View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by Qucosa - Publikationsserver der Universität Leipzig

UNIVERSITÄT LEIPZIG
Fakultät für Mathematik und Informatik
(Institut für Informatik)

Diplomarbeit

Dining Cryptographers Networks

Praktikable Werkzeuge zur Wahrung der Anonymität?

Leipzig, März 2003

vorgelegt von

Heiko Stamer
geb. am: 09. 12. 1977

Studiengang Informatik

Kurzzusammenfassung

Die vorliegende Arbeit beschäftigt sich mit einer speziellen Technik zur Wahrung von Anonymität in Rechnernetzen, den sogenannten *„Dining Cryptographers Networks“ (DC-Netze)*. Diese Netze haben im Vergleich zu anderen anonymisierenden Mechanismen die besondere Eigenschaft, daß ihre Sicherheit komplexitätstheoretisch mit nur wenigen Annahmen bewiesen werden kann. Eine Realisierung erfolgt dabei durch kryptographische Verfahren, welche in bestimmten Angriffsszenarien sogar die stärkere Forderung nach Robustheit des Anonymisierungsdienstes garantieren.

Neben der theoretischen Betrachtung dieser Algorithmen und Protokolle soll gleichzeitig eine freie prototypische Implementierung (`GnuDCN`) bereitgestellt werden, um die praktische Relevanz von DC-Netzen zu klären. Insbesondere kann somit deren Nachrichten- und Übertragungskomplexität mit der anderer Anonymisierungsverfahren verglichen werden, ohne dabei nur auf die üblichen komplexitätstheoretischen Abschätzungen angewiesen zu sein. Außerdem steht hier erstmals ein System zur Verfügung, das sich durch freie Verwend- und öffentliche Verifizierbarkeit seiner Quellen auszeichnet. Deshalb wird auch die allgemeine Bedeutung zugänglicher Quelltexte als Mittel einer Qualitätssicherung kritischer Softwarekomponenten verstärkt herausgestellt.

Die Messung und Analyse von `GnuDCN` zeigt leider deutlich, daß seine enorme Kommunikationskomplexität dem breiten Einsatz in der Praxis noch entgegensteht. Trotzdem ist das System in besonders sensiblen Umgebungen mit kleinem Teilnehmerkreis durchaus verwendbar, gerade weil es momentan eine der sichersten Methoden zur Wahrung der Anonymität darstellt.

Danksagung

Ich möchte hiermit allen Personen, die mich bei dieser Diplomarbeit moralisch und logistisch unterstützt haben, meinen Dank aussprechen. Ein besonderer Dank gilt Herrn Dr. Joachim Apel, der mir als Betreuer mit vielen Hinweisen und Gesprächen jederzeit zur Seite stand. Nicht vergessen möchte ich meine Eltern, die mir das Studium – und damit auch diese Arbeit – überhaupt erst ermöglichten. Schließlich danke ich auch meinen Kommilitonen Ingmar Bauer, Arnold Hagert und Frank-Michael Schleif für unsere konstruktiven Diskussionen und ihre Anregungen.

Weiterhin sage ich ‚Dankeschön' für das hervorragende Textsatzsystem TEX (D. E. Knuth), das Makropaket LATEX (L. Lamport) sowie die KOMA-Scriptklassen (F. Neukam, M. Kohm), welche meinen Schreibaufwand wesentlich verringerten.

Inhaltsverzeichnis

1. Einleitung **7**

1.1. Freiheitsrechte und informelle Selbstbestimmung 9

1.2. Gefahren und Risiken der Anonymität 11

1.3. Formen von Anonymität 12

1.4. Anonymität in einer vernetzten Gesellschaft 15

2. Kryptographische Grundlagen **19**

2.1. Informations- und Komplexitätstheorie 19

2.2. Kryptographie mit öffentlichem Schlüssel 25

2.3. Techniken für Authentizität 30

2.4. Anonymität durch Kryptographie 34

2.4.1. MIX-Netze 35

2.4.2. DC-Netze 38

2.5. Kryptographie mit mehreren Parteien 42

2.5.1. Geheimnisteilung 42

2.5.2. Schwellenwertverfahren 43

2.6. Signaturschemata für mehrere Unterzeichner 45

2.6.1. RSA-ähnliche Schwellenwertsignaturen 47

2.7. Münzwurfschemata für mehrere Beteiligte 54

2.8. Sonstige kryptographische Werkzeuge 55

2.8.1. Sichere Festlegschemata für Bitsequenzen 55

2.8.2. Generatoren für sichere Pseudozufallsbitsequenzen 56

3. Zuverlässige Verteilung **59**

3.1. Byzantinische Übereinstimmung 60

3.1.1. Synchrone Kommunikationsnetze 61

3.1.2. Asynchrone Kommunikationsnetze 62

3.2. Probabilistische Übereinstimmungsprotokolle 63

3.2.1. ABBA Protokoll 63
3.2.2. Ausschlußprotokoll für bösartige Teilnehmer 74
3.3. Konsistente Nachrichtenverteilung 76

4. Dining Cryptographers Networks 81
4.1. Zugriff auf den additiven Kanal 84
4.1.1. Kollisionserkennung 84
4.1.2. Reservierung von Sendeplätzen 85
4.2. Erhaltung der Robustheit bei Unbeobachtbarkeit 86
4.2.1. Ursprüngliches Fallenprotokoll von David Chaum 87
4.2.2. Verbesserungen von Waidner und Pfitzmann 88

5. Implementierung und Resultate 93
5.1. `GnuDCN` — eine robuste DC-Netz Implementierung 93
5.1.1. Designziele und konzeptioneller Aufbau 94
5.1.2. Umsetzung kryptographischer Grundlagen 97
5.1.3. Implementierung der Schichtprotokolle 98
5.2. `GnuDCN` — Messungen und Komplexitätsabschätzungen 101
5.3. `GnuDCN` — Bewertung der Praktikablilität 104

6. Zusammenfassung und Ausblick 105

A. Schnittstellen kryptographischer Verfahren 106
A.1. `ShoupRsaDTSS`-Klasse 106
A.2. `SchneierHashBCS`-Klasse 108
A.3. `BlumBlumShubPRBG`-Klasse 109
A.4. `CachinKursaweShoupABBA`-Klasse 111

Literatur 114

Software 118

Abkürzungen und mathematische Notationen

$O(f(n))$	Klasse von Funktionen, die *nicht schneller* als f wachsen
$\Theta(f(n))$	Klasse von Funktionen, die *gleich schnell* wie f wachsen
$\mathbb{P}$	Menge aller Primzahlen, $\{2, 3, 5, 7, 11, 13, 17, 19, 23, \ldots\}$
n	Eindeutige Darstellung, $n := p_1^{e_1} \cdot p_2^{e_2} \cdots p_k^{e_k}$, $p_i \in \mathbb{P}$, $e_i \geq 1$, $k \geq 1$
$\mathrm{ggT}(a, n)$	Größter gemeinsamer Teiler der Zahlen a und n
$\mathbb{Z}_n$	Natürliche Zahlen kleiner n, $\{0, \ldots, n-1\}$
$\mathbb{Z}_n^*$	Multiplikative Gruppe modulo n, $\{a \in \mathbb{Z}_n \mid \mathrm{ggT}(a, n) = 1\}$
$\varphi(n)$	Eulersche Funktion, $\varphi(n) := (p_1 - 1)p_1^{e_1-1} \cdots (p_k - 1)p_k^{e_k-1}$
$\lvert\mathbb{Z}_n^*\rvert$	Kardinalität der Menge, Gruppenordnung $\lvert\mathbb{Z}_n^*\rvert = \varphi(n)$
$\mathbb{QR}_n$	Quadratische Reste, $\{a \in \mathbb{Z}_n^* \mid \exists x \in \mathbb{Z}_n \ : \ x^2 \equiv a \pmod{n}\}$
$\mathbb{NQR}_n$	Quadratische Nicht-Reste, $\mathbb{Z}_n^* \setminus \mathbb{QR}_n$
$(\frac{a}{p})$	Legendre-Symbol, falls $a \in \mathbb{Z}$ und ungerades $p \in \mathbb{P}$

$$\left(\frac{a}{p}\right) := \begin{cases} 0, & \text{falls } \mathrm{ggT}(a, p) \neq 1 \\ 1, & \text{falls } a \in \mathbb{QR}_p \\ -1, & \text{falls } a \in \mathbb{NQR}_p \end{cases}$$

$(\frac{a}{n})$	Jacobi-Symbol, falls $a \in \mathbb{Z}$, $n = p_1^{e_1} \cdot p_2^{e_2} \cdots p_k^{e_k}$, $p_i \in \mathbb{P}$ ungerade

$$\left(\frac{a}{n}\right) := \left(\frac{a}{p_1}\right)^{e_1} \cdot \left(\frac{a}{p_2}\right)^{e_2} \cdots \left(\frac{a}{p_k}\right)^{e_k}$$

$\mathbb{J}_n$	Zahlen aus $\mathbb{Z}_n^*$ mit postivem Jacobi-Symbol, $\{a \in \mathbb{Z}_n^* \mid (\frac{a}{n}) = 1\}$
$\Pr[A]$	Wahrscheinlichkeit, daß A zutrifft
$\mathrm{GF}(2^n)$	Galois-Feld, endlicher Körper der Charakteristik 2, Ordnung 2^n
$\vec{a}$	Vektor über einem Vektorraum
$\oplus$	Exklusiv-Oder-Verknüpfung (XOR), Addition modulo 2
$\otimes$	Vektorprodukt, Kreuzprodukt
d. h.	das heißt
etc.	et cetera
ggf.	gegebenenfalls
i. S. d.	im Sinne der/des
o. B. d. A.	ohne Beschränkung der Allgemeinheit
u. U.	unter Umständen
z. B.	zum Beispiel
z. Z.	zur Zeit

1. Einleitung

Wer anonym schreibt und polemisiert,
hat eo ipso *die Präsumption gegen*
sich, daß er das Publikum betrügen,
oder ungefährdet anderer Ehre antasten will.
— Arthur Schopenhauer

Die vorliegende Arbeit beschäftigt sich mit einer speziellen Technik zur Wahrung von Anonymität in Rechnernetzen, den sogenannten *„Dining Cryptographers Networks“ (DC-Netze)*. Diese Netze haben, im Vergleich zu anderen anonymisierenden Mechanismen, die besondere Eigenschaft, daß ihre Sicherheit komplexitätstheoretisch mit nur wenigen Annahmen bewiesen werden kann. Eine Realisierung erfolgt dabei durch kryptographische Verfahren, welche in bestimmten Angriffsszenarien sogar die stärkere Forderung nach Robustheit garantieren.

Neben der theoretischen Betrachtung dieser Algorithmen und Protokolle soll gleichzeitig eine freie prototypische Implementierung (`GnuDCN`) bereitgestellt werden, um die praktische Relevanz von DC-Netzen zu klären. Insbesondere kann somit deren Nachrichten- und Übertragungskomplexität mit der anderer Anonymisierungsverfahren verglichen werden, ohne dabei nur auf die üblichen komplexitätstheoretischen Abschätzungen angewiesen zu sein. Außerdem steht damit erstmals ein System zur Verfügung, das sich durch freie Verwend- und öffentliche Verifizierbarkeit seiner Quellen auszeichnet.

Aufbau und Gliederung: Im ersten Abschnitt wird Anonymität im allgemeinen Kontext dargestellt. Die Betrachtung erfaßt ihre Bedeutung in einer modernen Gesellschaft, zeigt aber auch Gefahren und Risiken. Später soll an markanten Beispielen die zunehmende Notwendigkeit anonymisierender Mechanismen in unserer ‚digitalen und vernetzten Welt‘ skizziert werden.

Der zweite Abschnitt erarbeitet schrittweise Grundlagen, welche wir für eine sichere Wahrung der Anonymität benötigen. Zuerst wird kurz in die Kryptographie eingeführt und auch ihr komplexitätstheoretischer Hintergrund beleuchtet. Auf der geschaffenen Basis sollen dann zwei Techniken für Senderanonymität besprochen werden, wobei wir natürlich DC-Netze fokusieren. Eine kurze Analyse zeigt allerdings, daß es zu deren praktischer Umsetzung noch diverser Hilfmittel bedarf. Unser Schwerpunkt ist deshalb ein noch relativ neues kryptographisches Schema für sichere RSA-Schwellenwertsignaturen, um schließlich das Protokoll des folgenden Abschnitts zu implementieren.

Zuverlässige Verteilung ist in DC-Netzen eine zwingend notwendige Bedingung für Anonymität. Im dritten Kapitel widmen wir uns deswegen dieser Problematik, welche schon seit dem Altertum bekannt ist. Ausgehend von Lösungen

zur Byzantinischen Übereinstimmung soll deren Realisierung in synchronen und asynchronen Kommunikationsnetzen untersucht werden. Schließlich wird ein erst im Jahre 2000 von C. Cachin, K. Kursawe und V. Shoup beschriebenes Protokoll (Asynchronous Binary Byzantine Agreement) vorgestellt, welches unsere Forderungen nach Robustheit und Authentizität nahezu optimal erfüllt.

Gewappnet mit den Werkzeugen der zwei vorangegangenen Abschnitte können wir nun die Realisierung des eigentlichen „Dining Cryptographers Protocol“ besprechen und es für eine allgemeine Auslegung als anonymisierendes DC-Netz charakterisieren. Im Mittelpunkt steht dabei auch wieder die Robustheit[1] dieses Protokolls, welche durch B. Pfitzmanns und M. Waidners Erweiterungen des Originalverfahrens erreicht werden kann.

Die Titelfrage „Dining Cryptographers Networks — Praktikable Werkzeuge zur Wahrung der Anonymität?“ versucht das vorletzte Kapitel zu beantworten. Darin werden neben einigen Details zur Implementierung auch Messungen und Vergleiche vorgestellt. Diese Daten zeigen recht anschaulich, daß DC-Netze im praktischen Einsatz leider noch zu hohe Komplexitätsanforderungen stellen. Gleichzeitig sind sie jedoch die z. Z. sicherste Methode zur Wahrung der Anonymität und deshalb in besonders kritischen Umgebungen durchaus einsatzwürdig. Vorschläge zur Kombination mit herkömmlichen Verfahren runden den Hauptteil ab.

Im schließenden Abschnitt sollen behandelte Thematik sowie gewonnene Erkenntnisse möglichst kurz zusammengefasst und ein etwas längerer Ausblick auf zukünftige Entwicklungen – im Umfeld dieses speziellen Bereichs der Kommunikationssicherheit – gegeben werden.

Der Anhang zeigt neben Schnittstellendefinitionen der kryptographischen Klassen auch ein Abbildungs- und Tabellenverzeichnis. Die elektronische Version dieser Arbeit (Postscript-Format) und alle Quelltexte meiner prototypischen Implementierung `GnuDCN` sind hingegen auf einer beiliegenden CD-ROM enthalten.

[1] Schutz vor ‚lahm legendem Mißbrauch‘ (Denial of Service)

Der Begriff *Anonymität* ist allgemein etwas schwierig zu definieren. *Webster's Revised Unabridged Dictionary* [1] gibt dazu sinngemäß folgendes an: „Anonymität ist die Qualität oder der Zustand anonym, d. h. ohne Bekanntgabe eines Namens bzw. einer Identität, zu sein.“ Anders formuliert könnte man sich darunter also vorstellen, daß Objekte bzw. Subjekte innerhalb einer gegebenen Anonymitätsmenge nicht identifizierbar sind. Jene können demzufolge Handlungen ausführen, Informationen veröffentlichen oder an Prozessen teilnehmen, ohne daß damit in Zusammenhang stehende Daten, Aktionen oder Resultate auf ein konkretes Element dieser Menge hindeuten. Intuitiv betrachtet, sollte die spezielle Adjungierbarkeit[2] sogar eine Gleichverteilung i. S. d. Wahrscheinlichkeitstheorie darstellen, damit keinerlei Informationen über das betroffene Objekt/Subjekt entweichen.

Anonymität ist also scheinbar sehr eng mit dem Begriff der *Unzuordbarkeit* verbunden. Diese Annahme werden wir später (siehe Unterabschnitt 1.3) noch formalisieren, denn sie führt uns zu besonderen Formen von Anonymität. Vorerst wollen wir jedoch einen Blick auf ihre allgemeine Bedeutung als staatsbürgerliches Grundrecht werfen, ohne dabei ihre Gefahren oder Risiken zu verschweigen.

1.1. Freiheitsrechte und informelle Selbstbestimmung

Sich frei und unbeobachtet in bestimmten öffentlichen oder privaten Räumen bewegen zu können, ist für uns Menschen nahezu natürlich. Beim Einkaufen oder an der Wahlurne gehen wir davon aus, daß unsere (Kauf-)Entscheidung nicht mit unserer Identität verknüpft werden kann und wir damit vor unlauterer Werbung bzw. persönlichen Repressionen geschützt sind. Im Beichtstuhl oder beim Arztbesuch hoffen wir auf die Verschwiegenheit des Gegenübers hinsichtlich der anvertrauten Information. Anonymität ist in einer demokratischen Gesellschaft also offenbar allgegenwärtig und so selbstverständlich, daß wir sie kaum intensiv wahrnehmen oder gar speziell einfordern.

Vielleicht ist diese Tatsache auch der Grund, weshalb man in neuen Bereichen des gesellschaftlichen Lebens[3] mittlerweile unheimlich ‚spendabel‘ persönliche Daten (Name, Adresse, Alter, Telefonnummer, Gewohnheiten, Vorlieben, ...) offenbart. Oder ist es die juristische Absicherung, welche uns die zweifelhafte Gewissheit gibt, vor Mißbrauch weitreichend geschützt zu sein? In den meisten westeuropäischen Ländern ist nämlich, im Gegensatz zu den Vereinigten Staaten, der Umgang mit sensiblen personengebundenen Daten gesetzlich geregelt. Deutschland hat seit geraumer Zeit Datenschutzgesetze und auch Gremien, die deren Einhaltung überwachen sollen. Doch nehmen die Bürger ihre Rechte und die Beteiligten ihre Pflichten überhaupt wahr?

[2] Zuordbarkeit genau eines Elements der Anonymitätsmenge

[3] z. B. Talk- bzw. Realityshows im Fernsehen, Kundenbefragungen oder Internetnutzung

Helmut Bäumler, der Landesdatenschutzbeauftragte von Schleswig-Holstein, stellte in seiner Rede zur Eröffnung der Sommerakademie 2002 [2] vollkommen zu Recht fest, daß der Anonymitätsanspruch unserer Bürger einem Wandel unterliegt. Nicht immer steht heutzutage Diskretion im Vordergrund, wenn es um die Medienberichterstattung zu intimen oder persönlichen Dingen geht. Viele Betroffene opfern ihre Anonymität, um sich aus der breiten Masse hervorzuheben. Eine möglichst schrille Selbstdarstellung offeriert ja zumindest kurzweilige Prominenz und verspricht dadurch ggf. Einnahmequellen.

Schwierig wird die Situation, wenn sich – als Folge obiger Entwicklung – zukünftig Menschen rechtfertigen müssen, die *nicht* beobachtet oder befragt werden möchten. Spätestens dann wäre das *Recht auf informelle Selbstbestimmung*[4] ernsthaft gefährdet. Aufgrund dieser drohenden Problematik sollte man die Bürger also bewußt auffordern, häufiger von ihren Grundrechten Gebrauch zu machen und sie u. U. sogar einzuklagen.

Anhand dreier verschiedener Beispiele möchte ich an diesem Punkt die Bedeutung der Anonymität für eine moderne Gesellschaft nocheinmal klar herausstellen:

Geheime Wahlen: Über wichtige Personen oder Fragen wird in einem Entscheidungsprozeß oft geheim abgestimmt. Zum einem soll damit natürlich die allgemeine Vertraulichkeit gewährleistet werden, zum anderem möchte man die Beeinflußung der Entscheidungsträger aus dem privaten oder staatlichen Umfeld vermindern. Da abgegebene Stimmen nicht mehr einer konkreten Identität[5] zugeordnet werden können, bleiben auch persönliche Repressionen oder Zwangsmaßnahmen unadressierbar. Desweiteren kann der Wähler nur sehr umständlich einem Dritten beweisen, für wen oder was er gestimmt hat. Dadurch wird der *Käuflichkeit von Stimmen* begegnet. Für die Demokratie eines Staates sind geheime Wahlen gewissermaßen ‚das Salz in der Suppe', denn nur so gestaltet sich die Auswahl der Volksvertreter wirklich nach den Wünschen der Bürger und nicht etwa nach öffentlichen oder privaten Zwängen.

Anonymes Bezahlen: Bei Bezahlung mit Bargeld kann man hinreichend anonym einen Gegenstand oder eine Dienstleistung erwerben. Der Käufer kann damit verhindern, daß seine Konsumgewohnheiten oder sein Lebensstil der Allgemeinheit zugänglich wird. Der Verkäufer kann kein persönliches Kundenprofil anlegen, um etwaige Vorlieben zum eigenen Vorteil auszunutzen. Im Zuge einer

[4] Dieses Recht zählt seit dem Volkszählungsurteil des Bundesverfassungsgerichts mit zum allgemeinen Persönlichkeitsrecht gemäß Artikel 2 des GG. Die Richter haben damit eine wegweisende Entscheidung getroffen, da im beginnenden Zeitalter der Informationsgesellschaft auch weitgehende Grundrechte hinsichtlich des Umgangs mit persönlichen Informationen notwendig sind, um ein autonomes und selbstbestimmtes Leben zu gewährleisten.

[5] allenfalls einer bestimmten Trägermenge

wirtschaftlichen Wertschöpfungskette dient diese Anonymität auch der Wahrung von Geschäftsgeheimnissen (z. B. Rezepturen oder Prozeßabläufe).

Anonyme Post: Briefe und Pakete können auch ohne Absenderadresse zugestellt werden. Sie bieten damit eine besondere Art der Anonymität, auf die wir später noch genauer eingehen werden. Briefwahl oder anonyme Kundenbefragungen sind nützliche Anwendungsgebiete dieser Dienstleistung. Leider gibt es auch Mißbrauch, z. B. in Form von Drohungen/Erpressungen oder sogar Briefbomben. Noch vor kurzem haben biologisch kontaminierte Sendungen die nordamerikanische Bevölkerung in Angst und Schrecken versetzt.

1.2. Gefahren und Risiken der Anonymität

Die im letzten Beispiel angedeuteten Probleme zeigen deutlich, daß Anonymität immer differenziert zu betrachten ist. Wie jedes andere Freiheitsrecht hat sie Licht- und Schattenseiten. Straftäter dürfen sich nicht mit ihrer Hilfe dem staatlichen Zugriff entziehen können. Allerdings sollte man bedenken, daß auch unter dem Deckmantel anderer Freiheitsrechte, wie z. B. der Bewegungsfreiheit oder dem freien Erwerb von Waffen[6], Verbrechen verübt werden. Trotzdem schränkt der Staat diese Freiheiten nur sehr maßvoll ein; so sind beispielsweise nur das Betreten bestimmter Gebiete/Gebäude bzw. der Besitz besonders gefährlicher Waffen verboten. Eine moderne Gesellschaft muß deshalb immer einen gerechten Kompromiß zwischen Freiheit und Sicherheit treffen, welcher oft einer ‚schwierigen Gratwanderung mit gelegentlichen Fehltritten' entspricht.

Vollkommen andere Gefahren durch Anonymität spricht H.-G. Marcieniec [3] an, die ich allerdings nur bedingt teilen kann. Für ihn hat das Unerkanntsein rein subjektive Züge, z. B. um sich Verbindlichkeiten zu entziehen oder in anderen Menschen „Furcht oder Ärger zu erzeugen". Natürlich beinhaltet seine These, daß „Anonymität ein Mangel an Courage zum Vertrauen" darstellt, einige Wahrheiten, allerdings ist es – meiner Meinung nach – grundsätzlich falsch, die Form des Unerkanntseins von vornherein zu verteufeln. Eine offene Gesellschaft sollte diese Art der Ausdrucksmöglichkeit auch tolerieren, denn selbst Marcieniec stellt richtigerweise fest, daß „nur aus dem Offenen etwas zu entstehen vermag".

Ich möchte diese Diskussion mit einem Zitat von F. Sieberg abschließen, welches den Kern der Sache hervorragend trifft: „Der volle Genuß der Freiheit schließt ihren Mißbrauch mit ein."

[6] in manchen Ländern Schußwaffen oder sogar die vielzitierten „Massenvernichtungswaffen"

1.3. Formen von Anonymität

Wir wollen nun die Begriffsbildung für spätere Abschnitte präzisieren und stützen uns dabei auf die Vorschläge von [4]. Dabei müssen wir allerdings ein konkretes Umfeld festlegen, das durch Abstraktion aber durchaus variabel hinsichtlich seiner Anwendung bleibt:

Grundlage soll also im folgenden ein Kommunikationsnetzwerk sein, in dem Subjekte als Sender oder Empfänger von Nachrichten agieren. Wir behalten dabei immer einen möglichen Angreifer im Auge, welcher an Informationen über Beziehungen zwischen einzelnen Subjekten und Nachrichten interessiert ist. Weiterhin wird angenommen, daß das Wissen dieses imaginären Angreifers immer nur zunehmen kann, er aber nicht in der Lage ist, vom Inhalt der Nachrichten auf Absender oder Empfänger zu schließen. In der Praxis ist letzteres durch eine Verschlüsselung aller Nachrichten erreichbar.

Anonymität der Subjekte läßt sich in einer derartigen Umgebung genau dann definieren, wenn diese äußerlich über die selben Eigenschaften verfügen, so daß eine triviale Unterscheidung unmöglich wird.

Definition 1.1 (Anonymität)
Anonymität ist der Zustand der Unidentifizierbarkeit eines Einzelnen (Initiator) innerhalb einer Menge von Subjekten, der sogenannten Anonymitätsmenge.

Die Anonymitätsmenge enthält dabei alle möglichen Subjekte, welche zu diesem Zeitpunkt, wie der Initiator, eine bestimmte Handlung ausführen, eine Nachricht senden oder empfangen könnten. Die Stärke der Anonymität hängt von der Größe dieser Menge und der Gleichverteilung der beobachteten Ereignisse bzgl. aller Teilnehmer ab. Wie schon in der Einleitung angedeutet, spielt auch die *Unzuordbarkeit* zwischen Einträgen[7] eine wichtige Rolle.

Definition 1.2 (Unzuordbarkeit)
Unzuordbarkeit zwischen zwei oder mehreren Einträgen bedeutet, daß sie nicht mehr oder weniger in Beziehung zueinander stehen, als durch eine a-priori Wahrscheinlichkeit von vornherein bekannt ist.

Unzuordbarkeit ist also genau dann gewährleistet, wenn das a-priori Wissen über einen Sachverhalt dem a-posteriori Wissen entspricht, welches ein Angreifer nach Beobachtung des ausführenden Systems besitzt. Beispielsweise sollte nach einer erfolgten Beobachtung die Wahrscheinlichkeit, daß zwei bestimmte Nachrichten vom gleichen Sender stammen, sich nicht von der vornherein bekannten a-priori Wahrscheinlichkeit unterscheiden.

[7] Subjekte (Empfänger, Absender), Objekte (Nachrichten) oder Ereignisse (Aktionen)

Wenn wir Anonymität unter dem Gesichtspunkt der Unzuordbarkeit betrachten, ist für unseren Angreifer interessant, welches Subjekt eine bestimmte Nachricht sendet oder empfängt. Es ergeben sich deshalb folgende Anonymitätsformen:

Empfängeranonymität (recipient anonymity) herrscht, falls *keine* Nachricht einem empfangenden Subjekt zugeordnet werden kann. Folglich entspricht die Wahrscheinlichkeit dieser Zuordnung einer Gleichverteilung über alle möglichen Empfänger.

Senderanonymität (sender anonymity) herrscht, falls *keine* Nachricht einem sendenden Subjekt zugeordnet werden kann. Folglich entspricht auch hier die Wahrscheinlichkeit einer speziellen Zuordnung der Gleichverteilung über alle möglichen Sender.

Anonymität von Verbindungen (relationship anonymity) herrscht, falls *keine* Nachricht einem Sender- und Empfängersubjekt[8] zugeordnet werden kann. Diese Form ist schwächer als die beiden vorherigen, denn die Definition schließt nicht aus, daß entweder Sender *oder* Empfänger doch ermittelbar sind. Daraus darf allerdings nicht hervorgehen, welche zwei (oder mehr) Subjekte eine gegenseitige Nachrichtenverbindung errichtet haben.

Eine Schlüsselstelle scheint hier die Beobachtung des Systems durch den Angreifer zu sein. Um letztendlich obige Formen der Anonymität gewährleisten zu können, wünschen wir uns deshalb soetwas wie *Unbeobachtbarkeit*. Sie hätte gleichzeitig die stärkere Eigenschaft, daß nicht nur die Beziehung zwischen den Einträgen, sondern auch ihr Inhalt selbst geschützt wäre.

Definition 1.3 (Unbeobachtbarkeit)
Unbeobachtbarkeit ist der Zustand von Einträgen, welcher sie für potentielle Beobachter ununterscheidbar werden läßt.

Für Objekte/Nachrichten bedeutet diese Definition, daß jene gerade nicht vom ‚zufälligen Rauschen' zu unterscheiden sind. Analog zu den Formen der Anonymität können wir darauf aufbauend nun einführen:

Empfängerunbeobachtbarkeit (recipient unobservability) ist gegeben, falls von einem Außenstehenden *nicht* bemerkt werden kann, wann ein Subjekt der Beobachtungsmenge Nachrichten sendet.

Senderunbeobachtbarkeit (sender unobservability) ist gegeben, falls von einem Außenstehenden *nicht* bemerkt werden kann, wann ein Subjekt der Beobachtungsmenge Nachrichten empfängt.

Unbeobachtbarkeit von Verbindungen (relationship unobservability) ist gegeben, falls von Außenstehenden *nicht* bemerkt werden kann, daß Nachrichten von einer Menge möglicher Sender zu einer Menge möglicher Empfänger übertragen werden. Diese Form erlaubt die Beobachtung eines Ereignisses (Senden *oder* Empfangen); es darf jedoch daraus nicht hervorgehen, welche Subjekte eine gegenseitige Nachrichtenverbindung errichtet haben.

[8] bei Streuverteilung ggf. auch mehreren Empfängersubjekten (multicast)

Bleibt der Angreifertyp gleich, sollte Unbeobachtbarkeit höchstens eine echte Teilmenge der Informationen offenbaren, welche Anonymität dargelegt hätte. Wir können also implizieren, daß aus Unbeobachtbarkeit stets Anonymität folgt. Die Umkehrung dieser Aussage gilt natürlich nicht, denn selbst eine anonym gesendet und empfangene Nachricht deutet auf Kommunikation zwischen Subjekten hin.

Tabelle 1.1 (Beziehungen zwischen Begriffen)

Unbeobachtbarkeit	$\Rightarrow$	*Anonymität*
Empfängerunbeobachtbarkeit	$\Rightarrow$	*Empfängeranonymität*
Senderunbeobachtbarkeit	$\Rightarrow$	*Senderanonymität*
Unbeobachtbarkeit von Verbindungen	$\Rightarrow$	*Anonymität von Verbindungen*
Empfängeranonymität	$\Rightarrow$	*Anonymität von Verbindungen*
Senderanonymität	$\Rightarrow$	*Anonymität von Verbindungen*
Empfängerunbeobachtbarkeit	$\Rightarrow$	*Unbeobachtbarkeit von Verbindungen*
Senderunbeobachtbarkeit	$\Rightarrow$	*Unbeobachtbarkeit von Verbindungen*

Pseudonymität: Pseudonyme sind Bezeichner für Subjekte, also in unserem Fall für Sender oder Empfänger. Das konkrete Subjekt, das unter diesem Namen auftritt, ist der sogenannte *Pseudonymhalter*. Unter Pseudonymität verstehen wir nun (gemäß [4]) die Benutzung von Pseudonymen als Unterscheidungsmerkmal.

Digitale Pseudonyme sind i. A. beliebige Zeichenketten, die

1. innerhalb einer Unterscheidungsmenge eindeutig sind und
2. zur Authentifizierung von gesendeten Nachrichten taugen.

Damit kann eine gewisse ‚Zählbarkeit' (accountability) erreicht werden, wie sie in manchen Systemen – trotz gebotener Anonymität – gefordert wird. Hinsichtlich der Zuordnungsfähigkeit zwischen dem Pseudonym und seinem Halter lassen sich folgende Klassen unterscheiden:

(a) **Öffentliches Pseudonym (public pseudonym)**
Die Identität des Pseudonymhalters ist öffentlich und von Anfang an bekannt, z. B. öffentliche Verzeichnisse, Telefonbücher.

(b) **Nichtöffentliches Pseudonym (initially non-public pseudonym)**
Die Identität des Pseudonymhalters ist nur bestimmten Personen/Institutionen von Anfang an bekannt. Während der Benutzung erfahren aber ggf. Dritte diese Zuordnung, z. B. Bankkonten.

(c) **Unzuordbares Pseudonym (initially unlinked pseudonym)**
Die Identität des Pseudonymhalters ist zumindest am Anfang unbekannt. Eine

Ausnahme hiervon stellt natürlich der Halter selbst dar, da er meist die Zuordnung kennt oder zumindest leicht in Erfahrung bringen kann, z. B. biometrische Daten, DNA-Sequenzen.

Anonymität ist in solchen Umgebungen umso stärker, je weniger über die Zuordnung zwischen Pseudonym und Halter bekannt ist. Ergo hängt sie hier wesentlich vom Wissen der beteiligten Parteien und vom gewählten Angriffsmodell ab.

1.4. Anonymität in einer vernetzten Gesellschaft

Ich möchte in diesem Abschnitt keine ‚Schwarzmalerei' betreiben, allerdings sollen hier ungeschönt Entwicklungen und Technologien vorgestellt werden, die Anonymität in zunehmendem Maße gefährden. Die Beispiele stammen aus einem Telepolis-Buch [5] und haben die Aufgabe, für Unerkanntheit zu sensibilisieren:

Videoüberwachung: Anonymität ist in unserer mittlerweile weit vernetzten und damit leicht beobachtbaren Welt erheblich bedroht. Ein noch relativ harmloses Beispiel sind dabei die diversen Formen von Videoüberwachung. Ob auf öffentlichen Plätzen, an Geldautomaten oder aber auch im privaten Bereich durch installierte Webcams — wir werden auf Schritt und Tritt beobachtet. Vorreiter bei dieser zumindest zweifelhaften Technik ist Großbritannien, das Land mit der dichtesten Ausstattung an Überwachungskameras. Im Mai 1999 waren nach Aussage des britischen Innenministers ca. 1 Million Geräte installiert.

An sich stellen diese ‚künstlichen Augen' noch kein großes Problem dar, da aufgenommene Bilddaten meist in einer ‚rohen Form' vorliegen, welche nur schwer – also auch erst bei schwerwiegenden Vorfällen – analysiert und konkreten Individuen zugeordnet werden können. Gerade aber die schon im vorherigen Abschnitt besprochene Zuordnungsfähigkeit wird durch moderne Verfahren der Gesichts- oder Merkmalserkennung zunehmend erleichtert. Die Briten sind hier so weit fortgeschritten, daß sie ‚verdächtige Personen' sogar unter einer Verkleidung (Brillen, falsche Bärte, ...) erkennen können. Die neueste Errungenschaft sind intelligente Kameras, die Terroristen oder Autodiebe anhand ihres Verhaltens identifizieren sollen. Das von der *University of Leeds* entwickelte System ‚lernt', normales von verdächtigem Verhalten zu unterscheiden, und schlägt im Bedarfsfall Alarm. Da niemand ‚unangenehm' auffallen möchte, werden wir unser Verhalten langsam anpassen. In Zukunft ist also Vorsicht geboten, wenn man auf dem Parkplatz sein Fahrzeug nicht gleich findet und erst in Richtung eines anderen PKW läuft. Aber Sie haben sicherlich schon eine stichhaltige Erklärung bereit, um sich für den Fehlalarm zu rechtfertigen?!

Solche Systeme lassen sich natürlich auch am Arbeitsplatz einsetzen und sollen angeblich die Produktivität der Mitarbeiter steigern. Auf jeden Fall steigern sie die Gewinnmargen der Hersteller und die Paranoia der Bevölkerung.

Totale Überwachung (Echelon): Das von der NSA [6] und verbündeten Geheimdiensten betriebene Spionagenetzwerk *Echelon* ist mittlerweile der Öffentlichkeit bekannt. Es hat vornehmlich die Aufgabe satellitengestützte Kommunikation abzuhören und wurde nach dem Zweiten Weltkrieg installiert, um der ‚Roten Gefahr' auf informationstechnischer Ebene entgegenzutreten.

Die anfallenden Datenmengen sind enorm. Deshalb sind Techniken im Einsatz [7], welche wort- oder themenbasierte Zuordnungen finden. Abgehörte Informationen werden nicht nur zur diplomatischen bzw. militärischen Aufklärung genutzt, sondern auch immer häufiger, um sich wirtschaftliche Vorteile[9] zu verschaffen.

Die leitungsgebundene Kommunikation (insbesondere durch Lichtwellenleiter-Technologie) macht es den ‚offenen Ohren' der Geheimdienste heutzutage immer schwieriger, interessante Daten in Erfahrung zu bringen. Digitalisierung und Verschlüsselung bereiten den Nachrichtendiensten ernsthafte Probleme, die sich in den 80er Jahren in Exportbeschränkungen für kryptographische Algorithmen und Geräte äußerten. Mittlerweile hat man sich allerdings wieder von dem Gedanken verabschiedet, den Einsatz von Kryptographie weltweit kontrollieren zu können. Als Gegenmaßnahme versucht die NSA, durch eigene Forschungen und fragwürdige Standardisierungen[10] sich einen Wissens- bzw. Berechnungsvorsprung zu erhalten. Immerhin soll jener sagenumwobene Geheimdienst der weltweit größte Arbeitgeber von Mathematikern sein.

KfZ-Kennzeichen Detektion: Eine besondere Überwachungsmaßnahme hat unlängst Bayerns Innenminister Beckstein stolz der Öffentlichkeit vorgestellt. An mehreren Grenzübergängen zur Tschechischen Republik wurden Kameras eingerichtet, welche die Kennzeichen aller passierenden Kraftfahrzeuge lesen und mit einer Datenbank der Polizeibehörden vergleichen. Auffällige Autos werden den Zollbeamten angezeigt und können damit verstärkt kontrolliert werden. Datenschutzrechtliche Bedenken wegen der flächendeckenden Überwachung des Reiseverkehrs wies Beckstein mit dem Hinweis zurück, daß notfalls halt die gesetzlichen Grundlagen geändert werden müssen. Außerdem sollen die Daten ‚unbedenklicher Fahrzeughalter' umgehend nach dem Passieren gelöscht werden. Schade eigentlich, denn deutsche Versicherungskonzerne, die Werbeindustrie oder Reiseveranstalter würden sich bestimmt auch darüber freuen.

[9] Regierungskreise sprechen natürlich eher von Abwendung ‚wirtschaftlicher Nachteile'.

[10] Beispiel: Der DSS (Digital Signature Standard) war ursprünglich nur für eine Modullänge von 512 Bit ausgelegt. Erst nach erheblicher öffentlicher Kritik wurde sie auf 1024 Bit erhöht. [8]

Staubige Aussichten (Smart Dust): Kameras und Mikrophone werden zunehmend kleiner und damit unauffällig einsetzbar. Ein bedrohliches Szenario für die Privatheit stellt eine faszinierende Entwicklung der Berkeley-Universität dar: Amerikanischen Wissenschaftlern ist es gelungen, digitale Schaltkreise, Laserbasierte drahtlose Kommunikation, autonome Energieversorgung und MEMS[11]-Drehspiegel in einer Einheit von 5 Millimeter Länge zusammenzufügen. Ziel ist es ferner, diese Komponenten noch auf ein Raumvolumen von ein bis zwei Kubikmillimeter zu miniaturisieren.

Die Herausforderung besteht nun darin, diese im Aufklärungsraum verteilten ‚Staubkörner' mit einer Basisstation kommunizieren zu lassen. Distanzen von 150 Metern können mit der Datenrate von bis zu einem Kilobit pro Sekunde überbrückt werden. Es wurden auch schon Strecken bis zu 20 Kilometer erfolgreich getestet, allerdings fällt die Übertragungsrate dabei rapide ab.

Solche Miniatureinheiten könnten beispielsweise mit Mikrophonen oder elektromagnetischen Sensoren ausgestattet werden. Eine Überwachung kompromittierender Signale, welche z. B. bei Benutzung der Computertastatur erzeugt werden, scheint dadurch möglich. Inbesondere läßt sich dieser gefährliche Staub mit bloßem Auge nicht von harmlosem unterscheiden. Auch die militärische Anwendung zur Überwachung von Truppenbewegungen in feindlichen Ländern ist einfach zu bewerkstelligen, denn die Spionageeinheiten lassen sich leicht durch unbemannte Dronen oder Sprühflugzeuge im Zielgebiet verteilen.

Moderne Kommunikationsnetze (Internet): Digitale Netzwerke sind prinzipiell Datenerhebungsmaschinen von bislang unvorstellbarem Ausmaß. Jede Art von Aktivität kann mit relativ geringem Aufwand verfolgt, belauscht und aufgezeichnet werden. Jeder Benutzer hinterläßt Spuren im Netz, welche auch noch Monate später Personen oder Gruppen zugeordnet werden können.

Neben ‚Großen Brüdern'[12] tummeln sich im weltumspannenden Internet auch viele ‚Kleine Schwestern'. Konzerne, Organisationen und auch neugierige Einzelpersonen nutzen die anfallenden Daten, um kommerzielle oder persönliche Vorteile zu erlangen. Ein Beispiel hierfür sind sogenannte Bulkmail-Versender, die Millionen von elektronischen Briefkästen mit unerwünschter Werbepost überfluten. Weitaus bedrohlicher sind aber unlautere Betreiber von Webportalen, Einkaufssystemen oder Suchmaschinen. Sie verknüpfen Eingaben der Kunden mit zielgerichteter Werbung und können durch geeignete Identifizierungsmerkmale[13] sogar

[11] micro electromechanical systems

[12] Big Brother: Der Ausdruck stammt aus „1984", dem früher surreal erscheinenden Roman von George Orwell, und symbolisiert einen authoritären Überwachungsstaat. Heutzutage sind etliche der enthaltenen Visionen leider schon bittere Realität geworden.

[13] z. B. ‚Cookies' der WWW-Browser

Benutzerprofile [9] erstellen. Ein Verkauf der gewonnenen Daten an zwielichtige Unternehmen ist dabei nicht immer ausgeschlossen.

Hauptproblem ist die enorme Verdichtung persönlicher Informationen, welche das Internet durch etliche Dienste (WWW, Email, Diskussiongruppen, Einkaufs- oder Communityportale, Chatsysteme) bereitwillig preisgibt. Im kommerziellen Umfeld wird dem Benutzer zwar suggeriert, daß seine vertraulichen Angaben vor den Augen dritter Personen geschützt sind, allerdings halten die dafür verwendeten Schutzmechanismen häufig keiner seriösen Überprüfung stand. Aber schon allein die öffentlich zugänglichen Informationen bieten Neugierigen ein erschreckendes Potential. Moderne Suchmaschinen wie *GOOGLE* verknüpfen herkömmliche Webseiten, Bilder, Dokumente und Diskussionsgruppenbeiträge mit beliebigen Stichworten. Arbeitgeber könnten sich so schon im Vorfeld eine genaue Vorstellung vom Kandidaten machen und bei Mißfallen der persönlichen Meinung[14] gar nicht erst zum Gespräch einladen. Es ist davon auszugehen, daß sich die angesprochene Problematik in den nächsten Jahren weiter verstärkt; insbesondere weil Kindern und Jugendlichen zwar der Umgang mit dem Internet beigebracht, aber nur selten dessen Gefahren vermittelt werden.

Hingegen existieren kaum wirksame Werkzeuge zur Wahrung der Anonymität. Und selbst wenn sie leicht benutzbar sind, hat sich in der Bevölkerung noch lange nicht die Einstellung durchgesetzt, sie auch gelegentlich zu verwenden.

[14] z. B. durch ‚unschickliche' Beiträge in der Diskussionsgruppe `de.soc.menschenrechte`

2. Kryptographische Grundlagen

Es genügt nicht, zum Fluß zu kommen
mit dem Wunsche, Fische zu fangen,
man muß auch ein passendes Netz mitbringen.
— Chinesisches Sprichwort

Etwas antiquiert betrachtet, ist Kryptographie die Wissenschaft der Verschlüsselung. Sie war jahrhundertelang hauptsächlich militär- oder diplomatischen Kreisen vorbehalten und ist erst seit den 70er Jahren einer breiteren Öffentlichkeit zugänglich. Mittlerweile haben sich zur Geheimhaltung von Informationen auch modernere Aufgaben gesellt, wie z.B. Gewährleistung der Authentizität[15] von Nachrichten, Sicherstellung der Fairness für Mehrparteienprotokolle und Wahrung der Anonymität von Subjekten.

