mit Ansätzen aus der Spieltheorie Sybil Angriffe mathematisch simuliert wurden. Die mathematische Modellierung, [Unterabschnitt 5.1.1](#), erfolgte aufgrund der Erkenntnisse aus diesen Arbeiten.

3.9.5. Experiment: Testnetzwerk

Für die Umsetzung eines Testnetzwerks wurde geprüft, ob bereits funktionierende Implementationen existieren. Dafür wurden verschiedenen Plattformen wie GitHub und Codeberg durchsucht. Ebenfalls wurde uns von Dieter Arnold die Arbeit von Moritz Küttel [26] zur Verfügung gestellt. Das Hauptziel dieser Arbeit waren Performance-Messungen innerhalb von [I2P](#). Innerhalb dieser Arbeit wurden mehrere Möglichkeiten für die Erstellung eines Testnetzwerks erarbeitet. Ebenfalls wurde die Lösung des Entwickler LNS [27] auf GitHub geprüft. Dieser bietet ein Python Script, welches mit nativen Linux Tools mehrere Router auf dem lokalen System erstellen kann.

4. Ideen und Konzepte

Einleitend bei den Unterkapiteln sind die wichtigsten Informationen zusammengefasst. Die Kategorie beschreibt die Art des Angriffs. In Ausarbeitung wird beschrieben, ob es sich jeweils um eine Angriffsidee oder ein Angriffskonzept handelt. Eine Angriffsidee beschreibt einen eher einfachen Angriff, welcher nicht weiterverfolgt wurden. Bei einem Angriffskonzept handelt sich um eine oder mehrere Angriffsideen, welche weiter ins Detail analysiert und zu einem komplexen Konzept ausgearbeitet wurden. Eine Angriffsidee wurde nicht zu einem Angriffskonzept ausgearbeitet, wenn bereits früh erkannt wurde, dass die Angriffsidee von den [I2P](#) Entwickler bedacht und mitigiert wurde.

4.1. Tunnel-ID: Rückschluss auf Pos. in Tunnel

| | |
|-------------------------|--|
| Kategorie | Traffic Analyse |
| Ausarbeitung | Angriffsidee |
| Ersteinschätzung | Angriffsidee wurde verworfen, da die einzelnen Teilnehmer untereinander unterschiedliche Tunnel-IDs aushandeln. Daher ist ein Angriff nicht umsetzbar. |

Damit [Router](#) innerhalb eines Tunnels wissen, an welchen [Router](#) sie das Paket weiterleiten müssen, wir ihnen eine Tunnel-ID zugewiesen. In den empfangenen Paketen ist die Tunnel-ID in den Delivery Instruction enthalten. Durch diese Information kann der [Router](#) das Paket korrekt weiterleiten. Im [Lease](#) und somit im [LeaseSet](#) ist festgehalten, welche Tunnel-ID für diesen Tunnel zu verwenden ist.

4.1.1 Angriffsidee

Jeder Teilnehmer eines Tunnels erhält eine Tunnel-ID. Falls diese für jeden Teilnehmer im selben Tunnel die gleiche ist, könnte ein einfacher Lookup Angriff durchgeführt werden. Ein Angreifer wartet, bis sein [Router](#) in einem Tunnel eingebunden ist. Sobald er seine Tunnel-ID erhalten hat, kann er in seiner [NetDB](#) nach Leases mit derselben Tunnel-ID suchen. Falls ein Lease gefunden wird, kann von diesem auf das [LeaseSet](#) geschlossen und somit die B32-Adresse ermittelt werden. Weiter wird somit aufgedeckt, dass es sich beim aktuellen Tunnel um einen Inbound Tunnel handelt. Zusätzlich kann ermittelt werden, ob der Teilnehmer unmittelbar vor dem eigenen [Router](#), als [Inbound Gateway](#) agiert. Falls dies nicht der Fall ist, kann vermutet werden, dass der eigene [Router](#) der letzte Teilnehmer ist und somit direkt mit der Destination kommuniziert. Die Vermutung beruht auf der Annahme, dass der Tunnel aus drei Teilnehmern besteht.

4.1.2 Durchführbarkeit

Dieser Angriff funktioniert nicht, da, wie in der Tunnel Routing Dokumentation [41] beschrieben, jeder Teilnehmer eine eigene Tunnel-ID erhält. Jeder [Router](#) kennt nur

seine Tunnel-ID und die des nächsten Teilnehmers. Diese muss bekannt sein, damit der nächste Teilnehmer das Paket dem korrekten Tunnel zuordnen kann. Da die Tunnel-ID des vorherigen **Routers** nicht bekannt ist, ist es nicht möglich einen Tunnel anhand der eigenen Tunnel-ID zu rekonstruieren und einer Destination zuzuweisen.

4.2. Übernahme der NetDB durch Bootstrapping

| | |
|-------------------------|--|
| Kategorie | Bootstrapping |
| Ausarbeitung | Angriffskonzept |
| Ersteinschätzung | Angriffskonzept wurde verworfen. Gemäss dem Konzept wäre zwar eine Deanonymisierung theoretisch möglich, jedoch sehr aufwendig. Weiter ist der Angriff nur beim ersten Reseeding erfolgsversprechend. Da Angriffskonzepte aufgestellt wurden, welche einfacher durchzuführen und einen grösseren Erfolg versprochen, wurde dieses Konzept nicht weiter verfolgt. |

Ein **Router**, welcher neu ins **I2P** Netzwerk verbindet, fragt im Normalfall, wie in **Abchnitt 2.6** beschrieben, einen Reseed Server nach initialen **RouterInformationen** an. Dieser schickt dem bootstrappenden **Router** ein Paket, bestehend aus zufällig zusammengestellten **RouterInformationen**. Mit diesen kann der bootstrappende **Router** seine initiale Verbindung ins **I2P** Netzwerk bewerkstelligen. Weiter fragt der **I2PD Router** gemäss Implementation nur solange Reseed Server an, bis ein interner Threshold erreicht wurde. Falls es für einen manipulierten Reseed Server möglich ist, den beschriebenen Threshold mit einem Reseed-Antwort-Paket zu erreichen, würde für den Angriff ein einziger, manipulierter Reseed Server reichen. Der Reseed Prozess kann ebenfalls ausgelöst werden, falls der Threshold während dem normalen Betrieb unterschritten wird, oder zu wenige Floodfills bekannt sind. Dies variiert jedoch je nach **Router** Implementation.

4.2.1 Angriffsidee

Für diesen Angriff wird davon ausgegangen, dass ein in der **Router** Implementation integrierter Reseed Server unter der Kontrolle eines Angreifers ist. Bei einer Anfrage kontrolliert der besagte Reseed Server, welche RouterInfos dem anfragenden **Router** zurückgeben werden. Für eine vollständige Übernahme der **NetDB** wird von einer leeren **NetDB** ausgegangen. In diesem Zustand ist die **NetDB** jedoch nur beim initialen Bootstrap eines **Routers**. Ebenfalls erwähnten die **I2P** Entwickler während dem Expertengespräch einen Bug in der aktuellen Java-Implementation, welcher dazu führen kann, dass die **NetDB** verworfen wird.

Mit dem beschriebenen Angriff ist es für einen Angreifer möglich, die **NetDB** eines **Routers** vollständig mit manipulierten **Routern** zu füllen. Damit kontrolliert der Angreifer alle **Router** während dem initialen Tunnelaufbau. Dadurch ist es für den Angreifer möglich, die Tunnels zu rekonstruieren und somit das **RouterInfo** und entsprechend die IP-Adresse des Opfers zu identifizieren. Das Opfer wird nun sein **LeaseSet** zusammenstellen und dieses den Floodfills übermitteln, welche auch unter Kontrolle des Angreifers stehen. Der Angreifer kann das **LeaseSet** anhand der im Lease beschriebene Tunnels, von welchem er alle Informationen kennt, auf das Opfer zurückführen. Von diesem **LeaseSet** kann nun die B32-Adresse extrahiert werden.

Für einen Angreifer ist es mit diesem Angriff möglich, ein Opfer vollständig zu deanonymisieren.

misieren. Der Angreifer erhält alle notwendigen Informationen, um die IP-Adresse und die B32 Adresse des [Router](#)s miteinander zu verbinden.

4.2.2 Durchführbarkeit

Es wurden keine Mitigationen gefunden, welchen eine Übernahme der [NetDB](#) verhindern würden. Aufgrund der Komplexität ist ein solcher Angriff aber sehr schwierig umzusetzen. Die Hauptschwierigkeit bei einem Angriff dieser Art besteht darin, einen Reseed Server entweder zu übernehmen oder ihn in die [Router](#) Implementationen zu integrieren. Beide Möglichkeiten sind nicht auszuschliessen, benötigen aber eine längere Vorlaufzeit und genügend Mittel. Die Entwickler des [I2P Routers](#) haben zudem weitere Massnahmen ergriffen, mit welchen ein malicious Reseed Server erkannt werden soll. Die Massnahmen sind in der Dokumentation zur [NetDB](#) [37] in Kapitel *Bootstrap Attacks* beschrieben. Eine der Massnahme ist ein System zur Erkennung von verdächtigen Reseed Paketen. Weiter wird der Angriff erschwert, indem eine persistente Kopie der [NetDB](#) angelegt wird. Dadurch sollte der Reseeding Prozess nur bei der initialen Inbetriebnahme des [Router](#) notwendig sein. Da ein solcher Angriff wie beschrieben eine grössere Vorlaufzeit benötigt, eine höhere Komplexität mit sich zieht und schlussendlich gemäss aktuellem Wissensstand nur Teilnehmer deanonymisiert werden können, welche das allererste Mal eine Verbindung zu [I2P](#) erstellen, wird dieser Angriff innerhalb dieser Arbeit nicht weiter verfolgt.

4.3. Tunnelkontrolle

| | |
|-------------------------|---|
| Kategorie | Sybil Attack |
| Ausarbeitung | Angriffskonzept |
| Ersteinschätzung | Der Angriff wird in der Realisierung weitergezogen. So wird nach ersten Erkenntnissen vermutet, dass mit diesem Angriff eine gezielte Deanonymisierung einzelner Teilnehmern innerhalb des DIVA.EXCHANGE Netzwerks möglich ist. |

Falls ein Angreifer alle [Router](#) in einem Tunnel kontrolliert, kann er diesen vollständig überwachen. Ein Angreifer sieht jeglichen Datenverkehr, sowohl den eingehenden wie auch den ausgehenden. Dadurch kann er nachvollziehen, von wo der Datenverkehr kommt und wohin er geht. Dies ist ein bekanntes Problem bei allen Tunnel basierten Anonymisierungsnetzwerken. Demzufolge wurden Mitigationmassnahmen implementiert, welche eine solche Konstellation von [Routern](#) erschweren sollen. So soll in [I2P](#) kein [Router](#) wissen, welche Position er im Tunnel besitzt. Ausnahme dieser Regel sind die Endpoints und Gateways. Dies müssen aufgrund ihrer Aufgabe die Position im Tunnel kennen und wissen, ob sie in einem Inbound oder Outbound Tunnel sind. Die Standardeinstellungen aller gängigen [Router](#) Implementationen sieht zudem eine Länge von drei Hops vor. Falls ein Teilnehmer in einem Tunnel ermitteln kann, dass er der dritte Hop in einem Inbound Tunnel, oder der erste in einem Outbound Tunnel ist, besteht eine hohe Wahrscheinlichkeit, dass er direkt mit dem Tunnel-Besitzer kommuniziert.

4.3.1 Angriffsidee

Der grundlegende Gedanke ist, dass manipulierte [Router](#) in einen Inbound oder Outbound Tunnel eines Ziels eingeschleust werden, welche anschliessend Informationen über den

Besitzer der Tunnels sammeln. In Kombination mit weiteren Informationen, wie dem [LeaseSet](#) könnte eine Deanonymisierung eines Teilnehmers erreicht werden.

Abbildung 4.1 zeigt den grundlegenden Aufbau eines solchen Angriffs auf. Die beiden [0-Hop-Tunnels](#) auf der Seite des Angreifers (Eve) vereinfachen den Aufbau und die Umsetzung, jedoch mit dem Nachteil, dass die Identität des Angreifers preisgegeben wird. In der Grafik hat der Angreifer alle Tunnel Teilnehmer erfolgreich übernommen und kann somit den Tunnel rekonstruieren und somit die Verschleierung rückgängig machen.

