Sichere und effiziente Netzwerkcodierung

Dissertation

zur Erlangung des akademischen Grades Doktoringenieur (Dr.-Ing.)

vorgelegt an der
Technischen Universität Dresden
Fakultät Informatik

eingereicht von

Stefan Pfennig
geboren am 09. November 1986 in Oschatz

Gutachter:
Prof. Dr. Thorsten Strufe (Technische Universität Dresden)
Prof. Dr. Falko Dressler (Universität Paderborn)

Tag der Verteidigung: 18. Dezember 2018

Dresden, Januar 2019

Kurzfassung

Gegenwärtige Entwicklungen in der Informationstechnik verändern unsere Wirtschaft und Gesellschaft. Die Digitalisierung dringt in die verschiedensten Lebensbereiche vor und durch das Internet der Dinge steigt der Bedarf an schneller digitaler Kommunikation. Dabei sind die Kommunikationsnetzwerke und die Geräte der Teilnehmer so heterogen wie nie. Gerade für mobile Geräte und Sensoren ist die Energieeffizienz von immenser Bedeutung, da eine zu geringe Laufzeit der Geräte den Nutzen stark einschränken würde. Im gleichen Maß wächst auch der Bedarf an Sicherheit. Zum einen, weil durch Skandale in letzter Zeit auch einem Großteil der Menschen bekannt geworden ist, was durch Geheimdienste und ähnliche Institutionen nicht nur alles möglich ist, sondern auch tatsächlich durchgeführt wird. Zum anderen, weil die Daten immer privater (z. B. E-Health) werden, Geräte ständig am Körper getragen werden und Veränderungen an Daten stärkere Auswirkungen auf das Leben der Nutzer haben können (z. B. Smart Home, digitale Behörden).

Trotz dieser erhöhten Anforderungen an Sicherheit und Effizienz der Kommunikation findet die Datenübertragung heutzutage größtenteils mittels einer Technik aus den 1960er Jahren statt: der Paketvermittlung. Dabei werden Nachrichten in Pakete unterteilt und über verschiedene Zwischenknoten (z. B. Router/Switches) zum Empfänger geschickt. An diesen Zwischenknoten werden Pakete kurzzeitig gespeichert und dann weitergeleitet. Anfang des aktuellen Jahrtausends wurde eine Verbesserung der Effizienz beobachtet, wenn die Zwischenknoten in der Lage sind, verschiedene Pakete miteinander zu codieren. Die Idee der Netzwerkcodierung war geboren. Seitdem beschäftigen sich viele Arbeiten mit neuen Anwendungsmöglichkeiten und Vorteilen dieser Technik.

Die Netzwerkcodierung hat allerdings Auswirkungen auf die Anforderungen an die Sicherheit. Auf der einen Seite bietet sie neue Angriffsmöglichkeiten, die auf das Kombinieren von Paketen zielen. Zum anderen bietet gerade dieses Kombinieren auch einen gewissen Schutz gegenüber Abhörern. Daher wurden bereits einige Arbeiten und Verfahren zum Thema der sicheren Netzwerkcodierung präsentiert.

Ziel und Gegenstand der vorliegenden Untersuchungen sind die Gruppierung und Einordnung bestehender Verfahren und eine Auswertung bezüglich Effizienz und Sicherheit. Dabei soll auf die drei Hauptschutzziele der Informationstechnik (Vertraulichkeit, Integrität und Verfügbarkeit) eingegangen werden. Aufbauend auf diesen Ergebnissen sollen Optimierungen an den Verfahren oder Protokollen durchgeführt und gezeigt werden, dass sichere und effiziente Netzwerkcodierung möglich ist und die Vorteile gegenüber herkömmlicher Paketvermittlung überwiegen.

Dafür wurden verschiedene Aspekte der Sicherheit untersucht und die jeweils passende Methodik ausgewählt. Für die Untersuchung der Integrität und Verfügbarkeit wurde zuerst der Mindestaufwand ermittelt, der auch ohne aktive Angriffe notwendig ist. Dazu wurde eine theoretische Analyse der Effizienz durchgeführt. Später wurde auch der Aufwand für Übertragungen mit aktiven Angriffen und damit einhergehenden Paketverlusten untersucht. Hierzu wurde eine eigene Simulationsumgebung entwickelt, welche Grundlage für weitere Untersuchungen war. Für die Untersuchungen bezüglich der Vertraulichkeit

wurden die einzelnen Verfahren implementiert und Effizienzparameter gemessen. Die anschließenden Sicherheitsbetrachtungen der Verfahren wurden analytisch durchgeführt.
Die Ergebnisse der Untersuchungen sind im Folgenden zusammengefasst. Verfahren zur Integritätsabsicherung, welche verfälschte Pakete an den Zwischenknoten durch kryptographische Methoden erkennen, arbeiten effizienter als Verfahren, welche mittels Codierungstheorie am Empfänger die Fehler beheben. Die ratenlose Übertragung, d. h. das potentiell unbegrenzte Senden bis zur Empfangsbestätigung vom Empfänger, ist effizienter als die Verwendung festgelegter vordefinierter Redundanz. Für die ratenlose Übertragung wurden verschiedene Übertragungsprotokolle untersucht, welche die Bestätigungen zur Schätzung der Wahrscheinlichkeit einer erfolgreichen Übertragung verwenden, um sich an Angreifer anzupassen. Weiterhin werden Optimierungen und Adaptionsmöglichkeiten bezüglich verschiedener Effizienzparameter, wie Übertragungszeit, Kommunikationsaufwand und Energie aufgezeigt.
Bezüglich der kryptographischen Methoden wurden Vertreter unterschiedlicher Gruppen von Verfahren gegeneinander mit verschiedenen Netzwerken untersucht. Dabei stellte sich heraus, dass eine starke Netzwerkabhängigkeit bestand, sodass die Parameter, welche die Effizienz am stärksten beeinflussten, in ein generalisiertes Netzwerkmodell aufgenommen wurden. Anschließend konnte gezeigt werden, dass alle ausgewählten Verfahren je nach Netzwerk und Effizienzparameter Vorteile haben. Daher wurde ein adaptives Vorgehen vorgeschlagen, welches zu jedem Netzwerk ein optimales Verfahren der kryptographiebasierten Ansätze auswählt.
Bezüglich der Vertraulichkeit wurden verschiedene leichtgewichtige Verfahren implementiert und gegenüber herkömmlicher Ende-zu-Ende-Verschlüsselung verglichen. Zusätzlich wurde ein eigenes Verfahren namens eSPOC entwickelt, welches deutliche Verbesserungen hinsichtlich der Leistung bei gleichbleibender Sicherheit gegenüber den bisherigen Verfahren bietet. Außerdem wird die Sicherheit der Verfahren untersucht und bewertet und die Grenzen der algebraischen Sicherheit aufgezeigt. Abschließend wird die Verwendung der leichtgewichtigen Verfahren, insbesondere von eSPOC, für verteilte Speicher analysiert und die Vorteile aufgezeigt.
Die präsentierten Ergebnisse können direkt (Implementierungen) oder indirekt (Ergebnisse) in aktuelle Umsetzungen eingehen und damit Effizienzsteigerungen bewirken. Auch werden die unterschiedlichen Aspekte der sicheren Netzwerkcodierung übersichtlich dargestellt und Ansatzpunkte für weitere Entwicklungen gegeben. Dadurch kann die Forschung zur Netzwerkcodierung vorangebracht werden und die Technologie in der Zukunft mehrere Anwendungsfelder finden.

Inhaltsverzeichnis

Abkürzungsverzeichnis

ACK	Acknowledgement – Empfangsbestätigung
AES	Advanced Encryption Standard – Symmetrisches Verschlüsselungsverfahren [DBN+01]
ARQ	Automatic Repeat Request – Wiederholungsanfragen
CDN	Content Distribution Network – Verteilter Speicher für den Datenaustausch
CM	Codierungsmatrix
CodingRSNC	Gruppe integritätsichernder Verfahren, welche auf Codierungstheorie basieren und ratenlos übertragen werden
CodingSNC	Gruppe integritätsichernder Verfahren, welche auf Codierungstheorie basieren
COPE	Inter-Session-Netzwerkcodierung Verfahren [KRH+06]
CRC	Cyclic Redundancy Check – Verfahren zur Bestimmung eines Prüfwerts
CryptoRSNC	Gruppe integritätsichernder Verfahren, welche auf Kryptographie basieren und ratenlos übertragen werden
CryptoSNC	Gruppe integritätsichernder Verfahren, welche auf Kryptographie basieren
DART	Kryptographisches, integritätsicherndes Verfahren, welches auf Checksummen basiert [DCNR09]
ECDSA	Elliptic Curve Digital Signature Algorithm – Digitales Signaturverfahren
EncPay	Encrypted Payload – Ende-zu-Ende-Verschlüsselung der Nutzdaten
eSPOC	Enhanced Secure Practical Network Coding – eigenes Verfahren für die Vertraulichkeit [PF16]
GEM	Global Encoding Matrix – Globale Codierungsmatrix
GEV	Global Encoding Vector – Globaler Codierungsvektor
HEF	Homomorphic Encryption Function – homomorphe Verschlüsselungsfunktion
HTTP	Hypertext Transfer Protocol – Datenübertragungsprotokoll auf der Anwendungsschicht
IP	Internet-Protokoll
MAC	Message Authentication Code – Nachrichtenauthentifizierungscode
MAC-Schicht	Media Access Control – Sicherungsschicht
MORE	Intra-Session-Netzwerkcodierung Verfahren [CJKK07]
MTU	Maximum Transmission Unit – Maximal Übertragungseinheit
NC	Netzwerkcodierung
NC_d	Netzwerkcodierung mit dynamischer Redundanz
NC_{da}	Netzwerkcodierung mit dynamischer Redundanz und adaptiven Bestätigungen
NC_{dp}	Netzwerkcodierung mit dynamischer Redundanz und permanenten Bestätigungen
NC_s	Netzwerkcodierung mit statischer Redundanz
NC_{ss}	Netzwerkcodierung mit statischer Redundanz zu Beginn
NC_Φ	Netzwerkcodierung mit adaptiver Redundanz
NECO	Network Coding Simulator – Simulationsumgebung für Netzwerkcodierung [Fer09]

P-Coding	Verfahren für die Vertraulichkeit basierend auf Permutationen [ZJL+10]
PNC	Practical Network Coding – Verfahren für ungesicherte Netzwerkcodierung [CWJ03]
RAID	Redundant Array of Independent Disks – Verfahren, um Redundanz in Festplattenverbände einzubringen
RLNC	Random Linear Network Coding – Lineare Netzwerkcodierung mit zufälligen Koeffizienten
RSA	Asymmetrisches Kryptographieverfahren von Rivest, Shamir und Adleman
RT	Routing
RTT	Round Trip Time – Umlaufzeit
SIMD	Single Instruction, Multiple Data – Anwendung gleicher Operationen auf mehrere Datenströme
SPOC	Secure Practical Network Coding – Verfahren für Vertraulichkeit [VLB08]
TCP	Transmission Control Protocol – Verbindungsorientiertes Netzwerkprotokoll zwischen 2 Endpunkten auf der Transportschicht
TESLA	Protokoll zur Authentifizierung basierend auf Zeitasymmetrie [PCTS02]
UDP	User Datagram Protocol – Verbindungsloses Netzwerkprotokoll zwischen 2 Endpunkten auf der Transportschicht
XOR	Exclusive OR – exklusives Oder

Symbolverzeichnis

Netzwerk

$\mathbf{E}$	Menge der Kanten eines Netzwerks
$\mathbf{F}$	Menge der Zwischenknoten eines Netzwerks
$\mathbf{R}$	Menge der Empfänger eines Netzwerks
$\mathbf{S}$	Menge der Sender eines Netzwerks
$\mathbf{V}$	Menge der Knoten eines Netzwerks
G	Netzwerk bestehend aus Knoten und Kanten
e	Kante eines Netzwerks
f	Zwischenknoten eines Netzwerks
r	Empfänger eines Netzwerks
s	Sender eines Netzwerks

Variablen und Parameter

a	Beliebige Nachricht
b	Beliebige Nachricht
c	Minimaler Schnitt des Netzwerks
d	Breite des Netzwerks
e	Mehraufwand für Zusatzdaten
f	Berechnungsaufwand
g	Anzahl der Generationen
i	Laufvariable
j	Laufvariable
k	Anzahl der Pakete, die vom Angreifer abgehört wurden
l	Anzahl der MACs
ℓ	Anzahl der Hops im Netzwerk
m	Nachricht
n	Anzahl der Nutzdatensymbole
o	Generations-ID
p	Primzahl
q	Körpergröße
r	Reduktionsfaktor
s	Seed
t	Zeit

u	Anzahl an Positionen
v	Nutzdatenmenge
w	Paketgröße mit Header-Daten
x	Paketdaten
y	Codierte Paketdaten
z	Anzahl angegriffener Kanten
A	Bestätigung (Acknowledgement)
E	Übertragungseffizienz
M	MAC
N	RSA-Modul
O	Anzahl der Sendeoperationen
P	Relative Nutzdaten
R	Rate
S	Szenario
T	Anzahl an Zeiteinheiten
α	(Lokaler) Codierungskoeffizient
β	(Globaler) Codierungskoeffizient
δ	Kantenübertragungswahrscheinlichkeit
ϵ	Restwahrscheinlichkeit
η	Anzahl codierter Pakete pro Generation
θ	Laufvariable
κ	Schlüssel
λ	Laufvariable
ν	Verbindungsgrad
ξ	Bitsicherheit
ρ	Angriffswahrscheinlichkeit
τ	Effizienzverhältnis
ϕ	Wahrscheinlichkeit
Φ	Gewichtungsfaktor
χ	Wahrscheinlichkeit für partielle Invertierbarkeit

Vektoren und Matrizen

$\boldsymbol{\alpha}$	Menge von Codierungskoeffizienten $\widehat{=}$ Codierungskoeffzienten eines Pakets
$\boldsymbol{\beta}$	Menge von Codierungskoeffizienten $\widehat{=}$ Codierungskoeffzienten eines Pakets
$\boldsymbol{\kappa}$	Menge von Schlüsseln
$\boldsymbol{m}$	Menge von Nachrichten
$\boldsymbol{S}$	Menge von Seedwerten
$\boldsymbol{x}$	Menge von Paketdaten $\widehat{=}$ Paket
$\boldsymbol{y}$	Menge von Paketdaten $\widehat{=}$ Paket
$\boldsymbol{A}$	Menge von Codierungsvektoren $\widehat{=}$ Codierungsmatrix
$\boldsymbol{B}$	Menge von Codierungsvektoren $\widehat{=}$ Codierungsmatrix

$\boldsymbol{G}$	Menge von Paketen einer Generation $\widehat{=}$ Generationsmatrix
$\boldsymbol{H}$	Menge von Hashwerten $\widehat{=}$ Hashmatrix
$\boldsymbol{P}$	Paritätsmatrix
$\boldsymbol{S}$	Syndrommatrix
$\boldsymbol{Y}$	Menge von empfangenen Paketen einer Generation

Funktionen

chk(.)	Checksummenfunktion
dec(.)	Decodierungsfunktion
enc(.)	Codierungsfunktion
gen(.)	Generierungsfunktion
h(.)	Hashfunktion
in(v)	Knoten, die eine Nachricht an v senden können
inv(h, q)	Wahrscheinlichkeit für die Invertierbarkeit zufälliger Matrizen der Größe $h \times h$ in $\mathbb{F}_q$
inv$_\infty(q)$	Wahrscheinlichkeit für die Invertierbarkeit zufälliger großer Matrizen in $\mathbb{F}_q$
ld(.)	Logarithmus zur Basis 2
out(v)	Knoten, die eine Nachricht von v empfangen können
perm(.)	Permutationsfunktion
rg(.)	Rang einer Matrix
round(.)	Kaufmännisches Runden
sig(.)	Signaturfunktion

Algebraische Strukturen

$\mathbb{F}$	Körper
$\mathbb{N}$	Menge der natürlichen Zahlen
$\mathbb{Q}$	Menge der rationalen Zahlen
$\mathbb{U}$	Untervektorraum
$\mathbb{Z}$	Menge der ganzen Zahlen

1 Einführung

1.1 Motivation

Kommunikation beeinflusst seit jeher das Leben und die Entwicklung der Menschen. Dabei spielt nicht nur die gesprochene Sprache eine entscheidende Rolle, sondern auch Kommunikation über verschiedenste Signale. Schon vor über 2 000 Jahren gab es Leuchttürme, die vor Küsten gewarnt haben und Positionsbestimmungen ermöglichten. Die Feuerzeichen auf der Chinesischen Mauer warnten vor Angriffen. Auch Rauchsignale sind frühe Formen der Fernkommunikation.
Immer wieder hatten dabei neue Entwicklungen bei den Kommunikationswegen neue soziale, kulturelle und wirtschaftliche Entwicklungen in Gang gebracht. Allein der Buchdruck beispielsweise ermöglichte Kommunikation von Wissen auf erschwingliche Weise. Dies hatte signifikante Folgen auf die Gesellschaft und die Entwicklung der Wirtschaft. Die moderne elektromagnetische Kommunikation startete im 19. Jahrhundert mit den ersten Telegraphen, welche eine Nachricht über eine elektrische Leitung auch über weite Strecken übertragen konnten. Erste Überseekabel machten die Kommunikation über den Atlantik deutlich schneller als per Schiff möglich. Um eine Nachricht zu senden, wurde eine physische Leitung vom Sender zum Empfänger über Vermittlungen aufgebaut. Dann konnten die Nachrichten mittels Codierung, z. B. Morse-Code, übermittelt werden. Diese Art der Leitungsvermittlung war bis vor Kurzem noch der Standard bei der analogen Festnetztelefonie.
Die Vorteile der Leitungsvermittlung waren, dass die Vermittlungspunkte zwischen Sender und Empfänger sehr einfach sein konnten. In frühen Zeiten reichte ein einfaches Koppelfeld, um 2 Leitungen miteinander zu verbinden. Auch wenn sich die Technik zu moderneren elektronischen Vermittlungsstellen wandelte, mussten an den Zwischenknoten nach dem Verbindungsaufbau keine Aufgaben, wie z. B. eine Datenspeicherung, erledigt werden. Der Nachteil allerdings war die schlechte Skalierbarkeit und Effizienz der Lösung. Waren beispielsweise nur 5 Leitungen verfügbar und 10 Verbindungen benötigt, mussten 5 Kommunikationspaare warten, auch wenn eine Leitung nicht dauerhaft genutzt wurde. Dieses Problem ließ sich nur über mehr Leitungen lösen, was sehr hohe Kosten verursachte.
Eine Lösung des Problems und eine Erhöhung der Effizienz kam über die Entwicklung der Paketvermittlung in den 1960er Jahren. Anstatt dedizierte Leitungen für Kommunikationspartner bereit zu halten, werden die Nachrichten in Pakete zerlegt und dann über verschiedene Kommunikationskanäle zum Empfänger übertragen. Dadurch erhielt man eine deutliche Effizienzsteigerung, da ungenutzte Kapazitäten auf vorhandenen Leitungen fast vollständig ausgenutzt werden können. Zusätzlich wurde dadurch eine gewisse Fairness erzeugt, denn – zurückkommend auf das Beispiel – könnten nun alle 10 Kommunikationspaare miteinander kommunizieren, allerdings mit verminderter Übertragungsrate.
Aber auch die Paketvermittlung hatte einige Nachteile. Zum einen wurden neue Vermittlungsstellen benötigt, welche, anstatt eine Verbindung zwischen zwei Leitungen herzustellen, deutlich aufwendigere Aufgaben hatten. Empfangene Pakete mussten zwischengespeichert und anhand der Adressinformationen weitergeleitet werden. Dadurch, dass Pakete anstatt ganzer Nachrichten übertragen wurden, war auch

die Reihenfolge der Ankunft beim Empfänger nicht sichergestellt. Ferner ist so auch keine Garantie über die Übertragungsgeschwindigkeit möglich.

Trotz all dieser Nachteile ist die Paketvermittlung heutzutage der Standard in fast allen Kommunikationsnetzen, da die Nachteile durch Optimierungen und Protokolle, welche beispielsweise die Reihenfolge der Pakete wiederherstellen, fast vollständig ausgeglichen werden können. Somit überwiegen die Vorteile deutlich gegenüber den Nachteilen der Paketvermittlung, obwohl den Zwischenknoten mehr Arbeit abverlangt wird.

Gerade diese Art der Vermittlung machte erst das Internet und die damit verbundenen gesellschaftlichen und wirtschaftlichen Entwicklungen möglich. Doch nach Jahren der stetigen Entwicklungen gibt es mehr und mehr Probleme mit der aktuellen Kommunikation. Mit dem Internet der Dinge und Industrie 4.0 kommunizieren nicht nur Menschen untereinander oder mit Maschinen, sondern auch Maschinen mit Maschinen. Dadurch ergibt sich eine immense Steigerung des Kommunikationsbedarfs. Auch die Entwicklungen beim Cloud Computing lassen aktuelle Netzwerke an ihre Kapazitätsgrenzen kommen.

Mit der Netzwerkcodierung [ACLY00] besteht die Möglichkeit, die Nachteile der bisherigen Technologien zu umgehen und die Kommunikation der Zukunft zu ermöglichen. Bei der Netzwerkcodierung erhalten die Zwischenknoten eine weitere Aufgabe. Zwischen dem Speichern und Weiterleiten von Paketen wird ein Berechnungsschritt eingefügt. Dabei werden Pakete miteinander über eine Linearkombination verknüpft.

Diese Vorgehensweise bringt viele Vorteile, wie z. B. die Erhöhung des Durchsatzes, die Verbesserung der Ausfallsicherheit oder die Verringerung des Energiebedarfs. Andererseits gibt es natürlich auch Nachteile, wie der bereits geschilderte erhöhte Aufwand für die Zwischenknoten. Wie auch bei der Einführung der Paketvermittlung gilt es also, Optimierungen und Protokolle zu finden, welche die Nachteile minimieren und die Vorteile verstärken.

In vorliegender Dissertation soll es um einen wichtigen Aspekt bei der Kommunikation mittels Netzwerkcodierung gehen: die Sicherheit. Nur, wenn die Informationen vertraulich sowie integer übertragen werden und dem Empfänger verfügbar sind, sind diese auch nützlich. Für die Notwendigkeit der sicheren Kommunikation findet man schon frühe Beispiele. Für die geheime militärische Kommunikation wurde z. B. schon vor über $2\,000$ Jahren eine Verschiebechiffre (Cäsar-Algorithmus) angewandt, um beim Abfangen der Nachrichten keinerlei Informationen preiszugeben. Um die Herkunft und Originalität von Dokumenten nachzuweisen, wurden auch schon vor der modernen Kommunikation Siegel eingesetzt.

Gerade heute wächst die Bedeutung der Sicherheit bei der Kommunikation immens. Auf der einen Seite zeigen die Enthüllungen durch Snowden, wie viel Überwachung durch Geheimdienste stattfindet und was alles möglich ist. Auf der anderen Seite dringt die Kommunikationstechnik immer tiefer in das Privatleben der Menschen ein. War die Kommunikation über das Internet noch vor 20 Jahren für die meisten Menschen nur beruflicher Natur, besitzt heute ein Großteil der Menschen in Europa ein Smartphone für private Zwecke, welches eine dauerhafte mobile Internetverbindung besitzt. Wenn dann noch Themen wie E-Health, beispielsweise über Sensoren am Körper, oder Smart Home, also die Vernetzung und Automatisierung von Haustechnik, hinzukommen, nimmt die Relevanz der Sicherheit enorm zu. So könnten veränderte Patientenakten zu falschen und damit gesundheitsschädlichen Behandlungen führen. Weiterhin ist die Gewährleistung der Vertraulichkeit Voraussetzung, dass Menschen die E-Health Angebote überhaupt nutzen, da diese Daten z. B. nicht dem Arbeitgeber zugänglich sein sollten. Auch ein

elektronisches Türschloss, welches durch eine eingeschränkte Verfügbarkeit nicht öffnet, würde keine Akzeptanz erfahren.

Aus diesem Grund wird Sicherheit in einer zukünftigen Kommunikation einen integralen Bestandteil spielen und nicht nur im Nachhinein als zusätzliche (nicht-funktionale) Eigenschaft betrachtet werden. Somit wird auch die Netzwerkcodierung als solches nur dann eine breite Akzeptanz finden, wenn sowohl Effizienzsteigerungen gegenüber den vorherrschenden Technologien als auch ausreichende Sicherheit demonstriert werden können.

Diese Dissertation soll daher die Netzwerkcodierung unter den Aspekten Sicherheit und Effizienz betrachten und Optimierungen, Protokolle und Verfahren bereitstellen. Diese Betrachtungen und Ergebnisse sollen als Teil einer Argumentation dienen, damit die Netzwerkcodierung Einzug in Kommunikationsnetze erfahren kann und damit in Zukunft neue Technologien ermöglicht werden.

1.2 Ziele der Arbeit

Durch die Zusammenführung von Paketen bei der Netzwerkcodierung ergeben sich besondere Anforderungen an die Sicherheit. Daher soll im Folgenden auf die Argumente und Eigenschaften, die im Zusammenhang mit der Sicherheit stehen, eingegangen werden.

Das Hauptargument gegen Netzwerkcodierung aus Sicherheitssicht ist die Vulnerabilität gegenüber sogenannten Verschmutzungsangriffen (Pollution Attacks). Durch die Linearkombinationen, welche an den Zwischenknoten aus vielen eingegangenen Nachrichten gebildet werden, kann eine verfälschte Nachricht über Kombinationen bei mehreren Zwischenknoten sehr viele Nachrichten verfälschen. Dadurch wird die Integrität der Übertragung verletzt, d. h. die Nachrichten vom Sender zum Empfänger modifiziert.

In extremen Fällen kann ein Verschmutzungsangriff dazu führen, dass keinerlei Kommunikation innerhalb eines Netzwerks mehr möglich ist, da die Anzahl der verfälschten Nachrichten zu groß ist. Somit kann auch die Verfügbarkeit des Systems verletzt und die Kommunikation insgesamt verhindert werden.

Beim herkömmlichen Weiterleiten von Nachrichten hat ein solcher Angriff, also das Einschleusen verfälschter Pakete, nur eine geringe Auswirkung, da maximal so viele verfälschte Pakete beim Empfänger ankommen, wie vom Angreifer modifiziert wurden. Bei der Netzwerkcodierung nimmt der Anteil verfälschter Nachrichten durch das Zusammenfügen von Paketen rasant zu. Selbst ein verfälschtes Paket kann in einem sehr stark vermaschten Netzwerk schon zu einem Ausfall der Kommunikation führen.

Nun gibt es Methoden, welche die Integrität der Pakete absichern und damit die Verfügbarkeit des Systems gewährleisten können. Beispiele für solche Methoden sind digitale Signaturen oder Nachrichtenauthentifizierungscodes (Message Authentication Codes - MACs). Allerdings gibt es auch hier einige Besonderheiten zu beachten, da diese Sicherungsmechanismen auch mit der Bildung von Linearkombinationen kompatibel sein müssen. Deswegen werden homomorphe Verfahren verwendet und genau da liegt ein Problem der sicheren Netzwerkcodierung.

Da die Absicherung mittels homomorpher Verfahren einen großen Aufwand bedeutet, bleibt die Frage, ob sich der erhöhte Aufwand bei der Netzwerkcodierung durch die Vorteile lohnt. Ziel der Arbeit ist es verschiedene Situationen zu untersuchen und die jeweils effizientesten Lösungen zu präsentieren.

Zu Beginn der Arbeit soll zunächst eine Bestandsaufnahme bestehender Verfahren und Methoden durchgeführt werden. Anschließend sollen ähnliche Verfahren zusammengefasst oder verallgemeinert werden,

um möglichst unterschiedliche Verfahren zu evaluieren. Ausgehend davon sollen dann Abhängigkeiten bezüglich des Netzwerks untersucht werden.
Weiterhin soll gezeigt werden, dass durch möglichst einfache Optimierungen an den Verfahren und Übertragungsprotokollen die Effizienz gesteigert werden kann, sodass die Vorteile, welche die Netzwerkcodierung bringt, den Aufwand für die notwendigen Sicherheitsmaßnahmen mehr als kompensieren.
Ein zusätzliches Argument für die Netzwerkcodierung ergibt sich, wenn man nicht nur die Integrität und Verfügbarkeit, sondern auch die Vertraulichkeit betrachtet. Während für die ersten beiden Schutzziele Aufwand betrieben werden muss, birgt die Netzwerkcodierung einen gewissen Schutz hinsichtlich der Vertraulichkeit gegenüber beschränkten passiven Angreifern.
Da die Daten nicht im Klartext vorliegen, sondern meist nur codiert übertragen werden, ist es für Zwischenknoten nicht direkt möglich, den Inhalt der Nachrichten zu erkennen. Dadurch ist ein gewisser Basisschutz in Form einer Ende-zu-Ende-Verschlüsselung gegeben. Diese Eigenschaft, welche auch algebraische Sicherheit genannt wird, ist jedoch nicht ausreichend gegenüber einem Angreifer, welcher genügend Pakete abfangen kann. Dieser kann wie der Empfänger decodieren und die Vertraulichkeit wird verletzt.
Beim Einsatz von Netzwerkcodierung wird für die Vertraulichkeit daher meist noch eine zusätzliche Ende-zu-Ende-Verschlüsselung eingesetzt. Diese wiederum nutzt nicht die Vorteile der algebraischen Sicherheit aus und ist nur genauso effizient wie bei herkömmlicher Paketvermittlung. Dies motivierte die Entwicklung von Verfahren, welche eine leichtgewichtige Sicherheit versprechen, indem sie die algebraische Sicherheit ausnutzen.
Ziel der Arbeit ist diese leichtgewichtigen Verfahren zu untersuchen und zu prüfen, ob diese effizienter als herkömmliche Ende-zu-Ende-Verschlüsselung sind. Dabei soll neben der Effizienz auch die Sicherheit der Verfahren im Mittelpunkt stehen. Zusätzlich wird die Optimierung der bestehenden Verfahren ein Anliegen der Arbeit sein, damit der Nutzen der sicheren Netzwerkcodierung die Kosten noch stärker überwiegt.
Daher ist die Dissertation wie folgt aufgebaut. Kapitel 2 liefert zunächst eine umfassende und thematisch gegliederte Übersicht über den Stand des Gebiets rund um Netzwerkcodierung und deren Sicherheit. Danach folgen zwei Kapitel zur Thematik der Netzwerkcodierung mit Schutz der Integrität. In Kapitel 3 wird aus theoretischer Sicht der Mindestaufwand für den Schutz der Integrität untersucht. In Kapitel 4 wird zusätzlich von Angriffen auf die Kommunikation ausgegangen und die entsprechende Leistung und Effizienz in Simulationen untersucht. Anschließend folgt Kapitel 5 zum Thema der Vertraulichkeit mit den Schwerpunkten effiziente Netzwerkcodierungs-Verfahren, deren Sicherheit und die Anwendung bei verteilten Speichern. Zum Schluss gibt es in Kapitel 6 eine kurze Zusammenfassung der Ergebnisse und einen Ausblick auf weitere mögliche Schritte.

2 Stand des Arbeitsgebiets

2.1 Netzwerkcodierung

2.1.1 Grundlagen

Die Grundlagen für die Netzwerkcodierung setzten Ahlswede et. al in ihrer wegweisenden Arbeit „Network Information Flow“ [ACLY00]. Die Hauptidee bestand darin, dass man den Informationsfluss innerhalb eines Netzwerks für mehrere Empfänger maximieren kann, indem man die Daten an Zwischenknoten nicht einfach weiterleitet (Store-and-Forward), sondern miteinander kombiniert (Compute-and-Forward). Um die genaue Theorie dahinter zu erläutern, müssen zuerst einmal die grundlegenden Annahmen dargelegt werden.
Ein Netzwerk kann als Graph G abstrahiert werden. Dabei besteht $\mathsf{G} = (\mathbf{V}, \mathbf{E})$ aus einer Menge von Knoten $\mathbf{V}$ und einer Menge von gerichteten Kanten $\mathbf{E}$. Die Menge der Knoten $\mathbf{V}$ besteht für eine Übertragung aus einer Quelle (Sender) s, einer Menge von Zwischenknoten $\mathbf{F}$ und einer Menge von Senken (Empfänger) $\mathbf{R}$. Dabei gelte, dass $\{\{\mathsf{s}\}, \mathbf{F}, \mathbf{R}\}$ eine Partition von $\mathbf{V}$ darstellt. Anders ausdrückt gelte, dass $\{\mathsf{s}\}, \mathbf{F}$ und $\mathbf{R}$ paarweise disjunkte Mengen sind, jedes Element von $\mathbf{V}$ aber auch in einer der 3 Mengen enthalten sein muss. Die Kanten $\mathbf{E}$ seien der Einfachheit halber ohne unterschiedliche Gewichtung, d. h. gleich gewichtet mit 1. Eine beliebige Kante $\mathsf{e}_{ij} \in \mathbf{E} = (\mathsf{V}_i, \mathsf{V}_j)$ mit $i \neq j$ beschreibt eine Verbindung von Knoten V_i zu Knoten V_j, die für die Übertragung von Nachrichten benutzt werden kann.
Nun seien die Daten, die von s an alle $\mathsf{r}_j \in \mathbf{R}$ übertragen werden müssen, eine Menge von Nachrichten $\boldsymbol{m}$. Unter der Annahme, dass eine Nachricht $m_i \in \boldsymbol{m}$ über eine Kante in einer Zeiteinheit übertragen werden kann, lässt sich mithilfe des Satzes vom maximalen Fluss und minimalen Schnitts der Informationsfluss – oder anders ausgedrückt der Durchsatz – ermitteln. Vereinfacht ausgedrückt sagt der Satz aus, dass der maximale Fluss, also die Nachrichten, die pro Zeiteinheit übermittelt werden können, gleich dem minimalen Schnitt des Netzwerks ist. Dieser minimale Schnitt lässt sich ermitteln, indem man versucht, mit dem Entfernen von möglichst wenig Kanten $\mathbf{e}_{ij}$ von G das Netzwerk in 2 nicht miteinander verbundene Teilmengen zu trennen, wobei s und r_j nicht in der gleichen Teilmenge vorkommen dürfen.
Ein Beispiel für ein Netzwerk ist in Abbildung 2.1 dargestellt. Es handelt sich dabei um das Butterfly-Netzwerk. Unter der Annahme, dass nur Unicast-Verbindungen betrachtet werden, bei denen es nur einen Sender und einen Empfänger gibt, lässt sich ermitteln, dass der minimale Schnitt und damit der maximale Fluss zwischen s und r_1 bzw. zwischen s und r_2 gleich 2 ist. Das heißt im Beispiel, dass die Nachrichten a und b gleichzeitig und in einer Zeiteinheit von s zu r_1 oder zu r_2 übertragen werden können.
Im Multicast-Fall, bei dem einen Sender und mehrere Empfänger gibt, ist der maximale Fluss allerdings nicht auf herkömmlichem Wege zu erreichen. Knoten $\mathsf{f}_{2,1}$ muss sich entscheiden, ob er Nachricht a oder b weiterleitet. Dadurch kommt es bei einem der beiden Empfänger zu einem Informationsfluss von nur einer Nachricht. In [ACLY00] wird nun allerdings gezeigt, dass der maximale Fluss auch in Multicast-Szenarien möglich ist, sofern man den Knoten erlaubt, Berechnungen auf den Paketen durchzuführen. Abbildung 2.2 stellt eine mögliche Lösung des Problems dar. $\mathsf{f}_{2,1}$ führt eine XOR-Operation, d. h. ei-

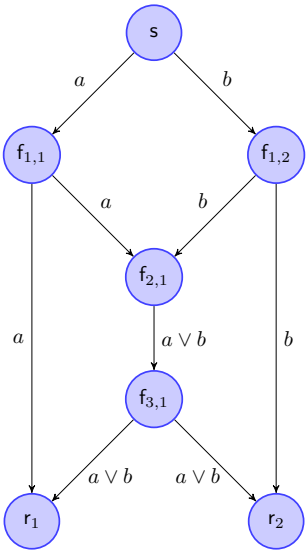


Abbildung 2.1: Übertragung ohne Netzwerkcodierung. Zwischenknoten $\mathsf{f}_{2,1}$ muss sich entscheiden, ob er Nachricht a oder b überträgt. Dadurch erhält r_1 oder r_2 nur eine der beiden Nachrichten.

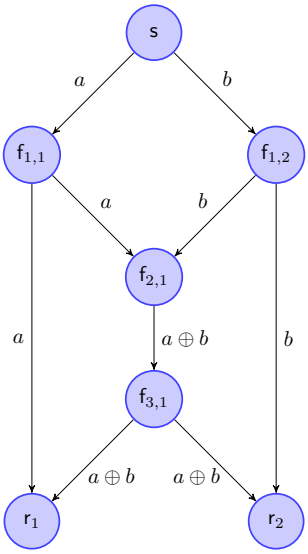


Abbildung 2.2: Übertragung mit Netzwerkcodierung. Zwischenknoten $\mathsf{f}_{2,1}$ kombiniert Nachrichten a und b mit der XOR-Operation. Dadurch können r_1 und r_2 die fehlende Nachricht wiederherstellen und besitzen dann beide Nachrichten a und b.

ne binäre Verknüpfung mit dem exklusiven ODER, durch. Die Empfängerknoten können die fehlende Nachricht decodieren, indem r_1 und r_2 jeweils ihre auf dem anderen Pfad empfangene Nachricht mit der Kombination nochmals mit XOR verknüpfen.

Dieses bahnbrechende Resultat wurde daraufhin in mehreren Arbeiten aufgegriffen und weiter entwickelt. Wichtig waren vor allem die theoretischen Betrachtungen am Anfang, die zeigten, welche Verknüpfungen optimale Ergebnisse erzielten. In „An Algebraic Approach to Network Coding“ [KM03] konnte gezeigt werden, dass die Netzwerkcodierung einen Zusammenhang zu algebraischen Codes hat, die in der Codierungstheorie benutzt werden. Außerdem wurden in dieser Arbeit Kapazitätsgrenzen untersucht und gezeigt, dass sich diese algebraischen Codes aufgrund ihrer starken mathematischen Grundlagen in der Wissenschaft sehr gut für die Netzwerkcodierung eignen.

Im selben Jahr beschäftigte sich auch die Arbeit „Linear Network Coding“ [LYC03] mit der Untersuchung optimaler Codes für Multicast-Übertragungen. In der Arbeit kamen Li et. al zu der Erkenntnis, dass lineare Codes ausreichend sind, um optimale Kapazität zu erreichen. Dazu konstruierten sie lineare Codes für Multicast-Übertragungen und zeigten, dass diese optimal sind.

Lineare Codes sind hierbei eine Untergruppe von algebraischen Codes, also Codes, welche auf algebraischen Strukturen basieren. Lineare Codes zeichnen sich dadurch aus, dass die Codewörter, welche eine endliche Länge n besitzen, Elemente eines Vektorraums $\mathbb{F}_q^n$ sind. Das heißt, jede Koordinate des Vektorraums (auch Symbol eines Codeworts genannt) ist ein Element eines endlichen Körpers $\mathbb{F}_q$. Daraus folgt, dass jedes Element aus $\mathbb{F}_q$ bezüglich der Addition als auch der Multiplikation genau ein inverses Symbol hat. Dadurch ist sichergestellt, dass ein Empfänger die Codewörter auch wieder decodieren kann. Im Übrigen gibt es genau zwei Klassen von endlichen Körpern: Primkörper (q prim) sowie Erweiterungskörper ($q = p^k$ wobei p prim), sodass $\mathbb{F}_q$ genau dann ein endlicher Körper ist, wenn $q = p^k$ mit p prim und $k \in \mathbb{N} \setminus \{0\}$.

In Abbildung 2.2 wurde die XOR-Verknüpfung angewandt, die auch als Addition über $\mathbb{F}_2$ gesehen werden kann. Man sieht, dass dieser kleine Körper nur wenige Möglichkeiten für Linearkombinationen bietet. Je mehr Elemente im Körper zur Verfügung stehen, umso mehr Möglichkeiten für eine Linearkombination gibt es und umso weniger lineare Abhängigkeiten zwischen diesen Möglichkeiten bestehen bei zufälligem Auswählen. Allerdings hat das Erhöhen der Körpergröße auch Auswirkungen auf den Berechnungsaufwand, sodass q auch nicht zu groß für eine effiziente Berechnung sein sollte.

Die Frage, wie groß nun der zugrunde liegende Körper $\mathbb{F}_q$ sein sollte oder muss, lässt sich nicht eindeutig beantworten. In der Arbeit „Network Coding for the Internet and Wireless Networks“ [CW07] wird die Netzwerkcodierung praktisch umgesetzt und ausgewertet. Es wird dabei neben dem Aufzeigen der Vorteile – wie erhöhtem Durchsatz, geringerem Energieverbrauch und niedriger Latenz – auch auf die Körpergröße q eingegangen. Es wurde gezeigt, dass eine Körpergröße von 2^8 ausreichend für eine gute Decodierbarkeit ist. Allerdings kann je nach Anwendungsfall eine kleinere Körpergröße bei rechentechnisch schwächeren Knoten sinnvoll sein. Die Autoren sehen viele Anwendungsmöglichkeiten, z. B. bei Dateidownloads, verteiltem Speicher oder Kommunikation in vermaschten Funknetzwerken. Dabei ist die Körpergröße nicht nur für die Decodierbarkeit (also die Kommunikationslast), sondern auch für den Speicherbedarf und die Geschwindigkeit (Berechnungsaufwand) entscheidend. Weiterhin gibt es verschiedene Arten der Netzwerkcodierung, auf die im nächsten Abschnitt eingegangen werden soll.

2.1.2 Verschiedene Arten der Netzwerkcodierung

Bisher wurde hauptsächlich auf die mögliche Codierung mit linearen Codes und den zugrunde liegenden endlichen Körper eingegangen. Nun soll es um die Varianten und Möglichkeiten gehen, wie die Netzwerkcodierung eingesetzt werden kann. Einen schnellen Überblick bietet auch die Arbeit „Network Coding: An Instant Primer“ [FWB06]. Die Autoren gehen dabei neben der grundsätzlichen Funktionsweise vor allem auf Anwendungsmöglichkeiten ein.

Die Idee von RLNC (Random Linear Network Coding – Lineare Netzwerkcodierung mit zufälligen Koeffizienten) ist, dass Zwischenknoten die Kombination eingehender Nachrichten nicht mit festen oder von einer zentralen Instanz vorgegebenen Koeffizienten berechnen, sondern diese zufällig wählen und anschließend den nachfolgenden Knoten mitteilen, damit eine Decodierung beim Empfänger möglich ist. Der Vorteil dabei ist, dass es keine Flaschenhälse wie zentrale Instanzen gibt und damit die Codierungen schneller ausgeführt werden können, die Größe des Netzwerks unproblematisch ist und Knoten dynamisch dazukommen oder gehen können.

Die Vorteile des RLNC wurden bereits 2003 in „The Benefits of Coding over Routing in a Randomized Setting“ [HKM$^+$03] aufgezeigt. Dabei wurden die Raten für eine erfolgreiche Decodierung in Abhängigkeit der Körpergröße q untersucht und gezeigt, dass diese für $q = 2^8$ höher als bei herkömmlicher Paketvermittlung mit zufälligem Routing sind.

Allerdings erreicht man bei zufälliger Wahl der Koeffizienten keine garantierte Decodierbarkeit beim Empfänger. Der Grund liegt im endlichen Vektorraum $\mathbb{F}_q^n$ der Codewörter. Mathematisch gesehen bilden alle gültigen Codewörter, also alle durch lineare Kombinationen erzeugbare Codewörter, einen h-dimensionalen Untervektorraum $\mathbb{U}$. Bildet die lineare Hülle der h Linearkombinationen den Untervektorraum $\mathbb{U}$, dann sind alle diese h Linearkombinationen innovativ. Anders ausgedrückt bedeutet dies, dass jedes Codewort genau dann den Rang der linearen Hülle der Linearkombinationen erhöht, wenn es innovativ ist. Lässt sich also ein Codewort durch bereits erhaltene Codewörter darstellen, so ist dieses nicht-innovativ und führt dazu, dass mehr als h Codewörter nötig sind, um $\mathbb{U}$ zu erzeugen und damit die Decodierbarkeit zu gewährleisten.

Je kleiner der zugrunde liegende endliche Körper $\mathbb{F}_q$ ist, umso geringer sind die Auswahlmöglichkeiten und damit die Anzahl der möglichen Koeffizienten, sodass es häufiger zu nicht-innovativen Paketen kommt. Eine andere häufige Darstellung ist, die Vektoren (Codewörter) als Zeilen in einer Matrix $\boldsymbol{G}$ aufzufassen. Jede hinzukommende Zeile, die den Rang der Matrix $\mathrm{rg}(\boldsymbol{G})$ erhöht, ist innovativ. Neue Zeilen, für die das nicht zutrifft, sind nicht-innovativ.

Neben der zufälligen Netzwerkcodierung gibt es auch die deterministische Netzwerkcodierung, die für spezielle Fälle sinnvoll sein kann. In „Deterministic Network Coding by Matrix Completion“ [HKM05] wird ein Verfahren gezeigt, das einen optimalen Netzwerkcode (d. h. mit optimaler Decodierbarkeit) bei kleiner Körpergröße erzeugt. Dabei wird ein Algorithmus verwendet, der versucht, beim Erstellen von Paketen dafür zu sorgen, dass der Rang der verschiedenen Empfangsmatrizen maximiert wird. Der Vorteil gegenüber der zufälligen Netzwerkcodierung ist, dass die Decodierbarkeit planbar und optimal ist und dass die Körpergröße sehr klein sein kann. Letzteres hat Vorteile bei dem Berechnungsaufwand für die Kombination und das Decodieren. Allerdings fällt die Flexibilität (Änderungen an der Topologie) und Dezentralität der zufälligen Variante weg. Weiterhin kann durch Optimierungen bei den Berechnungen durch z. B. Look-up-Tables der erhöhte Aufwand für größere Körper fast vernachlässigt werden.

In den meisten Arbeiten wird heute zufällige Netzwerkcodierung genutzt, zum einen wegen der bereits erwähnten Vorteile, zum anderen, weil es für statische Szenarien, bei denen ein optimaler deterministischer Code erzeugt werden könnte, auch ein vergleichbar guter Code mittels herkömmlicher Codierungstheorie, z. B. mittels Reed-Solomon-Code, erzeugt werden könnte. Weiterhin ist das Problem mit der schlechteren Decodierbarkeit für realistische Körpergrößen, wie z. B. $q = 2^8$, bezogen auf die Gesamtheit der Paketausfälle meistens zu vernachlässigen. Auch die zusätzlichen Daten, wie die Codierungskoeffizienten und der Berechnungsaufwand, sind eher marginal, sodass im Folgenden nur noch die zufällige Netzwerkcodierung betrachtet wird.

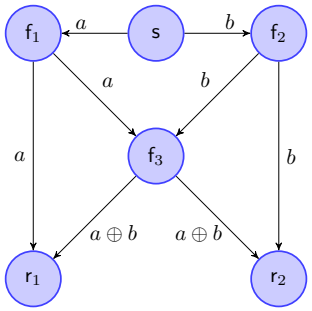


Abbildung 2.3: Intra-Session-Netzwerkcodierung. Die Nachrichten a und b von genau einem Sender s werden miteinander codiert. Da die Empfänger r_1 und r_2 genügend Linearkombinationen der Generation erhalten haben, können sie decodieren.

Nachdem Möglichkeiten vorgestellt wurden, wie kombiniert werden kann, sollen nun Möglichkeiten aufgezeigt werden, welche Nachrichten miteinander kombiniert werden können. Grob unterscheidet man zwei Kategorien. Die Intra-Session-Netzwerkcodierung und die Inter-Session-Netzwerkcodierung. Bei der Intra-Session-Netzwerkcodierung werden Nachrichten oder Pakete ausgehend von einer Session, d. h. einer Übertragung von einem Sender s zu einer Gruppe von Empfängern $\mathbf{R}$, miteinander codiert. In Abbildung 2.3 wird dieser Sachverhalt dargestellt.

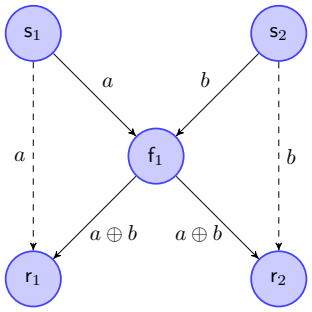


Abbildung 2.4: Inter-Session-Netzwerkcodierung. Nachrichten a und b von mehreren Sendern s_1 und s_2 werden miteinander verknüpft. Die gestrichelten Pfeile stellen das Mithören der Empfänger r_1 und r_2 bei der kabellosen Übertragung dar. Dadurch können sie die einzelnen Pakete decodieren.

Bei der Inter-Session-Netzwerkcodierung werden Nachrichten verschiedener Sessions und damit im Allgemeinen auch Nachrichten verschiedener Sender miteinander kombiniert. Dazu nimmt man an, dass Knoten durch Mithören auf dem drahtlosen Kanal zusätzliche Daten erhalten, um decodieren zu können. Abbildung 2.4 stellt diese Variante der Kombination dar.

Ein Beispiel für die Intra-Session-Netzwerkcodierung ist das MORE-Verfahren, dass in „Trading Structure for Randomness in Wireless Opportunistic Routing“ [CJKK07] vorgestellt wurde. Dabei wird eine zusätzliche Schicht zwischen Sicherungsschicht (MAC – Media Access Control) und Vermittlungsschicht (IP – Internet Protocol) in den Protokollstapel eingebracht. Übertragungen werden im MORE-Protokoll nur Ende-zu-Ende, also vom Empfänger zum Sender, bestätigt. Die Zwischenknoten agieren nicht selbstständig, sondern alle Übertragungen (auch erneute Übertragungen aufgrund von Verlusten) werden vom Sender initiiert. Die Berechnungen finden im endlichen Körper $\mathbb{F}_{2^8}$ statt. Die Autoren zeigten in dem in der Arbeit gewählten Testfall, dass mittels Intra-Session-Netzwerkcodierung ein höherer Durchsatz als bei herkömmlichem Routing erzielt werden kann.

Ein Protokoll für die Inter-Session-Netzwerkcodierung ist COPE, welches in der Arbeit „XORs in The Air: Practical Wireless Network Coding“ [KRH+06] erläutert wird. Es ist eine der ersten praktischen Umsetzungen der sogenannten opportunistischen Netzwerkcodierung. Ähnlich zu [CJKK07] wird auch hier eine zusätzliche Schicht zwischen der MAC- und IP-Schicht eingezogen. Die opportunistische Netzwerkcodierung umfasst das Mithören, das Codieren sowie das Melden, welche Pakete die Kommunikationspartner bereits haben. Zusätzlich werden die Übertragungen von (Zwischen-)Knoten zu (Zwischen-)Knoten bestätigt. Allerdings werden die Berechnungen, damit das Mithören und wieder Decodieren möglich ist, nur im Körper $\mathbb{F}_2$ durchgeführt. Insgesamt konnten die Autoren mit COPE einen Leistungszugewinn erzielen.

Weiterhin gab es natürlich auch Ambitionen, beide Varianten miteinander zu vereinen. Den ersten praktischen Versuch gab es mit „I^{2}NC: Intra- and Inter-Session Network Coding for Unicast Flows in Wireless Network“ [SMR11]. Basierend auf dem COPE-Protokoll, welches zwischen MAC- und IP-Schicht agiert, wird eine Intra-Session-Netzwerkcodierung über der IP-Schicht hinzugefügt, um beide Varianten in einer Umsetzung zu haben. Dabei werden die guten Fehlerkorrektureigenschaften der Intra-Session-Netzwerkcodierung gepaart mit der höheren Decodierbarkeit durch das Mithören in lokalen Bereichen (COPE) beschleunigt. Die Autoren zeigten die Vorteile der Kombination über eine Auswertung in einer Simulationsumgebung auf.

Ein alternatives, aber ähnliches Konzept hat auch CORE, welches in der Arbeit „CORE: COPE with MORE in Wireless Meshed Networks“ [KHH+13] erläutert wird. Bei CORE ist allerdings das Protokoll mehr als ein kombiniertes Protokoll anstatt zwei übereinander liegender Protokolle wie bei [SMR11] zu betrachten. Die Hauptidee ist aber auch hier, Intra-Session-Netzwerkcodierung (MORE) für die Übertragungen zu nutzen und in Regionen, wo Mithören von Paketen möglich ist, noch zusätzlich über die Inter-Session-Netzwerkcodierung den Durchsatz zu optimieren. Ein weiteres Augenmerk wird hier auch auf den Energieverbrauch und mögliche Einsparungen gelegt.

Insgesamt gibt es also Möglichkeiten, beide Technologien miteinander zu verbinden. Diese Arbeit befasst sich im Folgenden mit der Intra-Session-Netzwerkcodierung. Zum einen ist es bei dieser unerheblich, welches Übertragungsmedium genutzt wird, während bei der Inter-Session-Netzwerkcodierung nur Funknetze das Mithören ermöglichen. Zum anderen gibt es bei der Intra-Session-Netzwerkcodierung keine Einschränkungen bezüglich des Körpers, in dem gerechnet wird. Das hat für die sicheren Verfahren, die später vorgestellt werden, den Vorteil, dass sie auch sinnvoll angewendet werden können. Im Körper $\mathbb{F}_2$ sind fast alle Verfahren, die Sicherheit gewähren sollen, unsicher. Der nächste Teil befasst sich allerdings erst einmal mit der praktischen Umsetzung der Netzwerkcodierung.

2.1.3 Praktische Umsetzungen

Im folgenden Abschnitt soll es nun um die praktische Umsetzung von RLNC gehen. Die erste Arbeit dazu wurde mit „Practical Network Coding“ [CWJ03] veröffentlicht. Neben den technischen Betrachtungen zu Pufferspeichern oder der Pfadauswahl sind zwei Punkte besonders wichtig: zum einen die Frage, wie man, nachdem man mit zufälligen Koeffizienten (re)codiert hat, die Nachrichten am Empfänger wieder decodieren kann, zum anderen, daraus resultierend, wie das Paketformat aussehen soll, mit dem die Daten übermittelt werden können. Diese praktische Umsetzung und das Paketformat dienen als Grundlage aller weiteren Betrachtungen.

Die Daten, die übermittelt werden sollen, werden in Pakete $\boldsymbol{x}_j$ der Länge n unterteilt. Uncodierte Pakete ohne Koeffizienten sollen im Folgenden als native Pakete bezeichnet werden. Durch den zugrunde liegenden Körper $\mathbb{F}_{2^8}$ entspricht jedes Körperelement genau einem Byte der Daten. Genau h native Pakete formen dann eine Generation $\boldsymbol{G}$, innerhalb welcher Pakete miteinander codiert werden können. Dieser Parameter wird auch Generationsgröße h genannt. Jedes Paket wird mittels eindeutiger Generations-ID o einer Generation zugeordnet. Da die Generationen nacheinander oder verschachtelt (Pipelining) übertragen werden können, soll im Folgenden nur die Übertragung einer Generation betrachtet werden. Die Matrix $\boldsymbol{G}$ ist also ein Teil der Daten, wobei jede Zeile von $\boldsymbol{G}$ ein natives Paket $\boldsymbol{x}_j$ mit n Symbolen darstellt.

Nun soll verdeutlicht werden, wie die lokalen Koeffizienten übermittelt werden. Die Idee dahinter ist, dass man in jedes Paket den globalen Codierungsvektor $\boldsymbol{\beta}_j$ mit $\beta_{i,j} \in \mathbb{F}_{2^8}$ der Länge h integriert, indem man ihn vor die Daten $\boldsymbol{x}_j$ schreibt. Der Codierungsvektor zeigt dann an, aus welcher Kombination der h nativen Pakete ein codiertes Paket $\overline{x}_j$ besteht. Ein Paket besteht so aus zwei Teilen: $(\boldsymbol{\beta}_j|\boldsymbol{x}_j)$.

Initial (uncodiert) besteht ein Paket aus genau einem nativen Paket, weshalb $\boldsymbol{\beta}_j$ ein Nullvektor ist, bis auf $\beta_{j,j} = 1$ für $1 \leq j \leq h$. Für eine komplette uncodierte Generation $\boldsymbol{G}_{\mathsf{u}}$ ist dies in Formel (2.1) dargestellt:

$$\boldsymbol{G}_{\mathsf{u}} = \begin{pmatrix} \beta_{1,1} = 1 & \cdots & \beta_{1,h} = 0 & x_{1,1} & x_{1,2} & \cdots & x_{1,n} \\ \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \beta_{h,1} = 0 & \cdots & \beta_{h,h} = 1 & x_{h,1} & x_{h,2} & \cdots & x_{h,n} \end{pmatrix}. \tag{2.1}$$

Das (Re)Codieren erfolgt dann über eine Linearkombination von verfügbaren Paketen. Formel (2.2) zeigt eine codierte Generation $\boldsymbol{G}_{\mathsf{c}}$ mit η Paketen, wobei Formel (2.3) die Berechnung verdeutlicht. Eine codierte Generation kann auch mehr als h Zeilen enthalten, die aus Redundanzgründen verschickt werden können. Es gilt jedoch, dass der aufgespannte Untervektorraum aller codierten Vektoren maximal die Dimensionalität der Generationsgröße erreicht. Daher ist der Rang (rg) der Matrix $\boldsymbol{G}_{\mathsf{c}}$ unabhängig von $\eta \geq h$ und es gilt $\mathrm{rg}(\boldsymbol{G}_{\mathsf{c}}) \leq h$.

$$\boldsymbol{G}_{\mathsf{c}} = \begin{pmatrix} \overline{\beta}_{1,1} & \cdots & \overline{\beta}_{1,h} & \overline{x}_{1,1} & \overline{x}_{1,2} & \cdots & \overline{x}_{1,n} \\ \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \overline{\beta}_{\eta,1} & \cdots & \overline{\beta}_{\eta,h} & \overline{x}_{\eta,1} & \overline{x}_{m,2} & \cdots & \overline{x}_{\eta,n} \end{pmatrix}, \tag{2.2}$$

$$\text{mit } \overline{x}_{i,j} = \sum_{k=1}^{h} \overline{\beta}_{i,k} \cdot x_{k,j} \qquad \text{z. B.: } \overline{x}_{1,2} = \overline{\beta}_{1,1} \cdot x_{1,2} + \cdots + \overline{\beta}_{1,h} \cdot x_{h,2}. \tag{2.3}$$

Da Zwischenknoten wie Sender codieren können und es durch dieses Paketformat egal ist, ob die Pakete bereits codiert wurden oder nicht, lässt sich mithilfe eines beliebigen zufälligen lokalen Vektors $\boldsymbol{\alpha}$ in Länge der Anzahl k bereits empfangener Pakete $\boldsymbol{y}_j = (\overline{\boldsymbol{\beta}}_j | \overline{\boldsymbol{x}}_j)$ eine ausgehende Nachricht $\overline{\boldsymbol{y}}$ erzeugen, wobei gilt:

$$\overline{\boldsymbol{y}} = \sum_{j=1}^{k} \alpha_j \cdot \boldsymbol{y}_j. \tag{2.4}$$

Durch die Struktur der Pakete wird die aktuelle Codierung eines jeden Pakets im globalen Codierungsvektor, also in den ersten h Elementen eines Pakets, ersichtlich. Bleibt also nun noch zu klären, wie der Empfänger nach dem Empfang der Nachrichten wieder zu den nativen Paketen durch Decodierung kommt.

Da alle Operationen in einem endlichen Körper $\mathbb{F}_q$ stattfinden, gilt, dass jedes Element, bis auf das 0-Element, ein eindeutiges multiplikatives Inverses hat. Unter der Annahme, dass der Empfänger h innovative Pakete erhalten hat, kann dieser die inverse Matrix $\boldsymbol{B'}^{-1}$ zu der globalen Codierungsmatrix (Global Encoding Matrix – GEM) aus den globalen Codierungsvektoren (Global Encoding Vector – GEV) $\boldsymbol{B'}$ bilden. Diese inverse Matrix $\boldsymbol{B'}^{-1}$ multipliziert mit der Matrix aus empfangenen Paketen $\boldsymbol{Y}$ ergibt dann eine Einheitsmatrix $\mathbf{1}$ und die nativen Pakete $\boldsymbol{X}$. Der Weg der Daten vom Sender über Zwischenknoten zu den Empfängern sei im Folgenden in Matrixschreibweise verdeutlicht:

$$\underbrace{\boldsymbol{B}}_{\text{codieren}} * \left[\underbrace{\mathbf{1}}_{\text{GEM}} | \underbrace{\boldsymbol{X}}_{\text{Daten}}\right] = \left[\underbrace{\boldsymbol{B}}_{\text{GEM}} | \underbrace{\boldsymbol{BX}}_{\text{codierte Daten}}\right] \Rightarrow \underbrace{\boldsymbol{A}}_{\text{recodieren}} * \left[\underbrace{\boldsymbol{B}}_{\text{GEM}} | \underbrace{\boldsymbol{BX}}_{\text{codierte Daten}}\right] = \left[\underbrace{\boldsymbol{AB}}_{\text{GEM}} | \underbrace{\boldsymbol{ABX}}_{\text{codierte Daten}}\right]$$

$$\ldots \quad \underbrace{\boldsymbol{Y}}_{\text{empfangene Pakete}} = \left[\underbrace{\boldsymbol{B'}}_{\text{GEM}} | \underbrace{\boldsymbol{B'X}}_{\text{codierte Daten}}\right] \Rightarrow \underbrace{\boldsymbol{B'}^{-1}}_{\text{decodieren}} * \left[\underbrace{\boldsymbol{B'}}_{\text{GEM}} | \underbrace{\boldsymbol{B'X}}_{\text{codierte Daten}}\right] = \left[\underbrace{\mathbf{1}}_{\text{GEM}} | \underbrace{\boldsymbol{X}}_{\text{Daten}}\right].$$

Die Decodierung, also das Bilden von $\boldsymbol{B'}^{-1}$ erfolgt z. B. über den Gauß-Jordan-Algorithmus. Dabei muss man allerdings nicht erst warten, bis eine vollständige Generation, also genau h innovative Pakete, empfangen wurden. Es ist ebenfalls möglich, k von h Paketen so umzuformen, dass jeweils k Stellen pro Paket in der Einheitsmatrixform vorliegen. Dadurch verringert sich die Verzögerung durch die Decodierung am Ende, da paketweise in eine Zwischenform „decodiert“ werden kann. Dadurch kann die Latenz der Übertragung minimiert werden.

Im Folgenden sei ein Beispiel des erweiterten (hier mit Einheitsmatrix auf der rechten Seite) Gauß-Jordan-Algorithmus mit partieller Lösung gezeigt. Für die vereinfachte Darstellung sei der zugrunde liegende Körper im folgenden Beispiel der Primzahlkörper $\mathbb{F}_{13}$. Zu Beginn wurden 2 von 3 Pakete empfangen.

$$\begin{array}{ccc|ccc} 3 & 8 & 2 & 1 & 0 & 0 \\ 4 & 12 & 9 & 0 & 1 & 0 \end{array} \overset{*3^{-1}}{\Rightarrow} \begin{array}{ccc|ccc} 1 & 7 & 5 & 9 & 0 & 0 \\ 4 & 12 & 9 & 0 & 1 & 0 \end{array} \overset{*4}{\Rightarrow} \begin{array}{ccc|ccc} 1 & 7 & 5 & 9 & 0 & 0 \\ 0 & 3 & 11 & 10 & 12 & 0 \end{array}$$

Zu Beginn wird die erste Spalte „normalisiert“, d. h. wie zu einer Einheitsmatrix umgeformt. Dazu wird die Zeile selbst mit dem Inversen des jeweils ersten Elements multipliziert. Danach wird die entstehende Zeile mit den Koeffizienten an erster Stelle in den anderen Zeilen multipliziert und anschließend die Differenz aus den Zeilen gebildet. Dadurch entsteht eine Zeile mit Nullelement als ersten Koeffizienten.

$$\begin{array}{ccc|ccc} 1 & 7 & 5 & 9 & 0 & 0 \\ 0 & 3 & 11 & 10 & 12 & 0 \end{array} \overset{}{\underset{*3^{-1}}{\Rightarrow}} \begin{array}{ccc|ccc} 1 & 7 & 5 & 9 & 0 & 0 \\ 0 & 1 & 8 & 12 & 4 & 0 \end{array} \overset{-}{\underset{*7}{\Rightarrow}} \begin{array}{ccc|ccc} 1 & 0 & 1 & 3 & 11 & 0 \\ 0 & 1 & 8 & 12 & 4 & 0 \end{array}$$

Das Gleiche gilt auch für die zweite und alle folgenden Zeilen. Erst wird das Element j in Zeile j zu 1 umgeformt, indem mit dem inversen Element multipliziert wird. Danach werden alle anderen Zeilen an Stelle j auf 0 gebracht. Kommt nun eine neue Zeile – hier z. B. durch den Empfang eines neuen Pakets – hinzu, sind zusätzliche Schritte notwendig.

$$\begin{array}{ccc|ccc} 1 & 0 & 1 & 3 & 11 & 0 \\ 0 & 1 & 8 & 12 & 4 & 0 \\ 9 & 6 & 9 & 0 & 0 & 1 \end{array} \Rightarrow \begin{array}{ccc|ccc} 1 & 0 & 1 & 3 & 11 & 0 \\ 0 & 1 & 8 & 12 & 4 & 0 \\ 0 & 7 & 0 & 1 & 8 & 12 \end{array} \Rightarrow \begin{array}{ccc|ccc} 1 & 0 & 1 & 3 & 11 & 0 \\ 0 & 1 & 8 & 12 & 4 & 0 \\ 0 & 0 & 4 & 5 & 7 & 1 \end{array}$$

Zuerst müssen die ersten Spalten, die bereits in „Normalform“ waren, in der neuen Zeile zu 0 umgeformt werden. Das passiert durch die Differenzbildung mit einem Vielfachen der oberen normalisierten Zeilen.

$$\begin{array}{ccc|ccc} 1 & 0 & 1 & 3 & 11 & 0 \\ 0 & 1 & 8 & 12 & 4 & 0 \\ 0 & 0 & 4 & 5 & 7 & 1 \end{array} \Rightarrow \begin{array}{ccc|ccc} 1 & 0 & 1 & 3 & 11 & 0 \\ 0 & 1 & 8 & 12 & 4 & 0 \\ 0 & 0 & 1 & 11 & 5 & 10 \end{array} \Rightarrow \begin{array}{ccc|ccc} 1 & 0 & 0 & 5 & 6 & 3 \\ 0 & 1 & 0 & 2 & 3 & 11 \\ 0 & 0 & 1 & 11 & 5 & 10 \end{array}$$

Danach wird wie zuvor vorgegangen. Zuerst wird das Element j in Zeile j auf 1 umgeformt. Im Beispiel wird daher Zeile 3 mit 4^{-1} multipliziert. Anschließend wird diese Zeile mit dem Koeffizienten in Spalte j einer anderen Zeile multipliziert und das Ergebnis von der anderen Zeile abgezogen. Dadurch werden alle anderen Zeilen so umgeformt, dass der Koeffizient in dieser Spalte 0 wird. Am Ende erhält man die Einheitsmatrix auf der linken Seite. Die rechte Seite entspricht nun der inversen Matrix zu der ursprünglichen linken Seite.

Der Aufwand für alle Berechnungen von RLNC wurde in „Polynomial Time Algorithms For Network Information Flow“ [SET03] untersucht. Dabei zeigte sich, dass Polynomialzeit-Algorithmen ausreichend sind, um alle Berechnungen vom Codieren, Recodieren und Decodieren zu ermöglichen. Dabei hängt der Berechnungsaufwand hauptsächlich von der Generationsgröße h ab, sodass die Autoren schlussfolgern, dass für ein realistisches (nicht zu großes) h der Berechnungsaufwand nicht die Übertragungsrate limitiert und dadurch der Durchsatz durch die Netzwerkcodierung insgesamt erhöht werden kann.

Später wurde in “Network Coding vs. Erasure Coding: Reliable Multicast in Ad Hoc Networks“ [FOG08] auch der Zusammenhang zwischen Netzwerkcodierung und Fountain Codes, die auch als ratenlose (rateless) Codes bekannt sind, hergestellt. Fountain Codes zeichnen sich dadurch aus, dass es eine sehr hohe (fast unbegrenzte) Anzahl an gültigen Codewörtern gibt und somit diese für ratenloses Codieren geeignet sind. Dabei werden bei einer Übertragung so lange wie nötig neue Codewörter verschickt, bis der Empfänger die Decodierung bestätigt. Dies hat einen Vorteil in dynamischen Netzwerken, da automatisch auf

Änderungen reagiert wird und nicht wie normalerweise die festgelegte Rate (Redundanz zu Nutzdaten) angepasst werden muss.

Da es sich bei dem in [FOG08] angenommenen Systemmodell um ein Auslöschungsnetzwerk (d. h. manche Nachrichten gehen verloren, die restlichen Nachrichten sind jedoch unverfälscht) handelt, verglichen die Autoren in der Arbeit Ende-zu-Ende-Fountain-Codes mit Netzwerkcodierung und bewerteten die Leistungsfähigkeit. Dabei stellten sie fest, dass die Netzwerkcodierung mit zwischenzeitlichem Recodieren bessere Leistungen als herkömmliche Fountain Codes erreichen.

Eine ähnliche Arbeit zum Thema ist „Combined Fountain Code with Network Coding for Error-Tolerant Transmission Network“ [YA09]. Hier wird die Netzwerkcodierung durch den ratenlosen Ansatz erweitert, um höhere Adaptivität und Fehlertoleranz zu ermöglichen. Dabei werden neue Pakete an Zwischenknoten erzeugt, bis nachfolgende Knoten genügend Pakete besitzen.

Die für die ratenlose Übertragung notwendigen Benachrichtigungen lassen sich z. B. über Acknowledgements (ACK) oder automatische Wiederholungsanfragen (ARQ) realisieren. Je nachdem, ob man besonders wenige Nachrichten oder die Daten besonders schnell übertragen möchte, ließen sich damit ratenlose Übertragungen bewerkstelligen.

Die Frage, wie man die (möglicherweise Multicast-) Flüsse im Netzwerk organisiert, beantwortet die Arbeit „Dynamic Algorithms for Multicast With Intra-Session Network Coding“ [HV09]. Neben deren Ansatz – basierend auf Gegendruck (back pressure) – zeigen die Autoren vor allem, dass die Entscheidungen zur Flusskontrolle, wie auch für die Codierungen lokal getroffen werden müssen. Dadurch ist dieser dezentrale Ansatz gut für dynamische Netzwerke geeignet und benötigt keine globalen Routinginformationen.