Trotz ihrer Vorteile gab es wiederholt staatliche Bestrebungen, welche den Einsatz von Kryptographie beschränken oder sogar verbieten wollten. Zuletzt forderte der damalige Bundesinnenminister Manfred Kanther [10] im April 1997, daß Informationsquellen der Sicherheitsbehörden nicht durch zunehmende Verschlüsselung versiegen dürften. Eine Schlüsselhinterlegung, um Abhörmaßnahmen der Strafverfolgungsbehörden und Geheimdienste zu gewährleisten, schien unausweichlich. Die Bevölkerung antwortete mit Protestkampagnen im Internet. Zur Zeit hat sich die Regierung wieder von solchen unrealistischen Forderungen gelöst; das Bundeswirtschaftministerium fördert seit kurzem sogar die Entwicklung des `GNU Privacy Guard`[16], welcher für ‚Jedermann' verläßliche Verschlüsselung und digitale Signatur bereitstellt. Mit Blick auf unsere aktuelle ‚Sicherheitsgesetzgebung zur Terrorabwehr' sollte jeder kryptographiebesorgte Bürger die Situation auch weiter kritisch beobachten.

Die nun folgende Einführung soll wesentliche Grundlagen und Konzepte aus dem Gebiet der Kryptographie erschließen, auf die wir später häufig zurückgreifen müssen. Unser Schwerpunkt liegt vorerst auf Verfahren mit öffentlichem Schlüssel (*Public-Key-Algorithmen*), ohne die moderne Kommunikationssicherheit undenkbar wäre. Schließlich werden auch Möglichkeiten zur Wahrung der Anonymität aufgezeigt, um den Bezug zum ursächlichen Thema deutlich zu machen.

2.1. Informations- und Komplexitätstheorie

Beginnen wollen wir mit einem Ausflug in die theoretische Informatik, welche u.a. mit Hilfe der Komplexitätsanalyse Algorithmen und Probleme hinsichtlich ihres Verhaltens auf Digitalrechnern zu klassifizieren vermag.

[15] bestehend aus Identifikation (Herkunftsnachweis) und Integrität (Unveränderbarkeit)
[16] `http://www.gnupg.de/`

Komplexität von Algorithmen: Neben der Frage, ob ein Problem überhaupt auf derzeitigen Computern lösbar[17] ist, interessiert uns auch der dafür notwendige Berechnungsaufwand. Die Komplexität eines Lösungsalgorithmus wird dabei hauptsächlich von zwei Variablen bestimmt:

1. Rechenzeit
2. Speicherplatzbedarf

Um allgemeine Aussagen zu erhalten, werden beide Ressourcen gewöhnlich als Funktionen der Eingabegröße n betrachtet. Deren asymptotisches Wachstum kann mit der sogenannten O-Notation beschrieben werden:

Definition 2.1 (O-Notation)
Wir bezeichnen mit $O(f)$ eine Klasse von Funktionen, die **nicht schneller** *als f wachsen, d. h. ein Algorithmus g ist genau dann in $O(f)$, wenn Konstanten $c > 0$ und $n_0 > 0$ mit $\forall n \geq n_0 : g(n) \leq c \cdot f(n)$ existieren. Die Ressourcen werden also nach oben durch die Funktion $f : \mathbb{N} \to \mathbb{R}^+$ beschränkt, wobei wir von konstanten Faktoren und Termen niedrigerer Ordnung absehen.*

Wir wollen später diese Schreibweise auch benutzen, um die Kommunikationskomplexität von Protokollen zu kennzeichnen. Laufzeit und Aufkommen an Ein- und Ausgabedaten wird dann in Abhängigkeit von der Teilnehmerzahl n dargestellt. Für uns sind hierbei spezielle Unterkategorien wichtig:

Rundenkomplexität ist die erwartete Anzahl von Runden eines Protokolls, bevor es die Entscheidung bzw. das Ergebnis erreicht.

Nachrichtenkomplexität gibt die Zahl der gesendeten Nachrichten während der Laufzeit eines Protokolls an.

Bitkomplexität beschreibt die totale Länge der gesendeten Nachrichten in Bit.

In der Kryptographie werden Algorithmen meist nur hinsichtlich ihrer Zeitkomplexität klassifiziert, denn der Speicherplatzbedarf spielt nur eine untergeordnete Rolle. Folgende Tabelle zeigt gebräuchliche Bezeichnungen und einige Beispiele aus der Informatik, wobei $m > 1$ eine Konstante und $p(n)$ ein Polynom ist.

Tabelle 2.1 (Bezeichnungen und Beispiele zur O-Notation)

O-Notation	*Bezeichnung*	*Beispiel*
$O(1)$	*konstant*	*(unabhängig von der Eingabegröße)*
$O(\log n)$	*logarithmisch*	*binäre Suche*
$O(n)$	*linear*	*sequentielle Suche*
$O(n \log n)$	*linear logarithmisch*	*Sortieren mit Merge- oder Radixsort*
$O(p(n))$	*polynomial*	*Matrixmultiplikation*
$O(m^{p(n)})$	*exponentiell*	*Travelling Salesman Problem (TSP)*

[17] Stichwort: Entscheidbarkeit von Eigenschaften, z. B. Halteproblem bei Turing-Maschinen

Oft hängt Berechnungskomplexität inhärent mit den verwendeten Datenstrukturen zusammen. Deshalb darf man bei der Konstruktion effizienter Verfahren die Repräsentation der Daten nie außer Acht lassen. Wir werden im Abschnitt 5 (Implementierung und Resultate) noch darauf zurückkommen.

Komplexität von Problemen: Nicht nur Algorithmen, sondern auch die zugehörigen Probleme können in abstrakte Klassen eingeteilt werden. Dabei untersucht man jeweils den schwierigsten Einzelfall auf einem theoretischen Maschinenmodell, der sogenannten *Turing-Maschine*. Diese Apparatur ist ein endlicher Automat mit einem unendlich großen les- und schreibbaren Speicherband. Alle existierenden Rechnerformen werden durch dieses Modell abgedeckt, was allgemeine Schlüsse zur Schwierigkeit von Berechnungen ermöglicht. Innerhalb aller berechenbaren Probleme unterscheidet man folgende Komplexitätsklassen, wobei deren Beziehungen untereinander teilweise noch unklar sind:

$\mathcal{P}$: Probleme, die in polynomialer Zeit auf einer deterministischen Turing-Maschine gelöst werden können, d. h. es existieren dafür Lösungsalgorithmen mit der Zeitkomplexität $O(p(n))$.

Wir nennen solche Verfahren, obwohl sie durch große Konstanten oder Polynome hoher Ordnung noch weit von der tatsächlichen Lösbarkeit entfernt liegen könnten, trotzdem *effizient*, denn wir hoffen, polynomiales Wachstum mit technischem Fortschritt[18] in den Griff zu bekommen.

$\mathcal{NP}$: Probleme, die in polynomialer Zeit auf einer nichtdeterministischen Turing-Maschine gelöst werden können.

Ein nichtdeterministischer Automat verzweigt bei einem Rechenschritt in mehrere Zustände, die alle gleichberechtigt behandelt werden. Beispiele für nichtdeterministische Turing-Maschinen, die schwierige Probleme in polynomialer Laufzeit lösen, existieren momentan noch nicht. Allerdings darf daraus nicht etwa geschlossen werden, daß es sie nie geben wird.[19]

Die Klasse $\mathcal{NP}$ schließt offensichtlich $\mathcal{P}$ mit ein. Bis heute ist unklar, ob nicht doch alle Probleme aus $\mathcal{NP}$ auch in polynomialer Zeit auf herkömmlichen – sprich deterministischen – Turing-Maschinen lösbar sind. Es wurden zwar noch keine effizienten Algorithmen für $\mathcal{NP}$-vollständige Probleme gefunden, so daß viele Wissenschaftler die Gültigkeit der These $\mathcal{P} = \mathcal{NP}$ bezweifeln — ein entgültiger Beweis ist das allerdings nicht.

$\mathcal{NP}$-vollständig: Ein Problem $X \in \mathcal{NP}$ heißt $\mathcal{NP}$-vollständig, falls alle anderen Probleme aus $\mathcal{NP}$ *nicht schwieriger*[20] als X sind.

[18] Das Gesetz von Moore prophezeit, daß sich aller 1½ Jahre die Rechenleistung verdoppelt.
[19] Stichwort: Quantencomputer, analoge Berechnungsmethoden (Super-Turing-Maschinen [11])
[20] Wir werden diese informelle Schreibweise gleich durch Definition einer Relation präzisieren.

$\mathcal{P}$-SPACE: Probleme dieser Klasse können mit polynomialem Speicherplatz gelöst werden, nicht notwendigerweise aber in polynomialer Zeit.

Natürlich enthält $\mathcal{P}$-SPACE alle Probleme aus $\mathcal{NP}$. Aber auch hier ist unbekannt, ob es eine echte Schnittmenge gibt, d. h. Probleme, die auf einer nichtdeterministischen Turing-Maschine in Polynomialzeit unlösbar sind.

Analog zu $\mathcal{NP}$-vollständigen gibt es auch $\mathcal{P}$-SPACE-vollständige Probleme.

EXPTIME: Diese Klasse umfasst alle Probleme, die in exponentieller Zeit gelöst werden können. Trivialerweise enthält EXPTIME auch $\mathcal{P}$ und $\mathcal{NP}$.

Als Beispiel der Existenz solcher äußerst schwierigen Fragen sei die Entscheidbarkeit der *Presburger Arithmetik* angegeben. Dieses Problem kann nicht in Polynomialzeit gelöst werden, egal welches Maschinenmodell man zugrunde legt. Folglich gilt die Beziehung EXPTIME $\cap \mathcal{NP} \neq \emptyset$.

Wie schon angedeutet, wünschen wir uns eine Relation mit der die Schwierigkeit von Problemen vergleichbar wird:

Definition 2.2 ($\leq_{\mathcal{P}}$-Relation)
Ein Problem X ist **nicht schwieriger** *als Y (Kurzschreibweise: $X \leq_{\mathcal{P}} Y$), falls ein deterministischer Polynomialzeitalgorithmus existiert, welcher beliebige Aufgaben von X mit Hilfe eines hypothetischen Verfahrens für Y lösen kann.*

Umgangssprachlich formuliert: Der geforderte Algorithmus ‚übersetzt' beliebige Aufgabenstellungen für X ins Format von Y und wandelt erhaltene Ergebnisse wieder in adäquate Lösungen um. Dabei ist ersteinmal unerheblich, ob für Y Lösungsverfahren existieren und welche Komplexität sie haben.

Definition 2.3 ($\equiv_{\mathcal{P}}$-Relation)
Zwei Probleme sind zeitlich berechnungsäquivalent (Kurzschreibweise: $X \equiv_{\mathcal{P}} Y$), falls $X \leq_{\mathcal{P}} Y$ und $Y \leq_{\mathcal{P}} X$ gilt.

Mit obigen Definitionen sind wir in der Lage, Schlußfolgerungen zur Schwierigkeit von Problemen zu ziehen, selbst wenn wir keinen Lösungsansatz vor Augen haben.

Fakt 2.1 (Polynomialzeitreduktion)
Seien X, Y Probleme mit der Eigenschaft $X \leq_{\mathcal{P}} Y$. Es gelten die Folgerungen,

1. $Y \in \mathcal{P} \Rightarrow X \in \mathcal{P}$ *und*
2. $Y \in \mathcal{NP} \Rightarrow X \in \mathcal{NP}$,

da jeweils X nach Def. 2.2 in Polynomialzeit auf Y reduziert werden kann.

Schwierige zahlentheoretische Probleme: In der Kryptographie greift man häufig auf verschiedene mathematische Problemstellungen zurück, von denen angenommen wird, daß es sich um äußerst schwierige Aufgaben handelt. Obwohl

von ihnen meist nicht die EXPTIME Zugehörigkeit (bzw. die schwächere $\mathcal{NP}$-Vollständigkeit) bewiesen werden kann, sind sie doch für moderne Kryptosysteme unerläßlich geworden. Aufgrund der fehlenden Beweise nennt man solche Probleme schlicht *kryptographische Annahmen.*

Diese schwierigen Probleme werden zur Konstruktion sogenannter ‚Falltürfunktionen' benutzt, welche wiederum erst Kryptographie mit öffentlichem Schlüssel (siehe folgender Abschnitt 2.2) ermöglichen.

Definition 2.4 (Falltürfunktion)
Eine Funktion $f : A \to B$ heißt **Falltürfunktion**, *wenn*

1. *$f(a)$ leicht mit einem Algorithmus E berechenbar ist und*
2. *$f^{-1}(b)$ leicht mit einem Algorithmus D berechenbar ist.*

Es soll jedoch so gut wie unmöglich sein, D aus E zu bestimmen.

Einige für unsere Zwecke relevante kryptographische Annahmen sind:

Primfaktorisierung: Darunter verstehen wir das Problem, zu einer gegebenen positiven ganzen Zahl n ihre paarweise verschiedenen Primfaktoren $p_1, p_2, \cdots, p_k$ aus $\mathbb{P}$ zu finden, wobei $n = p_1^{e_1} \cdot p_2^{e_2} \cdots p_k^{e_k}$ und $e_i \geq 1$ gilt. Es existieren seit kurzem zwar Algorithmen, die in nahezu Polynomialzeit feststellen können, ob eine gegebene Zahl prim ist, allerdings ist die Zerlegung in Faktoren weitaus schwieriger. Momentan haben die schnellsten Algorithmen eine geschätzte Laufzeit [8], die sich proportional zu $e^{(1.9229+o(1))\ln(n)^{1/3}\ln(\ln(n))^{2/3}}$ verhält. Aufgrund des subexponentiellen Komplexitätswachstums dieser Faktorisierungsverfahren scheinen Produkte wie $n = p \cdot q$ mit mindestens 512 Bit langen und zufälligen Primfaktoren derzeit noch unangreifbar[21] zu sein. Das kann sich jedoch sehr schnell ändern, da auf dem Gebiet intensiv geforscht wird und in den letzten Jahren bedeutende Fortschritte bekannt wurden. Deshalb empfehlen einige Kryptographen schon jetzt, nur noch 1024 Bit lange Zahlen für p und q zu verwenden.

Diskreter Logarithmus: Ein anderes schwieriges Problem ist die Berechnung diskreter Logarithmen in endlichen Körpern oder Gruppen $\mathbb{Z}_n$.[22] Wir können in solchen Strukturen[23] Exponential- und Logarithmusfunktion definieren, die zueinander invers sind. Während modulare Exponentiation einfach geht, wurde von der Umkehrung gezeigt, daß damit n leicht faktorisierbar ist, d. h. FAKTOR $\leq_{\mathcal{P}}$ DLOG. Die Zahl n darf auch prim sein ($\mathbb{Z}_n$ Primkörper), muß allerdings immer so gewählt werden, daß die Ordnung von $\mathbb{Z}_n^*$ hinreichend groß ist und $n-1$ nicht nur in kleine Faktoren zerfällt, da man sonst effizient diskrete Logarithmen berechnen kann.

[21] d. h. nur mit Kenntnis von n nicht in beide Bestandteile p und q faktorisierbar

[22] aktuell häufig diskutiertes Beispiel: Gruppe der elliptischen Kurven

[23] Wichtig ist die Existenz einer multiplikativen Gruppe $\mathbb{Z}_n^*$; manche Algorithmen arbeiten dann sogar in Untergruppen $\mathbb{Z}_q^*$, wobei q ein großer Primfaktor von $n-1$ und n selbst prim ist.

Diffie-Hellman-Problem (Computational Diffie-Hellman assumption): Zu einer gegebenen zyklischen Gruppe $\mathbb{Z}_n^*$ mit Erzeuger α sowie den Elementen $\alpha^a \bmod n$ und $\alpha^b \bmod n$ (die Exponenten a und b sind dabei unbekannt) ist es schwierig, eine Zahl z mit $z = \alpha^{ab} \bmod n$ zu finden, wenn man keine diskreten Logarithmen in $\mathbb{Z}_n^*$ berechnen kann, d. h. CDH $\leq_{\mathcal{P}}$ DLOG. Interessanterweise wurde auch die Beziehung FAKTOR $\leq_{\mathcal{P}}$ CDH bewiesen, falls n zusammengesetzt ist.

DDH-Problem (Decisional Diffie-Hellman assumption): Die strengere Annahme besagt, daß es selbst bei Kenntnis der Werte n, α, $\alpha^a \bmod n$ und $\alpha^b \bmod n$ i. A. schwierig ist, die Zahl $\alpha^{ab} \bmod n$ von beliebigen Gruppenelementen $\alpha^c \bmod n$ auseinanderzuhalten, falls a, b und c zufällig gewählt sind. Mit anderen Worten: Es ist hart zu entscheiden, ob die Gleichung $c = ab$ gilt. Offensichtlich kann man DDH $\leq_{\mathcal{P}}$ CDH $\leq_{\mathcal{P}}$ DLOG feststellen, jedoch sind andere Zusammenhänge [12] noch unklar. Überraschenderweise ist das DDH-Problem in einigen endlichen Gruppen, wo CDH hart zu sein scheint, leicht lösbar. Dan Boneh [13] gibt einige Beispiele dazu an, zeigt aber auch, wo DDH vermutlich doch schwer ist. Insbesondere bleibt das Problem kompliziert, wenn die Gruppenordnung von $\mathbb{Z}_n^*$ keine kleinen Primfaktoren besitzt.

RSA-Problem (RSA assumption): Von diesem Problem hängt die Sicherheit des RSA-Algorithmus ab, welchen wir demnächst betrachten. Seien $n = p \cdot q$ (dabei sind p, q verschiedene ungerade – aber unbekannte – Primzahlen), e eine positive ganze Zahl mit $\mathrm{ggT}(e, (p-1)(q-1)) = 1$ und $c \in \mathbb{Z}_n$ beliebig gewählt. Man nimmt an, daß es sehr schwierig ist, ein $m \in \mathbb{Z}_n$ mit $m^e \equiv c \pmod{n}$ zu berechnen. Anders ausgedrückt besteht das Problem darin, e-te Wurzeln modulo n bzw. die Umkehrfunktion für $f(m) = m^e \bmod n$ zu bestimmen. Bis heute scheint das eine schwere Aufgabe zu sein, falls man die Faktorisierung von n nicht kennt. Mit Hilfe der Primzahlen p und q ist es hingegen sehr einfach, die Gruppenordnung $\varphi(n) = (p-1)(q-1)$ und damit auch ein inverses Element $d = e^{-1} \bmod \varphi(n)$ zu berechnen, welches dann $m = c^d \bmod n$ liefert. Deshalb gilt natürlich RSA $\leq_{\mathcal{P}}$ FAKTOR; die Gegenrichtung (folglich Äquivalenz) konnte bis heute allerdings weder gezeigt noch widerlegt werden.

Folgende Tabelle faßt noch einmal die wichtigsten Fragestellungen zusammen:

Tabelle 2.2 (kryptographische Annahmen)

Bezeichner	*Beschreibung*	*gegeben*	*gesucht*
FAKTOR	*Primfaktorisierung*	$n = p \cdot q$	p, q
DLOG	*diskreter Logarithmus*	$\alpha^r \bmod n$, n, α	r
CDH/DDH	*Diffie-Hellman*	$\alpha^a \bmod n$, $\alpha^b \bmod n$, n, α	$\alpha^{ab} \bmod n$
RSA	*Rivest-Shamir-Adleman*	$m^e \bmod n$, e, n	m

Nachdem wir notwendige zahlentheoretischen Probleme relativ ‚trocken' beschrieben haben, soll nun ihre Anwendung in der Kryptographie erläutert werden.

2.2. Kryptographie mit öffentlichem Schlüssel

Ein neues Paradigma führten 1976 die Wissenschaftler W. Diffie und M. Hellman mit dem Konzept der sogenannten *Public-Key-Cryptography* ein. Vorher bekannte Ansätze gingen immer von geheimen, beiderseitig genutzten Schlüsseln aus und wurden deshalb als symmetrische Kryptosysteme bezeichnet. Besondere Schwierigkeiten bereitete dabei der Austausch dieser geheimen Schlüsselinformation, weil meist kein sicherer Kommunikationskanal (persönliches Treffen, Kurierdienst, etc.) dafür zur Verfügung stand. Kryptographie mit öffentlichem Schlüssel löst diese Probleme teilweise und bietet zugleich neue Einsatzmöglichkeiten.

Jeder Kommunikationsteilnehmer mit Namen i verfügt dabei über ein Schlüsselpaar $(K^i_{\mathrm{PUB}}, K^i_{\mathrm{SEC}})$, bestehend aus dem öffentlichen Stück K^i_{PUB} und dem privaten Teil K^i_{SEC}. Letzterer darf nur dem Eigentümer bekannt sein und sollte deshalb besonderen Schutzmechanismen[24] unterliegen. Hingegen wird K^i_{PUB} offen verteilt; — auf die damit verbundenen Probleme gehen wir gleich ein. Möchte nun Alice an Bob eine geheime Nachricht m senden, kodiert sie den Inhalt mit Hilfe von Bobs öffentlichem Schlüssel und erhält $c := E(K^{\mathrm{Bob}}_{\mathrm{PUB}}, m)$. Lediglich Bob ist in der Lage, $m := D(K^{\mathrm{Bob}}_{\mathrm{SEC}}, c)$ wieder herzustellen, und zwar durch Kenntnis seines privaten Schlüsselteils. Hierbei sind E und D Algorithmen für Ver- bzw. Entschlüsselung, welche auf dem Prinzip einer Falltürfunktion (siehe Definition 2.4) basieren.

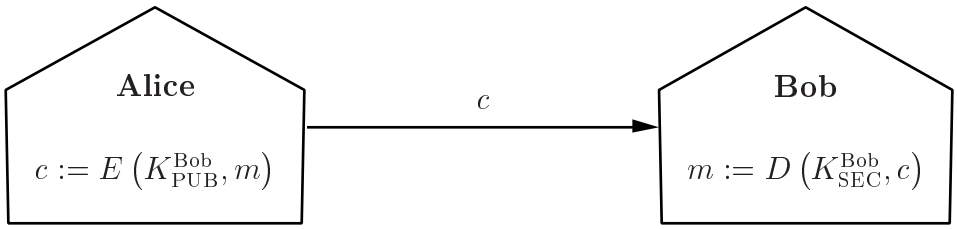

Abbildung 1: Prinzip der asymmetrischen Verschlüsselung

Man kann die Situation mit der am Briefkasten vor der Haustür vergleichen: Jeder Passant ist in der Lage, Sendungen einzuwerfen (entspricht der Verschlüsselung), aber nur der Eigentümer kann den Behälter öffnen (entspricht dem Entschlüsseln), um Nachrichten daraus zu lesen. Bei solchen *asymmetrischen Kryptosystemen* kann ein bösartiger Teilnehmer also beliebige Nachrichten verschlüsseln und so an zusätzliche Informationen gelangen. Die verwendeten Verfahren müssen deshalb gegen einen ‚Angriff mit gewähltem Klartext' (Chosen-Plaintext-Attack) resistent sein. Eine Lösung dieser Problematik bieten *hybride Kryptosysteme*, die hauptsächlich wegen ihrer Geschwindigkeitsvorteile eingesetzt werden. Dabei wird nur ein zufällig gewählter Sitzungsschlüssel K_{RND} mit der aufwendigen

[24] z. B. Speicherung auf einer Chipkarte oder einem verschlüsselten Datenträger

Public-Key-Kryptographie kodiert. Die eigentliche Nachricht m wird dann lediglich mit dem symmetrischen Verfahren S (z. B. Advanced Encryption Standard) verschlüsselt. Abbildung 2 zeigt diesen Vorgang schematisch.

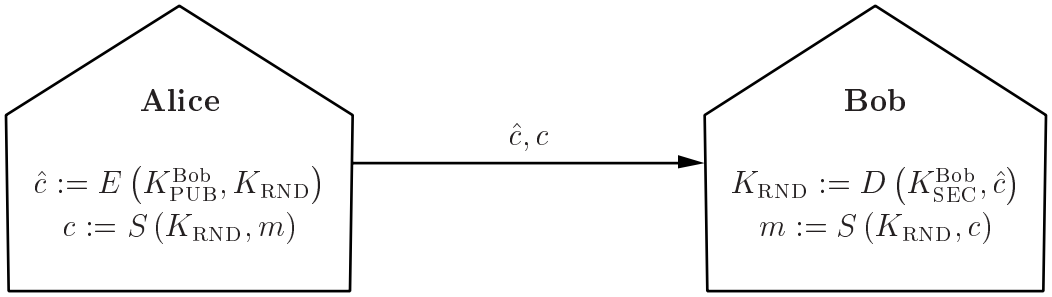

Abbildung 2: Prinzip der hybriden Verschlüsselung

Mit dieser Technik wird die sonst benötigte geheime Schlüsselverteilung symmetrischer Verfahren elegant umgangen. Leider gibt es jetzt eine andere Schwierigkeit: Da jeder nach belieben Schlüsselpaare generieren und verbreiten kann, ist unklar, ob der Schlüssel wirklich der Person gehört, mit welcher wir eigentlich kommunizieren möchten. Durch sogenannte ‚Man-in-the-middle'-Angriffe könnte der böswillige Teilnehmer Mallory die gesamte Korrespondenz problemlos mitlesen. Alice und Bob würden diese Observation nicht einmal bemerken.

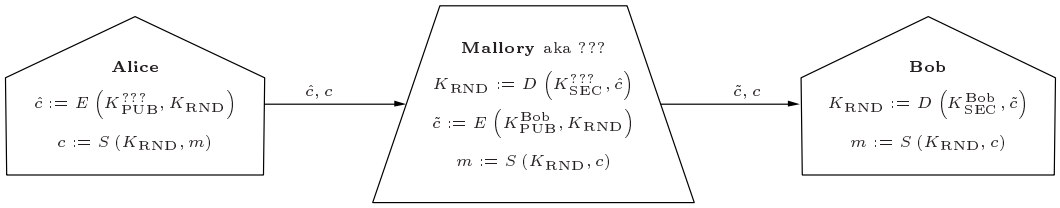

Abbildung 3: ‚Man-in-the-middle'-Angriff

Offensichtlich ist die Authentizität des Schlüsselpaars wichtig. Um diese Eigenschaft sicherzustellen, wurden einige übergreifende Zertifizierungsmodelle entwickelt, die allgemein unter dem englischen Stichwort *Public-Key-Infrastructure* zusammengefaßt werden. Wir wollen nun deren Vor- und Nachteile diskutieren.

Schlüsselinfrastruktur (PKI): Natürlich könnte auch weiterhin der öffentliche Schlüssel durch einem sicheren Kanal (z. B. persönliche Übergabe, ...) ausgetauscht werden, um dessen Authentizität (Zugehörigkeit zu einem Subjekt) zu garantieren. Diese Vorgehensweise ist aber in der Praxis ungeeignet, gerade wenn man mit Personen kommunizieren möchte, die weit entfernt leben oder einem unbekannt sind. Zur Lösung dieses Dilemmas wurden sogenannte *Zertifikate*, also

gewisse vertrauensvolle Bestätigungen (beispielsweise der Eigentümerschaft des Schlüssels), eingeführt, die in zwei verschiedenen Ansätzen zum Einsatz kommen:

Hierarchisches Zertifizierungsmodell: Diese Form ist überall dort anwendbar, wo eine hierarchische Grundstruktur vorliegt. Dies wäre bei Institutionen des Staates oder bei größeren Unternehmen bzw. Organisationen der Fall, die über eine baumartige innere Gliederung verfügen. Abbildung 4 zeigt einen möglichen Aufbau, wo die CA_i sogenannte *Certification Authoritys* und RA_i *Registration Authoritys* darstellen. Die Zertifizierungsstellen überprüfen Schlüssel und Identität eines Teilnehmers. Wenn bestimmte Bedingungen (Certification Policy) erfüllt sind, bestätigen sie die Authentizität durch ein Zertifikat. Dafür zugrundegelegte Regeln (beispielsweise Ausweiskontrolle) sollten für alle CA_i, RA_i gelten, um einen einheitlichen Standard innerhalb der Baumstruktur aufrechtzuerhalten.

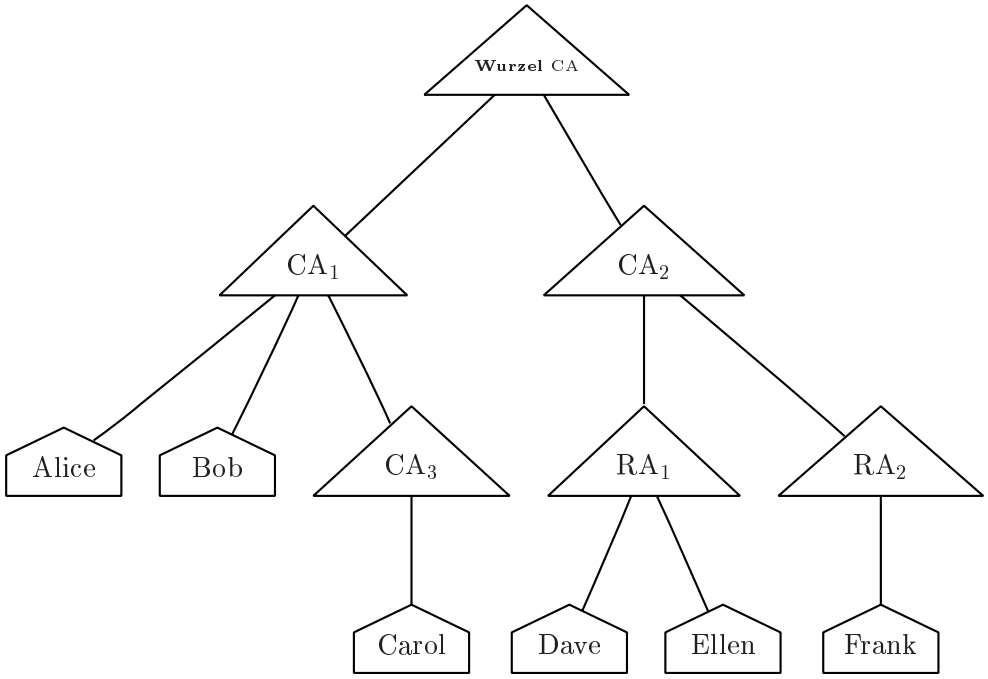

Abbildung 4: hierarchische Zertifizierungsstruktur

Die RA_i sind nur Hilfsinstanzen und stellen selbst keine Zertifikate aus. Damit wird der Kontrollaufwand für übergeordnete CA_i verteilt. Eine wichtige Rolle spielt die *Wurzel* CA, denn ihre Bestätigung authentifiziert die Instanzen CA_1, CA_2 und damit den gesamten Baum. Ihr öffentlicher Schlüssel muß allen Teilnehmern zugänglich sein, z. B. Publizierung im Bundesanzeiger. Dafür dürfen nur ‚schreibgeschützte' Medien verwendet werden, die nicht von außen veränderbar sind. Die CA_1 bestätigt wiederum durch Zertifikate die Authentizität der

Schlüssel von Alice, Bob und der CA_3. Um die Gültigkeit von Bobs Schlüssel zu überprüfen, muß Alice nur dessen CA_1-Zertifikat kontrollieren. Möchte sie hingegen mit Frank kommunizieren, verifiziert sie zuerst das Wurzelzertifikat von CA_2 und kann dann bedenkenlos Franks gültiges CA_2-Zertifikat akzeptieren. Ein Beispiel für Gestaltung hierarchischer Zertifikate ist der Standard `X.509`.

„Web of Trust“: Da eine hierarchische Struktur nicht immer vorhanden ist, wurde in `PGP/GnuPG` das Modell eines *Netz des Vertrauens* eingeführt. Die Grundannahme hierbei ist, daß man anderen Benutzern die Bestätigung von Schlüsseln zutraut, da diese sich durch persönlichen Kontakt oder Identifizierung von der Richtigkeit überzeugt haben könnten. Man kann sein Vetrauen solchen Personen gegenüber durch einen Parameter quantifizieren (volles, marginales oder kein Vertrauen), so daß für die Gültigkeit von Schlüsseln ggf. mehrere Bestätigungen erforderlich sind. Der Abbildung 5 können wir folgende Informationen entnehmen:

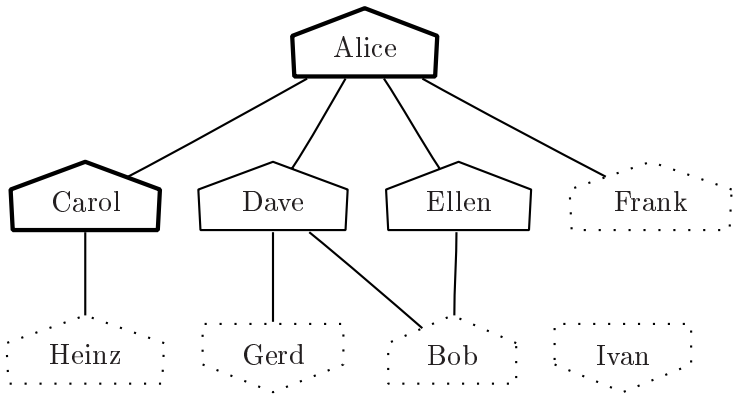

Abbildung 5: Aufbau eines *Netz des Vertrauens*

Alice kennt Carol, Dave, Ellen und Frank persönlich und hält deshalb deren Schlüssel für authentisch. Weiterhin hat sie zu Carol volles, zu Dave bzw. Ellen marginales und zu Frank kein Vertrauen, was die Bestätigung von Schlüsseln Dritter angeht. Heinz ist authentisch, weil er ein Zertifikat von Carol besitzt und Alice volles Vertrauen in deren Identifizierungsfähigkeiten setzt. Da sie Heinz aber nicht kennt, traut sie ihm auch nicht zu, andere Benutzer zu bestätigen. Dave und Ellen haben jeweils Bobs Schlüssel zertifiziert, welcher deshalb für Alice genauso authentisch ist. In beide hat Alice jedoch nur marginales Vertrauen, so daß genau eine Bestätigung nicht ausreichen würde. Dies ist bei Gerd der Fall. Ivan ist niemandem bekannt, weshalb sein Schlüssel ebenfalls ungültig ist.

Da in beiden Modellen Bestätigungen statt direktem Schlüsselaustausch Verwendung finden, liegt keine unbedingte Authentizität mehr vor. Vielmehr wird deren Sicherheit vom Bergiff des *Vertrauens* dominiert. Zertifikate bieten also nur eine gewisse Evidenz, daß der bestätigte Schlüssel auch wirklich authentisch ist.

RSA-Kryptosystem: Ein Algorithmus, der sich sowohl zur Verschlüsselung als auch für digitale Unterschriften (siehe nächster Abschnitt) eignet, wurde 1978 von R. Rivest, A. Shamir und L. Adleman [14] vorgestellt. Das mit den Anfangsbuchstaben seiner Autoren bezeichnete Verfahren ist aufgrund des einfachen Aufbaus sehr bekannt und weit verbreitet. Die Sicherheit basiert auf dem schon beschriebenen RSA-Problem (siehe Seite 24). Wir wollen wesentliche Teile des Systems hier nun kurz skizzieren:

Schlüsselerzeugung: Um das Schlüsselpaar (K_{PUB}, K_{SEC}) zu generieren, wählt man zufällig zwei große Primzahlen p, q und bildet deren Produkt $n := p \cdot q$. Weiterhin sucht man eine Zahl $e \in \mathbb{Z}_n$, die relativ prim (teilerfremd) zu $(p-1)(q-1)$ ist. Das Tupel $(n, e) =: K_{\text{PUB}}$ ist der öffentliche Schlüsselteil. Meist wird e als kleine Primzahl gewählt, damit die Verschlüsselung schneller funktioniert. Ein zu kleiner Wert bietet aber subtile Angriffsmöglichkeiten, weshalb gängige Implementierungen mindestens $e \geq 41$ wählen. Die Zahl e darf für mehrere Benutzer einer Gruppe gleich sein, während n zwingend verschiedene Werte haben muß. Der geheime Schlüssel besteht aus $d =: K_{\text{SEC}}$, wobei $d := e^{-1} \bmod (p-1)(q-1)$. Der erweiterte Euklidsche Algorithmus liefert uns diesen modularen Kehrwert in polynomialer Zeit. Unsere Primzahlen p, q müssen für Fremde unbekannt bleiben und sollten darum vernichtet werden.[25]

Ver- und Entschlüsselung: Zur Verschlüsselung wird eine Nachricht in Blöcke $m_i \in \mathbb{Z}_n$ zerlegt. Die eigentliche Transformation wird dann mittels $c_i := m_i^e \bmod n$ durchgeführt. Der rechtmäßige Empfänger kann Nachrichtenteile jeweils mit $m_i := c_i^d \bmod n$ dekodieren und zur Gesamtnachricht m zusammensetzen. Bei der Funktion $f(x) = x^e \bmod n$ handelt es sich also um eine Falltürfunktion, deren Umkehrung $f^{-1}(x) = x^d \bmod n$ anscheinend nur mit Kenntnis von d bzw. der Faktorisierung von n berechnet werden kann. Allerdings wurde bereits angedeutet, daß die Äquivalenz (FAKTOR $\underset{\mathcal{P}}{\equiv}$ RSA) zwischen Faktorisierung und RSA-Problem noch nicht bewiesen ist. Nach jahrelanger Kryptoanalyse liegt diese Vermutung allerdings sehr nahe.

Korrektheit: Wir wollen nun zeigen, daß RSA überhaupt funktioniert, d. h. $\forall x \in \mathbb{Z}_n$: $f^{-1}(f(x)) = x$ gilt. Dazu müssen wir die Kongruenz $x^{ed} \equiv x \pmod{n}$ beweisen:

Aus der Konstruktion von d ist bekannt, daß $ed = 1 + k(p-1)(q-1)$ für ein $k \in \mathbb{Z}$ gelten muß. Wir hatten ja $d := e^{-1} \bmod (p-1)(q-1)$ gewählt, wobei dieses Inverse nun gerade $ed \equiv 1 \pmod{(p-1)(q-1)}$ erfüllt. Folglich können wir ed ersetzen und müssen nur noch die Beziehung $x^{1+k(p-1)(q-1)} \equiv x \pmod{n}$ zeigen.

Da sowohl p als auch q verschiedene Primzahlen sind, gilt jeweils die Euler-Fermat-Identität $x^{p-1} \equiv 1 \pmod{p}$ bzw. $x^{q-1} \equiv 1 \pmod{q}$. Wir exponentieren beide zuerst mit $k(q-1)$ bzw. $k(p-1)$

$$\begin{aligned} x^{(p-1)^{k(q-1)}} &\equiv x^{k(p-1)(q-1)} &\equiv\ 1 \pmod{p} \\ x^{(q-1)^{k(p-1)}} &\equiv x^{k(p-1)(q-1)} &\equiv\ 1 \pmod{q} \end{aligned}$$

[25] Einige Optimierungen für die Entschlüsselung benötigen jedoch zumindest einen Faktor. Sichere Speicherung und Zugriffsschutz von p und d sind deshalb immer angeraten.

und multiplizieren dann mit x:

$$\begin{aligned} x^{1+k(p-1)(q-1)} &\equiv x \pmod{p} \\ x^{1+k(p-1)(q-1)} &\equiv x \pmod{q} \end{aligned}$$

Mit Hilfe des *Chinesischen Restsatzes (CRT)* und $n = p \cdot q$ können wir schließen:

$$x^{1+k(p-1)(q-1)} \equiv x \pmod{n}$$

□

Durch die Kommutativität der beteiligten Operationen wird deutlich, daß obiger Beweis auch zur Umkehrung $f(f^{-1}(x)) = x$ gelten muß. Diese Eigenschaft ist Grundlage für digitale RSA-Signaturen, welche wir im nächsten Unterabschnitt besprechen werden.

2.3. Techniken für Authentizität

Subjekte können ihre Identität/Berechtigung durch *Authentisierung* Systemen oder anderen Subjekten beweisen. Solche kryptographische Verfahren werden allgemein als Identifizierungsprotokolle bezeichnet.

Authentizität von Nachrichten (Objekten) umfaßt aber zwei wesentliche Punkte:

1. *Authentischer Absender:* Der Urheber der Meldung kann zweifelsfrei festgestellt oder zumindest einem festen Pseudonym zugeordnet werden. Dieser Herkunftsnachweis sollte quasi ‚unfälschbar' sein.

2. *Integrität des Inhalts:* Die Nachricht ist vor unbefugter bzw. ungewollter Veränderung geschützt. Da diese Forderung für viele Verteilnetze zu streng ist, sollte wenigstens sichergestellt sein, daß der Empfänger eine solche Änderung auch bemerkt. Er hätte somit die Chance, manipulierte Daten erneut anzufordern oder die Kommunikation fehlerbedingt abzubrechen.

Hashfunktionen: Um Integrität binärer Daten festzustellen, existieren heutzutage eine Vielzahl von Verfahren aus dem Bereich der Kodierungstheorie, z. B. CRC. Einige von ihnen sind sogar in der Lage, geringe Veränderungen zu erkennen und autonom zu korrigieren. Für kryptographische Belange sind solche Algorithmen aber häufig ungeeignet, weil sie folgende Eigenschaft nicht besitzen:

Definition 2.5 (Einwegfunktion)
Eine Funktion $f : A \to B$ heißt Einwegfunktion, wenn $b = f(a)$ für jedes $a \in A$ leicht zu berechnen ist, aber es für die meisten $b \in B$ ‚praktisch unmöglich' ist, ein a mit $a = f^{-1}(b)$ zu finden.

Die bereits in Definition 2.4 auf Seite 23 angegebene Falltürfunktion hat diesen Charakter, ist aber insofern ein Spezialfall, weil ihre Umkehrung mit zusätzlicher Information leicht berechnet werden kann.

Beide Definitionen wollen wir hier noch mit einigen Erklärungen versehen:

Bemerkung 2.1 (zu Def. 2.4 und Def. 2.5)

- *Es ist klar, daß solche Funktionen nur dann nützlich sind, falls f durch einen Polynomialzeitalgorithmus berechnet werden kann. Andererseits darf für die Umkehrung f^{-1} ein solcher gerade nicht existieren oder muß zumindest unbekannt sein. Die Forderung nach $\mathcal{NP}$-vollständigen oder schwierigeren Problemen ist i. A. nicht mit der Effizienz von f zu vereinbaren.*
- *Zusätzlich sollten Eingabeargument a, Funktionswert b und Falltürinformation ähnliche Größenverhältnisse aufweisen, damit unpraktikable Verfahren mit äußerst langen Parametern (respektive Schlüsseln) von vornherein ausgeschlossen sind.*

Kryptographische Hashfunktionen sollten gerade die Einwegeigenschaft 2.5 haben. Zusätzlich fordert man jedoch, daß es auch *schwierig* ist, konkrete Kollisionen, d. h. zu gegebenen a und $f(a)$ einen Wert $\hat{a}$ mit $f(a) = f(\hat{a})$, $a \neq \hat{a}$, zu finden. In bestimmten Bereichen wird sogar vollständige Kollisionsfreiheit (Injektivität der Abbildung) verlangt, welche allerdings der praktischen Bedingung widerspricht, beliebig lange Eingaben a in Funktionswerte $b = f(a)$ mit fixierter Länge $|b|$ zu transformieren. Mathematisch stellt man sich also eher eine Abbildung des Typs $f : \{0,1\}^* \to \{0,1\}^{|b|}$ vor. Einige Algorithmen sind speziell dafür konzipiert und bieten gleichzeitig eine gewisse Resistenz gegen gezielte Kollisionssuche. Außerdem wurden sie jahrelang von verschiedenen Wissenschaftlern kryptoanalysiert und gelten deshalb als ziemlich sicher [19]:

MD5: Die Hashfunktionen der MD-Familie wurden von Wissenschaftlern der Firma *RSA Data Security Inc.* entwurfen. MD5 ist eine direkte Verbesserung seines Vorgängers MD4 (Autor: R. Rivest) und liefert einen Hashwert der Länge 128 Bit. Es existieren zwar einige akademische Angriffe, dennoch scheint MD5 zumindest momentan noch sicher zu sein.

SHA1: Das Kürzel steht für den weltweit häufig genutzten *Secure Hash Algorithm*, welcher 1992 von der U. S. Standardisierungsbehörde vorgestellt und danach von der NSA noch leicht verbessert wurde. Neben seiner ‚staatlichen Empfehlung' zeichnet ihn noch der 160 Bit lange Funktionswert aus, der eine bessere Resistenz gegen Kollisionen verspricht.

