

# Analyse des OpenPGP Web of Trust

Studienarbeit  
von

**Alexander Ulrich**

Lehrstuhl für Rechnernetze und Internet  
Wilhelm-Schickard-Institut für Informatik  
Fakultät für Informations- und Kognitionswissenschaften  
Universität Tübingen

Erstgutachter: Prof. Dr. Peter Hauck  
Betreuer Mitarbeiter: Dipl.-Inform. Ralph Holz

Bearbeitungszeit: 08. Monat 2009 – 01. Monat 2010



---

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Tübingen, den ?. ???? 200?



# Inhaltsverzeichnis

<b>1 Einleitung</b>	<b>1</b>
1.1 Zielsetzung der Arbeit . . . . .	1
1.2 Gliederung der Arbeit . . . . .	1
<b>2 Grundlagen</b>	<b>3</b>
2.1 Kryptographie mit öffentlichen Schlüsseln . . . . .	3
2.1.1 Ziele von Kryptographie . . . . .	3
2.1.2 Prinzip . . . . .	3
2.1.3 Authentisierung von Schlüsseln . . . . .	5
2.1.3.1 Zentrale PKI (besser: hierarchisch) . . . . .	6
2.1.4 Kryptographische Hashfunktionen . . . . .	6
2.2 PGP/GnuPG . . . . .	6
2.2.1 Web of Trust . . . . .	7
2.2.2 Struktur von OpenPGP-Paketen . . . . .	8
2.2.3 Das GnuPG-Vertrauensmodell . . . . .	9
2.2.4 Das Keyserver-Netzwerk . . . . .	12
2.2.5 Zustandekommen von Signaturen im Web of Trust . . . . .	13
2.2.5.1 Übliche Zertifizierungsprozeduren . . . . .	13
2.2.5.2 Certificate Authorities im Web of Trust . . . . .	14
2.2.5.3 Keysigning-Parties . . . . .	15
2.2.5.4 Foobar FIXME . . . . .	15
2.3 Graphentheorie und Netzwerkanalyse . . . . .	16
2.3.1 Graphentheorie allgemein . . . . .	16
2.3.2 Netzwerkstatistiken . . . . .	17
2.3.3 Netzwerkmodelle . . . . .	18
2.3.4 Soziale Netzwerke . . . . .	18
2.3.5 Communities . . . . .	19
2.3.5.1 Grundlegende Begriffe . . . . .	19
2.3.5.2 Modularity . . . . .	20
2.3.5.3 Weitere Algorithmen . . . . .	20
2.4 Verwandte Arbeiten . . . . .	20
2.4.1 Analysen des OpenPGP-Web of Trust . . . . .	20
<b>3 Methoden und Materialien</b>	<b>23</b>
3.1 Warum eigene Software? . . . . .	23
3.2 Design und Implementierung . . . . .	25
3.2.1 Der SKS Keyserver . . . . .	25
3.2.2 Eigene Software . . . . .	25
3.2.2.1 Datenextraktion . . . . .	26

3.2.2.2	Datenauswertung . . . . .	28
3.3	Community-Analyse . . . . .	29
3.3.1	Communities in gerichteten Graphen . . . . .	32
<b>4</b>	<b>Ergebnisse und Diskussion</b>	<b>33</b>
4.1	Datensatz . . . . .	33
4.2	Allgemeine Merkmale des Netzwerkes . . . . .	34
4.2.1	Starke Zusammenhangskomponenten . . . . .	34
4.2.2	Netzwerkstatistiken der grössten starken Zusammenhangskomponente . . . . .	36
4.2.2.1	Gegenseitigkeit von Kanten . . . . .	36
4.2.2.2	Clustering und Small-World . . . . .	37
4.2.2.3	Verteilung der Grade, Skalenfreiheit . . . . .	38
4.2.3	Robustheit . . . . .	40
4.2.4	Nützlichkeit . . . . .	42
4.3	Eigenschaften einzelner Schlüssel, zeitliche Entwicklung . . . . .	45
4.3.1	Zeitliche Entwicklung . . . . .	45
4.3.2	Public-Key- und Hashalgorithmen . . . . .	46
4.4	Communities . . . . .	48
<b>5</b>	<b>Zusammenfassung und Ausblick</b>	<b>63</b>
<b>A</b>	<b>Schema der Tabellen</b>	<b>65</b>
<b>B</b>	<b>Analysewerkzeuge</b>	<b>67</b>
	<b>Abbildungsverzeichnis</b>	<b>69</b>
	<b>Literaturverzeichnis</b>	<b>71</b>

# 1. Einleitung

Hinweis: In die Einleitung gehört die Motivation und Einleitung in die Problemstellung. Die Problemstellung kann in der Analyse noch detaillierter beschrieben werden.

Bla fasel... foo

## 1.1 Zielsetzung der Arbeit

Arbeit befasst sich nicht mit den zugrundeliegenden kryptographischen Verfahren, sondern mit denen Rahmenbedingungen, in denen diese bei GnuPG verwendet werden.

Was ist die Aufgabe der Arbeit?

Bla fasel...

## 1.2 Gliederung der Arbeit

Die Arbeit ist im weiteren wie folgt gegliedert: Kapitel 2 erläutert die notwendigen Grundlagen. Dies sind zum einen die Prinzipien asymmetrischer Kryptographie (Abschnitt 2.1), Methoden zur Authentifizierung öffentlicher Schlüssel, die Weiterhin werden in Abschnitt 2.2 die Prinzipien von PGP und GnuPG sowie die Rahmenbedingungen, in denen PGP/GnuPG verwendet wird, beschrieben. Außerdem werden in 2.3 die graphentheoretischen und netzwerkanalytischen Verfahren eingeführt, die in dieser Arbeit verwendet werden. Abschnitt 2.4 bespricht verwandte Arbeiten.

Kapitel 3 beschreibt die verwendeten Methoden. Abschnitt 3.1 begründet zunächst, warum die Extraktion der benötigen Daten selbst implementiert wurde. Abschnitt 3.2 beschreibt die im Rahmen dieser Arbeit implementierte Software zur Extraktion und Analyse des Web of Trust, während Abschnitt 3.3 die Vorgehensweise für die Analyse der Community-Struktur in Abschnitt 4.4 erläutert.

Kapitel 4 stellt die erhaltenen Ergebnisse vor und diskutiert sie. Dies gliedert sich in Ergebnisse über die Struktur des Web of Trust (Abschnitt 4.2), einige Statistiken

über Eigenschaften einzelner Schlüssel und die zeitliche Entwicklung des Web of Trust (Abschnitt 4.3) sowie die Analyse der Community-Struktur (Abschnitt 4.4).

Kapitel 5 schliesslich fasst die Ergebnisse zusammen und bietet einen Ausblick auf Ansatzpunkte für zukünftige Arbeiten.

## 2. Grundlagen

### 2.1 Kryptographie mit öffentlichen Schlüsseln

Dieser Abschnitt erläutert zunächst die Prinzipien symmetrischer und asymmetrischer Kryptographie, geht dann auf das Problem der Schlüsselverteilung bei asymmetrischer Verschlüsselung ein und führt zu dessen Lösung die Konzepte Public Key Infrastructure (PKI) und Web of Trust (WoT) ein. Die Darstellung beruht auf [MeOV96].

#### 2.1.1 Ziele von Kryptographie

Wir betrachten den Fall einer Nachricht, die über einen Kanal transportiert wird. Im Kontext von Computernetzwerken, insbesondere des Internets, muss im Allgemeinen davon ausgegangen werden, dass unberechtigte Dritte in der Lage sind, Nachrichten auf einem Kanal abzuhören, zurückzuhalten und zu verändern. Menezes et al. [MeOV96] definieren essentielle Ziele von Informationssicherheit, die in diesem Fall relevant sind:

**Vertraulichkeit** Es soll sichergestellt werden, dass der Inhalt einer Nachricht nur berechtigten Kommunikationspartnern zugänglich ist.

**Authentizität** Eine Nachricht ist dann authentisch, wenn sie zweifelsfrei ihrem Absender zugeordnet werden kann. Authentifizierung betrifft also die Verifizierung des Ursprungs oder des Urhebers einer Nachricht. Dieses Problem betrifft nicht nur die Nachricht selbst, sondern auch Schlüsselmaterial.

**Integrität** Es soll sichergestellt werden, dass eine Nachricht während des Transportes nicht verändert wurde, bzw. dass eine Veränderung nicht unbemerkt möglich ist.

#### 2.1.2 Prinzip

*Symmetrische Kryptographie* basiert auf einem *Geheimnis* oder *Schlüssel*, der sowohl dem Absender als auch dem Empfänger bekannt ist. Der gleiche Schlüssel

wird dazu benutzt, eine Nachricht zu verschlüsseln und sie zu entschlüsseln. Diese Vorgänge können von jedem durchgeführt werden, der im Besitz des Schlüssels ist. Die Teilnehmer müssen also sicherstellen, dass keine unberechtigten Dritten in den Besitz des Schlüssels gelangen können. Ein Schlüssel kann natürlich bei einem direkten Treffen ausgetauscht werden. Die Verteilung oder Vereinbarung eines Schlüssels zwischen den Kommunikationspartnern ist aber dann schwierig, wenn ein direkter Austausch beispielsweise aufgrund von geographischer Entfernung nicht möglich ist und zwischen den Kommunikationspartnern noch kein sicherer Kanal<sup>1</sup> existiert. Dieses Problem wird als *Schlüsselverteilungsproblem* bezeichnet[MeOV96]. Außerdem stellt sich das Problem, dass für jede Menge von Kommunikationspartnern ein unterschiedlicher Schlüssel benötigt wird. Wenn davon ausgegangen wird, dass nur jeweils zwei Kommunikationspartner miteinander kommunizieren, wächst die Anzahl der benötigten Schlüssel also quadratisch mit der Anzahl der Teilnehmer.

Eine Methode, um das Problem der Schlüsselverteilung über einen unsicheren Kanal zu lösen, bietet *asymmetrische Kryptographie* oder *Public-Key-Kryptographie*. Dabei gibt es keinen einzelnen Schlüssel, über den alle Kommunikationspartner verfügen. Stattdessen verfügt jeder Teilnehmer über ein *Schlüsselpaar*, das aus einem *privatem* und einem *öffentlichen* Schlüssel besteht. Zwischen diesen beiden Schlüsseln muss zwar ein mathematischer Zusammenhang bestehen. Der private Schlüssel darf allerdings nicht mit akzeptablem Aufwand aus dem öffentlichen Schlüssel ableitbar sein. Das entscheidende Merkmal eines asymmetrischen Kryptographieschemas ist, dass der Schlüssel, mit dem eine Nachricht verschlüsselt wird, nicht mit dem Schlüssel identisch ist, mit dem sie wieder entschlüsselt werden kann.

Asymmetrische Kryptographie kann benutzt werden, um die in Abschnitt 2.1.1 definierten Ziele zu gewährleisten. Vertraulichkeit wird erreicht, indem der Absender einer Nachricht diese mit dem öffentlichen Schlüssel des Empfängers verschlüsselt. Diese Nachricht kann dann nur mit dem entsprechenden privaten Schlüssel des Empfängers entschlüsselt werden. Um Authentizität, Integrität und Nicht-Abstrebbarkeit zu gewährleisten, werden *digitale Signaturen* verwendet, die von einem Geheimnis (dem privaten Schlüssel) und der zu signierenden Nachricht abhängen. Ein digitales Signaturschema besteht aus einer Signaturverifizierungsfunktion und einer Signaturerzeugungsfunktion. Gegeben ein asymmetrisches Schlüsselpaar mit einem öffentlichen Schlüssel  $k$  und einem privaten Schlüssel  $k^{-1}$  sowie eine Nachricht  $m$  wird eine digitale Signatur  $d$  erzeugt, indem die Signaturerzeugungsfunktion auf  $k^{-1}$  und  $m$  angewandt wird. Um die Signatur  $d$  zur Nachricht  $m$  zu überprüfen, wird die Verifizierungsfunktion auf  $d$ ,  $m$  und den öffentlichen Schlüssel  $k$  angewandt. Sie gibt an, ob es sich um eine gültige Signatur bezüglich des zugehörigen privaten Schlüssels  $k^{-1}$  handelt oder nicht.

Der öffentliche Schlüssel kann öffentlich beliebig verfügbar gemacht und über einen unsicheren Kanal transportiert werden. Der private Schlüssel hingegen darf nur dem Besitzer bekannt sein, da nur mit ihm die Operationen Entschlüsselung und Erstellung von digitalen Signaturen möglich sind.

Asymmetrische Kryptographie löst das oben angesprochene Skalierungsproblem, da nicht mehr ein Schlüssel für jedes Paar von Kommunikationspartnern benötigt wird, sondern nur noch ein Schlüsselpaar für jeden Teilnehmer.

---

<sup>1</sup>ein sicherer Kanal bezeichnet einen Kanal, auf dem die oben definierten Eigenschaften Vertraulichkeit, Authentizität und Integrität gewährleistet sind

### 2.1.3 Authentisierung von Schlüsseln

Die Verteilung von öffentlichen Schlüsseln über unsichere Kanäle scheint zwar zunächst zwar zunächst das Schlüsselverteilungsproblem zu lösen. Öffentliche Schlüssel können über beliebige Wege verteilt werden, etwa per E-Mail, auf Webseiten oder auf Kesyservern (Abschnitt 2.2.4). Allerdings tritt dann ein weiteres Problem auf: Wie kann sichergestellt werden, dass ein öffentlicher Schlüssel authentisch ist, er also tatsächlich von dem vorgeblichen Besitzer stammt und dieser das Schlüsselmaterial kontrolliert? Die Lösung des Verteilungsproblems wird nur auf eine andere Ebene verschoben.

Sei angenommen, dass Benutzer  $B$  den öffentlichen Schlüssel von Benutzer  $A$  über einen unsicheren Kanal bezieht und dass keine Vorkehrungen getroffen werden, um die Authentizität des Schlüssels sicherzustellen. Ein aktiver Angreifer  $M$  könnte dann den echten Schlüssel von  $A$  abfangen und durch einen Schlüssel ersetzen, der zwar die Identität von  $A$  trägt, aber unter der Kontrolle von  $M$  steht.  $M$  wäre dann in der Lage, Nachrichten zu entschlüsseln, die von  $B$  mittels des vermeintlich  $A$  gehörenden Schlüssels verschlüsselt wurden.  $M$  könnte darüber hinaus solche Nachrichten mit dem echten Schlüssel von  $A$  verschlüsseln und sie weiterschicken. Auf diese Weise bliebe der Angriff unbemerkt. Ein solcher Angriff wird als Man-in-the-middle-Angriff bezeichnet.

Natürlich kann die Authentizität eines Schlüssels überprüft werden, wenn dieser bei einem persönlichen Treffen übergeben wird. Im Kontext elektronischer Kommunikation ist dies jedoch im Allgemeinen nicht möglich. Stattdessen muss sich dafür auf einen zuverlässigen Dritten (Trusted Third Party – TTP) verlassen werden, der die Authentizität des Schlüssels bestätigt. Eine TTP gibt eine Zusicherung über die Bindung von einem öffentlichen Schlüssel an eine Person oder allgemein eine Entität ab.

Diese Zusicherung hat die Form eines *Zertifikats*. Ein Zertifikat besteht aus einem Datenteil, der eine Identität (in Form eines Namens), den zugehörigen öffentlichen Schlüssel sowie eventuell zusätzliche Attribute enthält. Außerdem enthält ein Zertifikat einen Signaturteil. Dieser besteht aus einer digitalen Signatur über dem Daten teil, die mit dem Schlüssel der TTP erstellt wurde. Die Authentizität eines Zertifikats kann also überprüft werden, indem die Signatur auf dem Zertifikat mit dem öffentlichen Schlüssel der TTP (welcher natürlich selbst authentisch sein muss) überprüft wird.

Um sich auf die Zusicherungen der TTP verlassen zu können, muss ein Benutzer dieser *vertrauen*. Vertrauen bedeutet hier die Annahme, dass die TTP stets korrekte Zusicherungen abgibt. Um eine korrekte Zusicherung in Form eines Zertifikats für einen öffentlichen Schlüssel abgeben zu können, muss die TTP überprüfen, dass der Schlüssel tatsächlich der Person gehört, deren Identität auf dem Schlüssel vermerkt ist. Außerdem muss sichergestellt werden, dass diese Person auch im Besitz des entsprechenden privaten Schlüssels ist.

Selbstverständlich kann die Authentizität des öffentlichen Schlüssels einer TTP wiederum durch ein Zertifikat sichergestellt werden. Außerdem kann ein öffentlicher Schlüssel Zertifikate für mehrere andere Schlüssel abgeben und für einen öffentlichen Schlüssel können mehrere Zertifikate existieren. Auf diese Weise ergibt sich im Allgemeinen ein *gerichteter Graph* der als *Zertifikatsgraph* bezeichnet wird.

### 2.1.3.1 Zentrale PKI (besser: hierarchisch)

lassen sich dahingehend unterscheiden, welche Identitäten für das Ausstellen von Zertifikaten zuständig sind.

Ein bla bla bla ist eine Public-Key-Infrastruktur (PKI) nach dem X.509-Standard. In einer solchen PKI sind Certificate Authorities hierarchisch in einem Baum angeordnet. Jede CA zertifiziert diejenigen CAs, die in diesem Baum unterhalb von ihr liegen. zentral im Sinne von: Es muss einer einzigen zentralen Authority/Instanz vertraut werden.

kommerziell oder staatlich.

Zertifikatsgraph wird auf einen Baum beschränkt.

Üblicherweise wird die Authentizität des Schlüssels einer Certificate Authority von Benutzern nicht direkt überprüft. Stattdessen werden authentische Schlüssel der Certificate Authorities auf einem (mehr oder weniger) sicheren Weg an die Benutzer verteilt (beispielsweise als Teil des Betriebssystems oder des Webbrowsers).

Zu beachten ist, dass im Rest dieser Arbeit der Begriff "Schlüssel" ausschliesslich im Sinne des öffentlichen Teils eines OpenPGP-Schlüsselpaares verwendet wird.

Vielleicht Soghoyan (sp?) zitieren: X.509 wie es derzeit im Browser-Kontext verwendet wird, hilft nicht gegen MiTM.

### 2.1.4 Kryptographische Hashfunktionen

Eine grundlegende kryptographische Primitive sind *kryptographische Hashfunktionen*. Kryptographische Hashfunktionen sind unter anderem ein entscheidender Baustein für digitale Signaturschemata.

Eine Hashfunktion  $h$  ist eine Abbildung von Binärstrings beliebiger Länge auf Binärstrings einer festen Länge  $n$ . Die Ausgabe  $h(x)$  eines Binärstrings  $x$  wird als *Hashwert* (oder als *Fingerabdruck*) von  $x$  bezeichnet. Übliche Kriterien für eine gute Hashfunktion sind eine annähernde Gleichverteilung der Hashwerte und eine möglichst geringe Wahrscheinlichkeit von Kollisionen. Eine Kollision tritt dann auf, wenn  $h(x) = h(y)$  für zwei Eingaben  $x$  und  $y$  gilt.