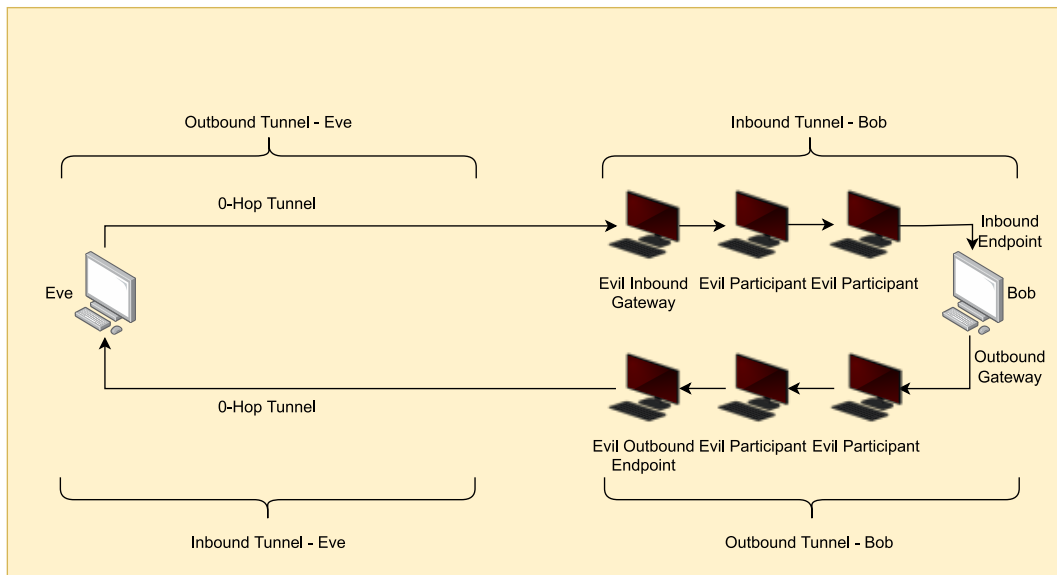


Abbildung 4.1.: Vollständige Tunnelkontrolle

Falls die obige Konstellation des Inbound oder Outbound Tunnels erreicht wurde, ist die [RouterInfo](#) und infolgedessen die IP-Adresse des Ziels bekannt. Da im obigen Beispiel Eve eine Verbindung zum Ziel aufgebaut hat, kennt Eve ebenfalls das [LeaseSet](#) und somit die B32-Adresse von Bob. Mit diesen Informationen ist es für Eve möglich, Bob zu deanonymisieren.

Ist die Kontrolle eines Inbound Tunnels von einem Ziel übernommen, kann versucht werden, aus bekannten [LeaseSets](#) die B32 Adresse des Ziels zu ermitteln. Dies kann erreicht werden, indem der Angreifer in seiner [NetDB](#) nach dem [Lease](#) des kontrollierten Tunnels sucht und diesen [Lease](#) dem entsprechenden [LeaseSet](#) zuordnen kann. Falls es sich beim Ziel um einen Teilnehmer des DIVA.EXCHANGE Netzwerkes handelt, dann kann dessen B32-Adresse aus der DIVA-Blockchain entnommen werden. Mit dieser B32 Adresse kann anschliessend das [LeaseSet](#) bei einem Floodfill angefordert werden. Falls die IP-Adresse und Tunnel-ID des von Eve kontrollierten [Inbound Gateway](#) in einem [LeaseSet](#) auftaucht, wurde das [LeaseSet](#) von Bob und entsprechend seine B32-Adresse identifiziert. Nun wäre ebenfalls eine Deanonymisierung von Bob möglich.

Dieser Angriff ist auch in Outbound Tunnels möglich, jedoch komplizierter in der Umsetzung. So müsste Eve eine legitime Anfrage an Bob senden, von welchem sie eine Antwort erwartet. Nach Erhalt der Antwort kann Eve den Outbound Tunnel an Bob zuweisen. Wichtig ist jedoch in diesem Fall, dass Eve über einen [0-Hop-Tunnel](#) kommunizieren muss, da sie ansonsten die IP-Adressen der Ziel-Outbound-Endpoints nicht auswerten kann.

Eine weitere Herausforderung ist, wie die **Router** in die Tunnels eingeschleust werden können. Eine Möglichkeit wäre, möglichst viele Router mittels Sybil Angriff ins **I2P** Netz einzuschleusen und anschliessend darauf zu warten, dass per Zufall ein Tunnel nur aus manipulierten **Routern** besteht. Dies wird jedoch als äusserst unwahrscheinlich eingeschätzt. Deswegen wurden folgende Ansätze aufgestellt, mit welchem der oben beschriebene Vorgang effizienter umgesetzt werden könnte.

4.3.2 Ansatz 1 - Partielle Tunnelkontrolle

Die Idee dieses Ansatzes entstand aus der Erkenntnis, dass nicht der komplette Tunnel übernommen werden muss. In der Standardkonfiguration von drei Hops, reicht es wenn der erste und der letzte **Router** im selben Tunnel übernommen werden kann. In einem Tunnel mit drei Hops wird dies möglich, wenn der manipulierte Participant-**Router** erkennt, dass er über den gleichen Participant **Router** kommuniziert, wie der manipulierte Gateway-**Router**. Ermöglicht wird dies, da die Tunnel Operation vorsieht, dass sich die Kommunikationspartner in einem Tunnel untereinander kennen. Wenn nun alle Evil Nodes die Informationen bezüglich ihres Kommunikationspartners einem zentralen Analyseserver mitteilen, könnte dieser die Tunnels rekonstruieren.

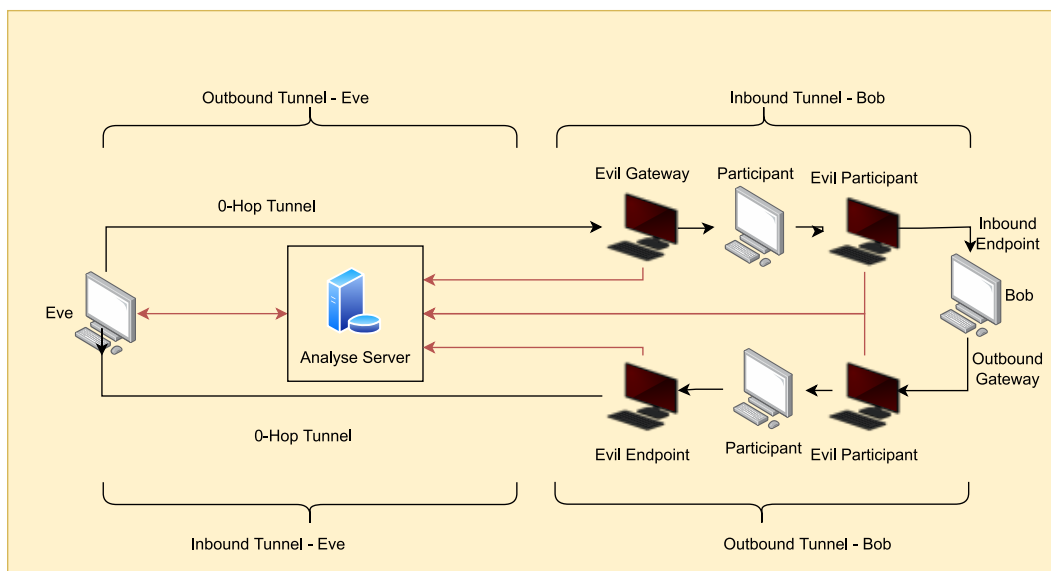


Abbildung 4.2.: Partielle Tunnelkontrolle

Bei diesem Verfahren gilt jedoch zu beachten, dass ein Participant in mehreren Tunnels teilnehmen kann. Demzufolge ist es möglich, dass ein malicious Gateway und ein malicious Participant mit demselben Router kommunizieren, obwohl sie nicht im selben Tunnel sind. Mittels Timing Attacks könnte jedoch überprüft werden, ob die betroffenen Teilnehmer im selben Tunnel sind.

4.3.3 Ansatz 2 - DoS auf Participants

Ein Tunnel in **I2P** hat eine maximale Laufzeit von 10 Minuten und wird anschliessend automatisch erneuert. Falls ein oder mehrere Teilnehmer beim Erneuerungsprozess nicht mehr zur Verfügung stehen, werden neue Teilnehmer für einen neuen Tunnel kontaktiert.

Beim Expertengespräch mit den I2P Entwicklern wurde in Erfahrung gebracht, dass die verbleibenden Teilnehmer in den neuen Tunnel übernommen werden. Der Hintergrund ist, dass möglichst wenig Rotation mit den Teilnehmern gewünscht ist. Dadurch wird einem Angreifer erschwert, manipulierte **Router** einzuschleusen. Die Idee des zweiten Ansatzes sieht vor, dass mit einem gezielten **DoS** ein oder mehrere **Router** aus einem Tunnel gedrängt werden. Diese sollen durch malicious **Router** ersetzt werden. **Router** mit einem **DoS** aus einem Tunnel zu verdrängen, ist gemäss dem Bedrohungsmodell von I2P ein bekannter Angriff und ist als Buddy Exhaustion bekannt. Buddy Exhaustion wird detailliert im Paper von Wang et al. [42] in der Sektion 5.2 beschrieben.

Um eine B32 Adresse zu deanonymisieren, könnte der Angreifer mit einem gezielten **DoS** den **Inbound Gateway** des Opfers dazu bringen, den Tunnel nach Ablauf der 10 Minuten nicht mehr zu erneuern. Das Opfer müsste dadurch einen neuen **Router** anfragen. Der Angreifer kann nun überprüfen, ob der neue Inboundgateway zu einem eigenen, manipulierten **Router** gehört. Falls dies nicht der Fall ist, kann der Angriff beliebig oft wiederholt werden. Anschliessend kann der nächste Teilnehmer des Tunnels auf die genau gleiche Art ausgetauscht werden. So kann schrittweise der Tunnel mit eigenen, manipulierten Routern übernommen werden, wodurch das **RouterInfo** und somit die IP-Adresse des Ziels identifiziert wird.

Bei diesem Vorgehen gilt ebenfalls zu beachten, dass die Teilnehmer in einem Tunnel mit einem nicht deterministischen Directory-Sort-Algorithmus sortiert werden. Wenn somit der **Inbound Gateway** eines Tunnel mit einem manipulierten **Router** ausgetauscht wird, wird der eingeschleuste **Router** nicht zwangsläufig zum Inboundgateway, sondern könnte auch als letzter Router im Tunnel eingesetzt werden.

4.3.4 Kombination der beiden Ansätze

Die Ansätze 1 & 2 könnten auch kombiniert werden. Dies hätte einen höheren Vorbereitungsaufwand zur Folge. Jedoch könnte der Angriff anschliessend effizienter durchgeführt werden. So könnte, wie in Ansatz 2 beschrieben, ein Evil **Router** beim Ziel als **Inbound Gateway** platziert werden. Anschliessend kann mit Ansatz 1 überprüft werden, ob der dritte **Router** bereits ein eigener, manipulierter **Router** ist.

4.3.5 Durchführbarkeit

Gemäss der Literaturrecherche, dem Interview und dem Expertengespräch konnten keine Informationen gesammelt werden, welche auf eine Verunmöglichung des Angriffs hindeuten würden. Der Angriff kann lediglich erschwert werden. Deswegen wurde der Angriff als durchführbar eingestuft und wird in dieser Arbeit weiterverfolgt.

4.4. Zero Hop Tunnels: Reseed Server

| | |
|-------------------------|---|
| Kategorie | Bootstrapping und Timing |
| Ausarbeitung | Angriffskonzept |
| Ersteinschätzung | Das Angriffskonzept wurde verworfen, da die erfolgreiche Durchführung, wie auch der Nutzen des Resultats als gering eingestuft wurde. |

Sobald ein Client von einem Reseed Server seine ersten initialen **RouterInformationen**

erhalten hat, kann er mit dem Aufbau seines Netzwerks beginnen, bestehend aus Outbound, Inbound und Exploratory Tunnels. Dies ist zwingend notwendig, um später anonym zu kommunizieren. Das Problem dabei ist, dass beim Erstellen des ersten Outbound Tunnels der Router direkt mit dem ersten Nodes kommunizieren muss, da er noch nicht in Besitz weiterer Tunnels ist. Um dies zu verschleiern, kommuniziert der Router über einen 0-Hop-Tunnel. Der Router verhält sich wie in einem normalen Tunnel und gibt somit vor, dass kein 0-Hop-Tunnel verwendet wird. Falls ein Angreifer dies Verhalten durchschauen könnte, wäre die Anonymität gefährdet.

4.4.1 Angriffsidee

Ein von einem Angreifer kontrollierter Reseed Server zeichnet alle Anfrage auf, welche er erhält. Zusätzlich stellt der Angreifer mittels Sybilangriff eine hohe Anzahl von manipulierten Routern im I2P Netzwerk zu Verfügung. Der Angreifer könnte nun die Anfragen an den Reseed Server mit den Anfragen an die manipulierten Router korrelieren. Der Angreifer überprüft nun, ob die Absender IP-Adresse bei der Anfrage zu einem Tunnelaufbau die gleiche ist, wie die einer zeitnahen Abfrage auf dem Reseed Server. Trifft dies zu, kann angenommen werden, dass es sich um einen Router handelt, welcher mittels 0-Hop-Tunnel seinen initialen Tunnel aufbaut.

Wie in den Tunnel Creation Spezifikation [38] beschrieben, baut ein Router immer einen Exploratory Outbound Tunnel als Erstes auf. Somit ist ebenfalls bekannt, dass der Router Angreifer in einem Exploratory Outbound Tunnel ist.

4.4.2 Durchführbarkeit

Dieser Angriff wäre möglich, nur sind sowohl die Erfolgchancen wie auch das Resultat nicht hilfreich. Beim erfolgreichen Ausführen dieses Angriffs, könnte ein Router wissen, dass er der erste Teilnehmer innerhalb eines Exploratory Outbound Tunnels ist. Der Exploratory Outbound Tunnel hilft aber nicht bei einer Deanonymisierung, da von einem Exploratory Tunnel nicht auf eine B32-Adresse geschlossen werden kann. Aus einem Blog-Eintrag [11] der DIVA.EXCHANGE-Entwickler geht hervor, dass Anfragen an Reseed Server sehr hoch sind, 100'000 - 150'000 Anfragen pro Monat. Dies würde bedeuten, dass ungefähr jeder der aktiven Router diesen Reseed Server 5 Mal angefragt haben müsste. Es ist anzunehmen, dass nicht nur neue Router sondern auch bereits vernetzte Router Reseedabfragen erstellen. Gemäss dem Expertengespräch mit den I2P Entwicklern sollte so häufige Reseedabfragen nicht notwendig sein, da diese ihre bekannten RouterInformationen persistent speichern und nach einem Neustart wieder verwenden können. Eine genauere Analyse dieses Phänomen wäre möglicherweise interessant, wird aufgrund von Zeitmangel und Komplexität jedoch nicht weiterverfolgt.

