Dezentralisierte asymmetrische Verschlüsselung über Tor

Die Lösung für sicheres Messaging?

Hendrik Lind

Facharbeit

Windthorst-Gymnasium Meppen

Seminarfach Informatik

4. Februar 2024

Inhaltsverzeichnis

1 Einleitung

Russland, China, Iran. In all diesen totalitären Staaten herrscht eine starke Zensur [vgl. Am23]. Rund 1,7 Milliarden Menschen sind allein nur in diesen drei Staaten von der Einschränkung der Meinungsfreiheit betroffen [vgl. Un22]. Wie können Bürger dieser Staaten ihre Meinung also verbreiten und andere Staaten auf staatskritische Probleme aufmerksam machen ohne sich selber in Gefahr zu bringen?

Bei herkömmlichen Messengern, wie WhatsApp, Signal und co., braucht die Außenwelt die Telefonnummern der im totalitären Staat wohnenden Bürgern und Reportern, um diese zu kontaktieren. Allerdings könnte ein totalitärer Staat, sich als Empfänger ausgeben, sodass Bürger/Reporter ihre private Nummer an den Staat überreichen und dieser somit jene Nummer rückverfolgen kann [vgl. Fä23]. Und genau hier liegt das Problem: Bürger und Reporter können nicht durch alltägliche Messenger mit der Außenwelt kommunizieren, da der Staat deren Nummer zurückverfolgen kann und somit weiter die Meinungsfreiheit einschränkt und unterbindet [vgl. ebd.].

Durch die zentrale Infrastruktur, welche die meisten Messenger, wie zum Beispiel WhatsApp und Signal verwenden, ist es für totalitäre Staaten, wie China, möglich, die IP-Adressen jener Server zu blockieren und somit für Bürger und Reporter unzugänglich zu machen [vgl. Wu+23; 23c].

Ein dezentralisierter Messenger, welcher Ende-zu-Ende verschlüsselt ist und über das Tor-Netzwerk kommuniziert, könnte bei diesen Problemen eine Lösung sein. Die Frage, ob ein solcher Messenger die Lösung für Bürger eines totalitären Staates ist, soll in dieser Arbeit geklärt werden.

Um diese Frage beantworten zu können, beschäftigt sich diese Arbeit in dem zweiten Kapitel mit der asymmetrischen Verschlüsselung, welche benötigt wird um die Ende-zu-Ende-Verschlüsselung (E2EE) umzusetzen und die Definition der E2EE, sowie der Sicherheitsbetrachtung der asymmetrischen Verschlüsselung [vgl. LB21]. Eine mögliche Lösung, um eine Anonymität über das Internet zu gewährleisten wird in Kapitel drei vorgeschlagen, wobei das Tor-Netzwerk eine wichtige Rolle spielt. Diese Arbeit befasst sich im vierten Kapitel mit einer Dezentralisierung der Infrastruktur, um eine weitere Sicherheitsebene zu schaffen. Zuletzt werden im fünften Kapitel die Vor- und Nachteile eines solchen Messengers betrachtet, im sechsten

Kapitel wird eine mögliche Umsetzung des Messengers beschrieben und im siebten Kapitel wird ein Fazit gezogen.

2 Asymmetrische Verschlüsselung

Um einen sicheren Nachrichtenaustausch zu gewährleisten, wird in dieser Arbeit die E2EE implementiert. Bei der E2EE wird von dem Sender die Nachricht, bevor sie an den Empfänger geschickt wird, verschlüsselt [vgl. Gr14]. Dazwischenliegende Akteure, wie zum Beispiel Server oder mögliche Angreifer, können demzufolge die Nachricht nicht lesen [vgl. ebd.]. Nur der Empfänger der Nachricht kann diese auch entschlüsseln. Als Ent- und Verschlüsselungsverfahren der Nachrichten wird die asymmetrische Verschlüsselung verwendet [vgl. ebd.]. Für diese Arbeit werde ich die gängigste asymmetrische Verschlüsselung, das RSA-Verfahren, verwenden.