Für eine kryptographische Hashfunktion werden zusätzliche Eigenschaften gefordert. Es darf nicht mit akzeptablem Berechnungsaufwand möglich sein, zwei Binärstrings  $x_1$  und  $x_2$  zu berechnen, so dass  $h(x_1) = h(x_2)$  gilt. Diese Eigenschaft wird als *Kollisionsresistenz* bezeichnet. Außerdem muss die Funktion resistent gegen *Urbild-Attacken*, sein, d.h. es darf mit akzeptablem Aufwand nicht möglich sein, zu einem Hashwert  $y$  einen Binärstring  $x$  zu berechnen, so dass  $h(x) = y$  gilt.

## 2.2 PGP/GnuPG

Die ursprüngliche Version von PGP wurde von Phil Zimmerman entwickelt. Der Quellcode von PGP wurde von Anfang an der Allgemeinheit zur Verfügung gestellt. PGP wird seit 1996 als kommerzielles Produkt vertrieben, blieb aber trotzdem kostenlos verfügbar[Wiki10]. Seit 1998 ist das verwendete Nachrichten- und Schlüsselformat durch den RFC 2440[CDFT98] und seit 2007 durch den überarbeiteten RFC 4880[CDFS<sup>+</sup>07] standardisiert. Seit 1999 existiert mit GnuPG[Gnu10]

eine freie Implementierung von OpenPGP unter der GNU General Public Licence, deren Entwicklung zeitweise vom deutschen Bundesministerium für Wirtschaft und Technologie gefördert wurde. GnuPG scheint inzwischen die am meisten verbreitete OpenPGP-Implementierung zu sein.

PGP benutzt asymmetrische Kryptographie für Verschlüsselung und digitale Signaturen und bietet damit ein Werkzeug, um Vertraulichkeit, Authentizität etc von Nachrichten zu erreichen. Die häufigste Anwendung von PGP ist die Verschlüsselung und Signierung von E-Mail-Nachrichten. Es kann aber auch benutzt werden, um beliebige Dokumente und Dateien zu verschlüsseln und zu signieren.

Wie die meisten kryptographischen Systeme, die asymmetrische Verfahren benutzen, ist PGP/GnuPG ein hybrides System. Die asymmetrischen Schlüssel werden nur benutzt, um einen symmetrischen Sitzungsschlüssel zu verschlüsseln, mit dem wiederum der eigentliche Nachrichteninhalt verschlüsselt wird. Der Grund dafür liegt unter anderem darin, dass symmetrische Verfahren in der Regel deutlich schneller sind als asymmetrische Verfahren.

PGP und GnuPG implementieren eine Vielzahl von symmetrischen und asymmetrischen sowie Hashalgorithmen. Nachdem GnuPG seit 1999 in der Standardeinstellung DSA/ElGamal-Schlüssel erzeugte, werden seit 2009 standardmäßig RSA-Schlüssel erzeugt (siehe Abschnitt 4.3).

OpenPGP-Schlüssel können mit einem Ablaufdatum versehen werden. Auch einzelne Teile eines Schlüssels, etwa einzelne UserIDs oder Signaturen, können ein Ablaufdatum erhalten. Ein Schlüssel kann aus verschiedenen Gründen nicht mehr benutzbar sein: Er kann kompromittiert worden sein, dass Schlüsselmaterial oder eine Passphrase können verloren gegangen sein und er kann schlichtweg durch einen anderen Schlüssel ersetzt worden sein. In solchen Fällen kann ein Schlüssel *widerrufen* werden, indem eine spezielle Widerrufssignatur durch den Besitzer auf dem Schlüssel angebracht wird. Ein abgelaufener oder widerrufener Schlüssel kann zwar noch benutzt werden, um Nachrichten zu entschlüsseln und Signaturen zu verifizieren. Er ist aber nicht mehr für Verschlüsselungs- und Signierungsoperationen verwendbar.

### 2.2.1 Web of Trust

Für die Authentifizierung von öffentlichen Schlüsseln benutzt PGP eine dezentrale Alternative zu dem in Abschnitt 2.1.3.1 beschriebenen hierarchischen und zentralen Modell. Die Fähigkeit, Zertifikate auszustellen, ist dabei nicht auf Certificate Authorities beschränkt. Stattdessen ist jeder Benutzer in der Lage, Zertifikate für öffentliche Schlüssel auszustellen. Ein öffentlicher Schlüssel kann von beliebig vielen Teilnehmern zertifiziert werden. Damit ergibt sich für die Zertifikate nicht nur ein Baum, sondern tatsächlich ein allgemeiner gerichteter Zertifikatsgraph. Dieser Graph wird als *Web of Trust*<sup>2</sup> bezeichnet.

Da jeder Teilnehmer Zertifikate ausstellen kann, stellt jeder Teilnehmer potentiell eine Trusted Third Party dar. Allerdings kann natürlich nicht jedem Teilnehmer dahingehend vertraut werden, korrekte Zertifikate auszustellen. Während im Fall

---

<sup>2</sup>Dieser Begriff ist irreführend, da der Zertifikatsgraph zunächst keine Informationen über Vertrauen enthält, sondern tatsächlich nur aus Zertifikaten besteht. Er wird in dieser Arbeit trotzdem benutzt, da er in der Literatur etabliert ist.

von X.509 das Vertrauen in die Certificate Authorities üblicherweise von vornherein festgelegt ist, muss im Fall des Web of Trust jeder Teilnehmer selbst entscheiden, welchen anderen Teilnehmern er vertraut. Er muss aufgrund von Vorwissen und Annahmen abschätzen, ob ein anderer Teilnehmer zuverlässig korrekte Zertifikate ausstellt.

Im Vergleich zu zentralen Modellen wird dem einzelnen Teilnehmern hier eine erheblich grössere Verantwortung und ein deutlich grösserer Aufwand zugewiesen, da er sowohl für die Überprüfung der Authentizität von Schlüsseln, für die er Zertifikate ausstellt, als auch für die Einschätzung der Zuverlässigkeit anderer Teilnehmer zuständig ist. Allerdings ergibt sich damit gleichzeitig ein erheblicher Gewinn an Flexibilität. Ein Teilnehmer ist eben nicht darauf angewiesen, sich auf zentrale Autoritäten zu verlassen. In zentralen Modellen hat er üblicherweise wenig Einfluss auf die Auswahl der als vertrauenswürdig angesehenen Certificate Authorities und er hat keine andere Wahl, als der Certificate Authority zu vertrauen, die ein bestimmtes Zertifikat ausgestellt hat. Vertraut er einer CA persönlich nicht, hat er keine direkte Möglichkeit, die von dieser zertifizierten Schlüssel zu authentifizieren. Im Web of Trust spielt es keine Rolle, ob ein öffentlicher Schlüssel von unvertrauenswürdigen Teilnehmern zertifiziert wurde, solange sich *ein* Zertifikatspfad mit vertrauenswürdigen Teilnehmern ergibt. Wird ein Schlüssel von mehreren als vertrauenswürdig eingeschätzten Teilnehmern zertifiziert, kann diese Zertifizierung als sicherer angesehen werden. Selbst wenn ein Angreifer einen Teilnehmer überlisten sollte, ein falsches Zertifikat auszustellen, ist die Wahrscheinlichkeit, dass er dies für mehrere Teilnehmer schafft, deutlich geringer.

Der gerichtete Zertifikatsgraph stellt eine Verallgemeinerung des Zertifikatsbaums von hierarchischen und zentralen Modellen dar. Tatsächlich kann das Modell des Web of Trust als eine Verallgemeinerung solcher Modelle angesehen werden. Es spricht zunächst nichts dagegen, mit den Mitteln des Web of Trust ein hierarchisches Modell auszudrücken und zu implementieren (siehe auch Abschnitt 2.2.5.2).. Die aktuelle Version des OpenPGP-Standards sieht einen Mechanismus vor, um einen Baum von Certificate Authorities kryptographisch sicher auszudrücken (sog. *trust signatures*).

Statt vom Ausstellen eines Zertifikats wird im Kontext von PGP üblicherweise vom *Signieren eines Schlüssels* gesprochen.

Der konkrete Algorithmus, mit dem PGP/GnuPG anhand eines Zertifikatsgraphen und einer Vertrauensdatenbank die Gültigkeit eines Schlüssels entscheiden, wird in Abschnitt 2.2.3 detailliert beschrieben.

## 2.2.2 Struktur von OpenPGP-Paketen

An dieser Stelle wird in groben Zügen beschrieben, wie ein OpenPGP-Schlüssel aufgebaut ist. Für Details wird auf den OpenPGP-Standard[CDFS<sup>07</sup>] verwiesen.

Ein OpenPGP-Schlüssel besteht aus einer Reihe von *Paketen*. Unterschiedliche Funktionen werden von unterschiedlichen Pakettypen wahrgenommen. Ein öffentlicher OpenPGP-Schlüssel enthält zunächst ein *Public-Key-Paket*, dass das eigentliche Schlüsselmaterial enthält. Weiterhin enthält ein Schlüssel eine Reihe von *User-ID*-Paketen, wobei mindestens ein solches Paket vorhanden sein muss. UserID-Pakete enthalten im Wesentlichen eine UserID bestehend aus einem Namen und einer E-Mail-Adresse. Sie geben also eine Identität an. Zu jedem UserID-Paket gehört eine

Reihe von *Signature*-Paketen. Diese bestehen aus einer digitalen Signatur über dem Public-Key-Paket und dem UserID-Paket. Eine Signatur auf einer UserID bindet den Schlüssel an diese Identität. Jede UserID verfügt über mindestens eine Selbstsignatur, die mit dem Schlüssel selbst erstellt wurde. Nach der obigen Definition eines Zertifikats stellt also jeder (öffentliche) OpenPGP-Schlüssel, auch wenn er keine Signaturen von Dritten enthält, ein selbstsigniertes Zertifikat dar. Werden zusätzliche Signaturen auf dem Schlüssel angebracht, ergibt sich gewissermassen ein Zertifikat mit mehreren Signaturteilen oder mehrere Zertifikate, die sich auf einen gemeinsamen Datenteil beziehen.

### 2.2.3 Das GnuPG-Vertrauensmodell

Öffentliche PGP-Schlüsse werden oft nicht in einer Weise übergeben, die die sofortige Verifikation des Schlüssels zulässt, beispielsweise bei einem persönlichen Treffen. Stattdessen werden Schlüsse häufig per E-Mail, über einen Keyserver (siehe Abschnitt 2.2.4) oder andere elektronische Wege ausgetauscht. Überprüfung der Authentizität eines Schlüssels ist deswegen von grosser Bedeutung.

Das Vertrauensmodell (Trust model – besserer Begriff?) ist der Algorithmus, mit dem PGP bzw. GnuPG entscheiden, ob ein Schlüssel anhand eines Web of Trust, also eines Zertifikatsgraphen mit Vertrauenswerten, als gültig betrachtet wird. Gültig meint an dieser Stelle, dass die Zugehörigkeit des Schlüssels zu dem angeblichen Besitzer bestätigt wurde.

PGP speichert Schlüssel in einem *Schlüsselring*. Jedem Schlüssel in einem Schlüsselring wird ein *Vertrauenswert* zugewiesen. Mögliche Werte sind

- Kein Vertrauen
- geringfügiges Vertrauen
- volles Vertrauen
- ultimatives Vertrauen

Der Vertrauenswert eines Schlüssels beschreibt, wie sehr der Besitzer des Schlüsselrings dem Besitzer des Schlüssels vertraut, zuverlässig korrekte Zertifikate zu erzeugen, also die Bindung von öffentlichen Schlüsseln an Personen zu verifizieren. Zur Überprüfung der Authentizität eines Schlüssels werden von GnuPG nur Signaturen betrachtet, die von Schlüsseln erzeugt wurden, deren Besitzern mindestens *geringfügig* vertraut wird. *Ultimativ* wird ausschliesslich implizit dem Besitzer des Schlüsselrings selbst vertraut. Es muss bemerkt werden, dass den einzelnen Vertrauenswerten keine festgelegte Semantik zugeordnet ist. Ihre Bedeutung ergibt sich eher anhand ihrer Auswirkungen im Algorithmus (siehe unten). Jeder Benutzer muss selbst entscheiden, anhand welcher Kriterien er welchen Personen wie stark vertraut. Diese Vertrauenswerte sind individuell. Jeder Benutzer verfügt über seine eigene, lokale Datenbank von Vertrauenswerten, die nicht mit anderen Benutzern geteilt werden<sup>3</sup>. Vertrauen ist also insbesondere nicht transitiv.

<sup>3</sup>Die Rede von “dem” Web of Trust ist insofern missverständlich. Eigentlich verfügt jeder Teilnehmer über sein persönliches Web of Trust, dass sich aus seiner persönlichen Vertrauensdatenbank und dem Netzwerk von Signaturen ergibt. Der Begriff “Web of Trust” wird im Rest dieser Arbeit aber nur im Sinne des Zertifikatsgraphen verwendet.

Ein Schlüssel wird von GnuPG genau dann als gültig betrachtet, wenn er die folgenden Bedingungen erfüllt:

1. Der Schlüssel wurde von ausreichend vielen *gültigen* Schlüsseln unterschrieben, d.h. er wurde mindestens entweder von
  - dem Besitzer des Schlüsselrings selbst (d.h. von einem Schlüssel mit *ultimativem Vertrauen*) unterschrieben
  - mindestens  $N$  gültigen Schlüsseln, denen voll vertraut wird, unterschrieben
  - mindestens  $M$  gültigen Schlüsseln, denen geringfügig vertraut wird, unterschrieben
2. Eine Signaturkette wird nur verwendet, wenn sie ausgehend vom Besitzer des Schlüsselrings maximal die Länge  $L$  hat.

Ein Schlüssel, der von weniger voll bzw. geringfügig vertrauenswürdigen Schlüsseln als notwendig unterschrieben wurde, wird als eingeschränkt gültig angesehen. Allerdings werden Schlüssel dieser Kategorie von GnuPG genauso wie nicht gültige Schlüssel behandelt.

GnuPG verwendet in der Standardeinstellung die Werte  $N = 1$ ,  $M = 3$  und  $L = 5$ . Damit wird *ein* Zertifikat, dass von einem Schlüssel mit vollem Vertrauen ausgestellt wurde, als ausreichend betrachtet. Für Schlüssel, denen nur geringfügig vertraut wird, ist ein einzelnes Zertifikat nicht ausreichend. Dieses muss noch durch 2 weitere solche Zertifikate bestätigt werden.

GnuPG erlaubt es einem Anwender, die Parameter  $N$ ,  $M$  und  $L$  selbst zu setzen und damit seine persönlichen Sicherheitsanforderungen umzusetzen. Je höher beispielsweise die notwendige Anzahl von Signaturen, um so kleiner ist der Schaden, den eine einzelne fehlerhafte Signatur anrichten kann. Ist die maximale Pfadlänge auf einen kleinen Wert begrenzt, so ist auch die maximale Anzahl der Signaturen auf dem Pfad kleiner, die potentiell fehlerhaft sein können. Andererseits verringert sich damit die Anzahl der Signaturen (Pfade im Web of Trust), die für die Verifizierung benutzt werden können, und damit die Anzahl verifizierbarer Schlüssel. Es muss also eine Abwägung zwischen dem Sicherheitsbedürfnis des Nutzers und der praktischen Benutzbarkeit getroffen werden.

OpenPGP-Signaturpakete werden in Typen (auch “certification level”) unterschieden, die angeben, wie gründlich der Ersteller der Signatur die Identität des Besitzers des signierten Schlüssels verifiziert hat. Diesen Typen wird durch den OpenPGP-Standard absichtlich keine klar definierte Semantik zugeschrieben, es werden lediglich Vorschläge für deren Bedeutung gemacht. Die möglichen Typen sind

**Generic** Der Ersteller der Signatur macht keine Aussage darüber, wie er überprüft hat, dass der Besitzer des Schlüssels die Person ist, die durch die UserID benannt wird.

**Persona** Der Ersteller hat die Identität des Schlüsselinhabers nicht verifiziert.

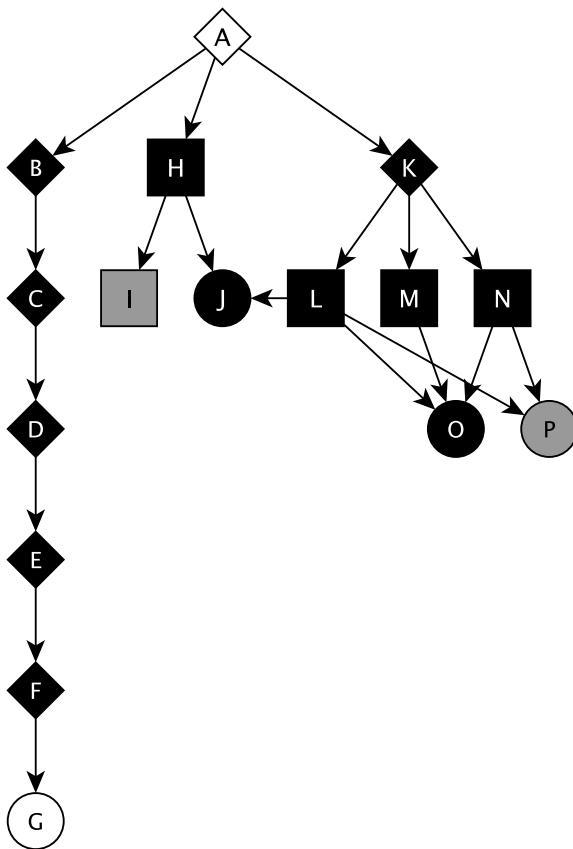


Abbildung 2.1: Beispiel für die Berechnung der Gültigkeit von Schlüsseln

**Casual** Der Ersteller hat die Identität des Schlüsselinhalters informell verifiziert.

**Positive** Der Ersteller hat die Identität des Schlüsselinhalters gründlich<sup>4</sup> verifiziert.

Mit den Standardeinstellungen werden Signaturen vom Typ “Generic” erzeugt. Diese Typen werden bei der Berechnung der Gültigkeit von Schlüsseln nur insofern verwendet, als Signaturen vom Typ “Persona” nicht beachtet werden. Die restlichen Typen werden gleich behandelt. Sie bieten aber die Möglichkeit, die Aussagekraft von Signaturen feiner zu untergliedern.

Abbildung 2.1 gibt ein Beispiel für die Berechnung der Schlüsselgültigkeit unter Verwendung der Standard-Parameter  $N = 1$ ,  $M = 3$  und  $L = 5$ . Die Schlüssel  $B$ ,  $H$  und  $K$  wurden direkt von  $A$ , dem Besitzer des Schlüsselrings, unterschrieben und sind deshalb voll gültig. Da  $L$ ,  $M$ , und  $N$  von  $K$  unterschrieben wurden und dieser über volles Vertrauen verfügt, sind sie ebenfalls voll gültig.  $O$  sowie  $J$  sind voll gültig, da sie jeweils von drei Schlüsseln mit geringfügigem Vertrauen unterschrieben wurden.  $I$  und  $P$  wurden dagegen jeweils nur von zwei Schlüsseln mit geringfügigem Vertrauen unterschrieben und sind deshalb nicht voll sondern nur eingeschränkt gültig.  $G$  wurde zwar von einem voll gültigen Schlüssel mit vollem Vertrauen unterschrieben. Allerdings überschreitet die Signaturkette zu  $A$  die maximale Länge von 5 und wird deshalb nicht benutzt.

<sup>4</sup>Es wird generell angenommen, dass mit diesem Typ eine Verifizierungsmethode wie in Abschnitt 2.2.5.1 beschrieben verbunden ist

Ein öffentlicher Schlüssel, der anhand dieser Regeln nicht als authentisch verifiziert werden kann, kann trotzdem zur Verschlüsselung und zur Verifizierung von Signaturen verwendet werden. Allerdings warnt GnuPG in diesem Fall vor der Verwendung.