4.5. Mustererkennung Datenverkehr

| | |
|-------------------------|--|
| Kategorie | Traffic Analysis |
| Ausarbeitung | Angriffskonzept |
| Ersteinschätzung | Ein Angriff dieser Art ist für das I2P Netzwerk fast unmöglich zu verhindern. Der Angriff wurde nicht verworfen, konnte jedoch aus Zeitgründen nicht in der Realisierung behandelt werden. |

[I2P](#) ist wie Tor ein Overlay Netzwerk, welches auf der bestehenden Internet-Infrastruktur basiert. Somit fließt jeglicher Datenverkehr über zentrale Anbieter wie Internet Provider. Diese haben die Möglichkeit, die Teilnehmer ihres Netzwerks zu überwachen. Die [I2P](#) Entwickler beschreiben in ihrem Bedrohungsmodell [21], dass [I2P](#) nicht dafür ausgelegt ist, die Teilnahmen am Netzwerk zu verschleiern.

4.5.1 Angriffsidee

Durch das gezielte Senden von Daten soll ein Muster generiert werden, welches den zurückgelegten Pfad des Datenstroms für den Angreifer ersichtlich macht. Falls der Angreifer dieses Muster in seinem überwachten Netzwerk erkennt, kann der Angreifer den Datenverkehr bis ans eigentliche Ziel verfolgen. Das Muster kann sowohl im Datenverkehr, wie auch in der verschickten Nachricht inkludiert werden. Der Fokus in diesem Angriff ist die Analyse des Datenverkehrs. Der eigentliche Inhalt der Nachricht ist für den Angreifer somit nicht von Bedeutung.

Ansatz 1: Globales Monitoring

Der Angreifer schickt einen Datenstrom mit einer definierten Grösse an sein Ziel. Als Beispiel kann ein Datenstrom mit einer Grösse von 500 MB an einen [I2P](#) Teilnehmer gesendet werden. Der Angreifer, überwacht nun, wo in seinem Netzwerk im gleichen Zeitfenster eine ähnlich grosse Datenmenge zusätzlich empfangen wird. Die Möglichkeit besteht, dass dies gleichzeitig an mehreren Endpunkten passiert. Daher wird dieser Vorgang über einen längeren Zeitraum mehrmals wiederholt. Per Ausschlussverfahren kann das Ziel so identifiziert werden. Da nicht verschleiert werden kann, ob ein Teilnehmer des überwachten Netzes auch Teilnehmer im [I2P](#) Netz ist, kann die Anzahl an überwachten Teilnehmern stark eingeschränkt werden. Yin und He beschrieben in ihrem Paper [44], wie mittels Machine-Learning-Techniken [I2P](#)-Nutzer identifiziert werden können. Ein weiterer Vorteil dieser Art des Monitoring-Angriffs ist, dass sich nur das Ziel, und nicht alle [Router](#) des Tunnels, im Netz des Angreifers befinden muss.

Ansatz 2: Gezieltes Monitoring

Falls dem Angreifer bereits Informationen über das mögliche Ziel bekannt sind, kann der Angreifer den Vorgang, wie im Ansatz 1 beschrieben, gezielter durchführen. Dies wäre für eine Strafbehörde eine Möglichkeit, ihren Verdacht zu validieren. Falls zum Beispiel eine Strafbehörde eine Person verdächtigt, einen bestimmten illegalen Service in [I2P](#) zu betreiben, könnte in Zusammenarbeit mit dem entsprechenden Internet Provider dieser Verdacht bestätigt oder auch widerlegt werden. Dies wird erreicht, indem ein Datenstrom definierter Grösse zum illegalen Service gesendet wird. Zusammen mit dem Internet

Provider oder Hostinganbieter kann überprüft werden, ob der gesendete Datenstrom bei der verdächtigen Person ankommt.

4.5.2 Bekannte Mitigationen

Wie im Bedrohungsmodell [21] der I2P Entwickler nachzulesen, existieren bereits diverse Massnahmen, welche eine Traffic-Analysis erschweren. Zum Beispiel verwendet I2P zufällig gewählte Ports, was ein einfaches Zuweisen des Datenverkehrs zu I2P erschwert. Der vollständig verschlüsselte Datenverkehr ist für diesen Angriff ebenfalls erschwerend, da es schwieriger wird ein Muster zu generieren. Moderne Verschlüsselungen sollte keine Rückschlüsse auf den Klartext geben und somit ein vorhandenes Muster im eigentlichen Datenverkehr verschleiern. Es konnte aber keine Mitigation identifiziert werden, welche das Senden eines Datenstroms an ein Teilnehmer in I2P erschweren oder gar verhindern würde.

4.5.3 Durchführbarkeit

Für das I2P Netzwerk ist es fast unmöglich, Angriff dieser Art zu verhindern. Dies liegt daran, dass nicht direkt das Protokoll, sondern die darunter liegende Infrastruktur angegriffen wird. Limitierend für diese Angriffsmöglichkeit ist, dass das Opfer im Netz des Angreifers sein muss. Falls das Ziel nicht bereits, wie in Ansatz 2 beschrieben, bekannt ist, muss der Angreifer die Kontrolle über grosse Teile des Internets überwachen können oder mit mehrere Parteien zusammenarbeiten, um ein solches Monitoring durchführen zu können.

4.6. Musterkennung: Nachricht/Tagging

| | |
|-------------------------|--|
| Kategorie | Traffic Analysis |
| Ausarbeitung | Angriffskonzept |
| Ersteinschätzung | Die Angriffs-idee wurde während dem Konzeptionieren des Angriffs verworfen. Möglichkeiten eines Angriffs wurden entweder bereits mitigi-ert, oder befanden sich ausserhalb des definierten Scopes. |

I2P legt viel Wert darauf, dass nicht erkennbar ist, in welcher Position ein Router innerhalb eines Tunnels ist. Ebenfalls sollte der Router nicht wissen, zu wem der Tunnel gehört, in welchem er aktuell Teilnehmer ist. Ein Router in einem Tunnel sollte nur seinen nächsten Hop sowie die dazugehörige Tunnel ID kennen.

4.6.1 Angriffs-idee

Ein Angreifer versieht eine Nachricht mit Tags, welche es ermöglichen, die Position eines vom Angreifer kontrollierten Routern zu bestimmen. Für die Tags könnten etwa ungenutzte Optionen in den Delivery Optionen verwendet werden. Falls es nun für einen Router im Tunnel möglich ist, diese Tags zu erkennen, wäre dem Router bekannt, dass er ein nachfolgender Teilnehmer im selben Tunnel ist. Anhand dieser Erkenntnis könnten weitere Informationen über das Ziel erfahren werden.

4.6.2 Durchführbarkeit

Bei den ausgehenden Tunnels wird die Nachricht vom Sender mit der in [Unterabschnitt 2.2.2](#) beschriebenen Layered-Encryption verschlüsselt. Jeder Tunnel Teilnehmer kann somit nur seine Schicht entschlüsseln. Darin findet er nur Router und Tunnelinformationen, welche für das Weiterleiten der Nachricht zwingend nötig sind. Weiter ist in der Tunnel Message Spezifikation [40] beschrieben, dass eine solche Nachricht eine feste Grösse von 1008 Bytes haben muss. Daher lassen sich keine zusätzlichen Informationen anhängen, welche möglicherweise für einen Angriff nutzbar wären. Der [Router](#) schickt die verschlüsselte Nachricht anhand der Delivery Instruction weiter.

Ein eingehender Tunnel erhält die entschlüsselte Tunnel Message, in Form eines verschlüsselten Texts, den nur der Empfänger der Nachricht entschlüsseln kann. Die folgenden Teilnehmer des Inbound Tunnels verschlüsseln die Nachricht wieder mit ihrem Schlüssel. Der Inbound Endpoint, respektive der eigentliche Empfänger, kann beim Empfang die verschiedenen Verschlüsselungsschichten wieder entschlüsseln. Da jeder Teilnehmer die Nachricht vollständig verschlüsselt, ist eine Erkennung von Mustern oder Tags innerhalb des Tunnels fast ausgeschlossen. Ob ein Muster aus der verschlüsselten Nachricht entnommen werden kann, wird innerhalb dieser Arbeit nicht weiter untersucht, da von einer perfekten Implementation ausgegangen wird. Die perfekte Implementation gilt ebenfalls für die gewählten Verschlüsselungsalgorithmen.

4.7. Peer Selection Ordering

| Kategorie | Implementierungsangriffe |
|-------------------------|---|
| Ausarbeitung | Angriffsidee |
| Ersteinschätzung | Die Angriffsidee wurde verworfen, da sie durch den Einsatz des „random keys“ mitigiert. |

Die Implementation des [I2P Router](#) sieht vor, dass alle Peers innerhalb von Tunnel Pools mit einem Directory-Sort-Algorithmus sortiert werden. Diese Information wurde im Expertengespräch mit den [I2P](#) Entwicklern aufgedeckt. Dieser Directory-Sort hat zur Folge, dass die Position der [Router](#) innerhalb eines Tunnels fixiert werden. Dies wurde so umgesetzt, um eine möglichst geringe Rotation der Teilnehmer innerhalb der Tunnels zu gewährleisten und so eine mögliche Tunnelkontrolle, wie in [Abschnitt 4.3](#) beschrieben zu erschweren. Falls die Sortierung jedoch deterministisch ist, könnte ein Angreifer die Position seiner [Router](#) innerhalb der Tunnels erzwingen.

4.7.1 Durchführbarkeit

Bei der Implementation dieser Funktion wurde darauf geachtet, dass die Sortierung nicht deterministisch ist. Wie in der [I2P](#) Dokumentation in Kapitel *Peer Ordering within Tunnels* der Tunnel Implementation [39] beschrieben, wird für jeden Tunnel Pool eine zufälliger Seed generiert. Dieser Seed wird in Kombination mit dem Hash des Peer für die Sortierung verwendet. Dieser Vorgang wird lokal auf dem entsprechenden [Router](#) durchgeführt. Somit ist es für einen Aussenstehenden nicht möglich, die Sortierung der Peers vorauszusagen. Eine Voraussage der Sortierung könnte möglich sein, falls die Position von mehreren Peers bekannt ist und so der Seed erraten werden kann. Es ist jedoch davon auszugehen, dass die dadurch gesammelten Informationen das Ziel bereits anderweitig

deanonymisieren würden. Aufgrund dieser Erkenntnis wird dieser Angriff innerhalb dieser Arbeit nicht weiterverfolgt.

4.8. Testnetzwerk

Für das Entwickeln eines Angriffs könnte ein Testnetzwerk für die Forschenden hilfreich sein. So können Angriffe in einem kontrolliertem Umfeld getestet werden. Zudem könnten Angriffe getestet werden, welche nur mit enormen Aufwand an Ressourcen möglich wären, wie ein Sybil Angriff. Weiter können auch Netzwerkanalysen einfacher durchgeführt werden, da alle Netzteilnehmer unter eigener Kontrolle sind. Auch das Durchführen von Bootstrapping-Angriffen, wie in [Abschnitt 4.4](#) beschrieben, sind mit einem Testnetzwerk einfacher durchzuführen, da auch die Reseed Server kontrolliert werden. Natürlich repräsentiert ein Testnetzwerk nicht die Realität. Es können aber Rückschlüsse und erste Erfahrungen gesammelt werden, welche das Planen, Entwickeln und Validieren eines Angriffs vereinfachen. Im [Abschnitt 5.3](#) wird beschrieben, wie ein solches Netzwerk umgesetzt werden kann.

5. Realisierung

Nach aktuellem Wissensstand lassen sich die beschriebenen Angriffe am einfachsten in einem Testnetzwerk (unter Laborbedingungen) nachstellen. So ist eine Durchführung im realen Netz, im Zuge dieser Bachelorarbeit, nicht realistisch. Es fehlen insbesondere die zeitlichen Ressourcen. Erschwerend kommt hinzu, dass aktuell kein **I2P** Testnetzwerk vorhanden ist, in welchem Angriffe simuliert oder durchgeführt werden könnten. Deswegen wird hier beschrieben, wie die Angriffe in einer künftigen Arbeit durchgeführt werden könnten. Weiter wird mittels mathematischer Modellierung aufgezeigt, wie sich gewisse Faktoren beim Angriff der Tunnelkontrolle verhalten.

5.1. Tunnelkontrolle Ansatz 1: Partielle Tunnelkontrolle

Der Angriff der Tunnelkontrolle lässt sich nach aktuellem Wissensstand nur mittels Sybilangriff durchführen. Eine möglichst gleichmässige Verteilung im **I2P** Netz wird den Angriff zusätzlich begünstigen. Aus diesem Grund sollte darauf geachtet werden, dass möglichst wenige **Router** von den anderen **I2P** Teilnehmern als malicious erkannt und geblockt werden. Der diesbezüglich grösste Störfaktor ist das Sybil Analysetool, welches mit heuristischen Analysen einen Sybilangriff versucht zu erkennen. **Router**, welche mit einem solchen Angriff in Verbindung gebracht werden, blockiert das Tool lokal auf dem eigenen **Router**. Da dieses Tool aktuell nur **Floodfills** analysiert, sollte der Angriff möglichst ohne **Floodfills** durchgeführt werden.