2.1 Grundlagen

Grundsätzlich gibt es bei der asymmetrischen Verschlüsselung ein Schlüsselpaar (Keypair), welches aus einem privaten Schlüssel (private key) und einem öffentlichen Schlüssel (public key) besteht [vgl. BSW15b]. Diese beiden Schlüssel hängen mathematisch zusammen, sodass der öffentliche Schlüssel Nachrichten nur verschlüsseln und nicht entschlüsseln kann [vgl. ebd.]. Nur der zum Schlüsselpaar dazugehörige private Schlüssel ist in der Lage, die verschlüsselte Nachricht wieder zu entschlüsseln (siehe Abb. 1) [vgl. ebd.].

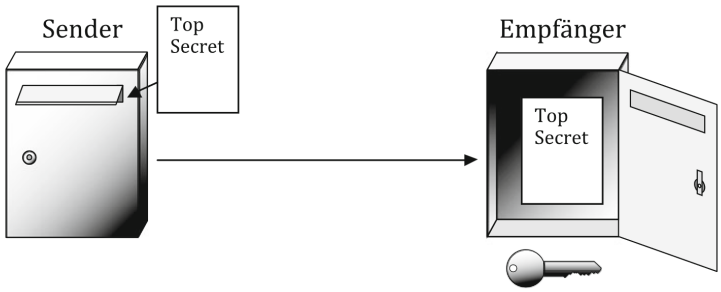

Abbildung 1: Jeder Sender kann mit dem öffentlichen Schlüssel die Nachricht „verschlüsseln“ (also in den Briefkasten eine Nachricht werfen), aber nur der Empfänger kann den Briefkasten mit seinem privaten Schlüssel öffnen [vgl. BSW15a]

2.2 Mathematische Betrachtung

Alle Variablen der folgenden Berechnungen liegen im Bereich $\mathbb{N}$ [vgl. Is16].

Für die Generierung des Schlüsselpaares benötigen wir zuerst zwei große zufällige Primzahlen, P und Q [vgl. ebd.]. Daraus ergibt sich $n = P*Q$, wobei $P \neq Q$, sodass P bzw. Q nicht durch $\sqrt{n}$ ermittelt werden kann [vgl. ebd.]. Der private Schlüssel besteht aus den Komponenten $\{n, d\}$ währenddessen der öffentliche Schlüssel aus $\{n, e\}$ besteht [vgl. Wa+13].

2.2.1 Eulersche Phi-Funktion

Die Eulersche Phi-Funktion spielt eine wichtige Rolle in dem RSA-Verfahren [vgl. Tu08]. Grundsätzlich gibt $\phi(x)$ an, wie viele positive teilerfremde Zahlen bis x existieren (wo also der größte gemeinsamer Teiler (gcd) 1 ist) [vgl. Ta13]. Somit ergibt $\phi(6) = 2$ oder bei einer Primzahl $\phi(7) = 7 - 1 = 6$ somit $\phi(x) = x - 1$, wenn x eine Primzahl ist, da jede Zahl kleiner als x teilerfremd sein muss [vgl. Tu08].

$$\begin{aligned}
&\phi(n) = \phi(P \cdot Q) \\
&\phi(n) = \phi(P) \cdot \phi(Q) \\
&\phi(P) = P - 1 \qquad\qquad \phi(Q) = Q - 1 \\
&\phi(n) = (P - 1) \cdot (Q - 1)
\end{aligned}$$

2.2.2 Generierung des Schlüsselpaares

Sowohl der private als auch der öffentliche Schlüssel besteht unter anderem aus folgender Komponente: $n = P \cdot Q$ [vgl. Ta96]. Für den öffentlichen Schlüssel benötigen wir die Komponente e, welche zur Verschlüsselung einer Nachricht benötigt wird [vgl. ebd.]. e ist hierbei eine zufällige Zahl, bei welcher folgende Bedingungen gelten [vgl. ebd.]:

$$e = \begin{cases} 1 < e < \phi(n) \\ \text{gcd}(e, \phi(n)) = 1 \\ e \text{ kein Teiler von } \phi(n) \end{cases}$$

Mit der errechneten Komponente e, welche Nachrichten verschlüsselt, kann der öffentliche Schlüssel nun an den Sender übermittelt werden.