RIPE-MD160: Dieser Algorithmus ist eine europäische Entwicklung der bekannten Autoren H. Dobbertin, A. Bosselaers, B. Preneel [20] und wurde mit dem Designziel eingeführt, besonders widerstandfähig gegen bekannte Angriffe

der Kryptoanalyse zu sein. Trotz der damit verbundenen Sorgfalt ist das Verfahren sehr effizient und wird deshalb in meiner `GnuDCN`-Implementierung auch als Standardalgorithmus verwendet. Die Länge der Hashausgabe beträgt auch hier 160 Bit, was für aktuelle Sicherheitsanforderungen ausreichend ist.

Die Problematik einer unzureichend gewählten Ausgabelänge zeigt das sogenannte *Geburtstagsparadoxon*.[26] Demnach sind für eine Kollision nur ca. $2^{|b|/2}$ zufällige Eingaben zu untersuchen, wobei $|b|$ die Länge des Hashwertes in Bit angibt.

Einweghashfunktionen könnten auch aus symmetrischen Blockchiffren oder komplizierten zahlentheoretischen Problemen gewonnen werden, in der Praxis sind aber aus Gründen der Performance häufig nur obige Verfahren im Einsatz.

Digitale Signatur: Unterschriften haben fünf ideale Eigenschaften [8], die sie für den allgemeinen Gebrauch so nützlich machen:

1. Unterschriften sind authentisch und können deshalb andere davon überzeugen, daß der Unterzeichner ein Dokument willentlich unterschrieben hat.
2. Unterschriften sind fälschungssicher und beweisen deshalb, daß nur der Unterzeichner und kein Dritter das Dokument unterschrieben hat.
3. Unterschriften sind nicht wiederverwendbar und können nicht in andere Dokumente übertragen werden.
4. Unterschriften machen das unterzeichnete Dokument unveränderbar.
5. Unterschriften können nicht zurückgenommen werden.

B. Schneier stellt natürlich fest, daß keine dieser Eigenschaften in der Realität uneingeschränkt zutrifft, weil es vielerlei Manipulationsmöglichkeiten gibt. Unterschriften könnten kopiert, überdeckt oder verändert werden. Sie garantieren auch nicht die Integrität des Dokuments, da es relativ einfach ist, nachträglich noch Text einzufügen. Solche Fälschungsversuche sind aber für Betrüger mit einem hohen Risiko behaftet, da sie von Experten nachgewiesen werden können und somit vor Gericht kaum Bestand hätten.

Digitale Dokumente sind im Gegensatz dazu leider leicht und spurlos veränderbar. Außerdem können Computerdaten mit nur marginalen Kosten beliebig vervielfältigt werden. Um elektronische Unterschriften zu verwirklichen, brauchen wir also Sicherungsmechanismen, die uns in Form der besprochenen asymmetrischen Kryptosysteme bereits begegnet sind. Zusätzlich verwenden wir eine Hashfunktion h aus dem vorangegangenen Unterabschnitt, z. B. RIPE-MD160.

[26] Dieses Phänomen aus der Stochastik besagt, daß die Wahrscheinlichkeit schon knapp über 50 % ist, aus einer Gruppe von 23 Personen zwei Leute zu finden, die am selben Tag ihren Geburtstag (365 Möglichkeiten) haben.

Abbildung 6 zeigt die Kommunikations- und Rechenschritte, welche zur Erzeugung und Verifizierung einer digitalen Signatur notwendig sind. Alice wendet zuerst die kryptograpische Hashfunktion h auf ihre Nachricht m an, wobei sie einen Rückgabewert w von fester Länge erhält. Zum Fortfahren muß sichergestellt sein, daß w in der Argumentenmenge ihrer asymmetrischen Algorithmen E, D liegt. Dann wendet Alice D auf w an, was aufgrund des bekannten privaten Schlüssels $K_{\text{SEC}}^{\text{Alice}}$ auch problemlos möglich ist. Sie erhält die Signatur s, welche in Verbindung mit der Nachricht m zu Bob geschickt wird. Bob möchte sicher sein, daß niemand Alices Dokument während der Übertragung verändert hat und verifiziert deshalb die Signatur. Er berechnet zuerst selbst $w := h(m)$ und dann die Umkehrung von s. Da er über den öffentlichen Schlüssel $K_{\text{PUB}}^{\text{Alice}}$ verfügt und ihn für authentisch hält, muß bei korrekter Übertragung $w = v$ gelten.

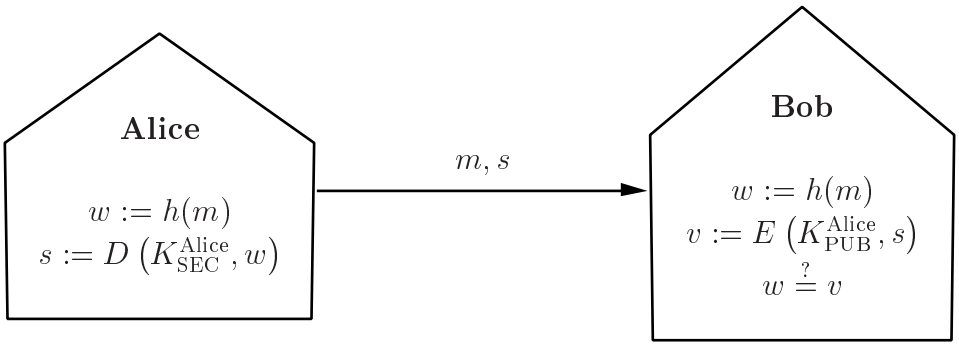

Abbildung 6: Prinzip der digitalen Unterschrift

Sollte Mallory sich am Dokument zu schaffen gemacht haben, ändert sich wegen der Kollisionsresistenz mit hoher Wahrscheinlichkeit auch der Hashwert $h(m)$. Da er aber Alices privaten Schlüssel nicht kennt, kann er auch keine neue Signatur s berechnen. Sein Betrug wird von Bob entdeckt.

Voraussetzung ist auch hier wieder die authentische Schlüsselverteilung, die mit Zertifizierungsstrukturen (siehe Seite 26) erreicht werden kann. Wir können nun aber beschreiben, von welcher Gestalt die darin vorkommenden *Zertifikate* sind.

Zertifikate sind digital signierte Informationen, die eine Zuordnung von Schlüsseln zu Subjekten ermöglichen. Sie stellen eine wichtige Anwendung, zugleich aber auch das ‚Rückgrat' der elektronischen Unterschrift dar. Welche konkreten Daten ein Zertifikat enthält, hängt vom zugrundegelegten Zertifizierungsstandard ab und kann teilweise stark differieren. Bestätigungen nach dem offenen Standard `OpenPGP` (RFC2440) sollten beispielsweise den Schlüssel, zugehörende Subjektmerkmale (Name und E-Mail-Adresse des Schlüsselinhabers), das Datum, eine Gültigkeitsdauer und die Schlüssel-ID des Zertifikatausstellers beinhalten. Mit letzerer können Relationen innerhalb einer hierarchischen oder netzartigen

Zertifizierungsstruktur hergestellt werden, um eine Entscheidung hinsichtlich der Authentizität des Schlüssels zu fällen.

RSA-Signaturen: Wir hatten bereits gezeigt, daß beim RSA Algorithmus sowohl $f^{-1}(f(x)) = x$ als auch $f(f^{-1}(x)) = x$ für beliebige Zahlen $x \in \mathbb{Z}_n$ gilt. Der Schlüsseleigner kann also leicht eine Signatur $s := w^d \bmod n$ mit $w := h(m)$ konstruieren, falls h in eine Teilmenge von $\mathbb{Z}_n$ abbildet. Der Verifizierer kann andererseits $v := s^e \bmod n$ (also eigentlich $w^{de} \bmod n$) allein mit Hilfe des öffentlichen Schlüssels berechnen und v mit $h(m)$ vergleichen. Stimmen beide Werte überein, ist die Signatur gültig und unsere Nachricht m authentisch.

Das Verfahren ist sehr effizient, da jeweils nur eine modulare Exponentation und die Hashwertberechnung ausgeführt werden muß. Mit Hilfe des Chinesischen Restsatzes sind zudem weitere Beschleunigungen möglich.

Ein Angreifer könnte versuchen, den Wert s seinen Manipulationen von m anzupassen. Ist h eine hinreichend kollisionsresistente Hashfunktion, wird dieses Vorgehen jedoch vereitelt. Die Sicherheit digitaler RSA-Signaturen hängt also jetzt nicht mehr nur vom RSA-Problem, sondern auch von den Eigenschaften der verwendeten Einweghashfunktion h ab. Wir wollen im folgenden davon ausgehen, daß sich h diesbezüglich ideal verhält. Diese Annahme heißt allgemein *Random-Orakel-Modell*, welches uns später noch genauer beschäftigen wird.

2.4. Anonymität durch Kryptographie

Grundlage aller modernen Verfahren zur Wahrung der Anonymität sind die wegweisenden Arbeiten von David Chaum [21, 22, 23]. Während eine begrenzte Empfängerunbeobachtbarkeit[27] releativ leicht durch Streuverteilung (Broadcast) und ggf. Verschlüsselung zu erreichen ist[28], mußten für die Senderunbeobachtbarkeit erst kompliziertere Konzepte entwickelt werden. Wir wollen jetzt zwei Lösungen betrachten, die einen sehr generischen Charakter haben und deshalb die Basis aller bekannten Anonymitätsmethoden darstellen.

[27] folglich auch Empfängeranonymität (siehe Implikationen von Tabelle 1.1 auf Seite 14)

[28] Dieses Vorgehen ist mit der Publikation in öffentlich zugänglichen Medien vergleichbar, wobei aber der Inhalt einer Nachricht für alle Teilnehmer außer Absender und Empfänger geheim bleiben muß. Relevante Informationen können durch individuelle Verschlüsselung (z. B. mit Public-Key-Kryptographie) geschützt werden, so daß nur rechtmäßige Kommunikationspartner darauf Zugriff haben. Leider verursachen die notwendigen Entschlüsselungstests der restlichen Teilnehmer einen Berechnungsaufwand proportional zu $O(n)$, da der konkrete Empfänger nicht mehr im Kopf der Nachricht angegeben werden darf.

2.4.1. MIX-Netze

Ein solches Netz besteht aus einer Reihe autonomer MIX-Knoten, deren öffentliche Schlüssel $K_{\mathrm{PUB}}^{\mathrm{MIX_i}}$ jedem Kommunikationsteilnehmer bekannt sind.

Die verteilten Knotenrechner agieren im Anforderungsfall: Sie entschlüsseln empfangene Nachrichten, entfernen kritische Metainformationen (Name, Adresse, IP-Nummer, etc.), anhand derer eine Identifizierung des Absenders möglich wäre, und speichern den Inhalt nebst Empfängeradresse in einer Datenbank zwischen. Ist eine ausreichende Anzahl Sendungen verschiedener Absender eingetroffen, werden alle bisher gespeicherten Meldungen umsortiert und schließlich an ihre Empfänger (meist andere MIX-Knoten) weitergeleitet. Durch diese Umordnung[29] und Verschlüsselung können eingehende Mitteilungen nicht mehr mit ausgehenden in Verbindung gesetzt werden. Die Verarbeitung erfolgt somit ‚schubartig', wobei ein Schub aus allen innerhalb einer bestimmten Zeitspanne eingetroffenen Nachrichten besteht. Folgende Abbildung zeigt die einzelnen Schritte graphisch:

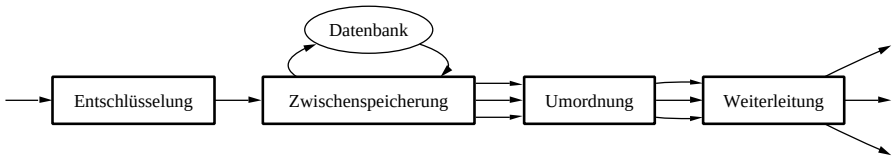

Abbildung 7: Verarbeitungsschritte eines MIX-Knotens

Zur Wahrung der Anonymität ist es wichtig, daß beteiligte Knotenbetreiber die Metainformationen wirklich entfernen und keinerlei Logbuch über Ein- und Ausgang von Mitteilungen führen. Bei professionellen Diensten wie dem `JAP Anonymity & Privacy Proxy` [24] der TU Dresden gibt es deshalb einige Regeln sowie eine Selbstverpflichtungserklärung [25], um die Vertrauenswürdigkeit sicherzustellen. Trotzdem sollte man sensible Daten nie über nur einen MIX leiten. Stärker wird das System, wenn die Nachrichten eine *Kaskade von MIX-Knoten* durchlaufen, deren Betreiber unterschiedliche Institutionen[30] mit divergierenden Interessen sind. Ein Austausch identitätsenttarnender Merkmale kann so zumindest erschwert werden. Auch die ‚physischen Knotenstandorte' im zugrundeliegenden Verteilnetz sollten unterschiedlich sein. Zwar verhindert Verschlüsselung die Zuordnungsfähigkeit, allerdings kann eine komplette Netzverkehrsanalyse Angreifern wesentlich subtilere Möglichkeiten bieten. Eine Kaskade hat also, solange das Datenaufkommen ausreicht und wenigstens ein MIX nicht korrupt ist, anonymitätsfördernde Wirkung. Der Weg, den eine Nachricht durch ein MIX-Netz nimmt, wird als *Indirektionspfad* bezeichnet, da das Basisprinzip dieser Technik

[29] Daher stammt auch der Name MIX, welcher im Sinne von ‚mixen' zu verstehen ist.
[30] z. B. Hochschulen, Datenschutzbeauftragte, Bürgernetzvereine, kirchliche Organisationen, ...

die Indirektion[31] ist. Der Sender legt diesen Pfad durch eine kaskadierte Verschlüsselung fest. Wir wollen dieses Vorgehen an einem Beispiel deutlich machen.

Alice möchte Bob unerkannt eine Mitteilung zusenden. Zuerst legt sie den Indirektionspfad für das MIX-Netz (siehe Abbildung 8) fest, welcher hier

$$\textbf{Alice} \hookrightarrow \mathrm{MIX}_2 \hookrightarrow \mathrm{MIX}_4 \hookrightarrow \mathrm{MIX}_5 \hookrightarrow \mathrm{MIX}_7 \hookrightarrow \textbf{Bob}$$

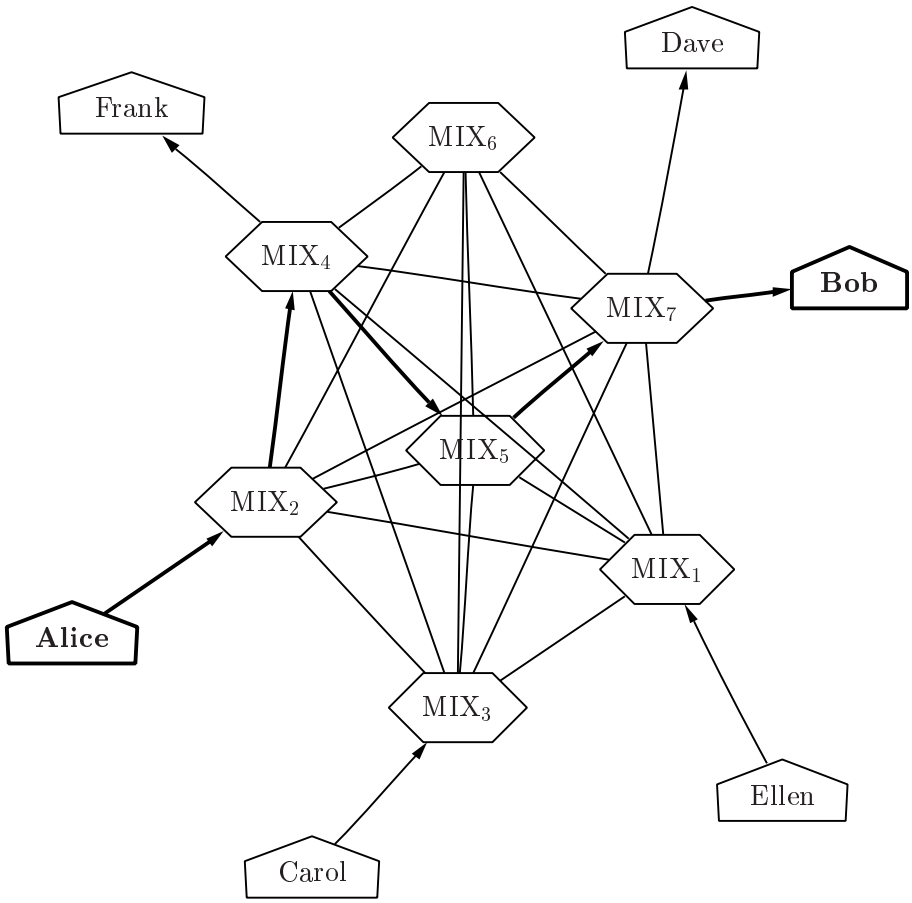

Abbildung 8: Indirektionspfad durch eine MIX-Kaskade

sei. Sie verschlüsselt dann ihre Nachricht m und die jeweilige Empfängeradresse,

[31] Senden auf indirektem Wege, d. h. über weiterleitende Stellvertreter oder Verteilzentren.

wobei sie den Pfad rückwärts travesiert:

$$\ldots \quad E\left(K_{\mathrm{PUB}}^{\mathrm{MIX}_7}, [\mathbf{Bob}, m]\right)$$

$$\ldots \quad E\left(K_{\mathrm{PUB}}^{\mathrm{MIX}_5}, \left[\mathrm{MIX}_7, E\left(K_{\mathrm{PUB}}^{\mathrm{MIX}_7}, [\mathbf{Bob}, m]\right)\right]\right)$$

$$\ldots \quad E\left(K_{\mathrm{PUB}}^{\mathrm{MIX}_4}, \left[\mathrm{MIX}_5, E\left(K_{\mathrm{PUB}}^{\mathrm{MIX}_5}, \left[\mathrm{MIX}_7, E\left(K_{\mathrm{PUB}}^{\mathrm{MIX}_7}, [\mathbf{Bob}, m]\right)\right]\right)\right]\right)$$

$$E\left(K_{\mathrm{PUB}}^{\mathrm{MIX}_2}, \left[\mathrm{MIX}_4, E\left(K_{\mathrm{PUB}}^{\mathrm{MIX}_4}, \left[\mathrm{MIX}_5, E\left(K_{\mathrm{PUB}}^{\mathrm{MIX}_5}, \left[\mathrm{MIX}_7, E\left(K_{\mathrm{PUB}}^{\mathrm{MIX}_7}, [\mathbf{Bob}, m]\right)\right]\right)\right]\right)\right]\right)$$

Nach der Kodierung sendet Alice das Ergebnis an den zuerst gewählten Knoten MIX_2, welcher den Anfang der Kaskade dekodieren kann. Er entfernt alle Absenderinformationen und speichert die Nachricht zwischen. Sind genügend andere Mitteilungen eingegangen, wird er Alices Sendung an den Empfänger MIX_4 weiterleiten. Dieser kann nur noch feststellen, daß die Nachricht offensichtlich von MIX_2 kommt und für MIX_5 adressiert ist. Erst der letzte Knoten MIX_7 erfährt den wirklichen Empfänger, wobei kaum mehr nachvollziehbar ist, woher die Mitteilung urspünglich stammt. Bob kann zwar erkennen, daß MIX_7 die Nachricht zugestellt hat, der wahre Absender bleibt ihm und Außenstehenden jedoch verborgen, da gleichzeitig auch noch Carol und Ellen senden. Wenn Alice ihre Identität im Inhalt preisgibt, muß sie das letzte Stück „$\mathrm{MIX}_7 \hookrightarrow \mathbf{Bob}$“ durch Kodierung mit Bobs öffentlichem Schlüssel absichern. Auf jeden Fall müssen alle Nachrichten eine uniforme Länge besitzen, da sonst einfach Zuordungen aufgrund von Längendifferenzen getroffen werden könnten.

Weitere Angriffsszenarien sind denkbar: So sollten die Teilnehmer des Mix-Netzes auch dann Daten senden, wenn sie eigentlich keine Nachrichten übermitteln wollen. Diese falschen Mitteilungen werden als *Dummy-Traffic* bezeichnet und dienen der Verschleierung von Kommunikationsbeziehungen, falls der Angreifer den kompletten Nachrichtenverkehr im Verteilnetz überwachen kann.

Ein besonders ‚hinterhältiger‘ Enttarnungsversuch sind die sogenannten *Replay-Attacken*. Dabei zeichnet der Beobachter eine gesendete Nachricht auf und spielt sie später nochmal ein, so daß dem MIX-Knoten eine bereits versendete Nachricht zur erneuten Bearbeitung vorgelegt wird. Dabei entsteht eine zur ursprünglichen Sendung identische Mitteilung, die eine Verfolgung innerhab des Netzes ermöglicht. Um dieses Vorgehen zu verhindern, muß jeder Knoten einen ‚Fingerabdruck‘[32] bereits gesendeter Nachrichten in seiner Datenbank ablegen. Damit prüft er jede eingehende Mitteilung auf Duplikateigenschaft und ignoriert sie gegebenfalls. Nach einer gewissen Zeitspanne werden ältere Einträge wieder entfernt, um ein endloses Anwachsen der Datenbank zu vermeiden.

[32] Funktionswert einer kryptographischen Hashfunktion über der Nachricht

Außerdem sind $n-1$-Angriffe bei geringen Nachrichtenaufkommen durchführbar, wobei n die Schubgröße[33] darstellt. Ein Beobachter, der mit $n-1$ Absendern zusammenarbeitet oder selbst diese Anzahl von Nachrichten erzeugt, kann den einzigen verbliebenen Sender ohne große Schwierigkeiten ermitteln. Allerdings muß er dazu den gesamten Netzwerkverkehr zwischen Überwachungs- bzw. Empfängersubjekt und dem MIX-Netz kontrollieren.

Die Arbeit „Anonymity and Unobservability in the Internet“ bietet einen guten Überblick und Vergleich aktueller MIX-Netz-Systeme. [27, 28]

Obige Beispiele zeigen deutlich, daß MIX-Netze starken Angreifern (z. B. Regierungen oder Geheimdiensten) lediglich begrenzt widerstehen können. Zumindest kann ihre Sicherheit diesbezüglich nur vage eingeschätzt, geschweige denn formal bewiesen werden. Deshalb wollen wir jetzt einen weiteren Vorschlag diskutieren.

2.4.2. DC-Netze

Hauptidee dieses Verfahrens [23] ist ‚überlagertes Senden‘ (Superposed Sending), mit dem Unbeobachtbarkeit erreicht werden kann. Klassisch wird die Aufgabenstellung oft als *Problem der abendessenden Kryptographen (Dining Cryptographers Problem)* beschrieben:

Unsere Kryptographen Alice, Bob, Carol, Dave, Ellen und Frank sitzen in einem beliebten Drei-Sterne-Restaurant. Der Ober informiert sie, daß mit dem Lokalbesitzer eine anonyme Bezahlung vereinbart wurde. Die könnte einer der sechs Kryptographen oder eine dritte Partei (beispielsweise die National Security Agency) veranlaßt haben. Alle Kryptographen würden natürlich akzeptieren, falls einer unter ihnen die Rechnung anonym begleichen möchte, bei der NSA haben sie jedoch Bedenken wegen ihrer heiklen Tischgespräche. Folgendes Protokoll löst ihr Entscheidungsproblem:

Jeder der Kryptographen wirft verdeckt hinter seiner Speisekarte eine ideale Münze genau so, daß lediglich sein rechter Nachbar das Ergebnis sieht. Er kennt nun die Seite der eigenen sowie der links von ihm gewurfenen Münze und kann laut sagen, ob beide die gleiche Seite zeigen oder ob das Gegenteil der Fall ist. Wenn er jedoch die anonyme Zahlung vereinbart hat, lügt er hier und behauptet genau die Umkehrung. Jetzt werden die von allen gesagten Ergebnisse ausgezählt: Eine ungerade Anzahl verschiedener Aussagen zeigt an, daß einer von ihnen zahlt. Um wen es sich jedoch konkret handelt bleibt anonym. Falls eine gerade Anzahl Unterschiede vorliegt, wurde das Mahl entweder doppelt bezahlt oder die NSA hat die Sache arrangiert. Für unsere ehrlichen Kryptographen ist es selbstverständlich, sich genau an die Protokollvereinbarungen zu halten, d. h. keiner stört absichtlich durch nicht konforme Aussagen.

[33] d. h. Anzahl der Mitteilungen, die ‚in einem Schub‘ verarbeitet werden

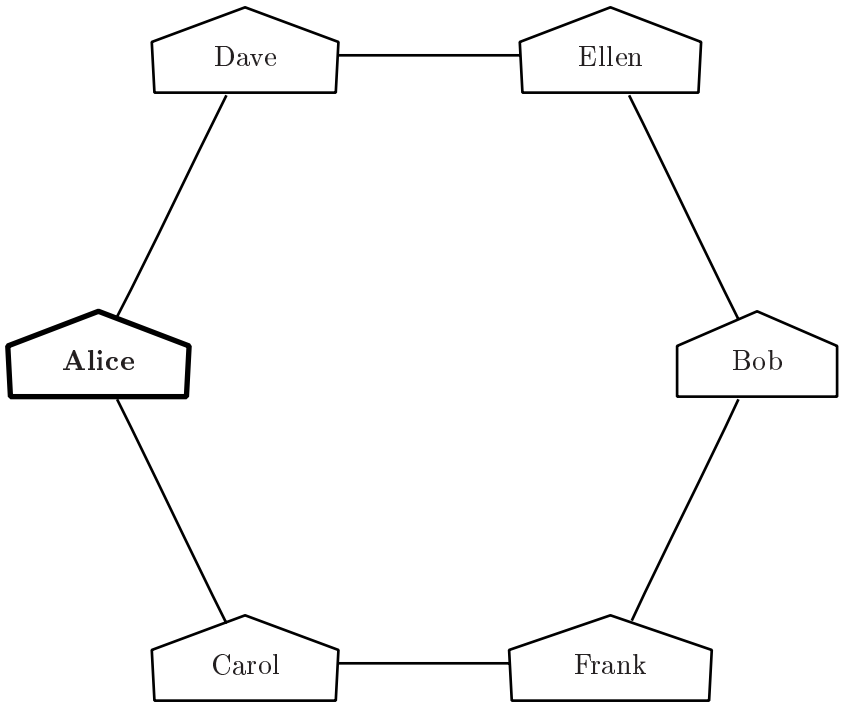

Abbildung 9: klassischer DC-Ring

Offensichtlich wird von jedem der n-Teilnehmer genau ein Bit an Information[34] überlagert gesendet. Längere Nachrichten können problemlos übertragen werden, indem man das Protokoll rundenweise wiederholt.

Doch wie stark ist dabei die Anonymität? Bei der klassischen DC-Ring Variante (Dinner-Tisch; siehe Abbildung 9) wird deutlich, daß schon zwei zusammenarbeitende Angreifer ihre Kollegen enttarnen können. Nehmen wir für einen Moment an, daß Alice die großzügige Spenderin des Abendmahls sei. Falls Dave und Carol ihre Münzwurfergebnisse austauschen, werden sie leicht erkennen, ob Alice wahrheitsgemäß berichtet oder etwa gelogen hat. Ebenso könnten Dave und Bob feststellen, daß Ellen nicht die Bezahlerin ist. Nach durchschnittlich $\Theta(n \lg n)$ Wiederholungen [30] mit jeweils anderen Ringpositionen der Teilnehmer wäre die Anonymität vollständig verloren.

Naheliegend ist deshalb eine n-zu-n Münzwurf-Topologie (siehe Abbildung 10), welche der maximalen Anzahl von $n-2$ Angreifern widersteht. Der Beweis dafür gelang David Chaum [23] bereits 1988. In solchen DC-Netzen können wir also theoretisch unbedingte Sender- bzw. Empfängerunbeobachtbarkeit erreichen, weil alle Mitspieler mit $n-1$ anderen Personen ihr Geheimnis teilen. Nur einer vollständigen Koalition gegen den Sender gelänge es, die Unbeobachtbarkeit auf-

[34] Zahler des Abendessens: ja/ nein

zudecken. Durch zusätzliche Verschlüsselung ist auch hier implizite Adressierung und damit Punkt-zu-Punkt-Kommunikation zwischen den Teilnehmern möglich: Alice kann ihre anonyme Nachricht für Bob kodieren. Jeder empfängt sie, aber nur Bob kann den Inhalt sinnvoll entschlüsseln.

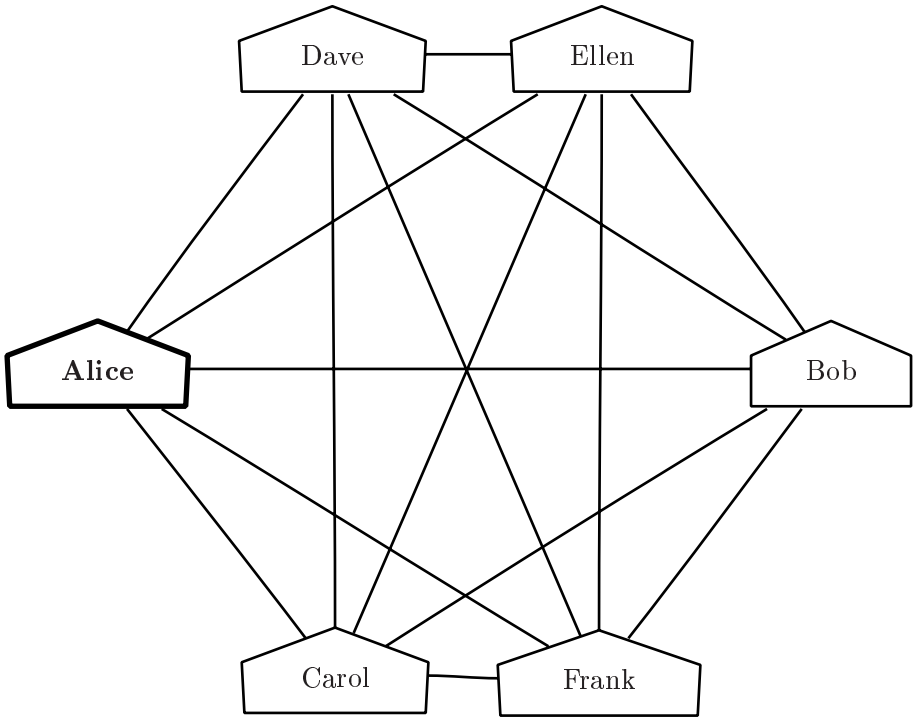

Abbildung 10: vollverbundenes n-zu-n DC-Netz

Trotzdem hätte eine naive Implementierung des DC-Protokolls noch schwerwiegene Nachteile, welche wir später Schritt für Schritt beseitigen müssen.

- *Ideale Münzen:* Möchte man das Protokoll auf deterministischen Rechnern umsetzen, stellt sich schnell die Frage, wie ein idealer – d. h. nicht vorhersagbarer – Münzwurf simuliert werden kann. Eine Möglichkeit ist die Verwendung tausender Zufallszahlen, welche beiden beteiligten Partnern zur Verfügung stehen. D. Chaum merkt an, daß optische Datenträger durchaus in der Lage sind, die für eine lange Konversation benötigten Mengen zu fassen. Es bleibt allerdings die Thematik des Schlüsselaustauschs, der hier aufgrund des enormen Speicherplatzbedarfs eine physische Medienübergabe erfordern würde. Wir werden in unserer `GnuDCN`-Implementierung dieses Problem durch den Einsatz von Pseudozufallsbitgeneratoren abmildern. Damit müssen nur noch geringe Schlüsselinformationen und ein definierter Anfangszustand übertragen werden, deren Geheimhaltung wir durch Public-Key-Kryptographie gewährleisten. Für unsere Zwecke eignet sich nur ein komplexitätstheoretisch sicherer Generator, da dessen Ausgabevorhersage äquivalent zum Lösen eines schwierigen Problems (hier Primfaktorisierung)

ist. Die Anonymität bleibt somit unter dieser kryptographischen Annahme beweisbar.

- *Kollisionen beim Senden:* Falls zwei Teilnehmer gleichzeitig Daten übermitteln, können die restlichen Stationen diese nicht voneinander trennen. Die Übertragung muß wiederholt werden. Es gibt diverse Lösungen [34] für diese Art von Synchronisationsproblemen. Wir werden sie im Abschnitt 4 genauer analysieren und auch gegenüberstellen.

- *Bösartige Störer:* Komplizierter ist die Sicherung der Robustheit des Protokolls. Böswillige Teilnehmer könnten die gegebene Anonymität ausnutzen, um das Netz dauerhaft zu stören. Chaum schlug zur Lösung vor, in unregelmäßigen Abständen sogenannte ‚Fallen' zu stellen, die keine sensitiven Informationen enthalten und damit auch nicht anonym bleiben müssen. Da Fallen herkömmlichen Sendungen gleichen, sind dauerhafte Störer mit hoher Wahrscheinlichkeit davon betroffen. Werden die bei Fallen verwendeten Zufallszahlen aller Teilnehmer später bekanntgegeben, ist der Schuldige leicht ermittelbar. Leider gab es im ursprünglichen Verfahren einige Angriffspunkte. Hier sorgen Erweiterungen [31, 32] von J. Bos, B. den Boer sowie B. Pfitzmann, M. Weidner für Abhilfe. Allerdings ist ihre Nachrichten- und Kommunikationskomplexität unheimlich hoch und limitiert somit das Einsatzgebiet von DC-Netzen. Trotzdem wird dieses sehr robuste Protokoll im Abschnitt 4 beschrieben.

- *Zuverlässige Verteilung:* In jedem Fall muß die Übertragung der Daten *zuverlässig* erfolgen, d. h. alle ehrlichen Teilnehmer empfangen die gleichen Werte.[35] Am Tisch der Kryptographen war das ohne Einschränkung gewährleistet, in der Realität erfüllen jedoch nur wenige physikalische Netzwerke (z. B. Satelliten- bzw. Radioverteilung, Ethernet-Bus-Topologie, ...) diese Forderung. Allgemein ist das Problem als *Byzantinische Übereinstimmung* bekannt. Im dritten Kapitel werden wir Lösungen besprechen, die sich für eine asynchrone Bearbeitung[36] eignen. Dafür benötigte kryptographische Mechanismen (sogenannte Schwellenwertsignaturen) sollen Gegenstand der folgenden Unterabschnitte dieser Arbeit sein.

Zusammenfassend läßt sich also feststellen, daß vollverbundene DC-Netze zwar unbedingte Senderanonymität[37] garantieren könnten, allerdings wirft ihre Realisierung wiederum eine Reihe von Problemen auf, deren Lösung jeweils nur komlexitätstheoretisch sicher[38] ist. Außerdem haben diese ‚Hilfsmittel' großen Einfluß auf die Laufzeit des Gesamtverfahrens. Wir müssen deshalb speziell ihre Einschränkungen sorgfältig untersuchen, um sowohl die realistische Betrachtung der Praxistauglichkeit als auch eine brauchbare Implementierung zu bekommen.

[35] Insbesondere wenn sich ein Sender fehlerhaft oder sogar bösartig verhält, indem er unterschiedliche Ausgaben verbreitet und damit die Teilnehmer inkonsistent stört.

[36] Gängige Weitverkehrsnetze (Internet) verteilen Datenpakete asynchron.

[37] Wir werden David Chaums Beweis dieser These im Abschnitt 4 nachvollziehen.

[38] Die Rechenkapazität der Angreifer ist beschränkt, d. h. sie können beispielsweise bestimmte zahlentheoretische Fragestellungen (vgl. Seite 23) nicht effizient beantworten.

2.5. Kryptographie mit mehreren Parteien

Sind sensible Informationen oder Gegenstände (Kennwörter, Schlüssel, ...) im Besitz eines Subjekts, verfügt es damit auch über die alleinige Anwendungsgewalt. In manchen Bereichen ist soetwas unerwünscht, man denke zum Beispiel an Abschußrampen für Atomraketen oder große Tresorräume. Hier kommen ‚Hardware'-Lösungen zum Einsatz: In einer Raketenstation sind meist zwei Schlüsselschalter installiert, die den Start nur dann auslösen, wenn sie simultan bedient werden. Die dafür nötigen verschiedenen Schlüssel werden an zwei zuverlässige Offiziere ausgegeben. Bei Tresorräumen wäre eine Situation denkbar, wo jeweils drei von insgesamt acht Angestellten nur zusammen mit dem Direktor die Öffnung vornehmen können. Damit ist auch hier ein alleiniger Zugriff unmöglich.

Wir wollen nun Methoden besprechen, mit denen ähnliche Anwendungen auch auf Digitalrechnern realisiert werden können. Zur Einführung stellen wir ein Verfahren vor, bei dem die Beteiligung aller Geheimnis-/Schlüsselträger notwendig ist[39], bevor wir die flexiblere Schwellenwertkryptographie[40] behandeln.

2.5.1. Geheimnisteilung

Eine Nachricht wird in mehrere Teile zerlegt. Jeder für sich ist ohne Bedeutung; erst alle zusammen ergeben das Geheimnis. Zur anfänglichen Aufteilung ist noch ein zuverlässiger Vermittler notwendig, dem jeder vertraut. Im richtigen Leben wäre die Person des Notars geeignet, welche wir fortan mit dem Namen „Trent" bezeichnen. Für zwei Geheimnisträger (Alice, Bob) bereitet er folgendes vor:

1. Trent erzeugt eine Zufallszeichenkette r. Sie muß exakt die gleiche Länge wie unsere Nachricht m besitzen.
2. Trent bildet $s := m \oplus r$, wobei $\oplus$ die bitweise Exklusiv-Oder-Verknüpfung (XOR) der beiden Teile ist.
3. Trent gibt das Schlüsselteil s an Alice und r an Bob weiter.

Wollen Alice und Bob das Geheimnis rekonstruieren, kombinieren sie einfach ihre zwei Teile und erhalten $m := s \oplus r$. Falls r von Trent wirklich zufällig gewählt wurde, handelt es sich um ein sogenanntes *One-Time-Pad* und die Sache ist sicher. Man kann dieses Verfahren auch für n Teilnehmer verallgemeinern: Trent bildet jeweils $n-1$ Zufallsketten r_i und berechnet

$$s := m \oplus \left(\bigoplus_{i=1}^{n-1} r_i \right) .$$

[39] vgl. Beispiel der Atomraketenbasis
[40] vgl. Beispiel des Banktresor

Die Rekonstruktion durch $m := s \oplus \left(\bigoplus_{i=1}^{n-1} r_i \right)$ erfolgt dann ohne Trents Hilfe.

Kritik: Trent besitzt hier die absolute Verfügungsgewalt. Er kann betrügen, indem er Unsinn verbreitet oder Schlüsselteile heimlich an Unbefugte weitergibt. Nur alle Geheimnisträger zusammen können entscheiden, ob m wirklich ein Geheimnis enthält oder ob Trent betrogen hat. Ein weiterer Nachteil ist die Robustheit: Geht auch nur ein einziger Bestandteil verloren, kann die Nachricht nicht mehr rekonstruiert werden. Um Betrug seitens der Teilnehmer zu verhindern, muß die Aufdeckung der r_i, s gleichzeitig erfolgen. Der böswillige Mallory könnte sonst seine Zeichenkette nicht preisgeben, aber trotzdem m in Erfahrung bringen. Die übrigen Kameraden würden dabei leer[41] ausgehen.

2.5.2. Schwellenwertverfahren

Wie bereits angedeutet, ist das obige Verfahren weder flexibel noch robust. Wir würden uns deshalb lieber eine Methode wünschen, mit welcher unser Geheimnis von verschiedenen Subjektmengen rekonstruiert werden kann. Die Anzahl der mindestens dazu notwendigen Schlüsselteile heißt *Schwellenwert*.

Definition 2.6 ($/n, k/$-Schwellenwertverfahren)
Diese Klasse von Verfahren teilt ein Geheimnis so in n verschiedene Stücke auf, daß zur Rekonstruktion immer mindestens k davon benötigt werden.

Mit einem $/14, 9/$-Schwellenwertverfahren wäre auch unser schönes Beispiel mit dem Banktresor lösbar: Der Direktor erhält sechs Schlüsselteile und die Angestellten jeweils einen. Alle acht Angestellten der Bank können gemeinsam den Tresorraum nicht öffnen, da sie insgesamt über weniger als 9 Teile verfügen. Auch der Direktor allein kann nicht einbrechen. Er bekommt erst dann Zugang, wenn drei beliebige Angestellte mit ihm zusammenarbeiten.

Heutzutage sind viele Varianten für Schwellenwertalgorithmen bekannt, um derartige Anwendungen zu realisieren. Ein besonders simples und zugleich sicheres Verfahren soll nun vorgestellt werden. Es ist außerdem Grundlage der später behandelten RSA-Schwellenwertsignaturen.

Polynomverfahren mit Lagrange-Interpolation: Dieses Verfahren wurde von Adi Shamir [35] bereits 1979 eingeführt. Das Geheimnis bestehe aus der Zahl z, die unter n Personen verteilt werden soll. Während immer k von ihnen gemeinsam den Wert aufdecken können, sind $k-1$ dazu nicht in der Lage.

[41] d. h. ohne Kenntnis der gesuchten Nachricht m

Trent wählt eine Primzahl p, wobei p größer als die Anzahl der Teilgeheimnisse und auch größer als z sein muß. Er bildet dann ein Polynom $F(x)$ vom Grad $k-1$ im Primzahlkörper $\mathbb{Z}_p$ ($p > n$, $p > z$):

$$F(x) = \left(a_{k-1}x^{k-1} + a_{k-2}x^{k-2} + \ldots + a_1 x + z\right) \bmod p$$

Die Koeffizienten werden zufällig mit $0 \leq a_i < p$ gewählt und dürfen niemandem außer Trent zugänglich sein. Am besten werden sie nach Abschluß der Geheimnisteilung vernichtet. Die Primzahl p wird hingegen öffentlich bekanntgegeben. Trent berechnet nun die einzelnen Teilstücke s_i durch Auswertung des Polynoms an n verschiedenen Stützstellen x_i, z. B. sei $\forall i = 1, \ldots, n: \; x_i := i$ gesetzt:

$$s_i = F(x_i) \qquad i = 1, \ldots, n$$

Im endlichen Körper $\mathbb{Z}_p$ können jetzt aus k Gleichungen der Art

$$a_{k-1}x_i^{k-1} + a_{k-2}x_i^{k-2} + \ldots + a_1 x_i + z \equiv s_i \pmod p$$

das Geheimnis z und die Koeffizienten a_i durch Lagrange-Interpolation bestimmt werden. Dazu berechnet die Teilnehmermenge $R \subseteq \{1, \ldots, n\}$ das Polynom

$$F'(x) = \sum_{i \in R} s_i \cdot \lambda_i^R(x) \qquad \text{mit} \qquad \lambda_i^R(x) = \prod_{j \in R \setminus \{i\}} \frac{(x-j)}{(i-j)}$$

und schließlich $z = F'(0)$. Wegen der Eindeutigkeit der Interpolation und weil alle Gleichungen linear unabhängig sind (entsprechende Vandermonde-Determinante ist teilerfremd zu p) gilt nämlich $\forall x \in \mathbb{Z}_p: \; F'(x) = F(x)$.

Diskussion: Auch mit großer Rechenleistung kann ein Angreifer aus $k-1$ Teilgeheimnissen z nicht rekonstruieren. Sogar exhausive Suche über alle $(p-1)p^{n-1}$ Kombinationen bringt keine Vorteile, denn die Zahlen $s_i \in \{1, \ldots, p-1\}$ sind (bei zufälliger Wahl der Koeffizienten a_i) gleichwahrscheinlich.

Betrugsversuche, Angriffe: Alice, Bob und Carol teilen ein Geheimnis und wollen es enthüllen. Folgende Szenarien sind denkbar:

Fall 1 Carol will die Rekonstruktion insgeheim verhindern. Beim Aufdecken gibt sie ein falsches Teilgeheimnis an. Die Nachricht z kann nicht gelesen werden und keiner kann nachweisen, warum das Öffnen nicht erfolgreich war. Es könnten zuletzt ja auch Trent oder Alice bzw. Bob geschummelt haben.

Fall 2 Beim Offenlegen wird Carol zuletzt gefragt und gibt einen falschen Teil an. Vorher hat sie sich jedoch die Teile von Ailce und Bob gemerkt. Carol kann später das Geheimnis z allein lüften, ohne daß die anderen davon erfahren.

Alice, Bob, Carol und Dave teilen das Geheimnis mit einem Verfahren des Typs $/4,3/$.

