Universität Regensburg
Wirtschaftswissenschaftliche Fakultät
Institut für Wirtschaftsinformatik

Diplomarbeit

Analyse von datenschutzfreundlichen Übertragungstechniken hinsichtlich ihres Schutzes vor Datenverkehrsanalysen im Internet

vorgelegt von

Dominik Herrmann
geb. am 2. Februar 1983 in Schrobenhausen
Matrikelnummer 1139614

Betreuer: Prof. Dr.-Ing. Hannes Federrath
Dipl.-Wirtsch.-Inf. Rolf Wendolsky

Aufgabenstellung

Angriffe, die sich der Analyse des Datenverkehrs (Traffic Analysis) bedienen, erlauben einem an der Kommunikation unbeteiligten Beobachter, Informationen über den Inhalt von verschlüsselten Datenübertragungen zu gewinnen, indem Zeitpunkte und Datenvolumen korreliert werden.

In der Diplomarbeit soll untersucht werden, inwieweit verschlüsselt übertragene Webseiten durch charakteristische Traffic-Fingerabdrücke identifiziert werden können. Dabei soll insbesondere der Sicherheitsgewinn durch verschiedene Echtzeit-Anonymisierungssysteme untersucht und miteinander verglichen werden. Die Untersuchungen sollen durch ein informationstheoretisches Modell fundiert werden.

Am Beispiel eines Burst-Proxies, der mehrere HTTP-Antworten bündelt, soll die Wirksamkeit von Gegenmaßnahmen überprüft werden.

Zusammenfassung

In dieser Diplomarbeit werden verschiedene datenschutzfreundliche Übertragungstechniken – OpenSSH-Tunnel, OpenVPN, ein IPsec-VPN, Stunnel, Tor, JonDonym und Shalon – hinsichtlich ihres Schutz gegen einen Website-Fingerprinting-Angriff untersucht. Dieser Angriff verfolgt das Ziel, verschlüsselt übertragene Webseiten durch die Analyse der Häufigkeitsverteilung der beim Abruf einer Webseite beobachtbaren IP-Paketgrößen zu identifizieren.

Das verwendete Website-Fingerprinting-Verfahren greift auf Text-Mining-Techniken zurück, wobei ein multinomialer Naïve-Bayes-Klassifizierer den Kern darstellt. Der Klassifizierer erzielt bei den Übertragungstechniken, die kein Traffic-Shaping durchführen, Erkennungsraten von über 90 % und übertrifft damit die bislang in der Literatur vorgestellten Website-Fingerprinting-Verfahren deutlich. Die Verfahren OpenSSH, OpenVPN, IPsec-VPN, Stunnel und Shalon bieten demnach – unter idealen Bedingungen – praktisch keinen Schutz gegen Website-Fingerprinting-Angriffe.

Bei Tor und JonDonym erzielt der Angriff hingegen lediglich Erkennungsraten von 3 % bzw. 20 %, was darauf zurückzuführen ist, dass diese Systeme die Nutzdaten in Datenstrukturen fixer Größe übertragen. Nutzer, die sich vor Website-Fingerprinting-Angriffen schützen möchten, sollten daher bevorzugt diese Anonymisierungssysteme verwenden.

Darüber hinaus werden zwei Gegenmaßnahmen implementiert und evaluiert: Zum einen eine modifizierte OpenSSL-Bibliothek, die Padding für TLS-Application-Records durchführt, zum anderen ein Burst-Proxy, der HTML-Seiten analysiert und die eingebetteten Objekte vorausschauend zum Browser schickt. Während das TLS-Padding praktisch wirkungslos bleibt, sinken die Erkennungsraten bei Verwendung des Burst-Proxies auf etwa 75 %.

Zuletzt wird gezeigt, dass sich Website-Fingerprinting auch unter praxisnahen Bedingungen durchführen lässt: Bei aktiviertem Browser-Cache reduzieren sich die Erkennungsraten nur unwesentlich, und die Anzahl der False Positives lässt sich durch geeignete Maßnahmen erheblich reduzieren. Ferner wird ein Verfahren vorgeschlagen, das einem aktiven Angreifer ermöglicht, Webseiten gezielt zu markieren, um ihre Identifizierbarkeit zu erhöhen.

~

In cases of defence 'tis best to weigh
The enemy more mighty than he seems

William Shakespeare (King Henry V, Act II, Scene IV)

~

Vorwort

Die vorliegende Diplomarbeit beschäftigt sich mit *Website-Fingerprinting*, einem Traffic-Analyse-Verfahren, das die Sicherheit von datenschutzfreundlichen Techniken im Internet beeinträchtigen kann. Vor etwa einem Jahr wurde meine Aufmerksamkeit zum ersten Mal auf dieses faszinierende Thema gelenkt. Die verwendeten Machine-Learning-Methoden und Data-Minning-Techniken bestechen durch Einfachheit und Cleverness zugleich. Im Rahmen der Diplomarbeit hatte ich die Möglichkeit, mich ausführlich mit Website-Fingerprinting-Verfahren zu beschäftigen.

Dennoch gestaltete sich die Anfertigung dieser Diplomarbeit erheblich aufwändiger als erwartet. Jedesmal, wenn für eine Forschungsfrage eine Antwort gefunden war, ergaben sich daraus eine Hand voll neuer Fragen. Nach sechsmonatiger Bearbeitungszeit bin ich erleichtert, diese Zeit der Höhen und Tiefen hinter mir zu lassen und mich neuen Herausforderungen (bzw. den neuen Fragen) zu stellen.

Mein Dank gilt Thomas Ernst, Hannes Federrath, Heike Gorski, Raimund Herrmann, Robert Hirschberger, Thomas Hirschvogel, Gerrit Hornung, Stefan Köpsell, Thomas Nowey, Stefan Schmidt, Christian Stüer, Benedikt Westermann und Rolf Wendolsky.

Regensburg, im März 2008

Dominik Herrmann

Inhaltsverzeichnis

1 Einleitung

1.1. Datenverkehrsanalysen

Der Begriff „Datenverkehrsanalyse“ (engl. *traffic analysis*) wird für eine Vielzahl von Verfahren verwendet. Datenverkehrsanalysen greifen normalerweise nicht auf den Inhalt der übermittelten Nachrichten zurück. Stattdessen wird der Datenverkehr auf einem höheren Abstraktionsniveau untersucht. Typischerweise sind dabei Auftretenszeitpunkte sowie Art und Größe von Datenpaketen und Verbindungen von Interesse. Durch die Analyse dieser Daten mit statistischen und informationstheoretischen Methoden lassen sich neue Erkenntnisse gewinnen.

In den vergangenen Jahren wurden zahlreiche Traffic-Analyse-Verfahren vorgestellt. Im Fokus steht vor allem die Identifizierung von Anwendungsprotokollen in verschlüsseltem Datenverkehr, die eine wirksame Durchsetzung von Sicherheitspolicies in Firewalls ermöglicht, ohne den Datenverkehr entschlüsseln zu müssen [EBR03; MP05; KPF05; ZNA05; WMM06b; WMM06a; EAM06; BT07; CDGS07]. Darüber hinaus können Datenverkehrsanalysen die Anwesenheit getunnelter Verbindungen aufdecken, um zu verhindern, dass Angreifer unbemerkt von außen auf ein Netzwerk zugreifen [BP04]. Es gibt sogar ein Traffic-Analyse-Verfahren, das die Zeichen, die in einer SSH-Sitzung eingegeben werden, ermitteln kann [SWT01]. Im Kontext von Anonymisierungssystemen spielen Datenverkehrsanalysen ebenfalls eine wichtige Rolle, da sie die Anonymität der Teilnehmer einschränken können [Ray01].

Die Untersuchung von Datenverkehrsanalysen ist von großer Bedeutung, da sich durch charakteristische Muster im Datenverkehr neue Erkenntnisse gewinnen lassen.

1.2. Website-Fingerprinting im Überblick

In dieser Arbeit liegt der Fokus auf einem speziellen Traffic-Analyse-Verfahren, das sich auf Webseiten, genauer: HTTP-Datenverkehr, beschränkt: *Website-Fingerprinting*.

Eine Einsatzmöglichkeit von Website-Fingerprinting soll an folgendem fiktiven Szenario illustriert werden:

> Alice gehört der Militanten Vereinigung (mv) an, welche die freiheitlich demokratische Grundordnung in Deutschland gefährdet. Sie ruft täglich mehrmals die Internetseite ihrer Vereinigung ab, die im Ausland gehostet wird – natürlich verschlüsselt über einen anonymisierenden Proxy-Server, den sie in ihrem Browser eingetragen hat.
>
> Eines Tages wird das Bundeskriminalamt (BKA) auf Alice aufmerksam (ihr Name wurde auf einer DVD gefunden, die ein Informant dem BKA verkauft hat). Die Polizei hat ein Problem: Sie kennt zwar den Wohnsitz von Alice, für eine richterliche Ermächtigung zur Durchsuchung der Wohnung oder gar zum Einsatz einer Remote Forensic Software reichen die Indizien jedoch nicht aus. Der Richter würde jedoch einen Durchsuchungsbeschluss ausstellen, wenn ihm der Staatsanwalt einen plausiblen Anfangsverdacht präsentieren würde, z. B. indem er nachweist, dass Alice sehr häufig die Seiten der Militanten Vereinigung abruft.
>
> Die Techniker des BKA haben in einer Diplomarbeit über einen viel versprechenden Website-Fingerprinting-Angriff gelesen, der es ermöglicht, auch bei Verwendung eines Anonymisierers den Besuch einer Webseite nachzuweisen. Sie protokollieren ein paar Tage lang den Datenverkehr des WLANs vor dem Haus von Alice. Der Angriff gelingt – die Analyse ergibt, dass Alice mit einer Wahrscheinlichkeit von über 90 % mindestens einmal täglich die Webseiten der Militanten Vereinigung abgerufen hat – diese Tatsache überzeugt den Richter, einen Durchsuchungsbeschluss auszustellen.

Der Begriff *Website-Fingerprinting* ist in der Literatur noch nicht definiert. Dieser Arbeit liegt folgende Definition zu Grunde:

Definition Website-Fingerprinting: *Beim Website-Fingerprinting verfolgt ein lokaler Angreifer das Ziel, durch die Anwendung von Traffic-Analyse-Verfahren probabilistische Aussagen über die Identität von verschlüsselt abgerufenen Webseiten zu machen.*

Mit Website-Fingerprinting kann ein *Angreifer* also ermitteln, welche Seite(n) ein einzelner Nutzer, der in diesem Kontext als *Opfer* bezeichnet wird, abgerufen hat. Der Angreifer belauscht hierzu den Kommunikationskanal seines Opfers. Ruft das Opfer die Webseiten durch Herstellen einer direkten Verbindung zum Ziel-Webserver ab, kann der Angreifer anhand des Host-Namens leicht auf den Inhalt der Seiten schließen bzw. bei unverschlüsselter Übertragung den Seiteninhalt direkt mitlesen. Hierzu bedarf es keiner ausgefeilten Traffic-Analyse-Techniken.

Website-Fingerprinting kommt erst dann ins Spiel, wenn das Opfer die Webseiten mit Hilfe einer *datenschutzfreundlichen Übertragungstechnik* abruft. Üblicherweise werden solche Techniken verwendet, um zu verhindern, dass ein Angreifer, der möglicherweise die Identität des Opfers kennt, ermitteln kann, welche Webseiten besucht werden.

Zwar kann der Angreifer den Inhalt der verschlüsselt übertragenen Seiten nicht mitlesen, er kann jedoch durch geeignete Traffic-Analyse-Verfahren probabilistische Aussagen darüber machen, zu welcher Webseite der beobachtete Datenverkehr passt. Solche probabilistischen Aussagen haben üblicherweise die Form „Mit einer Wahrscheinlichkeit von 95 % hat das Opfer soeben die Webseite *www-sec.uni-regensburg.de* abgerufen.“

Zwar lassen sich anhand der Kenntnis einer URL die tatsächlich abgerufenen Inhalte bzw. Eingaben in Formularfelder in der Regel nicht herausfinden, allein der Besuch einer Seite kann jedoch die Interessen und Gewohnheiten (etwa bei www.anonyme-alkoholiker.de) eines Benutzers preisgeben. Zudem lassen sich bei Kenntnis der Internetadresse die vom Opfer abgerufenen Seiten auch nachträglich noch aufrufen, um ihren Inhalt zu ermitteln.

Vorschau In der vorliegenden Diplomarbeit wird gezeigt, dass ein Angreifer unter Idealbedingungen durch Website-Fingerprinting ermitteln kann, welche Webseiten ein bestimmter Benutzer aufgerufen hat bzw. ob er eine bestimmte Webseite besucht hat. Dabei spielt es praktisch keine Rolle, welches datenschutzfreundliche System der Benutzer zum Surfen verwendet. Im Gegensatz zu anderen Datenverkehrsanalysen erfordert Website-Fingerprinting keinen verteilten, sondern lediglich einen lokalen passiven Angreifer, der sich in der Nähe des Opfers aufhält.

1.3. Zielsetzung

Diese Diplomarbeit adressiert einige Forschungsfragen zu Website-Fingerprinting-Verfahren. Darüber hinaus soll weiterer Forschungsbedarf identifiziert werden. Sie liefert vier wesentliche Forschungsbeiträge:

1. Entwicklung eines leistungsfähigen Website-Fingerprinting-Verfahrens auf Basis von Text-Mining-Techniken,
2. Untersuchung verschiedener datenschutzfreundlicher Übertragungstechniken hinsichtlich ihres Schutzes vor Website-Fingerprinting,
3. Untersuchung des Schutzes vor Website-Fingerprinting durch einen Burst-Proxy-Server,
4. Untersuchung der Praxistauglichkeit des Verfahrens durch Lockerung der Arbeitshypothesen.

Zur Erreichung dieser Forschungsbeiträge werden folgende Ziele verfolgt:

- Diskussion der bislang publizierten Website-Fingerprinting-Verfahren,
- Design und Implementierung eines leistungsfähigen Website-Fingerprinting-Verfahrens,
- Analyse des Website-Fingerprinting-Verfahrens mit Methoden aus der Informationstheorie und dem Data Mining,
- empirische Ermittlung der Effektivität von Website-Fingerprinting beim Einsatz unterschiedlicher datenschutzfreundlichen Übertragungstechniken im Internet,
- prototypische Entwicklung eines Burst-Proxy-Servers und Evaluation seiner Wirksamkeit,
- Untersuchung von Website-Fingerprinting unter praxisnahen Einsatzbedingungen.

1.4. Aufbau der Arbeit

In Kapitel 2 wird zunächst das Ziel von datenschutzfreundlichen Übertragungstechniken erläutert und der Nutzen von Website-Fingerprinting-Angriffen anhand eines informationstheoretischen Modells motiviert. Darüber hinaus wird der unterstellte Angreifer modelliert. Das Kapitel schließt mit der Darstellung unterschiedlicher Einsatzmöglichkeiten solcher Angriffe.

Kapitel 3 diskutiert verwandte Publikationen und grenzt diese Arbeit davon ab.

In Kapitel 4 werden die untersuchten datenschutzfreundlichen Übertragungstechniken vorgestellt, wobei zwischen Single-Hop- und Multi-Hop-Systemen unterschieden wird.

Kapitel 5 enthält Informationen zur Durchführung von Traffic-Analyse-Angriffen mit Data-Mining-Techniken. Nach der Definition der Arbeitshypothesen wird die gewählte Vorgehensweise erläutert. Darüber hinaus werden verschiedene Merkmale des Datenverkehrs hinsichtlich ihrer Eignung für einen Website-Fingerprinting-Angriff diskutiert. Die Beschreibung der Vorgehensweise bei der Erzeugung und Evaluierung großer Mengen von Testdaten rundet das Kapitel ab.

Das entwickelte Website-Fingerprinting-Verfahren – ein multinomialer Naïve-Bayes-Klassifizierer – wird in Kapitel 6 beschrieben. Im Anschluss daran wird der Klassifizierer mit den etablierten Website-Fingerprinting-Verfahren verglichen. Schließlich wird der Einfluss verschiedener Versuchsparameter analysiert.

In Kapitel 7 werden die zu untersuchenden datenschutzfreundlichen Übertragungstechniken hinsichtlich ihres Schutzes vor Website-Fingerprinting-Angriffen verglichen. Darüber hinaus wird versucht, die Ergebnisse mit Hilfe von Ähnlichkeitsanalysen und informationstheoretischen Betrachtungen theoretisch zu fundieren.

Daran schließt sich in Kapitel 8 die Betrachtung der Defensive an: In der Literatur vorgeschlagene Gegenmaßnahmen werden diskutiert, ein TLS-Padding-Verfahren und ein Burst-Proxy implementiert und evaluiert.

Kapitel 9 widmet sich der Praxistauglichkeit von Website-Fingerprinting, indem einige Arbeitshypothesen hinterfragt und auf Realitätsnähe geprüft werden. Das Kapitel geht auch auf juristische Aspekte ein.

Kapitel 10 schließt die Arbeit ab, indem die wichtigsten Erkenntnisse zusammengefasst werden. Darüber hinaus werden Ansatzpunkte für weitere Untersuchungen vorgestellt.

2

Motivation und Szenario

Dieses Kapitel gibt einen Überblick über die datenschutzfreundlichen Übertragungstechniken, die durch Website-Fingerprinting-Angriffe gefährdet werden. Darüber hinaus wird ein informationstheoretisches Modell aufgestellt, das den Nutzen solcher Angriffe verdeutlicht. Nach der Modellierung des unterstellten Angreifers wird aufgezeigt, wozu Website-Fingerprinting-Angriffe in der Praxis verwendet werden können.

2.1. Datenschutzfreundliche Übertragungstechniken

Im Rahmen dieser Arbeit werden verschiedene datenschutzfreundliche Übertragungstechniken hinsichtlich ihres Schutzes vor Website-Fingerprinting-Angriffen untersucht. Datenschutzfreundliche *Übertragungs*techniken gehören zur Gruppe der datenschutzfreundlichen Techniken *(privacy enhancing technologies)*. Witt sieht ihr Ziel in der Reduktion der „Risiken für die Privatsphäre der Betroffenen“ [Wit07, 133], indem sie die Nutzung von „Informations- und Kommunikationstechnik bei gleichzeitiger Reduzierung erforderlichen Personenbezugs“ [Wit07, 133] ermöglichen. Die im Folgenden betrachteten datenschutzfreundlichen *Übertragungs*techniken ermöglichen insbesondere den datenschutzfreundlichen Besuch von Internetseiten und adressieren damit das von Witt formulierte Prinzip des *Selbstdatenschutzes* (vgl. [Wit07, 133]).

2.1.1. Schutz der Beziehungsanonymität beim Websurfen

Datenschutzfreundliche Übertragungstechniken adressieren die Schutzziele Unbeobachtbarkeit und Anonymität (für eine ausführliche Darstellung dieser Konzepte sei auf [PH08] verwiesen). Aus Nutzersicht ist vor allem die Gewährleistung der *Beziehungsanonymität* wichtig – Außenstehende sollen nicht ermitteln können, *welche* Webseite von *welchem* Benutzer abgerufen wurde. Alle Nutzer, die einen Dienst zu einem gegebenen Zeitpunkt verwenden, kommen als Abrufer aller zu diesem Zeitpunkt übertragenen Webseiten in Frage.

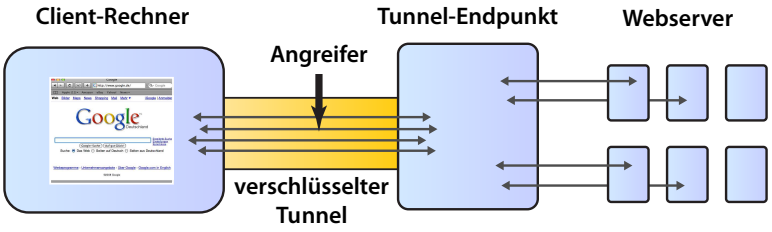


Abbildung 2.1.: Bei einem Single-Hop-System werden die HTTP-Requests in einem verschlüsselten Tunnel vom Client zu einem vertrauenswürdigen Host transportiert; dieser leitet die Anfragen an die Webserver weiter.

Sender und Empfänger eines IP-Pakets lassen sich bei normaler Kommunikation im Internet anhand der *Source-* und *Destination-IP-Adressen* ermitteln, die im Paket-Header enthalten sind. Bei HTTP lässt sich die Empfängeridentität mitunter auch aus dem Nachrichteninhalt ermitteln, und zwar dann, wenn in den HTTP-Requests die vollständige URL enthalten ist. Zur Wahrung der Beziehungsanonymität sind also zum einen die IP-Adressen von Sender und Empfänger zu verschleiern, zum anderen muss der Nachrichteninhalt verschlüsselt werden.

2.1.2. Ausprägungen datenschutzfreundlicher Systeme

Datenschutzfreundliche Übertragungstechniken lassen sich anhand ihrer Architektur voneinander abgrenzen: Single- und Multi-Hop-Dienste. Auf die Unterschiede zwischen diesen Architekturen wird in Kapitel 4 noch genauer eingegangen. Bei beiden Techniken werden die Nutzdaten (HTTP-Anfragen) in einem verschlüsselten Tunnel vom Client-Rechner zum Anonymisierungsdienst übertragen. Abbildung 2.1 veranschaulicht dieses Szenario.

Üblicherweise wird die Entscheidung für eine konkrete datenschutzfreundliche Übertragungstechnik auf Basis des persönlichen Schutzbedarfs und eines (möglicherweise unbewusst) unterstellten Angreifers getroffen. In vielen Fällen sind Single-Hop-Systeme ausreichend, um den Schutzbedarf zu decken: Die meisten Nutzer möchten verhindern, dass ihre Nachbarn erfahren, welche Webseiten sie am Laptop über ihr heimisches Funknetz abgerufen haben; die Verwendung von Verbindungsverschlüsselung (etwa mit *Wi-Fi Protected Access, WPA*) auf der Funkschnittstelle ist eine adäquate (Single-Hop-)Schutzmaßnahme. Möchte sich ein Nutzer in einem WLAN-Hotspot vor dessen Betreiber schützen, kann er ein *Virtual Private Network (VPN)* oder einen Proxy mit aktivierter Verschlüsselung verwenden. VPNs und Proxies gehören ebenfalls zur Gruppe der Single-Hop-Systeme.

Will der Anwender dem Betreiber eines Single-Hop-Dienstes nicht vertrauen müssen, kann er auf Multi-Hop-Anonymisierungsdienste zurückgreifen. Multi-Hop-Systeme stellen sicher, dass kein einzelner Knoten die Beziehungsanonymität der Kommunikationspartner aufheben kann. Die Verkettung von Sender und Empfänger wird dabei umso aufwändiger, je mehr Knoten zwischen Sender und Empfänger verwendet werden.

2.2. Informationstheoretische Veranschaulichung

Das Ziel von Website-Fingerprinting lässt sich anhand eines informationstheoretischen Modells veranschaulichen. Vereinfachend wird dabei unterstellt, dass sich das Opfer lediglich für Webseiten aus einer endlichen Menge $S = \{s_1, s_2, ...s_n\}$ von n Seiten interessiert, die auch dem Angreifer bekannt sind. Darüber hinaus sei eine Gleichverteilung der Seiten angenommen, d. h. das Opfer ruft jede Webseite mit der gleichen Wahrscheinlichkeit $p_{\text{Abruf}} = 1/n$ ab, wobei die einzelnen Abrufe voneinander unabhängig sind. Wie stehen die Chancen für den Angreifer, herauszufinden, welche Seite das Opfer zu einem bestimmten Zeitpunkt abgerufen hat?

Unter der Annahme, dass der Angreifer kein Website-Fingerprinting-Verfahren kennt und auch sonst keinerlei zusätzliches Wissen über die Seiten hat, ist aus seiner Sicht der Abruf jeder Seite gleich wahrscheinlich – er muss also raten. Die Wahrscheinlichkeit, dass er dabei die richtige Seite errät, ist in diesem Fall $p_{\text{Treffer}} = 1/n$. Eine solche Verteilung hat die höchstmögliche Entropie $H_{\max} = \log_2 n$, die Unsicherheit bezüglich des jeweils nächsten Abrufs ist also maximal.

In der Realität werden Webseiten allerdings unterschiedlich häufig abgerufen. Der Angreifer kann diese Tatsache ausnutzen, um seine Trefferquote zu erhöhen, wenn er Informationen über die Wahrscheinlichkeitsverteilung, der die Abrufe des Opfers folgen, sammeln und in seine Prognose einfließen lassen kann. Er rät dann nicht mehr zufällig, sondern prognostiziert bevorzugt Seiten mit niedrigem Informationsgehalt bzw. großer Auftretenswahrscheinlichkeit. Ist die angenommene Verteilung korrekt, kann er seine Trefferquote gegenüber dem gleichverteilten Fall verbessern, also gilt $p_{\text{Treffer}} > 1/n$.

Die Ermittlung der Wahrscheinlichkeitsverteilung für ein konkretes Opfer gestaltet sich für den Angreifer allerdings schwierig, da hierzu gerade die Informationen benötigt werden, die der Angriff liefern soll. In der Realität sinkt die Trefferquote beim Raten wegen der großen Menge der potentiell abgerufenen Seiten zudem schnell auf ein inakzeptables Niveau ab.

Informationstheoretische Zielformulierung Website-Fingerprinting verfolgt das Ziel, selbst bei unterstellter Gleichverteilung – wenn also a priori alle Webseiten gleichermaßen in Frage kommen – durch Analyse des Datenverkehrs eine *bessere Trefferquote zu erzielen als beim bloßen Raten*. Ist zusätzlich die Verteilung der Seitenabrufe a priori bekannt, kann diese Information beim Einsatz eines geeigneten Data-Mining-Verfahrens (z. B. dem Naïve-Bayes-Klassifizierer, vgl. Abschnitt 6.2.3) zur Verbesserung der Trefferquote verwendet werden.

Perfekter Schutz vor Website-Fingerprinting Im Umkehrschluss ist ein datenschutzfreundliches Übertragungsverfahren genau dann gegen Website-Fingerprinting resistent, wenn die Trefferquote des Angreifers auch nach der Analyse des Datenverkehrs genau der Trefferquote entspricht, die der Angreifer bei zufälligem Raten erzielt hätte.

2.3. Angreifermodell

2.3.1. Betrachtete Dimensionen

Das Angreifermodell wird auf Basis der Dimensionen entwickelt, die Diaz et al. zur Messung der Anonymität von datenschutzfreundlichen Techniken verwenden [DSCP02].

intern/extern Ein *interner* Angreifer kontrolliert einen oder mehrere Knoten, die zur Infrastruktur des datenschutzfreundlichen Übertragungssystems gehören. Hierzu zählen bei Single-Hop-Systemen der Proxy-Server bzw. das Tunnel-Ende, bei Multi-Hop-Systemen sind es die Mixe bzw. Router des Dienstes. Ein *externer* Angreifer hat keinen Zugriff auf die Infrastruktur des Dienstes. Er kann lediglich auf die Übertragungsleitungen bzw. auf die Internet-Router, die am Transport der IP-Pakete beteiligt sind, zugreifen.

aktiv/passiv Ein *aktiver* Angreifer ist in der Lage, den Datenverkehr zu verändern, zu unterdrücken oder eigene Nachrichten einzuschleusen. Ein *passiver* Angreifer kann den Datenverkehr lediglich beobachten.

global/verteilt/lokal Ein *globaler* Angreifer hat Zugriff auf alle Datenleitungen bzw. Hosts. Ein *lokaler* Angreifer kontrolliert lediglich einen örtlich begrenzten Teil der Kommunikationsinfrastruktur. Als Erweiterung des Modells aus [DSCP02] lässt sich noch der *verteilte* Angreifer charakterisieren: Er hat an mehreren Stellen Zugriff auf den Datenverkehr, kontrolliert im Unterschied zum globalen Angreifer jedoch nicht sämtliche Verbindungen.

2.3.2. Unterstelltes Angreifermodell: Lokaler Angreifer

Durch Kombination der in Abschnitt 2.3.1 erläuterten Dimensionen lassen sich *Angreifermodelle* bilden. Betrachtet man ein konkretes datenschutzfreundliches System, gibt ein Angreifermodell die maximale Stärke des Angreifers an, gegen den eine datenschutzfreundliche Technik gerade noch schützt. Im Folgenden werden einige gebräuchliche Angreifer vorgestellt:

Globale passive externe Angreifer (engl. *global passive adversary*) können den Datenverkehr aller Kommunikationsverbindungen abhören und durch Korrelation von übertragenen Datenmengen und Zeitpunkten potentiell die Kommunikationsbeziehungen aller Benutzer eines datenschutzfreundlichen Systems aufdecken.

Verteilte aktive externe Angreifer werden häufig bei der Analyse von Anonymisierungssystemen betrachtet; sie sind in der Lage, gezielt Nachrichten in den Dienst einzuspeisen, um dadurch die Anonymität anderer Nutzer aufzuheben.

Verteilte passive externe Angreifer können den Datenverkehr an einigen Stellen abhören und durch Korrelation von übertragenen Datenmengen und Zeitpunkten die Kommunikationsbeziehungen einiger Nutzer aufdecken.

Lokale passive externe Angreifer können den Datenverkehr an einer bestimmten Stelle im Netzwerk (nicht in der Infrastruktur des Dienstes) abhören.

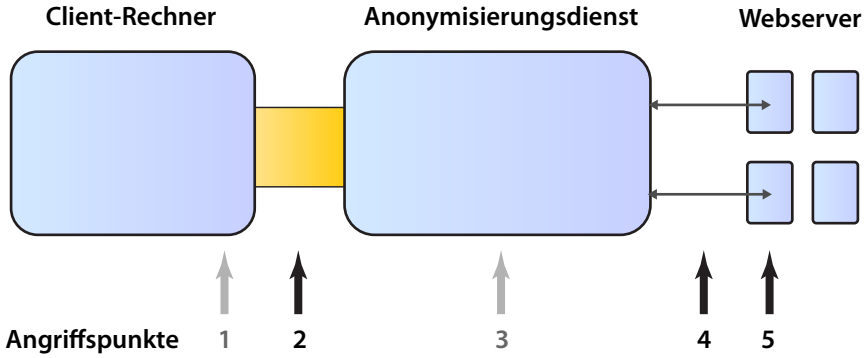


Abbildung 2.2.: Mögliche Angriffspunkte bei Anonymisierungsdiensten

Website-Fingerprinting ist ein passiver Angriff, der sowohl verteilt als auch lokal durchgeführt werden kann. Im weiteren Verlauf wird von einem *lokalen passiven externen Angreifer* ausgegangen. Die für Website-Fingerprinting vorausgesetzte Stärke des Angreifers ist also besonders gering.

Kann bereits ein lokaler Angreifer einen Website-Fingerprinting-Angriff durchführen, sind verteilte Angreifer dazu natürlich erst recht in der Lage. Da in der Literatur bereits viele Verkehrsanalyse-Techniken für verteilte Angreifer dokumentiert sind (für Tor etwa in [MD05; SW06]), welche die Beziehungsanonymität von datenschutzfreundlichen Systemen aufheben können, werden verteilte Website-Fingerprinting-Angriffe nicht im Detail betrachtet. Lediglich Abschnitt 9.3 geht kurz darauf ein, um aufzuzeigen, wie ein verteilter aktiver Angreifer die Effektivität des Angriffs steigern kann.

An dieser Stelle sei angemerkt, dass das Angreifermodell vieler datenschutzfreundlichen Techniken den hier unterstellten lokalen Angreifer durchaus zulässt. Üblicherweise geht man davon aus, dass nur ein wesentlich stärkerer (verteilter) Angreifer, der z. B. den Datenverkehr vor und hinter einem Anonymisierer korreliert, bei diesen Systemen die Sender- bzw. Empfängeridentitäten ermitteln und verknüpfen kann. Unterstellt man, dass Website-Fingerprinting auch bei Multi-Hop-Diensten funktioniert, kann damit auch ein schwächerer (lokaler) Angreifer diese Fähigkeit erlangen.

2.3.3. Angriffspunkte

Anhand von Abbildung 2.2, in der verschiedene Angriffspunkte für einen lokalen Angreifer dargestellt sind, wird das Angreifermodell nun präzisiert. Multi-Hop-Dienste werden zur Modellierung als „Black-Box“ aufgefasst: Der Datenverkehr zwischen den Knoten des Dienstes wird nicht betrachtet.

Es wird davon ausgegangen, dass der Angreifer keinen Zugriff auf den Client-Rechner oder die darauf befindliche Software hat (1). Darüber hinaus wird unterstellt, dass der Angreifer keinen Zugriff auf die Infrastruktur des Anonymisierungsdienstes hat (3). Der Angreifer hat lediglich Zugriff auf die Kommunikationsverbindung zwischen dem Client und dem Dienst (2).

An dieser Stelle hat ein Angreifer einen entscheidenden Vorteil: Häufig gehört die Source-IP-Adresse bei Datenpaketen, die bei (2) beobachtet werden können, dem tatsächlichen Sender der Daten. Aus der Source-IP-Adresse kann ein staatlicher „Angreifer“ über das automatische bzw. manuelle Auskunftsverfahren, das in den Paragraphen 112 und 113

TKG geregelt ist, die Identität des Senders ermitteln. Mit Inkraftreten des *Gesetzes zur Neuregelung der Telekommunikationsüberwachung und anderer verdeckter Ermittlungsmaßnahmen sowie zur Umsetzung der Richtlinie 2006/24/EG* sind Anbieter von Internetzugangsdiensten dazu verpflichtet, u. a. die dynamisch zugewiesene IP-Adresse sechs Monate auf Vorrat zu speichern (vgl. § 113a Abs. 1 und 4 TKG). Auch andere Angreifer sind möglicherweise in der Lage, die Identität des Senders anhand seiner IP-Adresse aufzudecken; so können Netzwerkadministratoren die Identität der Nutzer in der Regel anhand der IP-Adresse im LAN ermitteln. Eine Senderanonymität ist also in der Regel nicht gewährleistet. Zur Aufhebung der Beziehungsanonymität muss der Angreifer dann lediglich die abgerufene Webseite identifizieren, z. B. durch Website-Fingerprinting.

Protokolliert der Angreifer hingegen lediglich den Datenverkehr zwischen Anonymisierungsdienst und Webserver (4), ist die Aufhebung der Beziehungsanonymität nicht möglich, da die IP-Pakete lediglich die Adresse des Anonymisierungsdienstes enthalten und zunächst kein Rückschluss auf den tatsächlichen Sender möglich ist. Ein lokaler passiver externer Angreifer kann Website-Fingerprinting also nur dann effektiv durchführen, wenn er den Datenverkehr *vor dem Anonymisierungsdienst* abhören kann.

Ein aktiver Angreifer, der einen Webserver kontrolliert (5) und die Identität der Abrufer ermitteln will, kann einen besonders effektiven Website-Fingerprinting-Angriff durchführen, indem er den Datenverkehr der Seite durch Aufprägen eines besonders charakteristischen Musters markiert. Dieser Angriff macht allerdings nur dann Sinn, wenn der Angreifer gleichzeitig den Datenverkehr eines bzw. mehrerer Clients abhören kann (2). Diese Idee wird in Abschnitt 9.3 noch einmal aufgegriffen.

2.4. Einsatzmöglichkeiten

Website-Fingerprinting ist ein *Dual-Use-Tool*, ähnlich wie viele Security-Scanner oder Penetration-Testing-Tools: Aus der Sicht des Datenschutzes stellt es eine Bedrohung für die Benutzer dar, da ein Angreifer damit Interessen- und Nutzungsprofile erstellen kann, obwohl die Nutzer datenschutzfreundliche Techniken verwenden.

Website-Fingerprinting lässt sich jedoch auch für gemeinhin als gut empfundene Zwecke verwenden, zum Beispiel zur Strafverfolgung. Strafverfolger können damit – sofern sie dazu gesetzlich legitimiert sind – die Kommunikation und damit das Verhalten von Verdächtigen beobachten, die über verschlüsselte Verbindungen kommunizieren. Je nach gewählter Perspektive sind also unterschiedliche Forschungsziele im Hinblick auf Website-Fingerprinting-Verfahren zu verfolgen.

2.4.1. Erstellung von Nutzerprofilen

Mit Website-Fingerprinting kann die Privatsphäre von Internet-Nutzern verletzt werden. Benutzer, die aufwändige datenschutzfreundliche Techniken verwenden, gehen davon aus, dass Außenstehende bzw. Unbefugte nicht in der Lage sind, zu ermitteln, welche Internetseiten sie über den Dienst abgerufen haben. Ein Angreifer kann jedoch auf die Gewohnheiten und Interessen eines Benutzers schließen, indem er über längere Zeiträume dessen Datenverkehr protokolliert und durch Website-Fingerprinting die abgerufenen Webseiten ermittelt. Für eine solch umfangreiche Überwachung muss der Angreifer Fingerabdrücke für eine möglichst große Anzahl von Webseiten aufzeichnen, um diese dann

im abgehörten Datenverkehr des Opfers zu suchen. Das Ziel des Angreifers ist in diesem Fall also die Identifizierung möglichst vieler Seiten.

Vor diesem Hintergrund sind wirksame Gegenmaßnahmen zu entwickeln, um die Privatsphäre der Internetnutzer zu schützen.

2.4.2. Strafverfolgung

Im Rahmen der Strafverfolgung kann jedoch auch eine andere Fragestellung auftreten: Es ist zu ermitteln, ob ein bestimmter Benutzer eine ganz bestimmte (strafrechtlich relevante) Webseite abgerufen hat. Im Gegensatz zur Erstellung von umfangreichen Nutzungsprofilen, auf die im vorigen Abschnitt eingegangen wurde, interessiert sich der Anwender von Website-Fingerprinting – üblicherweise die Strafverfolgungsbehörden – in diesem Szenario für vergleichsweise wenige Seiten. Diese Seiten sollen jedoch möglichst zuverlässig identifiziert werden, um einem Benutzer den Besuch einer Webseite nachzuweisen. Abschnitt 9.4 geht auf einige juristischen Aspekte und Fragestellungen ein, die im Zusammenhang mit Website-Fingerprinting von Bedeutung sind.

Hier ist das Forschungsziel also gerade die Steigerung der Effektivität von Website-Fingerprinting-Angriffen.

3 Existierende Verfahren

In diesem Kapitel werden existierende Website-Fingerprinting-Verfahren vorgestellt, die die Basis für die Implementierung des Website-Fingerprinting-Verfahrens in Kapitel 6 bilden. Alle Verfahren verfolgen das Ziel, verschlüsselt übertragene Webseiten durch den Vergleich des Datenverkehrs mit bekannten Mustern zu identifizieren.

Website-Fingerprinting-Verfahren haben bislang drei Entwicklungsstufen durchlaufen: Im ersten Schritt stand die Identifizierung von HTTPS-gesicherten Webseiten anhand der Größe der HTML-Dateien und der darin eingebetteten Objekte im Fokus. Darauf aufbauend wurden Verfahren konzipiert, welche Seiten erkennen konnten, die über SSL-Proxies von beliebigen Webservern abgerufen wurden. Die neuesten Verfahren funktionieren sogar, wenn die Größe der übertragenen Objekte unbekannt ist und lediglich die Häufigkeitsverteilung der IP-Paketgrößen zur Verfügung steht.

3.1. Traffic-Analyse-Angriffe auf HTTPS

Bereits 1996 weisen Wagner und Schneier darauf hin, dass sich bei Servern, die SSL v3.0 verwenden, trotz Verschlüsselung die Länge der Request-URLs ermitteln lässt [WS96]. Die Spezifikation des SSL-3.0-Nachfolgers TLS 1.0 enthält daher eine ausdrückliche Warnung:

> "Any protocol designed for use over TLS must be carefully designed to deal with all possible attacks against it. Note that because the type and length of a record are not protected by encryption, care should be taken to minimize the value of traffic analysis of these values." [DA99]

Die nachfolgenden Angriffe zeigen, dass diese Warnung allerdings häufig ignoriert wird.

3.1.1. Identifizierung von Seiten beim Abruf von SSL-Webservern

Traffic-Analyse-Angriffe auf HTTPS-Verbindungen nutzen die Tatsache aus, dass Größe und Anzahl der in HTML-Seiten eingebetteten Objekte für viele Seiten charakteristisch sind. 1998 demonstrieren unabhängig voneinander Mistry und Cheng et al., dass

ein Angreifer allein durch die Analyse der Datenmenge, die über eine SSL-Verbindung transportiert wird, relativ zuverlässig auf die URL der abgerufenen Seite schließen kann [MR98; CA]. Diese ersten Angriffe unterliegen jedoch einer Einschränkung: sie operieren jeweils nur auf einem einzelnen Webserver und nicht über Host-Grenzen hinweg.

Die Autoren unterstellen, dass der Angreifer vorab für den jeweiligen Server die Größe aller HTML-Dateien sowie die Größe der darin eingebetteten Objekte ermitteln kann. Zur Evaluation rufen sie mit einem Web-Crawler alle HTML-Seiten eines Webservers ab und speichern jeweils URL, Größe der HTML-Datei sowie Anzahl und Gesamtgröße der darin eingebetteten Objekte in einer Tabelle ab.

Werden dieselben Seiten später mit SSL verschlüsselt abgerufen, ist eine Identifizierung leicht möglich, da bei den damals verwendeten SSL-Versionen das übertragene Datenvolumen nur geringfügig (und zudem deterministisch) von den tatsächlichen Objektgrößen abweicht und für jeden HTTP-Request eine separate HTTPS-Verbindung verwendet wird. Zur Identifikation einer unbekannten Seite suchen die Autoren in der Tabelle nach einem Datensatz, der zu den beobachteten Daten passt. Bereits dieses einfache Verfahren erzielt Erkennungsraten von über 90 %.

Der beschriebene Angriff ist heute allerdings nicht mehr anwendbar, da bei Webservern, die HTTP 1.1 unterstützen, mehrere HTTP-Requests und -Responses in einer SSL-Verbindung übertragen werden (z. B. durch HTTP-Pipelining, vgl. RFC 2616, section 8.1 [FGM$^+$99]). Eine präzise Ermittlung der Objektgrößen ist dann nicht mehr möglich.

3.1.2. Analyse der Entropie der Objektgrößen

Danezis veröffentlicht in einem Arbeitspapier die Ergebnisse verschiedener Traffic-Analyse-Experimente [Dan02], wobei er versucht, den Einfluss von TLS durch Aufrunden der Dateigrößen auf Vielfache von 8 Byte zu simulieren. Er ruft dazu eine große Anzahl von Seiten von der Seite *news.bbc.co.uk* ab und ermittelt die Verteilung der Größe der HTML-Dateien; als Entropie der Größenverteilung ermittelt er 9,8 Bit. Er argumentiert, dass sich demnach maximal $2^{9,8} \approx 891$ Webseiten anhand ihrer Dateigröße unterscheiden lassen. In einem weiteren Versuch berücksichtigt er auch die Anzahl der enthaltenen Bilder. Der zusätzliche Informationsgewinn fällt mit 0,3 Bit jedoch sehr gering aus.

Die Ergebnisse von Danezis haben allerdings keine Allgemeingültigkeit; sie beziehen sich lediglich auf das untersuchte Internetangebot der BBC. Die Berechnung des Informationsgehalts erfolgt zudem unter unzutreffenden Annahmen: Bei der Aufrundung auf 8-Byte-Blöcke bezieht sich der Autor auf die Block-Länge von TLS. Zwar definiert der TLS-Standard als *default block length* 8 Byte [DA99], die effektive Block-Länge hängt jedoch vom verwendeten Kryptoverfahren ab. Bei Verwendung von TLS mit AES [Nat01] beträgt die Block-Länge daher 16 Byte (siehe auch Abschnitt 8.3). Es ist davon auszugehen, dass die Entropie der Größenverteilung beim Einsatz von AES deutlich niedriger ist als der von Danezis ermittelte Wert.

Nichtsdestotrotz erscheint die von Danezis vorgeschlagene Methodik zur Ermittlung einer Obergrenze für die Anzahl der unterscheidbaren Seiten sinnvoll. Sie wird von vielen anderen Autoren (etwa [SSW$^+$02; LL06]) aufgegriffen, um deren Website-Fingerprinting-Verfahren damit zu evaluieren. In Abschnitt 7.1.2 wird im Detail auf die Ermittlung von Entropie und Informationsgehalt eingegangen.

3.2. Traffic-Analyse basierend auf Objektgrößen

Auch die Website-Fingerprinting-Verfahren, die in diesem Abschnitt vorgestellt werden, ziehen zur Identifizierung von Webseiten die Größe von HTML-Seiten und eingebetteten Objekten heran. Sie unterscheiden sich jedoch von den Verfahren, die in Abschnitt 3.1 vorgestellt wurden: Das Ziel der Verfahren aus Abschnitt 3.1 ist die Ermittlung der URL einer über HTTPS abgerufenen Seite, wenn der Webserver bereits bekannt ist. Das Ziel der nachfolgend beschriebenen Verfahren ist hingegen die Identifizierung von Webseiten, die über einen Proxy-Server abgerufen werden, der den Datenverkehr verschlüsselt (möglicherweise ebenfalls mit SSL bzw. TLS). Die zu identifizierenden Seiten können dabei von beliebigen Webservern stammen.

3.2.1. Website-Fingerprinting-Angriff auf SafeWeb

Hintz stellt einen Traffic-Analyse-Angriff vor, um auf die mangelhafte Sicherheit des *SafeWeb*-Dienstes[1] hinzuweisen [Hin02]. SafeWeb bietet einen Web-Proxy-Server an, der HTTP-Requests über SSL-gesicherte Verbindungen entgegennimmt. Hintz implementiert einen Website-Fingerprinting-Angriff, der es einem Angreifer ermöglicht, ein Nutzungsprofil eines SafeWeb-Benutzers zu erstellen.

Im Gegensatz zum Abruf von Webseiten über HTTPS ist es bei SafeWeb nicht möglich, den Host-Namen eines Webservers zu ermitteln, da dieser verschlüsselt an den Proxy-Server übertragen wird. Da SafeWeb jedoch für jeden HTTP-Request eine neue TCP-Verbindung aufbaut, lassen sich Anzahl und Größe der verschlüsselten Requests wie beim Angriff auf HTTPS (vgl. Abschnitt 3.1) leicht ermitteln.

Analog zur Vorgehensweise von Minsky und Cheng et al. unterstellt Hintz, dass der Angreifer in der Lage ist, vor dem eigentlichen Angriff die Verteilung der Dateigrößen für populäre Webseiten aufzuzeichnen und in einer Datenbank abzuspeichern. Der Angreifer kann die URL einer verschlüsselt übertragenen Seite ermitteln, indem er in seiner Datenbank nach dem Datensatz sucht, das mit der zu identifizierenden Seite hinsichtlich der Dateigröße am ehesten übereinstimmt. Hintz bezeichnet die *Größenverteilung der Objekte*, die er nach dem Abruf einer einzelnen Webseite aus der tcpdump-Logdatei extrahiert, als *website fingerprint*.

Hintz veröffentlicht eine rudimentäre Analyse der Effektivität seines Verfahrens: Bei ausgewählten Seiten stimmen typischerweise 45 bis 75 % der Verbindungen in der Größe exakt überein, wenn zwei Fingerabdrücke der gleichen Seite verglichen wurden. Fingerabdrücke von unterschiedlichen Seiten waren hingegen höchstens zu 6 % identisch. Wegen des kleinen Stichprobenumfangs – lediglich fünf Webseiten – haben diese Zahlen allerdings nur geringe Aussagekraft.

3.2.2. Vergleich von Traffic-Signaturen mit dem Jaccard-Koeffizienten

Sun et al. gehen bei ihren umfangreichen Traffic-Analyse-Untersuchungen ebenfalls davon aus, dass der Benutzer alle Webseiten mit SSL/TLS über einen Proxy-Server abruft, um Inhalte und Adressen der Webseiten vor Außenstehenden zu verbergen [SSW+02].

[1] SafeWeb Inc. wurde 2003 von der Symantec Corporation übernommen. Der von Hintz untersuchte SafeWeb-Dienst ist seitdem nicht mehr öffentlich zugänglich.

Wie bei Hintz basiert der Website-Fingerprinting-Angriff darauf, dass der Angreifer im verschlüsselten Datenverkehr die Größe der einzelnen Objekte ermitteln kann.

Sun et al. extrahieren aus dem Datenverkehr die Menge der beim Abruf einer Webseite beobachtbaren Objektgrößen – von den Autoren als *traffic signature* bezeichnet. Sie unterstellen, dass der Angreifer für alle Seiten, die ihn interessieren, solche Traffic-Signaturen erzeugt und abspeichert. Zur Ermittlung der URL einer verschlüsselt übertragenen Seite wird die Menge ihrer Objektgrößen paarweise mit allen bekannten Mengen verglichen (mit dem *Jaccard-Koeffizienten*, vgl. Abschnitt 6.2.1). Es wird dann die URL vorausgesagt, zu der die Menge mit der höchsten Übereinstimmung gehört.

Die Leistungsfähigkeit ihrer Technik demonstrieren die Autoren anhand von 100 000 URLs aus dem *DMOZ Open Directory*. Die Stichprobe besteht aus zwei Teilmengen: $S_{\text{interessant}}$ enthält 2191 „interessante“ Seiten, die mittels Traffic-Analyse identifiziert werden sollen, S_{normal} enthält 98 496 „uninteressante“ Seiten. Ihre Veröffentlichung ist insbesondere wegen der Einbeziehung *uninteressanter* Seiten von Bedeutung: In der Praxis werden Website-Fingerprinting-Verfahren mit dem Datenverkehr von Webseiten konfrontiert, für die kein Fingerabdruck in der Datenbank hinterlegt ist. Es ist daher entscheidend, dass das Verfahren diesen uninteressanten Datenverkehr ignoriert und möglichst keine Fehlalarme (*False Positives*, vgl. Abschnitt 6.4) erzeugt.

Zur Evaluation ihres Verfahrens ermitteln die Autoren die Traffic-Signaturen aller 100 000 Seiten durch automatisiertes Abrufen mit einem Browser. In Anlehnung an Danezis' Analyse gehen die Autoren der Frage nach, ob ein Angreifer mit ihrer Website-Fingerprinting-Methode alle Seiten im Internet identifizieren könnte. Eine Analyse der Stichprobe ergibt, dass die Verteilung der Elementgrößen nach Aufrunden auf Vielfache von 32 Byte (zur Simulation von Padding) eine Entropie von 8,4 Bit enthält. Im Mittel wurden für eine Webseite 11 Objekte heruntergeladen. Die Autoren ermitteln damit für die Menge der Objektgrößen, die beim Abruf einer Seite beobachtbar ist, einen durchschnittlichen Informationsgehalt von 67 Bit. Darauf aufbauend argumentieren sie, dass somit maximal $2^{67} \approx 1,48 \cdot 10^{20}$ Internetseiten zu unterscheiden seien, was völlig ausreiche, um alle Webseiten im Internet eindeutig zu identifizieren. Zum Vergleich: Schätzungen zufolge befanden sich im August 2005 im Index von Yahoo etwa $19,2 \cdot 10^9$ Seiten [Lee07]. Die auf diese Weise ermittelte obere Schranke hat jedoch nur eingeschränkte Aussagekraft (vgl. Diskussion in Abschnitt 7.1.2).

Mit ihrem Traffic-Analyse-Verfahren sind die Autoren in der Lage, 75 % der Seiten in $S_{\text{interessant}}$ eindeutig zu identifizieren – angesichts der relativ großen Stichprobe ein zufrieden stellendes Ergebnis. Gleichzeitig werden nur 1,5 % der Seiten aus S_{normal} fälschlicherweise Seiten aus $S_{\text{interessant}}$ zugeordnet, ein sehr niedriger Wert, der das Verfahren für den Praxiseinsatz qualifiziert. Bemerkenswert ist ferner die Tatsache, dass fast 40 % der Seiten in $S_{\text{interessant}}$ offenbar so dynamisch erzeugt werden, dass sich die Traffic-Signaturen von zwei direkt hintereinander durchgeführten Abrufen bereits voneinander unterschieden. Da das Verfahren selbst unter diesem Umständen eine hohe Erkennungsrate liefert, ist also ziemlich robust gegen Änderungen an den einzelnen Seiten.

Sun et al. unterstellen bei der Evaluation, dass der Benutzer den Browser-Cache deaktiviert hat. Zur Rechtfertigung dieser Maßnahme – die in der Realität wohl nur selten ergriffen wird – verweisen die Autoren auf einen Timing-Angriff, der es dem Betreiber eines Webservers ermöglicht, herauszufinden, ob der Benutzer zuvor eine andere Seite besucht hat [FS00]. Seit es das SafeCache-Add-On[2] für Firefox gibt, das den Browser-Cache

[2]Homepage: https://addons.mozilla.org/de/firefox/addon/1474

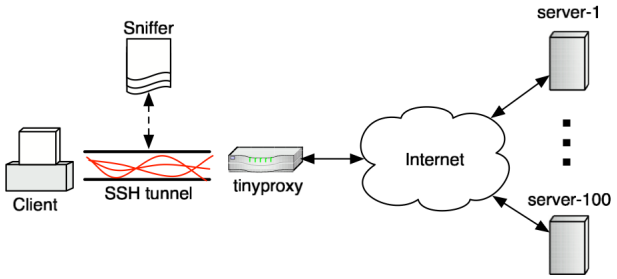


Abbildung 3.1.: Versuchsaufbau für Website-Fingerprinting bei Verwendung eines OpenSSH-Tunnels und HTTP-Proxies (Abbildung aus [BLJL05])

für unterschiedliche Seiten getrennt verwaltet, lässt sich der referenzierte Timing-Angriff allerdings streng genommen nicht mehr zur Rechtfertigung anführen.

3.3. Traffic-Analyse basierend auf IP-Paketen

Die bislang vorgestellten Verfahren setzen voraus, dass der Browser des Opfers für jedes übertragene Element eine neue Verbindung öffnet, so dass der Angreifer die Größe der einzelnen heruntergeladenen Seitenbestandteile erfährt. Wegen der hohen Verbreitung von HTTP 1.1, das mehrere HTTP-Requests über eine einmal aufgebaute Verbindung transportieren kann *(Pipelining*, vgl. RFC 2616, section 8.1 [FGM+99]), ist diese Annahme üblicherweise jedoch nicht erfüllt. Im Folgenden werden zwei Veröffentlichungen vorgestellt, die auch dann die übermittelten Webseiten identifizieren können, wenn sich Anzahl und Größe der eingebetteten Objekte nicht aus dem Datenverkehr ermitteln lassen.

3.3.1. Verwendung des Korrelationskoeffizienten

Bissias et al. unterstellen bei ihrer Untersuchung einen lokalen Angreifer, der sich im selben Netz aufhält wie sein Opfer [BLJL05]. Für die Untersuchung stellen die Autoren dieses Szenario nach (vgl. Abbildung 3.1), indem sie Webseiten über einen OpenSSH-Tunnel (mit aktivierter Verschlüsselung und Datenkompression) in Verbindung mit tinyproxy abrufen. Die Autoren wählen 100 populäre Webseiten aus, deren Startseiten über einen Zeitraum von drei Monaten im Stundentakt mit dem Firefox-Browser (mit deaktiviertem Cache) abgerufen werden. Die verschlüsselten SSH-Pakete werden mit Hilfe von tcpdump protokolliert.

