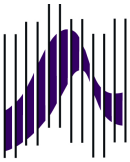
Hochschule
Ravensburg-Weingarten



Hochschule
Ravensburg-Weingarten
Technik | Wirtschaft | Sozialwesen

Kommunikation über einen steganografischen Covert Channel

Bachelorarbeit

Wintersemester 2018/19

im Studiengang Angewandte Informatik

an der Hochschule Ravensburg - Weingarten

von

Maximilian Nestle Matr.-Nr.: 27427

Abgabedatum : 13. Februar 2019

Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit mit dem Titel

Kommunikation über einen steganografischen Covert Channel

selbständig angefertigt, nicht anderweitig zu Prüfungszwecken vorgelegt, keine anderen als die angegebenen Hilfsmittel benutzt und wörtliche sowie sinngemäße Zitate als solche gekennzeichnet habe.

Weingarten, 13. Februar 2019

Maximilian Nestle

_____

Inhaltsverzeichnis

Kurzfassung

Abstract

Danksagung

Vorwort

1 Einleitung

1.1 Motivation

In der heutigen Zeit wird die Datensicherheit immer wichtiger, da immer mehr Daten im Internet preisgegeben werden. Um die Datenübertragung zu sichern wird meistens ein asymmetrisches Verschlüsselungsverfahren verwendet.
Dieses Verfahren bieten zwar ein hohes Maß an Sicherheit, hat aber auch Nachteile, wie zum Beispiel eine Erhöhung der Rechenzeit, die Verwaltung eines Key-Managers und die Bedrohung durch einen „Man-in-the-Middle" Angriff.
Eine mögliche Alternative ist die Steganographie. Dies ist die Kunst, Daten in legitimierten Datenkanälen zu verstecken ohne einen Verschlüsselung anzuwenden.
Steganographie ist zudem sehr unauffällig, da in unverschlüsselten Daten keine sensiblen Daten vermutet werden.
So wäre beispielsweise eine Anwendung denkbar, bei der ein Polizeipräsidium mit ihren verdeckten Ermittlern kommunizieren will. Da die Polizei davon ausgehen muss, dass die Kommunikation abgehört wird, würde einen verschlüsselte Kommunikation zu viel Aufsehen erregen. Zudem kann es sein, dass die Verschlüsselung schon geknackt wurde.
Hier kommt die Steganographie ins Spiel, die es möglich macht einen legitimen und unauffälligen Kanal für die Datenübertagung zu verwenden. So ein Kanal könnte der Stream beim Schauen eines Videos oder die Nachrichten einer Webseite sein.
Die Steganographie bietet gerade deswegen, da sie oft hinter den großen Verschlüsselungsverfahren in Vergessenheit gerät, eine sehr gut Methode um hoch sensible Daten zu versenden.

1.2 Aufgabenstellung und Zielsetzung

Ziel der Bachelorarbeit ist es, bei dem in der Motivation bereits beschriebenen Szenario, der Polizei eine Kommunikationsmöglichkeit zu schaffen. Dabei soll es möglich sein, dass Anweisungen, Treffpunkte aber auch Bilder an den verdeckten Ermittler übertragen werden können. Dabei soll die Kommunikation über ein Netzwerk stattfinden und steganografisch verschlüsselt werden.
Die entstehende Anwendung soll als ein „Proof of Concept" dienen und das Potential der Steganografie veranschaulichen.
Die Daten sollen nicht mit einem mathematischen Verfahren verschlüsselt werden, sondern in ein oder mehreren Protokollen „versteckt" eingebettet und übertragen werden. Dazu soll ein optimales Verfahren zur Dateninfiltration und -exfiltration gefunden werden. Das Verfahren sollte unauffällig, für Dritte schwer zu interpretieren und mit größt möglicher Übertragungsrate senden. Optimal wäre ein ähnliche Sicherheit zu gewährleisten, wie mit einer mathematischen Verschlüsselung.
Ziel ist es außerdem jedes Dateiformat übertragen zu können.

1.3 Aufbau

Als erstes folgt die Einführung in die Grundlagen. Danach beschäftigt sich die Arbeit mit der Findung eines optimalen steganographischen Verfahren, mit dem die Daten übertragen werden sollen. Anschließend wird dieses Verfahren in einer realen Anwendung übertragen. Als letztes wird die Anwendung bewertet und ein Fazit gezogen.

1.4 Eigene Leistung

Es werden steganografische Verfahren bewertet und das Optimum ausfindig gemacht. Aus dem gefundenen Ergebnis wird einen Anwendung als „Proof of Concept" implementiert,

die passend zur Zielsetzung die Datenkommunikation übernimmt. Das Programm wird danach evaluiert und auf mögliche Anwendungsgebiete getestet.

2 Grundlagen

2.1 Datenschutz

Der Datenschutz ist ein Überbegriff für das in Gesetzte festgelegten Recht, dass jede Person über die Preisgabe der personenbezogenen Daten bestimmen kann. Die Bundesbeauftragten für den Datenschutz und die Informationssicherheit (BfDI) definieren den Datenschutz wie folgt:

„Datenschutz garantiert jedem Bürger Schutz vor missbräuchlicher Datenverarbeitung, das Recht auf informationelle Selbstbestimmung und den Schutz der Privatsphäre" [Die18]

Das bedeutet, dass jeder der personenbezogene Daten, ohne Zustimmung des Betreffenden speichert oder weiterverarbeitet vor Gericht angeklagt werden kann. Damit dies nicht passiert haben die meisten Institute die mit personenbezogenen Daten umgehen einen Datenschutzbeauftragten, der die Einhaltung dieser Gesetzte überwacht.

2.2 Datensicherheit

Im Gegensatz zu dem Datenschutz bezieht sich die Datensicherheit nicht nur auf die personenbezogenen Daten, sondern auf alle Daten. Die Aufgabe der Datensicherheit werden durch das CIA-Prinzip beschreiben.
Zur Datensicherheit gehören nach diesem Prinzip alle Maßnahmen, die die **C**onfidentiality, **I**ntegrity und **A**vailability (Vertraulichkeit, Integrität, Verfügbarkeit) gewährleisten. [Nic18]

Eine zusätzliche Aufgabe ist die Sicherstellung der Authentizität. Viele dieser Aufgaben werden mit Hilfe der Kryptographie realisiert und umgesetzt.

2.2.1 Vertraulichkeit

Die Vertraulichkeit ist dann gewährleistet, wenn die Daten nicht von unbefugten Personen eingesehen werden können. Es muss also ein System verwendet werden, bei dem sich befugte Benutzer legitimieren können und unbefugte beim Interpretieren gehindert werden. In den meisten Fällen wird dies durch eine Verschlüsselung (symmetrisch oder asymmetrisch) umgesetzt. Alle legitimierten Benutzter erhalten den Schlüssel. Die Personen ohne Schlüssel können die Informationen nicht entschlüsseln - die Vertraulichkeit ist so garantiert.
Optimal wäre, wenn nicht legitime Benutzer, auch nicht an die verschlüsselten Daten kommen würde.

2.2.2 Integrität

Die Integrität beschäftigt sich damit, dass Daten nicht unbemerkt verändert oder abgefälscht werden. So soll zum Beispiel sichergestellt werden, dass eine Nachricht genau so beim Empfänger ankommt, wie sie abgesendet wurde.
Hierzu können Hash-Funktionen verwendet werden, die beim Verändern der Nachricht einen anderen Wert ergeben würden. Dabei müsste entweder die Hash-Funktion geheim sein oder der Hash-Wert verschlüsselt werden.

2.2.3 Verfügbarkeit

Der Dritte Punkt ist die Verfügbarkeit. Es soll immer sichergestellt wer, dass Daten aber auch Programm immer abrufbar sind. Hierzu gehören Mechanismen zur Vermeidung von DoS (Denial of Service) Angriffen. Diese Angriffe würden beispielsweise einen Server so überfordern, dass dieser keine Dateien mehr ausliefern kann - die Verfügbarkeit ist dann nicht mehr gewährleistet.

2.2.4 Authentizität

Die Authentizität bestätigt, dass Daten von der angegeben Informationsquelle stammen. Es ist ein Identitätsbeweis des Absenders gegenüber dem Empfängers.
Dies kann zum Beispiel mit einer Public-Key Verschlüsselung realisiert werden. So kann der Sender die Nachrichten mit seinem Private-Key verschlüsseln und jeder im Besitz des Public-Keys kann bestätigen, dass die Nachricht genau von dieser Person kommt.

2.3 Kryptographie

Kryptographie bedeutet „wörtlich: Die Lehre vom Geheimen schreiben" [Hel18] und beschäftigt sich mit der mathematischen Verschlüsselung von Informationen. Dabei gibt es zwei große Verschlüsselungsarten - die Symmetrischen und die Asymmetrischen Verschlüsselungen. Bei beiden Verfahren wird durch einen Schlüssel (meistens eine Zahl) und einem Algorithmus aus einer lesbaren Information eine Unlesbare. Um dies wieder Rückgängig zu machen wird ebenfalls ein Schlüssel und ein Algorithmus benötigt.
Eine der wichtigsten Grundprinzipien der Kryptographie wurde bereits im 19. Jahrhundert von A.Kerkhoffs aufgestellt. Eine der wichtigsten Ausagen hierbei ist, dass die Sicherheit einer Verschlüsselung nicht von dem Verschlüsselungsalgorithmus, sondern allein von dem Schlüssel abhängig sein soll. Das heißt, dass ein guter Verschlüsselungsalgorithmus öffentlich gemacht werden kann, ohne die Sicherheit zu gefährden. Ein Beispiel ist der RSA Algorithmus. Dies ist einer der heute verbreitetsten Algorithmen. Der Algorithmus ist für jeden öffentlich zugänglich, dies hat aber keine Auswirkung auf die Sicherheit, da die Sicherheit allein auf der Geheimhaltung des Passwortes basiert.
Dies hat zum Beispiel auch den Vorteil, dass bei einem Personalwechsel nicht der ganze Algorithmus ausgetauscht werden muss, sondern nur das Password.