5.1.1. Berechnung benötigter Router

Recherchen haben ergeben, dass das Errechnen der genau benötigten **Router** bei einem Sybil Angriff sehr komplex ist. Auch müssen mehrere unbekannte Faktoren in der Rechnung berücksichtigt werden. Deswegen wurde ein anderer Ansatz gewählt: Mit einer Aufwandsabschätzung sollte abgeschätzt werden können, wie erfolgversprechend ein Angriff sein wird. Dabei soll vorausgesagt werden können, welchen Einfluss die jeweiligen Parameter auf diese Art von Angriff haben. Diesbezüglich wurde folgende Formel aufgestellt.

$$I \approx \left(\frac{R}{N + R} \cdot p \right)^h \cdot \frac{a}{t!} \cdot \frac{T \cdot r \cdot d}{h! \cdot (t - h)!}$$

Diese Formel gibt als Output die Voraussage der erwarteten Anzahl an Tunnels wieder, welche mittels des Angriffs der „Tunnelkontrolle“ infiltriert werden können. Die einzelnen Komponenten und Variablen der Formel setzen sich wie folgt zusammen:

$\left(\frac{R}{N + R} \cdot p \right)^h \cdot \frac{a}{t!}$ repräsentiert die Wahrscheinlichkeit, dass die Bedingungen

für den Angriff „Tunnelkontrolle“ erfüllt werden, falls ein Tunnel komplett neu erstellt wird. $\frac{R}{N}$ repräsentiert dabei das Verhältnis der malicious Router R zu allen Router des Netzwerkes N . In dieser Formel wird einfachheitshalber angenommen, dass die malicious Router gleichmässig im Netz verteilt sind. Demzufolge dient dieses Verhältnis zugleich auch als Ausgangslage für die Wahrscheinlichkeit, dass ein malicious Router gewählt wird, falls bei einem Tunnelaufbau ein neuer Router gesucht wird. Da bei der Routerauswahl während dem Tunnelaufbau auch auf die Performance der einzelnen Router geachtet wird und performante Router bevorzugt behandelt werden, kann dies mittels der Performance p abgebildet werden. Weiter wird in der Formel berücksichtigt, dass nicht alle Positionen in einem Tunnel mit malicious Routern belegt werden müssen. So kann mit t die Tunnellänge und mit h die Anzahl der minimal benötigten Hops innerhalb eines Tunnels angegeben werden. Mit h wird somit berechnet, wie wahrscheinlich es ist, dass bei einem Tunnelaufbau die gewünschte Anzahl Router in einem Tunnel vorkommt.

Mit $\frac{a}{t!}$ wird ebenfalls die Permutation berücksichtigt. Dieser Teilabschnitt der Formel berechnet die Wahrscheinlichkeit (in Prozent), ob sich die Router in der richtigen Reihenfolge befinden. Dies wird erreicht, indem die Anzahl gültiger Permutationen a

durch die Anzahl aller Permutationen $\frac{t!}{h! \cdot (t-h)!}$ geteilt wird. Zur Berechnung aller Permutationen wird eine Permutation mit Wiederholung verwendet. Gemäss Studyflix [33] muss eine solche Permutation verwendet werden, wenn die Position der jeweiligen Elemente einer Gruppe nicht unterscheidbar ist. So ist bei diesem Angriff die exakte Position der einzelnen Router egal, solange das verlangte Muster eingehalten wird.

Mit $T \cdot r \cdot d$ können die weiteren Eigenschaften des I2P Netzwerkes wie die durchschnittliche Anzahl Tunnels pro Client T , die Anzahl Router welche all Ihre Tunnels erneuern werden r und wie viele Router vom Angriff betroffen sind d .

Die Daten für die Berechnung können wie folgt ermittelt werden:

- R = Anzahl bereitgestellter, malicious Router
- N = Gesamtgrösse des I2P Netzwerkes, vor dem Deployment der malicious Routern. Für die folgenden Berechnungen wurde dieser Wert von I2P Metric [19] entnommen.
- p = Gewichtung für die Performance der malicious Routern. 1 entspricht der durchschnittlichen Performance. < 1 entspricht einer unterdurchschnittlichen Performance und > 1 entspricht einer überdurchschnittlichen. Aktuell ist jedoch nicht bekannt, welchen Wertebereich diese Gewichtung besitzt. Dieser müsste bei der Durchführung des Angriffs erst ermittelt werden. Es wird deshalb empfohlen, den Wert von p auf 1 zu setzen.
- h = Anzahl der minimal benötigten Hops, welche für den Angriff benötigt werden. Wie in Abschnitt 4.3 Tunnelkontrolle beschrieben, sind beispielsweise bei einem Tunnel mit Länge 3 lediglich 2 Hops zu besetzen. h lässt sich ebenfalls mathematisch beschreiben als $\lfloor t - \frac{t}{2} + 1 \rfloor$. Gemäss dieser Formel werden bei einer maximalen Tunnellänge von 8 Hops, mindestens 5 „Evil Router“ benötigt. In Abbildung 5.1 wird dieses Szenario dargestellt.

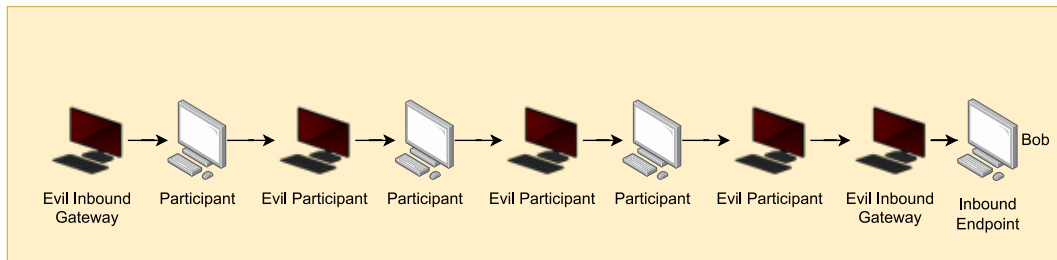


Abbildung 5.1.: Infiltrierter Tunnel mit maximaler Länge

- a = Anzahl gültige Permutationen. Wie aus [Abbildung 5.1](#) abzuleiten ist, gibt es mehrere Situationen, in welchen mehr als eine gültige Permutation existiert. Bei einer Tunnellänge von 8 Hops gibt es beispielsweise 4 gültige Permutationen.
- t = die erwartete Länge der Tunnels. Dieser Wert ist standardmässig 3. Es gilt zu beachten, dass Änderungen dieses Wertes auch h beeinflussen.
- T = Durchschnittliche Anzahl an Tunnels, welche pro [Router](#) erstellt werden. Dieser Wert muss abgeschätzt werden. Es gilt jedoch zu beachten, dass in der Standardkonfiguration von [I2PD](#) 5 Inbound- und 5 Outbound Tunnels definiert sind. Demzufolge kann 10 als Ausgangswert angenommen werden.
- r = Anzahl neuer [Router](#) pro Woche, welche Tunnels neu aufbauen müssen. Dieser Wert ist nicht bekannt und muss abgeschätzt werden. Als möglicher Anhaltspunkt könnte die Fluktuation dienen, welche von der Grafik „Number of routers and IP addresses“ von der Webseite I2P Metrics [19] abgelesen werden kann.
- d = Prozentsatz der Router, welche so konfiguriert sind, dass Sie anfällig für den Angriff sind (z. B. Standardkonfiguration). Dieser Wert ist nicht bekannt und muss abgeschätzt werden.
- I = Erwartet Anzahl an infiltrierten Tunnels pro Woche. Es gilt zu beachten, dass aufgrund der vielen Annahmen und unbekannten Faktoren dieser Wert nur einer groben Schätzung entspricht. Dieser Wert sollte als Vergleichswert interpretiert werden, mit welchem es möglich ist zu bestimmen, welche Faktoren einen Einfluss auf den besagten Angriff haben und wie sich diese auf den Angriff auswirken.

Beispiele

Bei den errechneten Beispielen wurden die Daten vom 10.05.2022 berücksichtigt. Anhand von I2P Metrics [19] und der Standardkonfiguration konnten zu diesem Datum folgende Parameter ermittelt werden:

- $N = 27'000$
- $a = 1$
- $T = 10$
- $r = 110$
- $d = 0.95$

Beim Parameter d handelt sich um einen Schätzwert, dass 95 % von allen aktiven [Router](#) die Standardkonfiguration von 3 Hops verwenden.

Somit gilt folgende Grundformel:

$$\left(\frac{R}{27000 + R} \cdot p\right)^h \cdot \frac{1}{t!} \cdot 10 \cdot 110 \cdot 0.95 \approx I$$

$$\frac{h! \cdot (t - h)!}{h! \cdot (t - h)!}$$

$$\left(\frac{5400}{27000 + 5400} \cdot 1\right)^2 \cdot \frac{1}{\frac{3!}{2! \cdot 1!}} \cdot 10 \cdot 110 \cdot 0.95 \approx 10$$

Somit wird vorhergesagt, dass mit einem Anteil von 20 % Netzanteil zum aktuellen Zeitpunkt ca. 10 Tunnels pro Woche infiltriert werden könnten.

5.1.2. Erkenntnisse

Anhand der aufgestellten Formel lassen sich unterschiedliche Hypothesen aufstellen. Bei den unten stehenden Funktionsplots werden unterschiedliche Sybil Attacken simuliert, bei welchen jeweils 10 %, 20 % und 200 % evil Router R in das bestehende Netz eingeschleust werden. 100 % entspricht dabei jeweils der Anzahl Router im I2P Netz, vor dem Einschleusen der evil Router. Falls das Netz beispielsweise aus 27'000 Routern besteht, entsprächen 200 % evil Router R 54'000 Routern. Das I2P Netz bestünde dadurch aus insgesamt 81'000 Routern und der Angreifer hätte somit eine Zweidrittelmehrheit.

Um das aufstellen des Funktionsplots zu vereinfachen, wurde der Parameter a auf 1 gesetzt.

1. Hypothese: Längere Tunnels erhöhen die Anonymität

Die Tunnellänge t ist der wichtigste Faktor in der Formel, da h auch als $\lfloor t - \frac{t}{2} + 1 \rfloor$ dargestellt werden kann. Demzufolge sind alle Fakultäten und Potenzen abhängig von t . Am folgenden Beispiel wird aufgezeigt, wie sich die Tunnellänge auf die Infiltrationsrate voraussichtlich auswirken wird. Auch wird aufgezeigt, wie sich die Tunnellänge auf unterschiedliche Grössen des Angriffs verhält. Die Funktionen wurden mit folgender Formel und Parametern geplottet.

$$f(t) = \left(\frac{R}{27000 + R} \cdot 1\right)^{\lfloor t - \frac{t}{2} + 1 \rfloor} \cdot \frac{1}{t!} \cdot 10 \cdot 110 \cdot 0.95 \approx I$$

$$\frac{1}{\lfloor t - \frac{t}{2} + 1 \rfloor! * (t - \lfloor t - \frac{t}{2} + 1 \rfloor)!}$$

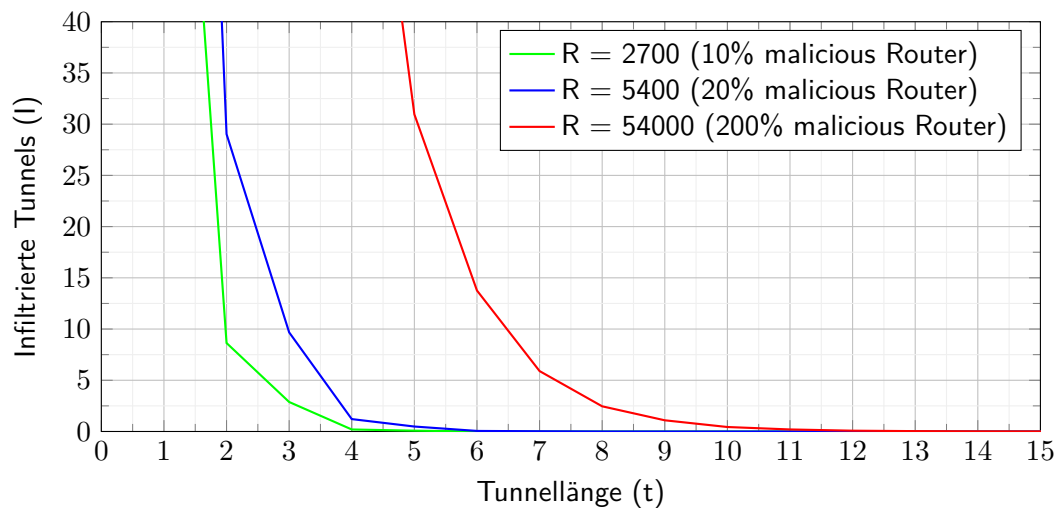


Abbildung 5.2.: Die Erhöhung der Tunnellänge hat einen signifikanten Einfluss

Wie aus dem Funktionsplot zu entnehmen ist, wächst die Schwierigkeit des Angriffs mit $\mathcal{O}(t!)$.

