Immanuel-Kant-Gymnasium Heiligenhaus

Schuljahr 2019/20

Anonymität garantiert?! –
Eine Sicherheitsanalyse des Tor-Netzwerks

vorgelegt von

Elias Mitropoulos

Grundkurs Informatik / Fachlehrer: Martin Tilmans

25. März 2020

Inhaltsverzeichnis

1. Einleitung

Spätestens seit dem Inkrafttreten der europäischen Datenschutzgrundverordnung (DSGVO) hat uns die Thematik der Anonymität und des Datenschutzes im Internet auch im alltäglichen Leben eingeholt. Neben dem lästigen zusätzlichen Papierkram, beispielsweise beim Online-Shopping, der durch diese Neuerung entstanden ist, hat die DSGVO bei vielen Menschen beispielsweise durch extrem lange Datenschutzerklärungen auch mit einem Irrtum endgültig aufgeräumt: Im Internet ist man nicht anonym! Beim Nutzen des Internets hinterlässt man nämlich einen gewaltigen Fußabdruck an Daten, auch „digital footprint" genannt. Über Cookies und sonstige log-Dateien werden große Mengen an Nutzerdaten erfasst und mithilfe dieser Daten anschließend das Nutzerverhalten analysiert. Viele Experten auf dem Gebiet der *Data Science* gehen davon aus, dass große Internetgiganten, wie beispielsweise Google oder Facebook, eine Person durch diese Datenanalysen ab einer bestimmten Aktivität besser einschätzen können als sie sich selbst (vgl. Grassegger/Krogerus 2018).

Diese Informationen haben bei mir die Frage aufgeworfen, ob es denn keine Möglichkeit gibt, sich im Internet anonym zu bewegen und seine Identität zu schützen.

Die Antwort auf diese Frage liefern moderne Anonymisierungsdienste wie das in diesem Bereich weit verbreitete Open-Source-Projekt Tor. Dieses stellt ein Anonymisierungsnetzwerk dar, was seinen Nutzern dabei hilft, sich gegenüber Websites und anderen Services im Internet anonym zu halten und beispielsweise den Standort, Geräteinformationen und Ähnliches zu verschleiern. Doch an einen Anonymisierungsdienst werden neben dem Schutz der Identität noch viele weitere Anforderungen gestellt, um überhaupt unter realen Gegebenheiten zu funktionieren.

Ein Anonymisierungsdienst wie Tor muss nutzerfreundlich sein. Er muss unabhängig von Hardware- und Softwareveränderungen funktionieren, also über ein eigenes Programm (Client) laufen. Des Weiteren sollte er ressourceneffizient arbeiten und wenig Latenz verursachen. Wichtig ist außerdem, dass er ausfallsicher funktioniert und skalierbar ist, also auf das Nutzeraufkommen entsprechend reagieren kann. Es sollte zudem die tatsächliche Identifikation des Nutzers (z.B. IP-Adresse) verborgen werden, zum Beispiel durch die Integration des Stellvertreterprinzips[1] (vgl. Miller 2006, S.17 f.). Darüber hinaus sollte auch die Benutzeroberfläche des Clients, mit dem auf den Dienst zugegriffen werden kann, einfach zu bedienen und klar strukturiert sein (vgl. Strefler 2006, S.33). Das alles sind anzustrebende Ziele, um viele Nutzer an den eigenen Dienst zu binden und ihnen zeitgleich bestmögliche Anonymität bieten zu können.

Aber wie sicher ist ein Anonymisierungsdienst, der versucht, die genannten Anforderungen zu erfüllen? In dieser Facharbeit soll das Tor-Netzwerk dahingehend untersucht werden, ob es absolute Anonymität gewährleisten kann. Dazu wird zunächst die Funktionsweise des Tor-Netzwerks beleuchtet. Anschließend wird eine tiefgehende Sicherheitsanalyse des Netzwerks

[1] **Stellvertreterprinzip:** Der Senderserver baut nicht mehr direkt und eigenverantwortlich eine Verbindung zum Empfänger auf, sondern wird über einen Zwischenserver geleitet. Dieser baut stellvertretend für den Sender eine Verbindung zum Empfänger auf (vgl. DATACOM Buchverlag: Web-Proxy 2020 und Vogel 2020).

durchgeführt, in der Angriffsmöglichkeiten auf das Netzwerk, weitergehende Gefahren für den Nutzer und die Tor-Community als Sicherheitskriterium Betrachtung finden. Abschließend werden die Analyseergebnisse in einem Fazit unter Beachtung der Ausgangsfrage reflektiert und bewertet.

2. Funktionsweise des Tor-Netzwerks

Das Tor-Netzwerk ist ein Overlay-Netzwerk[2], welches auf dem sogenannten Onion Routing basiert. Es wird von den Entwicklern und der allgemeinen Öffentlichkeit als „second-generation Onion Routing system“ (Dingledine et al. 2004, S.1) bezeichnet, weil bei Tor einige Veränderungen am Grundprinzip des Onion Routing vorgenommen wurden und es somit als Weiterentwicklung des klassischen Onion Routing gesehen werden kann.

Ein Onion Routing-Netzwerk besteht grundsätzlich aus zahlreichen Zwischensystemen, den Onion Routern, welche über langlebige Verbindungen alle miteinander verbunden sind. Bei einer Datenübertragung über das Netzwerk wird nicht direkt eine Verbindung vom Sender zum Empfänger aufgebaut. Der Verbindung sind mehrere Onion-Router zwischengeschaltet, die in ihrer Gesamtheit einen verschlüsselten Kanal, auch Circuit genannt, bilden. Die Onion-Router in einem Circuit kennen jeweils nur ihren unmittelbaren Vorgänger und ihren unmittelbaren Nachfolger. So ist sichergestellt, dass der Kommunikationskanal selbst innerhalb des Netzwerkes anonym gehalten wird. Beim Tor-Netzwerk kann ein Circuit gleichzeitig von mehreren Nutzern zur Datenübertragung genutzt werden und muss nicht immer vollständig durchlaufen werden. Die Nutzer initiieren auf einem Circuit eigene kleinere Kommunikationspfade, die sich durch eine individuelle Pfad-ID identifizieren lassen. Sie werden Streams genannt. Am Ende eines jeden Streams findet sich ein spezieller Exit-Onion-Router, durch welchen der übertragene Datenstrom das Netzwerk verlassen kann. Der Exit-Onion-Router baut dann stellvertretend für den Sender eine direkte Verbindung zum Empfänger auf.

2.1. Allgemeiner Kommunikationsablauf

Der Kommunikationsablauf im Tor-Netzwerk unterteilt sich in drei logische Phasen. Zuerst findet der Verbindungsaufbau statt, anschließend gibt es eine Phase, in der bidirektionale Kommunikation[3] zwischen dem Sender und dem Empfänger ermöglicht wird und dann die dritte Phase, in welcher die Verbindungstermination eingeleitet wird.

Möchte man eine Verbindung über das Tor-Netzwerk aufbauen, benötigt man zuerst einen Client, der eine Verbindung zu einem Onion-Proxy aufbauen kann. Dafür eignet sich beispielsweise der von den Tor-Entwicklern vertriebene „Tor-Browser“. Mithilfe dieses Clients können nach der Installation ein lokal auf dem eigenen Computer eingerichteter Onion-Proxy oder öffentliche Onion-Proxys erreicht werden. Der Onion-Proxy fungiert beim Verbindungsaufbau als Schnittstelle zwischen Client und dem Tor-Netzwerk. Wendet sich ein Client an einen

[2] **Overlay-Netzwerk**: Ein Rechnernetz baut auf einer existierenden Infrastruktur auf, nutzt aber eine eigene Struktur. Tor nutzt im Gegensatz zum herkömmlichen Internet alternative Routen zur Datenübertragung. Also baut es als Overlay-Netzwerk auf dem Internet als Underlay-Netzwerk auf (vgl. DATACOM Buchverlag: Overlay-Netz 2020).

[3] **Bidirektionale Kommunikation**: Bei einem Datenkommunikationskanal gibt es einen Hin- und einen Rückkanal. Das bedeutet hier, dass die Onion-Router innerhalb eines Streams nicht nur in der Lage sind, Datenströme weiterzusenden, sondern zeitgleich auch rückläufig empfangen können (vgl. DATACOM Buchverlag: Bidirektional 2020).

Onion-Proxy, so wählt der Proxy den neusten Circuit aus einem von einem Directory-Server verwalteten Pool an bereits erstellten Circuits aus, um über ihn später die zu sendenden Daten zu verschicken. In diesem Zusammenhang erhält er von diesem Directory-Server auch die öffentlichen Schlüssel aller sich auf dem Circuit befindenden Onion-Router, um später Onions erstellen zu können. Auf dem gewählten Circuit wählt der Proxy einen passenden Onion-Router als Exit-Onion-Router[4] aus und sendet ihm eine Kommando-Onion mit dem Kommando „begin“ („begin“-Onion) durch den Circuit. Erhält der gewählte Exit-Onion-Router die „begin“-Onion, stellt er eine Verbindung zum Zielserver her und antwortet dem Onion-Proxy mit einer „connected“-Onion, die über die gleiche Netzwerkroute zurück an den Onion-Proxy gesendet wird. Sobald der Onion-Proxy die erfolgreiche Antwort des Exit-Onion-Routers erhalten hat, ist ein neuer Stream, über welchen Daten übertragen werden können, aufgebaut. Der Onion-Proxy empfängt vom Nutzer beziehungsweise dessen Client anschließend die Daten, welche zur Übertragung über den neu geschaffenen Stream bestimmt sind (siehe *Abbildung 1*).