Für jeden Abruf archivieren die Autoren zwei Datensätze: Der *time trace* enthält die relativen Zeitabstände zwischen einzelnen IP-Paketen *(packet inter-arrival time)*, während der *size trace* die aufgetretenen Paketgrößen beinhaltet. Dabei wird die Reihenfolge der Pakete berücksichtigt. Aus je 24 aufeinanderfolgenden *size traces* und *time traces* werden schließlich durch elementweise Mittelwertbildung für jede Webseite ein *size profile* und ein *time profile* erstellt. Die Autoren haben in der Versuchsbeschreibung allerdings ein wichtiges Detail vergessen: Es bleibt unklar, ob bei der Bildung der Profile lediglich die empfangenen oder auch die gesendeten Pakete berücksichtigt werden.

Zur Identifizierung einer Webseite anhand ihres zunächst unbekannten *traces* wird die Ähnlichkeit zwischen dem *trace x* und allen *profiles y* ermittelt. Hierzu bilden die Autoren jeweils den Korrelationskoeffizient r nach Formel (3.1).

$$r_{xy} = \frac{\sum_{i=1}^{n}[(x_i - \overline{x})(y_i - \overline{y})]}{\sqrt{\sum_{i=1}^{n}(x_i - \overline{x})}\sqrt{\sum_{i=1}^{n}(y_i - \overline{y})}} \tag{3.1}$$

Die von den Autoren zur Veranschaulichung angegebene Formel für die *cross correlation* weicht allerdings von der üblichen Berechnungsvorschrift des Pearson-Korrelationskoeffizienten ab; dieser ist nach [Ben07, 283] wie folgt zu berechnen:

$$r_{xy} = \frac{\sum_{i=1}^{n}[(x_i - \overline{x})(y_i - \overline{y})]}{\sqrt{\sum_{i=1}^{n}(x_i - \overline{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \overline{y})^2}} \tag{3.2}$$

Der Wert r des Korrelationskoeffizienten spiegelt die Ähnlichkeit der zu vergleichenden Reihen wider. Die Profile werden dann absteigend nach r sortiert, um die Webseiten (genauer: die Profile) zu ermitteln, die am besten mit dem vorliegenden *trace* übereinstimmen. Die Autoren finden heraus, dass die *size traces* den größten Beitrag zur Erkennungsleistung liefern, wohingegen die *time traces* die Ergebnisse nur noch geringfügig verbessern.

Im Vergleich zu den bisher vorgestellten Verfahren ist die Effektivität des Verfahrens allerdings gering: Bei Verwendung von Profilen, die jeweils aus 24 *size traces* und *time traces* eines Tages bestehen, werden am nächsten Tag nur 23 % der Seiten auf Anhieb richtig identifiziert. Werden jeweils die zehn ähnlichsten Profile herangezogen (was 10-maligem Raten entspricht), so können immerhin etwa 60 % der Seiten identifiziert werden. Die Erkennungsraten fallen jeweils um einige Prozent, wenn der Zeitabstand zwischen Profilerstellung und Aufzeichnung des *traces* größer wird.

Als Ursache für die schlechten Erkennungsraten identifizieren die Autoren eine kleine Gruppe von Webseiten, die besonders schlecht identifizierbar ist – warum diese Seiten nicht zuverlässig erkannt werden, finden sie jedoch nicht heraus. Werden lediglich die 25 Seiten betrachtet, die sich am leichtesten identifizieren lassen, steigen die Erkennungsrate bei einmaligem Raten immerhin auf 40 % und bei 3-maligem Raten schließlich auf 100 %.

Bissias et al. zeigen also, dass es auch ohne Kenntnis der genauen Objektgrößen möglich ist, verschlüsselt übertragene Webseiten zu identifizieren. Mit dem Korrelationskoeffizienten lassen sich allerdings nur wenige Seiten zuverlässig identifizieren. Ihre Ergebnisse sind angesichts der Ungereimtheiten in der Publikation und der stellenweise geringen Präzision ohnehin nur eingeschränkt nachvollziehbar.

3.3.2. Anwendung von Data-Mining-Techniken

Die Basis für das in dieser Arbeit entwickelte Website-Fingerprinting-Verfahren ist die Veröffentlichung von Liberatore und Levine [LL06]. Ihre Ergebnisse basieren auf einer Stichprobe von 2000 Webseiten, die in einem Zeitraum von zwei Monaten viermal pro Tag (im Abstand von sechs Stunden) abgerufen wurden. Zur Auswertung des Datenverkehrs greifen die Autoren ausschließlich auf die Paketgrößen zurück – ihre *traces* berücksichtigen weder Zeitabstände noch die Reihenfolge zwischen den Paketen.

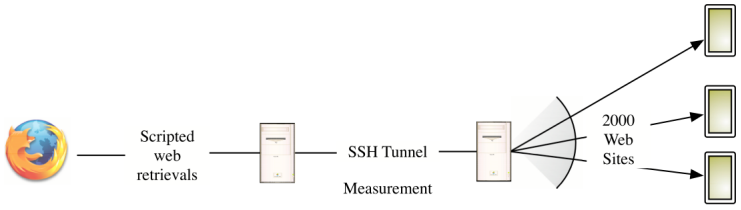


Abbildung 3.2.: Versuchsaufbau für Website-Fingerprinting mit OpenSSH-SOCKS-Proxy (Abbildung aus [LL06])

Auch hier kommt ein automatisierter Firefox-Browser mit deaktiviertem Caching zum Einsatz, wobei die Autoren den Browser zusätzlich so umkonfiguriert haben, dass er während der Messungen keine Anfragen für Software-Updates, Live-Bookmarks o. ä. durchführt. Wie Abbildung 3.2 zeigt, unterscheidet sich der Versuchsaufbau nur marginal vom Aufbau, den Bissias et al. gewählt haben. Allerdings wird hier der SSH-SOCKS-Proxy direkt – ohne Verwendung eines lokalen HTTP-Proxies – im Browser eingetragen.

Die Autoren untersuchen zwei statistische Verfahren auf ihre Website-Fingerprinting-Tauglichkeit: den bereits erwähnten *Jaccard-Koeffizienten* (vgl. Abschnitt 6.2.1) und einen *Naïve-Bayes-Klassifizierer* (vgl. Abschnitt 6.2.2) mit *Kernel-Density-Estimation* aus dem Weka-Toolkit (vgl. Abschnitt 5.5). Während der Jaccard-Klassifizierer lediglich das *Auftreten* einzelner Paketgrößen berücksichtigt, zieht der Naïve-Bayes-Klassifizierer (NB-Klassifizierer) auch die *Häufigkeit* des Auftretens zum Vergleich des Datenverkehrs verschiedener Seiten heran. Auch die Senderichtung der Pakete wird ausgewertet.

Informationsgehalt Die Autoren berechnen den Informationsgehalt der beobachteten Paketgrößen-Verteilung, die dem Klassifizierer, der auf dem Jaccard-Koeffizienten basiert, zur Verfügung steht. Hierzu ermitteln sie die Häufigkeitsverteilung der Paketgrößen über alle Instanzen hinweg, wobei die Auftretenshäufigkeit einer Paketgröße innerhalb eines *traces* nicht berücksichtigt wird. Die Entropie dieser Verteilung beträgt 7,53 Bit. Als obere Schranke für den Informationsgehalt, der dem Jaccard-Klassifizierer zur Verfügung steht, ermitteln die Autoren 137 Bit.[3] Sie argumentieren, dass sich bei dieser Häufigkeitsverteilung also theoretisch maximal $2^{137} \approx 1,7 \cdot 10^{41}$ Webseiten voneinander unterscheiden ließen – erheblich mehr als bei Website-Fingerprinting auf Basis von Objektgrößen (Sun et al. ermittelten einen mittleren Informationsgehalt von 67 Bit, vgl. Abschnitt 3.2.2). Da die Autoren dieselbe Formel verwenden wie Sun et al. ist die Aussagekraft ihrer Abschätzung allerdings gering.

Evaluation der Klassifizierer Die Autoren erzeugen mit den beiden Klassifizierungsverfahren *Profile* für die einzelnen Webseiten. In jedes Profil fließen t aufeinanderfolgende *traces* ein. Anhand der Profile sollen die beiden Klassifizierungsverfahren dann ermitteln, zu welcher Webseite ein unbekannter *trace* gehört, wobei der Zeitabstand Δ zwischen Profilerstellung und Klassifizierung variabel ist. Anhand ihrer Stichprobe analysieren sie isoliert voneinander eine Reihe von Aspekten:

[3] Die Autoren berechnen den Informationsgehalt nur für den Klassifizierer, der auf dem Jaccard-Koeffizienten, basiert, jedoch nicht für den Naïve-Bayes-Klassifizierer.

Einfluss von t Bereits mit einem Profil, das nur einen einzigen *trace* enthält, können bei einer Stichprobe von 1000 Seiten über 60 % der Seiten mit der Jaccard-Ähnlichkeitsmetrik korrekt identifiziert werden. Der NB-Klassifizierer identifiziert unter diesen Versuchsbedingungen hingegen lediglich etwa 40 % der Seiten korrekt. Mit steigendem t holt der NB-Klassifizierer schnell auf: Bei acht *traces* sind die Erkennungsraten beider Verfahren praktisch identisch: ca. 75 % der Seiten werden dann korrekt identifiziert.

Einfluss von Δ Die Erkennungsraten nehmen bei zunehmender Verzögerung nur leicht ab: selbst nach vier Wochen werden noch über 60 % der Seiten korrekt erkannt. Für einen erfolgreichen Angriff müssen die Profile also nicht unbedingt täglich aktualisiert werden.

Anzahl der Rateversuche Beide Verfahren identifizieren bereits über 70 % der Seiten auf Anhieb. Bei 10-maligem Raten lassen sich fast 90 % der Webseiten korrekt identifizieren.

Einfluss des Stichprobenumfangs Mit steigendem Stichprobenumfang N nehmen die Erkennungsraten beider Verfahren ab. Anhand einer Regressionsanalyse ermitteln die Autoren folgenden Zusammenhang: $accuracy = -0,03420 \log_2 N + 1,0679$. Für $N = 10^6$ ergibt sich eine Erkennungsrate von 38 %. Gemäß dieser Schätzung eignen sich die Verfahren also nur eingeschränkt zur Identifizierung großer Seitenbestände. Zudem ist davon auszugehen, dass bei diesen Größenordnungen die Performanz der Klassifizierer zum Flaschenhals wird – hierzu wäre eine Analyse der Zeitkomplexität erforderlich, welche die Autoren jedoch schuldig bleiben.

Zusammenfassend stellen die Autoren fest, dass – bei Abwesenheit von Gegenmaßnahmen – der Jaccard-Koeffizient bessere Ergebnisse liefert als der NB-Klassifizierer.

3.4. Abgrenzung zur Literatur

Das für den Vergleich von datenschutzfreundlichen Techniken entwickelte Website-Fingerprinting-Verfahren ist zwar an die Vorgehensweise von Liberatore et al. [LL06] angelehnt, es verwendet jedoch ein effektiveres und effizienteres Klassifizierungsverfahren, das in Abschnitt 6.2.3 vorgestellt wird.

Die Ergebnisse der bislang untersuchten Übertragungstechniken lassen sich nicht vergleichen, da unterschiedliche Testdaten und Website-Fingerprinting-Verfahren zum Einsatz kamen. Die vorliegende Arbeit verfolgt das Ziel, verschiedene Übertragungstechniken zu vergleichen, indem alle Systeme mit denselben Testdaten und demselben Website-Fingerprinting evaluiert werden.

In der Literatur wird die Wirksamkeit von Gegenmaßnahmen lediglich anhand von Simulationen evaluiert. In der vorliegenden Arbeit werden zwei Gegenmaßnahmen (TLS-Padding und Burst-Proxy) hingegen auf Basis von empirischen Untersuchungen analysiert.

Ferner gibt es bislang keine Publikation, in welcher der Einfluss des Browser-Caches auf die Erkennungsraten analysiert wird.

4

Untersuchte datenschutzfreundliche Techniken

Um zu ermitteln, inwiefern datenschutzfreundliche Techniken vor Website-Fingerprinting-Angriffen schützen, wurden mehrere Messungen und Versuche mit den einzelnen Techniken durchgeführt. Bei der Interpretation der Ergebnisse sind die Voraussetzungen und Charakteristika der untersuchten datenschutzfreundlichen Techniken zu berücksichtigen. Dieses Kapitel fasst die wichtigsten Eigenschaften der untersuchten Systeme zusammen. Die zu untersuchenden Systeme wurden so ausgewählt, dass damit ein möglichst größes Spektrum an Übertragungsverfahren evaluiert wird.

In den folgenden Abschnitten wird erläutert, wie sich die einzelnen Systeme für den datenschutzfreundlichen Zugriff auf das Internet nutzen lassen. Dabei wird insbesondere erläutert, wie ein Angreifer mit *tcpdump*[1] die verschlüsselten Datenpakete protokollieren kann. Der Aufruf von tcpdump erfolgt dabei immer mit `tcpdump -n -v -tt -i ethX Filter-String`, wobei *Filter-String* durch den jeweils angegebenen String zu ersetzen ist und *ethX* durch das Netzwerk-Interface, das zum Zugriff verwendet werden soll.[2]

4.1. Single-Hop-Systeme

Bei Single-Hop-Systemen wird ein verschlüsselter Tunnel aufgebaut, wobei sich das eine Tunnel-Ende auf dem Client-Rechner befindet und das andere Ende auf einem Host im Internet, der als Proxy-Server fungiert. Da die Nachrichten nur bis zum Proxy-Server verschlüsselt übertragen werden und dort im Klartext vorliegen, muss der Benutzer dem Betreiber des Proxy-Servers vertrauen. Dies ist in Einklang mit dem in Abschnitt 2.3.3 formulierten Angreifermodell, welches unterstellt, dass der Angreifer keinen Zugriff auf den Proxy-Server hat, sondern die Daten lediglich auf der Leitung zwischen Client und Proxy beobachten kann.

[1] Homepage: http://www.tcpdump.org/

[2] `-n` stellt IP-Adressen ohne Namensauflösung dar, `-v` blendet die Paketlänge ein, `-tt` gibt bei jedem IP-Paket die UNIX-Timestamp mit einer Auflösung im Millisekundenbereich aus.

4.1.1. OpenSSH

OpenSSH[3] ist eine freie Implementierung des SSH-Protokolls [YL06a]. Das SSH-Protokoll wird typischerweise zur sicheren Anmeldung an der Text-Konsole eines entfernten Rechners verwendet. Es beherrscht verschiedene Authentifizierungsverfahren (z. B. Passwörter oder asymmetrische Schlüssel) sowie eine Reihe von Verschlüsselungsverfahren (OpenSSH implementiert 3DES, Blowfish, AES, Arcfour und CAST128[4]). Darüber hinaus enthält das Protokoll Funktionen zur Dateiübertragung und zum *Port-Forwarding*.

Beim *Local-Port-Forwarding* öffnet der SSH-Client auf dem Client-Rechner einen TCP-Port. Alle Daten, die andere Programme auf dem Client-Rechner an diesen Port senden, werden über die verschlüsselte SSH-Verbindung an den SSH-Server übermittelt. Dieser leitet die Daten an einen zuvor festgelegten TCP-Port bei einer festgelegten Ziel-Adresse weiter. Beim *Remote-Port-Forwarding* verhält es sich genau anders herum: der SSH-Server öffnet einen Port und leitet alle dort auflaufenden Daten an den SSH-Client weiter.

Beim *Dynamic Forwarding* fungiert der SSH-Client hingegen als SOCKS-Proxy (vgl. RFC 1928 [LGL+96]). Alle SOCKS-fähigen Programme auf dem Client können diesen Proxy nutzen, um ihre Nachrichten auf der Strecke vom SSH-Client zum -Server verschlüsselt zu übertragen. Da der *Connection Layer* [YL06b] des SSH-Protokolls Multiplexing und Flusskontrolle für mehrere gleichzeitige Verbindungen beherrscht, ist es möglich, ohne spürbare Performance-Einbußen über einen solchen SSH-Tunnel im Internet zu surfen.

Die isolierte Aufzeichnung des SSH-Datenverkehrs zur Analyse des Schutzes vor Website-Fingerprinting ist leicht möglich, da die gesamte Kommunikation über TCP-Port 22 abgewickelt wird. Der passende libpcap-Filter lautet `tcp and port 22`.

4.1.2. OpenVPN

Im Vergleich zu den für die Anwendung nicht-transparenten OpenSSH-Tunneln wird bei einem *Virtual Private Network* (VPN) der datenschutzfreundliche Zugriff auf das Internet völlig transparent realisiert. Nachdem die Verbindung zum VPN-Server hergestellt ist, können alle Anwendungen den verschlüsselten Tunnel verwenden, ohne dass etwas an ihrer Konfiguration geändert werden muss.

OpenVPN[5] ist eine plattformunabhängige, freie Software zur Einrichtung eines VPNs auf Basis des TLS-Protokolls. OpenVPN nutzt hierzu Teile der OpenSSL-Bibliothek[6]. Als Transportprotokoll kann sowohl UDP als auch TCP verwendet werden.

OpenVPN kennt zwei Betriebsarten: *Bridging* und *Routing*. Im Briding-Modus wird ein verschlüsselter Tunnel hergestellt, über den Ethernet-Frames transportiert werden können (Schicht 2 im OSI-Referenzmodell), während der Tunnel im Routing-Modus lediglich IP-Pakete weiterleitet (Schicht 3 OSI-Referenzmodell). Im weiteren Verlauf wird lediglich der Routing-Modus beschrieben, da dessen Einrichtung einfacher ist und für den datenschutzfreundlichen Internetzugriff ausreicht.

Im Routing-Modus wird dem OpenVPN-Client neben seiner tatsächlichen IP-Adresse, die er zur Verbindung ins Internet nutzt, eine virtuelle IP-Adresse zugewiesen (z. B. 10.8.0.2).

[3] Homepage: http://www.openssh.com/
[4] vgl. http://www.openssh.com/faq.html
[5] Homepage: http://www.openvpn.org/
[6] Homepage: http://www.openssl.org/

Der VPN-Server verfügt ebenfalls über eine virtuelle Adresse (z. B. 10.8.0.1). Nachdem die Verbindung aufgebaut ist, kann der Client den Server über seine virtuelle Adresse ansprechen und Dienste auf dem Server nutzen.

Läuft auf dem OpenVPN-Server zusätzlich ein Web-Proxy-Server, kann der Client auf datenschutzfreundliche Weise auf das Internet zugreifen, indem er diesen Proxy-Server nutzt. Der Zugriff auf das Internet über das VPN ist auch ohne Proxy-Server möglich: Hierzu müssen die Weiterleitung von IP-Paketen *(IP-Forwarding)* sowie die Adressübersetzung *(Network Address Translation, NAT)* auf dem VPN-Server aktiviert werden. Auf dem Client-Rechner ist dann die virtuelle Adresse des OpenVPN-Servers als Standardgateway einzutragen.[7]

OpenVPN 2.0 verwendet standardmäßig den UDP-Port 1194. Der zugehörige libpcap-Filter lautet daher `udp and port 1194`.

4.1.3. Stunnel

Stunnel[8] ist ein Konsolenprogramm, das TLS-/SSL-Tunnel aufbaut und anderen Programmen, die nativ keine verschlüsselte Kommunikation unterstützen, zur Verfügung stellt. Auch Stunnel greift hierzu auf die Funktionen der OpenSSL-Bibliothek zurück. Die Funktionsweise ist mit dem Local-Port-Forwarding von OpenSSH vergleichbar: Stunnel öffnet nach dem Start auf dem Client-Rechner einen TCP-Port, der anderen Anwendungen auf dem Client-Rechner zur Verfügung steht. Verbindet sich ein Programm mit diesem Port, baut Stunnel einen Tunnel zu einem vordefinierten Ziel auf, das TLS bzw. SSL unterstützt. Das Programm kann dann mit dem Ziel-Server Daten austauschen, ohne sich um die Verschlüsselung zu kümmern. Unterstützt der anzusprechende Server kein SSL, kann man auf dem Server eine weitere Instanz von Stunnel einrichten, die den verschlüsselten Datenstrom entgegennimmt und im Klartext an einen vordefinierten Port auf dem Ziel-Server weiterleitet.

Zum datenschutzfreundlichen Zugriff auf das Internet ist ein vertrauenswürdiger Proxy-Server erforderlich, der Verbindungen über TLS oder SSL annehmen kann. Hierzu kommt etwa Squid 3.0[9] in Frage, der mit der Option *enable-ssl* übersetzt wurde (was in den Standarddistributionen nicht gegeben ist). Da aktuelle Browser keine verschlüsselten Proxy-Verbindungen unterstützen, kommt auf dem Client-Rechner Stunnel zum Einsatz, um einen „normalen“ Proxy-Server auf dem Client zu simulieren und die Daten verschlüsselt an den tatsächlichen Proxy-Server weiterzuleiten.

Steht keine Proxy-Server-Software zur Verfügung, die TLS unterstützt, ist auf dem Proxy-Server-Host eine weitere Instanz von Stunnel einzurichten, die mit der Client-Instanz kommuniziert und der eigentlichen Proxy-Server-Software den entschlüsselten Datenstrom zur Verfügung stellt. Für die Untersuchung von Stunnel wurde dieses Szenario zu Grunde gelegt, wobei als Proxy-Server tinyproxy[10], der kein Caching der heruntergeladenen Objekte durchführt, zum Einsatz kam.

[7] Verwendet der Client-Rechner DHCP, um die IP-Adresse der physischen Netzwerkschnittstelle zu beziehen, ist darauf zu achten, dass in der Routingtabelle ein Eintrag für den DHCP-Server angelegt wird. Andernfalls kann es passieren, dass der Client plötzlich seine IP-Adresse verliert, da er den DHCP-Server nicht (über das VPN) erreichen kann. Bei einer statisch zugewiesenen IP-Adresse tritt dieses Problem nicht auf.

[8] Homepage: http://stunnel.mirt.net/

[9] Homepage: http://www.squid-cache.org/

[10] Homepage: http://tinyproxy.sourceforge.net/

Stunnel hat einen technisch bedingten Nachteil, der seinen Schutz vor Website-Fingerprinting möglicherweise einschränkt: Im Gegensatz zu OpenSSH und OpenVPN wird beim Start von Stunnel kein permanenter Tunnel aufgebaut, sondern Stunnel verwendet für jede eingehende TCP-Verbindung eine neue TLS-Verbindung. Der Angreifer sieht dadurch die einzelnen Verbindungen zwischen Client und Webserver und kann aus der jeweils übertragenen Datenmenge auf die Größe der HTTP-Requests und -Responses schließen.

Stunnel verwendet wie OpenSSH einen nicht-standardisierten, jedoch festzulegenden Port. Die Aufzeichnung des verschlüsselten Datenstroms ist möglich, wenn man unterstellt, dass der Angreifer das eingesetzte datenschutzfreundliche Verfahren kennt oder ermitteln kann. Der zugehörige libpcap-Filter ist analog zu OpenSSH zu konstruieren.

4.1.4. Cisco IPsec-VPN

IPsec (standardisiert u. a. in RFC 2401 [KA98]) unterstützt zwei Betriebsarten: den *Transport-Mode*, der lediglich die Verbindung zwischen zwei Rechnern absichert, und den *Tunnel-Mode*, der dem Client-Rechner zusätzlich den Zugriff auf ein Netzwerk bzw. das Internet ermöglicht. Im Weiteren wird nur der Tunnel-Mode berücksichtigt, der auch beim Zugriff über das VPN der Universität Regensburg zum Einsatz kommt, das mit Cisco-Komponenten realisiert ist [Rec05]. Der gesicherte Internetzugriff wird über einen VPN-Client auf dem Client-Rechner realisiert, der einen Tunnel zum VPN-Gateway aufbaut. Durch die automatische Anpassung des Standard-Gateways wird nach dem Aufbau der VPN-Verbindung der gesamte Datenverkehr über diesen Tunnel geleitet.

Die wesentlichen Komponenten von IPsec sind zwei Protokolle bzw. ihre zugehörigen IP-Header: *AH (Authentication Header)* und *ESP (Encapsulated Security Payload)*. ESP ist ein eigenständiges Protokoll neben TCP und UDP, das im Tunnel-Mode verwendet wird, um die IP-Pakete zu kapseln und verschlüsselt zu übertragen. Um auch bei einem zwischengeschalteten NAT-Gateway auf ein IPsec-VPN-Gateway zugreifen zu können, wird üblicherweise *NAT-T (NAT Traversal in the IKE)* verwendet, das in RFC 3948 [HSV+05] stadardisiert ist. Bei NAT-T werden die ESP-Pakete in UDP-Paketen gekapselt, so dass auch NAT-Gateways, die nicht mit ESP-Paketen umgehen können, die Adressumschreibung durchführen können. Da weitere Protokolldetails für die Untersuchung irrelevant sind, sei auf die Literatur verwiesen (siehe etwa [Sch07]).

Der Cisco-Client verwendet NAT-T auf UDP-Port 10 000, so dass der zugehörige libpcap-Filter `udp and port 10000` lautet. Auf dem Client-Rechner lassen sich bei dem verwendeten Cisco-VPN allerdings nur die *empfangenen* Pakete aufzeichnen – die gesendeten Pakete sind aus unerklärlichen Gründen nicht auf dem Netzwerk-Interface zu sehen. Es wird sich jedoch zeigen, dass die Evaluation des Cisco-VPNs dadurch nicht beeinträchtigt wird.

4.2. Multi-Hop-Systeme

Single-Hop-Systeme haben den architekturbedingten Nachteil, dass der Benutzer dem Betreiber des Single-Hop-Proxy-Servers vertrauen muss – bei Multi-Hop-Systemen entfällt diese Einschränkung. Hier wird ebenfalls vom Client-Rechner aus ein verschlüsselter Tunnel aufgebaut. Jede Verbindung wird jedoch über mehrere Zwischenstationen weitergeleitet bevor sie den Ziel-Server erreicht.

Die untersuchten Multi-Hop-Systeme basieren auf dem Prinzip von Mixen [Cha81]. Auf dem Client-Rechner läuft ein lokaler Proxy-Server, der durch mehrfach hintereinander ausgeführte Verschlüsselung der Daten sicherstellt, dass kein Mix Sender *und* Empfänger einer Nachricht erfährt.

Die Nachrichten werden über eine Kette von n Mixen weitergeleitet. Zum Abruf einer Webseite chiffriert der lokale Proxy den eigentlichen HTTP-Request mit einem Schlüssel, den lediglich der letzte Mix kennt. Iterativ verschlüsselt er dann den erhaltenen Chiffretext und die Adresse des jeweiligen Empfängers mit einem Schlüssel, den nur dessen Vorgänger kennt. Das Ergebnis sendet er an den ersten Mix. Nur der erste Mix kennt also die Identität (bzw. die IP-Adresse) des Nutzers, nur der letzte Mix erfährt die URL und den Inhalt der abgerufenen Webseite.

4.2.1. JonDonym

JonDonym[11] ist ein kommerzieller Anonymisierungsdienst, der auf dem AN.ON-Projekt der Universitäten Dresden und Regensburg basiert. Das AN.ON-Projekt verfolgt das Ziel, ein Mix-basiertes Anonymisierungssystem zu entwickeln. Hierzu wurde das Konzept der ISDN-Mixe [PPW91] weiterentwickelt und für die Verwendung mit dem HTTP-Protokoll angepasst [BFK00; BFK01]. Der Fokus von AN.ON bzw. JonDonym liegt auf dem datenschutzfreundlichen Web-Zugriff, wobei inzwischen auf einigen Kaskaden auch andere Anwendungsprotokolle zugelassen sind. In der Architektur von JonDonym gibt es vier Kernkomponenten:

1. Die in Java implementierte *JonDo-Client-Software* stellt dem Client-Rechner einen Web-Proxy zur Verfügung.
2. Der Benutzer wählt im JonDo-Client eine Mix-Kaskade aus, die zum datenschutzfreundlichen Internetzugriff verwendet werden soll. Dabei besteht die Wahl zwischen kostenlosen und kommerziellen Kaskaden. Jede *Mix-Kaskade* besteht aus mehreren Mixen, die von unterschiedlichen Betreibern gehostet und betreut werden. Der letzte Mix leitet die Verbindungen an einen Cache-Proxy (Squid) weiter. Von dort aus werden die HTTP-Requests an die Ziel-Webserver geschickt.
3. Der *InfoService* sammelt Statusinformationen über Verfügbarkeit und Verkehrsaufkommen der einzelnen Mix-Kaskaden und stellt sie dem JonDo-Client zur Verfügung.
4. Die *Bezahl-Instanz* verwaltet das Guthaben der Benutzer, die kommerzielle Kaskaden verwenden wollen. Eine theoretische Beschreibung der Architektur des Bezahlsystems liefert [KM03].

Der datenschutzfreundliche Zugriff auf das Internet gestaltet sich bei JonDonym vergleichsweise einfach. Der Benutzer muss lediglich den JonDo-Client starten und im Browser als Proxy-Server eintragen. Die Geschwindigkeit der kostenlosen Kaskaden ist wegen der großen Benutzeranzahl häufig gering. Die Auslastung der kommerziellen Kaskaden ist deutlich niedriger – im Normalfall nimmt man bei ihnen keine Performance-Einbußen wahr. Für die Evaluation von JonDonym wurde daher eine kommerzielle Kaskade verwendet.

[11] Homepage: http://www.jondonym.de/

Für die Aufzeichnung von JonDonym-Traffic sind etwas kompliziertere Filter-Regeln erforderlich als bei Single-Hop-Systemen, da die Kommunikation zwischen JonDo und dem ersten Mix über verschiedene Ports und IP-Adressen stattfinden kann. Um den gesamten JonDonym-Datenverkehr zu erfassen, muss der Angreifer einen Filter konstruieren, der alle Zugangspunkte (erste Mixe) erfasst. Der Filter hat folgenden Aufbau: `tcp and host mix1.cascade1.jondonym.de and port 443 or tcp and host mix1.cascade2.jondonym.de and port 8080 ...`

Bei JonDonym werden die Nutzdaten in so genannten Mix-Paketen mit einer festen Größe von 998 Byte transportiert [FKL02]. In ein IP-Paket passt bei einer MTU von 1500 Byte also ein ganzes Mixpaket und der Anfang eines weiteren Pakets. Für den Test stand Version 00.09.010 des JonDo-Clients zur Verfügung. Um zu vermeiden, dass die Aufzeichnung des Datenverkehrs von Paketen beeinflusst wird, die nichts mit der übertragenen Webseite zu tun haben, wurde die Kommunikation mit dem Infoservice deaktiviert und eine (kommerzielle) Kaskade statisch ausgewählt. Die Messungen von JonDonym erfolgen also unter idealisierten Bedingungen, weshalb die ermittelten Erkennungsraten geringfügig *höher* sind als in der Praxis.

Der JonDo wurde so konfiguriert, dass er alle sechs Sekunden ein Dummy-Datenpaket erzeugte – das ist der minimal einstellbare Zeitabstand; diesen würde auch ein sicherheitsbewusster JonDo-Nutzer wählen.

4.2.2. Tor

Das Tor-Netzwerk[12] basiert auf dem Prinzip des Onion Routings [GRS96]. Es handelt sich dabei um ein Overlay-Netzwerk, welches das Ziel verfolgt, die Sender- und Beziehungsanonymität für beliebige TCP-basierte Internetdienste zu gewährleisten. Im Gegensatz zu Mix-Kaskaden ist hier die Reihenfolge der zu verwendenden Zwischenstationen nicht statisch festgelegt. Die Zwischenstationen werden erst zur Laufzeit dynamisch ausgewählt. In manchen Publikationen wird Tor deshalb als Mix-Netz bzw. *mix-net* bezeichnet (etwa in [Ray01]).

Nach Dingledine et al. hat Tor aus der Sicht des Benutzers drei wesentliche Komponenten [DMS04]:

1. Auf dem Client-Rechner läuft die Tor-Client-Software, die den Zugang zum Tor-Netz herstellt und die Ver- bzw. Entschlüsselung der Verbindungen übernimmt. Der Tor-Client stellt den Anwendungen einen lokalen SOCKS-Proxy-Server zur Verfügung. Die Tor-Software wird daher auch als *Onion-Proxy* bezeichnet.
2. Das Tor-Netzwerk besteht aus einer Vielzahl von *Onion-Routern*, die ebenfalls die Tor-Software ausführen.
3. In regelmäßigen Abständen laden die Onion Proxies eine Liste der aktuell verfügbaren Onion Router von redundant ausgelegten *Directory-Servern* herunter.

Der Onion Proxy konstruiert nach dem Start so genannte *circuits*, indem er mehrere (üblicherweise drei) Onion Router auswählt und miteinander verkettet. Zunächst stellt er eine TLS-Verbindung zum ersten Onion-Router her. Darüber baut er weitere TLS-Verbindungen zu anderen Onion Routern auf, bis die gewünschte *circuit length* erreicht

[12] Homepage: http://tor.eff.org/

ist. Am Ende dieses Kanalaufbauvorgangs hat der Onion-Proxy mit jedem der Router einen geheimen Sitzungsschlüssel vereinbart, und es existiert ein mehrfach verschlüsselter Kanal zwischen dem Onion-Proxy und dem letzten Onion-Router, der mit dem Zielserver kommuniziert.

Bei der Verwendung von Tor zum datenschutzfreundlichen Zugriff auf Internetseiten sind einige Besonderheiten zu beachten. So raten die Entwickler etwa davon ab, den Tor-Client direkt als SOCKS-Proxy im Browser einzutragen: Viele Browser verwenden zur Namensauflösung in diesem Fall den lokalen DNS-Server und geben damit die Host-Namen der abgerufenen Seiten preis. Stattdessen empfehlen die Entwickler, zusätzlich einen lokalen Proxy-Server (etwa Privoxy[13]) zu installieren und diesen als HTTP-Proxy-Server im Browser einzutragen. Seit einiger Zeit bieten die Tor-Entwickler ein Software-Paket bestehend aus vier Client-Komponenten zum Download an: *Tor*, *Privoxy*, *Vidalia* und *Tor Button*.

Wie bei JonDonym ist die zuverlässige Aufzeichnung von Tor-Traffic nicht trivial. Durch die große Anzahl von Onion Routern, die Verbindungen auf verschiedensten TCP-Ports entgegennehmen, ist der Aufwand zur Erstellung einer libpcap-Filterregel hoch. Man kann den Tor-Client zwar – analog zur Vorgehensweise bei JonDonym – zwingen, einen festgelegten *Entry-Node* zu verwenden, dies ist aber aus praktischen Gründen nicht sinnvoll: Im Gegensatz zur hohen Verfügbarkeit der kommerziellen Kaskaden von JonDonym sind einzelne Onion-Router häufig nicht zuverlässig erreichbar, was vor allem bei der Evaluation über einen Zeitraum von mehreren Tagen zum Problem werden kann. Darüber hinaus wäre der aufgezeichnete Datenverkehr wohl kaum für das gesamte Tor-Netzwerk repräsentativ, sondern zumindest teilweise von dem gewählten *Entry-Node* abhängig. Tor wird mit zwei verschiedenen Methoden untersucht:

1. Der Tor-Client wird angewiesen, für alle Verbindungen einen Proxy-Server (tinyproxy) zu verwenden. Der Datenverkehr zwischen dem Tor-Client und dem Proxy-Server lässt sich dann anhand des eindeutigen Proxy-Server-Ports leicht herausfiltern (analog zum Aufzeichnen des Datenverkehrs bei Stunnel).

2. Der Tor-Client wird angewiesen, sich nur mit Onion-Routern zu verbinden, die einen festgelegten Port verwenden. Dabei ist sicherzustellen, dass während der Aufzeichnung keine andere Anwendung auf dem Client-Rechner diesen Port für ihre Zwecke verwendet. Hierzu ist etwa der Port 9001, auf dem viele Onion-Router laufen, geeignet.

Die Datenpakete von Tor *(cells)* haben eine feste Größe von 512 Byte. In einem IP-Paket können bei einer MTU von 1500 also maximal zwei ganze *cells* sowie der Anfang einer dritten *cell* transportiert werden. Für die Untersuchung wurden Tor v0.1.2.17, Privoxy v3.0.6 sowie tinyproxy v1.6.3 verwendet.

Der Datenverkehr zu den Directory-Servern geht nicht in die Messung ein (da dieser über andere TCP-Ports abgewickelt wird). Die Nachrichten des Tor-Kontrollprotokolls, die im Hintergrund etwa zum Aufbau von *circuits* zwischen den Onion-Routern ausgetauscht werden, sind im aufgezeichneten Datenverkehr allerdings enthalten. Ein weiterentwickeltes Verfahren könnte versuchen, diese Nachrichten zu erkennen und aus dem Datenverkehr nachträglich wieder zu entfernen.

[13]Homepage: http://www.privoxy.org/

4.2.3. Shalon

Shalon (Scalable HTTP-based Anonymisation Lightweight Overlay Network) wurde im Rahmen der Diplomarbeit von Benedikt Westermann an der RWTH Aachen entwickelt. Es handelt sich dabei um ein Anonymisierungssystem, das auf der Verkettung von Proxy-Servern über verschachtelte TLS-Verbindungen basiert [Wes08]. Shalon wurde im Hinblick auf folgende Anforderungen entwickelt [Wes08, 33 ff.]: geringe Komplexität, Verwendung standardisierter Protokolle, kurze Latenzzeiten, hohe Transferraten, Skalierbarkeit und Schutz gegen lokale Angreifer.

Auf dem Client-Rechner wird die Shalon-Client-Software ausgeführt, die als lokaler HTTP- und SOCKS-Proxy-Server fungiert. Die Client-Software konstruiert nach dem Start im Hintergrund eine so genannte *Proxykette*, die aus mehreren Squid-Proxy-Servern (mit einkompilierter SSL-Unterstützung) besteht. Der Client baut Schritt für Schritt mit der HTTP-CONNECT-Methode eine TLS-gesicherte Verbindungen über die einzelnen Proxies auf.

Bei drei Proxies wird die Kette wie folgt konstruiert: Der Client verbindet sich mit dem ersten Proxy-Server und durchläuft den TLS-Handshake, so dass der Datenaustausch mit diesem Proxy verschlüsselt wird. Dann veranlasst er den ersten Proxy mit einem CONNECT-Request, eine Verbindung zum zweiten Proxy-Server aufzubauen. Auch die Verbindung zum zweiten Proxy-Server wird mittels TLS abgesichert. Nun kann der Client mit dem zweiten Proxy-Server kommunizieren, ohne dass der erste Proxy-Server die Klartexte sieht. Nach demselben Schema wird die Proxy-Kette ein weiteres Mal verlängert. Der dritte Proxy-Server baut dann die Verbindung zum Webserver auf.

Der Shalon-Client ermittelt die zu verwendenden Proxy-Server über eine Distributed Hashtable (DHT). Da die Funktionsweise der DHT für die Evaluierung von Website-Fingerprinting ohne Belang ist, wird an dieser Stelle nicht weiter darauf eingegangen.

Zur Evaluierung von Shalon wurde der Shalon-Client im Browser als HTTP-Proxy eingetragen. In diesem Betriebsmodus fungiert der letzte Proxy-Server in einer Kette zusätzlich als Cache-Proxy (ähnlich wie bei JonDonym). Der Shalon-Client wurde so konfiguriert, dass er einen festgelegten Entry-Node verwendet. Die Aufzeichnung des Datenverkehrs erfolgt damit wie bei Tor.

5 Vorgehensweise bei der Analyse

Das Website-Fingerprinting, das zum Vergleich der datenschutzfreundlichen Übertragungstechniken verwendet wird, basiert auf Data-Mining-Techniken. Dieses Kapitel erläutert den Kontext, in dem das Fingerprinting-Verfahren zum Einsatz kommt.

Nach der Formulierung der zentralen Arbeitshypothesen folgt eine kurze Beschreibung der relevanten Data-Mining-Konzepte. Im Kern des Kapitels wird ein geeignetes Datenverkehrsmerkmal für den Website-Fingerprinting-Angriff ausgewählt. Die Beschreibung der Vorgehensweise zur automatischen Erzeugung von Testdaten sowie wichtige Aspekte, die bei der Evaluierung von Data-Mining-Verfahren zu beachten sind, rundet das Kapitel ab. Abbildung 5.1 zeigt die Vorgehensweise im Überblick.

5.1. Arbeitshypothesen

Zur gezielten Analyse der datenschutzfreundlichen Systeme hinsichtlich ihres Schutzes gegen Website-Fingerprinting ist es erforderlich, die Komplexität der Problemstellung zu reduzieren. Hierzu werden in Anlehnung an [LL06] folgende Arbeitshypothesen aufgestellt:

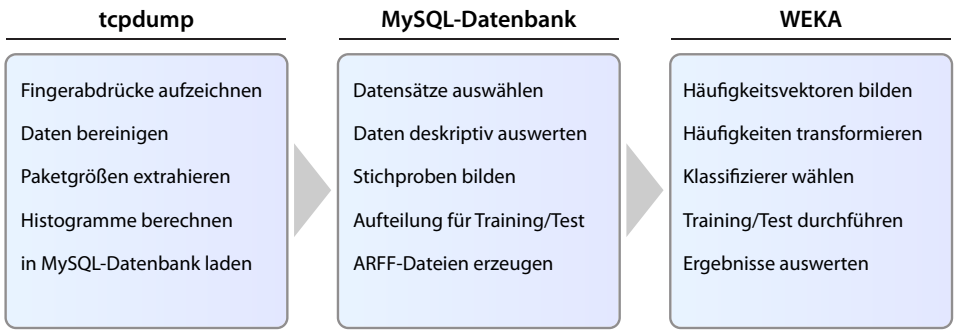


Abbildung 5.1.: Die Auswertung der Daten erfolgt in drei Phasen

Browser-Caching deaktiviert Es ist anzunehmen, dass die Analyseergebnisse bei aktiviertem Browser-Cache nur eingeschränkt reproduzierbar wären, da die heruntergeladenen Objekte dann erheblich davon abhingen, welche Objekte sich bereits im Browser-Cache befinden und wie viel Zeit zwischen den einzelnen Abrufen verstrichen ist. Daher wird unterstellt, dass das Opfer den Browser-Cache deaktiviert hat, so dass bei jedem Webseitenabruf alle eingebetteten Elemente heruntergeladen werden. In Abschnitt 9.1 wird diese Arbeitshypothese gelockert, um den Einfluss des Browser-Caches auf die Erkennungsraten zu untersuchen.

Angreifer kann einzelne Abrufe im Datenverkehr isolieren Es wird unterstellt, dass die Pause zwischen dem Abruf von zwei Webseiten größer ist als die Zeitabstände zwischen den IP-Paketen, die während des Abrufs übertragen werden. Diese Hypothese lässt sich durch die Denkpausen der Benutzer rechtfertigen.

Angreifer verwendet den gleichen Browser wie das Opfer Diese Arbeitshypothese ist erforderlich, da davon auszugehen ist, dass verschiedene Browser unterschiedliche Strategien zum Herunterladen von Webseiten verwenden. Die Analyseergebnisse könnten dadurch unkontrolliert beeinflusst werden. Der Angreifer soll daher in der Lage sein, auf seinem Rechner den gleichen Browser mit derselben Konfiguration zu verwenden wie das Opfer.

Angreifer kennt das eingesetzte Übertragungsverfahren Es ist davon auszugehen, dass verschiedene Übertragungsverfahren (etwa JonDonym und Tor) den Datenverkehr erheblich verändern. In Anlehnung an die Kryptanalyse soll unterstellt werden, dass der Angreifer das vom Opfer verwendete datenschutzfreundliche Übertragungsverfahren kennt und auch in der Lage ist, es selbst zu verwenden.

Die abgerufenen Webseiten sind beschränkt und dem Angreifer bekannt In Anlehnung an das informationstheoretische Modell aus Abschnitt 2.2 soll das Opfer nur Webseiten aus einer festgelegten (jedoch beliebig großen) Menge abrufen. Der Angreifer soll die in Frage kommenden Seiten kennen und in der Lage sein, bereits vor dem eigentlichen Angriff alle Webseiten ein- oder mehrmals abzurufen, um Trainingsdaten zu erzeugen. Diese Arbeitshypothese erleichtert die folgenden Erläuterungen. In Abschnitt 9.2 wird untersucht, inwiefern Website-Fingerprinting möglich ist, wenn diese Vereinfachung aufgehoben wird.

Angreifer hat Zugriff auf einen ähnlichen Internetzugang wie das Opfer Die technischen Eigenschaften unterschiedlicher Internetzugänge beeinflussen möglicherweise den tatsächlichen Datenverkehr. So haben ISDN- und DSL-Anschlüsse zum Beispiel unterschiedliche *Maximum Transmision Units (MTU)* bzw. Latenz-Zeiten. Zur Vereinfachung des Angriffs soll der Angreifer daher in der Lage sein, die Fingerabdrücke vor dem Angriff über den gleichen Internetzugang abzurufen wie das Opfer oder über einen Zugang, der aus technischer Sicht ähnlich ist.

5.2. Verwendete Data-Mining-Konzepte

Bei der Diskussion der bislang veröffentlichten Website-Fingerprinting-Verfahren in Kapitel 3 zeigte sich, dass Website-Fingerprinting-Verfahren, die auf einfachen statistischen Methoden (z. B. dem Korrelationskoeffizienten) basieren, schlechter abschneiden als Verfahren, die sich Data-Mining-Techniken bedienen. Das für die empirische Untersuchung entwickelte Verfahren basiert ebenfalls auf Data-Mining-Techniken. Die folgenden Abschnitte geben einen Überblick über die relevanten Data-Mining-Konzepte.

5.2.1. Definition und Zielsetzung

Es gibt eine Reihe von unterschiedlichen Definitionen und Interpretationen des Begriffs „Data Mining“. Die folgenden Ausführungen orientieren sich an Petersohns Begriffsbestimmung [Pet05, 8 f.]. Die Autorin referenziert eine Definition aus einem Artikel in der Zeitschrift *Wirtschaftsinformatik*: Beim Data Mining werden demnach Methoden aus Statistik, Künstlicher Intelligenz, Maschinellem Lernen und Mustererkennung verwendet, um relevante Muster in großen Datenbeständen zu identifizieren und sie dem Anwender zu präsentieren [HBM97].

Im weiteren Verlauf verweist Petersohn auf die Autoren Frawley, Piatetsky-Shapiro und Matheus, die sie als Wegbereiter des modernen Data Mining beschreibt [Pet05, 8 f.]. Als primäres Data-Mining-Ziel definiert das Trio “knowledge discovery”, also die nicht-triviale Extraktion von implizit vorhandenen Informationen aus Daten [FPSM91, 58].

5.2.2. Der KDD-Prozess nach Fayyad et al.

Fayyad, Piatetsky-Shapiro und Smyth erweitern das ursprüngliche Data-Mining-Konzept und definieren einen Prozess für “Knowledge Discovery in Databases”[1] (KDD), in den das eigentliche Data Mining eingebettet ist:

> “KDD refers to the overall process of discovering useful knowledge from data, and data mining refers to a particular step in this process. Data mining is the applications of specific algorithms for extracting patterns from data.” [FPSS96, 39]

Der KDD-Prozess ist nach Auffassung von Fayyad et al. sowohl iterativ als auch interaktiv, wobei in jeder Iteration andere Data-Mining-Techniken auf die Daten angewendet werden. In Abbildung 5.2 ist der typische Prozessablauf skizziert.

Die Autoren beschreiben neun Phasen [FPSS96, 42], die in jedem Data-Mining-Projekt zu durchlaufen sind: Nachdem das Ziel formuliert wurde (1), wird der zu analysierende Datensatz ausgewählt (2). Daran schließen sich Vorverarbeitung und Säuberung des ausgewählten Datensatzes (3) sowie Reduktion und Transformation der Rohdaten (4) an. Bevor das eigentliche Data Mining beginnt, ist die passende Data-Mining-Methode (z. B. Klassifizierung) zu bestimmen (5) und ein passender Algorithmus (z. B. Naïve-Bayes-Klassifizierer) zu wählen (6). Der ausgewählte Algorithmus wird dann verwendet, um das

[1] “Databases” wird hier nicht im Kontext von Datenbanksystemen, sondern im Kontext des Maschinellen Lernens verwendet. Eine Database ist dort eine Menge von Instanzen, die mittels Data-Mining-Techniken zu analysieren ist.

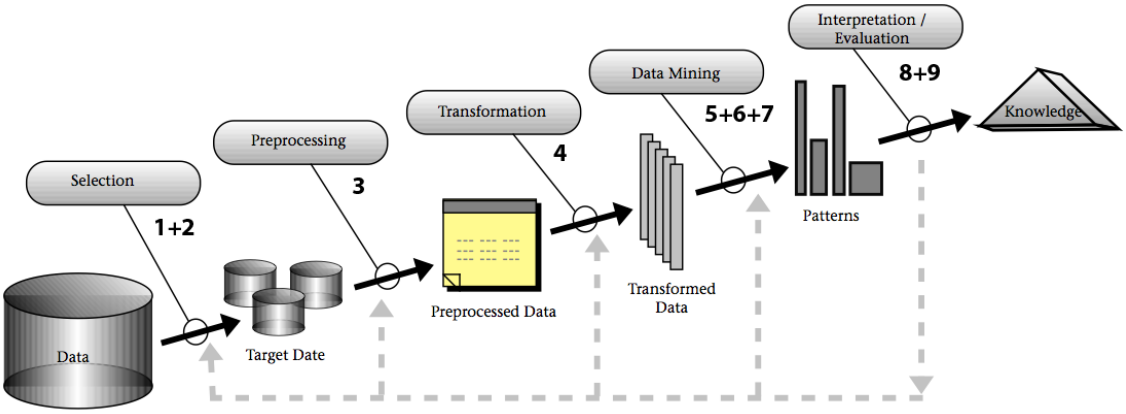


Abbildung 5.2.: Phasen im KDD-Prozess (Darstellung aus [FPSS96, 41])

eigentliche Data Mining, die Suche nach interessanten Mustern und Konzepten, durchzuführen (7). Danach sind die Ergebnisse zu evaluieren (8), und abschließend können die gewonnenen Erkenntnisse verwendet werden (9).

Die bisherigen Publikationen über Website-Fingerprinting konzentrieren sich auf die Phasen 6, 7 und 8, vernachlässigen jedoch die übrigen Phasen. In Abschnitt 6.3 wird sich zeigen, dass vor allem der Vorverarbeitung und Transformation der Daten eine hohe Bedeutung zukommt.

5.2.3. Website-Fingerprinting als Klassifizierungsproblem

Fayyad et al. unterscheiden verschiedene Data-Mining-Methoden, darunter Klassifizierung und Clustering [FPSS96, 44]. Bei der Klassifizierung ist ein *Klassifizierer* (manchmal auch: *Klassifikator*) gesucht, der in der Lage ist, unbekannte Daten einer vordefinierten Klasse zuzuordnen. In der Terminologie des Maschinellen Lernens gehören Klassifizierungsprobleme zum Gebiet *supervised learning*. Clustering ist hingegen ein Verfahren aus dem Gebiet *unsupervised learning*. Im Gegensatz zur Klassifizierung sind beim Clustering vor der Analyse noch keine Klassen bekannt, die Daten sollen stattdessen so gruppiert werden, dass die Datensätze in einem Cluster möglichst homogen sind, während die verschiedenen Cluster möglichst heterogen sein sollten.

Im Folgenden wird gezeigt, dass es sich beim Website-Fingerprinting um ein Klassifizierungsproblem handelt. Hierzu wird die Terminologie des *supervised learning* eingeführt (nach [WF05, 43 ff.]).

Der Datenverkehr, der beim Download einer Webseite entsteht, ist eine *Instanz*, die eindeutig einer *Klasse* (hier: der URL der Webseite) zugeordnet ist. Jede Instanz hat eine Menge von *Attributen* und *Attributwerten*. Die Auswahl geeigneter Attribute (engl. *feature selection*) und die Repräsentation ihrer Ausprägungen ist von hoher Bedeutung für die Erkennungsrate des Klassifizierungsverfahrens (Phase 3 im KDD-Prozess). Auf eine geeignete Instanzrepräsentation wird in Abschnitt 6.1 eingegangen. Stehen die zu verwendenden Attribute fest, werden die Attributwerte durch Transformation der aufgezeichneten Rohdaten ermittelt (Phase 4).

Ein Klassifizierungsproblem lässt sich – in Anlehnung an [MRS07, 256 f.] – wie folgt formalisieren. Gegeben sei eine Menge von Trainingsinstanzen $I_{\text{train}} = \{(\vec{x}_1, c_1), \ldots, (\vec{x}_n, c_n)\}$.

Jede Instanz $(\vec{x}_i, c_i) \in I_{\text{train}} \times \mathcal{C}$ ist ein Tupel bestehend aus einem (meist hochdimensionalen) Attributvektor $\vec{x}_i$ und dem zugehörigen Klassenlabel c_i. Darüber hinaus gibt es eine Menge von Testdaten I_{test}, deren Klassenlabel unbekannt ist. Es gilt einen Klassifizierer k: $\mathcal{X} \rightarrow \mathcal{C}$ zu bilden, der den Raum der Attributvektoren $\mathcal{X}$ auf den Raum der Klassenlabels $\mathcal{C}$ abbildet; k soll eine vorgelegte Testinstanz $i \in I_{\text{test}}$ auf ihr Klassenlabel $c \in \mathcal{C}$ abbilden.

Website-Fingerprinting mit einem Klassifizierer Website-Fingerprinting-Angriffe bestehen beim Einsatz eines Klassifizierers also aus zwei Schritten: Zunächst werden Trainingsdaten für alle Seiten aufgezeichnet, an deren Identifikation der Angreifer interessiert ist. Das Training erfolgt vor dem eigentlichen Angriff, also offline. Der eigentliche Angriff besteht darin, den Datenverkehr des Opfers aufzuzeichnen und vom Klassifizierer nach Mustern durchsuchen zu lassen, die im Modell enthalten sind. Der Angriff kann dabei entweder in Echtzeit (online) oder – wie bei der vorliegenden Untersuchung – auf Basis von abgespeichertem Datenverkehr (offline) erfolgen.

5.2.4. Systematische Evaluation von Klassifizierern

Bei der Verwendung von Klassifizierungsverfahren gibt es üblicherweise zwei Phasen: In der Trainingsphase lernt der Klassifizierer ein Modell der Trainingsdaten, in der Testphase verwendet er dieses Modell, um vorherzusagen, zu welcher Klasse eine ihm vorgelegte Instanz gehört. Die Ermittlung der Güte des erlernten Modells bezeichnet man als *Evaluation* [WF05, 144 ff.].

Zur Evaluation eines Klassifizierers bzw. zum Trainieren und Evaluieren eines Modells sind *vorklassifizierte Instanzen* erforderlich, mit denen der Klassifizierer trainiert und getestet werden kann. Bei vielen Data-Mining-Problemen ist die Erzeugung solcher Testdaten äußerst zeitaufwändig, da die vorklassifizierten Instanzen von einem Fachkundigen manuell den richtigen Klassen zugeordnet werden müssen.

Beim Website-Fingerprinting ist die Vorklassifizierung der Testdaten hingegen unproblematisch: Durch den skriptgesteuerten, vollautomatischen Abruf von Internetseiten – bei gleichzeitiger Protokollierung des Datenverkehrs – ist es möglich, den Datenverkehr einer Vielzahl von Webseiten ohne manuelle Eingriffe aufzuzeichnen. Der beim Abruf einer Webseite entstehende Datenverkehr wird dazu in einer Datei gespeichert, wobei die URL der Seite im Dateinamen oder in der Datei selbst hinterlegt wird. In Abschnitt 5.4 wird die Erzeugung der vorklassifizierten Testdaten im Detail beschrieben.