Es zeigt sich also auch bei der praktischen Umsetzung, dass keine zentrale Instanz notwendig ist, weder zum Austausch der Codierungsvektoren, noch zur Fehlerkorrektur oder für das Routing. Dies ist wichtig für viele Anwendungsszenarien mit nicht stabilen Netzwerken und gewährleistet gute Skalierbarkeit in wachsenden Netzwerken. Bevor auf die Effizienz und Sicherheit eingegangen wird, soll im nächsten Teil noch eine Übersicht zu Anwendungen der Netzwerkcodierung gegeben werden, da sich diese längst nicht mehr nur auf Durchsatzerhöhung beschränkt.

2.1.4 Anwendungen und Simulatoren

Durch die Eigenschaften höheren Durchsatz und robustere Übertragungen zu gewährleisten, eignet sich die Netzwerkcodierung für verschiedene Anwendungsfälle. In „Network Coding Applications“ [FS07] werden diese in verschiedene Klassen unterteilt.

Eine erste Klasse ist der Dateiaustausch zwischen verschiedenen Parteien. Dazu gehören Peer-to-Peer Systeme, also ein Netz von gleichberechtigten Rechnern, die Daten austauschen, sowie Content Distribution Networks (CDN). Bei Letzteren handelt es sich um Serversysteme, die verteilt arbeiten und Anfragen von Rechnern bezüglich Last verteilen und bearbeiten. Für diese Dateiaustauschsysteme kommt es vor allem auf effiziente Ausnutzung der Bandbreite mit so gering wie möglicher Netzwerkauslastung an. Ein Beispiel für ein solches System wird in der Arbeit „Building Scalable and Robust Peer-to-Peer Overlay Networks for Broadcasting using Network Coding“ [JLC07] aufgezeigt. Mithilfe von Overlays, also einem logischen Netz, das auf realen Verbindungen basiert, wird ein Netzwerk aufgebaut, welches durch die Anwendung von Netzwerkcodierung robuster und skalierbarer wird.

Eine zweite Klasse sind vermaschte Funknetzwerke oder Sensornetzwerke. Hier spielt zum einen eine effiziente Datenübertragung als auch die Möglichkeit von Multicasts eine Rolle. Meist sind die Sensoren oder mobilen Geräte nur leistungsschwach und dürfen nur wenig Energie verbrauchen. Das bereits erwähnte MORE-Protokoll [CJKK07] ist beispielsweise ein Verfahren, welches sich für solche Funknetze eignet.

Ein weiterer Anwendungsfall ist die Verwendung der Netzwerkcodierung als alternative Kanalcodierung für Auslöschungskanäle. Da es egal ist, welche Pakete bei dem Empfänger ankommen, solange es genügend sind, eignet sich die Netzwerkcodierung auch als Alternative zu herkömmlichen Kanalcodes wie Reed-Solomon-Codes. Ein weiterer Vorteil ist die Möglichkeit zur ratenlosen Codierung ähnlich zu Fountain Codes. Unter diese Rubrik fällt auch der bereits behandelte Vergleich von Netzwerkcodierung gegen Auslöschungscodes [FOG08].

Neben den Sicherheitsanwendungen wird auch auf die (zum Zeitpunkt der Veröffentlichung) sich neu entwickelnden Felder eingegangen. Aus heutiger Sicht besonders erwähnenswert sind dabei zum einen die On-Chip-Kommunikation, bei der besonders die Vermeidung von Flaschenhälsen durch die Netzwerkcodierung bei der Kommunikation vorteilhaft ist. Zum anderen wird auch auf die Nutzbarkeit von Netzwerkcodierung für verteilte Speicher eingegangen.

Ein Beispiel für eine verteilte Speicherlösung zeigt die Arbeit „Distributed Cloud Storage Using Network Coding“ [SFLP14]. Die Idee ist, seine Daten nicht in eine Cloud zu legen, sondern diese mittels Netzwerkcodierung auf Clouds unterschiedlicher Anbieter zu verteilen. Im Vordergrund stehen Durchsatzerhöhung und Sicherheit durch die Verteilung. Allerdings ist die Sicherheit nur sehr begrenzt, wie später erläutert werden wird. Ein anderes Beispiel ist die NCCloud, die in der Arbeit „NCCloud: A Network-Coding-Based Storage System in a Cloud-of-Clouds“ [CHLT14] vorgestellt wird. Sie funktioniert ähnlich zu [SFLP14] mit verschiedenen Anbietern. Allerdings wurde hier auch noch der Vergleich mit RAID-Systemen (Redundant Array of Independent Disks – Verfahren um Redundanz in Festplattenverbände einzubringen) gemacht und man kam zu dem Resultat, dass weniger Netzwerkverkehr im Reparaturfall (ein Anbieter oder Server fällt aus) notwendig ist.

Dies sollen nun die häufigsten Anwendungsfälle für die Netzwerkcodierung gewesen sein. Damit neue Ideen und Anwendungsfälle getestet werden können, wurden auch mehrere Simulatoren und Bibliotheken für die Simulation von Netzwerkcodierung entworfen. Es soll hier nur auf 2 Beispiele eingegangen werden.

Ein erster speziell für die Netzwerkcodierung entworfener Simulator ist NECO (NEtwork COding simulator) [Fer09]. Dieser ermöglicht es, zunächst ein Netzwerk und dessen Topologie festzulegen. Anschließend können Parameter bezüglich der Netzwerkcodierung frei gewählt werden, bevor die Simulation gestartet wird. Während der Ausführung bietet NECO eine Visualisierung der Kommunikation und zahlreiche Statistiken und Auswertungstools nach der Simulationszeit. Natürlich gibt es auch noch andere Simulatoren wie beispielsweise ns-3 [ns-17], doch leider sind die Anwendungsmöglichkeiten der verfügbaren Simulationsumgebungen für die in dieser Arbeit untersuchten Sachverhalte sehr begrenzt und erlauben nicht ohne weiteres, Verfahren zu ersetzen oder andere Szenarien als die Kommunikation zu betrachten.

Einen anderen Ansatz bietet die Bibliothek Kodo, welche in der Arbeit „Kodo: An Open and Research Oriented Network Coding Library“ [PHF11] erstmals veröffentlicht wurde. Es handelt sich dabei um eine optimierte und stetig weiterentwickelte Bibliothek für C++, die alle Funktionen, die für die Netz-

werkcodierung benötigt werden, einfach bereitstellt und effizient umsetzt. Auch für andere Sprachen, wie Python oder Javascript, sind Bindings vorhanden. Anders als ein Simulator ist Kodo mehr eine Bibliothek, bei der man auf unterschiedlichen Schichten mit der Netzwerkcodierung arbeiten kann. Vom bloßen Anwenden eines Codierers und eines Decodierers bis hin zum Modifizieren der einzelnen Körperoperationen ist alles möglich und Kodo dadurch universell einsetzbar. Auf der Gegenseite steht dafür die hohe Komplexität und bis auf einige Beispiele die Notwendigkeit des Programmierens, selbst für einfache Simulationen.
Da bereits mehrfach von Effizienz gesprochen wurde, soll es im nächsten Teil der Arbeit um Effizienzbetrachtungen gehen und aufgezeigt werden, welche Optimierungen der Netzwerkcodierung bereits vorgestellt wurden und an welchen Stellen weitere Optimierungen möglich sind.

2.2 Effizienzbetrachtungen

2.2.1 Algorithmische Betrachtungen

Dass die Netzwerkcodierung einige Effizienzverbesserungen bezüglich des Durchsatzes bringen kann, wurde bereits gezeigt. Jedoch bleibt die Frage, ob sich der Aufwand für die zusätzlichen Berechnungen und der Übertragungsmehraufwand für die Koeffizienten lohnen. Darum soll es nun um die effiziente Netzwerkcodierung gehen.
Jaggi et al. zeigten bereits in der Arbeit „Polynomial Time Algorithms for Multicast Network Code Construction'"[JLHE05], dass Polynomialzeitalgorithmen für das Codieren, Recodieren und Decodieren möglich sind. Weiterhin empfahlen sie mehrere Optimierungsmaßnahmen. Den Beginn macht ein schneller Test auf lineare Unabhängigkeit, der von allen beteiligten Knoten zum Prüfen verwendet werden kann, ob ein empfangenes Paket innovativ ist oder keinerlei neue Information enthält. Dann zeigen sie eine schnelle Konstruktion von deterministischen Netzwerkcodes bei ganzzahligen Kantenkapazitäten. Zusätzlich wurde gezeigt, dass Netzwerkcodes die Robustheit gegenüber Kantenausfällen optimieren.
In „A Network Coding Approach to Energy Efficient Broadcasting - From Theory to Practice“ [FWB06] wird der Fokus auf die Energie gelegt. Es konnte gezeigt werden, dass die Netzwerkcodierung eine höhere Energie-Effizienz als herkömmliche Verfahren aufweist. Dies ist vor allem für den Einsatz bei mobilen Geräten, also z. B. im Bereich der Ad-Hoc-Funknetzwerke, vorteilhaft.
Auch durch das ratenlose Codieren lässt sich der Durchsatz optimieren. Die Arbeit „ARQ for Network Coding“ [SSM08] bestätigt die vorteilhaften Eigenschaften der ratenlosen Netzwerkcodierung und zeigt, dass diese optimal bezüglich des dynamisch verändernden Netzwerks ist. Weiterhin wird die Idee aufgezeigt, ARQs, also automatische Wiederholungsanfragen, zu nutzen, um Feedback für die ratenlose Codierung zu erhalten.
Optimierungen bezüglich der Integration in bestehende Systeme werden in “Network Coding Meets TCP: Theory and Implementation“ [SSM$^+$11] aufgezeigt. Die Kernidee ist, dass fast alle Kommunikation über den TCP/IP-Stack abläuft und man deswegen die Netzwerkcodierung darin integrieren sollte, um geringe Kosten für die Umsetzung und höchstmögliche Kompatibilität zu erreichen. Dafür schlagen die Autoren vor, die TCP-Schicht auf Endgeräten durch eine TCP/NC-Schicht zu ersetzen, die nur geringe Änderungen aufweist und deshalb mit gängigen Netzwerkgeräten kompatibel ist. Die Änderungen umfassen eine geänderte Flusskontrolle und den Umgang mit dem Sliding Window sowie der ACK In-

terpretation. Es soll nun nicht mehr die letzte empfangene Paketnummer bestätigt werden, sondern der Rang. Das Sliding Window ist dabei die Anzahl der Pakete, die gleichzeitig geschickt werden, ohne dass eine Bestätigung zwischen den Paketen erfolgen muss. Dadurch erhöht sich der Durchsatz deutlich. Die Größe des Sliding Windows hängt von dem Empfang der Bestätigungen ab. Deswegen muss hier eine Anpassung an den Rang erfolgen. Durch die Integration in TCP eignet sich dieses Protokoll für schrittweises Ausrollen in Netzwerken und bietet gute Performance.

Dies sollen einige Beispiele für die Optimierung von Algorithmen gewesen sein, um eine höhere Effizienz zu erreichen. Ein anderes Optimierungspotenzial ist erkennbar, wenn man sich verdeutlicht, woher der zusätzliche Berechnungsaufwand kommt. Die Berechnungen für Codieren, Recodieren und Decodieren sind mathematische Operationen auf dem zugrunde liegenden endlichen Körper. Deswegen sollen im nächsten Abschnitt Möglichkeiten zur Optimierung an genau diesen Operationen aufgezeigt werden.

2.2.2 Körperoperationen

Eine Möglichkeit, den Aufwand für die Berechnungen zu senken, ist, die Operationen direkt in Hardware zu implementieren, anstatt diese in herkömmlichen CPUs zu berechnen. In „Energy-Aware Hardware Implementation of Network Coding“ [AMC11] wurde genau diese Möglichkeit beschrieben, umgesetzt und ausgewertet. Im Gegensatz zu der komplexen und aufwendigen Berechnung in Universalrechnern lässt sich ein endlicher (Erweiterungs-)Körper als lineares Schieberegister mit Rückkopplung realisieren. Die Ergebnisse der Arbeit zeigten, dass dies nicht nur mit einem niedrigen Energieverbrauch einhergeht, sondern auch eine sehr schnelle Ausführungszeit und damit niedrige Latenz ermöglicht.

Jedoch hat eine reine Hardwarelösung das Problem, dass es sich zum einen nur für einzelne Geräte lohnt, solch einen Aufwand zu betreiben und damit eine Beschränkung für die Verbreitung der Netzwerkcodierung darstellt. Zum anderen sind Hardwarelösungen nicht flexibel, wenn die Körpergröße verändert würde. Deswegen wird heute meist mit Lookup-Tabellen, zumindest für kleinere Körpergrößen bis $q < 2^8$, gearbeitet, da diese sich schnell berechnen lassen, keiner zusätzlichen Hardware bedürfen und einfach verändert werden können. Für größere Körper, wie z. B. $\mathbb{F}_{2^{16}}$, können auch Log-Lookup-Tabellen Berechnungen beschleunigen. Praktische Umsetzungen finden sich z. B. bei Kodo [PHF11].

Für größere Erweiterungskörper wie z. B. $\mathbb{F}_{2^{32}}$ gibt es allerdings keine effiziente Berechnung. Deswegen wird in „Network Coding Over The $2^{32} - 5$ Prime Field“ [PHVF13] eine Alternative aufgezeigt. Die Idee ist, dass man anstatt eines Erweiterungskörpers einen Primzahlkörper nimmt, der annähernd die gleiche Größe besitzt, aber deutlich schneller mittels einfacher Modulo-Operation berechenbar ist. Der Körper $\mathbb{F}_{2^{32}-5}$ ist so ein alternativer Körper. Das einzige Problem ist, dass mit den 32 Bit Länge eben nicht ganz 2^{32} Möglichkeiten dargestellt werden können, weshalb ein aufwendiges Datenmapping notwendig ist.

Ein Kompromiss zwischen Hardwareentwurf und Softwarelösung stellt die Arbeit „Efficient GF Arithmetic for Linear Network Coding using Hardware SIMD Extensions“ [GRU14] dar. Die Idee ist, dass heutige CPUs zusätzliche Hardwareerweiterungen, hauptsächlich für Multimedia-Zwecke, besitzen. Die SIMD-Erweiterungen aktueller ARM und x86-Prozessoren können z. B. ausgenutzt werden, um die Körperoperationen zu beschleunigen. Die Geschwindigkeit im Körper $\mathbb{F}_{2^8}$ steigt zwar deutlich gegenüber der herkömmlichen Berechnung, ist aber gegenüber der Lookup-Tabellen-Variante nur geringfügig schneller. Dafür sind die Überlegungen prinzipiell auch für größere Körper anwendbar, bei denen ebenfalls deutliche Gewinne erzielbar sind und Lookup-Tabellen nicht mehr einsetzbar sind.

In diesem Abschnitt sollte gezeigt werden, dass es durchaus Optimierungspotenzial für größere Körper gibt, diese aber durch äußere Umstände stark beschränkt sind. Für kleinere Körper bis $\mathbb{F}_{2^8}$ reichen Lookup-Tabellen – zumindest für normale Rechner – aus. Eine Lösung für sehr rechenschwache und energiebeschränkte Geräte zeigt die Arbeit „Coping with the Upcoming Heterogeneity in 5G Communications and Storage Using Fulcrum Network Codes“ [LPHF14]. Dort werden Fulcrum-Codes als Lösung für heterogene Strukturen erläutert.
Ein Fulcrum-Code ist eine Kombination aus 2 Netzwerkcodes, die in jeweils unterschiedlichen Körpern arbeiten. Es gibt einen inneren Code, der im $\mathbb{F}_2$ schnell und auch für schwache Systeme berechenbar ist, und darüber einen äußeren Code im $\mathbb{F}_{2^8}$. Zwischenknoten können dann entscheiden, ob sie nur die einfachen Berechnungen im inneren Code lösen, sprich einfache XOR-Verknüpfungen ausführen, oder ob sie, z. B. bei hoher Netzwerklast, eher mehr Berechnungsaufwand einsetzen und auch über den äußeren Code kombinieren und dadurch Kommunikationsaufwand sparen. Eine adaptive Wahl je nach Situation bezüglich des Netzwerks und Energiebudgets ist somit möglich. Allerdings ist es fraglich, inwieweit das Konzept der Fulcrum-Codes für sichere Verfahren anwendbar wäre.

2.3 Sicherheit

2.3.1 Integrität und Verfügbarkeit

2.3.1.1 Sichere Netzwerkcodierung basierend auf Codierungstheorie

Sicherheit ist ein sehr umfassender Begriff. Hier sei der Begriff im Folgenden als Informationssicherheit gemeint. Diese beschäftigt sich primär mit 3 allgemeinen Schutzzielen: der Vertraulichkeit, der Integrität und der Verfügbarkeit. Vertraulichkeit gewährleistet, dass nur diejenigen Zugriff auf die Informationen erhalten, die dazu berechtigt sind. Integrität stellt sicher, dass jegliche Änderungen an Daten durch Dritte erkennbar sind. Verfügbarkeit stellt sicher, dass Kommunikationspartner auch tatsächlich miteinander kommunizieren können und das System durch Angreifer nicht so gestört werden kann, dass keine Funktionsfähigkeit mehr gegeben ist.
Als Erstes soll die Sicherheit bezüglich der Integrität und der Verfügbarkeit im Fokus stehen. Hier kommt es zu einer nachteiligen Eigenschaft der Netzwerkcodierung durch das Zusammenführen von Paketen. Der aktive Angreifer möchte die Kommunikation zwischen Sender und den Empfängern stören. Daher versucht der Angreifer, Nachrichten so zu modifizieren, dass sie nicht mehr zur linearen Hülle der Generation gehört. Dadurch können die Empfänger nicht mehr richtig decodieren. Das Ganze wird zusätzlich dadurch verschlimmert, dass Zwischenknoten diese ungültige Nachricht mit anderen gültigen Nachrichten kombinieren und selbst ungültige Nachrichten erzeugen. Abbildung 2.5 verdeutlicht das Problem. Im schlimmsten Fall reicht das Modifizieren einer Nachricht aus, um alle bei den Empfängern ankommenden Pakete zu verfälschen. Dieser Angriff wird Verschmutzungsangriff (pollution attack) genannt und hat Folgen sowohl auf die Integrität als auch – in Folge dessen – auf die Verfügbarkeit.
Nun gibt es verschiedene Möglichkeiten, das Problem zu lösen. Dazu werden zwei Fälle unterschieden: Eine Erkennung und gegebenenfalls Korrektur basierend auf codierungstheoretischen Ansätzen, welche im Allgemeinen nur am Empfänger durchgeführt wird, oder eine Erkennung und Verwerfen von den Paketen basierend auf kryptographischen Verfahren, welche sowohl an den Zwischenknoten als auch am Empfänger durchgeführt wird.

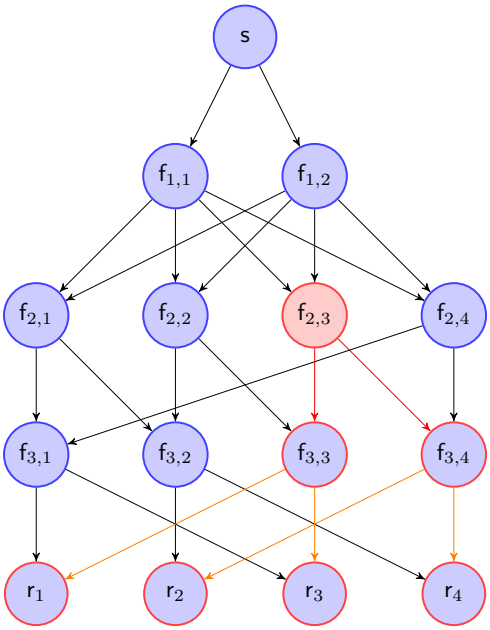


Abbildung 2.5: Verschmutzungsangriff. Der aktive Angreifer $f_{2,3}$ möchte die Kommunikation behindern und schickt verfälschte Pakete (rote Pfeile), welche in die weiteren protokollgerechten Linearkombinationen eingehen und dadurch beschädigte Pakete (orange Pfeile) verursachen. Dies führt dazu, dass alle Empfänger $r \in R$ nicht decodieren können.

Der nun folgende Teil beschäftigt sich mit Möglichkeiten zur Erkennung und Korrektur am Empfänger. Die Grundidee bei auf Codierungstheorie basierenden Mechanismen ist, dass die veränderte Nachricht zwar alle Nachrichten am Empfänger verfälschen kann, die veränderte Nachricht selbst aber in jeder Kombination mit gewissen Koeffizienten eingegangen ist. So ist es – sofern Redundanz zur Fehlerkorrektur vorhanden ist – möglich, die veränderte Nachricht aus fast allen Kombinationen „herauszurechnen“.

Ein erstes Verfahren, das auf Grundlage dieser Idee entstand, ist „Correction of Adversarial Errors in Networks“ [JLHE05], welches eine eher theoretische Arbeit zum optimalen Entwurf von Codes mit niedriger Komplexität ist. Anstatt realer Implementationen werden in der Arbeit zuerst einmal die theoretischen Raten und Grenzen für Unicast und Multicast aufgezeigt.

In „Resilient Network Coding in the Presence of Byzantine Adversaries“ [JLK+08] wurde daraufhin ein Algorithmus für den codierungstheoretischen Ansatz präsentiert, welcher sowohl optimal bezüglich der Rate als auch in Polynomialzeit berechenbar ist. Die gezeigten Raten basieren dabei auf verschiedenen Angriffstypen und Angreiferstärken, die verschiedene Abhörmöglichkeiten oder das Budget eines Angreifers begrenzen. Die Vorteile, in dieser Art mit Verschmutzungsangriffen umzugehen, wie informationstheoretische Sicherheit (auch nicht mit unbegrenzter Berechnungsstärke brechbar) oder keine Änderungsnotwendigkeit an den Zwischenknoten, werden aufgezeigt.

Eine Erweiterung der Arbeit von Jaggi [JLK+08] wurde in „Adversarial Models and Resilient Schemes for Network Coding’"[NL08] behandelt. Neben der Ausweitung auf verschiedene standardisierte Angreifermodelle (z. B. ein kausal allwissender Angreifer, d. h. ein allwissender Angreifer, welcher nur beschränkt Zugriff auf Nachrichten in der Zukunft hat, oder ein rechentechnisch beschränkter Angreifer) wurde eine sehr praxisnahe Implementierung aufgezeigt, um einen codierungstheoretischen Ansatz für die Netzwerkcodierung umzusetzen. Aber auch auf theoretischer Ebene gab es weitere Fortschritte.

In „Coding for Errors and Erasures in Random Network Coding“ [KK08] wurden weitere theoretische Grenzen aufgezeigt.
Ein Problem aller Verfahren basierend auf Codierungstheorie ist, dass kein ratenloses Codieren möglich ist. Daher kann nur bis zu einer vorher definierten Anzahl von fehlerhaften Nachrichten korrigiert werden. Schätzt man die Stärke des Angreifers zu hoch, werden mehr Nachrichten als nötig verschickt. Dies führt zu erhöhten Übertragungszeiten und geringerer Effizienz. Sendet man auf der anderen Seite zu wenig Redundanz, kann der Angreifer wieder die Verfügbarkeit des Systems verhindern. Doch es ist nicht nur schwer zu schätzen, wo und wie viele Angreifer im Netz sind. Schwerwiegender ist das Problem, dass ein Angreifer auch mehrere Nachrichten unterschiedlich verfälschen kann und somit die Fehlerkorrektur fast immer verhindern kann.
Eine mögliche Lösung zu diesem Problem zeigt die Arbeit „Robust Network Coding in the Presence of Untrusted Nodes“ [WSK10] auf. Das Problem, dass ein Angreifer beliebig viele Nachrichten verfälschen kann, wurde durch eine Limitierung umgangen. Die Idee war, dass in jeder Generation ein Knoten nur eine Nachricht übertragen darf. Dazu wurde eine Netzwerktransformation durchgeführt, um die Multicast-Kapazität und damit die Generationsgröße h auf die Anzahl der im Netz vorhandenen disjunkten Pfade zu reduzieren. Somit kann ein Angreifer an einem Ort im Netzwerk nur eine Nachricht der Übertragung angreifen. Dadurch lässt sich eine Abschätzung über die Stärke des Angreifers leichter durchführen, sodass man je nach Auslegung des Codes auch mit mehreren Angreifern (oder einem Angreifer an mehreren Stellen) zurechtkommt. Nachteilig ist, dass es durch die Reduktion der Generationsgröße h natürlich insgesamt zu einer geringeren Leistung durch die Netzwerkcodierung kommt. Außerdem ist die Anwendbarkeit stark von der Topologie abhängig.
Es bleibt also festzuhalten, dass codierungstheoretische Ansätze ihre Vorteile bei schwachen Zwischenknoten oder bestehender Infrastruktur mit ungesicherter Netzwerkcodierung haben, da sich jeweils Sender und Empfänger auf ein Verfahren einigen können und dieses für Zwischenknoten völlig transparent bleibt. Nachteilig sind die Einschränkungen bezüglich der automatischen Adaptivität wie beim ratenlosen Codieren. Hier müssen zwangsläufig, zumindest für die Übertragung einer Generation, Werte für die zu übermittelnde Redundanz festgelegt werden. Diese Nachteile haben kryptographische Verfahren nicht, weshalb nun im Folgenden auf diese alternativen Verfahren eingegangen wird.

2.3.1.2 Sichere Netzwerkcodierung basierend auf Kryptographie

Die Idee hinter allen kryptographiebasierten Ansätzen zur Verhinderung von Verschmutzungsangriffen ist, dass Zwischenknoten verfälschte Pakete erkennen können und dadurch nicht mit in die linearen Kombinationen einbringen. Ein Angreifer kann dadurch eventuell den gesamten Durchsatz etwas verringern, da kein Schutz gegen einen aktiven Angreifer möglich ist. Die Verfügbarkeit des Systems bleibt aber bestehen, da im Netz keine verschmutzten Pakete verbreitet werden. Nun gibt es in der Informationssicherheit altbekannte Methoden, um die Integrität von Daten zu gewährleisten, wie z. B. die Verwendung von Signaturen. Das Problem, welches die Netzwerkcodierung aber mit sich bringt, ist, dass Zwischenknoten und Empfänger auch beliebige Kombinationen von Paketen als gültig erkennen müssen. Übliche Signaturen oder MACs werden allerdings bei der Änderung eines Bits bereits ungültig. Das heißt, dass alle angewendeten Sicherungsmaßnamen homomorphe Eigenschaften mit sich bringen müssen. Das bedeutet, dass zu den Sicherungsmaßnamen zusätzlich eine Funktion existieren muss, welche die gültigen Prüfwerte für die originalen Pakete zu einem gültigen Prüfwert für die Linearkombination der Pakete

überführt. Mithilfe dieser Funktion können Zwischenknoten gültige Prüfwerte für Kombinationen von Paketen bilden.
Da es eine relativ große Menge von solchen Sicherungsmaßnahmen gibt, die auf kryptographischen Verfahren aufbauen, werden diese im Folgenden nochmals in Klassen unterteilt. In Frage kommen dazu homomorphe *Checksummen*, homomorphe *Hashwerte*, homomorphe *MACs* (symmetrische Nachrichtenauthentifizierungscodes) oder homomorphe *Signaturen*. Da im Folgenden nur homomorphe Verfahren betrachtet werden, wird auf die Bezeichnung „homomorph“ verzichtet.

Checksummen Hiermit sind nicht Algorithmen gemeint, die auf festen Berechnungen von Checksummen basieren, wie z. B. CRC (zyklische Redundanzprüfung). Da diese auch einem Angreifer bekannt wären und er veränderte Pakete berichtigen könnte, kommen authentifizierte Checksummen zum Einsatz. Eine solche Checksumme kann z. B. mithilfe einer zufälligen Matrix berechnet werden, welche dann als Art Schlüssel fungiert. Das Problem ist, dass jede Übertragung oder Generation einen neuen Schlüssel bräuchte. Nun könnte man diesen über eine pseudozufällige Funktion erzeugen oder das TESLA-Protokoll [PCTS02] nutzen.
Das Protokoll wird genauer in „The TESLA Broadcast Authentication Protocol“ [PCTS02] vorgestellt und hat erstmal nichts mit Netzwerkcodierung zu tun, ist aber die Basis für einige Protokolle. Das TESLA-Protokoll ist eine Alternative zu dem notwendigen Schlüsselaustausch zwischen Sender und Empfängern bei der symmetrischen Authentifizierung. Die Grundidee ist wie folgt: Ein Paket wird authentifiziert (z. B. mit einem MAC) mithilfe eines zufälligen symmetrischen Schlüssels. Dann wird das Paket versendet ohne MAC oder Schlüssel offenzulegen. Der Empfänger empfängt das Paket und speichert den Ankunftszeitpunkt. Danach veröffentlicht der Sender (z. B. über einen Broadcast) den MAC und den passenden Schlüssel. Das vorher versendete Paket wird nur akzeptiert, wenn die Erzeugung des Schlüsselpakets zeitlich nach dem Empfang des zu authentifizierenden Pakets liegt und MAC und Schlüssel zu dem Paket passen. Das hat den Vorteil, dass alle Berechnungen sehr leichtgewichtig sind.
Asymmetrische Kryptographie ist nur für die Checksummenpakete mit Schlüssel notwendig und diese Pakete bedürfen keinerlei Homomorphieeigenschaft, da sie nicht kombiniert werden. Nachteilig können sich natürlich die höhere Latenz durch die Zeitasymmetrie als auch die erhöhte Kommunikation durch zusätzliche Checksummenpakete auf die Leistung auswirken.
Im DART-Protokoll, welches in „Practical Defenses Against Pollution Attacks in Intra-Flow Network Coding for Wireless Mesh Networks“ [DCNR09] und in „Practical Defenses Against Pollution Attacks in Wireless Network Coding“ [DCNR11] vorgestellt wurde, bedient man sich genau dieses TESLA Ansatzes. Die Pakete werden genauso codiert wie in „Practical Network Coding“ [CWJ03]. Zusätzlich puffern nun Zwischenknoten die empfangenen Pakete in einem Zwischenspeicher für nicht verifizierte Pakete und versehen sie mit dem Ankunftsdatum. Der Sender verschickt periodisch Checksummenpakete, die eine Seed für den Schlüssel (hier eine Matrix), einen Zeitstempel und die Checksumme enthält. Das Paket ist, um es vor Veränderungen zu schützen, digital signiert. Wenn ein Checksummenpaket einen Zeitstempel enthält, welcher später als der Empfang eines Pakets ist, kann das Paket verifiziert werden. Dazu wird die Nachricht samt Koeffizienten sowie die Seed und die Checksumme verwendet. Wenn es sich um eine passende Checksumme handelt, wandert das Paket in den Zwischenspeicher der verifizierten Pakete, von wo aus es mit anderen innerhalb dieses Speicher liegenden und zur gleichen Generation gehörenden Paketen kombiniert werden kann. Zumindest eine gewisse zeitliche Synchronizität ist Vor-

aussetzung, damit das Verfahren funktioniert, wobei die Genauigkeit entscheidend für die Latenz der Übertragung ist.

Hashwerte Eine andere Möglichkeit, die Integrität zu gewährleisten, ist Hashwerte zu benutzen. Hashes sind Funktionen, die von einer großen Definitionsmenge (hier die Daten, auch unterschiedlicher Länge) auf eine kleinere Zielmenge (die Hashwerte) abbilden. Dabei gibt es je nach Anwendung verschiedene Anforderungen, wie Unumkehrbarkeit, Surjektivität oder Kollisionsresistenz. Ähnlich wie bei den Checksummen, sollen Zwischenknoten Änderungen an den Paketen bemerken, indem sie die Konsistenz zwischen Daten und Hashwerten überprüfen können.

Schon kurz nach der Entdeckung der Problematik der Verschmutzungsangriffe wurden Arbeiten veröffentlicht, die Hashwerte als Möglichkeit der Authentifizierung vorstellen. In „Byzantine Modification Detection in Multicast Networks using Randomized Network Coding“ [HLK+04] wird auf die Theorie hinter den Verschmutzungsangriffen eingegangen. Als Lösung wurde eine Erkennung an den Zwischenknoten vorgeschlagen, indem an jedes Paket eine gewisse Menge von Hashwerten angehangen wird. Ein Zwischenknoten kann dann die Gültigkeit der Werte für das Paket überprüfen. Je nach Anzahl der Hashwerte verringert sich die Wahrscheinlichkeit, dass verfälschte Pakete nicht erkannt werden.

Eine eher praktischere Umsetzung wurde im selben Jahr in der Arbeit „On-the-Fly Verification of Rateless Erasure Codes for Efficient Content Distribution“ [KFM04] vorgestellt. Dabei werden vor der Übertragung einer Generation die Hashwerte für alle nativen Pakete an alle (an der Kommunikation beteiligten) Knoten verteilt. Sollte ein Zwischenknoten nun ein codiertes Paket erhalten, kann dieser sowohl den Hashwert der Daten aus dem Paket berechnen als auch die Kombination der vorher übermittelten Hashwerte mit dem Codierungsvektor aus dem Paket. Ist beides gleich, kann das Paket recodiert und weitergeleitet werden, ansonsten wird es verworfen.

Der Vorteil des Verfahrens ist die Möglichkeit der schnellen Überprüfung, da man nicht erst auf zusätzliche Checksummenpakete warten muss. Weiterhin kommt es durch die homomorphe Eigenschaft der Hashwerte zu keinerlei Einschränkungen beim Recodieren. Allerdings gibt es auch Nachteile. Zum einen müssen zusätzliche Broadcasts mit den Hashwerten verschickt werden. Diese wiederum werden meist mit asymmetrischer Kryptographie signiert. Weiterhin ist es für sichere Hashes – hier im Besonderen für die Unumkehrbarkeit – notwendig, in einem größeren Körper zu arbeiten. Dies führt zum einen zu einem erhöhten Berechnungsaufwand, zum anderen auch zu größeren Codierungsvektoren und damit zu einem höheren Übertragungsaufwand.

In „Cooperative Security for Network Coding File Distribution“ [GR06] wurde dieses Verfahren auf das Gebiet des Datenaustauschs angewendet. In solchen Systemen ist es aufwendig, die Originalität der Daten an Zwischenknoten zu prüfen. Deswegen kann es auch hier zu Verfälschungen kommen, die einen großen Einfluss auf die Übertragung haben. Zur Lösung des Problems werden homomorphe Hashwerte, ähnlich wie in [KFM04], angewendet. Zusätzlich wird die Idee der kooperativen Sicherheit zur Verbesserung der Effizienz vorgeschlagen. Dabei informieren Nutzer sich gegenseitig über fehlerbehaftete Pakete und woher diese kommen. Dadurch ist es ausreichend nur einen Teil der Pakete zu überprüfen, um Verschmutzungen effektiv zu verhindern.

In dem gleichen Anwendungsgebiet bewegt sich auch die Arbeit „On the Practical and Security Issues of Batch Content Distribution Via Network Coding“ [LCL06]. Es wird auf die Effizienzprobleme von [KFM04] eingegangen. Zum einen gibt es zu viel Kommunikation und eine große Latenz durch die

Hashpakete, die vorher übermittelt werden müssen. Zum anderen benötigt man viele Berechnungen, da für jedes Paket der Hashwert und die Kombination aus den übermittelten Hashwerten berechnet werden müssen. Die Idee ist eine stapelweise Übertragung und Verifikation von Paketen. Dazu wird eine spezielle Hashfunktion vorgestellt, welche eine Einwegfunktion mit Falltür ist und auf einer modifizierten Permutationsfunktion basiert. Weiterhin werden die Daten für die Authentifizierung direkt in jedes Paket integriert, um die Latenz zu Beginn zu reduzieren.
Später wurde versucht, das ursprüngliche Verfahren [KFM04] nochmals zu verbessern. In der Arbeit „On the Security and Efficiency of Content Distribution via Network Coding“ [LLC12] wurden Erweiterungen, die speziell für den Datenaustausch geeignet sind, zusammengefasst. Neben dem geringeren Kommunikations- und Berechnungsaufwand zeigten die Autoren nochmalige Verbesserungen an dem Verfahren. Eine kleinere Generationsgröße gewährleistet, dass nur wenige Pakete miteinander kombiniert werden, um den Berechnungsaufwand niedrig zu halten. Außerdem soll die Kombination nur wenige Pakete enthalten, damit die globale Codierungsmatrix nur dünn besetzt ist, was Vereinfachungen bei der Decodierung mit sich bringt.
Eine ganz andere Möglichkeit zeigt „Null Keys: Limiting Malicious Attacks Via Null Space Properties of Network Coding“ [KL09] auf. Anstatt mit den homomorphen Hashwerten die Daten zu verifizieren, werden in diesem Verfahren sogenannte Nullvektoren (*Null Keys*) vom Sender zu den anderen Knoten mittels der Hashfunktionen authentifiziert. Nullvektoren sind Vektoren, die orthogonal zur linearen Hülle der Generation sind. Zwischenknoten können somit die Integrität der Daten über die Nullvektoren überprüfen. Gegenüber Angreifern, die die Nachrichten so verändern könnten, dass diese trotz Verfälschung orthogonal zu dem Nullvektor liegen, bietet das Verfahren eine gewisse Sicherheit, da die Nullvektoren auch kombiniert werden können und somit für einen Angreifer nicht einfach ersichtlich ist, welche Nullvektoren bei einem bestimmten Zwischenknoten bekannt sind. Dies gilt allerdings nur für bestimmte Angreifermodelle. Hat ein Angreifer beispielsweise die Möglichkeit auf vielen Kanten abzuhören, ist es ihm möglich, Kombinationen von Nullvektoren zu ermitteln und so verfälschte Pakete zu erzeugen, die nicht sofort erkannt werden können.

MACs MACs basieren auf symmetrischer Kryptographie und sind daher rechentechnisch besonders einfach zu erstellen und zu testen. Der Nachteil ist, ähnlich zu den Checksummen, dass für die Prüfung der symmetrische Schlüssel bekannt sein muss. Allerdings kann jeder, der den Schlüssel hat, auch neue gültige MACs berechnen. Dadurch ist es notwendig, zusätzliche Sicherungsmaßnamen zu ergreifen, um ein auf MACs basierendes Verfahren abzusichern. Weiterhin reicht eine gewöhnliche Ende-zu-Ende-Authentifikation nicht aus, da auch eine Überprüfung der Pakete durch die Zwischenknoten möglich sein muss.
Erste Untersuchungen dazu wurden in der Arbeit „Homomorphic MACs: MAC-Based Integrity for Network Coding“ [AB09] vorgestellt. Dabei entwarfen die Autoren zuerst ein MAC-Verfahren zur Überprüfung beim Empfänger, welches für Übertragungen mit Netzwerkcodierung geeignet war. Im Anschluss wandelten sie dieses in ein Broadcast-System, bei dem mehrere MACs und Schlüssel verwendet werden. Diese Schlüssel werden so an die Knoten verteilt, dass kein Knoten alle Schlüssel eines anderen Knotens besitzt (überdeckungsfreie Mengen). Dabei erzielte die Lösung gute Leistungen für Angriffe, bei denen Angreifer nur wenige Nachrichten verändern konnten. Für stärkere Angriffe empfehlen die Autoren, auf das TESLA-Protokoll zurückzugreifen.

Ein solches System, dass ähnlich wie DART [DCNR09] funktioniert, ist RIPPLE, welches in der Arbeit „RIPPLE Authentication for Network Coding“ [LYC+10] vorgestellt wurde. Die Ähnlichkeit rührt daher, dass auch hier das TESLA-Protokoll [PCTS02] zum Einsatz kommt, um die schrittweise Offenlegung der Schlüssel für die MACs umzusetzen. Es handelt sich also um ein Verfahren, bei dem an die Pakete homomorphe MACs angehangen werden und diese über die Zeitasymmetrie von TESLA abgesichert werden. Dadurch hat das Verfahren zwar einen kleinen Berechnungsaufwand, der aber durch eine höhere Latenz erkauft wird.

Weiterhin gibt es Arbeiten wie „Insecure ‘Provably Secure Network Coding’ and Homomorphic Authentication Schemes for Network Coding“ [Wan10], welche die Sicherheit des Verfahrens in Frage stellen und es für unsicher halten. Wang zeigte zum einen Sicherheitsprobleme und Risiken als auch Effizienzprobleme von RIPPLE [LYC+10] und anderen bestehenden Arbeiten. Zum anderen präsentierte er in derselben Arbeit auch ein eigenes homomorphes MAC-Verfahren, welches beweisbar sicher ist und ebenfalls auf dem TESLA-Protokoll [PCTS02] basiert. Allerdings ist für dieses Verfahren ein MAC pro Zwischenknoten mehr notwendig und das ganze Verfahren funktioniert nur für Wege, für welche die maximale Länge bekannt oder festgelegt worden ist.

Im Allgemeinen gibt es das Problem, dass die Sicherheit eines MACs von der Körpergröße q abhängt. Da nur q Möglichkeiten für einen MAC existieren, kann ein Angreifer mit der Wahrscheinlichkeit $\frac{1}{q}$ einen gültigen MAC zu einem modifizierten Paket erraten. Es gibt zwei Möglichkeiten, die Rate der Nichterkennung einer Modifizierung $\frac{1}{q}$ zu minimieren: Entweder man erhöht die Körpergröße q oder man benutzt mehrere, z. B. l, MACs, welche nicht direkt von einem Schlüssel abhängen. In beiden Fällen erhöht sich der Kommunikations- und Berechnungsaufwand. Dabei ist die Erhöhung der MAC-Anzahl effizienter, da der Aufwand linear steigt, während eine höhere Körpergröße eine superlineare Steigerung beim Berechnungsaufwand nach sich zieht.

Eine Optimierung stellt das TraceMac-Verfahren dar. Es wurde in „A Novel Homomorphic MAC Scheme for Authentication in Network Coding“ [CJ11] vorgestellt. Dabei wird für die Erhöhung der Sicherheit auf $\frac{1}{q^l}$ mit l MACs nur ein Schlüssel verwendet, welcher in die Berechnung der MACs eingeht. Dadurch wird eine höhere Sicherheit trotz kleinerer Schlüssel ermöglicht. Leider zeigte sich, dass dieses Verfahren angreifbar ist. In „Comment on ‘A Novel Homomorphic MAC Scheme for Authentication in Network Coding’“ [LLL+14] gehen die Autoren auf einen Angriff ein, der die Sicherheit wieder auf $\frac{1}{q}$ reduziert. Allerdings wurde wenig später in „Comments on ‘An Efficient Homomorphic MAC with Small Key Size for Authentication in Network Coding’“ [Kim15] eine alternative Konstruktion zur MAC-Generierung angegeben, mit der die Sicherheit $\frac{1}{q^l}$ mit nur einem Schlüssel erreichbar ist.

In der Arbeit „Cooperative Defense Against Pollution Attacks in Network Coding Using SpaceMac“ [LM12] wird das SpaceMac-Verfahren vorgestellt, das ebenfalls auf homomorphen MACs basiert. Das Verfahren bietet neben der Erkennung von verfälschten Paketen auch die Möglichkeit der Lokalisierung des Angreifers. Bei diesem kooperativen Verfahren ist allerdings eine zentrale Instanz notwendig. Weiterhin ist der Berechnungsaufwand höher als bei Verfahren mit Zeitasymmetrie. Vorteilhaft sind allerdings der geringere Kommunikationsaufwand und auch die kürzeren Verzögerungszeiten, sofern der Berechnungsaufwand für den Knoten kein Problem darstellt.

Eine weitere Alternative zu den Verfahren, die auf Zeitasymmetrie basieren, wird in der Arbeit „An Efficient MAC Scheme for Secure Network Coding with Probabilistic Detection“ [WLC12] vorgestellt. Dabei ist die Grundidee, dass allen Teilnehmern probabilistisch Schlüssel zugeteilt werden, die aus ei-

nem gemeinsamen Schlüsselpool stammen. Nun werden mehrere MACs an ein Paket angehangen, die auf unterschiedlichen Schlüsseln basieren. Die Wahrscheinlichkeit, dass ein Angreifer alle dabei verwendeten Schlüssel besitzt und eine Generation unbemerkt fälschen kann, geht gegen 0 mit steigender MAC- und Schlüsselanzahl. Auf der anderen Seite gilt das Gleiche für einen nachfolgenden Knoten. Deswegen können Zwischenknoten und Empfänger nur eine Teilmenge der verwendeten MACs prüfen. Wenn ein Angreifer Nachrichten fälscht und für die passenden Schlüssel die MACs ändert (tag pollution), ist es sehr wahrscheinlich, dass der nachfolgende Knoten Schlüssel besitzt, die in der Nachricht verwendet wurden, aber dem Angreifer nicht bekannt waren. Somit kann ein Zwischenknoten eine Fälschung mit hoher Wahrscheinlichkeit erkennen. Sollte es bei diesem probabilistischen Verfahren doch zu einer Verschmutzung kommen, wird diese zumindest sehr wahrscheinlich beim darauffolgenden Zwischenknoten erkannt.
Die Variante, die Schlüssel für die MACs durch einen Schlüsselpool zu verteilen, ist somit eine interessante Alternative zu den TESLA-basierten Protokollen. Der Aufwand für den Schlüsselpool und die MACs ist dabei abhängig von der Topologie und der angestrebten Sicherheit. Allerdings erreicht man je nach Parametrisierung hohe und schnelle Erkennung von Angriffen auch bei niedrigen zusätzlichen Kosten. Dabei kann es zu kurzzeitiger Verschmutzung im Netzwerk kommen.

Signaturen Häufig werden Signaturen als Schutz der Integrität von Daten verwendet. Das Problem wiederum bei der Netzwerkcodierung ist das Recodieren, welches homomorphe Signaturen verlangt, damit gültige Signaturen auch bei der Kombination von gültigen Signaturen entstehen können. Zusätzlich muss der Körper, in dem eine Signatur erstellt wird, sehr groß sein, damit die Sicherheit gegenüber beschränkten Angreifern gewährleistet ist.
Eines der ersten Verfahren – „Signatures for Content Distribution with Network Coding“ [ZKMH07] – kommt ebenfalls wie die MAC-Verfahren aus dem Bereich des Datenaustauschs. Eine homomorphe Signatur, die auf dem Diskreten-Logarithmus-Problem basiert und an ein Paket angehängt wird, ist die Grundlage des Verfahrens. Jeder Zwischenknoten kann damit überprüfen, ob Codierungsvektor, Daten und Signatur zueinander passen. Laut Aussage der Autoren ist der zusätzliche Aufwand sehr gering, da nur ein Signaturvektor pro Datei notwendig wird. Allerdings kann jede Signatur je nach gewähltem Sicherheitsparameter recht groß sein. Zusätzlich ist ein großer Körper für die Sicherheit vonnöten, was den Platz für die Koeffizienten des Codierungsvektors erhöht.
Eine Idee, um die Vergrößerung des Codierungsvektors durch den großen Körper zu umgehen, wurde in „Secure Network Coding Over the Integers“ [GKKR09] aufgezeigt. Das Verfahren selbst basiert auf der homomorphen Eigenschaft von RSA [RSA78], dass gültige Signaturen zu einer neuen gültigen Signatur kombiniert werden können. Dabei werden die Berechnungen nicht in einem endlichen Körper, sondern in einem Restklassenring $\mathbb{Z}/N\mathbb{Z}$ (die ganzen Zahlen modulo dem RSA-Modul N) durchgeführt. Dies hat zur Folge, dass im Codierungsvektor auch ganze Zahlen verwendet werden können. Die Idee ist nun mit kleinen Zahlen zu beginnen und nur kleine Koeffizienten zu verwenden, damit der Codierungsvektor relativ klein bleibt. Für eine Übertragung mit wenigen Zwischenknoten bedeutet dies wenig Zusatzkosten und eine einfache Implementierung. Allerdings heißt das auch, dass die Größe des Codierungsvektors von Zwischenknoten zu Zwischenknoten ansteigt. Die Größe ist zwar theoretisch begrenzt, da auch hier modulo N gerechnet wird, allerdings ist das erst nach vielen Zwischenknoten und Kombinationen möglich. Weiterhin sind die Koeffizienten als auch die Signatur durch die höheren Anforderungen an das

RSA-Modul riesig. Gängige Größen sind z. B. $2\,048$ Bit = 256 Byte. Das macht dieses Verfahren für Übertragungen über viele Zwischenknoten ineffizient.

In „Signing a Linear Subspace: Signature Schemes for Network Coding“ [BFKW09] wurde ein homomorphes Signaturverfahren mit konstanter Schlüssel- und Signaturgröße vorgestellt. Das ganze Verfahren basiert auf bilinearen Gruppen, um die Schlüssel und damit die Größe der Koeffizienten im Codierungsvektor klein zu halten. Weiterhin findet sich in der Arbeit ein Sicherheitsbeweis, der zeigt, dass ein leicht modifiziertes Verfahren auf das Diskrete-Logarithmus-Problem abgebildet werden kann.

Ein ähnliches Verfahren wurde auch in „Signatures for Network Coding“ [CJL09] gezeigt. Dieses homomorphe Signaturverfahren basiert ebenfalls wie [BFKW09] auf bilinearen Gruppen. Auf diesen wird zum Signieren und Testen das Weil-Pairing durchgeführt. Allen Paketen wird die Signatur angehangen, sodass Zwischenknoten in der Lage sind, die Signatur zu überprüfen. Wie auch bei [BFKW09] ist der Platzbedarf für die einzelnen Koeffizienten und die Signatur reduziert, allerdings ist der Berechnungsaufwand durch die bilinearen Gruppen sehr hoch. Zusätzlich sind weitere Daten, die per Broadcast vor der eigentlichen Übertragung gesendet werden müssen, notwendig.

Ein ebenfalls auf bilinearen Gruppen basierendes Verfahren, welches in „Efficient Network Coding Signatures in the Standard Model“ [CFW12] veröffentlicht worden ist, wurde im Standardmodell als sicher bewiesen. Dabei basiert es auf bilinearen Gruppen mit der Größe einer Primzahl p. Zusätzlich präsentieren die Autoren auch ein zweites Verfahren, welches über eine zusammengesetzte Zahl $n = p \cdot q$, ähnlich dem RSA-Modul, funktioniert. Auch dieses Verfahren konnte im Standardmodell als sicher bewiesen werden.

In der Arbeit „Linearly Homomorphic Signatures over Binary Fields and New Tools for Lattice-Based Signatures“ [BF11] wurde auf eine gänzlich andere Struktur gebaut. Das homomorphe Signaturverfahren arbeitet auf Gitterstrukturen (Lattices). Dabei wird die Tauglichkeit des Verfahrens aus theoretischer als auch praktischer Sichtweise untersucht. Die Gitterstruktur hat ihren Vorteil durch die geringere Körpergröße, die möglich ist. Selbst eine Größe von 2 statt 2^8 soll möglich sein. Weiterhin liefern die Autoren einen Sicherheitsbeweis nach dem Zufallsorakel-Modell (Random oracle model).

Auf der gleichen Struktur arbeitet auch das Verfahren, das in „Secure Network Coding Based on Lattice Signature“ [SPL14] veröffentlicht wurde. Das homomorphe Signaturverfahren, das auf Gittern arbeitet, erreichte darin eine hohe Effizienz. Die Simulationsergebnisse deuteten auf eine bessere Leistung und höhere Effizienz hin als bei einem auf RSA basierenden Signaturverfahren. Einschränkend ist jedoch zu sagen, dass die Arbeit sehr theoretisch ist und nur Simulationsergebnisse aufweist. Ein praktischer Einsatz von Signaturverfahren auf Gitterstrukturen wurde noch nicht genauer untersucht.

Einen Überblick zu bestehenden Verfahren, die auf MACs (und auch Signaturen) basieren, bietet „Homomorphic Signatures and Message Authentication Codes“ [Cat14]. Quintessenz dessen ist, dass es zur Zeit keine Verfahren mit vollständig homomorphen Signaturen und MACs gibt. Dadurch muss immer eine entsprechend hohe Körpergröße verwendet werden oder es müssen mehrere MACs benutzt werden, sofern man im häufig verwendeten Körper $\mathbb{F}_{2^8}$ arbeiten möchte.

Zusammenfassend ist festzuhalten, dass es zahlreiche Varianten und Weiterentwicklungen gibt, um Verschmutzungsangriffe mit kryptographischen Methoden zu verhindern. Vergleiche zwischen unterschiedlichen Strukturen fehlen jedoch, sodass es schwierig zu entscheiden ist, welches Protokoll am geeignetsten ist. Im Folgenden werden die Schutzziele der Integrität und Verfügbarkeit in den Hintergrund rücken und das Schutzziel der Vertraulichkeit im Fokus stehen.

2.3.2 Vertraulichkeit

2.3.2.1 Algebraische Sicherheit

Vertraulichkeit soll sicherstellen, dass nur berechtigten Empfängern Inhalte der Nachrichten erkenntlich werden. Dazu wird im Allgemeinen eine Verschlüsselung am Sender und eine Entschlüsselung am Empfänger verwendet. Im Gegensatz zur Integritätsabsicherung, die durch die Netzwerkcodierung notwendiger und aufwendiger wurde, bietet die Netzwerkcodierung bereits einen gewissen Schutz der Vertraulichkeit. Das liegt daran, dass im Normalfall lineare Kombinationen anstatt nativen Paketen gesendet werden und so Abhörer keine direkten Informationen über die Nachricht erhalten, sofern sie nicht decodieren können.

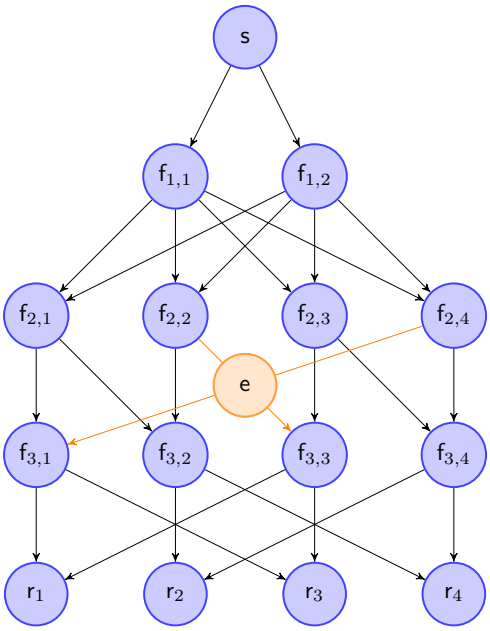


Abbildung 2.6: Abhören im Netz. Ein angreifender Knoten e versucht Nachrichten abzuhören. Es kann sich bei e auch um einen an der Kommunikation beteiligten Knoten handeln.

In einer ersten Arbeit zur Sicherheit der Netzwerkcodierung („Secure Network Coding“ [CY02]) wurde diese Eigenschaft, die sogenannte algebraische Sicherheit, festgestellt und untersucht. Die Grundidee ist, dass, wenn nur codierte Pakete verschickt werden und ein Angreifer nur $k < h$ Pakete abhören kann, er die Generation nicht decodieren kann. Zur Verdeutlichung des Schutzziels wird das in Abbildung 2.6 dargestellte Netz durch den Abhörer e angegriffen. Kann e genügend Pakete (mindestens h linear unabhängige Pakete) abhören, so kann er wie ein regulärer Empfänger die Nachricht decodieren und die Vertraulichkeit ist verletzt. Sollte er aber weniger als h linear unabhängige Pakete abgehört haben, ist es ihm nicht möglich, die Generation zu decodieren, da er die Codierungsmatrix nicht invertieren kann.
Eine Ausnutzung dieser Eigenschaft zur Erlangung der Vertraulichkeit wurde in der Arbeit „Secure Network Coding with a Cost Criterion“ [TM06] aufgezeigt. Die Idee war, die algebraische Sicherheit gegen beschränkte Abhörer (hier sind Abhörer gemeint, welche nur eine gewisse Anzahl an Kanten abhören können) einzusetzen, um zu verhindern, dass diese an die originalen Informationen kommen. Da es keine exakte Methode gibt zu bestimmen, wie die Nachrichten auf die einzelnen Pfade aufgeteilt werden und welcher Netzwerkcode am sinnvollsten wäre, präsentierten die Autoren zwei Heuristiken. Mithilfe dieser können sinnvolle Parameter festgelegt werden, die je nach Wichtung auf höhere Sicherheit oder

niedrigere Netzwerklast unterschiedlich ausfallen können. In der Auswertung, in der unterschiedliche Netzwerke und Angreiferstärken untersucht wurden, konnte gezeigt werden, dass es einen Trade-off zwischen niedriger Netzwerklast und erhöhter Vertraulichkeit gibt.

Eine weitergehende Analyse der algebraischen Sicherheit wurde später in „A Security Condition for Multi-Source Linear Network Coding“ [CY07] vorgestellt. Dabei wurden die notwendigen und hinreichenden Kriterien zur Vertraulichkeit dargestellt. Es wurde gezeigt, dass ein Abhörer mit der richtigen Parametrisierung daran gehindert werden kann, auch nur irgendeine Nachricht decodieren zu können. Dies gilt natürlich nur unter der Voraussetzung, dass der Abhörer beschränkt ist und nur eine bestimmte Anzahl an Paketen oder einen bestimmten Anteil der übertragenen Pakete abhören kann.

Auch die Arbeit „Random Linear Network Coding: A free cipher?“ [LMB07] beschäftigt sich mit der algebraischen Sicherheit. Neben dem Kriterium, dass der Rang der k abgehörten Nachrichten kleiner als die Generationsgröße h sein muss, wurde auch auf ein anderes Problem eingegangen. Grundsätzlich gilt, dass nur genau dann, wenn die Codierungsmatrix der empfangenen Nachrichten vollen Rang besitzt, die Matrix invertierbar und somit die gesamte Generation decodierbar ist. Allerdings ist das Schutzziel der Vertraulichkeit nicht auf eine komplette Generation, sondern allgemein auf Daten bezogen, sodass auch das Decodieren einer einzelnen Nachricht bereits das Vertraulichkeitskriterium verletzten würde.

Im Folgenden sei nun erläutert, wie es passieren kann, dass einzelne Nachrichten, aber nicht die gesamte Generation decodierbar sind. Sei $k < h$ und die k Nachrichten linear unabhängig, dann ist die Matrix der Codierungsvektoren eine $k \times h$ Matrix. Jede solche Matrix mit beliebigen Koeffizienten lässt sich (z. B. über den Gauß-Jordan-Algorithmus) in eine Matrix mit zwei Teilen umformen. Der erste Teil (z. B. links) ist dann eine $k \times k$ Einheitsmatrix. Der andere Teil eine $k \times (h-k)$ Matrix mit beliebigen Koeffizienten:

$$\boldsymbol{B}' = \begin{bmatrix} 1 & 0 & 0 & \overbrace{\beta_{1,4} & \beta_{1,5}}^{h-k} \\ 0 & 1 & 0 & \beta_{2,4} & \beta_{2,5} \\ 0 & 0 & 1 & \beta_{3,4} & \beta_{3,5} \end{bmatrix} \Bigg\} k. \tag{2.5}$$

Tritt nun in einer der k Zeilen bei allen $(h-k)$ Koeffizienten eine 0 auf so ist diese Nachricht, nicht aber die gesamte Matrix decodierbar. Die Wahrscheinlichkeit für sogenannte partiell diagonalisierbare Matrizen ist hauptsächlich abhängig von der Generationsgröße h, der Anzahl linear unabhängiger empfangener Pakete k und der Größe des endlichen Körpers q.

In [LMB07] wurde aufgezeigt, dass die Wahrscheinlichkeit für diagonalisierbare Zeilen in der Matrix asymptotisch gegen 0 sinkt, wenn die Größe des Körpers sowie die Generationsgröße gegen unendlich streben. Allerdings ist das in realen Systemen nicht möglich. Zusammenfassend kamen die Autoren zu dem Ergebnis, dass in realen Systemen die algebraische Sicherheit von Topologie und Parametern, wie z. B. der Generationsgröße h oder der Körpergröße q, abhängig ist. Dafür ist die von der Netzwerkcodierung inhärent angebotene algebraische Vertraulichkeit fast ohne zusätzlichen Aufwand zu bekommen.

In „Secure Network Coding on a Wiretap Network“ [CY11] wird eine Möglichkeit vorgestellt, die algebraische Sicherheit zu verwenden und die partiell diagonalisierbaren Matrizen zu verhindern. Die Idee ist, dass man davon ausgehen kann, dass ein Angreifer maximal r Nachrichten abhören kann. Darauf aufbauend kann ein optimaler Code konstruiert werden, sodass jede r-elementige Teilmenge aller Nachrichten zu keiner partiell diagonalisierbaren Matrix führt. Damit wäre sichergestellt, dass ein Angreifer keinerlei verwendbare Informationen erhält, sofern er maximal r Nachrichten abhört. Allerdings basiert dieses Verfahren auf deterministischen Codes und ist deswegen nicht dezentral und in dynamischen

Netzwerken einsetzbar. Weiterhin bleibt die Frage, wie Vertraulichkeit ermöglicht werden kann, wenn der Abhörer hinsichtlich der abzuhörenden Nachrichten nicht beschränkt werden kann.
Eine kryptographische Analyse in „An Information-Theoretic Cryptanalysis of Network Coding – is Protecting the Code Enough?“ [LVBM08] untersuchte, ob es ausreichend ist die Codierungsmatrix vor einem Angreifer zu verbergen. Die Idee ist, die algebraische Sicherheit so auszunutzen, dass Vertraulichkeit gewährleistet werden kann, selbst wenn ein Angreifer genügend Pakete erhält, aber nun nicht weiß, wie diese codiert sind. Dazu wurden in der Arbeit der Transinformationsfluss und dessen Grenzen bzw. Raten untersucht. Die Kernaussage der Analyse ist, dass das Verbergen der Koeffizienten generell ausreichend und sicher ist, wenn die Daten komprimiert sind, also eine optimale zufallsartige Verteilung von Bits gegeben ist. Einschränkend dazu gilt jedoch, dass es im Einzelnen auf die Wahl der Parameter ankommt, um z. B. das Ausprobieren aller Möglichkeiten (brute-force) zu unterbinden.
Zusammenfassend zeigt sich, dass Netzwerkcodierung einen gewissen Grundschutz der Vertraulichkeit bietet. Allerdings stößt die algebraische Sicherheit an zwei Stellen auf ihre Grenzen: zum einen, wenn ein Angreifer h oder mehr Pakete abhören kann und zum anderen, wenn es bei zufälliger Netzwerkcodierung zu partiell diagonalisierbaren Matrizen kommt. Die zuletzt vorgestellte Arbeit [LVBM08] zeigt jedoch einen gangbaren Weg, die algebraische Sicherheit auszunutzen, um Vertraulichkeit auch gegenüber stärkeren Angreifern zu gewährleisten, ohne gleich eine komplette Ende-zu-Ende-Verschlüsselung zu verwenden. Im nächsten Teil soll es daher um leichtgewichtige Verfahren gehen, die Vertraulichkeit gewährleisten können.

2.3.2.2 Leichtgewichtige Verfahren

Zu Beginn sei erläutert, wie man nach aktuellem Stand der Technik die Vertraulichkeit einer Datenübertragung schützen würde. Dazu würden die Nutzdaten zunächst mit einem Algorithmus, z. B. AES (Advanced Encryption Standard) [DBN+01] verschlüsselt werden. Danach würde normale Netzwerkcodierung durchgeführt. Abhörende (Zwischen-)knoten, die genügend Pakete zur Decodierung haben, könnten zwar decodieren, nicht aber die Daten entschlüsseln, da ihnen der Schlüssel für die AES Verschlüsselung fehlt. Dadurch wäre die Vertraulichkeit gewahrt, auch wenn beliebig viele Pakete abgehört werden könnten.
Der Empfänger muss nach Empfang von h linear unabhängigen Nachrichten die Daten nach dem Decodieren lediglich noch entschlüsseln, um an die Nutzdaten zu kommen. Dieses Verfahren bietet gewisse Vorteile. Zum einen ist eine Hardware Implementierung für AES möglich. Das AES Verfahren ist hinsichtlich Sicherheit überprüft und schnell. Es gibt auch verschiedene Schlüsselgrößen und Modi für unterschiedliche Sicherheitsanforderungen. Zum anderen kann das Verschlüsselungsverfahren AES leicht gegen ein anderes Verfahren ausgetauscht werden, da Netzwerkcodierung und Verschlüsselung auf unterschiedlichen Ebenen arbeiten. Auf der anderen Seite lässt die inhärente algebraische Sicherheit, welche die Netzwerkcodierung bietet, unbenutzt. Daher ist die Frage zu klären, ob man unter Ausnutzung dieser Eigenschaft effizientere und sichere Verfahren findet.
Ein solches leichtgewichtiges Verfahren ist SPOC (Secure Practical netwOrk Coding). Es wurde in „Lightweight Security for Network Coding“ [VLB08] vorgestellt. Anstatt die Nutzdaten zu verschlüsseln, werden die Codierungskoeffizienten verschlüsselt. Dies gewährleistet nach [LVBM08] unter der oben genannten Voraussetzung, dass die Daten zuvor komprimiert wurden, die Vertraulichkeit. Damit das Recodieren trotz nicht homomorpher Verschlüsselung (z. B. AES) weiterhin funktioniert, wird je-

dem Paket ein äußerer Koeffizientenvektor vorangestellt. Somit muss jeder Empfänger nach Erhalt der recodierten Nachrichten zuerst die äußere Matrix decodieren. Dadurch kommt er an die ursprünglichen, aber verschlüsselten inneren Koeffizienten. Diese werden dechiffriert und die inneren Codierungskoeffizienten anschließend decodiert.

Obwohl ein zusätzliches Codieren und Decodieren im Gegensatz zu der Ende-zu-Ende-Verschlüsselung mit AES hinzukommt, arbeitet das Verfahren ab einer gewissen Paketgröße schneller und effizienter. Dies liegt an dem Größenvorteil der zu verschlüsselnden Daten. Die Koeffizienten machen gegenüber den Nutzdaten einen viel kleineren Anteil am Paket aus und sind deswegen schneller verschlüsselt. Für realistische Größen, wie z. B. $h = 16$, $w = 1\,400$ Byte und einer AES-Blockgröße von 128 Bit, bedeutet SPOC zwar eine Verdopplung der Codierung und Decodierung von 16×16 Matrizen, jedoch werden dafür pro Generation statt $1\,392$ Nachrichtenblöcken der Generation nur 16 Koeffizientenblöcke verschlüsselt und wieder entschlüsselt.

Eine Möglichkeit, den Übertragungsmehraufwand zu optimieren wurde in „Secure Network Coding with Minimum Overhead Based on Hash Functions“ [AL09] vorgestelllt. Dabei wird die Menge der Koeffizientenvektoren eingeschränkt und die Koeffizienten durch eine kryptographische Hashfunktion mit nur einem Schlüssel erzeugt. Allerdings ergibt sich daraus ein hoher Aufwand für die Berechnung der Hashwerte an allen Knoten im Netzwerk, um auf die Koeffizienten zu kommen. Diese Verschiebung zu niedrigerem Kommunikationsmehraufwand und dafür erhöhtem Berechnungsaufwand ist jedoch nicht die effizienteste Lösung. Auch Vorschläge, die Codierungskoeffizienten mit einer homomorphen Funktion zu verschlüsseln, haben das Problem, einen geringeren Übertragungsaufwand mit einem erhöhten Rechenaufwand zu erkaufen.

Ein Verfahren mit relativ geringem Berechnungsaufwand und geringem Übertragungsaufwand ist P-Coding. Dabei beinhaltet das in „P-Coding: Secure Network Coding against Eavesdropping Attacks“ [ZJL+10] vorgestellte Verfahren eine neue Herangehensweise, um das Problem einer effizienten homomorphen Verschlüsselung zu lösen. Dafür wählten die Autoren ein zum Codieren orthogonales Verfahren: das Permutieren aller Nachrichten einer Generation. Das bedeutet, dass die codierten Pakete (Koeffizienten und Nutzdaten) einer Generation alle in gleicher Art und Weise permutiert werden. Nach der Übertragung, bei der ohne Probleme das Recodieren angewendet werden kann, werden die Elemente aller empfangener Pakete wieder in die richtige Reihenfolge mittels inverser Permutation gebracht und es kann decodiert werden. Dieses Verfahren ist ebenfalls effizienter als herkömmliche Ende-zu-Ende-Verschlüsselung und ist in Abhängigkeit von der Permutation und den Parametern, wie z. B. der Paketgröße, auch ausreichend sicher.

Es gab auch noch Versuche, den Übertragungsaufwand für Vertraulichkeit sorgende Verfahren weiter zu senken. In „A Novel Lightweight Algorithm for Secure Network Coding“ [WG12] präsentierten die Autoren eine Möglichkeit, anstatt der Koeffizienten nur ein Symbol zu senden, aus dem die anderen Koeffizienten abgeleitet werden können. Dadurch könnte der Kommunikationsaufwand weiter gesenkt werden. Es entsteht jedoch das Problem, dass der Suchraum für die Koeffizienten relativ gering ausfällt, sodass ein Abhörer alle Varianten durchprobieren könnte. Das Verfahren bietet daher eine viel zu geringe Sicherheit.

In „Data Obfuscation with Network Coding“ [HKPW12] wurde auch ein zu SPOC [VLB08] ähnliches Verfahren vorgestellt. Jedoch wurden die Koeffizienten mittels pseudozufälligem Schlüsselstrom verschlüsselt. Da die Seed, welche für den Schlüsselstrom notwendig ist, jedoch vorher an die Empfänger

übermittelt werden muss, ergibt sich leider kein Vorteil zu SPOC, da auch dort der Schlüssel für mehrere Generationen genutzt werden kann.

Eine Anwendung von P-Coding findet sich in „A Lightweight Encryption Scheme for Network-Coded Mobile Ad Hoc Networks“ [ZLJ+14]. Die Arbeit untersuchte das Ziel, in mobilen Ad-Hoc-Netzwerken nicht nur eine möglichst schnelle Übertragung zu ermöglichen, sondern auch besonders energieeffizient zu übertragen. Eine experimentelle Untersuchung zeigt neben dem Geschwindigkeitsvorteil auch den verringerten Energiebedarf gegenüber einer Ende-zu-Ende-Verschlüsselung mit AES. Weiterhin umfasst die Arbeit eine Sicherheitsanalyse der algebraischen Sicherheit und kommt zu dem Schluss, dass diese für starke Angreifer nicht ausreichend ist.

Es wird ersichtlich, dass es Verfahren gibt, bei denen die algebraische Sicherheit ausgenutzt werden kann, um ein effiziente und vertrauliche Kommunikation zu ermöglichen. Auch wenn der Fokus der Arbeit auf der Vertraulichkeit, der Integrität und der Verfügbarkeit liegt, soll im nächsten Teil der Vollständigkeit halber noch auf weitere Sicherheitsaspekte und Möglichkeiten bei der Netzwerkcodierung eingegangen werden. Weiterhin enthalten die vorgestellten Verfahren auch Techniken, die ebenfalls für die 3 Hauptschutzziele angewandt werden könnten.