Um den privaten Schlüssel zu berechnen benötigen wir die Komponente d, wel-

che zur Entschlüsselung verwendet wird [vgl. MS13].

$$\phi(n) = (P-1)(Q-1)$$
$$e \cdot d = 1 \mod \phi(n)$$

2.3 Sicherheit

Das RSA-Verfahren macht sich die Trapdoor-Einwegfunktion zu Nutze [vgl. Kr16]. Das bedeutet, dass eine Funktion mit modernen Computern unmöglich ist, zu invertieren, wenn eine Komponente fehlt (in diesem Fall d). Wenn diese jedoch gegeben ist, ist die Umkehroperation leicht [vgl. ebd.]. Zu sehen ist dies bei der Ver-/Entschlüsselung von Nachrichten mit dem öffentlichen und privaten Schlüssel [vgl. ebd.].

$c = m^e \mod n$ Verschlüsselung zu c mit m als Nachricht

$m = c^d \mod n$ Umkehroperation (Entschlüsslung) von c zu m

Um die verschlüsselte Nachricht c zu entschlüsseln, bräuchte ein möglicher Angreifer die Komponente des privaten Schlüssels d. Diese ist allerdings schwer zu berechnen, da, wie schon vorher bereits gezeigt, dafür $\phi(n)$ benötigt wird. Auch $\phi(n)$ ist rechenaufwendig, da dafür eine Primfaktorzerlegung von n benötigt wird [vgl. Ta13]. Wichtig hierbei ist aber, dass die Länge von n (Schlüssellänge) mindestens 3000 Bit betragen sollte, da sonst die Primfaktorzerlegung von n mit modernen Computern möglich sein könnte [vgl. BSI23].

2.4 Vergleich zur symmetrischen Verschlüsselung

Im Gegensatz zu der asymmetrischen Verschlüsselung, bei welcher sowohl der privaten als auch der öffentlichen Schlüssel verwendet werden, wird bei der symmetrischen Verschlüsselung nur ein Schlüssel verwendet, um Nachrichten zu verschlüsseln[vgl. IBM21]. Mit dem selben Schlüssel können Nachrichten auch wieder entschlüsselt werden [vgl. ebd.]. Im Vergleich zu der symmetrischen Verschlüsselung ist die asymmetrische Verschlüsselung langsamer, jedoch muss kein Schlüssel zur Entschlüsselung der Nachricht übermittelt werden [vgl. HK20].

3 Anonymität mit dem Tor-Netzwerk

Der zweite zentrale Aspekt dieser Arbeit ist die Anonymität über das Internet. Bei normalen Routing, wie wir es tagtäglich nutzen, kommuniziert der Client direkt mit dem Zielserver. Der Zielserver kann hierbei die IP-Adresse des Clients sehen, somit ist eine Rückverfolgung möglich [vgl. Fä23; El16]. Und genau bei diesem Problem setzt das Tor-Netzwerk an. Das Tor-Netzwerk besteht hierbei aus vielen Nodes , welche eingehende Tor-Verbindungen akzeptieren und weiterverarbeiten. Damit ein Client überhaupt eine Anfrage über das Netzwerk verschicken kann, sucht er sich zunächst einen Pfad durch das Netzwerk, genannt Circuit [vgl. 24e]. Dieser Circuit besteht dabei meist aus drei Nodes und ist für 10 Minuten gültig, bis der Client das Circuit erneuert (also einen neuen Pfad „sucht“) [vgl. 24d]. Der Entry Node , einer Relay Node und einer Exit Node , worüber später Anfragen an die Zielserver geschickt werden können [vgl. ebd.]. Dabei verschlüsselt der Client verschlüsselt die eigentliche Anfrage mehrmals, hüllt sie also in ein mehrere „Schalen“ ein, welche eine Onion bilden, und leitet diese über den Circuit an den Zielserver weiter[vgl. DMS04]. Anfangs wird die Onion von dem Tor-Client an die Entry Node geschickt [vgl. ebd.]. Bei jeder Node , also auch der Entry Node , wird eine Schale der Onion „geschält“ (die Nachricht also einmal entschlüsselt), welche Informationen zu dem nächsten Knotenpunkt enthält, somit schickt jene Node, die Anfrage an den nächsten Knotenpunkt weiter[vgl. Au18]. Sobald die Anfrage bei der Exit Node angekommen ist, entfernt diese die letzte „Schale“ der Anfrage, welche nun vollständig entschlüsselt ist (wenn die eigentliche Anfrage nicht mit HTTPS verschlüsselt wurde), und schickt diese an den Zielserver [vgl. ebd.]. Nur die Entry Node weiß somit die reale IP-Adresse des Clients und nur die letzte Node (Exit Node) weiß, an welchen Zielserver die Anfrage geschickt wurde [vgl. ebd.]. Wichtig hierbei ist allerdings, dass die Entry Node nicht weiß, was der Zielserver der Anfrage ist [vgl. ebd.]. Die Exit Node weiß im Gegenzug die reale IP-Adresse des Clients nicht. Allerdings könnte die Exit Node die IP-Adressen der Zielserver sehen oder möglicherweise Informationen der Nutzer auslesen (wenn die Anfrage nicht mit HTTPS verschlüsselt wurde) [vgl. Ch+11]. Hier herrscht also eine Schwachstelle des Tor-Netzwerkes.