2. Hypothese: Router Performance ist wichtiger als Router Anzahl

Diese Hypothese basiert auf der Annahme, dass p einen linearen Einfluss auf die Wahrscheinlichkeit hat, ob ein Router des Angreifers als Teilnehmer eines Tunnels gewählt wird. Falls p verdoppelt wird, sollte die Wahrscheinlichkeit der Selektion des Routers zugunsten des Angreifers verdoppelt werden. Da aktuell nicht bekannt ist, wie p in der Realität skaliert und wie p gemessen werden sollte, muss diese Hypothese zwingend überprüft werden. Die Erkenntnis dieser Überprüfung sollten in einen Angriff einbezogen werden.

Der unten aufgeführte Funktion wurde mit folgender Formel und folgenden Annahmen geplottet.

$$f(t) = \left(\frac{R}{27000 + R} \right)^{\lfloor t - \frac{t}{2} + 1 \rfloor} \cdot \frac{1}{t!} \cdot \frac{10 \cdot 110 \cdot 0.95}{\lfloor t - \frac{t}{2} + 1 \rfloor! * (t - \lfloor t - \frac{t}{2} + 1 \rfloor)!} \approx I$$

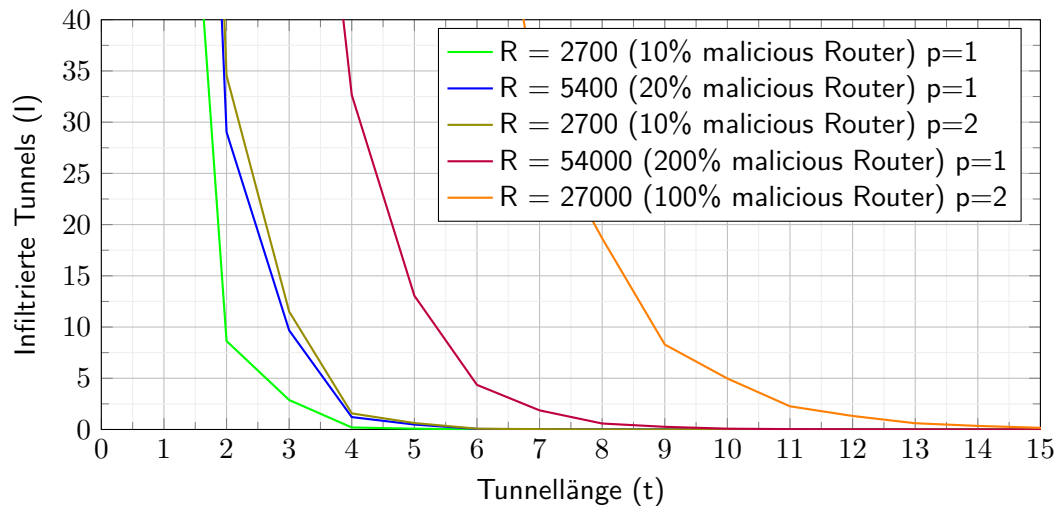


Abbildung 5.3.: Doppelte Routerperformance verglichen mit doppelter Anzahl Router

Abbildung 5.3 zeigt auf, dass p einen starken Einfluss auf einen potenziellen Angriff hat. Durch die Erhöhung von p kann die Anzahl der Router reduziert werden. Da p nach aktuellen Erkenntnissen primär durch die Bandbreite, Latenz und Erfolgsrate bei der Tunnelerstellung beeinflusst wird, sollte der Fokus auf der Verbesserung dieser Werte liegen. Durch die kleinere Anzahl an Routern, sollte es einfacher werden, einen Sybil Angriff durchzuführen.

3. Hypothese: Einfluss der Anzahl Tunnels ist linear

Gemäss der Formel ist der Einfluss der Tunnelanzahl T linear und lässt sich als $\mathcal{O}(T)$ darstellen. Die Sicherheit gegen diese Art von Angriff wird dabei erhöht, wenn die Tunnelanzahl verringert wird. Dies wird im folgenden Beispiel grafisch dargestellt. Der Funktionsplot basiert auf folgender Formel und folgenden Annahmen:

$$f(t) = \left(\frac{R}{27000 + R} \cdot 1 \right)^2 \cdot \frac{1}{\frac{3!}{2! + 1!}} \cdot T \cdot 110 \cdot 0.95 \approx I$$

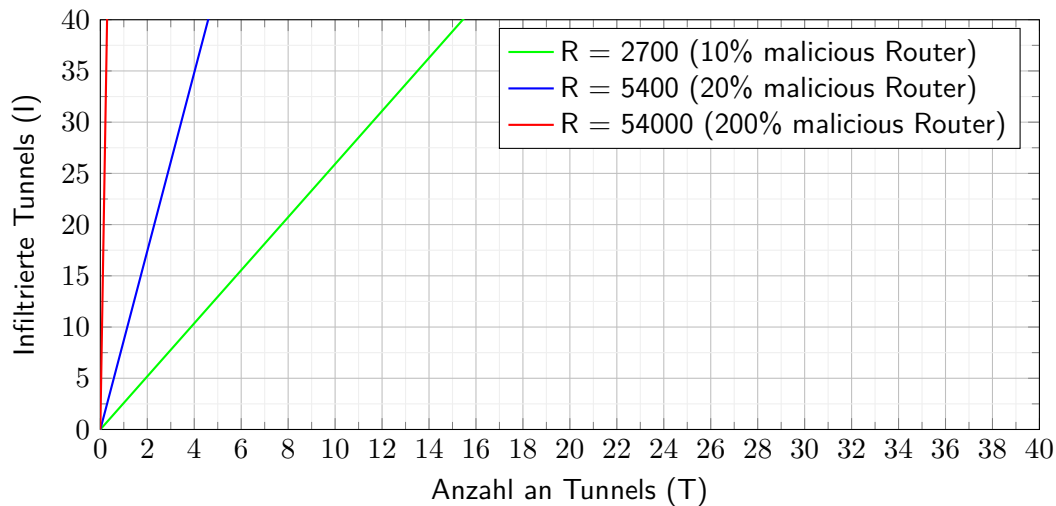


Abbildung 5.4.: Einfluss der Tunnellänge ist linear

Anhand dieses Funktionsplots kann aufgezeigt werden, dass die Anzahl Tunnel pro Router einen markanten Einfluss auf den Angriff hat. Eine Verdoppelung der Tunnelanzahl T entspricht ungefähr einer Verdoppelung der infiltrierten Tunnel I . Aus dieser Hypothese geht hervor, dass nur so viele Inbound und Outbound Tunnel erstellt werden sollten, wie für den störungsfreien Betrieb notwendig sind.

5.1.3. Modifikationen

Um den beschriebenen Angriff erfolgreich durchführen zu können, müssen mehrere Modifikationen an I2PD und am Testnetz vorgenommen werden. Weiter müssen Informationen aus dem DIVA.EXCHANGE Netzwerk ausgelesen und ausgewertet werden. Zudem wird ein Analyse- und Control-Server benötigt.

Modifikationen an I2PD

Für die Rekonstruktion der Tunnels müssen die manipulierten I2PD Router an einen Analyseserver Tunnelinformationen weiterleiten. Er muss insbesondere bekannt geben, von welchen IP-Adressen (RouterInfos), nach welchen IP-Adressen (RouterInfos) er seine Tunnels aufbaut.

Für die Validation, ob die zwei Router tatsächlich im selben Tunnel sind, oder über denselben Router, in unterschiedlichen Tunnels kommunizieren, müssen Logs über die versendeten Tunnelpakete gesammelt werden. Diese Logs sollten aufgezeichnet und dem Analyseserver zugestellt werden. Die I2PD Software muss dementsprechend so abgeändert werden, dass die Router Anfragen bezüglich Log-Aufzeichnung vom Analyseserver entgegennehmen und verarbeiten können. In den Logs sollte festgehalten werden, zu welchen Zeiten Datenpakete durch die angeforderten Tunnels empfangen und versendet wurden.

Angriffsspezifische Vorbereitungen

Das Testnetz muss so eingerichtet werden, dass es aus dem gewünschten Verhältnis an „normalen“ und „malicious“ Routern besteht. Es muss ein Weg gefunden werden, wie die

„malicious“ Routern möglichst gleichmässig verteilt werden über das **I2P** Netzwerk. Besonders Fokus sollte dabei auf das Sybil-Analyse-Tool des **i2p.i2p Routers** gelegt werden. Falls die eigenen **Router** dabei erkannt werden, erschwert dies einen Angriff erheblich. Das Analyse-Tool sollte umgegangen werden können, solange nicht mit **Floodfills** gearbeitet wird. Weiter muss ein Analyse- und Control-Server erstellt werden.

Informationen aus dem DIVA Netzwerk auslesen

Die B32-Adressen der DIVA Netzwerkteilnehmern sind alle in der DIVA Blockchain gespeichert und öffentlich einsehbar. Deshalb ist es nicht nötig, Modifikationen direkt bei der DIVA Software vorzunehmen, um an **LeaseSets** der Teilnehmenden zu gelangen. Es muss jedoch eine zusätzliche Anwendung erstellt werden, welche die B32-Adressen aus der Blockchain ausliest und anschliessend die **LeaseSets** der gefundenen Teilnehmer bei den entsprechenden **Floodfills** anfordert.

5.1.4. Variable Tunnellänge

Der Angriff setzt voraus, dass die Tunnellänge bekannt ist. Diesbezüglich wäre eine variable Tunnellänge die beste Mitigation. Falls die DIVA-Clients eine variable Tunnellänge von 3 bis 4 verwenden, kann bei einer erfolgreichen Infiltration der ersten drei Hops nur mit einer Wahrscheinlichkeit von 50 % vorausgesagt werden, ob es sich beim nächsten Hop effektiv um das Ziel handelt. Falls jedoch eine Infiltration von vier Hops nachgewiesen werden kann, ist klar, dass es sich beim nächsten Hop um das Ziel handelt. Für einen Angreifer bedeutet dies, dass nur Tunnels mit einer maximalen Anzahl Hops deanonymisiert werden können. Dadurch wird einerseits der Angriff erschwert. Andererseits verringert dies die Anzahl der Tunnels, welche anfällig für den Angriff sind. Alle Tunnels kürzer als die maximalen Anzahl Hops, können nicht mehr zuverlässig deanonymisiert werden.

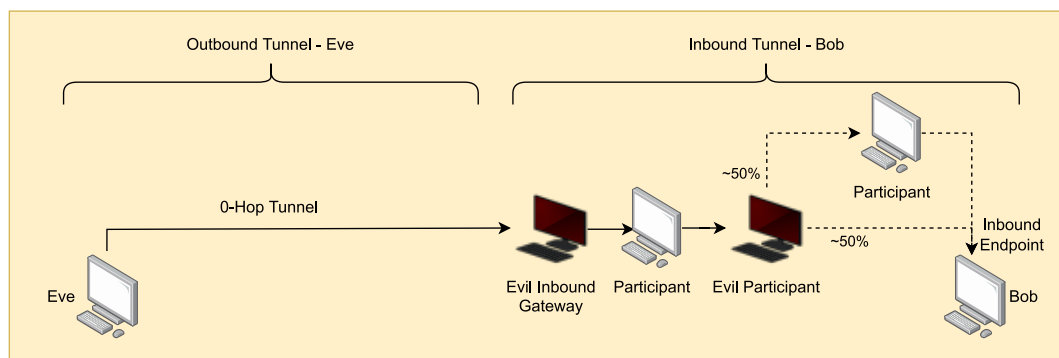


Abbildung 5.5.: Partielle Tunnelkontrolle mit randomisierter Tunnellänge

5.2. Tunnelkontrolle Ansatz 2: DoS auf Participants

Eine weitere Möglichkeit, wie dieser Angriff erfolgreicher durchgeführt werden könnte, wurde während der Zwischenpräsentation von Herrn Jan Alsenz (Experte dieser Arbeit) eingebracht. Diese Variante sieht vor, dass nicht darauf gewartet wird, bis per Zufall eine gewünschte Konstellation erreicht wird, sondern die Konstellation sollte mittels **Denial**

of Service (DoS) erzwungen werden. Dies wird erreicht, indem mittels eines DoS Tunnel-Participants dazu gezwungen werden, keine neuen Tunnels aufzubauen.

Aufwand der Durchführung

Bei den Recherchen konnten keine Anhaltspunkte gefunden werden, wie viel Aufwand betrieben werden muss, damit ein Tunnelteilnehmer mit DoS von der Tunnelerneuerung ausgeschlossen werden kann. Diesen Aufwand gilt es bei einer praktischen Durchführung herauszufinden. Bis dahin gilt die Annahme, dass dies erst der Fall sein wird, wenn das „Transit tunnels limit“, Standardwert 2'500 Tunnels, erreicht wird. Weiter wird ebenfalls ein Sybilangriff, analog zu [Abschnitt 5.1 Tunnelkontrolle Ansatz 1: Partielle Tunnelkontrolle](#) benötigt, um diesen Angriff durchzuführen. Deswegen ist Ansatz 2 als Zusatz zu Ansatz 1 zu verstehen, mit welchem dessen Erfolgchancen erhöht werden sollte. Falls diese Annahmen korrekt sind, könnte ein Angriff wie folgt funktionieren.

1. Schritt: Zielidentifizierung

Aus der Blockchain des DIVA.EXCHANGE Netzwerkes können Teilnehmer und deren B32-Adressen entnommen werden. Anhand dieser Adressen können bei den Floodfills einerseits LeaseSets angefordert werden. Andererseits kann so überprüft werden, ob ein Ziel erreichbar ist. Anschliessend wird von den erreichbaren Teilnehmern einer ausgewählt, auf welchen der Angriff durchgeführt werden sollte. Beim ausgewählten Teilnehmer wird nun ein Lease aus dem LeaseSet entnommen. Aus diesem Lease können die [Inbound Gateways](#) der Tunnels entnommen werden. Diese [Inbound Gateways](#) sind die ersten Ziele des Angriffs.