Es muss bemerkt werden, dass keine direkte Verbindung zwischen dem Vertrauenswert und dem Gültigkeitswert eines Schlüssels besteht. Ein Schlüssel, dem voll vertraut wird, kann trotzdem aufgrund von fehlenden Signaturketten ungültig sein. Signaturen dieses Schlüssels können dann nicht benutzt werden.

### 2.2.4 Das Keyserver-Netzwerk

Die Verbreitung von öffentlichen Schlüsseln ist auf beliebigen Wegen möglich, etwa per E-Mail oder auf Webseiten. Ein zentraler Ablageort für öffentliche Schlüssel erleichtert das Auffinden derselben aber erheblich. Im Kontext von PGP wird diese Aufgabe von *Keyservern* übernommen. Dabei handelt es sich im wesentlichen um Datenbanken von OpenPGP-Schlüsseln, die nach verschiedenen Kriterien (etwa nach KeyID oder UserID) durchsucht werden können. Benutzer können ihre öffentlichen auf diese Keyserver laden, um sie der Öffentlichkeit zur Verfügung zu stellen.

Schlüssel können von beliebigen Personen auf Keyserver geladen werden, nicht nur durch die Benutzer. Auf diesem Weg werden auch Signaturen (also Zertifikate), die durch einen Zertifizierer auf einem Schlüssel angebracht wurden, verbreitet. Der Zertifizierer lädt den veränderten Schlüssel auf einen Keyserver. Dieser fügt die neuen Signaturpakete dem bisherigen Schlüssel hinzu. Dabei werden stets nur Pakete zu Schlüsseln hinzugefügt, nicht überschrieben oder gelöscht. Auf diese Weise machen Keyserver den Zertifikatsgraphen zugänglich. Selbstverständlich können Zertifikate auch auf anderen (eventuell privaten) Wegen ausgetauscht werden.

Um Lastverteilung und Fehlertoleranz zu erreichen, existieren mehrere Keyserver. Diese sind als Netzwerk organisiert und synchronisieren ihren Datenbestand ständig untereinander. Benutzer finden also auf allen Keyservern des Netzwerks mit sehr geringer Verzögerung den gleichen Datenbestand<sup>5</sup>. Das öffentliche Keyserver-Netzwerk besteht derzeit (Mai 2010) aus 48 Keyservern[SKS]. Die üblicherweise verwendete Keyserver-Software wird in Abschnitt 3.2.1 beschrieben.

Eine wichtige Eigenschaft des Keyserver-Netzwerks ist es, dass sich Schlüssel, die einmal auf einen der dazugehörigen Keyserver abgelegt wurden, nicht mehr aus dem Netzwerk entfernen lassen. Zwar kann ein Schlüssel oder ein einzelnes OpenPGP-Paket aus der Datenbank von einzelnen Keyservern gelöscht werden. Allerdings wird er in diesem Fall durch die Synchronisierung mit anderen Keyservern des Netzwerks wieder auftauchen. Das Keyserver-Netzwerk insgesamt sieht keine Prozedur für die Entfernung von Schlüsseln vor. Das bedeutet, dass das öffentliche Keyserver-Netzwerk alle<sup>6</sup> jemals dort veröffentlichten Schlüssel enthält. Dieses Verhalten stellt für Benutzer kein wirkliches Problem dar. Der korrekte Weg, um einen Schlüssel, der aus beliebigen Gründen nicht mehr benutzbar ist, unbrauchbar zu machen, ist der Widerruf des Schlüssels. Umgekehrt würde eine Möglichkeit zur Löschung von

---

<sup>5</sup>In Tests des Autoren dauerte es normalerweise weniger als 10 Sekunden, bis neue Schlüssel auf alle Keyserver des Verbundes propagierte waren.

<sup>6</sup>Es kann nicht ausgeschlossen werden, dass einzelne Schlüssel durch Softwarefehler verloren gingen. Die zahlreichen Kopien des Datenbestandes auf den einzelnen Keyservern und die ständige Synchronisation machen dies jedoch unwahrscheinlich.

Schlüsseln einem Angreifer die Möglichkeit eröffnen, einzelne Schlüsselteile – beispielsweise Signaturpakete mit Widerrufssignaturen – zu entfernen und dadurch eigentlich widerrufene kompromittierte Schlüssel wieder gültig erscheinen zu lassen.

Der Widerruf eines Schlüssels oder einzelner Schlüsselbestandteile durch seinen Besitzer kann nur effektiv erfolgen, wenn die Information über den Widerruf an alle Personen verteilt wird, die den entsprechenden öffentlichen Schlüssel potentiell benutzen. Keyserver bieten die Möglichkeit, die Information über den Widerruf von Schlüsseln zentral und öffentlich verfügbar zu machen, indem das Widerrufszertifikat auf den Keyservern verfügbar gemacht wird. Allerdings kann dieses Vorgehen natürlich nur funktionieren, wenn Benutzer ihren Schlüsselring regelmäßig mit den Informationen eines Keyservern aktualisieren. Im Fall der Kompromittierung eines Schlüssels ist die Information aber zeitkritisch. Anwender sollten dann direkt informiert werden.

Indem ein Benutzer Signaturen auf einem Keyserver zur Verfügung stellt, macht er Informationen über seine Beziehungen für die Öffentlichkeit sichtbar. In Abschnitt 2.3.4 wird argumentiert, dass die Signaturbeziehungen einer Person ein Abbild sozialer Beziehungen von unterschiedlicher Art sind. Die rein topologische Information eines sozialen Netzwerks erlaubt detaillierte Aussagen über die Rolle einzelner Personen im Netzwerk[CaSW05]. Benutzer müssen sich daher bewusst sein, dass die Benutzung öffentlicher Keyserver einen Verlust von Privatsphäre mit sich bringt und müssen dies gegenüber einem möglichen Gewinn an Sicherheit und Bequemlichkeit abwägen.

## 2.2.5 Zustandekommen von Signaturen im Web of Trust

### 2.2.5.1 Übliche Zertifizierungsprozeduren

Bis jetzt wurde nur erklärt, dass eine Signatur die Zusicherung über die Bindung eines Schlüssels an eine Person, d.h. eine Identität, darstellt. In diesem Abschnitt wird näher darauf eingegangen, wie solche Zusicherungen *üblicherweise* zustande kommen. Außerdem werden einige Aspekte (FIXME Rahmenbedingungen)

Bemerkt werden muss, dass der Zertifizierungsprozess im Kontext des Web of Trust nicht formal definiert ist. Weder der OpenPGP-RFC noch andere Standarddokumente machen dazu Vorschriften. Nur Certificate Authorities, die PGP verwenden, und einige wenige Privatpersonen dokumentieren ihre Verfahren zur Zertifizierung in Form von sogenannten “Keypair-Policies”. Außerdem ist dem Autor dieser Arbeit keine Studie über tatsächlich verwendete Mechanismen bekannt. Deshalb wird an dieser Stelle nur anekdotische Evidenz als Eindruck von üblichen Praktiken präsentiert.

Die Person, die eine Signatur erstellt, wird im folgenden mit *Unterzeichner*, und die Person, deren Schlüssel unterschrieben wird, mit *Unterzeichneter* bezeichnet. Um die oben erwähnte Zusicherung abgeben zu können, muss der Unterzeichner die Identität des Unterzeichneten überprüfen und so sicherstellen, dass sie mit der auf dem Schlüssel angegebenen Identität (in Form der *User ID*) übereinstimmt. Diese Überprüfung sollte in Form eines *persönlichen* Treffens stattfinden, da sonst eine sinnvolle Identitätsüberprüfung im Allgemeinen nicht möglich ist. Der Unterzeichnete sollte zuerst den Fingerabdruck seines Schlüssels präsentieren. Anhand dieses Fingerabdrucks kann der Unterzeichnete überprüfen, dass der zu unterschreibende

Schlüssel tatsächlich der des Unterzeichneten ist. Der Unterzeichnete sollte dann ein Dokument präsentieren, dass seine Identität belegt. Um die Verwendung von gefälschten Dokumenten zu erschweren, wird üblicherweise ein amtliches Ausweisdokument mit Lichtbild gefordert (Personalausweis, Reisepass). Der Unterzeichner kann nun anhand des Dokumentes überprüfen, dass der Unterzeichner mit der auf dem Schlüssel vermerkten Identität identisch ist. Damit kann der Unterzeichner den Schlüssel unterschreiben und die obige Zusicherung abgeben. Normalerweise werden beide Teilnehmer ihre Schlüssel gegenseitig unterschreiben, also die Rollen von Unterzeichner und Unterzeichnetem wechseln.

Dies sind die üblichen Anforderungen an einen Zertifizierungsprozess, wie sie von ernsthaften Benutzern des Web of Trust erwartet werden. Allerdings hindert einen Teilnehmer nichts daran, diese Anforderungen beliebig abzuschwächen. Er kann beispielsweise auf die Identitätsüberprüfung von Personen verzichten, die ihm persönlich bekannt sind oder im Extremfall auf jegliche Überprüfung verzichten. Allerdings ist jeder Benutzer selbst dafür verantwortlich zu entscheiden, welchen Personen er dahingehend vertraut, korrekte Zusicherungen abzugeben.

### 2.2.5.2 Certificate Authorities im Web of Trust

Eine Reihe von Einrichtungen agiert im Kontext des Web of Trust als *Certificate Authorities* im Sinne von Abschnitt 2.1.3.1. Diese Einrichtungen bieten als Dienstleistung die Signierung von PGP-Schlüsseln nach Überprüfung der Identität an. Der Prozess bzw. die Kriterien, nach dem diese Signaturen zustande kommen, ist dokumentiert. Dieser Prozess unterscheidet sich üblicherweise nicht von dem in Abschnitt 2.2.5.1 beschriebenen Vorgehen.

- Der Heise-Verlag betreibt seit 1997 eine Certificate Authority im Rahmen der “Krypto-Kampagne”. Diese hat sich zum Ziel gesetzt, die Verbreitung von PGP zu fördern. Durch eine kostenlose Certificate Authority soll das Web of Trust gestärkt werden (FIXME cite).

Die Zertifizierung wird auf Messen gegen Vorlage des Personalausweises angeboten. Die von dieser CA verwendeten Schlüssel haben insgesamt ca. 22000 Signaturen erstellt.

- Die Certificate Authority CACert (FIXME link) zertifiziert unter anderem auch OpenPGP-Schlüssel. Der von dieser CA verwendete Schlüssel hat ca. 3400 Schlüssel signiert.
- Das Deutsche Forschungsnetzwerk (DFN) betrieb vermutlich bis zum Jahr 2009 eine Certificate Authority für PGP.

Das Konzept einer Certificate Authority steht nicht grundsätzlich im Widerspruch zu einem Web of Trust. Eine Verifizierungsprozedur wie in Abschnitt 2.2.5.1 macht für einen Schlüssel einer Certificate Authority keinen Sinn, da er üblicherweise nicht mit einer einzelnen Person verbunden ist. Ein Benutzer muss sicherstellen, dass der CA-Schlüssel authentisch ist und entscheiden, ob er dem Betreiber dahingehend vertraut, die dokumentierte Policy FIXME zuverlässig umzusetzen. Zusätzlich muss allerdings der CA-Schlüssel signiert werden, um seine Signaturen benutzen zu können. Werden diese Signaturen auf dem CA-Schlüssel veröffentlicht, dann ergeben sich

Signaturpfade, die zu diesem Schlüssel führen. Der CA-Schlüssel kann dann auch von Teilnehmern, die ihn nicht direkt unterschrieben haben, als Teil ganz normaler Signaturketten verwendet werden (sofern sie dem Betreiber der Certificate Authority vertrauen). Certificate Authorities fügen sich also problemlos in das dezentrale Web of Trust ein.

### 2.2.5.3 Keysigning-Parties

Das in Abschnitt 2.2.5.1 beschriebene übliche Verfahren betrifft zunächst nur zwei Personen. Wenn mehrere Personen zusammenkommen, um ihre Schlüssel gegenseitig zu unterschreiben, wird von einer *Keysigning-Party* gesprochen. Die Bandbreite dieser Veranstaltungen ist sehr gross. Sie reicht von informellen Zusammenkünften mehrerer Personen, die paarweise ihre Schlüssel unterschreiben, bis hin zu grösseren, formalisierten Veranstaltungen im Kontext von Konferenzen und Messen. Bei grösseren Veranstaltungen wird üblicherweise eine Anmeldung der Teilnehmer erwartet. Für grössere Veranstaltungen hat sich eine Reihe von "Protokollen" etabliert, um den Ablauf effizient zu organisieren [Bren08].

Grössere Keysigning-Parties fanden im Jahr 2009 beispielsweise auf diesen Veranstaltungen statt:

- Debconf (Konferenz des Debian-Projekts) mit XXX Teilnehmern
- FOSDEM (Open-Source-Konferenz) mit XXX Teilnehmern
- LinuxTag (Messe) mit XXX-Teilnehmern

Offensichtlich ist die Obergrenze der Signaturen, die aus einer Keysigning-Party mit  $n$  Teilnehmern entstehen können,  $n^2$ . Schon bei Veranstaltungen mit wenigen hundert Teilnehmern werden also erhebliche Mengen von Signaturen dem Web of Trust hinzugefügt.

Insbesondere auf kleinen und spontanen Veranstaltungen ist es jedem Teilnehmer überlassen, mit welchen anderen Teilnehmern er Signaturen austauscht. Üblich ist allerdings, dass (fast) alle Paare von Teilnehmern Signaturen austauschen. Ein hervorstechendes Merkmal vieler Keysigning-Parties ist also eine (fast) vollständige Vernetzung ihrer Teilnehmer.

### 2.2.5.4 Foobar FIXME

In einer Reihe von Open-Source-Projekten spielen PGP und das Web of Trust eine wichtige Rolle. Im Debian-Projekt beispielweise werden PGP-Schlüssel unter anderem benutzt, um alle bereitgestellten Softwarepakete durch den zuständigen Entwickler signieren zu lassen. Auf diese Weise wird überprüft, dass ein hochgeladenes Paket tatsächlich von dem verantwortlichen Projektmitglied stammt und nicht verändert wurde. Dazu muss jeder Debian-Entwickler über einen Schlüssel verfügen, der von mindestens einem anderen Debian-Entwickler verifiziert und unterschrieben wurde. Auf diese Weise bildet sich bereits innerhalb des Projekts ein Web of Trust, dass einen Teil des globalen Web of Trust darstellt. PGP scheint eine wichtige Rolle in der Kultur des Projekts zu spielen. Darauf weist beispielsweise hin, dass auf Treffen von Projektmitgliedern oft Keysigning-Parties abgehalten werden, um das Web of Trust zu stärken. Ausserdem signiert eine überdurchschnittliche Anzahl von Debian-Entwicklern ihre E-Mails auf projektinternen Mailinglisten.

## 2.3 Graphentheorie und Netzwerkanalyse

### 2.3.1 Graphentheorie allgemein

Dieser Abschnitt führt einige grundlegende graphentheoretische Begriffe nach [BrEr04] ein.

Ein *Netzwerk* bezeichnet eine Ansammlung von Objekten, zwischen denen bilaterale Beziehungen bestehen. Das hier betrachtete Netzwerk besteht aus einer Ansammlung von OpenPGP-Schlüsseln, zwischen denen Beziehungen in Form von Signaturen bestehen. Ein Netzwerk kann durch einen Graphen repräsentiert werden.

Ein *Graph*  $G = (V, E)$  besteht aus einer endlichen Menge  $V$  von *Knoten* und einer endlichen Menge  $E$  von *Kanten*, die je zwei Knoten miteinander verbinden. Die Anzahl  $|V|$  der Knoten wird mit  $n$  und die Anzahl der Kanten mit  $m$  bezeichnet. Zwei durch eine Kante verbundene, *benachbarte* Knoten heißen *adjazent*. In einem *ungerichteten* Graphen ist  $E$  eine Teilmenge von  $V \times V$ , also eine Menge von Kanten  $\{u, v\}$ , die zwei Knoten  $u \in V$  und  $v \in V$  verbinden. In einem *gerichteten* Graphen besteht eine Kante aus einem geordneten Paar  $(u, v)$  von Knoten. Eine gerichtete Kante  $(u, v)$  führt vom *Ursprung*  $u$  zum *Ziel*  $v$ .

Ein Graph ist *vollständig*, wenn er alle möglichen Kanten enthält.

Der *Grad*  $d(v)$  eines Knotens  $v$  in einem ungerichteten Graphen bezeichnet die Anzahl der Kanten, die diesen Knoten enthalten. Die *Nachbarschaft*  $N(v)$  bezeichnet die Menge der Knoten, die zu  $v$  adjazent sind. In einem gerichteten Graphen bezeichnet der *eingehende Grad*  $d^-(v)$  die Anzahl der Kanten, die den Knoten  $v$  als Ziel enthalten, und der *ausgehende Grad*  $d^+(v)$  die Anzahl der Kanten, die  $v$  als Ursprung enthalten.

Ein *Kantenzug* zwischen Knoten  $v_0$  und  $v_k$  in einem Graphen  $G = (V, E)$  ist eine alternierende Folge  $(v_0, e_1, v_1, e_2, \dots, v_{k-1}, e_k, v_k)$ , wobei  $v_i \in V$ ,  $e_i \in E$ , sowie  $e_i = \{v_{i-1}, v_i\}$  in einem ungerichteten Graphen bzw.  $e_i = (v_{i-1}, v_i)$  in einem gerichteten Graphen. Der Kantenzug hat die *Länge*  $i$ . Ein Kantenzug heißt *einfach*, wenn  $e_i \neq e_j$  für  $i \neq j$  gilt. Ein einfacher Kantenzug heißt *Pfad*, wenn  $v_0, \dots, v_k$  paarweise verschieden sind. Die *Distanz*  $d(u, v)$  von einem Knoten  $u$  zu einem Knoten  $v$  ist die Länge eines kürzesten Pfades zwischen  $u$  und  $v$ . Besteht kein Pfad von  $u$  nach  $v$ , so wird  $d(u, v) = \infty$  definiert.

Ein Graph  $G' = (V', E')$  ist ein *Teilgraph* eines Graphen  $G = (V, E)$ , wenn  $E' \subseteq E$  und  $V' \subseteq V$  gelten. Ein Teilgraph  $G' = (V', E')$  eines Graphen  $G$  ist durch eine Knotenmenge  $V'$  *induziert*, wenn  $E' = \{(u, v) : u \in V, v \in V, (u, v) \in E\}$  gilt.

Eine Partitionierung eines Graphen bezeichnet eine Zerlegung der Knotenmenge in disjunkte Teilmengen.

Ein ungerichteter Graph heißt *zusammenhängend*, wenn es zwischen allen Knotenpaaren  $u, v$  einen Pfad gibt. Eine *Zusammenhangskomponente* eines Graphen  $G$  ist ein *maximaler*, zusammenhängender induzierter Teilgraph von  $G$ . Ein gerichteter Graph ist *stark zusammenhängend*, wenn es für jeden Knoten einen gerichteten Pfad zu jedem anderen Knoten gibt. Eine *starke Zusammenhangskomponente* eines Graphen  $G$  ist dann ein maximaler, stark zusammenhängender induzierter Teilgraph von  $G$ .