Danach hat der Onion-Proxy die Aufgabe, die vom Client erhaltenen Daten in ein generisches Dateiformat zu überführen, das von allen Onion-Routern interpretiert werden kann. Für die Übertragung müssen die Daten in jeweils 512-Byte große Onions verpackt werden (vgl. Dingledine et al. 2004, S.5 f.). Weil die vom Nutzer zur Übertragung bestimmten Daten in der Regel nicht exakt der geforderten Größe entsprechen, muss der Onion-Proxy die Daten vor dem Verschicken entweder bei größerem Umfang fragmentieren, also auf mehrere Onions verteilen, oder mithilfe von Dummy-Traffic[5] eine Hochskalierung der betreffenden Onion vornehmen. Eine Onion besteht aus mehreren Verschlüsselungsschichten, die sich quasi um den zu übertragenden Nutzdaten befinden. Jeder Onion-Router innerhalb des Streams erhält vom Onion-Proxy zu Anfang einer jeden Übertragung einen Schlüssel, um die jeweils oberste Schale der Onion entschlüsseln zu können (siehe Kapitel 2.3).

Die Onions werden vom Onion-Proxy nach der Verbindungsaufbauphase an den ersten Onion-Router des Streams gesendet. Dieser löst die oberste Schale der Onion ab. Diese enthält unter anderem Adressinformationen des nächsten Onion-Routers im Stream, an den die Daten weitergeschickt werden sollen. Wenn der zweite Onion-Router die Onion von seinem Vorgänger erhalten hat, entschlüsselt er wiederum die oberste Schale der Onion und leitet die nun um wieder eine Schalenschicht kleinere Onion an seinen Nachfolger gemäß den in der abgeschälten Schale enthaltenen Adressinformationen weiter. Dieses Vorgehen wiederholen alle weiteren Onion-Router innerhalb des Streams, bis die Onion beim Exit-Onion-Router eintrifft. Der Exit-Onion-Router löst die letzte Schale der Onion ab, um an die zu übertragenden Nutzdaten zu kommen. Er wandelt die Daten wieder in ein spezifisches Dateiformat um, setzt sie, sofern die Daten zuvor fragmentiert wurden, wieder zusammen und leitet diese anschließend gemäß den in der Schale enthaltenen Adressinformationen weiter. Bei dem Empfänger handelt es sich um den Zielserver, den der tatsächliche Sender erreichen möchte.

[4] **Passender Onion-Router als Exit-Onion-Router:** Gemeint ist hier, dass es sich um einen Onion-Router mit einer positiven Exit-Policy handeln muss (siehe Kapitel 2.2).

[5] **Dummy-Traffic:** Platzhalter-Daten werden erzeugt, um künstlich eine bestimmte Größe eines Elements zu erhalten. Die Daten sind entgegen der zu übertragenen Nutzdaten an sich völlig sinnlos und dienen nur dem Mittel zum Zweck. Das Verfahren dahinter nennt sich „**padding**“.

Über eine gewisse Zeitspanne kann der Empfänger eines Datenstromes dem Sender über den erstellten Stream antworten, egal, ob er selbst Partizipant des Tor-Netzwerks ist oder nicht. Bei einer Antwort des Empfängers auf des Senders Datenstrom wird der Stream rückwärts durchlaufen, und auch das Bilden bzw. Abschälen einer Onion erfolgt quasi entgegengesetzt. Jeder Onion-Router des Streams, beginnend mit dem Exit-Onion-Router, ergänzt um die Nutzdaten der Antwort herum eine Schale einer Onion, wodurch der Onion-Proxy, der nun am Ende des Streams steht, eine vollumfänglich verschlüsselte Onion mit der Antwort des ehemaligen Empfängers erhält. Dadurch, dass der Onion-Proxy jeden Schlüssel mit jedem Onion-Router innerhalb des Streams kennt, ist er in der Lage, die „Antwort-Onion(s)“ alleine abzuschälen und die Antwort danach direkt in ein spezifisches Format umzuwandeln, gegebenenfalls zusammenzusetzen und an den Client weiterzuleiten (vgl. Miller 2006, S.23 f.) (siehe *Abbildung 2*).

Nach diesem Zeitraum, in dem durch den Stream eine bidirektionale Kommunikation zwischen dem Sender und dem Empfänger ermöglicht wird, ist eine direkte Antwort auf die Nachricht des Senders durch den Empfänger nicht mehr möglich. Der Stream wird dann nämlich durch eine „end-Onion“, die vom Onion-Proxy an den Exit-Onion-Router des Pfades geschickt wird, geschlossen. Der Exit-Onion-Router gibt die Verbindung zum Zielserver bei Erhalt dieser Onion auf und bestätigt die Schließung des Streams mit einer abschließenden „reply-end-Onion“ (vgl. Strefler 2006, S.39 f.) (siehe *Abbildung 3*).

Wenn der seltene Fall auftreten sollte, dass ein Onion-Router innerhalb des Streams nicht länger erreichbar ist, so informiert der Vorgänger des ausgefallenen Onion-Routers den Onion-Proxy über diesen Verbindungsabbruch, indem er eine „teardown-Onion“ an diesen sendet. Der Onion-Proxy leitet daraufhin die sofortige Schließung des betreffenden Streams ein (s.o.). Alle Datenübertragungen eines Circuit werden durch einen extra von Tor entwickelten Staukontrollmechanismus, der Signale vom Exit-Onion-Router an den Onion-Proxy erzeugt, verifiziert, um eine Überlastung eines Circuits durch das zu hohe Aufkommen von Streams über diesen Circuit ausschließen zu können (vgl. Strefler 2006, S.39 f.).

2.2. Zwischensysteme als Säulen des Netzwerks

Das Tor-Netzwerk bildet sich mithilfe vieler Freiwilliger, die ihre eigene Infrastruktur bestehend aus der eigenen Internetbandbreite und der Rechnerleistung des Computers, welcher als Onion-Router fungiert, für das anonyme Bewegen im Internet anderen Benutzern zur Verfügung stellen (vgl. Çalışkan et al. 2015, S.7 ff.).

Im Folgenden werden die verschiedenen Arten von Zwischensystemen, die als Säulen des gesamten Tor-Netzwerks zu sehen sind, dargestellt und erläutert.

Bei den meisten netzwerkspezifischen Zwischensystem des Tor-Netzwerks handelt es sich um Onion-Router. Jeder Onion-Router besitzt einen „long-term identity key“, durch welchen er eindeutig identifizierbar ist. Über die Onion-Router findet die Datenübertragung zwischen einem Sender und einem Empfänger statt, die über einen verschlüsselten Kommunikationspfand bidirektional kommunizieren können. Ein Onion-Router innerhalb eines Streams speichert neben seinem unmittelbaren Vorgänger und Nachfolger auch die eindeutige Pfad-ID des

Streams, um eintreffende Datenströme den richtigen Streams zuzuordnen, falls mehrere Streams über einen Circuit laufen oder ein Onion-Router in mehrere Circuits gleichzeitig eingebaut ist.

Die Betreiber eines Onion-Routers können sich über sogenannte „variable Exit-Policies“ zudem dazu bereit erklären, als Exit-Onion-Router zu fungieren. Exit-Onion-Router stellen in erster Linie ebenfalls ganz gewöhnliche Onion-Router dar, können jedoch, anders als klassische Onion-Router, am Ende eines Streams oder Circuits stehen. Die besondere Möglichkeit der „Exit-Policy“ wurde in der Tor-Software integriert, um mehr freiwillige Bereitsteller von Onion-Routern zu erreichen. Denn es lässt sich in der „Exit-Policy“ eines Onion-Routers auch einstellen, zu welchen Servern oder Ports man eine Verbindung verweigern möchte. Viele potenzielle Unterstützer scheuen sonst nämlich davor zurück, einen Exit-Onion-Router zu betreiben (siehe Kapitel 3.2) (vgl. Michaelis 2012, S.12 f.).

Doch wie wird das gesamte Tor-Netzwerk eigentlich verwaltet und zusammengehalten? Das geschieht bei Tor mithilfe sogenannter Directory-Server, die als zentrale Router-Verzeichnisse des gesamten Netzwerkes fungieren. Die Directory-Server sind besonders vertrauenswürdige Server, die Router-Informationen über jeden Onion-Router des Netzwerks verwalten und somit stetig ein Abbild des kompletten Tor-Netzwerks bereitstellen. Jeder Onion-Router schickt dazu in regelmäßigen Abständen eine von ihm signierte „Router-Description“ an alle Directory-Server, in welcher er über seinen eigenen Zustand informiert. Diese enthält den „long-term identity key“ des Onion-Routers, die Adresse und den öffentlichen Schlüssel des Onion-Routers sowie dessen Exit-Policy. Die Directory-Server gleichen die Ansammlungen an erhaltenen Router-Berichten regelmäßig ab, um ein einheitliches Abbild des Netzwerkes zu schaffen. Nur Onion-Router, die in den Berichten der Mehrheit der Verzeichnisserver auftauchen, werden auch in den abschließend signierten Netzwerkbericht aufgenommen und sind weiterhin Teil des Netzwerks. Die gesamten Netzwerkberichte werden vom Onion-Proxy benötigt, um eine Verbindung ins Netzwerk aufbauen zu können, denn über jene Berichte kann der Proxy auch auf einen Pool an bereits erstellten Circuits, die durch das Tor-Netzwerk führen, zugreifen und somit den Verbindungsaufbau einleiten (siehe Kapitel 2.1).