2. Schritt: Angriff

Es wird nun überprüft, ob die [Inbound Gateways](#) bereits eigene Router sind. Falls nicht, werden die identifizierten [Inbound Gateways](#) mit „Tunnel Build Messages“ geflutet. Dadurch wird das „Transit tunnels limit“ der [Inbound Gateways](#) erreicht und diese stehen nicht mehr zur Verfügung, was dazu führt, dass diese ausgetauscht werden müssen. Nach ca. 10 Minuten kann nun erneut das LeaseSet bei einem Floodfill angefordert werden. Von diesem LeaseSet können nun die neuen [Inbound Gateways](#) entnommen werden. Dies wird nun so oft wiederholt, bis ein eigener Router der [Inbound Gateway](#) des Ziels ist. Dieser Schritt wird nun mit jedem Teilnehmer wiederholt, bis alle Teilnehmer des Tunnels durch eigene [Router](#) ersetzt wurden.

Variable Tunnellänge

Der grösste Vorteil dieses Angriffs ist, dass anders als bei [Tunnelkontrolle Ansatz 1: Partielle Tunnelkontrolle](#), überprüft werden kann, ob der komplette Tunnel übernommen wurde. Dies ist möglich, da der Inbound Endpoint nicht anfällig auf den DoS-Angriff ist. So ist dieser kein Transit Tunnel Participant und steht auch als Inbound Endpoint zur Verfügung, wenn das *Transit Tunnels Limit* erreicht ist. Sollte der letzte eigene, malicious Router auch nach mehreren Durchläufen des DoS-Angriffs immer noch mit demselben Participant kommunizieren, kann davon ausgegangen werden, dass der nächste Participant in Wirklichkeit der Inbound Endpoint ist. Somit können auch I2P-Teilnehmern mit variablen Tunnels deanonymisiert werden.

Kombination aus Ansatz 1 & 2

Für eine möglichst effiziente Durchführung des Angriffs der Tunnelkontrolle wird eine Kombination der beiden Ansätze empfohlen. So können die Router, welche im 1. Ansatz ins Netzwerk integriert wurden, verwendet werden, um die DoS-Angriffe durchzuführen. Weiter kann mittels DoS-Angriff bei einer erfolgreichen Infiltration eines Tunnels mit drei Hops überprüft werden, ob es sich hier um das Ziel, oder um einen Participant handelt.

5.3. Testnetzwerk

Wie in [Unterabschnitt 3.9.5](#) beschrieben, wurde für das Testnetzwerk die erarbeiteten Lösungen von Moritz Küttel [26] und LNS [27] geprüft.

5.3.1. Entscheidung

Für die Entscheidung wurde keine Evaluation von verschiedenen Tools durchgeführt. Der Fokus dabei war, eine funktionstüchtige Lösung zu haben. Fehlende Funktionen können zu einem späteren Zeitpunkt ergänzt werden. Das Script von LNS sah dafür vielversprechend aus. Die ausgewählte Lösung hatte bereits viele Funktionen, wie die Einstellung von Router spezifischen Konfigurationen und die Verwendung von unterschiedlichen Router Implementationen pro Router.

5.3.2. Probleme

Das existierende Script war vollständig funktionsfähig. Als Problem während des Testens zeigte sich, dass die Entwickler von [I2PD](#) eine Prüfung von „Reserved Networkers“ implementiert haben. Dieser wurde mit dem Commit [2e0019c8c8c7f524710c9abf581e8794eaac267a](#) [6] eingeführt. [RouterInformationen](#) welche eine solche IP-Adresse beinhalten, werden von [Routern](#) als ungültig markiert und können somit nicht verwendet werden. Da diese [RouterInformationen](#) für das Reseeding verwendet werden, muss dieser Check umgegangen werden.

Mögliche Lösungen:

- Deaktivierung des Checks im Code
- Deaktivierung des Checks in der Konfiguration
- Anpassung der Konfiguration mit Public IP Netzwerken

Für initiale Tests wurde im [I2PD](#) Source Code der Check deaktiviert. Der Source Code wurde folgendermassen angepasst.

```

1  if (context.GetRouterInfo ().IsNTCP2 (true)
2      && peer.router->IsReachableBy (RouterInfo::eNTCP2V4))
3  {
4      address = peer.router->GetPublishedNTCP2V4Address ();
5      if (address && m_CheckReserved
6          && i2p::util::net::IsInReservedRange(address->host))
7          // address = nullptr;
8          LogPrint (eLogDebug, "IP Adresse would be in a reservedRange");
9  }
```

Die Zeile 7 im obigen Beispiel wurde dafür auskommentiert. Der Check wird normal durchgeführt, lässt aber die Adresse des Zielrouters bestehen und setzt die Variable nicht auf einen *nullptr*. Das Setzen auf einen *nullptr* führt dazu, dass der Router die IP-Adresse als ungültig interpretiert, zu welcher er nicht verbinden kann.

Alternativ zu dieser Anpassung könnten ebenfalls die verwendeten IP-Ranges zu öffentlichen angepasst werden. Die reservierten Ranges können von der Wikipedia Seite *Reserved IP addresses* [35] entnommen werden. Innerhalb des Source Codes des *I2PD Routers* wurde das undokumentierte Konfigurationsflag *reservedrange* gefunden. Dieses wird standardmässig mit dem Wert „true“ geladen. Was zur Folge hat, dass der entsprechende Check durchgeführt wird. Die Konfiguration auf „false“ zusetzen beim Starten des Routers schlug fehl. Aufgrund der fortgeschrittenen Zeit und der alternativen Lösung wurde dieser Sachverhalt nicht mehr mit dem Entwickler des *I2PD Routers* angeschaut.

5.3.3. Anpassung der Konfiguration

An der Konfiguration wurden Anpassungen vorgenommen, um das Erstellen von Routern zu vereinfachen. In der bisherigen Version musste jeder Router einzeln erfasst werden. In der aktualisierten Version können Router-Typen definiert werden. Dazu wird ein Netzwerk, inklusive Subnetzmaske und einem Offset, definiert. Dieser Offset definiert ab wann das Script beginnt IP-Adresse zu verteilen. Der minimale Offset ist 1, damit keine Kollision mit dem Netzwerkgateway zuträgt.

```

1  "networks": {
2    "fr": {
3      "gateway": "10.4.0.1",
4      "subnet_mask": "/16",
5      "node_offset": 10
6    },

```

Ein Router-Typ definiert die Router und ihre Parameter. Im folgenden Beispiel werden 10 *I2PD Router* erstellt. Sie sind dem Netzwerk „fr“ zugewiesen und keine Floodfills. Sie werden jedoch als Reseed Router für den Reseed Server oder für ein dateibasiertes Reseeding verwendet.

```

1  "router_types": [
2    {
3      "router": "i2pd",
4      "network": "fr",
5      "floodfill": false,
6      "quantity": 10,
7      "reseed": true
8    },
9  ]

```

Weitere Konfigurationsbeispiele sind auf der Plattform Codeberg im Repository *test-net.py* [28] im Verzeichnis *config* zu finden.

5.3.4. Funktionen

Das angepasste Script beinhaltet folgenden Funktionen.

- Automatisches Starten von [Routern](#) mit benutzerdefinierten Parametern.
- Automatisches Starten eines Reseed Servers mit benutzerdefinierten Parametern.
- Erstellen einer [NetDB](#) für den Reseed Server oder die [Router](#).
- Verwendung von unterschiedlichen [Router](#) Versionen.

6. Evaluation und Validation

6.1. Deanonymisierung von Teilnehmern des I2P Netzwerks

Die Community um [I2P](#) hat bereits sehr viel Arbeit investiert und wertvolle Gedankengänge umgesetzt, um eine sichere und anonyme Kommunikation zu gewährleisten. Viele der offensichtlichen Schwachstellen werden innerhalb des I2P-Protokollstacks behandelt und so gut wie möglich mitigiert. So stellte sich die praktische Durchführung einer Deanonymisierung eines Teilnehmers innerhalb des DIVA.EXCHANGE I2P Netzwerks als ein komplexes Unterfangen heraus, das während der Arbeit nicht umgesetzt werden konnte. Trotzdem konnten viele problematische Vektoren und deren Mitigationsmassnahmen aufgezeigt werden. Diese Informationen sind aus unsere Sicht sehr wertvoll. Sie zeigen auf, dass [I2P](#) keine einfachen und grundlegenden Probleme beinhaltet, was für unseren Auftraggeber DIVA.EXCHANGE sicher eine wünschenswerte Nachricht ist. Ebenfalls können die gesammelten Informationen als Grundlage für weiterführende Arbeiten verwendet werden. Die Arbeit bietet eine grundlegende Basis für den Einstieg in die Materie und eine gute Grundlage für weitere Forschungsarbeiten.

6.2. Informationsbeschaffung

Eine der grössten Herausforderungen bezüglich [I2P](#) war, dass das Protokoll keinen grosse Bekanntheit genießt. Für Recherchen konnte, wie in [Unterabschnitt 3.3.1](#) beschrieben, mehrere wissenschaftliche Arbeiten beigezogen werden. Aufgrund der stetigen Weiterentwicklung von [I2P](#) war ein Grossteil dieser Arbeiten leider nicht mehr aktuell. Sie helfen jedoch trotzdem beim Erarbeiten von möglichen Angriffen und dem allgemeinen Verständnis. Aktuelle Informationen konnten von der Webseite [geti2p.net](#) und aus dem Source Code der Router Implementationen entnommen werden. Dadurch war es auch möglich zu erkennen, welche Informationen und Erkenntnissen bei den wissenschaftlichen Arbeiten noch gültig waren. Zusätzlich existiert das Forum [zzz.i2p](#) welches für technische Fragen bezüglich [I2P](#) genutzt wurde. (Das Forum ist nur über das [I2P](#) Netzwerk zu erreichen.)

Ein Versuch über einen IRC Channel Informationen zu erhalten, ist leider fehlgeschlagen. Auf gestellte Fragen erfolgte keine Reaktion. Hingegen wurde auf Fragen im „zzz Forum“ [45] wie auch Anfragen per E-Mail an „zzz“ innert kürzester Zeit geantwortet. Für zukünftige Arbeiten in diesem Bereich ist es absolut empfehlenswert, möglichst früh Kontakte innerhalb der [I2P](#) Community zu knüpfen.

6.3. Methodik

Wie bereits im [Abschnitt 3.3](#) beschrieben, hat sich der Ansatz mit Aktionsforschung und einem Kanban Board für diese Arbeit nicht geeignet. Dies lag unter anderem daran, dass zu wenige Forschungsarbeiten vorlagen, auf welche aufgebaut werden konnte. Der Wech-

sel zur explorativen Forschung und das Führen von Steckbriefen für jeden Angriff ergab eine erhöhte Flexibilität und Tiefe. Dieses Vorgehen war sehr hilfreich, um die Komplexität abzubilden und die erarbeiteten Ergebnisse festzuhalten. Jeder dieser Steckbriefe enthielt alle Informationen und Quellen, welche notwendig waren, um den beschriebenen Angriff nachzuvollziehen. Durch das Einbeziehen von Christoph Egger und den I2P Entwicklern konnten viele Fragen geklärt und auf ihre Korrektheit geprüft werden. Die Experimente am Forschungsobjekt, vorwiegend an den beiden [Router](#) Implementationen, halfen mit die Abläufe besser zu verstehen.

6.4. Verbesserungspotenzial beim Vorgehen

Der Einstieg in die Materie war sehr anspruchsvoll, da [I2P](#) zu Beginn der Arbeit mehrheitlich unbekannt war. Während der Durchführung stellte sich der Informationsaustausch im Zweierteam als weitere Schwierigkeit heraus. So wurden Gegebenheiten im [I2P](#) teilweise unterschiedlich verstanden und interpretiert. Vor allem die hohe Komplexität, kombiniert mit der Schwierigkeit die Erkenntnisse in einfacher Form zu formulieren, stellte sich als grosse Herausforderung dar. Eine zielgerichtete Aufteilung der Tätigkeiten und ein systematischerer Austausch der gewonnenen Erkenntnisse wäre hilfreich gewesen. Wie im [Abschnitt 6.2](#) beschrieben, war das Zusammenstellen von Informationen über das [I2P](#) Protokoll eine weitere Schwierigkeit. Ein systematisches Vorgehen und das Erstellen einer Wissensdatenbank hätte gewisse Probleme entschärft.

6.5. Falsifikation

Antithese zur Tunnelkontrolle: längere Tunnels verbessern die Anonymität nicht

Basierend auf der Formel in [Unterabschnitt 5.1.1](#) kann festgehalten werden, dass das Erhöhen der Tunnellänge den Angriff auf die Tunnelkontrolle in einem faktoriellen Ausmass erschwert. Dies führt dazu, dass die Anzahl infiltrierter Tunnels I mit erhöhter Tunnellänge t logarithmisch abnimmt. Zusätzlich hat I die Charakteristik $\lim_{I \rightarrow 0}$ und kann daher nie 0 erreichen. Dies führt dazu, dass ab einer gewissen Tunnellänge der Schutz der Anonymität nicht weiter steigt. Es gilt anzumerken, dass wenn ein Angreifer die Anzahl malicious Router R erhöht, die „maximale“ Anonymität erst mit einem längeren Tunnel erreicht wird. Dies ist in [Abbildung 5.2](#) ersichtlich.