2.4 Symmetrische Verschlüsselung

Bei der symmetrischen Verschlüsselung wird zum Verschlüsseln und Entschlüsseln der gleiche Schlüssel verwendet.

$E_k(M) = C$
$D_k(C) = M$

[Ert01]

Der Schlüssel k wird dazu verwendet die Nachricht zu ver- und entschlüsseln. Das Problem bei symmetrischen Verschlüsselungen ist die Schlüsselübertragung, die auf jeden Fall geheim stattfinden muss. Bekannte Beispiele sind der DES und AES.

2.5 Asymmetrische Verschlüsselung

Bei einer asymmetrisch Verschlüsselung hat man zum verschlüsseln einen anderen Schlüssel wie zum entschlüsseln. Dieses System wird „Public-Key-Kryptographie" genannt, da es einen öffentlichen ($k1$) und einen privaten Schlüssel ($k2$) gibt. Dabei wird der $k1$ zum Verschlüsseln verwendet und $k2$ zum Entschlüsseln.

$E_{k1}(M) = C$
$D_{k2}(C) = M$

[Ert01]

Dieses System löst das Problem der Schlüsselübergabe, da der öffentliche Schlüssel ohne Bedenke an den Kommunikationspartner übertragen werden kann. Bei einem Möglichen Angriff kann der Angreifer mit dem Schlüssel nichts anfangen, da er mit ihm nicht entschlüsseln kann. Nur der private Schlüssel, der geheim bleibt und nicht versendet wird, kann dann die Entschlüsselte Nachricht dechiffrieren.
Diese Art von Algorithmus kann so auch zur Authentifizierung eingesetzt werden. Bekannte Asymmetrische Verschlüsselungen sind der RSA-Algorithmus, der Algorithmus von Diffi und Hellmann oder der Algorithmus von ElGamal.

2.6 Steganographie

Die Steganographie ist die Kunst vom verborgenem Schreiben. Je nachdem welche Literatur man verwendet wird die Steganographie als Unterpunkt der Kryptographie oder als einen eigene Disziplin gesehen. In dieser Arbeit wird die Steganographie eigenständig betrachtet und als alternative zur Kryptographie gesehen.
Beide, die Kryptographie und die Steganographie, sind Möglichkeiten Informationen geheim und von Dritten ungesehen zu übertragen. Wie bereits oben beschrieben beschäftigt sich die Kryptographie mit dem verschlüsseltem Schreiben. Die Steganographie hingegen benutzt keine Verschlüsselung, sondern versucht die geheime Information in einem unauffälligem oder legitimiertem Informationskanal zu verstecken.

Wie von Peter Purgathofer [Pur10] beschrieben hat die Steganographie eine große Bedeutung in der Geschichte, denn die Menschen waren gerade in Kriegszeiten schon immer auf der Suche nach einem sicher Weg Informationen zu übertragen.
So hat zum Beispiel der griechisch Spion Demaratos Wachstafeln dazu benutzt um Informationen zu verschicken. Nur hat er die nicht ins Wachs geschrieben sondern in das darunterliegende Holz.
Ebenfalls soll Histiaeus, der Tyrann von Milet, seine geheimen Nachrichten auf die Schädel der Sklaven tätowiert haben. Die Haare wuchsen nach und die Nachricht war verborgen.
Es gibt noch viele andere Beispiel Angefangen von unsichtbarer Tinte bis hin zu Morsezeichen in Gemälden, aber vor allem Künstlern wurde oft vorgeworfen, mit Hilfe von Steganographie Geheime Nachrichten zu verbreiten. So wurde zum Beispiel Mozart immer wieder beschuldigt freibeuterische Nachrichten in der „Zauberflöte" versteckt zu haben.

Die Beispiele der Geschichte zeigen deutlich wie die Steganographie funktioniert: Es gibt immer eine unauffällige Trägernachricht (Wachstafel, Sklave, Gemälde, Musikstück...).
In diese Trägernachricht wie die geheime Nachricht versteckt (Unter Wachs oder Haaren, Blickwinkel auf das Gemälde, Notenreihenfolge...)
Um die Nachricht zu entschlüsseln benötigt der Empfänger nur die Information wo sich die Nachricht befindet beziehungsweise wie sie versteckt wird. Das Schema von Gary C. Kessler [Kes15] macht dieses Prinzip sehr anschaulich:

Steganographisches Medium = Geheime Nachricht + Träger Nachricht + Steganografischer Schlüssel

Dabei darf der Steganografische Schlüssel nicht mit dem aus der Kryptographie verwechselt werden. Es handelt sich hier mehr um das Wissen wo und wie die geheime Nachricht verborgen ist.
Dabei bedient sich die Steganographie der „Security by Obscurity" (Sicherheit durch Unwissenheit), was bedeutet, dass die Sicherheit allein davon abhängt, ob das Geheimhaltungsverfahren unbekannt bleibt. Übrigens gehören kryptographische Verfahren die nicht unter Kerkhoffs Prinzip fallen auch zu „Security by Obscurity". Will man also ein solches System sicherer machen muss man dafür sorgen, dass das Verfahren so abwegig beziehungsweise obskur gestalten wird, sodass nie jemand auf die Idee kommt nach einer geheimen Nachricht zu suchen

Die Steganographie hat in der Geschichte eine relativ einfache aber sichere Methode geboten Nachrichten zu übertragen. Aber auch heute im Internet sind wir nahezu immer von Datenkanälen umgeben, die sich für die steganographische Datenübertragung eignen. Der Vorteil hierbei ist, dass meistens unter den ganzen kryptographisch verschlüsselten Datenpaketen die Steganographie vergessen wird.

2.7 Internet Protokolle

In den folgenden Kapiteln soll kurz das OSI-Schichtenmodell, auf welches das heutige Internet aufbaut, erklärt werden. Dabei repräsentiert jede Schicht eine Protokoll, das für die Kommunikation im Internet nötig ist.
Es werden nur die Protokolle betrachtet, die für dieses Projekt relevant sind.

2.7.1 Sicherungsschicht/Data Link Layer (Schicht 2)

Diese Schicht beinhaltet Protokolle, welche einen weitestgehend fehlerfreie Datenübertragung garantieren sollen. Außerdem wird der Zugriff auf das Übertragungsmedium ermöglicht. [Wik18]

Ethernet

Beim Ethernet-Protokoll werden die Daten in Pakete zerteilt. Diese können dann zwischen den Geräten im Netzwerk verschickt werden. Dabei ist das Ethernet-Protokoll immer nur im jeweiligen Netzwerksegment gültig. [Zis13] Die Adressierung wird mit Hilfe der MAC-Adressen realisiert. Diese Adresse ist einmalig und wird jedem netzwerkfähigem Gerät vom Hersteller zugeordnet.

7 Byte	1 Byte	6 Byte	6 Byte	4 Byte	2 Byte	bis 1500 Byte	max. 42 Byte	4 Byte
Präam-bel,	SFD	MAC-Adresse Ziel	MAC-Adresse Quelle	VLAN-Tag	Typ	Nutzdaten	PAD	FCS

Bild 2.1: Erweiterter Ethernet-Frame nach IEEE 802.1Q [Zis13]

2.7.2 Vermittlungsschicht/Network Layer (Schicht 3)

Die Protokolle dieser Schicht werden verwendet, um über Netzwerkgrenzen hinaus Nachrichten zu versenden. [Zis13] Dieses Protokoll liegt innerhalb der Nutzdaten des Ethernet-Pakets.

IPv4

Zur Adressierung werden IPv4 Adressen verwendet, die 32 bit (4 Byte) lang sind. Vergeben werden die Adressen von der IANA (Internet Assigned Numbers Authority). Jeder der aus dem Internet erreichbar sein will, muss sich bei der IANA oder einer untergeordneten

Organisation eine IP-Adresse oder Adressbereich geben lassen.

Das Internet Protokoll wird wie in folgender Abbildung gezeigt in das Ethernet Paket eingebettet.

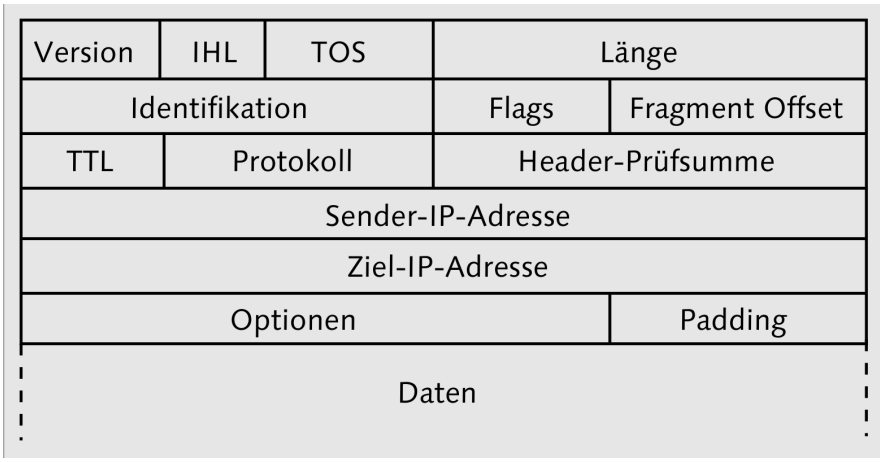


Bild 2.2: IPv4 Header [Zis13]

IPv6

Da die IPv4 Adressen langsam knapp werden, wurde das IPv6 Protokoll erstellt, welches über Adressen mit 6 Byte Länge verfügt. Dies bedeutet, dass deutlich mehr Adressen erstellt und vergeben werden können. Die Funktion ist aber mehr oder weniger die gleiche.