2.3.3 Sonstige Sicherheitsbetrachtungen

2.3.3.1 Anonymität

Wird versucht, Netzwerkcodierung und Anonymität zusammenzubringen, stößt man auf ein grundsätzliches Problem. Bei der Anonymität möchte man unter anderem gewährleisten, dass Nachrichten und Ströme verschiedener Sender, die in einen Knoten eingehen, nicht den Nachrichten, die von den Knoten ausgesendet werden, zuzuordnen sind. Bei der Netzwerkcodierung auf der anderen Seite hat man sowohl einen Codierungsvektor, in dem festgehalten wird, wie die nativen Nachrichten miteinander kombiniert wurden, als auch eine eindeutige Generations-ID o in jedem Paket. Es sollen nun Verfahren vorgestellt werden, die diese unterschiedlichen Anforderungen zusammenbringen. Bei der Anonymisierung wird auf zwei grundsätzliche Verfahren, das DC-Mix-Netzwerk [Cha88] oder das Onion-Routing [RSG98], zurückgegriffen.

In der Arbeit „An Efficient Privacy-Preserving Scheme against Traffic Analysis Attacks in Network Coding“ [FJZS09] wird erläutert, wie man die Netzwerkcodierung für das häufig benutzte Onion-Routing anwendbar machen kann. Die Idee ist die gleiche wie bei der Absicherung der Vertraulichkeit: Der Codierungsvektor muss verschlüsselt werden. Dazu wird eine homomorphe Verschlüsselung (homomorphic encryption function – HEF) auf den Codierungsvektor angewendet. Die Generations-ID wird durch ein sicheres Routing-Verfahren unkenntlich gemacht. Der Netzwerkfluss ist anschließend nicht mehr nachvollziehbar. Mit der Anonymität wird gleichzeitig auch Vertraulichkeit erreicht, weshalb es in [ZLJ+14] auch als alternatives Verfahren zu P-Coding aufgezeigt wurde. „Network Coding Based Privacy Preservation against Traffic Analysis in Multi-Hop Wireless Networks“ [FJZ+11] ist eine Nachfolgearbeit zu [FJZS09] und beinhaltet zusätzliche Sicherheitsbetrachtungen, Simulationen und Auswertungen.

Ein anderer Ansatz wurde in „Anonymous Communication with Network Coding against Traffic Analysis Attack“ [WWW+11] dargestellt. Das Verfahren ALNCode (Anonymous Linear Network Coding) nutzt die Idee, dass über das Kombinieren und Mischen verschiedener Codierungsvektoren der Codie-

rungsvektor anonymisiert werden kann. Es findet dadurch eine effektive Verschleierung der zusammengehörenden Pakete von Ein- und Ausgang statt.
Das ebenfalls zu Onion-Routing kompatible ANOC-Verfahren wurde in der Arbeit „ANOC : Anonymous Network-Coding-Based Communication with Efficient Cooperation“ [ZLJ+12] erläutert. Das Verfahren basiert auf Inter-Session-Netzwerkcodierung, wie z. B. COPE [KRH+06], welches in der Urform nicht mit Onion-Routing kompatibel ist, da für das Bilden der Kombinationen Informationen über die Koeffizienten notwendig sind, diese aber auch Rückschlüsse auf den Ursprung zulassen. Bei ANOC werden unter den Knoten Sitzungschlüssel ausgetauscht und das behelfsmäßige Dechiffrieren von mitgehörten Nachrichten eingeführt. Dies ermöglicht eine gewisse Kompatibilität zu dem Mithören und Codieren bei COPE.
Bei DC-Netzwerken wird häufig Dummy-Verkehr gesendet, um eine Zuordnung von Ein- und Ausgängen allein durch die Nachrichtenmenge zu verhindern. In „Priv-Code: Preserving Privacy Against Traffic Analysis through Network Coding for Multihop Wireless Networks“ [WXL12] wird die Idee des Priv-Codes (Privacy by NC) vorgestellt. Die Idee ist, anstatt zufälligen Dummy-Verkehr zu erzeugen, netzwerkcodierte Daten zu übermitteln. Dies hat den Vorteil, dass die Nachrichtenmenge auf den Ein- und Ausgängen konstant gehalten werden kann, zusätzlich aber auch ein höherer Durchsatz und eine bessere Robustheit im DC-Mix-Netzwerk erzielt wird.
Ein ähnlicher Ansatz ist auch in „Hide and Code: Session Anonymity in Wireless Line Networks with Coded Packets“ [SPLB12] zu finden. Wie bei [WXL12] werden codierte Pakete statt Dummy-Pakete verschickt. Die Arbeit umfasst auch eine Analyse der Latenzen und des Energieverbrauchs. Dabei wurde gezeigt, dass ein Trade-off zwischen Anonymität und Leistung sowie Energieverbrauch möglich ist, sodass das Verfahren entsprechend den Anforderungen parametrisiert werden kann.
Ganz ohne die Nachricht oder die Codierungsvektoren zu verschlüsseln, arbeitet ULNC (Untraceable Linear Network Coding). Das in „ULNC: An Untraceable Linear Network Coding Mechanism for Mobile Devices in Wireless Mesh Networks“ [WLW+16] vorgestellte Verfahren soll die Nachverfolgbarkeit in vermaschten Funknetzwerken verhindern und dabei möglichst effizient sein. Die Idee ist, die möglichen Codierungsvektoren mithilfe von hinreichenden und notwendigen Bedingungen einzuschränken, sodass Zwischenknoten immer nicht nachvollziehbare Kombinationen erzeugen. Dies passiert dadurch, dass eine lineare Korrelation zu anderen (nicht verwendeten) Codierungsvektoren gegeben ist. Damit wird Nachverfolgung deutlich erschwert.
Die vorgestellten Verfahren liegen zwar nicht im Fokus der Arbeit, aber die Verfahren zur Anonymität erzeugen immer auch Vertraulichkeit der Daten. Dadurch sind Entwicklungen in diesem Gebiet insgesamt interessant, um die Techniken auch direkt für die Vertraulichkeit anwenden zu können.

2.3.3.2 Sicherheit auf der Übertragungsschicht

Ein spezielles Gebiet im Bereich der Netzwerkcodierung ist die Codierung auf der Übertragungsschicht. Dabei ist das physische Medium, wie z. B. die Funkwellen, auf denen übertragen wird, gemeint. Zuerst sollen dazu kurz die Grundlagen der Netzwerkcodierung auf der Übertragungsschicht erläutert werden. Im Anschluss daran wird dann gezeigt, wie man diese für eine vertrauliche Übertragung ausnutzen könnte.
Alles begann mit einer Idee, welche in „Physical-Layer Network Coding“ [ZLL06] veröffentlicht wurde. Das Systemmodell, welches hinter der Idee steht, sei in Abbildung 2.7 dargestellt. Zwei Knoten s_1 und

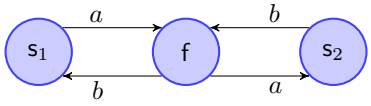


Abbildung 2.7: Systemmodell zur Netzwerkcodierung auf der Übertragungsschicht. s_1 und s_2 wollen über den Zwischenknoten f Nachrichten a und b austauschen.

s_2 wollen sich gegenseitig 2 Nachrichten über das Funkmedium übermitteln. Dabei sind diese Knoten aber so weit voneinander entfernt, dass diese auf die Hilfe eines Zwischenknotens f angewiesen sind, der die Nachrichten weiterleiten kann. Bei herkömmlichen Kanälen dauert die Übertragung 4 Schritte. Erst sendet einer der Sender, dann der andere und dann überträgt f eine Nachricht an den einen Sender und anschließend die andere an den nächsten Sender.

Wird nun die Netzwerkcodierung am Zwischenknoten genutzt, kann eine Übertragung gespart werden, denn nach dem sequentiellen Senden von a und b kann die lineare Kombination $a \oplus b$ auf dem Broadcast-Medium gesendet werden. Dadurch können sowohl s_1 und s_2 die fehlende Nachricht decodieren.

Die Idee in [ZLL06] ist nun, auch a und b gleich zu Anfang parallel zu übertragen. Die Wellen im Funkmedium können sich so überlagern, dass eine Linearkombination in der Luft durchgeführt wird. Dies reduziert die Übertragung auf 2 Schritte im aufgezeigten Systemmodell.

In der Arbeit „Compute-and-Forward: Error-Correcting Codes for Wireless Network Coding on the Physical Layer“ [NG08] wurde die Grundidee für diese sogenannte analoge Netzwerkcodierung, also der Linearkombination in der Luft, weiter optimiert. Zum einen wurde eine mögliche Kanalcodierung mit feinen und groben Gitterstrukturen beschrieben. Zum anderen wurde auch das Compute-and-Forward-Verfahren eingeführt. Wenn 2 Signale zur gleichen Zeit bei einem Empfänger ankommen, wird das eine Signal als Rauschen betrachtet, um das andere Signal decodieren zu können. Wenn man aber versucht, anstatt beide Signale einzeln das kombinierte Signal zu decodieren (im Sinne der Kanalcodierung), dann erreicht man eine höhere Decodierungsrate. Das bedeutet, man könnte die Parameter so wählen, dass ein Zwischenknoten nur die Kombination, aber nicht die einzelnen Signale decodieren kann. Dies ist eine interessante Eigenschaft für das Schutzziel Vertraulichkeit.

Untersuchungen zur Sicherheit auf der Übertragungsschicht gibt es schon sehr lange. In „The Wire-Tap Channel“ [Wyn75] wurde die Idee geboren, ohne Kryptographie für Vertraulichkeit zwischen zwei Knoten zu sorgen. Die Grundannahme war, dass ein Mithörer schlechtere Kanalcharakteristiken als der wirkliche Empfänger hat. Sei die Rate des Senders zum Empfänger R_r, die vom Sender zum Abhörer R_e. Es gilt $R_\mathsf{r} > R_\mathsf{e}$. Wenn man Nutzdaten mit einer Rate von maximal $R_\mathsf{r} - R_\mathsf{e}$ übermittelt und den Rest mit Entropie, also Zufall, auffüllt anstatt nur Nutzdaten zu senden, dann kann der Abhörer e keinerlei Informationen gewinnen. Die Übertragung der Nutzdaten vom Sender zum Empfänger erfolgt informationstheoretisch, d. h. auch gegen rechentechnisch unbeschränkte Angreifer, sicher.

Eine Kombination der beiden vorgestellten Ansätze ist in der Arbeit „Providing Secrecy with Lattice Codes“ [HY08] zu finden. Zusätzlich wurde das Verfahren auch für Übertragungen über mehrere Zwischenknoten erweitert. Die Sicherheit ist gegeben, solange sich alle beteiligten Knoten protokollgerecht verhalten.

Auch „Secure Compute-and-Forward in a Bidirectional Relay“ [VKT15] nimmt sich des Themas an und zeigt, dass die höhere Compute-and-Forward-Rate abzüglich der Rate, mit der Signale einzeln decodiert werden können, die sicher übertragbare Datenrate bei der Kommunikation ist.

Abschließend sei nochmal erwähnt, dass diese Verfahren nicht im Fokus dieser Arbeit stehen. Dies liegt zum einen an der eingeschränkten Praktikabilität. Es existieren nur wenige reale Umsetzungen, die auch nur zu Demonstrationszwecken fungieren, um zu zeigen, dass die Technik prinzipiell funktioniert. Zum anderen ist die Anwendbarkeit zur Zeit nur im Funkbereich gegeben und es werden hohe Anforderungen an die Synchronizität der Übertragungen gestellt. Es ist jedoch interessant, dass es fernab von den Sicherheitsmechanismen auf der Netzwerkschicht auch andere Möglichkeiten gibt. Eventuell lassen sich dadurch auch in Zukunft kombinierte Algorithmen generieren, die schichtübergreifende Sicherheit erzeugen.

3 Sichere und effiziente Netzwerkcodierung basierend auf Kryptographie für verschiedene Netzwerke

3.1 Einführung

3.1.1 Grundlagen

In diesem Kapitel soll es um Möglichkeiten gehen, die Integrität der übertragenen Pakete und damit die Verfügbarkeit der Daten am Empfänger bei der Netzwerkcodierung abzusichern. Dazu werden Verfahren, die auf kryptographischen Ansätzen basieren, untersucht und ihre Effizienz in verschiedenen Netzwerken verglichen. Dieses Kapitel basiert hauptsächlich auf den Arbeiten „Efficiency of Secure Network Coding Schemes“ [FPF12b], „Communication Overhead of Network Coding Schemes Secure against Pollution Attacks“ [FPF12a] sowie „Secure Network Coding: Dependency of Efficiency on Network Topology“ [PF13b].
Die Absicherung der Integrität der Daten ist wichtiger Bestandteil der Informationssicherheit. Nur wenn sichergestellt werden kann, dass Daten unverfälscht vom Sender zum Empfänger weitergeleitet wurden, ist die Information für den Empfänger nützlich. Sollte es dennoch zu Verfälschungen kommen, muss dies eindeutig erkennbar sein.
Aufgrund des Zusammenfügens von Nachrichten ist es bei der Netzwerkcodierung zwingend erforderlich Maßnahmen gegen Verschmutzungsattacken (Pollution Attacks) zu ergreifen. Im folgenden Kapitel wird dabei die Gruppe der auf Kryptographie basierenden Verfahren untersucht. Im Gegensatz zu Verfahren, die auf Codierungstheorie basieren, verhindern diese Verfahren Verschmutzungen aktiv, da Verfälschungen direkt an den Zwischenknoten erkannt werden. Eine Verhinderung aktiver Angriffe ist zwar generell nicht möglich, dafür aber eine Verhinderung der Ausbreitung und Verschmutzung im Netz.
Allerdings gibt es eine große Anzahl von verschiedenen Verfahren, bei denen nicht ersichtlich ist, wie ähnlich diese zueinander sind. Weiterhin gibt es zwar Vergleiche zwischen ähnlichen Verfahren, aber keine oder unzureichende Vergleiche zwischen unterschiedlichen Verfahrensgruppen. Daher ist ein Ziel dieses Kapitels, Verfahren nach der Verfahrensart zu vergleichen, zu gruppieren und anschließend die Effizienz an verschiedenen Netztopologien zu evaluieren.

3.1.2 Fragestellungen

Es ergeben sich folgende Fragestellungen, die in diesem Kapitel behandelt werden:

- Welche Hauptgruppen von Verfahren gibt es, die auf Kryptographie basieren?
- Welche dieser Gruppen bietet die höchste Effizienz bezüglich verschiedener Parameter?
- Welchen Einfluss hat das Netzwerk auf die Effizienz?

Um diese Fragen zu beantworten, wird im Folgenden das Systemmodell mit den Netzwerken, den Effizienzparametern sowie das Angreifermodell vorgestellt. Danach wird die Methodik erklärt, welche für die Untersuchungen zum Einsatz kam. Anschließend wird die Klassifizierung der Verfahren erläutert, bevor die Ergebnisse aufgezeigt und interpretiert werden. Eine kurze Zusammenfassung beendet das Kapitel und damit die Untersuchungen hinsichtlich der Integrität und Verfügbarkeit.

3.2 Systemmodell

3.2.1 Netzwerk

Um die Effizienz der Verfahren möglichst einfach, aber auch realistisch ermitteln und vergleichen zu können, muss ein Netzwerk vorhanden sein, für das die Effizienzparameter ermittelt werden können. Für zur Zeit übliche Sterntopologien, wie sie beispielsweise im Internet anzutreffen sind, kann das Potenzial der Netzwerkcodierung nur bedingt gezeigt werden. Deswegen werden eher vermaschte Netzwerke als Grundlage genommen, welche beispielsweise in Sensor-Netzwerken vorkommen und in Zukunft durch die Mobilisierung der Geräte eine größere Rolle spielen werden.
Damit nicht nur eine Aussage für ein Beispiel getroffen werden kann, wurden 3 parametrisierte Modelle als Netzwerk herangezogen. Die unterschiedlichen Modelle sollen dabei den Einfluss verschiedener Eigenschaften, wie z. B. die Netzwerkkapazität, welche durch den minimalen Schnitt ermittelbar sind, oder die Größe (Knotenanzahl, Länge, Breite) des Netzwerks, aufzeigen.
Ausgehend von diesen Modellen und den erzielten Ergebnissen wurde zusätzlich noch ein generalisiertes Modell entwickelt, um Aussagen über die Effizienz für eine möglichst große Anzahl von Netzwerktopologien machen zu können. Zusätzlich zeigt dieses Modell auch die Unterschiede zwischen den Gruppen von Verfahren auf.
Bezüglich der Parametrisierung der Netzwerkcodierung wurde bei den Untersuchungen davon ausgegangen, dass genau eine Generation mit der Generationsgröße h übertragen wird, welche dem maximalen Fluss/minimalen Schnitt des Netzwerks entspricht. Da für eine erste Analyse zunächst von einer Übertragung ohne Fehler bzw. Angriffe ausgegangen wird, wird somit jede Kante und jeder Knoten im Netz genau einmal pro Generation genutzt. Damit soll ein Vergleich der Verfahren bezüglich des besten Falls ermöglicht werden.
Da der Schwerpunkt der Untersuchungen auf der Kommunikation liegt, wird angenommen, dass die Knoten im Netzwerk so leistungsstark sind, dass die Geschwindigkeit nur durch die Übertragungsrate und nicht durch die Berechnungen an den Knoten begrenzt wird.

Parametrisierte Modelle Das erste Modell, dargestellt in Abbildung 3.1, soll den Einfluss der Netzwerkbreite d untersuchen. Dazu übermittelt ein Sender s Nachrichten an d Empfänger r_j, mit $1 \leq j \leq d$ jeweils über 2 Ebenen von Zwischenknoten $\mathsf{f}_{i,j}$. Zwischenknoten der ersten Ebene $\mathsf{f}_{1,j}$ erhalten eine Nachricht von s und leiten diese an ihren direkten Nachfolger $\mathsf{f}_{2,j}$ sowie dessen rechten Nachbarn $\mathsf{f}_{2,j+1}$ weiter. Für $\mathsf{f}_{1,d}$ gilt $\mathsf{f}_{2,1}$ als zweiter Empfänger. Die Zwischenknoten auf der zweiten Ebene (re)codieren ihre 2 Nachrichten und schicken diese, analog zur ersten Ebene, an den direkten Nachfolger und dessen rechten Nachbarn. Dabei spielt es für die Untersuchung keine Rolle, ob zweimal die gleiche Nachricht oder 2 unterschiedlich codierte Nachrichten versendet werden. Am Ende erhält jeder Empfänger dadurch

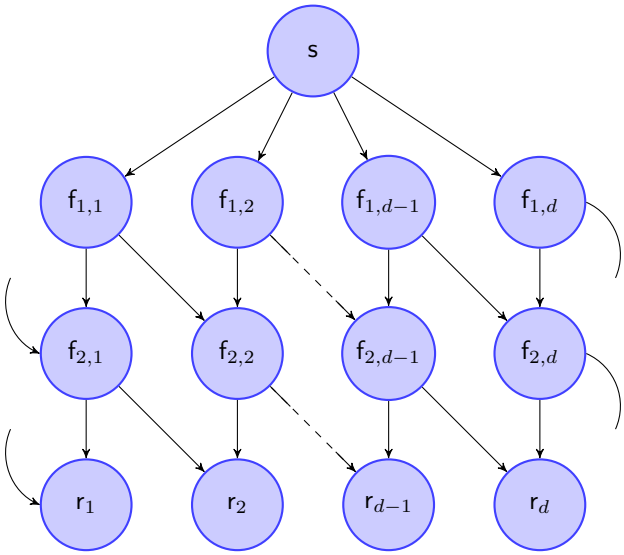


Abbildung 3.1: Modell 1. Die Anzahl der Hops ℓ ist 3. Die Netzwerkbreite d und damit auch die Anzahl der Empfänger $|\mathbf{R}|$ kann variiert werden.

2 codierte Nachrichten. Die Generationsgröße ist gleich dem maximalen Fluss/minimalen Schnitt. Somit gilt $h = 2$.

Abbildung 3.2 zeigt das zweite Modell. Hierbei soll der Einfluss der Tiefe des Netzwerks oder, alternativ ausgedrückt, die Anzahl der Hops ℓ vom Sender zum Empfänger untersucht werden. Dabei sendet ein Sender s je eine Nachricht an 2 Zwischenknoten $\mathsf{f}_{1,1}$ und $\mathsf{f}_{1,2}$. Diese leiten die Nachricht jeweils an ihre beiden Nachfolger $\mathsf{f}_{2,1}$ und $\mathsf{f}_{2,2}$, welche die Nachrichten (re)codieren und wieder 2 Nachrichten an ihre Nachfolger senden, bis jeweils 2 Nachrichten nach ℓ Hops bei den beiden Empfängern r_1 und r_2 ankommen. Auch hier gilt $h = 2$.

Modell 3, dargestellt in Abbildung 3.3, ist angelehnt an ein Netzwerk aus „Network Coding Applications“ [FS07] und stellt eine Verallgemeinerung des Butterfly-Netzwerks (Abbildung 2.1) dar. Dabei sollten die Vorteile der Netzwerkcodierung gezeigt werden, wenn keine direkte Verbindung zum Empfänger, sondern nur über einen Flaschenhals möglich ist. Im abgewandelten Fall versucht ein Sender s, jeweils d Nachrichten an d Empfänger gleichzeitig zu senden. Dafür sendet er d Nachrichten an eine Ebene von d Zwischenknoten. Diese haben jedoch jeweils nur $d-1$ Kanten zu den Empfängern $\mathbf{R}$, wobei jeweils die Kante $(\mathsf{f}_{1,j}, \mathsf{r}_j)$ nicht vorhanden ist. Dafür gibt es eine Verbindung von allen Zwischenknoten auf erster Ebene über 2 Zwischenknoten mit Flaschenhals, d. h. nur einer Kante, zu allen Empfängern. Um hier Vorteile durch die Netzwerkcodierung zu erreichen, muss der Knoten vor dem Flaschenhals (re)codieren. Durch diese Konstruktion ergibt sich ein maximaler Fluss von d. Somit gilt $h = d$ für Modell 3.

Generalisiertes Modell Nach Experimenten mit den 3 zwar parametrisierten, aber dennoch recht statischen Netzwerken sollten auch allgemeinere Aussagen zur Effizienz ermöglicht werden. Dazu wurde ein generalisiertes Netzwerkmodell entwickelt. Netzwerke lassen sich durch eine Vielzahl an Parametern beschreiben, wie z. B. die Anzahl der (Zwischen-)Knoten, Empfänger, Kanten oder der Eingangskanten pro Knoten. All diese Parameter haben einen mehr oder weniger großen Einfluss auf die Effizienz und machen ein allumfassendes generalisiertes Modell sehr kompliziert.

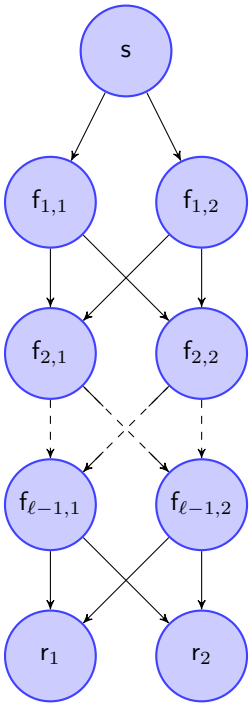


Abbildung 3.2: Modell 2. Das Netzwerk hat eine feste Breite von $d = 2$. Dafür ist die Anzahl der ℓ Hops beliebig wählbar.

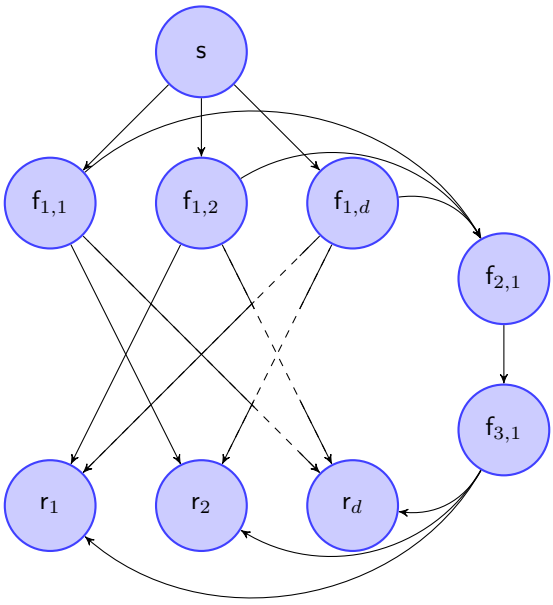


Abbildung 3.3: Modell 3. Eine direkte Kommunikation der Zwischenknoten auf der ersten Ebene $f_{1,j}$ mit den direkten Nachfolgern r_j ist nur über eine gemeinsam mit allen d Zwischenknoten erster Ebene genutzte Verbindung (Flaschenhals) möglich. Die variable Breite d entspricht der Anzahl der Empfänger $|\mathbf{R}|$ und dem maximalen Fluss des Netzwerks und damit der Generationsgröße h.

Da der Einfluss der meisten Parameter, wie z. B. Anzahl der ein- und ausgehenden Kanten, die Breite oder auch die Regelmäßigkeit des Netzwerks, jedoch recht gering ausfällt oder bezüglich der unterschiedlichen Verfahren relativ gleich ist, wie es z. B. bei der Anzahl der Knoten der Fall ist, werden nur die einflussreichsten Parameter für ein Modell herangezogen. Das hat mehrere Vorteile. Zum einen lässt sich das Modell leichter entwickeln und hat trotzdem eine gewisse Aussagekraft. Zum anderen ist die Anwendbarkeit auf eigene Netzwerktopologien einfacher, da nur wenige Parameter berücksichtigt werden müssen. Weiterhin reicht ein vereinfachtes Modell aus, um Unterschiede zwischen den Verfahren oder Gruppen von Verfahren aufzuzeigen.
Nach einigen Voruntersuchungen, insbesondere Experimenten mit den parametrisierten Modellen, lässt sich festhalten, dass 2 Parameter besonders großen Einfluss auf die Effizienz und damit die Wahl des Verfahrens haben:

- Die Multicast-Kapazität und damit die Generationsgröße h
- Die maximale Anzahl der Hops ℓ von Sender zu Empfänger

Da für einige Verfahren auch die Anzahl eingehender und ausgehender Kanten relevant ist, wird diese Größe mit etwa h grob abgeschätzt. Dadurch vereinfacht sich die Beschreibung eines gegebenen Netzwerkmodells auf nur zwei relevante Größen, h und ℓ.

3.2.2 Effizienzparameter

Um die Effizienz der Kommunikation beziffern zu können, müssen verschiedene Aspekte betrachtet werden. Da im Folgenden auf die Abhängigkeit von Netzwerken geschaut werden soll, werden nur Effizienzparameter betrachtet, welche nicht von den konkreten Implementierungen und den Umsetzungen der Operationen auf den zugrunde liegenden Systemen abhängen. Daher werden z. B. die zusätzlichen Operationen und Speicheraufwände ignoriert. An die Knoten wird nur die Anforderung gestellt, dass sie eine ausreichende Berechnungsstärke besitzen und die Verarbeitung der Pakete schneller als eine Übertragung zwischen zwei Knoten erfolgt. Um die Effizienz der Verfahren zu beschreiben, wurden daher die folgenden 3 Effizienzparameter ausgewählt:

- Relative Nutzdaten P
- Anzahl der Sendeoperationen im Netzwerk O
- Anzahl der Zeiteinheiten T

Die relativen Nutzdaten P errechnen sich als Quotient aus der Menge der Nutzdaten, welche den Empfängern nach dem Decodieren zur Verfügung stehen, und der Menge aller insgesamt vom Sender gesendeten Daten. Dabei werden sowohl Daten in Zusatzpaketen, die weitere Informationen enthalten, als auch Daten, die in den normalen Paketen enthalten sind, berücksichtigt. Dazu gehören z. B. die Generations-ID (8 Byte), Signaturen, MACs oder ähnliches. Da die Generationsgröße h variabel ist und somit unterschiedlich viele Daten pro Generation übertragen werden, wird auf diese relative Metrik zurückgegriffen. Insgesamt spiegelt die Größe der relativen Nutzdaten P den Mehraufwand für die Kommunikation und die Netzwerklast wider.

Der zweite Effizienzparameter ist die Anzahl der Sendeoperationen im Netzwerk O. Dabei wird davon ausgegangen, dass jeder Knoten, der ein Nachrichtenpaket oder ein eventuelles Zusatzdatenpaket sendet, genau eine Sendeoperation ausführt. Wird diese von einem anderen Knoten empfangen und weitergeleitet, so zählt dies als weitere Sendeoperation. Diese Größe dient zur groben Abschätzung der Last im gesamten Netz und damit zur Abschätzung des Energiebedarfs für das Senden einer Generation, welcher mit einer erhöhten Anzahl an Sendeoperationen steigen sollte. Da Sendeoperationen auch parallel ausgeführt werden können, ist die Anzahl an Sendeoperationen O immer größer oder gleich dem dritten Effizienzparameter, der Anzahl der für die Übertragung benötigten Zeiteinheiten T.
Die Anzahl der Zeiteinheiten T hängt hauptsächlich von der Anzahl der Hops ℓ im Netzwerk ab. Eine Zeiteinheit umfasst dabei die Zeit, die vergeht, um alle Nachrichten zu empfangen, diese miteinander zu (re)codieren und anschließend auf allen ausgehenden Kanten zu senden. Dabei sind auch alle notwendigen Zeiten für Prüfen oder Erzeugen von Signaturen oder anderen Sicherungsmaßnahmen enthalten. Da aufgrund der angenommenen Berechnungsstärke der Knoten davon auszugehen ist, dass die Berechnungen weniger Zeit als die Übertragungen benötigen, sind diese (bis auf kleine Latenzunterschiede) die Zeitfaktoren für eine Übertragung. Daher wurde als grobe Abschätzung das Konzept der Zeiteinheiten gewählt. Diese Größe ermöglicht einen Vergleich der unterschiedlichen Verfahren (z. B. symmetrisch oder asymmetrische Kryptographie), ohne spezielle Annahmen über Implementierung und System treffen zu müssen. Insgesamt sollten sich diese Zeiteinheiten in etwa proportional zu den tatsächlich benötigten Zeiten für eine reale Übertragung verhalten.

3.2.3 Angreifer

Die Verfahren, die untersucht werden, haben zum Ziel, Modifikationen eines aktiven Angreifers erkennbar zu machen. Dabei wird von einem Angreifer ausgegangen, welcher alle Informationen in den Paketen und Zusatzpaketen auf Protokollebene ändern kann. Es wird angenommen, dass ein möglicher Angreifer keinerlei Änderungen auf niedrigeren Schichten, wie der Übertragungsschicht o. ä., vornimmt. Weiterhin sei dem Angreifer erlaubt, Datenpakete zu blockieren oder zu verzögern. Dabei gelte dies nur eingeschränkt für die Zusatzinformationen, die bei einigen Verfahren gesendet werden. Da hier meist ein Broadcast zum Einsatz kommt, wird davon ausgegangen, dass diese Informationen auf einem anderen Weg zu einem beliebigen Knoten kommen können.
Der Angreifer selbst ist beschränkt hinsichtlich seiner Ressourcen und kann daher symmetrische oder asymmetrische Kryptographieverfahren mit ausreichender Schlüssellänge nicht brechen. Alle im Folgenden beschriebenen Sicherungsmaßnahmen werden daher als ausreichend betrachtet, um eine Veränderung an den Daten durch einen Angreifer zu erkennen.
Weiterhin gilt, dass der Angreifer potentiell jeden Knoten im Netz kontrollieren kann, da es sich beispielsweise um eine Schadsoftware handelt, die zwischen verschiedenen Knoten migriert wird. Es wird angenommen, dass Sender sowie Empfänger in dem gezeigten Szenario niemals Angreifer sind, da diese ihre Kommunikation schützen wollen. Im Übrigen könnten diese immer die eigene Kommunikation verhindern und sei es durch ein Verwerfen von Paketen.
Das Ziel des Angreifers ist, die Integrität und Verfügbarkeit der Kommunikation mittels Verschmutzungsangriff zu stören. Dabei gilt ein solcher Angriff als erfolgreich, wenn die Decodierbarkeit an den Empfänger schlechter ist, als es durch das Verwerfen des Pakets möglich gewesen wäre. Da durch das Erkennen und Aussortieren von verfälschten Paketen zwar nicht der Angriff auf das Paket, wohl aber

die Auswirkungen und die Verschmutzung im Netz abgewehrt werden können, wird bei der Analyse der Verfahren davon ausgegangen, dass es zu keinerlei Störung kommt. Zwar würde ein Angriff je nach Netzwerk und Lage des Angreifers die Übertragung eventuell langsamer machen, aber es würde keine Verschmutzung im Netzwerk entstehen. Bei dem im Folgendem vorgestellten Analysen wird zunächst der grundlegende Mehraufwand der Verfahren für eine Übertragung ohne Angriffe betrachtet. Dabei wird erwartet, dass der Aufwand für die sicheren Verfahren höher als bei ungesicherter Netzwerkcodierung ausfällt. Damit soll der Aufwand für die Integration von Sicherheit bewertet und ein Vergleich sicherer Verfahren ermöglicht werden.

3.3 Methodik

Zur Untersuchung der Effizienz wurde eine theoretische Analyse durchgeführt. Es wurde die Effizienz mit Sicherungsmaßnahmen, aber ohne aktive Angriffe, betrachtet. Dadurch ließen sich zum einen die Effizienzparameter, auch für die verschiedensten Parameterkombinationen, durch Formeln bestimmen. Zum anderen konnten die Einflüsse der Variablen in den Formeln gut sichtbar gemacht werden.

Es wurden alle ausgewählten Verfahren analysiert und die in der Literatur gängigen Parameterwerte (Länge der Signatur, Körpergröße etc.) festgelegt. Danach wurden die notwendige Paketstruktur sowie eventuelle zusätzliche Pakete und deren Größe festgelegt. Darauf aufbauend wurden dann die einzelnen Werte für die 3 Effizienzparameter für alle Verfahren und Modelle ermittelt.

Für das generalisierte Modell wurden entsprechende Formeln erarbeitet. Diese dienten als Grundlage für die Visualisierung in den Ergebnissen und konnten damit die Abhängigkeit der Effizienz von den einzelnen Parameterkombinationen darlegen.

Dazu wurden die Effizienzparameter für den Bereich von $2 \leq d \leq 20$ und $2 \leq \ell \leq 20$ für die parametrisierten Modelle bzw. im Bereich von $2 \leq h \leq 40$ und $2 \leq \ell \leq 200$ für das generalisierte Modell untersucht. Für die Paketgröße soll von einer Maximalgröße von $w \leq 1\,400$ Bytes ausgegangen werden, da diese Pakete über gängiges TCP/IP übertragen werden können, ohne dass Segmentierung auftritt.

Insgesamt müssen im Einzelfall natürlich die genauen Sicherheitsannahmen entsprechend aktueller Empfehlungen und Parameter angeschaut werden, da die Ergebnisse anders aussehen würden, wenn z. B. eine kleinere oder größere Signatur zum Einsatz käme. Unabhängig davon ist das Ziel der Untersuchung nicht die Bestimmung genauer Werte, sondern Tendenzen aufzuzeigen, bei welcher Art von Netzwerk welche Verfahren gut geeignet sind.

3.4 Klassifizierung existierender Verfahren

In Abschnitt 2.3.1.2 zum Stand der Verfahren basierend auf Kryptographie wurden verschiedene Verfahren vorgestellt. Die dabei vorgenommene Einteilung, basierend auf den absichernden Maßnahmen, wie Checksummen, Hashwerte, MACs und Signaturen, wird auch als Grundlage für die folgenden Analysen benutzt.

Aus jeder der 4 Klassen wurde ein Verfahren als Vertreter ausgewählt und soll im Folgenden näher beschrieben werden. Dabei spielen Paketformat auf Transportschicht, Ablauf der Kommunikation und eventuelle zusätzliche Pakete die größte Rolle, um die Effizienz zu ermitteln. Neben diesen sollen auch

sinnvolle Werte für notwendige Parameter wie Körpergröße oder Länge der Signatur/Hashes festgelegt werden.

Alle 4 benutzen die Netzwerkcodierung nach dem Verfahren Practical Network Coding (PNC) [CWJ03] als Basis. Das heißt, diese Pakete haben neben den Nutzdaten noch eine Generations-ID o der Größe 8 Bytes, um Wiederholungen zu verhindern, sowie einen Codierungsvektor bestehend aus h Elementen. Im Normalfall wird der zugrunde liegende Körper $\mathbb{F}_{2^8}$ gewählt, sodass genau 1 Byte pro Element benötigt wird. Damit ist die Größe des Codierungsvektors h Bytes, wodurch Nutzdaten in Höhe von $1\,400 - 8 - h$ Bytes pro Paket übertragen werden könnten.

In Tabelle 3.1 sind noch einmal die ausgewählten Verfahren und die wichtigsten Unterschiede für den Vergleich dargestellt.

Tabelle 3.1: Übersicht über die ausgewählten Verfahren

<table>
<tr><td></td><td>Basierend auf asymmetrischer Kryptographie</td><td>Basierend auf Zeitasymmetrie</td></tr>
<tr><td>Zusatzpakete werden benötigt</td><td>Homomorphe Hashwerte [GR06]</td><td>Authentifizierte Checksummen [DCNR09]</td></tr>
<tr><td rowspan="2">Zusatzinformationen werden in die Pakete eingebracht</td><td colspan="2">Homomorphe MACs [Wan10]</td></tr>
<tr><td>Homomorphe Signaturen [GKKR09]</td><td>—</td></tr>
</table>

Checksummen DART [DCNR09] wurde für die Gruppe der auf Checksummen basierenden Verfahren ausgewählt. Das Verfahren nutzt die Zeitasymmetrie nach dem TESLA-Protokoll [PCTS02]. Ansonsten nutzt es die Standardparameter von PNC, sodass es zu keinerlei Anpassungen im Datenpaket kommt.

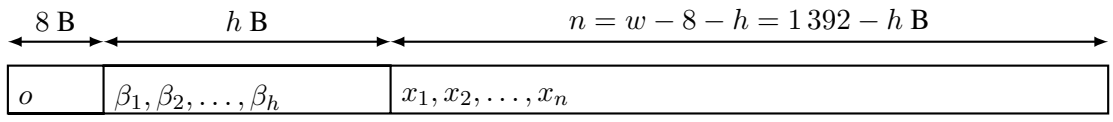


Abbildung 3.4: Struktur des Datenpakets beim Checksummenverfahren.

In Abbildung 3.4 ist das Paketformat für das Checksummenverfahren dargestellt. Für die Nutzdaten bleiben wie bei normalem PNC $1\,392 - h$ Bytes im Paket übrig.

Werden Pakete nun gesendet, können diese von einem (Zwischen-)Knoten empfangen werden. Dieser speichert die Pakete mit einem Zeitstempel in einem Speicher für nicht verifizierte Pakete. Zusätzlich werden in regelmäßigen Abständen Checksummenpakete vom Sender an alle Knoten im Netz verteilt. Mit diesen kann ein Zwischenknoten testen, ob die Pakete, die vor der Erstellung der Checksumme empfangen wurden, gültige Pakete sind. In diesem Fall werden die Pakete in einen Speicher für verifizierte

Pakete überführt. Mit diesen verifizierten Paketen können dann neue Kombinationen erzeugt und ausgesendet werden.
Um diese Verifizierung zu ermöglichen, muss ein Checksummenpaket alle notwendigen Daten enthalten und gegen Angreifer abgesichert sein. Damit eine Zuordnung zu der betreffenden Generation möglich ist, wird die 8 Byte große Generations-ID o dem Paket vorangestellt. Dann folgt die eigentliche Checksumme, welche die Größe eines festgelegten Sicherheitsparameters multipliziert mit der Generationsgröße hat. Bei Untersuchungen in [DCNR09] wurde herausgefunden, dass ein Sicherheitsfaktor von 2 ausreichend sei, weshalb von dieser Größe im Folgenden ausgegangen wird. Zusätzlich beinhaltet das Paket eine Seed s, um eine Matrix $\boldsymbol{H}_s$ zu erstellen, sowie den Zeitstempel $t_{\text{chk}_s(\boldsymbol{X})}$, welcher angibt, wann das Paket erzeugt wurde. Um Verfälschungen durch einen Angreifer zu umgehen, wird entsprechend dem Stand der Technik eine 128 Byte große Signatur (z. B. ECDSA) über alle Inhalte gebildet.

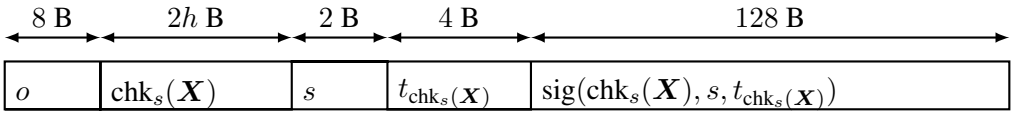


Abbildung 3.5: Struktur des Zusatzpakets beim Checksummenverfahren.

Die Paketstruktur eines Checksummenpakets ist in Abbildung 3.5 dargestellt. Neben der Generations-ID o und der eigentlichen Checksumme wird eine Seed s zur Erstellung von $\boldsymbol{H}_s$ und ein Zeitstempel $t_{\text{chk}_s(\boldsymbol{X})}$ von der Erstellzeit mitgesendet. Um Verfälschungen zu verhindern, wird eine 128 Byte große Signatur angehangen, welche dabei einen Großteil des gesamten Pakets ausmacht.
Für die Erstellung der Checksummenpakete muss der Sender s sich eine zufällige Seed s erstellen. Von dieser kann eine Matrix $\boldsymbol{H}_s$ abgeleitet werden, welche mit der gesamten (uncodierten) Generation $\boldsymbol{X}$ multipliziert wird. Dies ist dann $\text{chk}_s(\boldsymbol{X})$, welches zusammen mit den anderen notwendigen Daten, wie dem Zeitstempel, signiert und anschließend versendet wird. Die Prüfung am Zwischenknoten erfolgt dann über die Prüfung ob die Paketdaten $\overline{\boldsymbol{x}}_i$ multipliziert mit dem aus der Seed erzeugten $\boldsymbol{H}_s$ gleich dem Produkt aus Codierungsvektor $\overline{\boldsymbol{\beta}}_i$ und Checksumme $\text{chk}_s(\boldsymbol{X})$ sind.

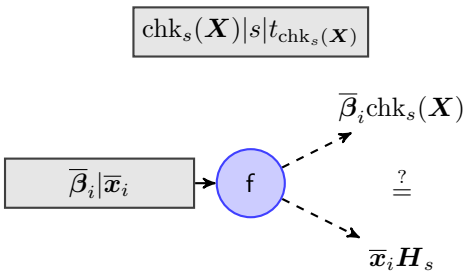


Abbildung 3.6: DART. Überprüfung der Validität eines empfangenen Pakets über die Checksummen.

Abbildung 3.6 zeigt den genauen Ablauf der Prüfung an einem Zwischenknoten f noch einmal schematisch. Wichtig für das Gelingen der Prüfung und eine relativ niedrige Latenz ist die zeitliche Synchronisierung der Knoten. Zwar kann man bei der Prüfung der Zeitstempel ein gewisses Δ_t zur Sicherheit einfügen, jedoch erhöht dies auch die Verzögerungszeiten, die bei der Übertragung zustande kommen.

Hashwerte Für die Absicherung über Hashwerte wurde das Verfahren ausgewählt, welches in „Cooperative Security for Network Coding File Distribution“ [GR06] vorgestellt wurde. Ähnlich zu DART

werden hier nur normale Pakete wie in PNC übermittelt. Die Sicherheit wird dann durch Zusatzpakete bewerkstelligt. Im Fall der homomorphen Hashes gibt es jedoch die Besonderheit, dass nicht in einem kleinen Erweiterungsköper $\mathbb{F}_{2^8}$ gearbeitet wird, sondern in einem großen Primzahlkörper $\mathbb{F}_p$ mit $|p| \approx 256$ Bit. Dies ist für die Sicherheit der kryptographischen Hashes notwendig und führt dazu, dass deswegen jeder Koeffizient im Codierungsvektor 32 Byte groß ist. Dies hat je nach Größe von h eine mehr oder weniger starke Reduzierung der Nutzdaten bei angenommener fester Paketgröße zur Folge.

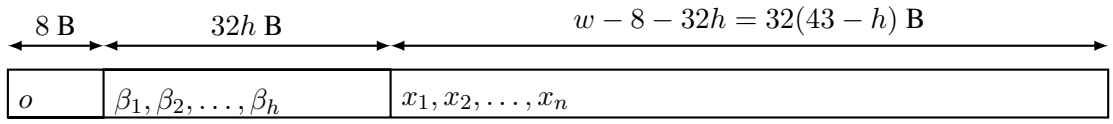


Abbildung 3.7: Struktur des Datenpakets beim Hashverfahren.

In Abbildung 3.7 ist die Paketstruktur beim Hashverfahren dargestellt. Durch den notwendigerweise vergrößerten Körper benötigt jeder Codierungskoeffizient 32 Byte, sodass der Nutzdatenanteil gegenüber PNC geringer ausfällt. Insgesamt bestehen die Nutzdaten aus $n = 43 - h$ Symbolen.
Zur Absicherung wird vor jeder Übertragung einer Generation ein zusätzliches Paket mit den je 128 Byte großen Hashwerten der uncodierten Vektoren $\boldsymbol{x}_i$ übertragen. Um diese zusätzlichen Hashpakete gegenüber aktiven Angriffen zu sichern, werden diese signiert.

Abbildung 3.8: Struktur des Zusatzpakets beim Hashverfahren.

Die Struktur für die Zusatzpakete ist in Abbildung 3.8 dargestellt. Da die maximale Paketgröße $w = 1\,400$ Byte beträgt, passen insgesamt 9 Hashwerte in ein Zusatzpaket, welches zusätzlich signiert ist. Sollte $h > 9$ gelten, müssen mehrere Pakete mit den Hashwerten erstellt werden.
Für die Erstellung der Hashwerte wird eine homomorphe Hashfunktion verwendet, sodass Zwischenknoten sowie der Empfänger die Integrität der Pakete überprüfen können, indem sie die vorher empfangenen Hashwerte mit dem Codierungsvektor des aktuellen Pakets multiplizieren. Nach dem Anwenden der Hashfunktion auf die Daten des aktuellen Pakets kann der Knoten beide Ergebnisse vergleichen und eventuelle Verfälschungen erkennen.
Die Überprüfung der Hashes an einem Zwischenknoten ist in Abbildung 3.9 dargestellt. Zu beachten ist, dass die Hashes der uncodierten Pakete nicht wie bei dem Checksummen-Verfahren einfach mit der Codierungsmatrix multipliziert werden können. Um die homomorphe Eigenschaft zu erhalten, wird eine andere Bildungsvorschrift verwendet, bei der die Hashwerte mit den Koeffizienten potenziert werden.

MACs Für die Absicherung durch MACs wurde ein Verfahren gewählt, dass wie DART auf Zeitasymmetrie basiert und in der Arbeit „Insecure ‘Provably Secure Network Coding’ and Homomorphic Authentication Schemes for Network Coding“ [Wan10] vorgestellt wurde. Um eine ausreichende Sicherheit zu gewährleisten, muss entweder in einem größeren Körper als bei PNC gearbeitet werden oder es müssen mehrere MACs pro Überprüfungsvorgang vorhanden sein. Obwohl eine Erhöhung der Anzahl der MACs aus Sicht eines erhöhten Nutzdatenanteils vorteilhafter ist, wird im Folgenden aus

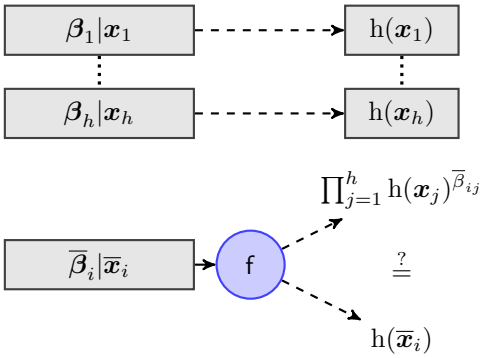


Abbildung 3.9: Homomorphe Hashes. Überprüfung der Validität eines empfangenen Pakets über vorher erhaltene und signierte Hashwerte.

Gründen der besseren Darstellbarkeit eine erhöhte Körpergröße angenommen. Deswegen werden alle Körperoperationen in $\mathbb{F}_{2^{128}}$ ausgeführt. Ein Element des Codierungsvektors hat folglich eine Größe von 16 Byte und ist damit deutlich größer als bei einer Variante mit mehreren MACs. Im Gegensatz zu den homomorphen Hashwerten und DART müssen die MACs zusätzlich im Paket untergebracht werden. Im MAC-Verfahren muss jeder Zwischenknoten auf dem Pfad zu den Empfängern einen MAC überprüfen. Daher entspricht die Anzahl der maximalen Hops ℓ der Anzahl der benötigten MACs im Paket.

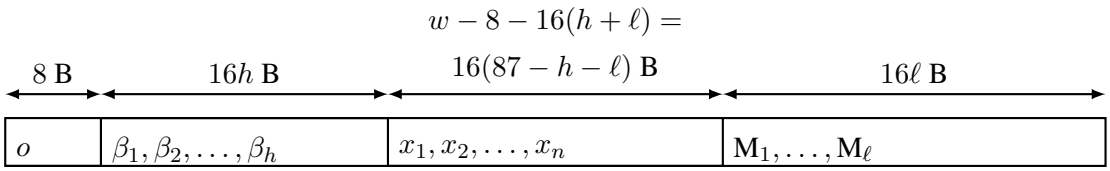


Abbildung 3.10: Struktur des Datenpakets beim MAC-Verfahren.

Abbildung 3.10 stellt die Struktur eines Datenpakets dar. Durch den größeren Körper werden pro Codierungskoeffizient 16 Byte benötigt. Zusätzlich dazu wird pro Hop ein MAC notwendig, welcher ebenfalls 16 Byte benötigt. Dadurch ergibt sich ein Nutzdatenanteil von $w - 8 - 16(h + \ell)$ Bytes.

Um die Pakete zu überprüfen, ist es notwendig, dass die symmetrischen Schlüssel für die MACs an die Zwischenknoten verteilt werden. Dies geschieht durch zusätzliche Pakete, die vom Sender s nach und nach an die Zwischenknoten verteilt werden. Ein Paket besteht dabei aus der Generations-ID o sowie einer Seed s_i, um den Schlüssel κ_i für die Überprüfung des MACs M_i zu erzeugen. Damit die Seedwerte nicht verfälscht werden können, wird eine Kette mithilfe einer Einwegfunktion erzeugt, sodass von s_j auf s_{j+1}, aber nicht auf s_{j-1} geschlossen werden kann. Die Veröffentlichung erfolgt rückwärts und die erste Seed $s_{\ell+1}$ dient als „ Anker der Kette“ für die Verifikation der anderen Seedwerte und muss signiert versendet werden.

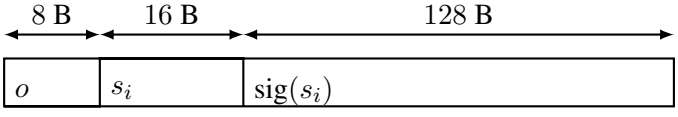


Abbildung 3.11: Struktur des Zusatzpakets beim MAC-Verfahren.

Die Paketstruktur für die $\ell + 1$ Zusatzpakete ist in Abbildung 3.11 dargestellt. Zu beachten ist, dass der letzte Teil, d. h. die asymmetrische Signatur, nur einmal für $s_{\ell+1}$ geschickt werden muss. Die weiteren ℓ Zusatzpakete sind dadurch mit 24 Byte sehr klein.
Die Erstellung der homomorphen MACs läuft entgegengesetzt zu der Veröffentlichung. Sender s erstellt sich eine Kette von $\ell + 1$ Seedwerten $\boldsymbol{s}$ und die daraus abgeleiteten Schlüssel $\boldsymbol{\kappa}$. Dann berechnet er die homomorphen MACs von M_1 bis M_ℓ, wobei mit M_j sowohl die Daten $\boldsymbol{x}_i$ als auch alle MACs von M_1 bis M_{j-1} abgesichert werden.
Nach dem Aussenden eines Pakets an den ersten Zwischenknoten wartet dieser Knoten auf die Veröffentlichung von dem passenden Zusatzpaket mit Seedwert. Dann wird geprüft, ob der Wert mit der Einwegfunktion zu der zuletzt erhaltenen Seed oder zum initialen Seedwert passt. Anschließend kann dann der letzte MAC im Paket überprüft und vom Paket entfernt werden. Danach kann das Recodieren von allen als gültig geprüften Paketen mit den homomorphen MACs stattfinden und das resultierende Paket an den nächsten Knoten gesendet werden. Die Empfänger können somit den letzten MAC überprüfen. Sollte ein Pfad nicht so lang wie die maximale Anzahl von Hops sein, kann der jeweilige Empfänger natürlich auch einen anderen MAC als M_1 zur Prüfung benutzen.

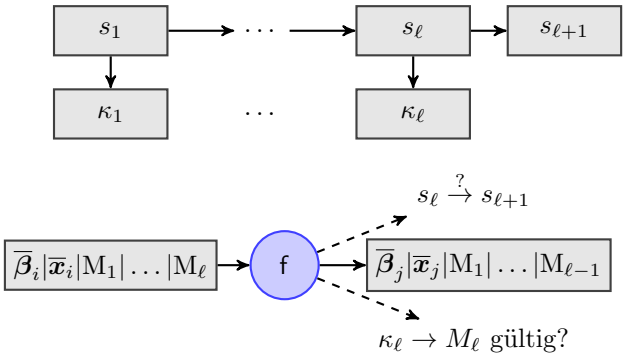


Abbildung 3.12: Homomorphe MACs, welche auf Zeitasymmetrie basieren. Überprüfung der Validität eines empfangenen Pakets über MACs. Die Schlüssel werden über eine Einwegfunktion in umgekehrter Reihenfolge veröffentlicht.

Abbildung 3.12 stellt diesen Vorgang noch einmal visuell dar. Die Berechnung neuer MACs bei der Kombination erfolgt genau wie bei den Codierungskoeffizienten und den codierten Daten, sodass beim Bilden der Linearkombination das komplette Paket gleich behandelt werden kann. Wichtig ist noch, dass die maximale Anzahl von Hops ℓ nicht überschritten werden darf, da andernfalls eine Integritätskontrolle ab da nicht mehr möglich ist, da weder ein weiterer MAC noch eine Seed für einen weiteren Schlüssel vorhanden sind.

Signaturen Als Beispiel für die homomorphen Signaturen wurde ein Verfahren gewählt, welches auf dem ursprünglichen RSA-Verfahren basiert. In der Arbeit „Secure Network Coding over the Integers“ [GKKR09] wurde dieses erstmalig vorgestellt. RSA in der „Textbook“-Implementierung besitzt eine homomorphe Eigenschaft, die es ermöglicht, aus gültigen Signaturen eine weitere gültige Signatur zu erstellen. Dabei funktioniert RSA nicht in einem endlichen Körper, sondern in einem endlichen Ring $\mathbb{Z}_N$, wobei die zusammengesetzte Zahl $N = p*q$ (mit p, q prim) selbst für eine kurzzeitige Absicherung mindestens 1 024 Bit groß sein muss. Die aktuellen Empfehlungen zu Schlüsselgrößen von RSA-Modulen gehen zwar von etwa 2 000 Bit aus [Bun17], dabei geht es allerdings um eine längerfristige Speicherung,

die hier nicht notwendig ist. Deswegen sind $1\,024$ Bit für eine kurzfristige Absicherung ausreichend. Daraus ergibt sich eine Signaturgröße von 128 Byte.
Damit die Koeffizienten des Codierungsvektors nicht auch eine Größe von 128 Byte benötigen, ist die Idee, über den Ganzzahlen $\mathbb{Z}$ zu rechnen und vorher abzuschätzen, um wie viel sich die Koeffizienten auf dem Weg zum Empfänger erhöhen. Es wird davon ausgegangen, dass die zufälligen Koeffizienten maximal 8 Bit groß sind, also wie bei PNC zwischen 0 und 255 zufällig gewählt werden. Pro Hop werden maximal h Nachrichten miteinander kombiniert. Daher ist der Platz für die Koeffzienten und damit auch für die Nutzdaten von h und ℓ abhängig.

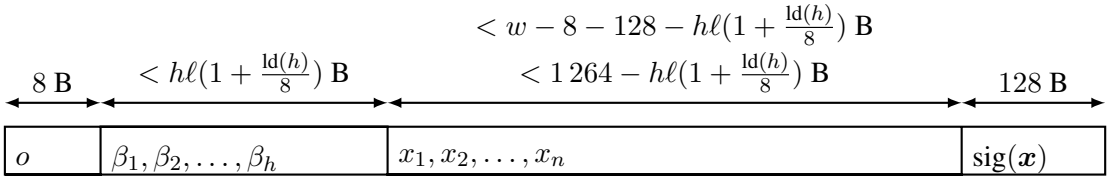


Abbildung 3.13: Struktur des Datenpakets beim Signatur-Verfahren.

In Abbildung 3.13 ist die Struktur des Datenpakets beim Signatur-Verfahren dargestellt. Aufgrund des Wachstums bei den Codierungskoeffizienten muss der Nutzdatenanteil schon zu Beginn so reduziert werden, dass genügend Platz für die Koeffizienten nach ℓ Hops verbleibt.
Je nach Größe von h und ℓ ergibt sich daraus schnell ein sehr großer Übertragungsaufwand für wenige Nutzdaten. Dafür müssen allerdings im Gegensatz zu den anderen Verfahren keine zusätzlichen Pakete übermittelt werden. Die Daten $\boldsymbol{x}$ sind im Übrigen Elemente des Rings $\mathbb{Z}_N$ und werden deswegen nicht aufgebläht.

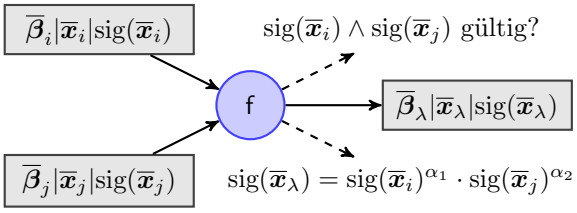


Abbildung 3.14: Signaturen basierend auf RSA. Überprüfung der Validität eines empfangenen Pakets über die Signatur. Zwischenknoten können aus mehreren Signaturen gültige Signaturen für recodierte Pakete erzeugen.

In Abbildung 3.14 ist das Prüfen der Signatur als auch das Erstellen einer gültigen Signatur für ein (re)codiertes Paket zu sehen. Im Gegensatz zu den anderen Verfahren muss bei der Linearkombination das Paket getrennt bearbeitet werden. Codierungskoeffizienten und Daten werden wie gewohnt mit einem lokalen Codierunsgvektor multipliziert und anschließend summiert. Die Signaturen werden mit den einzelnen Elementen des lokalen Codierunsgvektors potenziert und anschließend multipliziert.
Damit wurden nun alle 4 untersuchten Verfahren vorgestellt und die Verfahrensweise grob erläutert. Wie bereits an der Paketstruktur erkennbar ist, sind die 2 Parameter, die im generalisierten Netzwerkmodell verwendet werden, die größten Einflussfaktoren für die Effizienz. Dabei ist der Einfluss der beiden Parameter je nach Verfahren recht unterschiedlich. Daher ist schon jetzt davon auszugehen, dass es nicht ein effizientestes Verfahren gibt, sondern es auf die jeweiligen Netzwerke, die Anforderungen der Anwendung sowie auf die Gegebenheiten des Systems ankommt. Nachdem nun das Systemmodell, die

Methodik und die Verfahren erläutert wurden, werden im nächsten Abschnitt die Ergebnisse der Untersuchungen präsentiert.

3.5 Auswertungen und Ergebnisse

3.5.1 Effizienzbetrachtung der parametrisierten Modelle

Für die Auswertung der parametrisierten Modelle wurden jeweils für die 3 Modelle und die 3 Effizienzparameter Formeln entwickelt und für die Parameterbereiche von $2 \leq d \leq 20$ und $2 \leq \ell \leq 20$ überprüft. Diese Formeln wurden anhand der Definitionen der Effizienzparameter für alle ausgesuchten Verfahren und die Übertragung genau einer Generation entwickelt.
Im Folgenden werden die Ergebnisse für die unterschiedlichen Verfahren und parametrisierten Modelle diskutiert. Um eine bessere Vergleichbarkeit zu erhalten, werden diese mit unsicherer Netzwerkcodierung (PNC) sowie mit einfachem Routing verglichen. Unter diesem Routing ist das Senden einzelner Nachrichten an einen Empfänger zu verstehen. Zwar ist beim Routing wie auch bei PNC keine Integritätsabsicherung vorhanden, jedoch kann es beim Routing zu keinerlei Verschmutzungsattacken kommen.

Netzwerkmodell 1 Das erste Modell (Abbildung 3.1) wurde gewählt, um den Einfluss der Breite d eines Netzes und damit der Anzahl der Empfänger $|\mathbf{R}|$ bei gleichbleibender Hopanzahl ℓ und konstanter Generationsgröße h zu untersuchen.

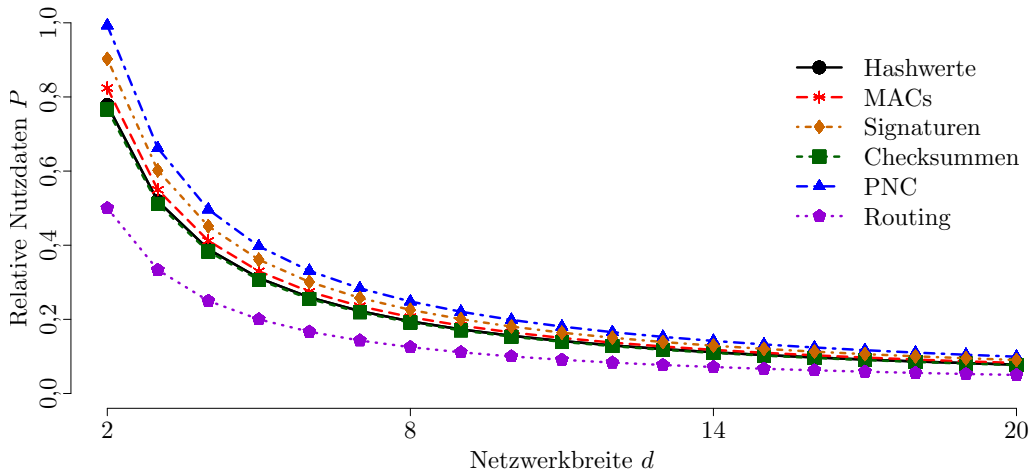


Abbildung 3.15: Relative Nutzdaten P in Abhängigkeit der Netzwerkbreite d in Modell 1.

In Abbildung 3.15 sind die Ergebnisse für die relativen Nutzdaten P dargestellt. Durch die Netzwerkcodierung liegen die Werte aller Verfahren höher als bei herkömmlichem Routing. Ungesicherte Netzwerkcodierung (PNC) zeigt wie erwartet die beste Leistung und stellt die maximal erreichbare Leistung für die abgesicherten Verfahren dar. Bezüglich der relativen Nutzdaten ist eine große Abhängigkeit von der Netzwerkbreite d zu sehen. Absolut gesehen vermindert sich der Anteil der Nutzdaten mit größerem d. Allerdings ist das Verhältnis der Werte der einzelnen Verfahren relativ zu PNC sehr konstant. Dadurch bleibt auch die Reihenfolge, dass Signaturen von den sicheren Verfahren am besten abschneiden

und Hashwerte und Checksummen am schlechtesten, für verschiedene Werte von d erhalten. Dies liegt daran, dass d bei allen Verfahren mit gleichem Faktor eingeht.

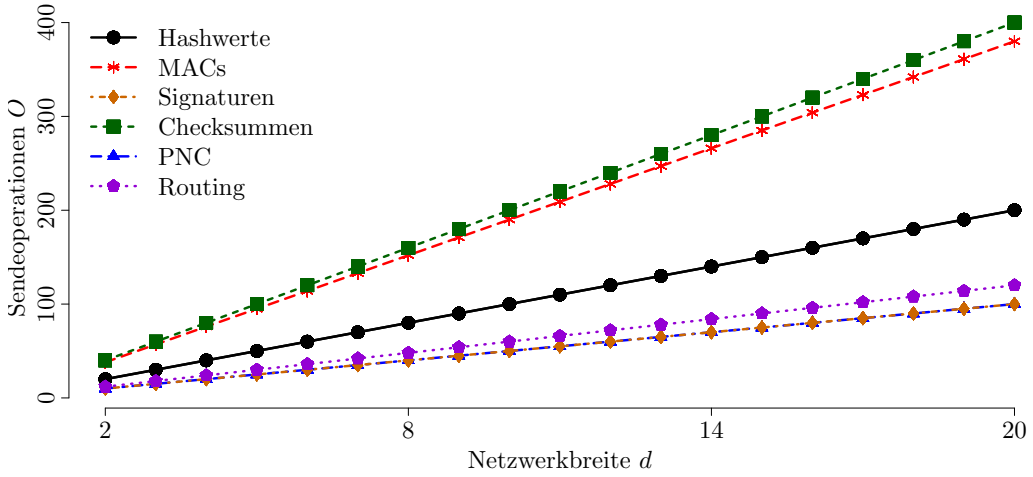


Abbildung 3.16: Anzahl der notwendigen Sendeoperationen O in Abhängigkeit der Netzwerkbreite d in Modell 1.

Abbildung 3.16 stellt die Abhängigkeit der Sendeoperationen O von der Netzwerkbreite d dar. Allgemein gesehen hängt O bei allen Varianten linear von h ab. Jedoch ist der Anstieg verschieden. Bei den MACs und den Checksummen erhöhen sich die notwendigen Sendeoperationen durch die Zusatzpakete am stärksten, weshalb die beiden Verfahren die schlechteste Leistung bezüglich des Effizienzparameters O zeigen. Auch die Hashwerte benötigen mehr Sendeoperationen als herkömmliches Routing, da mit steigender Breite d mehr Hashpakete am Anfang verteilt werden müssen. Jedoch ist der Anstieg nicht so stark wie bei den MACs und den Checksummen. Routing benötigt leicht mehr Sendeoperationen als PNC, während das Signaturverfahren mit PNC gleich aufliegt.

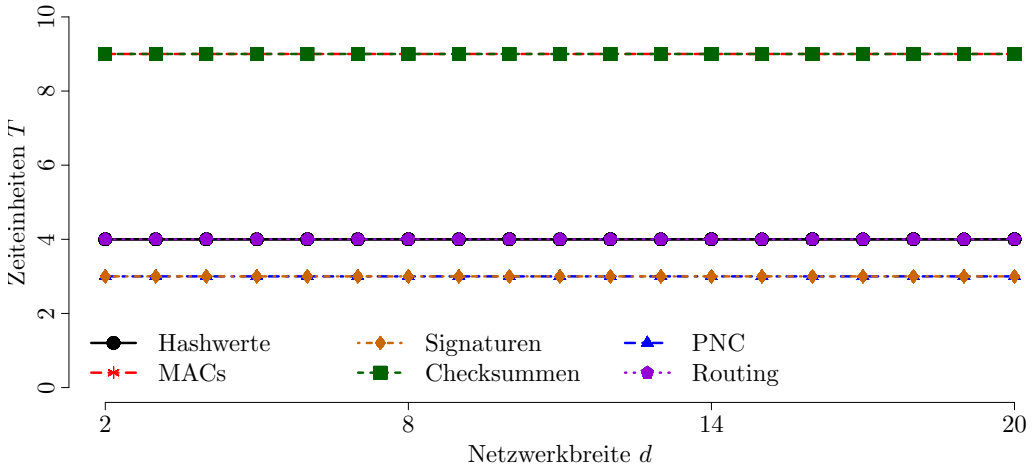


Abbildung 3.17: Anzahl der Zeiteinheiten T in Abhängigkeit der Netzwerkbreite d in Modell 1.

Abbildung 3.17 zeigt, dass die Netzwerkbreite d keinen Einfluss auf die benötigten Zeiteinheiten T hat. Diese ist in Netzwerkmodell 1 für alle Verfahren konstant. Durch das TESLA-Protokoll und das Senden von Zusatzinformationen zwischen den Paketübertragungen benötigen die Verfahren, welche auf Zeitasymmetrie basieren (MACs und Checksummen), die meisten Zeiteinheiten. Routing und die Hashwerte benötigen eine Zeiteinheit länger als das Minimum von 3 Zeiteinheiten, welches PNC und das Signaturverfahren benötigen.
Insgesamt schneidet bei diesem Modell das Signaturverfahren am besten ab, während die Verfahren, die auf MACs oder Checksummen basieren, die schlechtesten Werte zeigten.

Netzwerkmodell 2 Mit dem zweiten Modell (Abbildung 3.2) soll der Einfluss der Tiefe und damit der Hops ℓ eines Netzwerks auf die Effizienzparameter untersucht werden. Die Netzwerkbreite d sowie die Generationsgröße h sind in dem Modell konstant ($d = h = 2$).

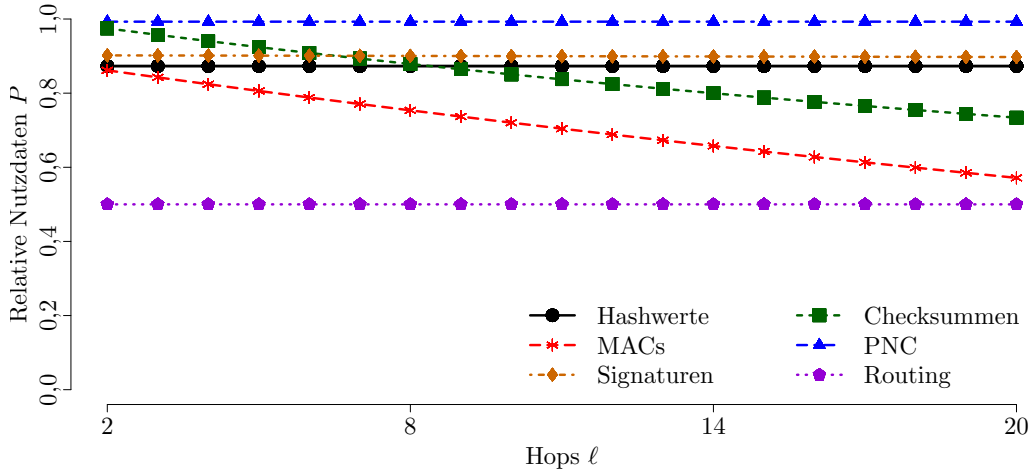


Abbildung 3.18: Relative Nutzdaten P in Abhängigkeit der Anzahl der Hops ℓ in Modell 2.