Fall 3 Mallory ist bei der Offenlegung statt Dave anwesend. Er wartet bis Alice, Bob und Carol ihre Teile offengelegt haben, berechnet dann z und kann damit auf das gültige Teilgeheimnis von Dave schließen, welches er den anderen zum Beweis seiner falschen Identität vorlegt. Mit dieser Prozedur hat er z erlangt, ohne daß die anderen etwas davon bemerken.

Wie wir gleich sehen werden, sind obige Betrugsmöglichkeiten in einigen speziellen Anwendungen gegenstandslos, weil dort zwar mit dem Geheimnis gerechnet, es aber nie vollständig offengelegt wird.

2.6. Signaturschemata für mehrere Unterzeichner

Im ABBA-Protokoll des Abschnitts 3.2.1 benötigen wir ein Signaturschema, welches gültige Unterschriften nur dann erzeugt, falls eine bestimmte Anzahl von Subjekten aktiv zusammenarbeitet. Wir wollen nun Eigenschaften, Bestandteile und Sicherheitsanforderungen dieser Verfahren formalisieren [37]:

Definition 2.7 ($/n,k,t/$-Schwellenwertsignaturschema)
Solche Schemata bestehen aus insgesamt n Subjekten, wobei bis zu t von ihnen bösartige Absichten verfolgen oder durch Angreifer kontrolliert werden könnten. Die Zusammenarbeit von mindestens k Subjekten ist notwendig und ausreichend, um eine gültige Unterschrift zu erzeugen.

Jeder der n Beteiligten ist in der Lage, einzelne Teilsignaturen für beliebige Dokumente zu erstellen. Andere Subjekte sollten deren Gültigkeit prüfen können und sie nur im positiven Fall speichern. Hat ein Teilnehmer k Stück von verschiedenen Unterzeichnern[42] gesammelt, kann er eine Gesamtsignatur berechnen. Diese ist auch durch Außenstehende verifizierbar und garantiert, daß mindestens k Subjekte das Dokument vorher bearbeitet (also gesehen) haben.

Bösartige Angriffe können jetzt nur noch stattfinden, falls die Anzahl der Betrüger den Schwellenwert k erreicht. Durch eine Verifikation der Geheimnisteile wird nämlich gesichert, daß diese wirklich verfahrenskonform sind. Insbesondere werden damit Fall 1, 2 (siehe Seite 44) der Angriffsszenarien gegenstandslos.

Wegen maximal zugelassenen t Betrügern sollte der Schwellenwert k vernüftigerweise durch $t < k \leq n - t$ beschränkt sein. Oft wird $k := t + 1$ gesetzt; andere Werte des Intervalls sind jedoch auch sinnvoll, wie die spätere Verwendung zeigt.

[42] aber bezüglich des gleichen Dokuments

Vorerst soll lediglich eine abstrakte Beschreibung solcher $/n, k, t/$-Schemata erfolgen. Im Abschnitt 2.6.1 betrachten wir schließlich ein konkretes Beispiel von RSA-ähnlicher Gestalt, welches dann auch in `GnuDCN` eingesetzt wird.

Zur Initalisierung und Schlüsselerzeugung ist hier wieder ein Vermittler (ehemals Trent) notwendig, dem alle Subjekte $1 \leq i \leq n$ uneingeschränkt vertrauen:

> *Vermittleralgorithmus (Trusted Dealer):* Der Vermittler erzeugt das Schlüsselpaar $(K_{\mathrm{PUB}}, K_{\mathrm{SEC}})$, zerteilt K_{SEC} sofort in Stücke $K^1_{\mathrm{SEC}}, \ldots, K^n_{\mathrm{SEC}}$ (z. B. mit dem Polynomverfahren aus 2.5.2), generiert Verifikationsschlüssel $K_{\mathrm{VER}}, K^1_{\mathrm{V}}, \ldots, K^n_{\mathrm{V}}$ für Teilsignaturen und übermittelt jedem Subjekt i folgende Daten geheim:
>
> $$\left(K_{\mathrm{PUB}}, K^i_{\mathrm{SEC}}, K_{\mathrm{VER}}, K^1_{\mathrm{V}}, \ldots, K^n_{\mathrm{V}}\right)$$
>
> Ferner wird der öffentliche Schlüssel K_{PUB} Außenstehenden bekanntgegeben.

Jedes Subjekt i darf nun folgende Bestandteile des $/n, k, t/$-Schemas ausführen:

> *Teilsignaturerzeugungsalgorithmus:* Mit Hilfe der Geheiminformation K^i_{SEC} kann eine i-Teilsignatur über einem beliebigen Dokument m erzeugt werden.
>
> *Teilsignaturverifikationsalgorithmus:* Mit den Schlüsseln K_{VER} und K^j_{V} kann die Gültigkeit einer j-Teilsignatur über m festgestellt werden.
>
> *Teilsignaturkombinationsalgorithmus:* Zu einem Dokument mit k gültigen Teilsignaturen verschiedener Subjekte kann die Gesamtsignatur für m erzeugt werden.
>
> *Gesamtsignaturverifikationsalgorithmus:* Allein mit dem öffentlichen Schlüssel K_{PUB} kann die Gültigkeit einer Gesamtsignatur über m festgestellt werden.

Wir erwarten natürlich gewisse Mindestanforderungen bezüglich der Sicherheit:

Robustheit: Es ist für einen Angreifer berechnungstechnisch unmöglich, k gültige Teilsignaturen zu liefern, mit denen der Teilsignaturkombinationsalgorithmus eine ungültige Gesamtsignatur erzeugt.

Unfälschbarkeit: Es ist berechnungstechnisch so gut wie unmöglich, eine gültige Gesamtsignatur ohne die Mithilfe (Teilsignatur) von wenigstens $k - t$ ehrlichen Subjekten zu erhalten.

Wie können solche $/n, k, t/$-Schemata nun implementiert werden? Die triviale Lösung ist die Verwendung herkömmlicher Signaturen. Dabei werden Einzelunterschriften zu einer Menge zusammengefaßt, in welcher mindestens k davon gültig sein müssen. Allerdings hat diese naive Herangehensweise erheblichen Speicher-

und Rechenaufwand[43] zur Folge, insbesondere wenn die Anzahl der Teilnehmer groß ist. Ein effizientes Schema auf Basis des RSA-Algorithmus wird deshalb Gegenstand des folgenden Unterabschnitts sein. Es ist gleichzeitig die Hauptgrundlage für alle weiteren Protokolle dieser Arbeit.

2.6.1. RSA-ähnliche Schwellenwertsignaturen

Überraschenderweise wurde erst im Jahre 1999 von Victor Shoup [36] ein Algorithmus vorgestellt, der im asynchronen Umfeld sowohl praktikabel als auch sicher ist. Dieses neue Schema hat einige erfreuliche Eigenschaften:

(E0) Seine Unfälschbarkeit und Robustheit kann – unter der Annahme, daß RSA- und DDH-Problem schwierig sind – im *Random-Orakel-Modell* bewiesen werden. Damit ist zwar nicht unbedingte Sicherheit erreicht, in realistischen Umgebungen bleibt die Angriffsabwehr dennoch stark.

(E1) Viele Bestandteile[44] des Verfahrens sind *nicht-interaktiv*, d.h. sie benötigen keinerlei Kommunikationaustausch zwischen Subjekten und sind damit autonom durchführbar. Die Verbreitung getätigter Unterschriften für den Teilsignaturkombinationsalgorithmus kann hingegen asynchron erfolgen.[45]

(E2) Es gibt zu jedem Dokument/jeder Nachricht m nur *genau eine* gültige Gesamtsignatur[46], unabhängig davon, welche Unterzeichner dafür Teilunterschriften geleistet haben.[47] Naive Schemaimplementierungen, z.B. mit Signaturmengen der Kardinalität k, erfüllen diese Forderung gerade *nicht*, da eine unterschiedliche Zusammensetzung der Unterzeichnersubjekte zu diversen, ebenfalls gültigen Gesamtunterschriften führt.

(E3) Jede Teilsignatur hat *ähnliche Größenverhaltnisse* wie der Schlüssel K_{PUB}. Das diesbezüglich konstante Verhalten macht den Algorithmus auch für hohe Subjektzahlen in n oder k praktikabel.

Der Arbeit von V. Shoup folgend, soll zuerst das einfachere Basisschema vorgestellt werden. Später beschreiben wir die allgemeinere Version, welche dann auch in der `ShoupRsaDTSS`-Klasse[48] von `GnuDCN` Verwendung findet.

[43] Wegen der Mehrfachsignaturen ist der Aufwand für Transport und Verifizierung einer Gesamtunterschrift damit linear in k.

[44] Teilsignaturerzeugungsalgorithmus, Teilsignaturverifikationsalgorithmus, Gesamtsignaturverifikationsalgorithmus

[45] Diese Bedingung ist für unseren speziellen Fall zwingend notwendig, da wir zuverlässige Verteilung durch Byzantinische Übereinstimmung in einem asynchronen Netz erreichen wollen. Eine solche Eigenschaft könnte sich aber auch für andere Bereiche als nützlich erweisen.

[46] Sie ist sogar fast eine normale RSA-Unterschrift. (siehe Abschnitt 2.3 auf Seite 34)

[47] Wir brauchen diese Eigenschaft im Abschnitt 2.7, um aus dem Algorithmus ein sicheres Münzwurfschema für mehrere Personen zu gewinnen.

[48] Implementierungsspezifische Details finden sich im Anhang A.1 auf Seite 106.

$k = t + 1$-Schema: Der vertrauenswürdige Vermittler wählt zufällig zwei lange Primzahlen p, q mit $p = 2p' + 1$ und $q = 2q' + 1$, wobei p', q' sogenannte *„Sophie-Germain-Primzahlen“*, also selbst wieder prim sind. Unsere Zahlen p, q haben jetzt besondere kryptographische Eigenschaften und werden oft auch mit dem Prädikat ‚stark‘ versehen. Wir können sie effizient mit einem Verfahren gewinnen, das im Abschnitt *„Remarks on Key Generation“* von [38] beschrieben ist:

1. Erzeuge eine ungerade Zufallszahl p' der gewünschten Länge.
2. Prüfe parallel, ob entweder p' oder $2p' + 1$ durch kleine Primzahlen teilbar sind. Falls ja, beginne wieder bei Schritt 1.
3. Prüfe, ob die Zahl 2 ein Zeuge für die Zusammengesetztheit von p' im Miller-Rabin-Test[49] ist. Falls ja, gehe zurück zu Schritt 1.
4. Setze $p := 2p' + 1$ und prüfe, ob $2^{p'} \equiv \pm 1 \pmod{p}$ gilt. Falls nein, beginne wieder bei Schritt 1.
5. Wende ℓ-mal den Miller-Rabin-Test mit zufälligen Basen auf p' an, bis die Fehlerwahrscheinlichkeit hinreichend gesunken (z. B. $\epsilon \leq 2^{-80}$ entspricht $\ell := 40$) ist. Falls p' dabei durchfällt, gehe zurück zu Schritt 1.

Das Vorgehen bringt erhebliche Geschwindigkeitvorteile gegenüber der naiven Implementierung. Mathematische Gründe für diese Tatsache sollen hier aber nicht dargelegt werden, sondern können direkt [38] entnommen werden. Trotz trickreicher Optimierung bleibt die Schlüsselerzeugung insgesamt sehr langsam. Da sie aber nur für eine Initalisierung notwendig ist, scheint es dennoch akzeptabel.

Aus praktischen Erwägungen (vgl. Primfaktorisierung) gilt jedoch auch immer $p', q' >> n$. Damit berechnet der Vermittler $\nu := p \cdot q$ sowie $\mu := p' \cdot q'$. Er wählt den öffentlichen Exponenten $e > n$ als relativ kleine Primzahl und setzt $K_{\text{PUB}} := (\nu, e)$. Weiterhin berechnet er $d \in \mathbb{Z}_\mu$ so, daß $de \equiv 1 \pmod{\mu}$ gilt. Dann wird das Teilungspolynom $f(x) = \sum_{i=0}^{k-1} a_i x^i$ gebildet. Dazu wählt der Vermittler zufällige Koeffizienten $a_i \in \mathbb{Z}_\mu$ für $1 \leq i \leq k - 1$ und besetzt $a_0 := d$ mit dem privaten Exponenten. Für jedes Subjekt $1 \leq i \leq n$ berechnet er

$$s_i := f(i) \bmod \mu \,. \tag{1}$$

Diese s_i bilden die geheimen Schlüsselteile K^i_{SEC} des Schwellenwertsignaturverfahrens. Unser Vermittler wählt nun zufällig einen Quadratrest[50] $v \in \mathbb{QR}_\nu$ und berechnet für alle s_i einen Verifikationsschlüssel K^i_{V} durch $v_i := v^{s_i} \bmod \nu$, der wieder in $\mathbb{QR}_\nu$ liegt. Nun können $K_{\text{PUB}} := (\nu, e)$, $K_{\text{VER}} := v$ und $K^i_{\text{V}} := v_i$ öffentlich verteilt werden. Hingegen darf $K^i_{\text{SEC}} := s_i$ nur dem jeweiligen Teilnehmer i

[49] ein probabilistischer Test auf Primzahleigenschaft (siehe Standardliteratur [8, 17])

[50] Man nennt den Wert $a \in \mathbb{Z}_n^*$ *quadartischer Rest modulo* n, falls eine andere Zahl $x \in \mathbb{Z}_n$ mit $x^2 \equiv a \pmod{n}$ existiert.

zugänglich sein. Aus den Werten v_i sind die zugehörigen, geheimen s_i für keinen Angreifer berechenbar, weil das Problem der diskreten Logarithmen in $\mathbb{QR}_\nu$ ebenfalls schwer ist. Mit diesen Daten sind folgende Schemabestandteile ausführbar:

Seien nun $\Delta := n!$ sowie die speziellen Lagrange-Konstanten $\lambda_{0,j}^S := \Delta \frac{\prod_{j' \in S \setminus \{j\}} -j'}{\prod_{j' \in S \setminus \{j\}} (j-j')}$ für jede k-elementige Teilmenge $S \subseteq \{1, \ldots, n\}$ mit Indizes $j \in S$ gegeben.

Teilsignaturerzeugungsalgorithmus: Man berechnet zuerst einen „Fingerabdruck" des Dokuments $x := h(m)$, wobei die kryptographische Hashfunktion h zum Einsatz kommt. Die i-Teilsignatur ergibt sich dann durch:

$$x_i := x^{2\Delta s_i} \bmod \nu \qquad (\in \mathbb{QR}_\nu) \tag{2}$$

Zur Verifikation ihrer Gültigkeit ist noch ein „Beweis der Korrektheit" anzufügen. Dazu zeigt man, daß der diskrete Logarithmus modulo ν von x_i^2 zur Basis

$$\tilde{x} := x^{4\Delta} \bmod \nu \tag{3}$$

dem diskreten Logarithmus modulo ν von v_i zur Basis v entspricht. Hierfür existiert ein bekanntes interaktives Protokoll [39], welches wir mit Hilfe einer Hashfunktion h' in einen nicht-interaktiven Algorithmus verwandeln. Dabei muß sichergestellt sein, daß man in der Gruppe $\mathbb{QR}_\nu$ arbeitet. Sei nun $|\nu|$ die Länge von ν in Bit und L_1 die Ausgabelänge dieser Hashfunktion h' (auch in Bit). Beide Einwegabbildungen können durchaus gleich sein, d. h. $h = h'$; eine Mehrargumentigkeit wird ggf. durch Hintereinanderschreibung (Konkatnation) simuliert. Das Subjekt i wählt eine Zufallszahl $r \in \{0, \ldots, 2^{|\nu|+2L_1} - 1\}$ und berechnet:

$$v' = v^r \bmod \nu, \quad x' = \tilde{x}^r \bmod \nu, \quad c = h'(v, \tilde{x}, v_i, x_i^2 \bmod \nu, v', x'), \quad z = s_i c + r$$

Der Korrektheitsbeweis ist dann das Tupel (z, c); eine komplette i-Teilsignatur besteht somit aus $(x_i, (z, c))$.

Teilsignaturverifikationsalgorithmus: Zur Überprüfung einer Teilunterschrift des Subjekts j über Dokument m wird zuerst wieder der Hashwert $x := h(m)$ und später die Logarithmusbasis $\tilde{x} := x^{4\Delta} \bmod \nu$ bestimmt. Nun kann mit unseren öffentlich bekannten Verifikationsschlüsseln v, v_j auch der Term

$$h'(v, \tilde{x}, v_j, x_j^2 \bmod \nu, v^z v_j^{-c}, \tilde{x}^z x_j^{-2c})$$

berechnet werden. Falls dessen Gleichheit mit c gilt, ist die j-Teilsignatur gültig.

Teilsignaturkombinationsalgorithmus: Angenommen ein Subjekt hat k gültige Teilunterschriften der Unterzeichner $S := \{i_1, \ldots, i_k\} \subseteq \{1, \ldots, n\}$ gesammelt, wobei $\forall j \in S : x_j^2 = x^{4\Delta s_j}$ für den Hashwert $x \in \mathbb{Z}_\nu^*$ mit $x = h(m)$ gilt. Man berechnet durch Lagrange-Interpolation (im Exponenten)

$$w := x_{i_1}^{2\lambda_{0,i_1}^S} \cdot x_{i_2}^{2\lambda_{0,i_2}^S} \cdot \cdots \cdot x_{i_k}^{2\lambda_{0,i_k}^S} \bmod \nu \tag{4}$$

und Werte a, b, welche die Gleichung $4\Delta^2 a + eb = 1$ erfüllen. Aufgrund der geltenden Bedingung $\mathrm{ggT}(4\Delta^2, e) = 1$, ist letzteres leicht mittels *Erweitertem Euklidschen Algorithmus* zu bewerkstelligen. Die Gesamtsignatur kann schließlich durch $y := w^a x^b \bmod \nu$ gebildet werden.

Gesamtsignaturverifikationsalgorithmus: Eine Signatur y kann man analog zum RSA-Verfahren durch Gültigkeitsprüfung der Beziehung $y^e \equiv x \pmod{\nu}$ verifizieren. Hier ist wieder $x := h(m)$ der selbst bestimmte Hashwert des Dokuments.

Wir wollen jetzt die Korrektheit des obigen Verfahrens betrachten und nachher eine Analyse seiner Sicherheit anschließen. Dabei folgen wir natürlich wieder [36]:

Man arbeitet hauptsächlich in einer zyklischen Gruppe $\mathbb{QR}_\nu$ der Ordnung μ. Aufgrund des Satzes von Euler gilt für alle Elemente $x \in \mathbb{QR}_\nu$ die Beziehung $x^\mu \equiv 1 \pmod{\nu}$ und somit $\forall x \in \mathbb{Z}_\nu^*, : x^{2\mu} \equiv 1 \pmod{\nu}$. Normale Gruppenoperationen werden modulo ν ausgeführt, während die Arithmetik im Exponenten quasi modulo μ erfolgt.

Der Vermittler wählt den globalen Wert $v \in \mathbb{QR}_\nu$ zufällig, so daß er mit hoher Wahrscheinlichkeit ein Erzeuger der Gruppe $\mathbb{QR}_\nu$ ist.[51] Deswegen sind auch die Geheimnisse $s_i \bmod \mu$ durch unsere Verifikationsschlüssel $v_i := v^{s_i} \bmod \nu$ vollständig festgelegt.

Für jede Teilmenge von k Punkten aus $\{0, \ldots, n\}$ bestimmen die Werte des Polynoms f an diesen Stellen auch dessen Koeffizienten fast immer eindeutig. Dies folgt aus der Tatsache, daß die zugehörige Vandermonde-Matrix i. A. modulo μ invertierbar ist, weil es keine kleinen Primfaktoren besitzt. Sind jedoch nur Funktionswerte von $k-1$ Punkten bekannt, bleibt die Verteilung hingegen gleichförmig und unabhängig in $\mathbb{Z}_\mu$.

Seien $\Delta := n!$ sowie alle Lagrange-Konstanten $\lambda_{i,j}^S := \Delta \frac{\prod_{j' \in S \setminus \{j\}} (i-j')}{\prod_{j' \in S \setminus \{j\}} (j-j')}$ für beliebige k-elementige Teilmengen $S \subseteq \{0, \ldots, n\}$ mit Indizes $i \in \{0, \ldots, n\} \setminus S$, $j \in S$ gegeben. Das Δ muß hier eingeführt werden, damit definitiv alle $\lambda_{i,j}^S$ ganze Zahlen aus $\mathbb{Z}$ sind.

Mit Hilfe der Interpolationsformel können wir schließen:

$$\Delta \cdot f(i) \equiv \sum_{j \in S} \lambda_{i,j}^S f(j) \pmod{\mu} \tag{5}$$

Deswegen erhalten wir im Teilsignaturkombinationsalgorithmus zuerst

$$w \equiv x^{2\Delta \sum_{i_j} 2\lambda_{0,i_j}^S s_{i_j}} \equiv x^{4\Delta^2 d} \pmod{\nu}$$

und schließlich

$$y \equiv w^a x^b \equiv x^{4\Delta^2 da + b} \pmod{\nu}.$$

Da unser Gesamtgeheimnis d nur im Exponenten berechnet wird, bleibt das Verfahren für weitere Dokumente anwendbar. Allerdings ist auch klar, daß einmal erzeugte Unterschriften später mißbräuchlich für das selbe Dokument eingesetzt werden könnten. Eine Zufallskomponente oder die Durchnummerierung verhindert derartige Angriffe.

[51] Genau $\varphi(\mu) = (p'-1)(q'-1)$ der insgesamt μ Elemente von $\mathbb{QR}_\nu$ sind Erzeuger.

Wegen $ed \equiv 1 \pmod{\mu}$ existiert ein $\ell \in \mathbb{Z}$, so daß $ed = 1 + \ell\mu$. Damit folgt bei Verifikation unserer Gesamtunterschrift natürlich

$$y^e \equiv x^{4\Delta^2 aed+eb} \equiv x^{4\Delta^2 a(1+\ell\mu)+eb} \equiv x^{4\Delta^2 a+eb} \underbrace{x^{4\Delta^2 \ell\mu}}_{=1} \equiv x \pmod{\nu}.$$

Satz 2.1 *Bei $k := t+1$ und im Random-Orakel-Modell für h' ergibt obiges Schema ein sicheres Schwellenwertsignaturverfahren (d. h. robust und unfälschbar), falls RSA-Standardsignaturen ihrerseits sicher sind.*

Random-Orakel-Modell: Dieses nützliche Analysewerkzeug wird in vielen Sicherheitsbeweisen kryptographischer Schemata verwendet. Der Definition aus [40] folgend, handelt es sich um ein idealisiertes Modell, wo kryptographische Hashfunktionen (siehe Abschnitt 2.3) durch „Black Boxen“ ersetzt werden. Teilnehmer und Angreifer werten diese Einwegabbildungen also nicht mehr direkt aus, sondern haben lediglich Zugriff auf ein Orakel, dessen interne Funktionsweise sie nicht kennen. Es liefert für beliebig lange Eingaben eine endliche Folge von Bits, welche im Modell durch eine echte Zufallsfunktion erzeugt werden. Für jeweils gleiche Argumentwerte sind natürlich auch die Ausgaben des Orakels dieselben. Im Gegensatz zu realen Hashfunktionen sind verschiedene Rückgabewerte aber weitgehend unabhängig voneinander, da ja eine Zufallsfunktion zum Einsatz kommt. Zwar ist ein Beweis innerhalb dieses Modells noch kein Garant für hohe Sicherheit, weil die in der ‚richtigen Welt‘ verwendeten Funktionen gerade auch dem Angreifer bekannt sind und damit keine echten Zufallsabbildungen darstellen, jedoch zeigt er zumindest eine gewisse heuristische Tendenz. Auf jeden Fall ist ein solches Kriterium besser als gar kein Sicherheitsbeweis.

Beweisidee: Wir gehen von der Sichtweise des Angreifers aus, welcher versucht, Robustheit oder Unfälschbarkeit zu verletzen. Seien hier o. B. d. A. $i_1, \ldots, i_{k-1}$ die von ihm kontrollierten Subjekte (vgl. $k = t+1$). Laut Interpolation gilt für alle $1 \le i \le n$:

$$\begin{aligned} s_i &\equiv f(i) \pmod{\mu} \\ d &\equiv f(0) \pmod{\mu} \end{aligned}$$

Erstaunlicherweise kann unser Angreifer zu gegebenen $x, y \in \mathbb{Z}_\nu^*$ mit $y^e = x$ und $S = \{0, i_1, \ldots, i_{k-1}\}$ problemlos das Bruchstück

$$\begin{aligned} x_\ell = x^{2\Delta s_\ell} &\equiv x^{2\Delta f(\ell)+2\Delta\xi\mu} \equiv x^{2(\sum_{i_j \in S} \lambda_{\ell,i_j}^S s_{i_j})+2\zeta\mu+2\Delta\xi\mu} \equiv \quad \ldots \\ \ldots &\equiv y^{2e(\lambda_{\ell,0}^S s_0+\lambda_{\ell,i_1}^S s_{i_1}+\cdots+\lambda_{\ell,i_{k-1}}^S s_{i_{k-1}})+2e\zeta\mu+2e\Delta\xi\mu} \equiv \quad \ldots \\ \ldots &\equiv y^{2(\lambda_{\ell,0}^S ed+e(\lambda_{\ell,i_1}^S s_{i_1}+\cdots+\lambda_{\ell,i_{k-1}}^S s_{i_{k-1}}))+2e\zeta\mu+2e\Delta\xi\mu} \equiv \quad \ldots \\ \ldots &\equiv y^{2(\lambda_{\ell,0}^S+e(\lambda_{\ell,i_1}^S s_{i_1}+\cdots+\lambda_{\ell,i_{k-1}}^S s_{i_{k-1}}))} \underbrace{y^{2\lambda_{\ell,0}^S\psi\mu+2e\zeta\mu+2e\Delta\xi\mu}}_{=1} \pmod{\nu} \end{aligned}$$

der ℓ-Teilunterschrift ohne Kenntnis von s_ℓ konstruieren. Dieser Angriff hat aber kaum Auswirkungen auf die Sicherheit, weil das x_ℓ erstens nur nachträglich und zweitens ohne Korrektheitsbeweis (z, c) zu einer vorliegenden Gesamtunterschrift y geliefert wird.

Der „Beweis der Korrektheit“ einer Teilunterschrift sollte die Eigenschaften *Korrektheit (soundness)* und *Geheimnisbewahrung (statistical zero-knowledge)* erfüllen. Hier benötigen wir das Random-Orakel-Modell für die Hashfunktion h'. Zuerst soll die Korrektheit gezeigt werden, d. h. ein Angreifer darf nur mit vernachlässigbarer Wahrscheinlichkeit den gültigen Korrektheitsbeweis zu einer falschen Teilsignatur liefern können. Sei x, x_ℓ zusammen mit dem richtigen Beweis (z, c) gegeben, wobei $c = h'(v, \tilde{x}, v_\ell, x_\ell^2, v', x')$ und

$$\tilde{x} = x^{4\Delta}, \quad v' = v^z v_\ell^{-c}, \quad x' = \tilde{x}^z x_\ell^{-2c}$$

sind. Alle vorkommenden Werte liegen offensichtlich in $\mathbb{QR}_\nu$, und wir nehmen auch weiterhin an, daß v Erzeuger dieser zyklischen Gruppe ist. Einheitlich haben wir also

$$\tilde{x} = v^\alpha, \quad v_\ell = v^{s_\ell}, \quad x_\ell^2 = v^\beta, \quad v' = v^\gamma, \quad x' = v^\delta$$

für bestimmte ganze Zahlen $\alpha, \beta, \gamma, \delta$. Weiterhin gilt bezüglich der Exponenten

$$\begin{aligned} z - cs_\ell &\equiv \gamma \pmod{\mu}, \\ z\alpha - c\beta &\equiv \delta \pmod{\mu}, \end{aligned}$$

das bei Multiplikation der ersten Kongruenz mit α und Subtraktion der zweiten zu

$$c(\beta - s_\ell \alpha) \equiv \alpha\gamma - \delta \pmod{\mu} \tag{6}$$

vereinfacht werden kann. Eine ℓ-Teilunterschrift ist genau dann korrekt erstellt, wenn

$$\beta \equiv s_\ell \alpha \pmod{\mu} \tag{7}$$

zutrifft. Das impliziert über $x_\ell^2 \equiv \tilde{x}^{s_\ell} \equiv x^{4\Delta s_\ell} \pmod{\nu}$ quasi die Richtigkeit bezüglich des verwendeten Nachrichtenhashwertes x. Im Angriffsfall sollte Kongruenz (7) also gerade nicht gelten, weshalb sie entweder für mod p' oder mod q' falsch sein muß. Damit ist der Wert c in Formel (6) eindeutig durch eine dieser Primzahlen bestimmt. Weil c im Random-Orakel-Modell jedoch gleichförmig verteilt und unabhängig von den Eingaben der Funktion h' ist, kann der Angriff höchstens mit vernachlässigbarer Wahrscheinlichkeit vom Verifizierer unbemerkt bleiben.

Jetzt soll die Geheimnisbewahrung betrachtet werden. Dafür wird ein sogenannter Simulator konstruiert, welcher die Angreiferseite ohne das Wissen von s_ℓ modelliert. Auch hier spielt das Random-Orakel-Modell eine Rolle: Immer wenn unser Orakel zu einer neuen Eingabe befragt wird, gibt es demnach eine Zufallsfolge aus; falls jedoch die Eingabe schon einmal vorkam, wird nur der zugehörige Wert geliefert. Erzeugt ein ehrlicher Teilnehmer ℓ den Korrektheitsbeweis für x_ℓ, wählt der Simulator $c \in \{0, \ldots, 2^{L_1} - 1\}$ und $z \in \{0, \ldots, 2^{|\nu|+2L_1} - 1\}$ zufällig aus. Zu gegebenen Werten x, x_ℓ wird dieses c als Ausgabe des Orakels an der Stelle $(v, \tilde{x}, v_\ell, x_\ell^2, v^z v_\ell^{-c}, \tilde{x}^z x_\ell^{-2c})$ definiert. Hat das ehrliche Subjekt sein r wirklich zufällig bestimmt, kann nur mit vernachlässigbarer Wahrscheinlichkeit der Wert c schon einmal vorgekommen sein. Der Simulator erzeugt deshalb ein Verhalten, das nahe bei einer statistisch perfekten Gleichverteilung liegt.

Aus der Korrektheit des Verfahrens resultiert die Robustheit des Schemas, aus der Geheimnisbewahrung seine Unfälschbarkeit. Letzteres aber nur unter der Voraussetzung, daß RSA-Standardsignaturen resistent gegen adaptive Angriffe mit gewähltem Klartext sind. Diese Annahme kann wiederum im Random-Orakel-Modell für h bewiesen werden, wenn man das RSA-Problem (vgl. Seite 24) als hart betrachtet.

$k \geq t+1$-Schema: Diese allgemeinere Variante vereinfacht den herkömmlichen Ansatz, indem sie das Random-Orakel-Modell vollständig für h und h' ausnutzt. Das resultierende Schema ist außerdem wesentlich effizienter. Es sollen hier nur die Veränderungen gegenüber dem $k = t + 1$ Verfahren dargestellt werden:

Der vertrauensvolle Vermittler berechnet die Geheimteile s_i statt mit Formel (1) durch

$$s_i := f(i)\Delta^{-1} \bmod \mu\,.$$

Den öffentlichen Verifikationsschlüsseln wird ein weiteres Element $u \in \mathbb{Z}_\nu^*$ hinzugefügt, dessen Jacobi-Symbol[52] negativ ist. Da das Symbol effizient berechnet werden kann, liefert zufällige Suche über $\mathbb{Z}_\nu^*$ schnell ein entsprechendes u.

Unsere Algorithmen werden folgendermaßen geändert: Sei $\hat{x} := h(m) \in \mathbb{Z}_\nu^*$, wir setzen

$$x := \begin{cases} \hat{x} \bmod \nu & \text{falls } \left(\frac{\hat{x}}{\nu}\right) = 1 \\ \hat{x}u^e \bmod \nu & \text{falls } \left(\frac{\hat{x}}{\nu}\right) = -1\,. \end{cases}$$

Damit ist das Jacobi-Symbol von x immer positiv. Außerdem gibt es noch die Vereinfachungen [vgl. Formeln (2) und (3)]

$$x_i := x^{2s_i} \bmod \nu, \quad \tilde{x} := x^4 \bmod \nu$$

und eine Änderung bei der Erzeugung der Gesamtunterschrift im *Teilsignaturkombinationsalgorithmus*. Dort werden die Werte $a, b \in \mathbb{Z}$ genau so gewählt, daß sie nun die Gleichung $4a + eb = 1$ erfüllen. Wurde vom Vermittler $e > n$ als kleine Primzahl gewählt, sind auch a, b nicht sehr groß und das Verfahren damit effizient. Das seltsame Auftreten von Δ wurde also größtenteils beseitigt. Trotz obiger Anpassungen ist y immernoch mit dem Standardverfahren a'la RSA verifizierbar.

Alle übrigen Bestandteile des Schemas laufen wie zuvor ab, benutzen jedoch immer die neuen Formeln, wenn sie x, x_i oder $\tilde{x}$ bestimmen müssen.

Satz 2.2 *Bei $k \geq t + 1$ und im Random-Orakel-Modell für h, h' ergibt obiges Schema ein sicheres Schwellenwertsignaturverfahren (d. h. robust und unfälschbar), falls jeweils RSA- und DDH-Problem schwer sind. Gilt hingegen lediglich $k = t + 1$, reicht bereits die RSA-Annahme aus.*

Auf den umfangreichen Beweis wird an dieser Stelle verzichtet; er kann der Orginalarbeit [36] entnommen werden. Es bleibt nur noch anzumerken, daß DDH hier eine sinnvolle Annahme darstellt, weil $|\mathbb{QR}_\nu|$ keine kleinen Primfaktoren besitzt.

[52] Das Jacobi-Symbol ist eine Verallgemeinerung des Legendre-Symbols. Während die Entscheidung der Quadratresteigenschaft damit nicht möglich ist, deutet sein negativer Wert aber immer auf einen quadratischen Nicht-Rest hin:

$$\begin{aligned} \left(\tfrac{a}{\nu}\right) = 1 \quad &\Rightarrow \quad a \in \mathbb{QR}_\nu \vee a \in \mathbb{NQR}_\nu \\ \left(\tfrac{a}{\nu}\right) = -1 \quad &\Rightarrow \quad a \in \mathbb{NQR}_\nu \end{aligned}$$

Glücklicherweise kann das Jacobi-Symbol auch ohne Primfaktorisierung effizient mit Komplexität $O(\nu^2)$ berechnet werden. Die obige Verwendung von u stellt sicher, daß x auf jeden Fall in $\mathbb{J}_\nu$ liegt, wobei $\mathbb{QR}_\nu \subset \mathbb{J}_\nu \subset \mathbb{Z}_\nu^*$ gilt.

2.7. Münzwurfschemata für mehrere Beteiligte

Wir benötigen noch eine weitere kryptographische Primitive für das ABBA-Protokoll. Um deterministische Angriffsversuche zu verhindern, ist eine globale Zufallsquelle für alle Teilnehmer notwendig. Analog zu 2.6 sollen hier deren Eigenschaften, Bestandteile und Sicherheitsanforderungen formalisiert werden [37]:

Definition 2.8 ($/n, k, t/$-Schwellenwertmünzwurfschema)
Solche Schemata bestehen aus insgesamt n Subjekten, wobei bis zu t von ihnen bösartige Absichten verfolgen oder durch Angreifer kontrolliert werden könnten. Die Zusammenarbeit von mindestens k Subjekten ist notwendig und ausreichend, um den Ausgang (Kopf oder Zahl) eines nichtvorhersagbaren Münzwurfs mit der Bezeichnung C zu erfahren.

Der sogenannte „Name der Münze“ $C \in \{0,1\}^*$ ist eine beliebige Zeichenkette, welche das gemeinsame Experiment parameterisiert. Wir können uns dieses Modell deshalb als unvorhersagbare Funktion $F : \{0,1\}^* \to \{0,1\}$ vorstellen, die nur bei Zusammenarbeit von k Subjekten berechnet werden kann. Gleiche Werte des Arguments C führen logischerweise zum selben Resultat $F(C)$.

Durch die zugelassenen t Betrüger sollte der Schwellenwert k wieder vernüftigerweise durch $t < k \leq n - t$ beschränkt sein. Die Festlegung $k := n - t$ wird später getroffen, um Unvorhersagbarkeit bei maximaler Robustheit zu garantieren.

Ein Subjekt i darf folgende Bestandteile des $/n, k, t/$-Schemas ausführen:

Münzteilerzeugungsalgorithmus: Mit Hilfe einer Geheiminformation kann der i-Münzteil für beliebige Namen C erzeugt werden.

Münzteilverifikationsalgorithmus: Durch Verifikationsschlüssel für j und dem Namen C kann die Gültigkeit eines j-Münzteils festgestellt werden.

Münzteilkombinationsalgorithmus: Aufgrund k gültiger – aber verschiedener – Münzteile von C kann die zugehörige Gesamtmünze konstruiert und ihr Wert $F(C)$ berechnet werden.

Wir erwarten ebenfalls gewisse Mindestanforderungen bezüglich der Sicherheit:

Robustheit: Es ist für einen Angreifer berechnungstechnisch unmöglich, einen Namen C und k gültige Münzteile zu liefern, mit denen der Münzteilkombinationsalgorithmus einen Wert ungleich $F(C)$ erzeugt.

Unvorhersagbarkeit: Es ist berechnungstechnisch so gut wie unmöglich, ohne die Mithilfe (Münzteile über C) von wenigstens $k - t$ ehrlichen Subjekten den Wert einer noch nicht berechneten Münze $F(C) \in \{0,1\}$ mit höherer Wahrscheinlichkeit als $1/2$ vorherzusagen. Diese Forderung impliziert natürlich sofort, daß auch

Sequenzen von mehreren Münzwürfen unvorhersagbar sind. Der Fehlervektor der Angreifervorhersage darf also nicht von einem Zufallsvektor unterscheidbar sein.

Aus jedem Schwellenwertsignaturschema mit zusätzlichen Eigenschaften (E1) sowie (E2) kann ein nicht-interaktives $/n, k, t/$-Schwellenwertmünzwurfschema gewonnen werden, indem man eine kryptographische Hashfunktion auf die Gesamtsignatur über C anwendet und ihre Ausgabe als Münzwurfwert $F(C)$ interpretiert. Die Forderung nach Unvorhersagbarkeit ist dann unter den Annahmen des Random-Orakel-Modells garantiert.

2.8. Sonstige kryptographische Werkzeuge

Für unsere DC-Netz-Implementierung `GnuDCN` besprechen wir jetzt noch einige allgemein bekannte kryptographische Hilfsmittel. Die zugehörigen Sicherheitsbeweise wurden in vielen Lehrbüchern [17, 8] ausführlich diskutiert und sollen deshalb hier nicht wiederholt werden.

2.8.1. Sichere Festlegschemata für Bitsequenzen

Manchmal ist es notwendig, daß ein Teilnehmer sich vor der Veröffentlichung einer Aussage $b \in \{0,1\}^*$ darauf festlegt. Wir wollen nachträgliche Änderungen erkennen, wobei die Festlegung selbst keine Information über das fixierte b preisgeben darf. Folgendes Beispiel erklärt den Sachverhalt:

Beispiel 2.1 *Alice möchte sich auf die Vorhersage eines Bits festlegen, ihre Wahl aber nicht sofort offenbaren. Bob möchte verhindern, daß Alice im Nachgang ihre Meinung ändert.*

Digitale Signatur löst dieses Problem nur unvollständig, insbesondere wenn mögliche Aussagen bzw. frühere Festlegungen öffentlich bekannt sind. Ein kryptographisch sicheres Festlegschema (commitment scheme) auf Basis von Einweghashfunktionen (siehe Abschnitt 2.3) schafft hier Abhilfe. Seine Implementierung als Klasse `SchneierHashBCS` wird im Anhang A.2 erläutert.

Dieses Verfahren [8] verlangt zusätzliche Interaktion zwischen den beteiligten Subjekten, kann aber vollkommen asynchron durchgeführt werden:

1. Alice erzeugt zwei Zeichenketten R_1, R_2 aus Zufallsbits.
2. Sie bildet die Nachricht $m := (R_1, R_2, b)$, wobei $b \in \{0,1\}$ ihr gewähltes Bit (bzw. eine Sequenz $b \in \{0,1\}^*$) ist.

3. Alice wendet eine Einweghashfunktion h auf die gesamte Nachricht m an und sendet $(h(m), R_1)$ an Bob. Damit hat sie sich festgelegt.Bob kann wegen der Unumkehrbarkeit von h Alices Aussage b nicht ermitteln.

Zur Offenlegung wird das Protokoll wie folgt weitergeführt:

4. Alice sendet die Nachricht m an Bob.
5. Nach Erhalt der Mitteilung kann Bob h darauf anwenden und vergleicht den Hashwert sowie R_1 mit den in Schritt 3 erhaltenen Daten. Bei Übereinstimmung akzeptiert er b als unveränderte und vorher festgelegte Aussage.

Fakt 2.2
Im Random-Orakel-Modell für h ist obiges Festlegschema kryptographisch sicher.

Diskussion: Alice kann hierbei kaum betrügen; sie müßte für geändertes $\bar{b}$ ein $\tilde{R}_2$ mit $h(R_1, R_2, b) = h(R_1, \tilde{R}_2, \bar{b})$ finden, was aber bei ausreichender Länge der R_i sowie einer kollisionsresistenten Hashfunktion h sehr unwahrscheinlich ist.

Die Zeichenkette R_2 wird eingeführt, damit Bob den Wert b nicht vorzeitig mittels Probieren erraten kann. Alice muß deshalb R_2 bis zur Offenlegung geheimhalten, weil Bob sonst $h(R_1, R_2, b)$ berechnen und durch Vergleich mit dem erhaltenen Hashwert die fixierte Aussage schon im Schritt 3 erkennen könnte.

2.8.2. Generatoren für sichere Pseudozufallsbitsequenzen

Als letzte Grundlage behandeln wir komplexitätstheoretisch sichere Pseudozufallsgeneratoren, welche uns als Ersatz für die ‚idealen Münzen' im DC-Netz dienen. Während über herkömmliche Mechanismen zur Erzeugung von ‚gutem Zufall' keine brauchbare Sicherheitsaussage getroffen werden kann, beruht die Unvorhersagbarkeit des später beschriebenen Verfahrens auf dem schwierigen Problem der Faktorisierung (vgl. Seite 23 und Bemerkungen aus [17]).

Definition 2.9 (Pseudozufallsbitgenerator)
Ein Pseudozufallsbitgenerator (PRBG) ist ein deterministischer Algorithmus, der zu einer gegebenen, echt zufälligen Binärsequenz (Startwert) fester Länge die Ausgabesequenz $z_1, z_2, \ldots$ mit $z_i \in \{0, 1\}$ liefert. Jene erfüllt Eigenschaften hinsichtlich ihrer ‚Zufälligkeit', die mit statistischen Tests überprüft werden können.

Definition 2.10 (kryptographisch sicherer Pseudozufallsbitgenerator)
Ein PRBG, der unter einer zahlentheoretischen Annahme folgenden Test besteht, heißt „kryptographisch sicher": Es darf kein Polynomialzeitalgorithmus existieren, welcher zu gegebenen Werten der Ausgabesequenz $z_1, z_2, \ldots, z_\ell$ das folgende Bit $z_{\ell+1}$ mit einer Wahrscheinlichkeit von mehr als $1/2$ vorhersagen kann.

Einen kryptographisch sicheren Generator wollen wir jetzt kurz betrachten. Seine Implementierung als Klasse `BlumBlumShubPRBG` wird im Anhang A.3 erläutert.

Verfahren von Blum, Blum und Shub: Dieser Algorithmus wurde von den Mathematikern L. Blum, M. Blum und M. Shub [41] vorgestellt, welche auch die Äquivalenz zum *Problem der Quadratischen Reste* (QREST)[53] zeigten. Erst später entdeckten andere Wissenschaftler, daß eine Vorhersage seiner Ausgabe sogar genauso schwierig ist, wie die Faktorisierung des Moduls n.