Der Header des IPv6 Protokolls ist in folgender Abbildung gezeigt.

2.7.3 Transportschicht/ Transport Layer (Schicht 4)

2.7.4 Kommunikationsschicht/Session Layer (Schicht 5)

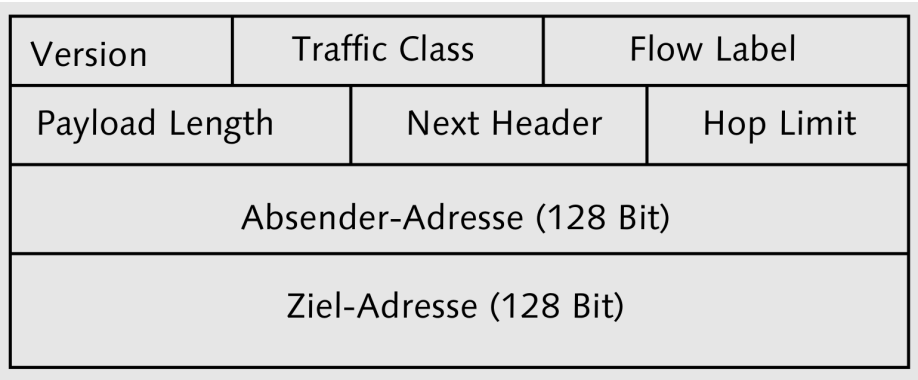


Bild 2.3: IPv6 Header [Zis13]

3 Was ist ein Covert-Channel?

3.1 Definition

Covert-Channels sind Netzwerk Kanäle, die nicht offen praktiziert, erklärt, engagiert, angesammelt oder gezeigt werden. [Gol03]

Hierbei kann ein Kanal eine beliebige Methode sein Informationen zischen zwei Geräten über ein Netzwerk auszutauschen. Bei Covert-Channels wird sich die größtmöglich Mühe gegeben, diesen Informationskanal vor Dritten unbemerkt zu halten.
Bei steganografischen Covert-Channels werden Methoden aus der Steganografie verwendet um diesen Kanal zu verbergen.

In dieser Arbeit handelt es sich bei Covert-Channel immer um Kanäle, die mit einem steganografischem Verfahren verschlüsselt werden.

3.2 Was zeichnend einen guten Covert-Channel aus?

Bei den weit verbreiteten mathematischen Verschlüsselungen muss sich meistens keine sorgen gemachen werden, ob der Datenaustausch von Dritten entdeckt werden kann, da hier die Daten ohne richtigen Key nutzlos sind.
Bei den Covert-Chanels ist dies problematischer, da ein entdeckter Kanal in der Regel direkt interpretiert werden kann. Ein Covert-Channel lebt, wie der Name auch schon verrät, davon wie gut dieser versteckt ist.
Dabei besitzt die Steganografie einen großen Vorteil: Ist das Verfahren zum Verstecken der Daten nicht bekannt, so ist es nahezu unmöglich den Covert-Channel zu finden, da

nicht klar ist wo und nach was gesucht werden muss (Security by Obscurity).
Ist hingegen klar, um welches Verfahren es sich handelt und besteht die Vermutung, dass eine Kommunikation über einen versteckten Kanal stattfindet so kommt man leicht an die Informationen.
Um einen Covert-Channel zu Bewerten müssen folgende Aspekte betrachtet werden:

Der erste ist die Fähigkeit, wie einfach sich ein Kanal verstecken lässt. Hier fließt die allgemeine Unauffälligkeit des Covert-Channels ein, aber auch die Eigenschaften des bereits herrschenden Netzwerkverkehrs, in den der Channel eingebettet werden soll.
Der zweite Aspekt ist die Unbekanntheit des Verfahrens, sodass nicht nach einem möglichen versteckten Kanal gesucht werden kann. Hier kann einen Methode zur individuellen Gestaltung des Channels eine Verbesserung bringen.
Natürlich muss auch die Datenübertragungsrate betrachtet. Diese ist meistens sehr gering aber hier gibt es auch große Unterschiede zwischen den einzelnen Verfahren.
Die Integrität der Daten müssen die Channels ebenfalls gewährleisten.

3.3 Aktive und Passive Covert Channels

3.4 Wozu sind Covert Channels nicht geeignet?

Hohe Datenübertragung Authenzität

4 Mögliche steganografische Covert Channel

Im folgenden Kapitel werden verschiedene, bereits existierende steganografische Covert-Channel betrachtet, die für das Erreichen der Zielsetzung in Frage kommen.

4.1 Klassische textbasierende Verfahren

Botschaften in Texten teilweise einzubetten ist mit der Steganographie möglich.
Gängig unter Textmanipulatoren ist die gezielte Wahl des ersten Buchstaben des Wortes, wobei die Aneinanderreihung dieser Buchstaben ein neues Wort ergibt.
Bei einem wissenschaftlich erarbeiteten Rückblick treten einige, nicht zu unterschätzende, Sicherheitslücken auf, wenn diese Art der Verschlüsselung angewendet wird.

(Im oberen Text ist zur Veranschaulichung eine Nachricht an einen potentiellen Prüfer eingebettet)

Eine weitere Methode ist die Satzzeichen zu verwenden um Informationen zu kodieren. So kann zum Beispiel ein Punkt 00, ein Komma 01, ein Fragezeichen 10 und ein Ausrufezeichen 11 bedeuten. [LC17]

Durch diese Verfahren lassen sich Covert-Channel konstruieren indem ein solcher Text beispielsweise per E-Mail versendet wird.

4.2 Zeitabhängige Covert-Chanel

Bei zeitabhängigen Covert-Channel werden die Daten so versendet, dass der Absendezeitpunkt oder der Abstand zwischen den Paketen die Information enthält. Dabei ähnelt dieses Verfahren dem in der Vergangenheit oft eingesetzten Morse Code. Hingegen beschränkt man sich bei diesen Covert-Channel meistens auf Binärdaten.
In der Abbildung unten sieht man einen beispielhaften Aufbau dieses versteckten Kanals.

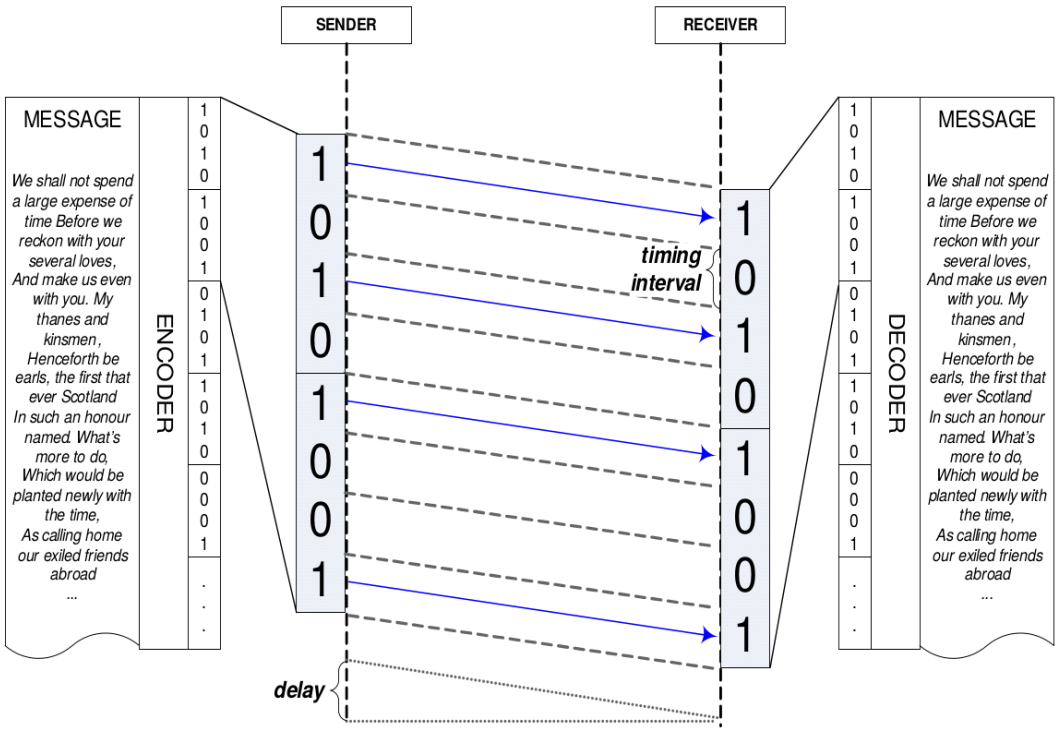


Bild 4.1: Kommunikation über einen Zeitabhängigen Kanal [CBS04]

Bei dem in Abbildung 4.1 dargestellte Covert Channel wird mit einem festen Zeitintervall gearbeitet. Dieses Zeitintervall muss sowohl dem Sender sowie dem Empfänger bekannt sein und muss vor dem Senden festgelegt werden. Wichtig ist ebenfalls die Synchronisation von Sender und Empfänger. Dazu muss entweder ein fixer Startzeitpunkt gewählt werden, oder ein bestimmtes Paket als Start der Übertragung festgelegt werden. [CBS04]

Sind dieses Parameter mit beiden Seiten abgeklärt, so kann die Datenübertragung starten.