Die einfachste Möglichkeit zum Testen eines Klassifizierers besteht darin, zur Evaluation in der Testphase dieselben Instanzen zu verwenden wie in der Trainingsphase. Die Fehlerrate einer solchen Evaluation ist jedoch viel zu optimistisch und hat daher nur eine relativ geringe Aussagekraft für die Güte des antrainierten Modells: Der Klassifizierer ist durch das Training gewissermaßen optimal auf die Trainingsdaten zugeschnitten. Dieses Phänomen wird in der Literatur als *resubstitution error* bezeichnet [WF05, 145].

Für eine aussagekräftigere Evaluation ist also sicherzustellen, dass sich Trainings- und Testinstanzen nicht überlappen. Die Menge der vorklassifizierten Instanzen wird dazu in zwei disjunkte Teilmengen mit Trainings- und Testinstanzen aufgeteilt *(split)*. Die Testinstanzen werden dem Klassifizierer beim Trainieren vorenthalten und erst im Testdurchlauf zur Klassifizierung vorgelegt (sog. *Holdout-Verfahren* [WF05, 146]).

Für eine realitätsnahe Evaluation von Website-Fingerprinting-Verfahren ist ein rein zufälliger Split unangebracht. Der Split sollte berücksichtigen, dass zwischen dem Aufnehmen der Trainingsdaten und dem eigentlichen Angriff eine gewisse Zeit verstrichen ist. Das Evaluationsverfahren, das zur Analyse der Testdaten verwendet wurde (vgl. Abschnitt 5.5.1), berücksichtigt diese Anforderung und stellt gleichzeitig sicher, dass die Mengen *stratifiziert* sind, d.h. dass in jeder Stichprobe für alle Klassen die gleiche Anzahl von Trainings- und Testinstanzen vorhanden ist [WF05, 149].

5.3. Charakteristische Merkmale des Datenverkehrs

Website-Fingerprinting basiert auf der Analyse spezifischer Merkmale des Datenverkehrs. Die bislang veröffentlichten Untersuchungsergebnisse (vgl. Kapitel 3) legen nahe, dass die Güte eines Website-Fingerprinting-Verfahrens erheblich von der Wahl des zu analysierenden Merkmals abhängt.

Bei der Untersuchung der in Frage kommenden Merkmale wird unterstellt, dass ein datenschutzfreundliches Übertragungsverfahren zum Einsatz kommt, das die Daten über einen dauerhaft aufgebauten Tunnel überträgt (wie bei [BLJL05; LL06]). Bis auf Stunnel trifft dies auf alle getesteten Übertragungstechniken zu. Da beim Abruf einer Webseite häufig mehrere Verbindungen gleichzeitig aufgebaut werden, hat der Angreifer keine Möglichkeit, die Größe der HTML-Seiten und der darin eingebetteten Objekte zu rekonstruieren. Er kann lediglich aggregierte Merkmale analysieren oder die einzelnen IP-Pakete auswerten.

5.3.1. Aggregierte Merkmale

Übertragenes Datenvolumen

Beim Abruf einer Webseite werden HTTP-Requests gesendet und HTTP-Responses empfangen (vgl. RFC 2616 [FGM$^+$99, Abschnitt 1.4]). Das übertragene Datenvolumen in Senderichtung (vom Client zum Server) ergibt sich aus der Größe aller HTTP-Requests, in Empfangsrichtung aus der Größe aller HTTP-Responses. Beim Abruf der Seite *www.google.com* werden zum Beispiel 14 944 Byte empfangen und 3260 Byte gesendet.[2]

Das übertragene Datenvolumen hängt von einer Reihe von Faktoren ab: In der *Empfangsrichtung* wirken sich primär inhaltliche Änderungen einer Seite auf die Datenmenge aus. *Caching* von Browser oder Proxy-Servern können zu einer signifikanten Verringerung des übertragenen Datenvolumens führen. Datenkompression auf dem Transportweg hat ebenfalls Auswirkungen auf das Volumen (etwa beim Einsatz des Content-Encodings *gzip*, vgl. RFC 1952 und 2616 [Deu96; FGM$^+$99, section 14.11]).

Die *gesendete* Datenmenge hängt teilweise von der empfangenen Datenmenge ab: Der Client muss die empfangenen Pakete in regelmäßigen Abständen mit einem TCP-Frame mit gesetztem ACK-Flag bestätigen [CK74]. Einen größeren Einfluss auf das gesendete Datenvolumen dürfte allerdings die Anzahl der in eine HTML-Seite eingebetteten Elemente haben: Für jedes Element muss ein HTTP-Request gesendet werden. Demnach sind vor

[2] Diese Werte wurden mit Firefox 2.0.7 unter Ubuntu Linux 7.10 am 22.11.2007 um 16:52 Uhr ermittelt. In die Messung flossen alle Pakete ein, die über den verwendeten OpenSSH-Tunnel auf Port 22 transportiert wurden. Die Messung erfolgte über das Datennetz der Universität Regensburg.

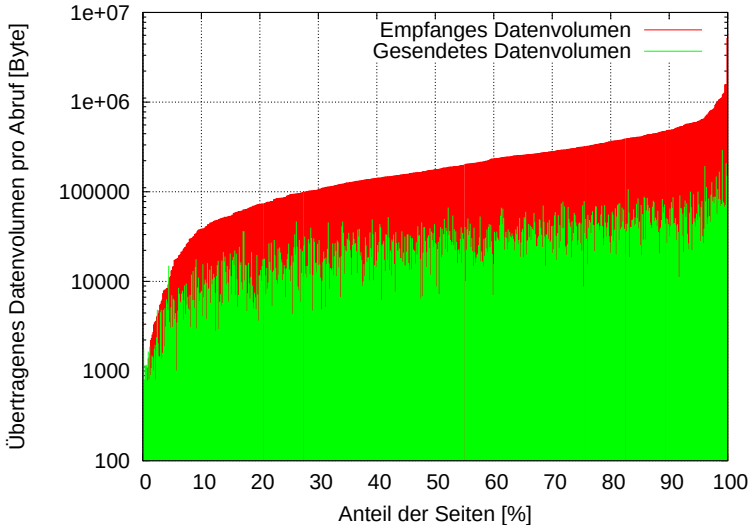


Abbildung 5.3.: Empfangenes/gesendetes Datenvolumen von 775 Webseiten (geordnet nach empfangenem Volumen) beim Abruf über OpenSSH

allem Anzahl und Art der HTTP-Header (abhängig von Browser und Webserver) sowie übermittelte Cookies und Formulardaten für die gesendete Datenmenge von Bedeutung.

Da das übertragene Datenvolumen eine stark aggregierte Kennzahl ist, erscheint sie für die Erstellung von Fingerabdrücken nur bedingt geeignet. Seiten mit völlig unterschiedlichen Inhalten können eine ähnliche oder sogar identische Gesamtgröße aufweisen, so dass eine trennscharfe Unterscheidung anhand ihrer Fingerabdrücke nicht möglich ist. Diese Problematik lässt sich bereits im verwendeten OpenSSH-Datensatz erkennen, in dem sich 8 der 775 Webseiten nicht eindeutig anhand des aggregierten Datenvolumens identifizieren lassen.[3] Es ist zu erwarten, dass die Anzahl der Kollisionen bei größeren Mengen von Webseiten erheblich steigt. Zudem ist die Robustheit des aggregierten Volumens gering: Bereits durch kleine Änderungen an der Seite kann sich das Gesamtvolumen erheblich verändern.

In Abbildung 5.3 ist die Verteilung der Datenvolumen für die 775 URLs aus dem OpenSSH-Datensatz dargestellt. Zur Visualisierung wurden die Seiten nach empfangenem Datenvolumen aufsteigend sortiert und der Reihe nach abgebildet. Die Darstellung ist wegen der großen Schwankungsbreite der Datenvolumen logarithmiert. Die beobachteten empfangenen Datenvolumen variieren zwischen 680 Byte und 5 925 880 Byte, wobei bei etwa 80 % der Seiten zwischen 20 000 und 500 000 Byte empfangen werden. Das gesendete Datenvolumen ist erheblich niedriger, wobei erkennbar ist, dass es zu einem gewissen Grad von der empfangenen Datenmenge abhängt.

Benötigte Zeit für den Abruf einer Webseite

Auch die benötigte Übertragungszeit ist nur eingeschränkt zur Erstellung von Fingerabdrücken geeignet, da sie von vielen Faktoren beeinflusst wird, die nicht mit der Webseite in Verbindung stehen. Schwankungen der verfügbaren Bandbreite, die Auslastung der Internetverbindung sowie die Belastung des Webservers können zu zusätzlichen, zufälligen

[3] Beispiel: Beim Abruf von www.vflfrohnlach.de und www.fh-erfurt.de über OpenSSH wurden jeweils exakt 148 844 Byte empfangen.

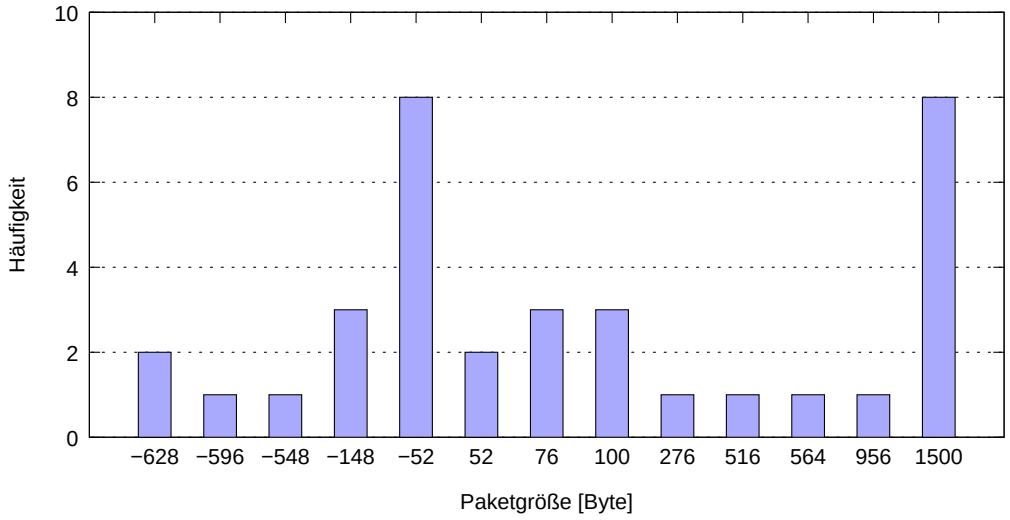


Abbildung 5.4.: Paketgrößen-Histogramm für Abruf von *www.google.com* über SSH-Tunnel (Empfangsrichtung: positive Paketgrößen, Senderichtung: negative Paketgrößen)

Latenzen führen, welche die wahre Übertragungszeit verschleiern. Selbst unter guten Bedingungen ist die Aussagekraft dieses Merkmals relativ gering, da sich bei exakt gleicher Übertragungsdauer völlig unterschiedliche Inhalte übermitteln lassen.

5.3.2. Merkmale basierend auf einzelnen IP-Paketen

Da aggregierte Merkmale in vielen Fällen keine eindeutige Identifizierung von Webseiten erlauben, werden im Folgenden die IP-Pakete betrachtet, die beim Abruf einer Webseite zwischen Client und Webserver ausgetauscht werden.

Größe und Häufigkeit

Gruppiert man die IP-Pakete, die während des Seitenabrufs übertragen werden, der Größe nach, kann man ein Paketgrößen-Histogramm erstellen, das die Häufigkeitsverteilung der Paketgrößen visualisiert. Ein solches Paketgrößen-Histogramm für den Abruf von *www.google.com* aus dem Netzwerk der Universität Regensburg ist in Abbildung 5.4 dargestellt. Zur effizienten Kodierung der Richtung wird die Paketgröße bei Paketen in Senderichtung (vom Client zum Server) mit negativem Vorzeichen versehen (vgl. auch [WMM06b], wobei hier allerdings die *empfangenen* Pakete ein negatives Vorzeichen tragen).

Die Größe und Anzahl der übertragenen IP-Pakete hängt eng mit dem übertragenen Datenvolumen zusammen. Die maximale Paketgröße wird durch die *maximum transmission unit (MTU)*, die auf dem Übertragungsweg zwischen Browser und Server möglich ist, bestimmt. Die MTU hängt von der Art des Internetzugangs ab: bei DSL-Zugängen sind Werte zwischen 1400 und 1500 Byte üblich. Beim Zugang über das Netzwerk der Universität Regensburg wird das Maximum von 1500 Byte verwendet.

Die Verteilung der Paketgrößen beim Abruf einer Webseite wird maßgeblich durch Größe und Anzahl der eingebetteten Objekte beeinflusst. Die Untersuchungsergebnisse vom Liberatore et al. deuten darauf hin, dass Paketgrößen-Histogramme ein viel versprechendes Merkmal zur Durchführung von Website-Fingerprinting sind (vgl. Abschnitt 3.3.2).

Die Abhängigkeit von der Internetanbindung auf Grund der Verwendung der absoluten Werte der Paketgrößen mag im ersten Moment als Nachteil erscheinen. Diese Einschränkung lässt sich jedoch durch die im KDD-Prozess vorgesehene Transformation der Rohdaten umgehen. Im konkreten Fall erscheint die Normalisierung bzw. Standardisierung der absoluten Paketgrößen sowie die anschließende Quantisierung in diskrete Intervallklassen angebracht (vgl. Ausblick in Abschnitt 10.2).

Zeitdifferenz zwischen IP-Paketen

Die Zeitdifferenzen zwischen den einzelnen Paketen werden in der Literatur ebenfalls zur Erzeugung von Fingerabdrücken verwendet. Diese Größe wird häufig als *packet interarrival time (IAT)* bezeichnet [BLJL05]. Dabei wird unterstellt, dass das deterministische Zugriffsmuster beim Abruf einer Webseite und der darin eingebetteten Objekte charakteristische Verzögerungen zwischen Paketen erzeugt.

Ähnlich wie die Gesamtübertragungsdauer wird auch die IAT durch Verzögerungen beeinflusst, die nicht durch die eigentliche Webseite bedingt sind. Es ist davon auszugehen, dass sich die Verteilung der Paketgrößen bei unterschiedlichen Internetzugängen nur geringfügig verändert, während die IATs weitaus größeren Schwankungen unterworfen sind. Eine Untersuchung dieser These steht allerdings noch aus. Da sich die Erkennungsraten in [BLJL05] beim Hinzuziehen der IATs nur marginal verbessert haben, werden sie im weiteren Verlauf nicht betrachtet.

Berücksichtigung der Paketreihenfolge

Die bisher vorgestellten Merkmale vernachlässigen die zeitliche Abfolge der einzelnen Pakete. Beim Abruf einer Webseite verändert sich die Charakteristik des Datenverkehrs während der Übertragung einer Seite: Zuerst wird die eigentliche HTML-Seite geladen, anschließend die eingebetteten Elemente. Das Website-Fingerprinting-Verfahren von Bissias et al. berücksichtigt zwar die Reihenfolge der Pakete [BLJL05], ihr Verfahren erzielt allerdings nur geringe Erkennungsraten.

Die Einbeziehung der Zeit oder der Paketreihenfolge kann zudem die Robustheit der Fingerabdrücke reduzieren. Die Reihenfolge, in der die eingebetteten Elemente heruntergeladen werden, variiert je nach verwendetem HTML-Parser. Wie Abbildung 5.5 verdeutlicht, unterscheidet sich der Datenverkehr bei verschiedenen Browsern erheblich. Es ist anzunehmen, dass Verfahren, die auf Paketgrößen-Häufigkeitsverteilungen basieren, besser mit unterschiedlichen Browsern zurecht kommen. Diese Vermutung muss allerdings noch untersucht werden.

5.3.3. Zusammenfassung

In diesem Abschnitt wurden die Vor- und Nachteile verschiedener Merkmale, die zum Website-Fingerprinting verwendet werden können, vorgestellt. Die Analyse zeigt, dass die Auswertung von IP-Paketgrößen – ohne Berücksichtigung der Reihenfolge – am ehesten dazu geeignet ist, robuste Website-Fingerabdrücke zu erzeugen. Diese Einschätzung ist auch mit den bislang publizierten Untersuchungsergebnissen in Einklang. Im weiteren Verlauf der Arbeit wird Website-Fingerprinting daher auf Basis der Paketgrößen-Häufigkeitsverteilung durchgeführt, wobei die empfangenen und die gesendeten Paketgrößen berücksichtigt werden.

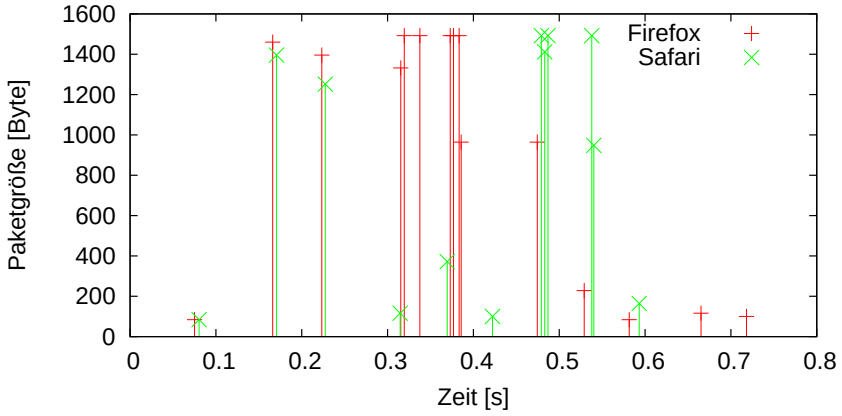


Abbildung 5.5.: Empfangene Pakete von *www.google.com* im Zeitverlauf für zwei verschiedene Browser beim Abruf über SSH-Tunnel

5.4. Erzeugung vorklassifizierter Testdaten

5.4.1. Auswahl der URLs für die Testdaten

In Abschnitt 5.2.4 wurde erläutert, dass zur systematischen Evaluation von Website-Fingerprinting-Verfahren vorklassifizierte Testdaten erforderlich sind. Es stellt sich zunächst die Frage, welche Webseiten abgerufen werden sollen. Zur Gewährleistung der Vergleichbarkeit mit bisherigen Publikationen sollten sich Anzahl und Art der abgerufenen Webseiten an den dort verwendeten Datensätzen orientieren. In der Literatur werden üblicherweise die n populärsten Seiten verwendet, wobei n und die Selektionsmethode variieren.

Während die Testdaten bei den Untersuchungen von Liberatore et al. 1000 Webseiten [LL06] enthalten, rufen Bissias et al. lediglich 100 Webseiten ab [SSW+02]. Sun et al. verwenden für ihre Untersuchungen hingegen 100 000 Seiten, wobei davon knapp 3000 Seiten für den Angreifer „interessant“ sind. Anhand der in Abschnitt 3.3.2 angegebenen Regressionsfunktion lässt sich allerdings erkennen, dass die Erkennungsraten bei Anzahlen im Bereich von wenigen tausend Seiten durchaus miteinander vergleichbar sind.

Die Evaluation der Website-Fingerprinting-Verfahren und der datenschutzfreundlichen Systeme erfolgt anhand der populärsten URLs, die im Jahr 2007 über den vom Autor entwickelten bzw. betriebenen Internet-Inhaltsfilter *FilterSurf* abgerufen wurden. FilterSurf[4] basiert auf zentral hinterlegten URL-Blacklisten. FilterSurf-Kunden installieren vor Ort den Proxy-Server *Squid* und binden einen speziellen Squid-Redirector *(FilterSurf-Redirector)* ein. Der Redirector übermittelt jede im Browser eingegebene URL in Echtzeit zur Kontrolle an einen der zentralen FilterSurf-Server. Der FilterSurf-Server protokolliert die angefragten URLs aller Nutzer in anonymisierter Form.

Zum Zeitpunkt der Datenerhebung nutzten etwa 50 Schulen und staatliche Einrichtungen den Dienst, darunter alle städtischen Schulen der Stadt Frankfurt. An einem typischen Tag bearbeiten die FilterSurf-Server etwa 130 000 Anfragen. Die genaue Anzahl der Nutzer lässt sich dadurch zwar nicht ermitteln, sie ist allerdings groß genug, um aus den protokollierten Daten die – für diese Nutzergruppe, die vermutlich primär aus Schülern besteht – populärsten URLs zu ermitteln.

[4]Homepage: http://www.filtersurf.de/

Chickenfoot	Browser-Sidebar mit JavaScript-ähnlicher Programmiersprache (http://groups.csail.mit.edu/uid/chickenfoot/)
CoScripter	Komfortsteigerung durch automatisches Navigieren Ausfüllen von Formularen (http://services.alphaworks.ibm.com/coscripter/)
Firewatir	Test-Erstellung und Low-Level-Automatisierung mit Ruby über JS-Shell (http://code.google.com/p/firewatir/)
Newbie	Komfortsteigerung durch automatisches Navigieren Ausfüllen von Formularen (http://www.newbielabs.com/)
Selenium	Test-Erstellung/Automatisierung mit Java, .NET, Perl, Python und Ruby (http://www.openqa.org/selenium/)

Tabelle 5.1.: Programme zur Automatisierung von Firefox und deren Zielsetzungen

Als Basis wurden die 2000 am häufigsten von FilterSurf überprüften URLs verwendet (Zeitraum: Januar bis September 2007). Alle URLs wurden manuell gesichtet, um eine hohe Datenqualität sicherzustellen: Doppelte Einträge, die zum Abruf derselben Seite führen (etwa *www.google.com* und www.google.de) wurden entfernt. Darüber hinaus wurden URLs entfernt, bei deren Abruf keine normale Webseite im Browser dargestellt wird (z. B. Update-Dienste oder Werbebanner). Für die Evaluation verbleiben damit 775 URLs.

5.4.2. Browser-Automatisierung

Die Erzeugung von vorklassifizierten Testdaten wird beim Website-Fingerprinting durch den automatischen Abruf von Webseiten durchgeführt. Eine einfache Möglichkeit besteht im Abruf der Seiten mit Kommandozeilen-Programmen wie *wget* oder *curl*. Allerdings rufen diese Programme die eingebetteten Elemente üblicherweise nacheinander ab, während echte Browser mehrere gleichzeitige Verbindungen zum Webserver aufbauen. Realistischer ist demnach der automatische Abruf mit einem Browser.

Für die Untersuchungen kommt daher Firefox 2.0 unter Ubuntu Linux zum Einsatz. Für diesen Browser gibt es eine Reihe von Open-Source-Programmen zur Automatisierung. Tabelle 5.1 listet die Automatisierungsprogramme auf, deren Einsatz in Erwägung gezogen wurde.

Die Wahl fiel auf *Firewatir*. Firewatir basiert auf *Watir*[5] ("Web Application Testing in Ruby"), einer Ruby-Bibliothek, mit der Webseiten-Entwickler automatisierte Tests für Webseiten schreiben können. Watir-Skripte simulieren hierzu Benutzeraktionen im Internet Explorer unter Windows. Firewatir ist eine plattformunabhängige Adaption dieses Konzepts für den Firefox-Browser. Da es Watir-Portierungen für andere Browser (u. a. Safari) gibt, lässt sich das im Folgenden beschriebene Konzepte bei Bedarf auch auf andere Browser anwenden.

Firewatir bedient sich zur Automatisierung einer Besonderheit des Firefox-Browsers: der *JavaScript-Shell*[6] (kurz: JSSh). Die JSSh ist nicht in der Standard-Binär-Distribution des Browsers enthalten. Es gibt zwei gleichwertige Installationsmöglichkeiten:

[5] http://www.openqa.org/watir/
[6] Homepage: http://croczilla.com/jssh

Listing 5.1: Firefox nimmt über den JSSh-Telnet-Server JavaScript-Befehle entgegen. Das Beispiel demonstriert den automatisierten Abruf der Webseite *www.google.com*.

```
>telnet localhost 9997
Connected to localhost.
Welcome to the Mozilla JavaScript Shell!
> var w = getWindows()[0]
> var browser = w.getBrowser()
> browser.loadURI("http://www.google.com/")
> exit()
```

Listing 5.2: Sequentieller Abruf von zwei Webseiten mit Firewatir in Ruby

```
# Firewatir-Datei laden und Modul einbinden
require 'firewatir'
include FireWatir

ff=Firefox.new
ff.goto("http://www.google.com")
ff.goto("http://www.ebay.com")
```

- Kompilieren von Firefox mit den Optionen *ac_add_options –enable-extensions=default,jssh* (in `.mozconfig`) oder
- Installation des JSSh-Add-Ons im Standard-Firefox-Browser (in dieser Arbeit verwendet)

Aus Sicherheitsgründen wird die JSSh nicht automatisch aktiviert; der Browser muss auf der Kommandozeile mit dem zusätzlichen Parameter *-jssh* gestartet werden. Erst dann nimmt die geöffnete Firefox-Instanz über einen integrierten Telnet-Server auf TCP-Port 9997 JavaScript-Befehle entgegen. In Abbildung 5.1 ist der Verlauf einer einfachen Sitzung dargestellt.

Das Firewatir-Ruby-Skript kapselt die Telnet-Kommunikation und stellt dem Entwickler eine komfortable Ruby-API zur Verfügung. In Abbildung 5.2 wird der Abruf von zwei Webseiten mit Firewatir veranschaulicht. Die Methode *goto* kehrt erst zurück, wenn die Webseite und alle eingebetteten Objekte vollständig geladen wurden (also in der Statuszeile des Browsers “Done” steht). Hierzu fragt Firewatir nach dem Absetzen der *loadURI*-Methode in einer Schleife in kurzen Zeitabständen wiederholt die Eigenschaft *getWindows()[0].getBrowser().webProgress.isLoadingDocument* ab. Setzt der Browser diese auf *true*, wird die Schleife abgebrochen und *goto* übergibt die Kontrolle wieder an seinen Aufrufer. Dieses Verhalten erleichtert die Erstellung von Website-Fingerprints erheblich. Die *goto*-Methode berücksichtigt sogar Umleitungen mittels META-Tags vom Typ HTTP-EQUIV und kehrt erst zurück, wenn der Browser die Seite, auf die umgeleitet wurde, komplett heruntergeladen hat.

Für den unbeaufsichtigten Abruf einer großen Anzahl von Webseiten sind einige Modifikationen an Firewatir erforderlich. Auf Basis des ursprünglichen Firewatir-Skripts wurde daher das Skript *autofox.rb* entwickelt, das eine robuste Automatisierung vieler Webseiten-Abrufe ohne Beaufsichtigung ermöglicht. Hierzu wurden folgende Änderungen implementiert:

- Fortfahren mit der nächsten Seite, wenn auf der aktuellen Seite HTML-Redirects auf Basis von META-Tags (HTTP-EQUIV-Refresh) mit einer Verzögerung von mehr als 20 Sekunden verwendet werden (solche Redirects werden auf einigen Nachrichten-Seiten zur Aktualisierung der Ansicht genutzt und würden zu einer Endlosschleife führen).
- Abbruch des Abrufs der aktuellen Seite und Fortfahren mit der nächsten Seite, wenn der Ladevorgang der aktuellen Seite mehr als 90 Sekunden benötigt.

Zur Erstellung der Fingerabdrücke werden die Webseiten des Test-Datensatzes in einer Endlosschleife abgerufen *(run_fingerprint_sessions.sh)*. Die von *autofox.rb* bereitgestellte API wird vom Skript *create_fingerprints.rb* verwendet, um für jeden Seitenabruf folgende Schritte durchzuführen:

1. Starten von tcpdump (*call_tcpdump.sh*) und Abspeichern der tcpdump-Prozess-ID,
2. Abruf der Webseite mit der oben beschriebenen *goto*-Methode,
3. Beenden von tcpdump durch Senden des Signals *SIGINT* an die abgespeicherte Prozess-ID,
4. Abspeichern des Protokolls mit dem Dateinamen `[URL]_YYYYMMDD-HHMMSS`.

Die Dauer eines Durchlaufs bei den 775 URLs hängt erheblich von der Geschwindigkeit des zu evaluierenden Übertragungsverfahrens ab, wobei Zeiten zwischen 30 Minuten (bei Shalon) und mehreren Stunden (bei Tor) zu beobachten waren.

5.4.3. Browser-Konfiguration

An der Konfiguration des Browsers sind einige Anpassungen erforderlich, um störende Einflüsse zu minimieren. Einerseits verringern diese Anpassungen die Realitätsnähe, andererseits wird erst durch sie eine unverfälschte und isolierte Analyse von Website-Fingerprinting-Verfahren möglich.

Die bisher beschriebenen Konfigurationsänderungen basieren auf den Beschreibungen in [BLJL05; LL06] und setzen die Arbeitshypothesen, die in Abschnitt 5.1 formuliert wurden, um.

Demnach ist zum einen der Browser-Cache zu deaktivieren (*network.http.use-cache:* false). Darüber hinaus ist das so genannte *Prefetching* abzuschalten – Firefox beginnt sonst im Hintergrund selbstständig mit dem Download von verlinkten Seiten *(network.prefetch-next:* false). Gleiches gilt für alle automatischen Ladevorgänge, etwa die Überprüfung auf Updates und die Aktualisierung von Live-Bookmarks.

Darüber hinaus werden JavaScript und alle anderen aktiven Inhalte abgeschaltet. Diese Anpassung ist damit zu rechtfertigen, dass viele sicherheitsbewusste Benutzer Browser-Add-Ons wie *NoScript*[7] verwenden, um sich vor Angriffen durch aktive Inhalte zu schützen. Die Deaktivierung aktiver Inhalte erleichtert auch den automatischen Abruf: Aktive

[7] Homepage: http://noscript.net/

Listing 5.3: Aus den von tcpdump erzeugten Dateien ist die Paketgröße und die Senderichtung zu ermitteln (IP-Adresse des Test-Rechners anonymisiert).

```
tcpdump: listening on eth0, [...] capture size 96 bytes
1195762218.139954 IP (... length: 148) 10.1.3.216.4807 > 88.198.xx.xx.22
1195762218.176269 IP (... length: 52) 88.198.xx.xx.22 > 10.1.3.216.4807
1195762218.293325 IP (... length: 100) 88.198.xx.xx.22 > 10.1.3.216.4807
1195762218.293362 IP (... length: 52) 10.1.3.216.4807 > 88.198.xx.xx.22
1195762218.295251 IP (... length: 548) 10.1.3.216.4807 > 88.198.xx.xx.22
[...]
12 packets captured
```

Inhalte können Objekte vom Webserver nachladen, nachdem die eigentliche Webseite bereits vollständig vorliegt. Das Ende solcher Ladevorgänge wäre über die Java-Script-Shell nur schwer erkennbar.

Die resultierende Konfiguration ist mit der des JonDoFox-Browsers[8], der für datenschutzfreundliches Surfen optimiert wurde, vergleichbar.

5.4.4. Speicherung der Testdaten

In Abschnitt 5.4.2 wurde erläutert, wie der Datenverkehr für den Abruf einzelner Webseiten automatisch mit tcpdump protokolliert wird. Da das entwickelte Website-Fingerprinting-Verfahren auf IP-Paketgrößen basiert, reicht es aus, den Header der übertragenen IP-Pakete auszuwerten. Listing 5.3 skizziert den Aufbau einer solchen TCP-Dump-Datei. Jede Zeile entspricht dabei einem IP-Paket. Die Größe eines Pakets kann direkt abgelesen werden, die Senderichtung lässt sich aus der Anordnung der IP-Adressen ermitteln.

Vor der Auswertung der Testdaten ist die Datenqualität zu analysieren. Um die Auswertungen nicht durch fehlerhafte Instanzen zu beeinflussen, werden dabei alle Dump-Dateien entfernt, in denen überhaupt keine Pakete enthalten waren (< 0.1 % der Dump-Dateien betroffen).

Die verbleibenden Testdaten werden in einer MySQL-Datenbank abgespeichert, um die präzise Ziehung von Stichproben zu erleichtern: Das Skript *store_trace_in_db.rb* liest eine Dump-Datei ein und extrahiert für jedes Paket das Tupel *(Senderichtung, Zeitpunkt, IAT, Paketgröße)*. Darüber hinaus berechnet es die diskrete Häufigkeitsverteilung der aufgetretenen Paketgrößen (Paketgrößen-Histogramm), die später für die Klassifizierung der Instanzen benötigt wird. Dieses Tupel wird dann in den Datenbanktabellen *histograms, traces, sites* abgespeichert. Eine Erläuterung des verwendeten relationalen Datenmodells enthält Anhang B.

5.5. Analyse mit Weka

Zur Auswertung der Testdaten wird *Weka*[9] (Version 3.5.7) verwendet. Weka *(Waikato Environment for Knowledge Analysis)* wurde an der University of Waikato in Neuseeland

[8] Download unter https://www.jondos.de/en/jondofox
[9] http://www.cs.waikato.ac.nz/~ml/weka/

entwickelt. Es enthält Implementierungen der wichtigsten Data-Mining-Algorithmen, bietet Werkzeuge zur Transformation der Attributwerte und ermöglicht die Visualisierung der Ergebnisse. Weka liest die zu analysierenden Instanzen aus so genannten ARFF-Dateien *(Attribute-Relation File Format)*, in denen die Trainings- und Testinstanzen als ASCII-Text vorliegen.

Für manuelle Analysen bietet sich der Einsatz des Weka-Explorers an. Für einen Vergleich verschiedener Verfahren ist jedoch der Weka-Experimenter besser geeignet. Diese Anwendung ermöglicht die automatische Klassifizierung mehrere Datensätze anhand von vordefinierten Konfigurationen und sie ist auch in der Lage, die Ergebnisse mit Hilfe von t-Tests auf signifikante Unterschiede zu überprüfen.

5.5.1. Erzeugung der ARFF-Dateien

Die Auswertung mit Weka erfolgt anhand von Stichproben aus der Grundgesamtheit aller Instanzen. Hierzu wird das Skript Skript *create_traintest_arff_files.rb* verwendet, das Paketgrößen-Histogramme aus der Datenbank exportiert und in ARFF-Dateien abspeichert.

Die erzeugten ARFF-Dateien haben einen genau festgelegten Aufbau. Eine ARFF-Datei enthält N_{train} Trainings- und N_{test} Testinstanzen für die Klassen (Webseiten) C. Für jede Klasse werden n_{train} und n_{test} Instanzen aus der Grundgesamtheit gezogen. Es gilt $N_{\text{train}} = |C| \cdot n_{\text{train}}$ und $N_{\text{test}} = |C| \cdot n_{\text{test}}$. Um genau festlegen zu können, welche Instanzen Weka jeweils für Training und Test verwendet, wird in Weka der "Percentage Split" in Verbindung mit der Einstellung "Preserve order for percentage split" ausgewählt. In Weka muss dann der prozentuale Anteil p_{train} eingetragen werden. Der Anteil der Trainingsinstanzen ermittelt sich als das Verhältnis $p_{\text{train}} = \frac{n_{\text{train}}}{n_{\text{train}}+n_{\text{test}}}$. Weka verwendet dann die ersten $N \cdot p_{\text{train}}$ Instanzen zum Trainieren des Klassifizierers. Danach folgen die Testinstanzen.

Ein Auszug einer solchen ARFF-Datei ist in Abbildung 5.6 dargestellt. Im Beispiel gilt $n_{\text{train}} = 2$ und $n_{\text{test}} = 10$. Der Anteil der Trainingsinstanzen ist $p_{\text{train}} = {}^{2}/_{12} = 0,\overline{16}$.[10]

create_traintest_arff_files.rb bietet eine Reihe von Möglichkeiten, die Auswahl der Trainings- und Testinstanzen zu beeinflussen. Zur realistischen Evaluation von Website-Fingerprinting dürfen die Instanzen nicht völlig zufällig aus der Grundgesamtheit gezogen werden. Stattdessen sollten sie aus unterschiedlichen Zeitfenstern stammen, um eine bestimmte zeitliche Verzögerung zwischen Training und Test zu simulieren. Im Beispiel werden Trainingsinstanzen ausgewählt, die zwischen dem 09.01.2008 und 11.01.2008 aufgezeichnet wurden. Die *Testinstanzen* werden im Beispiel hingegen zufällig aus einem Zeitfenster ausgewählt, das sechs Tage nach der letzten Trainingsinstanz beginnt.

Nach dem Dateikopf mit den Metadaten folgen die Instanzen in den einzelnen Zeilen. Jede Instanz hat zwei Attribute. Das Attribut *sizes* enthält die beobachteten Paketgrößen (durch Leerzeichen getrennt), und das Attribut *class* gibt den Namen der zugehörigen Klasse an. Darüber hinaus wird für jede Klasse die *Cosine Similarity* (siehe auch Abschnitt 7.1.1) zwischen den Trainings- und Testdaten ausgegeben, die die Ähnlichkeit der Häufigkeitsvektoren der beiden Mengen im Intervall $[0; 1]$ angibt. Diese Metrik ermöglicht

[10] In der aktuellsten Weka-Version (v3.5.7) gibt es im Weka-Explorer keine Möglichkeit den Anteilswert für den Percentage Split als Dezimalzahl einzugeben (nur Ganzzahlen sind zugelassen). Da diese Einschränkung die manuelle Untersuchung behindert, wurde dieser Mangel durch Anpassung des Quellcodes behoben.

```
% This is sample no. 11 of a total of 25 samples.
% You must set WEKA train/test percentage split to 16.66666666667 and activate the option 'Preserve Order for
% percentage split'. The instances have been selected from dataset 4.
% The first 2 instances per class are training instances [...] selected with condition
% (traces.timestamp BETWEEN '2008-01-09 00:00:00' AND '2008-01-11 00:00:00').
% The last 10 instances per class are test instances [...] selected with condition
% SELECT traces.id FROM [...] WHERE [...] AND traces.timestamp >= DATE_ADD([...], INTERVAL 6 day) AND
% traces.timestamp <= DATE_ADD([...], INTERVAL 8 day) [...]
@RELATION traces.varytrainingsize-train9-11_test17-19-size2.10_11.arff
@ATTRIBUTE sizes STRING
@ATTRIBUTE class {checkip.dyndns.org,www.lenovo-search.net/search/,[...]}
@DATA
%% TRAINING instances %%
'-164 -164 -452 -516 -516 -52 -52 -52 [...] 52 52 52 84 84 84 84 ', 'checkip.dyndns.org'
'-164 -164 -452 -516 -516 -52 -52 -52 -52 [...] 52 84 84 84 84 84', 'checkip.dyndns.org'
'-164 -500 -52 -52 -52 [...] 52 52 52 52 772 852', 'www.lenovo-search.net/search/'
'-148 -164 -500 -52 -52 [...] 52 52 52 852 852', 'www.lenovo-search.net/search/'
[...]
%% TEST instances %%
'-116 -164 -516 -52 -52 [...] 420 52 52 52 52 52 84 84 84 84', 'checkip.dyndns.org'
'-116 -164 -516 -52 -52 [...] 356 420 52 52 52 52 52 84 84', 'checkip.dyndns.org'
'-164 -452 -516 -52 -52 -52 356 [...] 52 52 52 52 52 84 84 84', 'checkip.dyndns.org'
[...]
% checkip.dyndns.org COSIM: 0.956 (similarity of training and test samples)
[...]
```

Abbildung 5.6.: Auszug aus einer ARFF-Datei zur Analyse mit Weka

eine (manuelle) Abschätzung der Unterschiede zwischen den Trainings- und Testdaten für jede Klasse.

5.5.2. Verarbeitung der ARFF-Dateien in Weka

Das *sizes*-String-Attribut wird in Weka durch den *StringToWordVector*-Filter wieder in einzelne Attribute zerlegt: Der Filter erzeugt für jede vorkommende Paketgröße ein numerisches Attribut, dem er die entsprechende Auftretenshäufigkeit innerhalb der Instanz zuordnet. Der StringToWordVector-Filter übernimmt nicht nur die Berechnung der Häufigkeitswerte, sondern ist darüber hinaus in der Lage, die in Abschnitt 6.3 beschriebenen Transformationen der Häufigkeitswerte durchzuführen. Diese müssen daher nicht selbst implementiert werden.

Verwendung des FilteredClassifiers Für eine faire Evaluierung der Verfahren ist zu beachten, dass bei der Auswahl des Klassifizierers nicht direkt das eigentliche Klassifizierungsverfahren eingetragen werden sollte. Stattdessen ist der so genannte *FilteredClassifier* zu verwenden. Dieser stellt sicher, dass während der Trainingsphase keinerlei Informationen über die Testdaten in das trainierte Modell einfließen [WF05, 401]. Der FilteredClassifier führt den StringToWordVector-Filter getrennt für die Trainings- und Testinstanzen aus, so dass die Häufigkeitsvektoren nur aus Paketgrößen bestehen, die in den Trainingsinstanzen vorkommen. Tritt in der Testphase eine neue Paketgröße auf, die in den Trainingsdaten nicht enthalten war, dann wird diese nicht berücksichtigt. In Voruntersuchungen hat sich gezeigt, dass die Erkennungsraten dadurch in der Tat geringfügig schlechter ausfallen. Da die bisherigen Veröffentlichungen keinerlei Hinweise auf

diese korrigierte Bildung der Häufigkeitsvektoren enthalten, sind die dort angegebenen Erkennungsraten vermutlich zu optimistisch.

5.5.3. t-Tests zur Ermittlung signifikanter Unterschiede

Während der Untersuchung stellte sich heraus, dass Änderungen an einzelnen Parametern zum Teil nur sehr geringe Genauigkeitsunterschiede hervorrufen. Zur fundierten Analyse werden daher stets mehrere (falls nicht anders erwähnt: 25) Stichproben aus allen zur Verfügung stehenden Instanzen gezogen. Das jeweilige Experiment wird dann für jede Stichprobe durchgeführt, wobei sich das Endergebnis als Mittelwert der einzelnen Durchläufe ergibt. Um die Reproduzierbarkeit der Experimente zu gewährleisten, werden die Stichproben mit Hilfe eines Pseudozufallszahlengenerators erzeugt, der vor der Ziehung mit der jeweiligen Stichprobennummer initialisiert wird. Die einzelnen Stichproben sind bei dieser Vorgehensweise zwar nicht notwendigerweise überlappungsfrei, die zur Verfügung stehende Grundgesamtheit wird dadurch aber bestmöglich ausgenutzt.

Mit dem in Weka vorhandenen *korrigierten t-Test für abhängige Stichproben* (*corrected resampled paired t-test*, [WF05, 157]) lässt sich ermitteln, ob sich die Erkennungsraten zweier Datensätze signifikant voneinander unterscheiden. Für den t-Test wird jeder Datensatz einmal als *Basis* verwendet und mit den anderen paarweise verglichen. Alle t-Tests werden auf einem Signifikanzniveau von $\alpha = 0,05$ durchgeführt. Demnach ist die Wahrscheinlichkeit, dass auf Basis des Testergebnisses die Aussage „die Erkennungsraten der beiden Verfahren sind *nicht identisch*“ zu Unrecht gemacht wird, kleiner als 5 %. Ist diese Irrtumswahrscheinlichkeit bei einem Vergleich höher als 5 %, kann die Nullhypothese („die Erkennungsraten der beiden Verfahren *sind identisch*“) hingegen nicht abgelehnt werden.

6

Website-Fingerprinting-Verfahren

In diesem Kapitel werden Klassifizierungsverfahren vorgestellt und evaluiert, die sich für Website-Fingerprinting-Angriffe eignen. Dabei wird insbesondere auf die Verfahren aus [LL06] eingegangen, die auf dem Jaccard-Koeffizienten und einem Naïve-Bayes-Klassifizierer basieren.

Darüber hinaus wird das verbesserte Website-Fingerprinting-Verfahren vorgestellt, das zur Untersuchung der datenschutzfreundlichen Systeme entwickelt und implementiert wurde: ein multinomialer Naïve-Bayes-Klassifizierer. Multinomiale Naïve-Bayes-Klassifizierer haben sich bereits bei der Kategorisierung von Textdokumenten bewährt [WF05, 94]. Beim Vergleich mit den etablierten Verfahren wird sich zeigen, dass dieses Text-Mining-Verfahren auch hervorragend für Website-Fingerprinting-Angriffe geeignet ist.

Zum Abschluss wird analysiert, wie verschiedene Parameter, etwa die Anzahl der Trainingsinstanzen und die Zeitdifferenz zwischen Training und Test, die Erkennungsleistung des entwicketen Klassifizierers beeinflussen.

6.1. Instanzrepräsentation mit Häufigkeitsvektoren

Zunächst wird gezeigt, dass die Problemstellung beim Website-Fingerprinting mit der Klassifizierung von Text-Dokumenten vergleichbar ist. Bei Textdokumenten entspricht eine Instanz einem String, der aus mehreren eindeutigen Termen besteht, die durch Deliminatoren (z. B. Leerzeichen) voneinander getrennt sind. Gegeben sei das Dokument[1]

$$d = \text{"amateurs hack systems, professionals hack people"}$$

mit sechs Wörtern und fünf Termen. Beim Text Mining werden die Terme bzw. deren Häufigkeiten häufig ohne Berücksichtigung der Reihenfolge betrachtet. Eine Instanz lässt sich demzufolge als Menge der darin vorkommenden Terme darstellen. Im Beispiel gilt dann $T_d = \{\text{amateurs}, \text{hack}, \text{systems}, \text{professionals}, \text{people}\}$.

[1] Das verwendete Zitat wird Bruce Schneier zugeschrieben [Ryb00].

Sollen die Termhäufigkeiten berücksichtigt werden, bietet sich zur Repräsentation ein *Multiset* (manchmal auch als *bag of words* bezeichnet) an, das im Gegensatz zu einer Menge einzelne Elemente mehrfach enthalten kann: $B_d = \{\text{amateurs, hack, hack, systems, professionals, people}\}$. Formal lässt sich ein Multiset x schreiben als $x = x_1^{m_1}, x_2^{m_2}, \ldots, x_n^{m_n}$, wobei x_i die Terme sind und m_i die jeweiligen Termhäufigkeiten. Die Termhäufigkeiten werden in der Text-Mining-Literatur auch mit f_{t_i} oder tf_{t_i} bezeichnet (vgl. [MRS07, 258 ff.]).

Beim Text-Mining werden Multisets in der Regel als Vektoren dargestellt: Für jede Instanz wird ein *Termhäufigkeitsvektor* gebildet, dessen Komponenten mit den Auftretenshäufigkeiten der einzelnen Terme belegt sind. Hierzu wird zunächst das Vokabular V ermittelt, das der Menge der M Terme entspricht, die in allen Trainingsinstanzen vorkommen. Jedes Dokument d lässt sich dann als M-dimensionaler Vektor $(\mathrm{f}_{t_1,d}, ..., \mathrm{f}_{t_M,d})$ darstellen, wobei $\mathrm{f}_{t_i,d}$ die Häufigkeit von Term i in d angibt.

Termhäufigkeitsvektoren beim Website-Fingerprinting Da in der Vektordarstellung der Inhalt der Terme keine Rolle spielt, kann man auch die Häufigkeitsverteilung von *Paketgrößen* als Vektor darstellen. Die Instanzen liegen hierzu wie in Abschnitt 5.5.1 beschrieben als String vor. Das folgende Beispiel demonstriert die Erzeugung von Termhäufigkeitsvektoren. Gegeben seien zwei Dokumente:

$$\begin{aligned} d_1 &= \text{"}-152\ 1500\ -52\ 1050\ 1500\ -80\text{"} \\ d_2 &= \text{"}-148\ 476\ 1500\ -80\text{"} \end{aligned}$$

Zur Ermittlung der Häufigkeitsvektoren ist zunächst das Vokabular V durch Enumeration aller Terme zu bilden. Daraus ergibt sich der Aufbau der Häufigkeitsvektoren $\mathbf{f}$:

$$\begin{aligned} V &= \{-152, 1500, -52, 1050, -80, -148, 476\} \\ \mathbf{f} &= (f_{-152}, f_{1500}, f_{-52}, f_{1050}, f_{-80}, f_{-148}, f_{476}) \end{aligned}$$

Die Häufigkeitsvektoren der beiden Dokumente lauten dann

$$\begin{aligned} \mathbf{f}_{d_1} &= (1, 2, 1, 1, 1, 0, 0) \\ \mathbf{f}_{d_2} &= (0, 1, 0, 0, 1, 1, 1) \end{aligned}$$

Das demonstrierte Verfahren ist einer der ersten Schritte beim Text-Mining. Liegen die Paketgrößen-Häufigkeitsvektoren in dieser Form vor, lassen sie sich wie normale Dokumente mit Text-Mining-Verfahren weiterverarbeiten.

6.2. Klassifizierungsverfahren

6.2.1. Jaccard-Koeffizient

Der *Jaccard-Koeffizient* ist eine Ähnlichkeitsmetrik für Mengen [vR79, 25], die häufig zur Erstellung von Clustern verwendet wird. Für zwei Mengen A und B berechnet sich der Jaccard-Koeffizient s_{AB} nach folgender Formel:

$$s_{AB} = \frac{|A \cap B|}{|A \cup B|} \tag{6.1}$$

Der Wertebereich dieser Metrik ist $[0;1]$. Je größer der Wert des Jaccard-Koeffizienten, desto größer ist die Übereinstimmung zwischen den beiden Mengen. Es spielt dabei keine Rolle, ob die Mengen nummerische Werte oder Merkmalsausprägungen nominal skalierter Attribute enthalten. Mit dem Jaccard-Koeffizienten kann man einen *Jaccard-Klassifizierer* konstruieren.

Wird der Jaccard-Koeffizient zum Vergleich von Website-Fingerabdrücken auf Basis von IP-Paketgrößen verwendet, entspricht eine Instanz der Menge der beobachteten IP-Paketgrößen, wobei die Auftretenshäufigkeit vernachlässigt wird. Dadurch toleriert der Jaccard-Klassifizierer kleine Veränderungen an den Webseiten. Allerdings lassen sich manche Webseiten wegen dieser Einschränkung nicht voneinander unterscheiden.

Beispiel Gegeben seien die Mengen der aufgetretenen Paketgrößen von zwei Webseiten: $A = \{-52, 516, 276, -148, 100\}$ und $B = \{-148, 100, 76, -52, 1500\}$. Dann ist die Vereinigungsmenge $A \cup B = \{100, 276, 1500, -148, 516, 76, -52\}$ und die Schnittmenge $A \cap B = \{100, -148, -52\}$. Der Jaccard-Koeffizient berechnet sich also zu $s_{AB} = 3/7 = 0,429$. □

Kontruktion des Jaccard-Klassifizierers Liegt für jede Webseite lediglich *eine einzige Trainingsinstanz* vor, bietet sich folgende Vorgehensweise zur Identifizierung unbekannter Seiten an: Für die unbekannte Instanz wird jeweils paarweise der Jaccard-Koeffizient mit allen Trainingsinstanzen ermittelt. Durch Sortieren in absteigender Reihenfolge lässt sich dann eine Rangfolge der wahrscheinlichsten Seiten erstellen. Wenn *mehrere Abrufe* einer Webseite zum Vergleich zur Verfügung stehen, müssen die Trainingsinstanzen auf geeignete Weise aggregiert werden. Ein pragmatischer Ansatz wird in [LL06] verwendet: Nur Paketgrößen, die in der Mehrheit der Trainingsinstanzen vorkommen, werden bei der Bildung des Jaccard-Koeffizienten berücksichtigt.

Für den Vergleich der verschiedenen Klassifizierungsverfahren in Abschnitt 6.4 wurde Weka um einen selbst entwickelten Jaccard-Klassifizierer erweitert. Zur Gewährleistung der Vergleichbarkeit mit den bislang publizierten Untersuchungsergebnissen wurde er nach den Vorgaben in [LL06] entwickelt.

Es gibt eine Reihe von weiteren Ähnlichkeitsmetriken, die mit dem Jaccard-Koeffizienten verwandt sind, u. a. Tanimoto-, Dice- und Sørensen-Koeffizient ([vR79, 24 ff.] und [KR05, 24 ff.]) sowie die *Cosine Similarity*, auf die ausführlich in Abschnitt 7.1.1 eingegangen wird. Inwieweit sich neben dem Jaccard-Koeffizienten auch diese anderen Metriken zum Vergleich von Website-Fingerabdrücken eignen, wurde allerdings noch nicht untersucht.

6.2.2. Naïve-Bayes-Klassifizierer mit Kernel-Density-Estimation

Liberatore et al. [LL06] verwenden einen Naïve-Bayes-Klassifizierer (NB-Klassifizierer) mit Normal-Kernel-Density-Estimation (den *NaiveBayes*-Klassifizierer von Weka). Auf eine detaillierte Herleitung des Klassifizierers wird an dieser Stelle verzichtet, da sie in [JL95] anschaulich durchgeführt wird.

Zur Klassifizierung werden die Häufigkeitsvektoren der Instanzen ermittelt (vgl. Abschnitt 6.1). Jede Komponente im Vektor wird als Attribut mit einer nummerischen Ausprägung interpretiert. Unter der naiven Annahme, dass alle Attribute einer Instanz voneinander unabhängig sind, lässt sich durch Anwendung der Formel von Bayes die Wahrscheinlichkeit, dass ein Häufigkeitsvektor $\mathbf{f} = (f_{t_1}, \ldots, f_{t_n})$ zur Klasse c_i gehört, wie folgt bestimmen:

$$p(c_i|\mathbf{f}) \sim p(c_i) \prod_{j=1}^{n} p(\mathbf{f}|c_i) \tag{6.2}$$

Die Wahrscheinlichkeiten $p(\mathbf{f}|c_i)$ werden anhand einer stetigen Wahrscheinlichkeitsverteilung ermittelt, die aus den verfügbaren Trainingsdaten geschätzt wird. Üblicherweise wird hier eine Normalverteilung der Attributausprägungen angenommen und die Wahrscheinlichkeit einer konkreten Ausprägung anhand der geschätzten Normalverteilung ermittelt. Diese einfache Methode führt jedoch zu erheblichen Schätzfehlern, wenn eine multimodale Häufigkeitsverteilungen vorliegt. Die Schätzung von $p(\mathbf{f}|c_i)$ wird daher durch die Überlagerung (Summe) mehrerer unabhängiger Normalverteilungen genauer an die Trainingsdaten angepasst. Diesen Vorgang nennt man *gaussian* bzw. *normal kernel density estimation*.

Im Gegensatz zum Jaccard-Koeffizienten ist der NB-Klassifizierer in der Lage die Häufigkeiten von Paketgrößen zur Klassifizierungsentscheidung heranzuziehen. Die erforderlichen Kernel-Schätzungen erhöhen jedoch laut [JL95] die Komplexität hinsichtlich Ausführungszeit und Speicherplatzbedarf. Bei k Attributen und n Trainingsinstanzen hat das Trainieren eine Zeitkomplexität von $O(nk)$, und der Test von m Testinstanzen von $O(mnk)$. Verzichtet man auf Kernel-Density-Estimation sinkt die Komplexität für den Test auf $O(mk)$.

6.2.3. Multinomialer Naïve-Bayes-Klassifizierer

In diesem Abschnitt wird der Kern des Website-Fingerprinting-Verfahrens vorgestellt, das für die Analyse von datenschutzfreundlichen Techniken entwickelt wurde: der *multinomiale Naïve-Bayes-Klassifizierer* (MNB-Klassifizierer). Dabei handelt es sich um ein populäres statistisches Klassifizierungsverfahren aus dem Bereich Information Retrieval bzw. Text Mining. Es wird vor allem zur Klassifikation von Textdokumenten eingesetzt. Zu den bekanntesten Anwendungen zählt die Analyse von E-Mails zur Spam-Erkennung.[2]

Die folgende theoretische Herleitung, die sich an die Darstellung in [MRS07, 258 ff.] anlehnt, erläutert die Funktionsweise des MNB-Klassifizierers anhand von Textdokumenten. Wie in Abschnitt 6.1 gezeigt wurde, lassen sich die Ausführungen jedoch analog auf die Pakethäufigkeitsverteilung von Webseiten übertragen. Der Datenverkehr, der beim Abruf

[2] z. B. in den Programmen *Mozilla Thunderbird* (http://www.mozilla.com/thunderbird/), *POPFile* (http://popfile.sourceforge.org/) und *SpamAssassin* (http://spamassassin.apache.org)

einer Webseite beobachtet wird, entspricht dabei einem Dokument, und die Paketgrößen entsprechen den Termen.