Um eine Sequenz von Pseudozufallsbits $z_1, z_2, \ldots, z_\ell$ der Länge ℓ zu generieren, werden folgende Schritte ausgeführt:

1. *Initalisierung:* Erzeuge zufällig zwei ausreichend große Primzahlen $p, q \in \mathbb{P}$, mit $p \neq q$ und $p, q \equiv 3 \pmod 4$. Bilde den Modul $n := p \cdot q$ (Blumsche Zahl) und vernichte ggf. p, q wieder.
2. Wähle einen zufälligen Startwert $s \in \mathbb{Z}_n^*$ und berechne $x_0 := s^2 \bmod n$.
3. Für $1 \leq i \leq \ell$,
   a) setze $x_i := x_{i-1}^2 \bmod n$ und
   b) wähle $z_i := x_i \bmod 2$ (am wenigsten signifikantes Bit von x_i).
4. $z_1, z_2, \ldots, z_\ell$ mit $z_i \in \{0, 1\}$ ist dann die Ausgabesequenz des Generators.

Die Forderung $p, q \equiv 3 \pmod 4$ garantiert uns, daß jedes $x_i \in \mathbb{QR}_n$ wenigstens eine Quadratwurzel besitzt, die selbst wieder Quadratrest modulo n ist. Damit bleibt die Gleichverteilung in der Ausgabe bewahrt und das Verfahren sicher, da quadratische Wurzeln modulo n nur mit Kenntnis von p und q berechenbar sind. Wenn zusätzlich der Wert $\mathrm{ggT}(\varphi(p-1), \varphi(q-1))$ klein ist[54], wird die Zykluslänge des Generators groß, so daß sich unsere Folge nicht allzu schnell wiederholt.

Fakt 2.3 (Vazirani [42])
Das Verfahren von Blum, Blum *und* Shub *ist ein kryptographisch sicherer Pseudozufallsbitgenerator im Sinne unserer Definition 2.10.*

Alle komplexitätstheoretisch sicheren Verfahren zur Pseudozufallszahlerzeugung sind langsam, weil sie für jedes Ausgabebit mindestens eine aufwendige modulare Exponentiation (hier modulare Quadrierung) benötigen. Es gab deshalb viele Untersuchungen [17], ob im Schritt 3. b) nicht mehr als nur ein Bit gleichzeitig genutzt werden könnte. Dabei stellte sich heraus, daß die Ausgabe sicher bleibt,

[53] Zu einer gegebenen Zahl $a \in \mathbb{Z}_n^*$ mit positivem Jacobi-Symbol $\left(\frac{a}{n}\right)$ soll die Fragestellung $a \in \mathbb{QR}_n$ entschieden werden. Ist die Faktorsierung von n bekannt, ist das Problem in Polynomialzeit lösbar, d. h. QREST $\leq_{\mathcal{P}}$ FAKTOR.

[54] Leider läßt sich dieses Kriterium nicht effizient überprüfen, da hierfür die Faktorisierung von $p-1$ bzw. $q-1$ bestimmt werden muß.

selbst wenn die j am wenigsten signifikanten Bits von x_i verwendet werden. Dabei ist j aber durch eine Funktion der Klasse $O(\log_2 \log_2 n)$ nach oben beschränkt.

3. Zuverlässige Verteilung

So wird die Zuverlässigkeit zum offenen Kreditbrief auf die allgemeine Achtung und das Vertrauen der Menschen.
— Samuel Smiles

Um Zuverlässigkeit in *Verteilten Systemen* zu erreichen, benötigen wir Protokolle, welche die Funktionalität auch im Fehlerfall sicherstellen. Hier ist jedoch Zusammenarbeit zwischen Prozessen verschiedener Maschinen notwendig, die wiederum fundamental vom Problem der Übereinstimmung verteilter Daten abhängt. So sind beispielsweise in replizierenden Dateisystemen zuverlässige Vergleiche bezüglich der Gültigkeit kopierter Informationen angeraten, während in Flugzeugsteuerungssystemen sogar eine gemeinsame Entscheidung redundanter Bordmodule herbeigeführt werden muß. Auch verteilte Datenbanken sollten sich über getätigte Transaktionen einig sein, damit ihre gespeicherten Informationen konsistent bleiben. Neben einer gewissen Korrektheit der Daten ist also vorallem die Konsistenz wichtig, damit alle beteiligten Rechner auf dieser Basis zur gleichen Entscheidung kommen. [43]

Ein offensichtlicher Ansatz zur Übereinstimmung ist die Wahl der herrschenden Mehrheitsmeinung. Falls keine Fehler auftreten, funktionieren solche Verfahren, sobald jedoch ein Prozeß verschiedene ‚Meinungen' verbreitet, versagen sie leider. Bevor wir bessere Lösungen behandeln, soll das eigentliche Problem formalisiert dargestellt und in ein Modell eingebettet werden:

Bei der einfachen Fragestellung soll die Übereinstimmung zu einem Bit erfolgen. Seien nun n Rechenknoten/Prozesse gegeben, wobei von ihnen eine beschränkte Menge anfänglich oder während der Laufzeit fehlerhaft arbeitet.

Definition 3.1 (Übereinstimmungproblem)
Jeder Prozeß i hat ein Anfangszustandsbit $x_i \in \{0,1\}$. Unser Problem besteht für die fehlerfreien Rechenknoten nun darin, daß sie gemeinsam den Übereinstimmungswert y erreichen wollen, d. h. wir benötigen ein Protokoll, welches jeden zuverlässigen Prozeß j mit dem Bit $y_j \in \{0,1\}$, $\forall j: \; y_j = y$ terminieren läßt.

Natürlich sollten die y_i vom Anfangszustand x_i abhängig sein, da sonst das Problem trivial durch Setzen von $y_i := 0$ gelöst werden könnte. Folgende Eigenschaften sind diesbezüglich also weiterhin zu beachten:

Nicht-Trivialität: Für jedes $y \in \{0,1\}$ gibt es einen Anfangszustand x_i und eine mögliche Ausführung des Protokolls, in der y unser Übereinstimmungswert ist.

Schwache Einstimmigkeit: Falls für alle i die Beziehung $x_i = x \in \{0,1\}$ gilt, folgt sofort $y = x$, wenn während der Ausführung keine Fehler aufgetreten sind.

Starke Einstimmigkeit: Falls $\forall i: \; x_i = x \in \{0,1\}$ gilt, folgt $y = x$ nun unbedingt.

Ein Protokoll heißt *t-resilient (unverwüstlich)*, wenn es selbst bei bis zu t fehlerhaften Prozessen konsistent und richtig arbeitet. Die auftretenden Fehler können dabei unterschiedlicher Gestalt sein. Wir bezeichnen das Szenario als *Crash*, falls sämtliche Operationen bis zu einem bestimmten Zeitpunkt korrekt ablaufen, danach aber keine Interaktion mehr erfolgt. Allgemeiner formuliert sind *Byzantinische Fehler* solche, bei denen keine Annahme über das Verhalten des betroffenen Prozesses zugrunde liegt. Damit sind beispielsweise auch mutwillige Abweichung, inkonsistente Nachrichtenverbreitung oder vorübergehendes ‚Totstellen' erfaßbar, welche Gegenstand bösartiger Reaktionen gegen das Protokoll sein könnten.

Die Praktikabilität der Übereinstimmungsprotokolle hängt hauptsächlich von ihrer Berechnungs- und Nachrichtenkomplexität ab. Wichtige Faktoren können die verstrichene Zeit bis zur Entscheidung, der erzeugte Nachrichtenaufwand oder der benutzte Rechnerspeicher sein. Insbesondere sind diese Größen mit der Anzahl der aufgetretenen Fehler korreliert. Deshalb ist es häufig eine nützliche Vorgehensweise, den Normalfall ohne Fehler effizient zu behandeln und nur bei Ausnahmesituationen in besondere Protokollschritte zu verzweigen. Allerdings sollte man nicht vergessen, daß in sehr großen Systemen i. A. die erwartete Fehlerquantität linear mit der Rechneranzahl wächst.

Zuverlässige Verteilung von Nachrichten kann entweder durch physisch sichere Verteilnetze[55] sofort erreicht oder aber mit Hilfe diverser Übereinstimmungsprotokolle in unzuverlässigen Umgebungen ‚simuliert' werden. Dabei geht man dann von folgendem Grundalgorithmus zum Senden einer Mitteilung aus:

1. Die Nachricht wird den gewünschten Beteiligten zugestellt.
2. Alle Empfänger werden sich über den Inhalt der Nachricht einig. Sie verwenden hierzu ein Übereinstimmungsprotokoll, welches gegen Byzantinische Fehler weitgehend robust ist.

Unser nächster Teil behandelt dafür geeignete Verfahren.

3.1. Byzantinische Übereinstimmung

Viele Protokolle beruhen auf dem sogenannten *„Problem der byzantinischen Generäle"*, das L. Lamport, R. Shostak und M. Pease [44] eindrucksvoll am historischen Beispiel erklären:

[55] Nachrichten müssen für alle Teilnehmer (inklusive Absender) unveränderbar sein. Deshalb entfallen Netzwerktechniken, bei denen Mitteilungen aktiv über andere Stationen weitergeleitet werden. Übrig bleibt nur eine geringe Anzahl von Möglichkeiten, die allgemein meist nicht praktikabel sind, z. B. Streuverteilung durch eigene Satelliten bzw. Funknetze. Störungen oder Medienfehler dürfen niemals den Mißbrauch der Verteileigenschaft erlauben.

Verschiedene Divisionen der byzantinischen Streitkräfte belagern eine feindliche Stadt. Jede Einheit wird vom eigenen General befehligt, der lediglich über Nachrichten mit anderen Kommandeuren in Verbindung steht. Um die Stadt erfolgreich einzunehmen, müssen die Truppenteile zusammenarbeiten, d. h. übereinstimmend festlegen, welcher Teil angegriffen werden soll bzw. aus welchem Territorium sie sich zurückziehen.

Die Verteidiger der Stadt waren jedoch nicht untätig. Sie bestachen einige Generäle mit Geld, um ihre Gunst zu erlangen und im Endziel eine geteilte byzantinische Streitmacht vorzufinden. Handeln die Truppen nämlich uneinheitlich, können sie leicht zurückgeschlagen werden. Diese Kunde verbreitete sich auch unter den loyalen Kommandeuren, welche daraufhin nach einem Verfahren mit folgenden Eigenschaften suchten:

1. Alle loyalen Generäle entscheiden sich für den selben Angriffsplan eines vom Kaiser bestimmten Oberbefehlshabers.
2. Eine geringe Anzahl Verräter können den ehrlichen Generälen keinen falschen Plan aufzwingen, falls der Oberbefehlshaber selbst loyal ist.

Da von vornherein unbekannt ist, welche Generäle Verrat begehen werden, ist das Problem nicht trivial lösbar. Eine zusätzliche Schwierigkeit besteht in der mündlichen Übertragung der Nachrichten. Ihr Transport durch Boten ist unzuverlässig, sie sind leicht veränderbar, und es existiert überhaupt keine Garantie, daß ein bestimmter Kommandeur die gleiche Meinung allen gegenüber vertritt.

Alle Verfahren, bei denen die Nachrichtenübertragung ungesichert erfolgt, dürfen nur von weniger als einem Drittel ($t < n/3$) Verrätern dominiert werden. Bei lediglich drei Generälen müssen sich also alle loyal verhalten. Die Autoren [44] zeigten aber, daß mit Authentizitätsmechanismen (unterschriebene, fälschungsichere Schriftstücke) sogar eine Lösung existiert, welche nur mindestens zwei ehrliche Befehlshaber ($t \leq n - 2$) erfordert. Die Generäle können dann nämlich nicht mehr einfach leugnen, welche Nachrichten sie wirklich von anderen empfangen haben. Leider funktioniert alles ausschließlich in einer synchronen Umgebung.

3.1.1. Synchrone Kommunikationsnetze

In Kommunikationsmodellen mit regelmäßiger Taktung, synchronisierten Uhren und akkuraten Zeitschranken für die Nachrichtenübertragung können Ausfälle oder Fehler leicht festgestellt werden. Synchrone Kommunikation läuft in einer Sequenz von Runden ab: In jeder Runde sendet der Prozeß/Teilnehmer zuerst beliebig viele Nachrichten an andere und empfängt danach für ihn bestimmte Mitteilungen. Eingegangene Nachrichten dieser Runde können also nicht das anfängliche Sendeverhalten beeinflussen. Außerdem ist am Ende der Runde eindeutig klar, welcher Prozeß keine Mitteilung gesendet hat. Für die synchrone Übertragung fassen wir hier wesentliche Erkenntnisse aus wissenschaftlichen Arbeiten [43, 44] noch einmal zusammen:

Fakt 3.1 (Zusammenhang zum Übereinstimmungsproblem)
Sei eine t-resiliente Lösung des Übereinstimmungsproblems gegeben, dann existiert auch eine t-resiliente Lösung des Problems der byzantinischen Generäle, die lediglich eine weitere Übertragungsrunde benötigt und zusätzlich $n-1$ Nachrichten (Verteilung des Übereinstimmungswertes) sendet.

Obiger Fakt erlaubt uns eine effiziente Konstruktion durch Verfahren des verwandten Übereinstimmungsproblems. Hinsichtlich der grundsätzlichen Lösbarkeit bei *Byzantinischen Fehlern* bleibt anzumerken:

Fakt 3.2 (Byzantinische Übereinstimmung durch Authentizität)
Es existiert ein t-resilientes synchrones Protokoll, welches das Problem der byzantinischen Generäle mit Hilfe authentischer Nachrichten löst.

Ein ‚effizientes' Protokoll mit Kommunikationskomplexität $O(nt)$ stammt von D. Dolev und H. R. Strong [45]. Ohne Authentizität der Mitteilungen gibt es nur eine Lösung, wenn die Anzahl der fehlerhaften Prozesse nicht zu groß wird.

Fakt 3.3 (Byzantinische Übereinstimmung ohne Authentizität)
Es existiert ein t-resilientes synchrones Protokoll ohne Authentizität, welches das Problem der byzantinischen Generäle löst, falls $t < n/3$ gilt.

3.1.2. Asynchrone Kommunikationsnetze

Unglücklicherweise findet man häufig asynchrone Umgebungen vor, wie es u. a. auch in unserer Aufgabenstellung zur Anonymität der Fall ist. Eine Erkennung böswilligen ‚Totstellens' ist hier fast unmöglich, da keine Annahmen zur Zeitverzögerung bei der Übertragung gemacht werden können. Insbesondere ist also nicht feststellbar, ob der Sender fehlerhaft oder lediglich langsam arbeitet. Diese Schwierigkeit führt zu so heiklen Effekten, daß viele Übereinstimmungsprobleme nicht mehr deterministisch lösbar sind.

M. J. Fischer, N. A. Lynch und M. S. Paterson [46] zeigten bereits im Jahre 1983 auf einer Konferenz für Datenbanken folgende schwächere Tatsache:

Fakt 3.4 (Unlösbarkeit der Übereinstimmung)
In einer voll asynchronen Umgebung existiert keine 1-resiliente Lösung des Übereinstimmungsproblems, falls gerade die Eigenschaft der Nicht-Trivialiät von ihr gefordert ist.

In ihrer Orginalarbeit wird als Störungsquelle zwar nur der vollständige Prozeßcrash in Betracht gezogen, da Byzantinische Fehler aber allgemeiner sind, gilt das Resultat selbstverständlich auch für das abgeleitete Problem der byzantinischen Generäle. Es gibt also kein t-resilientes Verfahren (mit $t > 0$) für unsere Fragestellung nach Byzantinischer Übereinstimmung, welches deterministisch arbeitet

und damit immer zu einer Entscheidung kommt. Das hat natürlich erheblichen Einfluß auf die Robustheit höherer Protokollschichten.

Probabilistische Algorithmen sind also scheinbar der einzige Ausweg, weil sie zumindest mit hoher Wahrscheinlichkeit Übereinstimmung erreichen können.

3.2. Probabilistische Übereinstimmungsprotokolle

Lösungsverfahren mit Einfluß von Zufall werden überall dort eingesetzt, wo keine exakt deterministischen Protokolle existieren. Beispielsweise liefern sie einen deadlock-freien Algorithmus zum *Problem der abendessenden Philosophen*[56] und helfen auch bei unserer Fragestellung nach Byzantinischer Übereinstimmung weiter. M. Ben-Or [47] beschrieb dazu ein Protokoll, das für $t < n/5$ mit Wahrscheinlichkeit nahe 1 funktioniert. T. Rabin, R. Canetti und andere Wissenschaftler verbesserten später die Bedingung an t auf das theoretische Maximum von $t < n/3$, allerdings sind ihre Verfahren aufgrund enormer Nachrichtenkomplexität höchst unpraktikabel. Erst im Jahre 2000 wurde von C. Cachin, K. Kursawe und V. Shoup [37] ein effizientes Protokoll für *„Asynchronous Binary Byzantine Agreement“ (ABBA)* beschrieben, welches jetzt vorgestellt werden soll.

3.2.1. ABBA Protokoll

Dieses Protokoll löst das Problem der binären Byzantinischen Übereinstimmung[57] in unserem asynchronen Umfeld und hat folgende praktische Eigenschaften:

- Es toleriert die maximale Zahl von $t < n/3$ byzantinischen Angreifern.
- Die erwartete Rundenkomplexität ist nahezu konstant.[58]
- Es besitzt eine fast optimale Nachrichtenkomplexität von $O(n^2)$ pro Runde.
- Die Länge der einzelnen Protokollnachrichten ist nicht viel größer als ein oder zwei RSA-Standardsignaturen.
- Ein vertrauensvoller Vermittler wird nur zur Initalisierung benötigt. Danach kann eine unbegrenzte Anzahl an Transaktionen[59] abgewickelt werden.

[56] Zugriffsproblem aus der Parallelverarbeitung: Da die Ressourcen (jeweils zwei Gabeln) nur exklusiv verwendet werden können, sind sie vor Benutzung zu sperren und danach wieder freizugeben. Durch unkoordiniertes Verhalten enstehen hierbei jedoch Deadlock-Situationen.

[57] d. h. Teilnehmer einigen sich auf einen Übereinstimmungswert aus der Menge $\{0, 1\}$

[58] Da es sich um ein probabilistisches Protokoll handelt, bleibt natürlich die absolute Rundenkomplexität im kaum wahrscheinlichen Extremfall unendlich groß. (vgl. Fakt 3.4)

[59] hier: Instanzen für Übereinstimmungsfindung zu jeweils einem Binärwert

Ausgangslage: Wir haben n Teilnehmer, bezeichnet mit $P_1, \ldots, P_n$, welche über ein unsicheres, asynchrones Netzwerk kommunizieren. Ein möglicher Angreifer (adversary) kann bis zu t von ihnen kontrollieren. Sein Angriffsverhalten ist *statisch*,[60] d. h. er muß vor Ausführung des Protokolls entscheiden, gegenüber welchen Teilnehmern er bösartig agiert. Damit sei f die aktuell beeinflußte Anzahl dieser unehrlichen Parteien (Kollaborateure, Opfer), wobei $0 \leq f \leq t$ gilt.

Während der Initalisierungsphase erzeugt ein vertrauensvoller Vermittler diverse Schlüsselsysteme und Anfangszustände für jeden Teilnehmer. Unser Angreifer kann zwar die geheimen Schlüssel der Opfer erlangen, alle Informationen der ehrlichen Parteien bleiben ihm jedoch verborgen.

Das zugrunde liegende Netzwerk ist unsicher, denn der Angreifer hat darüber die volle Kontrolle. Er kann die Auslieferung von Nachrichten nach seinen Wünschen beeinflussen, Inhalte verändern oder sogar neue Mitteilungen einfügen. In diesem formalen Modell kann man deshalb Netzwerk und Angreifer zusammenfassen. Die ehrlichen Teilnehmer sind komplett passiv, d. h. sie reagieren nur auf Anfragen, die der Angreifer passieren läßt. Genauer gesagt, findet nach der Initalisierungsphase eine Reihe von Grundschritten statt, die jeweils wie folgt ablaufen: Der Angreifer liefert eine Nachricht an einen ehrlichen Teilnehmer P_i aus, der dann seinen internen Zustand aktualisiert und ggf. eine Menge von Antwortnachrichten generiert. Jene Mitteilungen werden dem Netzwerk zur möglichen Auslieferung an andere Parteien übergeben. Der Angreifer wird also letztendlich entscheiden, ob er die Nachrichten sofort, irgendwann später oder überhaupt nicht weiterleitet.

Natürlich sollten alle Protokollteile nur (ggf. probabilistische) Polynomialzeitberechnungen erfordern, damit sie überhaupt praktikabel sind. Zusätzlich müssen wir deshalb einen Sicherheitsparameter für die kryptographischen Bestandteile einführen und von allen Protokollschritten verlangen, daß sie in diesem Parameter (z. B. Länge des Primfaktorprodukts) ebenfalls polynomial beschränkt sind.

Modell der Byzantinische Übereinstimmung: Wir wollen jetzt Operationen und Eigenschaften des Übereinstimmumgsprotokolls formalisieren. Jeder Instanz für eine zwischen den Teilnehmern durchzuführende binäre Einigung wird das eindeutige Kennzeichen TID zugeordnet, welches aus einer beliebig langen Zeichenkette besteht. Die konkrete Form wird von der zugrundeliegenden Applikation und ihrer Implementierung bestimmt.

[60] Alternativ könnte man auch ein adaptives Modell wählen, wo der Angreifer während der Laufzeit des Protokolls seine Opfer dynamisch festlegt. Dieses Szenario würde allerdings kryptographische Werkzeuge benötigen, die uns hier nicht zur Verfügung stehen. Außerdem ist der statische Ansatz in der Praxis gar nicht so unrealistisch, weil die Auswahl des Opfers meist unabhängig von den übers Netzwerk transportierten Protokolldaten erfolgt (z. B. eher durch unvorsichtige Systemadministratoren oder bestochene Mitarbeiter).

Der Angreifer könnte an P_i eine Nachricht der Form

$$(TID, \texttt{activate}, \alpha)$$

weiterleiten, in welcher $\alpha \in \{0, 1\}$ der Startwert zur Übereinstimmung ist. Falls eine solche Mitteilung wirklich ausgeliefert wird, bezeichnen wir P_i als TID-aktiviert, wobei sein Anfangszustand dem gegeben Wert α entspricht. Danach können bei P_i verschiedene Protokollnachrichten der Gestalt

$$(TID, j, i, \ldots)$$

eingehen, die neben dem Index des Senders $1 \leq j \leq n$ noch weitere Nutzdaten enthalten. Daraufhin aktualisiert P_i seinen internen Zustand und generiert ggf. eine Menge von Antwortmitteilungen der Form

$$(TID, i, j, \ldots) \ ,$$

wobei jetzt $1 \leq j \leq n$ der Empfängerindex ist. Schließlich wird er irgendwann auch die Nachricht

$$(TID, \texttt{decide}, \beta)$$

senden, wobei $\beta \in \{0, 1\}$ seine endgültige Entscheidung zum Übereinstimmungsproblem darstellt. Danach bezeichnen wir P_i als TID-entschieden; er darf seinen Zustand bei dieser Transaktion nicht mehr ändern.

Zur Vereinfachung nehmen wir an, daß alle Nachrichten authentisch verteilt werden. Digitale Signatur sowie Public-Key-Infrastruktur des vertrauensvollen Vermittlers ermöglichen uns die effiziente Umsetzung dieser Annahme. Der Angreifer wird damit insoweit eingeschränkt, daß, wenn P_i, P_j ehrlich sind und er eine Nachricht der Form $(TID, i, j, \ldots)$ an P_j ausliefert, diese Mitteilung auch wirklich von P_i zu einem früheren Zeitpunkt erzeugt wurde.

Folgende drei formale Eigenschaften muß das ABBA-Protokoll erfüllen:

Übereinstimmung: Alle ehrlichen, TID-entschiedenen Teilnehmer befinden sich nach Abschluß dieser Transaktion im gleichen Übereinstimmungszustand der Menge $\{0, 1\}$. Für den Angreifer ist es berechnungstechnisch unmöglich, zwei ehrliche Teilnehmer unterschiedlich entscheiden zu lassen.

Termination: Traditionell wird davon ausgegangen, daß alle Nachrichten zwischen ehrlichen Teilnehmern „irgendwann“ ausgeliefert werden. Damit entscheiden sich diese Parteien aber auch nur „irgendwann“ für den gemeinsamen Übereinstimmungswert. (Ein Angreifer kann die Verteilung also nicht dauerhaft unterbinden.)

Cachin, Kursawe und Shoup führen die neue Definition ein, daß, falls der Angreifer die Mitteilungen des Protokolls bis zu einem gewissen Grad sofort ausliefert, sich auch alle ehrlichen Teilnehmer ‚schnell‘ entscheiden. Ihr Verständnis von Termination setzt sich im wesentlichen aus zwei Bedingungen zusammen.

Deadlock-Freiheit: Der Angreifer kann unmöglich Situationen erzeugen, bei denen ein ehrlicher Teilnehmer sich nicht entscheiden wird, obwohl alle ehrlichen Parteien TID-aktiviert sind und ihre zugehörigen Nachrichten vom Netzwerk auch ausgeliefert wurden.

Schnelle Konvergenz: Sei TID_s für $s = 1, 2, \ldots$ das s-te Transaktionskennzeichen und X_s die Anzahl der dabei generierten Protokollnachrichten aller ehrlichen Teilnehmer. Es müssen zwei feste Polynome B, C in n und dem Sicherheitsparameter existieren, so daß für alle $s \geq 1, m \geq 1$

$$\Pr\left[X_s \geq mB + C\right] \leq 2^{-m} + \epsilon$$

gilt, wobei ϵ eine vernachlässigbare Funktion[61] im Sicherheitsparameter darstellt. Während letztere von der Berechnungsstärke des Angreifers abhängig ist, sind B, C lediglich durch das Übereinstimmungsprotokoll (hier ABBA) dominiert.

Die Deadlock-Freiheit verhindert triviale Protokolle, die niemals Transaktionen entscheiden oder Nachrichten generieren. Die schnelle Konvergenz sichert zeitliche Termination, weil der Angreifer B, C nicht beeinflussen kann. Damit entfallen Protokolle, welche lediglich Nachrichten hin-und-her senden, ohne dabei jemals zu einer Entscheidung zu gelangen.

Mit anderen Worten impliziert diese Definition, daß sich alle ehrlichen Parteien schnell für einen Übereinstimmungswert entscheiden, wenn das Netzwerk eine polynomial beschränkte Anzahl von Protokollnachrichten zu einer bestimmten Transaktion ausliefert. Die Forderung garantiert allerdings nicht, daß der Angreifer die Mitteilungen wirklich passieren läßt, da er in unserem Modell ja weitreichende Kontrolle über das zuständige Verteilnetz hat.

Richtigkeit: Falls alle Teilnehmer mit gleichem Anfangszustand TID-aktiviert sind und der Angreifer gerade keinen von ihnen kontrolliert ($f = 0$), dann entscheidet sich später jede Partei für diesen Startwert.

Obige Definition der Richtigkeit ist relativ schwach. Wir werden später eine Verbesserung kennenlernen, welche ehrliche Teilnehmer selbst bei $f > 0$ übereinstimmend richtig entscheiden läßt, wenn zumindest alle ehrlichen Parteien mit dem gleichen Anfangszustand diese Transaktion beginnen.

Kryptographie: Das Protokoll benutzt ungewöhnlich viele kryptographische Primitiven, die wir im Grundlagenabschnitt bereits ausgiebig besprochen haben:

- Digitale Signaturen, um die Authentizität der transportierten Protokollnachrichten (sichere Kanäle) zu gewährleisten.
- Ein $/n, n - t, t/$- sowie ein $/n, t + 1, t/$-Schwellenwertsignaturschema, um schnelle Byzantinische Übereinstimmung zu erreichen.

[61] z. B. sinkt ihr Funktionswert stärker als jedes Polynom in diesem Sicherheitsparameter

- Ein $/n, n-t, t/$-Schwellenwertmünzwurfschema, um die gemeinsame Zufallsquelle für den probabilistischen Teil zu realisieren. Sei $F(C) \in \{0,1\}$ dabei der Wert unserer Münze mit dem Namen C.

Protokoll: Wir wollen jetzt die Grundschritte von ABBA beschreiben. Seien ein Transaktionskennzeichen TID und für jeden Teilnehmer P_i der Anfangszustand $V_i \in \{0,1\}$ gegeben. Das Protokoll läuft in Runden $r = 1, 2, \ldots$ ab, wobei sich jede wiederum aus vier einzelnen Teilstücken zusammensetzt:

1. Der Wert der Münze mit Namen (TID, r) wird bestimmt und findet ggf. in den folgenden Schritten seine Anwendung. Er dient ehrlichen Parteien als Preferenz, falls in r unterschiedliche ‚Teilnehmermeinungen' vorliegen.
2. Jede Partei verteilt eine Vorauswahl (pre-vote) für den Übereinstimmungswert $b \in \{0,1\}$. Diese Wahl wird durch eine zugehörige Schwellenwertgesamtsignatur bestätigt, während die Protokollnachricht selbst durch Teilunterschrift von P_i gesichert ist.
3. Nach dem Erhalt von $n-t$ gültigen Vorauswahlmitteilungen sendet jede Partei eine Hauptauswahl (main-vote) $v \in \{0, 1, \texttt{abstain}\}$. Wie bereits bei der Vorauswahl geschehen, muß sie durch eine zugehörige Schwellenwertgesamtsignatur bestätigt werden. Auch die dafür verwendeten Protokollnachrichten sind wieder durch Teilunterschriften gesichert.
4. Nach dem Erhalt von $n-t$ gültigen Hauptauswahlmitteilungen (main-votes) werden jene untersucht. Sind alle zum gleichen Wert $b \in \{0,1\}$ ausgestellt, entscheidet sich die Partei für $U_i = b$ als Übereinstimmungswert dieser Transaktion TID und nimmt noch eine weitere Runde am Protokoll teil. Andernfalls fährt sie normal mit der nächsten Runde und Schritt 1 fort.

Eingabe: Anfangszustand $V_i \in \{0,1\}$, **Ausgabe:** Übereinstimmungswert $U_i \in \{0,1\}$

Zum Verständnis sind noch Details notwendig. Rufen wir uns in Erinnerung, daß eine Nachricht von P_i an P_j die Form $(TID, i, j, \ldots)$ hatte, wobei im Teil „…" Nutzlast transportiert werden konnte. Deshalb wird jetzt nur noch dieser Nutzteil beschrieben, denn TID, i sowie j ergeben sich aus dem Zusammenhang:

Schritt 1: Jede Partei P_i generiert mit dem $/n, n-t, t/$-Schwellenwertmünzwurfschema ihren i-Münzteil SC_i für die Münze mit Namen (TID, r). Danach sendet P_i an alle anderen im Nutzteil der Protokollnachricht:

$$(\texttt{coin}, r, SC_i)$$

Alle Teilnehmer sammeln hierauf $n-t$ verschiedene Münzteile für (TID, r) und können diese zur Gesamtmünze kombinieren, deren Wert $F(TID, r) \in \{0,1\}$ ihnen damit zugänglich ist. Der Schritt liefert uns also ein globales Zufallsbit in jeder Runde, welches wir unter Umständen verwenden müssen.

Schritt 2: Jeder Teilnehmer P_i bestimmt eine Vorauswahl $b \in \{0, 1\}$, wobei ihr Wert in der ersten Runde gleich V_i gesetzt oder sonst im Schritt 4 der vorherigen Runde $r - 1$ berechnet wird. Die Vorauswahl muß eine Bestätigung beinhalten, welche bei $r = 1$ ggf. überflüssig[62] ist. Falls jedoch $r > 1$ gilt, bekommen wir sie auf zwei unterschiedlichen Wegen:

- Entweder durch eine Schwellenwertgesamtsignatur über der Nachricht

$$(\texttt{pre-vote}, TID, r - 1, b)\,,$$

  was wir *harte Vorauswahl* zu b nennen,

- oder aber durch eine Schwellenwertgesamtsignatur über der Nachricht

$$(\texttt{main-vote}, TID, r - 1, \texttt{abstain})\,,$$

  was wir *weiche Vorauswahl* zu $b = F(TID, r)$ nennen.

Intuitiv ist eine harte Vorauswahl die Preferenz für den Wert b auf Grundlage der vorherigen Runde $r - 1$, während die weiche Vorauswahl den aktuellen Wert der Münze bevorzugt, da es offensichtlich widersprüchliche Meinungen einzelner Teilnehmer in der letzten Runde gab. Alle zur Bestätigung notwendigen Gesamtsignaturen können durch bisher gesammelte Teilunterschriften mit dem Kombinationsalgorithmus des Schwellenwertsignaturschemas berechnet werden.

Nach Erzeugung der Bestätigung JP_i generiert P_i eine i-Teilsignatur SP_i über der Nachricht

$$(\texttt{pre-vote}, TID, r, b)$$

und sendet schließlich folgende Mitteilung an alle anderen:

$$(\texttt{pre-vote}, r, b, JP_i, SP_i)$$

Schritt 3: Jede Partei P_i wartet auf $n - t$ ordnungsgemäß bestätigte und vom Sender teilsignierte Vorauswahlmitteilungen der aktuellen Runde r. Dann wird die Hauptauswahl $v \in \{0, 1, \texttt{abstain}\}$ nach folgendem Schema getroffen:

- Falls unter den $n - t$ Nachrichten mindestens eine Preferenz für 0 und eine andere für 1 existiert, ist die Hauptauswahl von P_i in dieser Runde $v := \texttt{abstain}$. Die zugehörige Bestätigung setzt sich aus den JP_x, JP_y der widersprüchlichen Vorauswahlmitteilungen von P_x und P_y zusammen. Da unsere Vorauswahl in Runde $r = 1$ ggf. unbestätigt ist, kann anfangs auch die Hauptauswahl nicht bestätigt werden.

- Ansonsten sind $n-t$ Vorauswahlnachrichten für den gleichen Wert $b \in \{0, 1\}$ eingegangen, welche allesamt auch gültige Teilsignaturen über

$$(\texttt{pre-vote}, TID, r, b)$$

  beinhalten. Damit ist P_i in der Lage, eine Gesamtsignatur zu erzeugen, die ihm als Bestätigung seiner Hauptauswahl $v := b$ dient.

[62] Vgl. kommender Abschnitt „Verbesserungen“, wo sie auch für $r = 1$ notwendig wird.

Nach Erzeugung der Bestätigung JM_i generiert P_i eine i-Teilsignatur SM_i über der Nachricht

$$(\texttt{main-vote}, TID, r, v)$$

und sendet schließlich folgende Mitteilung an alle anderen:

$$(\texttt{main-vote}, r, v, JM_i, SM_i)$$

Schritt 4: Jeder Teilnehmer P_i wartet auf $n - t$ ordnungsgemäß bestätigte und vom Sender teilsignierte Hauptauswahlmitteilungen der aktuellen Runde r. Sind alle Nachrichten für einen Wert $b \in \{0, 1\}$ ausgestellt, bekennt sich die Partei hierzu und ist damit TID-entschieden. Sie nimmt aber noch eine weitere Runde am Protokoll teil, um auch anderen ‚Mitspielern' die Entscheidung zu ermöglichen. Ansonsten fährt P_i normal mit der nächsten Runde und Schritt 1 fort. In beiden Fällen wird der Wert der Vorauswahl für $r + 1$ wie folgt bestimmt:

- Wenn P_i nur Nachrichten für `abstain` gesammelt hat, kann er eine Gesamtsignatur über der Mitteilung

  $$(\texttt{main-vote}, TID, r, \texttt{abstain})$$

  erzeugen und damit eine weiche Vorauswahl in der nächsten Runde treffen.

- Anderenfalls wurde zumindest eine der $n - t$ Hauptauswahlnachrichten für $b \in \{0, 1\}$ mit einer Gesamtsignatur über

  $$(\texttt{pre-vote}, TID, r, b)$$

  bestätigt, so daß P_i in der nächsten Runde eine harte Vorauswahl für gerade dieses b treffen kann.

Sicherheitsanalyse: Jetzt sollen wichtige Punkte des Sicherheitsbeweises skizziert werden. Der vollständige Beweisweg ist in der Orginalarbeit [37] zu finden.

Satz 3.1
Angenommen wir haben ein sicheres $/n, n - t, t/$-Schwellenwertsignaturschema, ein sicheres $/n, n - t, t/$-Schwellenwertmünzwurfschema und sichere digitale Signaturen zur Authentifikation der Nachrichten, dann löst das ABBA-Protokoll die asynchrone Byzantinische Übereinstimmung für $t < n/3$.

Beweisidee: Zu zeigen sind die besprochenen drei Eigenschaften: Übereinstimmung, Termination und Richtigkeit. Letztere ist in der schwachen Form mit $f = 0$ offensichtlich, da alle Teilnehmer den gleichen Anfangszustand V_i wählen. Für den restlichen Teil (Übereinstimmung, Termination) seien vorerst $f = t$ und das Transaktionskennzeichen TID fixiert.

Wir betrachten zunächst die Vorauswahl ehrlicher Teilnehmer für Runde $r \geq 1$. Da $n > 3t$ gilt, liegen maximal zu einem Wert $b \in \{0, 1\}$ wenigstens $n - 2t$

Preferenzen vor. Falls ein solches b existiert, setzen wir $\rho_r := b$, ansonsten bleibt ρ_r undefiniert. Dieses Symbol hat also genau ab dem Zeitpunkt einen konkreten Wert, wenn hierfür genügend gültige Vorauswahlnachrichten in r verteilt wurden.

Lemma 1 *Für $r \geq 1$ gelten (mit hoher Wahrscheinlichkeit) folgende Aussagen:*

(a) Falls eine ehrliche Partei die Hauptauswahl für $b \in \{0,1\}$ in Runde r verteilt oder akzeptiert, dann ist ρ_r definiert, und es gilt $\rho_r = b$.

(b) Falls eine ehrliche Partei die harte Vorauswahl für $b \in \{0,1\}$ in Runde $r+1$ verteilt oder akzeptiert, dann ist ρ_r definiert, und es gilt $\rho_r = b$.

(c) Falls eine ehrliche Partei die Hauptauswahl für `abstain` *in Runde $r+1$ verteilt oder akzeptiert, dann ist ρ_r definiert und $\rho_r = 1 - F(TID, r+1)$.*

(d) Falls r die erste Runde angibt, in welcher eine beliebige ehrliche Partei TID-entschieden ist, dann gilt für alle anderen ehrlichen Teilnehmer, welche sich irgendwann noch entscheiden werden, daß sie den gleichen Zustand und Übereinstimmungswert entweder in Runde r oder $r+1$ erreichen.

Um (a) zu zeigen, nehmen wir an, daß eine ehrliche Partei die Hauptauswahl für $b \in \{0,1\}$ in Runde r akzeptiert hat. Für die Bestätigung muß eine gültige Schwellwertgesamtsignatur über

$$(\texttt{pre-vote}, TID, r, b)$$

im Nutzteil enthalten sein. Die Eigenschaft der Unfälschbarkeit des Signaturschemas (siehe Seite 46) impliziert jedoch, daß wenigstens $(n-t)-t = n-2t$ ehrliche Teilnehmer eine Vorauswahl für b getroffen haben. Deshalb ist auch ρ_r definiert und gleicht genau diesem Wert. Aussage (b) folgt jetzt einfach aus (a) und der Tatsache, daß eine harte Vorauswahl für $b \in \{0,1\}$ in Runde $r+1$ mit obiger Gesamtsignatur der Hauptauswahl von r bestätigt werden muß.

Teil (c): Eine Hauptauswahl für `abstain` in Runde $r+1$ wird durch eine Vorauswahl für 0 und eine für 1 der Runde $r+1$ bestätigt. Diese Preferenzen können beide unmöglich weiche Vorauswahlen sein, mindestens eine von ihnen muß hart sein. Durch (b) wird sofort impliziert, daß jene harte Vorauswahl für ρ_r ausgestellt ist und damit die weiche für den Wert $1 - \rho_r$ plädiert. Weil außerdem $F(TID, r+1) = 1 - \rho_r$ gilt, folgt die Richtigkeit der Aussage (c).

Teil (d): Angenommen eine Partei P_i entscheidet $b \in \{0,1\}$ in Runde r, und kein anderer Teilnehmer ist vorher TID-entschieden, dann hat sie in dieser Runde $n-t$ Hauptauswahlmitteilungen für b akzeptiert. Durch (a) muß $b = \rho_r$ gelten, so daß jede andere ehrliche Partei, welche sich auch in dieser Runde entscheidet, auf jeden Fall den Wert ρ_r wählt. Von den $n-t$ durch P_i akzeptierten Nachrichten kommen mindestens $n-2t$ von ehrlichen Teilnehmern, die für b gestimmt haben.

Weil nun aber $n > 3t$ gilt, sind weniger als $(n-t)-t = n-2t$ Teilsignaturen über der Nachricht

$$(\texttt{main-vote}, r, \texttt{abstain})$$

von ehrlichen Teilnehmern erzeugt. Diese Tatsache impliziert, daß eine weiche Vorauswahl in Runde $r+1$ nicht bestätigt werden kann und damit die einzig gültigen Preferenzen harte Vorauswahlnachrichten für diese Runde sind. Mit Aussage (b) ist ihr Wert auf b festgelegt, somit wird sich auch die Hauptauswahl aller ehrlichen Teilnehmer nur für dieses b in Runde $r+1$ entscheiden können.

Die Übereinstimmungseigenschaft des ABBA-Protokolls folgt hier unmittelbar aus Teil (d). Es bleibt noch die Termination zu zeigen, welche Deadlock-Freiheit und schnelle Konvergenz beinhaltet:

Deadlock-Freiheit ist offensichtlich, denn alle ehrlichen Teilnehmer führen eine Runde nach der anderen aus, falls nur das Netzwerk genügend Protokollnachrichten zwischen ihnen verteilt. Zusammen mit (d) und der Bedingung, daß TID-entschiedene, ehrliche Parteien noch eine weitere Runde am Protokoll teilnehmen, folgt also dieser Teil der Terminationseigenschaft.

Schnelle Konvergenz ist schwieriger zu beweisen. Lemma 1 sagt aus, daß in gegebener Runde $r+1$ mit $r \geq 1$ die Menge der $n-t$ akzeptierten Hauptauswahlnachrichten einer ehrlichen Partei nur Preferenzen entweder für 0 oder 1 enthalten, keinesfalls aber für beide Werte. Also wird sich der Teilnehmer in dieser Runde entscheiden, es sei denn, wenigstens eine Mitteilung war auf `abstain` ausgestellt. Sollte er jedoch eine solche ‚Uneinigkeitsnachricht' akzeptieren, gilt $\rho_r = 1 - F(TID, r+1)$ durch Aussage (c) des Lemmas. Als Schlüsselpunkt für schnelle Konvergenz müssen wir demnach zeigen, daß dieser Wert ρ_r feststeht, bevor die Münze mit Namen $(TID, r+1)$ geworfen wird.

Mit dem Ausdruck *„ρ_r wird zu einem bestimmten Zeitpunkt ermittelt"* ist folgendes gemeint: Es existiert ein effizientes Verfahren W, welches alle bis zu diesem Zeitpunkt vom Angreifer (Netzwerk) durchgeführten Interaktionen in Verbindung mit den Parametern TID, $r \geq 1$ als Eingabe annimmt und daraufhin $w \in \{0, 1, ?\}$ ausgibt. Weiterhin gilt, daß im Falle $w \neq ?$ der Wert $\rho_r = w$ damit eindeutig festgelegt wird.[63] Die letzte Bedingung ist natürlich nur dann sinnvoll, wenn ρ_r in dieser Runde überhaupt definiert war.

Im Ausdruck *„die Münze mit Namen $(TID, r+1)$ wird zu einem bestimmten Zeitpunkt geworfen"* ist exakt der Zeitpunkt gemeint, an dem eine ehrliche Partei das $(n-2t)$-te Münzteil dafür erzeugt.

Lemma 2 *Es existiert eine Funktion W, die ρ_r, wie oben beschrieben, für alle $r \geq 1$ genau so festlegt, daß entweder ρ_r zum Zeitpunkt vor dem Wurf der Münze*

[63] Der Angreifer kann diese Festlegung nicht dauerhaft verhindern.

mit Namen $(TID, r+1)$ *ermittelt wird oder aber* ρ_{r+1} *zum Zeitpunkt vor dem Wurf der Münze mit Namen* $(TID, r+2)$ *ermittelt wird.*

Auf den Beweis des Lemmas soll hier verzichtet werden. Mit obiger Aussage und Unvorhersagbarkeitseigenschaft des Münzwurfschemas (siehe Seite 54) können wir implizieren, daß für ehrliche Teilnehmer die Wahrscheinlichkeit, mehr als $2r+1$ Runden zu absolvieren, durch $2^{-r}+\epsilon$ beschränkt wird.[64] Hiermit folgt auch die schnelle Konvergenz des ABBA-Protokolls sofort.