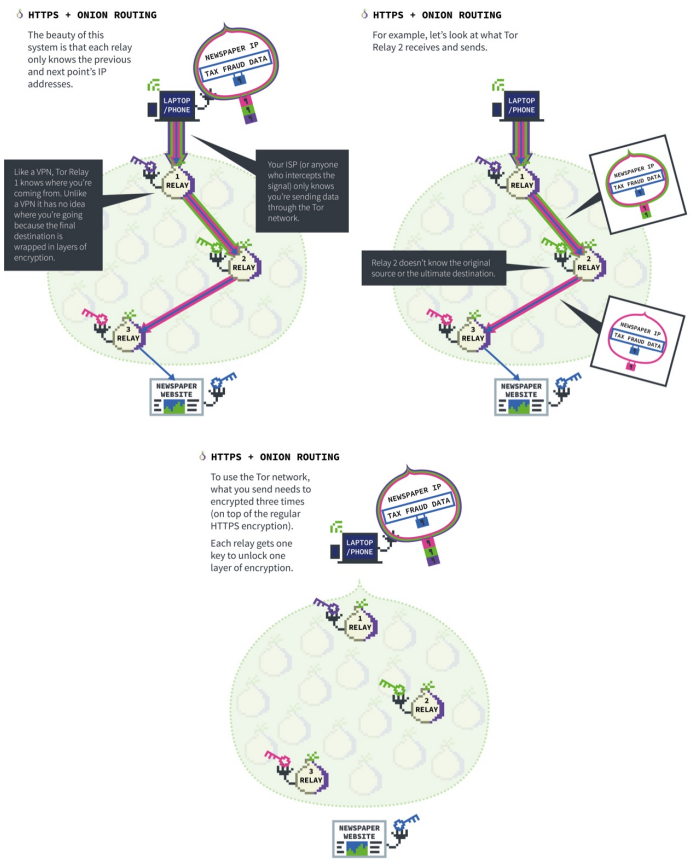

Abbildung 2: Tor Struktur und so [vgl. Tor22]

3.1 Onion Services

Onion Services lösen dieses Problem, da diese nicht auf die Exit Node angewiesen sind und nur innerhalb des Tor-Netzwerkes mit anderen Tor-Clients kommunizieren [vgl. 24h]. Sie nicht über das normale Internet erreichbar, wie die Zielserver im vorherigen Beispiel, sondern nur über das Tor-Netzwerk [vgl. 24g]. Bei dem Onion Service müssen im Kontrast zu normalen öffentlichen Servern keine Ports geöffnet werden, damit ein Client sich mit dem Server verbinden kann, da der Onion Service direkt mit dem Tor-Netzwerk über ausgehende Verbindungen kommuniziert (auch bekannt als NAT punching) und darüber sämtliche Daten geleitet werden [vgl. ebd.]. Für Außenstehende sieht der Onion Service also aus, wie als wäre er ein normaler Tor-Client [vgl. ebd.].