Manning et al. unterscheiden für die Belange des Text Mining zwischen dem *multivariaten Bernoulli-Modell* [MRS07, 263 f.] und dem *multinomialen Modell* [MRS07, 258 ff.]. Während beim Bernoulli-Modell (wie beim Jaccard-Koeffizienten, vgl. Abschnitt 6.2.1) lediglich die An- oder Abwesenheit von Termen eine Rolle spielt, ist das multinomiale Modell in der Lage, auch die Termhäufigkeiten zur Klassifizierung heranzuziehen. Im Folgenden wird daher lediglich das multinomiale Modell weiter verfolgt.

Die Zuordnung von Dokumenten aus $\mathcal{X}$ zu Klassen aus $\mathcal{C}$ erfolgt beim MNB-Klassifizierer unter zwei namensgebenden *naiven Annahmen* [MRS07, 266 f.]:

Positional independence assumption Die Reihenfolge der Terme in einem Dokument ist unerheblich. „Hans ruft Anna“ und „Anna ruft Hans“ sind demnach identisch.

Conditional independence assumption Die Terme innerhalb einer Klasse sind voneinander unabhängig. Mit anderen Worten: Wenn man weiß, dass ein Dokument der Klasse c_{China} den Term *„Taipei“* enthält, so kann man daraus nicht schließen, dass es daher wahrscheinlich auch den Term „Peking“ enthält, das mitunter in anderen Dokumenten der Klasse c_{China} vorkommt.

Beide Annahmen sind zwar bei echten Textdokumente (und gleichermaßen bei Website-Fingerabdrücken) nicht erfüllt – die Effektivität des MNB-Klassifizierers wird dadurch jedoch nicht beeinträchtigt.

Funktionsweise des MNB-Klassifizierers

Die Wahrscheinlichkeit, dass ein vorliegendes, zu klassifizierendes Dokument d der Klasse c angehört, sei mit $P(c|d)$ bezeichnet. Diese Wahrscheinlichkeit kann durch Anwendung des Satzes von Bayes umgeformt werden zu

$$P(c|d) = \frac{P(c)\,P(d|c)}{P(d)} \tag{6.3}$$

$P(c)$ ist die Wahrscheinlichkeit, dass ein beliebiges Dokument zur Klasse c gehört. Hätte man keinerlei Informationen über d, könnte man $P(c|d)$ lediglich durch die Verteilung der Klassen $P(C)$ schätzen (vgl. informationstheoretisches Modell in Abschnitt 2.2). Da die Wahrscheinlichkeitsverteilung der Klassen bereits vor der eigentlichen Klassifizierung aus den Trainingsdaten geschätzt werden kann, nennt man $P(c)$ *a-priori*-Wahrscheinlichkeit von c. Bei der Klassifizierung wird diese Grobschätzung nun durch ein zusätzliches Indiz, nämlich die Wahrscheinlichkeit $P(d|c)$, die angibt, wie gut das Dokument d zur Klasse c passt, und die Auftretenswahrscheinlichkeit $P(d)$ korrigiert.

Zur Klassifizierung ist die tatsächliche Wahrscheinlichkeit nicht von Interesse – es ist lediglich eine Rangfolge der Klassen zu bilden. Daher lässt sich der Ausdruck für $P(c|d)$ vereinfachen: Da $P(d)$ bei der Klassifizierung für alle Klassen konstant ist, kann diese Wahrscheinlichkeit vernachlässigt werden. Nur wenn neben der Klassifizierung auch der exakte Wert von $P(c|d)$ berechnet werden soll, muss $P(d)$ berücksichtigt werden.

Bei der Klassifizierung soll ausgehend von den *a-priori*-Wahrscheinlichkeiten der Klassen genau die Klasse ermittelt werden, die nach der Berücksichtigung der zusätzlich vorliegenden Informationen über die Testinstanz am besten passt. Diese Klasse wird mit

c_{map} *(maximum a posteriori)* bezeichnet und durch Anwendung von Formel (6.3) wie folgt ermittelt:

$$c_{\text{map}} = \arg\max_{c \in \mathcal{C}} \hat{P}(c|d) = \arg\max_{c \in \mathcal{C}} \hat{P}(c)\, \hat{P}(d|c) \tag{6.4}$$

Da die wahren Wahrscheinlichkeiten $P(c)$ und $P(d|c)$ nicht bekannt sind, müssen $\hat{P}(c)$ und $\hat{P}(d|c)$ aus den Trainingsdaten geschätzt werden. Nach dem Maximum-Likelihood-Prinzip ergibt sich

$$\hat{P}(c) = \frac{N_c}{N} \tag{6.5}$$

wobei N die Anzahl der Dokumente ist und N_c die Anzahl der Dokumente, die zur Klasse c gehören. Jetzt muss noch das zusätzliche Indiz $P(d|c)$ ermittelt werden. Liegen die Dokumente als Termhäufigkeitsvektoren vor, gilt:

$$P(d|c) = P\left[(\mathrm{f}_{t_1,d}, ..., \mathrm{f}_{t_M,d})\,|c\right] \sim \prod P(X = t_i|c)^{\mathrm{f}_{t_i,d}} \tag{6.6}$$

$P(X = t_i|c)$ gibt dabei die Wahrscheinlichkeit an, beim Ziehen mit Zurücklegen aus allen Termen, die in Dokumenten der Klasse c enthalten sind, genau Term t_i zu erhalten. Kommt Term t_i im Dokument nicht vor (d. h. $\mathrm{f}_{t_i,d} = 0$), wird der zugehörige Faktor zu 1, beeinflusst das Ergebnis also nicht. Andernfalls geht die Wahrscheinlichkeit mit der Auftretenshäufigkeit gewichtet in das Produkt ein. Das Gesamtprodukt der Term-Wahrscheinlichkeiten ist wegen der *conditional independence assumption* proportional zur Wahrscheinlichkeit, dass Dokument d zur Klasse c gehört.

Der rechte Ausdruck in der Formel ist nicht gleich ($=$) der Wahrscheinlichkeit $P(d|c)$, sondern lediglich proportional ($\sim$) dazu, da aus Gründen der Übersichtlichkeit der multinomiale Faktor $\left(\sum_{i=1}^{M} \mathrm{f}_{t_1,d}\right)! / \left(\prod_{\mathrm{i}=1}^{\mathrm{M}} \mathrm{f}_{t_i,d}!\right)$ vor dem Produkt weggelassen wurde. Dieser Faktor ist eigentlich wegen der *positional independence assumption* nötig. Da er für alle Klassen konstant ist, beeinflusst er die Klassifizierungsentscheidung jedoch nicht und muss daher auch nicht berechnet werden.

Zur Bestimmung von $\hat{P}(d|c)$ müssen die Wahrscheinlichkeiten $P(X = t_i|c)$ aus Formel (6.6) mit Hilfe der Testdaten geschätzt werden. Hierzu wird die relative Häufigkeit eines Terms t in allen Dokumenten, die zur Klasse c gehören, ermittelt:

$$\hat{P}^*(t|c) = \frac{T_{ct}}{\sum_{t' \in V} T_{ct'}} \tag{6.7}$$

T_{ct} bezeichnet dabei die Auftretenshäufigkeit von Term t in allen Dokumenten, die zur Klasse c gehören. Im Nenner werden die Häufigkeiten aller Terme des Term-Vokabulars V in Dokumenten der Klasse c aufsummiert, was gerade der Gesamtlänge aller Dokumente der Klasse c entspricht.

Allerdings gibt es ein Problem, wenn ein Term nicht in allen Klassen vorkommt: Gemäß Formel (6.7) wird in so einem Fall $P(t_i|c) = 0$ geschätzt. Bei der Bestimmung von $\hat{P}(d|c)$ ginge dann der Wert 0 in das Produkt ein, weshalb sich fälschlicherweise $\hat{P}(d|c) = 0$ ergäbe. Dieser Fehler lässt sich durch die so genannte *Laplace-Glättung* verhindern:

$$\hat{P}(t|c) = \frac{T_{ct} + 1}{\sum_{t' \in V}(T_{ct'} + 1)} \tag{6.8}$$

Beim Trainieren des MNB-Klassifizierers werden zunächst mit Formel (6.5) die *a-priori*-Wahrscheinlichkeiten $\hat{P}(c) \, \forall \, c \in \mathcal{C}$ berechnet. Dann werden die bedingten Wahrscheinlichkeiten $\hat{P}(t|c) \, \forall \, t \in V, \, \forall \, c \in \mathcal{C}$ gemäß Formel (6.8) ermittelt. Bei der Klassifikation einer Testinstanz lässt sich dann die wahrscheinlichste Klasse c_{map} bestimmen.

Aufbauend auf die obige Herleitung demonstriert das folgende – stark vereinfachte – Beispiel die Anwendung des multinomialen MNB-Klassifizierers zur Identifizierung von Webseiten anhand von IP-Paketgrößen.

Beispiel

Gegeben sind die in Tabelle 6.1 dargestellten sechs Trainingsdatensätze von drei unterschiedlichen Webseiten. Der MNB-Klassifizierer wird zunächst mit den Trainingsdaten trainiert, dann soll er die Klasse des Testdatensatzes prognostizieren.

	ID	Paketgrößen t_i	$\sum_i t_i$	Webseite
Trainingsdaten	1	52 52 70 1500 1500	3174	w_1
	2	52 60 70 1500 1500	3182	w_1
	3	100 500 500 600 1500	3200	w_2
	4	52 500 600 600 1500	3252	w_2
	5	52 70 70 1500 1500	3192	w_3
	6	60 60 70 1500 1500	3190	w_3
Testdaten	7	52 60 100 1500 1500	3212	**?**

Tabelle 6.1.: Vereinfachtes Anwendungsbeispiel eines MNB-Klassifizierers. Die Instanzen bestehen jeweils aus 5 Paketgrößen. Wegen der ähnlichen Gesamtgröße ist zunächst nicht erkennbar, zu welcher Webseite Instanz 7 am besten passt.

In der Trainingsphase werden zunächst die *a-priori*-Wahrscheinlichkeiten $\hat{P}(c)$ sind $\hat{P}(w_1) = \hat{P}(w_2) = \hat{P}(w_3) = 2/6 = 1/3$ ermittelt. Für das Vokabular gilt $|V| = |\{52, 60, 70, 100, 500, 600, 1500\}| = 7$. Dann werden die bedingten Wahrscheinlichkeiten $\hat{P}(t|c)$ für alle Klassen und Terme im Vokabular berechnet, wobei die oben beschriebene Laplace-Glättung verwendet wird:

$$\begin{aligned}
P(52|w_1) &= (3+1)/(10+7) = 4/17 \\
P(60|w_1) &= (1+1)/(10+7) = 2/17 \\
P(70|w_1) &= (2+1)/(10+7) = 3/17 \\
P(100|w_1) = P(500|w_1) = P(600|w_1) &= (0+1)/(10+7) = 1/17 \\
P(1500|w_1) &= (4+1)/(10+7) = 5/17
\end{aligned}$$

$$\begin{aligned}
P(52|w_2) &= (1+1)/(10+7) = 2/17 \\
P(60|w_2) = P(70|w_2) &= (0+1)/(10+7) = 1/17 \\
P(100|w_2) &= (1+1)/(10+7) = 2/17 \\
P(500|w_2) = P(600|w_2) &= (3+1)/(10+7) = 4/17 \\
P(1500|w_2) &= (2+1)/(10+7) = 3/17
\end{aligned}$$

$$\begin{aligned} P(52|w_3) &= (1+1)/(10+7) = 2/17 \\ P(60|w_3) &= (2+1)/(10+7) = 3/17 \\ P(70|w_3) &= (3+1)/(10+7) = 4/17 \\ P(100|w_3) = P(500|w_3) = P(600|w_3) &= (0+1)/(10+7) = 1/17 \\ P(1500|w_3) &= (4+1)/(10+7) = 5/17 \end{aligned}$$

Zur Klassifizierung der Testinstanz (ID=7) gemäß Formel (6.4) müssen nun die einzelnen $\hat{P}(c|d)$ berechnet werden. Diese setzen sich zusammen aus den a-priori-Wahrscheinlichkeiten $\hat{P}(c)$ und den bedingten Wahrscheinlichkeiten $\hat{P}(d|c)$, die nach Formel (6.6) zu ermitteln sind.

$$\begin{aligned} P(w_1|d) &\propto 1/3 \cdot 4/17 \cdot 2/17 \cdot (3/17)^0 \cdot 1/17 \cdot (1/17)^0 \cdot (1/17)^0 \cdot (5/17)^2 \approx 3,99 \cdot 10^{-4} \\ P(w_2|d) &\propto 1/3 \cdot 2/17 \cdot 1/17 \cdot (1/17)^0 \cdot 2/17 \cdot (4/17)^0 \cdot (4/17)^0 \cdot (3/17)^2 \approx 8,45 \cdot 10^{-6} \\ P(w_3|d) &\propto 1/3 \cdot 2/17 \cdot 3/17 \cdot (4/17)^0 \cdot 1/17 \cdot (1/17)^0 \cdot (1/17)^0 \cdot (5/17)^2 \approx 3,52 \cdot 10^{-5} \end{aligned}$$

Man erhält $c_{\text{map}} = w_1$. Der NB-Klassifizierer wird die Testinstanz daher der Webseite w_1 zuordnen.[3] Bei genauerer Betrachtung der Paketgrößen ist hierfür vor allem die größere Häufigkeit der Paketgröße 52 (dreimal bei den Instanzen von w_1, einmal bei den Instanzen von w_3) ausschlaggebend, die auch nicht durch die Paketgröße 60 (einmal bei den Instanzen von w_1, zweimal bei den Instanzen von w_3) wettgemacht werden kann. w_2 scheidet schon auf den ersten Blick wegen der geringen Ähnlichkeit aus.

Die *a-priori*-Wahrscheinlichkeiten $P(c)$ haben in diesem Beispiel keinen Einfluss auf die Klassifizierung, da für jede Webseite die gleiche Anzahl an Trainingsinstanzen vorliegt. In einem realen Szenario könnte man durch Beobachtung der tatsächlichen Abrufhäufigkeiten der zu untersuchenden Webseiten dem Klassifizierer zusätzliche Informationen zur Verfügung stellen. Er würde dann den Webseiten, die häufiger abgerufen werden (bzw. für die mehr Trainingsdatensätze vorliegen), den Vorzug geben. □

Implementierungsaspekte

Wie das Beispiel verdeutlicht, entstehen beim Aufmultiplizieren der bedingten Wahrscheinlichkeiten $\hat{P}(t|c)$ sehr kleine Werte. Bei der Implementierung eines MNB-Klassifizierers kann dies zu einem *floating point underflow* führen [MRS07, 258]. Dieses Problem lässt sich durch eine logarithmische Transformation der Wahrscheinlichkeiten lösen. Das Produkt kann dann (wegen $\log(xy) = \log(x) + \log(y)$) durch die Summe der logarithmierten Werte ersetzt werden.

Ferner kann es passieren, dass in den Testinstanzen Terme vorkommen, die in keiner Trainingsinstanz enthalten sind. Diese können zur Klassifizierung nicht herangezogen werden. Es gibt zwei Möglichkeiten, mit solchen Termen umzugehen: Entweder werden solche Terme zu einem virtuellen Term <*unknown*> zusammengefasst oder sie werden einfach nicht berücksichtigt – letztere Vorgehensweise ist im Weka-Filter *StringToWordVector* implementiert, wenn er vom *FilteredClassifier* aufgerufen wird.

Schließlich ist es denkbar, dass ein zu klassifizierendes Dokument zu einer Klasse gehört, die von den Trainingsinstanzen nicht abgedeckt wird. Im schlimmsten Fall ordnet der Klassifizierer eine solche Instanz einer falschen Klasse zu und erzeugt damit einen *False*

[3] In diesem Beispiel führt die Klassifizierung anhand des Jaccard-Koeffizienten zum selben Ergebnis.

Positive bei dieser Klasse (vgl. Abschnitt 6.4). Besser wäre es in diesem Fall, wenn der Klassifizierer eine solche Instanz überhaupt keiner Klasse zuweisen würde.

Da beim MNB-Klassifizierer nicht nur die Rangfolge der Klassen, sondern auch die absolute Wahrscheinlichkeit $\hat{P}(c|d)$, die ausdrückt wie wahrscheinlich ein Dokument zu einer Klasse gehört, ist es vorstellbar, für die Klassifizierung direkt eine Mindestwahrscheinlichkeit anzugeben. Zu beachten ist dabei allerdings, dass die absoluten Wahrscheinlichkeitswerte, die der MNB-Klassifizierer ermittelt, nur beschränkte Aussagekraft haben, da die einzelnen Terme – entgegen der naiven Annahme – *eben nicht* voneinander unabhängig sind.

Der *NaiveBayesMultinomial*-Klassifizierer von Weka berechnet zwar in der Methode *distributionForInstance(...)* die Wahrscheinlichkeiten $\hat{P}(c|d)$, verwendet jedoch bei der Klassifikation keine Mindestwahrscheinlichkeit. Zur Verwendung eines absoluten Schwellenwerts müsste daher zunächst der Klassifizierer angepasst werden. Bei den nachfolgenden Untersuchungen spielt diese Einschränkung allerdings ohnehin keine Rolle, da gewährleistet wird, dass in den Trainings- und Testdaten die gleichen Klassen vorkommen. In Abschnitt 9.2 wird evaluiert, wie sich ein angepasster MNB-Klassifizierer verhält, wenn in den Trainingsdaten nicht alle Klassen repräsentiert sind – was in der Praxis der Normalfall ist.

6.3. Optimierung des MNB-Klassifizierers

Es gibt eine Vielzahl von Optimierungsmöglichkeiten, die die Effektivität eines MNB-Klassifizierers bei Information-Retrieval- und Text-Mining-Problemstellungen erhöhen. Der eigentliche Algorithmus bleibt dabei unverändert. Die Optimierungen betreffen lediglich die Instanzrepräsentation.

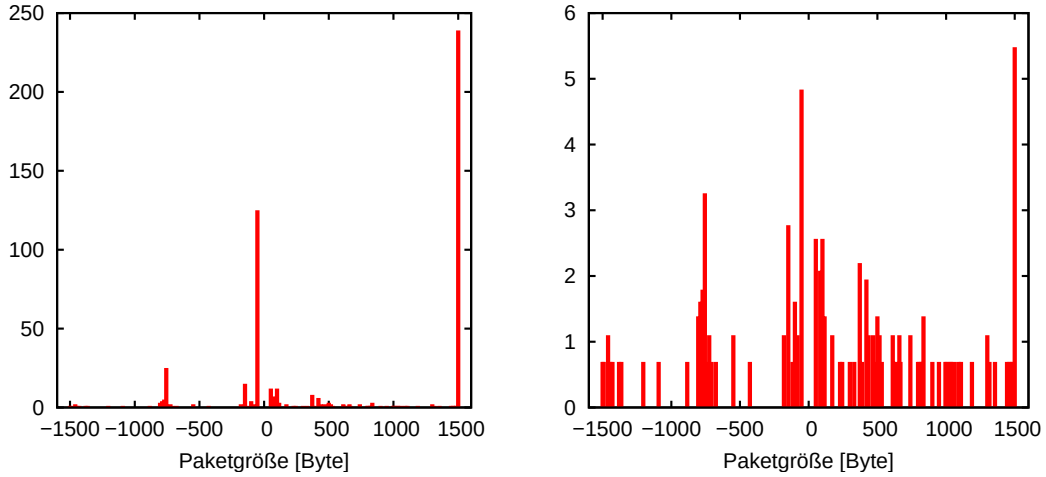
Beim Website-Fingerprinting basiert die Entscheidung des MNB-Klassifizierers auf den Auftretenshäufigkeiten der verschiedenen IP-Paketgrößen. Diese Vorgehensweise hat zwei unerwünschte Eigenschaften: Zum einen sind alle auftretenden Paketgrößen gleich stark gewichtet, unabhängig wie relevant sie sind. Zum anderen werden Klassen bevorzugt, in denen einzelne Paketgrößen besonders häufig vorkommen. Vergleichbare Probleme gibt es bei der Klassifizierung von Textdokumenten.

Im Folgenden werden drei Transformationen vorgestellt, die üblicherweise bei der Klassifizierung von Textdokumenten eingesetzt werden, um die störenden Effekte zu beseitigen. Inwiefern diese Optimierungen ihre Wirkung auch beim Website-Fingerprinting entfalten, wird in Abschnitt 6.6 untersucht.

6.3.1. Logarithmische Skalierung (TF-Transformation)

Werden bei der Klassifizierung der Instanzen die tatsächlichen Häufigkeitsvektoren verwendet, bevorzugt der MNB-Klassifizierer bei der Entscheidung diejenigen Instanzen, die einzelne Terme besonders häufig enthalten. Die Klassifizierungsentscheidung wird demnach durch überproportional häufige Terme verzerrt.

Beim Website-Fingerprinting ist das Ungleichgewicht aus technischen Gründen sogar besonders groß. Der IP-Stack versucht stets, die zu übertragenden Daten mit möglichst großen IP-Paketen zu versenden. Abbildung 6.1a zeigt diesen Umstand am Histogramm der Paketgrößen-Häufigkeitsverteilung beim Abruf der Seite *www.muenchen.de*.



(a) Tatsächliche Pakethäufigkeiten (b) Logarithmierte Häufigkeiten

Abbildung 6.1.: In diesem Häufigkeitsvektor von *www.muenchen.de* dominieren die Paketgrößen -52 und 1500. Nach einer logarithmischen Skalierung (hier: ln) ist ihr Einfluss erheblich geringer.

Durch eine logarithmische Skalierung der Häufigkeitswerte kann der Einfluss von dominanten Termen reduziert werden. Hierzu wird in Formel (6.6) statt der tatsächlichen Termhäufigkeit $f_{t,d}$ der logarithmierte Wert $f^*_{t,d}$ verwendet [WF05, 311]:

$$f^*_{t,d} = \log(1 + f_{t,d}) \tag{6.9}$$

Die Basis des Logarithmus spielt dabei keine Rolle. Abbildung 6.1b verdeutlicht die veränderten Größenverhältnisse nach Anwendung dieser *TF-Transformation*.

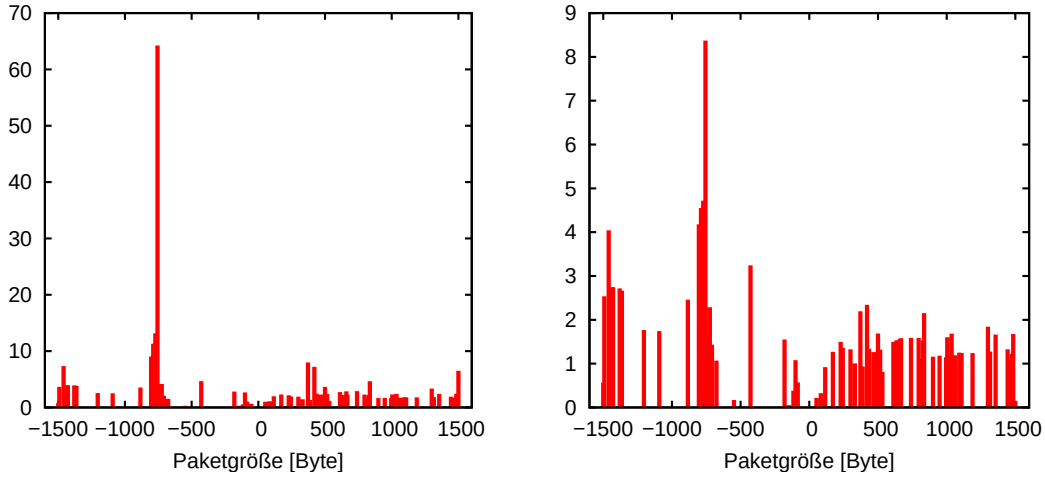
6.3.2. Berücksichtigung der Relevanz (TF-IDF-Transformation)

Neben der Bevorzugung von Klassen, die dominante Terme enthalten, gibt es noch einen zweiten unerwünschten Effekt. Alle Terme bzw. Paketgrößen haben bei der Klassifizierung dasselbe Gewicht. Es ist nahe liegend, dass die unterschiedlichen Paketgrößen nicht alle gleichermaßen zum charakteristischen Profil einer Webseite beitragen. So treten die kleinen TCP-ACK-Pakete bei praktisch allen Webseiten auf, und Pakete mit der Größe der MTU (meist 1500) sind ebenfalls in fast allen Instanzen enthalten.

Bei der Klassifikation von Textdokumenten wird daher üblicherweise die *document frequency* df_t ermittelt, die angibt, in wie vielen Dokumenten Term t enthalten ist. Je höher df_t, desto weniger eignet sich Term t als Diskriminator. Bei einer Menge von N Dokumenten wird daher die *inverse document frequency* idf_t wie folgt ermittelt [MRS07, 118 f.]:

$$idf_t = \log \frac{N}{df_t} \tag{6.10}$$

Je seltener Term t ist, desto größer ist der Wert von idf_t. Auch hier spielt die Basis des Logarithmus keine Rolle. Die *term frequency* $f_{t,d}$ von Term t in Dokument d wird nun gewichtet:



(a) IDF-Transformation (b) TF-IDF-Transformation

Abbildung 6.2.: Durch Anwendung der IDF-Transformation auf eine Instanz von *www.muenchen.de* werden die Größenverhältnisse erheblich verändert.

$$f^*_{t,d} = f_{t,d} \cdot idf_t \tag{6.11}$$

Diese Modifikation wird als *IDF-Transformation* bezeichnet. Der Wert von $f^*_{t,d}$ ist am größten, wenn t in sehr wenigen Dokumenten sehr häufig vorkommt – für diese also charakteristisch ist.

Während der Nutzen der IDF-Transformation bei Textdokumenten einleuchtet, gehen beim Website-Fingerprinting wertvolle Informationen verloren: Durch die IDF-Transformation werden die Pakethäufigkeiten und das Verhältnis der Häufigkeiten innerhalb einer Instanz verwischt. Der Nutzen der IDF-Transformation ist im Kontext von Website-Fingerprinting daher zweifelhaft. Wird die IDF-Transformation auf das in Abbildung 6.1a dargestellte Paketgrößen-Histogramm angewendet, verändern sich die Verhältnisse der Häufigkeiten erheblich. Wie Abbildung 6.2a zeigt, enthält die im Beispiel dargestellte Instanz von *www.muenchen.de* im Vergleich zu anderen Webseiten im Datensatz offenbar besonders viele Pakete mit einer Größe von etwa -750. Die ursprünglich dominanten Größen -52 und 1500 treten dagegen in den Hintergrund, da sie in allen Dokumenten häufig vorkommen.

Beim Textmining wird die IDF-Transformation üblicherweise mit der TF-Transformation kombiniert. Abbildung 6.2b zeigt das Resultat dieser *TF-IDF-Transformation*. Auch in diesem Histogramm ist der Einfluss der IDF-Transformation deutlich zu sehen.

6.3.3. Normalisierung der Vektorlänge

Es gibt noch einen weiteren Einflussfaktor, der das Klassifizierungsverhalten unerwünscht beeinflussen kann: Die Dokumentlänge bzw. die Anzahl der Pakete in einer Instanz. Auf den ersten Blick liefert diese Eigenschaft einen wertvollen Beitrag zur Diskriminierung. In empirischen Untersuchungen hat sich jedoch herausgestellt, dass bei der Klassifizierung von Textdokumenten die absoluten Termhäufigkeiten wenig aussagekräftig sind und der

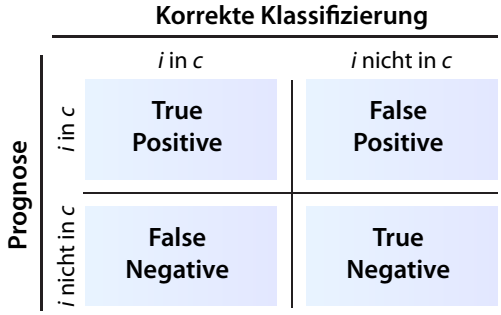


Abbildung 6.3.: Fehlerarten bei der Klassifizierung

MNB-Klassifizierer mit normalisierten Häufigkeiten erheblich bessere Ergebnisse liefert. Es ist also angebracht, den Einfluss der Normalisierung beim Website-Fingerprinting zu überprüfen.

Zur Normalisierung werden alle n-dimensionalen Termvektoren nach Anwendung der bereits beschriebenen Transformationen auf eine einheitliche euklidische Länge skaliert (im Normalfall auf die Länge 1). Diese Transformation wird manchmal als *cosine normalization* bezeichnet [MRS07, 128], da sie auch bei der Ähnlichkeitsmetrik *Cosine Similarity* eine wichtige Rolle spielt (vgl. Abschnitt 7.1.1). Die einzelnen Häufigkeiten werden hierzu durch die euklidische Länge des Termvektors dividiert:

$$f_{t,d}^{\text{norm}} = \frac{f_{t,d}^{*}}{\left\| \left(f_{t_1,d}^{*}, \ldots, f_{t_n,d}^{*} \right) \right\|} = \frac{f_{t,d}^{*}}{\sqrt{\sum_{t' \in V} \left(f_{t',d}^{*} \right)^2}} \tag{6.12}$$

In [WF05, 399] wird erläutert, dass die Normalisierung auf die Länge 1 beim Einsatz der Laplace-Glättung nicht optimal ist. Die dabei entstehenden Termhäufigkeiten sind dann so klein, dass der Einfluss des Glättungsfaktors die Klassifizierungsentscheidung verzerrt. Stattdessen schlagen die Autoren vor, die Vektoren auf die durchschnittliche Länge aller Instanzvektoren zu skalieren. Diese verbesserte Normalisierung kommt auch beim Weka-Filter *StringToWordVector* zum Einsatz, der für die Untersuchungen verwendet wird.

6.4. Erkennungsrate als Evaluationsmetrik

Bei der Evaluation von Klassifizierungsverfahren werden vier Ergebnisarten unterschieden, die in Abbildung 6.3 dargestellt sind [WF05, 162]. Gehört die zu klassifizierende Testinstanz i zur Klasse c und prognostiziert der Klassifizierer auch diese Zuordnung, wird die Instanz als *True Positive (TP)* gezählt. Für die anderen Klassen, denen der Klassifizierer die Instanz nicht zugeordnet hat, ist sie ein *True Negative (TN)*. Prognostiziert der Klassifizierer die falsche Klasse, handelt es sich aus der Perspektive der falschen Klasse um einen *False Positive (FP)* und aus der Perspektive der korrekten Klasse um einen *False Negative (FN)* .

Die bisherigen Publikationen (abgesehen von [SSW+02]) gehen bei der Evaluation von Website-Fingerprinting-Verfahren nicht näher auf die obigen Maßzahlen ein. Stattdessen konzentrieren sie sich auf die *Erkennungsrate*. Die Erkennungsrate entspricht dem Verhältnis $\frac{\text{TP}+TN}{\text{TP}+\text{FP}+TN+FN}$.

Variierter Parameter	Klassifizierungsverfahren
Datensatz	OpenSSH-Tunnel
Anzahl der Instanzen	$n_{train} = 4$, $n_{test} = 4$
Intervall zw. Training und Test	$\Delta_t = 6$ Tage
Klassifizierungsverfahren	Jaccard-Koeffizient, NB- und MNB-Klassifizierer
Transformation der Daten	keine bzw. TF, jeweils mit/ohne Normalisierung
Metrik	Erkennungsrate und Geschwindigkeit

Tabelle 6.2.: Parameter zum Vergleich der Klassifizierungsverfahren

In einigen Publikationen geben die Autoren zusätzlich an, wie sich die Erkennungsrate verändert, wenn der Klassifizierer nicht nur einmal raten darf, sondern n-mal. Das Klassifizierungsverfahren gibt in diesem Fall bei jeder Testinstanz die n wahrscheinlichsten Klassen aus. Es ist allerdings fraglich, ob die solchermaßen ermittelten Erkennungsraten in der Praxis überhaupt überhaupt relevant sind. Daher beschränkt sich diese Arbeit auf die Angabe der Erkennungsraten bei einmaligem Raten.

Die Erkennungsrate ist allerdings nur aussagekräftig, wenn jede Testinstanz auch einer Klasse zugewiesen werden kann. Dies ist insbesondere bei der Anwendung von Website-Fingerprinting in der Realität nicht der Fall, wenn der Großteil des beobachteten Datenverkehrs zu Webseiten gehört, die nicht im Trainingsdatensatz enthalten sind. In diesem Fall ist die Minimierung der False-Positive-Rate $\frac{\text{FP}}{\text{TN+FP}}$ möglicherweise wichtiger als die Maximierung der Erkennungsrate. Auf diese Problematik wird in Abschnitt 9.2 eingegangen.

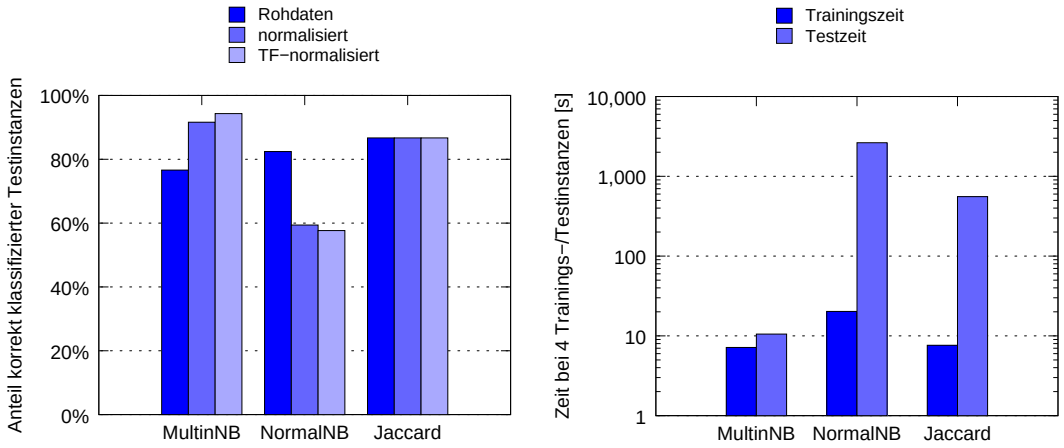
6.5. Vergleich der Klassifizierungsverfahren

Der Vergleich der Klassifizierungsverfahren erfolgt anhand der Effektivität, also der *Erkennungsrate*, und der Effizienz, die sich in der zur Klassifizierung benötigten *Zeit* niederschlägt.

Der entwickelte MNB-Klassifizierer muss sich an den etablierten Verfahren, die in der Literatur dokumentiert sind, messen lassen. Eine Einschätzung seiner Leistungsfähigkeit auf Basis der in der Literatur veröffentlichten Ergebnisse ist jedoch problematisch, da die dort abgedruckten Ergebnisse auf anderen Datensätzen erzielt wurden. Darüber hinaus werden in der Literatur keine Angaben über die Geschwindigkeit der implementierten Website-Fingerprinting-Verfahren gemacht, so dass ein Vergleich der Effizienz gar nicht möglich wäre. In diesem Abschnitt werden die verschiedenen Verfahren daher anhand eines einheitlichen Datensatzes miteinander verglichen.

Für die Analyse werden Instanzen aus dem OpenSSH-Datensatz mit den Weka-Klassifizierern *NaiveBayesMultinomial* (in Verbindung mit dem Filter *StringToWordVector*) und *NaiveBayes* (mit aktivierter Kernel-Density-Estimation) sowie dem selbst entwickelten *JaccardSimilarityClassifier* klassifiziert.

Erkennungsraten Die Erkennungsraten sind in Abbildung 6.4a dargestellt. Zunächst zu den etablierten Verfahren: NaiveBayes mit Kernel-Density-Estimation erzielt eine geringfügig schlechtere Erkennungsrate als der JaccardSimilarityClassifier (linker Balken im



(a) Erkennungsrate (b) Geschwindigkeit

Abbildung 6.4.: Beim Vergleich der Klassifizierer anhand von Erkennungsraten und Performanz schneidet der MNB-Klassifizierer *(MultinNB)* gut ab. *Jaccard* und *NormalNB* werden in den Abschnitten 6.2.1 und 6.2.2 beschrieben.

Diagramm) – dieser Zusammenhang wird auch in [LL06] beschrieben. Das Niveau der ermittelten Erkennungsraten sind etwas höher als in [LL06], wo der Jaccard-Klassifizierer bei vergleichbaren Bedingungen (4 Trainingsinstanzen, 7 Tage Zeitdifferenz, 1000 Webseiten) etwa 70 % der Instanzen korrekt klassifiziert. Diese Diskrepanz ist auf die bereits angedeuteten Unterschiede der Datensätze zurückzuführen.

Bei Verwendung der Rohdaten kann der MNB-Klassifizierer das Niveau der etablierten Verfahren zunächst nicht erreichen. Werden die Häufigkeitsvektoren jedoch durch Normalisierung und zusätzliche Anwendung der TF-Transformation optimiert, übertreffen die Erkennungsraten des MNB-Klassifizierers die der übrigen Verfahren erheblich – die Unterschiede sind statistisch signifikant.

Beim normalen NB-Klassifizierer ist die Verwendung optimierter Häufigkeitsvektoren kontraproduktiv. Beim JaccardSimilarityClassifier wirken sich die Transformationen gar nicht auf die Erkennungsraten aus, da bei diesem Verfahren nur die An- oder Abwesenheit einer Paketgröße eine Rolle spielt, nicht jedoch die Auftretenshäufigkeit.

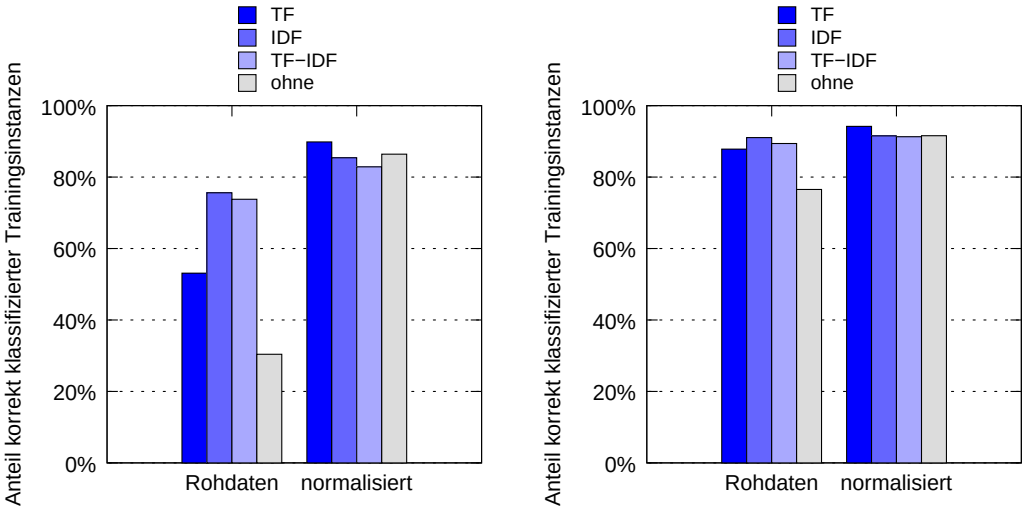
Performanz In Abbildung 6.4b ist die zur Klassifizierung benötigte Zeit auf einer logarithmischen Skala dargestellt. Der verwendete Datensatz enthält pro Klasse jeweils vier Trainings- und vier Testinstanzen.

Der MNB-Klassifizierer ist sowohl beim Training als auch beim Testen deutlich schneller als die etablierten Verfahren. Der NB-Klassifizierer benötigt etwa die hundertfache Zeit, um die Testinstanzen zu klassifizieren. Dies ist auf die hohe Zeitkomplexität zurückzuführen, auf die bereits in Abschnitt 6.2.2 hingewiesen wurde.

Da der selbst entwickelte Jaccard-Klassifizierer nicht auf Geschwindigkeit optimiert wurde, dauert bei ihm die Klassifizierung der Test-Instanzen ebenfalls sehr lange. Aus den Ergebnissen kann man daher nicht grundsätzlich auf eine schlechte Performanz des Jaccard-Klassifizierers schließen.

Variierter Parameter	Transformationsverfahren
Datensatz	OpenSSH
Anzahl der Instanzen	$n_{train} = \{1, 4\}$, $n_{test} = 10$
Intervall zw. Training und Test	$\Delta_t = 6$ Tage
Klassifizierungsverfahren	multinomialer Naïve-Bayes-Klassifizierer
Transformation der Daten	keine, TF, IDF, TF-IDF; jeweils mit/ohne Normalisierung
Metrik	Erkennungsrate

Tabelle 6.3.: Parameter zur Ermittlung des Einflusses der Transformationen



(a) 1 Trainingsinstanz

(b) 4 Trainingsinstanzen

Abbildung 6.5.: Die TF-Transformation erzielt zusammen mit der Normalisierung die besten Ergebnisse.

Bewertung Der MNB-Klassifizierer erzielt im Vergleich zu den etablierten Verfahren beim OpenSSH-Datensatz höhere Erkennungsraten bei geringerer Verarbeitungszeit. Da die Transformation der Häufigkeitsvektoren offensichtlich erheblichen Einfluss auf die Leistungsfähigkeit dieses Verfahrens hat, wird im nächsten Abschnitt noch detaillierter auf die Auswirkungen der unterschiedlichen Transformationen eingegangen.

6.6. Einfluss der Transformationen

In Abschnitt 6.3 wurden verschiedene Transformationen vorgestellt, die die Erkennungsleistung eines MNB-Klassifizierers bei Textdokumenten verbessern können. In diesem Abschnitt wird der Einfluss dieser Transformationen auf die Erkennungsraten beim Website-Fingerprinting untersucht, indem die verschiedenen Optimierungen anhand von Stichproben aus dem OpenSSH-Datensatz miteinander verglichen werden. Die Parameter des Experiments sind in Tabelle 6.3 aufgeführt.

In Abbildung 6.5 sind die Erkennungsraten für die verschiedenen Transformationen dargestellt. Zunächst fällt auf, dass die Erkennungsraten in erheblichem Maß von der Anzahl der Trainingsinstanzen abhängen – dieser Parameter wird daher in Abschnitt 6.7 separat

Variierter Parameter	Anzahl der Trainingsinstanzen
Datensatz	OpenSSH
Anzahl der Instanzen	$n_{train} = \{1, 2, 4, 8, 16\}$, $n_{test} = 10$
Intervall zw. Training und Test	$\Delta_t = 6$ Tage
Klassifizierungsverfahren	multinomialer Naïve-Bayes-Klassifizierer
Transformation der Daten	TF-Transformation, Normalisierung
Metrik	Erkennungsrate

Tabelle 6.4.: Parameter zur Ermittlung des Einflusses der Anzahl der Trainingsinstanzen

untersucht. Bei vier Trainingsinstanzen sind die Unterschiede zwischen den verschiedenen Verfahren erheblich geringer als bei nur einer Instanz. In beiden Diagrammen lassen sich jedoch dieselben Tendenzen erkennen:

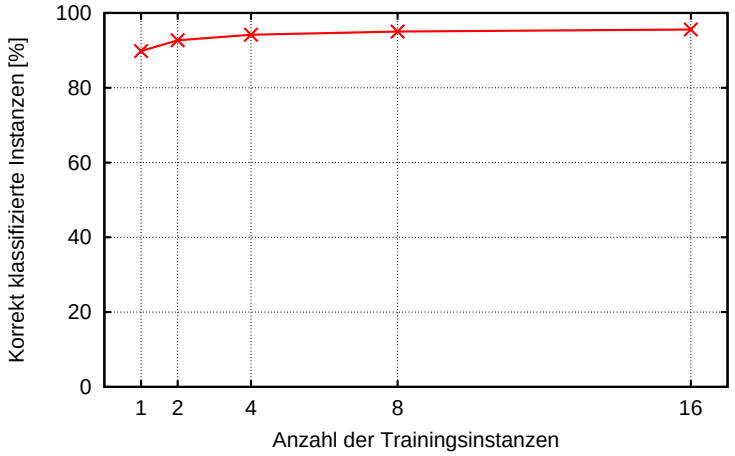
- Die Normalisierung der Häufigkeitsvektoren auf die durchschnittliche Länge (vgl. Abschnitt 6.3.3) verbessert die Erkennungsleistung im Falle einer einzigen Trainingsinstanz erheblich. Auch bei vier Trainingsinstanzen führt die Normalisierung zu signifikant besseren Erkennungsraten (vgl. Tabelle A.1).
- Die IDF-Transformation steigert zwar die Erkennungsleistung bei Verwendung der Rohdaten, in Verbindung mit normalisierten Häufigkeitsvektoren führt sie jedoch zu einer signifikanten Verschlechterung der (dann bereits sehr guten) Erkennungsraten.
- Die TF-Transformation zeichnet sich in Verbindung mit normalisierten Häufigkeitsvektoren durch eine signifikant höhere Erkennungsrate aus als alle anderen Verfahren (vgl. Tabelle A.1). Die Erkennungsrate beträgt bei vier Trainingsinstanzen bis zu 94,18 % bei einer Pause von sechs Tagen zwischen Training und Test.

Bewertung Die vorgestellten Transformationen aus dem Bereich des Text Mining verbessern die Erkennungsleistung beim Website-Fingerprinting erheblich. Wie in Abschnitt 6.3 vermutet, ist die IDF-Transformation hier allerdings kontraproduktiv. Die besten Ergebnisse werden beim Einsatz der TF-Transformation mit anschließender Normalisierung der Häufigkeitsvektoren erzielt.

6.7. Einfluss der Anzahl der Trainingsinstanzen

Für die praktische Durchführung von Website-Fingerprinting spielt die Anzahl der benötigten Trainingsinstanzen eine wichtige Rolle. Je weniger Instanzen zum Trainieren aufgezeichnet werden müssen, desto einfacher lässt sich der Angriff durchführen. Andererseits kann die Genauigkeit eines Naïve-Bayes-Klassifizierers erheblich gesteigert werden, wenn mehr Trainingsbeispiele zur Verfügung stehen (vgl. die Ergebnisse in [LL06]).

Im Folgenden wird am Beispiel des OpenSSH-Datensatzes überprüft, ob auch beim MNB-Klassifizierer die Anzahl der Trainingsinstanzen die Genauigkeit beeinflusst. Die Versuchsbedingungen sind in Tabelle 6.4 zusammengefasst.



Anzahl der Trainingsinstanzen	1	2	4	8	16
Korrekt klassifizierte Testinstanzen	89,82 %	92,70 %	94,18 %	95,02 %	95,59 %
Unterschied zur nächstkleineren Größe	–	+ 3,21 %	+ 1,60 %	+ 0,89 %	+ 0,60 %

Abbildung 6.6.: Die Anzahl der korrekt klassifizierten Testinstanzen nimmt mit größerer Anzahl der Trainingsinstanzen geringfügig zu.

Abbildung 6.6 zeigt die Ergebnisse für die verschiedenen Größen der Trainingsmenge. Selbst wenn der Klassifizierer *mit nur einer Instanz* trainiert wurde, werden im Durchschnitt fast 90 % der Testinstanzen erkannt. Mit steigender Anzahl der Trainingsinstanzen steigt die Erkennungsrate auf bis zu 96 %, wobei die Zuwachsraten dabei stetig abnehmen. Tabelle A.2 zeigt, dass die Steigerung der Erkennungsraten beim Wechsel von 1 auf 2 Instanzen, von 2 auf 4 Instanzen sowie von 4 auf 8 Instanzen jedoch nicht statistisch signifikant ist. Erst beim Sprung von 8 auf 16 Instanzen ergibt sich ein signifikanter (jedoch sehr kleiner) Unterschied. Eine signifikante Verbesserung ergibt sich allerdings auch beim Sprung von 1 auf 4 Trainingsinstanzen. Die Verwendung von 4 Trainingsinstanzen ist also ein effizienter Kompromiss.

Bewertung Der MNB-Klassifizierer erzielt bereits bei einer einzigen Trainingsinstanz eine Erkennungsrate von ca. 90 %, wobei sich die Erkennungsleistung durch zusätzliche Trainingsdaten noch weiter steigern lässt. Bei den etablierten Verfahren sind hingegen mindestens vier Trainingsinstanzen erforderlich.

6.8. Zeitliche Robustheit

Beim Website-Fingerprinting auf Basis der Paketgrößen wird die Tatsache ausgenutzt, dass die Paketgrößen-Häufigkeitsverteilung für eine Webseite charakteristisch ist. Die Häufigkeitsverteilung ergibt sich aus dem Aufbau der HTML-Seite und der darin eingebetteten Elemente. Ändert sich eine HTML-Seite, ändert sich auch ihr Fingerabdruck.

Es stellt sich die Frage, ob sich die typischen Änderungen, denen Webseiten im Verlauf der Zeit unterliegen, auf die Erkennungsraten auswirken. In verschiedenen Untersuchungen wird Art und Häufigkeit solcher Änderungen analysiert [WBM99; Koe99; Koe02; NCO04].

Variierter Parameter	Zeitdifferenz zwischen Training und Test
Datensatz	OpenSSH
Anzahl der Instanzen	$n_{train} = 4$, $n_{test} = 10$
Intervall zw. Training und Test	$\Delta_t = \{0, 5, 1, 2, 3, 4, 5, 6, 17\}$ Tage
Klassifizierungsverfahren	MNB-Klassifizierer
Transformation der Daten	TF mit Normalisierung
Metrik	Erkennungsrate

Tabelle 6.5.: Parameter zur Untersuchung der zeitlichen Robustheit

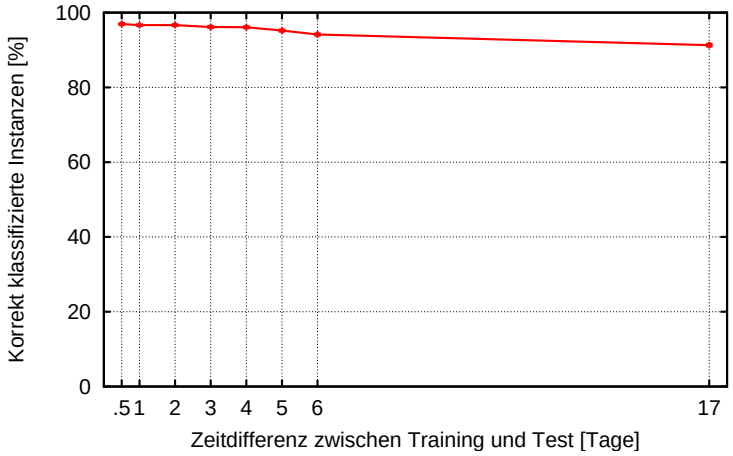


Abbildung 6.7.: Die Erkennungsrate sinkt bei steigender Zeitdifferenz Δ_t zwischen Training und Test. In der Grafik ist auch die Standardabweichung des Mittelwertes abgebildet (Dicke der Punkte), die hier zu vernachlässigen ist.

Ein wichtiges Ergebnis dieser Studien ist die Unterscheidung zwischen *inhaltlichen* (Änderungen an Text und Bildern) und *strukturellen* Änderungen (Änderung der Link-Struktur). Empirische Untersuchungen zeigen, dass im Internet die inhaltlichen Änderungen, die lediglich einen kleinen Teil einer Webseite verändern, dominieren, während strukturelle Änderungen seltener vorkommen.

Da kleine inhaltliche Änderungen die Paketgrößen-Häufigkeitsverteilung nur geringfügig beeinflussen, ist zu erwarten, dass das verwendete Website-Fingerprinting-Verfahren auf Basis von Paketgrößen-Häufigkeiten auch bei größeren Zeitdifferenzen zwischen Training und Test hohe Erkennungsraten liefert.

Zur Untersuchung dieser Hypothese werden die Webseiten über einen längeren Zeitraum über einen OpenSSH-Tunnel abgerufen. Der Abstand zwischen Training und Test Δ_t kann dadurch zwischen 12 Stunden und 17 Tagen variiert werden. Die verwendeten Parameter sind in Tabelle 6.5 aufgelistet. Aus Abbildung 6.7 lassen sich die Erkennungsraten für unterschiedliche Zeitabstände ablesen. Auch nach 17 Tagen werden noch 91,3 % der Instanzen korrekt erkannt.

In der Abbildung ist ein schwacher negativer Zusammenhang zwischen Erkennungsraten und der Zeitdifferenz zu erkennen, wobei die Unterschiede zur ersten Messung (nach

12 Stunden) erst ab dem vierten Tag statistisch signifikant sind (vgl. Tabelle A.3). Eine Regressionsanalyse deutet auf einen exponentiellen Zusammenhang hin: accuracy $\approx 97,11 \cdot e^{-3.735 \cdot 10^{-3} x}$.

Das Regressionsmodell ist angesichts des Beobachtungszeitraums von 17 Tagen allerdings nur eingeschränkt geeignet, um Aussagen über längere Zeiträume zu machen. Bei Liberatore et al. nehmen die Erkennungsraten allerdings auch nach einem Monat nur um etwa 10 % ab [LL06], so dass davon auszugehen ist, dass auch der MNB-Klassifizierer eine hohe zeitliche Robustheit aufweist.

Bewertung Der MNB-Klassifizierer erzielt beim OpenSSH-Datensatz auch nach mehreren Tagen noch eine hohe Genauigkeit. Der Angreifer muss die Datenbank mit den Trainingsdaten also nicht unbedingt täglich aktualisieren.

Der MNB-Klassifizierer weist zudem eine Eigenschaft auf, die ihn für Website-Fingerprinting über längere Zeiträume besonders geeignet erscheinen lässt: Manning et al. bescheinigen dem Naïve-Bayes-Klassifizierer eine gute Anpassungsfähigkeit an sich langsam verändernde Daten (sog. *concept drift*) [MRS07, 269]: Der Klassifizierer kann ein bereits antrainiertes Modell an graduelle Veränderungen anpassen, wenn er kontinuierlich mit aktualisierten Trainingsdaten versorgt wird.

7

Vergleich datenschutzfreundlicher Techniken

Der in Kapitel 6 untersuchte MNB-Klassifizierer hat sich als leistungsfähiges Website-Fingerprinting-Verfahren erwiesen. Ein wesentliches Ziel dieser Arbeit ist die Untersuchung der in Kapitel 4 vorgestellten datenschutzfreundlichen Übertragungstechniken hinsichtlich ihres Schutzes vor Website-Fingerprinting. In diesem Kapitel werden die Ergebnisse einer Testreihe mit den einzelnen Übertragungssystemen vorgestellt.

Zunächst werden zwei Evaluationsmetriken, Cosine Similarity und Informationsgehalt, vorgestellt, die für die Analyse der Ergebnisse benötigt werden. Daran schließt sich die Formulierung der Untersuchungshypothesen an, die anhand der Ergebnisse überprüft werden sollen. Die erzielten Erkennungsraten werden miteinander verglichen und durch visuelle Auswertung der Paketgrößenhistogramme und der erwähnten Evaluationsmetriken erklärt.

7.1. Verwendete Evaluationsmetriken

Um zu ermitteln, wie gut verschiedene datenschutzfreundliche Techniken vor Website-Fingerprinting-Angriffen schützen, wird die *Erkennungsrate* verwendet, die der MNB-Klassifizierer erzielt (vgl. Abschnitt 6.4). Je weniger Instanzen korrekt klassifiziert werden, desto besser schützt ein Übertragungsverfahren vor dem vorgestellten Website-Fingerprinting-Verfahren.