Ein Problem wird durch den Gesamtbeweis des Satzes 3.1 aber deutlich: Wenn sich die erste ehrliche Partei in Runde r entscheidet, gibt es unter Umständen auch Teilnehmer, die sich erst in $r+1$ entscheiden können. Sie befinden sich dann zwar im Zustand TID-entschieden, müssen – dem Protokoll folgend – aber noch in Runde $r+2$ mitarbeiten. Da die anderen Instanzen bereits nach $r+1$ beendet wurden, sind spätestens im Schritt 2 nicht mehr $n-t$ Nachrichten auslieferbar. Diese Situation ist weniger kritisch als es den Anschein hat, denn alle ehrlichen Teilnehmer haben ja den Übereinstimmungszustand zur Transaktion TID bereits erreicht. Lediglich manche Protokollinstanzen würden offenbar funktionslos weiterarbeiten. Mit einer Verbesserung aus dem folgenden Unterabschnitt können wir jedoch auch dieses Problem lösen.

Eingangs wurde die Sicherheitsanalyse für $f=t$ fixiert, was allerdings hier nur unnötig einschränkend wirkt. Für den allgemeineren Beweisweg mit $f<t$ sind noch kleinere Veränderungen anzubringen, die in der Orginalarbeit [37] ausführlich behandelt werden.

Verbesserungen: Jetzt besprechen wir die vorher angedeuteten Verbesserungen des ABBA-Protokolls und einige spezielle Optimierungen.

Starke Richtigkeit: Mit folgender Erweiterung soll auch bei $f>0$ abgesichert werden, daß alle ehrlichen Teilnehmer sich ultimativ für den Wert entscheiden, mit welchem sie übereinstimmend TID-aktiviert wurden. Dazu wird eine sogenannte Vorverarbeitungsrunde am Anfang jeder Transaktion ausgeführt, die ein zusätzliches $/n, t+1, t/$-Schwellenwertsignaturverfahren verwendet.

Jede Partei P_i generiert damit eine i-Teilunterschrift SR_i über

$$(\texttt{pre-process}, TID, V_i)$$

und sendet die Nachricht der Gestalt

$$(\texttt{pre-process}, V_i, SR_i)$$

[64] Die Vorhersagewahrscheinlichkeit pro Münzwurf beträgt ja genau $1/2$ und ϵ ist von vernachlässigbarer Größenordnung.

an alle anderen. Danach wartet P_i auf $n - t$ solcher Mitteilungen, unter denen wenigstens $t+1$ für den gleichen Wert[65] $b \in \{0, 1\}$ stimmen. Diese Teilsignaturen werden kombiniert und ergeben eine Schwellenwertgesamtsignatur JR_i über

$$(\texttt{pre-process}, TID, b)\,.$$

Weiterhin starten alle Parteien unser normales ABBA-Protokoll, allerdings nun mit dem Unterschied, daß die Vorauswahl der ersten Runde $r = 1$ gerade durch JR_i bestätigt werden muß. Analog fordert die Preferenz für `abstain` in der Hauptauswahl eine Bestätigung, welche wir durch zwei widersprüchliche Gesamtunterschriften JR_x, JR_y verschiedener Teilnehmer erreichen. Alle restlichen Runden laufen unverändert ab.

Falls alle ehrlichen Parteien mit b aktiviert wurden, kann eine gültige Vorauswahl in $r = 1$ nur für diesen Wert ausgestellt sein. Mit dem herkömmlichen Beweis von Satz 3.1 folgt also die Korrektheit der starken Richtigkeit.

Starke Termination: Diese Verbesserung beseitigt den unschönen Effekt, daß manche Protokollinstanzen nach ihrer Entscheidung blockieren könnten, weil sie endlos auf nie mehr eintreffende Nachrichten warten. Falls nun ein Teilnehmer P_i in Runde r den Zustand TID-entschieden für $b \in \{0, 1\}$ annimmt, besitzt er genügend Teilsignaturen, um eine gültige Schwellenwertgesamtunterschrift JF_i über

$$(\texttt{main-vote}, TID, r, b)$$

zu erzeugen. Er sendet die Nachricht

$$(\texttt{final-vote}, r, b, JF_i)$$

an alle anderen Parteien und beendet seine Instanz für TID sofort, ohne noch die geforderte, weitere Runde am Protokoll teilzunehmen. Jeder ABBA-Warteschritt für Nachrichten wird ferner so modifiziert, daß, wenn obige Nachricht nebst gültiger Bestätigung JF_j eingeht, die Partei sich sofort für b entscheidet, ihrerseits diese Terminationsmitteilung weiterleitet und ebenfalls die Instanz TID beendet.

Optimierungen: Einige Veränderungen können zwar die Laufzeit verkürzen oder die Nachrichtenkomplexität senken, rufen aber gleichzeitig andere Schwierigkeiten hervor. Ich führe deshalb nur einen unkritischen Punkt auf, welcher auch in der Implementierung verwendet wurde.

- Der Münzwurf ist in Runde $r = 1$ überflüssig und wird deshalb ans Ende (zyklische Permutation der Schrittnummern) verschoben und dann jeweils im Voraus für $r + 1$ durchgeführt.

[65] Dieser Wert ist bei $n > 3t + 1$ nicht notwendigerweise immer eindeutig. Eine Entscheidung, für welches b die Gesamtsignatur schließlich erstellt werden soll, kann beispielsweise durch Wahl der Mehrheitsmeinung erfolgen.

Probabilistische Sicherheitsanalyse: M. Kwiatkowska und G. Norman [49, 50] haben die Eigenschaften des ABBA-Protokolls mit dem *„Probabilistic Symbolic Model Checker“* untersucht. Sie konnten für $n \leq 20$ die Termination und für alle n die Übereinstimmung sowie die starke Richtigkeit mit automatisierten Werkzeugen nachweisen.[66] Zusammen mit dem herkömmlichen Sicherheitsbeweis gibt uns diese Tatsache eine beruhigende Gewissheit, daß das ABBA-Protokoll die von ihm geforderten Eigenschaften auch wirklich erfüllt.

3.2.2. Ausschlußprotokoll für bösartige Teilnehmer

Mit dem ABBA-Protokoll können wir ein asynchrones Verfahren konstruieren, welches identifizierte Störer von der weiteren Teilnahme ausschließt, ohne dabei inkonsistente Datenbestände bei ehrlichen Subjekten in Kauf zu nehmen. Es ist sowohl Voraussetzung für die später beschriebenen Verteilungsstrategie als auch für das im Abschnitt 4 erörterte DC-Netz-Protokoll und garantiert deren Robustheit[67] im Angriffsfall mit $t < n/3$. Hier muß im Gegensatz zur konsistenten Nachrichtenverteilung ein echtes Übereinstimmungsproblem gelöst werden, nämlich ob P_j in der aktuellen Korrespondenzmenge ehrlicher Parteien bleibt oder nicht.

Wir wollen einen Störer entweder *hart (dauerhaft)* oder nur *weich (global definierte Zeitspanne)* ausschließen. Im ersten Fall muß ein sicherer Beweis ς seiner Bösartigkeit vorliegen (z. B. digitale Signatur über falscher Mitteilung), welcher von allen Parteien unmittelbar verifiziert werden kann. Der zweite Fall behandelt fehlende oder verzögerte Nachrichten des Subjekts, die in einer asynchronen Umgebung sonst nicht handhabbar sind. Solche Fehler würden den Fortschritt des Anwendungsprotokolls (DC-Netz) und damit dessen Robustheit gefährden.

Auf dieser Grundlage kann ein Teilnehmer also seinen gewünschten Ausschlußvektor $\vec{\chi}_i = [\chi_1, \ldots, \chi_n]$ mit $\chi_j \in \{0, 1\}$ und den zugehörigen Beweisvektor $\vec{\varsigma}_i = [\varsigma_1, \ldots, \varsigma_n]$ (mit ‚leeren Vektorelementen‘ oder gültigem Beweis) bestimmen.

Ideen aus verschiedenen Arbeiten [52, 53] leiten uns zu folgendem Algorithmus, welcher bei jeder Ausführung ein anderes, eindeutiges Kennzeichen EID trägt.

- ↻ Jede Partei P_i durchläuft für $j = 1, \ldots, n$ die Schritte 1 bis 3:

  1. P_i verteilt zuverlässig die Nachricht

     $$([EID, P_j], \texttt{proof}, \varsigma_j)$$

     an alle Teilnehmer. Dazu wird das Protokoll aus dem nächsten Abschnitt 3.3 eingesetzt. Die Variable ς_j kann entweder leer sein, falls

[66] natürlich mit Simulation von bis zu $t < n/3$ byzantinischen Angreifern
[67] Funktionsfähigkeit bei Störungen (Byzantinische Fehler)

P_i keinen sicheren Hinweis für die Bösartigkeit von P_j hat, oder sie enthält einen allgemein verifizierbaren Störungsbeweis.

2. P_i wartet auf $n-t$ zuverlässig verteilte Nachrichten von verschiedenen Parteien zum laufenden Ausschlußverfahren $[EID, P_j]$. Danach wird jede Mitteilung untersucht, wobei ihr Sender P_ℓ durch Authentifizierung feststeht und der angefügte Beweis mit ς bezeichnet sei:

$\boxed{\varsigma \neq \emptyset}$ P_i versucht, den Beweis ς zu verifizieren. Im positiven Fall wird $\chi_j := 1$ und im negativen Fall $\chi_\ell := 1$ gesetzt. Außerdem muß der Schuldige (P_j oder P_ℓ) als *hart-ausgeschlossen* markiert werden.

$\boxed{\varsigma = \emptyset}$ Es wird sofort zur nächsten Nachricht übergegangen.

Damit sollten alle empfangenen `proof`-Mitteilungen untersucht sein.

3. P_i startet eine Instanz des ABBA-Protokolls mit dem Anfangszustand $V_i := \chi_j$ und dem Transaktionskennzeichen $TID := [EID, P_j]$. Der resultierende Übereinstimmungswert U_i entscheidet über den Ausschluß von P_j, wobei Vektor $\vec{\chi}_i$ durch $\chi_j := U_i$ aktualisiert wird.

Eingabe:
Ausschlußvektor $\vec{\chi}_i = [\chi_1, \ldots, \chi_n]$ mit $\chi_j \in \{0, 1\}$, 1 klassifiziert Ausschlußwunsch
Beweisvektor $\vec{\varsigma}_i = [\varsigma_1, \ldots, \varsigma_n]$, wobei $\varsigma_j = \emptyset$ keinen Beweis darstellt
Ausgabe:
übereinstimmender Ausschlußvektor $\vec{\chi}_i = [\chi_1, \ldots, \chi_n]$ mit $\chi_j \in \{0, 1\}$
einheitliche Markierung *hart-ausgeschlossener* Teilnehmer

Nach Ausführung des Protokolls haben alle ehrlichen Parteien einen konsistenten Ausschlußvektor $\vec{\chi}_i$ und einheitliche Markierungen. Jede Partei P_j kann deshalb in der Anwendungsschicht zeitweise vernachlässigt werden, wenn $\chi_j = 1$ gilt. Bei Vorhandensein einer harten Markierung darf P_j sogar dauerhaft entfernt bleiben. Der Algorithmus muß vernünftigerweise von allen ehrlichen Teilnehmern gestartet werden, was wir durch asynchronen – aber wohldefinierten – Timeout im Anwendungsprotokoll erreichen. Um die Wiederaufnahme vorher blockierter, aber nur *weich-ausgeschlossener* Subjekte zu ermöglichen, sollte das Ausschlußprotokoll zusätzlich ‚ab-und-zu' zur Ausführung kommen, z. B. in regelmäßigen Zeitabständen seit Initialstart der Anwendung. Dieses Vorgehen setzt keine Synchronisationsmechanismen voraus.

Das obige Verfahren läßt alle Teilnehmer über die Zusammensetzung ihrer neuen Korrespondenzmenge abstimmen. Seine Nachrichten- und Kommunikationskomplexität wird durch n-maligen Aufruf des ABBA-Protokolls dominiert und ist deshalb probabilistisch in die Klasse $O(n^3)$ einzuordnen.

3.3. Konsistente Nachrichtenverteilung

Wir beschreiben jetzt noch, wie mit Hilfe eines zusätzlichen Protokolls beliebig lange Nachrichten konsistent verteilt werden können und richten uns dabei nach den Vorschlägen aus [51, 52].

Definition 3.2 (Zuverlässige Verteilung)
Alle Protokolle für zuverlässige Verteilung erfüllen die folgenden Bedingungen:

Richtigkeit: *Falls ein ehrlicher Teilnehmer die Nachricht m sendet, erhalten alle anderen ehrlichen und aktivierten Parteien genau dieses m, vorausgesetzt das Netzwerk liefert auch alle zugehörigen Protokollnachrichten aus.*

Konsistenz: *Falls der ehrlicher Teilnehmer P_i die Nachricht m bekommt, hat jeder andere ehrliche Empfänger P_j ein m' mit $m = m'$ erhalten.*

Totalität: *Falls ein ehrlicher Teilnehmer eine Nachricht m empfängt, erhalten alle anderen ehrlichen und aktivierten Parteien genau dieses m, vorausgesetzt das Netzwerk liefert wirklich alle zugehörigen Protokollnachrichten aus.*

Integrität und Authentizität: *Jeder ehrliche Teilnehmer empfängt in einer Transaktion mit Kennzeichen $TID = (i, s)$ höchstens eine Nachricht m. Falls Partei P_i ehrlich ist, wurde m auch wirklich von ihr mit Sequenznummer $s \in \mathbb{N}$ gesendet.*

Effizienz: *Die Kommunikationskomplexität jeder Transaktion ist polynomial beschränkt.*

Das Blockieren[68] einzelner Parteien durch den Angreifer ist in unserem Szenario nicht verhinderbar, weil er ja vollständige Kontrolle über die Verteilstruktur hat. Um dennoch die Eigenschaften des Gesamtprotokolls zu gewährleisten, formuliert man folgende theoretische Bedingung an das Netzwerk:

Definition 3.3 (Zusammenhangsannahme)
Ehrliche Teilnehmer können vom Angreifer nicht dauerhaft in ihrer Kommunikation untereinander gehindert werden.

In der Praxis ist obige Annahme häufig zutreffend, da die darunterliegenden Verteilnetzwerke nur von besonders starken Angreifern beeinflußbar sind. Außerdem können durch technische Maßnahmen unabhängige Nachrichtenkanäle (Punkt-zu-Punkt Verbindungen) geschaffen werden, welche damit weit weniger Störfläche bieten, als die mittlerweile von der Wirtschaft oft hochpropagierten Client-Server-Modelle. Konsistenz ausgelieferter Mitteilungen bleibt jedoch auch im negativen Fall von dieser vorausgesetzten Robustheit unberührt.

[68] hier im Sinne fehlender Nachrichtenauslieferung zu verstehen

Aufbauend auf dem VCBC-Protokoll [52] ergibt sich folgende Herangehensweise:

Voraussetzung: Die gesamte Kommunikation zwischen unseren Teilnehmern ist durch digitale Signaturen abgesichert. Deshalb kann eine eingehende Nachricht auch problemlos dem Sendersubjekt zugeordnet werden. Einzelne Transaktionen sind wieder durch das Kennzeichen $TID = [i, s]$ eindeutig bestimmt. Es setzt sich aus dem Bezeichner für das Initialsubjekt i und der Sequenznummer $s \in \mathbb{N}$ zusammen, welche anfangs mit einer globalen Konstante (z. B. $s := 1$) vorbelegt ist. Anhand dieser Zahlen stellt die gewohnte $\leq$-Relation eine partielle Ordnung über der lokalen Sendeabfolge ehrlicher Parteien her. Um die Nachrichtenkomplexität zu reduzieren, wird ein $/n, n-t, t/$-Schwellenwertsignaturverfahren benutzt. Man kann einfach das Signaturschema des ABBA-Protokolls verwenden.

Senden: Das (möglicherweise unehrliche) Initialsubjekt P_i möchte die Nachricht m verteilen. Es sendet

$$(TID, \texttt{ini}, m)$$

an alle anderen Teilnehmer, welche bei Empfang damit TID-aktiviert werden, und erhöht seine interne Sequenznummer $s := s + 1$.

Empfangen: Folgende Teilaktionen werden zu eingehenden, authentischen Nachrichten einer bestimmten Gestalt von jeder Partei P_j ausgeführt:

$\boxed{(TID, \texttt{ini}, m')}$ geht vom Teilnehmer P_ℓ zum ersten Mal ein: Es wird überprüft, ob der Bezeichner des Initialsubjekts in TID dem wirklichen Sender P_ℓ entspricht. Falls ja, ist die Partei damit TID-aktiviert und erzeugt eine j-Teilsignatur SH_j über

$$(\texttt{hsh}, TID, h(m'))\,.$$

P_j sendet die Nachricht

$$(TID, \texttt{hsh}, h(m'), SH_j)$$

an P_ℓ zurück. Die Mitteilung m' wird erstmal nur temporär gespeichert $\widetilde{m}_{TID} := m'$, um sie nach Verifizierung der Konsistenz an höhere Protokollschichten auszuliefern.

$\boxed{(TID, \texttt{hsh}, h(m'), SH_\ell)}$ geht vom Teilnehmer P_ℓ zum ersten Mal ein: Ist P_j das Initialsubjekt (also $i = j$), prüft es die Gültigkeit der Teilunterschrift sowie die Übereinstimmung von $h(m)$ mit $h(m')$ und speichert SH_ℓ im postiven Fall zur späteren Verwendung ab.

Sind $n-t$ gültige Teilunterschriften verschiedener Parteien gesammelt, kann P_i eine Gesamtsignatur JH_i über

$$(\texttt{hsh}, TID, h(m))$$

erzeugen und sendet (hoffentlich) die Nachricht

$$(TID, \texttt{fin}, h(m), JH_i)$$

an alle übrigen Teilnehmer.

$\boxed{(TID, \texttt{fin}, h(m'), JH_i)}$ geht vom Teilnehmer P_ℓ ein: Ist die Gesamtunterschrift JH_i gültig, wird unterschiedlich fortgefahren, je nach dem ob P_j bereits TID-aktiviert ist oder nicht:

<u>TID-aktiviert:</u> P_j kontrolliert die Übereinstimmung von $h(\widetilde{m}_{TID})$ und dem empfangenen $h(m')$. Im positiven Fall wird $\widetilde{m}_{TID}$ an höhere Protokollschichten ausgeliefert und nebst JH_i zur Beantwortung von Anfragen weiterer Teilnehmer noch eine Zeit lang in der Datenbank R unter dem Schlüssel TID gespeichert. Liegt *keine* Übereinstimmung vor, wird

$$(TID, \texttt{req})$$

an alle anderen gesendet, um die korrekte Nachricht m' von ehrlichen Teilnehmern zu erlangen. Dieser negative Fall deutet auf eine Bösartigkeit des Senders P_i hin, welcher offensichtlich unterschiedliche `ini`-Nachrichten verteilt hat. Zusammen mit seiner digitalen Signatur unter dem falschen $\widetilde{m}_{TID}$ kann er des Betrugs überführt und später ausgeschlossen werden.

<u>nicht TID-aktiviert:</u> P_j ist damit TID-aktiviert und sendet die Anfrage

$$(TID, \texttt{req})$$

an alle anderen. Die fehlende `ini-Nachricht` kann entweder durch Bösartigkeit von P_i oder Übertragungsfehler bzw. Verzögerungen noch nicht eingetroffen sein. Die verbindliche Zuordnung der Ursache ist hier also nicht möglich. Es besteht allerdings eine sehr hohe Wahrscheinlichkeit, die korrekte Nachricht m' von ehrlichen Teilnehmern zu erlangen.

In beiden Fällen wird außerdem die Nachricht

$$(TID, \texttt{fin}, h(m'), JH_i)$$

an alle anderen Parteien weitergeleitet, um die Totalität zu wahren.

$\boxed{(TID, \texttt{req})}$ geht vom Teilnehmer P_ℓ ein: Partei P_j sucht in R die zum Datenbankschlüssel TID gehörende Nachricht $\widetilde{m}_{TID}$ nebst Bestätigung JH_i heraus und sendet im Erfolgsfall

$$(TID, \texttt{rpl}, \widetilde{m}_{TID}, JH_i)$$

an P_ℓ zurück. Ist momentan kein Datensatz verfügbar, wird die Anfrage nicht beantwortet.

$\boxed{(TID, \texttt{rpl}, m', JH_i)}$ geht vom Teilnehmer P_ℓ ein: P_j überprüft, ob er eine entsprechende Anfrage gestellt hat und ob JH_i eine gültige Gesamtunterschrift für m' ist. Im positiven Fall wird $\widetilde{m}_{TID} := m'$ gesetzt und an höhere Protokollschichten ausgeliefert. Außerdem sollte die Nachricht nebst Bestätigung wieder eine Zeit lang in der Datenbank R unter dem Schlüssel TID gespeichert bleiben, um auch anderen Parteien den Empfang (Eigenschaft der Totalität) zu sichern.

Funktionsweise: Die Idee geht auf M. Reiters Echo-Verteilung [53] zurück, wurde aber durch das Schwellenwertsignaturverfahren [52] in seiner Effizienz erhöht. Der Sender verteilt seine Nachricht und hofft, daß $n-t$ ‚Zeugen' eine Teilunterschrift dafür liefern. So kann er später eine Gesamtsignatur erstellen, welche allen anderen Parteien die konsistente Verteilung beweist. Kleine Erweiterungen garantieren zusätzlich die Totalitätseigenschaft, steigern damit allerdings unsere Nachrichtenkomplexität auf $O(n^2)$.

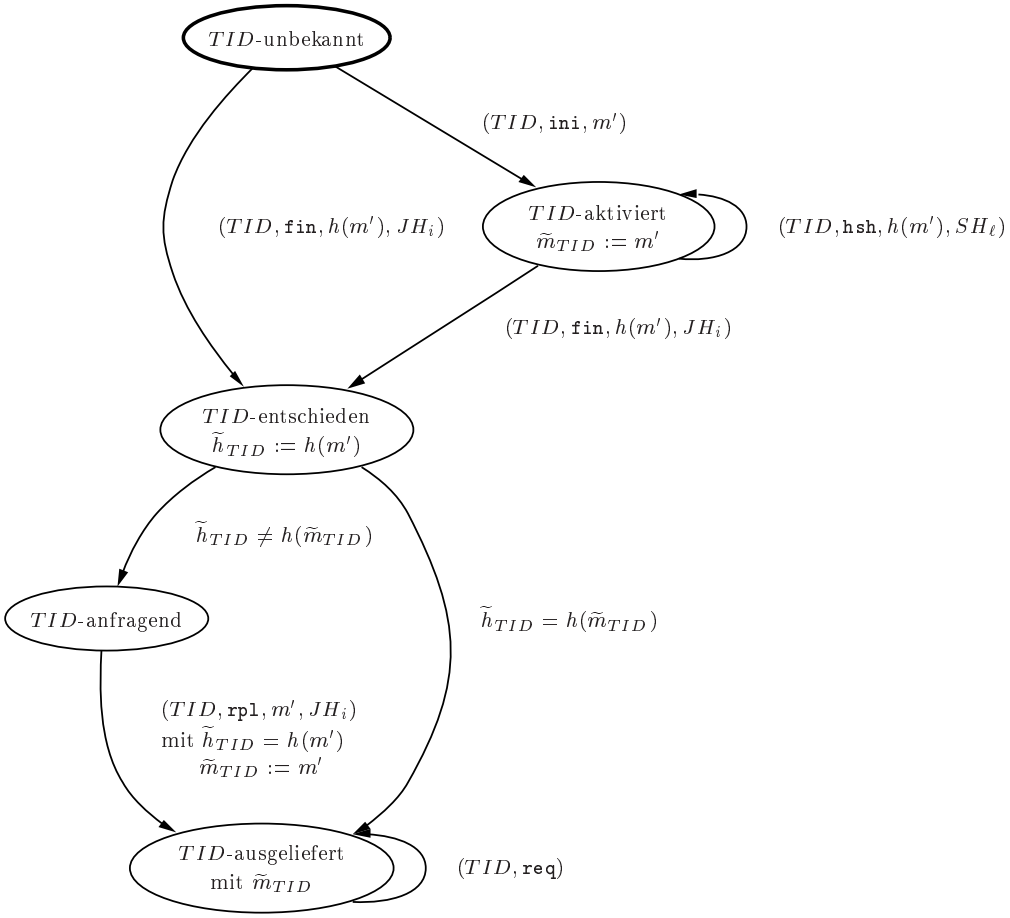

Abbildung 11: Zustandsdiagramm des Verteilungsprotokolls

Analyse: Richtigkeit, Konsistenz und indirekt auch Integrität hängen von der Kollisionsfreiheit der Abbildung h ab. Wir setzen hier wieder eine kryptographische Hashfunktion (siehe Abschnitt 2.3) ein, die für unsere Zwecke hinreichend resistent gegen gezielte Kollisionskonstruktion ist. Unter dieser Annahme garantiert das beschriebene Protokoll zuverlässige Verteilung.

Einige Details zur Implementierung finden sich im Kapitel 5.1.3.

4. Dining Cryptographers Networks

Freude wird jedesmal dein Abendbrot sein,
wenn du den Tag nützlich zugebracht hast.
— Thomas a Kempis

In diesem Kapitel wollen wir das bereits als Grundlage besprochene Konzept der anonymisierenden DC-Netze vertiefen und schließlich ein robustes Protokoll zur praktischen Realisierung angeben. Weiterhin soll der Sicherheitsbeweis von David Chaum nachvollzogen werden. Rufen wir uns dazu nocheinmal den allgemeinen Aufbau der *„Dining Cryptographers Networks“* in Erinnerung [23]:

Gegeben seien n Teilnehmer, die wieder mit P_i für $i = 1, \ldots, n$ bezeichnet sind. Sie stellen Knoten eines Graphen dar. Jedes Paar (P_i, P_j) von ihnen kennt den gemeinsamen, zufällig gewählten Schlüssel $K_{i,j} \in \{0, 1\} = K_{j,i}$ (Wert der geheim geworfenen Münze), welcher durch eine Kante im Graphen repräsentiert wird.

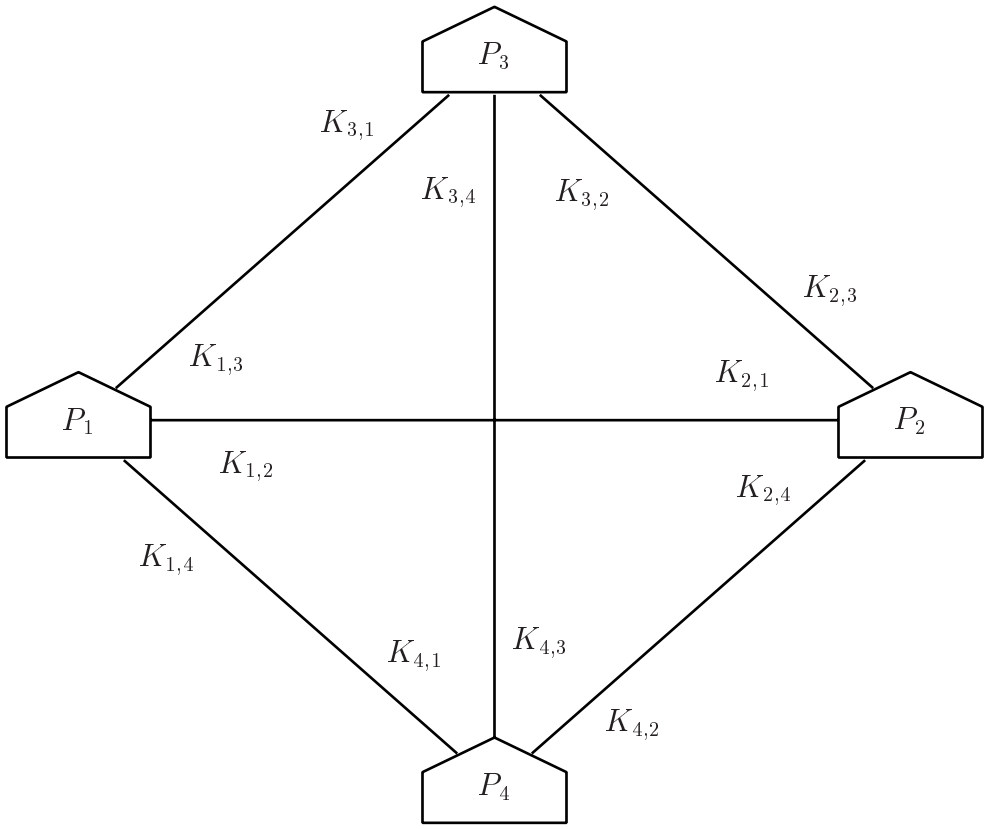

Abbildung 12: vollständiger Schlüsselgraph eines DC-Netzes mit $n = 4$

Die gewünschte Anonymität wird durch *überlagerndes Senden* erreicht, das sozusagen einen unbeobachtbaren, additiven Mehrfachzugriffskanal verwirklicht.

Möchte der Teilnehmer P_i eine Nachricht $b \in \{0,1\}$ senden, setzt er seinen Nachrichtenwert $M_i := b$. Alle übrigen Parteien $P_j, i \neq j$ wählen $M_j := 0$. Danach führen sie folgendes Protokoll gemeinsam durch:

1. Jeder Teilnehmer P_ℓ bildet seinen Ausgabewert

$$O_\ell := M_\ell \oplus \left(\bigoplus_{\substack{k=1 \\ k \neq \ell}}^{n} K_{\ell,k} \right) ,$$

   wobei $\oplus$ die Exklusiv-Oder-Verknüpfung (Addition modulo 2) ist.

2. Die Ausgabe O_ℓ wird an alle Subjekte zuverlässig verteilt.

3. Jeder Teilnehmer rekonstruiert die Nachricht durch

$$M := \bigoplus_{k=1}^{n} O_k \, .$$

Eingabe: Nachrichtenzeichen $M_\ell \in \{0,1\}$, Schlüsselbits $K_{\ell,k}$, **Ausgabe:** $M \in \{0,1\}$

Um längere Nachrichten (Bitsequenzen) zu übertragen, wiederholen wir den Ablauf rundenweise. Dabei müssen alle P_ℓ jeweils neue Schlüssel $K_{\ell,k}$ verwenden, die sie entweder vorher geheim ausgetauscht oder durch einen kryptographisch sicheren Pseudozufallsbitgenerator (siehe Abschnitt 2.8.2) erzeugt haben.

Birgit Pfitzmann und Michael Waidner [32] erwähnen, daß dieses Verfahren nicht nur in GF(2) funktioniert, sondern sich mit jeder beliebigen abelischen Gruppe durchführen läßt. Alle Schlüssel müssen dazu invers $K_{\ell,k} = -K_{k,\ell}$ gewählt und $\oplus$ durch die entsprechende Gruppenoperation ersetzt werden.

Die oben dargestellte Variante in GF(2) zeigt einige Parallelen zur Geheimnisteilung aus Abschnitt 2.5.1. Ist unserer Graph idealerweise vollständig verbunden, wird das Nachrichtenzeichen M_ℓ jedes Teilnehmers durch $n-1$ verschiedene Schlüsselbits unbeobachtbar. Bei der Rekonstruktion von M heben sich gleiche bzw. inverse Elemente wieder auf. Wir wollen jetzt den Beweis nachvollziehen.

Sicherheitsanalyse: Jeder Teilnehmer P_ℓ besitzt eigentlich zwei Geheimnisse:

(a) Schlüsselbits $K_{\ell,k}$, die er mit anderen gemeinsam hat, und

(b) sein Nachrichtenzeichen M_ℓ.

Bösartige Subjekte können obige Daten natürlich untereinander austauschen. Verbleibende (öffentlichen) Informationen über das DC-Netz sind der eigentliche

Schlüsselgraph und alle Ausgaben O_ℓ unserer Teilnehmer. Diese Daten müssen für die Unbeobachtbarkeit nicht geheim bleiben.

Kanten verbinden diejenigen Knoten, deren Teilnehmer jeweils Zugriff auf den gemeinsamen Schlüssel haben. Ohne Beschränkung der Allgemeinheit nehmen wir an, daß der Graph zusammenhängend ist. Ansonsten wird jeder zusammenhängende Teilgraph als einzelnes DC-Netz betrachtet. Eine Anonymitätsmenge besteht nun aus Knoten dieser Teilgraphen. So sind beispielsweise zwei Parteien in der gleichen Anonymitätsmenge, falls ein Pfad zwischen ihnen existiert und der allgegenwärtige Beobachter dessen Schlüsselbits nicht kennt.

Der Angriffsversuch in einem vollständig verbundenen Graphen ist also nur dann erfolgreich, wenn unser Beobachter mit einer Koalition von $n-1$ Partnern zusammenarbeitet. Hier wäre die Unbeobachtbarkeit aber sowieso verloren, da trivialerweise ermittelbar ist, welches P_i seinen Nachrichtenwert $M_i \neq 0$ gewählt hat. Im Fall des unvollständig verbundenen Graphen kann bereits eine Koalition aus zwei Parteien zusätzliche Informationen gewinnen.

Die Situation kann man für eine Runde wie folgt formalisieren: Es werden alle Kanten aus dem Graphen entfernt, von denen der Beobachter zugehörige Schlüsselbits erlangt hat. Aus dem verbliebenen Ergebnis betrachten wir jetzt jeden Teilgraphen C, dessen Kanten eine Anonymitätsmenge formen. Zu zeigen wäre, daß die Angreiferkoalition nicht mehr Informationen erhält als die Parität von M (also die Nachricht selbst). Intuitiv ist diese Tatsache klar, weil alle übrige Nachrichtenzeichen M_ℓ durch zufällig gewählte Schlüsselbits geschützt bleiben.

Der Teilgraph C bestehe aus n Knoten und m Kanten. Seine Inzidenzmatrix $\mathfrak{C}$ der Form $n \times m$ sei wie gewöhnlich definiert, wobei die Knoten die Zeilen und die Kanten die Spalten bezeichnen. Weiterhin sind $\vec{\mathfrak{K}} \in \mathrm{GF}(2)^m$, $\vec{\mathfrak{I}} \in \mathrm{GF}(2)^n$, $\vec{\mathfrak{A}} \in \mathrm{GF}(2)^n$ Wahrscheinlichkeitsgrößen. Dabei ist $\vec{\mathfrak{K}}$ gleichförmig über $\mathrm{GF}(2)^m$ verteilt, von $\vec{\mathfrak{I}}$ stochastisch unabhängig, und es gilt $\vec{\mathfrak{A}} = (\mathfrak{C}\vec{\mathfrak{K}}) \otimes \vec{\mathfrak{I}}$. Vektor $\vec{\mathfrak{K}}$ ist die Repräsentation der Schlüsselkanten $K_{\ell,k}$, $\vec{\mathfrak{I}}$ besteht aus Nachrichtenzeichen M_ℓ von Teilnehmern, und $\vec{\mathfrak{A}}$ ergibt sich durch die Ausgaben O_ℓ aller Knoten. Die Parität von $\vec{\mathfrak{A}}$ ist immer gleich der Parität von $\vec{\mathfrak{I}}$, solange jede Spalte von $\mathfrak{C}$ eine Nullparität hat. Wir erwarten nun, daß $\vec{\mathfrak{A}}$ nicht mehr Information über $\vec{\mathfrak{I}}$ offenlegt als die ungerade Parität im Fall $M_i = 1$.

Satz 4.1 *Sei $\vec{a} \in \mathrm{GF}(2)^n$ gegeben. Für jedes $\vec{i} \in \mathrm{GF}(2)^n$ mit gleicher Parität wie $\vec{a}$ und bei dem $\vec{\mathfrak{I}}$ mit einer Wahrscheinlichkeit größer Null vermutet wird, ist die bedingte Wahrscheinlichkeit daß $\vec{\mathfrak{A}} = \vec{a}$ unter der Voraussetzung $\vec{\mathfrak{I}} = \vec{i}$ genau 2^{1-n}. Damit entspricht auch die bedingte Wahrscheinlichkeit, daß $\vec{\mathfrak{I}} = \vec{i}$ unter der Voraussetzung $\vec{\mathfrak{A}} = \vec{a}$ ist, der a-priori Wahrscheinlichkeit für $\vec{\mathfrak{I}} = \vec{i}$.*

Beweis 4.1 *Angenommen $\vec{i} \in \mathrm{GF}(2)^n$ habe die gleiche Parität wie $\vec{a}$. Wir betrachten das System linearer Gleichungen $(\mathfrak{C}\vec{\mathfrak{K}}) \otimes \vec{i} = \vec{a}$ über Elementen aus*

$\mathrm{GF}(2)^n$. *Solange alle Spalten von* $\mathfrak{C}$ *gerade Parität haben, sind auch seine Zeilen linear abhängig über* $\mathrm{GF}(2)^m$. *Jede passende Teilmenge von Zeilen aus* $\mathfrak{C}$ *muß aber wegen der Vorgabe des zusammenhängenden Schlüsselgraphen linear unabhängig sein. Damit ist der Rang von* $\mathfrak{C}$ *genau* $n-1$, *so daß wir jeden Vektor mit gerader Parität als Linearkombination aus Spalten von* $\mathfrak{C}$ *schreiben können. Das impliziert die Lösbarkeit des Gleichungssystems, weil* $\vec{i} \otimes \vec{a}$ *auch diese gerade Parität hat. Solange die Menge der* m *Spaltenvektoren von* $\mathfrak{C}$ *den Rang* $m-1$ *hat, existieren jedoch genau* 2^{m-n+1} *Lösungen. Zusammen mit der Annahme, daß* $\vec{\mathfrak{K}}$ *von* $\vec{\mathfrak{I}}$ *unabhängig und gleichförmig verteilt ist, folgt die Gültigkeit des Satzes.* □

Die Verteilung der Ausgaben O_ℓ muß zuverlässig (vgl. Techniken des Kapitels 3) erfolgen, wobei für die Unbeobachtbarkeit vorallem deren Konsistenz wichtig ist. Besonders im Zusammenhang mit unserer Hauptforderung nach *Robustheit* stellt diese Art der Nachrichtenübermittlung eine zwingende Notwendigkeit dar. [54]

4.1. Zugriff auf den additiven Kanal

Ein schwieriges Problem bei der praktischen Implementierung ist die Koordination verschiedener Sender. Sind nämlich zwei oder mehr Nachrichtenzeichen (z. B. $M_i = 1$, $M_j = 1$, $i \neq j$) von ungerader Parität, wird das Ergebnis durch Überlagerung verfälscht. Hier kann nur ein Mehfachzugriffsprotokoll helfen. [32, 34]

Allgemein sollten Lösungen des Problems folgende Eigenschaften haben:

- Jeder Teilnehmer muß in der Lage sein, beliebig viele Daten zu übertragen.
- Alle Warte- oder Auslieferungszeiten werden möglichst kurz gehalten.
- Das Verfahren darf die gewonnene Unbeobachtbarkeit nicht gefährden.

Gerade der letzte Punkt ist nicht besonders einfach zu gewährleisten. Es existieren zwei grundverschiedene Herangehensweisen für diese Aufgaben.

4.1.1. Kollisionserkennung

Mit einem Fehlerortungssystem wird versucht, mögliche Sendekollisionen durch Techniken wie Prüfsummen oder Signaturen zu erkennen. Aufbauende Verfahren vermeiden dann bei Wiederholung des Vorgangs den simultanen Kanalzugriff. Ein Beispiel ist das bekannte ALOHA-Protkoll [55, 56], welches für satellitenorientierte Radiokommunikation entwickelt wurde. Man verwendet es in Netzwerken, wo alle Teilnehmer das gleiche Transportmedium (z. B. eine feste Radiofrequenz oder gemeinsames Kabel bei Ethernet-Bus-Topologie) benutzen.

Die Übertragung verläuft hier relativ einfach: Jeder Benutzer sendet verfügbare Daten sofort. Tritt eine Kollision auf, warten die Beteiligten eine zufällige Anzahl von Runden und probieren es danach erneut. Das Intervall dieser Zufallsauswahl wird mit steigender Versuchszahl immer größer, damit die Kollisionswahrscheinlichkeit sinkt. Obwohl das ALOHA-Protokoll für kleine n eigentlich sehr effizient ist, steht seinem praktischen Einsatz das unvorhersagbare Verhalten bei hohem Nachrichtenaufkommen entgegen. Außerdem kann es nur schwer *robust* gestaltet werden, weshalb es für unsere Zwecke leider weitgehend unbrauchbar bleibt. [34]

4.1.2. Reservierung von Sendeplätzen

Ein wesentlich besserer Ansatz ist senderbezogene Reservierung von Runden. Dieses Vorgehen scheint auf den ersten Blick die Unbeobachtbarkeit zu verletzen. Wir können es allerdings so modifizieren, daß uns sowohl Anonymität als auch Robustheit fast ohne Einschränkung erhalten bleibt. [23, 32, 34]

Man geht von der Grundidee aus, vor der eigentlichen Übertragung eine unbeobachtbare Reservierungsphase einzuführen. Hier einigen sich alle Teilnehmer auf eine festgelegte Reihenfolge für die Sendephase. Dazu stehen wieder diverse Methoden zur Verfügung, welche wir kurz behandeln und gegenüberstellen wollen.

Überlagerndes Reservieren nach J. Bos und B. den Boer [34, 57] ist eine mögliche Variante. Dazu sei ein endlicher Körper $\mathrm{GF}(p)$ mit Primzahl $p > R$ gegeben, wobei $R := n$ die Rundenanzahl der Reservierungsphase darstellt. Um einen Sendeplatz zu reservieren, wählt jeder Teilnehmer P_ℓ zufällig ein Nachrichtenzeichen $M_\ell \in \mathrm{GF}(p)$ aus. Dann wird das DC-Protokoll von Seite 82 über $\mathrm{GF}(p)$ für R Runden ausgeführt. In der r-ten Reservierungsrunde sendet P_ℓ die r-te Potenz, also $(M_\ell)^r \bmod p$, unbeobachtet an alle Parteien. Aus den im Schritt 3 entstehenden Potenzsummen (Addition als Operation der abelischen Gruppe) kann die Menge aller M_ℓ mit richtigen Vielfachheiten ermittelt werden. Doppelte Werte führen zu keiner Reservierung. Die arithmetische Ordnung über den verbleibenden Elementen bestimmt dann die Reihenfolge, in der später die Runden der Sendephase für einzelne Teilnehmer reserviert sind. Die Wahrscheinlichkeit von Kollisionen wird durch Wahl eines großen Körpers exponentiell klein.

Reservierungsleistentechnik: Unsere Version arbeitet in der additiven Gruppe des v-dimensionalen Vektorraums $\mathrm{GF}(2)^v$, wobei das $v \in \mathbb{N}$ eine ausreichend große natürliche Zahl darstellt. Die Reservierungsphase besteht hier aus $R := n^2$ Runden, um Kollisionen aufgrund des Geburtstagsparadoxons gering zu halten. Optimale Werte wurden von B. Bos [34] berechnet, sind allerdings immer quadratisch in n. Zur Reservierung eines Sendeplatzes wählt der Teilnehmer zufällig

einen Index $x \in \{1, \ldots, R\}$. Dann wird wieder das bekannte DC-Protokoll über $\mathrm{GF}(2)^v$ für R Runden ausgeführt. In der x-ten Reservierungsrunde sendet P_ℓ ein zufälliges Nachrichtenzeichen $M_\ell \in \{1, \ldots, 2^v - 1\}$, während er in den übrigen Runden jeweils $M_\ell := 0$ setzt. Beim Empfang erhöht P_ℓ einen internen Zähler $\hat{r} := \hat{r} + 1$, sobald $M \neq 0$ gilt. Falls er in Runde x die Gleichheit $M = M_\ell$ (scheinbar keine Kollision) feststellt, ist der $\hat{r}$-te Sendeplatz für ihn reserviert. Trotz aller Sorgfalt können Kollisionen auftreten, nämlich wenn sich einige Reservierungszeichen zum neutralen Element der Gruppe[69] überlagert haben. Die entsprechende Wahrscheinlichkeit bleibt aber bei großem v gering.

Die Parteien könnten die Position ihrer Nachrichten in der Sendephase beeinflussen, indem sie ihren Index x möglichst klein wählen. Soll ein solches Verhalten vermieden werden, kann eine zyklische Verschiebung der $\hat{r}$ um einen zufälligen Wert helfen. Dieser ist beispielsweise durch Addition aller empfangenen Zeichen (also $\sum_{\text{Runden}} M$) der Reservierungsphase für jeden Beteiligten ermittelbar. [34]

Fazit: Trotz seiner höheren Kommunikationskomplexität von $O(n^3)$ verwenden wir in `GnuDCN` die *Reservierungsleistentechnik*. Diese Effizienzeinschränkung ist notwendig, weil überlagerndes Reservieren einen erheblichen Implementierungsaufwand (Lösen schwieriger Gleichungssysteme, Polynomfaktorisierung) verursacht und deshalb die Grenzen dieser Diplomarbeit überschreiten würde.