Eine weitere Möglichkeit für engagierte Freiwillige, sich am Tor-Netzwerk zu beteiligen, sind die sogenannten Bridges. Bridges stellen spezielle, nicht in den Directory-Servern gelistete Entry-Onion-Router dar, die es ermöglichen, dass auch Menschen ins Tor-Netzwerk gelangen können, bei welchen die öffentlich gelisteten Onion-Router des Tor-Netzwerks und Tor-Verbindungen generell, beispielsweise durch den Staat, blockiert werden. Der Client eines derartig betroffenen Internetnutzers kann über eine bestimmte E-Mail-Adresse automatisiert eine kleine Liste sich ständig ändernder Bridge-Serveradressen durch eine unabhängige Verteilerstelle erhalten, die er dann an den Onion-Proxy übergeben kann. Dieser baut mithilfe dieser temporären Adressen eine Verbindung zur Bridge auf. Das Besondere dabei ist, dass derartig vom Nutzer initiierte Datenströme nicht als Tor-Verbindungen erkannt werden können. Sobald die Verbindung zur Bridge besteht, befindet man sich innerhalb des Tor-Netzwerks. Eine externe Beobachtung oder Kontrolle (zum Beispiel eine Zensur durch Behörden) ist dann nicht mehr möglich (vgl. Çalışkan et al. 2015, S.10).

2.3. Struktur und Funktion einer Onion

Die Datenübertragung innerhalb des Tor-Netzwerks erfolgt über immer gleichgroße Datenpakete, die auch als Zellen oder Onions bezeichnet werden. Die Onion ist eine verschachtelte Datenstruktur, die asymmetrisch verschlüsselt ist.

Jede Onion hat im Tor-Netzwerk eine exakte Größe von 512 Byte. Der Onion-Proxy kreiert die Onion, das geschieht von hinten nach vorne. Zunächst nutzt er den durch die Directory-Server ihm bekannten öffentlichen Schlüssel des Exit-Onion-Routers des Streams zur Verschlüsselung der ersten „Schale". Anschließend verschlüsselt der Onion-Proxy eine zweite „Schale", die über der ersten liegt, mit dem öffentlichen Schlüssel des vorletzten Onion-Routers im Stream. Dies wiederholt der Proxy solange, bis die Schale des ersten Onion-Routers im Stream oben aufliegt.

Wenn die fertige Onion, die nun mehrere Verschlüsselungsschichten umfasst, zum ersten Onion-Router des Streams geschickt wird, kann er die oberste Schale der Onion mit seinem privaten Schlüssel entschlüsseln, weil jene durch den Onion-Proxy mit dem zugehörigen öffentlichen Schlüssel verschlüsselt wurde[6]. Mit jeder Entschlüsselung wird die Onion optisch bzw. kryptografisch verändert (siehe *Abbildung 4*).

Eine Schale der Onion enthält immer die Circuit-ID, die angibt, über welchen Circuit die Onion läuft. Durch die eindeutige Circuit-ID kann sichergestellt werden, dass ein Onion-Router zeitgleich auch Teil mehrerer Circuits sein kann. Näher spezifiziert wird diese Information durch die ebenfalls in der Schale enthaltene eindeutige Stream-ID. Die Stream-ID ist entscheidend, damit der Exit-Onion-Router des Streams erkennt, dass er der letzte Onion-Router innerhalb des Streams ist. Ebenfalls in jeder Schale einer Onion enthalten sind die Adressinformationen des nachfolgenden Onion-Routers, an welchen die Onion weitergesendet werden soll. Außerdem enthält jede Schale das exp_time-Attribut, über welches geregelt wird, wie lange die Onion gültig ist und somit entschlüsselt werden kann. Übersteigt eine Onion diese Gültigkeitsdauer, wird sie zerstört. Es gibt in jeder Schale auch noch ein CMD-Attribut, über welches angegeben wird, was die Onion bezwecken möchte. Beispielweise könnte sie dazu eingesetzt werden, einen Stream aufzubauen („begin"-Onion) oder einen Stream schließen („end"-Onion). Das auch enthaltene length-Attribut gibt die Datengröße der restlichen Onion ohne die abgeschälte „Schale" an. Unter der Schale kommt die kleinere Onion nämlich zum Vorschein und kann weiterverarbeitet werden (vgl. Dingledine et al. 2004, S.5) (siehe *Abbildung 5*).

Ebenfalls in der Schale enthalten ist zusätzliches Schlüsselmaterial in Form symmetrischer Schlüssel („key-Seed"), welche jeweils die direkte Verbindung des betreffenden Onion-Routers zu seinem Vorgänger und Nachfolger absichern. Das trägt zur Sicherung des gesamten Kommunikationspfades nach der Erstellung des Streams bei (vgl. Miller 2006, S.22).

Es wird deutlich, dass eine Onion mit jedem Abschälen einer „Schale" immer kleiner wird. Um Angriffen entgegenzuwirken wird jede Onion nach dem Abschälen einer „Schale" mit dummy-

[6] Asymmetrische Verschlüsselungsweise (vgl. Gabler Wirtschaftslexikon 2020).

Traffic aufgefüllt und so auf die ursprüngliche Größe von 512 Byte gebracht (siehe Kapitel 3.1).

2.4. Versteckte Dienste im Tor-Netzwerk

Tor bietet nicht nur die Möglichkeit, dass sich Nutzer im Internet anonym bewegen können, sondern gibt auch Servern die Möglichkeit, sich gegenüber anderen Servern und Benutzern im Internet anonym zu halten.

Ein im Tor-Netzwerk integrierter anonymer Dienst wird Hidden Service genannt. Wenn ein Nutzer einen Kommunikationskanal zu einem anonymen Server aufbauen möchte, kommt ein neutraler Ort, ein Rendez-vous-Point, in Form eines anderen Onion-Routers ins Spiel, an welchem sich der Nutzer und der Server „treffen“ können (siehe *Abbildung 6*). Der Nutzer wählt einen beliebigen Onion-Router des Netzwerks mithilfe des Onion-Proxys als Rendez-vous-Point aus. Der Onion-Proxy kreiert daraufhin einen Circuit vom Nutzer hin zum gewählten Rendez-vous-Point. Anschließend sendet der Nutzer über seinen Onion-Proxy eine .onion-Domain[7] an einen Directory-Server. Der Directory-Server antwortet dem Onion-Proxy daraufhin mit der Adresse eines öffentlich bekannten Introduction-Points. Bei diesem Server handelt es sich um einen von vielen vom anonymen Server bereitgestellten Zwischenserver. Dieser Zwischenserver hält einen ständigen Stream zum anonymen Server offen, damit der tatsächliche Server auch netzwerkintern anonym bleibt. Der Onion-Proxy schickt mithilfe der Adresse des Introduction-Points eine Nachricht an diesen, um quasi ein „Treffen“ am Rendez-vous-Point zu vereinbaren. Der Introduction-Point leitet die Bitte um ein „Treffen“ am Rendez-vous-Point über den permanent offenen Stream zwischen ihm und dem tatsächlichen Server an jenen weiter. Sofern der Server eine Verbindung vom Nutzer zulassen möchte (es gibt auch eine Blacklist!), baut er einen Circuit zum Rendez-vous-Point auf, an dem der Nutzer bereits „wartet“. Haben sowohl der Nutzer als auch der tatsächliche Server einen offenen Stream zu dem als Rendez-vous-Point fungierenden Onion-Router aufgebaut, so ist es die Aufgabe dieses Onion-Routers, die beiden Streams miteinander zu verbinden. Wenn beide Streams zu einem Stream verbunden wurden, kann der Nutzer einen neuen Verbindungsaufbau an den tatsächlichen Server initiieren (vgl. Michaelis 2012, S.14 ff.). Hierbei ist gleich vorzugehen wie bei dem Wunsch, eine anonyme Verbindung zu einem Empfänger aufzubauen, welcher im „normalen“ Internet stationiert ist (siehe Kapitel 2.1).

3. Sicherheitsanalyse des Tor-Netzwerks

Die Nutzung des Tor-Netzwerks gilt als die populärste und sicherste Möglichkeit, sich im Internet anonym zu bewegen. Aber dass sich jeder Mensch über das Tor-Netzwerk scheinbar der generellen Protokollierung von Aktivitäten im Internet entziehen kann, macht Tor auch so interessant. Dieses Interesse mündet in Angriffen auf das Tor-Netzwerk, wodurch versucht wird, Benutzer des Tor-Netzwerks gezielt zu deanonymisieren. Es gibt aber auch Gefahren, die einen Nutzer bei der Kommunikation über das Tor-Netzwerk deanonymisieren könnten.

[7] **.onion-Domain:** Eine .onion-Domain ist ein 16-Zeichen langer Identifier. Sie ist keine klassische Top-Level-Domain, kann aber von den Tor-Netzwerk-Verwaltungssystemen, also den Directory-Servern, interpretiert werden.