### 2.3.2 Netzwerkstatistiken

In diesem Abschnitt werden einige Kennzahlen nach [BrSc04] definiert, die die Struktur eines Graphen charakterisieren.

Die Distanz zwischen zwei Knoten wurde bereits im vorherigen Abschnitt definiert. Ausgehend davon können die Distanzen *eines Knotens* durch die durchschnittliche Distanz dieses Knotens zu allen anderen Knoten charakterisiert werden:

$$\bar{d}(u) = \frac{1}{n-1} \sum_{v \neq u} d(u, v) \quad (2.1)$$

Für einen gesamten Graphen gibt die *charakteristische oder durchschnittliche Distanz* den Durchschnitt aller Distanzen in diesem Graphen an:

$$\bar{d} = \frac{1}{n^2 - n} \sum_{u \neq v \in V} d(u, v) \quad (2.2)$$

Die *Eccentricity* eines Knotens  $u$  ist definiert als die maximale Distanz zu einem anderen Knoten, also

$$\epsilon(u) = \max\{d(u, v) | v \in V\} \quad (2.3)$$

Davon ausgehend wird der *Durchmesser* eines Graphen als Maximum und der Radius als Minimum über die Eccentricity aller Knoten definiert. Durchmesser und Radius geben also die Ober- bzw. Untergrenze der Pfadlänge an, mit der ein Knoten alle anderen Knoten erreichen kann. Bemerkt werden muss, dass alle auf Distanzen basierenden Kennzahlen im Falle eines nicht (stark) zusammenhängenden Graphen unendlich sind. Dieses Problem wird im weiteren vermieden, indem diese Kennzahlen nur für (starke) Zusammenhangskomponenten berechnet werden.

Als Verallgemeinerung der Nachbarschaft eines Knotens wird die  *$h$ -Nachbarschaft* eines Knotens als

$$N_h(v) = \{u \in V | d(v, u) \leq h\} \quad (2.4)$$

definiert, d.h. die Menge der Knoten, zu denen die Distanz von  $v$  aus höchstens  $h$  beträgt.

Der *Clustering-Koeffizient* nach Watts und Strogatz ist ein Mass dafür, wie *transitiv* die Beziehungen in einem Netzwerk sind. Ein hoher Clustering-Koeffizient steht für eine hohe Wahrscheinlichkeit, dass zwei Nachbarn  $v$  und  $w$  eines Knotens  $u$  auch untereinander verbunden sind. Sei  $G = (V, E)$  ein ungerichteter Graph. Ein Dreieck  $\Delta = \{V_\Delta, E_\Delta\}$  ist ein vollständiger Teilgraph der Grösse 3 von  $G$ . Die Anzahl der Dreiecke *eines Knotens* wird mit  $\lambda(v) = |\{\Delta : v \in V_\Delta\}|$  bezeichnet. Ein *Tripel* an einem Knoten  $v$  ist ein Teilgraph von  $G$  bestehend aus 2 Kanten sowie  $v$  und 2 weiteren Knoten, so dass beide Kanten  $v$  enthalten. Die Anzahl der Tripel eines Knotens  $v$  kann durch den Grad  $d(v)$  ausgedrückt werden als  $\tau(v) = \binom{d(v)}{2}$ . Der *lokale Clustering-Koeffizient*  $c_v$  eines Knotens wird dann als

$$c(v) = \frac{\lambda(v)}{\tau(v)} \quad (2.5)$$

definiert.  $c(v)$  gibt also gewissermassen an, wie viele der möglichen Dreiecke an  $v$  auch tatsächlich Dreiecke sind. Der globale Clustering-Koeffizient von  $G$  ist dann

$$C(G) = \frac{1}{|V'|} \sum_{v \in V'} c(v) \quad (2.6)$$

wobei  $V' = \{v \in V : d(v) \geq 2\}$  gesetzt wird, um nicht definierte Werte für  $\tau(v)$  zu vermeiden.

Die *Transitivity* nach Newman et al. drückt zwar den gleichen Sachverhalt aus wie der in dieser Arbeit verwendete Clustering-Koeffizient, ist aber nicht äquivalent zu diesem und sollte nicht verwechselt werden.

### 2.3.3 Netzwerkmodelle

[BaAl99] baz

### 2.3.4 Soziale Netzwerke

Ein *soziales Netzwerk* ist nach Newman[Newm03] eine Menge von Personen bzw. Gruppen von Personen, zwischen denen Beziehungen bzw. Kontakte bestehen. Die Art dieser Interaktionen ist dabei beliebig: Soziale Netzwerke, die in der Literatur analysiert wurden, beinhalten Freundschaftsnetzwerke von Schulkindern, Netzwerke von sexuellen Kontakten und Netzwerke von Geschäftsbeziehungen (ebd.).

Die im Web of Trust vorhandenen Schlüssel können als Repräsentationen der Besitzer dieser Schlüssel gedacht werden. Allerdings ist dies keine Eins-zu-eins-Beziehung, da eine Person über beliebig viele Schlüssel verfügen kann. Die Beziehung dieser Personen im Web of Trust repräsentiert zunächst nur eine Signatur, also eine Zuschreibung über die Bindung zwischen dem signierten Schlüssel und dem Besitzer. Es scheint allerdings recht unwahrscheinlich, dass solche Signaturen zwischen Personen entstehen, die sich völlig unbekannt sind und in keiner Beziehung zueinander stehen. Stattdessen wird in dieser Arbeit davon ausgegangen, dass sich Signaturen in den meisten Fällen anhand von bereits bestehenden sozialen Kontakten zwischen den Besitzern der Schlüssel ergeben[CaBH02]. Das Spektrum möglicher Beziehungen ist dabei natürlich sehr weit und umfasst mindestens alle Gruppen von Personen, die potentiell ein Interesse an vertraulicher und authentifizierter E-Mail-Kommunikation haben: Es kann sich dabei um persönliche Bekanntschaften handeln, also etwa um Freunde und Bekannte. Genauso kann sich die Beziehung auch primär durch die Mitgliedschaft in einer gemeinsamen Gruppe oder Institution ergeben. Denkbar sind hier zum Beispiel Beziehungen, die sich aus Arbeitsverhältnissen oder einer Ausbildung ergeben, also Unternehmen oder akademische Einrichtungen. Genausogut kann es sich aber um die Mitgliedschaft in einem Open-Source-Projekt, einem Verein oder einer politischen Organisation handeln. Auch Zusammenhänge, die nicht über eine formale Mitgliedschaft in einer festen Gruppierung entstehen, sind möglich.

Zumindest im Fall von grösseren, formalen Keysigning-Parties scheint zunächst nicht notwendigerweise eine Verbindung zwischen den Teilnehmern (abgesehen von der Teilnahme selbst) zu bestehen. Es ist aber auch der Kontext relevant, in dem die Keysigning-Party stattfindet. Bei einer Konferenz oder Messe kann es sich um ein Ereignis mit einer klar definierten Zielgruppe handeln, die wiederum einer Gruppe

zuordnenbar ist (z.B. im Fall der “Debconf”-Konferenzen). Ist dies nicht möglich, so handelt es sich immerhin um eine Gruppe von Personen, die über ein gemeinsames Interessengebiet verfügen (beispielsweise im Fall der “LinuxTag”-Keysigning-Parties).

### 2.3.5 Communities

Dieser Abschnitt führt den Begriff der *Community* ein, definiert das Gütemass *Modularity* und beschreibt einige Algorithmen zur Erkennung von Communities in Netzwerken. Sofern nicht anders angegeben, wurde als Quelle ein Übersichtsartikel von Fortunato [Fort10] benutzt.

#### 2.3.5.1 Grundlegende Begriffe

In vielen – insbesondere sozialen – Netzwerken neigen die Teilnehmer dazu, *Gruppen* zu bilden. In sozialen Netzwerken bilden sich etwa Gruppen von Teilnehmern, die über familiäre, freundschaftliche oder professionelle Beziehungen miteinander verbunden sind. Ein besonders intensiv untersuchter Typ solcher Netzwerke entsteht aus Kooperations- oder Koautornetzwerken von Wissenschaftlern, in denen sich Wissenschaftler nach ihren Fachgebieten gruppieren. Andere Beispiele von Netzwerken, in denen Gruppenbildung stattfindet, sind Netzwerke aus der Biologie, beispielweise Netzwerke von Protein-Protein-Interaktionen, in denen sich funktionell ähnliche Proteine gruppieren, die an den gleichen biologischen Prozessen teilnehmen. Das hier zentrale Merkmal dieser Gruppen oder *Communities* ist die Tatsache, dass Mitglieder einer Gruppe normalerweise deutlich mehr Beziehungen zu anderen Mitgliedern dieser Gruppe haben als zu Teilnehmern ausserhalb. Eine solche Community sollte sich also auch in der Struktur des Graphen wieder finden, der das Netzwerk beschreibt. In der Tat teilen viele – insbesondere die oben angeführten – Beispiele von Netzwerken die Eigenschaft, dass die Kanten des Netzwerks nicht gleichmäßig verteilt sind. Stattdessen finden sich Gruppen von Knoten, in denen eine hohe Dichte von Kanten herrscht, während zwischen diesen Gruppen nur wenige Kanten bestehen.

In der Literatur scheint keine strenge Definition einer Community zu existieren, die allgemein akzeptiert ist. Stattdessen wird ein intuitives Verständnis verwendet, nach dem Communities *dichte* Teilbereiche eines Graphen sind, also Gruppen von Knoten, die untereinander “viele” Kanten haben, während zu nicht in der Gruppe liegenden Knoten nur “wenige” Kanten bestehen.

Da ein Graph über ein konkretes Netzwerk abstrahiert und nur noch die Struktur abbildet, werden auch bei der Community-Erkennung nur Informationen benutzt, die in der Topologie des Graphen kodiert sind.

[GiNe02]

#### 2.3.5.2 Modularity

Eine solche hierarchische Zerlegung macht noch keine Aussage darüber, wie gut die Zerlegung auf einer bestimmten Ebene die tatsächliche Community-Struktur wiedergibt. Um diese Frage zu beantworten, wurde ebenfalls von Newman als Mass für die Qualität einer Zerlegung die sogenannte *Modularity* definiert. Gegeben einen

ungerichteten Graphen  $G = (V, E)$  und eine Zerlegung dieses Graphen  $\mathcal{C} = C_1, \dots, C_n$  wird die Modularity  $Q$  definiert als

$$Q = \frac{1}{2m} \sum_{u,v \in V} \left( A_{uv} - \frac{d_i d_j}{2m} \right) \delta(C_u, C_v) \quad (2.7)$$

### 2.3.5.3 Weitere Algorithmen

Seit der Arbeit von Newman wurde eine Vielzahl weiterer Methoden zur Erkennung von Communities vorgestellt. An dieser Stelle werden 3 Methoden näher beschrieben, die in dieser Arbeit verwendet werden. Einen Gesamtüberblick bietet ein Übersichtsartikel von Fortunato [Fort10].

**Algorithmus von Clauset et al. (CNM)**

**Algorithmus von Blondel et al.**

**COPRA**

## 2.4 Verwandte Arbeiten

### 2.4.1 Analysen des OpenPGP-Web of Trust

An dieser Stelle wird nur auf Arbeiten eingegangen, die sich mit der Netzwerkstruktur des Web of Trust bei PGP beschäftigen. Für einen Überblick über Arbeiten zur Struktur von anderen Netzwerken sei auf [Newm03] und für Arbeiten zur Community-Struktur von anderen Netzwerken auf [Fort10] verwiesen.

Capkun et al. [CaBH02] untersuchten strukturelle Aspekte der grössten starken Zusammenhangskomponente eines PGP-Netzwerkes aus dem Jahr 2001. Dieses Netzwerk zeigte eine geringe charakteristische Distanz und einen hohen Clustering-Koeffizienten. Außerdem stimmte die Verteilung der aus- und eingehenden Grade mit einem Power-Law überein. Die Autoren argumentieren, dass Netzwerke, die aus Zertifikatsbeziehungen entstehen, naturgemäss den Small-World-Effekt zeigen, da sich die Zertifikate anhand schon bestehender Vertrauensbeziehungen ergeben. Die Autoren geben außerdem noch ein Netzwerkmodell an, dass Graphen erzeugen soll, die in den beschriebenen Charakteristiken dem PGP-Netzwerk ähneln.

Arenas et al. [BnPSDGA04] analysierten ein PGP-Netzwerk aus dem gleichen Jahr. Das Netzwerk wurde von vornherein in ein ungerichtetes Netzwerk umgewandelt, indem nicht gegenseitige Kanten entfernt wurden. Sie betrachteten die Gradverteilung, den Clustering-Koeffizienten und die Korrelation von Graden benachbarter Knoten. Dabei folgt die Verteilung der Knotengrade einem Power-Law mit einem exponentiellen Cutoff FIXME. Der Clustering-Koeffizient des Netzwerkes beträgt  $C = 0.4$ , ist also recht hoch. Die Grade benachbarter Knoten sind positiv korreliert. Außerdem benutzten die Autoren den in Abschnitt 2.3.5.1 erwähnten Algorithmus von Girvan und Newman, um eine Community-Zerlegung des Netzwerks zu berechnen. Die Verteilung der Grösse dieser Communities folgt wieder einem Power-Law.

Webzeugs (henning -> pgp.cs.uu.nl?, wotsap)

# 3. Methoden und Materialien

TODO UserID = Benutzerkennung?

## 3.1 Warum eigene Software?

Das bereits in Abschnitt 2.4 erwähnte *wotsap*-Projekt berechnet täglich die Struktur des Web of Trust. Die Daten werden für eine Web-Applikation benutzt, die grundlegende Statistiken über Schlüssel berechnet und Pfade zwischen Schlüsseln graphisch darstellt. Zusätzlich werden die Daten aber auch für weitere Analysen zur Verfügung gestellt. In diesem Abschnitt wird begründet, warum für die vorliegende Arbeit nicht auf diese Daten zurückgegriffen wurde, sondern die Datenextraktion selbst vorgenommen wurde.

Ein Ziel dieser Arbeit ist es, die Struktur des Web of Trust abseits der grössten starken Zusammenhangskomponente zu untersuchen. *wotsap* berechnet allerdings nur die Struktur eben dieser Zusammenhangskomponente. Schlüssel, die nicht in dieser Komponente enthalten sind, werden nicht beachtet. *wotsap* beginnt bei mehreren sehr gut vernetzten Schlüsseln, die sicher in der MSCC liegen. Von diesen ausgehend werden die Signaturen in der Art einer Breitensuche (rückwärts) verfolgt. Aufgrund dieser Methode scheint es mit vertretbarem Aufwand nicht möglich, die Extraktion auf alle Schlüssel auszudehnen.

Der Anwendungszweck der *wotsap*-Daten ist ausschliesslich die strukturelle Analyse des Netzes. Die über die reine Struktur hinausgehenden Daten, die über Schlüssel und Signaturen gespeichert werden, sind dabei auf ein Minimum reduziert: Für Schlüssel werden ausschliesslich die KeyID und die primäre UserID gespeichert, für Signaturen der Cert level und die Schlüssel. Diese Reduktion erlaubt zwar eine sehr kompakte Speicherung der Daten, macht den Datensatz aber für eine Auswertung weiterer Eigenschaften von Schlüsseln und Signaturen unbrauchbar. Das verwendete Dateiformat ist ausserdem recht unflexibel und lässt eine Speicherung weiterer Daten nur mit grösserem Aufwand zu.

Die *wotsap*-Daten beinhalten nur die zum jeweiligen Zeitpunkt gültigen Schlüssel und Signaturen. Dieser „Schnappschuss“ reicht für die strukturelle Analyse des Graphen aus. Zeitliche Entwicklungen, beispielsweise die Grösse des Datenbestandes, die

Verwendung bestimmter Verschlüsselungs- und Signaturalgorithmen und die Entwicklung einzelner Komponenten können damit aber nicht nachvollzogen werden.

Die *wotsap*-Methode liefert also nicht die im Rahmen dieser Arbeit benötigten Daten. Eine Anpassung der Software würde auf eine komplette Neuimplementierung hinauslaufen. Außerdem beruht die Extraktion der Daten bei *wotsap* auf der veralteten und kaum mehr benutzten *PKS*-Keyserver-Implementierung (siehe Abschnitt 3.2.1).

Darüber hinaus ist allerdings der *wotsap*-Datensatz fehlerhaft. Diese Fehler werden sowohl durch Fehler in der Implementierung als auch durch einen fehlerhaften Datenbestand auf dem verwendeten Keyserver verursacht: Die grösste starke Zusammenhangskomponente die durch XXX berechnet wurde (MSCC-1), enthält mit dem Stand vom 02.12.2009 ca. 44900 Schlüssel, während der *wotsap*-Datensatz (MSCC-2) vom gleichen Tag nur ca. 42130 Schlüssel enthält. Die Differenz zwischen den Datensätzen ergibt einerseits, dass ca. 1000 Schlüssel in MSCC-1 fehlen, die in MSCC-2 vorhanden sind. Eine stichprobenartige Analyse von 20 Schlüsseln zeigt, dass diese Schlüssel überwiegend durch Signaturketten an die MSCC angebunden sind, die aufgrund von widerrufenen Schlüsseln oder Schlüsselteilen unterbrochen sind. Dabei treten unter anderem Signaturen auf komplett widerrufenen Schlüsseln und Signaturen auf widerrufenen UserIDs auf. *wotsap* benutzt GnuPG, um die OpenPGP-Pakete eines Schlüssels zu parsen. Der Code zum Parsen der GnuPG-Ausgabe ist fehlerhaft und führt dazu, dass *wotsap* Widerrufssignaturen fälschlicherweise nicht beachtet.

Auf der anderen Seite sind ca. 3980 Schlüssel zwar in MSCC-1 vorhanden, nicht aber in MSCC-2. Um den Grund dafür zu finden, wurde für 10 zufällig ausgewählte Schlüssel aus dieser Menge eine Signaturkette (d.h. ein Pfad) zu einem Schlüssel gesucht, der sowohl in MSCC-1 als auch in MSCC-2 vorhanden ist. Ebenfalls wurde eine Kette in der anderen Richtung gesucht. Die Signaturen dieser Ketten wurden mittels GnuPG kryptographisch verifiziert, um sicherzustellen, dass sie gültig sind. Kann für beide Richtungen jeweils eine Kette erfolgreich verifiziert werden, so ist der betreffende Schlüssel per definitionem in der MSCC. Im vorliegenden Fall konnte das für alle Schlüssel der Stichprobe gezeigt werden. Der Fehler liegt also wiederum bei *wotsap*, dass diese Schlüssel fälschlicherweise ausschliesst. Als Ursache dafür ergab sich in allen betrachteten Fällen der fehlerhafte Bestand des Keyervers *wwwkeys.ch.pgp.net*, der von *wotsap* verwendet wird. Die Schlüssel wurden von *wotsap* nicht in die MSCC-2 übernommen, weil Teile der dazu notwendigen Signaturketten (komplette Schlüssel oder einzelne Signaturen) auf dem Keyserver nicht vorhanden sind. Besonders häufig traten dabei Schlüssel auf, deren Ablaufdatum durch eine neue Selbstsignatur verlängert wurde. Dieses Signaturpaket fehlte auf *wwwkeys.ch.pgp.net*, wodurch der betreffende Schlüssel fälschlicherweise als abgelaufen betrachtet wurde. Diese fehlenden Teile sind aber auf allen Keyservern des *SKS*-Verbundes vorhanden. Da die Ursache in allen untersuchten Fällen die gleiche war, kann angenommen werden, dass der Fehler systematisch ist und diesen Teil der Diskrepanz zwischen den Datensätzen vollständig erklärt. Als Ursache kommen eine mangelhafte Synchronisation zwischen *wwwkeys.ch.pgp.net* und dem *SKS*-Netzwerk sowie Fehler in der auf *wwwkeys.ch.pgp.net* verwendeten *PKS*-Version in Frage.