3.1.1 Verbindungsaufbau

Zunächst generiert der Onion Service ein Schlüsselpaar, bestehend aus einem öffentlichen und einem privaten Schlüssel [GeeksOnionService]. Unter anderem wird

nun aus dem öffentlichen Schlüssel die Adresse des Onion Services generiert und endet mit „.onion“ [vgl. ebd.]. Ein Beispiel für eine solche Adresse ist die der Suchmaschine DuckDuckGo: duckduckgogg42xjoc72x3sjasowoarfbgcmvfimaftt6twagswzczad.onion [vgl. 24c]. Der Onion Service sich nun mit dem Tor-Netzwerk wie ein normaler Client über einen Circuit , welcher aus drei Nodes besteht [vgl. 24g]. Der Service sendet eine Anfrage an das letzte Relay im Circuit , sodass es als Introduction Point dient und etabliert eine Langzeitverbindung jenem (der Circuit erneuert sich also nicht alle 10 Minuten wie bei dem Tor-Client) [vgl. ebd.]. Dieser Vorgang wiederholt sich zwei Mal, bis der Onion Service drei Introduction Points auf drei verschiedenen Circuits gefunden hat [vgl. ebd.]. Damit andere Clients den Onion Service erreichen können, erstellt der Onion Service einen Onion Service descriptor, welcher die Adressen der Introduction Points und Authentifizierungsschlüssel enthält, signiert diesen mit seinem privaten Schlüssel und schickt den descriptor an die Directory Authority [vgl. 24a; 24f]. Die Directory Authority ist ein Server, welcher Informationen über das Tor-Netzwerk, wie zum Beispiel die des Onion Services, speichert und verteilt [vgl. 24b]. Im Sinne dieser Arbeit, hat der Onion Service sich nun von Person A mit dem Tor-Netzwerk verbunden, jedoch braucht es noch Person B, welche sich über ihren Tor-Client mit dem Onion Service der Person B verbindet. Damit der Tor-Client von Person A sich allerdings verbinden kann, fragt der Client nun die Directory Authority an den signierten descriptor des Onion Services von Person B an den Client zu schicken [vgl. K 23]. Der Client besitzt nun die Adressen der Introduction Points und die Signatur des descriptors, sodass dieser mit dem öffentlichen Schlüssels der Onion Adresse die Signatur überprüfen kann [vgl. ebd.]. Der Client generiert nun 20 zufällige Bytes (Secret) und schickt diese an eine zufällig ausgewählte Relay Node , welche nun als Rendezvous Point dient [vgl. 23b]. Das Secret wird von dem Client auch an eine von den Introduction Points geschickt, sodass der Onion Service mit dem gleichen Secret eine Verbindung zu dem Rendezvous Point aufbauen kann [vgl. 23a]. Der Rendezvous Point leitet, wenn der Client und der Onion Service über deren Circuits miteinander verbunden sind, die Nachrichten zwischen den beiden weiter [vgl. 23b].

4 Dezentralisierung

4.1 Sicherheit

5 Vor- und Nachteile

5.1 Kriminalität

5.2 freie Meinungsäußerung

6 programmatische Umsetzung

7 Fazit

Literatur

[Ta96] D. Taipale. „Implementing the Rivest, Shamir, Adleman cryptographic algorithm on the Motorola 56300 family of digital signal processors“. In: Southcon/96 Conference Record. Juni 1996, S. 10–17. DOI: 10.1109/SOUTHC.1996.535035.

[DMS04] Roger Dingledine, Nick Mathewson und Paul Syverson. „Tor: The Second-Generation Onion Router“. In: 13th USENIX Security Symposium (USENIX Security 04). San Diego, CA: USENIX Association, Aug. 2004. URL: https://www.usenix.org/conference/13th-usenix-security-symposium/tor-second-generation-onion-router.

[Tu08] Clay S Turner. „Euler's totient function and public key cryptography“. In: Nov 7 (2008), S. 138.

[Ch+11] Sambuddho Chakravarty, Georgios Portokalidis, Michalis Polychronakis und Angelos D. Keromytis. „Detecting Traffic Snooping in Tor Using Decoys“. In: Recent Advances in Intrusion Detection. Hrsg. von Robin Sommer, Davide Balzarotti und Gregor Maier. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, S. 222–241. ISBN: 978-3-642-23644-0.

[MS13] Prerna Mahajan und Abhishek Sachdeva. „A study of encryption algorithms AES, DES and RSA for security“. In: Global Journal of Computer Science and Technology 13.15 (2013), S. 15–22.

[Ta13] Marius Tarnauceanu. A generalization of the Euler's totient function. 2013. DOI: 10.48550/arXiv.1312.1428. arXiv: 1312.1428 [math.GR]. URL: https://doi.org/10.48550/arXiv.1312.1428.