Im Folgenden sollen in einer theoretisch angelegten Sicherheitsanalyse generelle Angriffsmöglichkeiten auf Netzwerke aufgezeigt und auf ihre Umsetzbarkeit im Tor-Netzwerk überprüft werden. Des Weiteren werden Tor-spezifische Angriffsmöglichkeiten betrachtet. Anschließend sollen über die Angriffsmöglichkeiten hinaus weitere Gefahren für die Nutzer des Tor-Netzwerks beschrieben werden. Abschließend wird die Tor-Community als Sicherheitskriterium betrachtet.

3.1. Angriffsmöglichkeiten

Für die Analyse der Angriffsmöglichkeiten muss an dieser Stelle zunächst ein einheitliches Angriffsszenario definiert werden. Ein gezielter Angriff auf das Tor-Netzwerk geht immer von einem kompromittierten Onion-Router[8] aus. Wenn ein Stream nur aus kompromittierten Onion-Routern besteht, ist die Anonymität quasi nicht mehr (lange) zu erhalten. Im Fall Tor wurde jedoch untersucht, dass nur die NSA theoretisch die Kapazitäten hätte, eine ausreichend große Anzahl an Zwischensystemen des Tor-Netzwerks für solch großangelegte Angriffe zu kontrollieren (vgl. Sneier 2013). Daher ist dieser Fall weder besonders realistisch noch zielführend für diese Analyse. Dass die Anonymität gar nicht mehr gewährleistet werden kann gilt auch für den Fall, bei welchem der Nutzer einen kompromittierten Client oder Onion-Proxy benutzt. Die Angriffsanalyse bezieht sich also auf ein Szenario, bei welchem mindestens ein Onion-Router innerhalb des Streams ehrlich ist und der Nutzer einwandfreie Hard- und Software zum Eintritt in das Netzwerk verwendet.

3.1.1 Netzwerkspezifische Angriffe

Um im Allgemeinen Netzwerksysteme anzugreifen, sind viele Angriffsarten in Betracht zu ziehen. Bei einer Message-Volume-Attacke wird versucht, die Datenpakete über ihre Größe zu identifizieren. Ein derartiger Angriff ist über Tor nicht umsetzbar, denn alle über Tor übertragenen Datenpakete haben in Form von Onions dauerhaft eine einheitliche Größe von 512 Byte.

Eine Message-Coding-Attacke lässt sich durch einen Angreifer realisieren, wenn sich die Darstellung eines über das Netzwerk übertragene Datenpaket nicht ändert. Da alle Daten, welche über das Tor-Netzwerk übertragen werden, in Onions verpackt sind, ist ein solcher Angriff bezogen auf das Tor-Netzwerk nicht umsetzbar. Denn eine Onion verändert bei ihrem Weg durch das Netzwerk ständig ihre Darstellung, weil ständig eine Schale abgeschält wird und eine andere Schale zum Vorschein kommt. Das Erscheinungsbild der Onion ändert sich also ständig, eine Beobachtung anhand der Darstellung ist nicht möglich.

Ein denkbarer Angriff auf Netzwerksysteme ist auch die Flooding-Attacke, welche mit einer DOS-Attacke gleichzusetzen ist. Bei einem derartigen Angriff wird entweder versucht, ein eigens kontrolliertes Zwischensystem des Netzwerks soweit mit böswilligen oder sinnlosen Daten zu belegen, dass nur noch eine über das Netzwerk übertragene Datei das Zwischensystem passieren kann, oder es wird großangelegt versucht, sämtliche andere Zwischensysteme durch Überlastungen lahmzulegen und den Nutzer quasi dazu zu zwingen, eine Route mit

[8] **Kompromittierter Onion-Router**: Ein von einem Angreifer kontrollierter bzw. missbrauchter und somit schadhafter Onion-Router.

dem eigens kompromittierten Zwischensystem zu kreieren. Überlastungen des Netzwerks könnten zum Beispiel durch das Senden zu großer Datenmengen über eine Route oder auch das ständige Auf- und Abbauen von Routen initiiert werden. Gegen eine Flooding-Attacke gewährt das Tor-Netzwerk auf den ersten Blick keinen Schutz, jedoch ist im Tor-Netzwerk ein Staukontrollmechanismus integriert, der verhindert, dass einzelne Onion-Router wegen zu vielen eigenen Aktivitäten überlastet werden und ausfallen könnten. Das ständige Auf- und Abbauen von Streams über das Tor-Netzwerk ist erst einmal möglich, jedoch wird auch hier irgendwann ein Staukontrollmechanismus des verwendeten Onion-Proxys eingreifen und die Datenrate des Clients, von dem derartige Aktivitäten ausgehen, drosseln.

Durch die Zusammenarbeit verschiedener Zwischensysteme miteinander könnte versucht werden, den Datenverkehr der Nutzer auf vollständig vom Angreifer kontrollierte Netzwerkrouten umzuleiten und dann abzuhören. Eine solche Collusion-Attacke kann jedoch lediglich in Verbindung mit einer Message-Tagging-Attacke durchgeführt werden. Zunächst müssen die Daten, die auf eine vollständig kompromittierte Netzwerkroute umgeleitet werden sollen von einem Zwischensystem markiert werden, um identifizierbar zu sein. Danach könnte ein Angreifer diese markierten Dateien über ein weiteres kompromittiertes Zwischensystem auf die vollständig kompromittierte Netzwerkroute umleiten. Doch der Haken an so einer kombinierten Attacke auf das Tor-Netzwerk lässt sich bereits am Anfang erkennen. Es ist über das Tor-Netzwerk eigentlich nicht möglich, einzelne Dateien oder Datenströme zu markieren. Sofern die Markierungen auf der aus der Sicht des ersten kompromittierten Zwischensystems obersten Schale einer Onion platziert sind, was die einzige umsetzbare Möglichkeit darstellt, wird diese beim Entschlüsseln der Onion durch den nachfolgenden Onion-Router im Stream bemerkt. Der Nachfolger des ersten kompromittierten Onion-Routers würde nämlich feststellen, dass sein privater Schlüssel zum Entschlüsseln der obersten Schale nicht mehr passt, was daran liegt, dass die Verschlüsselung durch die Markierung des Vorgängers beeinflusst wurde. Daraufhin würde der Onion-Router einen sofortigen Verbindungsabbruch einleiten. Einzig und allein das Szenario, dass es sich bei dem in die Attacke involvierten zweiten Onion-Router um den unmittelbaren Nachfolger des ersten kompromittierten Onion-Routers handelt, kann nicht abgesichert werden. Der Angreifer würde die Markierung der Onion erkennen und den Zerstörungsinstinkt, den ein ehrlicher Onion-Router hätte, unterdrücken. Dann könnte der Datenstrom nach Belieben umgeleitet werden. Zur zusätzlichen Sicherung ist im Tor-Netzwerk ein Ende-zu-Ende-Integritätsschutz integriert, welcher alle Daten, bevor sie das Netzwerk über einen Exit-Onion-Router verlassen, auf ihre Unversehrtheit[9] überprüft.

Eine Intersection-Attacke beschreibt die Analyse von typischem Nutzerverhalten. Dazu werden über einen längeren Zeitraum große Datenmengen über kompromittierte Zwischensysteme des Netzwerks protokolliert, wodurch es später unter Umständen möglich ist, Nutzergruppen ausfindig zu machen, welche durch die erhaltenen Statistiken auffallen, beispielsweise durch immer ähnliche Onlinezeit (vgl. Konigorski 2006, S.4). Eine solche Attacke ist bezogen auf das Tor-Netzwerk ebenfalls nicht sehr realistisch. Denn das Tor-Netzwerk ist

[9] Es wird die Vertrauenswürdigkeit und die Erhaltung der Ursprungsdaten geprüft (vgl. DATACOM Buchverlag: Integrität 2020).

mittlerweile so weit verbreitet, dass ein hoher Grad an Anonymität durch Masse[10] gewährleistet werden kann. Viele benutzen das Tor-Netzwerk so regelmäßig, dass es schlichtweg fast unmöglich ist, ganz kleine Nutzergruppen oder sogar einzelne Nutzer anhand des Nutzerverhaltens zu identifizieren. Zusätzlich den Erfolg einer solchen Attacke erschweren können die Nutzer selbst, indem sie das Tor-Netzwerk unauffällig verwenden, beispielsweise durch eine möglichst wechselnde Onlinezeit und eine immer unterschiedliche Sitzungsdauer.

Ein Angriff, der dagegen auch bezogen auf das Tor-Netzwerk mehr Erfolg zu versprechen scheint, ist die Timing-Attacke. Eines der Ziele Tors ist es, möglichst geringe Latenzen und keinen Delay, also im allgemeinen keine Verbindungsverzögerungen, zu verursachen. Daraus folgt, dass der zeitliche Abstand, mit welchem ein durch das Tor-Netzwerk geschickte Datenpaket die einzelnen Zwischensysteme und auch den Empfänger erreicht, besonders gering sein soll. Dieses Bestreben Tors wird bei diesem Angriff negativ verwendet. Es wird nämlich versucht, den zeitlichen Abstand zwischen dem Ausgang von Daten eines Zwischensystems und dem Empfangen von Daten eines darauffolgenden Zwischensystems einem Datenstrom zuzuordnen und ihn so auf seinem Weg durch das Netzwerk zu beobachten. Es wird klar, dass auch eine solche Attacke im Tor-Netzwerk aufgrund des hohen Grades an Anonymität durch Masse schwer umzusetzen ist. Die Zuordnung eines Datenstroms zu einem einzigen Nutzer des Netzwerks ist aufgrund des extremen Nutzeraufkommens bei Tor generell schwierig, vor allem, wenn es sich um einen so minimalen Zeitabstand handelt, den es bei einem solchen Angriff zu beobachten gilt.