Zur Analyse der Klassifizierungsleistung werden zwei weitere Evaluationsmetriken verwendet, die in diesem Abschnitt eingeführt werden.

7.1.1. Cosine Similarity

Cosine Similarity ist eine Ähnlichkeitsmetrik, die im Information Retrieval häufig zur Suche von Textdokumenten anhand von Suchanfragen verwendet wird [MRS07, 121]. Eine weitere Anwendung ist das Clustering von Textdokumenten sowie die Evaluation der Güte von Clustering-Verfahren [SKK00]. Die Instanzen werden dabei wie in Abschnitt 6.1

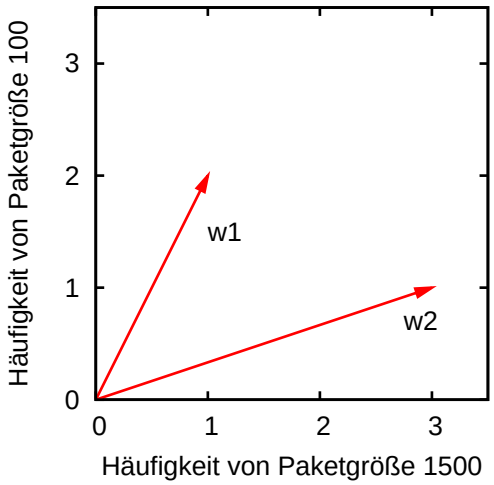


Abbildung 7.1.: Vereinfachtes Beispiel zur Repräsentation zweier Instanzen als Häufigkeitsvektoren $\mathbf{w}_1$ und $\mathbf{w}_2$.

beschrieben als Häufigkeitsvektoren dargestellt. Liegen die Termhäufigkeitsvektoren $\mathbf{f}_{d_1}$ und $\mathbf{f}_{d_2}$ zweier Dokumente vor, wird die Cosine Similarity wie folgt berechnet:

$$\text{sim}(d_1, d_2) = \frac{\mathbf{f}_{d_1} \cdot \mathbf{f}_{d_2}}{\|\mathbf{f}_{d_1}\| \, \|\mathbf{f}_{d_2}\|} \tag{7.1}$$

Im Zähler wird das Skalarprodukt der Vektoren berechnet, im Nenner das Produkt der euklidischen Längen der beiden Vektoren. Der Wert der Metrik entspricht dem Kosinus des Winkels zwischen den beiden Vektoren – je kleiner der Winkel, desto größer der Kosinus und desto ähnlicher sind sich die beiden Dokumente. Durch die Normierung ergibt sich ein Wertebereich im Intervall $[0; 1]$. Der berechnete Wert entspricht dem Winkel im Bogenmaß; bei einem Wert von 1 (entspricht 1π) beträgt der Winkel gerade 0°.

Die Cosine Similarity lässt sich auch für Paketgrößen-Häufigkeitsvektoren beim Website-Fingerprinting berechnen. Abbildung 7.1 veranschaulicht dies anhand von zwei einfachen Häufigkeitsvektoren, die lediglich aus zwei Paketgrößen bestehen. Die Vektoren $\mathbf{w_1} = (1, 2)$ und $\mathbf{w_2} = (3, 1)$ wurden wie in Abschnitt 6.1 demonstriert aus den Instanzen w_1 = ”100 1500 100” und w_2 = ”1500 100 1500 1500” gebildet, wobei $t_1 = 1500$ und $t_2 = 100$. Der Wert der Ähnlichkeitsmetrik beträgt $\text{sim}(\mathbf{w}_1, \mathbf{w}_2) = \frac{1 \cdot 3 + 2 \cdot 1}{\sqrt{1+4}\sqrt{9+1}} \approx 0,71$.

Die Cosine Similarity dient als Basis zur Erklärung der Klassifizierungsergebnisse. Hierzu werden die Homogenität der Instanzen innerhalb einer Klasse (Intra-Klassen-Ähnlichkeit) bzw. die Unterschiede der Instanzen zwischen den Klassen (Inter-Klassen-Ähnlichkeit) ermittelt. Ist die Ähnlichkeit der Instanzen, die zur selben Klasse gehören, größer als die Ähnlichkeit über Klassengrenzen hinweg, ist ein gutes Klassifizierungsergebnis zu erwarten. Andernfalls besteht die Gefahr, dass das gelernte Modell die Klassen nicht sauber voneinander trennen kann. Eine ausführliche Erläuterung der Vorgehensweise enthält Abschnitt 7.5.

7.1.2. Informationsgehalt der Häufigkeitsverteilungen

Entropie der Paketgrößen-Häufigkeitsverteilung

Eine weitere Metrik zur Analyse der Klassfizierungsergebnisse ist die Entropie der Paketgrößen-Häufigkeitsverteilung eines Datensatzes. Die informationstheoretische Definition der Entropie geht auf Shannon zurück [Sha48]. Die beobachteten Paketgrößen entsprechen den Symbolen im informationstheoretischen Modell. Der *Informationsgehalt des Symbols* x_i ist

$$h(x_i) = -\log_2 p(X = x_i) = \log_2 \frac{1}{p(X = x_i)} \tag{7.2}$$

wobei $p(X = x_i)$ die Wahrscheinlichkeit ist, dass Symbol x_i zu beobachten ist. Die Einheit von $h(x_i)$ ist *Bit*. Je seltener ein Symbol auftritt, desto höher ist die daraus gewonnene Information. Zur Berechnung der *Entropie der Häufigkeitsverteilung* von n unterschiedlichen Symbolen wird der durchschnittliche Informationsgehalt ermittelt:

$$H(X) = -\sum_{i=1}^{n} p(X = x_i)\ \log_2 p(X = x_i) \tag{7.3}$$

Die Entropie ist ein Maß für die Unsicherheit, die bei der Prognose des jeweils nächsten Symbols besteht. Sie erreicht ihren Maximalwert $H_{\max}(X) = \log_2 n$ für den Fall, dass alle Symbole gleich wahrscheinlich sind. Im anderen Extremfall – wenn immer das gleiche Symbol auftritt – gilt $H(X) = 0$.

Zum Vergleich unterschiedlicher Datensätze ist $H(X)$ nur eingeschränkt geeignet, da es von n abhängt. Es bietet sich an, hierfür die normierte Entropie $H_{\text{norm}} = \frac{H(X)}{H_{\max}(X)}$ zu verwenden, die im Bereich $[0; 1]$ liegt. Je niedriger der Wert von H_{norm}, desto größer ist die Redundanz in der Häufigkeitsverteilung.

Der Wert von H_{norm} ist spezifisch für die zu Grunde liegende Paketgrößen-Häufigkeitsverteilung. Er ist also völlig unabhängig vom verwendeten Klassifizierungssystem und daher besonders gut zum Vergleich unterschiedlicher Datensätze und Klassifizierer geeignet. Leider geben die Autoren in den bisherigen Publikationen lediglich die Entropie $H(X)$ an [Dan02; SSW+02; LL06], die sich aus dem oben genannten Grund zum Vergleich nicht so gut eignet.

Durchschnittlicher Informationsgehalt der Instanzen

Die Entropie der Paketgrößen-Häufigkeitsverteilung sagt jedoch nichts über den Informationsgehalt von einzelnen Instanzen aus. Dieser hängt davon ab, aus wie vielen Paketen eine Instanz besteht und wie die Auftretenshäufigkeiten verteilt sind. Unterstellt man, dass die Symbole *unabhängig* voneinander auftreten, ermittelt sich der Informationsgehalt einer Folge von m Symbolen, die der Verteilungsfunktion $p(X = x)$ entstammen, als

$$h_{\text{string}}(x_1, x_2, ..., x_m) = m \cdot H(X) \tag{7.4}$$

Der Informationsgehalt wird in der Literatur häufig zur Abschätzung der Anzahl der Webseiten, die durch ein Website-Fingerprinting-Verfahren unterschieden werden können, verwendet. Danezis ermittelt für die Verteilung der Dateigröße der HTML auf einem Webserver eine Entropie von 9,8 Bit [Dan02]. Eine obere Schranke für die Anzahl der unterscheidbaren Webseiten ist somit $2^{9,8} \approx 891$.

Liberatore et al. verwenden zur Klassifizierung von Webseiten die auftretenden IP-Paketgrößen [LL06]. Bei jedem Webseiten-Abruf sind daher mehrere Symbole zu berücksichtigen. Die Entropie der Paketgrößenverteilung beträgt $H(X) = 7,53$ Bit. Zur Berechnung der Informationsmenge, die dem Jaccard-Klassifizierer bei der Klassifizierung zur Verfügung steht, ermitteln die Autoren zunächst die durchschnittliche Anzahl der aufgetretenen Paketgrößen pro Abruf, $m_{\text{mean}} = 36,87$. Sie unterstellen, dass die Paketgrößen voneinander unabhängig sind und berechnen den durchschnittlichen Informationsgehalt, der dem Jaccard-Klassifizierer pro Webseite zur Verfügung steht, wie folgt:

$$h_{\text{Jaccard}}(X) = m_{\text{mean}} \cdot H(X) - \log_2(\lfloor m_{\text{mean}} \rfloor !) \tag{7.5}$$

Als durchschnittlichen Informationsgehalt ermitteln sie auf diese Weise $h_{\text{Jaccard}}(X) = 36,87 \cdot 7,53 - \log_2(36!) \approx 137$ Bit. Daraus schließen die Autoren, dass sich maximal 2^{137} Seiten mit ihrem Verfahren unterscheiden lassen.

Die Formel für die Berechnung von h_{Jaccard} haben Liberatore et al. offenbar von [SSW+02] übernommen. Sie berücksichtigt nach Auffassung der Autoren, dass der Jaccard-Klassifizierer die Reihenfolge der Paketgrößen ignoriert. In beiden Veröffentlichungen versäumen es die Autoren allerdings, eine stichhaltige Begründung für die Subtraktion von $\log_2(\lfloor m_{\text{mean}} \rfloor !)$ anzugeben – diese erscheint gar nicht nötig, da der Entropiewert $h_{\text{string}}(X) = m_{\text{mean}} \cdot H(X)$ ja bereits von der Reihenfolge der Paketgrößen unabhängig ist.

Unabhängig von den Unklarheiten hinsichtlich der Berücksichtigung der Reihenfolge überschätzt der berechnete Wert $2^{h_{\text{Jaccard}}(X)}$ die tatsächliche Anzahl der voneinander unterscheidbaren Seiten erheblich. Zum einen setzt Formel (7.4) voraus, dass die Paketgrößen voneinander unabhängig sind. Gerade dies ist beim Netzwerkverkehr jedoch nicht der Fall, da jedes Paket von der Gegenseite mit einem TCP-Acknowledgement bestätigt werden muss. Zum anderen wird bei dieser Berechnung unterstellt, dass sich die $2^{h_{\text{Jaccard}}(X)}$ Seiten auch tatsächlich voneinander unterscheiden und keine Kollisionen auftreten. Alle Seiten müssen also unterschiedliche Paketgrößen-Häufigkeitsverteilungen aufweisen. Auch dies dürfte in der Realität nicht der Fall sein. Daher kommt es zu einer deutlichen Überschätzung des wahren Informationsgehalts.

Verbesserung Es bietet sich an, zur Ermittlung der Entropie der Paketgrößen-Häufigkeitsverteilung dieselben Konzepte anzuwenden wie bei der Ermittlung der Entropie von natürlicher Sprache. Dort werden anstelle von einzelnen Buchstaben üblicherweise die Wahrscheinlichkeitsverteilungen von Bi- oder Trigrammen verwendet. Dadurch lässt sich die Entropie erheblich genauer bestimmen. Auch bei Paketgrößen-Häufigkeitsverteilungen könnte die Berücksichtigung von Paaren oder Gruppen, die aus mehreren Paketgrößen bestehen, eine präzisere Abschätzung ermöglichen.

Entropie von multisets

Zur Evaluierung des MNB-Klassifizierers sind Formeln (7.4) und (7.5) nicht geeignet: Bei der Betrachtung des Informationsgehalts, der dem Jaccard-Klassifizierer zur Verfügung steht, wird die Häufigkeit der einzelnen Paketgrößen vernachlässigt. Da der MNB-Klassifizierer mit Vektoren bzw. Multimengen *(multisets)* arbeitet, muss die Berechnung der Entropie entsprechend angepasst werden.

Bonchis et al. [BIC07, 168] leiten den Informationsgehalt einer Multiset-Instanz $y = x_1^{m_1}, x_2^{m_2}, \ldots, x_n^{m_n} = (m_1, m_2, \ldots, m_n)$ mit den Häufigkeitswerten m_i aus der Wahrschein-

lichkeit ihres Auftretens ab. Die Auftretenswahrscheinlichkeit einer Instanz ergibt sich über die Multinomialverteilung $\binom{\sum_{i=1}^{n} m_i}{m_1, m_2, \ldots, m_n} \prod_{i=1}^{n} p_i^{m_i}$ als

$$p\left[Y_{\text{multiset}} = (m_1, m_2, \ldots, m_n)\right] = \frac{(\sum_{i=1}^{n} m_i)!}{\prod_{i=1}^{n} (m_i!)} \prod_{i=1}^{n} p_i^{m_i} \tag{7.6}$$

wobei p_i die Auftretenswahrscheinlichkeit von x_i (also einer bestimmten Paketgröße) ist. Im Zähler des Bruchs wird die Fakultät der Länge der Instanz berechnet; das Ergebnis ist die Anzahl der Permutationsmöglichkeiten einer Zeichenkette dieser Länge. Im Nenner werden die Permutationsmöglichkeiten für die einzelnen Paketgrößen des Multisets aufmultpliziert. Der Informationsgehalt ergibt sich dann durch Anwendung von Formel (7.2):

$$h_{\text{multiset}}\left[Y_{\text{multiset}} = (m_1, m_2, \ldots, m_n)\right] = \log_2 \frac{1}{p(x)} = \log_2 \frac{\prod_{i=1}^{n} (m_i!)}{(\sum_{i=1}^{n} m_i)! \prod_{i=1}^{n} p_i^{m_i}} \tag{7.7}$$

Der durchschnittliche Informationsgehalt von k Multiset-Instanzen entspricht dem Mittelwert der Informationsgehalte aller Instanzen:

$$\overline{h}_{\text{multiset}}(Y) = \frac{1}{k} \sum_{j=1}^{k} h_{\text{multiset}}(Y = y_j) \tag{7.8}$$

Auch der durchschnittliche Informationsgehalt nach Formel (7.8) ist lediglich eine obere Schranke für die tatsächliche Entropie der Häufigkeitsverteilung der Multisets, da die x_i, die Paketgrößen, nicht voneinander unabhängig sind Aus diesem Grund wird bei der Analyse der datenschutzfreundlichen Übertragungstechniken darauf verzichtet, die Anzahl der unterscheidbaren Seiten abzuschätzen. Der Informationsgehalt wird lediglich zum Vergleich der Komplexitäten der Paketgrößen-Häufigkeitsverteilungen verwendet.

7.2. Untersuchungshypothesen

Die Untersuchungsergebnisse in Kapitel 6 lassen erwarten, dass die Paketgrößen-Häufigkeitsverteilung bei datenschutzfreundlichen Übertragungstechniken, die lediglich den Datenverkehr verschlüsseln, charakteristische Merkmale enthält. Diese Techniken würden dann keinen Schutz gegen Website-Fingerprinting-Angriffe mit dem MNB-Klassifizierer bieten. Werden zusätzlich zur Verschlüsselung jedoch *Traffic-Shaping*-Maßnahmen, welche die Häufigkeitsverteilung der Paketgrößen beeinflussen, ergriffen, ist zu erwarten, dass die Erkennungsraten bei Website-Fingerprinting-Angriffen erheblich geringer sind. Diese Vermutungen werden zu folgenden Hypothesen zusammengefasst:

Kein-Traffic-Shaping-Hypothese Da keines der untersuchten Single-Hop-Systeme in der Standardkonfiguration Traffic-Shaping durchführt, ist zu erwarten, dass diese Übertragungstechniken nur geringen Schutz vor Website-Fingerprinting bieten, der MNB-Klassifizierer müsste also hohe Erkennungsraten erzielen.

Variierter Parameter	Datensatz und Transformation
Datensatz	OpenSSH, OpenVPN, CiscoVPN, Stunnel, Tor, Tor+Tinyproxy, JonDonym, Shalon
Anzahl der Instanzen	$n_{train} = 4$, $n_{test} = 10$
Intervall zw. Training und Test	$\Delta_t = \{12, 24\}$ Stunden
Klassifizierungsverfahren	MNB-Klassifizierer
Transformation der Daten	keine, TF, IDF, TF-IDF; jeweils mit/ohne Normalisierung
Metrik	Erkennungsrate

Tabelle 7.1.: Parameter zur Untersuchung der Übertragungstechniken

Traffic-Shaping-Hypothese Die Multi-Hop-Systeme Tor und JonDonym tragen der Gefahr von Traffic-Analyse-Angriffen Rechnung, indem sie Traffic-Shaping-Maßnahmen implementieren: die Nutzdaten werden vor dem Versand in Datenstrukturen fixer Größe verpackt. Es ist daher zu erwarten, dass sie einen besseren Schutz vor Website-Fingerprinting-Angriffen bieten und die Erkennungsraten daher geringer ausfallen.

Tunnel-Hypothese Datenschutzfreundliche Systeme, die beim Start einen Tunnel anlegen, müssen nicht für jeden HTTP-Request eine neue Verbindung aufbauen, sondern können alle Pakete über den bestehenden Tunnel weiterleiten. Bei Diensten, die jede Verbindung separat weiterleiten, zum Beispiel Stunnel, werden zusätzliche Pakete für den Verbindungsaufbau erzeugt, die zum charakteristischen Muster einer Seite beitragen. Es ist daher zu erwarten, dass die Erkennungsraten bei Stunnel im direkten Vergleich mit den übrigen Verfahren höher sind.

Shalon-Hypothese Das Multi-Hop-System Shalon bietet in der für den Test vorliegenden Version keine Traffic-Shaping-Funktionalität. Zwar werden alle Pakete bei Shalon mehrmals hintereinander verschlüsselt, an der Charakteristik des Datenverkehrs ändert sich dadurch aber nichts. Auch hier sind hohe Erkennungsraten zu erwarten.

Packet-Size-Hypothese Es ist zu erwarten, dass die Erkennungsleistung bei Systemen, die Traffic-Shaping durchführen, von der Paketgröße abhängt. Bei kleinerer Paketgröße sind höhere Erkennungsraten zu erwarten, da dann eine größere Anzahl von Paketgrößen zu beobachten ist. Demnach müssten die Erkennungsraten bei Tor, bei dem eine *cell size* von 512 Byte zum Einsatz kommt, höher sein als bei JonDonym, das 998 Byte große Mix-Pakete verwendet.

7.3. Erzielte Erkennungsraten

7.3.1. Untersuchungsbedingungen

Die Untersuchungsbedingungen für den Vergleich der verschiedenen datenschutzfreundlichen Systeme sind in Tabelle 7.1 aufgeführt. Die Untersuchungen fanden unter denselben Rahmenbedingungen statt wie die Versuche in Kapitel 6: Mit jedem Übertragungsverfahren wurden die in Abschnitt 5.4.1 beschriebenen 775 Webseiten über einen Zeitraum von mehreren Tagen abgerufen.

System	Δ_t	$N_{\textbf{gesamt}}$	Geschwindigkeit
Tor	12 h	8 510	85 Instanzen/h
Tor+Tinyproxy	12 h	15 165	77 Instanzen/h
Stunnel	12 h	21 154	480 Instanzen/h
CiscoVPN	12 h	29 770	562 Instanzen/h
JonDonym	24 h	33 615	320 Instanzen/h
Shalon	24 h	58 688	543 Instanzen/h
OpenVPN	24 h	75 724	476 Instanzen/h
OpenSSH	24 h	131 960	498 Instanzen/h

Tabelle 7.2.: Verwendete Zeitabstände Δ_t zwischen Training und Test, Gesamtzahl N_{gesamt} der Instanzen in der Grundgesamtheit sowie durchschnittliche Anzahl der Instanzen, die pro Stunde heruntergeladen wurden, für die untersuchten Übertragungstechniken

Für die Evaluation werden zwei unterschiedliche Zeitdifferenzen Δ_t verwendet, da die Anzahl der verfügbaren Instanzen bei den verschiedenen Systemen stark variiert. Es gibt zwei Faktoren, welche die Anzahl der Instanzen pro Messung beeinflussen:

- Aus organisatorischen Gründen standen für die einzelnen Messungen unterschiedlich viele Tage zur Verfügung (zwischen drei und acht Tagen).
- Die Übertragungstechniken unterscheiden sich zudem in der Performanz: Vor allem bei Tor ist die Anzahl der Instanzen daher im Verhältnis zu den anderen Systemen deutlich geringer.

Aus diesen Gründen werden die einzelnen Systeme mit unterschiedlichen Intervallen Δ_t evaluiert: 12 bzw. 24 Stunden. Dieses Vorgehen ermöglicht die maximale Ausnutzung der zur Verfügung stehenden Testdaten für jedes System. Trainings- und Testdaten können dadurch aus einem größeren Zeitfenster ausgewählt werden, was zu repräsentativeren Stichproben führt.

Systeme, die mit 12 Stunden Zeitdifferenz evaluiert werden, schneiden dadurch nicht automatisch besser ab: In Abschnitt 6.8 wurde gezeigt, dass bei Zeitabständen von bis zu vier Tagen die Erkennungsraten keine signifikanten Unterschiede aufweisen. Tabelle 7.2 listet die Anzahl der Instanzen in der Grundgesamtheit zusammen mit den Zeitabständen Δ_t auf.

Darüber hinaus enthält die Tabelle Angaben zur erzielten Download-Geschwindigkeit. Hier ist vor allem auf die geringe Geschwindigkeit von Tor hinzuweisen, die dadurch zustande kommt, dass bei Tor viele Seiten nicht innerhalb der Maximalzeit (90 Sekunden, vgl. Abschnitt 5.4.2) vollständig heruntergeladen wurden.

7.3.2. Single-Hop-Systeme

Abbildung 7.2 zeigt die Erkennungsraten (Anteil der korrekt klassifizierten Testinstanzen) für die untersuchten Single-Hop-Systeme Stunnel, OpenSSH, OpenVPN und CiscoVPN. Bei allen Systemen lassen sich Erkennungsraten über 90 % erzielen. Die *Kein-Traffic-Shaping-Hypothese* wird durch die Ergebnisse bestätigt.

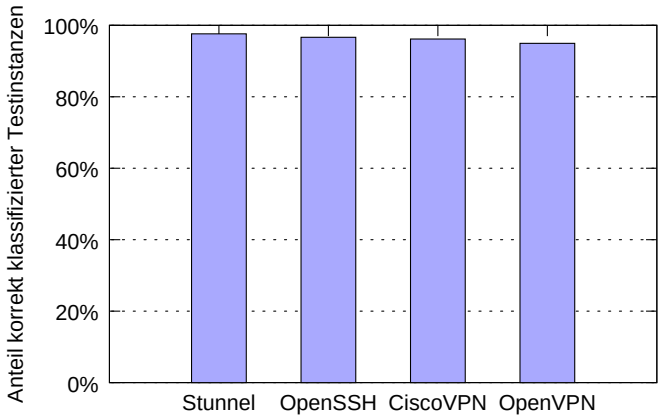


Abbildung 7.2.: Klassifizierungsergebnisse für Single-Hop-Systeme – der MNB-Klassifizierer erzielt bei allen Systemen hohe Erkennungsraten

Bei Stunnel, das keinen stehenden TLS-Tunnel aufbaut, sondern für jede TCP-Verbindung eine neue TLS-Verbindung erzeugt, erzielt der Klassifizierer die höchste durchschnittliche Erkennungsrate von 97,64 %, die *Tunnel-Hypothese* wird also bestätigt.

Selbst beim CiscoVPN, bei dem keine Pakete in Senderichtung aufgezeichnet wurden (vgl. Abschnitt 4.1.4), ist die Erkennungsrate sehr hoch. Offenbar reichen bereits die *empfangenen* IP-Pakete für eine trennscharfe Klassifizierung aus. Dass die *gesendeten* Pakete jedoch durchaus einen Beitrag zur Klassifizierung leisten, zeigt eine (nicht abgebildete) Untersuchung des OpenSSH-Datensatzes: werden hier lediglich die empfangenen Pakete betrachtet, fällt die Erkennungsrate um etwa 10 %.

7.3.3. Multi-Hop-Systeme

In Abbildung 7.3 sind die Erkennungsraten der Multi-Hop-Systeme Tor, Tor+Tinyproxy, JonDonym und Shalon dargestellt.

Die Erkennungsraten von Shalon, das kein Traffic-Shaping durchführt, liegen auf einem ähnlichen Niveau wie bei den Single-Hop-Systemen. Die mehrfache Verschlüsselung des Datenstroms reicht also erwartungsgemäß nicht aus, um die charakteristischen Muster im Datenverkehr zu eliminieren. Die *Shalon-Hypothese* wird durch die Ergebnisse bestätigt.

Tor und JonDonym verwenden Datenstrukturen fixer Größe für den Transport der Nutzdaten. Die Erkennungsraten sind daher erheblich niedriger als bei den Single-Hop-Systemen, die kein Traffic-Shaping durchführen. Die *Traffic-Shaping-Hypothese* wird durch diese Ergebnisse also bestätigt.

Überraschend ist, dass die Erkennungsraten bei beiden Tor-Messungen niedriger sind als bei JonDonym: Während sich bei JonDonym noch etwa 20 % der Instanzen identifizieren lassen, werden bei Tor nur etwa 3 % der Instanzen korrekt klassifiziert. Dabei macht es keinen Unterschied, ob der Datenverkehr bei Tor über Tinyproxy geleitet wird oder nicht. Die *Packet-Size-Hypothese* kann also anhand der Versuchsdaten nicht bestätigt werden.

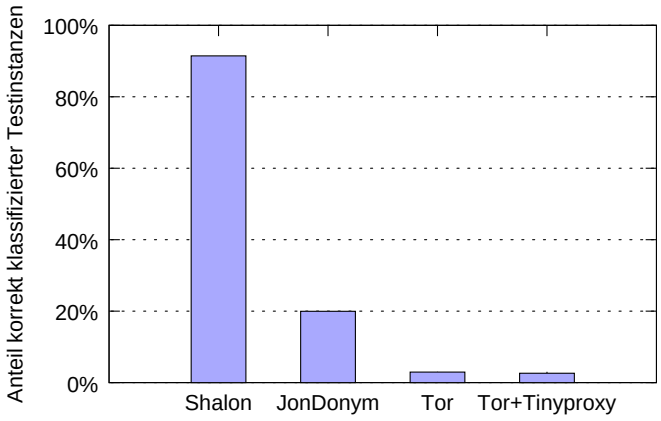


Abbildung 7.3.: Klassifizierungsergebnisse für Multi-Hop-Systeme – geringere Erkennungsraten bei Tor und JonDonym, hohe Erkennungsrate bei Shalon

Single-Hop			**Multi-Hop**		
Stunnel	97,64 (0,16)	TF-N	Shalon	91,45 (0,40)	TF-N
OpenSSH	96,65 (0,22)	TF-N	JonDonym	19,97 (0,47)	N
CiscoVPN	96,17 (0,24)	TF-N	Tor	2,96 (0,22)	N
OpenVPN	94,94 (0,30)	TF-N	Tor+Tinyproxy	2,64 (0,15)	N

Tabelle 7.3.: Durchschnittliche Erkennungsraten für die untersuchten Systeme (Standardabweichung in Klammern). Angegeben ist jeweils die Erkennungsrate, die mit der besten Transformationsmethode erreicht wurde. TF-N: TF-Transformation mit normalisierten Häufigkeitsvektoren; N: normalisierte Häufigkeitsvektoren

7.3.4. Interpretation

Tabelle 7.3 fasst die Erkennungsraten der verschiedenen Systeme zusammen. Der Vergleich der datenschutzfreundlichen Übertragungstechniken fällt im Wesentlichen genau so aus wie erwartet. Lediglich die *Packet-Size-Hypothese* konnte nicht anhand der Erkennungsraten von Tor und JonDonym bestätigt werden.

Einfluss der Optimierungen Die TF-Transformation in Verbindung mit normalisierten Häufigkeitsvektoren erweist sich für die Mehrheit der Systeme als beste Transformationsmethode und bestätigt das Ergebnis der Voruntersuchungen in Abschnitt (6.6). Bei Tor und JonDonym führt die Anwendung der TF-Transformation zu einem Einbruch der Erkennungsraten. Die besten Erkennungsraten werden hier, wie in Tabelle 7.3 dargestellt, mit normalisierten Häufigkeitsvektoren – ohne Anwendung der TF-Transformation – erzielt. Eine mögliche Ursache: Die Logarithmierung der Häufigkeiten verzerrt die Größenverhältnisse in der Häufigkeitsverteilung der Paketgrößen. Bei Tor und JonDonym ist der MNB-Klassifizierer offenbar gerade auf diese Unterschiede angewiesen.

Kein perfekter Schutz durch Anonymisierungsdienste Der Nutzen des Website-Fingerprinting-Angriffs mag bei bei Tor und JonDonym angesichts der niedrigen Erkennungsraten von 3 % bzw. 20 % gering erscheinen. Eine Erkennungsleistung von 3 % ist allerdings bereits wesentlich höher als die Trefferquote beim zufälligen Raten: Bei der vorliegenden Gleichverteilung liegt diese bei $p = \frac{1}{775} \approx 0,13\%$. Aus informationstheoretischer Sicht bieten beide Systeme also *keinen perfekten Schutz* gegen Website-Fingerprinting (vgl. Modell in Abschnitt 2.2). Die Ergebnisse zeigen, dass es auch einem lokal-begrenzten passiver Angreifer gelingen kann, die Beziehungsanonymität der Nutzer eines Anonymisierungsdienstes aufzuheben.

Architektur nicht als Ursache Aus den Ergebnissen kann man *nicht* schließen, dass Single-Hop-Systeme allein wegen ihrer Architektur vor Website-Fingerprinting-Angriffen schlechter schützen als Multi-Hop-Systeme. Die Übertragungstechniken, die guten Schutz vor Website-Fingerprinting bieten, verwenden im Anwendungsprotokoll Datenstrukturen mit fester Größe, während die getesteten Single-Hop-Systeme praktisch gar kein Traffic-Shaping durchführen. Die hohen Erkennungsraten bei Shalon zeigen, dass auch Multi-Hop-Systeme für Website-Fingerprinting anfällig sind, wenn sie keine entsprechenden Gegenmaßnahmen enthalten.

Die Ergebnisse werfen die Frage auf, warum die Erkennungsrate bei Tor lediglich 3 % beträgt, während bei JonDonym etwa 20 % der Instanzen identifiziert werden können. Wodurch entsteht diese Differenz? In den folgenden Abschnitten wird nach Gründen für diese Differenz gesucht.

7.4. Analyse der Histogramme

Möglicherweise lassen sich die unterschiedlichen Erkennungsraten auf charakteristische Muster in den Histogrammen zurückführen. Daher werden im Folgenden die Histogramme der Paketgrößenverteilung aller Instanzen in der Grundgesamtheit analysiert und miteinander verglichen.

7.4.1. Absolute Häufigkeiten

Die Histogramme für die nicht-transformierten Rohdaten sind in Abbildung 7.4 dargestellt, wobei die Darstellungsmethode aus Abschnitt 5.3.2 zur Anwendung kommt: Pakete in Empfangsrichtung werden durch positive, Pakete in Senderichtung durch negative Paketgrößen gekennzeichnet. Beim CiscoVPN fehlen wie in Abschnitt 4.1.4 angedeutet die Pakete in Senderichtung.

In den Histogrammen von OpenSSH, OpenVPN, Stunnel und Shalon dominiert die Paketgröße $+1500$, was der MTU auf den Verbindungsstrecken vom Testrechner zu den getesteten Diensten entspricht. Darüber hinaus treten viele kleine Pakete (mit 50 bis 100 Byte) in Empfangs- und Senderichtung auf – das sind die TCP-Acknowledgements des Clients, die keine Nutzdaten transportieren. Beim getesteten CiscoVPN, das zum Transport der IPsec-ESP-Pakete UDP verwendet, dominieren hingegen die Größen $+560$ und $+936$.

Bei Tor treten neben den TCP-Acknowledgements (-52,$+52$) primär die Paketgrößen -638, $+638$, $+1150$ und $+1500$ auf – was auf Tors *cell size* von 512 Byte zurückgeht

System	$n_{\text{sizes\_sent}}$	$n_{\text{sizes\_rcvd}}$	$n_{\text{sizes\_total}}$
OpenSSH	216	204	420
OpenVPN	1449	1449	2898
Stunnel	160	1445	1605
CiscoVPN	0	108	108
Tor	141	728	869
Tor+Tinyproxy	598	1330	1928
JonDonym	26	179	205
Shalon	457	1450	1907

Tabelle 7.4.: Anzahl der unterschiedlichen Paketgrößen in Sende- und Empfangsrichtung für alle Instanzen in den jeweiligen Grundgesamtheiten

(vgl. Abschnitt 4.2.2). Da bei JonDonym die Größe eines Mixpakets 998 Byte beträgt (vgl. Abschnitt 4.2.1), dominieren neben den TCP-Acknowledgements nur die Größen -1050,$+1050$ und $+1500$.

7.4.2. Logarithmierte Häufigkeiten

Bei linearer Skalierung der Histogramme sind Paketgrößen mit geringen Häufigkeiten nicht zu erkennen. Eine logarithmierte Darstellung kann hier Abhilfe schaffen (vgl. Abbildung 7.5).

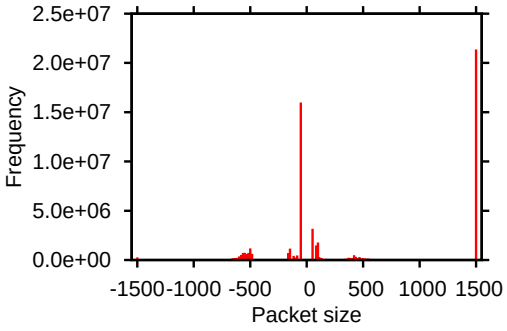
In den logarithmierten Histogrammen ist zu erkennen, dass bei OpenSSH, OpenVPN, Stunnel, CiscoVPN und Shalon praktisch alle Paketgrößen ab 52 Byte auftreten. In der logarithmierten Darstellung unterscheiden sich die Histogramme von JonDonym bzw. Tor von denen der Single-Hop-Systeme erheblich – ihre Histogramme sind wesentlich „zackiger“: durch die Verwendung von Datenstrukturen fixer Größe ist die Verteilung der Paketgrößen dort sehr unstetig.

Die Histogramme erwecken den Eindruck, dass sämtliche Größen zwischen 52 und 1500 Byte auftreten. Tatsächlich kommen jedoch lediglich Vielfache der jeweiligen Blockgröße der Chiffre vor. Bei OpenSSH sind zum Beispiel alle beobachteten Paketgrößen Vielfache von 8. Tabelle 7.4 enthält die Anzahl der unterschiedlichen Paketgrößen im Vergleich.

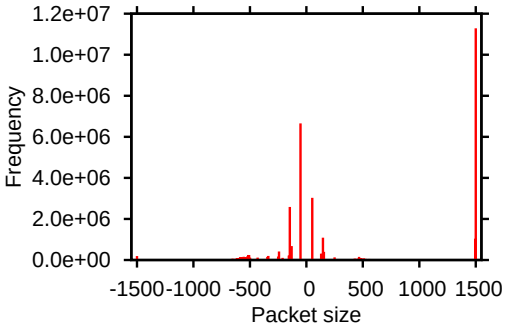
Interpretation

Die Anzahl der unterschiedlichen Paketgrößen variiert erheblich. Es lässt sich jedoch kein Zusammenhang zu den Erkennungsraten erkennen. Bei OpenVPN, das 2898 unterschiedliche Paketgrößen erzeugt, erzielt der Klassifizierer sehr hohe Erkennungsraten, bei Tor+Tinyproxy, bei dem 1928 Paketgrößen auftreten, ist die Erkennungsleistung hingegen sehr schlecht. Eine hohe Anzahl von unterschiedlichen Datenpaketen impliziert also nicht unbedingt eine hohe Erkennungsleistung. Analoges gilt bei einer niedrigen Anzahl von Paketgrößen: Während beim CiscoVPN, bei dem nur 108 Paketgrößen zu beobachten sind, sehr hohe Erkennungsraten erzielt werden, sind diese bei JonDonym, das nur 205 unterschiedliche Paketgrößen erzeugt, relativ gering.

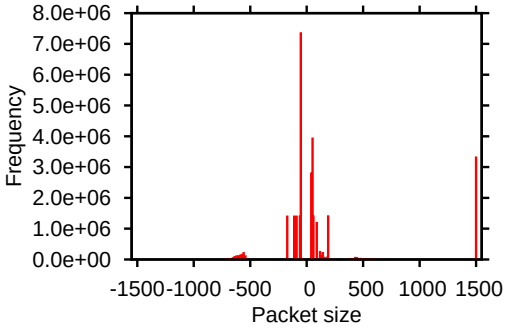
Überraschend ist allerdings, dass bei Tor und JonDonym überhaupt so viele unterschiedliche Paketgrößen generiert werden – die Systeme verwenden schließlich Datenstrukturen



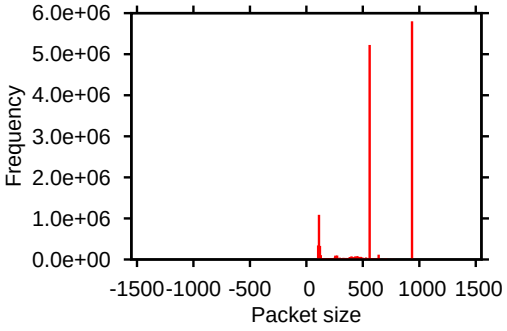
(a) OpenSSH



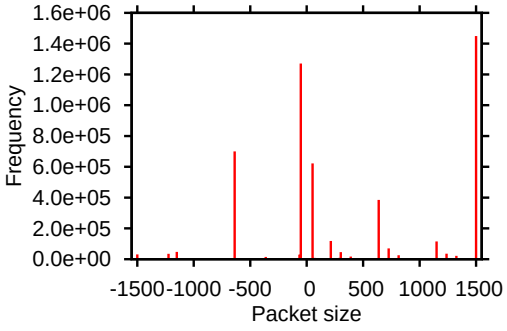
(b) OpenVPN



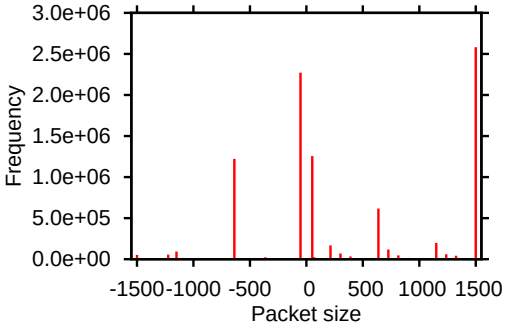
(c) Stunnel



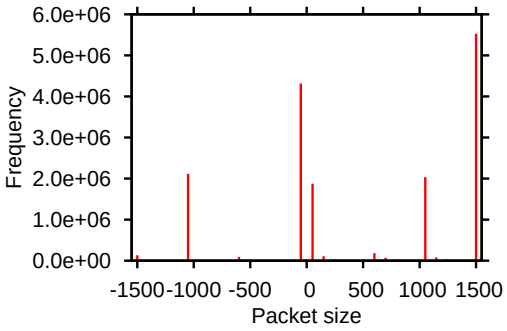
(d) CiscoVPN



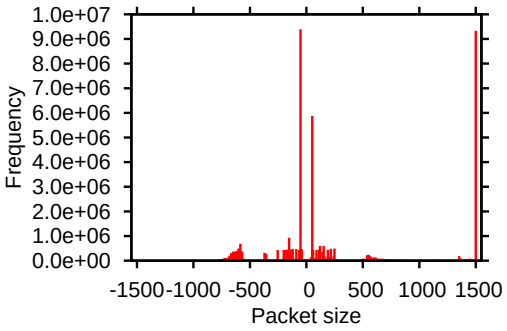
(e) Tor



(f) Tor+Tinyproxy



(g) JonDonym



(h) Shalon

Abbildung 7.4.: Die Histogramme der nicht-transformierten Pakethäufigkeiten unterscheiden sich nur geringfügig (Pakete vom Client zum Server haben negative Paketgröße).

fixer Größe. Dieses Phänomen lässt sich wie folgt erklären: Diese Dienste verwenden zwar intern fixe Paketgrößen für Mix-Pakete bzw. *onion cells*, aber der IP-Stack des Betriebssystems packt zur Optimierung des Durchsatzes möglichst viele Daten in ein IP-Paket – bis zum Erreichen der MTU der Übertragungsstrecke. Dabei werden die internen Datenstrukturen fixer Größe fragmentiert und auf mehrere Pakete aufgeteilt. Durch die Fragmentierung entstehen zwangsläufig einige IP-Pakete mit anderen Paketgrößen, insbesondere beim Leeren des Sendepuffers *(flush)* sowie beim Verbindungsabbau.

Weiterhin fällt auf, dass bei JonDonym die Anzahl der unterschiedlichen Paketgrößen in Senderichtung erheblich geringer ist als in Empfangsrichtung. Diese Diskrepanz ist möglicherweise durch die unterschiedlichen Implementierungen und Implementierungssprachen im Client (Java) bzw. im Mix (C++) zu erklären.

Die Auswertung der Histogramme kann nicht erklären, warum Website-Fingerprinting bei Tor niedrigere Erkennungsraten erzielt als bei JonDonym – im Gegenteil: Man könnte vermuten, dass die Histogramme bei JonDonym wegen der geringeren Anzahl der erzeugten Paketgrößen auch einen geringeren Informationsgehalt hätten, und daher der Klassifizierer bei JonDonym schlechter abschneiden müsste. Diese These wird in Abschnitt 7.6 untersucht.

7.5. Intra- und Inter-Klassen-Ähnlichkeit

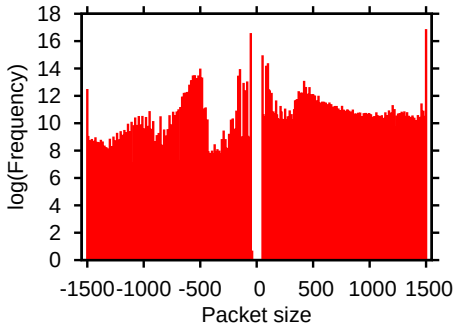
Zur Erklärung der unterschiedlichen Erkennungsraten bei Tor und JonDonym wird nun die Ähnlichkeit der Instanzen *innerhalb* einer Klasse sowie *zwischen* den Klassen analysiert – erstere wird im Folgenden als *Intra-Klassen-Ähnlichkeit* bezeichnet, letztere als *Inter-Klassen-Ähnlichkeit*. Die Ähnlichkeit der Instanzen hat maßgeblichen Einfluss auf das Klassifizierungsergebnis: Ein Klassifizierer kann ein trennschärferes Modell generieren, wenn die Instanzen einer Klasse möglichst homogen sind und die Instanzen verschiedener Klassen möglichst unterschiedlich sind (vgl. die Betrachtungen von Strehl hinsichtlich Intra- und Inter-Cluster-Similarity [Str02, 61]).

Als Ähnlichkeitsmetrik kommt dabei die beim Text Mining häufig verwendete Cosine Similarity zum Einsatz, die in Abschnitt 7.1.1 vorgestellt wurde. Intra- und Inter-Klassen-Ähnlichkeit ergeben sich jeweils als durchschnittlicher Wert der Ähnlichkeiten mehrerer Instanzpaare.

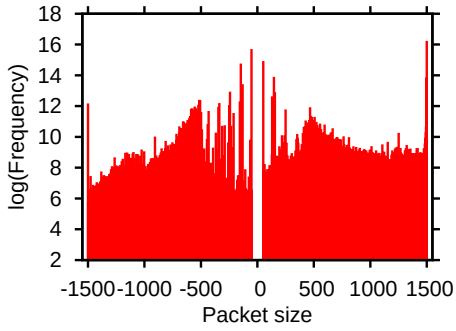
Erkennungsraten-Ähnlichkeiten-Hypothese

Ist der Ähnlichkeitswert für eine gegebene Instanz und die Instanzen *derselben* Klasse genauso groß wie der Ähnlichkeitswert zwischen der Instanz und den Instanzen *aller anderen* Klassen, kann der MNB-Klassifizierer für diese Instanz keine charakteristischen Merkmale finden. Sind die Ähnlichkeitswerte *aller* Instanzpaare völlig identisch, kann der MNB-Klassifizierer nur noch zufällig raten. Ein datenschutzfreundliches Übertragungssystem, das solche Instanzen erzeugt, bietet *optimalen Schutz* gegen den vorgestellten Website-Fingerprinting-Angriff.

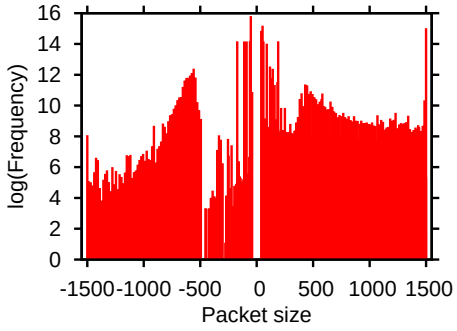
Wenn die (durchschnittlichen) Werte der Intra- und Inter-Klassen-Ähnlichkeit bei einer Klasse also *identisch* sind, deutet dies darauf hin, dass sich die Instanzen in der Klasse im Mittel genauso ähnlich sind wie im Vergleich zu Instanzen außerhalb der Klasse. In diesem Fall ist für diese Klasse eine vergleichsweise niedrige Erkennungsrate zu erwarten, da der Klassifizierer auf Basis der Trainingsdaten kein trennscharfes Modell erzeugen kann.



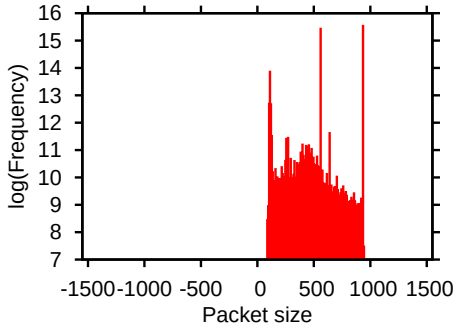
(a) OpenSSH



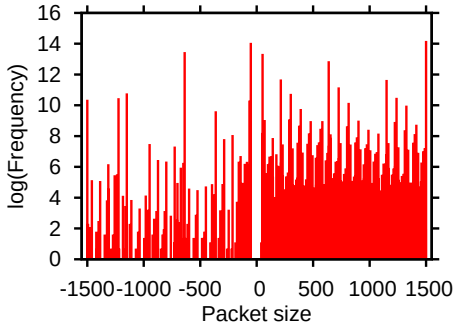
(b) OpenVPN



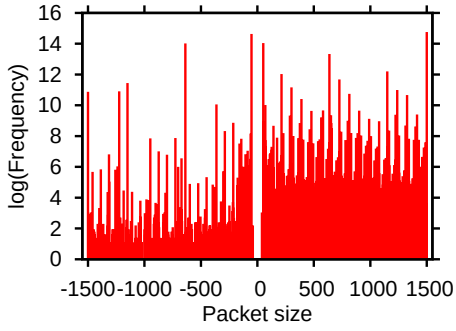
(c) Stunnel



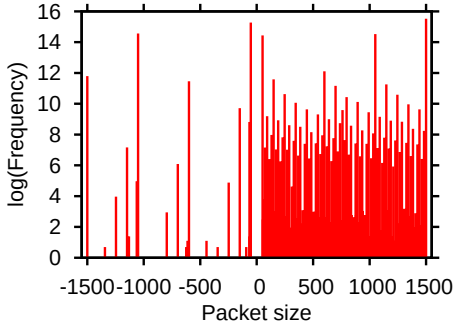
(d) CiscoVPN



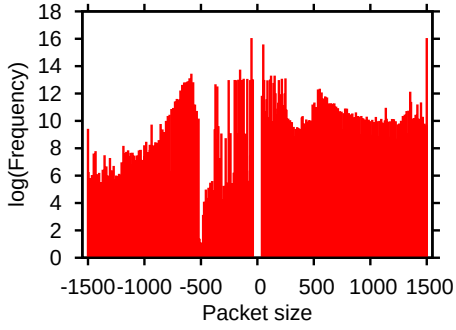
(e) Tor



(f) Tor+Tinyproxy



(g) JonDonym



(h) Shalon

Abbildung 7.5.: Die Histogramme der *logarithmierten* Pakethäufigkeiten unterscheiden sich deutlich (Pakete vom Client zum Server haben negative Paketgröße).

Ist die Differenz hingegen groß, ist zu erwarten, dass die Erkennungsraten hoch sind. Diese Hypothese soll durch Vergleich der durchschnittlichen Intra- und Inter-Klassen-Ähnlichkeitswerte für die Grundgesamtheiten überprüft werden

7.5.1. Vorgehensweise

Berechnung der Intra-Ähnlichkeit für eine Klasse Aus den Instanzen d einer Klasse c werden alle Instanz-Paare $\{d_j, d_k\}, d_j, d_k \in c \wedge j \neq k$ gebildet – bei n_d Instanzen in einer Klasse gibt es $n_{\text{pairs}} = \frac{n_d(n_d-1)}{2}$ Paare. Für alle Paare wird dann die Cosine Similarity $\text{sim}(d_j, d_k)$ berechnet, um die Intra-Klassen-Ähnlichkeit als Mittelwert der einzelnen Ähnlichkeiten $\text{sim}_{\text{intra}}(c) = \frac{1}{n_{\text{pairs}}} \sum_{j,k} \text{sim}(d_j, d_k)$ zu bestimmen.

Anhand des absoluten Wertes der Intra-Klassen-Ähnlichkeit lässt sich ablesen, wie stark sich der Datenverkehr beim Abruf der zugehörigen Webseite im Zeitverlauf geändert hat. Webseiten, die bei jedem Abruf exakt denselben Datenverkehr erzeugen, haben eine Intra-Klassen-Ähnlichkeit von 1.

Berechnung der Inter-Ähnlichkeit für eine Klasse Für jede Instanz d_j einer Klasse c werden Instanz-Paare $\{d_j, d_k\}, d_j \in c \wedge d_k \notin c$ mit allen Instanzen aus anderen Klassen gebildet. Bei n_d Instanzen pro Klasse und n_c Klassen gibt es pro Klasse $n_{\text{pairs}} = n_d^2(n_c - 1)$ Paare. Zur Berechnung der Inter-Klassen-Ähnlichkeit wird der Mittelwert aller Ähnlichkeitswerte für eine Klasse gebildet: $\text{sim}_{\text{inter}}(c) = \frac{1}{n_{\text{pairs}}} \sum_{j,k} \text{sim}(d_{j,c}, d_{k,c})$.

Die Inter-Klassen-Ähnlichkeit drückt aus, wie stark sich die Instanzen einer Klasse von den Instanzen aller anderen Klassen unterscheiden.

Implementierung der Berechnung Auf eine vollständige Analyse der Grundgesamtheit muss wegen dem hohen Zeit- und Speicherplatzbedarf der Berechnungen verzichtet werden. Zur Analyse werden daher aus den einzelnen Datensätzen Stichproben gezogen, die jeweils 11 Instanzen aus einem Zeitraum von 100 Stunden enthalten. Jede Stichprobe enthält also $775 \cdot 11 = 8525$ Instanzen.

Zur effizienten Berechnung der Ähnlichkeitswerte kommen zahlreiche Optimierungen zum Einsatz. Zum einen werden die Ruby-Bibliotheken *linalg*[1] und *narray*[2] verwendet, welche die aufwändigen Vektor-Operationen Skalarprodukt und Norm (euklidische Länge) in nativ übersetztem Code durchführen. Durch Abspeichern der bereits berechneten Vektornorm-Werte in einer Lookup-Tabelle werden unnötige Operationen vermeiden. Beim OpenSSH-Datensatz mit 8525 Instanzen dauert die optimierte Berechnung aller $\binom{8525}{2} \approx 36 \cdot 10^6$ Cosine-Similarity-Werte etwa 30 Minuten, während die ursprüngliche, nicht-optimierte Fassung des Skripts etwa die zehnfache Zeit benötigt hätte.

7.5.2. Vergleich der Ähnlichkeitswerte

In Tabelle 7.5 sind die Werte für die Intra- und Inter-Klassen-Ähnlichkeit für die Datensätze von JonDonym und Tor abgebildet. Dabei handelt es sich jeweils um die durchschnittlichen Ähnlichkeitswerte der 775 Klassen (URLs).

[1] Homepage: http://rubyforge.org/projects/linalg/
[2] Homepage: http://narray.rubyforge.org/

System	$\text{sim}_{\text{intra}}$	$\text{sim}_{\text{inter}}$	Δ_{sim}	Erkennungsrate
OpenVPN	0,9612	0,7701	0,1911	0,9494
CiscoVPN	0,9839	0,8275	0,1564	0,9617
OpenSSH	0,9777	0,8270	0,1507	0,9665
Shalon	0,9662	0,8261	0,1401	0,9145
JonDonym	0,9854	0,9145	0,0709	0,1997
Stunnel	0,9954	0,9337	0,0617	0,9764
Tor	0,9409	0,9037	0,0372	0,0296
Tor+Tinyproxy	0,9459	0,9088	0,0371	0,0264

Tabelle 7.5.: Zwischen den Ähnlichkeitsdifferenzen $\Delta_{\text{sim}} = \text{sim}_{\text{intra}} - \text{sim}_{\text{inter}}$ und den Erkennungsraten aus Abschnitt 7.3.4 besteht ein Zusammenhang. Die Darstellung ist nach Δ_{sim} sortiert.

Die eingangs formulierte *Erkennungsraten-Ähnlichkeiten-Hypothese*, nach der eine Abhängigkeit zwischen den Erkennungsraten des Klassifizierers und der Differenz der Ähnlichkeitswerte $\Delta_{\text{sim}} = \text{sim}_{\text{intra}} - \text{sim}_{\text{inter}}$ besteht, wird durch die Ergebnisse in Tabelle 7.5 bestätigt. Der Pearson-Korrelationskoeffizient nach Formel (3.2) beträgt $r_{xy} = 0,79$, es besteht also ein leicht positiver Zusammenhang. Die Unterschiede zwischen $\text{sim}_{\text{intra}}$ und $\text{sim}_{\text{inter}}$ sind jeweils hochsignifikant (Irrtumswahrscheinlichkeiten beim t-Test für abhängige Stichproben sind kleiner als 0,1 %).

Bei OpenSSH, CiscoVPN und OpenVPN ist die Differenz Δ_{sim} vergleichsweise groß, so dass der MNB-Klassifizierer ein trennscharfes Modell erzeugen kann und eine hohe Erkennungsrate erzielt. Bei Tor und JonDonym sind die Differenzen wesentlich geringer, wobei vor allem bei Tor die Intra-Klassen-Ähnlichkeit nur noch geringfügig höher ist als die Inter-Klassen-Ähnlichkeit. In den Ergebnissen ist ein Ausreißer enthalten: Bei Stunnel erzielt der Klassifizierer hohe Erkennungsraten, obwohl der Wert Δ_{sim} gering ist. Eine große Ähnlichkeitsdifferenz Δ_{sim} ist also offenbar hinreichend für hohe Erkennungsraten, jedoch nicht notwendig.