[Wa+13] Hongjun Wang, Zhiwen Song, Xiaoyu Niu und Qun Ding. „Key generation research of RSA public cryptosystem and Matlab implement“. In: PROCEEDINGS OF 2013 International Conference on Sensor Network Security Technology and Privacy Communication System. 2013, S. 125–129. DOI: 10.1109/SNS-PCS.2013.6553849.

[Gr14] Andy Greenberg. „Hacker Lexicon: What Is End-to-End Encryption?“ In: WIRED (Nov. 2014). URL: https://www.wired.com/2014/11/hacker-lexicon-end-to-end-encryption (besucht am 16. 01. 2024).

[BSW15b] Albrecht Beutelspacher, Jörg Schwenk und Klaus-Dieter Wolfenstetter. „Ziele der Kryptographie“. In: Moderne Verfahren der Kryptographie: Von RSA zu Zero-Knowledge. Wiesbaden: Springer Fachmedien Wiesbaden, 2015, S. 1–7. ISBN: 978-3-8348-2322-9. DOI: 10.1007/978-3-8348-2322-9_1. URL: https://doi.org/10.1007/978-3-8348-2322-9_1.

[El16] Elektronik Kompendium. TCP/IP. Nov. 2016. URL: https://www.elektronik-kompendium.de/sites/net/0606251.htm (besucht am 23. 01. 2024).

[Is16] Ni Made Satvika Iswari. „Key generation algorithm design combination of RSA and ElGamal algorithm“. In: 2016 8th International Conference on Information Technology and Electrical Engineering (ICITEE). 2016, S. 1–5. DOI: 10.1109/ICITEED.2016.7863255.

[Kr16] N. Krzyworzeka. „Asymmetric cryptography and trapdoor one-way functions“. In: Automatyka / Automatics 20.2 (2016), S. 39–51. ISSN: 1429-3447. DOI: 10.7494/automat.2016.20.2.39.

[Au18] Autumn. „How does Tor ∗really∗ work?“ In: (Feb. 2018). URL: https://hackernoon.com/how-does-tor-really-work-c3242844e11f (besucht am 23. 01. 2024).

[HK20] Aljaafari Hamza und Basant Kumar. „A Review Paper on DES, AES, RSA Encryption Standards“. In: 2020 9th International Conference System Modeling and Advancement in Research Trends (SMART). 2020, S. 333–338. DOI: 10.1109/SMART50582.2020.9336800.

[IBM21] IBM Documentation. März 2021. URL: https://www.ibm.com/docs/en/ztpf/2020?topic=concepts-symmetric-cryptography (besucht am 23. 01. 2024).

[LB21] Ben Lutkevich und Madelyn Bacon. „end-to-end encryption (E2EE)“. In: Security (Juni 2021). URL: https://www.techtarget.com/searchsecurity/definition/end-to-end-encryption-E2EE (besucht am 16. 01. 2024).

[Un22] United Nations. World Population Prospects - Population Division. Jan. 2022. URL: https://population.un.org/wpp/Download/Files/1_Indicators%20(Standard)/EXCEL_FILES/1_General/WPP2022_

GEN_F01_DEMOGRAPHIC_INDICATORS_COMPACT_REV1.xlsx (besucht am 13. 01. 2024).

[Am23] Amnesty International. Amnesty International Report 2022/23. London WC1X 0DW, United Kingdom: International Amnesty Ltd, 2023, S. 307–312, 122–128, 196–201. ISBN: 978-0-86210-502-0. URL: https://www.amnesty.org/en/wp-content/uploads/2023/04/WEBPOL1056702023ENGLISH-2.pdf (besucht am 13. 01. 2024).

[Fä23] Jan Fährmann. „Rechtliche Rahmenbedingungen der Nutzung von Positionsdaten durch die Polizei und deren mögliche Umsetzung in die Praxis–zwischen Strafverfolgung und Hilfe zur Wiedererlangung des Diebesguts“. In: Private Positionsdaten und polizeiliche Aufklärung von Diebstählen. Nomos Verlagsgesellschaft mbH & Co. KG. 2023, S. 141–176. ISBN: 978-3-8487-5905-7.