3.1.2 Tor-spezifische Angriffe

Auch wenn sich Tor gegen viele allgemein bekannte Netzwerkangriffe geschickt absichert, sind durch die Architektur des Tor-Netzwerks andere, teils auch neuartige Tor-spezifische Angriffe möglich, die im Folgenden betrachtet werden sollen.

Eine Man-in-the-Middle-Attacke ist möglich, sofern der Exit-Onion-Router eines Streams kompromittiert ist. Beim Exit-Onion-Router liegen die über das Tor-Netzwerk übertragenen Daten, sofern sie nicht noch anderweitig zusätzlich verschlüsselt wurden, nämlich unverschlüsselt vor, weil der Exit-Onion-Router, fungierend als Schnittstelle zwischen Netzwerk und Internet, die letzte Schale der Onion und somit die letzte Verschlüsselungsschicht des Datenpaketes abschälen bzw. entschlüsseln muss. Ein kompromittierter Onion-Router, welcher vom Onion-Proxy als Exit-Onion-Router eines Streams gewählt wurde, erhält so die Möglichkeit, sich als tatsächlicher Operator zwischen den Sender und den Empfänger zu schalten und die übertragenen Daten einfach nach Informationen zu durchsuchen oder sogar die Rolle des eigentlichen Kommunikationspartners zu übernehmen (vgl. Strefler 2006, S.42). Eine Maßnahme gegen eine solche Man-in-the-Middle-Attacke durch einen kompromittierten Exit-Onion-Router ist die bei Tor implementierte Ende-zu-Ende-Integritätsüberprüfung, bei welcher die Unversehrtheit der über das Netzwerk übertragenen Pakete geprüft wird (siehe Kapitel 3.1.1).

[10] **Anonymität durch Masse:** Der einzelne Nutzer kann in der Masse an Nutzern untertauchen. Ein einzelnes Datenpaket wird in der Masse an Paketen, die zeitähnlich übertragen werden, nicht einfach bemerkt. Dadurch entsteht bereits ein zumindest grundlegender Schutz.

Theoretisch sind auch Angriffe auf die Directory-Server des Tor-Netzwerks möglich. Wenn ein Angreifer irgendwie die Möglichkeit erhält, die Kontrolle über die Mehrheit an aktiven Directory-Servern des Netzwerks zu erlangen, kann er das komplette Netzwerk lahmlegen. Denn der Angreifer könnte verhindern, dass die von ihm kompromittierten Directory-Server weiterhin von den andern Directory-Servern erhaltene Netzwerkberichte signieren. Dadurch, dass dann keine Mehrheit an Directory-Servern die Berichte der ehrlichen Directory-Server signieren würde, käme es wegen baldiger Ungültigkeit der älteren, noch von einer Mehrheit an Directory-Servern signierten Netzwerkberichte zu einem Stillstand des Netzwerks, weil sich die Directory-Server nicht auf einen neuen, gültigen Netzwerkbericht einigen könnten. Darüber hinaus könnte ein Angreifer das Netzwerk auch spalten (Partition-Attacke), indem er von ihm kompromittierte Onion-Router installiert und sie von der von ihm kontrollierten Mehrheit an Directory-Servern zu einem neuen Netzwerkbericht signieren lässt (vgl. Strefler 2006, S.42 f.). Das wäre quasi der Supergau, weil der Angreifer so ein Szenario schaffen könnte, dass alle Onion-Router in dem „neu-signierten“ Netzwerk von ihm kompromittiert sind (siehe Kapitel 3.1). Allerdings ist ein solches Szenario höchst unrealistisch, weil es sich bei den Directory-Servern um höchstvertrauenswürdige Server handelt, die zum großen Teil von den Tor-Entwicklern selbst verwaltet werden. Daher ist dieser Angriffsmöglichkeit, trotz der immensen Kraft, die sie theoretisch hätte, nicht die höchste Beachtung zu schenken.

Die von den Tor-Entwicklern implementierten Hidden-Services können, obwohl sie sich rein netzwerkintern bewegen, auch Opfer von Angriffen werden. Ein anonymer Server bietet eine begrenzte Anzahl an Introduction-Points an, damit die Nutzer eine Verbindung zu ihm aufbauen können (siehe Kapitel 2.4). Die Introduction-Points eines anonymen Servers sind, damit ein Verbindungsaufbau möglich wird, logischerweise über die Directory-Server gelistet, wodurch sie quasi öffentlich bekannt sind. Weil sie als Introduction-Points in Verbindung mit den Hidden Services stehen, sind diese für Angreifer besonders interessant und aufgrund ihrer begrenzten Anzahl entsteht dadurch eine zusätzliche Gefahr. Denn wenn alle Introduction-Points eines anonymen Servers von einem Angreifer kompromittiert werden oder durch eine Flooding-Attacke stillgelegt werden, kann der versteckte Dienst nicht mehr erreicht werden. Um diesem Problem entgegenzuwirken, wurde die Hidden Service-Architektur kürzlich um eine Serverschicht ergänzt. Ein anonymer Server kann nun mehrere Valet-Tickets kreieren, welche über einen abgeschotteten Kommunikationskanal (Außenband-Signalisierung[11]) direkt an den interessierten Nutzer gesendet werden, wenn dieser über seinen Onion-Proxy einen Directory-Server mit der .onion-Domain anfragt. Dieses Ticket enthält dann Kontaktinformationen zu einem von im Gegensatz zu Introduction-Points viel zahlreicher vorhandenen Valet-Nodes. Der Nutzer kann damit über seinen Onion-Proxy einen Stream zu dem Valet-Node aufbauen, welcher gleichzeitig übrigens ganz normaler Onion-Router ist. Der Valet-Node baut daraufhin einen Stream zu einem Introduction-Point des anonymen Servers auf,

[11] **Außenband-Signalisierung:** Es gibt spezielle Kanäle, die außerhalb des für die Nutzdaten vorgesehenen Übertragungsbereichs liegen und lediglich für die Signalisierung von Schlüsselinformationen benutzt werden. Sie können von Außenstehenden nicht eingesehen werden (vgl. DATACOM Buchverlag: Außenband-Signalisierung 2020).

sodass es den Introduction-Points ermöglicht wird, anonym zu bleiben (siehe *Abbildung 7*). Sie müssen somit nicht länger von den Directory-Servern gelistet werden.

3.2. Weiterführende Gefahren für den Nutzer

Das Tor-Netzwerk versucht, jedem Nutzer absolute Anonymität zu gewährleisten. Doch neben Angriffen auf das Netzwerk gibt es noch weitere Gefahren für den Nutzer. Die Nichtbeachtung dieser Gefahren kann dazu führen, dass der Nutzer sich selbst deanonymisiert oder einen Angriff auf ihn selbst stark begünstigt.

Eine große Gefahr für den Nutzer stellt die Benutzung von Diensten Dritter, welche nicht ausdrücklich für die Nutzung über das Tor-Netzwerk entwickelt oder optimiert wurden, über Clients, wie dem „Tor Browser“, dar. Beispielsweise ist das beliebte Programm Bit-Torrent, mit welchem sich Dateien anonym übertragen lassen, nicht für die Nutzung über Tor optimiert. Es basiert zwar auf einem dem Onion Routing ähnlichen Prinzip, denn es baut für jede Dateiübertragung ebenfalls eine verschlüsselte Übertragungskette auf. Allerdings könnte Bit-Torrent mit dieser eigenen Kette den vom Tor-Netzwerk bereitgestellten Stream nicht beachten und somit die Struktur des Tor-Netzwerks bei einer Dateiübertragung aushebeln. Das Nutzen von Bitcoin über Tor stellt eine ähnliche Gefahr dar. Nach einer Recherche der Universität zu Luxemburg ergab das Kombinieren der beiden Anwendungen die Möglichkeit einer Man-in-the-Middle-Attacke. Dabei konnte volle Kontrolle über den Informationsfluss des Nutzers, der Bitcoin über das Tor-Netzwerk verwendet, erlangt werden (vgl. Çalışkan et al. 2015, S.14). Ein weiteres Beispiel ist die Nutzung von nicht-anonymen Plug-ins über das Tor-Netzwerk, wie beispielsweise „Flash Player“ oder „Document-Viewer“. Diese können unbemerkt Cookies setzen, einen Tor-Netzwerk-Stream damit unbewusst untergraben und so schlussendlich die tatsächliche IP-Adresse des Nutzers aufdecken. Außerdem können solche Plug-ins eine direkte Verbindung zum Zielserver suchen und den von Tor bereitgestellten Stream nicht beachten. So könnte die IP-Adresse des Nutzers ebenfalls sichtbar werden (vgl. Strefler 2006, S.33 f.).