Die Natur der Fehler legt nahe, dass hauptsächlich solche Schlüssel fehlen bzw. fälschlich einbezogen wurden, die nur über eine sehr geringe Anzahl von redundanten Pfaden zur bzw. von der MSCC verfügen. Gäbe es mehr redundante Pfade, so

hätten einzelne fehlerhafte Informationen (fehlende bzw. fälschlicherweise als gültig betrachtete Schlüssel oder Signaturen) einen geringeren Einfluss.

Die Anzahl fehlender bzw. fälschlich einbezogener Schlüssel (ca. 9% fehlend, ca. 2% fälschlich einbezogen relativ zu MSCC-1) ist so gross, dass qualitative Fehler in der Graphenstruktur zu erwarten sind. Insbesondere Aussagen über die Verteilung von Knotengraden und ähnliche Aussagen anhand dieser Daten sind mit Vorsicht zu geniessen. Eine Reihe von Arbeiten verwendet die *wotsap*-Daten als Beispiel für ein empirisches soziales bzw. Small-World-Netzwerk [BrSc06] [HeGu09] [Dell07]. Sofern die Ergebnisse dieser Arbeit auf experimentellen Resultaten aufbauen, die mit diesen Daten gewonnen wurden, sollten sie daher mit korrekten Daten überprüft werden. Der Keyserver *wwwkeys.ch.pgp.net* sollte von Anwendern nicht mehr benutzt werden.

## 3.2 Design und Implementierung

Dieser Abschnitt beschreibt die im Rahmen dieser Arbeit erstellte Software zur Extraktion und Analyse des Web of Trust. Abschnitt 3.2.1 beschreibt zunächst die Keyserver-Software *SKS*, auf die der Extraktionsteil aufbaut. Abschnitt 3.2.2 beschreibt dann die entwickelte Software selbst und begründet das Design.

### 3.2.1 Der SKS Keyserver

SKS (Synchronizing Key Server) löst die bis dahin verbreitetste Keyserver-Software PKS (Public Key Server) ab und wird inzwischen auf so gut wie allen Keyservern benutzt. Mit *pgp.mit.edu* wurde im Juli 2009 der letzte grosse Keyserver auf SKS umgestellt. PKS benutzt ein auf E-Mails basierendes Protokoll zur Synchronisation zwischen Keyservern, das ausgesprochen ineffizient ist. Ausserdem kommt es mit einigen Bestandteilen des aktuellen OpenPGP-Standards wie z.B. *PhotoIDs* und mehreren Unterschlüsseln nicht zurecht. PKS-Keyserver können über ein Webinterface, ein E-Mail-Interface und das auf HTTP basierende HKP-Protokoll<sup>1</sup> abgefragt werden.

Im Unterschied dazu kommunizieren SKS-Keyserver zur Synchronisation direkt (ohne Umwege über Mailserver) miteinander. Dazu wird ein binäres „Gossip“-Protokoll benutzt, das auf einem effizienten Algorithmus zum Abgleich von Datensätzen beruht [MiTZ03]. SKS unterstützt alle Bestandteile des OpenPGP-Standards.

SKS ist in der Sprache Objective Caml (OCaml) implementiert und benutzt die Datenbank Berkeley DB als Datenablage. Ein Datenbankeintrag besteht dabei aus einem kompletten OpenPGP-Key, d.h. einer Reihe von OpenPGP-Paketen. SKS ist in zwei Prozesse aufgeteilt: der *db*-Prozess beantwortet Datenbank-Abfragen für die verschiedenen Abfragemöglichkeiten (Webinterface, HKP) während der *recon*-Prozess für den Abgleich der Datenbank mit anderen Keyservern zuständig ist.

### 3.2.2 Eigene Software

Die im Rahmen dieser Arbeit entwickelte Software besteht aus zwei Teilen: Der Extraktionsteil liest Schlüssel aus der Datenbank eines Keyservers aus, parst sie, reduziert sie dabei auf die benötigten Daten und speichert sie in einer Datei ab. Die reduzierten Daten werden in einer SQL-Datenbank abgelegt und können dort abgefragt werden. Der zweite Teil besteht aus einer Reihe von Werkzeugen zur Analyse dieser Daten entsprechend der Zielsetzung dieser Arbeit.

---

<sup>1</sup><http://tools.ietf.org/html/draft-shaw-openpgp-hkp-00>

### 3.2.2.1 Datenextraktion

Die Extraktion der Daten ist recht zeitaufwendig. Auf der zur Verfügung stehenden Hardware (FIXME Poolrechner Hardware) werden ca. XX Stunden benötigt. Durch die Aufteilung kann die Extraktion der Daten einmalig auf dem Keyserver vorgenommen werden, während die Daten anschliessend auf beliebigen Rechnern zur Verfügung stehen.

Der Extraktionsteil wurde direkt in die SKS-Software integriert. Dadurch ergeben sich mehrere Vorteile:

- Das Interface von SKS für den Datenbankzugriff kann wiederverwendet werden
- SKS enthält einen rudimentären OpenPGP-Parser, d.h. eine Reihe von Funktionen, die die Byte-Struktur einzelner OpenPGP-Pakete sowie die Paket-Struktur des Schlüssels parsen und die darin enthaltenen Informationen in Datenstrukturen zugänglich machen. Durch die Verwendung dieser Funktionen kann auf eine eigene Implementierung eines OpenPGP-Parsers verzichtet werden.
- Die in SKS definierten Datenstrukturen für die Auswertung von OpenPGP-Paketen und weitere Hilfsfunktionen, beispielsweise für den Umgang mit OpenPGP-Fingerprints, können verwendet und erweitert werden

Derzeit läuft der Extraktionsteil in Form eines eigenen Prozesses, der einmalig den kompletten Datenbestand ausliest. Dieser Teil könnte aber auch direkt in den *db*-Prozess integriert werden, so dass der laufende Keyserver konstant neue oder geänderte Schlüssel reduziert und in der SQL-Datenbank ablegt. Auf diese Weise könnte ein ständig aktueller Datenbestand ohne den Aufwand des vollständigen Auslesens realisiert werden. Diese Möglichkeit wurde im Rahmen dieser Arbeit nicht verfolgt, kann aber mit geringem Aufwand realisiert werden.

Da der Extraktionsprozess nur lesend auf die SKS-Datenbank zugreift, kann er die Daten auslesen, während die *recon*- und *db*-Prozesse laufen. Eine Unterbrechung des SKS-Betriebes ist nicht notwendig.

Als Sprache für die Implementierung wurde OCaml ausgewählt. Im Extraktionsteil bestand dafür aufgrund der Integration in SKS keine Wahl. Der Analyseteil wurde ebenfalls in OCaml implementiert. Da die dort verwendeten Daten ausschliesslich aus der SQL-Datenbank stammen und dort nur primitive Datentypen (Strings, Ganz- und Fliesskommazahlen) gespeichert werden, können aber für diesen Teil auch problemlos andere Sprachen verwendet werden.

Der Extraktionsteil iteriert über alle in der SKS-Datenbank enthaltenen Schlüssel. Jeder Schlüssel wird durch die in SKS enthaltenen Funktionen geparsst. Der Extraktionsteil FIXME muss dann noch

- entscheiden, ob der Schlüssel zurückgezogen oder abgelaufen ist. Dazu werden die Selbstsignaturen auf den einzelnen UserIDs betrachtet. Falls zu einer UserID mehrere Selbstsignaturen vorliegen, wird entsprechend der Empfehlung im OpenPGP-Standard nur die aktuellste verwendet. Liegt ein Rückrufzertifikat vor oder ist der Schlüssel abgelaufen, wird das Ablauf- bzw. Rückzugsdatum gespeichert.

- Fremdsignaturen sammeln. Dazu wird jede UserID betrachtet und die darauf angebrachten Signaturen ohne Duplikate gespeichert. Hier wird ebenfalls nur die neueste Signatur verwendet. Handelt es sich bei der neuesten Signatur um eine Rückrufsignatur, wird das Datum des Rückrufs gespeichert und zusätzlich die nächstältere Signatur gesucht, auf die sich der Rückruf bezieht.
- den Schlüssel und jede Fremdsignatur auf die benötigten Daten reduzieren. Für den Schlüssel sind dies (falls vorhanden) Rückzugs- und Ablaufdatum, die KeyID, die Liste aller UserIDs und die Information, welche davon die primäre UserID ist, das Erstellungsdatum des Schlüssels und der verwendete Public-Key-Algorithmus sowie dessen Schlüssellänge.

Eine Fremdsignatur wird reduziert auf die KeyID des Unterzeichners, den Zertifizierungslevel, das Erstellungsdatum der Signatur, falls vorhanden Ablauf- und Rückzugsdatum, den verwendeten Signaturalgorithmus und den verwendeten Public-Key-Algorithmus.

Grundsätzlich werden Schlüssel nur dann verworfen, wenn sie nicht parsebar, d.h. keine Abfolge gültiger OpenPGP-Pakete sind. Schlüssel, die zurückgezogen oder abgelaufen sind, werden unter Angabe des jeweiligen Datums trotzdem gespeichert. Auf diese Weise ist sichergestellt, dass der Datensatz möglichst vollständig ist. Für einen beliebigen Zeitpunkt stehen alle dann gültigen Schlüssel und Signaturen zur Verfügung. Es kann also gewissermaßen ein „Snapshot“ des Web of Trust zu einem beliebigen Zeitpunkt analysiert werden.

Sind alle Schlüssel extrahiert, werden noch solche Signaturen entfernt, deren erstellender Schlüssel im Datenbestand nicht vorhanden ist. Außerdem werden für Signaturen, die von einem Unterschlüssel erstellt wurden, die KeyID des signierenden Schlüssels auf die des Hauptschlüssels geändert. Dies ist notwendig, weil das hier verwendete Datenmodell keine Informationen über Unterschlüssel enthält.

Erwähnt werden muss, dass der Parse-Vorgang nur prüft, ob die Paketfolge eines Schlüssels dem OpenPGP-Standard entspricht. Es wird keinerlei kryptographische Verifizierung der Selbst- und Fremdsignaturen eines Schlüssels vorgenommen. Grundsätzlich können ohne Probleme Signaturpakete auf einem Schlüssel angebracht und diese auf einem Keyserver veröffentlicht werden, auch wenn der Ersteller nicht über das private Schlüsselmaterial für die kryptographische Signatur verfügt. Keyserver verifizieren keine Signaturen, so dass der Signaturteil des Pakets mit beliebigem Inhalt gefüllt werden kann. GnuPG und PGP verifizieren natürlich Signaturen, so dass eine solches Signaturpaket dort nicht verwendet wird und damit kein wirkliches Angriffspotential bietet. Eine Möglichkeit für den Extraktionsteil bestünde darin, die Signaturen jedes OpenPGP-Schlüssels mit GnuPG verifizieren zu lassen. Dagegen sprechen zwei Gründe: Einerseits wäre der Aufruf von GnuPG sehr zeitaufwendig. Für jeden Schlüssel müsste der Schlüssel in einen GnuPG-Schlüsselring eingefügt werden. Zusätzlich müssten noch alle Schlüssel, die den jeweiligen Schlüssel signiert haben, einzeln in der SKS-Datenbank gesucht und in den Schlüsselring eingefügt werden, um die Signaturen verifizieren zu können. Letztendlich kostet der Aufruf von GnuPG selbst Zeit. Für die Anzahl der hier verarbeiteten Schlüssel (2,6 Millionen) scheint dieser Ansatz daher ungeeignet. Sicherlich sind Schlüssel mit defekten Signaturpaketen auf den Keyservern vorhanden. Allerdings müsste deren Anzahl sehr gross sein, um signifikante Änderungen in der Struktur des Graphen und den

statistischen Auswertungen der Schlüsseleigenschaften zu erreichen. Eine grosse Zahl solcher Pakete scheint aber unwahrscheinlich, da zumindest unter Sicherheitsaspekten ein Angreifer damit keinen offensichtlichen Gewinn erreichen kann.

Die so extrahierten Daten werden in einer SQL-Datenbank (PostgresQL) abgelegt. Auf diese Weise kann darauf verzichtet werden, eigene Selektionsmechanismen zu implementieren. Stattdessen kann die gewünschte Datenmenge einfach als SQL-Abfrage formuliert werden. Auf diese Weise ergibt sich eine deutlich flexiblere Abfragemöglichkeit. Die Ablage in der Datenbank ist zwar etwas langsamer als die Daten im Speicher zu halten. Andererseits müssen die Daten aber nicht jedesmal komplett in den Speicher geladen und dort gehalten werden. Außerdem werden effiziente Indexstrukturen der Datenbank genutzt und müssen nicht selbst implementiert werden.

### 3.2.2.2 Datenauswertung

Die Datenauswertung besteht aus einer Reihe von unabhängigen Werkzeugen, die die jeweiligen Analyseaufgaben implementieren. Diese sind als Kommandozeilenprogramme in separaten Prozessen realisiert. Eine Übersicht über die Aufteilung der Aufgaben auf die einzelnen Werkzeuge findet sich in Anhang B.

Die zur Community-Analyse verwendeten Algorithmen (siehe Abschnitt 3.3) wurden aus Zeitgründen nicht selbst implementiert. Für Fastmod FIXME wurde die Implementierung aus dem Softwarepaket igraph[CsNe06] verwendet. Für die restlichen Algorithmen wurden Implementierungen der jeweiligen Autoren verwendet.

## Parallelisierung mit MPI

Die in Abschnitt 2.3.2 definierten Kennzahlen Eccentricity, Durchmesser, Radius,  $h$ -Nachbarschaften und charakteristische Distanz ergeben sich sämtlich aus den Distanzen in einem Graphen. Der übliche Ansatz zur Berechnung dieser Kennzahlen ist, die Distanzmatrix

$$D = (d(u, v))_{u, v \in V} \quad (3.1)$$

zu berechnen, die die Distanz  $d(u, v)$  in Zeile  $u$  und Spalte  $v$  enthält[BrSc04]. Aus dieser Matrix aller Distanzen im Graphen lassen sich alle oben angeführten Kennzahlen berechnen.  $D$  kann berechnet werden, indem das single-source-shortest-path-Problem (SSSP) für alle  $n$  Knoten oder das all-pairs-shortest-path-Problem (APSP) gelöst werden. Für den vorliegenden Fall eines ungewichteten Graphen lässt sich SSSP in  $\mathcal{O}(n)$  durch Breitensuche lösen. Für die Berechnung der Distanzmatrix ergibt sich dann eine Komplexität von  $\mathcal{O}(nm)$ .

Dieser Ansatz ist hier aufgrund der Grösse des vorliegenden Graphen nicht geeignet. Für diesen gilt  $m > n$ , so dass die Berechnung quadratisch in der Anzahl der Knoten ist. Das sequentielle Lösen von SSSP für jeden Knoten auf der zur Verfügung stehenden Hardware hätte einen Zeitaufwand von ca. 32 Stunden erfordert. Das Vorhalten der quadratischen Distanzmatrix erfordert eine erhebliche Menge Speicher (ca. 8 GiB, wenn Distanzen durch 4-Byte-Ganzzahlen repräsentiert werden).

Allerdings ist es nicht notwendig, die gesamte Distanzmatrix vorzuhalten. Eccentricity,  $h$ -Nachbarschaften und die durchschnittliche Distanz eines Knotens sind bestimmbar, wenn SSSP *für diesen Knoten* gelöst wurde. Durchmesser und Radius ergeben sich allein aus den Eccentricity-Werten für alle Knoten. Die Berechnung

kann also parallelisiert werden, indem die Knotenmenge  $V$  in Teilmengen  $V_1, \dots, V_k$  aufgeteilt wird, die jeweils einer Berechnungseinheit zugewiesen werden. Jede Berechnungseinheit  $i$  löst dann SSSP für alle Knoten aus  $V_i$ . Die Ergebnisse müssen dann nur noch kombiniert werden, um Radius, Durchmesser und charakteristische Distanz des Graphen zu bestimmen.

Dieser Ansatz wurde auf einem Cluster realisiert, dessen Knoten mittels Message-Passing-Interface (MPI) kommunizieren. Eine zentrale *Master*-Instanz zerlegt die Knotenmenge des Graphen und schickt jeder der  $k$  *Worker*-Instanzen die Menge  $V_k$ , für die diese zuständig ist. Jede *Worker*-Instanz  $k$  löst SSSP für alle Knoten aus  $V_k$  sequentiell und berechnet damit die gesuchten Kennzahlen. Die Ergebnisse werden an die *Master*-Instanz übermittelt, die sie kombiniert.

Da die Berechnung für die einzelnen Knotenmengen  $V_k$  komplett unabhängig läuft, beschränkt sich die notwendige Kommunikation zwischen den Instanzen auf das Zuteilen der Knotenmenge zu einer *Worker*-Instanz und die Übertragung der Ergebnisse zu der zentralen *Master*-Instanz. Dieser Ansatz skaliert also annähernd linear in der Anzahl der Berechnungseinheiten.

Die Berechnung wurde mit 24 *Worker*-Instanzen durchgeführt, wobei eine *Worker*-Instanz auf einem Core einer Quad-Core-Xeon-CPU ausgeführt wird. Insgesamt wurden also 7 Clusterknoten benutzt. Die notwendige Zeit für das Berechnen sämtlicher oben angeführter Kennzahlen reduzierte sich damit auf ca. 20 Minuten.

### 3.3 Community-Analyse

In Abschnitt 2.2.5 wurden Mechanismen beschrieben, die zur Entstehung des Web of Trust beitragen. Ausgehend davon kann die Annahme getroffen werden, dass die Vernetzung wesentlich von zwei Faktoren beeinflusst wird: Teilnehmer vernetzen sich einerseits aufgrund ihrer Zugehörigkeit zu einer *sozialen Gruppe*. Dabei kann es sich um Open-Source-Projekte wie z.B. Debian, akademische Einrichtungen und Firmen handeln, aber auch um eine Gruppe von Freunden oder Bekannten. Andererseits vernetzen sich Teilnehmer auf Keysigning-Parties mit einer relativ grossen Anzahl Benutzer, die ihnen nicht unbedingt bekannt sind und mit denen sie keine starke Gruppenzugehörigkeit verbindet. Offensichtlich ist allerdings, dass diese Mechanismen sich nicht gegenseitig ausschliessen. Die bereits in 2.2.5 beschriebene Bandbreite von Keysigning-Parties reicht von Ad-hoc-Veranstaltungen mit wenigen Teilnehmern, die durchaus der selben Institution angehörig sein können, bis zu grossen, formalisierten Veranstaltungen auf Konferenzen und Messen mit etlichen Teilnehmern.