Dabei wird einen Datenpaket innerhalb des Zeitintervalls als 1 interpretiert, falls kein Paket gesendet wird als 0. So lassen sich beliebige Daten betragen. [CBS04]

Da die Datenübertragung sehr stark von der Netzwerkgeschwindigkeit abhängig ist, muss man zusätzlich ein Verfahren zur Sicherstellung der Integrität implementieren.
Das zusätzlich Versenden des Hashwertes, der über einen bestimmten Anteil der Nachricht gebildet wird, könnte die Integrität garantieren.
Eine andere Methode bei der die Integrität jedoch nicht vollständig garantiert, aber simpler umzusetzen ist, ist die Verwendung eines Paritätsbit. [CBS04]

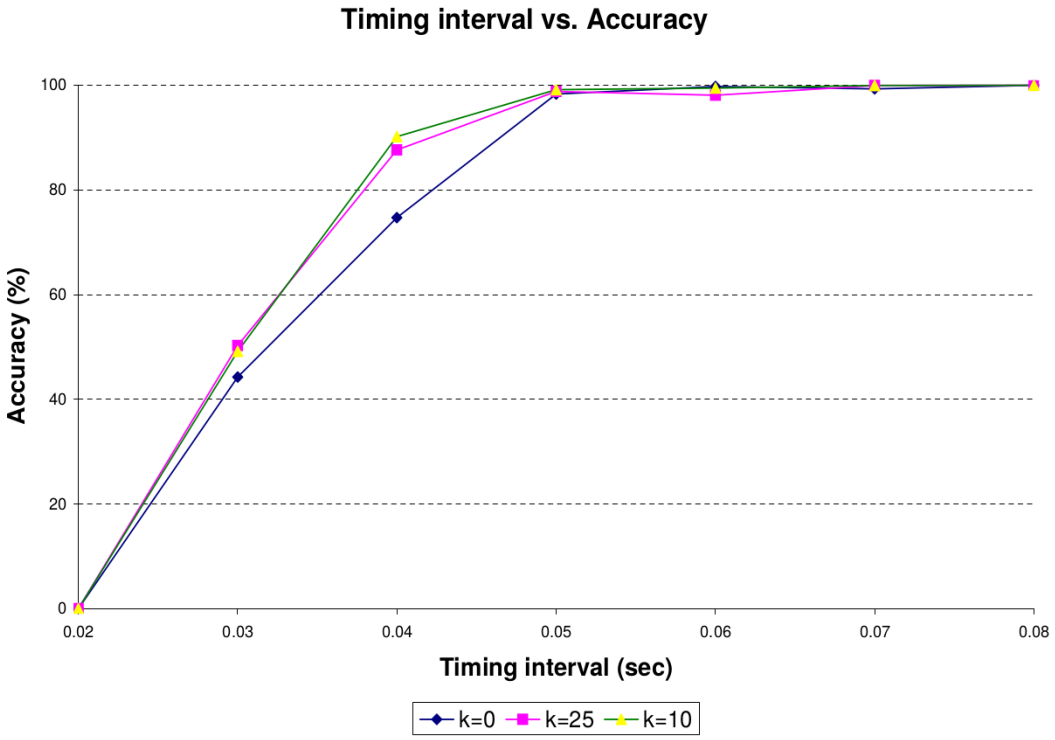


Bild 4.2: Zusammenhang zwischen Zeitintervall und Genauigkeit [CBS04]

Die Datenübertragung über diesen Channel ist mit einem Bit pro Paket relativ gering. Die Datenübertragung ist hier aber auch direkt vom verwendeten Zeitintervall abhängig. Deshalb gilt es einen möglichst kleinen Zeitintervall zu wählen und diesen variabel, optimal an die herrschenden Netzwerkbedingungen anzupassen.
Die Problematik ist hier, dass mit der Reduzierung des Netzwerkhinteralls die Fehleranfälligkeit ebenfalls zunimmt.

Abbildung 4.2 zeigt, wie das gewählte Zeitintervall und die Genauigkeit zusammenhängen. So lässt sich ab einer langsamen Datenübertragung mit einem Intervall von 0.05 Sekunden eine Genauigkeit von ca. 100 Prozent garantieren. Wobei diese Zahlen mit Vorsicht zu genießen sind da sie sehr stark vom jeweiligen Netzwerk abhängig sind. Der Ersteller dieser Grafik [CBS04] hat ebenfalls einen Wert k in seine Implementierung eingebaut, der die Länge einer Pause angibt, die bei einer Übertragungsverzögerung eingelegt werden kann. Diese Verbessert zwar die Genauigkeit der Datenübertragung die Geschwindigkeit wird aber erheblich reduziert.

Bei dieser Art von Covert-Channel kann jedes beliebige Protokoll eingesetzt werden. Es bietet sich aber an die Protokolle auf den umliegenden Netzwerkverkehr anzupassen um ihn so unauffällig wie möglich zu machen.

4.3 Storage Channel

Bei Storage Channels benutzt man Speicherattribute [Wen12b] im Protokollheadern, um unbemerkt Daten zu übertragen. Hier wird beim Internet Protokoll angefangen, da in dieser Arbeit über Netzwerkgrenzen hinaus kommuniziert werden soll.

4.3.1 IPv4

Der IPv4 Header bietet einige Möglichkeiten Daten in Speicherattribute zu verstecken. (Header in Kapitel Grundlagen)

Type of Service
Die letzten beiden Bits diese Feldes sind unbenutzt und können so zur Datenübertragung verwendet werden. (2 Bits)

Identification
Bietet 16 Bits die theoretisch frei wählbar sind. (16 Bits)

Reserved Flag
Das erste Bit der Flags ist für zukünftige Benutzung reserviert und ist derzeit noch unbenutzt. (1 Bit)

Fragment Offset
Der Fragment Offset wird dazu verwendet, um Pakete nach einer Fragmentierung wieder zusammenzusetzen. Geht man davon aus, dass die Paket Fragmente sich frei konfigurieren lassen bietet sich die Chance 13 Bit zu verwenden. Dies ist aber fast unmöglich zu realisieren. ($<$ 13 Bit)

Time to Live
Dieses 8 Bit Feld lässt sich frei wählen. Jedoch muss man bei der Benutzung wissen, wie viele Netzwerkstationen, die dieses Feld herunterzählen, auf dem Weg liegen. Auch ein zu kleiner Wert kann dazu führen, dass die Nachricht nicht ankommt. ($<$ 8 Bit)

Total Length
Die Gesamtlänge des Pakets lässt sich auch manipulieren. Diese Länge wird mit einem 16 Bit Wert angegeben. Jedoch ist dieser Wert durch die Mindestgröße eingeschränkt. Durch Fragmentierung des Pakets oder das Hinzufügen von Optionen ändert sich dieser Wert. ($<$ 16 Bit)

Options
Dem Ip Header können Optionen hinzugefügt werden, in die sich ebenfalls Daten einbetten lassen.

Padding
Die durch das IHL Feld angegebene Headerlänge muss ein vielfaches von 4 Byte erreicht werden. Durch die Verwendung von Options wird diese Länge nicht immer erreicht und wird deshalb mit Padding aufgefüllt. Dieses Padding lässt sich theoretisch auch umwandeln und zur Datenübertragung verwenden.

4.3.2 IPv6

Bei IPv6 kann man in der Regel die äquivalenten Speicherattribute verwenden, wie bei IPv4. Hier unterscheidet sich meistens nur die Namensgebung. [Wen12b]

4.3.3 TCP

Im TCP Protokoll bieten sich ebenfalls Möglichkeiten Daten unbemerkt zu transportieren. So kann der Source Port zur Codierung verwendet werden. Ebenfalls möglich ist die Benutzung des optionalen TCP Timestamps, bei dem zum Beipiel die letzten Bits manipuliert werden. [Wen12b]

Ein weiter Möglichkeit ist das Manipulieren der Sequence Number. [Wen12b] Diese Nummer gibt die Reihenfolge der Datenpakete an. Dabei wird sie am Anfang der Datenübertragung vom Sender errechnet und dann alle 4 Mikrosekunden um eins hochgezählt. Sollte diese 32 Bit Nummer überlaufen so wird sie wieder auf null zurückgesetzt. So wird Sichergestellt, dass jedes Paket eine einzigartige Sequence Number bekommt [Inf81]
Um diese Nummer nun zu Verändern muss eine Übersetzungsschicht eingebaut werden die, die übertragenen Geheimdaten abfängt und wieder mit der Richtigen Sequence Number ersetzt. [Wen12b]

4.3.4 Traffic Normalizers

Storage Channels haben einen großen Schwachpunkt, der ihnen das Leben schwer macht. Traffic Normalizer schreiben die oben beschriebenen Headerattribute gezielt um oder verwerfen diese wenn auffällige Werte gesetzt sind.
Ein Traffic Normalizer auf IP Ebene könnte zum Beispiel das TTL Feld manipulieren, die Flags „Don't Fragment" und „Reserved" auf 0 senzen, die Options löschen oder Pakete bei denen das IHL Feld größer als 5 ist verwerfen. [Wen12a]
Außerdem denkbar ist, dass Padding und die ungenutzten Bits des Type of Service Feldes auf 0 gesetzt werden.
Ein solches System, in ähnlicher Weise auf alle Netzwerkschichten angewendet, ist eine sehr effektive Methode um gegen Storage Channels vorzugehen.