Die Benutzung von HTTP anstelle von HTTPS stellt eine weitere Gefahr für den Nutzer dar. Beim Exit-Onion-Router eines Streams liegen die übertragenen Daten, sofern nicht zuvor zusätzlich verschlüsselt, unverschlüsselt vor (siehe Kapitel 3.1.2). Dadurch ist dort ein hohes Angriffspotential gegeben. Dieses Problem kann HTTPS lösen, weil mit HTTPS eine zusätzliche Verschlüsselungsschicht um die übertragenen Daten gepackt wird. Der Exit-Onion-Router hat dann nicht mehr die Möglichkeit, die Daten theoretisch einfach auszuspähen.

Eine weitere Gefahr stellen für die Nutzer des Tor-Netzwerks natürlich auch software-technische Bugs dar, die entweder von der Netzwerkarchitektur Tors direkt oder von Clients, mit denen auf das Tor-Netzwerk zugegriffen werden kann, ausgehen. In der Vergangenheit gab es beispielsweise den sogenannten Heartbleed-Bug. Dabei konnte eine Schwachstelle in bestimmten schadhaften Versionen der vom „Tor-Browser“ implementierten Open-SSL-Verschlüsselungssoftware zu einem Angriff auf Nutzer, welche zum Eintritt in das Tor-Netzwerk den „Tor-Browser“ mit jenen bestimmten Open-SSL-Versionen nutzten, verwendet werden. Sensible Daten wie Passwörter oder Kreditkarteninformationen der betroffenen Nutzer konnten offengelegt werden, was zur Deanonymisierung von vielen Tor-Benutzern führte. Der Bug

wurde schließlich durch ein Update des „Tor-Browsers“ behoben (vgl. Tor Projekt: [tor announce] 2019).

Logischerweise stellt auch der Ausfall des gesamten Tor-Netzwerks eine Gefahr für den Nutzer dar. Es wäre theoretisch möglich, dass beispielsweise alle Directory-Server gleichzeitig ausfallen oder aus irgendwelchen Gründen keine Onion-Router erreichbar sind. Aufgrund der virtuellen Größe und der geographischen Streuung der einzelnen Onion-Router und Directory-Server ist ein gänzlicher Ausfall des Tor-Netzwerks jedoch annähernd ausgeschlossen.

3.3. Tor-Community als Sicherheitskriterium

Darum, dass es gar nicht erst zu tatsächlichen Angriffen auf das Tor-Netzwerk kommen kann und das Tor-Netzwerk weiterverbreitet und weiterentwickelt wird, kümmert sich die Tor-Community. Die Tor-Community formiert sich aus engagierten und interessierten Unterstützern des Open-Source-Projekts Tor.

Von der Tor-Community wird Recherchearbeit betrieben, um das Tor-Ökosystem aufrechtzuerhalten. An solchen Recherchearbeiten haben neben der Community auch die eigentlichen Tor-Entwickler, sowie andere Forscher und Wissenschaftler teil. So werden beispielsweise großangelegte Angriffssimulationen in von Tor eigens geförderten Forschungsprojekten an Netzwerk-Prototypen durchgeführt, um eventuelle Schwachstellen der Tor-Clients oder auch der Netzwerkarchitektur bereits im Voraus zu erkennen. Auch werden Recherchen zum Zweck der Effizienzsteigerung des Tor-Netzwerks durchgeführt (vgl. Tor Projekt Research 2019). So wurde unter anderem der im klassischen Onion-Routing implementierte Zwischen-Padding-Ansatz[12] nicht in der Architektur Tors aufgenommen, weil sich nach gründlichen Recherchen ergab, dass der dadurch höhere Ressourcenverbrauch den minimal gewonnenen Grad an zusätzlicher Anonymität nicht rechtfertige (vgl. Dingledine et al. 2004, S.1). Die Tor-Community hat mit ihrer aktiv unterstützenden Arbeit großen Anteil daran, dass Tor als Open-Source-Projekt seine Maxime von allumfassender Anonymität kombiniert mit hoher Nutzerfreundlichkeit im Internet weiterverfolgen kann. Es soll erreicht werden, dass immer mehr Menschen anfangen, das Tor-Netzwerk regelmäßig zu nutzen. Denn eine höhere Anzahl an Nutzern erhöht langfristig auch den Grad an Anonymität.

4. Fazit

Ziel dieser Facharbeit war es, theoretisch zu untersuchen, ob das Tor-Netzwerk seinen Nutzern tatsächlich absolute Anonymität bieten kann. Beim Betrachten der Funktionsweise des Tor-Netzwerks wurde in diesem Zusammenhang deutlich, dass die Tor-Architektur durch das Onion Routing-Prinzip und eigene Erweiterungen des Ansatzes einen enormen Schutz der Identität bietet. Allerdings stellte sich im Rahmen der anknüpfenden Sicherheitsanalyse des Tor-Netzwerks heraus, dass auch das Tor-Netzwerk trotz zahlreicher in die Architektur

[12] **Zwischen-Padding-Ansatz:** Auffüllen der schwankenden Datenrate auf einer direkten Verbindung zweier Onion-Router auf ein stetig konstantes Level mithilfe von zwischen diesen hin- und hergeschicktem dummy-Traffic. Dadurch sollte das Beobachten und Identifizieren von Nutzern anhand schwankender Datenraten (u.a. wegen geringen Nutzerzahlen!) unmöglich gemacht werden.

integrierter Maßnahmen zur Sicherung des Netzwerkes angegriffen werden und die Anonymität der Nutzer dadurch gefährdet werden kann.

Die Annahme einer absoluten Anonymität ist rückblickend betrachtet eine utopische Vorstellung, weil sich jedes noch so sichere Informationssystem im Internet teils unberechenbaren Gefahren ausgesetzt sieht. Sogar die Nutzer selbst können ihre Anonymität durch unvorsichtiges Nutzen des Tor-Netzwerks in Gefahr bringen. Es stellt sich also auf zunächst tieferer Ebene die Frage, ob Tor alles in seiner Macht Stehende tut, um bestmögliche Anonymität zu schaffen.

Die Antwort auf diese Frage ist eindeutig: Nein. Denn die Tor-Entwickler formulierten bereits bei der Erstveröffentlichung Tors das oberste Ziel des damals neuen Anonymisierungsdienstes. Tor sollte ein Anonymisierungsnetzwerk sein, welches den Nutzern ein größtmögliches Maß an Anonymität im Internet bietet, jedoch gepaart mit einfacher Nutzbarkeit der Software[13]. Doch genau darin liegt der Knackpunkt, wenn man die Fragestellung dieser Facharbeit betrachtet. Tor geht es zwar zum größten Teil um Anonymität im Internet, aber auch um andere Attribute, wie beispielweise Latenzminimierung, Performance- und Effizienzsteigerung und Reichweitenexpansion. Doch viele Maßnahmen, die von den Tor-Entwicklern zugunsten der anderen Attribute getroffen werden, gehen zu Lasten der Anonymität. Als Beispiel lässt sich das Nicht-Integrieren des Zwischen-Padding-Ansatzes nennen.

Wegen dieser Maxime ist Tor heutzutage aber auch so populär. Das Projekt wird von zahlreichen Unterstützern getragen, die dazu beitragen, dass Tor stets weiterentwickelt wird. Der derzeitige Aufwärtstrend bezüglich der Größe des Netzwerks lässt vermuten, dass das Tor-Netzwerk auch zukünftig seinen Nutzern enorme Anonymität bieten kann.

Es wird insgesamt deutlich, dass Tor stets auf einen begründeten Kompromiss zwischen Anonymität und Nutzbarkeit bedacht ist. Eine absolute Anonymität im Internet kann das Tor-Netzwerk seinen Nutzern nicht bieten, wohl aber die bestmögliche, die derzeit weltweit zu erlangen ist.

[13] Einer der Gründer des Tor-Projekts, Roger Digledine, formulierte derartige Ansätze sinngemäß in verschiedenen seiner schriftlichen Ausarbeitungen (beispielweise in Dingledine et al. 2004).

5. Anhang

5.1. Literaturverzeichnis

Çalışkan, Emin / Minárik, Tomáš / Osula, Anna-Maria: Technical and Legal Overview of the Tor Anonymity Network, Tallinn 2015

DATACOM Buchverlag: Außenband-Signalisierung :: out-of-band signalling. https://www.it-wissen.info/Aussenband-Signalisierung-out-of-band-signalling.html (Zugriff: 28.01.2020)

DATACOM Buchverlag: Bidirektional (bidirectional) :: BiDi. https://www.itwissen.info/Bidirektional-bidirectional-BiDi.html (Zugriff: 18.01.2020)

DATACOM Buchverlag: EzE (Ende-zu-Ende-Verbindung) :: E2E (end to end). https://www.it-wissen.info/Ende-zu-Ende-Verbindung-end-to-end-E2E.html (Zugriff: 18.01.2020)

DATACOM Buchverlag: Integrität :: integrity. https://www.itwissen.info/Integritaet-integrity.html (Zugriff: 25.01.2020)

DATACOM Buchverlag: Overlay-Netz :: overlay network. https://www.itwissen.info/Overlay-Netz-overlay-network.html (Zugriff: 18.01.2020)

DATACOM Buchverlag: PzP (Punkt-zu-Punkt-Verbindung) :: P2P (point to point). https://www.itwissen.info/Punkt-zu-Punkt-Verbindung-PzP-point-to-point-P2P.html (Zugriff: 28.01.2020)