4.2. Erhaltung der Robustheit bei Unbeobachtbarkeit

Bösartige Störer können die durch das DC-Protokoll gegebene Unbeobachtbarkeit ausnutzen, um ehrliche Teilnehmer an einer anonymen Kommunikation dauerhaft zu hindern. Wir wünschen uns deshalb eine weitere Eigenschaft des Verfahrens.

Robustheit: Störer, die mehr als eine zulässige Bandbreite (Anzahl der Nachrichtenzeichen $M_\ell \neq 0$) oder reservierte Runden anderer Teilnehmer verwenden, können im Nachhinein mit hoher Wahrscheinlichkeit erkannt und vom weiteren Protokollablauf ausgeschlossen werden.

Könnte man beliebige Runden des DC-Protokolls respektive Reservierung öffentlich nachrechnen (‚aufdecken'), so wäre die Lokalisierung von Störern sehr einfach. Natürlich ist ein solches Vorgehen für *sensitive Runden* (enthalten echte Nachrichten) nicht mit der Unbeobachtbarkeit vereinbar. *Nichtsensitive Runden* können hingegen gefahrlos aufgedeckt werden, da sie keine Nutzdaten beinhalten.

[69] Im Vektorraum $\mathrm{GF}(2)^v$ handelt es sich also um den Nullvektor $\vec{0}$.

Als Erweiterung müßte man also

1. nach einer Störung sensitive und nichtsensitive Runden sicher voneinander unterscheiden, und
2. verhindern können, daß ein Angreifer gezielt nur sensitive Runden stört.

Die Lösung dieser Probleme besteht nun darin, daß ehrliche Teilnehmer gezielt nichtsensitive Nachrichten (sogenannte Fallen) senden, die für alle anderen Parteien von echten, sensitiven Nachrichten nicht zu unterscheiden sind, für die aber nach einer Störung der eindeutige Beweis ihrer Nichtsensitivität erbracht werden kann. Alle bekannten Verfahren zur Gewährleistung der Robustheit basieren auf dieser Idee, weshalb sie kurz *Fallenprotokolle* genannt werden. Sie setzten ebenfalls zuverlässige Verteilung der Protokollmitteilungen voraus. [23, 32]

4.2.1. Ursprüngliches Fallenprotokoll von David Chaum

Ein erstes Verfahren wurde bereits von David Chaum [23] im Jahre 1988 skizziert. Das Fallenprotokoll sollte Robustheit gewährleisten, stellte sich jedoch später als unsicher heraus. Wir wollen es hier trotzdem behandeln, um ein Gefühl für notwendige Verbesserungen zu bekommen.

Fallenprotokoll: Jeder Teilnehmer P_i belegt in der Reservierungsphase genau einen Sendeplatz mit Index $\hat{r}_i$. Vor jeder Sendephase wird zufällig entschieden, ob er diesen Platz für eine echte Mitteilung oder Falle verwendet soll. Bei letzterem wählt P_i die zufällige Nachricht y_i und legt sich dann auf die Mitteilung

$$(\texttt{FALLE AN INDEX}, \hat{r}_i, \texttt{MIT DATEN}, y_i)$$

fest, indem er ihren Hashwert als *Fallenankündigung* veröffentlicht. Während der Sendephase setzt er in Runde $\hat{r}_i$ dann sein Nachrichtenzeichen auf $M_i := y_i$. Zusätzlich müssen alle Ausgabe O_i durch eine weitere Festlegung geschützt werden, damit bösartige Teilnehmer ihre Ausgaben nicht in Abhängigkeit von anderen bestimmen können. Bemerkt P_i bei der Rekonstruktion eine Störung seiner Falle, veröffentlicht er den Fallenbeweis (d. h. die Werte $\hat{r}_i$ und y_i). Der Angreifer wird durch Aufdecken aller Schlüsselbits dieser Runde identifiziert.

Aufdecken von Runden: Wird eine Runde $\hat{r}$ aufgrund eines gültigen Fallenbeweises aufgedeckt, veröffentlicht jeder Teilnehmer P_i sein dazugehörendes Nachrichtenzeichen M_i nebst allen Schlüsseln $K_{i,k}$. Die Berechnung von Schritt 3 des DC-Protokolls kann somit für jede Partei nachvollzogen und mit der festgelegten Ausgabe O_i verglichen werden.

Angenommen der böse Teilnehmer P_a habe gestört: Gibt P_a sich selbst als böse zu erkennen, z. B. indem seine Ausgabe nicht stimmt oder er gar nichts veröffentlicht, so wird er von der weiteren Teilnahme ausgeschlossen. Möchte er sich nicht direkt verraten, so muß er mindestens einen Schlüssel $K_{a,k}$ fälschen. Ist der Partnerknoten P_k ehrlich, wird dieser Versuch wegen $K_{a,k} \neq K_{k,a}$ erkannt, und P_k entfernt die Kante aus seinem Schlüsselgraphen. Wenn P_k ebenfalls bösartige Absichten verfolgt, kann er P_a zwar decken, muß dazu aber selbst einen Schlüssel abändern. Letztendlich wird stets ein böser Teilnehmer ausgeschlossen oder zumindest eine Schlüsselkante eines bösartigen Knotens entfernt.

Ist der Graph nach einem Angriff partitioniert, so wird das Fallenprotokoll in den einzelnen Teilgraphen getrennt ausgeführt. Da nie Kanten zwischen ehrlichen Parteien entfernt werden, sind selbige stets in einer Partition. Unser Angreifer verfügt nur über eine beschränkte Anzahl von Schlüsselpaaren bzw. Kollaborateuren, weshalb spätestens nach genauso vielen Störungen nur noch ehrliche Teilnehmer miteinander kommunizieren.

Diskussion: Aufgrund konsistenter Verteilung der Nachrichten kann jeder Teilnehmer lokal über das Entfernen von Kanten bzw. Knoten entscheiden und wird zu demselben Ergebnis wie andere kommen. Im asynchronen Umfeld sind jedoch ausbleibende Nachrichten ein großes Problem, das wir in `GnuDCN` durch unser Ausschlußprotokoll (siehe Abschnitt 3.2.2) lösen müssen.

Einen schwerwiegenden Fehler entdeckte M. Waidner [54]: Der Angreifer kann sogar die Unbeobachtbarkeit verletzen, indem er Fallenbeweise für nicht von ihm reservierte Runden veröffentlicht. Je nach Reservierungstechnik trifft er mit gewisser Wahrscheinlichkeit einen verwendeten Sendeplatz und kann dann für den Ausschluß des eigentlichen Besitzers sorgen. Verändert er statt einer echten Nachricht andere Fallen, wird der Angriff zwar erkannt, man kann den bösen Teilnehmer aber nicht vom rechtmäßigen Fallensteller unterscheiden.

Kritisch ist hier zusätzlich noch die Tatsache, daß alle Fallensteller eines Reservierungsabschnitts nicht gleichzeitig auch Sender von echten Nachrichten sein können. Damit wird die gewonnene Anonymität unnötigerweise eingeschränkt.

4.2.2. Verbesserungen von Waidner und Pfitzmann

Wir stellen jetzt das verbesserte Gesamtprotokoll vor, welches Hauptgrundlage unserer `GnuDCN`-Implementierung ist. Dabei greifen wir auf die spezielle Reservierungsleistentechnik aus Abschnitt 4.1.2 zurück. Die Veränderungen von M. Waidner und B. Pfitzmann [32] verhindern alle ursprünglichen Angriffe durch eine bessere Verkettung der Fallen mit ihren Ankündigungen.

Das Verfahren besteht aus drei Hauptphasen, in denen jeweils das DC-Protokoll (vgl. Seite 82) für Unbeobachtbarkeit zum Einsatz kommt: *Reservierungsphase*, *Ankündigungsphase* und *Sendephase*. Außerdem gibt es zwei *nichtanonyme Palaverphasen* und ggf. eine *Teil- bzw. Vollaufdeckphase*, um Störer auszuschließen. Die Abbildung auf dieser Seite zeigt den vereinfachten Gesamtablauf graphisch.

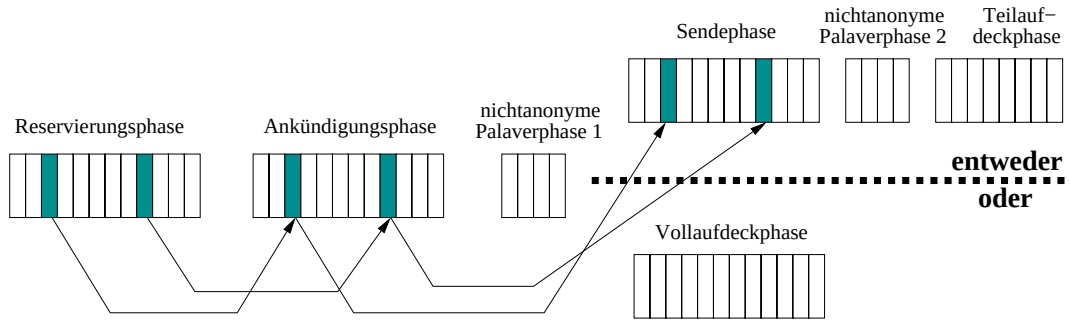

Abbildung 13: Phasen des verbesserten Fallenprotokolls

Jeder Teilnehmer P_i unternimmt in der *Reservierungsphase* den Versuch, zwei Sendeplätze $\hat{x}_i$ und $\hat{y}_i$ zu belegen. Ist die Reservierung erfolgreich, stellt er mindestens eine Falle. Bekommt er aufgrund einer Kollision nur einen Platz zugewiesen, darf er keine echte Nachricht verschicken, weil sonst die Unbeobachtbarkeit eingeschränkt wird. Grundsätzlich ist das Reservierungs- und Ankündigungsverhalten unabhängig von wirklichen Sendewünschen zu gestalten, da diese zwei Phasen u. U. vollständig aufgedeckt werden. In der $\hat{x}_i$-ten bzw. $\hat{y}_i$-ten Runde der *Ankündigungsphase* legt P_i sich auf `FALLE` bzw. `NACHRICHT` an diesem Index fest.

Vor Beginn der Sendephase werden Störungen der ersten beiden Teile in einer nichtanonymen *Palaverphase 1* angezeigt und führen dann unmittelbar zur *vollständigen Aufdeckung* dieser Phasen, ohne daß es überhaupt zum Senden der sensitiven Informationen kommt. Sind keine Fehler gemeldet, wird gemäß obiger Reservierung die *Sendephase* durchgeführt. Treten dabei Störungen auf, kommt es lediglich zu einer *Teilaufdeckung* der entsprechenden Runde, allerdings nur, wenn eine nichtsensitive Falle betroffen war und der Besitzer durch einen passenden Fallenbeweis diese Tatsache auch zeigen kann. Der dafür notwendige Ankündigungsbeweis wird in der nichtanonymen *Palaverphase 2* bekanntgegeben.

Alle Teilnehmer, die zugehörende Nachrichten nicht innerhalb einer festgelegten Zeitspanne (global definierte Warteschritte) senden, werden durch unser asynchrones Protokoll (siehe Abschnitt 3.2.2) *weich*-ausgeschlossen. Damit wird zwar die Unbeobachtbarkeit etwas eingeschränkt[70], für den Fortschritt des Gesamtverfahrens ist diese Vorgehensweise im asynchronen Umfeld jedoch unvermeidbar.

[70] z. B. wenn der Angreifer eine ehrliche Partei blockiert (vgl. Angriffsmodell des Verteilnetzes)

Ausgabenfestlegung: Im DC-Protokoll darf es den Subjekten nicht möglich sein, ihre Werte O_i in Abhängigkeit von anderen Ausgaben dieser Runde zu wählen. Dafür setzt man in der Regel ein kryptographisch sicheres Festlegschema (siehe Abschnitt 2.8.1 auf Seite 55) ein. Zuerst muß die eindeutige Festlegung $(h((R_1, R_2, O_i)), R_1)$ getroffen und an alle verteilt werden. Danach hat P_i die Ausgabe O_i und den Beweis (R_1, R_2) zu veröffentlichen. Reservierungs- und Sendephase sind so vor abhängigen Manipulationen geschützt.

Reservierungsphase: Wir verwenden unsere spezielle Reservierungsleistentechnik über dem Vektorraum $\mathrm{GF}(2)^v$.[71] Weil nun jeweils zwei Plätze $\hat{x}_i, \hat{y}_i$ zu reservieren sind, führt man insgesamt $2n^2$ Runden des DC-Protokolls (mit Ausgabenfestlegung) durch. Die Positionen x_i, $y_i \in \{1, \ldots, 2n^2\}$, $x_i \neq y_i$ werden vorher zufällig gewählt, um Kollisionen möglichst zu vermeiden. In der x_i-ten bzw. y_i-ten Reservierungsrunde sendet P_i dann ein zufälliges Nachrichtenzeichen $M_i \in \{1, \ldots, 2^v - 1\}$, ansonsten wird $M_i := 0$ gesetzt. Empfängt er das Zeichen unverändert ($M = M_i$), gilt Sendeplatz $\hat{x}_i := \hat{r}$ bzw. $\hat{y}_i := \hat{r}$ für ihn als reserviert; bemerkt er eine Kollision $M \neq M_i$, bleiben die Variablen $\hat{x}_i$ bzw. $\hat{y}_i$ undefiniert. Am Ende prüft P_i noch, ob $2 \leq \hat{r} \leq 2n$ Reservierungsversuche[72] unternommen wurden und meldet den negativen Fall zur späteren Palaverphase 1 als Störung.

Ankündigungsphase: Ankündigungen werden wiederum durch ein Festlegschema implementiert, d. h. jeder Teilnehmer P_i muß sich öffentlich auf die Werte `FALLE` oder `NACHRICHT` festlegen. Die Sicherheit dieses Schemas hat nicht nur Einfluß auf die Robustheit sondern auch auf die Unbeobachtbarkeit: Könnte der Angreifer die Nichtfallenankündigung einer anderen Partei als Fallenankündigung öffnen, würde beim Aufdecken der Sender dieser sensitiven Nachricht identifiziert. Zur anonymen Festlegverbreitung werden $\hat{r}$ Runden des DC-Protokolls (ohne Ausgabenfestlegung) über $\mathrm{GF}(2)^w$ durchgeführt, wobei w die maximale Länge einer Festlegung[73] ist. In der $\hat{x}_i$-ten bzw. $\hat{y}_i$-ten Ankündigungsrunde sendet P_i dann seine Festlegung als Nachrichtenzeichen M_i und prüft, ob er sie unverändert empfängt. Im negativen Fall liegt eine bösartige Störung vor, welche er in der folgenden Palaverphase 1 meldet. Die Ankündigungsphase muß nicht durch Ausgabenfestlegung über O_i geschützt werden, weil P_i bereits beim Aufdecken der Reservierungsphase (siehe Vollaufdeckphase) seine rechtmäßige Benutzung des Sendeplatzes beweisen kann.

[71] Im Prototyp `GnuDCN` ist der Einfachheit halber $v := 8$ gesetzt, so daß Sendeplätze damit genau durch ein Byte repräsentiert werden. Außerdem fixieren wir die Leistenlänge auf 2048, weil für alle $n \leq 32$ die Beziehung $2n^2 \leq 2048$ gilt.

[72] Hier besteht ein Unterschied zur Arbeit von Waidner und Pfitzmann [32], welche genau $2n$ Reservierungsversuche voraussetzen. Wir formulieren diese Bedingung weniger scharf, weil in unserer Leiste der Länge $2n^2$ Kollisionen durchaus wahrscheinlich sind.

[73] also maximale Anzahl der Bits von $(h(R_1, R_2, \texttt{FALLE}), R_1)$ bzw. $(h(R_1, R_2, \texttt{NACHRICHT}), R_1)$

Palaverphase 1: Jeder Teilnehmer, der eine Störung der ersten beiden Phasen feststellt, zeigt diese Tatsache nichtanonym durch Senden der `FAIL`-Nachricht an. Liegt mindestens eine solche Mitteilung vor, wird sofort die Vollaufdeckphase initiiert. Ansonsten wird nach dem Eingang von n `OK`-Nachrichten verschiedener Parteien die eigentliche Sendephase gestartet.

Vollaufdeckphase: Zeigte in der Palaverphase 1 wenigstens ein Teilnehmer eine Störung an, so werden alle Runden der ersten beiden Phasen aufgedeckt und jede Partei prüft lokal das Verhalten jedes anderen Teilnehmers nach: Zuerst werden die Regeln des DC-Protokolls geprüft; darauf aufbauend kontrolliert man die Reservierungs- und Ankündigungsphase. Nach jeder Störung wird entweder eine Kante oder ein Knoten (böser Teilnehmer) aus dem Schlüsselgraphen mittels Ausschlußprotokoll entfernt. Wird keine Protokollabweichung festgestellt, so sind alle Parteien böse, die in der Palaverphase einen Fehler angezeigt haben, und werden deshalb ausgeschlossen. Wegen der zuverlässigen Verteilung gelangen alle ehrlichen Teilnehmer zum selben Ergebnis.

Sendephase: Zeigte in der Palaverphase 1 kein Teilnehmer eine Störung an, so werden nun gemäß Reservierung die Sendeplätze verwendet. Dazu hat P_i nochmal $\hat{r}$ Runden des DC-Protokolls (mit Ausgabenfestlegung) über $\mathrm{GF}(2)^z$ durchzuführen, wobei z die maximale Länge einer Nutznachricht in Bit ist. Wird eine Runde als Falle genutzt, so wählen alle, natürlich auch der Fallensteller, als Nachrichtenzeichen $M_i := 0$. Um zu verhindern, daß Angreifer echte Mitteilungen von Fallen unterscheiden können, wird wieder Ausgabenfestlegung betrieben. Fallensteller beobachten außerdem, ob eine Störung $M \neq 0$ ihres Sendeplatzes vorliegt und melden diesen Sachverhalt ggf. in der folgenden Palaverphase 2.

Palaverphase 2: Hier veröffentlicht jede Partei die Anzahl ihrer gestörten Fallen und reserviert damit den zur Teilaufdeckung benötigten Bandbreitenanteil.

Teilaufdeckphase: In einer festen Ordnung öffnet jeder Fallensteller die zu seiner gestörten Falle gehörende Ankündigung und überzeugt damit alle ehrlichen Teilnehmer zweifelsfrei, daß die Schlüsselbits dieses Sendeplatzes aufgedeckt werden können. Weil in Fallen keine sensitiven Informationen enthalten sind, ist eine Aufdeckung problemlos möglich. Sollte kein Fehlverhalten feststellbar sein, wird die anzeigende Partei (durch Palaverphase 2 identifiziert) ausgeschlossen.

Fazit: Falls störende Angreifer nicht die Ausgabenfestlegung entschlüsseln und damit Ankündigungen klassifizieren können, erhält dieses Fallenprotokoll die Unbeobachtbarkeit bei gleichzeitiger Robustheit.

Tabelle 4.1 (theoretisches Nachrichtenaufkommen der Phasen)

Reservierungsphase	*Ankündigungsphase*	*Palaverphase 1*	*Vollaufdeckphase*	*Sendephase*	*Palaverphase 2*	*Teilaufdeckphase*
$4n^3$	$2n^2$	n	$2n^3 - n^2 - n$	$4n^2$	n	$n^3 - n^2$

Die Tabelle 4.1 zeigt das theoretische Gesamtnachrichtenaufkommen einzelner Phasen des verbesserten DC-Protokolls. Daraus läßt sich auch im Störfall eine Kommunikationskomplexität von $O(n^3)$ ableiten. Die eigentliche Nachrichtenkomplexität kann zwar durch das Zusammenfassen bestimmter Mitteilungen noch reduziert werden, eine Auswirkung auf die Anzahl der zu transportierenden Informationen hat solches Vorgehen jedoch nicht. Zusätzlich muß noch unser asynchrones Ausschlußprotokoll (inklusive ABBA) betrachtet werden, das probabilistisch (mit hoher Wahrscheinlichkeit) wiederum in die Klasse $O(n^3)$ fällt.

Alle erzeugten Nachrichten werden dann zuverlässig verteilt, wobei jede Transaktion nocheinmal den Aufwand $O(n^2)$ verursacht. Die Gesamtkommunikationskomplexität eines robusten DC-Netzes ist deshalb in die Klasse $O(n^5)$ einzuordnen. Mit der Reservierungstechnik von J. Bos und B. den Boer [57] oder einer Fixierung der Reservierungsleistenlänge[74] könnte im optimalen Fall (d. h. ohne Störungen/ Ausfälle) der Grad noch um eins verringert werden.

[74] Diese Methode zur Reduktion der Komplexität verwenden wir in `GnuDCN`, um überhaupt ein praktikables Transportvolumen für kleine Gruppen (z. B. $n \leq 32$) zu erreichen. Auf damit verbundene Probleme (Kollisionen) gehen wir später noch ein.

5. Implementierung und Resultate

Wir sind uns mehr als die frühere Naturwissenschaft dessen bewußt, daß es keinen sicheren Ausgangspunkt gibt, von dem aus Wege in alle Gebiete des Erkennbaren führen, sondern daß alle Erkenntnis gewissermaßen über einer grandiosen Tiefe schweben muß; daß wir stets irgendwo in der Mitte anfangen müssen, über die Wirklichkeit zu sprechen mitbegriffen, die erst durch ihre Anwendung allmählich einen schärferen Sinn erhalten, und daß selbst die schärfsten aller Anforderungen an logische und mathematische Präzision genügenden Begriffssysteme nur tastende Versuche sind, uns in begrenzten Bereichen der Wirklichkeit zurechtzufinden.
— Werner Heisenberg

In diesem Kapitel betrachten wir unsere Implementierung `GnuDCN` und die bei ihrem Entwurf verfolgten Ziele. Alle ‚Bausteine' der vorherigen Abschnitte werden jetzt unter technischen Gesichtspunkten zusammengefügt und formen ein robustes, vollverbundenes DC-Netz. Damit eröffnet sich erstmals die faszinierende Möglichkeit, konkrete Messungen und Versuche zur Praxistauglichkeit solcher Systeme durchzuführen. Den Abschluß bildet dann die Beantwortung der zentralen Frage: Sind DC-Netze praktikable Werkzeuge zur Wahrung der Anonymität?

5.1. `GnuDCN` — eine robuste DC-Netz Implementierung

Alle Teile von `GnuDCN`, was für *GNU Dining Cryptographers Network* steht, wurden in der Programmiersprache C/C++ realisiert. Die Quelltexte umfassen zirka 200 000 Zeichen[75] in 7700 Zeilen und sind damit noch von überschaubarer Größenordnung. Die Sprache C/C++ war hauptsächlich durch drei Punkte vorgegeben:

- freie Verfüg- und Verwendbarkeit diverser Programmbibliotheken
  Deren Einsatz vermeidet aufwendige Eigenentwicklungen, reduziert gleichzeitig die Programmkomplexität und ermöglicht eine schnellere Fehlersuche.
- stabile und hochoptimierende Compiler, z. B. GNU `gcc`, `g++`[76]
  Gerade im kryptographischen Umfeld sind dadurch erzielte Laufzeitverbesserungen oft signifikant.
- Verbreitung, Portabilität und Standardisierung der Programmiersprache

Wir wollen jetzt einen Blick auf die dominierenden Designziele beim Entwurf von `GnuDCN` wagen und später den grundlegenden Aufbau des Systems besprechen.

[75] Zum Vergleich: Diese Diplomarbeit enthält ca. 300 000 Zeichen in etwa 4000 Zeilen.
[76] siehe `http://www.gnu.org/software/gcc/`

5.1.1. Designziele und konzeptioneller Aufbau

Hauptaufgabe von `GnuDCN` ist natürlich die *Wahrung der Senderanonymität*. Als weitergehende Ziele standen bei der Implementierung jedoch im Mittelpunkt:

Robustheit: Für n vorher festgelegte Subjekte (statische Konfiguration) soll der robuste Betrieb eines DC-Netzes möglich sein, solange die Anzahl bösartiger Störer (oder blockierter Teilnehmer) kleiner als $n/3$ bleibt. Die Unbeobachtbarkeit darf auch im negativen Fall nur mit vernachlässigbarer Wahrscheinlichkeit gefährdet werden.

Sicherheit: Wir greifen nur auf gesicherte kryptographische Annahmen zurück — also solche, die in jahrelangen Analysen *keine* wesentlichen Schwächen gezeigt haben. Für alle darauf aufbauenden Algorithmen und Protokolle sollte ein Beweis ihrer Sicherheit vorliegen.

Offenheit: Verwendete Programmbibliotheken (siehe Seite 118) sowie die eigentlichen Quelltexte von `GnuDCN` sollten öffentlich und frei verfügbar sein, z. B. unter der bekannten *GNU General Public License* [60]. Spezielle wissenschaftliche Untersuchungen [62, 63] haben gezeigt, daß nur eine vollständige Offenlegung überhaupt effektive Sicherheitsanalysen ermöglicht und damit letztendlich auch das Risiko von Fehlern minimiert. Außerdem wird die Weiterentwicklung und der Einsatz in anderen freien Projekten gefördert.

Beim herkömmlichen *Closed-Source-Modell* können versteckte trojanische Pferde oder Hintertüren[77] nur schwer entdeckt werden und stellen die Vertrauenswürdigkeit solcher Software auf eine harte Probe. Das scheinheilig anmutende Paradigma der *„Security by Obscurity“*[78] ist zwar weit verbreitet, wird allerdings in seriösen Kreisen der IT-Sicherheit durchweg abgelehnt.

Flexibilität: Die Architektur von `GnuDCN` soll weitgehend unabhängig gestaltet werden, damit Teilfunktionen (z. B. zuverlässige Verteilung) auch als Einzelkomponenten benutzbar sind. Eine Gliederung in verschiedene Aufgabenschichten mit gemeinsamer Schnittstelle scheint deshalb sinnvoll. Neben der höheren Flexibilität bestehender Module können so spätere Erweiterungen relativ einfach integriert werden.

Prototypischer Charakter: Die `GnuDCN`-Entwicklung sollte in erster Linie einer Bewertung von Machbarkeit und Praktikabilität des DC-Netz-Ansatzes im Rahmen dieser Studie dienen. Es handelt sich also eher um ein prototypisches System und ist deshalb weniger für den produktiven Einsatz geeignet.

Aus diesem Ziel ergeben sich eine Reihe von notwendigen Einschränkungen/Grenzen, welche wir demnächst noch genauer beschreiben werden.

[77] Ein verstecktes Hinzufügen der Absenderadresse ist beispielsweise ohne weiteres möglich.
[78] Sicherheit durch Unbekanntheit

Nachnutzung: Kryptographische Teilsysteme, wie beispielsweise das Schwellwertsignaturschema, sollen als objektorientierte Klassen (C++) auch für andere Zwecke separat nutzbar sein.

Gemäß diesen Vorstellungen wurde ein System entwickelt, welches nach der initialen *Konfiguration* durch den vertrauenswürdigen Vermittler eine beliebige Anzahl von Nachrichten im *Anwendungsbetrieb* senderanonym transportieren kann.

Konfiguration: Der vertrauenswürdige Vermittler erzeugt für jeden Teilnehmer eine individuelle Konfigurationsdatei mit folgendem Inhalt:

- normales RSA-Schlüsselpaar für Signatur und Geheimhaltung
- öffentliche Schlüsselkomponenten aller anderen Subjekte[79]
- Schlüssel des $/n, n-t, t/$- bzw. $/n, t+1, t/$-Schwellenwertsignaturschemas
- grundsätzlicher Aufbau des DC-Netz-Schlüsselgraphen (Knoten, Kanten)
- Anfangszustände und Moduli der Pseudozufallsbitgeneratoren, welche für beide Enden einer Schlüsselkante $K_{\ell,k}$ (DC-Netz) identisch sind

Er initalisiert damit viele kryptographische Bestandteile und legt auch deren Sicherheitsparameter (z. B. Länge des jeweiligen Primzahlprodukts) fest.

Das `GnuDCN`-Programm `TrustedDealer` erwartet als Parameter die Anzahl der Teilnehmer n und eine gewünschte Schlüssellänge in Bit. Damit erzeugte Konfigurationsdateien müssen später noch sicher, d. h. geheim und authentisch, an die entsprechenden Subjekte verteilt werden. Hierfür ist u. U. eine bestehende Public-Key-Infrastruktur (siehe ggf. Seite 26) mit dem Vermittler notwendig, falls die Konfiguration nicht persönlich von Trent vorgenommen wird.

Anwendungsbetrieb: Die `GnuDCN`-Module für die senderanonyme Verteilung sind schichtartig angeordnet, um einer höheren Abstraktion und *Flexibilität* zu genügen. Abbildung 14 auf Seite 96 zeigt diese Gliederung.

Die *Anwendung* (`SimpleInterface`) sendet/empfängt anonym verteilte Nachrichten über einen IPC-Kanal[80] vom unteren Modul `AnonymousBroadcast`. Jenes wiederum benötigt zuverlässige Verteilung, welche durch den tiefer arbeitenden `ReliableBroadcast`-Prozeß bereitgestellt wird. Schließlich kümmert sich noch unser Systemmodul `TechnicalBroadcast` um die eigentliche Verteilung, wobei es auf das übliche TCP/IP-Netzwerkprotokoll zurückgreift.

[79] Da Trent für alle Parteien vertrauenswürdig ist, bleibt uns zumindest hier eine komplizierte Public-Key-Infrastruktur erspart.

[80] Interprozeßkommunikation durch systemweite Nachrichtenwarteschlangen

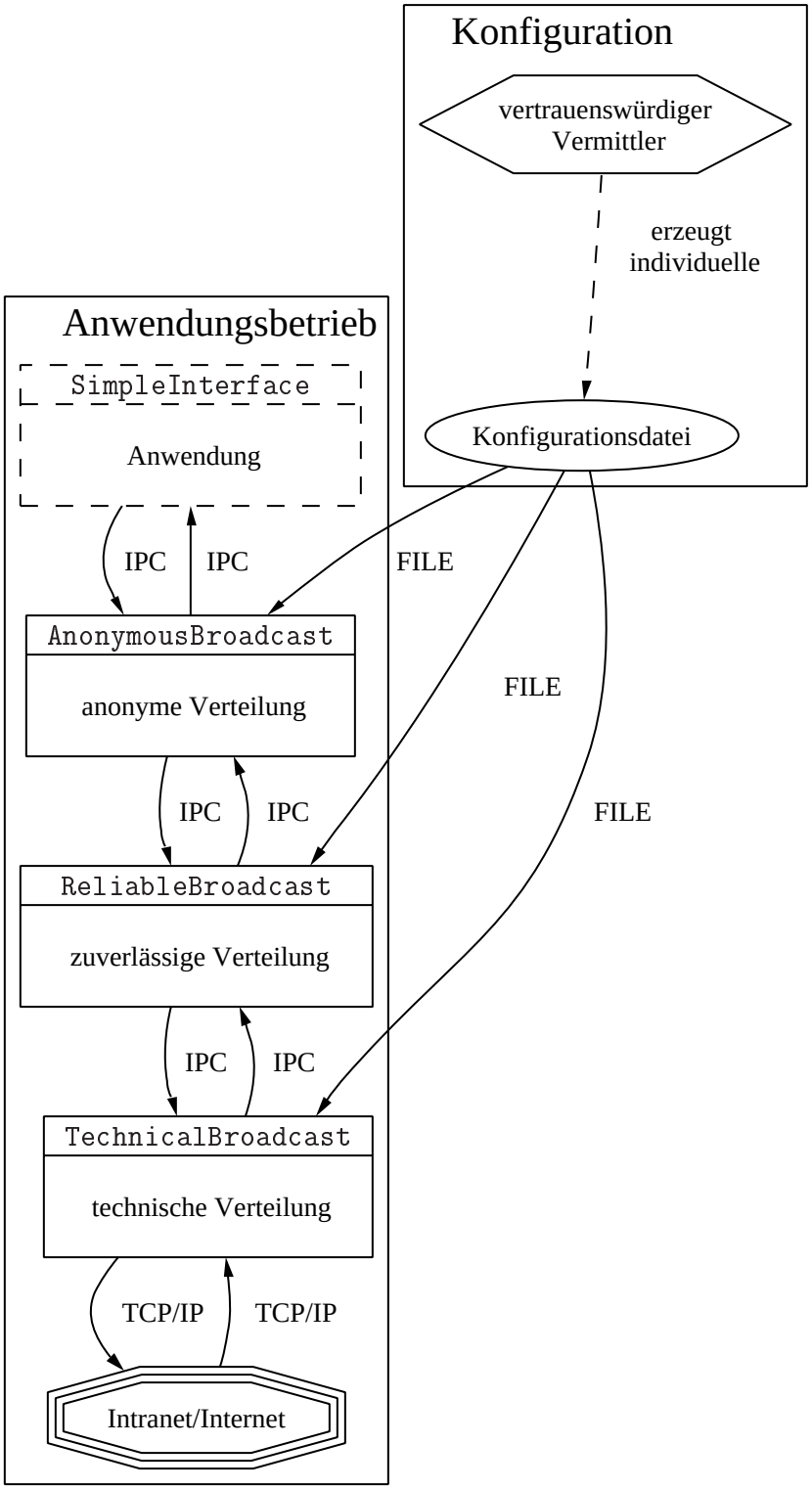

Abbildung 14: Aufbau und Schichten von GnuDCN

Alle Module verfügen über den selben Aufbau ihrer IPC-Schnittstelle, so daß auch eine andere (z. B. mehrfache) Kombination problemlos möglich ist. Für unsere Aufgabenstellung bleibt jedoch nur die beschriebene Anordnung sinnvoll.

Einschränkungen: Leider müssen wir aufgrund des prototypischen Charakters viele Einschränkungen beim Betrieb von `GnuDCN` in Kauf nehmen:

- Die Teilnehmermenge ist *statisch*, d. h. neue Subjekte können nur durch Rekonfiguration vom Vermittler hinzugefügt werden.
- Nach jeder (allerdings beliebig langen) Benutzung muß das `GnuDCN`-System neu konfiguriert werden. Ein wiederholtes Verwenden der gleichen kryptographischen Initalisierung (insbesondere der Pseudozufallsbitgeneratoren) gefährdet sonst die Sicherheit grundlegend.
- Die maximale Nutzdatengröße ist betriebssystembedingt auf 1024 Zeichen pro Nachricht beschränkt. (3328 Zeichen IPC-Warteschlangenlimit)
- Alle Reservierungsrunden des DC-Netz-Protokolls wurden für kleine Gruppen ($n \leq 32$) fixiert, um dessen gravierende Nachrichtenkomplexität zu senken. Sollen erheblich mehr Subjekte teilnehmen, muß entweder der Transportaufwand erhöht oder die Kollisionsbehandlung verändert werden.
- Die technische Verteilung der Nachrichten erfolgt der Einfachheit halber über einen zentralen Server. Andere Modelle (z. B. Punkt-zu-Punkt Verbindungen, Multicast-Routing) bieten zwar einen größeren Außenschutz unserer Robustheit, sind jedoch nicht Gegenstand dieser Arbeit.

5.1.2. Umsetzung kryptographischer Grundlagen

Zwei wichtige Fremdbibliotheken [A, B] erleichtern uns die Implementierung:

`libgmp` stellt alle notwendigen arithmetischen Operation für beliebig lange Zahlen bereit. Auch komplizierte Funktionen, wie Berechnung des Jacobisymbols, sind in dieser Bibliothek enthalten und zudem effizient umgesetzt.

`libgcrypt` bietet uns kryptographische Primitiven, von denen wir insbesondere die Hashfunktion, einige Routinen für sicheren Pseudozufall sowie Verschlüsselungs- und Authentifizierungsalgorithmen verwenden.

Beide Funktionssammlungen wurden für viele Rechnerarchitekturen optimiert und sind unter diversen Betriebssystemen lauffähig. Sie stehen damit der Portierung von `GnuDCN` auf andere Plattformen nicht entgegen.

`SchneierHashBCS` **Festlegeschema (aus Abschnitt 2.8.1)** Diese Klasse ist eine objektorientierte Implementierung des unter den Grundlagen bereits beschriebenen Verfahrens.[81] Wir setzen sie im Rahmen der Ausgabenfestlegung und Fallenankündigung des verbesserten DC-Protokolls (siehe Kapitel 4) ein. Ihre speziellen Attribute und Methoden sind im Anhang A.2 genauer erklärt.

`BlumBlumShubPRBG` **Pseudozufallsbitgenerator (aus Abschnitt 2.8.2)** Dieser kryptographisch sichere Generator liefert eine Sequenz von Pseudozufallsbits, welche wir im DC-Protokoll als Schlüssel für die Kanten des Graphen benötigen. Dazu muß der Konstruktor jeweils paarweise mit dem gleichen Primzahlprodukt aufgerufen und sein Anfangszustand identisch initalisiert werden. Auch für diese Klasse werden Attribute und Methoden im Anhang A.3 erläutert.

`ShoupRsaDTSS` **Schwellenwertsignaturschema (aus Abschnitt 2.6.1)** Die Klasse erfaßt alle Bestandteile des im zweiten Kapitel ausführlich behandelten Signaturschemas in seiner verbesserten und effizienteren $k \geq t+1$ Version. Sie wird bei der konsistenten Verteilung und im ABBA-Protokoll (siehe unten) instanziert. Unsere diesbezügliche Implementierung braucht viele Funktionen der beiden Fremdbibliotheken, insbesondere modulare Arithmetik, Hashfunktion RIPE-MD160 [20] und den Pseudozufall. Klassenattribute und -methoden des Schemas sind im Anhang A.1 kurz zusammengestellt.

`CachinKursaweShoupABBA` **ABBA-Protokoll (aus Abschnitt 3.2.1)** Eines der wichtigsten Teilprojekte war die objektorientierte Umsetzung des komplizierten ABBA-Protokolls.[82] Kontroll- und Nachrichtenfluß wurden durch kooperative Nebenläufigkeit realisiert, um Problemen mit ungenügend reentranten Fremdbibliotheken aus dem Wege zu gehen. Die Klasse wird in unserem asynchronen Ausschlußprotokoll eingesetzt und sichert damit die Robustheit des DC-Netzes. Ihre Attribute und Methoden zeigt der Anhang A.4 in einer Übersicht.

5.1.3. Implementierung der Schichtprotokolle

Alle Module sind eigenständige Programme, die über IPC-Kanäle miteinander kommunizieren. Notwendige Konfigurationsdaten werden in Form einer `.dot`-Datei vom vertrauenswürdigen Vermittler bereitgestellt. Die syntaktische Verarbeitung erfolgt dann mit Hilfe der Fremdbibliothek `libgraph` [C].

[81] Hashfunktion RIPE-MD160 [20] und notwendiger Pseudozufall (Entropie aus Betriebssystem- oder I/O-Ereignissen) werden von der Fremdbibliothek `libgcrypt` [B] geliefert.

[82] inklusive der Erweiterungen für *starke Richtigkeit* und *starke Termination*

`SimpleInterface` – **eine einfache Anwendung** Für Testzwecke wurde innerhalb der Anwendungsschicht lediglich ein triviales Programm implementiert. Es verarbeitet Texteingaben zu Nachrichten, speist sie in den gewünschten IPC-Kanal ein und gibt auch empfangene Mitteilungen wieder aus. Dabei werden Funktionen der `libreadline`-Bibliothek [D] verwendet, um etwas mehr Eingabekomfort zu erreichen. In dieser Schicht wären also alle Anwendungen/Protokolle anzusiedeln, welche eine senderanonyme Verteilung ihrer Nachrichten benötigen.

`AnonymousBroadcast` – **anonyme Verteilung** Dieses Modul realisiert das im Kapitel 4 vorgestellte und verbesserte DC-Netz-Protokoll. Nach jeder Runde bzw. beim Eintreten eines Nachrichtentimeout wird zusätzlich das asynchrone Ausschlußprotokoll[83] gestartet, um bösartige Teilnehmer konsistent zu entfernen.

`ReliableBroadcast` – **zuverlässige Verteilung** Nachrichten werden durch dieses Modul zuverlässig verteilt, sofern es von einer höheren Schicht gewünscht ist. Grundlage für die Implementierung war unser behandeltes Protokoll aus Abschnitt 3.3, welches auch ein Schwellenwertsignaturschema verwendet.

`TechnicalBroadcast` – **technische Verteilung** Diese Schicht hat zwei Aufgaben: Zuerst werden ausgehende Nachrichten mit einer normalen RSA-Signatur versehen, um ihre Authentizität sicherzustellen, später erfolgt dann die technische Verteilung an alle Teilnehmer. Dafür gibt es verschiedene Möglichkeiten, von denen in unserem Prototyp `GnuDCN` nur der erste Punkt realisiert wurde.

Client-Server-Modell: Ein zentraler Server nimmt Mitteilungen des Klienten entgegen und sendet sie an alle beteiligten Parteien. Die Funktionsfähigkeit wäre beim Ausfall dieses Computers gefährdet, weshalb er in einer hochverfügbaren und unabhängigen Umgebung betrieben werden muß.

Punkt-zu-Punkt: Die Teilnehmer unterhalten unabhängige Nachrichtenkanäle und sind für die Verteilung selbst verantwortlich. Der Ausfall eines einzelnen Rechners ist dadurch nicht mehr so kritisch. Adressierungsfragen muß der vertrauenswürdige Vermittler durch globale Festlegung behandeln.

Multicast-Routing: Die Verteilung wird vollständig den Multicast-Protokollen des Netzwerks überlassen, wobei die Funktionsfähigkeit von ihnen abhängt.

Die zweite oder dritte Form scheint aus praktischen Gesichtspunkten am vernünftigsten. Ihre Migration sei für produktive Systeme dringend angeraten.

Zusammenfassend sind Abhängigkeiten (`-->`) zwischen einzelnen Modulen, Klassen und Bibliotheken in der kompakten Graphik 15 auf Seite 100 dargestellt.

[83] basierend auf der asynchronen binären Byzantinischen Übereinstimmung (ABBA)

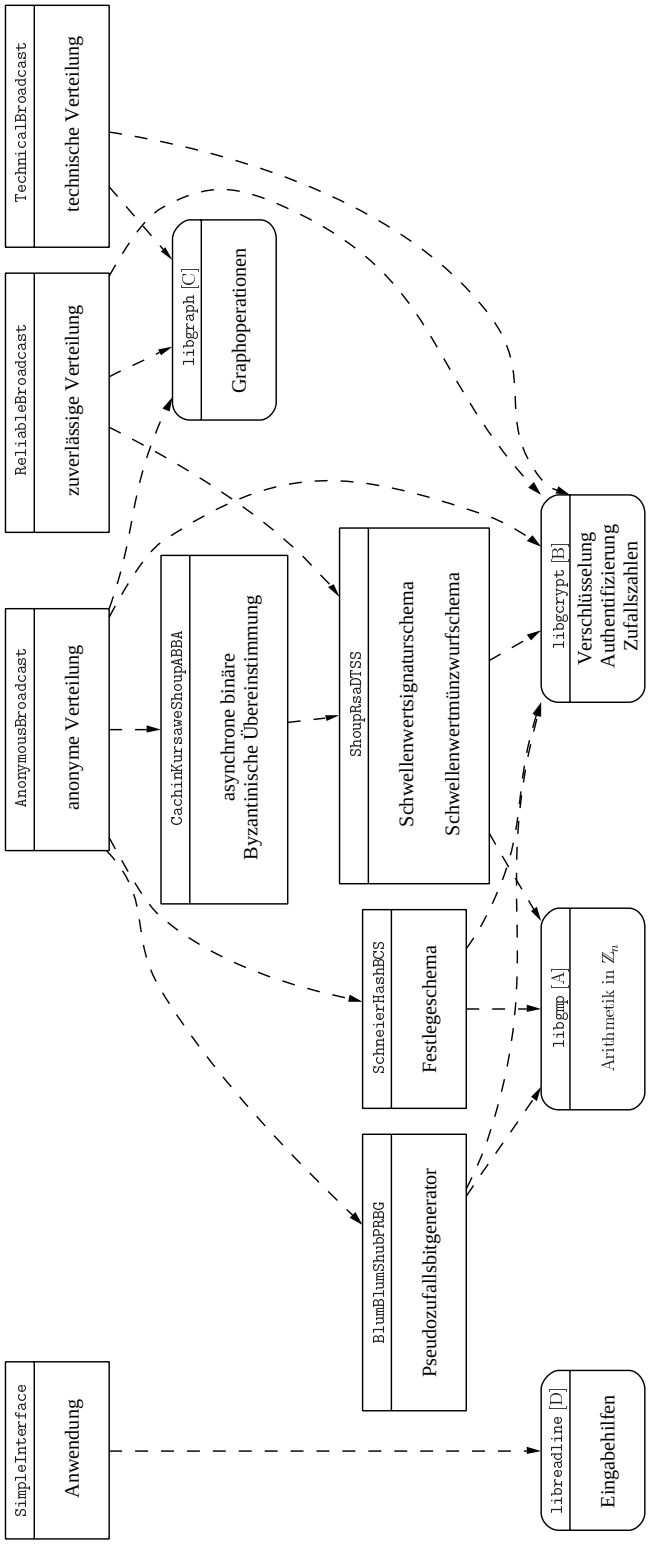

Abbildung 15: Abhängigkeiten zwischen Modulen, Klassen und Bibliotheken

5.2. GnuDCN — Messungen und Komplexitätsabschätzungen

Am Ende des vierten Kapitels hatten wir bereits festgestellt, daß die Kommunikationskomplexität eines robusten DC-Netzes die Größenordnung $O(n^5)$ erreicht. Die Fixierung der Reservierungsleistenlänge für kleine Gruppen ($n \leq 32$) senkt den Aufwand jedoch erheblich: Im optimalen Fall (ohne Störungen/Kollisionen) arbeitet GnuDCN in der Ordnung $O(n^3)$, wobei seine Nachrichtenkomplexität pro Runde[84] den Wert $10n^3 + 24n^2 + 8n$ nicht unterschreiten kann. Selbst bei Störungen oder Ausfällen von bis zu t Teilnehmern steigt der Aufwand maximal auf $6n^4 + 17n^3 + 2n^2 + 2n$ an. Diese Ober- und Untergrenzen ergeben sich durch Komposition des Nachrichtenaufkommens der einzelnen Schichtprotokolle, allerdings ohne Betrachtung des TechnicalBroadcast-Moduls. Dessen Komplexität hängt leider zu stark von seiner Implementierungsstrategie (siehe Seite 99) und der zugrundeliegenden Netzwerktopologie ab.