Es soll nun untersucht werden, inwiefern sich diese postulierten Entstehungsmechanismen in der Struktur, also den topologischen Eigenschaften, des Web of Trust wiederfinden. Angenommen wird, dass sich ein hoher Anteil der Teilnehmer primär anhand dieser beiden Mechanismen vernetzen und der Anteil an Signaturen zwischen Teilnehmern, die nicht durch eine Keysigning-Party oder eine soziale Gruppe entstanden ist, deutlich geringer ist. In diesem Fall ist zu erwarten, dass sich soziale Gruppen und die Ergebnisse von Keysigning-Parties als Gruppen von Knoten im Netzwerk wiederfinden, die untereinander eine hohe Anzahl von Kanten hat, d.h. *dicht* vernetzt ist, während sich zu Knoten ausserhalb der Gruppe nur relativ wenige

Kanten finden. Diese Eigenschaft entspricht exakt der in Abschnitt 2.3.5 beschriebenen Definition von *Communities*. Es erscheint als Methode deshalb sinnvoll, mit vorhandenen Methoden zur Community-Erkennung eine Zerlegung des Graphen zu berechnen und dann zu untersuchen, inwiefern sich die einzelnen so berechneten Communities entsprechend der Annahme auf soziale Gruppen bzw. Keysigning-Parties abbilden lassen. Beispielsweise könnten sich eine (oder mehrere) Communities ergeben, die von Mitgliedern des Debian-Projekts dominiert werden.

Dazu sind Kriterien nötig, um eine berechnete Community einer sozialen Gruppe, einer Keysigning-Party oder beidem zuzuordnen. Für Keysigning-Parties ist anzunehmen, dass die Signaturvorgänge zwischen den Teilnehmern einer KSP innerhalb eines eng begrenzten Zeitraums nach Stattfinden der KSP vorgenommen werden. Als einfaches Kriterium für die Erkennung einer Keysigning-Party kann also eine starke zeitliche Korrelation der Signaturvorgänge zwischen der Mehrzahl der Community-Mitglieder verwendet werden. Konkret wird hier eine Community als Produkt einer KSP betrachtet, wenn 80% der Signaturen zwischen den Mitgliedern innerhalb eines Monats vorgenommen wurden.

Die Zuordnung zu sozialen Gruppen wird dadurch erschwert, dass – abgesehen von den Schlüsseln und Signaturen selbst – keine empirischen Daten über die Gruppenzugehörigkeit vorliegen. Die einzigen Daten, die eine (primitive) Zuordnung erlauben, sind die in den UserIDs enthaltenen E-Mail-Adressen. Über die Top-Level-Domain (TLD) kann ein User einem Land zugeordnet werden, sofern es sich nicht um eine der generischen TLDs (com, org, net) handelt. Über die Second-Level-Domain kann versucht werden, einen Nutzer einer Institution zuzuordnen. Ein User, der beispielsweise über eine E-Mail-Adresse mit der Domain debian.org verfügt, ist sicherlich ein Mitglied des Debian-Projekts. Hilfreich dabei ist, dass Teilnehmer dazu tendieren, alle ihre Adressen auf ihren Schlüsseln zu vermerken, so dass viele Schlüssel mehrere UserIDs und E-Mail-Adressen haben (siehe Abschnitt 4.3). Dadurch werden die verfügbaren Informationen über Gruppenzugehörigkeiten erhöht. Adressen verbreiteter “Freemail”-Anbieter wie Gmail, GMX und Yahoo sowie Adressen von Internet Service Providern dürfen dabei natürlich nicht verwendet werden, weil ihre Verwendung nichts über eine Gruppenzugehörigkeit aussagt.

Ähnlich wie für Keysigning-Parties werden wieder einfache Schwellwerte verwendet, um eine Community einer sozialen Gruppe, also einer Domain, zuzuordnen: Eine Community kann einer Domain *zugeordnet* werden, wenn mindestens 80% der Mitglieder eine UserID mit dieser Domain haben. Außerdem wird eine Community von einer Domain *dominiert*, wenn ein erheblicher Anteil – konkret mindestens 40% – der Mitglieder eine UserID mit dieser Domain haben.

Die Schwellwerte wurden nicht anhand empirischer Daten festgelegt sondern drücken nur ein intuitives Verständnis für die überwiegende Mehrzahl der Mitglieder einer Gruppe bzw. einen erheblichen Anteil der Mitglieder einer Gruppe aus.

Wie bereits in Abschnitt 2.3.5 beschrieben, existiert – insbesondere für ungerichtete Graphen – eine Vielzahl von Methoden zur Zerlegung eines Graphen in Communities. Dabei hat sich bislang kein “bester” Algorithmus herauskristallisiert, auch wenn einige Algorithmen für eine Reihe von Benchmark-Graphen bessere Ergebnisse liefern als andere [LaFo09]. Das übliche Gütemass zum Vergleich von Community-Zerlegungen eines Graphen, dessen tatsächliche Community-Struktur nicht bekannt

ist, ist der Modularity-Wert (siehe Abschnitt 3.3). Um zu vermeiden, dass eine vergleichsweise schlechte Einteilung durch einen Algorithmus die Ergebnisse verfälscht, wurden mehrere Algorithmen zur Community-Erkennung auf ungerichteten Graphen verwendet, um ihre Ergebnisse vergleichen zu können: Fast-Modularity (FM) von Newman et al., sowie der Algorithmus von Blondel et al. (LP) (siehe Abschnitt 2.3.5). Die Auswahl wurde aus folgenden Gründen getroffen:

- Fast-Modularity war einer der ersten praktikablen Algorithmen zur Community-Erkennung in grossen Netzwerken und wird weiterhin häufig verwendet. Allerdings liefert er in der Literatur durchgängig schlechtere Ergebnisse als moderne Algorithmen [Fort10] und wird deshalb hier nur als Vergleichsbasis verwendet.
- Der Algorithmus von Blondel et al. erzielte im Vergleich von Lancichinetti et al. [LaFo09] mit die besten Ergebnisse.

Ein weiteres Auswahlkriterium war, dass für diese Algorithmen Implementierungen öffentlich mit Sourcecode verfügbar sind. Ausserdem existieren Algorithmen, die zwar für kleinere Graphen bessere Ergebnisse liefern als die hier verwendeten, aufgrund ihrer Berechnungskomplexität für grosse Graphen wie den hier vorliegenden aber nicht geeignet sind.

Eine Zuordnung von einer Person zu genau einer sozialen Gruppe scheint unrealistisch. Statt dessen ist davon auszugehen, dass eine Person im Allgemeinen zu mehreren Institutionen oder Gruppen zugehörig ist. Ebenso ist es möglich, dass sie sich einerseits im Kontext einer sozialen Gruppe vernetzt, andererseits aber auch an Keysigning-Parties teilnimmt. Eine solche multiple Zugehörigkeit sollte sich im Netzwerk durch eine Mitgliedschaft eines Knotens in mehreren Communities ausdrücken. Wie in 2.3.5 erwähnt, existiert eine Klasse von Algorithmen, die ein Netzwerk in *überlappende* Communities zerlegt, so dass ein Knoten Teil von mehreren Communities sein kann. Um zu untersuchen, inwiefern dieser Überlappungseffekt die Zuordnung von Knoten zu sozialen Gruppen und KSPs beeinflusst, wurde zusätzlich noch eine überlappende Zerlegung mit dem Algorithmus COPRA [Greg10], einer Verallgemeinerung von Label-Propagation auf berechnet. Dabei handelt es sich um eine Verallgemeinerung des bereits erwähnten Algorithmus Label-Propagation. COPRA erwartet einen Parameter  $v$ , der angibt, zu wie vielen Communities ein Knoten gehören kann. Um hier das Optimum bezüglich der Modularity zu finden, wurden Zerlegungen mit Werten  $v = 2, \dots, 15$  berechnet. Zum Vergleich wurde eine Zerlegung mit  $v = 1$  berechnet, die keine Überlappung erlaubt. Da COPRA nicht-deterministisch ist, wurden für jeden Wert 10 Berechnungen durchgeführt, um die Abweichung zu ermitteln.

Während Modularity die Güte einer Zerlegung nur bezogen auf die Struktur des Graphen misst, drücken die Zuordnungszahlen die Güte einer Partitionierung in Bezug auf die oben formulierte Hypothese über die Entstehung des Netzwerks aus. Wenn diese Hypothese zutrifft, ist eine Zerlegung in Bezug darauf “besser”, wenn eine grössere Anzahl von Communities einer sozialen Gruppe oder Keysigning-Party zugeordnet werden kann. Es kann nicht vorrausgesetzt werden, dass eine Zerlegung mit hoher Modularity automatisch auch eine Zerlegung mit hohen Zuordnungszahlen ist, dass sich also die real existierenden Communities direkt in der Struktur des Graphen wiederfinden.

### 3.3.1 Communities in gerichteten Graphen

Der Graph des Web of Trust ist inhärent gerichtet, da eine Signatur nur in eine Richtung gilt. Die meisten verfügbaren Methoden zur Community-Erkennung setzen im Gegensatz dazu einen ungerichteten Graphen voraus. Methoden, die auf gerichteten Graphen arbeiten, sind erst in jüngster Zeit vorgestellt worden [LeNe08] [RoBe08]. Der bis jetzt am häufigsten verwendete Ansatz für die Community-Erkennung in gerichteten Netzwerken ist es, die Richtung der Kanten zu ignorieren und den Graphen als ungerichteten Graphen zu behandeln. Auch die Arbeiten, die den Web of Trust-Graphen unter Community-Gesichtspunkten analysieren oder ihn als Benchmark für den Vergleich von Algorithmen verwenden, gehen so vor [BnPSDGA04]. Auch wenn dieses Vorgehen akzeptable Resultate liefern kann, ist doch klar, dass durch das Ignorieren der Richtung ein Informationsverlust stattfindet, der die Ergebnisse negativ beeinflussen kann. In der vorliegenden Arbeit wurde dieser übliche Ansatz des Ignorierens der Kantenrichtung verwendet, da die Methoden für ungerichtete Netzwerke besser etabliert und Implementationen dieser Algorithmen einfacher zugänglich sind. Im Fall des Web of Trust ist allerdings ein erheblicher Anteil der verbundenen Knotenpaare nur in eine Richtung verbunden, d.h. es existiert keine “Gegenkante” (siehe Abschnitt 4.2). Damit ist auch der Informationsverlust durch die Reduzierung erheblich. Es ist daher naheliegend, dass eine Community-Zerlegung des Graphen unter Berücksichtigung der Richtungsinformationen eine Zerlegung berechnen könnte, die die Struktur des Graphen besser wiedergibt und auch unter dem Gesichtspunkt der Zuteilung zu sozialen Gruppen und Keysigning-Parties bessere Resultate liefert. Um den Einfluss der Richtung zu überprüfen, wurde eine Zerlegung mit dem Algorithmus “infomap” von Rosvall et al. [RoBe08] berechnet.

# 4. Ergebnisse und Diskussion

In Abschnitt 4.1 wird zunächst der berechnete Datensatz allgemein charakterisiert.

## 4.1 Datensatz

Die Extraktion der hier verwendeten Daten wurde, wie in Abschnitt 3.2.2.1 beschrieben, auf einem vom Verfasser betriebenen Keyserver am 02.12.2009 vorgenommen. Die Datenbank des Keyservers war zu diesem Zeitpunkt mit dem Rest des Keyserver-Netzwerkes vollständig abgeglichen. Der berechnete Datensatz enthält 2.725.504 Schlüssel und 1.145.337 zugehörige Signaturen, wobei hier keine Selbstsignaturen enthalten sind. 52.000 Schlüssel wurden während der Extraktion als defekt verworfen. Eine Stichprobe von 30 verworfenen Schlüssel ergab, dass diese auch von GnuPG nicht akzeptiert werden, weil sie entweder über keine UserID-Pakete verfügen oder aus anderen Gründen keine standardkonformen OpenPGP-Schlüssel darstellen. Von den nicht-defekten Schlüsseln sind 417.163 Schlüssel abgelaufen und 100.071 Schlüssel wurden zurückgezogen. Die in Relation zur Anzahl der Schlüssel niedrige Anzahl von Signaturen weist schon darauf hin, dass ein erheblicher Teil der (gültigen) Schlüssel nicht oder kaum vernetzt ist. In der Tat verbleiben nach Abzug von Schlüsseln, die weder ein- noch ausgehende Signaturen haben, nur 325.410 gültige Schlüssel und 816.785 zugehörige gültige Kanten, die den Graphen ausmachen.

Die grosse Mehrheit der im Keyserver-Netzwerk vorhandenen Schlüssel ist also komplett unvernetzt und nimmt von vorneherein nicht am Web of Trust teil. Die Überprüfung der Authentizität dieser Schlüssel anhand öffentlich verfügbarer Informationen ist nicht möglich. Die Besitzer dieser Schlüssel haben – sofern sie die Schlüssel überhaupt einsetzen – ebenfalls keine Möglichkeit, die Authentizität anderer Schlüssel zu prüfen. Über die Gründe für diese geringe Vernetzung kann hier nur spekuliert werden. Möglich ist etwa, dass die Benutzer schlichtweg keine Notwendigkeit in der Authentifizierung von Schlüsseln sehen, weil ihnen Man-in-the-middle und ähnliche Angriffe nicht bekannt sind, oder das im Web of Trust verwendete Modell zur Verifizierung von Schlüsseln zu komplex erscheint. Es kann selbstverständlich nicht ausgeschlossen werden, dass die Verifizierung über Signaturen läuft, die nicht auf öffentliche Keyserver geladen wurden. Beachtet werden muss auch, dass der Datensatz

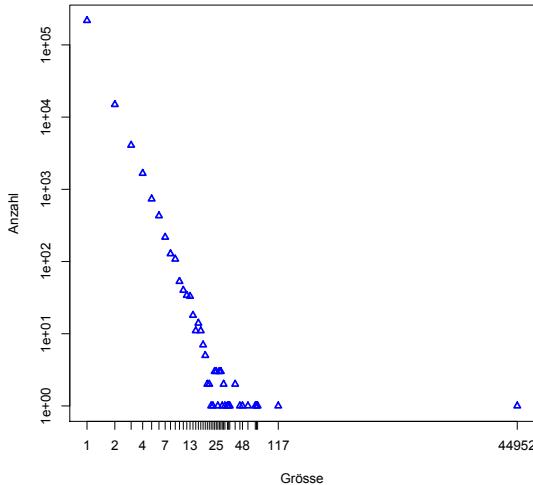


Abbildung 4.1: Grössenverteilung der starken Zusammenhangskomponenten

keine präzise Angabe über die momentane Anzahl von PGP-Benutzern erlaubt, da sich die Schlüssel über einen Zeitraum von 20 Jahren angesammelt haben.

## 4.2 Allgemeine Merkmale des Netzwerkes

### 4.2.1 Starke Zusammenhangskomponenten

Der Graph wurde zunächst in seine 240.382 starken Zusammenhangskomponenten zerlegt. Die Verteilung der Komponentengrößen in Abbildung 4.1 zeigt dabei zwei Extreme: Es existiert eine einzelne gigantische Komponente mit ca. 45000 Knoten. Demgegenüber steht eine geringe Anzahl von kleinen Komponenten bis zur Grösse 3 , insbesondere aber über 100.000 einelementige Komponenten, also einzelne Knoten, die nur *entweder* ein- oder ausgehende Kanten haben, und über 10.000 zweielementige Komponenten, also durch zwei Kanten verbundene Knotenpaare. Ein erheblicher Anteil der überhaupt vernetzten Knoten ist also wiederum nur sehr wenig vernetzt.

Abbildung 4.2 zeigt die Struktur der Zusammenhangskomponenten bis zur Grösse 6, die mit anderen Komponenten verbunden sind. Es ergibt sich eine sternförmige Struktur, bei der die kleineren Komponenten fast ausschliesslich mit der grössten Komponente verbunden sind und es so gut wie keine weitere Vernetzung gibt. Der Aufwand, der nötig ist, um zwei starke Zusammenhangskomponenten verschmelzen zu lassen, ist sehr gering: dazu ist nur genau eine Kante in jede Richtung nötig. Je mehr Kanten (also Signaturen) in eine Richtung verlaufen, desto wahrscheinlicher ist es, dass es zu einer dieser Kanten auch eine Gegenkante, also eine entsprechende Signatur in Gegenrichtung, gibt. Es kann also davon ausgegangen werden, dass die Komponenten nur durch sehr wenige Kanten, die eben nur in eine Richtung verlaufen können, verbunden sind. Daraus ergibt sich, dass die Mitglieder der kleinen Komponenten nur eine sehr geringe Signaturaktivität aufweisen können. Würde eine nennenswerte Anzahl von Signaturen entstehen, so wäre die Wahrscheinlichkeit für das Verschmelzen von Komponenten sehr hoch. Die Anzahl kleiner Komponenten müsste in diesem Fall deutlich geringer sein.

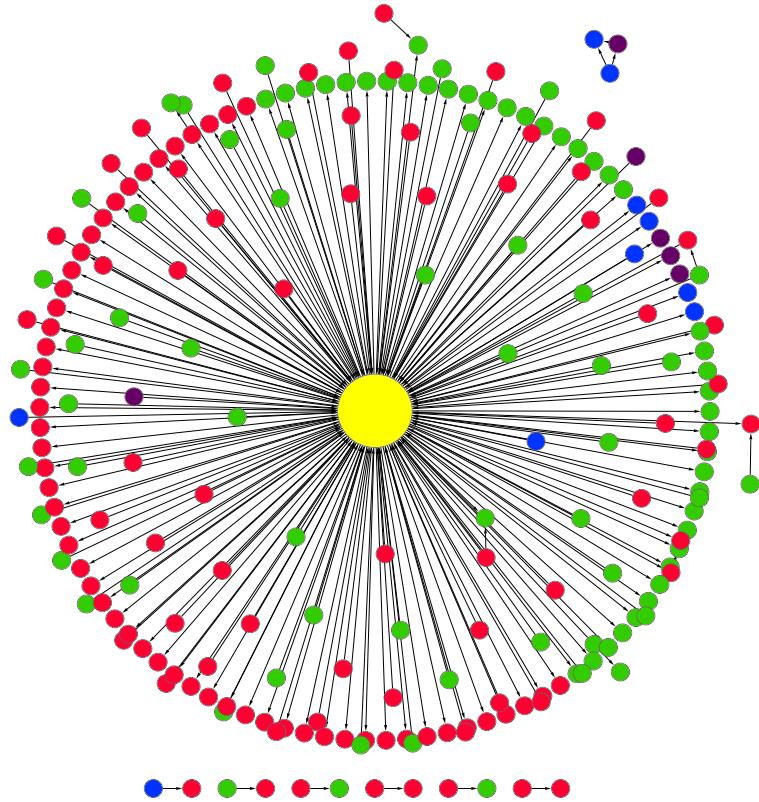


Abbildung 4.2: Sternförmige Struktur der starken Zusammenhangskomponenten bis zur Grösse 6 (rot = Grösse 6-10, grün = Grösse 11-20, blau = Grösse 21-30, violett = Grösse  $\geq 31$ , gelb = MSCC)

Die *Nützlichkeit* des Web of Trust für die Teilnehmer, deren Schlüssel nicht in der grössten starken Zusammenhangskomponente enthalten sind, ist gering: Die Menge der Schlüssel, die von einem Teilnehmer anhand von Signaturketten verifiziert werden kann, beschränkt sich zunächst auf die eigene Komponente und ist damit sehr klein. Zwar gibt es auch – wie oben gezeigt – Vernetzung zwischen kleineren Komponenten und der grössten Komponente. Es existieren ca. 18000 Knoten, die Knoten aus der grössten Komponente direkt über eine einzige Kante erreichen können. Allerdings muss hier beachtet werden, dass das in PGP/GnuPG verwendete Verifizierungsmodell die Länge der verwendbaren Signaturketten begrenzt – in der Standardeinstellung von GnuPG auf die maximale Länge 5 (siehe Abschnitt 2.2.3). Schon innerhalb der grössten Komponente sind viele kürzeste Pfade länger als dieses Maximum (siehe Abschnitt 4.2.2). Durch den zusätzlichen Schritt zu einem Knoten innerhalb der grössten Komponente verlängert sich der Pfad. Insbesondere wenn es sich bei diesen Knoten um schwach vernetzte Knoten handelt, die am “Rand” der grössten Komponente liegen, ist die Menge der auf Pfaden benutzbarer Länge erreichbaren Knoten beschränkt. Kann die grösste Komponente nur indirekt über andere Knoten erreicht werden, reduziert sich diese Menge noch weiter. Gleiches gilt auch für die Erreichbarkeit der Menge von ca. 92000 Knoten, zu denen eine Kante von Knoten der grössten Komponente aus besteht. Beachtet werden muss allerdings die Existenz von zentralen Certificate Authorities im eigentlich dezentralen Web of Trust. Mit den drei seit 1997 im Rahmen der “Krypto-Kampagne” der Zeitschrift c’t (siehe Abschnitt 2.2.5) verwendeten Zertifizierungsschlüsseln wur-

den insgesamt 23813 derzeit gültige Schlüssel unterschrieben. Von diesen liegen nur 2578 Schlüssel innerhalb der grössten Komponente. Wenn also nur dem verwendeten Zertifizierungsprozess und damit diesen drei Zertifizierungsschlüsseln vertraut wird, sind immerhin etwa 20000 Schlüssel aussserhalb der grössten Komponente verifizierbar. Es zeigt sich hier auch die Flexibilität des Web of Trust-Konzeptes, dass die Integration von zentralen Komponenten in das eigentlich dezentrale Netz erlaubt.