DATACOM Buchverlag: Web-Proxy :: web-proxy. https://www.itwissen.info/Web-Proxy-web-proxy.html (Zugriff: 05.01.2020)

Dingledine, Roger / Mathewson, Nick / Syverson, Paul: Tor: The Second-Generation Onion Router, San Diego 2004

Federrath, Hannes / Martius, Kai: Anonymität und Authentizität im World Wide Web, in: ITG-Fachtagung „Internet – frischer Wind in der Telekommunikation“, Stuttgart 1998, S. 91–96

Gabler Wirtschaftslexikon: Public-Key-Verfahren. Definition. wirtschaftslexikon.gabler.de/definition/public-key-verfahren-46898 (Zugriff: 03.01.2020)

Grassegger, Hannes / Krogerus, Mikael: «Ich habe nur gezeigt, dass es die Bombe gibt», in: Tages-Anzeiger vom 20. März 2018

Konigorski, Marco: Techniken zur Sicherung von Anonymität im Internet, in: Gamer, Thomas / Sorge, Christoph u.a. (Hrsg.): Datenschutz in Kommunikationsnetzen. Seminar SS06, Karlsruhe 2006, S. 3–15

Michaelis, Kai H.: Eine Analyse des Tor-Protokolls, Bochum 2012

Miller, Konrad: Onion Routing, in: Gamer, Thomas / Sorge, Christoph u.a. (Hrsg.): Datenschutz in Kommunikationsnetzen. Seminar SS06, Karlsruhe 2006, S. 17–24 u. S. 27–31

Rouse, Margaret: Was ist Payload? https://www.computerweekly.com/de/definition/Payload (Zugriff: 19.01.2020)

Schneier, Bruce: Attacking Tor: how the NSA targets users’ online anonymity, in: The Guardian vom 04. Oktober 2013

Spektrum der Wissenschaft: symmetrisches Verschlüsselungsverfahren. https://www.spektrum.de/lexikon/mathematik/symmetrisches-verschluesselungsverfahren/11416%20-%20Definition%20symmetrisches%20Verschlüsselungsverfahren (Zugriff: 03.01.2020)

Strefler, Mario: Tor und JAP: Umsetzung von Anonymitätstechniken, in: Gamer, Thomas / Sorge, Christoph u.a. (Hrsg.): Datenschutz in Kommunikationsnetzen. Seminar SS06, Karlsruhe 2006, S. 33–43

Tor Project: Research. research.torproject.org (Zugriff: 28.12.2019)

Tor Project: [tor-announce] Tor security advisory: Old Tor Browser Bundles vulnerable. lists.torproject.org/pipermail/tor-announce/2013-August/000089.html (Zugriff: 31.12.2019)

Vogel, Martin: Proxy – Was bedeutet das? https://martinvogel.de/lexikon/proxy.html (Zugriff: 04.01.2020)

5.2. Abbildungsverzeichnis

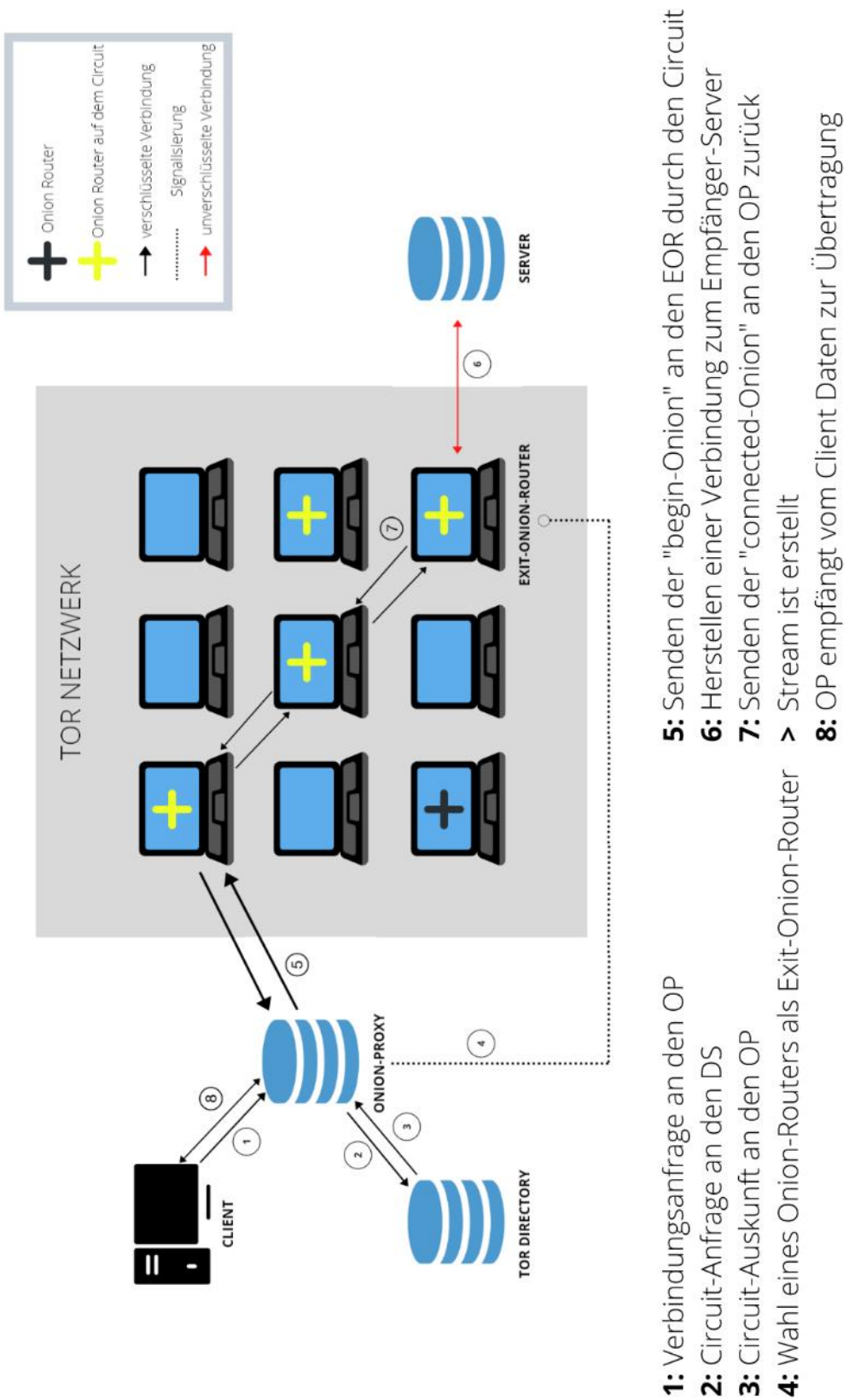


Abbildung 1: Verbindungsaufbau und Streaminitiierung beim Tor-Netzwerk (Elias Mitropoulos).