Erklärung der unterschiedlichen Erkennungsraten von Tor und JonDonym

Die Ähnlichkeitsanalyse liefert eine mögliche Erklärung für die unterschiedlichen Erkennungsraten bei Tor und JonDonym: bei JonDonym ist die Reinheit der einzelnen Klassen höher als bei Tor. Die Instanzen einer Klasse unterscheiden sich bei JonDonym dadurch noch so stark von den Instanzen der übrigen Klassen, dass manchmal eine Klassifizierung möglich ist. Bei Tor sind die Unterschiede innerhalb einer Klasse hingegen offenbar so groß, dass sich nur noch sehr wenige Instanzen ihrer Klasse zuordnen lassen.

Eine mögliche Ursache für den homogeneren Datenverkehr bei JonDonym: Der Datenverkehr wurde über eine relativ gering ausgelastete, festgelegte Kaskade übertragen. Er ist dadurch frei von störenden Einflüssen. Bei Tor wurden die Daten im Messzeitraum dagegen über verschiedene Onion-Router transportiert, die unterschiedliche Tor-Versionen einsetzen, auf unterschiedlichen Betriebssystemen laufen und unterschiedliche Zuverlässigkeit aufweisen.

Hinzu kommt, dass Zuverlässigkeit und Datendurchsatz bei Tor erheblichen Schwankungen unterliegen, wie aktuelle Studien zeigen [WHF07; PPR08]. Die geringe Zuverlässigkeit wirkte sich auch auf die Messung aus: Bei Verwendung von Tor wurde der Download vie-

ler Seiten nicht innerhalb der Maximalzeit von 90 Sekunden abgeschlossen. Dabei waren häufig lange Pausen zu beobachten, in denen der Browser keine Daten empfangen hat. Die Betrachtung der Pakete im Zeitverlauf (nicht abgebildet) zeigt jedoch, dass auch in diesen scheinbaren Pausen Daten übertragen wurden. Dieser Datenverkehr wird vom Tor-Client selbst erzeugt, um im Hintergrund kontinuierlich neue *circuits* zu erzeugen. Die dabei entstehenden, zusätzlichen Datenpakete sind in den jeweiligen Instanzen enthalten und möglicherweise die Ursache für die geringe Ähnlichkeitsdifferenz Δ_{sim} und die niedrigen Erkennungsraten.

7.6. Informationstheoretische Auswertung

Abschließend wird die Entropie der Häufigkeitsverteilung der Paketgrößen sowie der durchschnittliche Informationsgehalt der Instanzen (Multisets) ermittelt. Anhand der Entropie der Häufigkeitsverteilung kann man abschätzen, wie viele Informationen eine datenschutzfreundliche Übertragungstechnik dem Datenverkehr entzieht. Mit dem mittleren Informationsgehalt lässt sich die Komplexität aller Instanzen in der Grundgesamtheit beurteilen.

Gelingt es, zwischen diesen informationstheoretischen Eigenschaften und den Erkennungsraten einen (kausal erklärbaren) Zusammenhang herzustellen, kann man datenschutzfreundliche Übertragungssysteme im Hinblick auf ihren Schutz vor Website-Fingerprinting in Zukunft möglicherweise direkt anhand ihrer Paketgrößen-Häufigkeitsverteilung beurteilen. Dadurch ließe sich auch die Wirksamkeit verschiedener Gegenmaßnahmen auf effiziente Weise evaluieren – ohne jedes Mal eine aufwändige empirische Untersuchung durchführen zu müssen. In diesem Abschnitt wird folgende Hypothese untersucht:

Paketgrößen-Entropie-Hypothese Es ist zu erwarten, dass ein Zusammenhang zwischen der Entropie der Paketgrößen-Verteilung und der Erkennungsrate besteht. Datenschutzfreundliche Übertragungstechniken, deren Paketgrößenverteilung eine hohe Redundanz aufweist, sollten also besser vor dem Website-Fingerprinting-Angriff schützen als Systeme, bei denen die Entropie der Paketgrößen-Verteilung hoch ist.

7.6.1. Entropie der Paketgrößen-Verteilung

Tabelle 7.6 enthält die Anzahl der unterschiedlichen Paketgrößen n je Datensatz sowie die zugehörige maximale Entropie $H_{\max}(X) = \log_2 n$. Darüber hinaus werden die aus dem Histogramm tatsächlich ermittelte Entropie $H(X)$ sowie die normierte Entropie $H_{\text{norm}} = \frac{H(X)}{H_{\max}(X)}$ angegeben.

Augenscheinlich besteht tatsächlich ein Zusammenhang zwischen den Erkennungsraten und der normierten Entropie $H_{\text{norm}}(X)$ der Paketgrößen-Häufigkeitsverteilung – dies wird durch den Pearson-Korrelationskoeffizient, der sich gemäß Formel (3.2) zu $r_{xy} = 0,92$ berechnet, bestätigt. Geringe Werte von $H_{\text{norm}}(X)$ sind gleichbedeutend mit einer hohen Redundanz $R = 1 - H_{\text{norm}}$. Die Untersuchungsergebnisse deuten darauf hin, dass eine hohe Entropie der Verteilung der Paketgrößen hinreichend und notwendig ist, um hohe Erkennungsraten zu erzielen. Die *Paketgrößen-Entropie-Hypothese* kann damit grundsätzlich bestätigt werden. Einschränkend ist allerdings zu bemerken, dass die betragsmäßigen Unterschiede der normierten Entropie nicht besonders groß sind.

System	n	$H_{\max}(X)$	$H(X)$	$H_{\text{norm}}(X)$	Erkennungsrate
OpenSSH	420	8,71	4,26	0,49	0,9665
OpenVPN	2898	11,50	5,51	0,48	0,9494
CiscoVPN	108	6,75	3,18	0,47	0,9617
Shalon	1907	10,90	4,92	0,45	0,9145
Stunnel	1605	10,65	4,21	0,40	0,9764
JonDonym	205	7,68	2,66	0,35	0,1997
Tor	869	9,76	3,24	0,33	0,0296
Tor+Tinyproxy	1928	10,91	3,20	0,29	0,0264

Tabelle 7.6.: Entropiewerte der Paketgrößen-Häufigkeitsverteilungen der getesteten Systeme sortiert nach der normierten Entropie $H_{\text{norm}} = \frac{H(X)}{H_{\max}(X)}$. Dargestellt sind zudem die Anzahl der unterschiedlichen Paketgrößen n, die maximale Entropie H_{max} und die tatsächliche Entropie H der Paketgrößen-Häufigkeitsverteilungen.

System	n	$H_{\text{norm}}(X)$	$\overline{h}(Y)$	Erkennungsrate
Shalon	1907	0,45	448 Bit	0,9145
OpenVPN	2898	0,48	433 Bit	0,9494
OpenSSH	420	0,49	378 Bit	0,9665
Stunnel	1605	0,40	356 Bit	0,9764
CiscoVPN	108	0,47	276 Bit	0,9617
Tor	869	0,33	202 Bit	0,0296
Tor+Tinyproxy	1928	0,29	195 Bit	0,0264
JonDonym	205	0,35	97 Bit	0,1997

Tabelle 7.7.: Der durchschnittliche Informationsgehalt $\overline{h}(Y) = \frac{1}{k}\sum_{i=1}^{k} h(Y = y_i)$ aller k Instanzen in der Grundgesamtheit der normierten Entropie $H_{\text{norm}}(X)$ bzw. den Erkennungsraten bei den untersuchten Systemen. Die Darstellung ist nach $\overline{h}(Y)$ sortiert.

7.6.2. Durchschnittlicher Informationsgehalt der Instanzen

Einige Autoren verwenden den durchschnittlichen Informationsgehalt zur Abschätzung, wie viele Webseiten ein Klassifizierer maximal voneinander unterscheiden kann (vgl. Abschnitt 3.1.2, 3.2.2 und 3.3.2). Da es hierbei wie in Abschnitt 7.1.2 dargestellt zu einer systematischen Überschätzung kommt und die ermittelten Zahlen ohnehin nur eine geringe Aussagekraft besitzen, wird im Folgenden auf eine Abschätzung der unterscheidbaren Seiten verzichtet. Es wird lediglich untersucht, ob ein Zusammenhang zwischen Informationsgehalt und Erkennungsraten besteht.

Da der MNB-Klassifizierer zur Identifizierung die Instanzen als Häufigkeitsvektoren bzw. Multisets repräsentiert, ist der durchschnittliche Informationsgehalt nach Formel (7.8) zu ermitteln: $\overline{h}(Y) = \frac{1}{k}\sum_{i=1}^{k} h(Y = y_i)$, wobei $h(Y = y)$ nach Formel (7.7) berechnet wird. Tabelle 7.7 fasst die Ergebnisse für die betrachteten Systeme zusammen.

Auch zwischen dem Informationsgehalt und den Erkennungsraten besteht in den Test-

daten ein Zusammenhang: der Pearson-Korrelationskoeffizient beträgt $r_{xy} = 0,83$. Bei komplexeren Instanzen erzielt der Klassifizierer offenbar bessere Ergebnisse.

Auffällig ist vor allem die Tatsache, dass der durchschnittliche Informationsgehalt der Instanzen in den Tor-Datensätzen höher ist als bei JonDonym. Dieses Ergebnis stimmt mit der visuellen Analyse der Histogramme in Abschnitt 7.4.2 überein. Obwohl die Instanzen bei JonDonym weniger Informationen preisgeben, ist die Erkennungsrate bei JonDonym höher als bei Tor. Offenbar ist ein niedriger Informationsgehalt allein kein Garant für niedrige Erkennungsraten. Zur Prognose der Erkennungsrate eines Systems erscheint die normierte Entropie $H_{\text{norm}}(X)$ daher geeigneter.

Das Niveau der ermittelten durchschnittlichen Informationsgehalt-Werte liegt bei den Systemen mit hohen Erkennungsraten deutlich über dem Wert von 130 Bit, den Liberatore et al. beim Jaccard-Klassifizierer ermittelt haben. [LL06]. Dieser Unterschied lässt sich durch die Berücksichtigung der Auftretenshäufigkeiten bei der Berechnung des Informationsgehalts der Multisets erklären.

7.7. Zusammenfassung und Interpretation

In diesem Kapitel wurde gezeigt, dass datenschutzfreundliche Systeme, die keine Traffic-Shaping-Maßnahmen implementieren, nur geringen Schutz vor dem Website-Fingerprinting auf Basis von Paketgrößen-Häufigkeitsverteilungen bieten. Betroffen sind erwartungsgemäß die Single-Hop-Systeme: OpenSSH, OpenVPN, das Cisco-IPsec-VPN sowie Stunnel.

Der MNB-Klassifizierer ist bei diesen Systemen in der Lage, bei 4 Trainingsinstanzen zwischen 94 und 97 % der Instanzen der korrekten URL zuzuordnen, wenn zwischen Training und Test nicht mehr als 12 Stunden verstrichen sind. Da Shalon in der getesteten Version ebenfalls keine Traffic-Shaping-Funktion enthält, erzielt der Klassifizierer auch hier eine hohe Erkennungsrate (91 %). Damit übertrifft der MNB-Klassifizierer die Erkennungsraten der Website-Fingerprinting-Verfahren in [BLJL05; LL06] deutlich.

Bei Tor und JonDonym, die beide Datenstrukturen fixer Größe zum Datentransport verwenden, fallen die Erkennungsraten erheblich niedriger aus (3 % bzw. 20 %). Das niedrige Niveau der Erkennungsraten ist auf die Verwendung von Datenstrukturen fixer Größe zurückzuführen, wobei bei JonDonym trotz größerer Paketgröße (998 Byte im Vergleich zu 512 Byte bei Tor) höhere Erkennungsraten erzielt werden. Die niedrigeren Erkennungsraten von Tor gehen jedoch möglicherweise nur auf seine mangelhafte Zuverlässigkeit zurück. Eine abschließende Bewertung der *Packet-Size-Hypothese* ist in dieser Arbeit nicht möglich.

Die Anzahl der unterschiedlichen Paketgrößen hat offenbar keinen Einfluss auf die Erkennungsrate. Es konnte jedoch gezeigt werden, dass die Homogenität der Instanzen innerhalb einer Klasse im Vergleich zu den Instanzen der übrigen Klassen die Erkennungsraten beeinflusst. Darüber hinaus existiert in den Testdaten eine positive Korrelation zwischen der normierten Entropie der Paketgrößen-Häufigkeitsverteilung (sowie dem durchschnittlichen Informationsgehalt der Instanzen) und den Erkennungsraten. Diese Erkenntnis ist möglicherweise die Basis für eine effizientere Evaluation der Wirksamkeit von Gegenmaßnahmen – ohne dass es jedes Mal einer aufwändigen empirische Ermittlung der Erkennungsraten bedarf.

8

Gegenmaßnahmen

In den bisherigen Kapiteln wurde gezeigt, dass Website-Fingerprinting bei datenschutzfreundlichen Übertragungstechniken, die keine Vorkehrungen gegen Traffic-Analyse treffen, hohe Erkennungsraten erzielt. Dieses Kapitel widmet sich dem Schutz vor Website-Fingerprinting-Angriffen, die auf der Häufigkeitsverteilung von IP-Paketgrößen basieren.

Nach der Beschreibung der Anforderungen werden einige Traffic-Shaping-Techniken für Webseiten vorgestellt. Vor allem Padding, also das Anfügen von Dummy-Daten, erweist sich als viel versprechende Gegenmaßnahme. An die theoretischen Betrachtungen schließt sich die empirische Untersuchung der Wirksamkeit von zwei konkreten Verfahren an: Padding von Application-Records bei TLS-Verbindungen und ein HTTP-Burst-Proxy.

8.1. Anforderungen an Gegenmaßnahmen

Zur Erreichung einer hohen Nutzerakzeptanz sollten Gegenmaßnahmen das folgende Anforderungsprofil erfüllen:

- Die zentrale Anforderung ist die *Verhinderung oder Erschwerung des eigentlichen Website-Fingerprinting-Angriffs*. Dies erfordert die Verschleierung charakteristischer Merkmale des Datenverkehrs, so dass sich aus der Häufigkeitsverteilung der Paketgrößen nicht mehr auf die eigentliche Webseite schließen lässt. Optimale Gegenmaßnahmen reduzieren die Erkennungsraten bei solchen Angriffen auf das Niveau, das beim zufälligen Raten erreicht werden würde.
- Zur Erreichung dieses Ziels sollten Gegenmaßnahmen *keine irreversiblen Veränderungen* an den übertragenen Nutzdaten vornehmen. Ihre Verwendung sollte *für Client und Server transparent* sein und keine Veränderungen an den Gewohnheiten der Betreiber oder Nutzer erforderlich machen.
- Darüber hinaus sollten sie die subjektiv wahrgenommene Performanz aus der Sicht des Benutzers nur minimal beeinträchtigen.

- Wünschenswert ist darüber hinaus, dass eine Gegenmaßnahme *plattformunabhängig* ist, also mit beliebigen Browsern und Betriebssystemen funktioniert.

Diese Anforderungen können zum Beispiel von einem speziellen Proxy-Server mit zugehöriger Client-Software erfüllt werden.

8.2. Gegenmaßnahmen in der Literatur

8.2.1. Vorschläge für Gegenmaßnahmen nach Sun et al.

Sun et al., die Website-Fingerprinting bei TLS-Verbindungen auf der Basis der Größe der übertragenen Objekte vorstellen, untersuchen in ihrer Veröffentlichung auch eine Reihe von Gegenmaßnahmen [SSW+02]. Im Folgenden werden ihre Vorschläge im Hinblick auf die Anwendbarkeit beim Website-Fingerprinting mit IP-Paketgrößen diskutiert.

Webserver: Padding der Objektgrößen Beim Padding der Objektgrößen hängt der Webserver an die Objekte Dummy-Daten an, so dass die Anzahl der beobachtbaren Objektgrößen abnimmt. Der Browser wird so angepasst, dass er das zusäzliche Padding ignoriert. Alternativ dazu kann eine spezielle Client-Software zum Einsatz kommen, die das Padding automatisch entfernt und dem Browser die Originaldaten zur Verfügung stellt.

Sun et al. evaluieren die Wirksamkeit dieser Maßnahme in einer Simulation: Beim Padding der Größen auf Vielfache von 128 Byte sind immer noch mehr als die Hälfte der Seiten eindeutig identifizierbar, beim Aufrunden auf 4 Kilobyte sind es noch 18 % der Seiten. Erst bei 16 Kilobyte sinkt der identifizierbare Anteil unter 5 %. Eine hohe Wirksamkeit hat also auch hohe Kosten (in Form von zusätzlichem Datenverkehr).

Die Veränderungen der Objektgrößen wirkt sich auch auf die Paketgrößen-Häufigkeitsverteilung aus, so dass diese Gegenmaßnahme auch bei Website-Fingerprinting auf der Basis von IP-Paketgrößen durchaus Erfolg versprechend ist. In Abschnitt 8.3 wird eine ähnliche Gegenmaßnahme empirisch untersucht.

Webserver: Padding durch versteckte Objekte Der Betreiber eines Webservers fügt deterministisch oder zufällig versteckte Objekte in die Webseiten ein, die vom Browser zwar heruntergeladen, jedoch nicht angezeigt werden. Auch diese Gegenmaßnahme wirkt sich auf die Häufigkeitsverteilung der IP-Paketgrößen aus, erscheint also grundsätzlich auch zur Abwehr von Website-Fingerprinting auf Basis von Paketgrößen sinnvoll.

Webserver: Nachahmung anderer Seiten Hier soll der Webdesigner versuchen, das Layout einer anderen Seite nachzuahmen, so dass der Datenverkehr möglichst ähnlich ist. Obwohl auch diese Gegenmaßnahme grundsätzlich geeignet erscheint, ist sie wegen des hohen Aufwands kaum praktikabel – ihr Verbreitungspotenzial ist also gering.

Client: Range-Requests Der Client ruft die Objekte der Webseite nicht in einem Stück ab, sondern verwendet den HTTP-Header *Range* (vgl. RFC 2616, section 14.35.2 [FGM+99]), um sie in einzelnen Teilen abzurufen und lokal wieder zusammenzusetzen. Diese Idee lässt sich noch weiterentwickeln: Die abzurufenden Bereiche werden zufällig (möglicherweise sogar überlappend) festgelegt, so dass charakteristische Muster in der Paketgrößen-Häufigkeitsverteilung eliminiert werden. Range-Requests werden allerdings nicht von allen Webservern unterstützt, zudem ist die Verwendung bei dynamisch generierten Seiten problematisch.

Client: Prefetching von verlinkten Seiten Der Client ruft vorausschauend Seiten ab, die in der herunterzuladenden Seite referenziert werden. Heute verfügbare Browser führen Prefetching bereits unter bestimmten Umständen durch (z. B. Firefox). Auch hier entstehen allerdings charakteristische Muster im Datenverkehr, wenn die abzurufenden Seiten deterministisch ermittelt werden.

Web-Server: Automatisches Senden von verlinkten Seiten Der Webserver sendet nicht nur die angeforderte Seite, sondern auch alle verlinkten Seiten. Der Client muss in der Lage sein, den Datenstrom zu dekodieren und die Seiten lokal zwischenspeichern. Auch hier entstehen allerdings charakteristische Muster im Datenverkehr, wenn die abzurufenden Seiten deterministisch ermittelt werden.

Content-Negotiation initiiert von Client oder Webserver Mittels Content-Negotiation können Webserver und Client den abzurufenden Inhalt flexibel aushandeln. Auch hier entstehen allerdings charakteristische Muster im Datenverkehr, wenn die abzurufenden Seiten deterministisch ermittelt werden.

Client: Verwendung eines Ad-Blockers Bei Verwendung eines Ad-Blockers werden nicht alle Objekte, die in einer HTML-Seite eingebettet sind, abgerufen. Der Datenverkehr ändert sich dadurch im Vergleich zum normalen Abruf ohne Ad-Blocker. Es ist zu erwarten, dass ein Klassifizierer den modifizierten Datenverkehr nicht mehr zuverlässig den Webseiten zuordnen könnte. Unterstellt man jedoch wie in Abschnitt 5.1 formuliert, dass der Angreifer den Browser und das verwendete Übertragungsverfahren kennt, kann er selbst den jeweiligen Ad-Blocker verwenden und passende Trainingsinstanzen erzeugen.

Client: Gleichzeitige Verwendung mehrerer Web-Browser Der Benutzer ruft mehrere Seiten in mehreren Browser-Fenstern parallel ab. Dadurch überlappen sich die Verbindungen, und der Datenverkehr der einzelnen Seiten wird verschleiert. Der gleichzeitige Abruf mehrerer Seiten ist auch beim Website-Fingerprinting auf Basis von IP-Paketgrößen eine erfolgversprechende Gegenmaßnahme.

Server: HTTP-Pipelining (zwei Blöcke) Der Client lädt zunächst die HTML-Seite vom Webserver herunter (und erzeugt dadurch einen beobachtbaren Datenblock). Dann sendet er auf einmal die HTTP-Requests für alle eingebetteten Objekte an den Webserver, welcher die einzelnen Objekte dann zusammen in einem zweiten Datenblock übermittelt. In der Simulation können bei Verwendung dieser Gegenmaßnahme nur noch 36 % der Seiten eindeutig identifiziert werden.

Ein ähnlicher Mechanismus ist Bestandteil des HTTP-1.1-Standards (vgl. RFC 2616, section 8.1 [FGM+99]) und wird von vielen Browsern und Servern unterstützt: Der Server schließt die Verbindung nach der Übertragung eines Objektes nicht sofort, sondern wartet auf weitere HTTP-Requests vom Client *(persistent connections* bzw. *request pipelining)*.

Während sich dadurch die tatsächlichen Objektgrößen durchaus verschleiern lassen, ist die Auswirkung auf die Häufigkeitsverteilung der IP-Paketgrößen gering – ob die HTTP-Requests und -Responses über mehrere TCP-Verbindungen ausgetauscht werden oder über eine einzige, macht keinen Unterschied. Bei der Evaluation von Website-Fingerprinting in den Kapiteln 6 und 7 war HTTP-1.1-Pipelining im Browser stets aktiviert. Angesichts der hohen Erkennungsraten beeinträchtigt es den vorgestellten Website-Fingerprinting-Angriff offenbar nicht oder nur in geringem Maße.

Server: HTTP-Pipelining (ein Block) Bei dieser Gegenmaßnahme sendet der Client lediglich einen HTTP-Request für die HTML-Seite an einen vertrauenswürdigen Proxy-Server – dieser analysiert den Inhalt der angeforderten Seite und liefert automatisch die angeforderte Seite und alle eingebetteten Elemente in einem großen Datenblock zurück. In der Simulation sinkt die Erkennungsrate dadurch auf 7 %.

Es ist zu erwarten, dass dieses Verfahren auch die Paketgrößen-Häufigkeitsverteilung erheblich beeinträchtigt, da weniger Pakete in Senderichtung übertragen werden müssen und durch das Senden großer Datenblöcke die Anzahl der charakteristischen Paketgrößen abnimmt. In Abschnitt 8.4 wird die Implementierung eines solchen Burst-Proxies vorgestellt und ihre Wirksamkeit evaluiert.

Bewertung

Viele der von Sun et al. untersuchten Gegenmaßnahmen funktionieren nur, wenn der Angreifer das verwendete Verfahren nicht kennt. Die verwendeten Arbeitshypothesen (vgl. Abschnitt 5.1) und das unterstellte Angreifermodell (vgl. Abschnitt 2.3) machen solche Gegenmaßnahmen, die auf dem Prinzip *security by obscurity* basieren, allerdings obsolet.

Sun et al. gehen offenbar davon aus, dass die Gegenmaßnahmen direkt in der Browser- bzw. Webserver-Software implementiert werden, was sich jedoch nicht mit dem Anforderungsprofil aus Abschnitt 8.1 verträgt. Es ist jedoch ebenso vorstellbar, dass client- bzw. serverseitige Proxies das Traffic-Shaping durchführen, so dass die Anforderungen *Transparenz* und *Plattformunabhängigkeit* gewährleistet werden können.

8.2.2. Simulation von IP-Paketgrößen-Padding

Liberatore et al. [LL06] zeigen, dass Paketgrößen-Padding eine effektive Maßnahme gegen Website-Fingerprinting-Angriffe ist, die auf der Paketgrößen-Häufigkeitsverteilung basieren. Die Autoren simulieren und evaluieren vier statische Padding-Varianten, bei denen die IP-Pakete mit Dummy-Daten auf eine festgelegte Größe aufgefüllt werden.

Lineares Padding Hier wird die Paketgröße auf das nächste Vielfache von 50 Byte erhöht – ein minimaler Eingriff, der die Einzigartigkeit der Pakete theoretisch um den Faktor 50 reduziert. Das zu übertragende Datenvolumen steigt in der Simulation um etwa 3 %.

Padding	Jaccard	Bayes	Datenvolumen
ohne	0,721	0,680	100 %
linear	0,477	0,588	103,4 %
exponentiell	0,056	0,485	108,9 %
Mice-and-Elephants	0,003	0,279	147,8 %
MTU	0,001	0,077	245,3 %

Tabelle 8.1.: Einfluss von verschiedenen Paket-Padding-Verfahren auf die Wirksamkeit von Website-Fingerprinting und übertragenes Datenvolumen. Dargestellt ist die Genauigkeit bei den Verfahren *Jaccard-Koeffizient* und *Naïve-Bayes-Klassifizierer* (Darstellung nach [LL06])

Exponentielles Padding Hier wird die Paketgröße auf die nächste Zweierpotenz (maximal jedoch auf die MTU) erhöht, was die Anzahl der beobachtbaren Paketgrößen auf $\lceil \log_2 \mathrm{MTU} \rceil$, also bei einer MTU von 1500 Byte auf 11, reduziert. In der Simulation nimmt die Paketgröße beim exponentiellen Padding jedoch im Schnitt lediglich um 9 % zu, was damit zu begründen ist, dass die Länge der Mehrzahl der Pakete bereits der MTU entspricht. Diesen Befund bestätigen auch die Histogramme in Abschnitt 7.4.

Mice-and-Elephants-Padding Hier wird die Paketlänge entweder auf 100 oder 1500 Byte gesetzt, je nachdem, ob das ursprüngliche Paket kleiner oder größer als 100 Byte ist. Das übertragene Datenvolumen steigt dadurch in der Simulation um etwa 48 %.

MTU-Padding Alle Pakete werden auf die MTU erweitert, was in der Simulation zu einer Vergrößerung des übertragenen Datenvolumens um 145 % führt.

Bei der Evaluation der Padding-Verfahren gehen die Autoren davon aus, dass der Angreifer in der Lage ist, das eingesetzte Padding-Verfahren zu erkennen und dieses auch bei der Generierung der Profile verwendet. Es stellt sich heraus, dass der Naïve-Bayes-Klassifizierer erheblich besser mit Padding zurechtkommt als der Jaccard-Klassifizierer. Dieses Ergebnis belegt, dass Auftretenshäufigkeiten für die Klassifizierung relevant sind. Tabelle 8.1 zeigt die Erkennungsraten von Jaccard- und NB-Klassifizierer beim Einsatz der untersuchten Padding-Verfahren.

8.2.3. Implementierung von IP-Paketgrößen-Padding

Die Simulationsergebnisse aus dem vorherigen Abschnitt deuten darauf hin, dass das Padding von einzelnen IP-Paketen Website-Fingerprinting auf Basis der Paketgrößen-Häufigkeitsverteilung erheblich erschweren kann. In diesem Abschnitt wird eine praktische Implementierung eines solchen IP-Paketgrößen-Padding-Verfahrens diskutiert.

Kiraly et al. implementieren und evaluieren eine Erweiterung für das IPsec-Protokoll: *Traffic-Flow-Confidentiality (TFC)* [KTB+07]. TFC soll Traffic-Analyse-Angriffe auf Basis von IP-Paketen verhindern. Es wird als eigenständiger Sub-Layer in das IPsec-Protokoll integriert. Abbildung 8.1 skizziert die Einbettung von TFC in ein ESP-Paket.

Die Absicherung der IPsec-Verbindung übernimmt eine dedizierte Kontroll-Logik. Diese kann Dummy-Pakete erzeugen, alle ESP-Pakete auf eine einheitliche Größe auffüllen

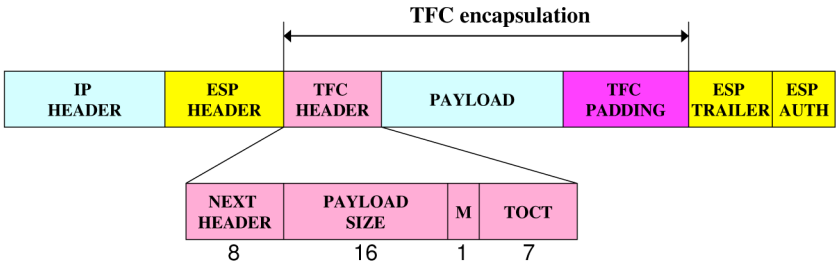


Abbildung 8.1.: Traffic-Flow-Confidentiality für IPsec (Abbildung aus [KTB+07])

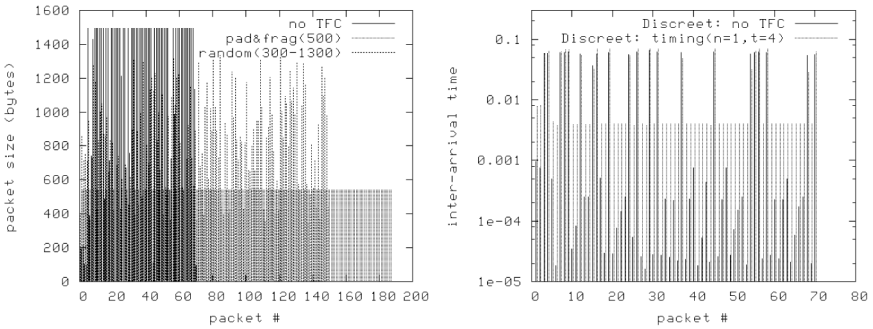


Abbildung 8.2.: TFC führt zu einheitlichen Paketgrößen in einem einheitlichen Zeitraster (Abbildung aus [KTB+07])

sowie die Zeitabstände zwischen den versendeten Paketen vereinheitlichen. Dadurch wird Angriffen, die auf der Häufigkeitsverteilung von IP-Paketgrößen oder den *inter-arrival times* zwischen einzelnen Paketen basieren, die Grundlage entzogen. Die einzige Information, die der Angreifer noch erhält, ist die Gesamtübertragungszeit sowie das übertragene Datenvolumen. Abbildung 8.2 zeigt die Auswirkungen von TFC am Beispiel einer Webseite. Der linke Graph zeigt den Einfluss von Padding (zum einen auf zufällige Werte zwischen 300 und 1300 Byte, zum anderen auf die feste Größe 500 Byte). Im rechten Graph ist die Vereinheitlichung der *inter-arrival times* zu sehen.

Die Evaluierung von Website-Fingerprinting bei Verwendung von TFC steht noch aus. Die Abbildungen lassen jedoch erwarten, dass der Klassifizierer nur sehr niedrige Erkennungsraten erzielen kann.

Kiraly et al. zeigen, dass es grundsätzlich möglich ist, durch extensive Traffic-Shaping-Maßnahmen den Datenverkehr so zu verändern, dass sich daraus keine charakteristischen Muster mehr extrahieren lassen. Das vorgestellte Verfahren hat jedoch eine Einschränkung: Es erfordert einen Eingriff in das Betriebssystem. Die TFC-Erweiterung ist derzeit nur als Patch für die IPsec-Implementierung im Linux Kernel 2.6 verfügbar. Da sie das Kompilieren eines eigenen Kernels erfordert, ist ihr Verbreitungspotential auf einen kleinen Nutzerkreis beschränkt.

Bewertung Die TFC-Implementierung von Kiraly et al. erscheint zwar geeignet, um Website-Fingerprinting-Angriffe zu verhindern, sie verletzt allerdings die Anforderung der *Plattformunabhängigkeit*. Wünschenswert wäre ein Traffic-Shaping-Verfahren, das Paket-Padding in der Sitzungs- oder Anwendungsschicht durchführt.

Eine Implementierung der TFC-Konzepte in höheren Schichten des OSI-Referenzmodells erscheint jedoch problematisch, da die Erzeugung der IP-Pakete in der Sitzungs- und Anwendungsschicht nicht direkt beeinflusst werden kann. Zwar besteht die Möglichkeit IP-Pakete mit *Raw-Sockets* nach eigenen Vorgaben zu konstruieren, bei vielen Betriebssystemen sind jedoch lediglich privilegierte Benutzer (z. B. *root*) in der Lage, solche Sockets zu öffnen. Dadurch wird das Verbreitungspotential eingeschränkt.

In den folgenden Abschnitten wird daher untersucht, ob auch Traffic-Shaping-Maßnahmen, die in der Sitzungs- bzw. Anwendungsebene greifen, vor Website-Fingerprinting-Angriffen schützen.

8.3. Padding bei TLS

In diesem Abschnitt wird eine Traffic-Shaping-Maßnahme vorgestellt, die in der Sitzungsschicht arbeitet. Es soll ermittelt werden, ob sich zusätzliches Padding von TLS-Records zur Abwehr von Website-Fingerprinting-Angriffen eignet. Diese Padding-Methode ähnelt damit dem Padding der Objektgrößen bei Sun et al., auf das in Abschnitt 8.2.1 eingegangen wurde.

Zunächst wird der Padding-Mechanismus von TLS erläutert, im Anschluss daran wird eine Möglichkeit vorgestellt, TLS-Records mit zusätzlichem Padding zu versehen. Schließlich wird der Datenverkehr, der beim Abruf von Webseiten über eine entsprechend angepasste Version von OpenSSL entsteht, mit dem bekannten Website-Fingerprinting-Verfahren evaluiert.

8.3.1. Minimales Padding auf die Blockgröße

Viele datenschutzfreundliche Übertragungstechniken greifen zur Verschlüsselung auf SSL bzw. TLS zurück. TLS 1.0 bzw. der Nachfolger 1.1 spezifizieren eine minimale Blockgröße von 8 Byte (vgl. RFC 2246 und RFC 4346 [DA99; DR06]). TLS implementiert Padding lediglich für Blockchiffren, bei den ebenfalls unterstützten Stromchiffren RC4 (40 und 128 Bit) kommt kein Padding zum Einsatz. Aktuelle Versionen unterstützen die Blockchiffren 3DES und AES (vgl. RFC 3268 [Cho02]).

Die TLS-Implementierung in der OpenSSL-Bibliothek, die von vielen Linux-Programmen verwendet wird, etwa bei wget sowie dem Apache-Webserver, der gemäß der Netcraft Secure Server Survey im Juni 2006 den Markt mit einem Marktanteil von ca. 45 % dominierte [Net06], handelt bevorzugt die Blockchiffre AES aus (z. B. mit der CipherSuite: TLS_DHE_RSA_WITH_AES_256_CBC_SHA). Da die Blockgröße von AES 128 Bit (16 Byte) beträgt [Nat01], werden die TLS-Records also üblicherweise mit Dummy-Daten auf das jeweils nächste Vielfache von 16 Byte aufgefüllt.

Dies lässt sich bei HTTPS-Übertragungen demonstrieren, indem eine spezielle dynamische Seite abgerufen wird, deren Größe in 1-Byte-Schritten verändert werden kann. Abbildung 8.3 zeigt die empfangenen Datenmengen für verschiedene Größen dieser Seite: Das tatsächlich übertragene Datenvolumen steigt in regelmäßigen Abständen (alle 16 Byte) sprunghaft an.

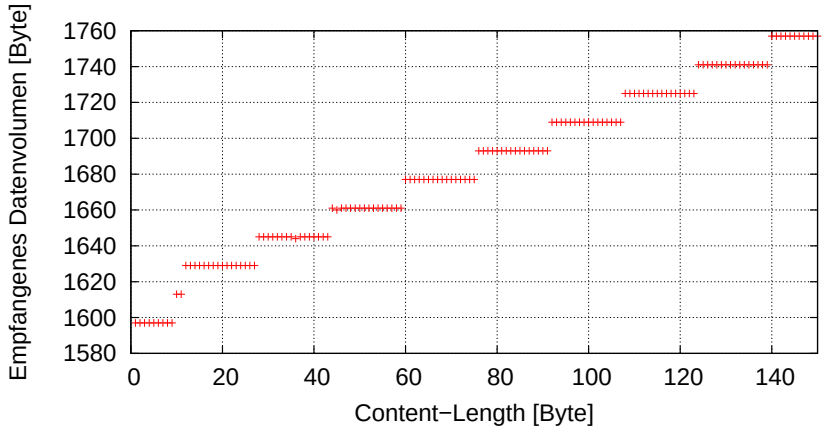


Abbildung 8.3.: Übertragung einer Webseite mit einer Größe zwischen 0 und 150 Byte über TLS mit wget. Bei der Verwendung der AES-Blockchiffre wird beim TLS-Protokoll eine Blockgröße von 16 Byte verwendet.

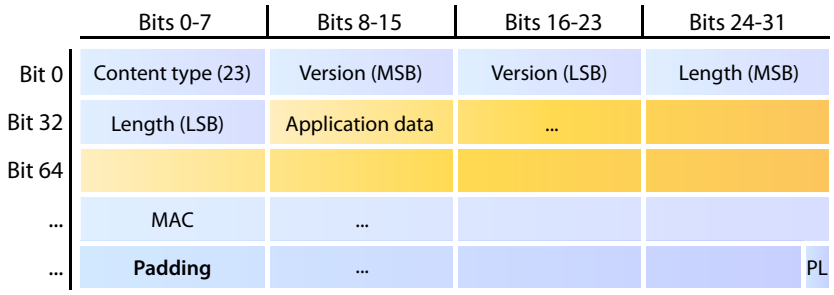


Abbildung 8.4.: Schematischer Aufbau eines TLS-Application-Records. Im letzten Byte wird die Padding Length (PL) gespeichert.

8.3.2. Zusätzliches Padding bei TLS

Padding auf die Blockgröße wird bei TLS und SSL gleichermaßen durchgeführt. Erkennungsraten von über 90 % bei Stunnel zeigen jedoch, dass es gegen Website-Fingerprinting so gut wie wirkungslos ist. TLS bietet allerdings zusätzlich die Möglichkeit, in TLS-Application-Records bis zu 255 Byte zusätzliches Padding zu verwenden:

> „Padding that is added to force the length of the plaintext to be an integral multiple of the block cipher's block length. The padding MAY be any length up to 255 bytes, as long as it results in the TLSCiphertext.length being an integral multiple of the block length. Lengths longer than necessary might be desirable to frustrate attacks on a protocol that are based on analysis of the lengths of exchanged messages. Each uint8 in the padding data vector MUST be filled with the padding length value." (RFC 4346, section 6.2.3.2. [DR06])

Abbildung 8.4 zeigt den schematischen Aufbau eines TLS-Application-Records. Die aktuell[1] verfügbaren Implementierungen der OpenSSL-Bibliothek machen davon jedoch kei-

[1]Zum Zeitpunkt der Einreichung der Arbeit ist die OpenSSL-Version 0.9.8g (http://www.openssl.org/source/openssl-0.9.8g.tar.gz) aktuell.

nen Gebrauch. Sie beschränken sich auf das minimal erforderliche Blockgrößen-Padding. Eine Quellcode-Analyse von GnuTLS[2], dem TLS-Code in der Standard-Implementierung der Java Cryptography Architecture (JCA)[3] sowie von BouncyCastle[4] ergibt, dass keine dieser Implementierungen das in RFC 4346 vorgesehene zusätzliche Padding unterstützt.

Listing 8.1 zeigt den relevanten Quellcode-Ausschnitt der OpenSSL-Bibliothek (Version 0.9.8g). Der Kommentar /* *Add weird padding of upto 256 bytes* */ legt nahe, dass die OpenSSL-Entwickler bereits darüber nachgedacht haben, das zusätzliche Padding zu implementieren. Es stellt sich die Frage, wieso diese Möglichkeit von den Implementierungen nicht umgesetzt wird. Eine Nachfrage auf der Mailingliste *openssl-dev@openssl.org* wurde mit Verweis auf *Symbian*-basierte Mobiltelefone beantwortet, die das zusätzliche Padding angeblich nicht verarbeiten können.[5]

Listing 8.1: Quellcode-Ausschnitt aus `ssl/t1_enc.c`, Zeilen 524 ff.

```
int tls1_enc(SSL *s, int send)
  {
  [...]
    l=rec->length;
    bs=EVP_CIPHER_block_size(ds->cipher);

    if ((bs != 1) && send)
      {
      i=bs-((int)l%bs);

      /* Add weird padding of upto 256 bytes */

      /* we need to add 'i' padding bytes of value j */
      j=i-1;
      [...]
        for (k=(int)l; k<(int)(l+i); k++)
          rec->input[k]=j;
        l+=i;
        rec->length+=i;
        }
  [...]
  }
```

8.3.3. Implementierung und Evaluation zusätzlichen TLS-Paddings

Eine Implementierung von Traffic-Shaping-Maßnahmen in der OpenSSL-Bibliothek hätte einen entscheidenden Vorteil: Alle Programme, die darauf zurückgreifen, profitieren davon, ohne dass weitere Anpassungen am Quellcode nötig sind.[6]

Betrachtet man jedoch die Funktionsweise des TLS-Record-Padding genauer, stellt man fest, dass sein Einfluss sehr gering ist. Dies lässt sich am Beispiel eines 50 Kilobyte großen Objekts erläutern. Aus Gründen der Übersichtlichkeit wird die Größe der TCP/IP-Header

[2] Homepage: http://www.gnu.org/software/gnutls/
[3] siehe [Sun07a] und [Sun07b]
[4] Homepage: http://www.bouncycastle.org/
[5] Der Thread ist im Archiv der Mailingliste einsehbar (http://www.mail-archive.com/openssl-dev@openssl.org/msg23671.html).
[6] Der Objektcode der Programme muss jedoch gegen die modifizierte OpenSSL-Bibliothek gelinkt werden.

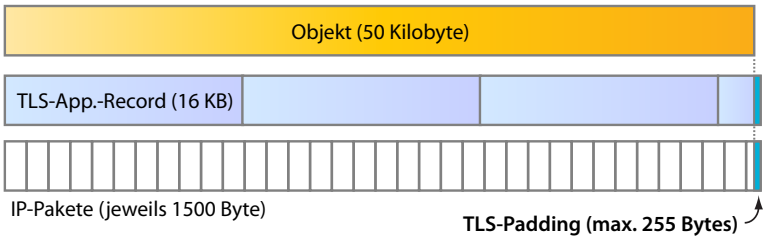


Abbildung 8.5.: Auswirkung von TLS-Padding auf IP-Paketgrößen: Nur das letzte IP-Paket ist betroffen.

bei der Veranschaulichung vernachlässigt. Ein TLS-Application-Record fasst maximal 16 Kilobyte an Nutzdaten. Das 50 Kilobyte große Objekt wird also in vier Application-Records aufgeteilt, drei davon fassen jeweils 16 Kilobyte, der letzte enthält die verbleibenden 2 Kilobyte. Bei einer MTU von 1500 Byte wird ein vollständig ausgefüllter TLS-Records im Idealfall in 10 IP-Pakete mit der Maximalgröße von 1500 Byte aufgeteilt. Es verbleibt ein Rest von 2 Kilobyte. Dieser wird jedoch nicht separat übertragen, sondern zusammen mit dem nächsten Application-Record. Das eingesetzte TLS-Record-Padding wirkt sich demnach lediglich auf die Größe des allerletzten Pakets aus – und dort führt es zu völlig willkürlichen Paketgrößen. Abbildung 8.5 veranschaulicht diesen Fragmentierungsprozess.

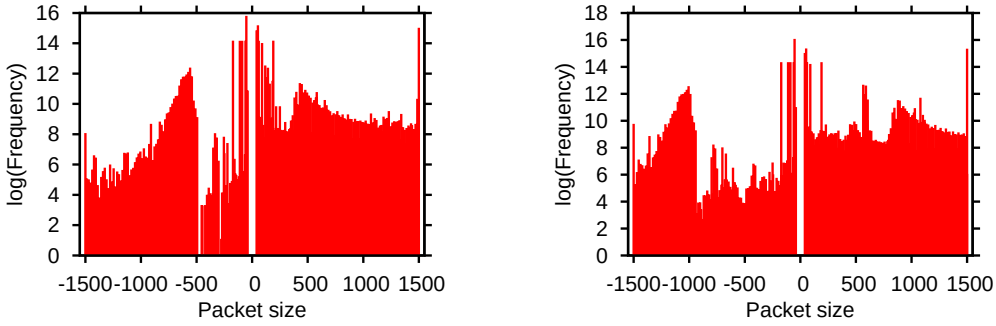
Die schematische Betrachtung lässt vermuten, dass TLS-Padding nur eine geringe Wirksamkeit hat. Im Folgenden soll untersucht werden, wie sich Padding von TLS-Records auf die Effektivität von Website-Fingerprinting auswirkt. Hierzu werden zwei Hypothesen aufgestellt:

TLS-Padding-Histogramm-Hypothese Zusätzliches TLS-Padding ist nicht geeignet, um die Anzahl der beobachtbaren Paketgrößen sowie die Entropie der Paketgrößen-Häufigkeitsverteilung zu verringern.

TLS-Padding-Erkennungsraten-Hypothese Zusätzliches TLS-Padding führt nicht zu niedrigeren Erkennungsraten bei einem Website-Fingerprinting-Angriff, der auf der Paketgrößen-Häufigkeitsverteilung basiert.

Zur Überprüfung dieser Hypothese wurden die Test-Webseiten noch einmal über Stunnel abgerufen, wobei Stunnel gegen eine angepasste OpenSSL-Bibliothek gelinkt wurde. Die OpenSSL-Bibliothek wurde so modifiziert, dass alle TLS-Application-Records mit der größtmöglichen Anzahl von Padding-Bytes erweitert werden. Dabei wird – so wie es der Standard vorschreibt – darauf geachtet, dass die resultierende Record-Länge wieder ein Vielfaches der Blockgröße der Chiffre ist.

Abbildung 8.6 stellt die Histogramme von Stunnel mit und ohne TLS-Padding einander gegenüber. Im Histogramm ist beim Einsatz von zusätzlichem TLS-Padding ein Trend zu größeren Paketen erkennbar. Dieser Trend bestätigt sich in der nummerischen Auswertung. Demnach ist die durchschnittliche Paketgröße in der Tat um 57 Byte gestiegen (vgl. Tabelle 8.2). Das Padding hat also durchaus einen messbaren Effekt auf die Häufigkeitsverteilung.



(a) Stunnel normal (b) Stunnel (max. padding)

Abbildung 8.6.: Einfluss des zusätzlichen TLS-Paddings auf die logarithmierte Paketgrößen-Häufigkeitsverteilung ist gering

Parameter	Stunnel normal	mit max. TLS-Padding
Mittlere Paketgröße (Byte)	288	345
Anzahl beobachteter Paketgrößen n	1605	1636
Normierte Entropie $H_{\text{norm}}(X)$	0,40	0,39
Erkennungsrate (TF, Normalisierung)	97,64 %	97,56 %

Tabelle 8.2.: Vergleich der Datensätze beim Abruf mit unmodifiziertem Stunnel sowie bei Verwendung maximal möglichen TLS-Paddings

Allerdings ist die Anzahl der beobachtbaren Paketgrößen nicht gesunken, sondern sogar leicht angestiegen, und die Entropie hat sich im Vergleich zur Stunnel-Messung ohne zusätzliches TLS-Padding praktisch nicht verändert. Dieses Ergebnis bestätigt die eingangs aufgestellte *TLS-Padding-Histogramm-Hypothese*. Die Tabelle zeigt außerdem, dass sich die Erkennungsrate beim Einsatz von zusätzlichem TLS-Padding nicht verändert hat. Auch die *TLS-Padding-Erkennungsraten-Hypothese* wird also bestätigt. Das zusätzliche TLS-Padding bleibt damit wirkungslos und schützt nicht vor dem Website-Fingerprinting-Angriff mit dem MNB-Klassifizierer.

8.4. Implementierung des Burst-Proxies

Die Veröffentlichungen, die in Abschnitt 8.2 vorgestellt wurden, zeigen, dass durch extensiven Einsatz von Paket-Padding in der Netzwerkschicht ein Website-Fingerprinting-Angriff erschwert oder sogar verhindert werden kann. Padding auf Sitzungsebene reicht dazu nicht aus, wie die Untersuchung von zusätzlichem Padding von TLS-Records in Abschnitt 8.3.3 gezeigt hat.

Es stellt sich die Frage, ob der vorgestellte Website-Fingerprinting-Angriff tatsächlich nur durch Padding von IP-Paketen verhindert werden kann, oder ob Maßnahmen in höheren Schichten dazu ebenfalls geeignet sind. Der in Abschnitt 8.2 bereits angesprochene Burst-Proxy ist eine solche Gegenmaßnahme, die in der Anwendungsschicht arbeitet. Im Folgenden werden Entwicklung und Evaluation des Prototyps eines Burst-Proxies beschrieben.

Ein Burst-Proxy, manchmal auch als *Prefetching Proxy* bezeichnet, verhält sich aus Sicht des Browsers wie ein normaler HTTP-Proxy. Im Gegensatz zu einem normalen Proxy analysiert der Burst-Proxy jedoch die HTML-Seiten, die über ihn abgerufen werden. Er extrahiert die Links aller eingebetteten Objekte, lädt diese vom Webserver herunter und schickt sie zusammen mit der HTML-Seite an die Client zurück. Da heutige Browser mit einem solchen Datenpaket nichts anfangen können, ist eine Client-Software erforderlich, welche die empfangenen Daten dekodiert und dem Browser zum richtigen Zeitpunkt zur Verfügung stellt. Dadurch lässt sich die Anzahl der HTTP-Requests, die der Browser über das Netzwerk an den Webserver senden muss, erheblich reduzieren.

8.4.1. Burst-Proxy bei Crowds

Die Idee des Burst-Proxies ist nicht neu – ein Burst-Proxy ist auch im Konzept des Crowds-Anonymisierungsdienstes vorgesehen [RR98]. Dort erfüllt er jedoch einen anderen Zweck. Bei Crowds leitet die Client-Software, "jondo", einen HTTP-Request nach dem Zufallprinzip entweder an einen anderen jondo weiter oder sie sendet ihn direkt an den Webserver. Erhält ein jondo j_2 einen Request von einem anderen j_1, weiß j_2 daher nicht, ob j_1 der Urheber des Requests ist oder die Nachricht seinerseits von einem anderen jondo empfangen hat.

Die Entwickler von Crowds identifizieren jedoch ein Problem: Da der Browser nach dem Parsen der HTML-Seite typischerweie sofort HTTP-Requests für alle enthaltenen Objekte versendet, kann j_2 durch Messen der Zeit zwischen der Auslieferung der HTML-Seite und dem Eintreffen der Folge-Requests abschätzen, ob j_1 der Urheber der Requests ist. Hierzu benötigt j_2 lediglich eine Abschätzung für die Paketlaufzeit.

Die Entwickler schlagen daher vor, dass der letzte jondo in einer Kette alle HTML-Seiten analysiert und die eingebetteten Elemente selbstständig herunterlädt und zurückschickt. Der jondo, der im Browser des Clients als Proxy-Server eingetragen ist, muss dann den Datenstrom dekodieren und dem Browser die HTTP-Responses zur richtigen Zeit zur Verfügung stellen.

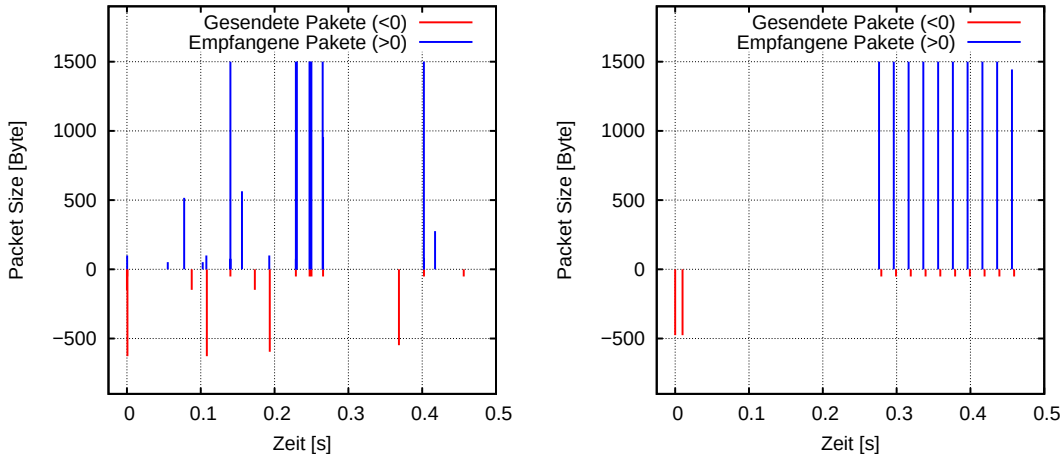
8.4.2. Motivation

Natürlich weist der Datenverkehr auch bei der Verwendung eines Burst-Proxies charakteristische Merkmale auf. In den vorherigen Kapiteln wurde jedoch gezeigt, dass Website-Fingerprinting auf eine hohe Entropie der Pakethäufigkeitsverteilung angewiesen ist. Der Burst-Proxy soll die Entropie der Häufigkeitsverteilung verringern, indem er die in einer HTML-Seite eingebetteten Elemente selbstständig herunterlädt und gebündelt an den Browser zurückschickt, wodurch *Paket-Bursts* entstehen.

Abbildung 8.7 skizziert die erhoffte Veränderung des Datenverkehrs durch die Verwendung eines Burst-Proxies. Zum einen sollte die Anzahl der gesendeten Pakete erheblich sinken, zum anderen ist zu erwarten, dass die Anzahl der unterschiedlichen Paketgrößen sinkt.

8.4.3. Systemarchitektur

Abbildung 8.8 stellt den Aufbau des Burst-Proxies dar. Der Burst-Proxy besteht aus zwei Komponenten, dem *Local End* und dem *Remote End*. Das Local End läuft auf dem Rechner



(a) Abruf über OpenSSH (b) Abruf über Burst-Proxy

Abbildung 8.7.: Es ist zu erwarten, dass der Informationsgehalt des Datenverkehrs von *www.google.com* durch den Burst-Proxy erheblich reduziert wird (idealisierte Darstellung).

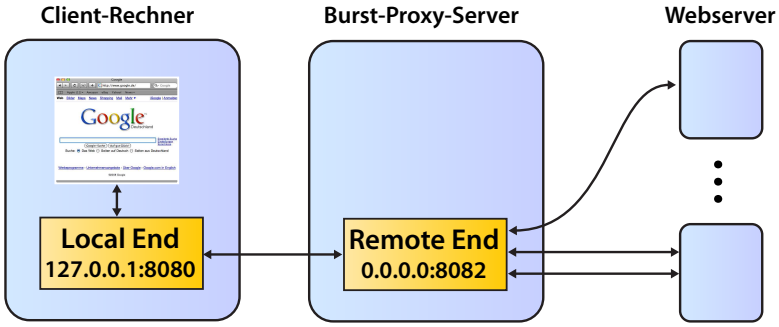


Abbildung 8.8.: Die zwei Komponenten des Burst-Proxies: Local und Remote End

des Benutzers, das Remote End auf einem dedizierten Server im Internet. Die Kommunikation erfolgt über TCP-Verbindungen.

Beide Komponenten öffnen einen TCP-Port für eingehende Verbindungen, wobei das Local End diesen auf der Loopback-Adresse öffnet, so dass es nicht von anderen Netzwerk-Teilnehmern verwendet werden kann. Das Remote End nimmt hingegen Verbindungen auf seiner öffentlichen Netzwerkschnittstelle entgegen.