Insgesamt lassen diese Daten den Schluss zu, dass ausschliesslich die grösste starke Zusammenhangskomponente Schlüssel von Personen enthält, die sich durch regelmässige Signaturaktivitäten am Web of Trust beteiligen und es zur Verifizierung von Schlüsseln benutzen.

#### 4.2.2 Netzwerkstatistiken der grössten starken Zusammenhangskomponente

Die weitere Untersuchung der Topologie des Graphen konzentriert sich auf die grösste starke Zusammenhangskomponente. Starke Zusammenhangskomponenten als Einheit der Betrachtung machen Sinn, weil die Verwendung des Graphen für die Verifizierung von Schlüsseln gerade die Erreichbarkeit voraussetzt und weil eine Reihe von Netzwerkstatistiken nur definiert sind, wenn zwischen allen Paaren von Knoten Pfade existieren. Da im vorherigen Abschnitt argumentiert wurde, dass die grösste starke Zusammenhangskomponente die einzige ist, die in grösserem Massstab vernetzt ist und in der Signaturaktivitäten stattfinden, scheint eine Untersuchung der restlichen Komponenten nicht sinnvoll.

Der induzierte Teilgraph der grössten starken Zusammenhangskomponente besteht aus 44.952 Knoten und 442.960 gerichteten Kanten. Dieser Teilgraph ist deutlich grösser als die bisher in der Literatur verwendeten PGP-Netzwerke: ein gerichtetes Netzwerk von ca. 12.000 Knoten bei [CaBH02] und ein Netzwerk von 10.700 Knoten, bei dem alle einseitigen Signaturen gelöscht wurden bei [BnPSDGA04] und [Greg10]. Beide Netzwerke stellen die grösste starke Zusammenhangskomponente dar und wurden im Jahr 2001 extrahiert.

##### 4.2.2.1 Gegenseitigkeit von Kanten

Der *Reciprocity*-Wert eines gerichteten Graphen gibt den Anteil von Kanten an, die in beide Richtungen verlaufen. Für die grösste Komponente ist dieser Wert 0,510. Das bedeutet, dass es für eine gegebene Kante eine Chance von 51% gibt, dass eine entsprechende “Gegenkante” in umgekehrter Richtung existiert. Die Reciprocity eines zufälligen Graphen, der über die *gleiche Gradsequenz* wie die grösste Komponente verfügt, hat im Vergleich eine Reciprocity von nur 0,006. Dass der reale Wert deutlich grösser ist, ist nicht verwunderlich: Das Web of Trust ist eben kein Produkt eines zufälligen Prozesses, sondern von konkreten Mechanismen, nämlich gegenseitigen Signierungen von Schlüsseln. Eher erstaunlich ist, dass der Wert nicht noch höher ist, dass also nicht noch mehr Signaturen auf Gegenseitigkeit beruhen. In Abschnitt 2.2.5 wurde dargestellt, dass die Signierung bei Keysigning-Parties und privaten Treffen üblicherweise gegenseitig verläuft, dass also beide Signaturpartner den Schlüssel des jeweils anderen unterschreiben. Es ist eine Reihe von Gründen denkbar, die dazu führen, dass der Anteil einseitiger Signaturen so hoch ist: Eine Rolle könnten dabei Certificate Authorities spielen. Diese signieren zwar eine Vielzahl von Schlüsseln,

empfangen aber selbst weniger Signaturen<sup>1</sup>. Ausserdem können Signaturen aus verschiedenen Gründen nicht vorgenommen werden oder nicht auf Keyserver hochgeladen werden. Schlussendlich kann nicht davon ausgegangen werden, dass hinter jeder Signatur ein sinnvoller Prozess der Identitätsüberprüfung steht. Teilnehmer könnten etwa andere Schlüssel „zum Experimentieren“ unterschreiben. PGP und GnuPG zeigen eine Warnung an, wenn die E-Mail-Signatur eines Schlüssels überprüft wird, der nicht als *valide* eingestuft wird. Teilnehmer könnten dann versucht sein, Schlüssel zu unterschreiben, deren Authentizität sie nicht überprüft haben, um diese Warnung zu unterdrücken, selbst wenn dadurch der Sinn des Web of Trust untergraben wird.

Abbildung 4.4a zeigt die Verteilungsfunktion des Verhältnisses von ausgehendem zu eingehendem Grad für alle Knoten  $u$  in der grössten starken Zusammenhangskomponente, also  $\frac{d^+(u)}{d^-(u)}$ . Aus dieser Verteilung ergibt sich, dass eine positive Korrelation zwischen dem eingehenden und dem ausgehenden Grad eines Knotens besteht. Für etwa die Hälfte der Knoten (21.817) unterscheidet sich der eingehende Grad um maximal 20% vom ausgehenden Grad. Diese Korrelation folgt aus der hohen Anzahl gegenseitiger Kanten und der Natur des Signaturprozesses, bei dem Schlüssel üblicherweise gegenseitig signiert werden.

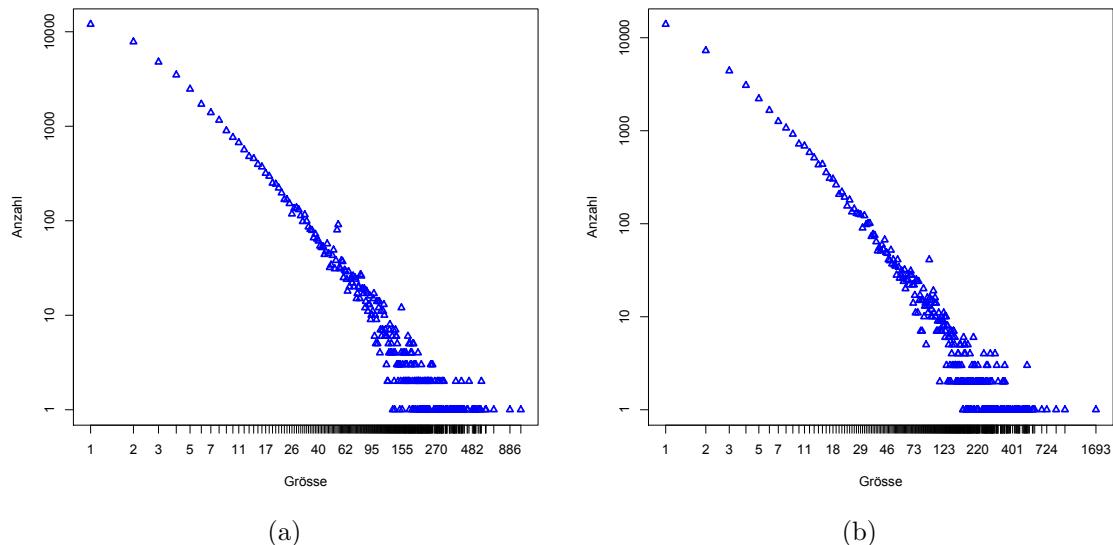


Abbildung 4.3: Verteilung der eingehenden (a) und ausgehenden (b) Knotengrade in der grössten starken Zusammenhangskomponente

#### 4.2.2.2 Clustering und Small-World

Für die Berechnung des Clustering-Koeffizienten wurde der eigentlich gerichtete Graph als ungerichteter Graph betrachtet, da es keine anerkannte Definition des Clustering-Koeffizienten für gerichtete Graphen zu geben scheint. Diese Vorgehensweise scheint die in der Literatur übliche zu sein. Selbstverständlich ist anzunehmen, dass das Ergebnis verfälscht wird, wenn gerichtete Kanten als ungerichtete Kanten betrachtet werden. Allerdings wurde bereits gezeigt, dass ein erheblicher Anteil der

<sup>1</sup>Beispielsweise verfügt ein CA-Schlüssel, der 1684 Schlüssel in der grössten Komponente unterschrieben hat, selbst nur über 683 Signaturen von Schlüsseln aus dieser Komponente

Kanten symmetrisch ist, also in beide Richtungen verläuft, und eine positive Korrelation zwischen dem ausgehenden und dem eingehenden Grad von Knoten besteht. Der Schaden sollte sich also in Grenzen halten.

Der durchschnittliche Clustering-Koeffizient für die MSCC beträgt  $C = 0,460$ . Das bedeutet, dass *im Durchschnitt* etwa die Hälfte der Nachbarn eines Knoten wieder verbunden sind. Der Graph zeigt also ein erhebliches Mass an Clustering. Zum Vergleich beträgt der Clustering-Koeffizient für ein *zufälliges* Netzwerk mit der gleichen Anzahl von Knoten und Kanten  $C = 0,00025$  und für ein zufälliges Netzwerk, dass ausserdem über die gleiche Sequenz von Graden verfügt,  $C = 0,013$ . In der MSCC findet sich also wesentlich mehr Clustering, als in einem zufällig entstandenen Netzwerk zu erwarten wäre.

Newman und Park bemerken, dass ein hohes Mass an Clustering charakteristisch für soziale Netzwerke ist und sie von vielen nicht-sozialen Netzwerken unterscheidet [NePa03]. Als möglichen Grund geben sie an, dass sich die Knoten in sozialen Netzwerken üblicherweise in *Communities* einteilen lassen, also Gruppen von Knoten, die untereinander stärker vernetzt sind als nach aussen. Intuitiv ist naheliegend, dass in einer solchen Community die Wahrscheinlichkeit, dass zwei Nachbarn eines Knoten durch eine Kante verbunden sind, hoch ist. In Abschnitt 4.4 wird gezeigt, dass der vorliegende Graph in der Tat über eine markante Community-Struktur verfügt.

Der Durchschnitt aller Distanzen in der grössten starken Zusammenhangskomponente beträgt 12,14 (siehe auch Abbildung 4.6a für die Verteilung der durchschnittlichen Distanzen pro Knoten). Dieser Wert ist in Relation zur Anzahl der Knoten im Netzwerk gering und zeigt, dass der Graph den “Small-World”-Effekt aufweist, dass also Paare von Knoten mit hoher Wahrscheinlichkeit durch einen Pfad von geringer Länge verbunden sind. Dieser Effekt tritt in vielen untersuchten Netzwerken auf und ist insbesondere typisch für soziale Netzwerke [Newm03].

Interessant ist die geringe Distanz auch, weil die Teilnehmer des Web of Trust aus verschiedensten Ländern und damit aus verschiedensten geographischen Regionen stammen. Die übliche Prozedur für das Signieren von Schlüsseln setzt ein direktes, d.h. räumliches Treffen der teilnehmenden Personen voraus. Eine Interpretation der geringen Distanz wäre, dass eine Anzahl von Teilnehmern geographisch so mobil ist, dass sich Signierungen an weit auseinanderliegenden Orten ergeben, die für die Verbindungen zwischen geographisch eigentlich weit entfernten Bereichen im Netzwerk sorgen. Allerdings wurde hier nur der Durchschnitt der Distanzen betrachtet. Es kann nicht ausgeschlossen werden, dass sich der niedrige Durchschnitt für die einzelnen Knoten nur ergibt, weil die Knoten aus dem eigenen Land bzw. der eigenen geographischen Region auch im Netzwerk sehr nahe liegen. In Abschnitt 4.4 wird gezeigt, dass sich die meisten Communities anhand von Top-Level-Domains einem Land oder Sprachraum zuordnen lassen. Interessant wäre an dieser Stelle eine Untersuchung, inwiefern sich die geographische Distanz der Länder auch in der Distanz im Netzwerk der entsprechenden Communities wiederspiegelt.

#### 4.2.2.3 Verteilung der Grade, Skalenfreiheit

Der durchschnittliche ausgehende Grad (und damit auch der durchschnittliche eingehende Grad) beträgt 9,29. Allerdings zeigen Abbildung 4.3a und 4.3b, dass dieser Durchschnitt auf höchst ungleichmässige Weise zustande kommt. Während eine

Mehrheit der Knoten einen sehr kleinen ein- bzw. ausgehenden Grad hat, existiert eine signifikante Anzahl von Knoten, deren Grad deutlich über dem Durchschnitt liegt. Dieses Verhalten ist konsistent mit einer Power-Law-Verteilung der Grade und die in etwa geraden Linien im doppelt logarithmischen Plot der Verteilungen liegen dies zusätzlich nahe. Allerdings wird in [ClSN09] argumentiert, dass ein “Nachweis” einer Power-Law-Verteilung und insbesondere die Berechnung des Power-Law-Koeffizienten etwa mit linearer Regression auf einem doppelt logarithmischen Plot (vorgeschlagen etwa bei [BrSc04]) zu stark verfälschten Ergebnissen führen kann. Stattdessen wird hier die von Clauset et al. vorgeschlagene Vorgehensweise benutzt, die beste Power-Law-Anpassung mittels der Maximum-Likelihood-Methode zu berechnen. Damit ergibt sich für die Verteilung der ein- bzw. ausgehenden Grade ein Power-Law-Koeffizient von XX bzw. XY. Diese Werte liegen im üblichen Rahmen für Netzwerke, die aus sozialen Interaktionen entstehen (FIXME: cite). Allerdings ergibt die Überprüfung der Qualität der Anpassung mittels des Kolmogorov-Smirnov-Tests Werte, die mit  $p = 0,012$  und  $p = 0,011$  deutlich unter dem in [ClSN09] empfohlenen Schwellwert von  $p = 0,1$  liegen und damit ein Power-Law als plausible Hypothese für die Verteilung ausschliessen.

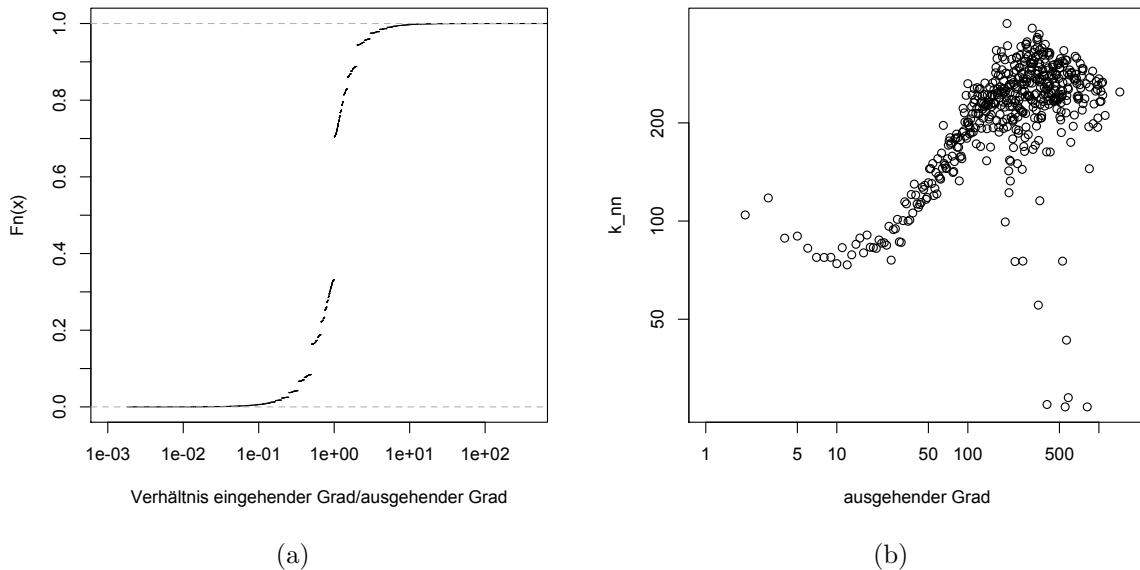


Abbildung 4.4: Kumulative Verteilungsfunktion von  $\frac{d^+(u)}{d^-(u)}$  für alle Knoten  $u$  in der MSCC (a). Verteilung von  $k_{nn}$  über alle ausgehenden Grade (b).

Es handelt sich bei dem vorliegenden Netzwerk also nicht um ein skalenfreies Netzwerk im engeren Sinne. Es stellt sich allerdings die Frage, ob das überhaupt relevant ist. Zum einen zeigt die Verteilung der Grade Merkmale, die mit charakteristischen Merkmalen einer Power-Law-Verteilung übereinstimmen, nämlich die hohe Variabilität der Grade und insbesondere die Anwesenheit einer signifikanten Anzahl von Knoten mit hohem Grad. Die Verteilung *ähnelt* also immerhin einer Power-Law-Verteilung. Eine zentrale Rolle bei den Eigenschaften, die skalenfreien Netzwerken zugeschrieben werden, spielt die Existenz eines Kerns bestehend aus “Hubs”, also Knoten mit hohem Grad, die untereinander verbunden sind. Knoten mit niedrigem Grad sind primär mit diesen Hubs verbunden. Dieser Kern hält das Netzwerk zusam-

men und sorgt für geringe Distanzen. Allerdings haben Li et al. [LADW05] gezeigt, dass die Existenz einer Power-Law-Verteilung nicht ausreicht, um die Existenz von Hubs nachzuweisen. Es existieren Netzwerke, die zwar eine Power-Law-Verteilung der Grade haben, bei denen die Knoten mit hohem Grad aber eher an der *Peripherie* liegen und keine zentrale Rolle im Sinne von Hubs übernehmen. Hubs ergeben sich erst dann, wenn Knoten mit hohem Grad primär mit Knoten vernetzt sind, die wiederum einen hohen Grad haben. Umgekehrt kann angenommen werden, dass ein Netzwerk, welches in charakteristischen Aspekten einem skalenfreien Netzwerk *ähnelt*, auch ähnliche Eigenschaften aufweisen *kann*.