4.4 Benutzung verschiedener Protokolle

Covert-Channel können durch die Verwendung verschiedener Protokolle realisiert werden. Hier kann man zwischen Protocol Hopping Covert Channels und Protocol Channels unterscheiden. [Wen12b]

4.4.1 Protocol Channels

Bei Protocol Channel werden die Daten mit Hilfe mehrere Protokolle kodiert. Eine mögliche Kodierung könnte Folgende sein:

HTTP -> 00
DNS -> 01
ICMP -> 10
POP -> 11
[Wen12b]

So ist man in der Lage binäre Daten mit vier Protokolle zu versenden. In Abbildung 4.3 ist dieses Prinzip veranschaulicht.

Die Wahl der Protokolle ist hier abhängig von den im Netzwerk verwendeten Protokollen. Natürlich funktioniert dieses Prinzip auch mit zwei Protokollen. Denkbar wäre hier die Verwendung von IPv4 und IPv6 da diese Protokolle unter Umständen unterschiedliche Wege durchs Internet nehmen und so noch unauffälliger werden. Wie auch bei den zeitabhängigen Covert-Channel beträgt hier die Übertragungsrate 1 oder 2 Bit pro Paket. Die Übertragungsgeschwindigkeit wir durch die Netzwerkgeschwindigkeit eingeschränkt.

Eine denkbare Unterart dieses Kanals ist die Verwendung verschiedener Options im Header wodurch die Codierung realisiert wird.

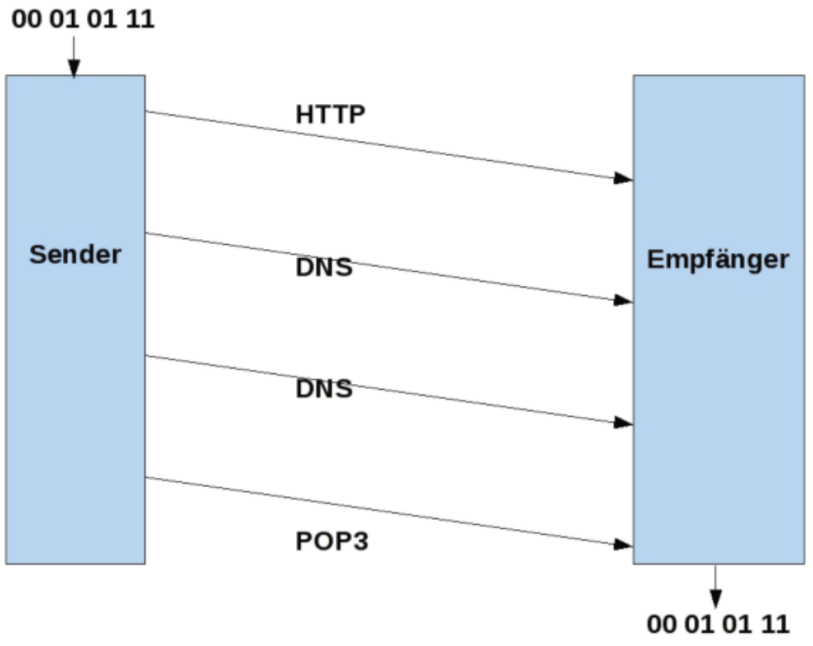


Bild 4.3: Protokol Channel [Wen12b]

4.4.2 Protocol Hopping Covert Channels

Dieser Kanal ist eine Mischung des Protocl-Channels und des Storage Channel. Dabei werden verschiedene Storage-Channels zu einem zusammengefasst und abwechseln Daten übertragen.

In Abbildung 4.4 wird dargestellt wie dies realisiert werden kann. Die Binärdaten werden hier über zufällig gewählte Storage-Chanel übertragen. Dadurch wird bewirkt, dass der Kanal unauffälliger wird, da sich ein reales Netzwerk simulieren lässt.

Im Beispiel unten werden jeweils 4 Bit an den Storage Channel übergeben und an den Empfänger weitergeleitete.

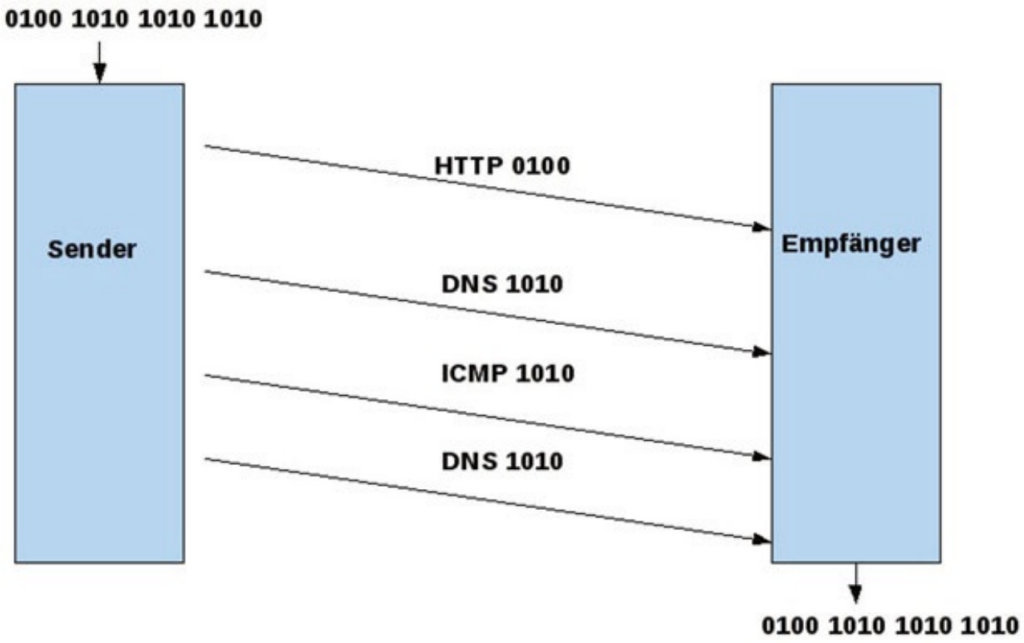


Bild 4.4: Protokol Hoping Covert Channel [Wen12b]

4.5 Verwendung der Nutzdatengröße

Die Größe des Pakets lässt sich über die Versendung unterschiedlicher Daten einfach manipulieren. Daraus ergeben sich etliche Varianten um einen Covert Channel zu Erschaffen. Beispielsweise kann zwischen großen/kleinen Paketen oder Gerade/Ungerade Datenanzahl unterschieden werden und so binär Daten übertragen.
Aber auch direkt in die Größe der Nutzdaten können Informationen versteckt werden: Will man 8 Bit pro Paket übertragen, benötigt man die Werte von 0 bis 255. Nun muss die Größe der Nutzdaten so angepasst werden, dass sie jeweils den zu übertragenden Werten entspricht. Da Nutzdaten von null oder einem Byte relativ selten sind, empfiehlt sich die Verwendung eines statischen oder flexiblen Offset der addiert wird um die Paketgröße anzuheben.
Dieser Kanal lässt sich auf alle Protokolle, die zur Datenübertragung fähig sind, anwenden.

4.6 Verstecken der Informationen in den Nutzdaten

4.6.1 Verwenung von Bilddateien

RGB Basierende Formate

Bildformate die auf die RGB Formate basieren sind zum Beispiel BMP (Windows Bitmap) oder auch GIF (Graphics Interchange Format). Diese Formate geben die Pixelfarbe basierend auf dem **R**ot-, **G**rün- und **B**lauwert. In diesen Werten können Daten versteckt werden. [KP00] So können beispielsweise die letzten beiden Bits manipuliert werden. Diese Veränderung ist für das menschliche Auge nicht zu erkennen, da die letzten Bits kaum eine Auswirkung auf die Höhe der Zahl haben.
Bei einer Farbtiefe von 24 Bit beträgt die maximale Änderung der Farbwerte nur 3 Farbstufen von insgesamt 256 möglichen.

00000000 (0) -> 00000011 (3)
11111111 (255) -> 11111100 (252)

Auf diese Weise lassen sich 6 Bit pro Pixel übertragen. In einem unkomprimierten HD Bild mit einer Auflösung von 1280x720 Pixeln (ca. 22 MByte) lassen sich 0,6912 Mbyte Daten übertragen.

In Abbildung **??** wird dieses System veranschaulicht:

Bildformate mit Alpha Kanal

Bildformate wie PNG (Portable Network Graphics) können zusätzlich zu den RGB Kanälen auch auch einen Alpha Kanal besitzen. Dies ist ein zusätzlicher Wert der die Transparenz des jeweiligen Pixel angibt. Bei einem Wert von 0 ist der Pixel „unsichtbar" und bei 255 ist der Pixel komplett sichtbar. Hier könnte man die Daten wie oben in den letzten beiden Werten speichern.
Da die Transparenz keine direkte Auswirkung auf die Farbe der Pixel hat, kann man jedoch auch mehr Daten speichern.