Der Web-Browser wird so konfiguriert, dass er das Local End als HTTP-Proxy-Server verwendet. Im Local End ist die IP-Adresse des zu verwendenden Remote Ends hinterlegt. Das Remote End kommuniziert mit den einzelnen Webservern. Zunächst soll ein Überblick über die grundsätzliche Funktionsweise der beiden Komponenten gegeben werden, bevor im nächsten Abschnitt die Design-Alternativen und -Entscheidungen diskutiert werden.

Funktionsweise des Remote End

Geht beim Remote End ein HTTP-Request ein, wird der Request an den zugehörigen Webserver weitergeleitet. Handelt es sich bei der HTTP-Response um eine HTML-Seite,

ermittelt das Remote End die URLs aller Objekte, die darin eingebettet sind, um diese selbstständig von den jeweiligen Webservern herunterzuladen. Das Remote End übermittelt dann die HTML-Seite und alle eingebetteten Objekte an das Local End. Handelt es sich bei der HTTP-Response nicht um eine HTML-Seite, verhält sich das Remote End wie ein normaler HTTP-Proxy-Server und übermittelt die Daten an das Local End.

Funktionsweise des Local End

Eingehende HTTP-Requests leitet das Local End an das Remote End weiter. Dann wartet es, bis das Remote End eine HTTP-Response zurückschickt. Die Verbindung zum Browser wird während dieser Wartezeit offen gehalten.

Trifft die HTTP-Response ein, sendet das Local End diese an den Browser. Nachdem der Browser die Seite analysiert hat, wird er weitere HTTP-Requests an das Local End senden, um die eingebetteten Objekte abzurufen. Das Local End überprüft, ob die angefragten URLs in der Menge der URLs enthalten sind, die das Remote End übermittelt hat. Falls der Browser eine URL verlangt, die das Remote End nicht selbstständig abgerufen hat (weil diese etwa dynamisch durch JavaScript generiert wird), wird ein weiterer HTTP-Request an das Remote End übermittelt.

8.4.4. Design-Entscheidungen

URLs nur im Remote End parsen

Es gibt bei einem Burst-Proxy zwei Möglichkeiten für das Protokoll zwischen Local und Remote End: Entweder analysiert nur das Remote End die HTML-Seiten und teilt dem Local End mit, welche URLs es gefunden hat und senden wird, oder das Remote End *und* das Local End analysieren die Seite unabhängig voneinander. Der letztere Fall erscheint problematisch, da Inkonsistenzen auftreten, wenn die beiden Komponenten bei der Analyse zu einem unterschiedlichen Ergebnis kommen. Das Local End wartet dann unter Umständen auf URLs, die das Remote End gar nicht schickt. Da es bei einem verteilten System nicht immer möglich ist, Clients und Server auf den exakt gleichen Software-Versionsständen zu halten, müsste bei der Weiterentwicklung stets auf die Abwärts- und Aufwärtskompatibilität geachtet werden.

Beim Burst-Proxy-Prototyp wurde daher die erstgenannte Alternative implementiert: Hier analysiert ausschließlich das Remote End den Seitenquelltext und teilt dem Local End in einer dedizierten Protokollnachricht *(urllist)* die Liste der gefundenen URLs mit.

Bündel-Strategien

1-Bündel-Strategie Im Idealfall sendet ein Burst-Proxy die HTML-Datei zusammen mit allen eingebetteten Objekten *in einem einzelnen Bündel* an das Local End, so dass die Anzahl der vollständig gefüllten IP-Pakete maximiert wird und die Daten in einem möglichst homogenen Datenstrom zum Local End fließen. Diese Methode macht den Burst-Proxy zu einer sehr wirksamen Gegenmaßnahme. Nachdem der Browser die Anfrage an das Local End geschickt hat, muss er jedoch warten, bis das Remote End mit dem Download aller eingebetteten Objekte fertig ist und mit dem Transfer der HTML-Seite und der Objekte an das Local End beginnt. Bei größeren Seiten kommt es dadurch zu störenden Wartezeiten.

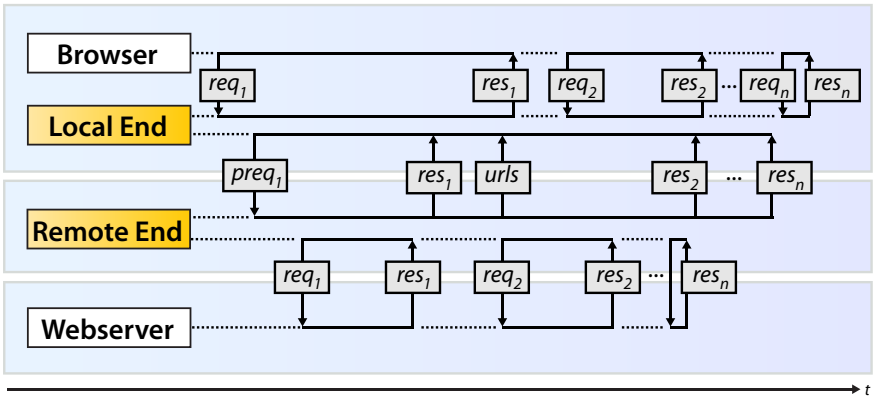


Abbildung 8.9.: Ausgetauschte Nachrichten beim Abruf einer Webseite mit $n-1$ eingebetteten Elementen über den Burst-Proxy

2-Bündel-Strategie Bei dieser Alternative sendet das Remote End *insgesamt zwei Bündel*: Nachdem es die HTML-Seite analysiert hat, übermittelt es diese zusammen mit der *urllist* an das Local End. Parallel dazu führt das Remote End den Download der eingebetteten Objekte durch. Das Local End leitet die HTML-Seite umgehend an den Browser weiter, so dass dieser bereits den Text der Seite darstellen und mit der Analyse der Seite beginnen kann.[7] Die eingebetteten Objekte sendet das Remote End dann in einem zweiten Datenblock zum Local End. Erst dann kann der Browser die Bilder darstellen. Auch hier kommt es – bei Seiten mit vielen Bildern – zu erhebllichen Wartezeiten.

n-Bündel-Strategie Die dritte Alternative ist eine Erweiterung des soeben beschriebenen Verfahrens mit zwei Bündeln, wobei die Anzahl der Bündel höher ist. Bei der n-Bündel-Strategie sammelt das Remote End anhand einer festgelegten Entscheidungsregel einen Teil der eingebetteten Objekte, um diese bereits vorzeitig an das Local End zu schicken. Der Nachteil dieser Strategie: Der Angreifer kann mehrere Datenblöcke auf der Leitung beobachten und daraus auf den Aufbau der Webseite schließen. Der Vorteil: Die Wartezeiten werden deutlich reduziert, da der Browser bereits frühzeitig mit dem Rendering der Seite beginnen kann.

Im Prototyp ist die n-Bündel-Strategie implementiert. Das Remote End überträgt immer dann ein Bündel von Objekten an das Local End, sobald die Größe der bereits empfangenen Objekte 50 Kilobyte übersteigt. Der Speicherbedarf dieser Strategien wird in Abschnitt 8.4.6 diskutiert.

8.4.5. Abläufe beim Abruf einer Webseite

Abbildung 8.9 zeigt die Schritte, die beim Abruf einer typischen Internet-Seite durchlaufen werden.

In der Darstellung sendet der Browser zunächst einen HTTP-Request req_1 zum Abruf einer HTML-Seite an das Local End. Das Local End erzeugt durch Hinzufügen eines

[7] Diese Aussage trifft auf die aktuellen Browser nicht immer zu: Firefox beginnt mit dem Rendering einer Seite unter Umständen erst dann, wenn er die Größe aller Bilder kennt, die in ihrem IMG-Tag kein WIDTH- und HEIGHT-Attribut tragen.

X-Accept-Prefetching-Headers daraus einen prefetching request $preq_1$ und sendet ihn an das Remote End des Burst-Proxies. Das Remote End liest den Prefetching-Header aus, rekonstruiert req_1 und sendet den Request an den Webserver. Dieser antwortet mit der HTTP-Response res_1. Das Remote End dekodiert eventuell vorhandene Transfer- (z. B. *chunked*) und Content-Encodings (z. B. *gzip*), bis der Quelltext der HTML-Seite vorliegt.

Daraufhin ermittelt das Remote End die URLs aller eingebetteten Objekte (z. B. Bilder, JavaScript- und CSS-Dateien) und beginnt damit, diese vom Webserver herunterzuladen. Hierzu erzeugt es auf der Basis von req_1 eine Reihe von HTTP-Requests, wobei jeweils URL und Referrer-Header angepasst werden. Parallel dazu sendet es res_1 an das Local End zurück, um die Wartezeit für den Browser zu minimieren (vgl. Abschnitt 8.4.4).

Zusammen mit der HTML-Seite wird auch die *urllist*, die Liste der URLs der eingebetteten Objekte, an das Local End zurückgeschickt (*urls*). Nachdem das Local End res_1 und *urls* empfangen hat, übermittelt es die HTTP-Response res_1 an den Browser. Der Browser kann die Seite dann parsen und eventuell bereits den Text darstellen. Kurz darauf wird der Browser HTTP-Requests für die in der Seite eingebetteten Objekte an das Local End senden.

Da der Parser des Browsers unter Umständen mehr URLs gefunden hat als die Remote End, muss das Local End bei jedem HTTP-Request entscheiden, wie damit zu erfahren ist. Handelt es sich um eine URL, die in der *urllist* enthalten ist, wird das Remote End die zugehörige HTTP-Response „irgendwann“ an das Local End schicken. Das Local End hält die Verbindung zum Browser dann so lange aufrecht, bis die Response vom Remote End empfangen wurde. Diese wird dann an den wartenden Browser weitergeleitet.

Fordert der Browser jedoch eine URL an, die nicht in *urllist* enthalten ist, wird das Remote End diese Seite von sich aus auch nicht übermitteln. Das Local End baut daher eine weitere Verbindung zum Remote End auf, um diese Anfrage zu bearbeiten.

Rekursives Prefetching

CSS-Dateien, die in HTML-Dateien referenziert werden (mit *<link rel="stylesheet" type="text/css" href="/path/to/file.css">*) erfordern eine Sonderbehandlung: In solchen CSS-Dateien können ebenfalls einzubettende Bilder enthalten sein (etwa mit dem Attribut *background: url("/path/to/picture.png")*. Darüber hinaus gibt es die Möglichkeit, in einer CSS-Datei eine andere CSS-Datei zu inkludieren (mit der Anweisung *@import url("erweitert.css")*). Liegt der Quelltext einer CSS-Datei vor, ermittelt das Remote End die darin enthaltenen URLs und fügt diese in die Warteschlange des Thread-Pools ein, der sich um den Download der eingebetteten Elemente kümmert (vgl. Abschnitt 8.4.7).

Dieses rekursive Prefetching birgt die Gefahr, dass es durch Zirkelbezüge zu einer Endlosschleife kommt. Das Remote End adressiert dieses Problem, indem es für jeden Request die URLs der eingebetteten Objekte, die es bereits gefunden hat, in einer Lookup-Tabelle *(HashMap)* speichert.

Burst-Proxy-Protokoll

Abbildung 8.10 skizziert die Nachrichten, die zwischen den Beteiligten ausgetauscht werden. Das Protokoll basiert auf HTTP 1.1 und verwendet das Transfer-Encoding *chunked* zur Übertragung der einzelnen eingebetteten Objekte. In so genannten *chunk-extensions* übermittelt das Remote End an das Local End die Größe der jeweiligen HTTP-Header sowie die Gesamtgröße eines Objekts (in Hexdarstellung vor dem Strichpunkt). Der Typ „response“ kündigt die eigentliche HTTP-Response für den ursprünglichen HTTP-Request

an, der Typ „urllist“ enthält die Liste der gefundenen eingebetteten Objekte und der Typ „prefetched“ wird verwendet, um auf die Übertragung eines eingebetteten Objekts mit dem angegebenen URL-Index hinzuweisen. Nicht abgebildet ist die Verwendung der Chunk-Extensions *HE* und *BE*, die das Transfer-Encoding von Header und Body festlegen: Das Remote End kann so konfiguriert werden, dass es Header und Body mit *gzip* komprimiert, um das übertragene Datenvolumen zu reduzieren.

8.4.6. Speicherbedarf im Local und Remote End

Beim in Abbildung 8.9 dargestellten Ablauf wird unterstellt, dass der Browser die Requests $req_{2...n}$ an das Local End sendet, *bevor* die zugehörigen Responses $res_{2...n}$ vom Remote End an das Local End übermittelt wurden. In diesem Fall könnte das Local End die Daten „live“ vom Remote End an den Browser weiterleiten.

In der Realität ist diese Annahme jedoch nicht erfüllt – es kann passieren, dass der Browser langsamer ist als das Remote End. Das Local End könnte in diesem Fall die Kommunikation mit dem Remote End so lange anhalten, bis der Browser einen HTTP-Request für das nächste in der Warteschlange befindliche Element stellt. Da die Reihenfolge, in welcher der Browser die HTTP-Requests sendet, jedoch nicht vorhersehbar ist, könnte es zu einem Deadlock kommen.

Beim Prototyp des Burst-Proxies wird dieses Problem daher wie folgt gelöst: Das Local End empfängt unabhängig vom Browser alle HTTP-Responses vom Remote End und legt sie in einer HashMap ab. Stellt der Browser den dazugehörigen HTTP-Request, wird die HTTP-Response aus der HashMap geholt und zurückgeschickt. Eine präzise Ermittlung eines geeigneten Löschzeitpunktes ist allerdings problematisch: Das Local End kann nicht beurteilen, *ob* und *wann* der Browser den HTTP-Request für ein Objekt, das in der HashMap aufbewahrt wird, stellen wird. Es ist auch nicht möglich, die Objekte in der HashMap, für die der Browser bereits einen HTTP-Request gestellt hat, sofort nach der Übertragung aus der HashMap zu löschen, da es nicht ausgeschlossen ist, dass der Browser beim Laden der Seite eine URL mehrmals anfordert.[8] Daher wird im Prototyp folgende Heuristik verwendet: Ist der Download aller eingebetteten Objekte abgeschlossen, werden die HTTP-Responses nach einer (einstellbaren) Verzögerung von 10 Sekunden wieder aus der HashMap entfernt. Der Speicherbedarf auf dem Local End ist jedoch vernachlässigbar: Bei den Versuchen befanden sich in der HashMap nur einige hundert Objekte, der Speicherbedarf lag stets unter 10 Megabyte.

In Abhängigkeit von der gewählten Bündel-Strategie muss auch das Remote End die HTTP-Responses im Speicher sammeln. Bei der Verwendung der 1- und 2-Bündel-Strategie muss das Remote End sämtliche eingebetteten Objekte zwischenspeichern, ehe es diese an das Local End senden kann. Bei der n-Bündel-Strategie ist der Speicherbedarf geringer, wenn die einzelnen Objekte klein sind. Das Remote End kann nach jedem Versand eines Bündels die darin enthaltenen Objekte aus dem Speicher entfernen.

Behandlung großer Objekte Große Objekte stellen jedoch – zumindest beim Prototyp – ein grundsätzliches Problem dar: Local und Remote End leiten ein 3-Megabyte-Bild, das in einer HTML-Seite eingebettet ist, erst dann weiter, wenn sie die Daten von der

[8] Firefox hat bei Tests mit dem Burst-Proxy bei einigen Webseiten tatsächlich ein und dasselbe Bild mehrmals vom Local End angefordert, wenn das Bild mehrmals in der Webseite vorkam (da der Cache deaktiviert war).

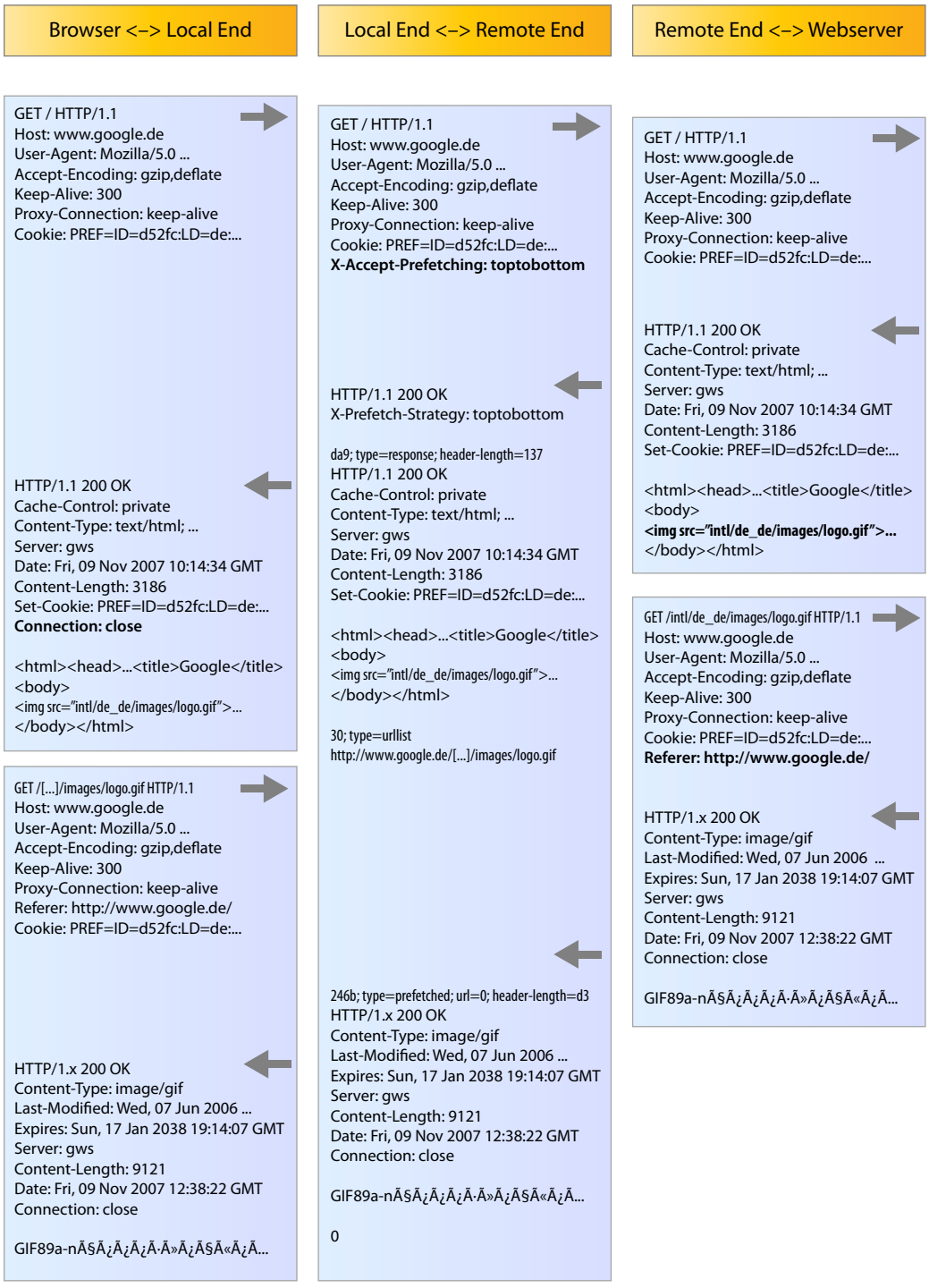


Abbildung 8.10.: Nachrichten beim Abruf von *www.google.de* über den Burst-Proxy

Gegenstelle vollständig empfangen haben. Dies gewährleistet zwar, dass beim Versand des Bildes ein homogener Datenstrom erzeugt wird, erhöht jedoch den Speicherbedarf erheblich. Dieses Problem lässt sich vermeiden, indem das Protokoll so geändert wird, dass auch Teile eines Objekts in einem Bündel übermittelt werden können. In Abschnitt 8.6 wird noch einmal auf diese *Chunk-Bündel-Strategie* eingegangen.

8.4.7. Implementierungsaspekte

HTTP-Proxy-Implementierung Als Basis für die Implementierung des Burst-Proxies wurde MyProxy[9] von Alexander Dietrich verwendet. MyProxy steht unter der GNU General Public License (GPLv2) und ist in Java implementiert (Voraussetzung: Java Runtime Environment v1.4). MyProxy enthält eine nahezu vollständige Implementierung des HTTP-1.1-Protokollsm, kann allerdings keine HTTPS-Verbindungen bearbeiten.

Erweiterung von MyProxy um zusätzliche Handler Da einige Webseiten im Testdatensatz über HTTPS abgerufen werden, wurde die HTTP-CONNECT-Methode implementiert *(ConnectRequestHandler)*. Bei HTTPS-Verbindungen kann der Burst-Proxy allerdings keine eingebetteten Objekte herunterladen, da er den Quelltext der Webseite nicht entschlüsseln kann. Er beschränkt sich daher auf die Weiterleitung des verschlüsselten Datenstroms.

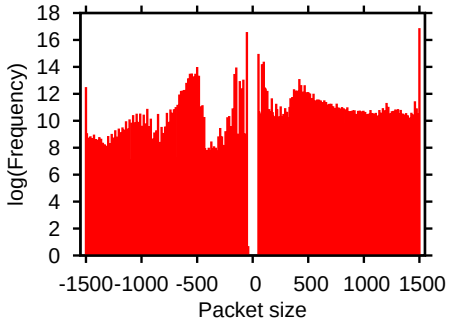
Darüber hinaus wurde der Handler-Pool von MyProxy durch einen flexibleren Thread-Pool ersetzt, der mehrere Worker-Threads verwalten kann. Das Remote End fügt die herunterzuladenden Objekte in die Warteschlange des Thread-Pools ein; die Worker-Threads kümmern sich dann um den parallelen Download. Der eigentliche Burst-Proxy-Code verteilt sich über mehrere dedizierte Request-Handler *(LocalPrefetchRequestHandler, RemotePrefetchRequestHandler, PrefetchingHandler)*.

Da Local End und Remote End dieselbe Code-Basis verwenden, kann man den modifizierten MyProxy durch eine einfache Konfigurationsanpassung entweder als Local End oder als Remote End betreiben. Änderungen am Quellcode sind hierzu nicht erforderlich.

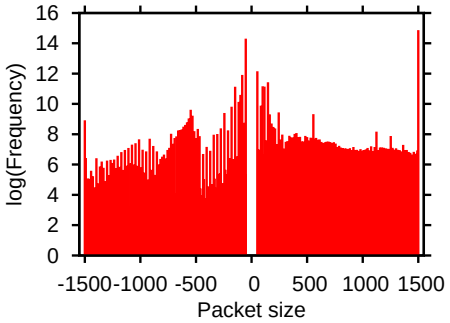
HTML- und CSS-Parser Zur Ermittlung der eingebetteten Objekte wird der Jericho-HTML-Parser[10] verwendet, der im Dokumentbaum nach folgenden HTML-Tags sucht: LINK, IMG, SCRIPT und STYLE. Der HTML-Parser beachtet auch das BASE-Tag, mit dem Web-Designer die Basis-URL für alle relativen Links in einer HTML-Seite festlegen können. Der HTML-Parser ist allerdings nicht in der Lage, die URLs von Objekten zu finden, die durch JavaScript oder Plugins nachgeladen werden.

Der CSS-Parser verwendet einen regulären Ausdruck, um verlinkte Grafiken oder andere CSS-Dateien zu finden. Hierzu wird in CSS-Dateien und STYLE-Tags nach *url\([\s"']*([^)]+?)[\s"']*\)* gesucht. Das Durchsuchen aller STYLE-Attribute in einer HTML-Datei dauert mit den aktuellen Parsern allerdings sehr lange, daher bleiben STYLE-Attribute im Prototyp unberücksichtigt.

[9] Homepage: http://dietrich.cx/wiki/Development/MyProxy
[10] Homepage: http://jerichohtml.sourceforge.net/



(a) OpenSSH normal



(b) OpenSSH mit Burst-Proxy

Abbildung 8.11.: Bei Verwendung des Burst-Proxies steigt der Anteil der Pakete mit einer Paketgröße von 1500 Byte

8.5. Evaluation des Burst-Proxies

In diesem Kapitel wird die Wirksamkeit des Burst-Proxies analysiert. Zur Gewährleistung der Vergleichbarkeit erfolgt die Evaluation des Burst-Proxies unter denselben Versuchsbedingungen, die auch für die anderen datenschutzfreundlichen Übertragungstechniken verwendet wurden (vgl. Tabelle 7.1, $\Delta_t = 12$ Stunden).

Burst-Proxy-Histogramm-Hypothese Ein wirksamer Burst-Proxy beeinflusst die Paketgrößen-Häufigkeitsverteilung: Die Anzahl der Pakete mit der MTU sollte steigen, die Anzahl der unterschiedlichen Paketgrößen sowie die normierte Entropie hingegen sinken.

Burst-Proxy-Erkennungsraten-Hypothese Bei der Verwendung des Burst-Proxies sind deutlich geringere Erkennungsraten zu erwarten, da durch die Bündelung von HTTP-Responses der Informationsgehalt der Instanzen reduziert wird.

Da beim Prototyp die Kommunikation zwischen Local und Remote End nicht verschlüsselt wird, muss die Verschlüsselung bei der Datenübertragung separat durchgeführt werden. Hierzu wird beim Prototyp OpenSSH verwendet. Bei der nachfolgenden Evaluierung des Burst-Proxies kommuniziert das Local End daher über Local-Port-Forwarding, das in Abschnitt 4.1.1 beschrieben wurde, mit dem Remote End.

Abbildung 8.11b zeigt das Histogramm der Paketgrößen-Verteilung, die sich bei Verwendung des Burst-Proxies über OpenSSH ergibt. Die Benchmark, OpenSSH ohne Burst-Proxy, ist in Abbildung 8.11a dargestellt. Es ist zu erkennen, dass die Anzahl der Pakete mit einer Größe von 1500 Byte im Verhältnis zu den anderen Paketgrößen ansteigt. Eine Analyse der Grundgesamtheiten bestätigt den visuellen Eindruck: der Anteil dieser Pakete ist von 32 % auf 51 % gestiegen (vgl. Tabelle 8.3). Dadurch steigt auch die mittlere Paketgröße.

Gleichzeitig ist die Anzahl der beobachtbaren Paketgrößen leicht gesunken, was zu einer erheblich geringeren Entropie führt. Diese Ergebnisse bestätigen die *Burst-Proxy-Histogramm-Hypothese*. Auch die *Burst-Proxy-Erkennungsraten-Hypothese* wird bestätigt: Der MNB-Klassifizierer erkennt nur noch knapp 75 % der Instanzen, wenn diese über einen Burst-Proxy abgerufen werden.

Parameter	OpenSSH normal	mit Burst-Proxy
Mittlere Paketgröße (Byte)	685	834
Anteil der Pakete mit 1500 Byte	0,32	0,51
Anzahl beobachteter Paketgrößen n	420	398
Normierte Entropie $H_{\text{norm}}(X)$	0,49	0,30
Erkennungsrate (TF, Normalisierung)	96,65 %	74,62 %

Tabelle 8.3.: Einfluss des Burst-Proxies auf Erkennungsraten und Histogramm

Die Ergebnisse zeigen, dass der Burst-Proxy Website-Fingerprinting-Angriffe, die auf der Analyse der Paketgrößen-Häufigkeitsverteilungen basieren, erschwert. Es ist davon auszugehen, dass durch Optimierung der HTML- und CSS-Parser die Erkennungsrate noch weiter reduziert werden kann. Wie in Abschnitt 8.4.7 beschrieben, ist der im Prototyp verwendete Parser zu langsam, um *url(...)*-Werte in STYLE-Attributen zu suchen – eine stichprobenhafte Durchsuchung der Testseiten ergab, dass einige Seiten davon Gebrauch machen.

Erwartete Wirksamkeit bei client-seitigen Skripten Die Evaluation der Wirksamkeit des Burst-Proxies wurde mit einem angepassten Browser durchgeführt: Alle aktiven Inhalte (JavaScript, Java, Plugins) wurden deaktiviert (vgl. Abschnitt 5.4.3). Es ist davon auszugehen, dass die Erkennungsraten ansteigen, wenn die Ausführung aktiver Inhalte zugelassen ist, da der Burst-Proxy diese Requests nicht vorhersehen und bündeln kann. Angesichts der zunehmenden Verbreitung von Webseiten, die AJAX und andere JavaScript-Funktionen verwenden, hat auch der Burst-Proxy nur einen begrenzten Anwenderkreis, nämlich Nutzer, die bereit sind zu Gunsten einer höheren Anonymität auf die Verwendung einiger Seiten zu verzichten.

8.6. Weiterentwicklung des Burst-Proxies

Für den Praxiseinsatz sind jedoch eine Reihe von Erweiterungen und Anpassungen im Burst-Proxy erforderlich. Zuallererst ist natürlich an die Optimierung der verwendeten HTML- und CSS-Parser zu denken, die derzeit nur einen Teil der eingebetteten Objekte identifizieren können. Dabei ist auch zu überprüfen, ob neben den bereits untersuchten HTML-Tags auch IFRAME-Tags, eingebettete RSS-Feeds sowie eine eventuell vorhandene *favicon.ico*-Datei vom Remote End selbstständig heruntergeladen werden können. Darüber hinaus erscheint vor allem die Analyse anderer Bündel-Strategien viel versprechend:

Analyse weiterer Bündel-Strategien

Das Bündeln von Objekten erfolgt derzeit anhand einer Mindestgröße, die etwa einem Fünftel der durchschnittlich empfangenen Datenmenge entspricht. Das Remote End erzeugt also durchschnittlich fünf Bündel mit jeweils mindestens 50 Kilobyte. Eine Untersuchung anderer Mindestgrößen steht allerdings noch aus. Möglicherweise lässt sich allein dadurch die Performanz und die Wirksamkeit des Burst-Proxies optimieren.

Es sind allerdings auch Bündel-Strategien vorstellbar, die nicht auf ganzen Objekten beruhen: Stattdessen könnten die Datenpakete *(Chunks)* hier auch aus Teilen von Objekten

bestehen. Dadurch ließe sich zum einen die Größe der Chunks präzise festlegen (zum Beispiel auf die empirisch ermittelte Durchschnittsgröße von einer Vielzahl von Bildern im Internet). Zum anderen müsste das Remote End nicht mehr die vollständigen Objekte im Speicher vorhalten. Dadurch ließe sich der Speicherbedarf erheblich reduzieren. Durch die Verwendung von Chunks laufen auch Denial-of-Service-Angriffe ins Leere, die bei einer objektbasierten Bündel-Strategie im Remote End einen Speicherüberlauf verursachen können, indem extrem große Bilder (z. B. 100 Megabyte) in HTML-Seiten eingebettet werden.

Durch den Übergang zu Chunks sind zudem Entscheidungsregeln vorstellbar, die auf anderen Parametern basieren. Während bei einer *größenabhängigen* Regel ein Chunk gesendet wird, wenn das erforderliche Mindestvolumen zum Transport bereit steht, sind auch *zeitabhängige* Regeln möglich, die alle *n* Sekunden einen Chunk senden, der die in diesem Intervall eingetroffenen Daten (sowie möglicherweise zusätzliche Dummy-Daten) enthält.

Die Verwendung von Chunks erhöht jedoch die Komplexität des Protokolls, da das Remote End dem Local End mitteilen muss, welchen Teil welcher Datei es jeweils überträgt.

Auflösung von HTTP-Redirects im Remote End Das Remote End könnte die Anzahl der ausgetauschten Nachrichten reduzieren, wenn es HTTP-Redirects erkennen und selbstständig auflösen würde. Derzeit sendet das Remote End bei einem Redirect lediglich den Redirect-Header zum Local End zurück, was dazu führt, dass der Browser kurz darauf einen Request für das Ziel der Umleitung stellt. Ein Angreifer könnte dieses Verhalten ausnutzen, um durch die Verkettung von HTTP-Redirects eine charakteristische Abfolge von Nachrichten zwischen Local und Remote End zu provozieren. Aktuelle Browser brechen die Verfolgung solcher Umleitungen zwar nach einigen Iterationen ab, diese könnten jedoch möglicherweise bereits ausreichen, um ein charakteristisches Muster im Datenverkehr zu erzeugen. Eine HTTP-Redirect-Auflösung im Remote End müsste solche „unendlichen“ Redirects ebenfalls erkennen und rechtzeitig abbrechen – andernfalls besteht die Gefahr eines Denial-of-Service-Angriffs.

9 Website-Fingerprinting in der Praxis

In Abschnitt 5.1 wurde eine Reihe von Arbeitshypothesen aufgestellt, die die Analyse von Website-Fingerprinting erleichtern. Diese Arbeitshypothesen sind jedoch in der Realität möglicherweise nicht erfüllt. Im Folgenden werden ausgewählte Hypothesen anhand empirischer Daten näher untersucht, um die Praktikabilität des vorgestellten Verfahrens einzuschätzen.

Darüber hinaus wird ein Verfahren zur Markierung von Webseiten vorgestellt, das es einem verteilten aktiven Angreifer ermöglicht, die Effektivität eines Website-Fingerprinting-Angriffs zu erhöhen. Das Kapitel schließt mit der Diskussion von juristischen Aspekten, die im Zusammenhang mit dem gezeigten Website-Fingerprinting-Angriff relevant sind.

9.1. Website-Fingerprinting bei aktiviertem Cache

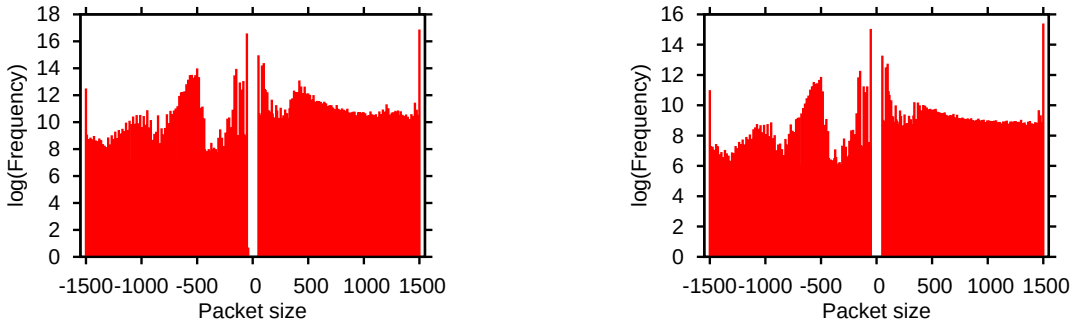
Die bisherigen Ergebnisse wurden bei deaktiviertem Browser-Cache erzielt. In der Realität ist der lokale Cache des Browsers jedoch häufig aktiviert. Zur Abschätzung des Einflusses des Browser-Caches wurden die zu testenden Webseiten über einen OpenSSH-Tunnel abgerufen, wobei der Cache aktiviert war. Die eingestellte Cache-Größe von 5 Gigabyte wäre sogar ausreichend, um alle 775 Seiten vollständig im Cache vorzuhalten. Tabelle 9.1 stellt die Untersuchungsbedingungen dar.

Durch den automatischen Abruf gibt es in jeder Klasse im Extremfall zwei Sorten von Instanzen: Zum einen die Instanzen, die den gesamten Datenverkehr enthalten (wenn es noch keinen Cache-Eintrag gab oder dieser veraltet war), zum anderen die Instanzen, bei denen die Seite großenteils aus dem Cache abgerufen werden konnte. Es ist davon auszugehen, dass sich diese beiden Instanzsorten deutlich voneinander unterscheiden. Der Klassifizierer erzielt nur dann hohe Erkennungsraten, wenn er beide Ausprägungen einer Klasse erkennt.

Die Zeitdifferenz zwischen zwei Abrufen derselben Seite betrug im Versuch etwa 1,3 Stunden. Diese Zeitspanne sollte kurz genug sein, so dass sich der Inhalt bei vielen Seiten zwischen zwei Abrufen nicht oder nur geringfügig geändert hat. Zumindest ein Teil

Variierter Parameter	Datensatz
Datensatz	OpenSSH, OpenSSH_withcache
Anzahl der Instanzen	$n_{train} = 4$, $n_{test} = 10$
Intervall zw. Training und Test	$\Delta_t = 12$ Stunden
Klassifizierungsverfahren	MNB-Klassifizierer
Transformation der Daten	TF mit Normalisierung
Metrik	Erkennungsrate

Tabelle 9.1.: Parameter zur Untersuchung des Einflusses des Browser-Caches



(a) OpenSSH normal (b) mit aktiviertem Browser-Cache

Abbildung 9.1.: Aktivierter Browser-Cache hat keinen sichtbaren Einfluss auf die Paketgrößen-Häufigkeitsverteilung

der Seite kann dann direkt aus dem lokalen Cache entnommen werden. Typischerweise sendet der Browser in einem solchen Fall einen konditionalen HTTP-Request (mit *If-Modified-Since*-Header) an den Webserver. Dieser entscheidet, ob die Kopie der Seite im Browser-Cache noch aktuell ist. Falls dies der Fall ist, wird in der HTTP-Response lediglich ein kurzer Header übertragen. Andernfalls wird in der Response der Seiteninhalt erneut übertragen.

Der Einfluss der Browser-Caches wird anhand der folgenden Hypothesen untersucht:

Browser-Cache-Ähnlichkeiten-Hypothese Es ist davon auszugehen, dass die Aktivierung des Browser-Caches Einfluss auf die Homogenität der Instanzen hat. Bei aktiviertem Browser-Cache sollten die Ähnlichkeit der Instanzen innerhalb der Klassen abnehmen und die Ähnlichkeit über Klassengrenzen hinweg zunehmen.

Browser-Cache-Erkennungsraten-Hypothese Bei aktiviertem Browser-Cache sollten erheblich weniger Daten übertragen werden. Ferner ist anzunehmen, dass die Anzahl der unterschiedlichen Pakete sinkt und damit dem Klassifizierer weniger Informationen zur Verfügung stehen. Dies sollte zu niedrigeren Erkennungsraten führen.

Ergebnisse

In Abbildung 9.1 sind die Histogramme von OpenSSH bei deaktiviertem bzw. aktiviertem Browser-Cache dargestellt. Obwohl die Messungen von OpenSSH ohne und mit

Parameter	OpenSSH normal	mit Browser-Cache
Gesendetes Datenvolumen pro Instanz	46 KB	30 KB
Empfangenes Datenvolumen pro Instanz	286 KB	246 KB
Anzahl beobachteter Paketgrößen n	420	393
Normierte Entropie $H_{\text{norm}}(X)$	0,49	0,44
Intra-Klassen-Ähnlichkeit	0,9777	0,9718
Inter-Klassen-Ähnlichkeit	0,8270	0,8653
Erkennungsrate (TF, Normalisierung)	96,65 %	91,70 %

Tabelle 9.2.: Vergleich der Datensätze beim Abruf mit OpenSSH sowie mit aktiviertem Browser-Cache

Cache fast zwei Monate auseinander liegen, sind die Histogramme kaum voneinander zu unterscheiden. Dies deutet zum einen auf die hohe zeitliche Robustheit von Paketgrößen-Häufigkeitsverteilungen hin. Zum anderen scheint die Aktivierung des Browser-Caches keinen erheblichen Einfluss auf die übertragenen Pakete zu haben.

Tabelle 9.2 führt die relevanten Statistiken auf. In der Tat ist das übertragene Datenvolumen gesunken. Auch die Anzahl der beobachtbaren Paketgrößen hat abgenommen. Die normierte Entropie der Häufigkeitsverteilung ist jedoch nur geringfügig gefallen. Die Erkennungsrate ist zwar bei aktiviertem Cache signifikant kleiner, sie beträgt knapp 92 %. Die *Browser-Cache-Erkennungsraten-Hypothese* wird also bestätigt.

Die Aktivierung des Browser-Caches hat jedoch bei diesem Test keine Auswirkung auf die Ähnlichkeit der Instanzen innerhalb der Klassen, und die Ähnlichkeit der Instanzen über Klassengrenzen hinweg steigt nur geringfügig. Die *Browser-Cache-Ähnlichkeiten-Hypothese* lässt sich durch die Ergebnisse nicht bestätigen. Offenbar wirkt sich der Einsatz von Caching – entgegen der anfänglichen Vermutung – nur minimal auf die Homogenität der Instanzen in den Klassen aus.

Bewertung

Der Versuch zeigt, dass der Einfluss des Browser-Caches zwar messbar ist, jedoch die Erkennungsleistung nur minimal verringert. Website-Fingerprinting ist also auch bei aktiviertem Browser-Cache durchführbar. Der Angreifer sollte jedoch sicherstellen, dass die Trainingsinstanzen sowohl Instanzen enthalten, bei denen die Seite bereits im Cache vorhanden war, als auch solche, bei denen große Teile der Seite über das datenschutzfreundliche System heruntergeladen wurden. Möglicherweise lassen sich die Erkennungsraten auch durch eine Erhöhung der Anzahl der Trainingsinstanzen verbessern.

Die bisherigen Untersuchungsergebnisse, die aufgrund der Arbeitshypothesen in Abschnitt 5.1 bei deaktiviertem Browser-Cache durchgeführt wurden, überschätzen die in der Praxis erzielbaren Erkennungsraten bei aktiviertem Cache also nur moderat.

9.2. Effektivität bei unbekannten Testinstanzen

Bislang wurde davon ausgegangen, dass die Menge der Webseiten, die der Nutzer abrufen kann, beschränkt und dem Angreifer bekannt ist. Zur Evaluierung der Genauigkeit des

Website-Fingerprinting-Verfahrens war es daher ausreichend, den Anteil der korrekt klassifizierten Instanzen (die *True Positives*) zu betrachten. Diese Beschränkung soll nun zur Erhöhung der Realitätsnähe aufgehoben werden.

In einem separaten Versuch mit dem OpenSSH-Datensatz werden dem Klassifizierer in der Trainingsphase nicht mehr Trainingsinstanzen aller Klassen zur Verfügung gestellt, sondern nur 10 % der URLs (also 78 der 775 URLs), die zufällig ohne Zurücklegen aus der Menge der URLs gezogen werden. Dabei handelt es sich um die für den Angreifer interessanten Seiten, deren Besuch er durch den Website-Fingerprinting-Angriff erkennen will. In der Testphase muss der Klassifizierer dann Instanzen aus allen Klassen klassifizieren. Zunächst soll der bekannte MNB-Klassifizierer untersucht werden. Er wird wie in den bisherigen Versuchen auf die TF-transformierten, normalisierten Häufigkeitsvektoren angewendet. Der Versuch wird mit 25 Stichproben wiederholt, wobei 4 Trainings- und 10 Testinstanzen sowie ein Intevall Δ_t von 12 Stunden verwendet werden. In den ARFF-Dateien befinden sich also jewels $4 \cdot 78 + 10 \cdot 775 = 8062$ Instanzen.

Da der Klassifizierer nur die Klassen von $78/775$ der Instanzen kennt, beträgt die bestmögliche Erkennungsrate bei diesem Versuch 10,06 %. Wendet man den MNB-Klassifizierer von Weka auf die oben beschriebenen ARFF-Dateien an, erzielt er eine durchschnittliche Erkennungsrate von 9,88 %.

Im ersten Moment sieht dieses Ergebnis äußerst zufriedenstellend aus. Für den Praxiseinsatz ist der Klassifizierer damit allerdings nicht zu gebrauchen: Eine Erkennungsrate von 9,88 % bedeutet, dass der Klassifizierer 90,12 % der Instanzen (7265) der falschen Klasse zugewiesen hat. Der MNB-Klassifizierers von Weka prognostiziert bei jeder Instanz eine URL, egal wie sicher er sich dabei ist. Bildlich gesprochen alarmiert ein solcher Klassifizierer den Angreifer bei *jeder* Webseite, wobei es sich in 90 % der Fälle um einen Fehlalarm handelt.

MNB-Klassifizierer mit Schwellwert Das oben beschriebene Problem bezieht sich auf die Kosten, die mit einer Fehlklassifizierung verbunden sind. In der Realität sind die Kosten eines Fehlalarms (False Positive) beim Website-Fingerprinting nicht zu vernachlässigen. Der Verringerung der False Positives kommt dann ebenfalls eine hohe Bedetung zu.

Zur Abschätzung der Leistungsfähigkeit des MNB-Klassifizierers in solchen Szenario wurde der MultinomialNaiveBayes-Klassifizierer von Weka angepasst, so dass der Benutzer eine Mindestwahrscheinlichkeit für eine Klassifizierung angeben kann. Ist selbst bei der wahrscheinlichsten Klasse c_{map} die Wahrscheinlichkeit $\hat{P}(c|d)$ geringer als der Schwellwert, ordnet der Klassifizierer die Testinstanz überhaupt keiner Klasse zu – das entspricht genau dem erwünschten Verhalten bei einer Testinstanz, für die es keine Trainingsdaten gibt.

Die Ergebnisse der Evaluation des angepassten MNB-Klasifizierers werden im Folgenden anhand von zwei Kennzahlen diskutiert: Der Anteil der korrekt klassifizierten interessanten Seiten $p_{\text{interesting}} = \frac{n_{\text{interesing,correct}}}{n_{\text{interesting}}}$ sowie der Anteil der korrekterweise nichtklassifizierten uninteressanten Seiten $p_{\text{uninteresting}} = \frac{n_{\text{uninteresting,unclassified}}}{\text{n}_{\text{uninteresting}}}$. Im Optimalfall beträgt der Anteilswert in beiden Fällen 1: Dann hat der Klassifizierer alle interessanten Seiten korrekt klassifiziert und bei keiner der uninteressanten Seiten einen Fehlalarm ausgelöst. In Abbildung 9.2 sind die resultierenden Wertepaare $\left(p_{\text{interesting}}, p_{\text{uninteresting}}\right)$ für unterschiedliche Mindestwahrscheinlichkeiten dargestellt (Kurve „ohne UNKNOWN-Klasse“). Die höchste Mindestwahrscheinlichkeit hat der Punkt links oben, die niedrige

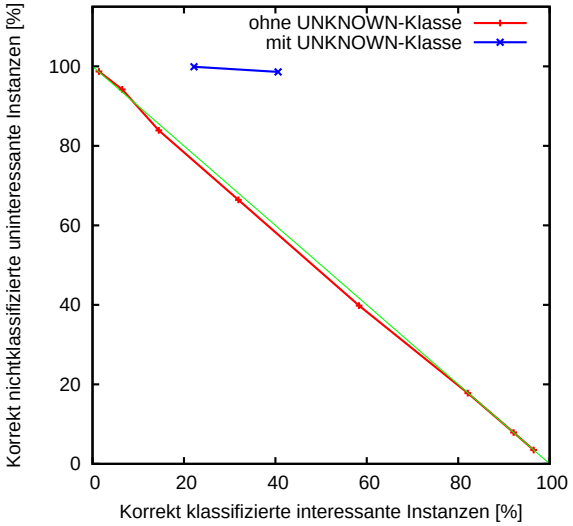


Abbildung 9.2.: Beim MNB-Klassifizierer muss man einen Kompromiss zwischen $p_{\text{interesting}}$ und $p_{\text{uninteresting}}$ eingehen („ohne UNKNOWN-Klasse“). Bei Verwendung einer UNKNOWN-Klasse steigt die Trennschärfe des Klassifizierers deutlich.

Mindestwahrscheinlichkeit, der Punkt rechts unten.[1] Die einzelnen Wertepaare sind durch Punkte markiert, die durch eine Linie verbunden sind.

Die Darstellung ist folgendermaßen zu lesen: wenn der Klassifizierer 96,5 % der interessanten Instanzen korrekt erkennt, werden lediglich 3,5 % der uninteressanten Instanzen korrekterweise nicht klasifiziert – also wird bei 96,5 % der uninteressanten Seiten ein Fehlalarm produziert. Wird durch eine höhere Mindestwahrscheinlichkeit die Anzahl der Fehlalarme reduziert, so dass nur noch 58 % der uninteressanten Seiten zu einem Fehlalarm führen, werden auch nur noch 58 % der interessanten Seiten erkannt. Ein optimales Klassifizierungsergebnis ergibt sich im Punkt (100 %,100 %), also in der rechten oberen Ecke.

Man kann beim MNB-Klassifizierer also anhand der Mindestwahrscheinlichkeit einstellen, ob man mehr Wert auf eine hohe Anzahl korrekt klassifizierter Instanzen legt, oder ob die Reduktion der False Positives im Vordergrund steht (also die Anzahl der korrekt nichtklassifizierten uninteressanten Instanzen maximiert werden soll).

Die Verbindungslinie zwischen den markierten Punkten deckt sich fast perfekt mit der Gerade $y = 100 - x$. Es besteht also ein indirekt proportionaler Zusammenhang zwischen $p_{\text{interesting}}$ und $p_{\text{uninteresting}}$. In dem Maß, in dem die False Positives zurückgehen (also $p_{\text{uninteresting}}$ zunimmt), fällt auch die Erkennungsrate bei den interessanten Seiten ($p_{\text{interesting}}$ nimmt ab). Die Reduktion der Mindestwahrscheinlichkeit wirkt sich gleichermaßen auf die Erkennungsrate bei den interessanten als auch bei den uninteressanten Seiten aus. Für eine angemessene Reduktion der False Positives muss die Mindestwahrscheinlichkeit dadurch so hoch sein, dass auch viele True Positives durch das Raster fallen und dann vorsichtshalber keiner Klasse zugeordnet werden.

[1] Da der MNB-Klassifizierer von Weka zur Optimierung nicht die exakten Wahrscheinlichkeiten schätzt, liegen alle Wahrscheinlichkeitswerte in der Nähe von 1,0. Die verwendeten Mindestwahrscheinlichkeiten sind daher nicht besonders aussagekräftig: $\{0.99, 0.9999, 0.999999, 0.99999999, 0.9999999999, 0.999999999999, 0.99999999999999, 0.9999999999999999\}$

Erhöhung der Trennschärfe Wünschenswert ist jedoch ein Klassifizierer, der eine niedrige Anzahl von False Positives mit einer hohen Erkennungsrate vereinbart, also eine hohe Trennschärfe aufweist. Die Verbindungslinie zwischen den einzelnen Punkten $(p_{\text{interesting}}, p_{\text{uninteresting}})$ müsste also einen konkaven Verlauf in der oberen Diagrammhälfte haben. Abschließend soll gezeigt werden, dass auch beim MNB-Klassifizierer die Möglichkeit besteht, die Trennschärfe zu erhöhen.

Der Angreifer zeichnet bei der Vorklassifizierung der Trainingsinstanzen nicht nur den Datenverkehr der Webseiten auf, für die er sich tatsächlich interessiert, sondern auch den Datenverkehr von Webseiten, die für ihn gar nicht interessant sind. Wenn der Angreifer weiß, welche (uninteressanten) Seiten sein Opfer typischerweise besucht, ist es sinnvoll, dass er insbesondere diese Seiten abruft. Kann er keine Annahmen über das Opfer machen, bietet es sich an, den Datenverkehr besonders populärer Seiten aufzuzeichnen.

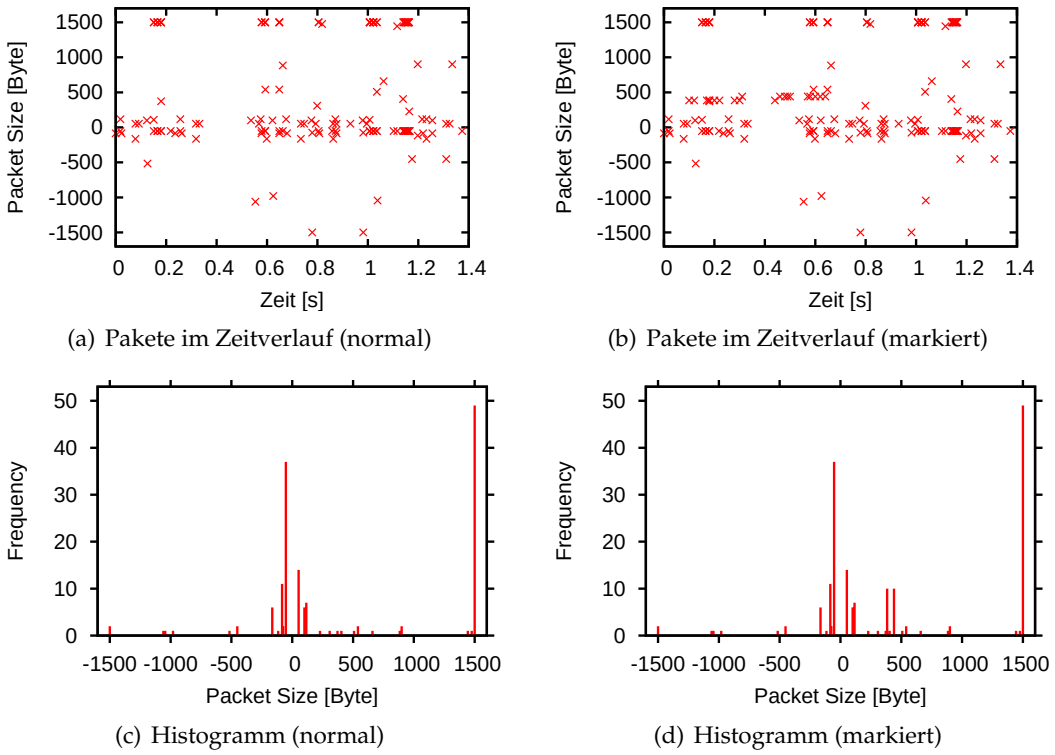
In der Trainingsphase werden dem Klassifizierer dann sowohl Instanzen der interessanten Seiten präsentiert als auch Instanzen der uninteressanten Seiten. Im Unterschied zur bisherigen Vorgehensweise befinden sich die Instanzen aller uninteressanten Seiten jedoch in einer gemeinsamen Klasse – der UNKNOWN-Klasse. Diese Klasse entspricht dadurch dem *Prototyp* aller uninteressanten Instanzen, so dass der Klassifizierer in der Testphase die Möglichkeit hat, eine „durchschnittlich" aussehende Instanz der UNKNOWN-Klasse zuzuordnen. Dadurch ist es möglich, den Klassifizierer mit einem niedrigeren Schwellenwert für die Mindestwahrscheinlichkeit zu betreiben, was die Anzahl der True Positives erhöhen kann.

Zur Evaluation dieses Verfahrens werden dem MNB-Klassifizierer in der Trainingsphase zum einen die $4 \cdot 78 = 312$ Trainingsinstanzen der 78 interessanten Seiten zur Verfügung gestellt. Darüber hinaus sind in den Trainingsdaten $4 \cdot 697 = 2778$ Trainingsinstanzen enthalten, die aus den anderen 697 URLs gezogen wurden. Diese gehören zur Klasse *UNKNOWN*. In Abbildung 9.2 sind zwei Wertepaare für $(p_{\text{interesting}}, p_{\text{uninteresting}})$ eingezeichnet (mit UNKNOWN-Klasse). Die Trennschärfe des Klassifizierers hat sich dadurch erheblich verbessert: Wenn der Klassifizierer 98,6 % der uninteressanten Seiten korrekterweise keiner Klasse zuordnet, also es lediglich in 1,4 % der Fälle zu einem False Positive kommt, werden immerhin 40,6 % der interessanten Instanzen korrekt identifiziert. Dieser Wert ist allerdings erheblich schlechter als bei Sun et al.: Der dort eingesetzte Klassifizierer erreichte den Punkt $(75\%; 1.5\%)$. Allerdings basiert der dort verwendete Klassifizierer auf Objektgrößen und nicht auf der Paketgrößen-Häufigkeitsverteilung.