In Abbildung 3.18 wird die Abhängigkeit der relativen Nutzlast P von der Hopanzahl ℓ dargestellt. Wie bei Netzwerkmodell 1 sind die relativen Nutzdaten durch die Netzwerkcodierung bei PNC etwa doppelt so groß wie bei herkömmlichem Routing. Jedoch ist P sowohl für diese beiden Verfahren, als auch für die Verfahren, welche Hashwerte oder Signaturen verwenden, konstant und nicht von ℓ abhängig. Bei den Verfahren, die auf MACs oder Checksummen basieren, sinkt hingegen der Anteil der Nutzdaten mit steigender Anzahl ℓ. Die Ursache liegt wieder an den Zusatzpaketen, welche gesendet werden müssen. Bei dem MAC-Verfahren kommt noch negativ hinzu, dass die Nutzdaten pro Paket geringer werden, da pro Hop ein weiterer MAC im Paket benötigt wird. Somit hat dieses Verfahren von allen sicheren Verfahren den niedrigsten Anteil an Nutzdaten. Für die Checksummen gilt, dass diese mit einem sehr hohen Nutzdatenanteil für eine kleine Hopanzahl die beste Leistung erzielen. Für größere ℓ überträgt das Signaturverfahren den höheren Anteil an Nutzdaten. Das Verfahren mit den homomorphen Hashwerten hat einen ähnlichen, wenn auch nicht ganz so hohen Nutzdatenanteil wie das Signaturverfahren.
In Abbildung 3.19 werden die Sendeoperationen O in Abhängigkeit der Netzwerktiefe ℓ dargestellt. Während O bei den Verfahren mit MACs und Checksummen mit größer werdendem ℓ quadratisch steigt, weisen die anderen Verfahren nur lineares Wachstum auf. Dadurch ist die Anzahl der Sendeoperationen

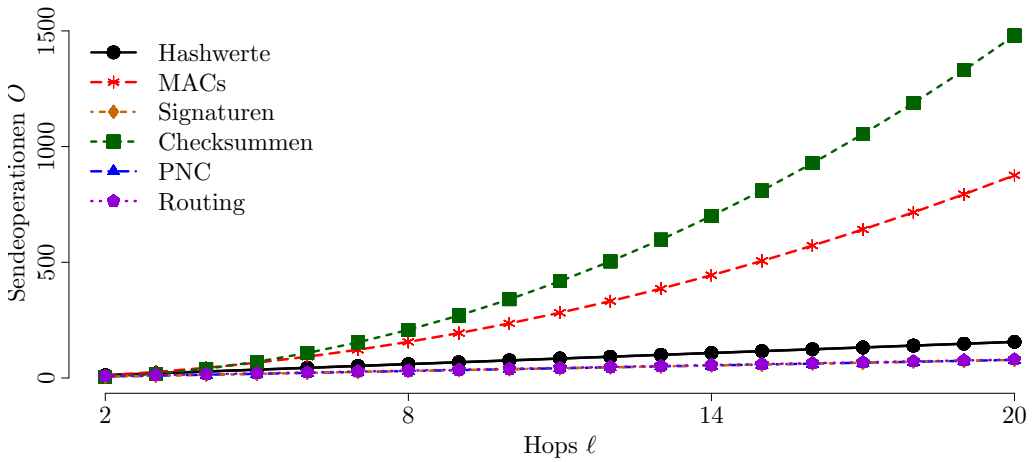


Abbildung 3.19: Anzahl der notwendigen Sendeoperationen O in Abhängigkeit der Anzahl der Hops ℓ in Modell 2.

bei MACs und Checksummen am größten. Auch die Hashwerte benötigen mehr Sendeoperationen als das herkömmliche Routing. Routing liegt nur ganz leicht über der minimalen Anzahl an Sendevorgängen, die sowohl PNC als auch das Signaturverfahren erreicht. Auch hier kommt letzterem zu Gute, dass keine zusätzlichen Pakete wie bei den anderen Verfahren benötigt werden.

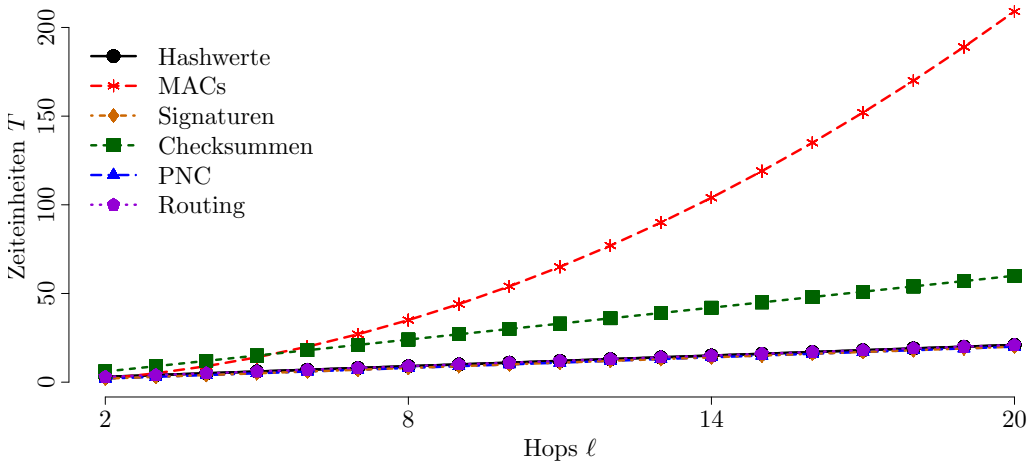


Abbildung 3.20: Anzahl der Zeiteinheiten T in Abhängigkeit der Anzahl der Hops ℓ in Modell 2.

Abbildung 3.20 stellt noch den Zusammenhang zwischen den benötigten Zeiteinheiten T und der Anzahl der Hops ℓ dar. Das Diagramm zeigt einen sehr ähnlichen Verlauf wie Abbildung 3.19. In diesem Fall ist allerdings das Verfahren, welches MACs verwendet, zumindest ab einer Hopanzahl von $\ell > 4$ deutlich langsamer als das auf Checksummen basierende Verfahren. Dieses benötigt zwar auch mehr Zeiteinheiten als alle anderen Verfahren bis auf das MAC-Verfahren, jedoch ist der Anstieg linear mit ℓ. Der Grund liegt darin, dass die Zusatzpakete bei dem Checksummen-Verfahren im Prinzip dauerhaft gesendet werden können, während beim MAC-Verfahren eine Taktung notwendig ist, welche sicherstellt, dass eine

neue Seed erst dann veröffentlicht wird, wenn alle Zwischenknoten die vorherige Seed erhalten haben, die Nachrichten überprüft haben und kombinierte Nachrichten weiterschicken konnten.
Insgesamt zeigt sich in Netzwerkmodell 2 eine stärkere Abhängigkeit von ℓ für unterschiedliche Verfahren. Während für Netzwerke mit einer kleinen maximalen Hopanzahl ℓ Verfahren, die auf Checksummen basieren, sehr effizient und sinnvoll erscheinen, gilt für tiefere Netzwerke, dass ein solches Verfahren deutlich ineffizienter bezüglich der 3 Effizienzparameter als beispielsweise ein Signaturverfahren ist.

Netzwerkmodell 3 Das Netzwerkmodell 3 (Abbildung 3.3) hat ähnlich wie Modell 1 eine variable Netzwerkbreite d. Jedoch ändert sich hier neben der Anzahl der Empfänger $|\mathbf{R}|$ auch die Generationsgröße h ($d = h = |\mathbf{R}|$). Dafür ist die Anzahl der Hops $\ell = 4$ konstant. Insgesamt sollte dieses Netzwerk mit Flaschenhals einen deutlichen Vorteil der Netzwerkcodierung gegenüber herkömmlichem Routing aufzeigen.

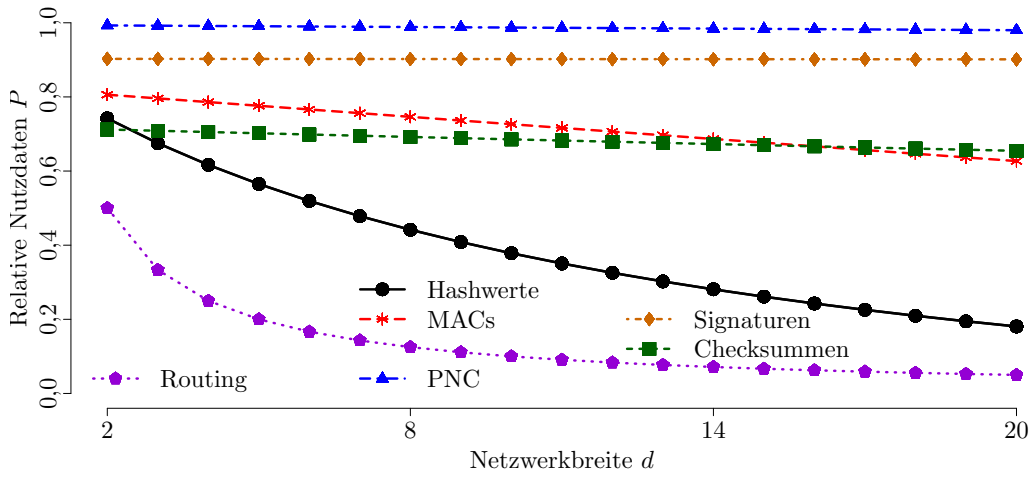


Abbildung 3.21: Relative Nutzdaten P in Abhängigkeit der Netzwerkbreite d und der Generationsgröße h in Modell 3. Dabei gilt $h = d = |\mathbf{R}|$.

Abbildung 3.21 zeigt die Abhängigkeit der relativen Nutzlast P von der Netzwerkbreite d sowie der Generationsgröße h für alle untersuchten Verfahren. Wie erwartet zeigt die Netzwerkcodierung deutliche Vorteile gegenüber herkömmlichen Routing, insbesondere bei steigender Netzwerkbreite d und damit einhergehend steigender Generationsgröße h. Die sicheren Verfahren zeigen sich differenzierter als in den anderen beiden Modellen. Zwar ist auch hier die relative Nutzlast P bei den Signaturen am höchsten, jedoch ändert sich die Reihenfolge der anderen Verfahren in Abhängigkeit von d. Während das Verfahren mit Checksummen zuerst (bei $d = 2$) die niedrigsten relativen Nutzdaten besitzt, ist ab $d \geq 4$ das Verfahren mit Hashwerten am schlechtesten. Ab etwa $d \geq 16$ ist das auf Checksummen basierende Verfahren besser als das bis dahin effizientere MAC-Verfahren.
In Abbildung 3.22 wird der Einfluss der Netzwerkbreite d und der Generationsgröße h auf die Anzahl der Sendeoperationen O dargestellt. Ähnlich wie bei den anderen beiden Modellen benötigen die Verfahren mit MACs und Checksummen aufgrund der zusätzlich notwendigen Pakete die meisten Sendeoperationen O. Aufgrund der großen Generationsgröße müssen aber auch bei den homomorphen Hashwerten relativ viele Zusatzpakete geschickt werden, sodass auch hier relativ viele Sendeoperationen notwendig

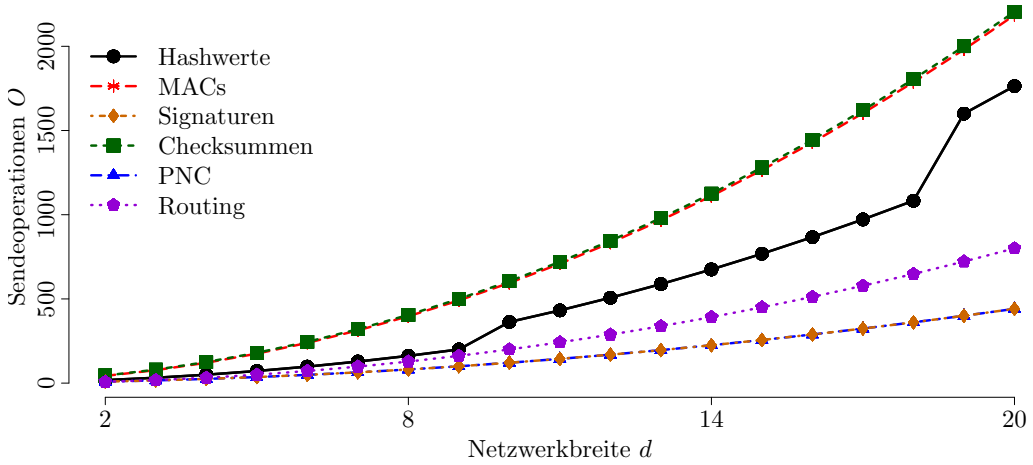


Abbildung 3.22: Anzahl der notwendigen Sendeoperationen O in Abhängigkeit der Netzwerkbreite d und der Generationsgröße h in Modell 3. Dabei gilt $h = d = |\mathbf{R}|$.

sind. Im Diagramm sieht man Sprünge bei diesem Verfahren bei etwa $d = 9$ und $d = 18$, da bei den gewählten Parametern jeweils 9 Hashwerte in ein Hashpaket passen und beim Übersteigen dieser Werte dann ein weiteres Hashpaket geschickt werden muss. Durch die für die Netzwerkcodierung günstige Netzwerktopologie gibt es einen klaren Vorteil von PNC gegenüber Routing, wobei auch das Signaturverfahren auf einer Linie mit PNC liegt und somit die geringste Anzahl an Sendeoperationen O von den sicheren Verfahren benötigt.

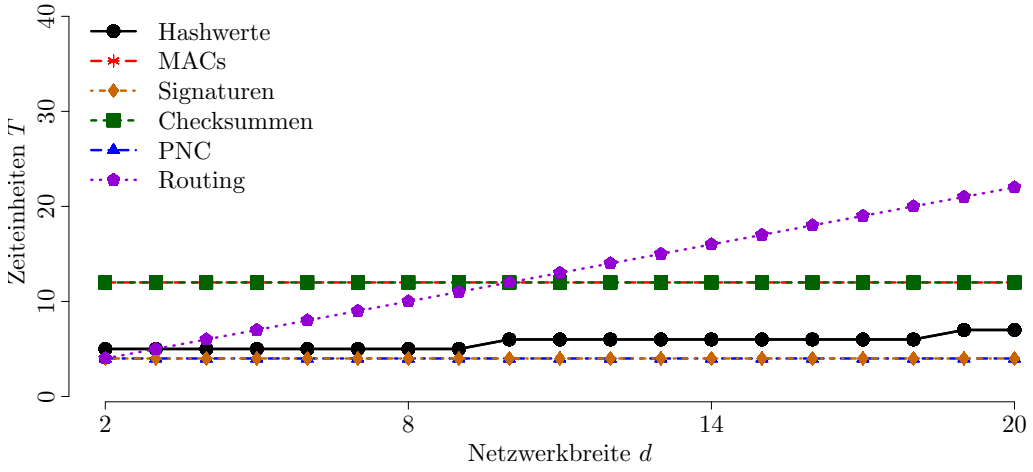


Abbildung 3.23: Anzahl der Zeiteinheiten T in Abhängigkeit der Netzwerkbreite d und der Generationsgröße h in Modell 3. Dabei gilt $h = d = |\mathbf{R}|$.

Analog dazu stellt Abbildung 3.23 die Auswirkungen von Netzwerkbreite d und Generationsgröße h auf die Anzahl der Zeiteinheiten T dar. Auch hier sieht man die Vorteile der Verfahren, die auf Netzwerkcodierung beruhen, gegenüber Routing. Beim Routing steigt der Zeitaufwand linear zur Netzwerkbreite d und damit zur Generationsgröße h. Bei den anderen Verfahren ist die Anzahl der Zeiteinheiten konstant.

Eine Einschränkung von dieser Aussage gibt es nur bei den homomorphen Hashwerten. Dort erhöht sich mit einer um 9 größeren Generationsgröße, ähnlich zu den Sendoperationen, der Zeitbedarf um eine Einheit, da ein weiteres Hashpaket verteilt werden muss. Ansonsten liegen PNC und Signaturen wieder gleichauf und benötigen die wenigsten Zeiteinheiten. MACs und Checksummen haben den höchsten Zeitbedarf von den sicheren Verfahren.

Zusammenfassend zeigt sich, dass sich das Signaturverfahren im Großen und Ganzen bei allen Modellen und Effizienzparametern überdurchschnittlich verhält. Weiterhin fällt auf, dass die Verfahren mit MACs und Checksummen mit größerer Netzwerktiefe und damit der Hopanzahl ℓ schlechter werden und folglich eher für kleinere Netze geeignet sind.

Insgesamt handelt es sich bei den Untersuchungen jedoch nur um erste Vergleiche. Für den praktischen Einsatz sind aber noch einige Details zu beachten. Zum einen wurde der Bereich von ℓ und d bzw. h beschränkt. Die Ergebnisse sollten daher auch auf größere Netzwerke und größere Generationen ausgeweitet werden. Zum anderen gelten die Angaben nur für 3 konstruierte Modelle. Um allgemein gültige Aussagen treffen zu können, sollte ein generalisierte Netzwerkmodell zum Einsatz kommen.

Somit lassen sich nach dieser Untersuchung zwei Aussagen festhalten. Erstens: Es gibt nicht ein bestes Verfahren, welche für alle Parameter und Netzwerke die höchste Effizienz zeigt. Zweitens: Die Effizienz ist bei allen Verfahren stark von den zugrunde liegenden Parametern und der Netzwerktopologie abhängig. Deswegen soll im nächsten Teil stärker auf diese Netzwerkabhängigkeit eingegangen werden und je nach Parametern das beste Verfahren aufgezeigt werden. Dabei liegt der Fokus nach wie vor auf der Übertragung von Daten.

3.5.2 Effizienzbetrachtung des generalisierten Modells

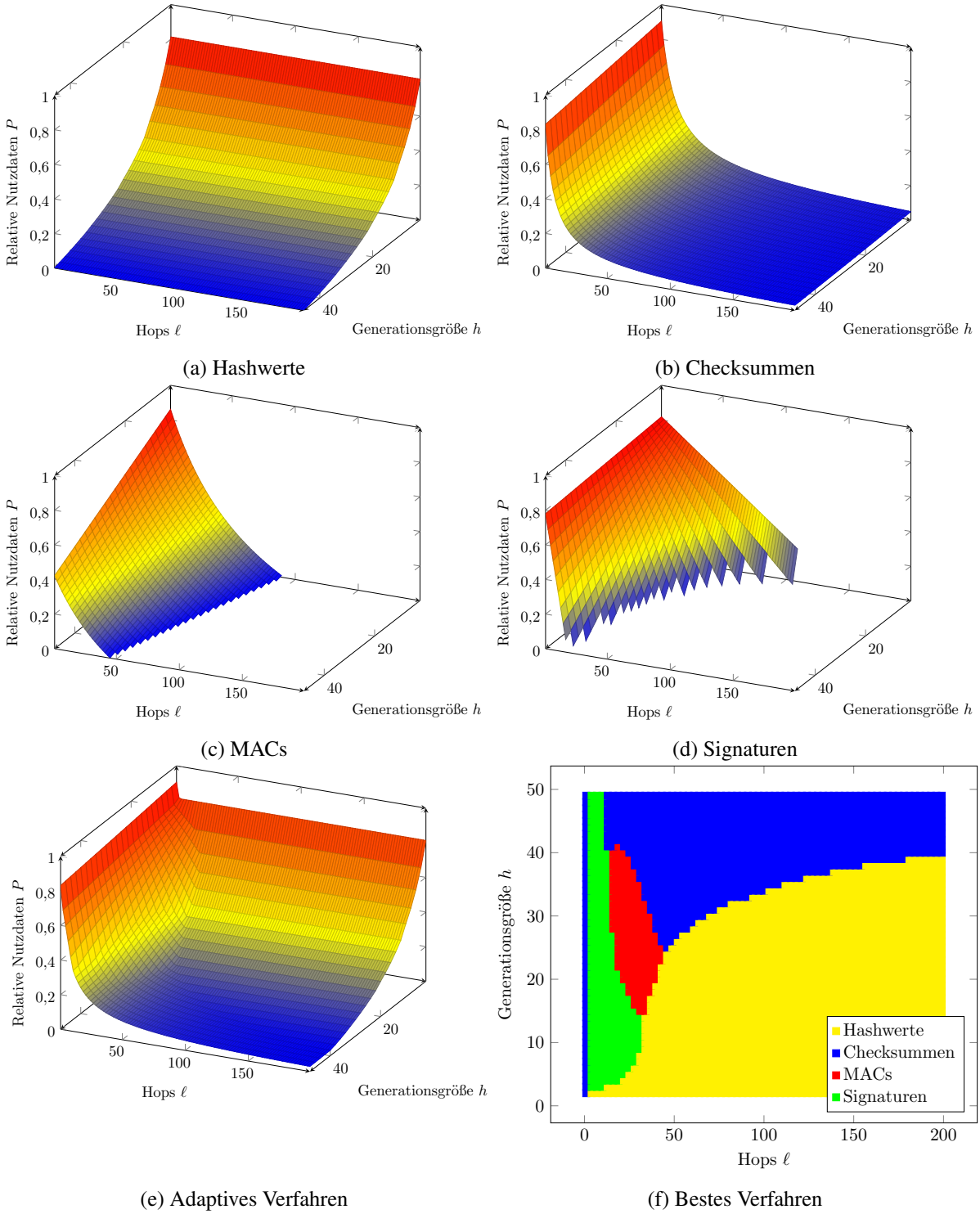
Um die Netzwerkabhängikeit der Verfahren zu untersuchen, wurden für die Effizienzparameter für die 4 Vertreter der sicheren Verfahren in Abhängigkeit von der Generationsgröße h und der maximalen Anzahl an Hops ℓ Formeln erstellt, um die Effizienzparameter berechnen zu können.

Für die relativen Nutzdaten wurde die genaue Größe der Nutzdaten pro Paket und der Zusatzdaten beachtet. Dadurch ergibt sich z. B. für das Verfahren, welches auf homomorphen Hashwerten basiert, die folgende Formel:

$$P_{\text{Hashwerte}} = \frac{h\,(w - 32h - 8)}{h\left(w + 128h + \left\lceil \frac{128h}{w-136} \right\rceil * 136\right)}. \tag{3.1}$$

Da eine Generation aus h Datenpaketen besteht, werden als Nutzdaten h Pakete der Größe w versendet. Davon werden noch 8 Byte für die Generations-ID o sowie 32 Byte pro Codierungskoeffizient verwendet. Im Nenner sind die h vollen Datenpakete sowie die Hashpakete, die an h nachfolgende Knoten (Abschätzung, dass die Anzahl der abgehenden Kanten etwa der Generationsgröße entspricht) gesendet werden müssen. Dabei gilt, dass ein Hash 128 Byte groß ist und alle $\left\lceil \frac{128h}{w-136} \right\rceil$ Hashpakete eine zusätzliche Generations-ID (8 Byte) sowie eine Signatur (128 Byte) benötigen. Analog zu Formel (3.1) können die Formeln für die anderen Verfahren ermittelt werden:

$$P_{\text{Checksummen}} = \frac{h\,(w - h - 8)}{h\,(w + \ell\,(142 + 2h))}, \tag{3.2}$$



(a) Hashwerte

(b) Checksummen

(c) MACs

(d) Signaturen

(e) Adaptives Verfahren

(f) Bestes Verfahren

Abbildung 3.24: Relative Nutzdaten P in Abhängigkeit der Generationsgröße h und der Anzahl an Hops ℓ im generalisierten Modell.

$$P_{\text{MACs}} = \frac{h\,(w - 16\,(h + \ell) - 8)}{h\,(w + 24\ell + 152)}, \tag{3.3}$$

$$P_{\text{Signaturen}} = \frac{\left(w - 264 - \ell\,(h+5)\left(1 + \frac{\lceil \text{ld}(h) \rceil}{8}\right)\right)}{w}. \tag{3.4}$$

In Abbildung 3.24 sind die Ergebnisse für die relativen Nutzdaten P in Diagrammen dargestellt. Dazu wurden in die Formeln (3.1) - (3.4) verschiedene Parameterpaare von h und ℓ eingesetzt und die relativen Nutzdaten P berechnet. Die ersten 4 Diagramme (a) - (d) entsprechen den Werten für die jeweiligen Verfahren. Diagramm (e) stellt die relativen Nutzdaten dar, wenn ein adaptives Verfahren immer das bestmögliche Verfahren auswählen würde. Diagramm (f) zeigt für alle getesteten Parameterkombinationen von h und ℓ, welches der untersuchten Verfahren jeweils auszuwählen wäre.

Die Diagramme verdeutlichen die unterschiedlichen Eigenschaften der Verfahren. Während bei den homomorphen Hashwerten die Anzahl der Hops ℓ keinerlei Einfluss, dafür aber die Generationsgröße h einen großen Einfluss auf den Anteil der Nutzdaten hat, ist es bei den Checksummen fast umgekehrt. Die Generationsgröße hat nur einen minimalen Einfluss, während die Anzahl der Hops für den Effizienzparameter stark entscheidend ist.

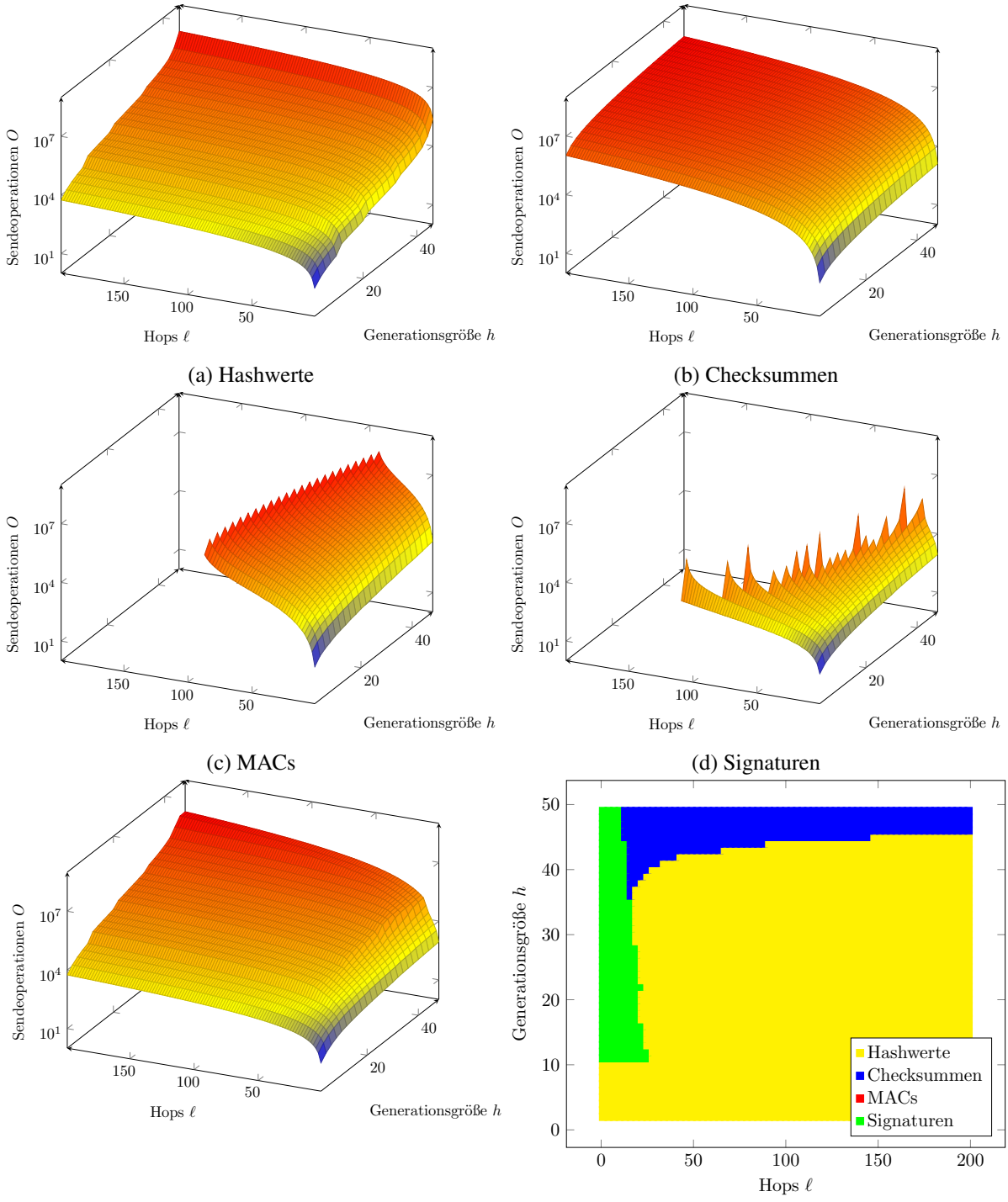
Die Verfahren, welche auf MACs sowie Signaturen basieren, zeigen eine Abhängigkeit von sowohl Hops ℓ als auch der Generationsgröße h. Weiterhin fällt hier auf, dass für große Werte von ℓ und h keine Ergebnisse existieren. Das liegt daran, dass diese Verfahren für bestimmte Kombinationen nicht anwendbar sind. Zum einen bedeutet eine hohe Hopanzahl ℓ für das MAC-Verfahren, dass ℓ MACs im Datenpaket eingebracht werden müssen. Da aber die Paketgröße begrenzt ist, wird die Anzahl und der Speicherbedarf für die MACs zu hoch. Ein Paket von $w = 1\,400$ Byte kann neben der Generations-ID o und einer Generationsgröße $h = 2$ beispielsweise nur 85 MACs umfassen. Ähnliches gilt für das Signaturverfahren. Mit wachsender Generationsgröße und Hopanzahl wächst die Größe der Koeffizienten des Codierungsvektors, sodass die Paketgröße ab einem bestimmten Punkt nicht mehr ausreicht.

Insgesamt am interessantesten ist Abbildung 3.24 (f). Ausgangspunkt für die Untersuchung war die Erkenntnis, dass es nicht ein bestes Verfahren gibt und es auf das Netzwerk ankommt. In dieser Grafik sieht man gut, dass jedes der 4 Verfahren bei einer gewissen Kombination von h und ℓ am besten ist. Für eine niedrige Hopanzahl ℓ bietet sich das Signaturverfahren an. Ist $\ell > 50$, lohnt sich der Einsatz von Hashwerten, sofern h nicht allzu groß ist. Ist sowohl ℓ als auch h groß, sollte auf das Verfahren mit den Checksummen zurückgegriffen werden. Bei ungefähr $15 \leq h \leq 40$ und $20 \leq \ell \leq 40$ bietet das MAC-Verfahren das beste Nutzdatenverhältnis.

Für die Auswertung der Sendeoperationen wird wieder als Beispiel die Formel für das Verfahren mit den Hashwerten vorgestellt und erläutert, bevor auf die eigentlichen Ergebnisse eingegangen wird.

Zusätzlich zu h und ℓ sind die Sendeoperationen für die Hashwerte noch von der Anzahl aller Kanten $|\mathbf{E}|$ im Graph abhängig. Da $|\mathbf{E}|$ allerdings als Faktor in allen Formeln vorkommt, kann dieser einfach auf eine beliebige positive Zahl gesetzt werden. Das Verhältnis zwischen den Verfahren bleibt davon unberührt.

Außerdem wird anstatt des Übertragens einer Generation eine bestimmte Nutzdatenmenge $v = 20\,000$ Byte gewählt. Das hat den Vorteil, dass man die Verfahren untereinander besser vergleichen kann, da ansonsten Verfahren ohne zusätzliche Pakete bevorzugt würden. Die Festlegung auf $v = 20\,000$ Byte hat dabei keinerlei Einfluss auf die Relation zwischen den Verfahren, sondern nur auf die absoluten Zahlen.



(a) Hashwerte

(b) Checksummen

(c) MACs

(d) Signaturen

(e) Adaptives Verfahren

(f) Bestes Verfahren

Abbildung 3.25: Anzahl der notwendigen Sendeoperationen O in Abhängigkeit der Generationsgröße h und der Anzahl an Hops ℓ im generalisierten Modell.

Daher ergibt sich für das Verfahren, welches auf homomorphen Haswerten basiert, folgende Formel für die Anzahl der notwendigen Sendeoperationen:

$$O_{\text{Hashwerte}} = \left\lceil \frac{v}{(w - 32h - 8)} \right\rceil * \left(1 + \left\lceil \frac{128h}{w - 136} \right\rceil\right) * |\mathbf{E}| \,. \tag{3.5}$$

Der erste Faktor der Formel (3.5) gibt an, wie viele Pakete für die Datenmenge übertragen werden müssen. Dazu kommt dann im zweiten Faktor noch die Anzahl an Hashpaketen, die vorher übertragen werden muss. Da jede Nachricht über eine Kante geht, wird das Ganze mit $|\mathbf{E}|$ multipliziert. Analog kann für die anderen Verfahren die Formel für die Sendeoperationen ermittelt werden:

$$O_{\text{Checksummen}} = \left\lceil \frac{v}{(w - h - 8)} \right\rceil * (\ell + 1) * |\mathbf{E}| \,, \tag{3.6}$$

$$O_{\text{MACs}} = \left\lceil \frac{v}{(w - 16(h + \ell) - 8)} \right\rceil * (\ell + 2) * |\mathbf{E}| \,, \tag{3.7}$$

$$O_{\text{Signaturen}} = \left\lceil \frac{v}{\left(w - 264 - \ell(h + 5)\left(1 + \frac{\lceil \text{ld}(h) \rceil}{8}\right)\right)} \right\rceil * |\mathbf{E}| \,. \tag{3.8}$$

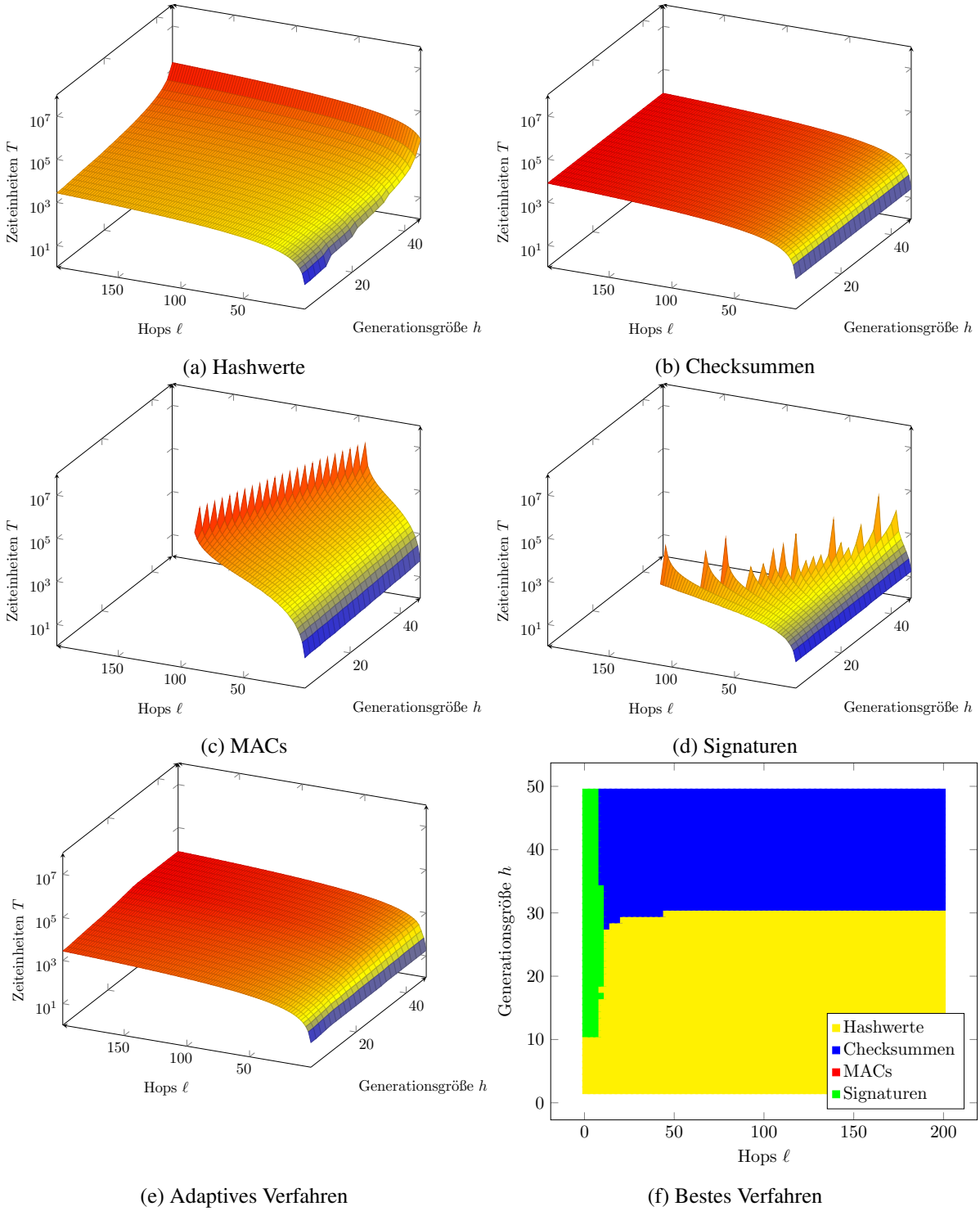
Unter Anwendung der Formeln (3.5) - (3.8) wurde Abbildung 3.25 erstellt. Diese ist analog zu der vorherigen Abbildung aufgebaut und stellt die benötigten Sendeoperationen dar. Abbildungen 3.25 (a) - (d) zeigen die Werte für die einzelnen Verfahren während Abbildungen 3.25 (e) und (f) die besten Verfahren veranschaulichen. Zu beachten sind die geänderten Achsenrichtungen im Vergleich zu Abbildung 3.24, um eine bessere Ablesbarkeit zu gewährleisten.

Auch hier gilt, dass das Verfahren mit Hashwerten eher von der Generationsgröße, das Verfahren mit den Checksummen eher von der Anzahl der Hops ℓ abhängt. Signatur- und MAC-Verfahren sind von beiden Variablen abhängig, sind aber wieder nur mit eingeschränkten Wertepaaren einsetzbar.

Alles in allem zeigt sich ein ähnliches Bild wie bei den relativen Nutzdaten. Zwar ist das Verfahren, welches auf MACs basiert, in keiner Kombination von h und ℓ das beste Verfahren, jedoch gilt wieder, dass für Netzwerke mit niedriger Hopanzahl das Signaturverfahren die besten Resultate erzeugt, zumindest wenn $h > 10$. Bei großer Hopanzahl $\ell > 25$ und großer Generationsgröße $h > 40$ bietet sich das Checksummenverfahren an. Für die anderen Fälle benötigt das Verfahren, welches auf homomorphen Hashwerten basiert, am wenigsten Sendeoperationen, um eine bestimmte Menge v an Daten zu übertragen.

Als letztes soll noch auf die benötigten Zeiteinheiten eingegangen werden. Wie bisher wird davon ausgegangen, dass in einer Zeiteinheit Pakete empfangen, verarbeitet und anschließend verschickt werden können. Daher könnte sich eine leichte Verschiebung der Ergebnisse hin zu Verfahren ohne zusätzliche Pakete ergeben, da diese meist kleineren Pakete schneller übertragen werden können. Nichtsdestotrotz korreliert diese Metrik stark mit der letztendlichen Zeit, sofern man davon ausgeht, dass nicht die Berechnungen an den Knoten, sondern die Sendevorgänge der limitierende Faktor bei der Kommunikation sind.

Wieder gilt für die bessere Vergleichbarkeit, dass eine Nutzdatenmenge $v = 20\,000$ Byte übertragen werden soll. Dabei wird aber kein Pipelining eingesetzt. Das bedeutet, dass eine neue Generation erst



(a) Hashwerte

(b) Checksummen

(c) MACs

(d) Signaturen

(e) Adaptives Verfahren

(f) Bestes Verfahren

Abbildung 3.26: Anzahl der Zeiteinheiten T in Abhängigkeit der Generationsgröße h und der Anzahl an Hops ℓ im generalisierten Modell.

gesendet werden kann, wenn alle Pakete der letzten Generation am Empfänger angekommen sind. Für das Hashverfahren sieht die Formel für die Zeiteinheiten wie folgt aus:

$$T_{\text{Hashwerte}} = \left\lceil \frac{v}{(w - 32h - 8)} \right\rceil * \left(\ell + \left\lceil \frac{128h}{w - 136} \right\rceil \right). \tag{3.9}$$

Der linke Faktor in Formel (3.9) berechnet, wie viele Pakete übertragen werden müssen, um die bestimmte Datenmenge v zu übertragen. Der rechte Faktor ist die Summe aus der Anzahl der Hops ℓ (da soviel Zeit vergeht, bis alle Pakete einer Generation beim Empfänger sind) und der Anzahl der Hashpakete pro Generation, welche vor der eigentlichen Übertragung geschickt werden müssen und somit den gesamten Vorgang pro Hashpaket um eine Zeiteinheit pro Generation verzögern. Analog dazu sind die Formeln für die anderen sicheren Verfahren wie folgt:

$$T_{\text{Checksummen}} = \left\lceil \frac{v}{(w - h - 8)} \right\rceil * (3\ell - 1), \tag{3.10}$$

$$T_{\text{MACs}} = \left\lceil \frac{v}{(w - 16(h + \ell) - 8)} \right\rceil * \left(0{,}5\ell^2 + 1{,}5\ell\right), \tag{3.11}$$

$$T_{\text{Signaturen}} = \left\lceil \frac{v}{\left(w - 264 - \ell(h + 5)\left(1 + \frac{\lceil \text{ld}(h) \rceil}{8}\right)\right)} \right\rceil * \ell. \tag{3.12}$$

In Abbildung 3.26 sind die Ergebnisse unter Nutzung der Formeln (3.9) - (3.12) wieder graphisch dargestellt. Auch hier zeigen die Abbildungen 3.26 (a) - (d) die Werte der einzelnen Verfahren, (e) und (f) die Ergebnisse der jeweils besten Verfahren.

Auch für die Zeiteinheiten gelten ähnliche Aussagen wie zuvor. Die Hashwerte hängen mehr von h, Checksummen eher von ℓ und die beiden anderen Verfahren von beiden Variablen ab. Insgesamt sind die Regionen, wo die einzelnen Verfahren das beste Ergebnis liefern, etwas verrutscht. Das Signaturverfahren lohnt sich nur noch für $\ell \leq 12$ mit $h > 10$. Das MAC-Verfahren bietet nie die beste Leistung und das Verfahren mit Checksummen lohnt sich schon ab $h > 30$ und $\ell > 12$. Das Hashverfahren benötigt daher für den Fall $h < 10$ als auch für den Fall $h < 30$ und $\ell > 12$ die wenigsten Zeiteinheiten.

Ein Diagramm, bei dem das beste Verfahren für alle 3 Effizienzparameter überlagert dargestellt wurde, ist in Abbildung 3.27 zu sehen. Dabei zeigt sich, dass gesamt betrachtet die Regionen mit der besten Effizienz je nach Effizienzparameter relativ ähnlich sind. Das bedeutet, dass ein Verfahren, welches für ein bestimmtes Netzwerk mit h und ℓ gut für die relativen Nutzdaten P ist, auch relativ gute Leistungen bezüglich der anderen beiden Effizienzparameter hat. Die in Abbildung 3.27 dargestellten Flächen ohne Gitter sind solche Regionen, in denen ein Verfahren für alle 3 Effizienzparameter die beste Leistung erbringt. Diese Überdeckung der Regionen zeigt sich auch bei den Ergebnissen mit den parametrisierten Modellen. Weiterhin gilt, dass potentiell jedes Verfahren als effizient gelten kann, je nach h und ℓ im gegebenen Netzwerk sowie der Gewichtung der Effizienzparameter.

Allerdings sieht man gerade in den Abbildungen 3.24 - 3.26 (e) (Adaptives Verfahren), dass man versuchen sollte, ein Netzwerk so zu konstruieren, dass es eine geringe Tiefe und damit relativ wenig Hops ℓ besitzt. Ist eine eigene Konstruktion nicht möglich, sollte zumindest die Generationsgröße h nicht zu hoch gewählt werden, um eine möglichst hohe Effizienz zu erreichen.

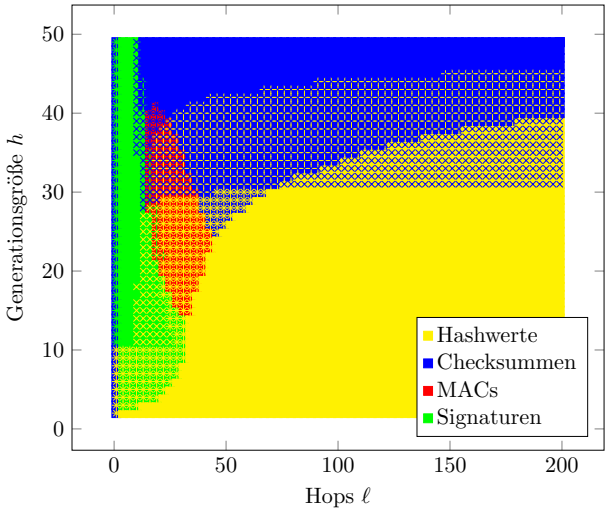


Abbildung 3.27: Überlagerte Darstellung des besten Verfahrens für alle 3 Effizienzparameter in Abhängigkeit der Generationsgröße h und der Anzahl an Hops ℓ im generalisierten Modell. Der Untergrund entspricht den relativen Nutzdaten P, das vertikale und horizontale Gitter den Sendeoperationen im Netzwerk O, das diagonale Gitter den Zeiteinheiten T.

3.6 Zusammenfassung

In diesem Kapitel sollte untersucht werden, welche Gruppen von integritätssichernden Verfahren auf Basis der Kryptographie existieren und wie deren Effizienz ist. Nach der Auswahl und Erläuterung verschiedener Vertreter konnten die unterschiedlichen Eigenschaften anhand 3 parametrisierter Netzwerkmodelle gezeigt werden. Es wurde ein umfassender Vergleich mit verschiedenen Effizienzparametern durchgeführt. Jedoch wurde ersichtlich, dass es nicht ein bestes Verfahren gibt, welches für alle Fälle die besten Effizienzparameterwerte erreicht.

Daher wurde das generalisierte Netzwerkmodell entwickelt, welches nur die entscheidendsten Parameter für das Netzwerk enthält. Mithilfe von Formeln konnten die Werte für die verschiedenen Verfahren sowie für die unterschiedlichen Effizienzparameter berechnet werden. Weiterhin wurde untersucht, bei welchen Werten welches Verfahren die besten Effizienzwerte erreicht. Somit konnte gezeigt werden, dass jedes Verfahren seine Vorteile je nach Beschaffenheit des Netzwerks hat.

Jedoch gibt es einige Einschränkungen bezüglich der Aussagekraft der Untersuchungen. Um das generalisierte Modell möglichst einfach zu halten, wurden nur 2 Parameter aufgenommen. Möglich ist, dass bei Netzwerken, für die Annahmen, wie z. B. dass die Generationsgröße in etwa der Anzahl ausgehender Kanten entspricht, nicht gelten, abweichende Ergebnisse entstehen könnten, wobei dies nur in Spezialfällen auftreten sollte. Weiterhin wurden Größen von Signaturen, Hashes etc. festgelegt. Dadurch ist die Beeinflussung der Effizienz durch diese Größen statisch und nicht variabel. So wird eine Erhöhung der Signaturgröße eine Verschiebung der Regionen der besten Verfahren mit sich bringen. Auch wurden nur 4 Beispiele für Gruppen von Verfahren ausgewählt. Es ist durchaus möglich, dass eine Änderung des Signatur- oder MAC-Verfahrens zu Verschiebungen bei den Ergebnissen führen könnte.

Weiterhin zeigen alle sicheren Verfahren bei gewissen Netzwerken eine relativ schlechte Leistung. Die Annahme, dass Verfahren, welche auf Kryptographie basieren, geeigneter sind als solche, die auf Codierungstheorie basieren, ist daher noch einmal zu prüfen. Zudem wurde in dieser Untersuchung davon ausgegangen, dass die Generationsgröße an den maximalen Fluss im Netzwerk angepasst ist. Allerdings

wurde in der Literatur auch sogenanntes ratenloses Codieren vorgestellt [SSM08]. Dabei wird die Generationsgröße h unabhängig vom Netzwerk gewählt und der Empfänger meldet dem Sender, wie viele Nachrichten er noch benötigt. Diese Vorgehensweise ist gerade vorteilhaft, wenn man davon ausgeht, dass Angreifer tatsächlich aktiv angreifen und Nachrichten verfälschen. Diese realistischeren Szenarien müssen weiter untersucht werden. Daher soll im nächsten Teil auf die Auswirkungen von Verfälschungen bei unterschiedlichen Paradigmen (Kryptographie oder Codierungstheorie) sowie bei ratenloser oder herkömmlicher Netzwerkcodierung eingegangen werden.
Zusätzlich dazu sollen Möglichkeiten gezeigt werden, wie die Bestätigungen, welche bei der ratenlosen Codierung zum Einsatz kommen, zur Beschleunigung der Übertragung genutzt werden können. Dabei soll vor allem auch das Kosten-Nutzen Verhältnis verschiedener Übertragungsmöglichkeiten ermittelt und ausgewertet werden.
Ein anderer Punkt, welcher vor allem bei den Zeiteinheiten aufgefallen ist, ist, dass jede Generation nacheinander, also sequentiell abgearbeitet werden muss. Algorithmische Optimierungen, wie z. B. das Pipelining, könnten die Übertragungszeit und damit die Effizienz verbessern. Auch Implementierungen in bestehende Infrastrukturen, wie TCP/IP, und Optimierungen, z. B. über ein „Sliding Window“ [SSM+11], sind mögliche Ansätze für Effizienzverbesserungen.
Allerdings muss auch erwähnt werden, dass die tatsächlichen Zeiten der Ausführung eine wichtige Rolle für den Vergleich der Verfahren spielen. Bei diesen Analysen wurde davon ausgegangen, dass es sich bei den Knoten um sehr leistungsstarke Knoten handelt, sodass beispielsweise der Unterschied zwischen asymmetrischen und symmetrischen Verschlüsselungsoperationen gegenüber der Übertragungszeit zu vernachlässigen sind. Von daher müssten für realistische Untersuchungen des Zeitverhaltens tatsächliche Implementierungen in festgelegten Systemen stattfinden, auf welche aufgrund der Allgemeingültigkeit der Ergebnisse auch im nächsten Kapitel verzichtet wird.

4 Sichere und effiziente Netzwerkcodierung in Gegenwart von Angriffen

4.1 Einführung

4.1.1 Grundlagen

In diesem Kapitel soll erneut die Absicherung der Integrität und der Verfügbarkeit der Kommunikation mit Netzwerkcodierung untersucht werden. Dabei soll es weniger um die Effizienz bestimmter Verfahren gehen, sondern vielmehr um die theoretisch erreichbare Effizienz bei der Verwendung verschiedener grundlegender Paradigmen. Weiterhin sollen algorithmische Verbesserungen erarbeitet, geprüft und bewertet werden. Das Kapitel basiert hauptsächlich auf den Arbeiten „Comparison of Different Secure Network Coding Paradigms Concerning Transmission Efficiency“ [PF13a] und „Adjustable Redundancy for Secure Network Coding in a Unicast Scenario“ [PF14].
Im vorherigen Kapitel wurde nur die Effizienz der auf Kryptographie basierenden Ansätze untersucht, da davon ausgegangen wurde, dass diese gegenüber Verfahren, welche auf Codierungstheorie basieren, eine höhere Effizienz ermöglichen. Diese Annahme soll zu Beginn des Kapitels noch einmal überprüft werden. Weiterhin soll in diesem Kapitel die Effizienz nicht nur für den grundlegenden Fall ohne Angriffe untersucht werden. In diesem Kapitel wird von der Gegenwart eines aktiven Angreifers ausgegangen, der Pakete verfälscht und damit die Kommunikation aktiv stört.
In diesem Zusammenhang stellt sich die Frage, wie genügend Pakete vom Sender zu den Empfängern gelangen. Zum einen gibt es die Möglichkeit, die Paketausfälle durch eine vordefinierte Anzahl redundanter Pakete auszugleichen. Zum anderen können die Empfänger den Sender über ihren Status informieren, der darauf hinweist, wann ein Empfänger genügend Pakete empfangen hat.
Nachdem die Fragen der Netzwerkabhängigkeit im vorherigen Kapitel geklärt wurden, soll in diesem Kapitel auch geklärt werden, welche Verbesserungen auf Protokollebene erzielt werden können. Dazu sollen verschiedene Ideen vorgestellt, implementiert und ausgewertet werden.

4.1.2 Fragestellungen

Grundsätzlich ergeben sich folgende Hauptfragestellungen:

- Welche Ansätze (basierend auf Kryptographie oder Codierungstheorie) sind effizienter?
- Welche Übermittlungsart (vordefinierte Redundanz oder ratenlose Übermittlung mit Bestätigungen vom Empfänger) ist am effizientesten bei Paketausfällen durch aktive Angreifer?
- Kann die Effizienz durch Übermittlungsprotokolle gesteigert werden? Welche Protokolle eignen sich besonders dafür?

Zum Klären der Fragestellungen wird im Folgenden wieder das Systemmodell mit der Netzwerktopologie, den Effizienzparametern und dem Angreifer besprochen. Nach der Vorstellung der Methodik der Untersuchungen werden zuerst die Fragen bezüglich der Ansätze und der Übermittlungsart geklärt. Anschließend wird auf mögliche Protokolle und deren Effizienzverbesserung eingegangen. Das Ende des Kapitels stellt eine Zusammenfassung mit kurzem Ausblick dar.

4.2 Systemmodell

4.2.1 Netzwerk

Da in vorherigen Untersuchungen bereits auf die Netzwerkabhängigkeit eingegangen wurde und gezeigt werden konnte, dass ein generalisiertes Modell im Allgemeinen ausreicht, wird für die Untersuchungen in diesem Kapitel nur ein allgemeines Netzwerkmodell verwendet. Dieses ist gleichmäßig aufgebaut und hat einen Sender, der eine direkte Verbindung mit d nachfolgenden (Zwischen-)Knoten hat. Diese wiederum haben jeweils ν Verbindungen zu der nächsten Ebene, welche aus ebenfalls d Knoten besteht. Es gibt insgesamt ℓ solcher Ebenen, wobei die letzte Ebene aus d Empfängern besteht.

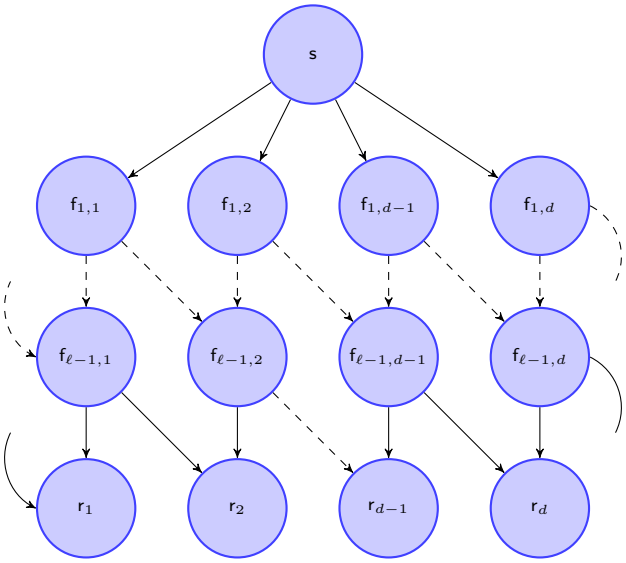


Abbildung 4.1: Allgemeines Netzwerkmodell mit Netzwerkbreite d, Netzwerktiefe ℓ und Verbindungsgrad $\nu = 2$.

Abbildung 4.1 zeigt das Netzwerkmodell für eine variable Netzwerkbreite d sowie eine variable Netzwerktiefe ℓ. Da ein parametrisierter Verbindungsgrad ν nur schwer darstellbar ist, ist dieser in der Abbildung 4.1 auf $\nu = 2$ festgelegt. Jedoch sind Werte von $\nu = 1$, also nur eine direkte Verbindung zum Nachfolgeknoten, bis zu $\nu = d$, was vollvermaschte Ebenen bedeutet, möglich. Damit die Anzahl der Eingangskanten auch ν entspricht, gilt folgende Vorschrift für die Kanten:

$$(\mathsf{f}_{\theta,i}, \mathsf{f}_{\theta+1,j}) \in \mathbf{E} \Leftrightarrow j - i \pmod{d} < \nu. \tag{4.1}$$

Diese Vorschrift gilt analog auch für die Kanten $(\mathsf{f}_{\ell-1,i}, \mathsf{r}_j)$.
Für die Untersuchung auf Protokollebene wird von einem noch einfacheren Modell ausgegangen. Zwar ist das Protokoll relativ unabhängig von dem zugrunde liegenden Netzwerk, jedoch wird für eine einfa-

chere Simulation von einer Punkt-zu-Punkt-Übertragung (Unicast) über genau einen Pfad ausgegangen. Das bedeutet, dass es einen Sender s und einen Empfänger r gibt, die genau über einen Pfad über $\ell - 1$ Zwischenknoten kommunizieren.

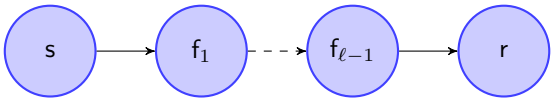


Abbildung 4.2: Netzwerkmodell für Unicastverbindungen. Spezialfall vom allgemeinen Systemmodell mit $d = \nu = 1$.

Wie Abbildung 4.2 zeigt, ist dies nur ein Spezialfall des allgemeinen Netzwerkmodells mit $d = \nu = 1$. Wichtig für die Einführung von Feedback vom Empfänger zum Sender ist auch, dass ein Kanal zwischen Empfänger und Sender besteht. Dieser kann z. B. ein dedizierter Kanal mit beschränkter Bandbreite sein oder genau entgegengesetzt zu der eigentlichen Übertragung erfolgen. Im Folgenden wird von letzterem ausgegangen.

4.2.2 Effizienzparameter

Für die Auswertung, welche Paradigmen bei der Netzwerkcodierung am effizientesten sind, wird die Metrik der Paketübertragungseffizienz E_p eingeführt. Sei $\mathrm{in}(\mathsf{v})$ eine Funktion, die aus allen validen Paketen die Knoten $\mathsf{v} \in \mathbf{V}$ empfangen hat, eine Matrix formt. Weiterhin sei $|\mathrm{out}(\mathsf{v})|$ die Anzahl der von Knoten v gesendeten Pakete. Zusätzlich gelte für die Decodierbarkeit folgende Funktion:

$$\mathrm{dec}(\boldsymbol{Y}) = \begin{cases} h, & \text{wenn } \mathrm{rg}(\boldsymbol{Y}) = h \\ 0, & \text{sonst} \end{cases}. \tag{4.2}$$

Das bedeutet, dass ein Empfänger h Pakete decodieren kann, sofern die aus den empfangenen Linearkombinationen zusammengesetzte Matrix vollen Rang hat. Ansonsten kann er kein einziges der Pakete decodieren. Weiterhin sei die Paketübertragungseffizienz E_p wie folgt definiert:

$$E_p = \frac{\min_{\mathsf{r} \in \mathbf{R}} \mathrm{dec}((\mathrm{in}(\mathsf{r})) \cdot |\mathbf{R}|}{|\mathrm{out}(\mathsf{s})|}. \tag{4.3}$$

Die Minimum-Funktion ist notwendig, da die Übermittlung einer Generation nur dann als erfolgreich bezeichnet werden kann, wenn alle Empfänger alles decodieren können. Andernfalls müssen die Daten der Generation erneut übertragen werden. Bei einem Netzwerk mit Netzwerkbreite $d = 4$ und damit $|\mathbf{R}| = 4$ und vollem Verbindungsgrad $\nu = d = 4$ ergibt sich bei beliebiger Netzwerktiefe ℓ, dass im fehlerfreien Fall ein Sender 4 Pakete schicken muss ($|\mathrm{out}(\mathsf{s})| = 4$). Wenn alle Empfänger dann 4 Nachrichten erhalten und decodieren können gilt $E_p = 4$. Sofern es nur einen Empfänger gibt, gilt $E_p = 1$ als Idealfall. Ein Wert größer 1 kann also nur im Multicast-Szenario auftreten.

Die Paketübertragungseffizienz E_p bezieht sich jedoch nur auf die Anzahl der Pakete. Um genauer bestimmte Verfahren miteinander zu vergleichen, ist es sinnvoll, den Parameter auf die Nutzdaten auszuweiten. Wie im vorherigen Kapitel gezeigt, kann von den jeweiligen Verfahren der relative Nutzdatenanteil P (für eine Generation) berechnet werden. Wird dieser Anteil mit der Paketübertragungseffizienz multipliziert, erhält man die Datenübertragungseffizienz E_d:

$$E_d = E_p \cdot P. \tag{4.4}$$

Diese Effizienzparameter werden für den Vergleich der verschiedenen Paradigmen verwendet. Die beiden genannten Effizienzparameter spiegeln hauptsächlich den Anteil der Nutzdaten und damit den Netzwerkverkehr wider. Zusätzlich dazu werden noch andere Parameter ausgewertet.
Ähnlich wie im vorherigen Kapitel wird die Anzahl der Zeiteinheiten T ausgewertet. Pro Zeiteinheit können dabei Pakete empfangen, (re)codiert und weitergeschickt werden. Das bedeutet auch, dass für eine Übertragung über i Hops mindestens i Zeiteinheiten benötigt werden. Dieser Parameter soll sich relativ zum tatsächlichen Zeitaufwand verhalten. Da dieser aber von dem gegebenen Netzwerk und der Leistungsfähigkeit der Zwischenknoten abhängig ist, ist die Angabe der Zeiteinheiten universeller und auch geeignet die Unterschiede zwischen Verfahren und Paradigmen aufzuzeigen.
Auch die Anzahl der Sendeoperationen O wird wieder als Effizienzparameter herangezogen. Dabei werden alle im Netzwerk übertragenen Pakete vom Sender zu den Empfängern gezählt. Jede Sendeoperation eines Zwischenknotens erhöht die Anzahl der Sendeoperationen O ebenfalls. Diese Metrik sollte mit dem jeweiligen Energieverbrauch im Netzwerk korrespondieren. Auch hier gilt wie für die Zeiteinheiten, dass der Energieverbrauch von den Netzwerkeigenschaften und der Ausstattung der Knoten abhängt und mit dieser theoretischen Größe eine bessere Vergleichbarkeit zwischen den Verfahren und Paradigmen möglich ist.
Zusätzlich werden für Verfahren, welche nicht auf eine vordefinierte Redundanz, sondern auf Rückmeldung über z. B. Bestätigungen (Acknowlegdements) setzen, die Anzahl dieser Bestätigungen A vom Empfänger zum Sender ausgewertet. Auch hier wird jede Weiterleitung eines Zwischenknotens als weiteres Senden einer Bestätigung gezählt. Im Prinzip entsprechen diese den Sendeoperationen in die entgegengesetzte Richtung. Da es aber möglich wäre, die Bestätigungen über einen extra Kanal zu übertragen, werden Sendeoperationen und Bestätigungen getrennt betrachtet.
Insgesamt stehen damit genügend Effizienzparameter zur Verfügung, welche die Netzwerkauslastung, den Energieverbrauch und den Zeitaufwand widerspiegeln. Alle Parameter werden in diesem Kapitel unter Berücksichtigung eines aktiven Angreifers und dem damit einhergehenden Paketverlust ermittelt. Die Möglichkeiten und Ziele des Angreifers werden im nächsten Abschnitt detailliert beschrieben.

4.2.3 Angreifer

Ein relevanter Aspekt des Angreifermodells für die hier angestrebten Vergleiche ist die Verbreitung des Angreifers. Bei Angreifern gibt es prinzipiell eine Unterscheidung nach den Systemteilen, die der Angreifer kontrollieren kann. Zum einen gibt es einen Angreifer, welcher Knoten des Übertragungsnetzwerks kontrollieren kann, zum anderen einen außenstehenden Angreifer.
Der Angreifer, welcher Zugriff auf einen Teil der Knoten $\mathbf{V}'$ des Übertragungsnetzwerks $\mathbf{V}$ (mit $\mathbf{V}' \subset \mathbf{V}$) hat, kann alle ausgehenden Kanten der Knoten $\mathbf{V}'$ kontrollieren. Außenstehende Angreifer hingegen können prinzipiell nur eine Menge von Kanten im Netzwerk $\mathbf{E}' \subseteq \mathbf{E}$ angreifen. In den folgenden Untersuchungen wird von solchen nicht an der Übertragung beteiligten Angreifern ausgegangen. Prinzipiell kann dadurch jede Kante angegriffen werden. Weiterhin wird von einem aktiven Angreifer ausgegangen, welcher Pakete nicht verzögert oder verwirft, sondern bloß verfälscht, da nur dies eine Verschmutzung im Netz verursachen kann. Um zu bestimmen, ob ein Angreifer eine Nachricht auf einer Kante $\mathsf{e} \in \mathbf{E}$

verändert hat, wird jeder Kante eine Angriffswahrscheinlichkeit ρ zugewiesen. Da Fehler bei der Übertragung durch Kanalcodierung auf niedrigerer Schicht ausgeschlossen werden, bedeutet dies, dass die Übertragungswahrscheinlichkeit für unveränderte Pakete $1 - \rho$ beträgt.
Im Prinzip könnte die Netzwerkcodierung auch die Berichtigung verfälschter Pakete durch Übertragungsfehler übernehmen, jedoch ist es gerade für die Vergleichbarkeit von Verfahren, die auf Codierungstheorie bzw. Kryptographie basieren, sinnvoll, nur beabsichtigte Veränderungen zu berücksichtigen. Das liegt daran, dass bei Verfahren zur Absicherung der Integrität, die auf Kryptographie basieren, absichtliche und zufällige Verfälschungen zum Verwerfen der betroffenen Pakete an den Zwischenknoten führen würden und damit ähnliche Konsequenzen hätten. Bei codierungsbasierten Ansätzen würden Zwischenknoten unbeabsichtigte Fehler bei der Übertragung durch die Fehlerkorrektur auf tieferen Schichten erkennen und im Gegensatz zu absichtlich (durch einen intelligenten Angreifer) veränderten Paketen nicht weiterleiten. Dadurch käme es zu einem Unterschied bei den Auswirkungen, sodass im Folgenden nur von Angriffen ausgegangen wird.

4.3 Methodik

Im folgenden Teil soll die Methodik vorgestellt werden, mit welcher die Untersuchungen durchgeführt wurden. Weil es sich bei den Angriffen um Wahrscheinlichkeiten handelt und sich die Folgen eines Angriffs für eine Übertragung je nach Zeitpunkt und Ort unterscheiden, wäre eine theoretische Betrachtung mittels analytischer Modelle schwierig. Da es für die Effizienzparameter nicht auf die Berechnungsstärke der Knoten oder auf die Übertragungsgeschwindigkeiten der Kanten ankommt, sondern eher allgemeinere Konstrukte wie Zeiteinheiten T und Sendeoperationen O ausgewertet werden sollen, bietet sich eine Simulation der Übertragungen an.
Allerdings eigneten sich verfügbare Simulatoren, wie z. B. NECO [Fer09], nicht für die vorgesehenen Untersuchungen. Das Problem bei allen getesteten Simulatoren für die Netzwerkcodierung bestand darin, dass diese nicht für sichere Netzwerkcodierung vorbereitet sind und nur ungesicherte Verfahren simulieren können. Zudem ließen sich keine großen Körper auswählen, wie sie für einige sichere Verfahren benötigt werden. Weiterhin war eine Auswertung mit eigenen Effizienzparametern nur sehr beschränkt möglich. Daher wurde sich für die Entwicklung einer eigenen Simulationsumgebung entschieden.
Um eine eigene Simulationsumgebung zu schaffen, mussten zwei Eigenschaften gegeben sein. Zum einen benötigt man für die Netzwerkcodierung die Möglichkeit, in endlichen Körpern zu arbeiten. Das bedeutet, dass entsprechende Datentypen oder Bibliotheken dafür verfügbar sein mussten. Weiterhin ist eine Unterstützung für Matrizenoperationen wie Multiplikation oder Invertierung notwendig. Beide Eigenschaften sowie eine akzeptable Ausführungszeit fanden sich in der Mathematiksoftware SageMath [The17] (ursprünglich SAGE für „System for Algebra and Geometry Experimentation“). Diese basiert auf Python sowie einigen Bibliotheken, die jedoch durch die Benutzung von Cython teilweise in C-Code übersetzt wurden und damit eine hohe Performance bieten.
Insgesamt gibt es verschiedene Möglichkeiten, SageMath zu nutzen, wobei für die vorliegenden Untersuchungen Python-Programme erstellt wurden, welche die notwendigen Sage-Bibliotheken importieren. Für den Vergleich der Paradigmen wurden dazu 4 verschiedene Verfahren implementiert, je 2 auf Codierungstheorie und auf Kryptographie basierend. Jeweils eine der beiden Implementierungen erfolgte mit vordefinierter Redundanz, die andere mit ratenlosem Codieren.

Aufbauend auf dieser Implementierung entstand auch eine Simulationsumgebung, welche es möglich macht verschiedene Übertragungsprotokolle zu prüfen. Dafür wird zuerst im Hauptprogramm die Anzahl der zu übertragenden Daten und das Netzwerk festgelegt. Danach können noch Parameter wie Angriffsrate, Anzahl der Wiederholungen, Übertragungsverfahren etc. festgelegt werden. Die einzelnen Operationen der Knoten sind in Klassen gekapselt und lassen sich damit leicht modular austauschen.
Die Auswertung der Ergebnisse erfolgt mittels ausgegebener csv-Datei, welche dann für die unterschiedlichen Parameter, wie z. B. die Generationsgröße und die Angriffsrate, die Effizienzparameter gemittelt über die Anzahl der Simulationsläufe beinhaltet.
Um Ausführungszeit einzusparen, werden nicht alle Abläufe simuliert. So wird z. B. für die auf Kryptographie basierenden Ansätze angenommen, dass ein Verfälschen eines Pakets zu einer Erkennung am Nachfolgeknoten führt und dieser dann das Paket verwirft, ohne dass eine Implementierung eines Verfahrens, wie z. B. mit homomorphen Signaturen, erfolgt. Für die auf Codierungstheorie basierenden Verfahren wurde als Beispielverfahren [JLK+08] implementiert, welches auf einer Syndromberechnung basiert, damit die Auswirkungen am Empfänger untersucht werden können. Auch der Rang der Codierungsmatrix wird ausgewertet, um den Anteil nicht-innovativer Pakete zu bestimmen.
Insgesamt ist die Simulationsumgebung sehr variabel und leicht veränderbar. Daher wurde sie auch schon für weitere Forschungsfragen (z. B. [PFC+14], [PFC+16]) eingesetzt. Da die genauen Untersuchungsparameter für die vorliegenden Auswertungen unterschiedlich sind, werden diese in den einzelnen Abschnitten erläutert, bevor die Ergebnisse der Tests dargelegt werden.