Obige Abschätzung soll jetzt mit Meßwerten von GnuDCN untermauert werden.

Tabelle 5.1 (Messung einer störungsfreien Runde von GnuDCN)

		Gesamtnachrichtenkomplexität aller Teilnehmer		
n	t	SimpleInterface	AnonymousBroadcast	ReliableBroadcast
4	1	4	176	1232
7	2	7	518	5180
10	3	10	1040	13520
17	5	17	2958	59160
25	8	25	6350	177800

Tabelle 5.2 (Messung einer störungsfreien Runde von GnuDCN)

		Gesamtkommunikationskomplexität in kByte		
n	t	SimpleInterface	AnonymousBroadcast	ReliableBroadcast
4	1	4	246.76	1375.21
7	2	7	727.18	5635.58
10	3	10	1460.63	14489.01
17	5	17	4159.28	62626.41
25	8	25	8967.23	186655.27

Nachrichten- und Kommunikationskomplexität wurden für die Gruppenstärken $n = 4, 7, 10, 17, 25$ jeweils an den IPC-Kanalausgängen unserer GnuDCN-Module gemessen. Vorerst betrachten wir nur den Fall der optimalen Runde ohne Störung bzw. Kollision. Die ReliableBroadcast-Werte (siehe Tabelle 5.1) liegen deshalb recht nahe am theoretischen Minimum von $10n^3 + 24n^2 + 8n$. Diagramm 16 (auf der nächsten Seite) gibt dieses Verhalten überaus deutlich wieder, wobei seine Y-Achse logarithmisch zur Basis 2 skaliert wurde.

[84] In einer Runde kann jede Partei höchstens eine Nachricht senderanonym verteilen.

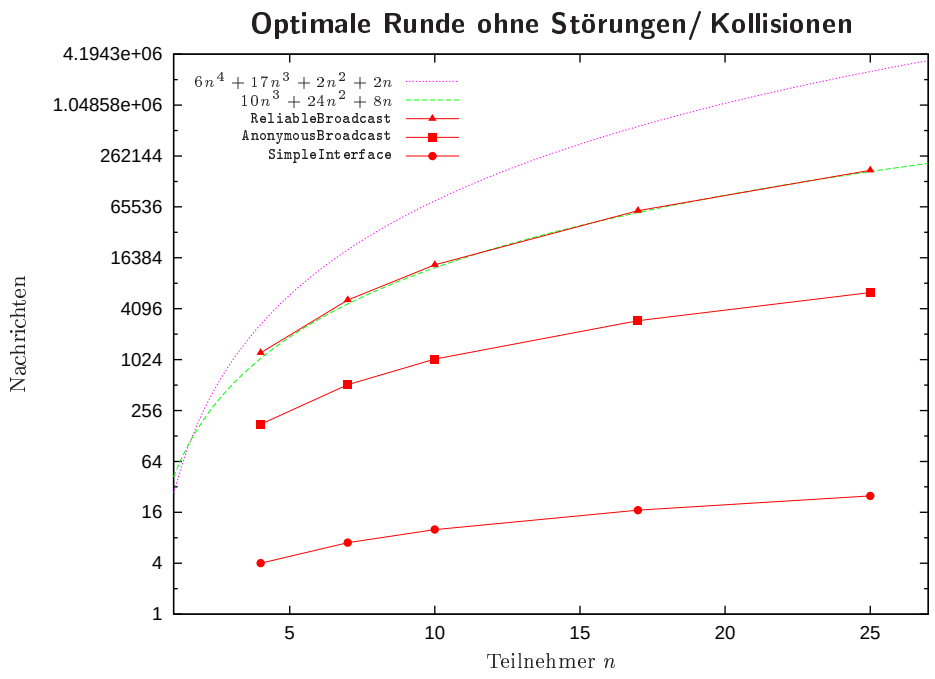

Abbildung 16: normale Nachrichtenkomplexität von `GnuDCN`

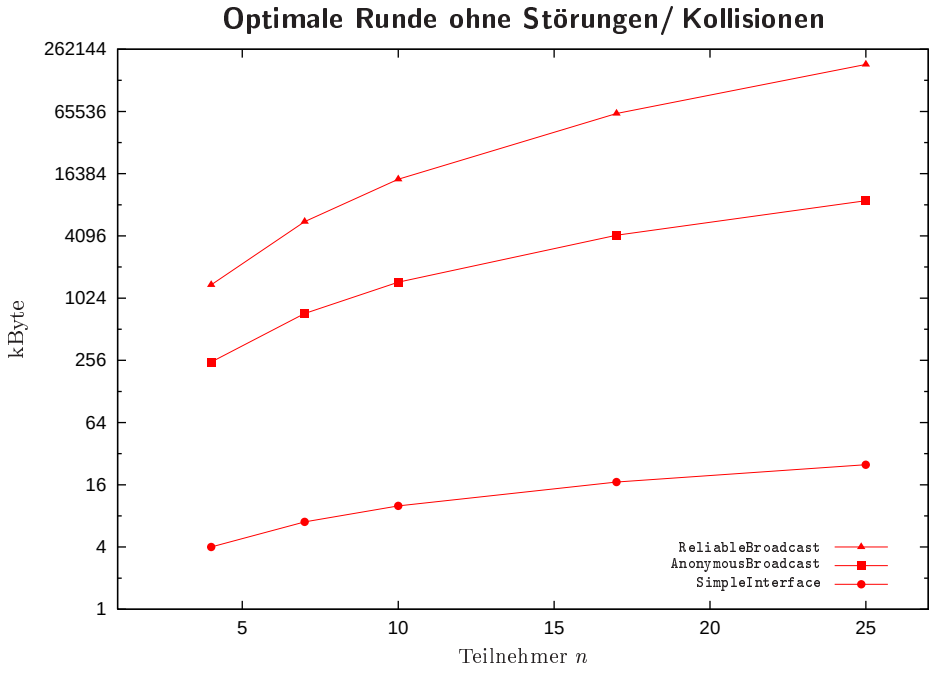

Abbildung 17: normale Kommunikationskomplexität von `GnuDCN`

Eine weitere Meßreihe zeigt das Verhalten von GnuDCN im Störfall. Exemplarisch wurde hierfür die Vollaufdeckung der Reservierungs- und Ankündigungsphase (verbessertes DC-Protokoll) betrachtet, da sie proportional den höchsten Aufwand verursacht. Um t bösartige Teilnehmer vollständig auszuschließen, müßten wir allerdings $tn - t$ solche Runden durchlaufen, weil minimal jeweils nur eine ‚schlechte' Kante des Graphen entfernt wird. Tabelle 5.3 und Diagramm 18 stellen die erhöhte Nachrichtenkomplexität einer einzelnen Vollaufdeckung dar.

Tabelle 5.3 (erhöhte Nachrichtenkomplexiät von GnuDCN)

		Gesamtnachrichtenkomplexität aller Teilnehmer		
n	t	SimpleInterface	AnonymousBroadcast	ReliableBroadcast
4	1	4	216	1512
7	2	7	945	9450
10	3	10	2520	32760
17	5	17	11305	226100
25	8	25	34425	963900

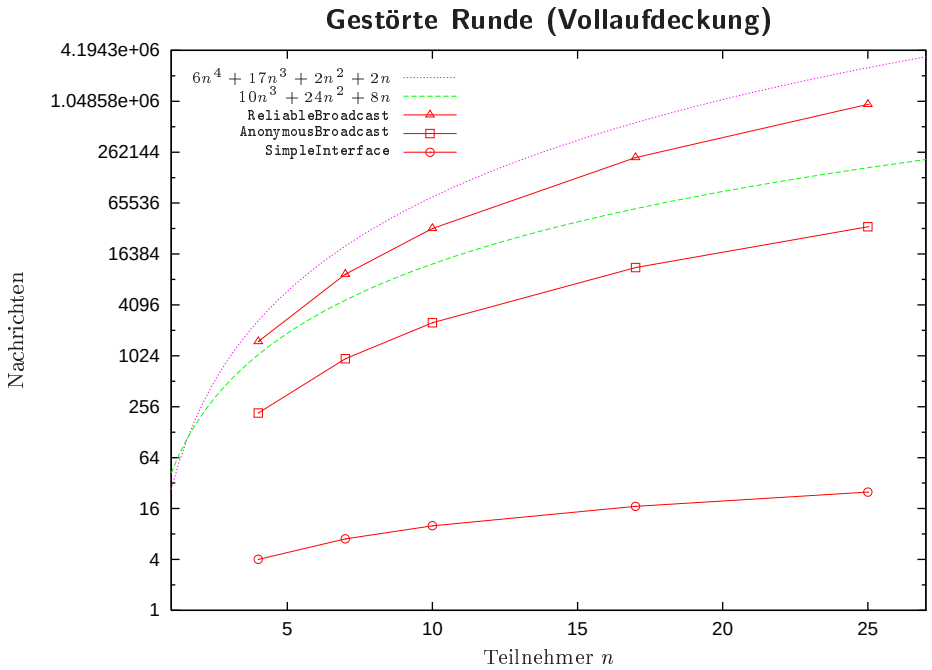

Abbildung 18: erhöhte Nachrichtenkomplexität von GnuDCN

Fazit: Alle gemessenen Daten bestätigen unsere Annahme, daß sich die Kommunikationskomplexität von GnuDCN im optimalen Fall kubisch $O(n^3)$ und im Störfall biquadratisch $O(n^4)$ verhält. Mit dem hinzukommenden linearen Aufwand von TechnicalBroadcast (aktuelle Implementierung) erreichen wir dann insgesamt die angedeutete Komplexitätsklasse $O(n^5)$.

5.3. GnuDCN — Bewertung der Praktikablilität

Zwei wesentliche Gründe stehen einem praktischen Einsatz zur Zeit entgegen:

- die enorme Nachrichten- und Kommunikationskomplexität
- die statische Konfiguration durch den vertrauenswürdigen Vermittler

Der erste Punkt wiegt schwer, denn beispielsweise müssen für $n = 25$ Teilnehmer in einer Runde[85] ohne Störungen zirka 4.45 Gigabyte übertragen werden. Zum zweiten Grund präsentieren wir später im Ausblick noch Lösungsvorschläge.

Trotzdem können DC-Netze bereits heute in besonders kritischen Umgebungen nützlich sein, was wir an zwei Fallbeispielen zeigen möchten:

Anonyme Abstimmung: Eine kleine Gruppe (z. B. $n = 10$) möchte eine geheime Abstimmung in einem Hochgeschwindigkeitsnetz (LAN) durchführen. Es könnte sich dabei beispielsweise um den Vorstand eines Unternehmens/ Vereins handeln, der eine unangenehme Entscheidung treffen muß. Da kein vertrauenswürdiger Vermittler zur Verfügung steht, gehen die Teilnehmer wie folgt vor:

1. Gemeinsame Beschaffung eines möglichst vertrauenswürdigen Rechners mit Laufwerk für wechselbare und beschreibare Medien. Danach erfolgt die Installation von `GnuDCN` auf dieser Plattform.
2. Jeder Teilnehmer überprüft die Quelltexte von `GnuDCN`, des Betriebssystems und des Compilers auf Korrektheit, um böswillige Veränderungen oder Hintertüren zu erkennen.[86]
3. Gemeinsame Ausführung des `GnuDCN`-Programms `TrusedDealer 10 1024` für die Schlüsselkonfiguration mit 1024 Bit Primzahlproduktlänge, wobei die `.dot`-Dateien auf verschiedenen Medien gespeichert und so nur dem jeweiligen Teilnehmer ausgehändigt werden.
4. Löschen des vom Programm `TrustedDealer` verwendeten Arbeitsspeichers.
5. Jeder Teilnehmer installiert `GnuDCN` auf dem eigenen (vertrauenswürdigen) Rechner und konfiguriert es mit den erhaltenen Daten.
6. Durchführung des gewünschten Abstimmungsprotokolls durch Verteilung senderanonymer Nachrichten mit Hilfe von `GnuDCN`.

Die Sicherheit basiert hier nur marginal auf Vertrauen. (Kontrollmöglichkeit)

Kombination mit MIX-Netzen: Man kann `GnuDCN` für die Kommunikation zwischen Knoten verwenden, um die Anonymität eines MIX-Netzes zu stärken und die notwendige Kaskadenlänge zu reduzieren. Die initiale Konfiguration muß von allen Knotenbetreibern wieder gemeinsam durchgeführt werden, wenn man nicht auf eine global vertrauenswürdige Instanz zurückgreifen möchte.

[85] Jeweils ein Kilobyte (also insgesamt 25 kB) können wir dabei senderanonym verteilen.

[86] Wenn man den Autoren der Software vertraut, kann diese Kontrolle durch Verifizierung einer digitalen Signatur über den Quelltexten vorgenommen werden.

6. Zusammenfassung und Ausblick

In dieser Arbeit haben wir uns mit Formen der Anonymität und Werkzeugen zu ihrer Wahrung, insbesondere den sogenannten DC-Netzen von D. Chaum [23], auseinandergesetzt. Motivation für die umfangreiche Bearbeitung war eine Studie von M. Wright et al., welche diverse anonymisierende Techniken gegenüberstellt und deren Sicherheit in bestimmten Angriffsszenarien analysiert. Dabei taucht im Abschnitt 7.2 [30] die interessante Frage auf, welchen Aufwand/Kosten vollverbundene DC-Netze verursachen. Unser Untersuchungsgegenstand war also die Nachrichten- bzw. Kommunikationskomplexität dieser Verfahren und eine abschließende Bewertung vom praktischen Standpunkt aus.

Die Einleitung hatte hierbei die wichtige Aufgabe, den Begriff der *Anonymität* zu erklären und ihre Bedeutung in einer immer stärker digitalisierten Welt hervorzuheben. Das zweite Kapitel führte zunächst allgemein in das Gebiet der Kryptographie ein und behandelte dann zwei Lösungsmöglichkeiten zur Wahrung der Anonymität in Rechnernetzen. Wir fokusierten dabei den DC-Netz-Ansatz, welcher allerdings noch weitere komplizierte Verfahren (z. B. Schwellenwertsignatur- und Schwellenwertmünzwurfschema) für seine Realisierung benötigte. Nach der umfassenden Diskussion ihrer Sicherheit sollte dem Leser im dritten Abschnitt das Problem der zuverlässigen Verteilung erläutert und Wege zur Gewährleistung von Eigenschaften wie Konsistenz aufgezeigt werden. Unser Hauptteil behandelte dann das eigentliche DC-Netz-Protokoll, wobei ein besonderes Interesse seiner Robustheit, d. h. Erhaltung der Funktionsfähigkeit im Angriffsfall, galt.

Aufbauend auf den besprochenen Verfahren und Protokollen wurde der Prototyp `GnuDCN` entwickelt, mit dem wir schließlich die Messungen und Abschätzungen des fünften Kapitels durchführten. Dabei bestätigte sich natürlich die Vermutung, daß robuste DC-Netze einen enormen Nachrichten- und Kommunikationsaufwand in der Größenordnung $O(n^5)$ verursachen. Eine allgemeine Praktikabilität ist damit leider nicht gegeben, jedoch kann die Verwendung im kleinen Teilnehmerkreis durchaus Sinn haben. Anhand von zwei kurzen Beispielen wurde auf diese Möglichkeit hingewiesen, wobei insbesonders die Kombination mit MIX-Netzen ein vielversprechendes Einsatzgebiet eröffnet. Meiner Meinung nach sollten hier weitere Untersuchungen und Forschungsarbeiten ansetzten.

Ein zentrales Problem ist dabei die statische Konfiguration der Teilnehmer durch den vertrauenswürdigen Vermittler. Steht er nicht zur Verfügung, kann unter Umständen die gemeinsame Erzeugung auf vertrauenswürdiger Hardware weiterhelfen. Insgesamt ist diese Lösung jedoch kaum befriedigend, besonders wenn die beteiligten Parteien physisch nicht zusammentreffen können. Seit kurzer Zeit existieren jedoch sichere Protokolle [64, 65, 66], um derartige Probleme über gemeinsame Mehrparteienberechnung [67, 68] zu lösen. Deren Integration könnte den Verwendungsbereich von `GnuDCN` wesentlich vergrößern.

A. Schnittstellen kryptographischer Verfahren

A.1. ShoupRsaDTSS-Klasse

Diese Klasse implementiert das RSA-ähnliche $/n, k, t/$-Schwellenwertsignaturschema von V. Shoup [36], welches wir im Abschnitt 2.6.1 besprochen haben. Dabei wurde nur das verbesserte $k \geq t+1$-Verfahren realisiert. Zusätzlich stehen aber zwei Methoden zur Verwendung als $/n, k, t/$-Schwellenwertmünzwurfschema bereit.

Typen:

```
typedef    unsigned long int    mpz_ui
```

Kon- und Destruktoren:

```
ShoupRsaDTSS
   (mpz_ui l, mpz_ui k, mpz_ui t, mpz_ui size)
```

Erzeugt ein /`l`, `k`, `t`/-Schwellenwertsignatur- bzw. Schwellenwertmünzwurfschema mit genau `size`-Bit langen Primfaktoren (Sicherheitsparameter). Diesen Konstruktor ruft unser vertrauenswürdiger Vermittler während der Konfigurationserstellung auf.

```
ShoupRsaDTSS
   (string secret_block)
```

Initalisiert ein Schwellenwertsignatur- bzw. Schwellenwertmünzwurfschema mit den geheimen Schlüsselinformationen aus der Zeichenkette `secret_block`. Dieser Konstruktor wird also von allen Subjekten verwendet, um das vom Vermittler vorkonfigurierte Schema schließlich einzurichten.

```
~ShoupRsaDTSS
   ()
```

Gibt benutzte Ressourcen und den Speicher der Klasseninstanz wieder frei.

Methoden:

```
const char *secret_block
   (mpz_ui share_idx)
```

Liefert die geheimen Schlüsselinformationen des `share_idx`-ten Subjekts als nullterminierte Zeichenkette, falls das Schema erzeugt und nicht initalisiert wurde. Es handelt sich hierbei also nur um eine Exportfunktion für den vertrauenswürdigen Vermittler, welcher die Daten dann den jeweiligen Subjekten übergibt.

```
unsigned long identify_owner
   () const
```

Liefert das vom Vermittler zugeordnete Kennzeichen des aufrufenden Subjekts. Diese eindeutige Zahl soll im folgenden mit i bezeichnet werden.

```
const char *generate_share
   (const char *msg, size_t msg_length)
```

Teilsignaturerzeugungsalgorithmus: Erstellt eine i-Teilunterschrift über der `msg_length` Zeichen langen Nachricht `msg` und liefert diese als nullterminierte Zeichenkette zurück.

```
unsigned long identify_share
   (string input) const
```

Liefert das vom Vermittler zugeordnete Kennzeichen desjenigen Subjekts, welches die Teilunterschrift `input` erzeugt hat.

```
bool validate_share
   (const char *msg, size_t msg_length, string input) const
```

Teilsignaturverifikationsalgorithmus: Prüft die Gültigkeit der Teilunterschrift `input` über der `msg_length` Zeichen langen Nachricht `msg` und liefert im positiven Fall den Wert `true` zurück.

```
const char *combine_shares_to_signature
   (const char *msg, size_t msg_length, vector<string> input)
```

Teilsignaturkombinationsalgorithmus: Erzeugt aus k gültigen Teilunterschriften verschiedener Subjekte eine Gesamtsignatur über der `msg_length` Zeichen langen Nachricht `msg` und liefert sie als nullterminierte Zeichenkette zurück. Die Teilunterschriften werden im Vektor `input` übergeben und müssen nicht geordnet sein.

```
bool validate_signature
   (const char *msg, size_t msg_length, string input) const
```

Gesamtsignaturverifikationsalgorithmus: Prüft die Gültigkeit der Gesamtunterschrift `input` über der `msg_length` Zeichen langen Nachricht `msg` und liefert im positiven Fall den Wert `true` zurück.

```
const char *signature_to_coin
   (string input)
```

Wandelt eine Gesamtunterschrift `input` über der Nachricht C in einen Münzwert $F(C)$ um und gibt denselben als nullterminierte Zeichenkette zurück.

```
bool last_bit_of_coin
   (string input) const
```

Ermittelt das letzte Bit des Münzwertes `input` — also die sogenannte Seite der Münze. Mit ähnlichen Funktionen können aus einem Münzwert bis zu 160 unabhängige Bits gewonnen werden, falls die kryptographische Hashfunktion eine solche Länge erlaubt.

A.2. SchneierHashBCS-Klasse

Diese Klasse implementiert das sichere Festlegeschema [8] aus Abschnitt 2.8.1.

Typen:

```
typedef    unsigned long int    mpz_ui
```

Attribute:

```
pair<string, string>    commitment
```

Enthält die von Methode `define_commitment(...)` erstellte Fest- und Offenlegung.

Kon- und Destruktoren:

```
SchneierHashBCS
   (mpz_ui size)
```

Initalisiert das Festlegeschema mit dem Sicherheitsparameter `size`, welcher die Länge der Zufallszeichenketten R_1 und R_2 in Bit spezifiziert.

```
~SchneierHashBCS
   ()
```

Gibt benutzte Ressourcen und den Speicher der Klasseninstanz wieder frei.

Methoden:

```
void define_commitment
   (const char *msg, size_t msg_length)
```

Erstellt eine Fest- und Offenlegung über der `msg_length` Zeichen langen Nachricht `msg` und speichert sie im Attribut `commitment` ab.

```
bool verify_commitment
   (const char *msg, size_t msg_length, string input1, string input2)
```

Überprüft die Festlegung `input1` sowie die Offenlegung `input2` über der `msg_length` Zeichen langen Nachricht `msg` und liefert im positiven Fall den Wert `true` zurück.

A.3. BlumBlumShubPRBG-Klasse

Diese Klasse implementiert den kryptographisch sicheren Pseudozufallsbitgenerator von L. Blum, M. Blum und M. Shub [41] aus Abschnitt 2.8.2, dessen Unvorhersagbarkeit auf dem schwierigen Problem der Faktorisierung seines Moduls n beruht. Die Ein- und Ausgabe von durch Zeichenketten (`string` oder `const char*`) repräsentierten Zahlen erfolgt jeweils in kanonischer Darstellung zur Basis `mpz_io_base = 36`.

Typen:

```
typedef    unsigned long int    mpz_ui
```

Kon- und Destruktoren:

```
BlumBlumShubPRBG
  (mpz_ui size)
```

Erzeugt einen neuen Pseudozufallsbitgenerator mit dem Sicherheitsparameter `size`, der die Länge verwendeter Primfaktoren in Bit angibt. Diesen Konstruktor ruft unser vertrauenswürdiger Vermittler während der Konfigurationserstellung auf.

```
BlumBlumShubPRBG
  (string p_str, string q_str)
```

Initalisiert den Pseudozufallsbitgenerator mit dem Faktor p (`p_str`) und q (`q_str`), falls beide Primzahlen intern bekannt sind.

```
BlumBlumShubPRBG
  (string n_str)
```

Initalisiert den Pseudozufallsbitgenerator mit dem Produkt $n := p \cdot q$ (`n_str`). Dieser Konstruktor wird von allen Subjekten verwendet, um den vom Vermittler vorkonfigurierten Generator einzurichten.

```
~BlumBlumShubPRBG
  ()
```

Gibt benutzte Ressourcen und den Speicher der Klasseninstanz wieder frei.

Methoden:

```
const char *get_n
  ()
```

Liefert das Produkt $n := p \cdot q$ als nullterminierte Zeichenkette zurück.

```
const char *get_x_i
  ()
```

Liefert den Zustandswert x_i als nullterminierte Zeichenkette zurück.

```
const char *get_p
  ()
```

Liefert den Primfaktor p als nullterminierte Zeichenkette zurück (falls intern bekannt).

```
const char *get_q
  ()
```

Liefert den Primfaktor q als nullterminierte Zeichenkette zurück (falls intern bekannt).

```
void set_seed
  ()
```

Belegt den Zustandswert x_i des Generators mit einer Zufallszahl.

```
void set_seed
  (string x_0_str)
```

Setzt den Wert x_i durch einen Anfangszustand x_0. Dabei wird die von `x_0_str` repräsentierte Zahl solange inkrementiert oder modular reduziert, bis $x_0 \in \mathbb{Z}_n^*$ gilt.

```
bool random_bit
  ()
```

Iteriert den Generator ($x_i := x_i^2 \bmod n$) und liefert das niederwertigste Bit von x_i als Pseudozufall (`true`, `false`) zurück.

```
char random_char
  ()
```

Iteriert den Generator ($x_i := x_i^2 \bmod n$) und liefert die niederwertigsten 8 Bit von x_i als Pseudozufallszeichen zurück. Dazu muß der Zustandswert x_i mindestens $2^8 = 256$ Bit lang sein, um die Sicherheit (Unvorhersagbarkeit) nicht zu gefährden. [8]

A.4. CachinKursaweShoupABBA-Klasse

Diese Klasse implementiert das probabilistische ABBA-Protokoll (Asynchronous Binary Byzantine Agreement) der Autoren C. Cachin, K. Kursawe und V. Shoup [37], welches im Abschnitt 3.2.1 Gegenstand unserer Betrachtung war. Als Verbesserungen wurden sowohl die *starke Richtigkeit* als auch die *starke Termination* eingearbeitet.

Kon- und Destruktoren:

```
CachinKursaweShoupABBA
   (string tid, int sid, bool init_decide, unsigned long n,
   unsigned long t, ShoupRsaDTSS* dtss_a, ShoupRsaDTSS* dtss_p,
   unsigned long d, bool b)
```

Initalisiert eine ABBA-Instanz mit dem Transaktionskennzeichen `tid` und der Sitzungsnummer `sid`. Der Binärwert `init_decide` repräsentiert den vorgegebenen Anfangszustand $V_i \in \{0, 1\}$, während `n` die Anzahl der Teilnehmer n und `t` die Zahl t (bösartige/ fehlerhafte Parteien) darstellt. Damit das ABBA-Protokoll robust funktioniert, muß die Bedingung $t < {}^{n}/_{3}$ gelten. Weiterhin werden ein initalisiertes $/n, n - t, t/$-Schwellenwertsignaturschema `dtss_a` sowie $/n, t + 1, t/$-Schwellenwertsignaturschema `dtss_p` benötigt. Im Wert `d` übergibt man die Anzahl der `run(...)`-Aufrufe bis zum Start des Protokolls (Zeitverzögerung) und in `b` eine Preferenz (Bias) für die Vorauswahl; i. A. wird `d = 0, b = init_decide` gesetzt.

```
~CachinKursaweShoupABBA
   ()
```

Gibt benutzte Ressourcen und den Speicher der Klasseninstanz wieder frei.

Methoden:

```
bool run
   ()
```

Diese Methode muß für den Fortschritt des Protokolls regelmäßig aufgerufen werden (kooperative Nebenläufigkeit). Bei Termination wird `true` zurückgegeben.

```
bool decide
   () const
```

Liefert den Übereinstimmungswert $U_i \in \{0, 1\}$ zurück, sofern das ABBA-Protokoll ordentlich terminiert `run() == true` ist.

```
void in
   (string in_data)
```

Übergibt eine Nachricht `in_data` an die interne Eingabewarteschlange.

```
const char *out
   ()
```

Liefert eine nullterminierte Nachricht von der internen Ausgabewarteschlange zurück,

welche über das Netzwerk an alle Parteien verteilt werden muß. Ist keine auszuliefernde Mitteilung vorhanden, wird einfach `NULL` zurückgegeben.

```
const char *black
   ()
```

Liefert die vom Vermittler zugeordneten Kennzeichen derjenigen Subjekte, welche nachvollziehbare Fehler (z. B. ungültige Signatur) im Protokoll ausgelöst haben. Diese Parteien können dann mit einem Ausschlußverfahren entfernt werden. Ist kein Kennzeichen vorhanden, wird `NULL` zurückgegeben.

```
const char *red
   ()
```

Liefert die vom Vermittler zugeordneten Kennzeichen derjenigen Subjekte, welche nicht kausale Handlungen (z. B. Nachricht zum falschen Zeitpunkt) im Protokoll ausgelöst haben. Diese Parteien können dann für eine bestimmte Zeit gesperrt werden, damit ‚lahm legende' Angriffe (Denial of Service) nicht die Robustheit gefährden. Ist kein Kennzeichen vorhanden, wird `NULL` zurückgegeben.

Abbildungsverzeichnis

1. Prinzip der asymmetrischen Verschlüsselung 25
2. Prinzip der hybriden Verschlüsselung 26
3. ‚Man-in-the-middle'-Angriff 26
4. hierarchische Zertifizierungsstruktur 27
5. Aufbau eines *Netz des Vertrauens* 28
6. Prinzip der digitalen Unterschrift 33
7. Verarbeitungsschritte eines MIX-Knotens 35
8. Indirektionspfad durch eine MIX-Kaskade 36
9. klassischer DC-Ring 39
10. vollverbundenes n-zu-n DC-Netz 40
11. Zustandsdiagramm des Verteilungsprotokolls 79
12. vollständiger Schlüsselgraph eines DC-Netzes mit $n = 4$ 81
13. Phasen des verbesserten Fallenprotokolls 89
14. Aufbau und Schichten von `GnuDCN` 96
15. Abhängigkeiten zwischen Modulen, Klassen und Bibliotheken . . . 100
16. normale Nachrichtenkomplexität von `GnuDCN` 102
17. normale Kommunikationskomplexität von `GnuDCN` 102
18. erhöhte Nachrichtenkomplexität von `GnuDCN` 103

Tabellenverzeichnis

Beziehungen zwischen Anonymitätsbegriffen 14

Bezeichnungen und Beispiele zur O-Notation 20

kryptographische Annahmen 24

theoretisches Nachrichtenaufkommen der Phasen 92

normale Nachrichten- und Kommunikationskomplexiät von `GnuDCN` . . 101

erhöhte Nachrichtenkomplexiät von `GnuDCN` 103

Literatur

[1] *Webster's Revised Unabridged Dictionary*, C. & G. Merriam Co., 1913
Elektronische Version, DICT Development Group, `http://www.dict.org/`

[2] H. Bäumler: *„Unser Recht auf Anonymität“*
Eröffnungsrede zur Sommerakademie 2002 des Unabhängigen Landeszentrums für Datenschutz, Schleswig-Holstein

[3] H.-G. Marcieniec: *Anonymität, Schule und Demokratie*
Beitrag in einer Schülerzeitung, Alsfeld 1965, `http://www.marcieniec.de/`

[4] A. Pfitzmann, M. Köhntopp, A. Shostack: *Anonymity, Unobservability, and Pseudonymity – A Proposal for Terminology*, Draft v0.12
Designing Privacy Enhancing Technologies, LNCS 2009, pp. 1 ff.

[5] C. Schulzki-Haddouti (Hrsg.):
Vom Ende der Anonymität – Die Globalisierung der Überwachung,
Verlag Heinz Heise, 2. aktualisierte Auflage, 2001, ISBN 3-88229-192-3

[6] J. Bamford: *NSA – Die Anatomie des mächtigsten Geheimdienstes der Welt*,
Goldmann-Verlag, 2001, ISBN 3-442-15151-1

[7] R. Höhndorf: *Echelon Vortrag*
Vortragsreihe des GAOS e. V., 2001, `http://gaos.org/vortrag/echelon/`

[8] B. Schneier: *Angewandte Kryptographie*
Addison-Wesley, 1996, ISBN 3-89319-854-7

[9] P. Gentsch: *Tante-Emma-Prinzip*
IX Magazin für professionelle Informationstechnik, 12/2002, ISSN 0935-9680

[10] M. Kanther: *Mit Sicherheit in die Informationsgesellschaft*
Eröffnungsrede zum 5. IT-Sicherheitskongress, 28. April 1997

[11] H. T. Siegelmann: *Neural Networks and Analog Computation*,
Birkhäuser Verlag, 2002, ISBN 0-8176-3949-7

[12] E. Bresson, O. Chevassut, D. Pointcheval: *Group Diffie-Hellman Problems*
Proceeding of the Workshop on Selected Areas in Cryptography (SAC'02),
LNCS 2595, Springer-Verlag, pp. 325-338, 2002

[13] D. Boneh: *The Decision Diffie-Hellman Problem*
Proceedings of the Third Algorithmic Number Theory Symposium,
LNCS 1423, Springer-Verlag, pp. 48–63, 1998

[14] R. Rivest, A. Shamir, L. Adleman:
A Method for Obtaining Digital Signatures and Public-Key Cryptosystems,
Communications of the ACM, Vol. 21 (2), pp. 120–126, 1978

[15] J. Apel: *Kryptographie*, Vorlesungsskript WS1998/99, Universität Leipzig

[16] A. Salomaa: *Public-Key Cryptography*
EATCS Monographs on Theoretical CS (Eds.: Brauer, Rozenberg, Salomaa), Springer Verlag, 1990, ISBN 3-540-52831-8

[17] A. Menezes, P. van Oorschot, S. Vanstone: *Handbook of Applied Cryptography*, CRC Press, 1996, `http://www.cacr.math.uwaterloo.ca/hac/`

[18] J. Pieprzyk, B. Sadeghiyan: *Design of Hashing Algorithms*
LNCS 756, Springer-Verlag, 1993

[19] S. Bakhtiari, R. Safavi-Naini, J. Pieprzyk: *Cryptographic Hash Functions: A Survey*, Technical Report 95-09, University of Wollongong, 1995

[20] H. Dobbertin, A. Bosselaers, B. Preneel:
RIPEMD-160, a strengthened version of RIPEMD,
Fast Software Encryption, LNCS 1039, Springer-Verlag, pp. 71–82, 1996

[21] D. Chaum: *Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms*, Communications of the ACM, Vol. 24 (2), pp. 84–88, 1981

[22] D. Chaum: *Security without identification: Transaction systems to make big brother obsolete*, Comm. of the ACM, Vol. 28 (10), pp. 1030-1044, 1985

[23] D. Chaum: *The Dining Cryptographers Problem: Unconditional Sender and Recipient Untracebility*, Journal of Cryptology, Vol. 1, pp. 65–75, 1988

[24] `JAP` – Anonymity & Privacy, MIX-Anoymisierungsdienst der TU Dresden, `http://anon.inf.tu-dresden.de/`

[25] Selbstverpflichtungserklärung der `JAP`-Betreiber
`http://anon.inf.tu-dresden.de/Selbstverpflichtung.html`

[26] Technischer Hintergrund des `JAP`-Systems
`http://anon.inf.tu-dresden.de/JAPTechBgPaper.pdf`

[27] O. Berthold, H. Federrath, M. Köhntopp: *Project „Anonymity and Unobservability in the Internet“*, Proceedings of the Workshop on Freedom and Privacy 2000, ACM ISBN 1-58113-256-5, pp. 57–65, 2000

[28] E. Franz, A, Jerichow, A. Pfitzmann: *Systematisierung und Modellierung von Mixen*, Proc. Verläßliche IT-Systeme (VIS'97), DuD-Fachbeiträge, Vieweg

[29] M. Rennhard, S. Rafaeli, L. Mathy, B. Plattner, D. Hutchison:
An Architecture for an Anonymity Network
10th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE 2001), ISBN 0-7695-1269-0

[30] M. Wright, M. Adler, B. Levine, C. Shields: *An Analysis of the Degradation of Anonymous Protocols*, Proceedings of the ISOC Network and Distributed System Security Symposium (NDSS 2002), 2002

[31] J. Bos, B. den Boer: *Detection of Disrupters in the DC Protocol*, LNCS 434 (Eurocrypt '89), Springer-Verlag, 1989

[32] M. Waidner, B. Pfitzmann: *The Dining Cryptographers in the Disco: Unconditional Sender and Recipient Untraceability with Computationally Secure Serviceability*, LNCS 434 (Eurocrypt '89), Springer-Verlag, 1989

[33] D. Martin: *Local Anonymity in the Internet*, Dissertation (PhD), Boston University, School of Arts and Sciences, 1999

[34] J. Bos: *Practical Privacy* Dissertation (PhD), Technische Universität Eindhoven, 1992

[35] A. Shamir: *How to share a secret* Communications of the ACM, Vol. 22, pp. 612–613, 1979

[36] V. Shoup: *Practical Threshold Signatures* LNCS 1807 (Eurocrypt 2000), Springer-Verlag, 2000

[37] C. Cachin, K. Kursawe, V. Shoup: *Random Oracles in Constantinople: Practical Asynchronous Byzantine Agreement using Cryptography*, 19th ACM Symposium on Principles of Distributed Computing (PODC), 2000

[38] R. Cramer, V. Shoup: *Signature Schemes Based on the Strong RSA Assumption*, ACM Transactions on Information and System Security (TISSEC), Vol. 3 (3), pp. 161–185, 2000

[39] D. Chaum, T Pedersen: *Wallet Databases with Observers* Advances in Cryptology (CRYPTO '92), pp. 89–105, 1992

[40] R. Gennaro, V. Shoup: *Securing Threshold Cryptosystems against Chosen Ciphertext Attack*, Journal of Cryptology, Vol. 15, pp. 75–96, 2002

[41] L. Blum, M. Blum, M. Shub: *A Simple Unpredictable Pseudo-Random Number Generator*, SIAM Journal on Computing, Vol. 15 (2), pp. 364–383, 1986

[42] U. V. Vazirani, V. V. Vazirani: *Efficient and secure pseudo-random number generation*, Advances in Cryptology (CRYPTO'84), LNCS 196, Springer-Verlag, pp. 193–201, 1985

[43] M. J. Fischer: *The Consensus Problem in Unreliable Distributed Systems*, LNCS 158 (Fds. Computation Theory), pp. 127–140, Springer-Verlag, 1983

[44] L. Lamport, R. Shostak, M. Pease: *The Byzantine Generals Problem*, ACM Trans. on Prog. Languages and Systems, Vol. 4 (3), pp. 382–401, 1982

[45] D. Dolev, H. R. Strong: *Authenticated Algorithms for Byzantine Agreement*, SIAM Journal of Computation, Vol. 12 (4), pp. 656–666, 1983

[46] M. J. Fischer, N. A. Lynch, M. S. Paterson: *Impossibility of distributed consensus with one faulty process*, Proc. Second ACM Symposium on Principles of Database Systems, 1983 Journal of the ACM, Vol. 32 (2), pp. 374-382, 1985

[47] M. Ben-Or: *Another advantage of free choice: Completely asynchronous agreement protocols*, Proc. 2nd ACM Symposium on Principles of Distributed Computing, 1983

[48] R. Canetti, T. Rabin: *Fast asynchronous Byzantine Agreement with optimal resilience*, Proc. 25th ACM Symp. on Theory of Computing, pp. 42-51, 1993

[49] M. Kwiatkowska, G. Norman: *Automated Verification of a Randomized Byzantine Agreement Protocol*, Proc. Formal Techniques for Networked and Distributed Systems (FORTE'02), LNCS 2529, Springer-Verlag, pp. 194–209, 2002

[50] Probabilistic Symbolic Model Checker: „Byzantine Agreement Protocol“ `http://www.cs.bham.ac.uk/~dxp/prism/byzantine/index.html`

[51] C. Cachin: *Distributing Trust on the Internet*, Conference on dependable systems and networks (DSN-2001), 2001

[52] C. Cachin, K. Kursawe, F. Petzold, V. Shoup: *Secure and Efficient Asynchronous Broadcast Protocols*, Proc. CRYPTO 2001, pp. 524–541, 2001

[53] M. K. Reiter: *Reliable and Atomic Group Multicast in Rampart*, Proceedings 2nd ACM Conference on Computer and Communications, 1994

[54] M. Waidner: *Unconditional Sender and Recipient Untraceability in spite of Active Attacks*, LNCS 434 (Eurocrypt '89), Springer-Verlag, 1989

[55] D. W. Davies, D. L. A. Barber, W. L. Price, C. M. Solomides: *Computer Networks and their Protocols*, John Wiley and Sons, 1979

[56] A. S. Tanenbaum: *Computer Networks*, Prentice-Hall, 1988

[57] J. Bos, B. den Boer: *Detection of Disrupters in the DC Protocol*, LNCS 434 (Eurocrypt '89), Springer-Verlag, 1989

[58] R. van der Meyden, K. Su: *Symbolic Model Checking the Knowledge of the Dining Cryptographers*, erscheint 2003

[59] B. Schneier: *Secrets & Lies – IT-Sicherheit in einer vernetzten Welt* dpunkt.verlag/Wiley-VCH, 2001, ISBN 3-89864-113-9

[60] Free Software Foundation: *GNU General Public License (GPL)*
`http://www.gnu.org/copyleft/`

[61] O'Reillys Taschenbibliothek: *Open Source*, 1999, ISBN 3-89721-222-6

[62] K. Köhntopp, M. Köhntopp, A. Pfitzmann: *Sicherheit durch Open Source?* Datenschutz und Datensicherheit (DuD) 24/9, Vieweg-Verlag, 2000

[63] Bundesministerium für Wirtschaft und Technologie: *Alternative Betriebssysteme*, Leitfaden für kleine und mittlere Unternehmen, BMWi, 2001

[64] I. Damgård, M. Koprowski: *Practical threshold RSA signatures without a trusted dealer*, LNCS 2045 (EUROCRYPT 2001), Springer-Verlag, 2001

[65] P.-A. Fouque, J. Stern: *Fully distributed threshold RSA under standard assumptions*, LNCS 2248 (ASIACRYPT 2001), Springer-Verlag, 2001

[66] J. Algesheimer, J. Camenisch, V. Shoup: *Efficient Computation Modulo a Shared Secret with Application to the Generation of Shared Safe-Prime Products*, LNCS 2442 (CRYPTO 2002), Springer-Verlag, 2002

[67] O. Goldreich: *Secure Multi-Party Computation*,
`http://www.wisdom.weizmann.ac.il/~oded/pp.html`

[68] D. Boneh, M. Franklin: *Efficient generation of shared RSA keys*, LNCS 1233 (CRYPTO'97), Springer-Verlag, 1997

Software

[A] GNU MP Library (`libgmp`) Version $\geq$ 4.1:
Rechenoperationen für beliebig lange/genaue Zahlen
`http://swox.com/gmp/`

[B] GNU Crypto Library (`libgcrypt`) Version $\geq$ 1.1.10:
kryptographische Grundfunktionen (Pseudozufallszahlen, RMD160)
`ftp://ftp.gnupg.org/gcrypt/alpha/libgcrypt/`

[C] AT&T Graph Visualizations Tools (`libgraph`) Version $\geq$ 1.8.5:
Operationen auf Graphen, Konfiguration, Darstellung
`http://www.graphviz.org/`
`http://www.research.att.com/sw/tools/graphviz/license/`

[D] GNU Readline Library (`libreadline`) Version $\geq$ 4.1:
Zeileneingabehilfe
`http://www.gnu.org/directory/readline.html`

Selbständigkeitserklärung

Ich versichere, daß ich die vorliegende Arbeit selbständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel angefertigt habe.

Leipzig, den 11. März 2003 .....

Heiko Stamer