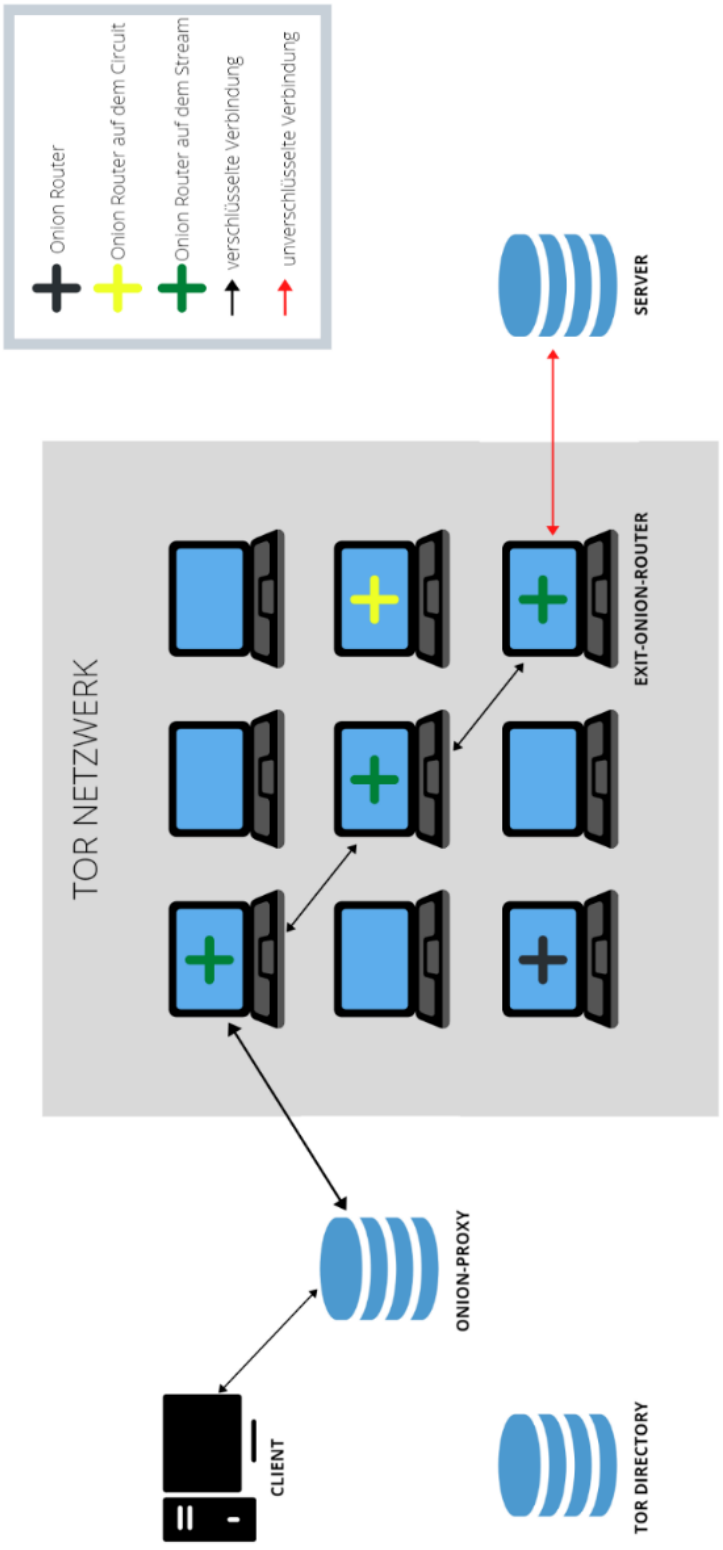
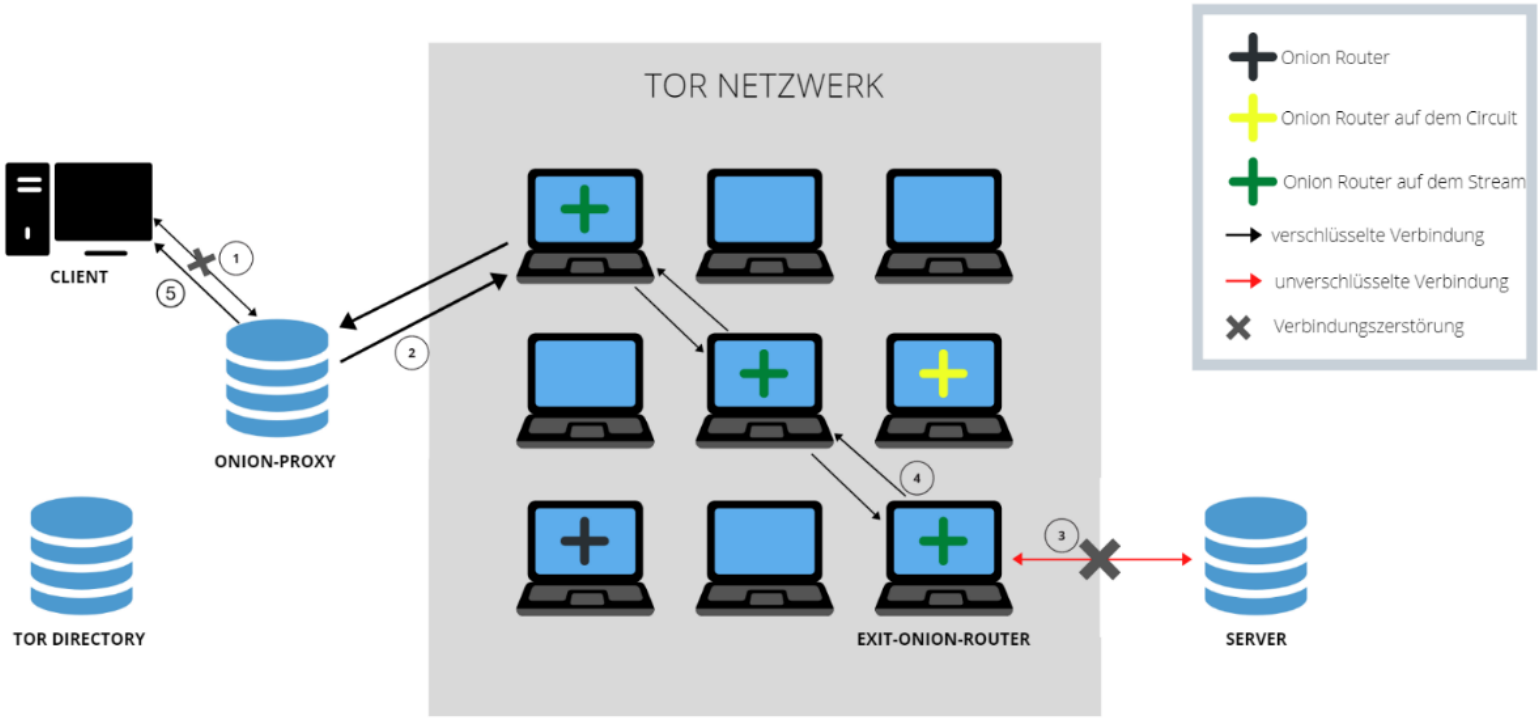


Abbildung 2: Bidirektionale Kommunikation beim Tor-Netzwerk (Elias Mitropoulos).



1: Verbindungszerstörung zum Client hin
2: Senden der "end-Onion" an den EOR
3: Verbindungszerstörung zum Server hin
4: Senden der "reply-end-Onion" an den OP zurück
> Stream wurde aus dem Verkehr gezogen
5: Benachrichtigung an den Client

Abbildung 3: Verbindungstermination beim Tor-Netzwerk (Elias Mitropoulos).

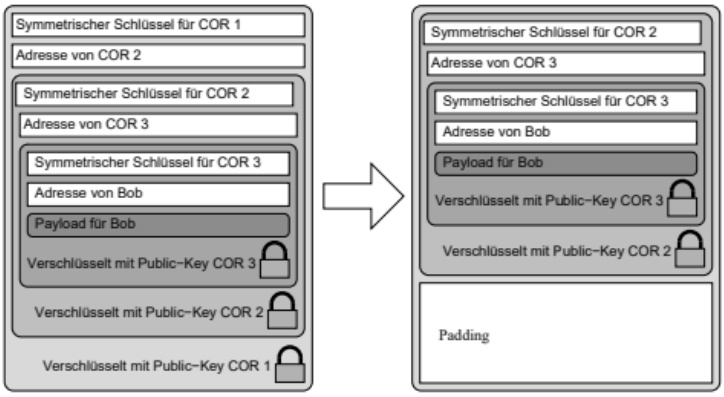


Abbildung 4: Abschälen einer Zwiebelschale (vgl. Miller 2006, S.22).

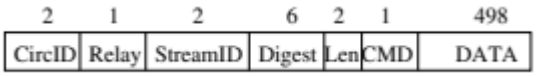


Abbildung 5: Bestandteile einer Onion-Schale (vgl. Dingledine et al. 2004, S.5).

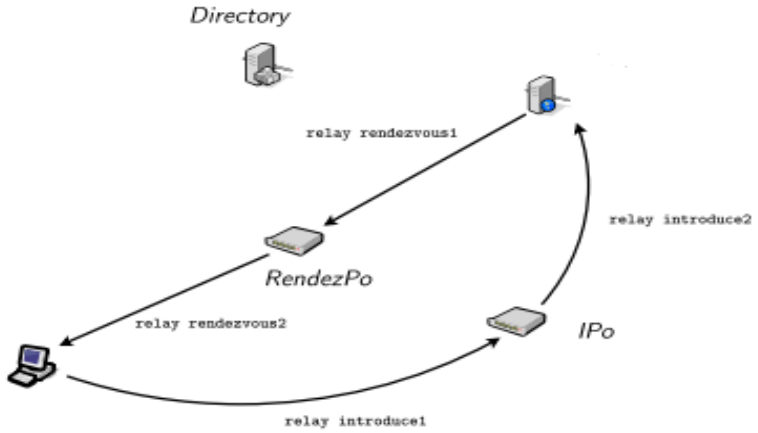


Abbildung 6: Etablierung eines Kanals mit einem Hidden Service (vgl. Michaelis 2012, S.16).

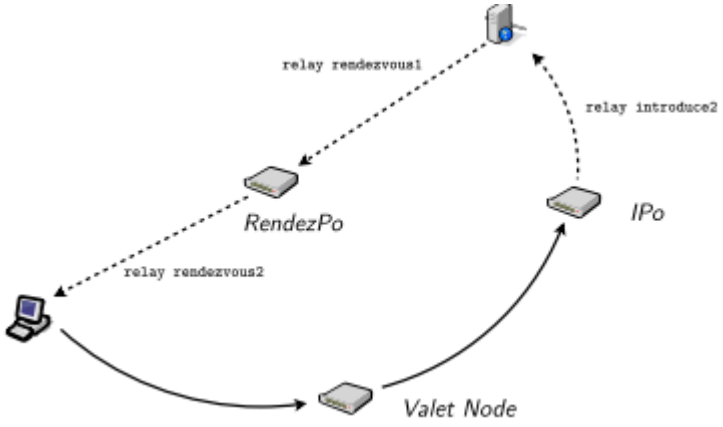


Abbildung 7: Etablierung eines Kanals mit einem Hidden Service über Valet-Nodes (vgl. Michaelis 2012, S.17) [Anmerkung: Die Änderungen sind durch dicke Pfeile hervorgehoben].

---

6. Selbstständigkeitserklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und ohne fremde Hilfe verfasst und keine anderen Hilfsmittel als angegeben verwendet habe. Insbesondere versichere ich, dass ich alle wörtlichen und sinngemäßen Übernahmen aus anderen Werken als solche kenntlich gemacht habe.

Heiligenhaus, 25. März 2020
……
(Ort, Datum)

……
(Unterschrift)