Die verwendete Standardlänge von 3 Hops für Tunnels in [I2P](#) beruht auf dem im Jahr 2010 veröffentlichten Paper [3]. Darin wurde die optimale Pfadlänge im Tor-Netzwerk analysiert. Da im DIVA.EXCHANGE Netzwerk jedoch die B32-Adressen den einzelnen Teilnehmern in einer öffentlich einsehbarer Blockchain gespeichert sind und die B32-Adressen gemäss DIVA.EXCHANGE möglichst selten geändert werden, sollte die Tunnellänge so hoch wie möglich angesetzt werden. So sind über die Identität der Teilnehmer des DIVA.EXCHANGE-I2P-Netzwerks mehr Informationen bekannt als bei Teilnehmern des Tor-Netzwerks.

Die Antithese falsifiziert somit die ursprüngliche Hypothese in ihrer Kernaussage nicht. Die Antithese liefert jedoch die Erkenntnis, dass längere Tunnels die Anonymität nur bis zu einer gewissen Länge verbessern. Ab einer Länge von 7 - 8 Hops, haben weitere Hops keinen signifikanten Einfluss mehr auf die Anonymität. Eine Verlängerung der Tunnellänge hat einen direkten Einfluss auf die Performance der Tunnels. Durch praktische Tests, könnte geprüft werden, welche Tunnellänge ein optimales Verhältnis von Anonymität zu Performance innerhalb des DIVA.EXCHANGE-I2P-Netzes repräsentiert.

Antithese zur Tunnelkontrolle: eine grosse Anzahl Tunnels verbessert die Anonymität

Mit dem Funktionsplot, abgebildet in [Abbildung 5.4](#), kann diese Antithese deutlich widerlegt werden. Jeder zusätzliche Tunnel erhöht die Wahrscheinlichkeit, dass der Angreifer zufällig in einem Tunnel die gewünschte Konstellation an malicious Routern erreicht.

Antithese zur Tunnelkontrolle: die Router Anzahl ist wichtiger als die Router Performance

In [Abbildung 5.3](#) konnte zwar aufgezeigt werden, dass eine Verdopplung der Performance p ein grösseres I zur Folge hatte, als eine Verdoppelung der malicious Routern R . Da jedoch nicht bekannt ist, wie p skaliert, kann diese Antithese nicht ausgeschlossen werden. Was jedoch die ursprüngliche Hypothese bekräftigt, ist die Tatsache, dass, gemäss dem Expertengespräch mit den I2P Entwicklern, performante Router bei der Tunnelauswahl bevorzugt behandelt werden. Bei einer Folgearbeit sollte somit die Hypothese und Antithese weiterverfolgt werden, bis eine der beiden Thesen definitiv ausgeschlossen werden kann.

6.6. Testnetzwerk

Während der Planungsphase zeigte sich, dass die Umsetzung von Angriffen im I2P-Netzwerk komplizierter sind als ursprünglich angenommen. Einfacher könnten Angriffe in einem Testnetzwerk, unter Laborbedingungen durchgeführt und analysiert werden. Das in [Abschnitt 5.3](#) beschriebene Script zur Erstellung eines Testnetzwerkes kann verwendet werden, um Angriffe in einem kontrollierten Umfeld zu testen. Da dies erst gegen Ende dieser Arbeit zu Verfügung stand, konnte keiner der beschriebenen Angriffe innerhalb eines Testnetzwerks validiert werden.

7. Ausblick

7.1. Fokus: Router Implementationen

Zum Zeitpunkt dieser Arbeit existieren zwei bekannte Implementationen des [I2P Routers](#). Innerhalb dieser Arbeit wurde davon ausgegangen, dass diese keine Fehler in der Implementation aufweisen. Für weitere Forschungsprojekte würde sich anbieten, diese genauer zu analysieren. Fehler innerhalb der Kryptografie oder der sonstigen Funktionen könnten ebenfalls eine Deanonymisierung der Benutzer zulassen.

7.2. Fokus: DIVA.EXCHANGE

Der Hauptfokus dieser Arbeit lag auf [I2P](#) selbst. Für weitere Arbeiten könnte der Fokus auf den DIVA.EXCHANGE selbst gelegt werden, wie dieser das [I2P](#) Netzwerk verwendet. Es besteht die Möglichkeit, dass die auf den DIVA.EXCHANGE angepassten Konfigurationen oder bei den übertragenen Daten eine Deanonymisierung möglich wären. Eine Analyse der DIVA.EXCHANGE Software könnte mithelfen zu beweisen, dass diese ebenfalls anonym ist. Da jeder DIVA.EXCHANGE Teilnehmer auch einen [I2P](#) Service bereitstellt, ist auch für jeden eine öffentlich B32 Adresse bekannt.

7.3. Praktische Umsetzung: Tunnelkontrolle

Das Angriffskonzept wurde innerhalb dieser Arbeit theoretisch erarbeitet. Es wurde eine mathematische Simulation durchgeführt, die aufzeigt, welche Ressourcen für eine praktische Umsetzung benötigt werden. Um diese Theorie zu bestätigen, müsste eine praktische Umsetzung erfolgen. Vorerst müssten die technischen Hürden der Analyseapplikation und der Anpassungen der [Router](#) Implementation bewältigt werden. Danach könnte dieser Angriff innerhalb eines Testnetzwerks geprüft werden. Falls er dort umgesetzt werden kann, sollte der Angriff im öffentlichen [I2P](#) Netzwerk geprüft werden.

7.4. Mitigationsmassnahmen: Tunnelkontrolle

Die optimale Mitigation des Angriffs der Tunnelkontrolle wird erreicht, indem so wenige Tunnels wie nötig und diese so lange wie möglich konfiguriert werden. Die Tunnellänge ist die grösste, durch DIVA.EXCHANGE beeinflussbare Grösse des Angriffs der Tunnelkontrolle. Da längere Tunnels die Latenz erhöhen und die Stabilität verringern, muss mit praktischen Tests die optimale Tunnellänge ermittelt werden. Weiter kann der Angriff mit variablen Tunnellängen erheblich erschwert werden, beschrieben in [Tunnelkontrolle Ansatz 1: Partielle Tunnelkontrolle](#). Aktuell gilt die Annahme, dass variable Tunnellängen die allgemeine Sicherheit erhöhen. Ob variable Tunnellängen wie in [Kombination aus Ansatz 1 & 2](#) beschrieben den Angriff erleichtern können, müsste zuerst noch überprüft

werden. Weiter beeinflusst die Tunnelanzahl den Angriff, da jeder Tunnel eine Gelegenheit für einen Angreifer darstellt. Der Einfluss ist jedoch nicht so markant wie die Tunnellänge. Da jedoch die allgemeine Angriffsfläche erhöht wird, wird empfohlen, die Anzahl Tunnels so klein wie möglich zu halten.

7.5. Testnetzwerk

Das angepasste Testnetz des Entwicklers LNS konnte erfolgreich in Betrieb genommen werden. In Zukunft könnte es um zusätzliche Funktionen erweitert werden. Während dieser Arbeit wurde nur die Integration des [I2PD](#) Routers geprüft, da dieser auch von DIVA.EXCHANGE verwendet wird. Die Integration des [i2p.i2p Routers](#) wurde nicht geprüft.

7.6. Fazit

Eine Deanonymisierung von Teilnehmern innerhalb des [I2P](#) ist nicht auszuschliessen. Der im Detail untersuchte Angriff der Tunnelkontrolle zeigt, dass sich unter bestimmten Umständen eine Deanonymisierung nicht verhindern lässt. Dies ist den Entwicklern von [I2P](#) bekannt. Es sind bereits verschiedene Massnahmen implementiert, die einen solchen Angriff erschweren sollen. Durch diese Massnahmen entstehen zusätzliche, neue Angriffsvektoren. Innerhalb dieser Arbeit konnten keine offensichtlichen Mängel festgestellt werden. Da es sich bei der Entwicklung des [I2P](#) Netzwerks um einen fließenden und sich ständig weiterentwickelnden Prozess handelt, ist weitere Forschung auf diesem Gebiet notwendig. Die aktive Weiterentwicklung des Protokollstacks und der [Router](#) könnten jederzeit neue Angriffsmöglichkeiten bieten.


Anhänge

A. Aufgabenstellung

Projektidee

Antrag zur Ausschreibung einer studentischen Projektarbeit

1. Projektidee

| | |
|-----------------------------------|---|
| Titel: | De-Anonymisierung von Teilnehmern des vollständig verteilten, Blockchain-basierten Handelsnetzwerk für digitale Werte („DIVA.EXCHANGE“). |
| Ausgangslage und Problemstellung: | <p>Ausgangslage: Das freie Software- und Netzwerkprojekt DIVA.EXCHANGE (https://diva.exchange) entwickelt den Softwareprototypen DIVA.</p> <p>Technisch besteht diese freie und quelloffene Software aus einer Anonymisierungsschicht („I2P“), einer auf einer Blockchain basierenden Datenhaltung („Divachain“) und der darauf aufbauenden Handels- und Verwaltungssoftware („DIVA Frontend“).</p> <p>DIVA hat den Zweck aufzuzeigen, wie die Aufbewahrung, der Handel und der Zahlungsverkehr mit digitalen Werten ganz ohne zentrale Dienstleister funktioniert – sicher und mit kompromisslosem Schutz der Privatsphäre.</p> <p>Es handelt sich um ein langfristiges Forschungsprojekt.</p> <p>Aus welchen Komponenten die Gesamtlösung besteht, kann aktuell wie folgt dargestellt werden:</p>  <p>Problemstellung: Die Software „DIVA“ basiert auf dem Anonymisierungsnetzwerk „I2P“. Es soll aufgezeigt werden, wie die Teilnehmer des Netzwerkes de-anonymisiert werden können. De-Anonymisierung ist definiert als „Offenlegung der IP Adresse“ eines am DIVA.EXCHANGE Netzwerk teilnehmenden Knotens.</p> |

Verzeichnisse

Abbildungsverzeichnis

| | |
|---|----|
| 1.1. DIVA.EXCHANGE: Funktionsweise [9] | 1 |
| 2.1. Unterschied zwischen Tunnel Encryption und Transport (Garlic) Encryption [1] | 4 |
| 2.2. End-to-End Message Bundling [15] | 5 |
| 2.3. Inbound/Outbound Tunnels | 7 |
| 2.4. Beispiel Search und Store | 10 |
| 2.5. Vereinfachte Visualisierung eines Bootstrapping-Vorgangs | 12 |
| 3.1. Trello: Kanban-Board für die Projektplanung und Ideen-Datenbank | 15 |
| 3.2. Abarbeitung von Angriffsideen | 21 |
| 4.1. Vollständige Tunnelkontrolle | 27 |
| 4.2. Partielle Tunnelkontrolle | 28 |
| 5.1. Infiltrierter Tunnel mit maximaler Länge | 37 |
| 5.2. Die Erhöhung der Tunnellänge hat einen signifikanten Einfluss | 39 |
| 5.3. Doppelte Routerperformance verglichen mit doppelter Anzahl Router | 40 |
| 5.4. Einfluss der Tunnellänge ist linear | 41 |
| 5.5. Partielle Tunnelkontrolle mit randomisierter Tunnellänge | 42 |

Abkürzungsverzeichnis

BWL Betriebswirtschaftslehre.

DHT Distributed Hash Table.

DoS Denial of Service.

I2NP I2P Network Protocol.

I2P Invisible Internet Project.

IP Internet Protocol.

ZHdK Zürcher Hochschule der Künste.

Glossar

0-Hop-Tunnel Ein 0-Hop-Tunnel ist im eigentlichen Sinn kein Tunnel. Er besteht nur aus einem Sender der Nachricht und keinen weiteren Teilnehmer. Der Sender verhält sich trotzdem wie ein Tunnel, damit die Gegenseite nicht weiss, ob sie mit einem Tunnel oder direkt mit dem Sender kommuniziert .

Floodfill Der Begriff Floodfill hat verschiedenen Bedeutungen. Einerseits wird damit eine Technik bezeichnet, mit der die aufgeteilte [NetDB](#) von einer Teilmenge sogenannter Floodfill Router verwaltet wird. Andererseits wird Floodfill als umgangssprachliche Bezeichnung für Floodfill Router verwendet. Ein Floodfill Router erhält LeaseSets von benachbarten Routern, die er an eine Auswahl bekannter Floodfill Router weiterleitet. Auch wenn ein Sender das LeaseSet seines Ziels nicht kennt, kann er einem Floodfill Router eine Anfrage stellen. Der Floodfill übermittelt ihm dann, falls ihm bekannt, das entsprechende LeaseSet .

i2p.i2p Mit i2p.i2p wird die Implementation des Java [Routers](#) bezeichnet. Sie gilt als originale Implementation, da sie vom Erfinder von [I2P](#) entwickelt wurde .

I2PD Der I2PD ist eine Implementation eines [I2P Router](#)s in C++ [16] .

Inbound Endpoint Beim Inbound Endpoint handelt es sich um den letzten Teilnehmer in einem Inbound Tunnel. Der Inbound Endpoint ist der eigentliche Empfänger der Nachricht und somit in der Lage, diese vollständig zu entschlüsseln .

Inbound Gateway Mit Inbound Gateway wird der erste Teilnehmer des Inbound Tunnels bezeichnet. Die [RouterInfo](#) des Inbound Gateways wird zusammen mit der Tunnel-ID im entsprechenden Lease angegeben. Ähnlich wie der Outbound Gateway kann er Aufgaben wie Padding und Fragmentierung der Nachrichten übernehmen .