4.4 Paradigmen

4.4.1 Kryptographische gegenüber codierungstheoretischen Verfahren

Im letzten Kapitel wurde angenommen, dass die Verfahren, welche auf Kryptographie basieren, effizienter sind als die, welche auf Codierungstheorie basieren. Hier soll noch einmal der Nachweis erbracht werden, dass durch die Verfahren, welche auf Codierungstheorie basieren, Verschmutzungsattacken nur begrenzt verhindert werden können und dass die Paketübertragungseffizienz E_p durch das Prüfen von Paketen an den Zwischenknoten bei den auf Kryptographie basierenden Verfahren höher ist.
Insgesamt soll dabei nicht auf 2 Vertreter der Gruppen zurückgegriffen werden, sondern vielmehr der grundlegende Unterschied untersucht werden. Bei den auf Kryptographie basierenden Verfahren überprüft jeder Zwischenknoten die Integrität der empfangenen Pakete. Erst danach werden gültige Pakete recodiert und weitergeleitet. Veränderte Pakete werden verworfen. Dadurch gibt es keinerlei Verschmutzung im Netzwerk. Es ist lediglich notwendig, dass die Empfänger aus ihren empfangenen Paketen eine Matrix $\boldsymbol{G}$ mit Rang $\mathrm{rg}(\boldsymbol{G})$ der Generationsgröße h aufstellen können. Ist dies der Fall, so können sie decodieren, andernfalls benötigen sie noch $h - \mathrm{rg}(\boldsymbol{G})$ innovative Pakete.
Bei den auf Codierungstheorie basierenden Verfahren hingegen werden die verfälschten Pakete durch die Zwischenknoten recodiert und der Empfänger muss nun die fehlerhaften Pakete wieder herauscodieren. Zwar sind viele Pakete nun keine gültigen Pakete mehr, aber möglicherweise ist nur ein verfälschtes Paket in den ungültigen Paketen enthalten. Über eine erhöhte Redundanz kann es möglich sein, dass der Empfänger wieder genügend gültige Pakete besitzt und anschließend decodieren kann.
Ein Beispielverfahren, welches auch für die Implementierung genutzt wurde, ist aus der Arbeit „Resilient Network Coding in the Presence of Byzantine Adversaries“ [JLK+08] entnommen worden. Die genaue

Arbeitsweise soll im Folgenden erklärt werden, damit erkenntlich wird, wie ein codierungsbasiertes Verfahren funktioniert.

Für die Erläuterung sei ein Netzwerk gegeben, welches eine Kapazität von c hat. Das heißt, dass der maximale Fluss/minimale Schnitt zu allen Empfängern c ist. Im einfachsten Fall soll von vordefinierter Redundanz ausgegangen werden. Das bedeutet, es sollen c Pakete pro Runde übertragen werden. Unter der Annahme, dass z Kanten durch einen Angreifer verändert werden, können im Optimalfall $c - z$ Nachrichten fehlerfrei übertragen werden. Das Verfahren aus [JLK+08] garantiert, dass die optimale Rate erreicht werden kann. Somit ist die Generationsgröße h so zu wählen, dass $h = c - z$ gilt.

Soll nun eine Generation $\boldsymbol{G} = [\mathbf{1}|\boldsymbol{X}]$ übertragen werden, wird eine Paritätsprüfungsmatrix $\boldsymbol{P}$ der Größe $w \times c$ erstellt. Darauf aufbauend wird eine Hashmatrix $\boldsymbol{H} = \boldsymbol{GP}$ der Größe $h \times c$ berechnet. Es gibt auch eine Möglichkeit, $\boldsymbol{H}$ unabhängig von der Generation $\boldsymbol{G}$ zu bestimmen. Diese wurde in „Adversarial Models and Resilient Schemes for Network Coding“ [NL08] vorgestellt und beinhaltet eine Modifikationsmatrix $\boldsymbol{L}$, welche in die Pakete der Generation integriert und entsprechend gewählt wird, sodass gilt $\boldsymbol{H} = \boldsymbol{GP} = [\mathbf{1}|\boldsymbol{X}|\boldsymbol{L}]\boldsymbol{P}$. Dies soll im Folgenden – der Übersichtlichkeit halber – aber nicht weiter betrachtet werden.

Nun können aus der Generation $\boldsymbol{G}$ genau c Pakete codiert werden, z. B. mit einer zufälligen Codierungsmatrix $\boldsymbol{B}$ der Größe $c \times h$. Diese Pakete werden über das Netzwerk transportiert und gegebenenfalls durch Angreifer verändert. Der Empfänger formt sich aus den c empfangenen Paketen eine Matrix $\boldsymbol{Y} = [\boldsymbol{B}'|\boldsymbol{X}']$ der Größe $c \times w$. Mithilfe des Wissens über $\boldsymbol{P}$ und $\boldsymbol{H}$ kann er eine Syndrommatrix $\boldsymbol{S}$ der Größe $c \times c$ erzeugen, die wie folgt berechnet wird: $\boldsymbol{S} = \boldsymbol{YP} - \boldsymbol{B}'\boldsymbol{H}$.

Besteht $\boldsymbol{S}$ nur aus dem Element 0, bzw. gilt $\mathrm{rg}(\boldsymbol{S}) = 0$, so wurde mit hoher Wahrscheinlichkeit kein Paket verändert und der Empfänger kann aus h beliebigen Paketen die Generation $\boldsymbol{G}$ decodieren. Sollte gelten $\mathrm{rg}(\boldsymbol{S}) \geq 1$, dann wurden Pakete verändert. Der Rang der Syndrommatrix $\mathrm{rg}(\boldsymbol{S})$ entspricht dabei der Anzahl der veränderten Pakete. Sollte allerdings gelten, dass ein verändertes Paket in mehreren Paketen von $\boldsymbol{Y}$ linear codiert wurde, erhöht sich der Rang der Syndrommatrix nicht. Gilt daher $\mathrm{rg}(\boldsymbol{S}) \leq c - h$, dann kann der Empfänger die Syndrommatrix mithilfe einer Matrix $\boldsymbol{A}$ so multiplizieren, dass zumindest h Zeilen mit Nullvektoren entstehen. Wird $\boldsymbol{Y}$ anschließend ebenfalls mit $\boldsymbol{A}$ multipliziert, können die h Zeilen aus $\boldsymbol{AY}$ entsprechend den Nullvektoren in $\boldsymbol{AS}$ ausgewählt werden. Diese h Zeilen entsprechen unveränderten Paketen, sodass die Generation $\boldsymbol{G}$ decodiert werden kann.

Nachdem die Verfahrensweise codierungsbasierter Ansätze erläutert wurde, kann nun auf das Testszenario eingegangen werden, welches beide Paradigmen vergleichen möchte. Gegeben sei das allgemeine Netzwerkmodell aus Abbildung 4.1 mit den Parametern $\ell = 3$ und $d = \nu = 4$. Es handelt sich also um ein Netzwerk mit 3 Ebenen und einem Sender s, wobei alle Ebenen untereinander vollvermascht sind. Da auf der ersten Ebene kein Recodieren ausgeführt wurde, wird in der Untersuchung davon ausgegangen, dass der Angreifer hier nicht verändert hat. Diese Annahme wurde getroffen, da der Fokus auf der Untersuchung der Auswirkungen von Verschmutzungsattacken liegt. Für die übrigen 32 Kanten, auf denen recodierte Pakete gesendet werden, wird von einer Angriffswahrscheinlichkeit von $\rho = 0{,}1$ ausgegangen. Das bedeutet, dass pro Sendevorgang einer Generation bzw. Runde im Mittel 3,2 Kanten angegriffen werden.

Die Festlegung auf z (der Anzahl an Kanten, welche angegriffen werden können und die Decodierung für den Empfänger trotzdem noch möglich ist) müsste also bei etwa 3 erfolgen. Ein größeres z ist im Beispiel nicht möglich, da sonst keine Information übertragen werden könnte. Dies heißt für $z = 3$, dass

$h = 1$ gesetzt werden müsste und somit nur ein Informationspaket pro Übertragung übermittelt wird. Gemäß der Paketübertragungseffizienz E_p gilt, dass im Fall eines Empfängers, welcher nicht decodieren kann, die gesamte Generation erneut an alle Empfänger übertragen werden muss. Daher ist die Frage, mit welchem z die höchste Paketübertragungseffizienz E_p erreicht wird. Da $1 \leq h \leq c$ gilt, folgt $z \in [0, 1, 2, 3]$.

In der Simulation, welche für jeden Wert von z genau 1 000 Mal wiederholt wurde, wurde für die auf Kryptographie basierenden Verfahren ermittelt, ob der Rang der Matrix $\boldsymbol{Y}$ der gültigen Pakete bei allen Empfängern ausreichend groß war, also $\mathrm{rg}(\boldsymbol{Y}) = h$ gilt. Für die codierungsbasierten Verfahren wurde überprüft, ob der Rang der Syndrommatrix ausreichend klein war, also $\mathrm{rg}(\boldsymbol{S}) \leq z$ gilt.

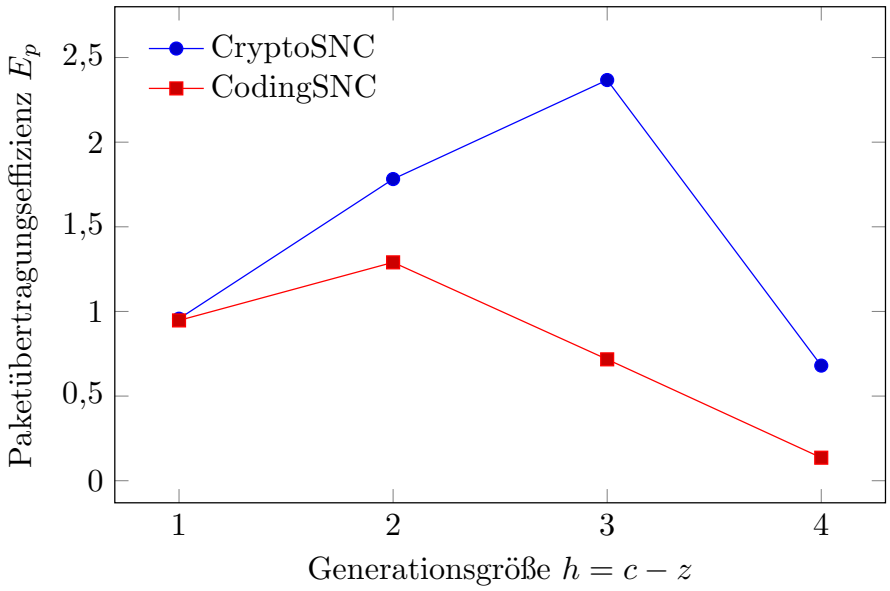


Abbildung 4.3: Paketübertragungseffizienz E_p in Abhängigkeit der Generationsgröße h für Integrität-sichernde Verfahren, welche auf Kryptographie (CryptoSNC) bzw. auf Codierungstheorie (CodingSNC) basieren.

Abbildung 4.3 zeigt die Auswertung der Simulationsergebnisse für verschiedene Werte der Generationsgröße h. Die auf Kryptographie basierenden Verfahren (CryptoSNC) zeigen für alle 4 getesteten Generationsgrößen eine höhere Paketübertragungseffizienz E_p als Verfahren, welche auf Codierungstheorie basieren (CodingSNC). Die höchste Effizienz erreicht CryptoSNC bei $h = 3$ mit $E_p \approx 2{,}4$. Bei CodingSNC liegt die höchste Effizienz bei $h = 2$ mit $E_p \approx 1{,}3$.

Interessant an dem Ergebnis ist vor allem, dass, obwohl im Mittel 3,2 Kanten angegriffen werden, $z = 3$ für beide Verfahren nicht die höchste Effizienz bietet. Zwar werden bei $z = 3$ rund 95% der Generationen an den Empfängern decodiert und damit mehr als in den anderen Fällen, dafür wird im Erfolgsfall aber auch nur 1 Paket decodiert, was sich für beide Verfahren in einer relativ niedrigen Rate von $E_p \approx 0{,}95$ auswirkt.

Insgesamt zeigt das Beispiel, dass das Herausrechnen am Empfänger nicht so gut funktioniert, wie das vorherige Aussortieren über Verfahren, welche auf Kryptographie basieren. Trotzdem gibt es natürlich auch Anwendungsszenarien, für die die codierungstheoretischen Verfahren besser geeignet sind. Ein Beispiel wäre, wenn nur rechentechnisch sehr schwache Knoten, wie z. B. in einem Sensornetzwerk,

vorhanden sind. Aber davon ausgehend, dass Zwischenknoten die Pakete überprüfen können, haben die Kryptographie-Verfahren wie erwartet deutliche Effizienzvorteile.

4.4.2 Ratenlose Codierung gegenüber vordefinierter Redundanz

Bisher wurde in dieser Arbeit davon ausgegangen, dass die Generationsgröße h an die Kapazität c des Netzwerks angepasst werden muss, damit pro Runde eine Generation übertragen wird. Jedoch ist diese Einschränkung nicht notwendig, wenn die Sender Rückmeldung von den Empfängern erhalten können. Diese Rückmeldung könnten einerseits Bestätigungen sein, dass die Knoten etwas erhalten haben (Acknowledgements) oder andererseits Anfragen, dass einen Knoten noch Informationen fehlen oder er diese nicht erhalten hat (Requests). Zwar funktionieren beide Möglichkeiten unterschiedlich und Requests eignen sich eher für wenige Fehler, Acknowledgements eher für häufigere Fehler, jedoch sind beide Möglichkeiten vergleichbar, sodass im Folgenden nur von Bestätigungen im Sinne von Acknowledgements ausgegangen wird.

Das bedeutet, ein Sender könnte so lange Pakete einer Generation schicken, bis alle Empfänger per Bestätigungsmeldung mitteilen, dass sie genügend Pakete zum Decodieren erhalten haben. Ist dem Empfänger die gewählte Generationsgröße h des Senders bekannt oder kann diese aus dem Paket entnommen werden bedeutet dies, dass die Empfänger den Empfang der Pakete bestätigen, wenn die Matrix $\boldsymbol{Y}$ der empfangenen Pakete vollen Rang, also $\mathrm{rg}(\boldsymbol{Y}) = h$, besitzt. Sei $\boldsymbol{G} = [\mathbf{1}|\boldsymbol{X}]$, dann lassen sich fast beliebig viele (q^h) unterschiedliche Pakete erzeugen und ratenlos übertragen, indem für jedes neue Paket ein Zufallsvektor $\boldsymbol{\beta}$ der Länge h erstellt wird und $\boldsymbol{\beta G}$ als Paket übertragen wird.

Diese Möglichkeit der ratenlosen Übertragung steht im Kontrast zu der Übertragung mit vordefinierter Redundanz. Wie im vorherigen Abschnitt erläutert, wird hier die Generationsgröße h passend zur Kapazität c und der geschätzten Anzahl verfälschter Pakete z gewählt. Dadurch wird versucht eine Generation so zu übertragen, dass jede Kante im Netzwerk nur einmal für die Übertragung der Nutzdaten verwendet wird. Das hat den Vorteil, dass bei bekannter maximaler Angriffs- oder Fehlerrate auf Bestätigungen verzichtet werden kann, da davon auszugehen ist, dass $h = c - z$ Pakete unverfälscht den Empfänger erreichen und dieser decodieren kann.

Für eine weitere Simulation wurde wieder das zuletzt betrachtete vollvermaschte Netzwerk mit $\ell = 3$ und $d = \nu = 4$ verwendet. Die Angriffswahrscheinlichkeit $\rho = 0{,}1$ auf den unteren beiden Ebenen wurde ebenfalls beibehalten. Für die Auswertung wurde die Decodierbarkeit bei den Empfängern betrachtet und die Paketübertragungseffizienz E_p aus $1\,000$ Simulationsläufen gemittelt.

Beim direkten Vergleich der Paketübertragungseffizienz E_p für eine Generationsgröße von $h = 4$ mit dem ratenlosen Verfahren basierend auf Kryptographie (CryptoRSNC) wurde ermittelt, dass nach der Übertragung von 4 Nachrichten in der ersten Runde bereits in $18{,}7\,\%$ der Fälle alle Empfänger decodieren konnten. Nach der zweiten Runde mit zusätzlichen 4 Nachrichten konnten in weiteren $81{,}1\,\%$ der Fälle alle Empfänger decodieren. In den restlichen $0{,}2\,\%$ der Fälle reichte dann eine dritte Runde und somit insgesamt 12 Nachrichten. Bezogen auf die (ratenlose) Paketübertragungseffizienz E_{p_r} ergibt sich daraus ein Wert von:

$$E_{p_r} = 0{,}187 * \frac{4}{1} + 0{,}811 * \frac{4}{2} + 0{,}002 * \frac{4}{3} = 2{,}373. \tag{4.5}$$

Dieser ist ähnlich dem besten Wert für das Verfahren mit vordefinierter Redundanz (CryptoSNC) bei $h = 3$ und $z = 1$ mit $E_p = 2{,}367$. Werden die codierungtheoretischen Ansätze betrachtet, zeigt sich ein anderes Bild. Bei $h = 4$ erreicht das ratenlose Verfahren (CodingRSNC) mit $E_p = 1{,}847$ eine höhere Paketübertragungseffizienz als das analoge Verfahren mit vordefinierter Redundanz (CodingSNC) im besten Fall, d. h. bei $h = z = 2$ mit $E_p = 1{,}29$. Trotzdem bleibt auch hier ein deutlicher Abstand zwischen kryptographischen und codierungstheoretischen Verfahren.
Diese Ergebnisse galten für $h = c$, jedoch kann bei der ratenlosen Netzwerkcodierung h auch unabhängig von c gewählt werden, um eine höhere Effizienz zu erreichen. Im Folgenden sollen die ratenlosen Verfahren für unterschiedliche Generationsgrößen $4 \leq h \leq 40$ getestet werden.

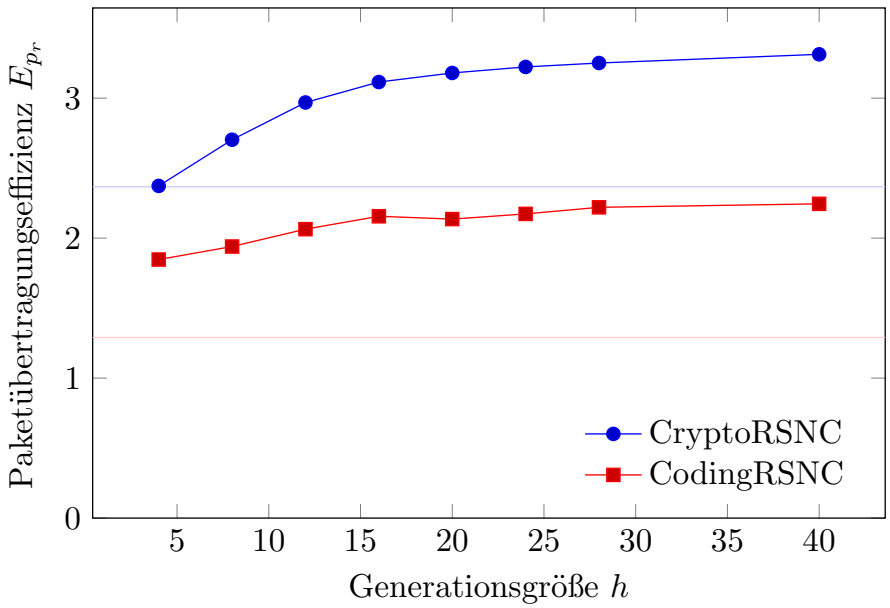


Abbildung 4.4: Paketübertragungseffizienz E_{p_r} in Abhängigkeit der Generationsgröße h für die ratenlosen Versionen CryptoRSNC und CodingRSNC. Die besten Ergebnisse für E_p bei vordefinierte Redundanz sind mit den hellen Linien für CryptoSNC (blau) und CodingSNC (rot) dargestellt.

Abbildung 4.4 zeigt die Ergebnisse der Simulation. Dabei wird die Paketübertragungseffizienz E_{p_r} in Abhängigkeit von der Generationsgröße h für die ratenlosen Verfahren (CryptoRSNC/CodingRSNC) dargestellt. Zum einfacheren Vergleich sind auch die besten Werte für E_p bei Verfahren mit fester Redundanz (CryptoSNC/CodingSNC) mit helleren Linien abgebildet.
Insgesamt zeigt sich, dass die Paketübertragungseffizienz für ratenlose Verfahren mit steigender Generationsgröße monoton steigt. Dabei ist der Anstieg anfangs stärker als bei größerem h. Es ist davon auszugehen, dass E_p sich asymptotisch an einen Maximalwert annähert. Bei einer Generationsgröße von $h = 40$ erreicht das ratenlose Kryptographieverfahren einen Wert von $E_{p_r} = 3{,}313$. Bei dem ratenlosen codierungstheoretischen Ansatz wurde ein Wert von $E_{p_r} = 2{,}245$ ermittelt.
Festzuhalten ist damit, dass bei ratenloser Netzwerkcodierung der Unterschied zwischen kryptographischen und codierungsbasierten Verfahren bestehen bleibt und zumindest bezüglich der Kommunikation bei den auf Kryptographie basierenden Verfahren deutliche Einsparungen bestehen. Weiterhin zeigt sich, dass die Verfahren mit ratenloser Netzwerkcodierung (Rückkanal vorausgesetzt) eine bessere Effizienz als die Verfahren mit festem Redundanzanteil erreichen. Dabei zeigt sich ein Zusammenhang zwischen

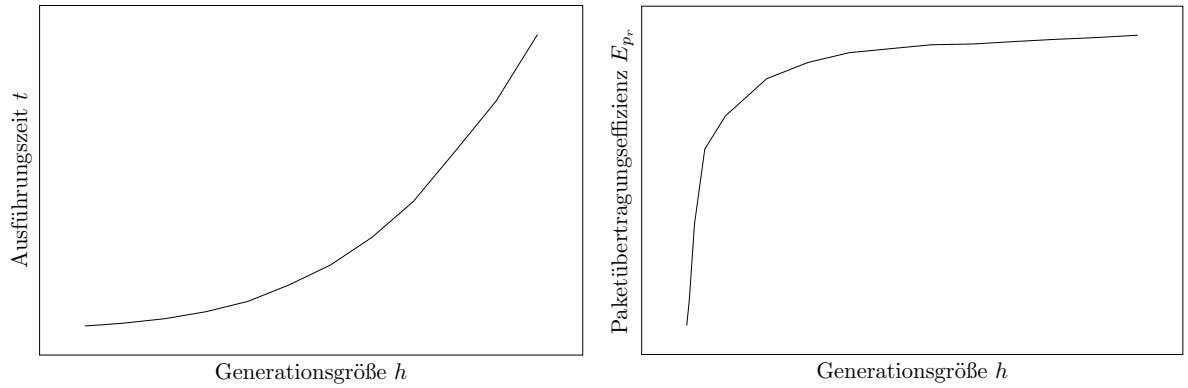
höherer Generationsgröße und höherer Paketübertragungseffizienz. Daher könnte geschlussfolgert werden, dass die ratenlose Netzwerkcodierung mit kryptographischer Absicherung und einer möglichst hohen Generationsgröße die beste Leistung erzielt. Jedoch soll im nächsten Abschnitt gezeigt werden, dass es Einschränkungen bezüglich der Generationsgröße gibt, sodass mit steigendem h die Effizienz durchaus sinken kann.

4.4.3 Einschränkungen bei der Generationsgröße

Wird nur die Paketübertragungseffizienz E_p als Metrik für die Effizienz einbezogen, dann gilt die Aussage, das mit höherer Generationsgröße h eine höhere Effizienz erreicht werden kann. Zwar nähert diese sich asymptotisch einem Maximalwert, jedoch bringt eine Erhöhung von h auch für Werte über 40 noch eine leichte Verbesserung. Eine zu große Generationsgröße h hat allerdings einen negativen Einfluss auf die Effizienz, genauer auf

- den Berechnungsaufwand für das Codieren/Decodieren und damit den Energieverbrauch,
- den Speicherbedarf an den Knoten sowie
- den Kommunikationsaufwand und damit die Netzwerkbelastung.

Um den Berechnungsaufwand zu untersuchen, wurde ein weiteres Netzwerk mit $\ell = 3$, $\nu = 2$ und $d = 4$ untersucht (Abbildung 4.1). Dabei wurde neben der Paketübertragungseffizienz E_{p_r} für das ratenlose kryptographische Verfahren auch die Ausführungszeit t der Simulation gemessen.



(a) Ausführungszeit t (b) Paketübertragungseffizienz E_{p_r}

Abbildung 4.5: Abhängigkeit der Ausführungszeit t sowie der Paketübertragungseffizienz E_{p_r} von der Generationsgröße h.

In Abbildung 4.5 ist der Zusammenhang ohne konkrete Werte dargestellt, da die absoluten Zahlen systemabhängig sind. Es zeigt sich auch hier, dass sich mit Erhöhung von h die Paketübertragungseffizienz E_{p_r} asymptotisch an einen Maximalwert annähert. Währenddessen steigt allerdings die Ausführungszeit superlinear an. Dies liegt an den Matrixoperationen, deren Aufwand bei größerer Generationsgröße h stärker als quadratisch ansteigt.
Insgesamt gesehen hat dies zur Folge, dass für relativ kleine h eine Erhöhung durchaus sinnvoll ist, da der Zeitaufwand nur minimal steigt, während sich die Paketübertragungseffizienz E_{p_r} deutlich verbessert. Ist die Generationsgröße allerdings schon recht hoch, so führt eine weitere Erhöhung nur zu einer

minimalen Verbesserung von E_{p_r}, während die Ausführungszeit deutlich ansteigt. Es gibt hier also – abhängig von dem zugrunde liegenden System und der Gewichtung von Berechnungsaufwand zu Übertragungseffizienz – einen optimalen Arbeitspunkt, der gewählt werden kann.

Auch der Speicherbedarf ist ein Aspekt der Effizienz. Teilweise ist der Speicher an (Zwischen-)Knoten limitiert oder es muss ein gewisser Aufwand (Geld, Energie) für ein Mehr an Speicher aufgebracht werden. Da Zwischenknoten recodieren, sollten diese die empfangenen Pakete der aktuellen Generation zwischenspeichern. Je größer die Generation ist, umso mehr Pakete müssen daher gespeichert werden. Dieser Speicherbedarf ist zwar nur linear zu h, bedeutet aber gerade an den Zwischenknoten, dass für eine Verdopplung von h auch eine Verdopplung an Speicherplatz nötig ist. Auch hier gilt – je nach Kosten für den Speicher – dass ein Optimum existiert, bis zu dem die Steigerung der Paketübertragungseffizienz E_{p_r} die Kosten für den zusätzlichen Speicher kompensiert. Nach diesem Punkt ist eine Steigerung von h allerdings nicht mehr sinnvoll. Zudem kommt hinzu, dass der Nutzdatenanteil pro Paket durch die größere Generationsgröße h sinkt und somit noch mehr Speicher benötigt wird.

Diese Eigenschaft betrifft auch den Kommunikationsaufwand. Mit steigender Generationsgröße h steigt auch die Anzahl der notwendigen Elemente des Codierungsvektors in jedem Datenpaket. Bei gleichbleibender Paketgröße w sinkt somit der Nutzdatenanteil der Datenpakete. Insgesamt erhöht sich somit der Kommunikationsaufwand relativ zur Nutzdatenmenge.

Zusätzlich zu dieser Problematik ist bei den sicheren Verfahren der Nutzdatenanteil durch die integritätssichernden Maßnahmen, wie z. B. der Integration von Signaturen in den Paketen, ohnehin verringert. Bei einigen Verfahren ist auch die Größe der Symbole und damit der Elemente des Codierungsvektors erhöht, sodass bei steigendem h der Nutzdatenanteil noch stärker abnimmt. Um dies zu überprüfen, wurde die Datenübertragungseffizienz E_d für die 4 beispielhaften kryptographischen Verfahren aus dem vorherigen Kapitel ermittelt. Die Berechnung wurde anhand Formel (4.4) durchgeführt und schließt die zusätzlich benötigten Pakete nicht mit ein. Trotzdem sollte der Einfluss von h zu sehen sein, da die Zusatzdaten für jedes Verfahren relativ gleich sind und sich nur zwischen den Verfahren unterscheiden. Somit ist die Aussagekraft der Datenübertragungseffizienz E_d bezüglich absoluter Zahlen nur als grobe Tendenz zu werten, während die Relationen eines jeden Verfahrens abhängig von h den realen Werten entsprechen sollten.

Abbildung 4.6 zeigt die Datenübertragungseffizienz E_d in Abhängigkeit der Generationsgröße h auch für das unsichere Netzwerkcodierungsverfahren PNC sowie für die optimalen Werte, welche E_{p_r} entsprechen. Dieser Wert verhält sich wie erwartet, indem er mit steigender Generationsgröße h monoton steigt und sich asymptotisch einem Maximum annähert. Doch schon bei dem nicht abgesicherten PNC gibt es durch den leicht verringerten Nutzdatenanteil ein Optimum bei $h \approx 40$, wobei der Wert E_d nur leicht unter E_{p_r} liegt.

Bei den einzelnen sicheren Verfahren sind die Optima für die Datenübertragungseffizienz E_d schon bei geringerer Generationsgröße h erreicht und liegen wie erwartet unter der Datenübertragungseffizienz von PNC. Für die Checksummen liegt das Optimum bei $h \approx 25$, für das Signaturverfahren bei $h \approx 16$, für das MAC-Verfahren bei $h \approx 8$ und für das Verfahren mit Hashwerten bei $h \approx 2$.

Unabhängig von den absoluten Werten und dem Netzwerk lässt sich also festhalten, dass ratenlose Netzwerkcodierung, die auf Kryptographie basiert, die beste Effizienz erreicht. Die Parametrisierung, d. h. welches Verfahren mit welchen Parametern bei welcher Generationsgröße zum Einsatz kommen sollte, ist zum einen abhängig vom Netzwerk, wie im vorherigen Kapitel gezeigt wurde, zum anderen von der

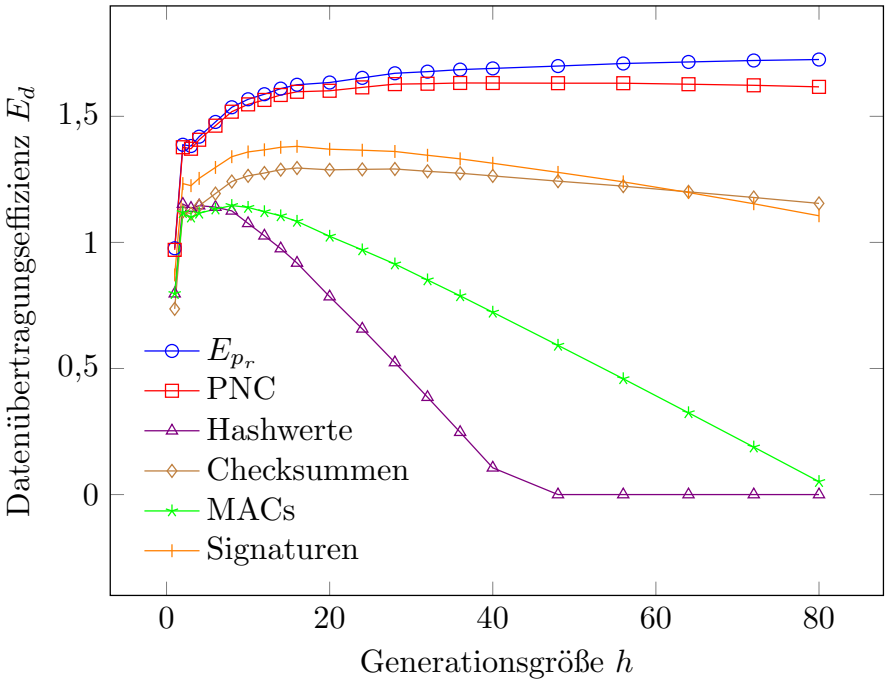


Abbildung 4.6: Datenübertragungseffizienz E_d in Abhängigkeit der Generationsgröße h für ausgewählte Beispiele der verschiedenen auf Kryptographie basierenden Verfahren.

Kostenfunktion für Kommunikation, Berechnungen sowie Speicherplatz. Es gibt jedoch immer einen optimalen Punkt, bis zu welchem die gesamte Effizienz steigt, danach aber wieder abfällt. Diese optimalen Punkte sind mithilfe einer Simulation zu finden. Um die Effizienz weiter zu steigern, werden im nächsten Abschnitt Protokolle bzw. Übertragungsverfahren vorgestellt, welche eine effiziente Übertragung ermöglichen.

4.5 Effiziente Übertragungsprotokolle

4.5.1 Protokolle

Um verschiedene Übertragungsprotokolle vergleichen zu können, werden zunächst die Annahmen und Parameter für die Simulation erläutert. Danach werden verschiedene Übertragungsprotokolle vorgestellt. Anschließend werden sie miteinander verglichen und ausgewertet.

Da in den vorherigen Untersuchungen bereits die Netzwerkabhängigkeit der einzelnen kryptographischen Verfahren evaluiert und die Vorteile der ratenlosen Netzwerkcodierung betrachtet wurde, wird im Folgenden davon ausgegangen, dass die Netzwerkcodierung durch effiziente Sicherheitsmaßnahmen, d. h. durch auf Kryptographie basierende Verfahren, abgesichert ist, ohne dass noch einmal genauer ein Verfahren betrachtet wird. Dies ist möglich, da der Ablauf der Übertragung weitgehend unabhängig von dem gewählten sicheren Verfahren ist.

Weiterhin wird zur vereinfachten Darstellung von einer Unicast-Verbindung ausgegangen. Für das Netzwerkmodell gilt damit $d = \nu = 1$. In Abbildung 4.2 wurde bereits ein Beispiel dieses Netzwerks dargestellt, welches für die Untersuchung der Protokolle verwendet wurde. Das bedeutet, dass von einem

Sender s Nachrichten über ℓ Hops zu genau einem Empfänger r übertragen werden. Für die folgenden Beispiele sei $\ell = 4$.

Da abstrakte Zeiteinheiten T verwendet werden, wird für die einzelnen Protokolle davon ausgegangen, dass ein Sender, nachdem er ein Paket übertragen hat, weiß, wann er eine Bestätigung vom Empfänger zu erwarten hat. Sollte der Sender keine Bestätigung empfangen, weiß er, dass entweder das Paket oder die Bestätigung nicht fehlerfrei übertragen werden konnte und kann entsprechend reagieren. Für das Beispiel mit $\ell = 4$ beträgt die Umlaufzeit (Round Trip Time – RTT) dann $2\ell + 1$. Das bedeutet der Sender s, welcher zum Zeitpunkt $T = 0$ ein Paket schickt, erwartet zu $T = 9$ die Bestätigung und kann zu diesem Zeitpunkt sofort reagieren.

Weiterhin wird bei der Übertragung von genau einer Generation ausgegangen. Für dieses Szenario werden die Zeiteinheiten T, die Anzahl der Sendeoperationen O und die Anzahl der Bestätigungen A für die erfolgreiche Übertragung dieser Generation gemessen, also vom Senden des ersten Pakets bis zum Empfang der finalen Bestätigung. Für die Übertragung von mehreren Generationen müssten diese Zahlen entsprechend vervielfacht werden, wenn man von einer sequentiellen Übertragung (Generation nach Generation) ausgeht. Die Möglichkeit des Pipelinings wird im Folgenden nicht betrachtet, da dann viele weitere Parameter eingeführt und untersucht werden müssten (Anzahl maximal aktiver Generationen, Speicherplatz an Zwischenknoten, maximale Latenzzeiten etc.).

Für den Angreifer wird wieder eine bestimmte Angriffsrate ρ angenommen, welche auch in einer Kantenübertragungswahrscheinlichkeit $\delta = 1 - \rho$ ausgedrückt werden kann. Angriffe können prinzipiell auf jeder der ℓ Kanten zwischen Sender s und Empfänger r erfolgen.

Nachdem die grundlegenden Annahmen und Parameter erläutert wurden, soll in den folgenden Abschnitten auf die Möglichkeiten eingegangen werden, wie die Pakete möglichst effizient bezüglich der Zeiteinheiten T, aber auch bezüglich der notwendigen Sendeoperationen O und Bestätigungen A übertragen werden können.

Grundlegende Verfahren Als einfachstes Vergleichsverfahren wird ein einfaches Weiterleitungsverfahren, hier als Routing (RT) benannt, verwendet. Dabei werden die h Pakete nacheinander übertragen und auf die Bestätigungen gewartet. Dies ist vergleichbar mit einem Sendefenster der Größe h bei TCP. Sollte eine Bestätigung nicht empfangen werden, so wird das dazugehörige Paket nochmals versendet. Der Nachteil gegenüber der Netzwerkcodierung ist, dass nicht der Rang, sondern die Ankunft des einzelnen Pakets bestätigt wird. Zum einen ist dem Sender damit nicht ersichtlich, ob das Paket nicht zugestellt wurde oder ob nur die Bestätigung verloren gegangen ist. Zum anderen wäre es im Falle von mehreren Empfängern schwierig, das richtige Paket zu senden, da unterschiedliche Pakete bei den Empfängern fehlen könnten. Da nur ein Empfänger vorhanden ist, entfällt dieser Nachteil, sollte aber für reale Anwendungen berücksichtigt werden.

Zum Vergleich dazu dient die Netzwerkcodierung (NC), bei der wie beim Routing die h Nachrichten nacheinander übertragen werden. Sollte der Sender eine Bestätigung nicht erhalten, so wird dieser ein neues (codiertes) Paket losschicken. Durch den Empfang einer späteren Bestätigung kann der Sender nachvollziehen, ob davor nur die Bestätigung (wenn höchstmöglicher Rang bestätigt wird) oder ob das Paket auf dem Hinweg (wenn der Rang kleiner als die Anzahl bis dahin möglicher empfangener Pakete) nicht übertragen werden konnte.

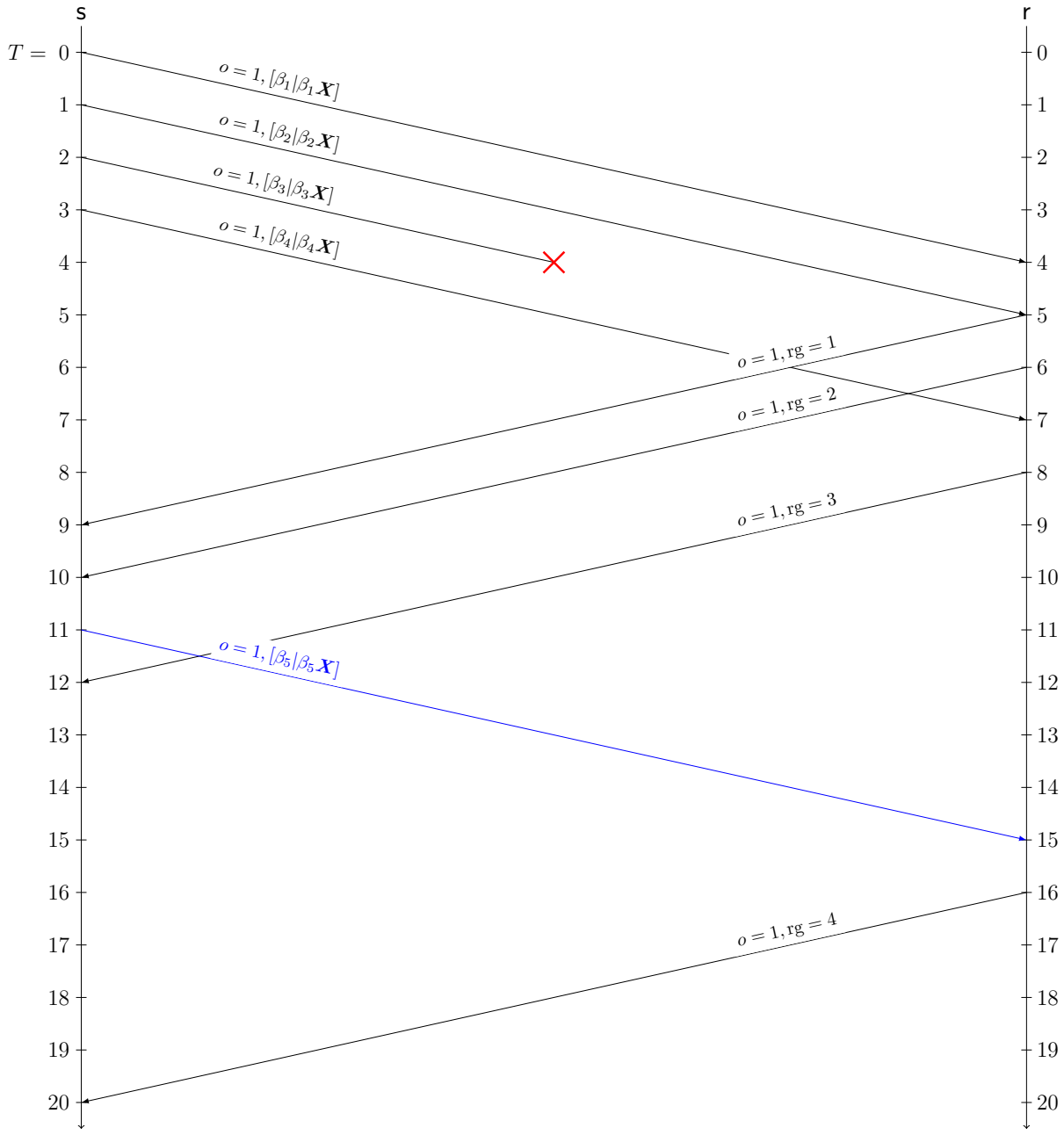


Abbildung 4.7: Beispielhafte Übertragung einer Generation $h = 4$ mit trivialer Netzwerkcodierung (NC).

In Abbildung 4.7 ist ein Beispiel für eine Übertragung mit Netzwerkcodierung (NC) mit $h = 4$ gezeigt. Die 4 am Anfang übertragenen Pakete (schwarz dargestellt) sind zwingend notwendig. Bei der Übertragung kommt es bei dem dritten Paket zu einem Angriff, sodass das Paket nicht zum Empfänger übertragen wird und dieser keine Bestätigung sendet. Da der Sender zum Zeitpunkt $T = 11$ eine Bestätigung erwartet, sendet er bei Ausbleiben der Bestätigung sofort ein neues Paket. Zum nächsten Zeitpunkt $T = 12$ bekommt er die Bestätigung des Empfängers, dass sein Rang noch nicht voll ist und kann daraus schließen, dass das dritte Paket auf dem Hinweg ausgefallen ist. Nach Empfang der Bestätigung zum Zeitpunkt $T = 20$, dass der Empfänger nun vollen Rang hat, ist die Übertragung der Generation beendet und es kann mit einer weiteren Generation begonnen werden.

Bekannte Ausfallrate Wenn dem Sender die durch den Angreifer oder Übertragungsfehler bedingte Ausfallrate bekannt ist, kann dieser versuchen, gleich so viele Pakete zu senden, dass genau h Pakete ankommen und dadurch weniger Wartezeit für die Bestätigungen benötigt wird. Die Gesamtübertragungswahrscheinlichkeit δ_{sr} von Sender s zum Empfänger r lässt sich leicht über die Kantenübertragungswahrscheinlichkeit δ berechnen. Es gilt $\delta_{\mathsf{sr}} = \delta^{\ell}$.

Ist δ_{sr} bekannt oder kann vom Sender s annähernd genau geschätzt werden, besteht die Möglichkeit zur Übertragung mittels Netzwerkcodierung mit statischer Redundanz zu Beginn ($\mathrm{NC_{ss}}$). Für das folgende Beispiel wird von $\delta_{\mathsf{sr}} = \frac{2}{3}$ ausgegangen.

In Abbildung 4.8 wird ein beispielhafter Ablauf für $h = 4$ gezeigt. Auch hier werden zuerst die 4 notwendigen Pakete nacheinander gesendet. Da der Sender die Gesamtübertragungswahrscheinlichkeit δ_{sr} kennt, kann er mittels $\mathrm{round}(\frac{h}{\delta_{\mathsf{sr}}})$ berechnen, wie viele zu sendende Pakete η im Mittel notwendig sind, damit der Empfänger h Paket erhält. Die Funktion $\mathrm{round}()$ der entspricht mathematischen Rundung auf 0 Nachkommastellen . Im Beispiel werden daher 2 weitere, redundante Pakete (blau gekennzeichnet) geschickt. Wie erwartet, erhält der Empfänger nur 4 der 6 Pakete und schickt jeweils bei Empfang eines Pakets die Bestätigung mit dem aktuellen Rang. Der Sender schickt für jede nicht empfangene Bestätigung ein weiteres Paket bis zum Zeitpunkt $T = 18$, als die Bestätigung des vollen Rangs eintrifft.

Wie man im Beispiel sehen kann, bringt dieses Verfahren eine Zeitersparnis, da nicht erst auf den Ausfall gewartet werden muss, bis neue Pakete nachgesendet werden können. Stattdessen erhöht man die Redundanz gleich zu Beginn, sodass ausreichend Pakete empfangen werden sollten. Diese pessimistische Einstellung steht damit der optimistischen Einstellung von NC entgegen, bei der davon ausgegangen wird, das die notwendige Anzahl an zu sendenden Paketen gleich der Generationsgröße ist und der Sender erst bei einem Ausbleiben der Bestätigung reagiert. Auf der anderen Seite werden bei $\mathrm{NC_{ss}}$ auch deutlich mehr Nachrichten gesendet, die eventuell nicht notwendig sind.

Eine Abwandlung davon kommt zustande, wenn man sich eine relativ niedrige Gesamtübertragungswahrscheinlichkeit δ_{sr} vorstellt. Kommt es zu einem Ausfall, was der Sender durch die fehlenden Bestätigungen bemerkt, so weiß dieser, dass die Nachricht, die er nun nachsendet, auch nur mit einer bestimmten Wahrscheinlichkeit ankommt. Die Idee für die Netzwerkcodierung mit statischer Redundanz ($\mathrm{NC_s}$) ist also, nicht nur die Redundanz zum Start einzufügen, sondern auch im Fall eines Paketausfalls $\mathrm{round}(\frac{1}{\delta_{\mathsf{sr}}})$ Pakete nachzusenden. Durch die Rundungsfunktion wird aber erst für $\delta_{\mathsf{sr}} < \frac{2}{3}$ mehr als ein Paket nachgesendet. Für $\delta_{\mathsf{sr}} \geq \frac{2}{3}$ gilt $\mathrm{NC_{ss}} = \mathrm{NC_s}$.

Der Erfolg und die Effizienz der beiden letzten Strategien, welche die bekannte Ausfallrate ausnutzen, liegt hauptsächlich an einer guten Schätzung oder dem Wissen über diese Rate. Wird die Ausfallrate zu

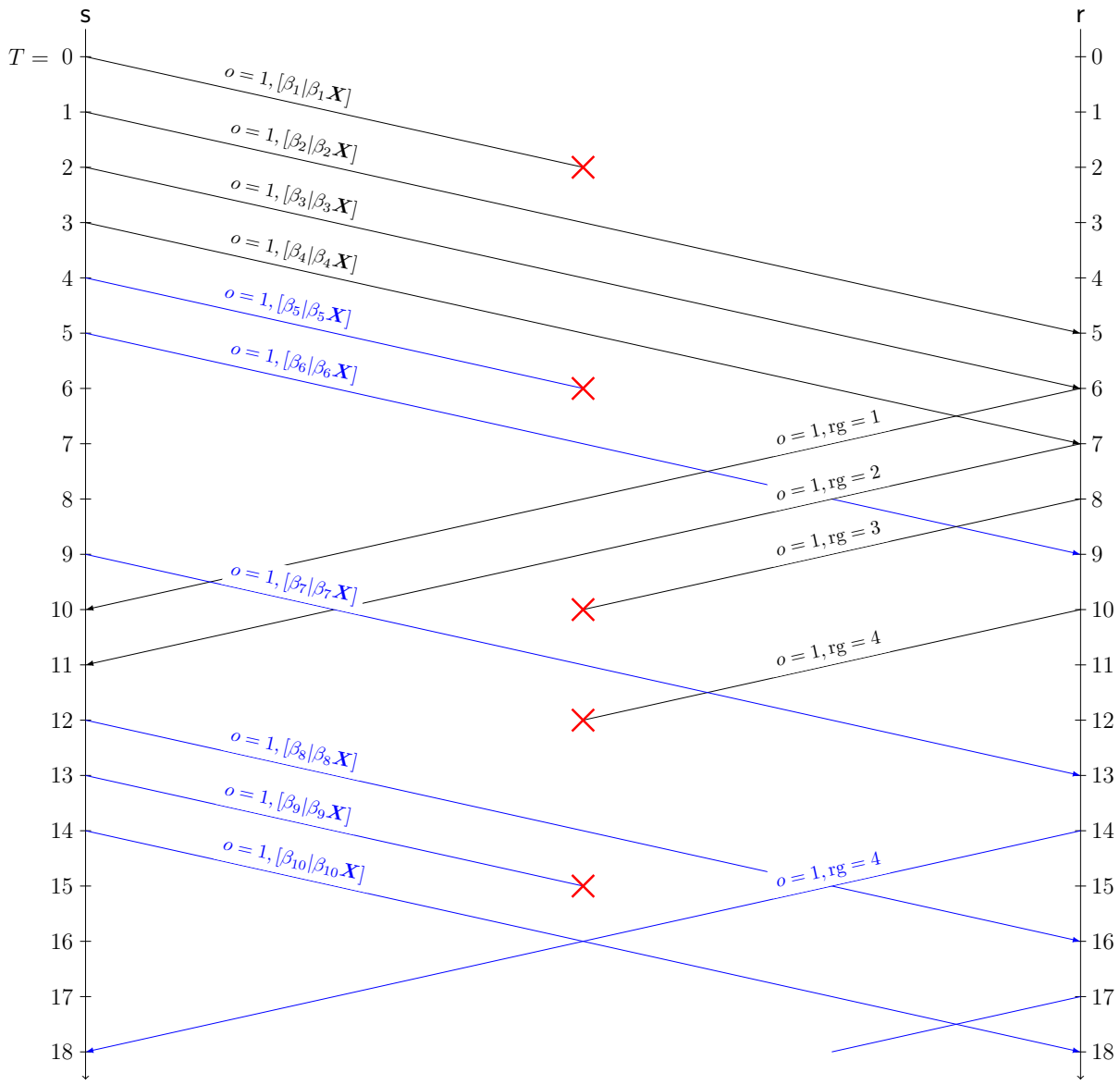


Abbildung 4.8: Beispielhafte Übertragung einer Generation $h = 4$ mit statischer Redundanz zu Beginn (NC_{ss}).

gering eingeschätzt, kommt es trotzdem zu langen Verzögerungen durch Paketausfälle. Wird sie zu hoch eingeschätzt, werden viele redundante Pakete umsonst übertragen. Insgesamt sind diese Ansätze eher theoretischer Natur, da ein Wissen über den Angreifer und somit über die Ausfallwahrscheinlichkeit nicht vorausgesetzt werden kann. Da sich aber gezeigt hat, dass dieses Wissen Vorteile für die Übertragungszeit haben kann, sollen im nächsten Abschnitt Möglichkeiten aufgezeigt werden, wie die Ausfallrate anhand der Bestätigungen abgeschätzt werden kann.

Schätzen der Ausfallrate Im Gegensatz zu den vorherigen Strategien wird nun versucht, die Gesamtübertragungswahrscheinlichkeit δ_{sr} über den Empfang der Bestätigungen abzuschätzen. Das Schätzen der Ausfallrate bringt im Übrigen auch Vorteile in dynamischen Netzwerken oder falls ein Angreifer seine Angriffsrate verändert. Im Gegensatz zu einer festen angenommenen Wahrscheinlichkeit wird sich eine Schätzung über die Bestätigungen an die aktuelle Situation anpassen.

Der Ansatz wird als Netzwerkcodierung mit dynamischer Redundanz ($\mathrm{NC_d}$) bezeichnet. Dazu werden zu Beginn h Pakete übertragen und auf deren Bestätigung gewartet, da noch kein Wissen über die Gesamtübertragungswahrscheinlichkeit δ_{sr} besteht. Erreicht den Sender eine Bestätigung, kann er den aktuellen Rang rg ablesen. Weiterhin weiß er, welchen Rang er zu diesem Zeitpunkt erwarten könnte, wenn fehlerfrei übertragen werden würde. Dieser erwartete Rang sei $\widetilde{\mathrm{rg}}$. Nun lässt sich zu jedem Zeitpunkt, an dem eine Bestätigung erwartet wird, die aktuelle Gesamtübertragungswahrscheinlichkeit δ_{sr} über das Verhältnis der Ränge schätzen: $\widehat{\delta_{\mathsf{sr}}} = \frac{\mathrm{rg}}{\widetilde{\mathrm{rg}}}$. Über die Berechnung $\mathrm{round}(\frac{h}{\widehat{\delta_{\mathsf{sr}}}})$ kann ermittelt werden, wie viele Pakete η insgesamt zu senden wären. Ist diese Zahl η niedriger als die Zahl bereits gesendeter Pakete, werden weitere Pakete gesendet. Ansonsten wird auf die nächste Bestätigung oder deren Ausfall gewartet. Dabei wird die Schätzung von δ_{sr} zu jedem Zeitpunkt aktualisiert. Je länger eine Übertragung dauert bzw. je größer die Generationsgröße h ist, umso genauer sollte eine Schätzung anhand der Bestätigungen ausfallen.

Um das ganze Verfahren zu verdeutlichen, ist ein Beispiel mit $h = \ell = 4$ in Abbildung 4.9 dargestellt. Zu Beginn werden nur $h = 4$ Nachrichten nacheinander gesendet, da keinerlei Informationen über δ_{sr} vorhanden sind. Zum Zeitpunkt $T = 9$ erwartet der Sender eine Bestätigung, d. h. $\widetilde{\mathrm{rg}} = 1$, jedoch ist keine Bestätigung angekommen, sodass $\widehat{\delta_{\mathsf{sr}}} = 0$ geschätzt wird. In diesem Sonderfall lässt sich die Anzahl der zu sendenden Pakete η nicht über die Formel bestimmen und wird daher auf $\eta = \infty$ gesetzt. Deswegen wird ein Paket gesendet, da keine genauere Schätzung möglich ist. Zum nächsten Zeitpunkt $T = 10$ ist $\widehat{\delta_{\mathsf{sr}}} = \frac{1}{2}$, da 1 von 2 erwarteten Bestätigungen angekommen ist. Daher gilt, dass die Anzahl der zu sendenden Pakete $\eta = \mathrm{round}(\frac{h}{\widehat{\delta_{\mathsf{sr}}}}) = 8$ größer ist als die Anzahl der bisher gesendeten Pakete (5) und deswegen auch hier ein weiteres Paket gesendet wird, obwohl eine Bestätigung empfangen wurde. Bei $T = 11$ ist $\widehat{\delta_{\mathsf{sr}}} = \frac{2}{3}$, sodass $\mathrm{round}(\frac{4}{\widehat{\delta_{\mathsf{sr}}}}) = 6$ gleich den bereits gesendeten Paketen ist und nicht erneut gesendet werden muss. Durch den Ausfall der Bestätigung muss zu den Zeitpunkten $T = 12$ und $T = 13$ jeweils ein weiteres Paket gesendet werden, da $\widehat{\delta_{\mathsf{sr}}} = \frac{1}{2}$ geschätzt wird und somit 8 Pakete übertragen werden müssen. Zum Zeitpunkt $T = 18$ wird eine weitere Bestätigung erwartet, welche nicht empfangen wird, was einen weiteren Sendevorgang hervorruft. Zum Zeitpunkt $T = 19$ allerdings wird durch den Empfang der Bestätigung des vollen Rangs die Übertragung beendet.

Das Beispiel zeigt also eindrücklich, dass eine Adaption anhand der Bestätigungsmeldungen möglich ist. Durch das Senden von nur h Paketen am Anfang ist das Verfahren auch nicht ineffizient bezüglich der

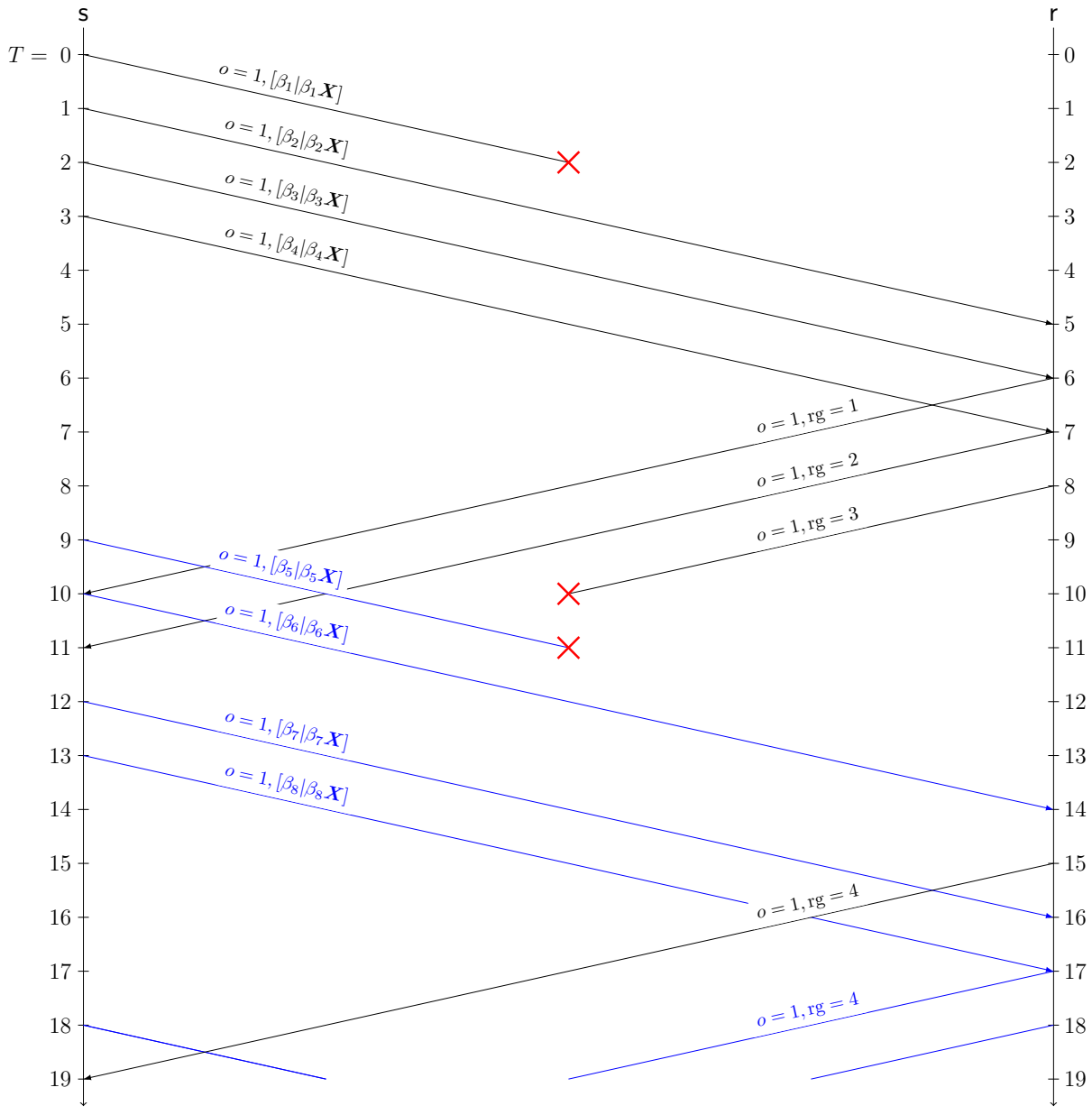


Abbildung 4.9: Beispielhafte Übertragung einer Generation $h = 4$ mit dynamischer Redundanz ($\mathrm{NC_d}$).

Kommunikation, sollte $\delta_{\mathsf{sr}} = 1$ gelten. Allerdings sieht man auch noch einige Punkte, an denen optimiert werden könnte.

Sollten z. B. h Nachrichten zu Anfang vollständig übertragen werden, aber die letzte Bestätigung des vollen Rangs auf dem Weg zum Sender verloren gehen, so muss eine komplette Umlaufzeit abgewartet werden, bis der Empfänger ein weiteres Paket erhält, um dann auch nur den vollen Rang zu bestätigen. Um diese Wartezeit zu verkürzen, könnte das Verfahren so verändert werden, dass ab Empfang des Pakets, welches zum vollen Rang führt, der Empfänger permanente Bestätigungen an den Sender schickt. Im Folgenden sei dieses Verfahren als Netzwerkcodierung mit dynamischer Redundanz und permanenten Bestätigungen ($\mathrm{NC}_{\mathrm{dp}}$) deklariert.

Obwohl diese Verfahrensweise zu einer schnellen Beendigung der Übertragung führen wird, hat sie auch ihre Nachteile. Gerade bei einer hohen Anzahl von Hops ℓ und einer relativ hohen Gesamtübertragungswahrscheinlichkeit δ_{sr} werden bis zum Eintreffen des Bestätigungspakets mindestens $\ell - 1$ weitere Bestätigungen versendet, was einen erhöhten Kommunikationsaufwand bedeutet, der eventuell nicht notwendig ist. Weiterhin ist problematisch, dass der Empfänger nicht weiß, wann er mit dem Senden von Bestätigungen aufhören kann. Dieser Zeitpunkt wäre im Zweifelsfall erst erreicht, wenn er ein Paket einer neuen Generation erhält. Dies kann dazu führen, dass er etwa 2ℓ Bestätigungen schicken muss, obwohl vielleicht schon die erste empfangen wurde.

Um dieses Problem zu umgehen, wird vorgeschlagen, die permanenten Bestätigungen durch eine Schätzung von δ_{sr} zu optimieren. Dieses Verfahren namens Netzwerkcodierung mit dynamischer Redundanz und adaptiven Bestätigungen ($\mathrm{NC}_{\mathrm{da}}$) hat jedoch ein Problem. Im Gegensatz zum Sender, der weiß, wann er die ersten Pakete geschickt hat und wann er die Bestätigungen erhalten hat, weiß der Empfänger nicht, wenn er kein Paket enthält, ob das Paket nicht gesendet wurde oder ob es auf dem Weg verloren gegangen ist. Jedoch kann man sich mit einer sehr groben Schätzung behelfen. Dazu wird als Erstes die Differenz Δ_T zwischen den Zeitpunkten des ersten Empfangs und des ersten Bestätigens des vollen Rangs ermittelt. Im Beispiel in Abbildung 4.9 wäre $\Delta_T = 15 - 5 = 10$. In dieser Zeit hätte der Empfänger also bis zu 10 Pakete erhalten können, hat aber nur $h = 4$ erhalten. Um nun abzuschätzen, wie viele Bestätigungen zu senden sind, berechnet man $\mathrm{round}(\frac{\Delta_T}{h})$, was im Beispiel zu 3 Sendungen führen würde. Die Schätzung ist dabei immer relativ pessimistisch, da Δ_T meist größer als die Anzahl gesendeter Pakete ist. Dadurch ist aber relativ sicher, dass der Sender mindestens eine Bestätigung erhält, ohne dass dabei zu viele Bestätigungen geschickt werden müssen.

Allen 3 Verfahren, welche die Ausfallrate abschätzen, ist gemein, dass diese ein festes Verhältnis von Verzögerungszeit zu Kommunikationsaufwand besitzen. Anders ausgedrückt wird aus der Schätzung der Gesamtübertragungswahrscheinlichkeit geschlussfolgert, dass $\eta = \mathrm{round}(\frac{h}{\widehat{\delta_{\mathsf{sr}}}})$ Pakete gesendet werden müssen. Dies stimmt jedoch nur im Mittel. Das bedeutet, das im Falle einer perfekten Schätzung von beispielsweise $\delta_{\mathsf{sr}} = \frac{1}{2}$ genau $2h$ Pakete gesendet werden sollten. Auf der einen Seite reichen diese $2h$ Pakete in etwa der Hälfte der Fälle nicht aus, sodass hier Optimierungspotenzial für schnellere Übertragungen vorhanden ist, wenn mehr als $2h$ Pakete geschickt werden würden. Auf der anderen Seite ist es auch nicht unwahrscheinlich, dass mit weniger als $2h$ Paketen auch alle Pakete empfangen werden könnten, sodass hier Optimierungspotenzial für die Netzwerkauslastung bzw. für eine geringere Kommunikation vorhanden ist.

Um unterschiedlichen Zielstellungen gerecht werden zu können, wird ein Gewichtungsfaktor Φ eingeführt, welcher eine Anpassung des Verhältnisses zwischen Zeitaufwand und Kommunikationsaufwand

ermöglicht. Das Verfahren wird im Folgenden als Netzwerkcodierung mit adaptiver Redundanz (NC_Φ) bezeichnet. Es gelte $0 < \Phi < 1$ und Φ beschreibt die Wahrscheinlichkeit, dass eine bestimmte Anzahl η zu sendender Pakete für eine bestimmte Gesamtübertragungswahrscheinlichkeit δ_{sr} ausreicht. Diese Wahrscheinlichkeit lässt sich über die Binomialverteilung ermitteln. Sei $\widehat{\delta_{\mathsf{sr}}}$ die geschätzte Gesamtübertragungswahrscheinlichkeit, dann lässt sich die Wahrscheinlichkeit ϕ', dass genau h von η Paketen vollständig übertragen werden, wie folgt ermitteln:

$$\phi' = \binom{\eta}{h} \widehat{\delta_{\mathsf{sr}}}^h (1 - \widehat{\delta_{\mathsf{sr}}})^{\eta - h}. \tag{4.6}$$

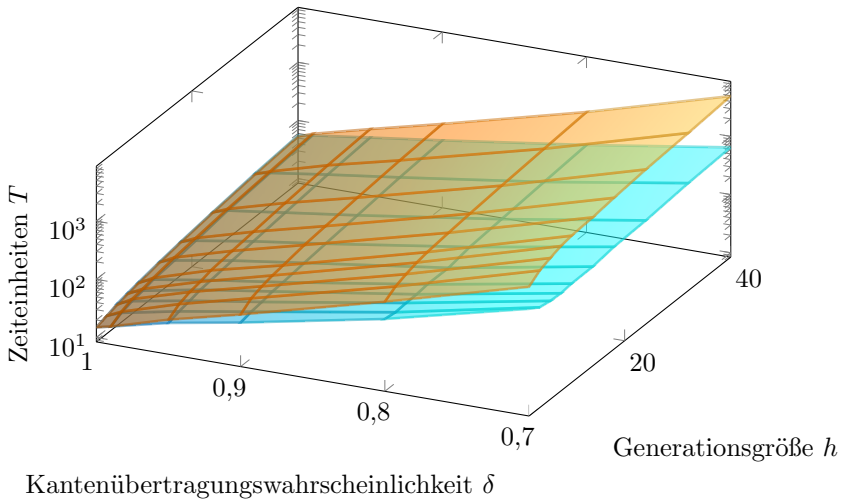
Da eine erfolgreiche Decodierung auch bei Empfang von mehr als h Paketen möglich ist, müssen diese Fälle ebenfalls beachtet werden. Dies ist mithilfe der kumulativen Wahrscheinlichkeit Φ' möglich:

$$\Phi' = \sum_{i=h}^{\eta} \binom{\eta}{i} \widehat{\delta_{\mathsf{sr}}}^i (1 - \widehat{\delta_{\mathsf{sr}}})^{\eta - i}. \tag{4.7}$$

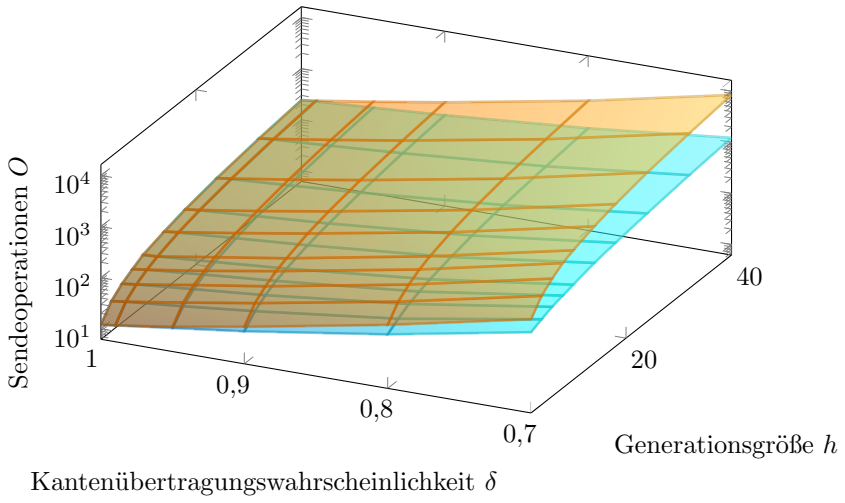
Je nach Wahl des Gewichtungsfaktors Φ wird anschließend das minimale η gesucht, welches $\Phi' \geq \Phi$ erfüllt. Die Annahme ist, dass ein kleines Φ zu einem geringeren η führt und damit weniger Pakete gesendet werden, dafür die Übertragung aber länger dauert. Bei einem großen Φ werden im Gegensatz dazu mehr Pakete geschickt, dafür sollte sich die Übertragungszeit reduzieren, da bei Paketausfall nicht so lange Wartezeiten entstehen.
Für das Beispiel in Abbildung 4.9 mit $\widehat{\delta_{\mathsf{sr}}} = \frac{2}{3}$ und $h = 4$ wären bei einem Gewichtungsfaktor $\Phi = 0{,}2$ genau $\eta = 5$ Pakete notwendig. Für $\Phi = 0{,}5$ müssten $\eta = 6$ Pakete, wie bei NC_d, geschickt werden, bei $\Phi = 0{,}8$ dann $\eta = 7$ Pakete oder für einen sehr hohen Wert von z. B. $\Phi = 0{,}99$ sogar $\eta = 11$ Pakete.
Im nächsten Abschnitt werden die Ergebnisse der Simulationen mit diesen verschiedenen Möglichkeiten ausgewertet. Dabei soll gezeigt werden, welche Optimierungen vorteilhaft und welche überflüssig sind oder die Effizienz sogar nachteilig beeinträchtigen und ob NC_Φ die erwarteten Eigenschaften besitzt.