[K 23] Arun K. L. „Detailed Anatomy of the Tor Network | Structure of the Tor Network“. In: Sec Master (Okt. 2023). URL: https://thesecmaster.com/detailed-anatomy-of-the-tor-network-structure-of-the-tor-network (besucht am 23. 01. 2024).

[BSI23] Bundesamt für Sicherheit in der Informationstechnik. BSI TR-02102-1 Kryptographische Verfahren: Empfehlungen und Schlüssellängen. Bundesamt für Sicherheit in der Informationstechnik, Jan. 2023, S. 39–41. URL: https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/TechnischeRichtlinien/TR02102/BSI-TR-02102.pdf?__blob=publicationFile&v=9 (besucht am 21. 01. 2024).

[23a] The introduction protocol [INTRO-PROTOCOL] - Tor Specifications. Dez. 2023. URL: https://spec.torproject.org/rend-spec/introduction-protocol.html (besucht am 04. 02. 2024).

[23b] The rendezvous protocol - Tor Specifications. Nov. 2023. URL: https://spec.torproject.org/rend-spec/rendezvous-protocol.html (besucht am 04. 02. 2024).

[23c] Understanding System Design Whatsapp & Architecture. Mai 2023. URL: https://pwskills.com/blog/system-design-whatsapp/#Client-Server_Architecture (besucht am 22. 01. 2024).

[Wu+23] Mingshi Wu, Jackson Sippe, Danesh Sivakumar, Jack Burg, Peter Anderson, Xiaokang Wang, Kevin Bock, Amir Houmansadr, Dave Levin und Eric Wustrow. „How the Great Firewall of China Detects and Blocks Fully Encrypted Traffic". In: 32nd USENIX Security Symposium (USENIX Security 23). Anaheim, CA: USENIX Association, Aug. 2023, S. 2653–2670. ISBN: 978-1-939133-37-3. URL: https://www.usenix.org/conference/usenixsecurity23/presentation/wu-mingshi (besucht am 14. 01. 2024).

[24a] Deriving blinded keys and subcredentials [SUBCRED] - Tor Specifications. Jan. 2024. URL: https://spec.torproject.org/rend-spec/deriving-keys.html (besucht am 04. 02. 2024).

[24b] directory authority | Tor Project | Support. Feb. 2024. URL: https://support.torproject.org/glossary/directory-authority (besucht am 02. 02. 2024).

[24c] duckduckgo onion at DuckDuckGo. Feb. 2024. URL: https://duckduckgo.com/?q=duckduckgo+onion&atb=v160-7&ia=web (besucht am 02. 02. 2024).

[24d] How can we help? | Tor Project | Support. Feb. 2024. URL: https://support.torproject.org/#ChangePaths (besucht am 04. 02. 2024).

[24e] Kanal | Tor Project | Hilfe. Jan. 2024. URL: https://support.torproject.org/de/glossary/circuit (besucht am 31. 01. 2024).

[24f] src/core/or · main · The Tor Project / Core / Tor · GitLab. Feb. 2024. URL: https://gitlab.torproject.org/tpo/core/tor/-/blob/main/src/feature/hs/hs_descriptor.c?ref_type=heads (besucht am 04. 02. 2024).

[24g] Tor Project | How do Onion Services work? [Online; accessed 2. Feb. 2024]. Jan. 2024. URL: https://community.torproject.org/onion-services/overview.

[24h] Tor Project | Talk about onions. Jan. 2024. URL: https://community.torproject.org/onion-services/talk (besucht am 31. 01. 2024).

Anhang

[BSW15a] Albrecht Beutelspacher, Jörg Schwenk und Klaus-Dieter Wolfenstetter. „Kryptologische Grundlagen“. In: Moderne Verfahren der Kryptographie: Von RSA zu Zero-Knowledge. Wiesbaden: Springer Fachmedien Wiesbaden, 2015, S. 9–30. ISBN: 978-3-8348-2322-9. DOI: 10.1007/978-3-8348-2322-9_2. URL: https://doi.org/10.1007/978-3-8348-2322-9_2.

[Tor22] Tor Project. Tor Project | All About Tor. Aug. 2022. URL: https://community.torproject.org/training/resources/all-about-tor/#/0/7 (besucht am 23. 01. 2024).

Quellcode: https://github.com/sshcrack/enkrypton