9.3. Gezielte Markierung von Webseiten

In diesem Abschnitt wird ein Verfahren vorgestellt, mit dem man den Datenverkehr von Webseiten gezielt um charakteristische Merkmale erweitern kann, um ihre Erkennung mit dem MNB-Klassifizierer zu erleichtern. In Abschnitt 9.2 wurde gezeigt, dass 60 % der Instanzen in der Praxis nicht identifizierbar sind, wenn die Minimierung der False Positives im Vordergrund steht. Das im Folgenden vorgestellte Verfahren verhilft solchen Webseiten zu einer hohen Erkennungsrate.

Angreifermodell Das Verfahren zur Markierung von Webseiten erfordert im Gegensatz zu den bisher beschriebenen Website-Fingerprinting-Angriffen einen verteilten aktiven Angreifer, der Zugriff auf den Webserver der Seite hat, die er markieren möchte (Position 5



(a) Pakete im Zeitverlauf (normal)

(b) Pakete im Zeitverlauf (markiert)

(c) Histogramm (normal)

(d) Histogramm (markiert)

Abbildung 9.3.: Markierung von Webseiten durch Einfügen charakteristischer Pakete (10 Pakete mit 384 Byte und 10 Pakete mit 440 Byte)

in Abbildung 2.2). Darüber hinaus muss der Angreifer wie bei einem normalen Website-Fingerprinting-Angriff an einer Stelle im Netzwerk Zugriff auf den Datenverkehr haben, wo er die Identität des Nutzers anhand der in den IP-Paketen enthaltenen IP-Adressen ermitteln kann (etwa Position 2 in Abbildung 2.2).

Vorgehensweise Die Vorgehensweise wird am Beispiel der Webseite *www.mainfranken-chat.de* beim Abruf über OpenSSH erläutert (bei Tor oder JonDonym ist das Verfahren in der demonstrierten Form nicht durchführbar). Diese Seite konnte im Versuch in Abschnitt 9.2 kein einziges Mal vom MNB-Klassifizierer identifiziert werden. Das Auftreten der Pakete im Zeitverlauf und das Histogramm einer Instanz dieser Seite sind in den Abbildungen 9.3a und 9.3c dargestellt.

Bei der Markierung der Seite nutzt der Angreifer die Tatsache aus, dass auch bei einer großen Anzahl von Webseiten nicht alle möglichen Paketgrößen auftreten. Zur Ermittlung geeigneter Paketgrößen betrachtet der Angreifer zunächst die Häufigkeitsverteilung aller ihm bekannten Instanzen (das Histogramm des OpenSSH-Datensatzes ist in Abbildung 7.4 dargestellt). Dann wählt er einige Paketgrößen aus, die im Histogramm der Grundgesamtheit und der zu markierenden Seite besonders selten oder überhaupt nicht vorkommen – im OpenSSH-Histogramm kommen unter anderem die Paketgrößen 384 Byte und 440 Byte in Frage.

Das Ziel des Angreifers ist es, die Webseite so zu modifizieren, dass diese charakteristischen Paketgrößen im Datenverkehr enthalten sind, der beim Abruf der Seite entsteht.

Auf geeignete Methoden zur Erzeugung charakteristischer Paketgrößen wird noch eingegangen. Nach der Modifikation erzeugt der Angreifer durch automatisierten Abruf der Seite mit einem Browser wie gehabt vorklassifizierte Instanzen, mit denen er den MNB-Klassifizierer trainiert. Durch die charakteristischen Paketgrößen unterscheidet sich die Seite nun erheblich von allen anderen und ist für den Klassifizierer erheblich leichter zu identifizieren. In den Abbildungen 9.3b und 9.3d ist der Datenverkehr nach Einbetten von jeweils 10 Paketen mit 384 Byte und 440 Byte dargestellt.

Nach diesen Vorbereitungen protokolliert der Angreifer wie bei einem normalen Website-Fingerprinting-Angriff den verschlüsselten Datenverkehr eines bzw. mehrerer Nutzer, um zu ermitteln, ob die markierte Webseite abgerufen wurde.

Modifikation der Seite

Da unterstellt wird, dass der Angreifer Zugriff auf den Webserver hat, der die Webseite ausliefert, hat er die Möglichkeit in den Quelltext der HTML-Seite zusätzliche (im Layout nicht sichtbare) Objekte einzubinden, die eine genau festgelegte Größe haben. Hierzu sind besonders Bilddateien geeignet, da diese von fast allen Browsern ohne Rückfrage abgerufen werden. Während sich die Größe der einzubindenden Objekte aus den ermittelten Paketgrößen ergibt, kann der Angreifer die Anzahl in gewissen Grenzen frei wählen. Er muss einen Kompromiss finden zwischen höherer Eindeutigkeit (was für mehr Bilder spricht) und der Minimierung des zusätzlichen Datenverkehrs (damit die Markierung nicht auffällt).

Die Größe der Objekte muss etwas kleiner sein als die gewünschte Paketgröße, da in dem IP-Paket neben dem eigentlichen Objekt auch noch der HTTP-Header übertragen wird. Zudem ist das Padding auf die Blockgröße zu berücksichtigen. Durch Ausprobieren sollte sich die optimale Größe jedoch ermitteln lassen. Darüber hinaus muss der Angreifer verhindern, dass der Browser mehrere Objekte mittels HTTP-Pipelining in einer einzigen Verbindung abruft. Dies lässt sich zum Beispiel bewerkstelligen, indem die zusätzlichen Objekte dynamisch von einem PHP-Skript generiert werden, das den HTTP-Headers *Connection: close* einfügt. Dadurch wird der Webserver dazu veranlasst, die Verbindung nach der Übertragung eines Objekts zu schließen.

Wasserzeichen

Eine Erweiterung des beschriebenen Konzepts ist die Einbettung von Wasserzeichen in den Datenverkehr. Auf eine umfassende Diskussion solcher Verfahrens soll an dieser Stelle verzichtet werden. Stattdessen wird die Vorgehensweise lediglich beispielhaft illustriert.

Der Angreifer identifiziert zunächst $n_{\text{packetsize}}$ Paketgrößen, die im typischen Datenverkehr sehr selten auftreten. Mit einer konservativen Kodierung könnte er dann $n_{\text{packetsize}}$ Bit in die Paketgrößen-Häufigkeitsverteilung einbetten, indem der Wert 0 eines Bits durch die Paketgrößen-Auftretenshäufigkeit 10 dargestellt wird und der Wert 1 durch die Häufigkeit 15. Auch leistungsfähigere Kodierungen, die den Datenverkehr weniger auffällig verändern, sind denkbar: Sie könnten sich der Techniken bedienen, die zur Einbettung von Wasserzeichen in Medieninhalten verwendet werden.

Erkennen von markierten Webseiten im Datenverkehr

Bei der systematischen Evaluierung von Website-Fingerprinting unter Laborbedingungen ist durch den Versuchsaufbau gewährleistet, dass in einer Instanz genau der Datenverkehr enthalten ist, der beim Abruf der jeweiligen Webseite entsteht. In der Praxis ist es jedoch

eine Herausforderung, aus einem kontinuierlichen Datenstrom einen Website-Abruf zu extrahieren – es gibt Hinweise darauf, dass die Denkpausen-Arbeitshypothese (vgl. Abschnitt 5.1) in der Realität nicht erfüllt ist [KAA06; CCW+07]. Im Folgenden wird ein Verfahren vorgeschlagen, das den Abruf individuell markierter Webseiten automatisch erkennen kann.

In den bisherigen Versuchen wurde gezeigt, dass bei den meisten Seiten in der Empfangsrichtung die größtmögliche Paketgröße (die MTU) dominiert. Dieser Umstand erlaubt es, einen *Paketgrößen-Sniffer* zu konstruieren, der speziell markierte Webseiten auch ohne aufwändiges Klassifizierungsverfahren identifizieren kann. Hierzu misst der Sniffer die *Konzentration der einzelnen Paketgrößen* innerhalb eines festgelegten Zeitintervalls, zum Beispiel 10 Sekunden. Steigt die Konzentration der Paketgrößen, die zur Markierung einer Webseite verwendet wurden, plötzlich an, ist dies ein Indiz dafür, dass gerade die markierte Webseite abgerufen wird.

Um auszuschließen, dass eine andere Webseite durch Zufall ebenfalls Pakete mit der charakteristischen Größe erzeugt und dadurch fälschlicherweise identifiziert wird, könnte der Angreifer die Pakete mit den charakteristischen Größen zusätzlich in einem festgelegten Verhältnis in der Webseite einbetten, so dass die Wahrscheinlichkeit einer zufälligen Übereinstimmung reduziert wird: Die Wahrscheinlichkeit, dass mindestens 10 Pakete der Größen 440, 348 und 130 Byte in einem 10-Sekunden-Intervall auftreten *und* ihre Auftretenshäufigkeiten noch dazu im Verhältnis 1:2:3 stehen, dürfte bei nicht-markierten Webseiten extrem gering sein.

Würde ein solcher *Paketgrößen-Sniffer* bei Internetzugangsanbietern oder dem ersten Knoten von Anonymisierungsdiensten platziert werden, könnte der Datenverkehr von vielen Teilnehmern gleichzeitig nach dem charakteristischen Muster durchsucht werden, mit dem eine Webseite markiert wurde. Zum Glück gibt es derzeit keine rechtliche Grundlage, auf Grund derer die Anbieter von Telekommunikationsdiensten zu einer solchen Maßnahme verpflichtet werden können.

9.4. Juristische Aspekte von Website-Fingerprinting

Es ist nur eine Frage der Zeit, bis die Zuverlässigkeit von Website-Fingerprinting-Angriffen so hoch ist, dass sie für die Zwecke der Strafverfolgung interessant werden. Während bislang die technischen Herausforderungen im Vordergrund standen, wird in diesem Abschnitt der Frage nachgegangen, inwiefern solche Angriffe auf Grund der derzeitigen (deutschen) Rechtslage durchführbar sind.

Möglichkeit zur Aufzeichnung des Datenverkehrs

Telekommunikationsinhalte sind durch das Fernmeldegeheimnis (Art. 10 Abs. 1 GG) vor unbefugtem Abhören geschützt. Dieses Grundrecht wird jedoch durch Regelungen in der Strafprozessordnung (StPO), dem G-10-Gesetz, dem Zollfahndungsgesetz sowie den Landesgesetzen beschränkt.

Die Betreiber von Telekommunikationsanlagen, die für die Öffentlichkeit bestimmt sind, sind gemäß § 110 TKG verpflichtet, Vorkehrungen für die Umsetzung von Überwachungsmaßnahmen zu treffen. Die technischen und organisatorischen Grundsätze sind in der Telekommunikations-Überwachungsverordnung (TKÜV) festgelegt. Zum Kreis der Verpflichteten gehören insbesondere Internetzugangsanbieter (§ 3 TKÜV S. 2 Nr. 1). Zentrale

Internet-Backbones, die von einer Vielzahl von Nutzern verwendet werden, sind von der Überwachung allerdings ausgeschlossen.

Anordnung einer Überwachungsmaßnahme

Die Überwachung des Datenverkehrs eines Benutzers *(Individualkontrolle)* ist nur bei Vorliegen der in § 100a StPO genannten Voraussetzungen möglich, zum Beispiel bei Verdacht auf bestimmte schwere Straftaten (Katalogstraftaten wie Mord, Totschlag, Raub, Erpressung, Hochverrat, Gefährdung des demokratischen Rechtsstaates, Gefährdung der äußeren Sicherheit und andere). Zur Anordnung einer Individualkontrolle ist eine richterliche Genehmigung erforderlich. Bei Gefahr im Verzug kann eine Überwachungsmaßnahme für die Dauer von höchstens drei Tagen auch direkt durch die Staatsanwaltschaft angeordnet werden. (§ 100b Abs. 1 StPO).

Das *Gesetz zur Beschränkung des Brief-, Post- und Fernmeldegeheimnisses*, auch *G-10-Gesetz* (G10G) genannt, ermöglicht unter bestimmten Umständen die *strategische Kontrolle*, bei der das Fernmeldegeheimnis ohne Bezug auf bestimmte Personen beschränkt wird. Diese Maßnahme ist nur bei bestimmten schwerwiegenden Gefahren (etwa eines bewaffneten Angriffs oder terroristischen Anschlags, der Verbreitung von Kriegswaffen, der Verbringung von Betäubungsmitteln in erheblicher Menge oder der Geldwäsche) zulässig (§ 5 G-10-G).

Voraussetzung für Website-Fingerprinting

Nach der Anordnung einer Überwachungsmaßnahme ist sowohl die Telekommunikation, die für die zu überwachende Kennung bestimmt ist, als auch die Telekommunikation, die von ihr ausgeht, an die berechtigte Stelle zu übermitteln (§ 5 Abs. 1 TKÜV). Bei Internetzugangsdiensten erstreckt sich die Überwachung also auf den Datenverkehr in Sende- und Empfangsrichtung.

Die Einzelheiten zur Durchführung einer Telekommunikationsüberwachung sind in einer „Technischen Richtlinie“ (TR TKÜ) geregelt [Bun06]. Die Überwachungskopie ist demnach über ein kryptographisch abgesichertes VPN unverzüglich der berechtigten Stelle zur Verfügung zu stellen.

Die berechtigten Stellen sind in § 2 Nr. 3 TKÜV definiert: Zur Durchführung der Individualkontrolle sind nach § 100b Abs. 3 S. 1 StPO die Staatsanwaltschaften und ihre im Polizeidienst tätigen Ermittlungspersonen berechtigt. Zur strategischen Kontrolle sind gemäß § 1 Abs. 1 Nr. 1 G10G die Verfassungsschutzbehörden des Bundes und der Länder, der militärische Abschirmdienst sowie der Bundesnachrichtendienst berechtigt.

Die genannten Behörden sind demnach schon heute unter den beschriebenen Voraussetzungen zur Durchführung von Website-Fingerprinting-Angriffen in der Lage.

Verwertbarkeit der Erkenntnisse

Probabilistische Erkenntnisse über den Besuch einer Seite sind vor Gericht als Beweismittel verwendbar – genauso wie echte Fingerabdrücke oder DNA-Spuren. Es wird aber wohl kaum gelingen, einen Internetnutzer, dem man mit einer Wahrscheinlichkeit von 97 % den Besuch einer Seite mit kinderpornografischem Material nachweisen kann, rechtskräftig zu verurteilen, wenn bei der Hausdurchsuchung kein belastendes Material zu finden ist. Die probabilistischen Erkenntnisse können jedoch ausreichen, um einen Anfangsverdacht zu begründen, der zum Beispiel zur Anordnung einer Hausdurchsuchung erforderlich ist.

In diesem Zusammenhang ist allerdings darauf hinzuweisen, dass die Voraussetzungen für eine Hausdurchsuchung im Allgemeinen niedriger sind als für die Überwachung der Telekommunikation, die nur bei schweren Straftaten zulässig ist. Der zur Telekommunikationsüberwachung erforderliche Anfangsverdacht reicht also normalerweise ohnehin aus, um unmittelbar eine Hausdurchsuchung anzuordnen. Der Telekommunikationsüberwachung ist der Vorzug zu geben, wenn man sich durch die Ermittlung der besuchten Seiten zusätzliche Erkenntnisse oder Beweismittel verspricht.

Die Vorgehensweise der Strafverfolgungsbehörden im einführenden Beispiel (s. Abschnit 1.2) ist damit sowohl aus technischer als auch aus rechtlicher Sicht plausibel. Wie die aktuellen Ermittlungen wegen Steuerhinterziehung zeigen, lässt sich in der Realität eine Hausdurchsuchung allerdings bereits anhand der Tatsache begründen, dass der Name eines Bürgers auf einer DVD gefunden wurde [DJK+08].

10 Zusammenfassung und Ausblick

10.1. Untersuchungsergebnisse

In dieser Arbeit wurde ein Website-Fingerprinting-Angriff vorgestellt, der es einem lokalen Angreifer ermöglicht, die Beziehungsanonymität von datenschutzfreundlichen Übertragungstechniken aufzuheben. Hierzu wurde ein leistungsfähiges Fingerprinting-Verfahren entwickelt, das auf dem multinomialen Naïve-Bayes-Klassifizierer basiert. Es behandelt den Datenverkehr, der beim verschlüsselten Abruf von Webseiten zu beobachten ist, wie ein Textdokument. Durch geschickte Transformationen der Auftretenshäufigkeiten der unterschiedlichen Paketgrößen ist ein Angreifer damit in der Lage, unter Idealbedingungen bis zu 97 % der Seitenabrufe korrekt zu identifizieren. Keines der etablierten Fingerprinting-Verfahren erzielt unter vergleichbaren Bedingungen so hohe Erkennungsraten in so kurzer Zeit. Hinzu kommt, dass das Klassifiziererungsverfahren dieses hohe Niveau bereits bei einer einzigen Trainingsinstanz erreicht, wohingegen bei etablierten Verfahren mindestens vier Trainingsinstanzen erforderlich sind, um akzeptable Erkennungsraten zu erzielen.

Das zentrale Ergebnis dieser Arbeit ist die Evaluation der Effektivität des entwickelten Website-Fingerprinting-Angriffs beim Einsatz von unterschiedlichen datenschutzfreundlichen Übertragungstechniken. Die Erkennungsraten bei OpenSSH, OpenVPN, dem Cisco-IPsec-VPN, Stunnel und dem auf TLS basierten Multi-Hop-System Shalon liegen – unter idealen Versuchsbedingungen – bei über 90 %. Lediglich die Anonymisierungssysteme JonDonym und Tor reduzieren die Effektivität des Angriffs. Bei diesen Systemen werden 20 % bzw. 3 % der Seitenabrufe korrekt identifiziert. Die vergleichsweise niedrige Erkennungsrate bei Tor geht jedoch offenbar nicht auf dessen Systemdesign zurück. Sie kommt vermutlich durch Störfaktoren und die geringe Zuverlässigkeit des Tor-Netzwerks zu Stande. Dadurch ist der Datenverkehr beim mehrmaligen Abruf derselben Seite über Tor so inhomogen, dass eine korrekte Zuordnung nur noch selten möglich ist.

Bei der Untersuchung der Grundgesamtheiten stellte sich heraus, dass es einen Zusammenhang zwischen den Erkennungsraten und der normierten Entropie der Paketgrößen-Häufigkeitsverteilung gibt. Dies ist insofern bemerkenswert, da die Erkennungsraten von der Anzahl der unterschiedlichen Paketgrößen in den Histogrammen unabhängig sind.

Darüber hinaus wurde untersucht, inwieweit Gegenmaßnahmen, die in der Sitzungs- bzw. Anwendungsschicht wirken, vor dem Website-Fingerprinting-Angriff schützen können. Während die Verwendung von zusätzlichem Padding am Ende von TLS-Application-Records praktisch wirkungslos ist, sinken die Erkennungsraten um über 20 %, wenn ein Burst-Proxy verwendet wird. Die Effektivität des Burst-Proxies lässt sich durch robustere HTML- und CSS-Parser vermutlich noch weiter steigern.

In den bisherigen Publikationen wird Website-Fingerprinting stets unter Idealbedingungen durchgeführt. So wird etwa unterstellt, dass der Benutzer den Cache seines Browsers deaktiviert hat und der Angreifer bereits vor dem Angriff die Grundgesamtheit der Seiten kennt, die für einen Abruf in Frage kommen. Zur Untersuchung der Praxistauglichkeit des vorgestellten Website-Fingerprinting wurden diese Arbeitshypothesen gelockert. Die Aktivierung des Browser-Caches führte nur zu einer geringfügigen Verschlechterung der Erkennungsrate: Bei Verwendung von OpenSSH wurden immer noch über 90 % der Seiten korrekt identifiziert. Kennt der Angreifer hingegen vor dem Angriff die Menge der Webseiten nicht, die sein Opfer abruft, rückt die Reduzierung der *False Positives*, uninteressante Seiten, bei denen der Klassifizierer fälschlicherweise anspricht, in den Vordergrund. Die Reduzierung der False Positives auf 1 % führte bei dem entwickelten Verfahren allerdings zu einem Einbruch der Erkennungsrate: Im Versuch sank diese auf 40 %. Hier gibt es noch Untersuchungsbedarf.

Während Website-Fingerprinting unter Idealbedingungen sehr gut funktioniert, ist es noch ein weiter Weg bis zur uneingeschränkten Praxistauglichkeit, wobei einige Herausforderungen bereits in dieser Arbeit angeschnitten wurden. Liegt jedoch erst einmal ein praxistaugliches Verfahren vor, dürften vor allem Strafverfolgungsbehörden daran interessiert sein. Es ermöglicht ihnen die Überwachung von Verdächtigen und Straftätern, die ihre Kommunikation mit datenschutzfreundlichen Techniken verschleiern. Der Einsatz von Website-Fingerprinting zur Erstellung von Nutzerprofilen unbescholtener Bürger erscheint hingegen weniger relevant – in einer Zeit, in der die meisten Internet-Nutzer bereit sind, ihre persönlichen Daten und Interessen öffentlich im Internet zur Verfügung zu stellen, gibt es einfachere Methoden zur Erstellung von Nutzungsprofilen als die Analyse des verschlüsselten Datenverkehrs mit Website-Fingerprinting-Verfahren.

10.2. Erweiterungsmöglichkeiten und Ausblick

Verbesserungen des Website-Fingerprinting-Verfahrens

Das vorgestellte Fingerprinting-Verfahren erzielt zwar unter idealen Bedingungen hohe Erkennungsraten, seine Trennschärfe bei Vorliegen unbekannter Testdaten ist jedoch verbesserungswürdig (vgl. Abschnitt 9.2). Vielleicht lässt sich die Trennschärfe jedoch durch das *Hinzuziehen des aggregierten Datenvolumens* erhöhen. Durch einen Vergleich des aggregierten Datenvolumens der prognostizierten Klasse mit dem tatsächlich beobachteten Datenvolumen wäre eine Plausibilitätskontrolle der Klassifizierungsentscheidung möglich. Übersteigt die Abweichung einen Schwellwert, könnten entsprechend der Rangfolge der Wahrscheinlichkeiten $P(c|d)$ weitere Klassen in Erwägung gezogen werden.

Darüber hinaus könnte man zur Erhöhung der Robustheit *zusätzliche Transformationen in den Klassifizierer integrieren*. Insbesondere *Standardisierung*, *Normalisierung* und *Diskretisierung* kommen hier in Frage [WF05, 398]. Während nach der Standardisierung alle

Attributwerte im Intervall $[0;1]$ liegen, werden die Attributwerte bei der Normalisierung so transformiert, dass gilt $\overline{x} = 0$, $\sigma^2 = 1$. Die Diskretisierung verringert die Anzahl der unterschiedlichen Paketgrößen, indem Intervallklassen gebildet werden (z. B. $0 \leq x < 100, 100 \leq x \leq 200, \ldots$).

Der vorgestellte Klassifizierer vernachlässigt die Reihenfolge der Pakete. Man könnte jedoch *zeitlich aufeinanderfolgende Pakete zu einer virtuellen Paketgröße zusammenfassen*: Werden nacheinander die Pakete 1500 und -52 beobachtet, sind folgende Aggregationen denkbar: 1448, 1552 oder –1552. Auch die Bildung von Mengen oder Tupeln ist möglich: $\{-52, 1500\}$ bzw. $(-52, 1500)$ oder $(1500, -52)$. Zusätzlich kann die Anzahl der zu aggregierten Pakete variiert werden. Zur Analyse der Wirkungsweise dieser Aggregationen sind weitere Versuche erforderlich.

Schließlich ist auch die *Untersuchung von anderen Klassifizierern*, z. B. basierend auf der Cosine Similarity oder anderen Ähnlichkeitsmetriken denkbar.

Burst-Proxy

Auf Erweiterungsmöglichkeiten für den Burst-Proxy wurde bereits in Abschnitt 8.6 eingegangen. Zusätzlich sollte untersucht werden, ob die Wirksamkeit des Burst-Proxies beim *Zulassen aktivierter Inhalte* wie erwartet sinkt, und ob sie bei der *Verwendung von Chunk-Bündel-Strategien* erwartungsgemäß ansteigt.

Weitere Messungen

Bei einer weiteren Messung könnten zusätzlich *Formular-Proxies* (z. B. CGI-Proxy[1], PHProxy[2]) untersucht werden, die über HTTPS erreichbar sind. Da diese Systeme kein Traffic-Shaping durchführen, ist zu erwarten, dass der MNB-Klassifizierer bei diesen Systemen hohe Erkennungsraten erzielt.

Ferner ist der von Kiraly et al. beschriebene TFC-Mechanismus [KTB+07] hinsichtlich seines Schutzes vor Website-Fingerprinting zu evaluieren. Für den Test wird ein Linux-basiertes IPsec-VPN mit angepasstem Kernel benötigt. TFC könnte perfekten Schutz vor Website-Fingerprinting bieten.

Zur besseren Einschätzung der Praxistauglichkeit sind die verbleibenden Arbeitshypothesen zu überprüfen. Hierzu müsste man *Messungen mit verschiedenen Browsern und Internetzugängen* durchführen: Bislang gibt es keine Erkenntnisse darüber, ob Website-Fingerabdrücke, die in verschiedenen Umgebungen erzeugt wurdem, zueinander „kompatibel“ sind.

Bei zukünftigen Untersuchungen sollte außerdem die *Änderungsrate der Testseiten gemessen* werden. Hierzu kann man die Testseiten in regelmäßigen Abständen mit *wget* herunterladen und die Dateigrößen der einzelnen Objekte anhand ihrer Cosine Similarity vergleichen. Nur so lässt sich überprüfen, ob besonders gute Erkennungsraten bei einzelnen Messungen darauf zurückzuführen sind, dass sich die Seiten während der Messung weniger geändert haben als bei anderen Messungen.

Eine fundierte Analyse der *Packet-Size-Hypothese* (vgl. Abschnitt 7.2) war auf Grund der inhomogenen Instanzen im Tor-Datensatz nicht möglich. Um störende Einflussfaktoren auszuschließen könnte man die Messung in einer kontrollierten Versuchsumgebung

[1] Homepage: http://jmarshall.com/tools/cgiproxy/
[2] Homepage: http://sourceforge.net/projects/poxy/

mit einem dedizierten Tor-Netzwerk wiederholen. Zur Untersuchung der Packet-Size-Hypothese könnte man dann die *cell size* von Tor durch Anpassung des Quellcodes variieren.

Rechnerische Evaluierung von Gegenmaßnahmen

Ein zentrales Problem bei der Entwicklung von Gegenmaßnahmen ist die vergleichsweise aufwändige empirische Überprüfung ihrer Wirksamkeit. Möglicherweise lässt sich der Zusammenhang zwischen Erkennungsraten und der Entropie der Paketgrößen-Verteilungen ausnutzen, um eine robuste Metrik zu konstruieren, die eine rechnerische Beurteilung der Wirksamkeit ermöglicht.

A

Ergebnisse der t-Tests

Wie in Abschnitt 5.5.3 erläutert kommen t-Tests zum Einsatz, um die Ergebnisse auf statistische Signifikanz zu untersuchen. Für die Irrtumswahrscheinlichkeit gilt $\alpha = 0,05$. Die Ergebnisse der t-Tests werden tabellarisch wiedergegeben. Die Tabellen verwenden eine einheitliche Notation: Ein Plus (+) in einer Zelle bedeutet, dass in der zugehörigen *Spalte* dargestellte Variante signifikant besser ist als die Variante in der jeweiligen *Zeile*; bei einem Minus (–) ist das Verfahren in der Spalte hingegen signifikant schlechter. Ist eine Zelle leer, kann die Nullhypothese nicht abgelehnt werden – die Ergebnisse weisen dann keine signifikanten Unterschiede auf.

Transformation	TF-N	IDF-N	TFIDF-N	ohne-N	TF	IDF	TF-IDF	ohne
TF-N	·	–	–	–	–	–	–	–
IDF-N	+	·			–		–	–
TF-IDF-N	+		·		–		–	–
ohne-N	+			·	–		–	–
TF	+	+	+	+	·	+	+	–
IDF	+				–	·	–	–
TF-IDF	+	+	+	+	–	+	·	–
ohne	+	+	+	+	+	+	+	·

Tabelle A.1.: Signifikanztest für Transformation der Rohdaten bei 4 Trainingsinstanzen: Die Genauigkeit des MNB-Klassifizierers hängt erheblich von der Transformation der Rohdaten ab. Die TF-Transformation erweist sich in Verbindung mit normalisierten Häufigkeitsvektoren (Suffix *N*) als beste Transformation – sie erzielt signifikant höhere Erkennungsraten als alle anderen Verfahren (+ in allen Zeilen).

Anzahl der Trainingsinstanzen	1	2	4	8	16
1	·		+	+	+
2		·		+	+
4	–	–	·		+
8	–	–		·	+
16	–	–	–	–	·

Tabelle A.2.: Der t-Test für abhängige Stichproben ermittelt u. a. einen signifikanten Unterschiede der Erkennungsraten bei einer bzw. vier Trainingsinstanzen.

Zeitdifferenz (Tage)	0.5	1	2	3	4	5	6	17
0,5	·				–	–	–	–
1		·				–	–	–
2			·			–	–	–
3				·		–	–	–
4	+				·	–	–	–
5	+	+	+	+	+	·	–	–
6	+	+	+	+	+	+	·	–
17	–	–	–	–	–	–	–	·

Tabelle A.3.: Der t-Test ermittelt keine signifikanten Unterschiede der Erkennungsraten innerhalb von drei aufeinanderfolgenden Tagen; ab dem vierten Tag verschlechtern sich die Erkennungsraten statistisch signifikant.

B

Datenmodell für Auswertung

Abbildung B.1 skizziert das Datenmodell, das zur Abspeicherung der TCP-Dump-Dateien verwendet wurde. Das Ergebnis eines Versuchs (Download von Webseiten über SSH-Tunnel, OpenVPN, Tor, usw.) wird durch den Entitätstyp *dataset* repräsentiert. In jedem Versuch werden eine Reihe von URLs (Entitätstyp *site)* abgerufen, wobei nicht notwendigerweise in allen Versuchen die gleichen URLs verwendet werden müssen.

Für jede URL entstehen im Laufe des Versuchs mehrere *traces* (die einzelnen tcpdump-Protokolle). Neben dem Zeitpunkt des Abrufs werden für jeden *trace* eine Reihe von aggregierten Informationen hinterlegt. (Anzahl und Größe der empfangenen bzw. gesendeten Pakete). Ein *trace* besteht aus mehreren Paketen (Entität *packet*), deren Reihenfolge durch aufsteigende Werte für das Attribut *id* festgehalten ist. Für jedes Paket werden Informationen zu Übertragungsrichtung (empfangen $\widehat{=}$ 1, gesendet $\widehat{=}$ 0), Paketgröße (bei Paketen in Senderichtung ist die Größe negativ), zeitlichen Abstand zum vorherigen Paket (die inter-arrival time *iat*) sowie Zeitdifferenz zum ersten Paket (Attribut *abstime*) festgehalten.

Aus Performance-Gründen werden die Häufigkeitshistogramme für jeden *trace* vorberechnet. Die Daten für ein Paketgrößenhistrogramm setzen sich aus mehreren Entitäten des Typs *histogram* zusammen: Für jede auftretende Paketgröße wird die Häufigkeit abgespeichert.

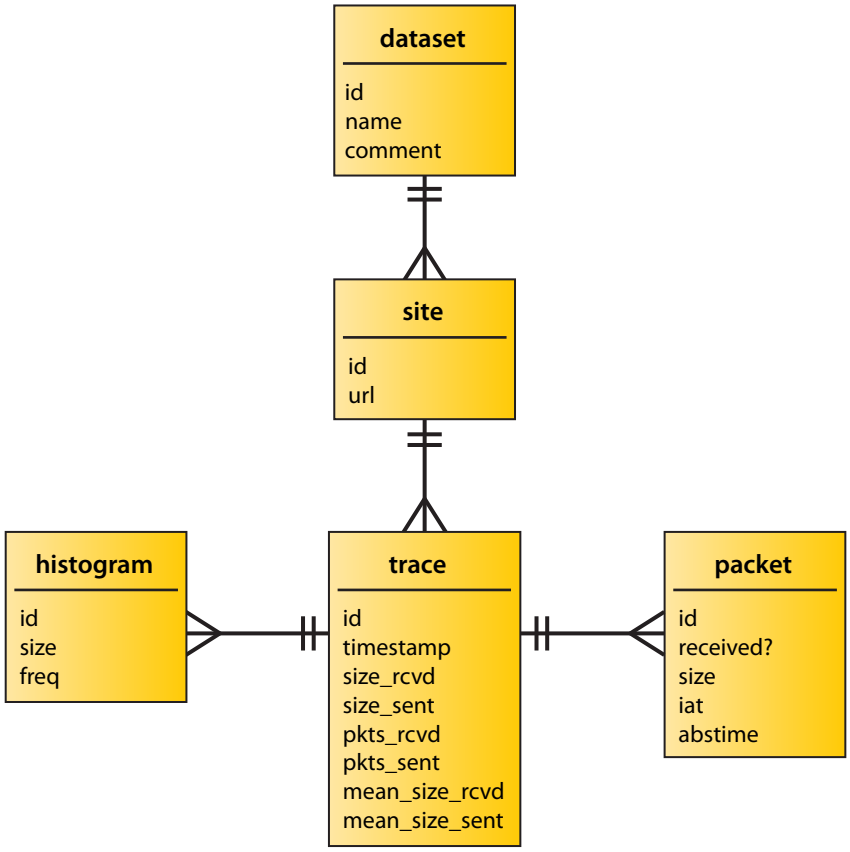


Abbildung B.1.: Datenmodell für die Auswertung von Website-Fingerabdrücken

C

Entwicklungserzeugnisse

Die folgende Aufstellung enthält die entwickelten Software-Komponenten:

- Implementierung eines Website-Fingerprinting-Verfahrens auf Basis von *FilteredClassifier*, *MultinomialNaiveBayes*-Klassifizierer und *StringToWordVector*-Filter in Weka,
- Implementierung des *JaccardSimilarityClassifiers* für Weka,
- Erweiterung des MultinomialNaiveBayes-Klassifizierers von Weka zur Verwendung von Mindestwahrscheinlichkeiten *(MyMultinomialNaiveBayes)*,
- Entwicklung eines Ruby-Skripts *(autofox.rb)* zum automatischen Abruf von Webseiten mit Firefox und Aufzeichnen des Datenverkehrs mit tcpdump,
- MySQL-Datenbank *fingerprints* mit dem aufgezeichneten Datenverkehr für die getesteten datenschutzfreundlichen Systeme,
- Entwicklung von Ruby-Skripten zur Befüllung der MySQL-Datenbank mit Netzwerk-Dumps *(store_dataset_in_db.sh, store_trace_in_db.rb)* sowie zur Erzeugung von ARFF-Dateien *(create_traintest_arff_files*.rb)* zur Analyse mit Weka auf Basis von Stichproben, die aus der Grundgesamtheit aller Instanzen nach einstellbaren Kriterien gezogen werden,
- Entwicklung von Ruby-Skripten zur Berechnung der Cosine Similarity von Termhäufigkeitsvektoren *(compute_cosim_for_histogram_by_url_and_traceid_linalg.rb, compute_intra_and_inter_class_cosim.rb)*, zur Berechnung der Entropie der Paketgrößen-Häufigkeitsverteilung *(compute_entropy_for_csv_file_from_sql.rb)* sowie zur Berechnung des mittleren Informationsgehalts von Multisets *(compute_multiset_information_content.rb)*,
- Erweiterung der OpenSSL-Bibliothek: Unterstützung des zusätzlichen Paddings bei TLS *(openssl-0.9.7m-maxpadding.tar.bz2)*,
- Design und Entwicklung des Prototyps eines Burst-Proxies, der HTTP-Responses gebündelt abruft *(MyProxy)*.

Literaturverzeichnis

[Ben07] Benninghaus, Hans: *Deskriptive Statistik. Eine Einführung für Sozialwissenschaftler*. Vs Verlag, 11. Auflage, September 2007.

[BFK00] Berthold, Oliver, Hannes Federrath, and Marit Köhntopp: *Project "anonymity and unobservability in the internet"*. In *CFP '00: Proceedings of the tenth conference on Computers, freedom and privacy*, pages 57–65, New York, NY, USA, 2000. ACM.

[BFK01] Berthold, Oliver, Hannes Federrath, and Stefan Köpsell: *Web MIXes: a system for anonymous and unobservable Internet access*. In *International workshop on Designing privacy enhancing technologies*, pages 115–129, New York, NY, USA, 2001. Springer-Verlag New York, Inc..

[BIC07] Bonchis, C., C. Izbasa, and G. Ciobanu: *Information Theory over Multisets*. In Gutierrez-Naranjo, M.A., Gh. Paun, A. Romero-Jimenez, and A. Riscos-Nunez (editors): *Proceedings of the Fifth Brainstorming Week on Membrane Computing*, pages 73–86, Sevilla (Spain), January 29th - February 2 2007.

[BLJL05] Bissias, George, Marc Liberatore, David Jensen, and Brian Neil Levine: *Privacy Vulnerabilities in Encrypted HTTP Streams*. In *Proc. Privacy Enhancing Technologies Workshop (PET)*, pages 1–11, May 2005.

[BP04] Borders, Kevin and Atul Prakash: *Web Tap: Detecting Covert Web Traffic*. In *CCS '04: Proceedings of the 11th ACM conference on Computer and communications security*, pages 110–120, New York, NY, USA, 2004. ACM Press.

[BT07] Bernaille, Laurent and Renata Teixeira: *Early Recognition of Encrypted Applications*. In Uhlig, Steve, Konstantina Papagiannaki, and Olivier Bonaventure (editors): *Passive and Active Network Measurement*, volume 4427 of *Lecture Notes in Computer Science*, pages 165–175. Springer, 2007.

[Bun06] Bundesnetzagentur für Elektrizität, Gas, Telekommunikation, Post und Eisenbahnen (Herausgeber): *Technische Richtlinie zur Umsetzung gesetzlicher Maßnahmen zur Überwachung der Telekommunikation – Entwurf. Ausgabe 5.0*. http://www.eco.de/dokumente/20060404-TR-TKUE-Entwurf-5-0.pdf, besucht: 2008-03-18, 2006.

[CA] Cheng, Heyning and Ron Avnur: *Traffic Analysis of SSL Encrypted Web Browsing*. http://www.cs.berkeley.edu/~daw/teaching/cs261-f98/projects/final-reports/ronathan-heyning.ps, visited: 2008-01-10.

[CCW+07] Coull, S.E., M.P. Collins, C.V. Wright, F. Monrose, and M.K. Reiter: *On Web Browsing Privacy in Anonymized NetFlows*. In *Proceedings of the 16th USENIX Security Symposium*, Boston, MA, August 2007.

[CDGS07] Crotti, Manuel, Maurizio Dusi, Francesco Gringoli, and Luca Salgarelli: *Traffic Classification through Simple Statistical Fingerprinting*. SIGCOMM Computer Communication Review, 37(1):5–16, 2007.

[Cha81] Chaum, David: *Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms*. Communications of the ACM, 4(2), February 1981.

[Cho02] Chown, P.: *RFC 3268 Advanced Encryption Standard (AES) Ciphersuites for Transport Layer Security (TLS)*, June 2002.

[CK74] Cerf, V. and R. Kahn: *A Protocol for Packet Network Intercommunication*. IEEE Transactions on Communications [legacy, pre - 1988], 22(5):637–648, 1974.

[DA99] Dierks, T. and C. Allen: *RFC 2246 The TLS Protocol Version 1.0*, January 1999.

[Dan02] Danezis, George: *Traffic Analysis of the HTTP Protocol over TLS*. http://homes.esat.kuleuven.be/~gdanezis/TLSanon.pdf, visited: 2007-10-03, 2002.

[Deu96] Deutsch, P.: *RFC 1952 GZIP file format specification version 4.3*, May 1996.

[DJK+08] Dohmen, Frank, Ulrich Jaeger, Dirk Kurbjuweit, Gunther Latsch, Alexander Neubacher, René Pfister, Christian Reiermann, Barbara Schmid, Jörg Schmitt und Holger Stark: *Der Schatz des BND*. Der Spiegel, (8/2008):20, Februar 2008.

[DMS04] Dingledine, Roger, Nick Mathewson, and Paul Syverson: *Tor: The Second-Generation Onion Router*. In *SSYM'04: Proceedings of the 13th conference on USENIX Security Symposium*, pages 21–21, Berkeley, CA, USA, 2004. USENIX Association.

[Dov05] Dovrolis, Constantinos (editor): *Passive and Active Network Measurement, 6th International Workshop, PAM 2005, Boston, MA, USA, March 31 - April 1, 2005, Proceedings*, volume 3431 of *Lecture Notes in Computer Science*. Springer, 2005.

[DR06] Dierks, T. and E. Rescorla: *RFC4346 The Transport Layer Security (TLS) Protocol Version 1.1*, April 2006.

[DS03] Dingledine, Roger and Paul F. Syverson (editors): *Privacy Enhancing Technologies, Second International Workshop, PET 2002, San Francisco, CA, USA, April 14-15, 2002, Revised Papers*, volume 2482 of *Lecture Notes in Computer Science*. Springer, 2003.

[DSCP02] Díaz, Claudia, Stefaan Seys, Joris Claessens, and Bart Preneel: *Towards Measuring Anonymity*. In Dingledine, Roger and Paul F. Syverson [DS03], pages 54–68.

[EAM06] Erman, Jeffrey, Martin Arlitt, and Anirban Mahanti: *Traffic Classification Using Clustering Algorithms*. In *MineNet '06: Proceedings of the 2006 SIGCOMM workshop on Mining network data*, pages 281–286, New York, NY, USA, 2006. ACM Press.

[EBR03] Early, James P., Carla E. Brodley, and Catherine Rosenberg: *Behavioral Authentication of Server Flows*. In *ACSAC '03: Proceedings of the 19th Annual Computer Security Applications Conference*, page 46, Washington, DC, USA, 2003. IEEE Computer Society.

[FGM+99] Fielding, R., J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee: *RFC 2616 Hypertext Transfer Protocol – HTTP/1.1*, June 1999.

[FKL02] Federrath, Hannes, Stefan Köpsell und Heinrich Langos: *Anonyme und unbeobachtbare Kommunikaton im Internet*. In: Schubert, Sigrid E., Bernd Reusch und Norbert Jesse (Herausgeber): *GI Jahrestagung*, Band 19 der Reihe *LNI*, Seiten 481–487. GI, 2002.

[FPSM91] Frawley, William J., Gregory Piatetsky-Shapiro, and Christopher J. Matheus: *Knowledge Discovery in Databases: An Overview*. In *Knowledge Discovery in Databases*, pages 1–30. AAAI/MIT Press, 1991.

[FPSS96] Fayyad, Usama M., Gregory Piatetsky-Shapiro, and Padhraic Smyth: *From Data Mining to Knowledge Discovery in Databases*. AI Magazine, 17(3):37–54, 1996.

[FS00] Felten, Edward W. and Michael A. Schneider: *Timing Attacks on Web Privacy*. In *CCS '00: Proceedings of the 7th ACM conference on Computer and communications security*, pages 25–32, New York, NY, USA, 2000. ACM Press.

[GRS96] Goldschlag, David M., Michael G. Reed, and Paul F. Syverson: *Hiding Routing Information*. In *Proceedings of the First International Workshop on Information Hiding*, pages 137–150, London, UK, 1996. Springer-Verlag.

[HBM97] Hagedorn, Jürgen, Nicolas Bissantz, and Peter Mertens: *Data Mining (Datenmustererkennung): Stand der Forschung und Entwicklung*. Wirtschaftsinformatik, 39(6):601–612, 1997.

[Hin02] Hintz, Andrew: *Fingerprinting Websites Using Traffic Analysis*. In Dingledine, Roger and Paul F. Syverson [DS03], pages 171–178.

[HSV+05] Huttunen, A., B. Swander, V. Volpe, L. DiBurro, and M. Stenberg: *RFC 3948 UDP Encapsulation of IPsec ESP Packets*, January 2005.

[JL95] John, George H. and Pat Langley: *Estimating Continuous Distributions in Bayesian Classifiers*. In Besnard, Philippe and Steve Hanks (editors): *UAI*, pages 338–345. Morgan Kaufmann, 1995.

[KA98] Kent, S. and R. Atkinson: *RFC 2401 Security Architecture of the Internet Protocol*, November 1998.

[KAA06] Koukis, D., Spyros Antonatos, and Kostas G. Anagnostakis: *On the Privacy Risks of Publishing Anonymized IP Network Traces*. In *Communications and Multimedia Security*, pages 22–32, 2006.

[KM03] Köpsell, Stefan und Andreas Müller: *Bezahlsystem für einen Mixkaskadenbasierten Anonymisierungsdienst*. In: Grimm, Rüdiger, Hubert B. Keller und Kai Rannenberg (Herausgeber): *GI Jahrestagung (Schwerpunkt "Sicherheit – Schutz und Zuverlässigkeit")*, Band 36 der Reihe *LNI*, Seiten 305–316. GI, 2003.

[Koe99] Koehler, Wallace: *An analysis of Web page and Web site constancy and permanence*. Journal of the American Society for Information Science, 50(2):162–180, 1999.

[Koe02] Koehler, Wallace: *Web Page Change and Persistence – A Four-Year Longitudinal Study*. Journal of the American Society for Information Science and Technology, 53(2):162–171, 2002.

[KPF05] Karagiannis, Thomas, Konstantina Papagiannaki, and Michalis Faloutsos: *BLINC: multilevel traffic classification in the dark*. In *SIGCOMM '05: Proceedings of the 2005 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 229–240, New York, NY, USA, 2005. ACM Press.

[KR05] Kaufman, Leonard and Peter J. Rousseeuw: *Finding Groups in Data: An Introduction to Cluster Analysis (Wiley Series in Probability and Statistics)*. Wiley-Interscience, March 2005.

[KTB+07] Kiraly, Csaba, Simone Teofili, Giuseppe Bianchi, Renate Lo Cigno, Matteo Nardelli, and Emanuele Delzeri: *Traffic Flow Confidentiality in IPsec: Protocol and Implementation*. In *Preproceedings Third IFIP/FIDIS Summer School "The Future of Identity in the Information Society"*, August 2007.

[Lee07] Lee, Lorant: *The Physics Factbook: Number of Web Pages*. http://hypertextbook.com/facts/2007/LorantLee.shtml, visited: 2007-12-22, 2007.

[LGL+96] Leech, M., M. Ganis, Y. Lee, R. Kuris, D. Koblas, and L. Jones: *RFC 1928 SOCKS Protocol Version 5*, March 1996.

[LL06] Liberatore, Marc and Brian Neil Levine: *Inferring the Source of Encrypted HTTP Connections*. In *CCS '06: Proceedings of the 13th ACM conference on Computer and communications security*, pages 255–263, New York, NY, USA, 2006. ACM Press.

[MD05] Murdoch, Steven J. and George Danezis: *Low-Cost Traffic Analysis of Tor*. In *SP '05: Proceedings of the 2005 IEEE Symposium on Security and Privacy*, pages 183–195, Washington, DC, USA, 2005. IEEE Computer Society.

[MP05] Moore, Andrew W. and Konstantina Papagiannaki: *Toward the Accurate Identification of Network Applications*. In Dovrolis, Constantinos [Dov05], pages 41–54.

[MR98] Mistry, Shailen and Bhaskaran Raman: *Quantifying Traffic Analysis of Encrypted Web-Browsing*. http://bmrc.berkeley.edu/people/shailen/Classes/SecurityFall98/paper.ps, visited: 2008-02-05, December 1998.

[MRS07] Manning, C. D., P. Raghavan, and H. Schütze: *Introduction to Information Retrieval (preliminary draft printed on November 17, 2007)*. Cambridge University Press, 2007.

[Nat01] National Institute of Standards and Technology (NIST): *Advanced Encryption Standard (AES). Federal Information Processing Standards Publication 197*. http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf, visited: 2007-12-28, November 2001.

[NCO04] Ntoulas, Alexandros, Junghoo Cho, and Christopher Olston: *What's New on the web? The Evolution of the Web from a Search Engine Perspective*. In *WWW '04: Proceedings of the 13th international conference on World Wide Web*, pages 1–12, New York, NY, USA, 2004. ACM.

[Net06] Netcraft Ltd: *The Netcraft Secure Server Survey*. http://news.netcraft.com/SSL-Survey/, visited: 2007-12-28, June 2006.

[Pet05] Petersohn, Helge: *Data Mining. Verfahren, Prozesse, Anwendungsarchitektur*. Oldenbourg Wissenschaftsverlag, Juli 2005.

[PH08] Pfitzmann, Andreas and Marit Hansen: *Anonymity, Unlinkability, Undetectability, Unobservability, Pseudonymity, and Identity Management – A Consolidated Proposal for Terminology*. http://dud.inf.tu-dresden.de/literatur/Anon_Terminology_v0.31.pdf, visited: 2008-02-25, February 2008.

[PPR08] Panchenko, Andriy, Lexi Pimenidis, and Hannes Renner: *Performance Analysis of Anonymous Communication Channels Provided by Tor*. In *Proceedings of the Third International Conference on Availability, Reliability and Security, ARES 2008, Barcelona*, March 2008.

[PPW91] Pfitzmann, Andreas, Birgit Pfitzmann und Michael Waidner: *ISDN-MIXes: Untraceable Communication with Small Bandwidth Overhead*. In: *Kommunikation in Verteilten Systemen, Grundlagen, Anwendungen, Betrieb, GI/ITG-Fachtagung*, Seiten 451–463, London, UK, 1991. Springer-Verlag.

[Ray01] Raymond, Jean François: *Traffic Analysis: Protocols, Attacks, Design Issues, and Open Problems*. In *International workshop on Designing privacy enhancing technologies*, pages 10–29, New York, NY, USA, 2001. Springer-Verlag New York, Inc..

[Rec05] Rechenzentrum, Universität Regensburg: *VPN an der Uni Regensburg*. http://www-soft.uni-regensburg.de/soft/v/vpnclient/doc/vpn.pdf, besucht: 2008-03-01, 2005.

[RR98] Reiter, Michael K. and Aviel D. Rubin: *Crowds: Anonymity for Web Transactions*. ACM Transactions on Information and System Security, 1(1):66–92, June 1998.

[Ryb00] Rybczynski, William: *Information Systems Security User Awareness: Social Engineering and Malware*. http://www.giac.org/certified_professionals/practicals/gsec/0200.php, visited: 2008-03-02, November 2000.

[Sch07] Schmeh, Klaus: *Kryptografie. Verfahren, Protokolle, Infrastrukturen*. dpunkt Verlag, Heidelberg, 3. überarbeitete und erweiterte Auflage, 2007.

[Sha48] Shannon, C. E.: *A Mathematical Theory of Communication*. The Bell System Technical Journal, 27:379–423, 623–656, July, October 1948.

[SKK00] Steinbach, M., G. Karypis, and V. Kumar: *A Comparison of Document Clustering Techniques*. In Proceedings of Workshop on Text Mining, 6th ACM SIGKDD International Conference on Data Mining (KDD'00), pages 109–110, August 20–23 2000.

[SSW+02] Sun, Qixiang, Daniel R. Simon, Yi Min Wang, Wilf Russell, Venkata N. Padmanabhan, and Lili Qiu: *Statistical Identification of Encrypted Web Browsing Traffic*. In *SP '02: Proceedings of the 2002 IEEE Symposium on Security and Privacy*, page 19, Washington, DC, USA, 2002. IEEE Computer Society.

[Str02] Strehl, Alexander: *Relationship-based Clustering and Cluster Ensembles for High-dimensional Data Mining*. PhD thesis, Faculty of the Graduate School of the University of Texas at Austin, May 2002.

[Sun07a] Sun Microsystems: *Java™ Cryptography Architecture (JCA) Reference Guide for Java™ Platform Standard Edition 6 – How the JCA Might Be Used in a SSL/TLS Implementation*. http://java.sun.com/javase/6/docs/technotes/guides/security/crypto/CryptoSpec.html#SSLTLS, visited: 2008-03-10, 2007.

[Sun07b] Sun Microsystems: *Java™ Secure Socket Extension (JSSE) for Java™ Platform Standard Edition 6 Reference Guide*. http://java.sun.com/javase/6/docs/technotes/guides/security/jsse/JSSERefGuide.html, visited: 2008-03-10, 2007.

[SW06] Shmatikov, Vitaly and Ming Hsiu Wang: *Timing analysis in low-latency mix networks: Attacks and defenses*. In Gollmann, Dieter, Jan Meier, and Andrei Sabelfeld (editors): *ESORICS*, volume 4189 of *Lecture Notes in Computer Science*, pages 18–33. Springer, 2006.

[SWT01] Song, Dawn Xiaodong, David Wagner, and Xuqing Tian: *Timing Analysis of Keystrokes and Timing Attacks on SSH*. In *SSYM'01: Proceedings of the 10th conference on USENIX Security Symposium*, pages 25–25, Berkeley, CA, USA, 2001. USENIX Association.

[vR79] Rijsbergen, C. J. van: *Information Retrieval*. Butterworth, 1979.

[WBM99] Warren, Paul, Cornelia Boldyreff, and Malcolm Munro: *The Evolution of Websites*. In *IWPC '99: Proceedings of the 7th International Workshop on Program Comprehension*, page 178, Washington, DC, USA, 1999. IEEE Computer Society.

[Wes08] Westermann, Benedikt: *Entwicklung und Evaluation eines einfachen und effizienten Anonymisierungsverfahrens basierend auf offenen Standards*. Diplomarbeit, Rheinisch-Westfälische Technische Hochschule Aachen, Januar 2008.

[WF05] Witten, Ian H. and Eibe Frank: *Data Mining: Practical Machine Learning Tools and Techniques, Second Edition (Morgan Kaufmann Series in Data Management Systems)*. Morgan Kaufmann, June 2005.

[WHF07] Wendolsky, Rolf, Dominik Herrmann, and Hannes Federrath: *Performance Comparison of Low-Latency Anonymisation Services from a User Perspective*. In Borisov, Nikita and Philippe Golle (editors): *Privacy Enhancing Technologies*, volume 4776 of *Lecture Notes in Computer Science*, pages 233–253. Springer, 2007.

[Wit07] Witt, Bernhard C.: *Datenschutz kompakt und verständlich. Eine praxisorientierte Einführung (Edition kes)*. Vieweg Verlag, 2007.

[WMM06a] Wright, Charles V., Fabian Monrose, and Gerald M. Masson: *On Inferring Application Protocol Behaviors in Encrypted Network Traffic*. Journal of Machine Learning Research, 6:2745–2769, 2006.

[WMM06b] Wright, Charles V., Fabian Monrose, and Gerald M. Masson: *Using Visual Motifs to Classify Encrypted Traffic*. In *VizSEC '06: Proceedings of the 3rd international workshop on Visualization for computer security*, pages 41–50, New York, NY, USA, 2006. ACM Press.

[WS96] Wagner, David and Bruce Schneier: *Analysis of the SSL 3.0 protocol*. In *WOEC'96: Proceedings of the 2nd conference on Proceedings of the Second USENIX Workshop on Electronic Commerce*, pages 29–40, Berkeley, CA, USA, 1996. USENIX Association.

[YL06a] Ylonen, T. and C. Lonvick: *RFC 4251 The Secure Shell (SSH) Protocol Architecture*, January 2006.

[YL06b] Ylonen, T. and C. Lonvick: *RFC 4254 The Secure Shell (SSH) Connection Protocol*, January 2006.

[ZNA05] Zander, Sebastian, Thuy Nguyen, and Grenville J. Armitage: *Self-Learning IP Traffic Classification Based on Statistical Flow Characteristics*. In Dovrolis, Constantinos [Dov05], pages 325–328.

Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbständig angefertigt sowie keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe. Diese Arbeit wurde bisher keiner anderen Prüfungsbehörde vorgelegt.

Regensburg, den 19.03.2008

Dominik Herrmann