4.5.2 Resultate

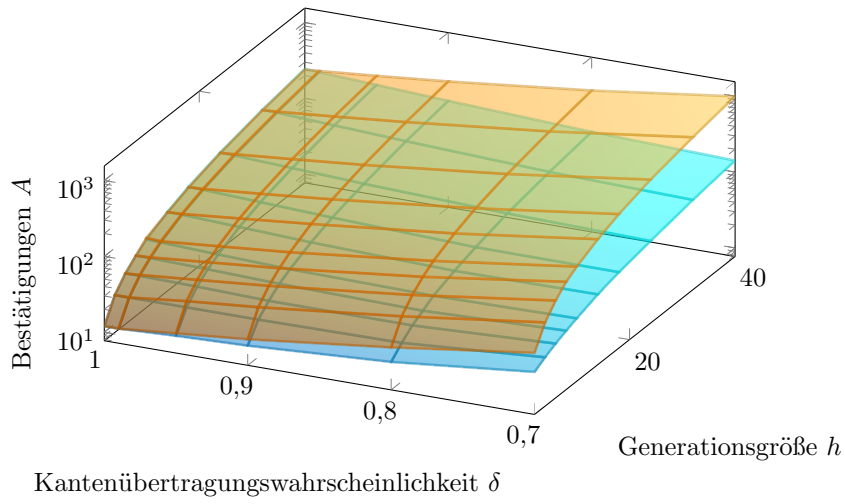
Um die vorgestellten Möglichkeiten zu vergleichen, wurden die Verfahren in die entwickelte Simulation, welche auf SageMath [The17] basiert, eingebaut. Dabei wurde genau eine Generation pro Simulationslauf übertragen. Jede Simulation eines Protokolls wurde für verschiedene Simulationsparameter durchgeführt. Dabei wurden verschiedene Generationsgrößen $h \in [2; 4; 6; 8; 10; 14; 18; 24; 32; 40]$, verschiedene Kantenübertragungswahrscheinlichkeiten $\delta \in [1; 0{,}99; 0{,}95; 0{,}90; 0{,}80; 0{,}70]$ sowie verschiedene Werte für den Gewichtungsfaktor $\Phi \in [0{,}1; 0{,}33; 0{,}45; 0{,}5; 0{,}55; 0{,}66; 0{,}8; 0{,}9]$ verwendet. Da es durch die Kantenübertragungswahrscheinlichkeit δ zu zufälligen Angriffen auf Kanten kam, wurde jeder Simulationslauf für jedes Verfahren und Parametertupel jeweils $1\,000$ Mal durchgeführt und die Ergebnisse gemittelt.
Da sich in vorherigen Untersuchungen herausgestellt hatte, dass die Anzahl der Hops ℓ kaum einen Einfluss auf die relativen Zahlen, sondern nur auf die absoluten Zahlen hatte, wurde für alle Simulationen $\ell = 6$ gewählt. Für die Sicherheitsmaßnahmen wurde ein kryptographisches System angenommen, sodass angegriffene Pakete an den Zwischenknoten erkannt und verworfen wurden.
Zu Beginn wurden die beiden grundlegenden Verfahren miteinander verglichen. Abbildung 4.10 zeigt die 3 Effizienzparameter für Routing (RT) in orange und für Netzwerkcodierung (NC) in blau. In allen 3



(a) Zeitaufwand



(b) Kommunikationsaufwand



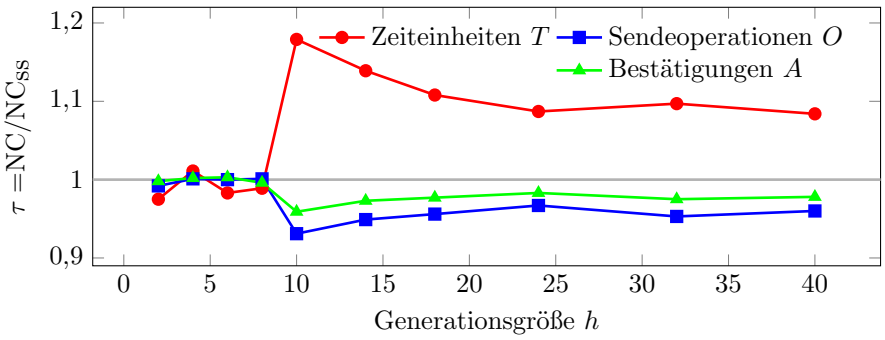
(c) Aufwand für Bestätigungen

Abbildung 4.10: Vergleich der grundlegenden Verfahren Routing (RT) in orange und Netzwerkcodierung (NC) in blau für die 3 Effizienzparameter Zeiteinheiten T, Sendeoperationen O und Bestätigungen A.

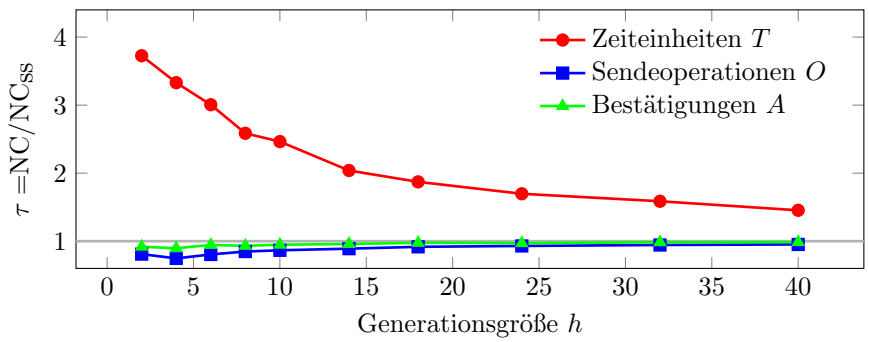
Diagrammen sieht man ein ähnliches Verhalten. Wie zu erwarten steigt der Aufwand mit einer größeren Generationsgröße h. Da dabei aber auch mehr Daten übertragen werden, lässt das keine Rückschlüsse auf die Paketübertragungseffizienz zu. Weiterhin ist für hohe Werte der Kantenübertragungswahrscheinlichkeit δ kaum ein Unterschied zwischen RT und NC auszumachen. Mit steigender Angriffsrate und damit kleinerem δ zeigt NC deutliche Vorteile gegenüber RT.

Dies liegt an zwei hauptsächlichen Eigenschaften. Zum einen handelt es sich bei nahezu der Hälfte der Paketverluste um Verluste der Bestätigungen. Bei RT sind diese Pakete für den Sender verloren und er muss nochmals übermitteln, während bei NC eine spätere Bestätigung die Ankunft vorheriger Pakete mit bestätigen kann. Zum anderen muss bei RT genau das fehlende Paket nachgesendet werden, während bei NC eine beliebige Kombination ausreicht. Zwar wird hier nur ein Netzwerk mit nur einem Pfad und einem Empfänger betrachtet, aber wenn es mehrere Empfänger geben sollte, würde der Unterschied noch eklatanter aussehen.

Insgesamt bleibt also festzuhalten, dass NC im Fehlerfall deutliche Vorteile gegenüber RT bei Unicast-Übertragungen bietet. Von daher soll NC als Basis für die weiteren Untersuchungen der vorgeschlagenen Protokolle dienen. Da es im Folgenden weniger um die absoluten Werte als vielmehr um die relativen Veränderungen gehen soll, werden die Ergebnisse der Effizienzparameter von NC durch die Ergebnisse der Effizienzparameter der anderen Übertragungsprotokolle geteilt. Dadurch ergibt sich ein Verhältnis τ, für das Folgendes gilt: Ist das Verhältnis $\tau < 1$, so ist NC in diesem Parameter effizienter als das Vergleichsverfahren; ist $\tau > 1$, so ist das andere Verfahren effizienter als NC.



(a) Kantenübertragungswahrscheinlichkeit $\delta = 0{,}99$



(b) Kantenübertragungswahrscheinlichkeit $\delta = 0{,}80$

Abbildung 4.11: Vergleich von $\mathrm{NC}_{\mathrm{ss}}$ zu NC für zwei verschiedene Kantenübertragungswahrscheinlichkeiten in Abhängigkeit der Generationsgröße h.

In Abbildung 4.11 ist dieses Verhältnis von NC zu $\mathrm{NC_{ss}}$ in Abhängigkeit der Generationsgröße dargestellt. Das obere Diagramm zeigt dabei das Verhältnis $\tau = \mathrm{NC/NC_{ss}}$ für eine hohe Kantenübertragungswahrscheinlichkeit $\delta = 0{,}99$, das untere zeigt das Verhältnis für einen niedrigeren Wert $\delta = 0{,}80$. Insgesamt gesehen spiegelt die Abbildung die Erwartungen wider. Durch zusätzliche Pakete zu Beginn wird der Bedarf an Zeiteinheiten einer Übertragung vermindert. Gleichzeitig erhöht sich jedoch die Anzahl an Sendeoperationen und Bestätigungen.

Im Detail kommt es natürlich auf die Generationsgröße h und die Kantenübertragungswahrscheinlichkeit δ an. Ist die Ausfallrate relativ niedrig, wie es bei $\delta = 0{,}99$ der Fall ist, gibt es für $h < 10$ kaum einen Unterschied zwischen den Verfahren. Das liegt daran, dass für $\ell = 6$ Hops und einer geringen Generationsgröße keine zusätzlichen Pakete zu Beginn geschickt werden, da durch Rundungen genau h Pakete ausreichen. Erst ab $h \geq 10$ werden zusätzliche Pakete geschickt, welche dann aber auch eine Erhöhung der Sendeoperationen nach sich zieht. Insgesamt werden so etwa $10\,\%$ weniger Zeiteinheiten T für etwa $5\,\%$ mehr Sendeoperationen O ausgetauscht. Der Mehraufwand für die Bestätigungen A ist etwa halb so hoch wie für die Sendeoperationen O. Zwar kann sich der Einsatz hier für bestimmte Fälle lohnen, jedoch ist der Unterschied doch recht gering.

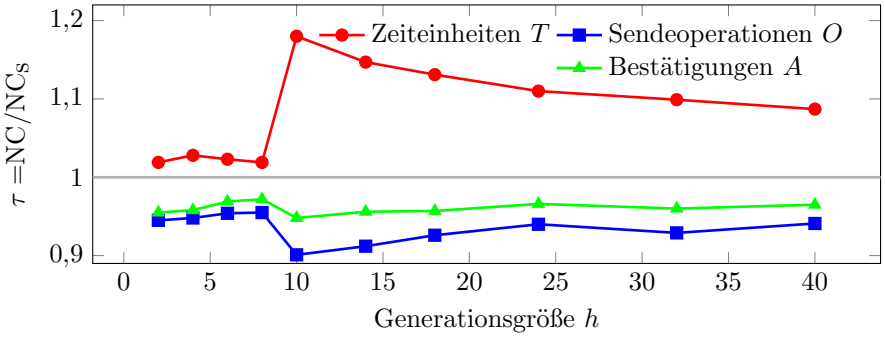
Anders sieht es bei $\delta = 0{,}80$, also einer hohen Ausfallrate, aus. Hier ist die Zeiteinsparung gerade für kleine Generationsgrößen h sehr deutlich. Dafür erhöht sich der Aufwand für die Sendeoperationen ebenfalls deutlich gegenüber $\delta = 0{,}99$. Gesamt gesehen überwiegt aber der Vorteil der deutlich geringeren Anzahl an Zeiteinheiten. Der Grund dafür ist, dass gerade bei kleinen Generationen die Wartezeiten bei Ausfällen den größten Anteil der Ausführungszeit ausmachen und hier die statische Redundanz zu Beginn diese Wartezeit in den meisten Fällen überflüssig macht. Je größer h ist, umso geringer ist der Anteil der Wartezeit an der Gesamtübertragung, weshalb der Vorteil von $\mathrm{NC_{ss}}$ gegenüber NC absinkt. Dafür kommt es für $h \geq 10$ allerdings auch kaum noch zu einem merklichen Kommunikationsmehraufwand.

Insgesamt ermöglicht dieses Protokoll, gerade wenn Angriffe stattfinden, eine deutliche Verminderung der Zeiteinheiten auf Kosten eines leicht erhöhten Kommunikationsaufwands. Ob dieses Verhältnis durch weitere Optimierungen, wie zusätzlicher Redundanz bei den Bestätigungen ($\mathrm{NC_s}$), noch gesteigert werden kann, zeigt die nächste Untersuchung.

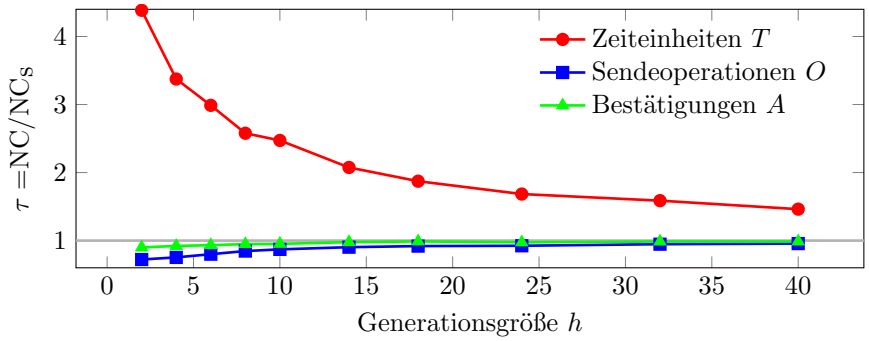
Abbildung 4.12 ist analog zur vorherigen Abbildung aufgebaut. Das obere Diagramm gibt τ für $\delta = 0{,}99$ in Abhängigkeit der Generationsgröße h an. Das untere zeigt das Verhältnis für $\delta = 0{,}80$. Jedoch bildet τ dieses Mal das Verhältnis von NC zu $\mathrm{NC_s}$ ab. Dabei zeigt sich für das obere Diagramm kaum ein Unterschied, was zu erwarten war, weil die geringe Ausfallwahrscheinlichkeit dazu führt, dass auch hier bei Paketausfall nur ein Paket nachgesendet werden muss. Dadurch verhält sich $\mathrm{NC_s}$ im Prinzip wie $\mathrm{NC_{ss}}$.

Bei dem unteren Diagramm ist das Verhältnis jedoch ein wenig verändert. Da mit der Kantenübertragungswahrscheinlichkeit $\delta = 0{,}80$ für die Gesamtübertragungsrate gilt, dass $\delta_{\mathsf{sr}} = \delta^{\ell} \approx 0{,}26$ und damit $\delta_{\mathsf{sr}} < \frac{2}{3}$, werden bei Ausbleiben einer Bestätigung mehrere Pakete nachgesendet. Dadurch verbessert sich zumindest für kleinere Generationsgrößen der Geschwindigkeitsvorteil nochmals. Auf der anderen Seite erhöht sich aber auch der Aufwand für die zusätzlichen Sendeoperationen und die zusätzlichen Bestätigungen.

In Zahlen gesprochen und über alle Generationsgrößen und Angriffsraten gemittelt reduziert $\mathrm{NC_{ss}}$ den Zeitaufwand um $47\,\%$ gegenüber NC, während die Zeiteinheiten bei $\mathrm{NC_s}$ mit $48\,\%$ Reduktion gegenüber NC nur geringfügig besser sind. Dafür benötigt $\mathrm{NC_s}$ etwa $13\,\%$ mehr Sendeoperationen und $5\,\%$ mehr



(a) Kantenübertragungswahrscheinlichkeit $\delta = 0{,}99$



(b) Kantenübertragungswahrscheinlichkeit $\delta = 0{,}80$

Abbildung 4.12: Vergleich von $\mathrm{NC_s}$ zu NC für zwei verschiedene Kantenübertragungswahrscheinlichkeiten in Abhängigkeit der Generationsgröße h.

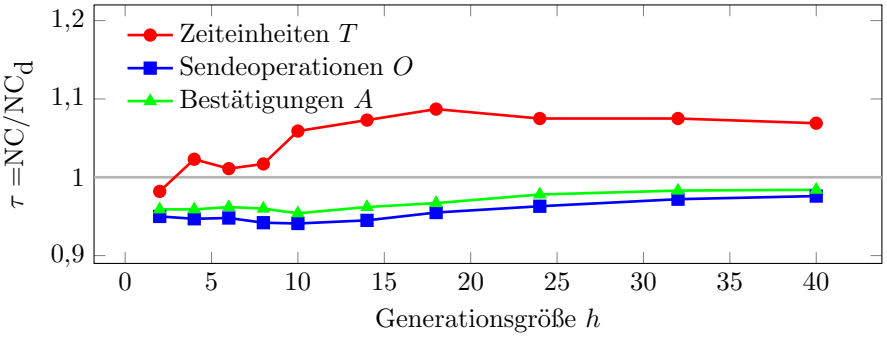
Bestätigungen. Für $\mathrm{NC_{ss}}$ ist der Mehraufwand mit $17\,\%$ bei den Sendeoperationen und $7\,\%$ bei den Bestätigungen deutlich höher, um den oben angesprochenen minimalen Zeitgewinn zu erreichen.

Insgesamt gesehen ist die Lösung $\mathrm{NC_s}$ also nur dann vorzuziehen, wenn die Zeit das entscheidende Kriterium ist, da ansonsten der Vorteil zu gering ist. Sind sowohl geringe Kommunikation als auch geringe Zeit gefragt, ist das Verhältnis von $\mathrm{NC_{ss}}$ besser. Beide Protokolle stellen eher einen theoretischen Fall dar, da Angriffsrate und Ausfallrate für die Berechnung der Kantenübertragungswahrscheinlichkeit δ notwendig sind. In praktischen Anwendungsfällen ist zumindest die Angriffsrate unbekannt. Daher soll als nächstes das Verfahren $\mathrm{NC_d}$ evaluiert werden, welches die Ausfallrate anhand der Bestätigungen abschätzt (Abbildung 4.9).

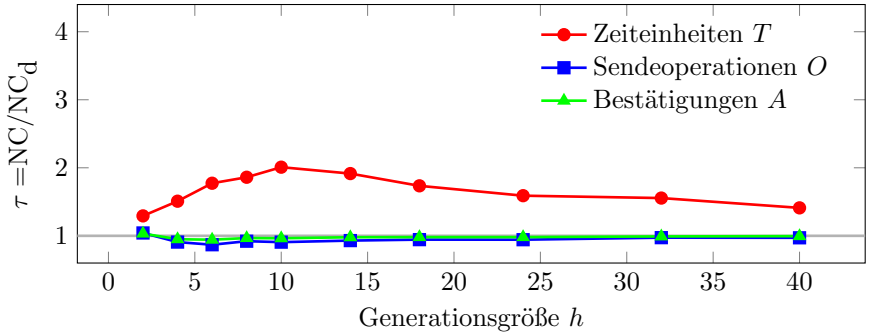
Abbildung 4.13 zeigt die Ergebnisse der Auswertung. Wieder gilt oberes Diagramm für $\delta = 0{,}99$, unteres Diagramm für $\delta = 0{,}80$. Zur besseren Vergleichbarkeit der Abbildungen zueinander wurden jeweils die gleichen Skalen für die 3 Verfahren verwendet. Insgesamt gilt auch für $\mathrm{NC_d}$, dass weniger Zeiteinheiten T benötigt werden, dafür mehr Sendeoperationen O und auch Bestätigungen A notwendig sind.

Insgesamt sieht der Verlauf der Kurven eher gleichmäßiger aus. Bei großem δ und über alle Generationsgrößen gemittelt sind die Zeitvorteile unter $10\,\%$. Dafür wird auch nur bis zu $5\,\%$ Mehraufwand bei der Kommunikation benötigt. Bei kleinerem δ sind die Zeitvorteile deutlicher, obwohl auch hier der Kommunikationsaufwand nur leicht erhöht wird.

Im Vergleich zu den anderen Verfahren $\mathrm{NC_{ss}}$ und $\mathrm{NC_s}$ fällt auf, dass die Werte für $h \geq 16$ bei den untersuchten Kantenübertragungswahrscheinlichkeiten jeweils sehr ähnlich sind, während für $h < 16$ die Zeitvorteile geringer sind. Bei $\delta = 0{,}99$ und $h < 10$ ist sogar der Zeitvorteil kaum gegeben, obwohl



(a) Kantenübertragungswahrscheinlichkeit $\delta = 0{,}99$



(b) Kantenübertragungswahrscheinlichkeit $\delta = 0{,}80$

Abbildung 4.13: Vergleich von $\mathrm{NC_d}$ zu NC für zwei verschiedene Kantenübertragungswahrscheinlichkeiten in Abhängigkeit der Generationsgröße h.

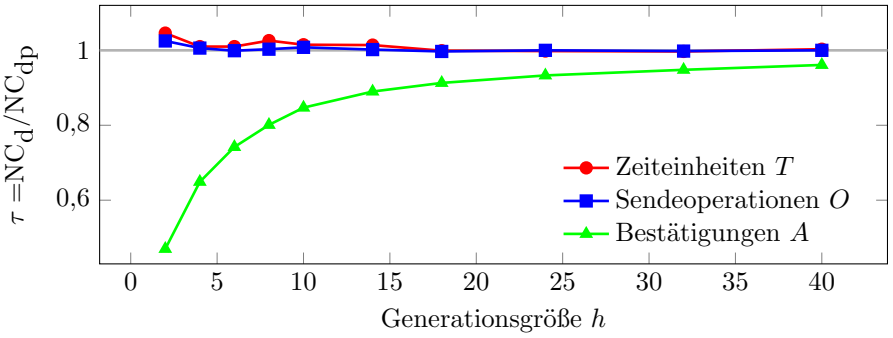
deutlich mehr Nachrichten gesendet werden müssen. Der Grund liegt an der Schätzung. Je höher die Generationsgröße h ist, umso mehr Nachrichten müssen zu Beginn geschickt werden und umso mehr Bestätigungen erhält der Sender, um δ_{sr} genauer abschätzen zu können. Deswegen funktioniert das Ganze besser für hohe Generationsgrößen als für kleinere.

Somit erzielt das Verfahren, gemittelt über alle getesteten δ und h, nur eine Verringerung von etwa $25\,\%$ an Zeiteinheiten, was weniger als bei $\mathrm{NC_{ss}}$ oder $\mathrm{NC_s}$ ist. Andererseits werden auch nur $6\,\%$ mehr Sendeoperationen und $3\,\%$ mehr Bestätigungen benötigt, um diese Zeitersparnis zu erreichen, was ebenfalls deutlich weniger Kosten als bei $\mathrm{NC_{ss}}$ oder $\mathrm{NC_s}$ verursacht. Die Gründe für die weniger starke Zeitersparnis liegen wie schon erwähnt an den kleinen Generationsgrößen, welche eine Schätzung sehr schwierig machen.

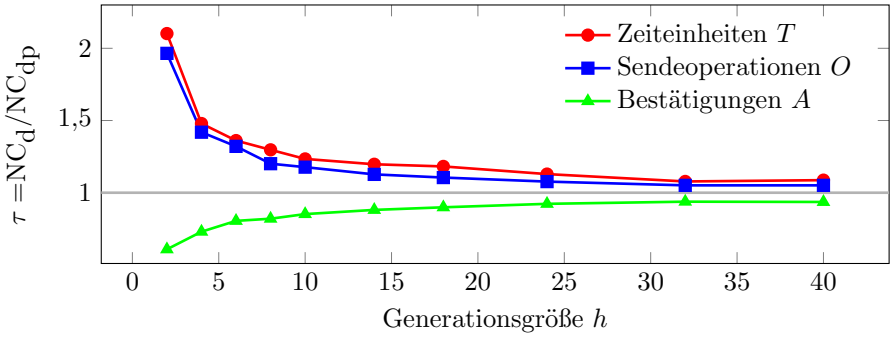
Einschränkend dazu muss gesagt werden, dass in realistischen Anwendungen einerseits ein Startwert (typische Ausfallrate) gesetzt werden könnte, um weitere Zeiteinheiten zu sparen. Andererseits würde man bei einer Übertragung von mehreren Generationen zu Beginn einer neuen Generation δ_{sr} auf die Schätzung von der letzten Generation zurückgreifen. Dadurch können weitere Vorteile erzielt werden und $\mathrm{NC_d}$ könnte eine sehr gute Leistung zeigen.

Da auch für das Schätzen der Ausfallrate 3 Optimierungsvorschläge (permanente Bestätigungen, adaptive Bestätigungen und Gewichtungsfaktor Φ) gemacht wurden, sollen die folgenden Protokolle nicht mit Netzwerkcodierung (NC), sondern mit der Netzwerkcodierung mit dynamischer Redundanz ($\mathrm{NC_d}$) verglichen werden. Dabei werden wiederum die Ergebnisse der Effizienzparameter von $\mathrm{NC_d}$ durch die Ergebnisse der Effizienzparameter der optimierten Protokolle geteilt und das relative Verhältnis τ darge-

stellt. Für $\tau > 1$ ergibt sich eine Verbesserung durch die Optimierung; bei $\tau < 1$ erzielt NC_d bessere Ergebnisse.



(a) Kantenübertragungswahrscheinlichkeit $\delta = 0{,}99$



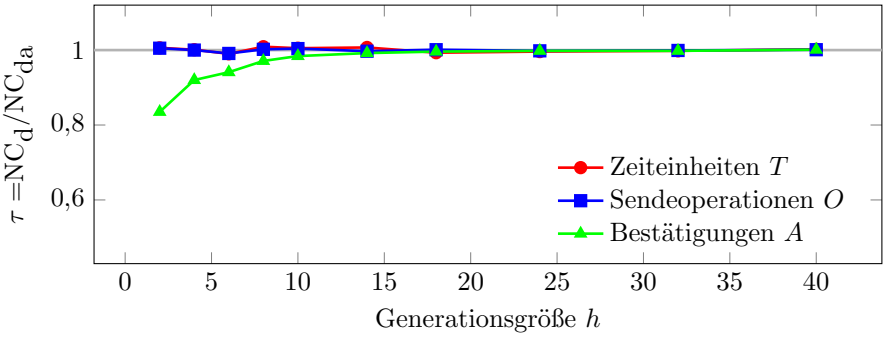
(b) Kantenübertragungswahrscheinlichkeit $\delta = 0{,}80$

Abbildung 4.14: Vergleich von NC_{dp} zu NC_d für zwei verschiedene Kantenübertragungswahrscheinlichkeiten in Abhängigkeit der Generationsgröße h.

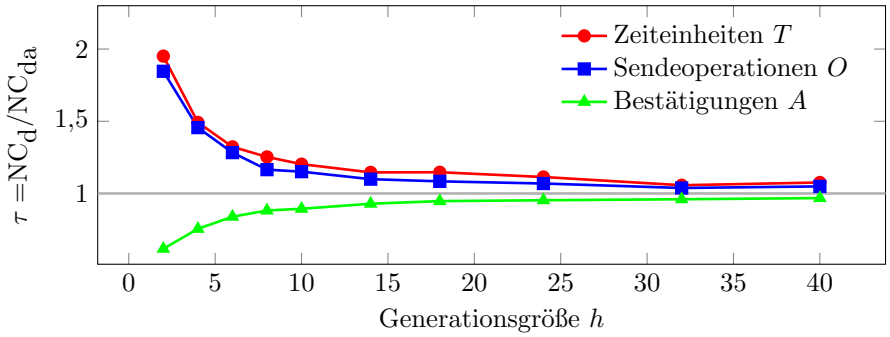
In Abbildung 4.14 werden die Werte für Netzwerkcodierung mit dynamischer Redundanz und permanenten Bestätigungen NC_{dp} relativ zu NC_d gezeigt. Für eine niedrige Ausfallrate (Diagramm oben) gibt es kaum Verbesserungen bezüglich der Zeiten und der Sendeoperationen und wenn nur bei niedriger Generationsgröße. Dafür ist dort die Anzahl der notwendigen Bestätigungen natürlich deutlich höher als ohne die permanenten Bestätigungen. Bei $\delta = 0{,}80$ ist der Mehraufwand für die Bestätigungen auch stärker je niedriger die Generationsgröße ist, jedoch bringt die Optimierung hier bezüglich der notwendigen Sendeoperationen und Zeiteinheiten eine deutliche Verbesserung bei den kleineren Generationsgrößen.

Somit erscheint die Methode durchaus sinnvoll, sofern eine gewisse Menge an Paketen bzw. Bestätigungen ausfällt. Betrachtet man jedoch auch für das Senden der Bestätigungen die vorher angesprochene Möglichkeit, bei weiteren Generationen auf den Schätzwert der vorherigen Generationen zurückzugreifen, sähe das Ergebnis leicht anders aus. Man würde dann keine so starke Steigerung für $h \leq 12$ erzielen und nur eine leichte Zeitersparnis und weniger Sendeoperationen auf Kosten von deutlich mehr Bestätigungen machen. Aber auch dies kann je nach Anwendungsfall sinnvoll sein.

Die Werte für Netzwerkcodierung mit dynamischer Redundanz und adaptiven Bestätigungen NC_{da} in Relation zu NC_d sind in Abbildung 4.15 dargestellt. Beim Vergleich mit Abbildung 4.14 lassen sich die Veränderungen durch die Abkehr von permanenten zu adaptiven Bestätigungen sehen. Während der Unterschied für $\delta = 0{,}80$ nur sehr gering ausfällt, ist dieser bei $\delta = 0{,}99$ deutlich. Dies liegt daran, dass



(a) Kantenübertragungswahrscheinlichkeit $\delta = 0{,}99$



(b) Kantenübertragungswahrscheinlichkeit $\delta = 0{,}80$

Abbildung 4.15: Vergleich von $\mathrm{NC_{da}}$ zu $\mathrm{NC_d}$ für zwei verschiedene Kantenübertragungswahrscheinlichkeiten in Abhängigkeit der Generationsgröße h.

bei $\delta = 0{,}80$ die Schätzung der Ausfallrate recht hoch ist und das dazu führt, dass auch bei $\mathrm{NC_{da}}$ sehr viele Bestätigungen nach dem Empfang des letzten Pakets gesendet werden.

Bei niedrig geschätzter Ausfallrate ($\delta = 0{,}99$) wird hingegen die Anzahl der Bestätigungen gegenüber $\mathrm{NC_{dp}}$ reduziert. Gerade für kleinere Generationsgrößen reduziert sich der Mehraufwand für die zusätzlichen Bestätigungen deutlich. Für größere Generationen $h \geq 10$ ist kaum ein Mehraufwand für die Bestätigungen zu sehen. Allerdings gibt es, wie bei $\mathrm{NC_{dp}}$, bei $\delta = 0{,}99$ keinerlei Verbesserungen hinsichtlich der Zeiteinheiten T oder der Sendeoperationen O.

Insgesamt betrachtet korrigiert also $\mathrm{NC_{da}}$ nur den Mehrverbrauch von Bestätigungen, der bei $\mathrm{NC_{dp}}$ auftritt, wenn diese aufgrund der kleinen Angriffsrate gar nicht benötigt werden. Ob eine der beiden verbesserten Varianten von $\mathrm{NC_d}$ wirklich notwendig ist, ist vom Anwendungsfall abhängig, jedoch ist der Gewinn – gemessen an Aufwand und Kosten für zusätzliche Bestätigungen – relativ gering. Von daher scheint die Möglichkeit sinnvoll, $\mathrm{NC_d}$ nicht bezüglich der Bestätigungsrate zu optimieren, sondern die Redundanz mit einem Gewichtungsfaktor Φ zu bestimmen. Im Folgenden sollen deswegen die Bestätigungen A nicht weiter betrachtet werden. Bei dem Vergleich von dynamischer Netzwerkcodierung mit Gewichtungsfaktor Φ ($\mathrm{NC_\Phi}$) mit $\mathrm{NC_d}$ soll hauptsächlich auf das Verhältnis von Sendeoperationen O zu Zeiteinheiten T geschaut werden.

Abbildung 4.16 zeigt die Abhängigkeit des Verhältnisses von Sendeoperationen O zu Zeiteinheiten T von $\mathrm{NC_\Phi}$ relativ zu $\mathrm{NC_d}$ für unterschiedliche Werte von Φ sowie verschiedene Generationsgrößen h. Grundlage für die Darstellung ist eine Kantenübertragungswahrscheinlichkeit von $\delta = 0{,}80$.

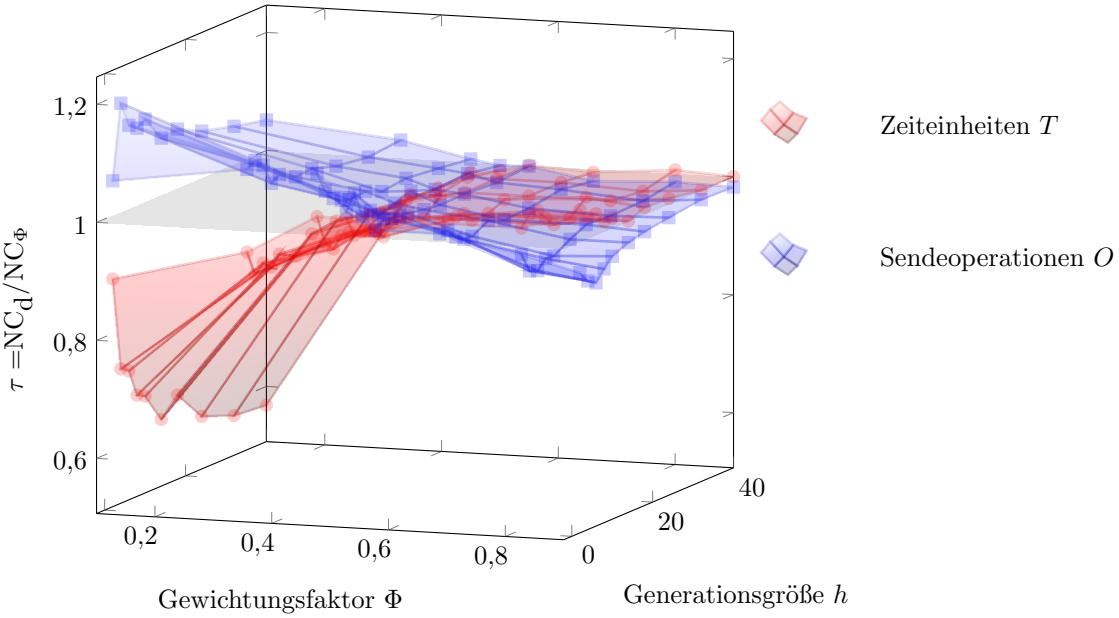


Abbildung 4.16: Vergleich von $\mathrm{NC_\Phi}$ zu $\mathrm{NC_d}$ in Abhängigkeit vom Gewichtungsfaktor Φ und der Generationsgröße h für die Kantenübertragungswahrscheinlichkeit $\delta = 0{,}80$.

Als Erstes ist festzuhalten, dass die Generationsgröße relativ wenig Einfluss auf das Verhältnis von Zeiteinheiten T zu Sendeoperationen O hat. Zwar schwächt sich der Vorteil auf die Zeit und der Nachteil auf die Sendeoperationen mit größer werdendem h bei gleichem Φ ab, jedoch liegt dies hauptsächlich an der schlechteren Schätzung bei kleinen Generationsgrößen. Weiterhin kann man das Verhalten ausgleichen, indem Φ je nach Bedarf verändert wird.

Bei der Wahl eines recht kleinen Φ werden wie erwartet einige Sendeoperationen O gespart und dafür mehr Zeiteinheiten T benötigt. Würde man im Übrigen $\Phi = 0$ setzen, so würde sich $\mathrm{NC_\Phi}$ wie die Variante NC verhalten und nur die unbedingt notwendigen Sendeoperationen durchführen, was natürlich zu einer höheren Verzögerungszeit führen würde. Für große Φ ist der Kommunikationsaufwand leicht gegenüber $\mathrm{NC_d}$ erhöht, allerdings werden dafür noch einige Zeiteinheiten eingespart. Somit lässt sich mit dem Verfahren das notwendige Verhältnis stufenlos einstellen.

Interessant ist auch der mittlere Bereich. Unter der Annahme, dass Sendeoperationen und Zeiteinheiten in etwa gleich wichtig sind, kann man die Summe aus beiden Werte bilden. Diese Summe über alle h gemittelten Werte im Bereich $0{,}33 \leq \Phi \leq 0{,}55$ ist niedriger als bei $\mathrm{NC_d}$, was einer leichten Einsparung der Sendeoperationen und auch der Zeiteinheiten in dem Bereich entspricht.

Die Simulationsergebnisse zeigen, dass mit verschiedenen Möglichkeiten die Übertragungsgeschwindigkeit bei Angriffen bei gleichzeitig moderater Erhöhung des Kommunikationsaufwands verbessert werden kann. Insbesondere das leicht in der Praxis umzusetzende $\mathrm{NC_d}$ bietet eine gute Basis. Sollte eine Anwendung individuellere Verhältnisse von Zeit und Netzwerklast benötigen, so steht mit $\mathrm{NC_\Phi}$ ein Werkzeug zur Verfügung, die Wechselbeziehung zwischen Zeit und Kommunikation optimal und stufenlos einzustellen.

4.6 Zusammenfassung

In diesem Kapitel konnte mithilfe von Simulationen gezeigt werden, dass integritätssichernde Verfahren, welche auf Kryptographie basieren, Vorteile gegenüber den auf Codierung basierenden Verfahren haben. Dabei wurde weniger auf die speziellen Verfahren, sondern vielmehr auf die generellen Unterschiede eingegangen.
Weiterhin konnte gezeigt werden, dass die ratenlose Übertragung von Paketen entscheidende Vorteile gegenüber vordefinierter Redundanz hat. Zum einen kann sich das ganze System durch die Rückmeldung des Empfängers an den Angreifer anpassen, sodass dieser weder als zu stark (hohe Kosten) noch als zu schwach (Verfügbarkeit eingeschränkt) eingeschätzt wird. Zum anderen ist es so einfacher möglich, die Generationsgröße unabhängig vom Netzwerk zu wählen. Das ist besonders wichtig, falls nur beschränktes oder gar kein Wissen über das Netzwerk besteht. Zusätzlich wurde gezeigt, dass – je nach Ausprägung des kryptographischen Verfahrens – die Größe der Generation einen entscheidenden Einfluss auf die Effizienz hat.
Auch die Rolle der Übermittlungsprotokolle wurde genauer untersucht. Aufbauend auf den grundlegenden Lösungen wurden Möglichkeiten zur Steigerung der Effizienz aufgezeigt und entsprechende Protokolle beschrieben. In der anschließenden Evaluation wurde die Wirksamkeit dieser Vorschläge untersucht. Dabei zeigte sich, dass eine Schätzung der aktuellen Übertragungsrate durch den Sender dazu führt, dass durch zusätzlich gesendete redundante Pakete eine Zeitersparnis ermöglicht wird, wobei sich der Mehraufwand für die zusätzliche Kommunikation in Grenzen hält. Ferner wurde auch das Verfahren NC_{Φ}, welches in der Wechselbeziehung zwischen hoher Geschwindigkeit und wenigen Paketübertragungen eine stufenlose Adaption möglich macht, entwickelt und untersucht.
Auch die Entwicklung des Simulators bietet viele Möglichkeiten für zukünftige Arbeiten, die Netzwerkcodierung, Sicherheit und Protokolle zu untersuchen und eigene Entwicklungen zu evaluieren. Dies geschah z. B. bereits bei der Untersuchung, ob sichere Netzwerkcodierung in Hochleistungsrechnern Vorteile bringt, welche in „Analysis of Parallel Applications on a High Performance-Low Energy Computer“ [CIF+14] veröffentlicht wurde. Auch fanden Ergebnisse des Simulators Eingang in HAEC-SIM [BCF+15], einem parallelen ereignisorientierten Simulator. Die in HAEC-SIM integrierten Kommunikationsmodelle basieren auf den Untersuchungen in diesem Kapitel und wurden auch mithilfe der in diesem Kapitel beschriebenen, auf SageMath basierenden Simulationsumgebung verifiziert und in „Simulation Models Verification for Resilient Communication on a Highly Adaptive Energy-Efficient Computer“ [PFC+16] publiziert.
Mit diesem und dem vorherigen Kapitel zusammen wurden grundlegende Aspekte des Einsatzes von Netzwerkcodierungsverfahren zur Sicherung von Integrität und Verfügbarkeit beleuchtet. Daher soll sich im nächsten Kapitel dem Schutz der Vertraulichkeit gewidmet werden, damit alle Hauptschutzziele der Informationssicherheit betrachtet werden.

5 Effiziente vertrauliche Netzwerkcodierung

5.1 Einführung

5.1.1 Grundlagen

Im folgenden Kapitel soll es um den Schutz der Vertraulichkeit bei der Netzwerkcodierung gehen. Dabei werden hauptsächlich die Inhalte der Arbeiten „eSPOC: enhanced Secure Practical Network Coding for Better Efficiency and Lower Latency“ [PF16] sowie „Security Aspects of Confidential Network Coding“ [PF17] verwendet.
Die Absicherung der Vertraulichkeit ist, neben der Absicherung der Integrität und Verfügbarkeit, die Hauptaufgabe der Informationssicherheit. Nur wenn Daten vor dem Zugriff Dritter geschützt werden können, ist eine Benutzung von Netzwerkcodierung in realen Anwendungen denkbar. Im Gegensatz zur Integritätsabsicherung, bei der die Netzwerkcodierung stärkere Anforderungen an das System stellt als bei herkömmlicher Übertragung, erhält man durch die Benutzung von Netzwerkcodierung einen gewissen Basisschutz, die algebraische Sicherheit.
Da diese allein, wie bereits in verschiedenen Arbeiten (z. B. [LMB07, CY07]) aufgeführt, nur für sehr beschränkte Angreifer als ausreichend zu betrachten ist, wird sich dieses Kapitel zum größten Teil mit Verfahren auseinander setzen, die eine höhere Sicherheit bieten, zugleich aber auch nur einen sehr geringen Mehraufwand bedeuten. Als Vertreter dieser leichtgewichtigen vertraulichen Verfahren wurden für die Untersuchungen SPOC [VLB08] und P-Coding [ZJL$^+$10] ausgewählt, da diese Verfahren einen niedrigeren Berechnungsaufwand als beispielsweise homomorphe Verschlüsselungsverfahren aufweisen sollten.
Aufbauend auf diesen Verfahren und der Effizienz gegenüber Ende-zu-Ende-Verschlüsselung wurde ein eigenes Verfahren namens „enhanced Secure Practical Network Coding“ (eSPOC) entwickelt, welches eine höhere Effizienz bietet. Neben der verbesserten Effizienz sollen auch die Sicherheitsaspekte von eSPOC und den leichtgewichtigen Verfahren im Allgemeinen im Focus stehen. Weiterhin beschäftigt sich das Kapitel mit der Anwendbarkeit der Verfahren für verteilte Speicher (Clouds) im Internet.

5.1.2 Fragestellungen

Im Folgenden seien die zu behandelnden Fragestellungen dieses Kapitels zusammengefasst:

- Welche Effizienzsteigerungen können leichtgewichtige Verfahren zur Vertraulichkeitsabsicherung gegenüber herkömmlicher Ende-zu-Ende-Verschlüsselung bringen?
- Ist das entwickelte Verfahren eSPOC bei gleicher Sicherheit effizienter als bisherige Verfahren?
- Wie sicher sind leichtgewichtige Verfahren und welche Angriffsmöglichkeiten gibt es?
- Ist eSPOC im Bereich verteilter Speicher anwendbar und wenn ja, welche Vorteile bietet diese Kombination?

Dazu sollen im Folgenden das Systemmodell und die Effizienzparameter eingeführt werden. Danach wird die Methodik erklärt und der Ablauf der Untersuchungen erläutert. Im Anschluss wird, ausgehend von den bisherigen Verfahren, auf das eSPOC-Verfahren sowie die verbesserten Eigenschaften eingegangen. Das Verfahren wird danach in den Ergebnissen bezüglich der Messungen bewertet. Anschließend folgt eine Diskussion bezüglich der Sicherheit von eSPOC sowie aller betrachteten leichtgewichtigen Verfahren. Nach den theoretischen Betrachtungen zur Anwendbarkeit und den Vorteilen im Bereich der Cloudspeicher folgt eine kurze Zusammenfassung aller Ergebnisse.

5.2 Systemmodell

5.2.1 System

Für die Vergleichbarkeit der Verfahren untereinander wird eine simple Unicast-Kommunikation über einen Pfad von einem Sender `s` zu einem Empfänger `r` betrachtet. Dabei wird genau eine Generation $\boldsymbol{X}$ an Daten übertragen. Obwohl bei allen Verfahren das Recodieren durchgeführt werden kann, wird auf eine Betrachtung der Aufwendungen dafür verzichtet. Dies hat folgende Gründe: Die Änderungen des Protokolls betreffen bei den Verfahren zum Schutz der Vertraulichkeit nur den Sender, der eine Art Verschlüsselung umsetzen muss, und den Empfänger, der die empfangenen Daten wieder entschlüsseln muss. Das Recodieren an den Zwischenknoten bzw. die Aufwendungen für die Übertragung sind unabhängig von dem jeweiligen Verfahren und werden deswegen nicht für den Vergleich mit einberechnet.
Für den Angreifer gilt, dass er nur an den Inhalten der Pakete interessiert ist, potenziell jedes übertragene Paket abhören kann, sich aber sonst protokollgerecht verhält. Verändernde Angriffe sind nicht Teil dieses Kapitels und verursachen unabhängig von dem jeweiligen Verfahren den gleichen Schaden. Bezüglich des Schlüssels zwischen Sender und Empfänger wird angenommen, dass dieser bereits vor Beginn vorliegt. Ein Austauschprotokoll hätte im Übrigen keinen Einfluss auf die unterschiedlichen Verfahren, da jedes Verfahren auf einen gemeinsamen Schlüssel angewiesen ist. Um die Vergleichbarkeit zu gewährleisten, basieren alle Verschlüsselungsprimitive auf dem AES-Standard [DBN+01] mit gleichen Parametern.
Abbildung 5.1 zeigt das Modell für die herkömmliche Art der Ende-zu-Ende-Verschlüsselung (EncPay) vor der eigentlichen Netzwerkcodierung. Die Daten $\boldsymbol{X}$ werden zuerst verschlüsselt und dann mit einer Codierungsmatrix (CM) $\boldsymbol{A}$ codiert. Da es im Netzwerk zum Recodieren kommen kann, wird am Empfänger die (recodierte) Codierungsmatrix $\boldsymbol{A'}$ und die entsprechenden (re)codierten und verschlüsselten Daten $\boldsymbol{A'} * \mathrm{enc}(\boldsymbol{X})$ empfangen. Diese müssen vom Empfänger decodiert und anschließend entschlüsselt werden. Somit erhält der Empfänger Zugriff auf die Daten $\boldsymbol{X}$, ohne dass ein Angreifer, der alle Übertragungen hätte abfangen und somit ebenfalls decodieren könnte, die Daten entschlüsseln könnte, da er den Schlüssel nicht besitzt.

5.2.2 Effizienzparameter

Im Folgenden werden die Effizienzparameter für den Vergleich zwischen Ende-zu-Ende-Verschlüsselung (EncPay), den leichtgewichtigen Verfahren (P-Coding, SPOC) sowie dem selbst entwickelten Verfahren (eSPOC) erläutert. Dabei soll es zunächst nur um die messbaren Effizienzparameter bezüglich der Leis-

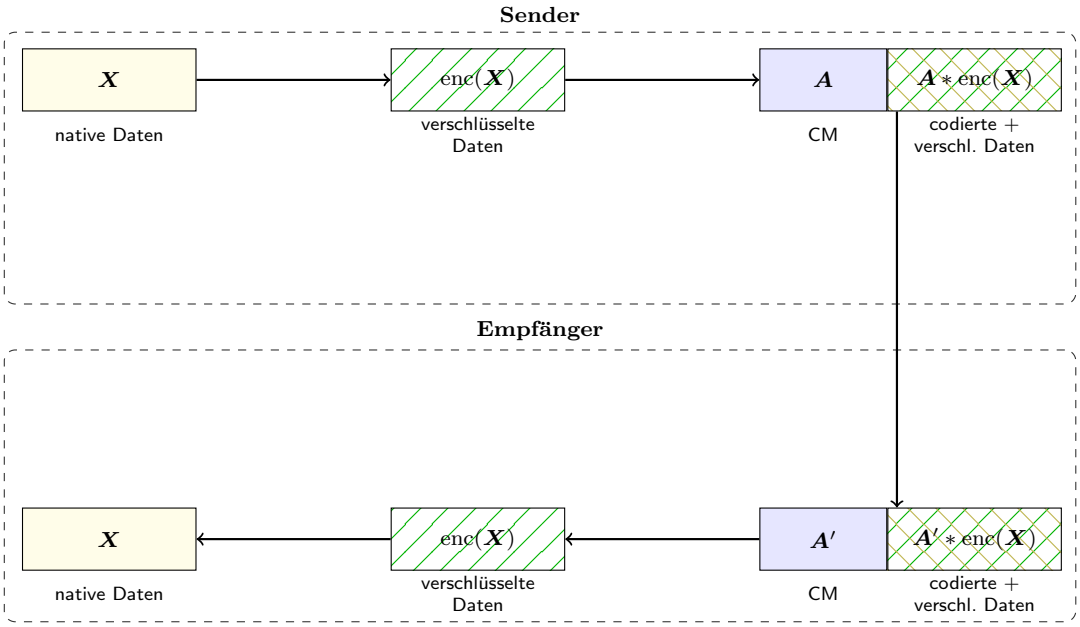


Abbildung 5.1: Modell für die Übertragung von Ende-zu-Ende verschlüsselten Daten (EncPay). Das Modell betrachtet nur die Bearbeitung notwendiger Schritte am Sender und am Empfänger.

tungen gehen. Für eine Sicherheitsanalyse wird in Abschnitt 5.5 der jeweilige Aufwand eines Angreifers untersucht und es werden Aspekte diskutiert, die für eine Umsetzung wichtig sein könnten.

Eine einfach zu bestimmende Effizienzgröße ist die des Mehraufwands gegenüber herkömmlicher Paketvermittlung für die Kommunikation bei Nutzung eines der Vertraulichkeit gewährenden Verfahrens. Dazu wird ausgehend von den zu übertragenden Paketen ermittelt, wie viele Nutzdaten enthalten sind. Da im Folgenden von dem zugrunde liegenden Körper $\mathbb{F}_{2^8}$ ausgegangen wird, sind Angaben in Symbolen und Byte äquivalent. Die Größe der zu übertragenden Daten ist gleich der Paketgröße w. Die für eine Übertragung immer benötigten Headerdaten für darunterliegende Protokolle, wie z. B. IP oder TCP, werden dabei, wie bei der Paketgröße w, ignoriert, da diese für alle Verfahren identisch sind. In den Messungen wurde das UDP-Protokoll verwendet, weswegen die Größe für die UDP-Header bzw. IP-Header keine Rolle spielt. Auch der Speicherbedarf für die Angabe der Nummer der Generations-ID wird vernachlässigt. Bei der Netzwerkcodierung werden allerdings immer ein paar Bytes für den Codierungsvektor belegt, die dann nicht mehr als Nutzdatensymbole n genutzt werden können. Dadurch ergibt sich ein gewisser Mehraufwand e, der wie folgt ermittelt wird:

$$e = \frac{w - n}{n}. \tag{5.1}$$

Der Wertebereich für den Mehraufwand liegt also in den positiven rationalen Zahlen $e \in \mathbb{Q}^+$. Je näher e an 0 ist, umso geringer ist der Mehraufwand. Ein Wert von $e = 1$ bedeutet beispielsweise, dass nur die Hälfte der Paketdaten wirkliche Nutzdaten sind.

Die zweite Effizienzgröße ist die Zeit für die Ausführung aller notwendigen Berechnungen (Berechnungsaufwand f). Da nur Sender und Empfänger betrachtet werden, wird die Zeit für die Ausführung aller Funktionen, vom Verschlüsseln und Codieren über das Versenden hin zum Decodieren und Entschlüsseln, bis zu dem Zeitpunkt, an dem die Nutzdaten komplett wieder vorliegen, berücksichtigt. Der

Berechnungsaufwand f ist zwar abhängig vom verwendeten System, sollte aber bei der Nutzung gleicher Primitive und Funktionen vergleichbare Ergebnisse auf verschiedenen Plattformen erzeugen. Im Speziellen ist damit die Hardware-Unterstützung für kryptographische Operationen gemeint, wobei darauf geachtet wird, dass alle Verfahren auf den gleichen kryptographischen Verfahren basieren. Da einige Funktionen parallel berechenbar sind, spiegelt der Berechnungsaufwand nicht unbedingt den zeitlichen Aufwand für die gesamte Ausführung wider. Vielmehr entspricht der Berechnungsaufwand und damit die Gesamtausführungszeit aller Operationen dem letztendlichen Energieverbrauch, der somit – zumindest grob – abgeschätzt werden kann.
Da die Funktionen der einzelnen Verfahren erst in einem späteren Abschnitt erläutert werden, soll hier erwähnt werden, dass nur die zusätzlichen Funktionen gemessen werden. Das ohnehin notwendige Schreiben der Daten in Pakete oder die für die Übertragung über das Netzwerk relevanten Funktionen werden also für die Auswertung ignoriert. Bei der Auswertung des Berechnungsaufwands f sind kleinere Werte besser als größere. Um nicht den fälschlichen Bezug zur Zeit, sondern vielmehr zur Energie zu haben, werden die Ausführungszeiten der einzelnen Funktionen mit der Leistung des Systems multipliziert.
Natürlich ist aber auch die Zeit eine relevante Größe beim Thema Effizienz. Hier ist jedoch die Übertragung als solches der begrenzende Faktor. Das heißt, je nach Länge der Übertragung spielen die zusätzlichen Berechnungen keine Rolle, da diese parallel während der Übertragung der nächsten Daten berechnet werden können. Man kann jedoch diese Übertragungszeit komplett ignorieren und sich die Latenzen, also die Verzögerungen, die durch die Verfahren entstehen, anschauen. Diese Latenz ist wichtig bei kurzen Übertragungen und für Anwendungen, die niedrige Verzögerungszeiten benötigen. Die Idee dazu ist, die Zeit t_{pre} beim Sender zu messen, die vom Zugriff auf die Daten bis zum Versenden des ersten Pakets vergeht. Analog dazu wird die Zeit t_{post} beim Empfänger gemessen, die vom Empfang des letzten Pakets bis zu dem Zeitpunkt, bei dem alle Daten in entschlüsselter und decodierter Form vorliegen, vergeht. Der betrachtete Effizienzparameter t_{lat} ist die Summe aus beiden Zeiten:

$$t_{\text{lat}} = t_{\text{pre}} + t_{\text{post}}. \tag{5.2}$$

Auch hier ergibt sich ein rein positiver Wertebereich. Je kleiner der Wert für t_{lat} ist, umso kürzer ist die Verzögerung. Weiterhin gibt es jeweils 2 unterschiedliche Werte für die Zeiten, welche sich auf die Bearbeitung einer gesamten Generation oder auf die Bearbeitung eines Pakets beziehen. Für eine bessere Unterscheidung der Werte wird für beispielsweise t_{pre} im Folgenden t_{pre_G} für die generationsbasierte Berechnung bzw. t_{pre_P} für die paketweise Berechnung verwendet. Die Notwendigkeit für die genaue Unterscheidung ergab sich bei der Auswertung der Ergebnisse und wird dort näher behandelt.

5.3 Methodik

Für den Vergleich der Effizienz der Ansätze und zum Nachweis der Leistungsfähigkeit des selbst entwickelten Verfahrens eSPOC wird auf eine Messung auf einem Testsystem zurückgegriffen. Zwar lässt sich der Mehraufwand für die Übertragungen im Netzwerk auch leicht analytisch ermitteln, dies gilt allerdings nicht für den Berechnungsaufwand oder die Latenzen. Zwar ließen sich die Anzahl und Art der Rechenoperationen abschätzen, jedoch käme es auch dabei auf eine konkrete Umsetzung an und eine praktische Vergleichbarkeit wäre trotzdem nicht gegeben. Die Ausführungszeiten für eine Verschlüs-

selung ist nicht mit der einer Codierung vergleichbar. Deswegen ist eine Simulation der Zeiten nicht möglich und eine Messung hier eine sinnvolle Wahl.
Dazu erfolgte die Implementierung der Verfahren. Um eine möglichst hohe Leistung zu erreichen, wurde dabei auf die Softwarebibliothek Kodo [PHF11] zurückgegriffen, da alle Funktionen und Primitive bezüglich der Geschwindigkeit optimiert wurden. Für die Berechnung der kryptographischen Primitive wurde die Bibliothek Crypto++ [Cry16] mit der AES-Verschlüsselung und 128 Bit Schlüssel als Grundlage gewählt. Alle Verfahren wurden in C++ implementiert.
Als Testsystem wurde der HAEC Playground (genauer beschrieben in [MPB+17]) gewählt. Dabei handelt es sich um ein Cluster bestehend aus mehreren Odroid XU-4 Einplatinenrechnern [RB15]. Jeder Odroid XU-4 Rechner besitzt 2 GB DDR3 Arbeitsspeicher, Gigabit-Ethernet, eine 16 GB eMMC als Speichermedium und eine ARM big.LITTLE CPU, bestehend aus 4 leistungsfähigen Cortex A-15 Kernen zu 2 000 MHz und 4 sparsamen Cortex A-7 Kernen zu 1 400 MHz. Die einzelnen Knoten (Sender, Zwischenknoten, Empfänger) waren für die Messungen auf unterschiedliche Einplatinenrechner verteilt. Da die Leistung eines aktiven A15 Kerns auf etwa 1 W geschätzt wird, soll für die Energieabschätzung die Ausführungszeit mit 1 W multipliziert werden.
Die Messungen wurden für verschiedene Szenarien mit unterschiedlicher Paket- und Generationsgröße und für die unterschiedlichen Verfahren durchgeführt. Dabei wurde jede Einstellung 1 000 Mal ausgeführt, um etwaige Messungenauigkeiten auszugleichen. Als Körper wurde durchgehend $\mathbb{F}_{2^8}$ verwendet. Insgesamt wurden 5 verschiedene Verfahren getestet:

EncPay Ende-zu-Ende-Verschlüsselung der Daten (Abbildung 5.1)

P-Coding Permutieren der Daten [ZJL+10] (Abbildung 5.3)

SPOC Secure Practical Network Coding [VLB08] (Abbildung 5.2)

eSPOC Eigenes Verfahren [PF16] aufbauend auf SPOC (Abbildung 5.6)

PNC nicht abgesichertes Practical Network Coding [CWJ03] als Messbasis

Dabei wurden 3 verschiedene Parametereinstellungen aus Generationsgröße h sowie Paketgröße w für die Messungen ausgewählt. Für die Nachrichtenlänge pro Paket n gilt $w = n + 2h$ für SPOC bzw. $w = n + h$ für alle übrigen Verfahren. Die 3 Szenarien sind:

S1: $h = 16$ und $w = 1\,360$ (Gängige Bedingungen)

S2: $h = 16$ und $w = 500$ (Kleinere Paketlänge)

S3: $h = 64$ und $w = 1\,408$ (Höhere Generationsgröße)

Die Bedingungen in S1 stellen in der Literatur gängige Parameter da. Die Paketgröße w ist so gewählt, dass die Daten mit einigen Zusatzinformationen, wie z. B. der Generations-ID, in ein gewöhnliches IP-Paket passen. Zudem ist sowohl die Paketgröße als auch die Generationsgröße und damit auch die Anzahl der Codierungskoeffizienten ein Vielfaches von 16. Dies hat den Vorteil, dass bei einer Verschlüsselung mit AES nur vollständige Blöcke zu verschlüsseln sind, da die Blocklänge 128 Bit beträgt.
Die beiden anderen Bedingungen sind Varianten, um den Einfluss der Paketgröße und der Generationsgröße zu untersuchen. Bei S2 wird einfach eine kleinere Paketgröße angenommen. Die kann z. B.

in Netzen mit einer kleineren maximalen Übertragungseinheit (Maximum Transmission Unit - MTU) vorkommen. Bei S3 wurde im Gegensatz zu S1 die Generationsgröße vervierfacht. Größere Generationen erhöhen im Allgemeinen die Latenz und den Berechnungsaufwand, verringern aber dafür die Netzwerklast etwas. Dass in diesem Fall die Paketgröße w leicht steigt, liegt daran, dass die Nutzdaten bei S1 und S3 mit $n = 1344$ konstant gehalten wurden und die Codierungskoeffizienten mehr Platz benötigen. Die anschließenden Sicherheitsbetrachtungen sind hauptsächlich analytisch. Der vorgestellte Angriff bei bekanntem Klartext basiert auf einer Simulation, welche in SageMath [The17] implementiert wurde. Die Betrachtungen bezüglich der Anwendbarkeit und der Vorteile für verteilte Cloud-Speicher sind ebenfalls theoretisch oder basieren auf Simulationen.

5.4 eSPOC

5.4.1 Leichtgewichtige Verfahren

Neben dem unsicheren Basisverfahren (PNC), welches nur die Daten codiert, bevor sie übertragen werden, wurde auch schon die herkömmliche Ende-zu-Ende-Verschlüsselung (EncPay) erläutert. Wie im Kapitel zum Stand der Technik dargelegt, gibt es auch leichtgewichtige Verfahren, die den inhärenten Basisschutz der Netzwerkcodierung ausnutzen. Die zwei Verfahren SPOC [VLB08] und P-Coding [ZJL+10] sollen im folgenden Abschnitt genauer erklärt werden.

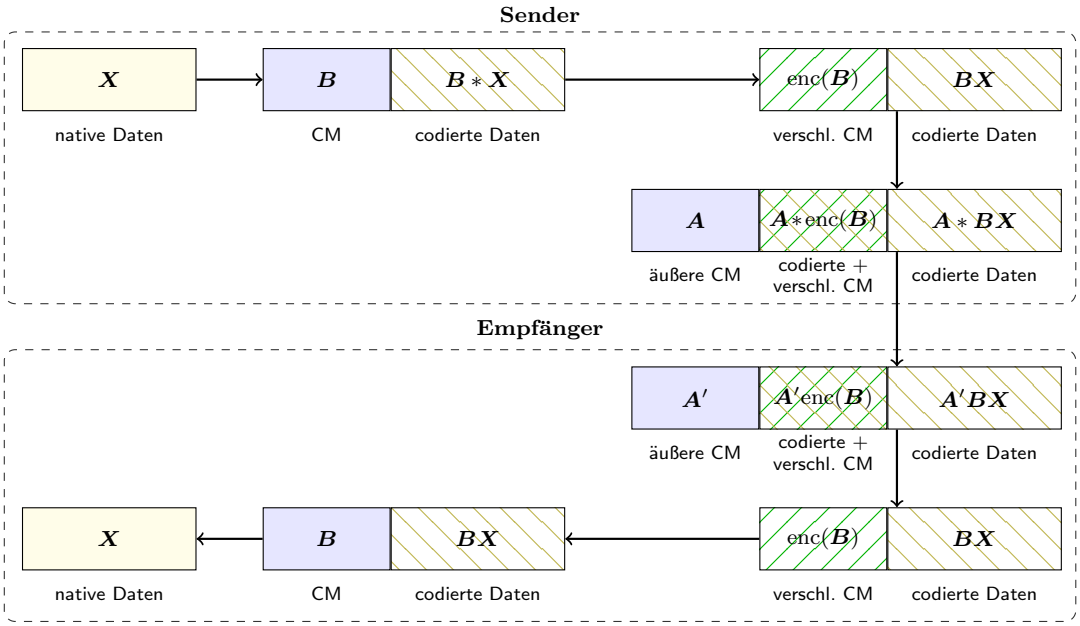


Abbildung 5.2: Modell für die Übertragung mit SPOC analog zu Abbildung 5.1. Bei SPOC wird eine zusätzliche äußere Codierungsmatrix (CM) verwendet, was zu einem Übertragungsmehraufwand führt. Dafür müssen die Daten nicht verschlüsselt werden.

Abbildung 5.2 zeigt das Verfahren SPOC im relevanten Bereich des Senders s und des Empfängers r. Die Daten $\boldsymbol{X}$ werden, wie bei der Netzwerkcodierung üblich, mit einer globalen Codierungsmatrix $\boldsymbol{B}$ codiert. Um nun die Codierungsmatrix vor Abhörern zu schützen, wird diese verschlüsselt. Da die Verschlüsselung allerdings nicht homomorph bezüglich der Matrixmultiplikationen ist, muss eine zusätzliche äußere Codierungsmatrix $\boldsymbol{A}$ verwendet werden. Diese Matrix, im einfachsten Fall die Einheitsmatrix

$\boldsymbol{I}$, wird nun mit der inneren Codierungsmatrix und den Daten zusammen multipliziert. Das Ergebnis, ein Paket mit innerer und äußerer Codierungmatrix sowie den codierten Daten, wird dann übertragen.

Auf der Empfängerseite kommen nun etwas vergrößerte Pakete an, wenn nicht vorher die Nutzdaten pro Paket etwas verkleinert wurden. Diese werden wie folgt verarbeitet: Zuerst muss die äußere Codierung rückgängig gemacht werden. Dazu wird $\boldsymbol{A}'$ mittels Gauß-Jordan-Algorithmus invertiert und mittels Multiplikation damit die innere verschlüsselte Codierungsmatrix und die zugehörigen codierten Daten wiederhergestellt. Anschließend muss diese verschlüsselte innere Codierungsmatrix entschlüsselt werden, um dann erneut (hier aber mit $\boldsymbol{B}^{-1}$) multiplizieren zu können, damit die Daten $\boldsymbol{X}$ decodiert werden.

Wie man sieht, sind bei SPOC im Gegensatz zu EncPay mehr Schritte notwendig, um eine gesicherte Übertragung zu bewerkstelligen. Weiterhin ist der Anteil der Nutzdaten durch die zusätzliche Codierungsmatrix geringer als bei EncPay. Trotzdem wird erwartet, dass das Verfahren zumindest für realistische Generations- sowie Paketgrößen schneller und effizienter als EncPay arbeitet, da hier nur die Codierungsmatrix $\boldsymbol{B}$ und nicht die Daten $\boldsymbol{X}$ verschlüsselt und entschlüsselt werden müssen. Die Größe der Codierungsmatrix ist, in Abhängigkeit von Generations- sowie Paketgröße, normalerweise deutlich geringer als die der Daten. Somit sollte der Aufwand für das doppelte Codieren und Decodieren durch die deutlich verringerten Verschlüsselungsoperationen mehr als ausgeglichen worden sein.

Die Sicherheit von SPOC basiert auf der Tatsache, dass die inneren Codierungsinformationen verschlüsselt sind und somit ein Angreifer, der genügend Pakete abhören kann, nur die äußere Codierung rückgängig machen kann. Um die innere Codierung zu lösen, müsste er entweder den Schlüssel raten, das kryptographische Verfahren brechen oder die Matrix $\boldsymbol{B}$ erraten. Je nach Generationsgröße und Schlüssel gilt dies für einen beschränkten Angreifer als ausgeschlossen. Das Konzept basiert auf einer Untersuchung, in der gezeigt wurde, dass es ausreicht, als Geheimnis, die Codierungsmatrix zu schützen [LVBM08], wenn die Daten redundanzfrei sind, z. B. durch eine vorherige Kompression.

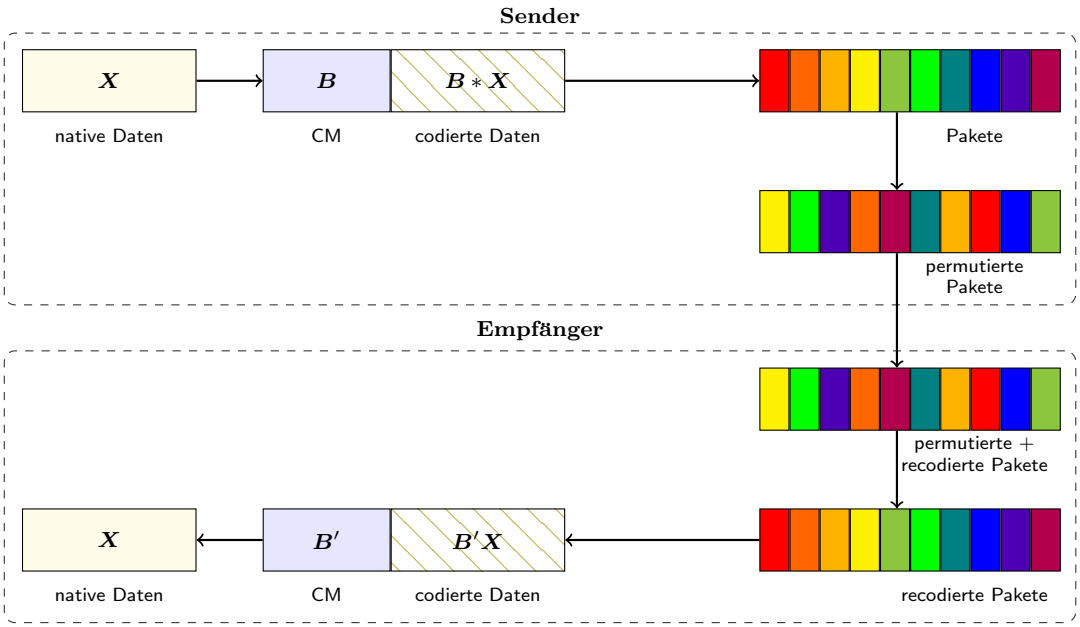


Abbildung 5.3: Modell für die Übertragung mit P-Coding. Anstatt zu verschlüsseln, werden Codierungsmatrix (CM) und Daten zusammen permutiert. Da diese Operation homomorph zu dem Recodieren ist, wird keine zusätzliche Codierungsmatrix benötigt.

Das alternative leichtgewichtige Verfahren P-Coding ist in Abbildung 5.3 dargestellt. Nach dem Codieren der Daten $\boldsymbol{X}$, d. h. der Multiplikation der Matrix $\boldsymbol{B}$ mit der Matrix $\boldsymbol{X}$, werden alle Pakete einer Generation in der gleichen Art und Weise permutiert. Dabei werden Codierungskoeffizienten und Daten zusammen und nicht getrennt voneinander permutiert. Die Permutation wird auf die einzelnen Symbole des Pakets, also auf die Elemente des zugrunde liegenden Körpers, angewendet. Die Permutation muss bei P-Coding zufällig, aber auch am Empfänger reproduzierbar sein. Deshalb eignet sich ein Pseudozufallszahlengenerator auf Basis einer kryptographischen Funktion für die Erzeugung des Permutationsvektors.

Bei der Übertragung kann ebenfalls ohne Einschränkung recodiert werden, da die Permutation homomorph zu den Matrizenmultiplikationen ist. Dadurch kann der Empfänger auch zuerst die inverse Permutation auf die empfangenen Pakete anwenden und anschließend mittels $\boldsymbol{B}^{-1}$ decodieren.

Bei Anwendung von P-Coding ergibt sich kein erhöhter Mehraufwand bei der Kommunikation relativ zu EncPay oder PNC. Weiterhin ist die Permutation einfacher als eine Verschlüsselung auszuführen. Zusätzlich gibt es keine doppelten Operationen oder sonstigen Mehraufwand. Deswegen wird erwartet, dass auch P-Coding bessere Ergebnisse als EncPay erreicht. Im Vergleich zu SPOC muss allerdings das komplette Paket mit in die Permutation einfließen, was bei langen Paketen ebenfalls aufwendig sein könnte. Bei einem Vergleich von P-Coding und SPOC wird es auf die Paketgröße und Generationsgröße ankommen. Größere Pakete sollten mit SPOC effizienter übermittelt werden, da der Berechnungsaufwand im Gegensatz zu P-Coding nur leicht steigen sollte. Bei hohen Generationsgrößen wiederum wird erwartet, dass P-Coding vorteilhafter als SPOC ist.