Inbound Participants Zu den Inbound Participants gehören alle Teilnehmer im Inbound Tunnel, welche keine Endpoints oder Gateways sind. Ihre Aufgabe ist die Weiterleitung der erhaltenen Nachrichten an den nächsten Teilnehmer des Tunnels .

Inbound Tunnel Mit Inbound Tunnel wird der Tunnel für den eingehenden Datenverkehr bezeichnet. Der Gateway des Inbound Tunnels ist als Lease im LeaseSet der Destination hinterlegt .

Kademlia Kademlia ist ein Algorithmus für das Verteilen einer Datenbank in P2P Netzwerken. Sie bietet den Vorteil, dass die Datenbank nicht allen Teilnehmern vollständig bekannt sein muss. Kademlia beschreibt primär wo und wie die Informationen abgespeichert und wie diese gefunden werden können. .

Konsensus Konsensus ist die übereinstimmende Meinung der Mehrheit aller betroffenen Teilnehmer einer Gruppe oder eines kompletten Netzwerkes zu einer bestimmten Frage, ohne verdeckten oder offenen Widerspruch.

Lease Der Lease enthält die Tunnel ID und RouterIdentity eines Inbound Tunnels.

LeaseSet Das LeaseSet beinhaltet alle derzeit autorisierten Leases für ein bestimmtes Ziel. Ebenfalls enthalten ist ein Public Key für die asymmetrische Verschlüsselung und einen zusätzlichen Public Key für das Prüfen von Signaturen des Ziels. Neben der [RouterInfo](#) ist das LeaseSet eine der beiden in der Netzdatenbank gespeicherten Strukturen. Es wird mit dem SHA256 der enthaltenen Destination verschlüsselt.

NetDB Die NetDB ist eine für das [I2P](#) Netzwerk entwickelte, dezentralisierte Datenbank. Sie beinhaltet die RouterInfos und LeaseSets. Die Datenbank wird für das Finden von weiteren Nodes innerhalb des Netzwerks verwendet. Die NetDB wird durch Floodfills verteilt und verwaltet .

Outbound Endpoint Beim Outbound Endpoint handelt es sich um den letzten Teilnehmer des Outbound Tunnels. Er leitet die entsprechenden Datenpakete an den jeweiligen Inbound Gateway weiter .

Outbound Gateway Der Outbound Gateway ist der erste Teilnehmer des Outbound Tunnels. Er erstellt die Tunnelpakete für den Outbound Tunnel und ist somit ursprünglicher Absender der Nachrichten. .

Outbound Participants Mit Outbound Participants werden alle Teilnehmer im Outbound Tunnel bezeichnet, welche keine Endpoints oder Gateways sind .

Outbound Tunnel Der Outbound Tunnel ist der Tunnel für den ausgehenden Datenverkehr.

Router Der Begriff Router wird in [I2P](#) für die Applikation verwendet, welche den Zugriff ins [I2P](#) Netzwerk verwaltet. Sie übernimmt die Aufgabe des Bereitstellens von Schnittstellen, zum Beispiel eines SOCKs Proxys, verwaltet aber auch die bekannten Teilnehmer und Tunnels..

RouterInfo In der RouterInfo werden alle Daten definiert, die ein Router über sich im Netz zur Verfügung stellen will. Enthalten in der RouterInfo sind primär die RouterIdentity, mit welcher ein Router eindeutig identifiziert werden kann und die RouterAddress. In der RouterAddress sind einerseits die Kosten zur Berechnung der Erreichbarkeit enthalten, wie auch die eigentliche IP-Adresse und Portnummer des Routers .

Routingkey Der Routingkey wird verwendet, um die logische Position, wie auch Distanz mittels Kademia zu bestimmen. Er setzt sich aus dem Hash, bestehend aus Routeridentifikation und Datum zusammen.

Literaturverzeichnis

- [1] „A Gentle Introduction to How I2P Works - I2P.“ (), Adresse: <https://geti2p.net/en/docs/how/intro> (besucht am 03.06.2022).
- [2] Z. Balevsky, *MuWire - Easy Anonymous File-Sharing*, 26. Mai 2022. Adresse: <https://github.com/zlatinb/muwire> (besucht am 26.05.2022).
- [3] K. Bauer, J. Juen, N. Borisov, D. Grunwald, D. Sicker und D. McCoy, „On the Optimal Path Length for Tor,“ S. 17, 2010. Adresse: <https://cs.uwaterloo.ca/~k4bauer/papers/bauer-hotpets2010-paper17.pdf>.
- [4] BWL Lexikon. „Explorative Forschung » Definition, Erklärung,“ BWL-Lexikon.de. (), Adresse: <https://www.bwl-lexikon.de/wiki/explorative-forschung/> (besucht am 04.05.2022).
- [5] W. Casey, A. Kellner, P. Memarmoshrefi, J. A. Morales und B. Mishra, „Deception, identity, and security: The game theory of sybil attacks,“ *Communications of the ACM*, Jg. 62, Nr. 1, S. 85–93, 19. Dez. 2018, ISSN: 0001-0782, 1557-7317. DOI: 10.1145/3190836.
- [6] „Check if NTCP2 address is valid before connection attempt · PurpleI2P/i2pd@2e0019c,“ GitHub. (), Adresse: <https://github.com/PurpleI2P/i2pd/commit/2e0019c8c8c7f524710c9abf5> (besucht am 20.05.2022).
- [7] „Common Structures Specification - I2P.“ (), Adresse: <https://geti2p.net/spec/common-structures> (besucht am 23.03.2022).
- [8] „Contact - I2P.“ (), Adresse: <https://geti2p.net/en/contact> (besucht am 26.05.2022).
- [9] „Die Distributed Ledger Technologie von DIVA.EXCHANGE Ist Iroha : DIVA.EXCHANGE.“ (), Adresse: <https://www.diva.exchange/de/privatsphaere/die-distributed-ledger-technologie-von-diva-exchange-ist-iroha/> (besucht am 04.05.2022).
- [10] diva.exchange. „Diva.exchange i2p codeberg repo,“ Codeberg.org. (), Adresse: <https://codeberg.org/diva.exchange/i2p> (besucht am 23.05.2022).
- [11] DivaExchange. „Reseed Traffic Stats: Strong Traffic Increase,“ r/i2p. (4. Feb. 2022), Adresse: www.reddit.com/r/i2p/comments/skcdcc/reseed_traffic_stats_strong_traffic_increase/ (besucht am 13.05.2022).
- [12] J. (Douceur, „The Sybil Attack,“ in *Proceedings of 1st International Workshop on Peer-to-Peer Systems (IPTPS)*, Jan. 2002. Adresse: <https://www.microsoft.com/en-us/research/publication/the-sybil-attack/>.
- [13] C. Egger, „Practical Attacks Against The I2P Network,“ S. 32, 2013. DOI: 10.1007/978-3-642-41284-4_22.
- [14] B. Evers, J. Hols, E. Kula, J. Schouten und J. A. Pouwelse, „Thirteen Years of Tor Attacks,“ S. 30,
- [15] „Garlic Routing - I2P.“ (), Adresse: <https://geti2p.net/en/docs/how/garlic-routing> (besucht am 06.05.2022).

- [16] „Home - I2pd Documentation.“ (), Adresse: <https://i2pd.readthedocs.io/en/latest/> (besucht am 07.06.2022).
- [17] „I2NP Specification - I2P.“ (), Adresse: <https://geti2p.net/spec/i2np> (besucht am 05.05.2022).
- [18] *I2P*, The I2P Project, 7. Apr. 2022. Adresse: <https://github.com/i2p/i2p.i2p/blob/50ce3c2856ee55816c301600cbbc84395f590cb5/router/java/src/net/i2p/router/networkdb/reseed/Reseeder.java> (besucht am 08.04.2022).
- [19] „I2P Metrics.“ (), Adresse: <https://i2p-metrics.np-tokumei.net/network-size> (besucht am 11.05.2022).
- [20] „I2P Tunnels Configuration - I2pd Documentation.“ (), Adresse: <https://i2pd.readthedocs.io/en/latest/user-guide/tunnels/#serverhttp-tunnels> (besucht am 06.05.2022).
- [21] „I2P's Threat Model - I2P.“ (), Adresse: <https://geti2p.net/en/docs/how/threat-model> (besucht am 06.05.2022).
- [22] *I2pd*, PurpleI2P, 7. Apr. 2022. Adresse: <https://github.com/PurpleI2P/i2pd/blob/b7e20b9b86165a0eb2ba5bcf9a580f3824a38462/libi2pd/Reseed.cpp> (besucht am 08.04.2022).
- [23] „I2Ps Bedrohungsmodell - I2P.“ (), Adresse: <https://geti2p.net/de/docs/how/threat-model> (besucht am 30.05.2022).
- [24] „Index to Technical Documentation - I2P.“ (), Adresse: <https://geti2p.net/en/docs> (besucht am 05.05.2022).
- [25] Z.-Z. H. der Künste. „Aktionsforschung | ZHdK.ch,“ ZHdK. (), Adresse: <https://www.zhdk.ch/forschung/ehemalige-forschungsinstitute-7626/iae/glossar-972/aktionsforschung-3810> (besucht am 04.05.2022).
- [26] M. Küttel, „Untersuchung der Performanz des Invisible Internet Protocols (I2P),“ S. 91,
- [27] l-n-s, *Testnet.Py*, 25. Nov. 2018. Adresse: <https://github.com/l-n-s/testnet.py> (besucht am 19.05.2022).
- [28] letum. „Testnet.py,“ Codeberg.org. (), Adresse: <https://codeberg.org/letum/testnet.py> (besucht am 05.06.2022).
- [29] Y. Marcus, E. Heilman und S. Goldberg, „Low-Resource Eclipse Attacks on Ethereum's Peer-to-Peer Network,“ S. 15,
- [30] P. Maymounkov und D. Mazières, „Kademlia: A Peer-to-Peer Information System Based on the XOR Metric,“ in *Peer-to-Peer Systems*, Ser. Lecture Notes in Computer Science, P. Druschel, F. Kaashoek und A. Rowstron, Hrsg., bearb. von G. Goos, J. Hartmanis und J. van Leeuwen, Bd. 2429, Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, S. 53–65, ISBN: 978-3-540-44179-3. DOI: 10.1007/3-540-45748-8_5.
- [31] J. Müller, „Analysis of the I2P Network - Information Gathering and Attack Evaluations,“ *Berner Fachhochschule*, S. 60, 16. Juni 2016. Adresse: https://www.benoist.ch/research/thesis/FS16/Analysis_of_the_I2P_Network-Bachelorarbeit_Jens_Mueller.pdf.
- [32] „Peer Profiling and Selection - I2P.“ (), Adresse: <https://geti2p.net/en/docs/how/peer-selection> (besucht am 22.05.2022).

- [33] „Permutation mit und ohne Wiederholung,“ Studyflix. (), Adresse: <https://studyflix.de/statistik/permutation-mit-wiederholung-1070> (besucht am 07.06.2022).
- [34] „Reseed Hosts - I2P - <https://geti2p.net/en/docs/reseed>.“ (), Adresse: <https://geti2p.net/en/docs/reseed> (besucht am 18.03.2022).
- [35] *Reserved IP addresses*, in *Wikipedia*, 18. Apr. 2022. Adresse: https://en.wikipedia.org/w/index.php?title=Reserved_IP_addresses&oldid=1083413470 (besucht am 05.06.2022).
- [36] Q. Tan, Y. Gao, J. Shi, X. Wang, B. Fang und Z. Tian, „Toward a Comprehensive Insight Into the Eclipse Attacks of Tor Hidden Services,“ *IEEE Internet of Things Journal*, Jg. 6, Nr. 2, S. 1584–1593, Apr. 2019, ISSN: 2327-4662. DOI: 10.1109/JIOT.2018.2846624.
- [37] „The Network Database - I2P.“ (), Adresse: <https://geti2p.net/en/docs/how/network-database> (besucht am 15.05.2022).
- [38] „Tunnel Creation Specification.“ (), Adresse: <https://geti2p.net/spec/tunnel-creation> (besucht am 13.05.2022).
- [39] „Tunnel Implementation - I2P.“ (), Adresse: <https://geti2p.net/en/docs/tunnels/implementation> (besucht am 11.05.2022).
- [40] „Tunnel Message Specification - I2P.“ (), Adresse: <https://geti2p.net/spec/tunnel-message> (besucht am 11.05.2022).
- [41] „Tunnel Routing - I2P.“ (), Adresse: <https://geti2p.net/en/docs/how/tunnel-routing> (besucht am 13.05.2022).
- [42] Q. Wang, P. Mittal und N. Borisov, „In search of an anonymous and secure lookup: Attacks on structured peer-to-peer anonymous communication systems,“ S. 11, Okt. 2010. DOI: 10.1145/1866307.1866343.
- [43] „What Is an Eclipse Attack? | The Radix Blog | Radix DLT.“ (), Adresse: <https://www.radixdlt.com/post/what-is-an-eclipse-attack> (besucht am 05.05.2022).
- [44] H. Yin und Y. He, „I2P Anonymous Traffic Detection and Identification,“ in *2019 5th International Conference on Advanced Computing & Communication Systems (ICACCS)*, März 2019, S. 157–162. DOI: 10.1109/ICACCS.2019.8728517.
- [45] „Zzz.I2p: Development Discussions.“ (), Adresse: <http://zzz.i2p/> (besucht am 08.06.2022).