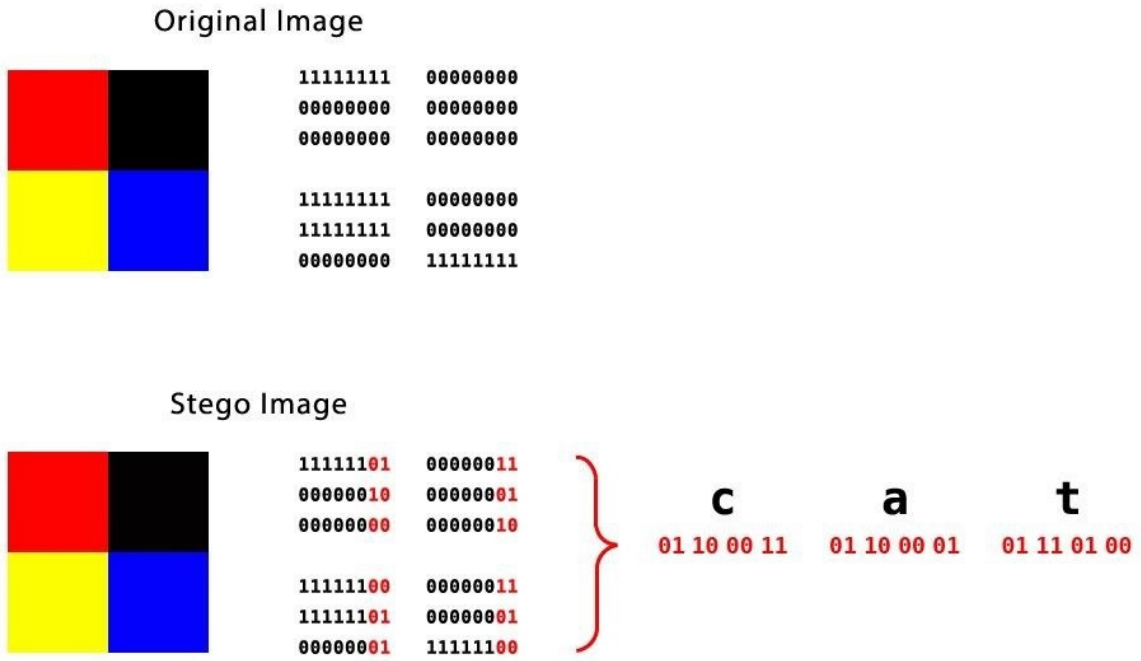


Bild 4.5: Codierung in den 2 letzten Bits [Use18]

Verwendung von „Shamir's Secret Sharing" Methode

Die Shamir's Secret Sharing Methode ist ein mathematisches Verfahren, das es möglich macht, ein Geheimnis auf mehrere Instanzen aufzuteilen. Dabei sind das Geheimnis und die dabei einstehenden „Shares" Integer-Werte.
Die einzelnen Instanzen können dabei keine Rückschlüsse auf das Geheimnis schließen. Allein wenn man den Großteil der „Shares" besitzt kann man das Geheimnis rekonstruieren.

Der folgende Abschnitt basiert auf dieser Literatur: [LT10]

Verschlüsseln:
Verwendet wird ein Geheimnis d, eine Anzahl von n Instanzen und eine Schwelle $k < n$.

1. Es wird eine Primzahl p zufällig gewählt.

2. Es werden k-1 Werte $c_1, c_2...., c_{k-1}$ mit den Werten zwischen 0 und p-1 vergeben.

3. Für $x_1, x_2.., x_n$ jeweils eine eindeutige reale Zahl wählen.

4. Mit Hilfe einer Polynomgleichung mit dem Grad k-1 werden nun n Gleichungen und somit auch Werte berechnet. $i = 1, 2, ..., n$

$$F(x_i) = (d + c_1 x_i + c_2 x_i^2 + ... + c_{k-1} x_i^{k-1}) mod(p)$$

5. Nun können die n Shares erstellt werden. Diese sind wie folgt Aufgebaut $(x_i, F(x_i))$

Das Geheimnis ist nun in einer Funktion „abgespeichert". Die Shares sind Punkte, die auf dieser Funktion liegen. Damit die Funktion wieder rekonstruiert werden kann, müssen mindestens k der n Shares vorhanden.

Entschlüsseln:

Um die Nachrichten zu entschlüsseln müssen k Shares in die Formel oben eingesetzt werden.
Das hierdurch entstandene Gleichungssystem mit den Unbekannten d und c_1 bis c_{k-1}, kann zum Beispiel mit dem Gaußsches Eliminationsverfahren oder durch die Lagrangesche Interpolationsformel gelöst werden.

Dieses Verfahren kann man dazu verwenden, um Nachrichten unauffällig in den alpha Kanal eines PNG Bildes zu integrieren.
Die zu übertragende Nachricht M wird hierzu in Segmente von t Bit, mit $t = 3$ unterteilt. Bei der Umwandlung der Segmente in Dezimalzahlen entsteht so ein neues Array $M' = d_1, d_2...$ bei dem die Werte zwischen 0 und 7 liegen.

Im Gegensatz zu dem original Algorithmus greift man hier zusätzlich auf die Werte c_1, c_2 und c_3 zurück um hier Ebenfalls ein „Geheimnis" einzubetten. Zusammen mit d können nun k Werte integriert werden.

So ergibt sich:

$d = m_1, c_1 = m_2, c_2 = m_3, c_3 = m_4$

Folgende Werte werden definiert:

$p = 11$
$x_1 = 1, x_2 = 2, x_3 = 3, x_4 = 4$

x kann hier eine beliebige Zahl annehmen und so als eine Art Verschlüsselung dienen. Fraglich ist, ob es sich dann noch um reine Steganografie handelt.

Die Werte von q_1 bis q_4 erben sich dann durch Einsetzten in sie Polynomgleichung.

$q_1 = F(x_1) = (m_1 + m_2x_1 + m_3{x_1}^2 + m_4{x_1}^3)mod(p)$
$q_2 = F(x_2) = (m_1 + m_2x_2 + m_3{x_2}^2 + m_4{x_2}^3)mod(p)$
$q_3 = F(x_3) = (m_1 + m_2x_3 + m_3{x_3}^2 + m_4{x_3}^3)mod(p)$
$q_4 = F(x_4) = (m_1 + m_2x_4 + m_3{x_4}^2 + m_4{x_4}^3)mod(p)$

Die Werte q_1 bis q_4 können nun beispielsweise in den alpha Kanal eines PNG Bildes eingefügt werden. Durch die modulo Funktion entstehen so entstehen Werte zwischen 0 und 10. Damit das Bild eine möglichst niedrige Transparenz bekommt muss der alpha Wert so groß wie möglich sein. Deshalb wird zu q jeweils der Wert 245 addiert, um so nah wie möglich an 255 zu kommen.
Die neu entstandenen Werte q'_1 bis q'_4 kann man nun in den alpha Kanal einfügen und über eine sensible Datenverbindung übertragen.

Zum Entschlüsseln muss man nun wieder 245 von den Werten des alpha Chanel subtrahieren. Durch das Erstellen und Lösen des Gleichungssystems, mit den oben gezeigten Formeln, können die Wert m_1 bis m_4 wieder berechnet werden.

4.6.2 Verwenung von PDF Dateien

Auch in PDF Dateien können Daten versteckt eingebettet werden. Das PDF Format setzt sich aus einer Reihe von Befehlen zusammen in der die Formatierung der Seite angegeben wird. So können Elemente zur Positionierung von Texten eingesetzt werden

um Informationen einzubetten. [ZCC07] Open Source Programme machen es für jeden möglich auf diese Weise Daten zu verstecken.

5 Bewertung der Covert Channel

In diesem Kapitel wird sich damit beschäftigt, welcher der im vorhergehenden Kapitel vorgestellten Covert Channel für die Problemstellung geeignet ist. Es soll nun der optimale Kanal gefunden werden, der das Problem der Kommunikation zwischen Polizeipräsidium und Informant lösen kann.

Betrachtet man die **Textbasierenden Verfahren** etwas genauer wird schnell klar, dass diese nicht für größere Datenmengen geeignet sind und sich auch sehr schwierig als Algorithmen darstellen lassen.

Die **Zeitabhängigen Verfahren** sind sehr unauffällig da die Datenpakete nicht direkt manipuliert werden müssen. Es muss sich jedoch um die Integrität der Daten gekümmert werden, da der Kanal sehr stark von den Netzwerkbedingungen abhängt. Die Übertragungsgeschwindigkeit ist mit einem Bit pro Paket ist sehr gering, jedoch lässt sich dieser Kanal sehr gut als passiver Covert-Channel realisieren.

Storage Channel sind relativ auffällig, vor allem wenn die Pakete genauer angeschaut werden. Zudem gibt es das Problem der Netzwerk Normalisierung, die den Storage Channel stark einschränken würde. Die Methode bei der die TCP Sequence Number manipuliert wird ist hingegen für dieses Projekt denkbar, da sie nicht durch die Normalisierung verändert werden kann und pro Paket 32 Bit übertragen kann. Bei einer sehr genaueren Analyse kann hier aber ebenfalls auffallen, dass die Nummern nicht in der richtigen Reihenfolge versendet werden.

Protocol Channel sind im Gegensatz zu zeitabhängigen Verfahren nicht von den Netzwerkbedingungen abhängig und machen so eine Verfahren gegen Integritätsverlust überflüssig.

Es ist schwierig einen realen Netzwerkkanal zu realisieren, da die Reihenfolge der Pakete/Protokolle zufällig ist. Dies ist bei realen Netzwerken nicht der Fall. So kommen zum Beispiel DNS Anfragen viel seltener vor als TCP Pakete.
Die Veränderung der Pakete ist bei diesem Covert Channel nicht nötig. Eine implementierung als passiver Channel ist aber nicht möglich

Projekt Hopping Channels haben die gleichen negativen Eigenschaften wie die Storage Channels und kommen deshalb nicht für diese Projekt in frage.