Die Sicherheit des Verfahrens basiert auf der Permutation. Für einen Angreifer ist zum einen nicht ersichtlich, welche Symbole ursprüngliche Codierungsinformationen oder Daten darstellen. Aber selbst wenn dies dem Angreifer bekannt wäre und er alle Codierungsinformationen in der richtigen Reihenfolge hätte, könnte er zwar die permutierten Nutzdaten decodieren, aber diese wären durch eine Art Permutationschiffre geschützt. Ein Angriff auf diese Verschlüsselung ist dabei nicht Erfolg versprechend, da es sich bei den Daten um vorher komprimierte Daten handelt. Die einzige Möglichkeit für einen Angreifer ist, in solch einem Fall den Schlüssel für die Permutation zu raten, die kryptographische Funktion zu brechen oder die richtige Reihenfolge der Daten zu erraten. Es wird angenommen, dass dies für einen beschränkten Angreifer nicht möglich ist.

Um eine möglichst hohe Vergleichbarkeit der Verfahren untereinander zu gewährleisten, wurde für die Implementierung aller Verfahren dieselbe kryptographische Funktion, genauer eine Verschlüsselung mit AES und einem 128 Bit Schlüssel, gewählt, um Daten sowie Codierungsvektoren zu verschlüsseln bzw. die geheime Permutation zu erzeugen. Dies gilt auch für das im folgenden Abschnitt vorgestellte, selbst entwickelte Verfahren. Dieses wurde entwickelt, um einige Nachteile von SPOC, z. B. den doppelten Platzbedarf für den Header, zu beseitigen.

5.4.2 Eigenes Verfahren

Aufbauend auf dem Verfahren SPOC wurde ein eigenes Verfahren namens eSPOC entwickelt, welches zum Ziel hat, bei gleichbleibender Sicherheit die Effizienz zu verbessern. Der folgende Abschnitt soll die Entwicklung der einzelnen Ideen darstellen und erläutern, warum eSPOC eine bessere Effizienz und Leistung in den Messungen zeigen sollte.

Als erstes besteht das Problem der zwei Codierungsmatrizen bei SPOC, was zu einem erhöhten Mehraufwand für die Kommunikation führt. Da die äußere Matrix zwingend für den Transport und damit für das Recodieren benötigt wird, ist der einzige Ansatzpunkt für eine Einsparung also die innere Matrix. Diese beinhaltet die verschlüsselte Version von $\boldsymbol{B}$. Da $\boldsymbol{B}$ sowieso nur zufällig auf Senderseite erstellt und verschlüsselt wurde, ist die Idee, ganz auf die Übertragung zu verzichten und stattdessen $\boldsymbol{B}$ mithilfe einer pseudozufälligen Funktion auf beiden Seiten mit dem Schlüssel zu generieren. Damit entfällt im Vergleich zu SPOC der Kommunikationsmehraufwand.

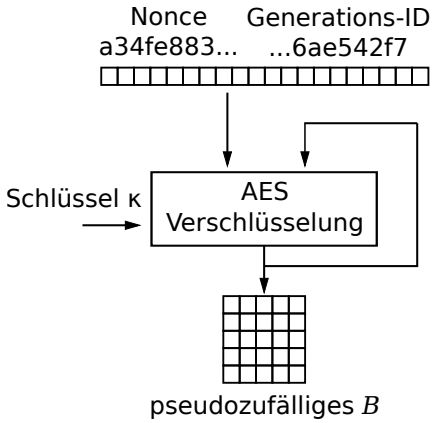


Abbildung 5.4: Ein Pseudozufallsgenerator ist eine Möglichkeit, um $\boldsymbol{B}$ auf beiden Seiten zu generieren. Zuerst wird der Initialisierungsvektor mit dem AES Schlüssel κ verschlüsselt. Das Ergebnis ist Eingang der nächsten Runde. Somit lässt sich bei Sender und Empfänger das gleiche $\boldsymbol{B}$ generieren.

Eine Möglichkeit, die in Abbildung 5.4 dargestellt ist, ist dabei, AES im Output-Feedback-Mode zu betreiben und die Generations-ID sowie eine Nonce (welche öffentlich bekannt sein kann) als Initialisierungsvektor zu verwenden. Dadurch lässt sich auf beiden Seiten ein Bitstrom erzeugen, aus dem $\boldsymbol{B}$ abgeleitet werden kann. Natürlich darf die Kombination aus Generations-ID und Nonce nicht für unterschiedliche Generationen wiederholt werden. Sollte ein so erzeugtes $\boldsymbol{B}$ nicht vollen Rang haben, so muss es verworfen werden und mit einer anderen Generations-ID neu erzeugt werden. Ein weiterer Vorteil bei dieser Verfahrensweise ist, dass $\boldsymbol{B}$ sofort nach dem Empfang der ersten Nachricht einer Generation beim Empfänger erzeugt werden kann. So muss nicht erst auf das letzte Paket gewartet werden, um die äußere Codierungsmatrix zu decodieren, bevor $\boldsymbol{B}$ entschlüsselt werden kann. Dadurch lässt sich die Verzögerungszeit nach Empfang des letzten Pakets reduzieren.

Nachdem nun also nur noch die äußere Codierungsmatrix und die codierten Daten übermittelt werden, kann auch der Berechnungsaufwand minimiert werden. Anstatt am Ende zu decodieren, dann zu entschlüsseln und wieder zu decodieren, kann man, da $\boldsymbol{B}$ ja bekannt ist, die äußere Codierungsmatrix $\boldsymbol{A}$ mit $\boldsymbol{B}$ multiplizieren und muss dann nur einmal mit dem Inversen $(\boldsymbol{AB})^{-1}$ decodieren.

In Abbildung 5.5 wird der gesamte geänderte Ablauf von eSPOC (b) im Vergleich zu SPOC (a) nochmal in Matrixschreibweise aufgezeigt. Anstatt $\boldsymbol{B}$ zufällig zu erzeugen und dann zu verschlüsseln, wird diese Matrix mithilfe eines pseudozufälligen Zahlengenerators erzeugt und dann nicht dem Paket hinzugefügt. Auf Empfängerseite wird ebenfalls $\boldsymbol{B}$ erzeugt, anstatt die Matrix zu entschlüsseln. Anstatt $\boldsymbol{A}$ und später $\boldsymbol{B}$ aufwendig zu invertieren und mit $\boldsymbol{ABX}$ bzw. $\boldsymbol{BX}$ zweimal zu multiplizieren, wird $\boldsymbol{A}$ mit $\boldsymbol{B}$ multipliziert, das Inverse $(\boldsymbol{AB})^{-1}$ gebildet und nur einmal mit $\boldsymbol{ABX}$ multipliziert.

$[\boldsymbol{X}]$
1 $[\mathbf{1}|\boldsymbol{X}]\,|*\boldsymbol{B} \rightarrow [\boldsymbol{B}|\boldsymbol{BX}]$
2 $|\text{enc} \rightarrow [\text{enc}(\boldsymbol{B})|\boldsymbol{BX}]$
3 $[\mathbf{1}|\text{enc}(\boldsymbol{B})|\boldsymbol{BX}]$
4 $|*\boldsymbol{A} \rightarrow [\boldsymbol{A}|\boldsymbol{A}*\text{enc}(\boldsymbol{B})|\boldsymbol{ABX}]$
5 $|*\boldsymbol{A}^{-1} \rightarrow [\mathbf{1}|\text{enc}(\boldsymbol{B})|\boldsymbol{BX}]$
6 $|*\text{dec} \rightarrow [\boldsymbol{B}|\boldsymbol{BX}]$
7 $|*\boldsymbol{B}^{-1} \rightarrow [\mathbf{1}|\boldsymbol{X}]$
$[\boldsymbol{X}]$

(a) SPOC

$[\boldsymbol{X}]$
1 $[\boldsymbol{X}]\,|*\boldsymbol{B} \rightarrow [\boldsymbol{BX}]$
2 $[\mathbf{1}|\boldsymbol{BX}]$
3 $|*\boldsymbol{A} \rightarrow [\boldsymbol{A}|\boldsymbol{ABX}]$
4 $|*\boldsymbol{B} \rightarrow [\boldsymbol{AB}|\boldsymbol{ABX}]$
5 $|*(\boldsymbol{AB})^{-1} \rightarrow [\mathbf{1}|\boldsymbol{X}]$
$[\boldsymbol{X}]$

(b) eSPOC

Abbildung 5.5: Ablauf bei SPOC und eSPOC. Bei eSPOC entfallen gegenüber SPOC die Ver- und Entschlüsselung von $\boldsymbol{B}$ sowie die aufwendige Invertierung von $\boldsymbol{A}$. Dafür wird bei eSPOC auf Empfängerseite das empfangene $\boldsymbol{A}$ mit dem generierten $\boldsymbol{B}$ multipliziert.

Weiterhin können die Verzögerungszeiten reduziert werden, da nicht zwangsläufig alle Berechnungen auf Matrizen angewendet werden müssen. Um die Verzögerungszeiten beim Empfänger nach Empfang des letzten Pakets so gering wie möglich zu halten, ist es möglich, bei jedem empfangenen Paket sofort den Codierungsvektor mit $\boldsymbol{B}$ zu multiplizieren. Das Ganze sieht für Paket i einer Generation wie folgt aus:

$$[\boldsymbol{\alpha}_i|\boldsymbol{\alpha}_i\boldsymbol{BX}]\,|\boldsymbol{\alpha}_i*\boldsymbol{B} \rightarrow [\boldsymbol{\alpha}_i\boldsymbol{B}|\boldsymbol{\alpha}_i\boldsymbol{BX}]\,. \tag{5.3}$$

Der Aufwand für eine Multiplikation von Matrix $\boldsymbol{A}$ mit Matrix $\boldsymbol{B}$ ist zwar identisch mit dem Aufwand für alle Multiplikation von Vektoren $\boldsymbol{\alpha}_i$ mit Matrix $\boldsymbol{B}$, aber diese Vorgehensweise ermöglicht einige Vorteile, da ein Paket der Form $[\boldsymbol{\alpha}_i\boldsymbol{B}|\boldsymbol{\alpha}_i\boldsymbol{BX}]$ partiell invertiert werden kann. Das führt dazu, dass zum Zeitpunkt des Empfangs des letzten Pakets nur noch wenig Aufwand für das Invertieren von $\boldsymbol{AB}$ notwendig ist, da nur noch eine Zeile und eine Spalte bearbeitet werden muss. Danach kann $(\boldsymbol{AB})^{-1}$ mit den codierten Daten $\boldsymbol{ABX}$ multipliziert werden und die kompletten Daten der Generation stehen zur Verfügung. Dadurch wird die zeitliche Verzögerung auf ein Minimum reduziert.
Wollte man diese Optimierung im Übrigen auf SPOC anwenden, könnte man das partielle Invertieren nur auf die äußere Codierungsmatrix anwenden. Das heißt, nach Empfang des letzten Pakets müsste nach wie vor neben dem Bearbeiten der letzten Zeile und Spalte für die Invertierung von $\boldsymbol{A}$ und dem Multiplizieren mit $\boldsymbol{ABX}$ dann das komplette enc($\boldsymbol{B}$) entschlüsselt, anschließend die komplette Matrix $\boldsymbol{B}$ invertiert und dann nochmals mit den Daten $\boldsymbol{BX}$ multipliziert werden. Hier sollte sich also bei den Messungen ein deutlicher Zeitunterschied ergeben.
Abbildung 5.6 stellt zusammenfassend die notwendigen Schritte bei eSPOC analog zu Abbildungen 5.2 und 5.3 dar. Die Matrix $\boldsymbol{B}$ wird auf beiden Seiten pseudozufällig erzeugt und muss daher nicht übertragen werden. Die Anzahl der Berechnungsschritte wurde gegenüber SPOC deutlich reduziert.

5.4.3 Ergebnisse

Mehraufwand für die Kommunikation Der Kommunikationsmehraufwand e wird allein durch die einzelnen Abbildungen der Verfahren ersichtlich. Tabelle 5.1 legt die numerischen Ergebnisse in den unterschiedlichen Szenarien dar. Wie erwartet, hat nur SPOC einen etwa doppelt so hohen Mehraufwand, da

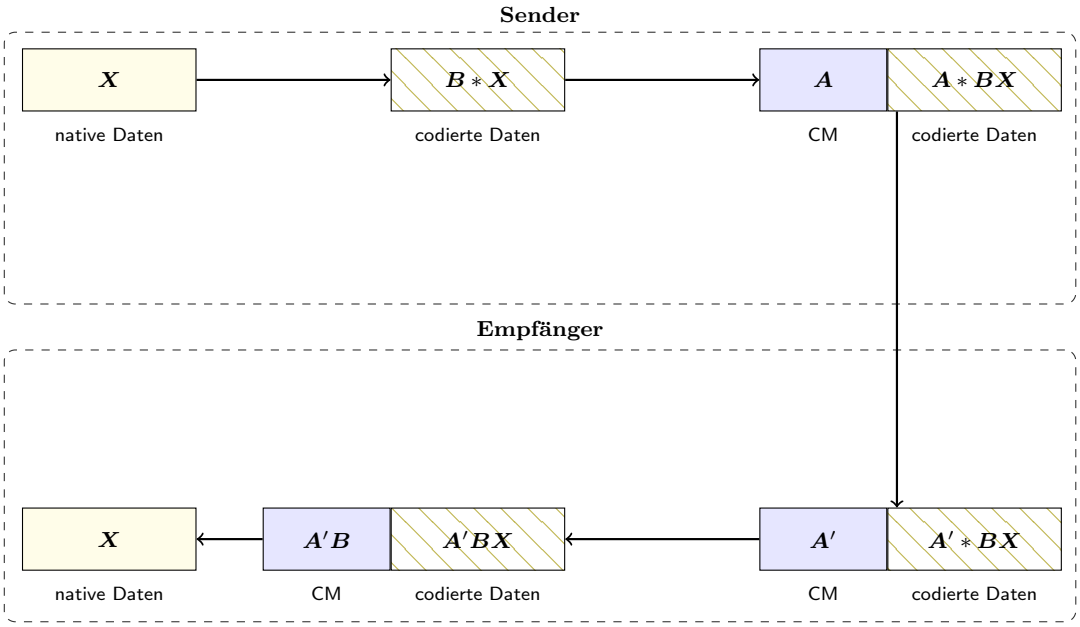


Abbildung 5.6: Modell für die Übertragung mit eSPOC. Das benötigte $\boldsymbol{B}$ wird mithilfe einer pseudozufälligen Funktion und eines Schlüssels auf beiden Seiten berechnet und muss somit nicht übertragen werden.

Tabelle 5.1: Mehraufwand e für die Kommunikation für die 3 verschiedenen Szenarien.

Mehraufwand e	S1	S2	S3
SPOC	2,41 %	6,84 %	10,00 %
EncPay / P-Coding / eSPOC / PNC	1,19 %	3,31 %	4,76 %

2 Codierungsmatrizen übermittelt werden müssen und somit pro Paket weniger Platz für Nutzdaten übrig bleiben. Alle anderen Verfahren liegen gleich mit der Messbasis für die ungesicherte Netzwerkcodierung (PNC) und zeigen bei dieser Effizienzgröße keinerlei Mehrkosten, die durch Sicherheit entstehen.

Weiterhin wird der Einfluss der Paketgröße w und der Generationsgröße h bei den unterschiedlichen Szenarien ersichtlich. Je größer die Pakete sind und je geringer die Generationsgröße ist, umso weniger Mehraufwand e für die Kommunikation ist notwendig.

Berechnungsaufwand Bei den folgenden Betrachtungen sei noch einmal darauf hingewiesen, dass zwischen Energieverbrauch, Ausführungszeit und Berechnungsaufwand zwar kein linearer Zusammenhang besteht, trotzdem aber Tendenzen und grobe Abschätzungen möglich sind, sodass die hier gemessenen Werte alle auf den Ausführungszeiten auf einem A-15 Core basieren.

Abbildung 5.7 zeigt die Aufwendungen für alle Operationen f für das Szenario S1, welches übliche Bedingungen repräsentiert. Alle Operationen auf Senderseite sind ungemustert, die auf Empfängerseite gemustert. Alle Angaben entsprechen der Energie, die gebraucht wird, um genau eine Generation der Größe h zu übertragen, unter der Annahme, dass die Berechnungen 1 W Leistung benötigen. Die Fehlerbalken stehen für die Standardabweichungen der Summe über alle Funktionen bei Sender und Empfänger. Zusätzlich sei erwähnt, dass bei P-Coding aus Implementierungsgründen zuerst permutiert

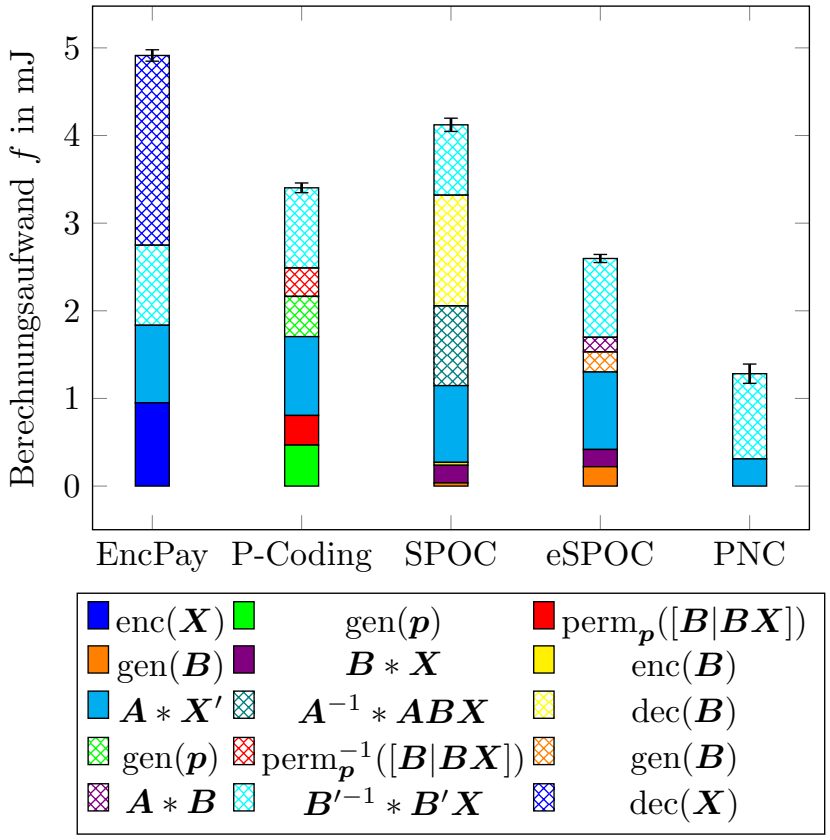


Abbildung 5.7: Berechnungsaufwand f nach einzelnen Funktionen für S1: $h = 16$ und $w = 1\,360$.

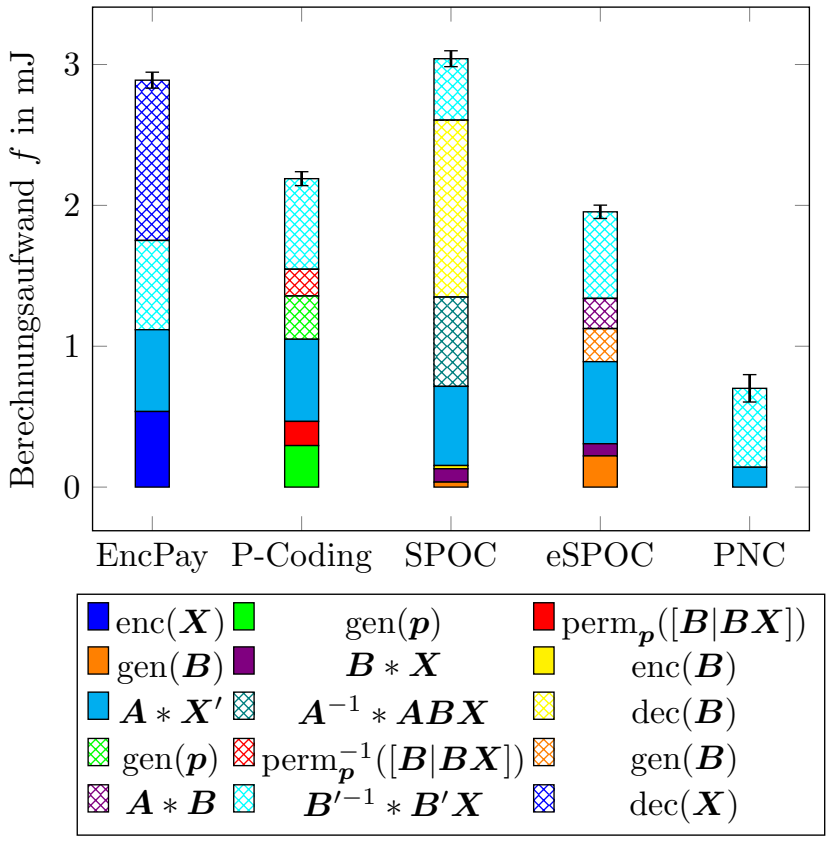


Abbildung 5.8: Berechnungsaufwand f nach einzelnen Funktionen für S2: $h = 16$ und $w = 500$.

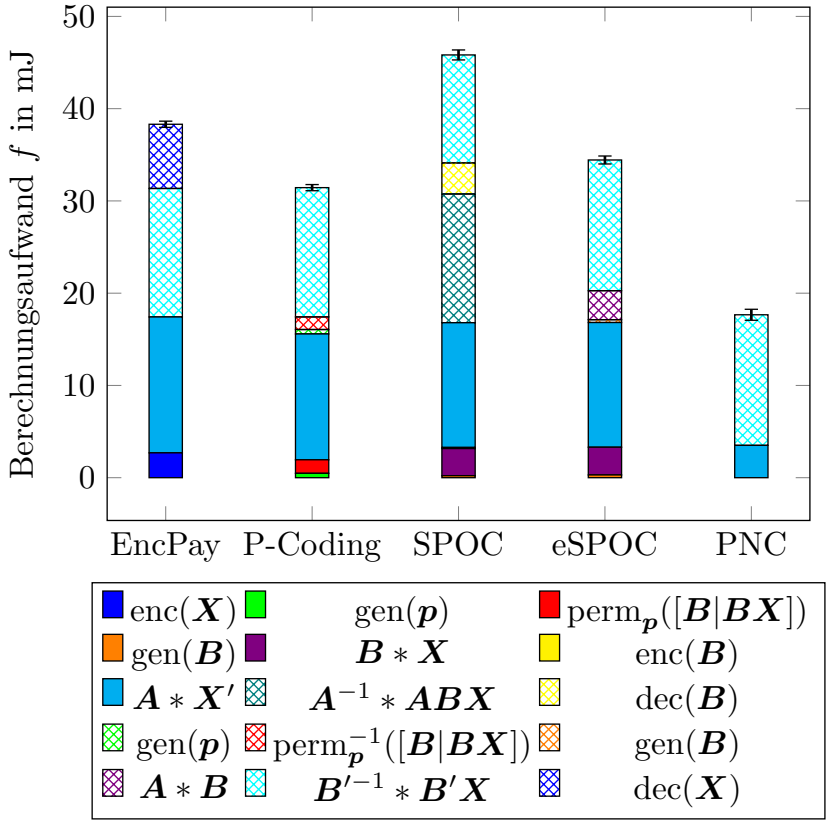


Abbildung 5.9: Berechnungsaufwand f nach einzelnen Funktionen für S3: $h = 64$ und $w = 1\,408$.

und anschließend codiert wurde, was aber keinen Unterschied macht, da sich die Funktionen homomorph zueinander verhalten.

Wie erwartet, ist der Aufwand f für das nicht abgesicherte PNC am geringsten und für das herkömmliche Verfahren EncPay durch die Ent- und Verschlüsselung am größten. SPOC verursacht von den 3 leichtgewichtigen Verfahren den größten Aufwand, hat aber auf Senderseite den geringsten Wert für f. Dies liegt an dem aufwendigen Verfahren auf Empfängerseite, welcher decodieren, entschlüsseln und nochmals decodieren muss. P-Coding hat einen geringeren Aufwand als SPOC, da das Permutieren auf beiden Seiten weniger aufwendige Berechnungen als das Ver- und Entschlüsseln verursacht. Am besten von den sicheren Verfahren schneidet eSPOC ab.

In Abbildung 5.8 zeigt sich ein ähnliches Bild für das Szenario, in dem der Einfluss einer kleineren Paketgröße getestet wird (S2). Allerdings liegt der Aufwand f für EncPay nun unter dem für SPOC, da bei kleineren Paketen weniger Daten verschlüsselt werden müssen, während der Aufwand für Codieren/Decodieren und Verschlüsseln/Entschlüsseln der Codierungsmatrix ähnlich hoch bleibt. Auch der Abstand zwischen eSPOC und P-Coding veringert sich, da die Permutationsoperation einfacher wird, die Codierungsmatrix aber gleich groß bleibt. Trotzdem erzielt auch hier eSPOC unter den sicheren Verfahren den geringsten Wert für den Effizienzparameter f .

Wird die Generationsgröße stark erhöht, wie in S3, dann ergibt sich ein Bild, welches in Abbildung 5.9 dargestellt ist. Ähnlich wie in S2 steigt der Aufwand f für SPOC und eSPOC überproportional zu EncPay und P-Coding, die von kurzen Paketen in Relation zur Generationsgröße profitieren. Somit ist P-Coding hier das beste der 4 sicheren Verfahren. Jedoch ist das Szenario für die Praxis ungeeignet, wenn man sich

den Gesamtenergiebedarf anschaut. Für das Übertragen von der 4 fachen Menge an Nutzdaten (relativ zu S1) benötigt man etwa 10 Mal so viel Energie. Die Übertragung von 4 Generationen mit den Parametern von S1 erscheint dann sinnvoller.
Noch anzumerken ist das seltsame Verhalten bei der Matrixmultiplikation mit $\boldsymbol{A}$ bei PNC. Bei allen 3 Szenarien würde man einen ähnlich hohen Balken wie bei den anderen Verfahren erwarten, jedoch ist dies reproduzierbar nicht der Fall. Eine mögliche Erklärung für das Phänomen ist, dass wahrscheinlich zusätzliche Optimierungen innerhalb der Kodo-Bibliothek bei dem einfachen Fall vorgenommen wurden, die bei zusätzlichen Operationen bei den sicheren Verfahren nicht angewendet wurden.

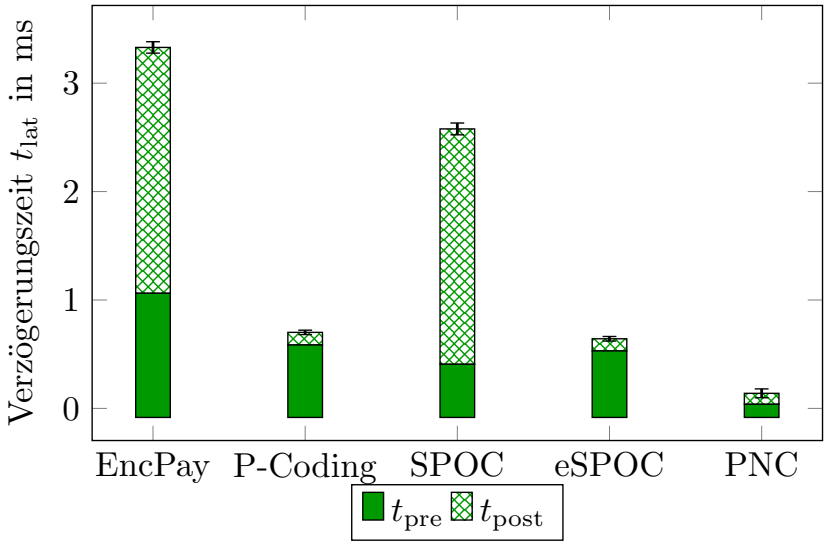


Abbildung 5.10: Verzögerungszeiten für Sender (t_{pre}) und Empfänger (t_{post}) für S1: $h = 16$, $w = 1\,360$.

Verzögerungszeit Abbildung 5.10 zeigt die gemessenen Latenzen für Szenario S1. Ähnlich wie beim Berechnungsaufwand ist das unsichere PNC am schnellsten, EncPay verursacht die größten Verzögerungen t_{lat}. Von den 3 leichtgewichtigen Verfahren ist eSPOC am schnellsten, SPOC am langsamsten. Der Unterschied zwischen beiden Verfahren ist hauptsächlich auf die bereits erwähnten Aufwendungen beim Empfänger zurückzuführen, der vor Erhalt des letzten Pakets nur wenig im Voraus berechnen kann. Abbildung 5.11 zeigt analog dazu die Werte für t_{lat} und Szenario S2. Wie auch beim Berechnungsaufwand zeigt bei diesen Parametern EncPay bessere Eigenschaften als SPOC. P-Coding und eSPOC besitzen deutlich geringere Verzögerungszeiten, wobei P-Coding geringfügig besser abschneidet.
In Abbildung 5.12 für Szenario S3 zeigt sich, dass sowohl SPOC als auch eSPOC einen stärkeren Zuwachs an der Verzögerungszeit t_{lat} als die anderen Verfahren haben. Besonders niedrig bleibt die Verzögerungszeit bei P-Coding. Die Ursache, weshalb eSPOC hier deutlich höhere Verzögerungen als P-Coding aufweist, liegt an der Vorverarbeitung am Sender s.
In der Implementierung der Verfahren wurde zwar auf die schnelle Decodierung beim Empfänger Wert gelegt, die Vorverarbeitung wurde jedoch nicht optimiert. Es wurde für die Messung von P-Coding die paketweise Vorverarbeitungszeit t_{pre_P} gemessen, d. h. vor dem Versenden des ersten Pakets wurde nur dieses codiert, permutiert und dann gesendet. Bei eSPOC wurde dazu im Gegensatz die generationsweise Vorverarbeitungszeit t_{pre_G} gemessen, da erst die komplette Generation mit $\boldsymbol{B}$ multipliziert wurde, bevor die erste Nachricht gesendet wurde.

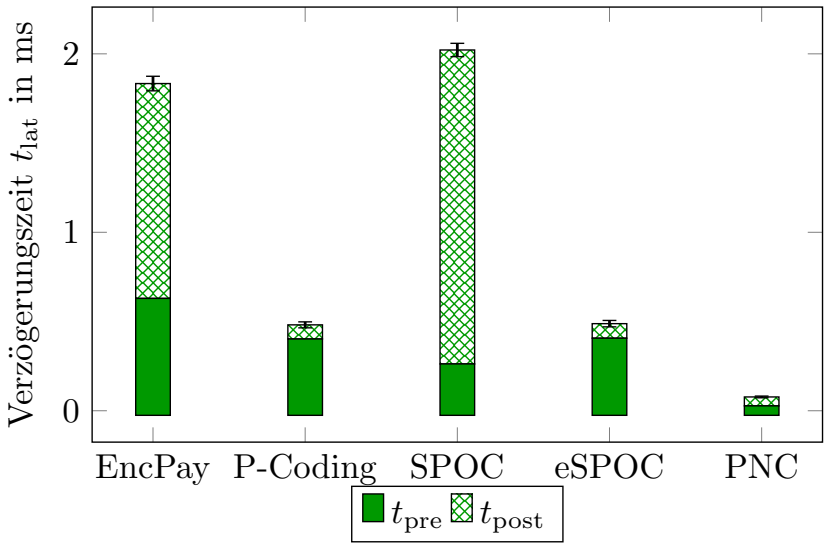


Abbildung 5.11: Verzögerungszeiten für Sender (t_{pre}) und Empfänger (t_{post}) für S2: $h = 16$, $w = 500$.

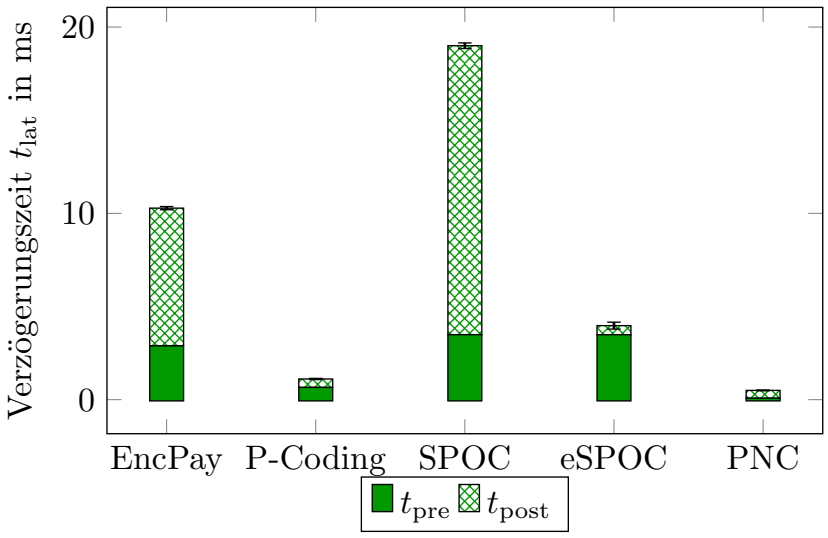


Abbildung 5.12: Verzögerungszeiten für Sender (t_{pre}) und Empfänger (t_{post}) für S3: $h = 64$, $w = 1\,408$.

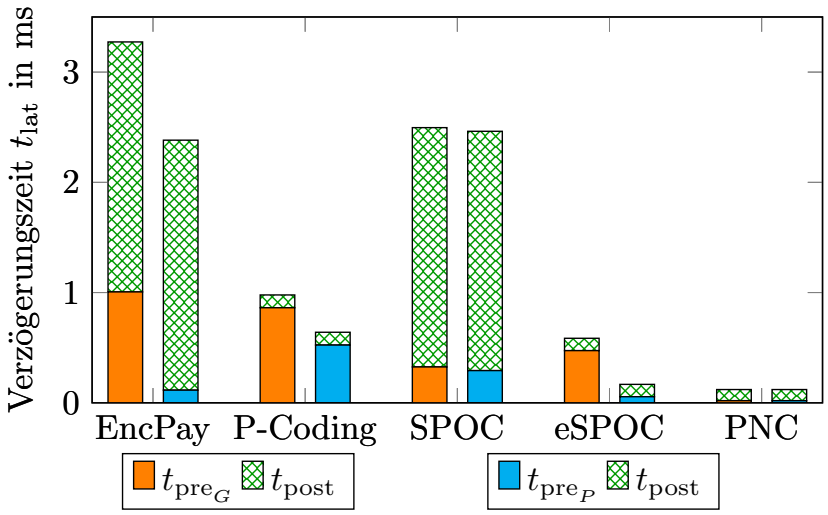


Abbildung 5.13: Differenzierte Verzögerungszeiten für die paketweise Vorverarbeitungszeit t_{pre_P} sowie die generationsweise Vorverarbeitungszeit t_{pre_G} für S1: $h = 16$, $w = 1\,360$.

Wird nun also bei den Zeiten nach t_{pre_P} und t_{pre_G} differenziert, ergeben sich die Werte, die in Abbildung 5.13 für Szenario S1 dargestellt sind. Zuallererst sind nur Unterschiede auf Senderseite zu erkennen, da die Empfängerseite bei allen Verfahren schon auf paketweise Zeiten t_{post_P} optimiert war. Wird nur auf die paketweise Implementierung der Verfahren geachtet, sieht man, dass SPOC und EncPay ähnlich hohe Verzögerungszeiten verursachen. P-Coding hat zwar eine niedrigere Latenz, aber einen deutlichen Abstand zu eSPOC, welches fast so niedrige Zeiten wie PNC erreicht.

Damit konnte gezeigt werden, dass es möglich ist, die durch eSPOC eingebrachten Verzögerungszeiten so zu optimieren, dass diese in Relation zu der ungesicherten Netzwerkcodierung sehr klein sind. Im Vergleich zu den anderen Verfahren hat eSPOC hier einen klaren Vorteil.

Interessant sind vor allem auch die Unterschiede zwischen t_{pre_P} und t_{pre_G} bei den einzelnen Verfahren. Während sich bei EncPay ein relativ großer Unterschied zeigt, ist dieser bei SPOC marginal. Dies liegt daran, dass bei EncPay die Nutzdaten anstatt bei 16 Paketen nur bei dem ersten verschlüsselt werden mussten. Dadurch gibt es eine Verbesserung annähernd um Faktor h. Bei SPOC hingegen muss sowohl bei t_{pre_P} als auch bei t_{pre_G} zuerst einmal die gesamte Generation codiert und anschließend die Codierungsmatrix verschlüsselt werden, bevor dann entweder ein Paket (t_{pre_P}) oder alle Pakete (t_{pre_G}) mit einer äußeren Matrix codiert werden. Dadurch ist die Verbesserung nur sehr gering. Bei P-Coding muss zwar die Permutation für alle Pakete erzeugt werden, aber nur auf eines angewendet werden. Weiterhin ist ein paketweises Codieren möglich. Dadurch werden etwa 30% der Zeit eingespart. Bei eSPOC muss nur $\boldsymbol{B}$ für die Generation erzeugt werden und anschließend kann ein lokaler Vektor $\boldsymbol{\alpha}_i$ mit $\boldsymbol{B}$ multipliziert werden, um dann mit $\boldsymbol{\alpha}_i\boldsymbol{B}$ die Daten eines Pakets zu codieren. Dadurch gibt es eine Verbesserung, die fast so groß wie bei EncPay ist.

Insgesamt lässt sich festhalten, dass die leichtgewichtigen Verfahren, wie SPOC und P-Coding, klare Vorteile gegenüber herkömmlicher Ende-zu-Ende-Verschlüsselung haben. Zusätzlich zeigt sich, dass das selbst entwickelte Verfahren eSPOC bei realistischen Parametern nochmals deutliche Vorteile bringt. Besonders bei den Latenzen zeigt sich nur eine leichte Erhöhung der Verzögerungszeiten gegenüber der nicht abgesicherten Netzwerkcodierung. Neben der Effizienz ist allerdings auch wichtig, wie sicher die

Verfahren nun im Vergleich zur Ende-zu-Ende-Verschlüsselung sind. Darauf soll im nächsten Abschnitt eingegangen werden.

5.5 Sicherheitsanalyse

5.5.1 Algebraische Sicherheit

Bevor auf die spezifischen Sicherheitsverfahren eingegangen wird, soll es in diesem Abschnitt zunächst um die Gründe gehen, warum algebraische Sicherheit nicht ausreicht. Dazu soll im Detail noch einmal die algebraische Sicherheit erklärt werden und dann auf partiell invertierbare Matrizen sowie den Einfluss des zugrunde liegenden Körpers für die mathematischen Operationen eingegangen werden.
Von algebraischer Sicherheit wird bei der Netzwerkcodierung gesprochen, da die Nachrichten nicht im Klartext, sondern codiert übermittelt werden. Sofern keine triviale Codierungsmatrix (Einheitsmatrix) verwendet wird, lassen sich die Klartexte der Nachrichten nur ermitteln, indem die von den empfangenen Nachrichten abzuleitende Codierungsmatrix invertiert wird und die Pakete damit multipliziert werden. Die Grundannahme ist nun, dass bei einer Generation der Größe h genau h nicht linear abhängige codierte Nachrichten empfangen werden müssen, um decodieren zu können.
Betrachtet man das aus Sicht eines rein abhörenden Angreifers, bedeutet dies, dass dieser, wenn er $k < h$ linear unabhängige Nachrichten abhört, nicht an die Inhalte der Nachrichten kommt, da er keine invertierbare Codierungsmatrix besitzt. Sollte er allerdings $k = h$ Nachrichten abgehört haben, so kann er wie der Empfänger decodieren und erhält die Daten $\boldsymbol{X}$. Unabhängig von allen Daten kann es zusätzlich zu sogenannten partiell invertierbaren (auch partiell diagonalisierbaren) Matrizen kommen. Jede $k \times h$ Matrix $\boldsymbol{B'}$ mit linear unabhängigen Zeilen lässt sich zu einer $k \times k$ Einheitsmatrix und einer $k \times (h-k)$ Matrix mit beliebigen Koeffizienten umformen:

$$\boldsymbol{B'} = \left.\begin{bmatrix} 1 & 0 & 0 & \overbrace{\beta_{1,4} & \beta_{1,5}}^{h-k} \\ 0 & 1 & 0 & \beta_{2,4} & \beta_{2,5} \\ 0 & 0 & 1 & \beta_{3,4} & \beta_{3,5} \end{bmatrix}\right\} k. \tag{5.4}$$

In Formel (5.4) ist die resultierende Matrix für ein Beispiel dargestellt. Kommt es nun in einer Zeile dazu, dass die restlichen $h-k$ Koeffizienten zufällig 0 sind, dann ist die Matrix partiell diagonalisierbar und die entsprechende Nachricht könnte im Klartext gelesen werden.
Daraus ergeben sich nun genau 2 Fälle, bei denen die algebraische Sicherheit gegenüber abhörenden Angreifern e nicht ausreicht, da die Vertraulichkeit nicht gewahrt wäre:

1. Der Angreifer e kann h Pakete einer Generation abhören.

2. Der Angreifer e kann die Matrix der empfangenen Codierungsvektoren partiell diagonalisieren.

Während Punkt 1 noch relativ leicht über ein Angreifermodell beschrieben werden kann und auch für die Planung eines Systems theoretisch bestimmbar ist, ist die Wahrscheinlichkeit für eine partiell invertierbare Matrix nicht ohne weiteres bestimmbar. Deshalb soll im Folgenden untersucht werden, wie wahrscheinlich dieser Fall in Abhängigkeit von der Anzahl der abgehörten linear unabhängigen Nachrichten k, der Generationsgröße h und der Körpergröße q eintritt.

Bei Betrachtung der Formel (5.4) lässt sich erkennen, dass die Wahrscheinlichkeit einer 0 an einen der beliebigen Koeffizienten $\beta_{i,j}$ von der Körpergröße abhängt. Unter der Annahme, dass die Koeffizienten zufällig verteilt sind, beträgt die Wahrscheinlichkeit für eine 0 an einer bestimmten Stelle $\frac{1}{q}$. Die Wahrscheinlichkeit, dass dann $h - k$ beliebige Koeffizienten in einer bestimmten Zeile den Wert 0 haben, beträgt $\frac{1}{q^{h-k}}$. Damit kann dann die Wahrscheinlichkeit χ bestimmt werden, dass irgendeine der k Zeilen einer Generation partiell invertierbar ist. Es gilt:

$$\chi = 1 - \left(1 - \frac{1}{q^{h-k}}\right)^{k}. \tag{5.5}$$

Da meist nicht nur eine Generation übertragen wird, sondern eine größere Nachricht oder eine Datei, werden g Generationen übermittelt. Somit lässt sich Formel (5.5) auf Daten beliebiger Größe erweitern. Die Wahrscheinlichkeit χ, dass unter g Generationen der Größe h bei Abhören von k linear unabhängigen Nachrichten pro Generation mindestens eine Zeile aller Matrizen invertierbar ist, beträgt:

$$\chi = 1 - \left(1 - \frac{1}{q^{h-k}}\right)^{k \cdot g}. \tag{5.6}$$

Um die Abhängigkeiten visuell deutlich zu machen, wurden 3 Abbildungen erstellt, die auf der Formel (5.6) basieren und die Zusammenhänge besser darstellen sollen.

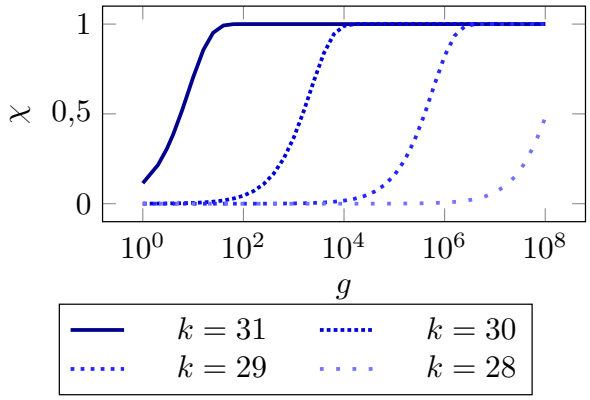


Abbildung 5.14: Wahrscheinlichkeit χ einer partiell decodierbaren Nachricht in Abhängigkeit der Anzahl übertragener Generationen g und der Anzahl abgehörter Nachrichten k pro Generation bei konstanter Generationsgröße $h = 32$ und $q = 256$.

In Abbildung 5.14 wird die Abhängigkeit der partiellen Decodierbarkeit von der Anzahl der Generationen g und der Anzahl abgehörter Nachrichten k dargestellt. Unter der Annahme, dass die Generationsgröße $h = 32$ und die Körpergröße $q = 256$ beträgt, sind für 4 unterschiedliche Werte von k die Wahrscheinlichkeiten für eine partielle Invertierung χ bei g Generationen gezeigt. Man sieht, dass mit linear kleiner werdendem k exponentiell viele Generationen g übertragen werden müssen, um die gleiche Wahrscheinlichkeit χ zu erreichen. Während beispielsweise bei 1 000 Generationen bei $k \geq 30$ die Wahrscheinlichkeit χ nahe bei 1 liegt, ist diese für $k < 30$ fast 0.

In Abbildung 5.15 soll nun die Abhängigkeit von der Anzahl der Generationen g und der Körpergröße q gezeigt werden. Für eine feste Generationsgröße $h = 32$ und eine feste Anzahl linear unabhängiger abgehörter Pakete $k = 28$ wird die Wahrscheinlichkeit χ für 4 verschiedene Körpergrößen in Abhängigkeit der Generationsanzahl g abgebildet. Wie zu erwarten, führen kleinere Körper zu einer erhöhten Wahr-

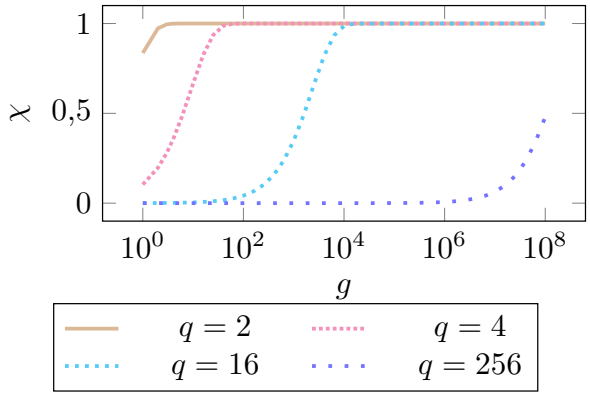


Abbildung 5.15: Wahrscheinlichkeit χ einer partiell decodierbaren Nachricht in Abhängigkeit der Anzahl übertragener Generationen g und der Körpergröße q bei konstanter Generationsgröße $h = 32$ und konstanter Anzahl abgehörter Nachrichten $k = 28$ pro Generation.

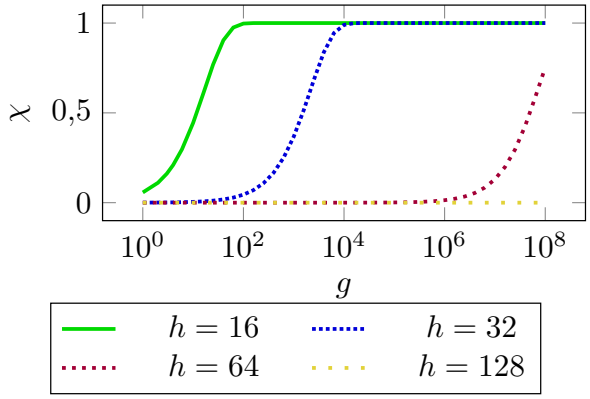


Abbildung 5.16: Wahrscheinlichkeit χ einer partiell decodierbaren Nachricht in Abhängigkeit der Anzahl übertragener Generationen g und der Generationsgröße h bei konstanter Körpergröße $q = 256$ und konstantem Verhältnis abgehörter Nachrichten $k = \frac{15}{16}h$ pro Generation.

scheinlichkeit für die partielle Decodierbarkeit. Der Körper $q = 256$ zeigt die niedrigste Wahrscheinlichkeit von den dargestellten Körpern. Während für den Körper $\mathbb{F}_2$ schon wenige Generationen ausreichen, um nahezu sicher eine Generation decodieren zu können, werden für $\mathbb{F}_{2^2}$ etwa 10 Generationen benötigt, um eine Wahrscheinlichkeit $\chi > 0.5$ zu erreichen. Um bei $\mathbb{F}_{2^4}$ auf diese Wahrscheinlichkeit zu kommen, müssen etwa 1 000, bei $\mathbb{F}_{2^8}$ rund 100 000 000 Generationen übertragen werden. Höhere Körpergrößen führen andererseits auch zu einem größeren Berechnungsaufwand und mehr Zusatzdaten in Relation zu den Nutzdaten.

Abbildung 5.16 zeigt nun noch die Abhängigkeit von der Anzahl der Generationen g und der Generationsgröße h. Bei einem festen Verhältnis von abgehörten Nachrichten $k = \frac{15}{16}h$ und einer festen Körpergröße von $q = 256$ wird für 4 unterschiedliche Generationsgrößen h die Wahrscheinlichkeit χ bei g Generationen dargestellt. Man sieht, dass größere Generationen mehr Sicherheit gegen partielle Diagonalisierbarkeit erreichen. Sollten $\frac{15}{16}$ der Nachrichten einem Angreifer bekannt sein, so benötigt dieser bei $h = 16$ nur etwa 100 Generationen, um nahezu sicher zu sein, dass er mindestens eine Generation decodieren kann. Verdoppelt man die Generationsgröße, sind schon mehr als 10 000 Generationen notwendig. Für $h = 64$ müssten schon einige Terabyte an Daten übertragen werden, um überhaupt eine realistische

Chance zu haben, eine Generation decodieren zu können. Für $h = 128$ sieht es noch schlechter für einen Angriff aus. Allerdings erhöhen große Generationen auch den Anteil von Codierungsdaten zu Nutzdaten und den Berechnungsaufwand.
Ausgehend von Formel (5.6) können auch Formeln zur Berechnung der Körpergröße q bzw. der Generationsgröße h abgeleitet werden, um eine vorgegebene Wahrscheinlichkeit χ zu erreichen:

$$q = \left(1 - (1-\epsilon)^{\frac{1}{k}}\right)^{\frac{1}{k-h}}, \tag{5.7}$$

$$h = k - \frac{\log\left(1 - (1-\epsilon)^{\frac{1}{k}}\right)}{\log(q)}. \tag{5.8}$$

Damit ließen sich nun passende h und q für eigene Anforderungen ermitteln. Jedoch geht eine Erhöhung von q und h immer mit höherem Berechnungsaufwand und höherem Platzbedarf einher. Außerdem muss der Angreifer relativ gut bekannt sein. Wird die Stärke zu hoch angesetzt, wären Berechnungsaufwand und Kommunikationsaufwand nutzlos erhöht. Ist die Schätzung zu niedrig, dann hat der Angreifer die Chance, doch Daten einzusehen und die Vertraulichkeit zu verletzen.
So mag es zusammenfassend durchaus Szenarien geben, bei denen der Basisschutz der algebraischen Sicherheit ausreichend ist. Mit den oben genannten Formeln lässt sich die Gefahr richtig abschätzen oder die Systemparameter entsprechend setzen. Für normale universelle Anwendungen und nicht abschätzbare Angreifer sind jedoch die herkömmliche Ende-zu-Ende-Verschlüsselung oder die leichtgewichtigen Verfahren zu empfehlen. Wie es um die Sicherheit dieser Verfahren steht, soll im folgenden Teil erläutert werden.

5.5.2 Sicherheit von eSPOC

In [LVBM08] wurde gezeigt, dass das Versenden von $\boldsymbol{B}$ über einen gesicherten Kanal erfolgen muss. Dann habe ein Angreifer nur die Chance, die Koeffizienten (also $\boldsymbol{B}$) zu erraten oder direkt die Daten $\boldsymbol{X}$ aus den codierten Daten $\boldsymbol{BX}$ herauszufinden. Es konnte gezeigt werden, dass, sofern $\boldsymbol{X}$ keine Redundanz enthält (also aus von Zufall nicht zu unterscheidenden Daten besteht) und $\boldsymbol{B}$ keine Nullelemente enthält, das Schützen von $\boldsymbol{B}$ ausreicht, um die Vertraulichkeit zu gewährleisten. Dies wird als Sicherheitsbegründung für SPOC herangezogen. Der nun folgende Teil soll verdeutlichen, dass eSPOC zumindest genauso sicher wie SPOC ist, um dann in den folgenden Teilen die Sicherheitseigenschaften beider Verfahren gegenüber P-Coding vergleichen zu können. Doch davor müssen erst die Grundlagen für die Sicherheitsbetrachtungen erläutert werden.
Unter der Annahme, dass die Verschlüsselung der Codierungsmatrix mit Schlüssel κ bei SPOC sicher ist und die Daten codiert vorliegen, besitzt ein passiver Angreifer, der auf alle Nachrichten Zugriff hat, zwei Möglichkeiten, um an die Daten zu kommen:

1. Erraten des Schlüssels κ, um enc$(\boldsymbol{B})$ zu entschlüsseln
2. Erraten von $\boldsymbol{B}$

Das direkte Herausfinden der Daten aus $\boldsymbol{BX}$ ist äquivalent zum Erraten von $\boldsymbol{B}$, da die gleichen Operationen durchgeführt werden müssen wie bei einer Multiplikation mit $\boldsymbol{B}^{-1}$. Um den Suchraum für den ersten Punkt abschätzen zu können, muss bekannt sein, welches Verfahren mit welcher Schlüssellänge

verwendet wurde. Der Einfachheit halber wird für alle Verfahren zur Absicherung der Vertraulichkeit von einer Verschlüsselung auf Grundlage von AES ausgegangen. Dabei sollen jeweils 128 Bit große Schlüssel verwendet werden. Daraus ergibt sich für das Erraten des Schlüssels κ ein Suchraum von 2^{128}. Für eine vereinfachte Darstellung wird dies im Folgenden als eine Bit-Sicherheit $\xi = 128$ beschrieben. Für den zweiten Punkt, Erraten von $\boldsymbol{B}$, benötigt man die Körpergröße q sowie die Generationsgröße h, um die Bit-Sicherheit zu ermitteln. Allerdings sind nicht alle der $q^{(h^2)}$ möglichen Variationen von $\boldsymbol{B}$ auch sinnvoll. Das liegt daran, dass $\boldsymbol{B}$ vollen Rang haben muss, damit $\boldsymbol{B}$ invertierbar ist und am Ende die Daten für den Empfänger decodierbar sind. Daher ist wichtig zu klären, wie wahrscheinlich eine zufällig erzeugte Matrix invertierbar ist.

Es existieren einige Abschätzungen, die besagen, dass die Wahrscheinlichkeit bei etwa $\frac{q-1}{q}$ liegt (z. B. [CWJ03]). Um diese Aussage zu überprüfen, soll die Wahrscheinlichkeit im Folgenden detaillierter angeschaut werden. Damit man sich die Wahrscheinlichkeit für die Invertierbarkeit herzuleiten kann, sollte man induktiv vorgehen. Betrachtet man eine Matrix $\boldsymbol{X}$ der Größe 1×1 und ist das einzige Element $x_{1,1}$ unabhängig und identisch verteilt zufällig aus $\mathbb{F}_q$ ausgewählt, dann ist die Matrix $\boldsymbol{X}$ genau dann invertierbar, wenn $x_{1,1} \neq 0$. Die Wahrscheinlichkeit für eine invertierbare Matrix beträgt somit $1 - \frac{1}{q}$. Erweitert man $\boldsymbol{X}$ zu einer 2×2 Matrix, so spielt der Wert für $x_{1,2}$ keine Rolle, solange $\boldsymbol{x}_2$ kein Vielfaches von $\boldsymbol{x}_1$ ist und $\boldsymbol{x}_2 \neq \boldsymbol{0}$. Die Wahrscheinlichkeit für eine invertierbare Matrix beträgt dann $\left(1 - \frac{1}{q}\right)\left(1 - \frac{1}{q^2}\right)$. Wird das Ganze fortgesetzt, ergibt sich folgende Funktion für die Invertierbarkeit quadratischer Matrizen:

$$\text{inv}(h, q) = \prod_{i=1}^{h} \left(1 - \frac{1}{q^i}\right). \tag{5.9}$$

Tabelle 5.2: Wahrscheinlichkeiten $\text{inv}(h, q)$ für die Invertierbarkeit zufälliger quadratischer Matrizen mit Körpergröße q und Größe $h \times h$.

h	$q = 2$	$q = 4$	$q = 16$	$q = 256$
1	0,5	0,75	0,937 5	0,996 09
2	0,375	0,692 14	0,933 84	0,996 08
4	0,307 62	0,689 44	0,933 60	0,996 08
8	0,289 92	0,688 54	0,933 59	0,996 08
$\to \infty$	0,288 79	0,688 54	0,933 59	0,996 08

In Tabelle 5.2 werden die Wahrscheinlichkeiten $\text{inv}(h, q)$ auf Grundlage von Formel (5.9) dargestellt. Ein steigendes h hat nur für kleinere Körpergrößen q eine Relevanz. Für etwas größere Körper, z. B. $\mathbb{F}_{16}$ oder $\mathbb{F}_{256}$, ist der Wert für eine 2×2 Matrix schon sehr ähnlich zu dem einer beliebig großen Matrix. Darum soll im Folgenden auch hauptsächlich der Wert der Invertierbarkeit für $h \to \infty$ benutzt werden. Um diese Vereinfachung darzustellen, sei sie wie folgt gekennzeichnet: $\text{inv}_\infty(q) = \text{inv}(\infty, q)$. Es zeigt sich weiterhin, dass die Abschätzung von $\frac{q-1}{q}$ in [CWJ03] zumindest für größere Körper relativ passend ist. Beim zugrunde liegenden Körper $\mathbb{F}_2$ liegt die Invertierbarkeit größerer Matrizen mit unter 29% jedoch deutlich unter der Abschätzung von 50%.

Zurückkehrend zu der Frage, wie groß der Suchraum für das Erraten von $\boldsymbol{B}$ ist, lässt sich also festhalten, dass ein Angreifer einen Suchraum von etwa $\text{inv}_\infty(q) * q^{(h^2)}$ für das Erraten hätte. In Bit-Sicherheit umgeformt entspricht dies:

$$\xi = \text{inv}_{\infty}(q) * \text{ld}(q) * h^2. \tag{5.10}$$

Für ein realistisches Szenario mit $h = 16$ und $q = 256$ erhält man eine Bit-Sicherheit von $\xi \approx 2\,039$. Dieser Suchraum ist so groß, dass es für einen Angreifer einfacher wäre, direkt den Schlüssel κ zu erraten. Nur bei kleinen Generationsgrößen mit kleinen Körpern muss sichergestellt werden, dass der Suchraum noch ausreichend groß ist.

Da nun beide Suchräume ausreichend groß sind und in [VLB08] und [LVBM08] nachgewiesen wurde, dass SPOC als sicher – zumindest unter den genannten Einschränkungen – zu betrachten ist, muss nun noch gezeigt werden, dass eSPOC mindestens genauso sicher ist. Dazu muss auf die Unterschiede und auf die Auswirkungen auf die Sicherheit eingegangen werden.

Anstatt $\boldsymbol{B}$ zufällig zu erzeugen und dann verschlüsselt zu übermitteln, wird bei eSPOC $\boldsymbol{B}$ auf beiden Seiten durch eine pseudozufällige Funktion erzeugt. Aus Übertragungssicht ist es nicht sicherer, enc($\boldsymbol{B}$) zu übertragen als ganz auf die Übertragung zu verzichten. Somit bleibt die Frage, ob die Erzeugung von $\boldsymbol{B}$ bei eSPOC unsicherer ist. Anstatt zufällige Werte zu nehmen, wird eine pseudozufällige Funktion verwendet, die einen Schlüssel benötigt. Hier kann davon ausgegangen werden, dass pseudozufälliger Zufall, richtige Parameterwahl vorausgesetzt, eine mindestens genauso hohe Güte wie von gängigen Zufallsgeneratoren erzeugter Zufall besitzt. Daher bleibt dem Angreifer nur ein Angriff auf die Zufallsfunktion übrig. Hier müsste der Schlüssel κ bekannt sein oder erraten werden, um das richtige $\boldsymbol{B}$ zu erzeugen, oder die Verschlüsselung gebrochen werden. Doch wenn κ vorhanden oder das Verschlüsselungsverfahren gebrochen wäre, dann könnte auch enc($\boldsymbol{B}$) in SPOC entschlüsselt werden und man würde auch hier die Daten decodieren können. Darum kann davon ausgegangen werden, dass eSPOC mindestens genauso sicher wie SPOC ist.

5.5.3 Leichtgewichtige Verfahren im Vergleich

Im letzten Punkt wurde die Sicherheit für SPOC und eSPOC in Formel (5.10) aufgezeigt. Dabei ist die Bit-Sicherheit für die herkömmliche Ende-zu-Ende-Verschlüsselung EncPay rein von der Schlüssellänge κ abhängig. Im vorliegenden Fall sei $\xi = 128$ für EncPay. Um einen Vergleich zwischen den Verfahren basierend auf Sicherheit zu machen, muss zuerst die Sicherheit von P-Coding geklärt werden. Danach kann ein Vergleich aller Verfahren, in Abhängigkeit von der Generationsgröße h und der Körpergröße q, durchgeführt werden.

Die Sicherheit des Verfahrens P-Coding basiert auf der Permutation. Unter der Annahme, dass es sich wie bei (e)SPOC um komprimierte Daten der Länge n (d. h. n Elemente pro Paket) handelt, ist ein Angriff auf die Permutationschiffre nur durch Erraten der Permutation oder des Schlüssels κ zur Erzeugung der Permutation möglich. Sei die Paketgröße $w = h + n$, dann gilt, dass es $w!$ Permutationen gibt. Ähnlich wie bei (e)SPOC sind jedoch nicht alle Permutationen sinnvoll, wenn der Angreifer eine komplette Generation abgehört hat. Das liegt daran, dass h der w Spalten vor der Permutation die Codierungsmatrix dargestellt haben. Da diese aber vollen Rang haben muss, gibt es einige Einschränkungen für die Permutation. Eine gültige inverse Permutation bringt also eine permutierte Generation mit Codierungsmatrix und Daten wieder so in Form, dass die ersten h Spalten vollen Rang haben. Die restlichen n Spalten können beliebig permutiert werden. Mithilfe der Formel (5.9) lässt sich bestimmen, wie hoch der Anteil an letztendlich sinnvollen Permutationen in Abhängigkeit von h, n und q ist:

$$\chi(h,n,q) = \frac{\sum_{i=0}^{h} \left(\binom{h}{h-i} \binom{n}{i} * \text{inv}(i,q) \right)}{\binom{h+n}{h}}. \tag{5.11}$$

Unter der Annahme, dass für realistische Szenarien $n >> h$ gilt, ist der Einfluss der h Spalten, die definitiv eine vollrangige Matrix bilden, gegenüber den restlichen n Spalten auf die Wahrscheinlichkeit sehr gering. Daher kann man, um die Berechnung zu vereinfachen, $\chi(h,n,q)$ so approximieren, dass auch die h Spalten als zufallsverteilt angenommen werden. Dadurch ergibt sich folgende Vereinfachung für die Wahrscheinlichkeit, eine gültige Permutation zu erhalten:

$$\chi(h,n,q) \approx \text{inv}(h,q) \approx \text{inv}_\infty(q). \tag{5.12}$$

Um daraus die Bit-Sicherheit ξ zu ermitteln, muss nur noch die Wahrscheinlichkeit mit den Möglichkeiten multipliziert werden. Es gilt:

$$\xi \approx \text{ld}(w!) \cdot \text{inv}_\infty(q). \tag{5.13}$$

Natürlich muss auch hier erwähnt werden, dass es für einen Angreifer ab einer gewissen Nachrichtenlänge und damit einer Bit-Sicherheit von $\xi > 128$ einfacher ist, gleich den Schlüssel κ zur Erzeugung der Permutation als erst den Permutationsvektor selbst zu erraten. Unabhängig von dieser Aussage soll im Folgenden ein Vergleich der Sicherheit erfolgen. Dazu sind nun alle notwendigen Formeln aufgestellt, sodass (e)SPOC und P-Coding vergleichbar sind.

Da in Formel (5.10) als auch in Formel (5.13) der Faktor $\text{inv}_\infty(q)$ vorkommt, lässt sich dieser Faktor herauskürzen und die Gleichheit der Sicherheit beider Verfahren über folgende Formel beschreiben:

$$\text{ld}(w!) = \text{ld}(q) \cdot h^2. \tag{5.14}$$

In Abbildung 5.17 ist nun anhand Formel (5.14) dargestellt, an welchen Punkten sich (e)SPOC und P-Coding vom Suchraum für den Angreifer gleichen. Dabei wurden für 4 verschiedene Körpergrößen q die Verhältnisse von Generationsgröße h und der gesamten Paketgröße w dargestellt. Konfigurationen von h und w, die über der Linie der jeweiligen Körpergröße q liegen, besitzen einen größeren Suchraum für P-Coding. Analog dazu bietet (e)SPOC die größere Sicherheit in Konfigurationen von h und w, die unter einer solchen Linie liegen. Die schwarze Linie zeigt die Sicherheit von EncPay als auch der zugrunde liegenden Funktionen, um $\boldsymbol{B}$ zu verschlüsseln oder zu erzeugen, sowie die Permutation zu erstellen.

So zeigt sich beispielsweise, dass für $q = 256$ die Verfahren P-Coding mit $w > 34$ oder (e)SPOC mit $h > 4$ eine ausreichende Sicherheit bezüglich des Suchraums haben und äquivalent zu EncPay wären. Steigt bei gleichbleibender Generationsgröße h die Paketgröße w, dann wird P-Coding sicherer. Steigt hingegen die Generationsgröße bei fixer Paketgröße w, dann wird (e)SPOC sicherer. Dabei kostet dieses Erhöhen des Suchraums auch mehr Energie, wie man in den Abbildungen 5.7, 5.8 und 5.9 sehen kann. Eine Erhöhung von h vergrößert vor allem den Berechnungsaufwand für (e)SPOC (Abbildung 5.7 zu 5.9), eine Erhöhung von w vor allem für P-Coding (Abbildung 5.7 zu 5.8). Eine Vergrößerung der Sicherheit bedeutet damit auch mehr Berechnungsaufwand für Sender und Empfänger.

Alles in allem lässt sich festhalten, dass die Sicherheit der leichtgewichtigen Verfahren auf einem möglichst großen Suchraum für den Angreifer basiert und dass sich dieser analytisch vergleichen und mithilfe

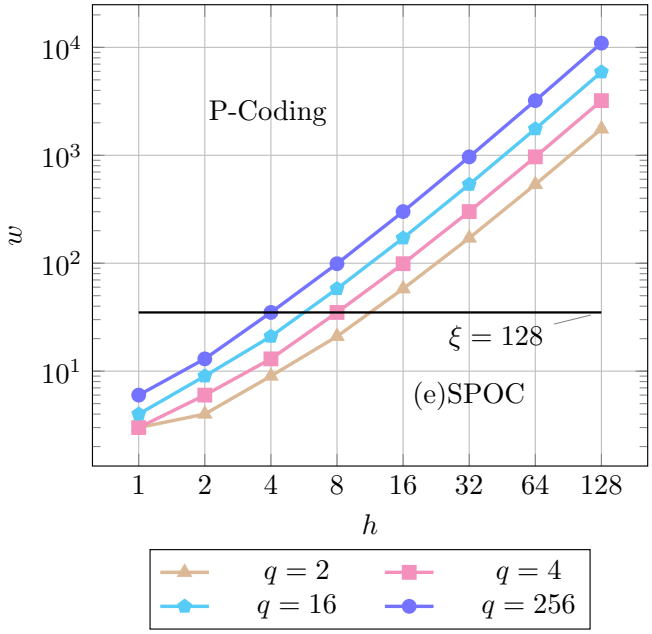


Abbildung 5.17: Gleiche Suchraumgröße für (e)SPOC und P-Coding in Abhängigkeit der Parameter w, h und q. Die schwarze Linie entspricht der Grenze, über der das Erraten von κ weniger Aufwand bedeutet als $\boldsymbol{B}$ oder die Permutation zu erraten. Dies entspricht auch der Sicherheit von EncPay.

der gezeigten Formeln auch ermitteln lässt. Wichtig ist hierbei zu bedenken, dass ab einer gewissen Größe der Parameter h und w der Suchraum für den Schlüssel κ deutlich geringer ist und somit kein Mehr an Sicherheit erreichbar ist. Stattdessen erhöht sich hauptsächlich der Berechnungsaufwand und damit der Energiebedarf der Verfahren. Jedoch kann es im Einzelfall durchaus sinnvoll sein, die Parameter so zu wählen, wie der nächste Abschnitt zeigen soll.

5.5.4 Angriffe bei bekanntem Klartext

Der nachfolgende Teil soll auf ein wenig beachtetes Problem bei den leichtgewichtigen Verfahren hinweisen. Dabei ist das Ziel weniger, die genaue Analyse oder quantitative Aussagen treffen zu können, sondern anhand einer Simulation für ein Beispiel die Probleme aufzuzeigen und ein Bewusstsein für das Problem zu schaffen.

Bisher wurde für den Angreifer immer angenommen, dass dieser zwar alle codierten Nachrichten kenne, aber bis auf dieses Wissen keinerlei Informationen hat. Jedoch ist dies nicht immer der Fall. Es ist durchaus möglich, dass ein Angreifer einzelne Segmente oder einen Teil des Klartexts kennt, z. B. Headerdaten bei HTTP-Anfragen. Deswegen soll im Folgenden ein erweitertes Angreifermodell zum Tragen kommen, bei dem der Angreifer nicht nur h linear unabhängige (codierte) Pakete abgehört hat, sondern auch u Koeffizienten von der (uncodierten) Datenmatrix $\boldsymbol{X}$.

Der Einfachheit halber gilt die Annahme, dass der Angreifer die exakte Position innerhalb der Generation kennt. Sollte das nicht der Fall sein, müsste er die maximal möglichen $h * n$ Positionen in $\boldsymbol{X}$ ausprobieren. Weiterhin wurde zur Veranschaulichung des Problems das Verfahren (e)SPOC gewählt, da es auf einfachen linearen Operationen basiert. Grundsätzlich gelten die Probleme aber ebenfalls für P-Coding oder auch, sofern man nur auf die algebraische Sicherheit setzt, für PNC.

Die Frage ist nun, wie sehr sich ein Wissen von u Positionen auf den Suchraum für $\boldsymbol{B}$ auswirkt. Alternativ – aus Sicht des Angreifers formuliert – kann man fragen, um wie viel sich der Suchraum für den Angreifer bei einem Klartextwissen von u Koeffizienten reduziert. Weiterhin stellt sich die Frage, ob die Position einen Einfluss auf die Reduzierung des Suchraums hat.
Um die Frage des Einflusses der Position zu klären, sollte man den linearen Zusammenhang zwischen der (uncodierten) Matrix $\boldsymbol{X}$ und der (codierten) Matrix $\boldsymbol{BX}$ beachten. Sollte ein Element in $\boldsymbol{X}$ der Größe q bekannt sein, so reduziert sich die Anzahl der möglichen $\boldsymbol{B}$ um den Reduktionsfaktor $r = \frac{1}{q}$. Sollten u Elemente bekannt sein, kommt es auf die lineare Abhängigkeit der Elemente untereinander an. Im schlechtesten Fall (oder für den Angreifer im besten Fall) sind alle der u bekannten Elemente linear unabhängig. Dann ergibt sich für den Reduktionsfaktor r folgende untere Schranke:

$$r = \frac{1}{q^u}. \tag{5.15}$$

Allerdings kann man sich auch einen besten Fall (für den Angreifer der schlechteste Fall) vorstellen. Sollten z. B. alle Symbole aus einer Zeile von $\boldsymbol{X}$ stammen, sind diese linear voneinander abhängig und der gesamte Suchraum ist nur um $r = \frac{1}{q^h}$ verkleinerbar. Diese Abhängigkeit der Elemente einer Zeile wurde in einigen vorherigen Simulationen untersucht und festgestellt, dass sich der Suchraum im Falle von $u = h$ um $r = \frac{1}{q^{h-1}}$ verkleinert hat. Allgemein kann deswegen folgende Formel als Abschätzung für die obere Schranke dienen:

$$r = \frac{1}{q^u} \cdot \left(1 + q^{u-h}\right) = \frac{1}{q^u} + \frac{1}{q^h}. \tag{5.16}$$

Die beiden Schranken in Formel (5.15) und (5.16) sind keine absoluten Schranken, sondern gelten im Mittel. In den meisten Fällen werden die u bekannten Elemente wohl eine gewisse, aber nicht zu starke lineare Abhängigkeit haben, also irgendwo zwischen den beiden Schranken liegen. Beispiele wären mehrere zusammenhängende Teile in unterschiedlichen Zeilen oder Blöcke aus Zeilen und Spalten in $\boldsymbol{X}$.

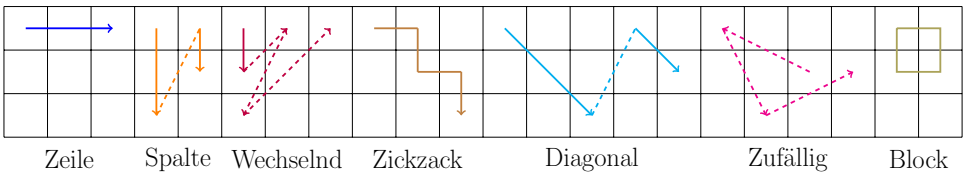


Abbildung 5.18: Verschiedene Muster der u für den Angreifer bekannten Stellen.

Für einen exemplarischen Test sind in Abbildung 5.18 dafür 7 verschiedene Muster dargestellt, von denen ein Angreifer u Positionen im Klartext weiß. Das Muster „Zeile“ bedeutet, dass ein Angreifer u Symbole einer Zeile kennt. Dabei spielt die Reihenfolge und ob die Elemente zusammenhängen oder nicht keine Rolle. Bei „Spalte“ weiß ein Angreifer u Elemente in einer beliebigen Spalte. Sollte $u > h$ gelten, dann wird eine weitere beliebige Spalte ausgewählt. Als Kombination aus beiden gibt es „Wechselnd“, wobei die Hälfte der Elemente in einer Zeile und die andere Hälfte in einer Spalte bekannt ist. Alternativ ist dazu „Zickzack“ zu sehen. Hierbei sind immer Elemente über Zeilen und Spalten miteinander verbunden. Das bedeutet, dass bei jedem bekannten Element (bis auf die Randfälle) jeweils genau ein anderes Element in der gleichen Zeile und in der gleichen Spalte bekannt ist. Muster, die eine geringere

lineare Abhängigkeit zueinander haben sollten, sind „Diagonal“ und „Zufällig“. Bei ersterem sind nur Elemente aus unterschiedlichen Zeilen und Spalten bekannt. Sollte $u > h$ gelten, dann sind Elemente von bereits verwendeten Zeilen, nicht aber von bereits verwendeten Spalten bekannt. Bei „Zufällig“ sind, wie der Name andeutet, alle u bekannten Symbole zufällig und gleichmäßig aus $\boldsymbol{X}$ bekannt. Bei Muster „Block“, welches nur für nichtprime $u = i * j$ mit $i \approx j$ möglich ist, gilt, das i Elemente einer Zeile in j Spalten der Matrix $\boldsymbol{X}$ bekannt sind.

Für eine kleine Simulation wurde jeder der Fälle für ein Szenario mit $q = 2$, $h = 16$, $n = 30$ und $u \in [1, 30]$ getestet. Dabei wurden für jedes Muster und jedes u genau 50 000 000 zufällige $\boldsymbol{B}$ gewählt und bestimmt, ob diese zu den u bekannten Elementen in $\boldsymbol{X}$ kompatibel sind. Dadurch ließ sich der Faktor r ableiten.

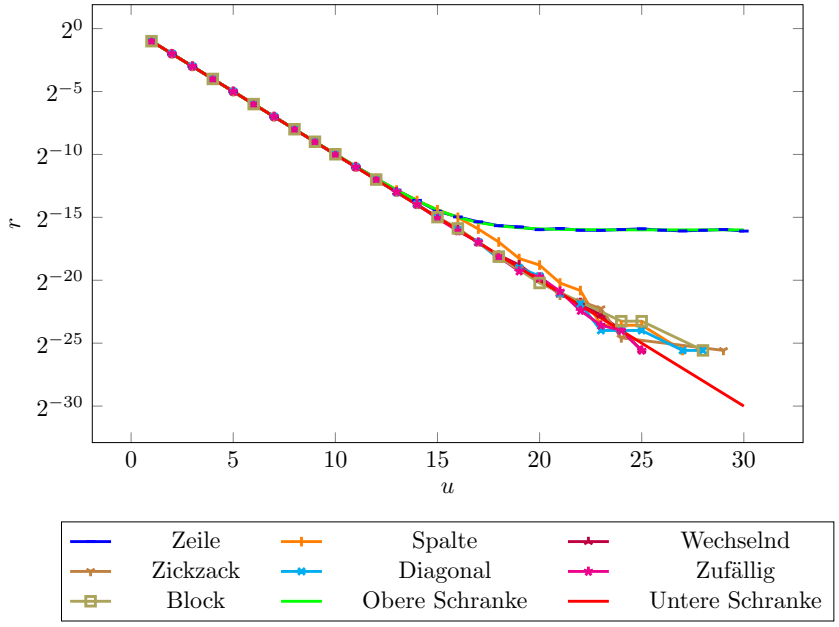


Abbildung 5.19: Ergebnisse für r bei verschiedenen Mustern und einer variierenden Anzahl u an dem Angreifer bekannten Stellen.

In Abbildung 5.19 sind die Ergebnisse für die Simulation und die obere und untere Schranke dargestellt. Man sieht, dass sich alle Muster innerhalb der Schranken bewegen. Das Muster „Zeile“ ist aus Angreifersicht das schlechteste, da der Suchraum um maximal $\frac{1}{q^h}$ verringert wird. Alle anderen Muster sind anscheinend näher an der unteren Schranke. Bei genauer Betrachtung sieht man, dass sich „Spalte“ ähnlich wie „Zeile“ bis zu $u = h$ verhält, dann aber durch Hinzufügen einer neuen Spalte nicht durch h beschränkt wird. Auch „Block“ zeigt eine leicht schlechtere Verkleinerung des Suchraums für den Angreifer. Das beste Ergebnis aus Angreifersicht erzielt „Zufällig“, wobei die Muster „Wechselnd“, „Zickzack“ und „Diagonal“ eine ähnlich gute Verminderung des Suchraums ermöglichen.

Wie erwähnt soll diese Simulation nicht erschöpfend sein, sondern auf die Probleme und Gefahren bei der Verwendung leichtgewichtiger Verfahren hindeuten. Unter der Annahme eines realistischen Szenarios mit $h = 16$, $n = 1\,400$, $q = 8$ und $|\kappa| = 128$ Bit liegt die Größe des Suchraums von (e)SPOC mit $\xi \approx 2\,039$ deutlich über dem des Schlüssels. Trotzdem sind diese Parameter durchaus sinnvoll, wenn Teile der Nachricht bekannt sind. Im Fall, dass der Angreifer neben $\boldsymbol{BX}$ auch 15 der 16 Zeilen in $\boldsymbol{X}$ kennt, hat der Angreifer immer noch einen Suchraum für die fehlende Zeile in $\boldsymbol{B}$ von etwa $\xi \approx 127,5$, was in etwa

dem Aufwand entspricht, den Schlüssel zu knacken. Dies klingt durchaus akzeptabel und sicher. Auf der anderen Seite könnte es im besten Fall für den Angreifer auch ausreichen, 239 Elemente (die nicht linear voneinander abhängig sind) in $\boldsymbol{X}$ zu kennen, um genauso den Suchraum unter $\xi = 128$ des Schlüssels zu bekommen. Letztes Szenario ist allerdings nicht sehr realistisch, da meist zusammenhängende Teile von dem Klartext bekannt sein dürften und ab einer gewissen Anzahl an Elementen aus $\boldsymbol{X}$ immer lineare Abhängigkeiten zwischen den Elementen bestehen werden.
Zusammenfassend ist anzumerken, dass es daher durchaus sinnvoll ist, Daten, die ein Angreifer kennen könnte (also z. B. öffentlich zugängliche Texte, Bilder oder andere Inhalte), nicht in Generationen mit unbekannten und schützenswerten Informationen zu kombinieren. Ist dies unumgänglich, muss zumindest sichergestellt werden, dass nicht innerhalb eines Pakets solche Daten miteinander vermischt werden und die Sicherheitsparameter entsprechend hoch gewählt werden. Insgesamt ist es sehr schwierig, Aussagen über die generelle Sicherheit zu machen, da es zum einen auf die Daten (gibt es Protokolle auf höheren Schichten, sodass relativ viel Klartext bekannt ist oder sind die Daten komprimiert), zum anderen auf die Wahl der Parameter h, q und zumindest für P-Coding auch n ankommt.
Trotz allem bieten die leichtgewichtigen Verfahren für einen überschaubaren Mehraufwand deutlich höhere Sicherheit als der Basisschutz über die algebraische Sicherheit. Weiterhin bieten die leichtgewichtigen Verfahren bei richtiger Parameterwahl und komprimierten, unbekannten Daten durchaus genügend Schutz, wobei die Sicherheit herkömmlicher Ende-zu-Ende-Verschlüsselung robuster gegenüber „falschen“ (d. h. unkomprimierten oder öffentlich bekannten) Daten oder Parametern ist. Dabei ist dann aber auch der Preis für die erhöhten Energieanforderungen und der erhöhte Zeitbedarf zu berücksichtigen.

5.6 eSPOC Storage

Ein relativ junges Gebiet ist die Nutzung von Netzwerkcodierung für verteilte Speicher. Es wurde gezeigt, dass durch das Aufteilen von Daten auf Cloud-Speicher von verschiedenen Anbietern deutliche Vorteile bezüglich des Durchsatzes, der Latenz und der Robustheit erreicht werden können. Dies wird dadurch erreicht, dass – ähnlich wie bei RAID Sytemen – die Clouds auch redundante Daten speichern.
Bezüglich der Sicherheit wird entweder auf die algebraische Sicherheit gesetzt oder bei erhöhten Anforderungen Ende-zu-Ende-Verschlüsselung angewendet. Da es im Gegensatz zur Kommunikation bei solchen Systemen hauptsächlich auf einen hohen Durchsatz und weniger auf eine niedrige Latenz ankommt, werden die Daten meist in größeren „Paketen“ gespeichert. Das heißt, es werden normal große Generationen von etwa $10 \leq h \leq 40$ verwendet, dafür aber große Paketgrößen von z. B. 1 MB gewählt, um den Mehraufwand für die Speicherung der Codierungmatrix recht niedrig zu halten.
Wie im vorherigen Abschnitt dargelegt, ist (e)SPOC für den Berechnungsaufwand und die Geschwindigkeit bei solchen Parametern sehr vorteilhaft, da diese Verfahren hauptsächlich mit großem h aufwendiger werden, sich jedoch kein Mehraufwand für große Pakete bei gleichbleibender Generationsgröße h ergibt. Dementgegen wird EncPay mit größer werdenden Paketen ineffizienter. Das heißt, dass es noch stärkere Vorteile haben sollte, anstatt EncPay die Verfahren SPOC oder eSPOC für die Absicherung zu nehmen. Im Folgenden wird nur das Verfahren eSPOC betrachtet, da gegenüber SPOC nochmals deutliche Leistungsvorteile zu erwarten sind.
Es stellt sich wiederum die Frage, ob zusätzliche Absicherung der Vertraulichkeit überhaupt notwendig ist oder ob nicht doch die algebraische Sicherheit ausreicht. Das Angreifermodell ist nämlich im Ge-

gensatz zur Kommunikation ein anderes. Unter der Annahme, dass der Datenaustausch zwischen dem Nutzer und dem Cloud-Speicher-Anbieter gesichert geschieht, ist der einzige mögliche Angreifer der Cloud-Anbieter selbst. Das ist auch der Grund, weshalb die Netzwerkcodierung hier als vorteilhaft erachtet wird. Anstatt einem Anbieter vertrauen zu müssen, kann man seine Daten codieren und anschließend auf mehrere Clouds aufteilen. Nur wenn alle Cloud-Anbieter zusammenarbeiten, wäre es ihnen möglich, die Daten zu decodieren.

Bei dieser vereinfachten Darstellung scheint es so, als wäre eine zusätzliche Absicherung nicht notwendig, sofern man auf die Annahme vertraut, dass nicht alle Anbieter zusammenarbeiten. Jedoch werden zwei Gesichtspunkte vernachlässigt.

Erstens kann es, wie zuvor gezeigt, zu partiell diagonalisierbaren Matrizen kommen. Das bedeutet, dass, wenn z. B. alle bis auf einen Anbieter zusammenarbeiten, es zu einer gewissen Wahrscheinlichkeit vorkommen kann, dass einzelne „Pakete“ decodiert werden können. Dagegen gibt es ein relativ einfaches Mittel, was schon zu einer bedeutenden Verminderung der Gefahr führen kann. Wenn man sich noch einmal Abbildung 5.16 anschaut, sieht man, dass die Wahrscheinlichkeit, ein Paket decodieren zu können, geringer wird, wenn h größer wird, auch wenn k relativ zu h gesehen konstant bleibt. Für das Beispiel sehe eine Verbesserung also wie folgt aus: Anstatt eine Generationsgröße zu wählen, die der Anzahl der Cloud-Anbieter entspricht, kann ein Vielfaches gewählt werden und dann mehrere Datensätze pro Generation an die Clouds geschickt werden. Zu beachten ist jedoch, dass auch diese Methode ihre Grenzen hat, da zum einen der Aufwand für die Decodierung mit wachsendem h superlinear steigt, zum anderen mehr Speicherplatz für die Codierungsmatrix benötigt wird.

Zweitens müssen die Daten auf die Clouds relativ gleichmäßig verteilt werden und es darf keine Redundanz geben, damit die Annahme, dass alle bis auf eine Cloud zusammenarbeiten dürfen, bestehen bleibt. Ohne Redundanz ist jedoch kein großer Vorteil durch das verteilte Speichern zu erwarten. Es gibt keine Robustheit gegenüber Ausfällen von Clouds und die Latenz bestimmt sich durch die langsamste der Clouds. Ein Lösungsansatz hierfür ist, die Bedingung, dass alle zusammenarbeiten müssen, um decodieren zu können, etwas abzuschwächen, um dann Vorteile wie höhere Geschwindigkeit oder Ausfallsicherheit zu erhalten. Das bedeutet, dass man sich einen Kompromiss aus Vertraulichkeit und Redundanz sucht.

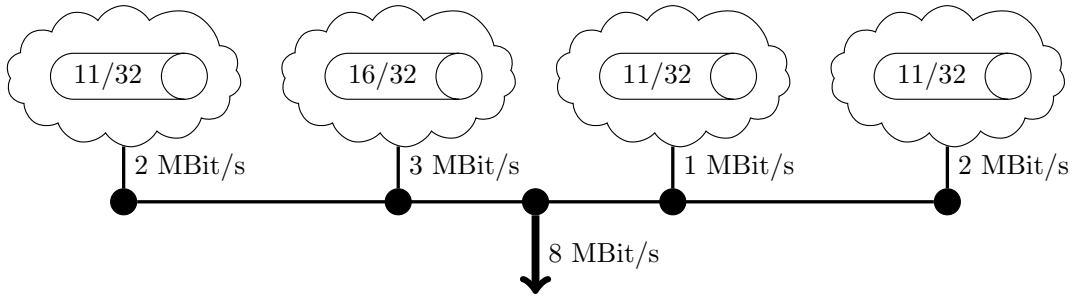


Abbildung 5.20: Beispiel einer Konfiguration, bei der zwei zusammenarbeitende Clouds nicht decodieren können und bei Aufall einer Cloud immer noch genügend Daten vorhanden sind, um Verfügbarkeit zu gewährleisten.

Abbildung 5.20 ist einem Beispiel aus [SFLP14] nachempfunden. Die Idee ist, die Pakete aus einer Generation (hier mit Generationsgröße $h = 32$) so aufzuteilen, dass 3 Clouds ausreichen, um alle Da-

ten wiederherzustellen (Ausfallsicherheit durch Redundanz). Weiterhin sind aber 2 zusammenarbeitende Clouds nicht in der Lage, die Generation zu decodieren. Aufgrund der unterschiedlichen Geschwindigkeiten der Clouds hat die zweite Cloud ein etwas erhöhtes Datenvolumen, um die höchstmögliche Downloadgeschwindigkeit (entspricht der Summe aller Einzelgeschwindigkeiten) zu erzielen. Man erreicht $8 = 2 + 3 + 1 + 2$ MBit/s. Weiterhin besitzen 2 Clouds maximal 27 von 32 Paketen und können deswegen nicht decodieren. 3 Clouds sind aber im Besitz von mindestens 33 Paketen, was eine Decodierung ermöglicht.
Zwar ist nun die Fehlertoleranz gegeben und auch die Geschwindigkeit im ausfallfreien Fall gut, jedoch bleibt die Frage, wie sich die Leistungsfähigkeit verhält, wenn eine Cloud ausfällt.

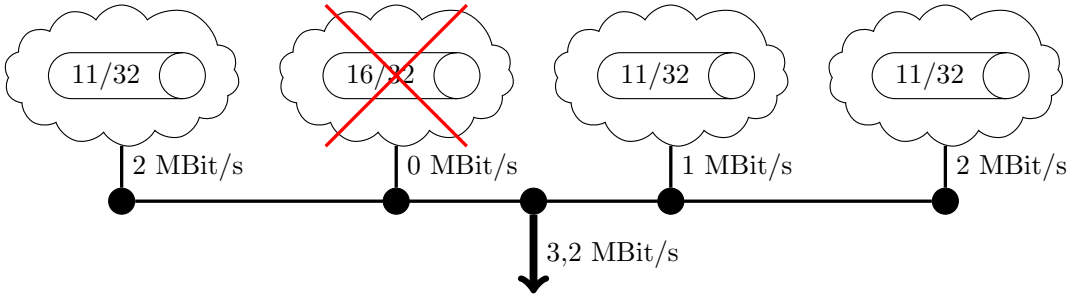


Abbildung 5.21: Eine Cloud in der Konfiguration aus Abbildung 5.20 ist nicht mehr verfügbar. Durch die geringe Redundanz besitzen Cloud 1 und 4 nicht genügend Daten, sodass auf Cloud 3 gewartet werden muss, was den Datendurchsatz verringert.

In Abbildung 5.21 ist beispielsweise die zweite Cloud ausgefallen und somit nicht mehr erreichbar. Von den theoretisch möglichen 5 MBit/s können nur 3,2 MBit/s abgerufen werden, da Cloud 1 und Cloud 4 keine zusätzlichen redundanten Pakete besitzen. Zur Ermittlung der Übertragungsgeschwindigkeit und der bestmöglichen Verteilung wurde ein kleines Skript in Python erstellt und angewendet.
Man kann die Werte aber auch analytisch nachvollziehen, wenn man beispielsweise davon ausgeht, dass in einer Zeiteinheit bei 1 MBit/s genau ein Paket abgerufen werden kann. Bei dem in Abbildung 5.21 dargestellten Sachverhalt kann dann bis zum Zeitpunkt 5,5 mit 5 MBit/s heruntergeladen werden. Es fehlen dann noch 4,5 Pakete, welche nur noch bei Cloud 3 verfügbar sind. Nach weiteren 4,5 Zeiteinheiten sind dann insgesamt 32 Pakete heruntergeladen und die Decodierung kann beginnen. Bei insgesamt 10 Zeiteinheiten wird also zu 55 % mit 5 MBit/s und zu 45 % mit 1 MBit/s heruntergeladen. Dies ergibt eine durchschnittliche Geschwindigkeit von $0{,}55 * 5$ MBit/s $+ 0{,}45 * 1$ MBit/s $= 3{,}2$ MBit/s.

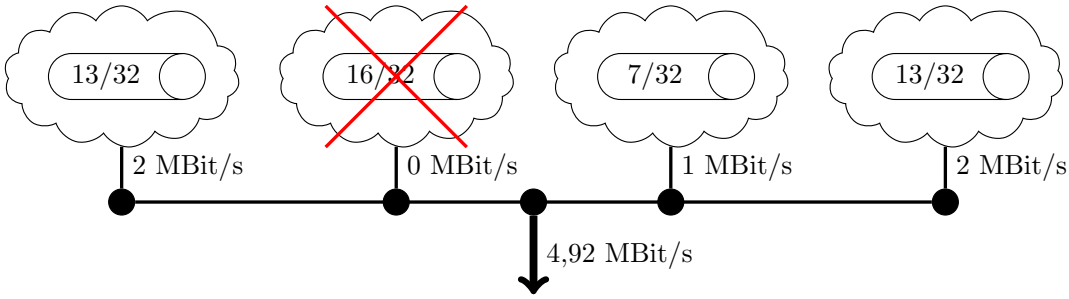


Abbildung 5.22: Das Beispiel aus Abbildung 5.21 mit leicht verschobenen Anteilen pro Cloud erreicht einen höheren Durchsatz im Fehlerfall.

Bei diesem Beispiel lassen sich die Anteile der Daten auf den Clouds noch relativ leicht so verschieben, dass die Beschränkungen für die Vertraulichkeit und Verfügbarkeit trotzdem erfüllt sind. Abbildung 5.22 zeigt das geänderte Beispiel. Dass am Ende nur eine Geschwindigkeit von $4{,}92$ MBit/s von möglichen 5 MBit/s erreicht werden, liegt an der Stückelung der Pakete. Dieser Effekt wird in den folgenden Beispielen immer wieder auftreten, ist aber relativ unkritisch, da er bei der Übertragung mehrerer Generationen relativiert wird.
Zu beachten ist auch, dass, obwohl die Sicherheitskriterien eingehalten wurden, mit der jetzigen Aufteilung 2 Clouds (z. B. Cloud 1 und 2) maximal 29 statt wie davor nur 27 von $h = 32$ Paketen einsehen können. Damit erhöht sich also auch die Gefahr für partielle Decodierbarkeit. Diese Gefahr soll für das nächste Beispiel auch erstmal ignoriert werden und es als ausreichend erachtet werden, wenn 2 Clouds maximal $31 < h$ Pakete besitzen.
Wie gezeigt wurde, existieren Beispiele für Szenarien, bei denen es im Fehlerfall zu einer Abschwächung der Übertragungsrate kommt, da auf algebraische Sicherheit anstatt auf ein leichtgewichtiges Verfahren zur Absicherung der Vertraulichkeit gesetzt wurde. Doch gerade bei sehr heterogenen Cloud-Anbietern (auf die Geschwindigkeit bezogen) kommt es zu diesem Problem nicht nur im Fehlerfall, sondern kann auch im Regelbetrieb auftreten.

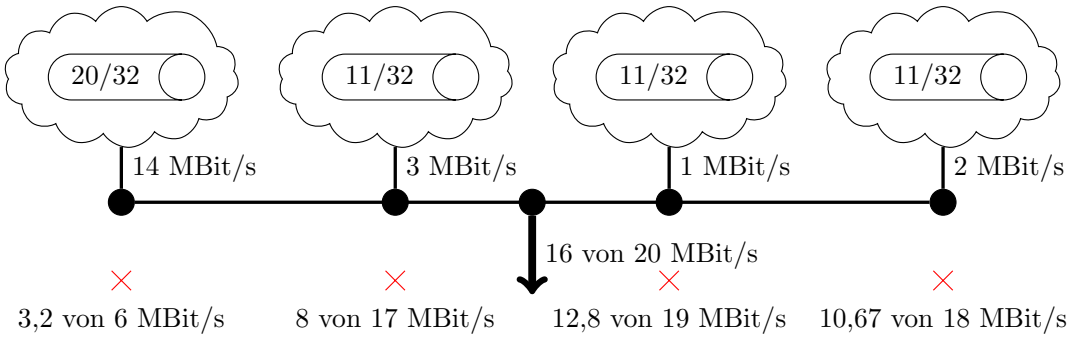


Abbildung 5.23: Sehr heterogene Clouds mit algebraischer Sicherheit erreichen auch im normalen Betrieb nicht die maximal erreichbaren Übertragungsraten. Die Werte unter dem roten Kreuz ergeben sich bei Ausfall der darüber liegenden Cloud.

Abbildung 5.23 zeigt ein Beispiel, bei dem es selbst im Regelbetrieb zu Geschwindigkeitseinbußen kommt. Es existieren 4 Clouds, von denen eine sehr schnell, die anderen eher langsam sind. Um zu gewährleisten, dass 3 Clouds mindestens 32 Pakete besitzen, haben die 3 langsameren Clouds jeweils 11 Pakete. Da die erste Cloud sehr schnell ist, lohnt es sich, dort soviel wie möglich Pakete zu speichern. Doch da zwei Clouds maximal 31 Pakete pro Generation besitzen dürfen, kann somit Cloud 1 nur 20 Pakete besitzen, obwohl es von der Geschwindigkeit her sinnvoller wäre, mehr Pakete dort zu lagern.
Man sieht, dass selbst im Regelbetrieb nur 16 von möglichen 20 MBit/s erreicht werden. Das liegt daran, dass die Übertragungen von Cloud 1 vorzeitig enden, da keine Pakete mehr vorhanden sind und der Rest über die 3 langsameren Clouds abgerufen werden muss. Im Fehlerfall, wenn eine Cloud ausfällt (in der Abbildung je rotes $\times$ unter der Cloud), werden auch nur gut die Hälfte (im Durchschnitt $57{,}8\%$) der maximalen Geschwindigkeit erreicht.
Es ist also auch möglich, dass durch das Verwenden der algebraischen Sicherheit nicht nur Sicherheitsprobleme (z. B. partielle Decodierbarkeit), sondern auch Leistungsprobleme, eventuell sogar im Normalfall aufgrund der Beschränkungen bezüglich der Anzahl der Pakete pro Cloud, auftreten können. Daher

scheint es sinnvoll, auf eine Absicherung der Vertraulichkeit zu setzen, um diese Beschränkungen zu umgehen.

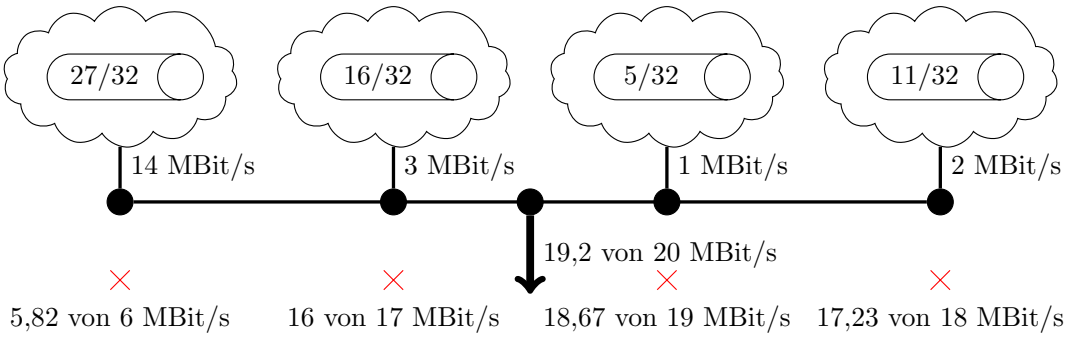


Abbildung 5.24: Sehr heterogene Clouds mit nahezu optimalen Übertragungsraten, die auch bei Fehlerfällen gelten, bei Verwendung leichtgewichtiger Verfahren wie eSPOC.

Abbildung 5.24 zeigt eine Konfiguration, die mit leichtgewichtigen Verfahren möglich wäre und bei nur etwa 11% mehr Speicherbedarf optimale Übertragungsraten, auch bei Ausfall einer beliebigen Cloud, ermöglicht. Dass die Werte nicht der Summe der Werte entsprechen, liegt an der bereits erwähnten Stückelung. Auch hier speichern 3 beliebige Clouds mindestens h Pakete und ermöglichen die Decodierung bei Ausfall einer Cloud. Allerdings kann nun zwar eine beliebige Anzahl an Clouds (auch alle) zusammenarbeiten, aber durch die Absicherung der Vertraulichkeit die Originaldaten nicht decodieren.
Es zeigt sich also, dass durch die Benutzung der algebraischen Sicherheit als Absicherungsmaßnahme gegenüber der Vertraulichkeit große Einschränkungen bezüglich der Geschwindigkeit und der Redundanz gemacht werden müssen. Eine erhöhte Vertraulichkeit geht immer zu Lasten der Verfügbarkeit und umgekehrt. Ende-zu-Ende-Verschlüsselung wie auch die leichtgewichtigen Verfahren ermöglichen größtmögliche Verfügbarkeit bei gleichzeitiger Vertraulichkeit, sofern genügend Speicherplatz vorhanden ist. Die leichtgewichtigen Verfahren, besonders (e)SPOC, haben jedoch den Vorteil, dass bei der Speicherung großer Daten die Generationsgröße im Vergleich zur Codierungsmatrix recht gering ausfällt. Somit ermöglicht (e)SPOC einen immensen Zeit- und Energievorteil gegenüber Ende-zu-Ende-Verschlüsselung.

5.7 Zusammenfassung

In diesem Kapitel wurde das Gebiet der Vertraulichkeit in Bezug auf die Netzwerkcodierung näher beleuchtet. Die häufig als Vorteil der Netzwerkcodierung genannte algebraische Sicherheit wurde bewertet und deren Grenzen aufgezeigt. Nach allen Betrachtungen kann diese nur als Basisschutz dienen und ist zusätzlich schwer gegenüber Angreifer unbekannter Stärke zu parametrisieren. Weiterhin birgt das Vertrauen auf die algebraische Sicherheit Gefahren durch die partielle Diagonalisierbarkeit von Matrizen.
Die Standardmöglichkeit der Ende-zu-Ende-Verschlüsselung ist zwar möglich und sicher, lässt aber die Vorteile der Netzwerkcodierung ungenutzt. Bessere Leistungen ermöglichen leichtgewichtige Verfahren, welche die algebraische Sicherheit ausnutzen, um mit relativ wenig Aufwand eine deutliche Verbesserung der Vertraulichkeit zu schaffen. Wie gezeigt werden konnte, erreichten diese Verfahren bessere Leistungen gegenüber der Ende-zu-Ende-Verschlüsselung. Zum einen zeigt sich das beim niedrigeren

Berechnungsaufwand und damit geringeren Energieverbrauch. Zum anderen verringert sich auch die Latenz und damit die Gesamtdauer der Kommunikation.
Das selbst entwickelte Verfahren eSPOC zeigte nochmal erheblich verbesserte Werte und bietet so die Möglichkeit, den Basisschutz der algebraischen Sicherheit mit minimalen Kosten zu ergänzen. Auch bezüglich der Sicherheit konnte gezeigt werden, dass eSPOC den anderen leichtgewichtigen Verfahren, je nach Parameterwahl, ebenbürtig ist. Trotzdem sei auf die Angriffsmöglichkeit bei bekanntem Klartext verwiesen, sodass die Sicherheit von eSPOC auch ihre Grenzen hat.
Auch die Frage, ob eSPOC im Bereich von verteilten Speichern eine Rolle spielen könnte, kann nach den aufgeführten Betrachtungen bejaht werden. Die Nachteile, die entstehen, wenn auf algebraische Sicherheit gesetzt wird, betreffen nicht nur die Sicherheit selbst, sondern auch die Leistungsfähigkeit. Daher ist es empfehlenswert, auf eSPOC zu setzen, da dieses Verfahren bei großen Paketen und relativ kleiner Körpergröße eine deutliche Kostenersparnis gegenüber den anderen sicheren Verfahren bietet.

6 Zusammenfassung und Ausblick

Im Folgenden sollen noch einmal alle wichtigen Ergebnisse und Erkenntnisse aufgezeigt werden, die in dieser Dissertation behandelt wurden. Dabei soll auch immer ein Ausblick auf zukünftige Entwicklungen und Möglichkeiten für Anknüpfungspunkte angefügt werden.

In zwei Kapiteln wurde sich der Absicherung der Integrität bei der Netzwerkcodierung gewidmet. Aufgrund der Linearkombinationen, die Knoten bei der Netzwerkcodierung bilden können, ist dies besonders wichtig, da es sonst zu Verschmutzungsangriffen kommen kann.

Es wurden unterschiedliche Methoden vorgestellt und diese in verschiedene Gruppen unterteilt. Grob aufgegliedert, gibt es zwei Arten von Verfahren. Bei den codierungstheoretischen Ansätzen werden den Generationen zusätzliche redundante Pakete hinzugefügt. Sollte es zu einer Verfälschung kommen, wird diese über die redundanten Pakete am Empfänger erkannt und korrigiert. Bei Verfahren, welche auf Kryptographie basieren, wird an jedem Zwischenknoten die Integrität der Pakete vor dem Bilden einer Linearkombination überprüft und ungültige Pakete aussortiert, damit keine Verschmutzung entstehen kann.

Die Auswertungen zeigen, dass die auf Kryptographie basierenden Verfahren insgesamt gesehen deutliche Vorteile in Bezug auf die Anzahl zu sendender Nachrichten bieten und unter der Bedingung, dass Zwischenknoten in der Lage sind, Nachrichten zu überprüfen, den codierungstheoretischen Verfahren vorzuziehen sind.

In der Gruppe der Kryptographie-Verfahren wurden 4 verschiedene Klassen der Absicherung ausgemacht und für jede Klasse ein Vertreter gewählt. Bei den Untersuchungen der Effizienz zeigte sich, dass kein Verfahren für alle gewählten Netzwerktopologien die besten Ergebnisse erzielte. Es stellte sich eine starke Abhängigkeit der Effizienz der Verfahren hauptsächlich von 2 Parametern des Netzwerks, der Generationsgröße und der Länge des längsten Pfads, heraus. Aufgrund dessen wurde ein generalisiertes Netzwerkmodell entwickelt, welches auf beliebige Netzwerke anwendbar ist.

Bei der Untersuchung der Effizienzparameter konnte gezeigt werden, dass alle Verfahren ihre Daseinsberechtigung haben und jeweils für gewisse Parameterkonfigurationen die höchste Effizienz erreicht wird. Daher wurde ein adaptives Verfahren vorgeschlagen, welches zu einer gegebenen Netzwerktopologie das effizienteste Verfahren auswählt.

Aber auch eine andere Klassifizierung wurde untersucht. Dabei wurde vordefinierte Redundanz gegenüber ratenlosem Codieren mit Bestätigungen des Empfängers überprüft. Durch diese Rückmeldungen ergab sich eine deutlich bessere Anpassung an einen Angreifer als bei vorher fest definierter Redundanz. Deswegen ist ratenloses Codieren, sofern möglich, bei der Netzwerkcodierung immer empfehlenswert.

Ausgehend von diesem Ergebnis wurden Möglichkeiten untersucht, die Rückmeldungen auszunutzen, um die Übertragung zu beschleunigen und das mit möglichst wenig zusätzlichen Nachrichten. Neben der Idee, die Redundanzrate dynamisch anhand der Rückmeldungen anzupassen, wurde auch ein adaptives Übertragungsprotokoll erarbeitet, welches eine stufenlose Optimierung in der Wechselbeziehung zwischen Zeitbedarf und Sendeoperationen ermöglicht.

Insgesamt wurden so Techniken untersucht und optimiert und ein adaptives Verfahren geschaffen, das je nach Netzwerk eine optimale Leistung erzielt. Es konnte auch gezeigt werden, dass die Netzwerkcodierung trotz des zusätzlichen Aufwands für die Integritätsabsicherung Vorteile gegenüber herkömmlichem Routing hat. Gerade im praktischen Einsatz wäre die Absicherung bei herkömmlichem Routing ebenfalls notwendig.

Aus methodischer Sicht wurde ein Simulator basierend auf SageMath entwickelt, welcher die Übermittlungen mit sicherer Netzwerkcodierung nachbildet. Zahlreiche Parameter und eine einfache Struktur erlauben eine schnelle Anpassung an geeignete Effizienzparameter oder Übermittlungsprotokolle. Somit ergibt sich eine Grundlage für weitere Untersuchungen.

Zukünftige Verbesserungen liegen in der Auswahl und Optimierung einzelner Verfahren. So besteht die Möglichkeit, bei dem MAC-Verfahren weitere Optimierungen umzusetzen. Um den Berechnungsaufwand zu verringern und den Nutzdatenanteil hoch zu halten, könnten dort anstatt eines MACs im großen Körper mehrere MACs im kleinen Körper pro Prüfvorgang verwendet werden. Auch um die Zeitverzögerungen gering zu halten, könnte von einem TESLA-basierten Verfahren auf ein Verfahren mit Schlüsselpool gewechselt werden. Weiterhin sollten andere Signaturverfahren, welche nicht auf RSA basieren, für größere Generationen und tiefere Netzwerke besser geeignet sein. Zusätzlich ist ein Zusammenlegen mit Verfahren für die Vertraulichkeit wünschenswert.

Die Vertraulichkeit wurde in einem weiteren Kapitel untersucht. Darin wurde die algebraische Sicherheit vorgestellt und deren Grenzen aufgezeigt. Für höhere Sicherheitsansprüche müssen folglich zusätzliche Maßnahmen ergriffen werden. Dabei gibt es neben der Standardvariante, dem Verschlüsseln der Nutzdaten, auch leichtgewichtige Verfahren, welche einen reduzierten Berechnungsaufwand versprechen.

Ausgewählte Verfahren wurden implementiert, um deren Effizienz gegenüber der Standardvariante zu vergleichen. Weiterhin wurde ein eigenes optimiertes Verfahren (eSPOC) entwickelt, welches deutliche Vorteile gegenüber allen anderen Verfahren hat.

Auch die Sicherheit der verschiedenen Verfahren wurde analysiert und festgestellt, dass für einen begrenzten Angreifer die leichtgewichtigen Verfahren je nach Parameterwahl ausreichend sicher sind. Hier zeigte sich bei eSPOC, dass es mindestens so sicher wie andere leichtgewichtige Verfahren ist.

Ein letztes wichtiges Ergebnis ist die Anwendbarkeit von eSPOC im Bereich der verteilten Speicher. Analytisch konnten deutliche Vorteile gegenüber nur auf algebraischer Sicherheit basierender Verfahren gezeigt werden, insbesondere bei Ausfällen ganzer Clouds. Eine Einbindung in praktischen Systemen erscheint daher sinnvoll.

Methodisch steht hier die Implementierung von leichtgewichtigen Verfahren und des eigens entwickelten eSPOC mittels Kodo im Vordergrund. Dies erlaubt einen Einsatz der Verfahren auf nahezu beliebigen Plattformen ohne größere Änderungen am Quellcode. Diese Implementierungen bieten die Basis für verschiedene Untersuchungen und spätere Anwendungen.

Offen für zukünftige Untersuchungen bleibt noch die tatsächliche Umsetzung der leichtgewichtigen Verfahren für verteilte Speicher, wobei die vorhandenen Implementierungen verwendet werden können. Weiterhin ist auch hier die Kombination mit Verfahren für die Integrität zu evaluieren.

Insgesamt gesehen konnte die Arbeit zeigen, dass auch sichere Netzwerkcodierung eine bessere Effizienz als herkömmliche Paketvermittlung bieten kann. Dabei wurden Methoden aufgezeigt, wie die Effizienz dieser Methoden weiter verbessert werden kann. Neben der Konsolidierung verfügbarer Verfahren wurde erarbeitet, wann welche Methoden zum Einsatz kommen sollten.

Die präsentierten Ergebnisse eröffnen neue Schritte in eine Kommunikationstechnik mit Netzwerkcodierung, welche Probleme der aktuellen Übertragungsformen, wie z. B. Flaschenhälse oder schlechte Skalierbarkeit, mit einem verringerten Energiebedarf lösen können. Sicherheit ist dabei ein integraler Bestandteil der Kommunikationsprotokolle und muss nicht, wie so oft, im Nachhinein hinzugefügt werden.

Literaturverzeichnis

[AB09] AGRAWAL, Shweta ; BONEH, Dan: Homomorphic MACs: MAC-based integrity for network coding. In: *Lecture Notes in Computer Science* 5536 LNCS (2009), Nr. 1, S. 292–305

[ACLY00] AHLSWEDE, Rudolf ; CAI, Ning ; LI, Shuo-Yen R. ; YEUNG, Raymond W.: Network Information Flow. In: *IEEE Transactions on Information Theory* 46 (2000), Nr. 4, S. 1204–1216

[AL09] ADELI, Majid ; LIU, Huaping: Secure Network Coding with Minimum Overhead Based on Hash Functions. In: *IEEE Communications Letters* 13 (2009), Nr. 12, S. 956–958

[AMC11] ANGELOPOULOS, Georgios ; MÉDARD, Muriel ; CHANDRAKASAN, Anantha P.: Energy-Aware Hardware Implementation of Network Coding. In: *NETWORKING Workshops* (2011), S. 137–144

[BCF+15] BIELERT, Mario ; CIORBA, Florina M. ; FELDHOFF, Kim ; ILSCHE, Thomas ; NAGEL, Wolfgang E.: HAEC-SIM: A Simulation Framework for Highly Adaptive Energy-Efficient Computing Platforms. In: *Proc. of Eighth Conference on Simulation Tools and Techniques*, 2015, S. 128–138

[BF11] BONEH, Dan ; FREEMAN, David M.: Linearly Homomorphic Signatures over Binary Fields and New Tools for Lattice-Based Signatures. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 6571 (2011), S. 1–16

[BFKW09] BONEH, Dan ; FREEMAN, David ; KATZ, Jonathan ; WATERS, Brent: Signing a Linear Subspace: Signature Schemes for Network Coding. In: JARECKI, Stanisław (Hrsg.) ; TSUDIK, Gene (Hrsg.): *Public Key Cryptography – PKC: 12th International Conference on Practice and Theory in Public Key Cryptography, Irvine, CA, USA, March 18-20, 2009. Proceedings*. Springer Berlin Heidelberg, 2009, S. 68–87

[Bun17] BUNDESAMT FÜR SICHERHEIT IN DER INFORMATIONSTECHNIK: *Kryptographische Verfahren: Empfehlungen und Schlüssellängen*. BSI - Technische Richtlinie, 2017

[Cat14] CATALANO, Dario: Homomorphic Signatures and Message Authentication Codes. In: *Security and Cryptography for Networks* 8642 (2014), S. 514–519

[CFW12] CATALANO, Dario ; FIORE, Dario ; WARINSCHI, Bogdan: Efficient Network Coding Signatures in the Standard Model. In: *PKC: Public Key Cryptography* (2012), S. 680–696

[Cha88] CHAUM, David: The Dining Cryptographers Problem: Unconditional Sender and Recipient Untraceability. In: *Journal of Cryptology* 1 (1988), Nr. 1, S. 65–75

[CHLT14] CHEN, Henry C. H. ; HU, Yuchong ; LEE, Patrick P C. ; TANG, Yang: NCCloud: A Network-Coding-Based Storage System in a Cloud-of-Clouds. In: *IEEE Transactions on Computers* 63 (2014), Nr. 1, S. 31–44

[CIF+14] CIORBA, Florina M. ; ILSCHE, Thomas ; FRANZ, Elke ; PFENNIG, Stefan ; SCHEUNERT, Christian ; MARKWARDT, Ulf ; SCHUCHART, Joseph ; HACKENBERG, Daniel ; SCHÖNE, Robert ; KNÜPFER, Andreas ; NAGEL, Wolfgang E. ; JORSWIECK, E. A. ; MÜLLER, Matthias S.: Analysis of Parallel Applications on a High Performance - Low Energy Computer. In: *Proc. of Conference on Parallel Processing (Euro-Par)*, 2014, S. 474–485

[CJ11] CHENG, Chi ; JIANG, Tao: A Novel Homomorphic MAC Scheme for Authentication in Network Coding. In: *IEEE Communications Letters* 15 (2011), Nr. 11, S. 1228–1230

[CJKK07] CHACHULSKI, Szymon ; JENNINGS, Michael ; KATTI, Sachin ; KATABI, Dina: Trading Structure for Randomness in Wireless Opportunistic Routing. In: *ACM SIGCOMM Computer Communication Review* 37 (2007), Nr. 4, S. 169–180

[CJL09] CHARLES, Denis ; JAIN, Kamal ; LAUTER, Kristin: Signatures for Network Coding. In: *International Journal of Information and Coding Theory* (2009), S. 857–863

[Cry16] CRYPTO++ COMMUNITY: *Crypto++ Library 5.6.5*. `http://www.cryptopp.com/`. Version: 2016

[CW07] CHOU, Philip A. ; WU, Yunnan: Network Coding for the Internet and Wireless Networks. In: *IEEE Signal Processing Magazine* 24 (2007), Nr. 5, S. 77–85

[CWJ03] CHOU, Philip A. ; WU, Yunnan ; JAIN, Kamal: Practical Network Coding. In: *41st Annual Allerton Conference* (2003), S. 1–10

[CY02] CAI, Ning ; YEUNG, Raymond W.: Secure Network Coding. In: *Proceedings. IEEE International Symposium on Information Theory* (2002), S. 323

[CY07] CAI, Ning ; YEUNG, Raymond W.: A Security Condition for Multi-Source Linear Network Coding. In: *IEEE International Symposium on Information Theory – Proceedings* (2007), S. 561–565

[CY11] CAI, Ning ; YEUNG, Raymond W.: Secure Network Coding on a Wiretap Network. In: *IEEE Transactions on Information Theory* 57 (2011), Nr. 1, S. 424–435

[DBN+01] DWORKIN, Morris J. ; BARKER, Elaine B. ; NECHVATAL, James R. ; FOTI, James ; BASSHAM, Lawrence E. ; ROBACK, E. ; DRAY JR., James F.: Announcing the Advanced Encryption Standard (AES). In: *Technology Laboratory, National Institute of Standards* 2009 (2001), S. 8–12

[DCNR09] DONG, Jing ; CURTMOLA, Reza ; NITA-ROTARU, Cristina: Secure network coding for wireless mesh networks: Threats, challenges, and directions. In: *Computer Communications* 32 (2009), Nr. 17, S. 1790–1801

[DCNR11] DONG, Jing ; CURTMOLA, Reza ; NITA-ROTARU, Cristina: Practical Defenses Against Pollution Attacks in Wireless Network Coding. In: *ACM Transactions on Information and System Security* 14 (2011), Nr. 1, S. 1–31

[Fer09] FERREIRA, Diogo: NECO : NEtwork COding simulator. In: *SIMUTools* (2009), S. 1–21

[FJZ+11] FAN, Yanfei ; JIANG, Yixin ; ZHU, Haojin ; CHEN, Jiming ; SHEN, Xuemin S.: Network Coding Based Privacy Preservation against Traffic Analysis in Multi-Hop Wireless Networks. In: *IEEE Transactions on Wireless Communications* 10 (2011), Nr. 3, S. 834–843

[FJZS09] FAN, Yanfei ; JIANG, Yixin ; ZHU, Haojin ; SHEN, Xuemin: An Efficient Privacy-Preserving Scheme against Traffic Analysis Attacks in Network Coding. In: *Proceedings of IEEE INFOCOM* (2009), S. 2213–2221

[FOG08] FUJIMURA, Atsushi ; OH, Soon Y. ; GERLA, Mario: Network Coding vs. Erasure Coding: Reliable Multicast in Ad Hoc Networks. In: *Proceedings – IEEE Military Communications Conference MILCOM* (2008), S. 1–7

[FPF12a] FRANZ, Elke ; PFENNIG, Stefan ; FISCHER, André: Communication Overhead of Network Coding Schemes Secure against Pollution Attacks / TU Dresden. 2012. – Forschungsbericht

[FPF12b] FRANZ, Elke ; PFENNIG, Stefan ; FISCHER, André: Efficiency of Secure Network Coding Schemes. In: *Proc. of Communications and Multimedia Security*, Springer, 2012, S. 145–159

[FS07] FRAGOULI, Christina ; SOLJANIN, Emina: Network Coding Applications. In: *Foundations and Trends® in Networking* 2 (2007), Nr. 2, S. 135–269

[FWB06] FRAGOULI, Christina ; WIDMER, Jörg ; BOUDEC, Jean-Yves L.: A Network Coding Approach to Energy Efficient Broadcasting: from Theory to Practice. In: *IEEE INFOCOM* (2006), S. 1–11

[GKKR09] GENNARO, Rosario ; KATZ, Jonathan ; KRAWCZYK, Hugo ; RABIN, Tal: Secure Network Coding Over the Integers. In: *IEEE Communications Letters* 13 (2009), Nr. 12, S. 956–958

[GR06] GKANTSIDIS, Christos ; RODRIGUEZ, Pablo R.: Cooperative Security for Network Coding File Distribution. In: *IEEE INFOCOM* (2006), S. 1–13

[GRU14] GÜNTHER, Stephan M. ; RIEMENSBERGER, Maximilian ; UTSCHICK, Wolfgang: Efficient GF Arithmetic for Linear Network Coding using Hardware SIMD Extensions. In: *International Symposium on Network Coding, NetCod – Conference Proceedings* (2014), S. 1–6

[HKM+03] HO, Tracey ; KOETTER, Ralf ; MÉDARD, Muriel ; KARGER, David R. ; EFFROS, Michelle: The Benefits of Coding over Routing in a Randomized Setting. In: *Proceedings. IEEE International Symposium on Information Theory* (2003), S. 442

[HKM05] HARVEY, Nicholas J. A. ; KARGER, David R. ; MUROTA, Kazuo: Deterministic Network Coding by Matrix Completion. In: *SODA Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms* (2005), S. 489–498

[HKPW12] HESSLER, Alban ; KAKUMARU, Takahiro ; PERREY, Heiner ; WESTHOFF, Dirk: Data Obfuscation with Network Coding. In: *Computer Communications* 35 (2012), Nr. 1, S. 48–61

[HLK+04] HO, Tracey ; LEONG, B ; KOETTER, Ralf ; MÉDARD, Muriel ; EFFROS, Michelle ; KARGER, David R.: Byantine Modification Detection in Multicast Networks using Randomized Network Coding. In: *Proc. IEEE Int. Symp. on Info. Theory* 54 (2004), Nr. 6, S. 2798–2803

[HV09] HO, Tracey ; VISWANATHAN, Harish: Dynamic Algorithms for Multicast With Intra-Session Network Coding. In: *IEEE Transactions on Information Theory* 55 (2009), Nr. 2, S. 797–815

[HY08] HE, Xiang ; YENER, Aylin: Providing Secrecy with Lattice Codes. In: *46th Annual Allerton Conference on Communication, Control, and Computing* (2008), S. 1199–1206

[JLC07] JAIN, Kamal ; LOVÁSZ, László ; CHOU, Philip A.: Building Scalable and Robust Peer-to-Peer Overlay Networks for Broadcasting using Network Coding. In: *Distributed Computing* 19 (2007), Nr. 4, S. 301–311

[JLHE05] JAGGI, Sidharth ; LANGBERGT, Michael ; HO, Tracey ; EFFROS, Michelle: Correction of Adversarial Errors in Networks. In: *Proceedings. International Symposium on Information Theory (ISIT)* (2005), S. 1455–1459

[JLK+08] JAGGI, Sidharth ; LANGBERG, Michael ; KATTI, Sachin ; HO, Tracey ; KATABI, Dina ; MÉDARD, Muriel ; EFFROS, Michelle: Resilient Network Coding in the Presence of Byzantine Adversaries. In: *IEEE Transactions on Information Theory* 54 (2008), Nr. 6, S. 2596–2603

[KFM04] KROHN, Maxwell N. ; FREEDMAN, Michael J. ; MAZIÈRES, David: On-the-Fly Verification of Rateless Erasure Codes for Efficient Content Distribution. In: *Proceedings – IEEE Symposium on Security and Privacy* (2004), S. 226–239

[KHH+13] KRIGSLUND, Jeppe ; HANSEN, Jonas ; HUNDEBØLL, Martin ; LUCANI, Daniel E. ; FITZEK, Frank H. P.: CORE: COPE with MORE in Wireless Meshed Networks. In: *Vehicular Technology Conference* (2013), S. 1–6

[Kim15] KIM, Young S.: Comments on "An Efficient Homomorphic MAC with Small Key Size for Authentication in Network Coding". In: *IEEE Transactions on Computers* 64 (2015), Nr. 12, S. 3619–3620

[KK08] KOETTER, Ralf ; KSCHISCHANG, Frank: Coding for Errors and Erasures in Random Network Coding. In: *IEEE International Symposium on Information Theory (ISIT)* 54 (2008), Nr. 8, S. 3579–3591

[KL09] KEHDI, Elias ; LI, Baochun: Null Keys: Limiting Malicious Attacks Via Null Space Properties of Network Coding. In: *Proceedings – IEEE INFOCOM* (2009), S. 1224–1232

[KM03] KOETTER, Ralf ; MÉDARD, Muriel: An Algebraic Approach to Network Coding. In: *IEEE/ACM Transactions on Networking* 11 (2003), Nr. 5, S. 782–795

[KRH+06] KATTI, Sachin ; RAHUL, Hariharan ; HU, Wenjun ; KATABI, Dina ; MÉDARD, Muriel ; CROWCROFT, Jon: XORs in The Air: Practical Wireless Network Coding. In: *SIGCOMM* 16 (2006), Nr. 3, S. 497–510

[LCL06] LI, Qiming ; CHIU, Dah-ming ; LUI, John C.: On the Practical and Security Issues of Batch Content Distribution Via Network Coding. In: *Proceedings of the IEEE International Conference on Network Protocols* (2006), S. 158–167

[LLC12] LI, Qiming ; LUI, John C. S. ; CHIU, Dah-Ming: On the Security and Efficiency of Content Distribution via Network Coding. In: *IEEE Transactions on Dependable and Secure Computing* 9 (2012), Nr. 2, S. 211–221

[LLL+14] LI, Chen ; LU, Rongxing ; LI, Hui ; CHEN, Le ; LI, Xiaoqing: Comment on "A Novel Homomorphic MAC Scheme for Authentication in Network Coding". In: *IEEE Communications Letters* 18 (2014), Nr. 12, S. 2129–2132

[LM12] LE, Anh ; MARKOPOULOU, Athina: Cooperative Defense Against Pollution Attacks in Network Coding Using SpaceMac. In: *IEEE Journal on Selected Areas in Communications* 30 (2012), Nr. 2, S. 442–449

[LMB07] LIMA, Luísa ; MÉDARD, Muriel ; BARROS, João: Random Linear Network Coding: A free cipher? In: *IEEE International Symposium on Information Theory (ISIT)* (2007), Nr. 2, S. 546–550

[LPHF14] LUCANI, Daniel E. ; PEDERSEN, Morten V. ; HEIDE, Janus ; FITZEK, Frank H. P.: Coping with the Upcoming Heterogeneity in 5G Communications and Storage Using Fulcrum Network Codes. In: *11th International Symposium on Wireless Communications Systems, ISWCS – Proceedings* (2014), S. 997–1001

[LVBM08] LIMA, Luísa ; VILELA, João P. ; BARROS, João ; MÉDARD, Muriel: An Information-Theoretic Cryptanalysis of Network Coding – is Protecting the Code Enough? In: *International Symposium on Information Theory and its Applications* (2008), S. 7–10

[LYC03] LI, Shuo Y. R. ; YEUNG, Raymond W. ; CAI, Ning: Linear Network Coding. In: *IEEE Transactions on Information Theory* 49 (2003), Nr. 2, S. 371–381

[LYC+10] LI, Yaping ; YAO, Hongyi ; CHEN, Minghua ; JAGGI, Sidharth ; ROSEN, Alon: RIPPLE Authentication for Network Coding. In: *Proceedings – IEEE INFOCOM* (2010), S. 1–9

[MPB+17] MATTHIESEN, Bho ; PFENNIG, Stefan ; BIELERT, Mario ; ILSCHE, Thomas ; LONNSTROM, Andrew ; LI, Tao ; CABRERA, Juan A. ; SCHEUNERT, Christian ; FRANZ, Elke ; SANTINI, Silvia ; STRUFE, Thorsten ; JORSWIECK, Eduard A. ; NAGEL, Wolfgang E. ;

NGUYEN, Giang T. ; FITZEK, Frank H.: Secure and Energy-Efficient Interconnects for Board-to-Board Communication. In: *Proc. of Conference on Ubiquitous Wireless Broadband (ICUWB)*, 2017, S. 1–7

[NG08] NAZER, Bobak ; GASTPAR, Michael: Compute-and-Forward: Error-Correcting Codes for Wireless Network Coding on the Physical Layer. In: *5th IEEE Annual Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks Workshops* (2008), S. 1–5

[NL08] NUTMAN, Leah ; LANGBERG, Michael: Adversarial Models and Resilient Schemes for Network Coding. In: *IEEE International Symposium on Information Theory – Proceedings* (2008), S. 171–175

[ns-17] NS-3 DEVELOPERS: *ns-3 network simulator*. 2017. – `http://www.nsnam.org`

[PCTS02] PERRIG, Adrian ; CANETTI, Ran ; TYGAR, Doug ; SONG, Dawn: The TESLA Broadcast Authentication Protocol. In: *CryptoBytes* 5 (2002), Nr. 2, S. 2–13

[PF13a] PFENNIG, Stefan ; FRANZ, Elke: Comparison of Different Secure Network Coding Paradigms Concerning Transmission Efficiency. In: *Computer Aided Modeling and Design of Communication Links and Networks (CAMAD), 2013 IEEE 18th International Workshop on* IEEE, 2013, S. 18–22

[PF13b] PFENNIG, Stefan ; FRANZ, Elke: Secure Network Coding: Dependency of Efficiency on Network Topology. In: *IEEE International Conference on Communications (ICC)* IEEE, 2013, S. 2100–2105

[PF14] PFENNIG, Stefan ; FRANZ, Elke: Adjustable Redundancy for Secure Network Coding in a Unicast Scenario. In: *Proc. of International Symposium on Network Coding*, 2014, S. 1–6

[PF16] PFENNIG, Stefan ; FRANZ, Elke: eSPOC: enhanced Secure Practical Network Coding for Better Efficiency and Lower Latency. In: *Proc. of the NetCod 2016, part of the IEEE GLOBECOM*, 2016, S. 1–6

[PF17] PFENNIG, Stefan ; FRANZ, Elke: Security Aspects of Confidential Network Coding. In: *Proc. of IEEE ICC 2017 Communication and Information Systems Security Symposium*, 2017, S. 1–6

[PFC+14] PFENNIG, Stefan ; FRANZ, Elke ; CIORBA, Florina M. ; ILSCHE, Thomas ; NAGEL, Wolfgang E.: Modeling Communication Delays for Network Coding and Routing for Error-Prone Transmission. In: *Proc. of Conf. on Future generation Communication Technologies (FGCT)*, 2014, S. 19–24

[PFC+16] PFENNIG, Stefan ; FELDHOFF, Kim ; CIORBA, Florina M. ; BIELERT, Mario ; FRANZ, Elke ; ILSCHE, Thomas ; REIHER, Tobias ; NAGEL, Wolfgang E.: Simulation Models Verification for Resilient Communication on a Highly Adaptive Energy-Efficient Computer. In: *Proc. of the 24th High Performance Computing Symposium (HPC 2016), part of the 2016 Spring Simulation Multi-Conference, SpringSim*, 2016, S. 1–6

[PHF11] PEDERSEN, Morten V. ; HEIDE, Janus ; FITZEK, Frank H. P.: Kodo: An Open and Research Oriented Network Coding Library. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* (2011), S. 145–152

[PHVF13] PEDERSEN, Morten V. ; HEIDE, Janus ; VINGELMANN, Peter ; FITZEK, Frank H. P.: Network Coding Over The 2^32 - 5 Prime Field. In: *IEEE International Conference on Communications (ICC)* (2013), S. 1515–1520

[RB15] ROY, Rob ; BOMMAKANTI, Venkat: *Odroid-XU4 User Manual*. https://magazine.odroid.com/wp-content/uploads/odroid-xu4-user-manual.pdf, 2015

[RSA78] RIVEST, Ron L. ; SHAMIR, Adi ; ADLEMAN, Leonard: A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. In: *Communications of the ACM* 21 (1978), Nr. 2, S. 120–126

[RSG98] REED, Michael ; SYVERSON, Paul F. ; GOLDSCHLAG, David: Anonymous Connections and Onion Routing. In: *IEEE Symposium on Security and Privacy* 16 (1998), Nr. 4, S. 482–494

[SET03] SANDERS, Peter ; EGNER, Sebastian ; TOLHUIZEN, Ludo: Polynomial Time Algorithms For Network Information Flow. In: *15th ACM Symposium on Parallel Algorithms and Architectures* (2003), S. 286–294

[SFLP14] SIPOS, Márton ; FITZEK, Frank H. P. ; LUCANI, Daniel E. ; PEDERSEN, Morten V.: Distributed Cloud Storage Using Network Coding. In: *Consumer Communications and Networking Conf. (CCNC)* (2014), S. 139–144

[SMR11] SEFEROGLU, Hulya ; MARKOPOULOU, Athina ; RAMAKRISHNAN, Kadangode K.: I2NC: Intra- and Inter-Session Network Coding for Unicast Flows in Wireless Network. In: *Proceedings – IEEE INFOCOM* (2011), S. 1035–1043

[SPL14] SHANG, Tao ; PEI, Hengli ; LIU, Jianwei: Secure Network Coding Based on Lattice Signature. In: *China Communications* 11 (2014), Nr. 1, S. 138–151

[SPLB12] SOUSA-PINTO, Hugo ; LUCANI, Daniel E. ; BARROS, João: Hide and Code: Session Anonymity in Wireless Line Networks with Coded Packets. In: *Information Theory and Applications Workshop, ITA – Conference Proceedings* (2012), S. 262–268

[SSM08] SUNDARARAJAN, Jay K. ; SHAH, Devavrat ; MÉDARD, Muriel: ARQ for Network Coding. In: *IEEE International Symposium on Information Theory – Proceedings* (2008), S. 1651–1655

[SSM+11] SUNDARARAJAN, Jay K. ; SHAH, Devavrat ; MÉDARD, Muriel ; JAKUBCZAK, Szymon ; MITZENMACHER, Michael ; BARROS, João: Network Coding Meets TCP: Theory and Implementation. In: *Proceedings of the IEEE* 99 (2011), Nr. 3, S. 490–512

[The17] THE SAGE DEVELOPERS: *SageMath, the Sage Mathematics Software System (Version 7.5.1)*, 2017. – `http://www.sagemath.org`

[TM06] TAN, Jianlong ; MÉDARD, Muriel: Secure Network Coding with a Cost Criterion. In: *4th International Symposium on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks, WiOpt* (2006), S. 1–6

[VKT15] VATEDKA, Shashank ; KASHYAP, Navin ; THANGARAJ, Andrew: Secure Compute-and-Forward in a Bidirectional Relay. In: *IEEE Transactions on Information Theory* 61 (2015), Nr. 5, S. 2531–2556

[VLB08] VILELA, João P. ; LIMA, Luísa ; BARROS, João: Lightweight Security for Network Coding. In: *IEEE International Conference on Communications* (2008), S. 1750–1754

[Wan10] WANG, Yongge: Insecure "Provably Secure Network Coding" and Homomorphic Authentication Schemes for Network Coding. In: *IACR Eprint archive* (2010), S. 1–18

[WG12] WANG, Xiao ; GUO, Wangmei: A Novel Lightweight Algorithm for Secure Network Coding. In: *Advances in information Sciences and Service Sciences (AISS)* 4 (2012), Nr. 20, S. 675–680

[WLC12] WANG, Boyang ; LI, Hui ; CAO, Jin: An Efficient MAC Scheme for Secure Network Coding with Probabilistic Detection. In: *Frontiers of Computer Science* 6 (2012), Nr. 4, S. 429–441

[WLW+16] WANG, Jin ; LU, Kejie ; WANG, Jianping ; ZHU, Junda ; QIAO, Chunming: ULNC: An Untraceable Linear Network Coding Mechanism for Mobile Devices in Wireless Mesh Networks. In: *IEEE Transactions on Vehicular Technology* 65 (2016), Nr. 9, S. 7621–7633

[WSK10] WANG, Da ; SILVA, Danilo ; KSCHISCHANG, Frank R.: Robust Network Coding in the Presence of Untrusted Nodes. In: *IEEE Transactions on Information Theory* 56 (2010), Nr. 9, S. 4532–4538

[WWW+11] WANG, Jianping J. ; WANG, Jianping J. ; WU, Chuan ; LU, Kejie ; GU, Naijie: Anonymous Communication with Network Coding against Traffic Analysis Attack. In: *Proceedings IEEE INFOCOM* (2011), S. 1008–1016

[WXL12] WAN, Zhiguo ; XING, Kai ; LIU, Yunhao: Priv-Code: Preserving Privacy Against Traffic Analysis through Network Coding for Multihop Wireless Networks. In: *Proceedings IEEE INFOCOM* (2012), S. 73–81

[Wyn75] WYNER, Aaron D.: The Wire-Tap Channel. In: *Bell System Technical Journal* 54 (1975), Nr. 8, S. 1355–1387

[YA09] YANG, Jing ; AN, Jianping: Combined Fountain Code with Network Coding for Error-Tolerant Transmission Network. In: *Proceedings – 5th International Conference on Wireless Communications, Networking and Mobile Computing, WiCOM* (2009), S. 4–7

[ZJL+10] ZHANG, Peng ; JIANG, Yixin ; LIN, Chuang ; FAN, Yanfei ; SHEN, Xuemin: P-Coding: Secure Network Coding against Eavesdropping Attacks. In: *Proc. IEEE INFOCOM*, 2010, S. 1–9

[ZKMH07] ZHAO, Fang ; KALKER, Ton ; MÉDARD, Muriel ; HAN, Keesook J.: Signatures for Content Distribution with Network Coding. In: *IEEE International Symposium on Information Theory* (2007), S. 556–560

[ZLJ+12] ZHANG, Peng ; LIN, Chuang ; JIANG, Yixin ; LEE, Patrick P. C. ; LUI, John C. S.: ANOC: Anonymous Network-Coding-Based Communication with Efficient Cooperation. In: *IEEE Journal on Selected Areas in Communications* 30 (2012), Nr. 9, S. 1738–1745

[ZLJ+14] ZHANG, Peng ; LIN, Chuang ; JIANG, Yixin ; FAN, Yanfei ; SHEN, Xuemin: A Lightweight Encryption Scheme for Network-Coded Mobile Ad Hoc Networks. In: *IEEE Transactions on Parallel and Distributed Systems* 25 (2014), Nr. 9, S. 2211–2221

[ZLL06] ZHANG, Shengli ; LIEW, Soung-Chang ; LAM, Patrick P.: Physical-Layer Network Coding. In: *Proceedings of the Annual International Conference on Mobile Computing and Networking (MobiCom)* (2006), S. 358–365

Abbildungsverzeichnis

Tabellenverzeichnis

Danksagung

Im Folgenden möchte ich mich bei den Personen bedanken, die mich während der Bearbeitung des Dissertationsthemas unterstützt haben und ohne die diese Arbeit nicht möglich gewesen wäre.
Zuallererst möchte ich mich bei Prof. Dr. Thorsten Strufe, meinem betreuenden Hochschullehrer, bedanken. Seine unkomplizierte Art und seine Ideen ermöglichten es Probleme schnell überwinden. Weiterhin danke ich für die vielen Freiheiten, die er mir während der Bearbeitung gelassen hat.
Auch Prof. Dr. Frank Fitzek möchte ich danken, dass dieser als Zweitbetreuer sowie als Fachreferent meiner Arbeit fungiert. Die Zusammenarbeit mit ihm und seinem Team bereicherte diese Arbeit inhaltlich und ermöglichte neue Sichtweisen.
Ebenso bedanke ich mich bei Prof. Dr. Falko Dressler von der Universität Paderborn dafür, dass er sich so schnell bereit erklärt hat, diese Arbeit als externer Gutachter zu betreuen.
Mein größter Dank gebührt Dr. Elke Franz, die mir seit über 6 Jahren in unserer gemeinsamen Arbeit zur Seite steht und die mich stets bei neuen Ideen und Untersuchungen unterstützt hat. So hat Sie auch am Entstehen dieser Arbeit – neben ihren wertvollen Tipps beim Schreiben und Korrekturlesen – einen immensen Anteil.
Weiterhin möchte ich auch allen Mitarbeitern am Lehrstuhl für Datenschutz und Datensicherheit für die Anregungen und Gespräche danken. Speziell Dr. Dagmar Schönfeld danke ich für das Korrekturlesen dieser Arbeit.
Des Weiteren danke ich dem kompletten HAEC-Team für meine Projektstelle, die es mir ermöglichte, mein Thema zu bearbeiten und diese Arbeit zu erstellen. Zusätzlich bedanke ich mich für die tollen Kooperationen mit anderen Projektgruppen, speziell innerhalb der Architekturgruppe.
Zu guter Letzt gilt auch meiner gesamten Familie ein großer Dank. Insbesondere seien hier meine Frau, die mir in allen Situationen den Rücken freigehalten hat, und meine Kinder, welche auch manchmal auf Zeit mit mir verzichten mussten, erwähnt, da ich nur dadurch meiner Arbeit nachgehen konnte.