Die Verwendung der **Nutzdatengröße** für die Datenübertragung ist sehr unauffällig. Die Pakete bleiben bis auf die Nutzdaten regulär und ziehen so fast keine Aufmerksamkeit auf sich. Der Kanal ist nicht von Netzwerkbedingungen abhängig und die Datenübertragung kann 8 Bit pro Paket betragen. Es muss jedoch eine unauffällige Methode ausgearbeitet werden, bei der es möglich ist variable Datenpakete zu versenden. Es ist hier ebenfalls nicht möglich den Kanal passiv zu gestalten.

Werden **Bilddateien** zum Einbetten der Daten verwendet, muss sich überlegt werden wie die Übertragung der Bilder stattfinden soll. Hier muss eine unauffällige Möglichkeit gefunden werden, um diesen Austausch zu realisieren. Hier lassen sich viele Daten auf einmal überragen. Hingegen ist diese Methode vor allem durch die Medien sehr bekannt geworden. Ein aktuelles Beispiel ist eine Sicherheitslücke in Android, wo durch das Öffnen eines PNG Bildes Schadcode mit den Rechten des Benutzers ausgeführt werden kann. [De19].

Da der zeitabhängigen Covert-Channel die Pakete nicht manipulieren muss und da beim Netzwerkverkehr, bis auf den zeitlichen Offset, keine Veränderung vorgenommen werden muss, soll dieses Projekt mit diesem Kanal durchgeführt werden. Zudem ist die Möglichkeit den Channel passiv zu realisieren und die allgemeine Unbekanntheit des Channels weitere positive Aspekte.

6 Projekt Umsetzung

6.1 Konstruktion des Covert Channel

Wie in den Grundlagen schon beschrieben, kann man einen steganografischen Kanal mit folgender Formel veranschaulichen:

Steganographischer Covert Channel = Geheime Daten + Trägerkanal + Steganografischer Schlüssel

6.1.1 Geheime Daten

Die geheimen Daten beinhalten die Information, welche versteckt übertragen werden. Diese sollen jedes beliebige Format annehmen können. Zum Senden werden die Daten in ihre binären Form übertragen. So muss sich keine Sorge um das Datenformat gemacht werden.
Da ein sehr geringe Übertragungsgeschwindigkeit erwartet wird ist es hilfreich, wenn diese Daten so klein wie möglich ausfallen.
Zum Entwickeln wird hierzu die Datei *test.txt* verwendet, die als Inhalt den String „Hallo Welt" besitzt.

6.1.2 Steganografischer Schlüssel

Der steganografische Schlüssel bildet sich aus dem, im vorhergehenden Kapitel gewählten, Covert Channel. Er bestimmt auf welche Art die Daten in den Träger infiltriert und später exfiltriert werden.
Aufgrund der Wahl des zeitabhängigen Covert-Channel können folgende Schlüssel definiert werden.

Schlüssel zur Dateneinfiltration:

Manipuliere den Datenstrom des Trägerkanals so, dass die geheimen Daten kodiert werden.

Schlüssel zur Datenexfiltration:

Lese die Zeitabstände der Datenpakete im Trägerkanal und dekodiere diese.

6.1.3 Trägerkanal

Es wird ein Trägerkanal benötigt in diesen die geheimen Nachrichten eingebettet werden sollen.
Für den gewählten Covert Channel kann prinzipiell jedes Netzwerkprotokoll verwendet werden. Da die Übertragung über Netzwerkgrenzen hinaus stattfinden soll muss jedoch mindestens das Internet Protokoll verwendet werden. Die Verwendung von ICMP Paketen ist durch die Filterung von Firewalls nicht geeignet. Die verbleibenden und sinnvollen Möglichkeiten sind demnach entweder TCP oder UDP. Da immer nur ein Bit durch ein Datenpaket übertragen wird wird ein Datenstrom (Stream) mit vielen Datenpaketen benötigt.
Die Entscheidung Beschränkt sich nun auf einen UDP oder TCP Stream. UDP hat das Problem, dass es sich hierbei um eine verbindungsloses Protokoll handelt und man nicht sicher sein kann, dass die Daten in der Richtigen Reihenfolge oder überhaupt ankommen. Durch die Verwendung eines zeitlichen Covert Channel muss aber sowieso ein System zur Sicherstellung der Integrität implementiert werden.

Auf der anderen Seite sind UDP Pakete selten geworden, da HTTP und somit die Webserver auf TCP aufbauen. Die Streams von Firmen wie Netflix oder YouTube basieren heute alle auf dem TCP/IP Stack.
So ist die Verwendung eines TCP Streams die Beste Lösung, wobei die verbindungsorientierte Datenübertragung von TCP sich zusätzlich positiv auf die Zuverlässigkeit des Datenkanals auswirken wird.

6.2 Aufbau des Systems

In diesem Kapitel sollen die einzelnen Bausteine, die im vorhergehenden Abschnitt beschriebenen wurden, zu einem kompletten System zusammengefügt werden. Der Aufbau diesen Systems dient dann später als Struktur bei der Programmierung.
Im allgemeinen können hier zwei Systeme entstehen, entweder ein System mit einem aktiven oder passiven Covert-Channel.

6.2.1 Aktiv

Soll ein aktiver Covert Channel erstellt werden, so ist der Sender gleichzeitig als Server realisiert. Dieser sendet aktiv Datenpakete an den Empfänger. Durch die Codierung, der geheimen Nachricht in die Zeitabstände zwischen den Paketen, gelangt die Information zum Empfänger. In *Abbildung 6.1* wird dieses System vereinfacht dargestellt.

6.2.2 Passiv

Bei der passiven Alternative verbindet sich der Empfänger über einen Proxy mit einem beliebigen Webserver, der einen konstanten Datenstrom generiert. Dies könnte zum Beispiel ein Video- oder Audiostream sein.
Dieser Datenstrom läuft über den Proxy, der nun die Möglichkeit hat den Datenstrom zu

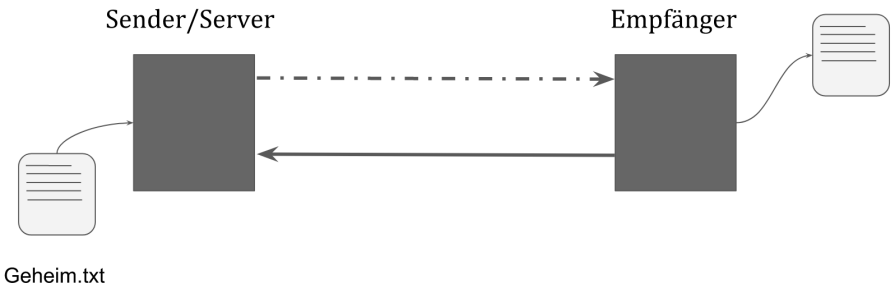


Bild 6.1: Aufbau eines aktiven Systems

manipulieren. Der Sender der geheimen Nachrichten nimmt in diesem System die Rolle des Proxys ein. Er muss die Daten nicht verändern sondern nur verzögern.

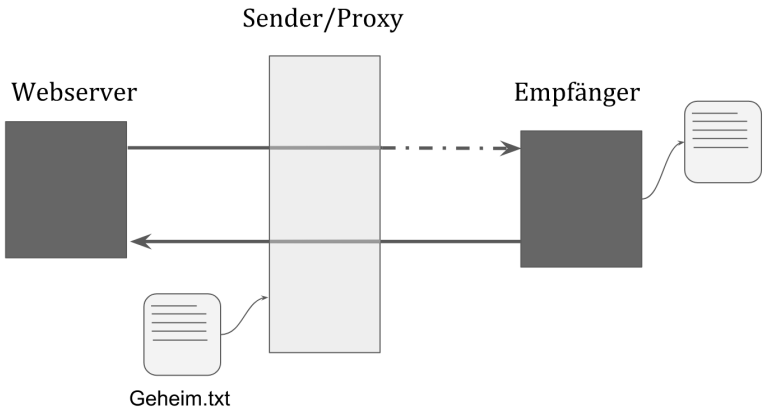


Bild 6.2: Aufbau eines passiven Systems

6.3 Kodierung und Dekodierung

Der folgende Abschnitt beschäftigt sich damit, wie die Daten in den Trägerkanal codiert werden können indem nur der Zeitpunk des Versendens manipuliert wird.
Folgend werden zwei Methoden vorgestellt, die zur Kodierung der Binärdaten in frage kommen.

6.3.1 Mit festen Zeitrastern

Das in [CBS04] vorgestellte Verfahren basiert auf einer Generierung von Zeitintervallen. Diese werden sowohl beim Sender und Empfänger erstellt. Dabei haben die Intervalle des Empfängers einen zeitlichen Offset der ungefähr der Übertragungsdauer entspricht. Dies dient dazu, dass Pakete die vom Sender abgesendet werden beim Empfänger im gleichen Intervall ankommen. Nachdem Sender und Empfänger synchronisiert sind, kann man mit der Datenübertragung starten. Dabei wird ein Paket, dass in einem Zeitintervall ankommt als binäre 1 interpretiert. Kommt kein Paket so wird eine 0 geschrieben. Durch die Wahl längerer Zeitintervalle kann die Fehleranfälligkeit verringert werden, wobei jedoch die Übertragungsgeschwindigkeit ebenfalls abnimmt.
Für diese Art der Codierung ist es unabdingbar, dass die Uhren von Sender und Empfänger möglichst genau übereinstimmen.
Zur Veranschaulichung ist die Kodierung in *Abbildung 6.3* dargestellt.

6.3.2 Basierend auf den Paketabständen

Zur Kodierung kann ebenso die Veränderung der Paketabstände benutzt werden. Hierzu wird zwischen einer kleinen oder großen Pause zwischen zwei Nachrichten unterschieden. Eine große Pause wird als binäre 1 interpretiert, eine kleine Pause als 0.
Je nachdem wie gut die Netzwerkbedingungen sind kann hier durch eine gezielte Verkleinerung der Pausen eine Erhöhung der Übertragungsgeschwindigkeit generiert werden. Durch eine Verlängerung dieser Pausen sinkt jedoch die Fehleranfälligkeit.
Zu sehen ist diese Art der Kodierung in *Abbildung 6.4*

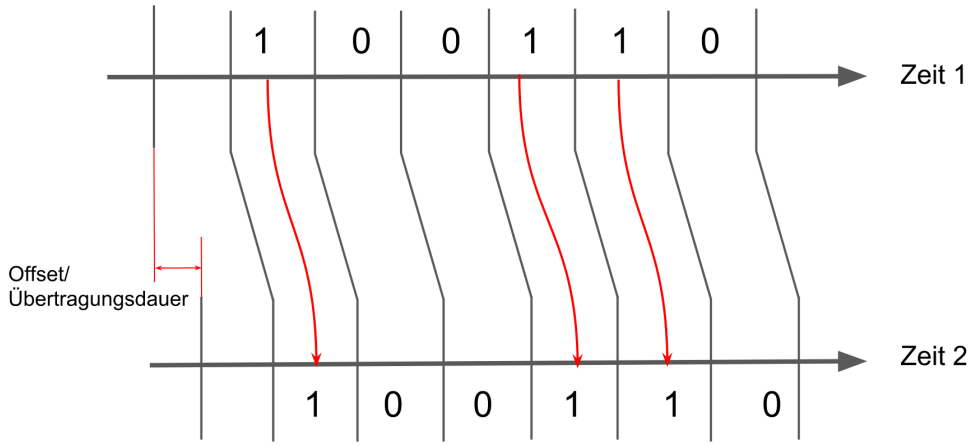


Bild 6.3: Kodierung mit festen Zeitrastern

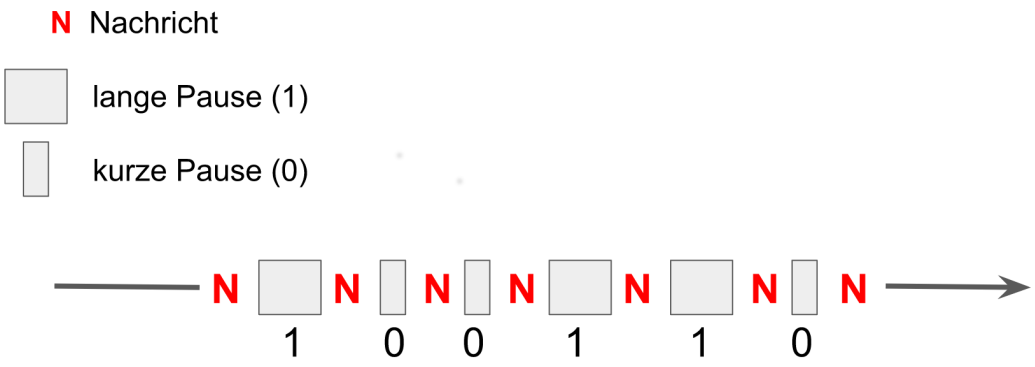


Bild 6.4: Kodierung anhand der Paketabständen

6.3.3 Bewertung der Kodierung

Bei der Kodierung mit Zeitrastern muss sich um die Synchronisierung gekümmert werden. Außerdem gilt es den Offset passend zu wählen. Diese Methode hat jedoch den Vorteil, dass zum Senden einer 0 keine Nachricht benötigt wird. Kommt es jedoch zu Fehlern im Offset oder zu plötzlichen Übertragungsschwankung ist dieses System sehr Anfällig, da

die Nachrichten in andere Zeitintervalle hineingeraten könnten.

Werden die Paketabstände zu Kodierung verwendet, so wird zum Senden einer 0 eine Nachricht benötigt. Dadurch werden mehr Nachrichten benötigt, die kontinuierlich zu Verfügung sehen müssen.
Dabei besteht ein Vorteil darin, dass sich nicht um die Synchronisierung gekümmert werden muss. Zusätzlich lässt sich bei schlechten Netzwerkbedingungen die Übertragung pausieren. Hierzu kann einfach die Datenübertragung gestoppt werden. Bei der Codierung mit Rastern würde dieses abrupte Stoppen der Datenübertragung als 0en interpretiert werden. Bei einer reinen Betrachtung der Differenz zwischen den langen und kurzen Pausen ist der Sender in der Lage die Sendegeschwindigkeit beliebig anzupassen.

Aufgrund der Vorteile durch die Codierung mit Hilfe der Paketabstände soll diese Methode in dieser Arbeit verwendet werden.

6.4 Wahrung der Integrität

6.5 Optimierung

Buffer Pausenverkleinerung

6.6 Implementierung

6.6.1 Aktiv

6.6.2 Passiv

6.7 Bewertung der Ergebnisse

7 Etische Aspekte

vieleicht hinter Grundlagen

8 Schlussbemerkungen und Ausblick

A Ein Kapitel des Anhangs

Literatur

[De19] Dennis Schirrmacher : *Patchday: Das Öffnen von PNG-Bildern kann Android-Geräte kompromittieren.* `https://www.heise.de/security/meldung/Patchday-Das-Oeffnen-von-PNG-Bildern-kann-Android-Geraete-kompromittieren-430html`, 2019. Abrufdatum: 12.02.2019.

[CBS04] Cabuk, Serdar, Carla E Brodley Clay Shields: *IP covert timing channels: design and detection. Proceedings of the 11th ACM conference on Computer and communications security*, 178–187. ACM, 2004.

[Die18] Die Bundesbeauftragten für den Datenschutz und die Informationssicherheit (BfDI): *Was ist Datenschutz?* `https://www.bfdi.bund.de/DE/Datenschutz/Ueberblick/ueberblick-node.html`, 2018. Abrufdatum: 16.10.2018.

[Ert01] Ertel, Wolfgang: *Angewandte Kryptographie. Fachbuchverlag Leipzig.* Carl Hanser Verlag), ISBN, 2001.

[Gol03] Goltz, James P.: *Under the radar:A look at three covert communications channels.* 2003.

[Hel18] Hellmann, Roland: *IT-Sicherheit: eine Einführung.* Walter de Gruyter GmbH & Co KG, 2018.

[Inf81] Information Sciences Institute University of Southern California: *TRANSMISSION CONTROL PROTOCOL DARPA INTERNET PROGRAM PROTOCOL SPECIFICATION.* `https://tools.ietf.org/html/rfc793`, 1981. Abrufdatum: 01.02.2019.

[Kes15] Kessler, Gary C.: *An Overview of Steganography for the Computer Forensics Examiner (Updated Version, February 2015)*, 2015.

[KP00] Katzenbeisser, Stefan Fabien Petitcolas: *Information hiding techniques for steganography and digital watermarking.* Artech house, 2000.

[LC17] Lockwood, Robert Kevin Curran: *Text based steganography.* International Journal of Information Privacy, Security and Integrity, 3(2):134–153, 2017.

[LT10] Lee, Che-Wei Wen-Hsiang Tsai: *A new steganographic method based on information sharing via PNG images. 2nd International Conference on Computer and Automation Engineering (ICCAE)*, 2010.

[Nic18] Nico Grundmeier: *Sicherheitslücken im Internet.* `http://www.informatik.uni-oldenburg.de/~iug10/sli/indexd917.html?q=node/19`, 2018. Abrufdatum: 16.10.2018.

[Pur10] Purgathofer, Peter: *Eine kurze Geschichte der Steganographie.* Die Funktion verdeckter Kommunikation: Impulse für eine Technikfolgenabschätzung zur Steganographie, 9:65, 2010.

[Use18] User: Black Slash: *How to Hide Secret Data Inside an Image or Audio File in Seconds.* `https://null-byte.wonderhowto.com/how-to/steganography-hide-secret-data-inside-image-audio-file-seconds-0180936/`, 2018. Abrufdatum: 22.01.2019.

[Wen12a] Wendzel, Steffen: *The Problem of Traffic Normalization Within a Covert Channel's Network Environment Learning Phase. Sicherheit*, 12, 149–161, 2012.

[Wen12b] Wendzel, Steffen: *Tunnel und verdeckte Kanäle im Netz: Grundlagen, Protokolle, Sicherheit und Methoden.* Springer-Verlag, 2012.

[Wik18] Wikipedia contributors: *OSI-Modell — Wikipedia, The Free Encyclopedia.* `https://de.wikipedia.org/wiki/OSI-Modell`, 2018. Abrufdatum: 03.01.2019.

[ZCC07] Zhong, Shangping, Xueqi Cheng Tierui Chen: *Data Hiding in a Kind of PDF Texts for Secret Communication.* IJ Network Security, 4(1):17–26, 2007.

[Zis13] Zisler, Harald: *Computer-Netzwerke*, 2013.