Um zu überprüfen, ob die gut vernetzten Knoten als Hubs fungieren, wurde zunächst die Korrelationsfunktion  $k_{nn}$  berechnet[PSVV01]. Diese verbindet für gerichtete Graphen einen Grad  $d$  mit dem Durchschnitt des Grads der Nachbarn der Knoten, die diesen Grad  $d$  haben. Sie misst also, wie hoch der Grad von Knoten ist, die mit Knoten vom Grad  $d$  verbunden sind. Abbildung 4.4b zeigt die Verteilung von  $k_{nn}$  über alle Grade  $d$ . Aus der ansteigenden Kurve für ansteigende Grade kann geschlossen werden, dass Knoten mit hohem Grad dazu tendieren, zu anderen Knoten mit hohem Grad verbunden zu sein. Diese Knoten scheinen also in der Tat einen *Kern* des Netzwerkes zu bilden. Um dies zu untermauern, wurde zusätzlich noch der Assortativity-Koeffizient  $r$  [Newm02] berechnet, welcher zwischen -1 und 1 liegt. Ein positiver Wert von  $r$  bedeutet, dass eine positive Korrelation zwischen den Graden der Knoten besteht. Im gegensätzlichen Fall einer negativen Korrelation ist  $r$  negativ. Es ergibt sich ein Wert  $r = 0,113$ . Aus diesem positiven Wert folgt also ebenfalls, dass Knoten tendenziell mit Knoten verbunden sind, die einen ähnlichen Grad haben. Der  $r$ -Wert ähnelt den Werten, die für andere soziale Netzwerke berechnet wurden[Newm03].

Zwar scheint die in [LADW05] definierte  $s$ -Metrik ein besser geeignetes Mass für die Charakterisierung der Korrelation von Graden benachbarter Knoten zu sein als der Assortativity-Koeffizient  $r$ . Diese wurde hier aber nicht verwendet, da die Implementierung des Algorithmus für die Berechnung des Normalisierungswertes  $s_{max}$  zu aufwändig war.

#### 4.2.3 Robustheit

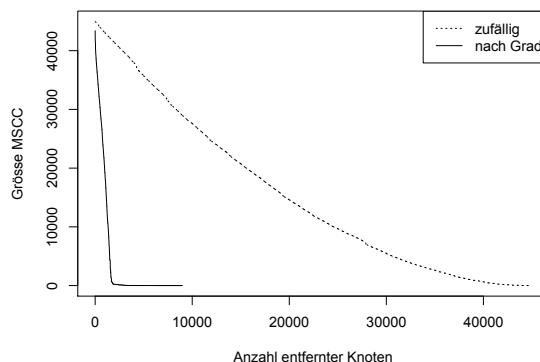


Abbildung 4.5: Entwicklung der Grösse der grössten starken Zusammenhangskomponente, wenn Knoten zufällig oder nach absteigender Grösse entfernt werden

In diesem Zusammenhang stellt sich auch die Frage nach der *Robustheit* des Netzwerkes. Eine Eigenschaft, die skalenfreien Netzwerken üblicherweise zugeschrieben wird[AlJB00], ist eine typische Reaktion auf das Entfernen von Knoten: Während die Entfernung zufällig ausgewählter Knoten kaum Schaden anrichtet, reagiert das Netzwerk sehr verwundbar auf die Entfernung von Knoten, die sehr gut vernetzt sind. Schon die Entfernung weniger sehr gut vernetzter Knoten führt dazu, dass das Netzwerk zerfällt.

Dieser Unterschied kann aufgefasst werden als der Unterschied zwischen einem gerichteten *Angriff* und einer zufälligen *Schädigung*. Ein Angreifer mit der Intention, das Netz an sich zu schädigen, wird solche Knoten auswählen, deren Entfernung einen möglichst grossen Effekt hat. Demgegenüber kann angenommen werden, dass die Wahrscheinlichkeit einer zufälligen Schädigung in etwa gleichverteilt über alle Schlüssel ist. Ein Angreifer könnte beispielsweise versuchen, Knoten zu entfernen, indem er den Schlüssel an sich kompromittiert oder den Zugriff auf das Schlüsselmaterial unmöglich macht, so dass der Schlüssel widerrufen werden muss. Demgegenüber können Schlüssel auf “normalem” Wege aus dem Netz verschwinden, wenn sie ablaufen oder beispielsweise aufgrund des Verlusts der Passphrase widerrufen werden müssen.

Eine offensichtliche Metrik für die Wichtigkeit eines Knotens und damit eine Strategie für die Auswahl des Zielknotens ist der Knotengrad. Je höher der Grad eines Knotens, desto wichtiger ist er potentiell für das Netzwerk. Um die Robustheit unter dieser Auswahlstrategie zu testen, wurden aus der grössten starken Zusammenhangskomponente sukzessive Knoten in der Reihenfolge absteigender ausgehender Grade bzw. in einer zufälligen Reihenfolge entfernt. Abbildung 4.5 zeigt die Entwicklung der Grösse der grössten Komponente, wenn mehr und mehr Knoten entfernt werden. Bei der Entfernung zufällig ausgewählter Knoten ist das Netzwerk ausgesprochen stabil. Die grösste Komponente schrumpft fast im gleichen Masse, wie Knoten entfernt werden. Dieses Ergebnis ist nicht überraschend. Da ein hoher Anteil der Knoten einen sehr niedrigen Grad hat, ist bei zufälliger Auswahl die Chance, einen solchen Knoten zu treffen, hoch. Das Verschwinden von Schlüsseln durch Ablauen ist ein normaler und häufiger Prozess. So verfügen etwa 5700 Schlüssel in der grössten starken Zusammenhangskomponente über ein Ablaufdatum, werden also irgendwann aus dem Netz verschwinden. Dieser Fall richtet also kaum globalen Schaden im Netzwerk an.

Der Schaden durch die Entfernung von Knoten mit hohem ausgehendem Grad ist deutlich grösser. Hier führt schon die Entfernung von deutlich weniger Knoten zu einem raschen Verfall bis hin zur völligen Auflösung des Netzwerkes. Aber auch hier richtet die Entfernung *einzelner* Knoten noch keinen signifikanten Schaden an. Erst ab etwa 100 Knoten – was der Entfernung aller Knoten mit einem ausgehenden Grad grösser 250 entspricht – sinkt die Grösse der Komponente erheblich. Die Möglichkeiten eines Angreifers dürften sich eher im Entfernen einzelner Knoten erschöpfen, so dass der zu erwartende Schaden gering ist. Trotzdem ist die Abhängigkeit des Netzwerks von wenigen Knoten bemerkenswert. Der rasche Verfall bei der Entfernung dieser Knoten illustriert die Wichtigkeit dieser zentralen Knoten, die als “Hubs” das Netzwerk zusammenhalten.

Dieses Verhalten ist konsistent mit dem Verhalten, dass viele – insbesondere soziale – Netzwerke aus der realen Welt zeigen, die als skalenfrei eingeordnet werden *FIXME cite*.

Angemerkt werden muss, dass neben der Grösse der Komponente auch eine Be- trachtung der Entwicklung der Distanzen in der Komponente sinnvoll wäre. Dass die Komponentengrösse sich im Fall zufälliger Reihenfolge stabil verhält, muss nicht bedeuten, dass sich die durchschnittlichen Distanzen nicht deutlich vergrössern. Ver- grössern sich die Distanzen und verlängern sich damit die nötigen Signaturketten, so sinkt die Nützlichkeit des Netzwerkes (siehe Abschnitt 4.2.4). Für die Entfer- nung von Knoten anhand des Grades wird aufgrund der Ähnlichkeit zu skalenfreien Netzwerken erwartet, dass sich die Distanzen schon durch die Entfernung weniger Knoten merklich erhöhen. Aufgrund des notwendigen Rechenaufwandes konnte dies im Rahmen dieser Arbeit aber nicht durchgeführt werden.

Der Grad eines Knoten ist nicht unbedingt das geeignetste Mass für die globale “Wichtigkeit” des Knotens. Durch eine Konzentration auf *zentrale* Knoten, also et- wa solche mit einer hohen Betweenness Centrality, lässt sich möglicherweise mehr Schaden anrichten. Dies lässt sich am Beispiel von Certificate Authorities im Web of Trust illustrieren: CA-Schlüssel haben zwar üblicherweise einen hohen ausgehenden Grad, haben also viele Schlüssel signiert und damit relativ kurze Wege zu vielen Schlüsseln. Allerdings haben sie gleichzeitig einen niedrigen eingehenden Grad, weil ein CA-Schlüssel üblicherweise nicht mit einer Person direkt verbunden ist, und damit das übliche Verifizieren der Identität wenig Sinn macht. Damit tauchen CA- Schlüssel nur auf relativ wenigen kürzesten Pfaden auf, da sie selbst eher schlecht erreichbar sind. Ihre Entfernung richtet also – zumindest in Bezug auf die Distanzen – einen relativ geringen Schaden an.

#### 4.2.4 Nützlichkeit

Dieser Abschnitt betrachtet die Nützlichkeit des Netzwerkes für seinen eigentlichen Zweck, die Verifizierung von Schlüsseln. Diese Nützlichkeit ist dann hoch, wenn für viele Knoten eine hohe Zahl von Knoten potentiell verifizierbar ist.

Eine Mindestvoraussetzung für die Verifizierung eines Schlüssels ist, dass eine Signaturkette zu diesem Schlüssel aufgebaut werden kann, dass also der Knoten im Graph erreicht werden kann. Dies ist in einer starken Zusammenhangskomponente der Fall, da laut Definition ein Pfad von jedem Knoten zu jedem anderem Knoten besteht. Allerdings ist die Erreichbarkeit an sich hier nicht von Bedeutung. Vielmehr muss die Erreichbarkeit unter den Beschränkungen des PGP/GnuPG-Vertrauensmodells (siehe Abschnitt 2.2.3) betrachtet werden. Hier gilt insbesondere die Einschränkung, dass Signaturketten in der Standardeinstellung maximal die Länge 5 haben dürfen. Selbstverständlich bedeutet das Vorhandensein einer Signaturkette von hinreichend geringer Länge noch nicht, dass ein Schlüssel tatsächlich verifizierbar ist. Zusätzlich muss noch jedem Glied dieser Kette *vertraut* werden. Wird Personen nur *geringfügig* vertraut, sind zusätzlich noch weitere redundante Signaturen nötig. Da diese Ver- trauensinformationen aber nicht öffentlich verfügbar sind, wird hier als obere Grenze für die Anzahl verifizierbarer Schlüssel nur die maximale Pfadlänge verwendet.

Die grösste Komponente enthält eine erhebliche Anzahl von Knoten mit einem aus- gehenden Knotengrad von 1. Ein solch niedriger Grad stellt eine erhebliche Ein- schränkung für die Benutzbarkeit dieses Schlüssels für die Verifizierung von anderen Schlüsseln dar. Da ein Schlüssel *A* mit ausgehendem Grad 1 nur einen Schlüssel *B* signiert hat, gibt es auch nur genau einen Schlüssel, über den eine Signaturkette aufgebaut werden kann. Natürlich wird dadurch die Menge insgesamt erreichbarer

Knoten stark eingeschränkt. Ausserdem aber existiert damit keine Redundanz. Der Besitzer des Schlüssels  $A$  muss dem Besitzer des Schlüssels  $B$  vertrauen, wenn er irgend eine Signaturkette benutzen will.  $B$  stellt einen *Single point of failure* dar, da  $A$  vom Netz getrennt wird, wenn  $B$  etwa abläuft oder zurückgezogen wird. Äquivalent ist die Verifizierbarkeit eines Knotens  $A$  stark eingeschränkt, wenn er einen eingehenden Grad von 1 hat und nur von einem einzigen Schlüssel  $B$  signiert wurde. Für die Verifizierung von  $A$  muss zwingend dem Besitzer von  $B$  vertraut werden. Ist  $B$  nicht mehr benutzbar, wird  $A$  vom Netz getrennt.

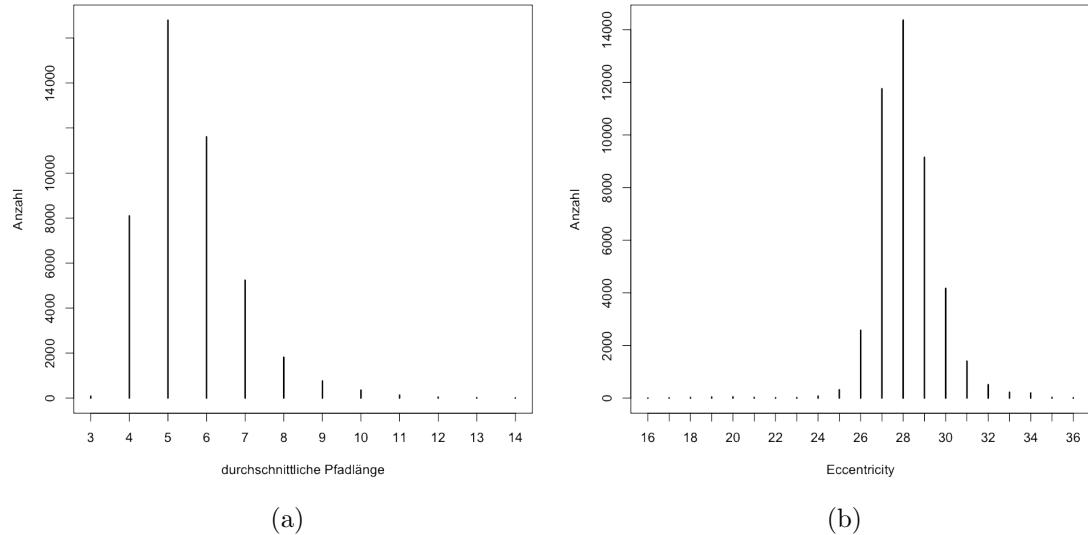


Abbildung 4.6: Verteilung der (gerundeten) durchschnittlichen Distanzen (a) und der Eccentricity (b) in der grössten starken Zusammenhangskomponente

Aus Abbildung 4.6b ist ersichtlich, dass die Eccentricity, also die *maximale* Distanz eines Knotens zu einem anderen Knoten, zwischen dem Minimum 16 (dem *Radius* des Graphen) und dem Maximum 36 (dem *Durchmesser* des Graphen) liegt. Für die meisten Knoten liegt sie bei 26 bis 31. Damit wird die maximale Pfadlänge deutlich überschritten. Für alle Knoten gilt also, dass die am weitesten entfernten Knoten grundsätzlich nicht erreichbar sind. Aber auch schon die *durchschnittliche* Distanz (Abbildung 4.6) ist für eine erhebliche Anzahl von Knoten grösser als die maximal erlaubte Länge. Klar ist also, dass für die Mehrzahl der Knoten längst nicht alle Knoten unter der Beschränkung der Pfadlänge erreichbar sind.

Die Grösse der 5-Nachbarschaft eines Knotens gibt die Anzahl der Knoten an, die unter dieser Beschränkung erreichbar sind. Allerdings macht es Sinn, nicht nur die 5-Nachbarschaft, sondern auch die  $n$ -Nachbarschaften für  $n = 2, \dots, 4$ , also kürzere Signaturketten, zu betrachten. Da jedem einzelnen Glied einer Signaturkette vertraut werden muss, steigt mit der Länge der Kette auch die Wahrscheinlichkeit, dass dieses Vertrauen eben nicht in alle Glieder vorhanden ist. Da das Web of Trust ein soziales Netzwerk darstellt, kann angenommen werden, dass im Allgemeinen mit der Entfernung zu einem Knoten im Graph auch die soziale und geographische Entfernung zu der entsprechenden Person steigt. Damit sinkt auch die Wahrscheinlichkeit, dass weiter entfernten Personen aufgrund von persönlicher Bekanntschaft vertraut werden kann. Dass einer Person, deren Schlüssel selbst signiert wurde, vertraut wird,

ist wahrscheinlicher als dass einer Person vertraut wird, deren Schlüssel nicht selbst signiert wurde. Je länger also eine Signaturkette, desto niedriger die Chance, dass sie tatsächlich benutzbar ist.

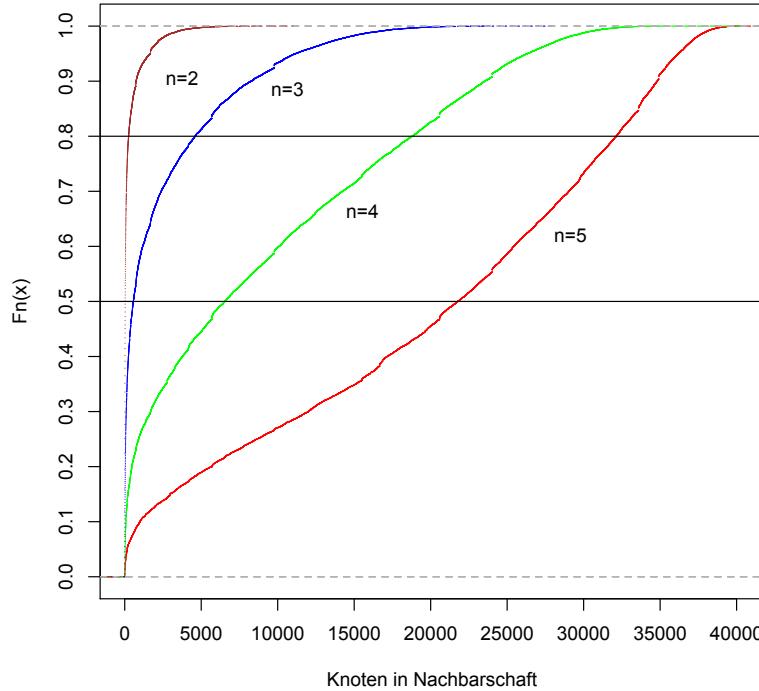


Abbildung 4.7: Kumulative Verteilungsfunktion der Nachbarschaftsgrößen in der grössten starken Zusammenhangskomponente

Abbildung 4.7 zeigt die kumulative Verteilungsfunktion der Grösse dieser Nachbarschaften. Auffällig ist hier zunächst das steile Wachstum der Kurve für die 2-Nachbarschaft, woraus geschlossen werden kann, dass diese Nachbarschaft für fast alle Knoten sehr klein ist. In der Tat liegt das 2. Quantil bei 30 und das 3. Quantil bei 160. Für die Mehrzahl der Schlüssel ist also die Menge der Schlüssel, für deren Verifizierung nur einer Person voll vertraut werden muss, sehr gering. Immerhin steigt die Grösse der Nachbarschaften mit wachsender Distanz deutlich: Für  $n = 3$  liegt das 3. Quantil bereits bei 3.371. Für  $n = 4$  und  $n = 5$  beträgt es 16.340 bzw. 30.530. Mittels längerer Signaturketten kann also die Mehrzahl der Teilnehmer einen signifikanten Anteil der in der grössten Zusammenhangskomponente vorhandenen Schlüssel *potentiell* erreichen. Interessanterweise liegt das Maximum der Nachbarschaftsgrößen für  $n = 5$  bei nur 40.180. Nicht einmal für die am besten vernetzten Schlüssel sind also alle Schlüssel mit beschränkter Kettenlänge erreichbar.

Wie bereits erwähnt, enthält die grösste Zusammenhangskomponente die Schlüssel mehrerer zentraler Certificate Authorities. Interessant ist eine Abschätzung, wie sehr das eigentlich dezentrale Web of Trust auf diese zentralen Komponenten aufbaut. Um dies festzustellen wurden die Schlüssel der grösseren Certificate Authorities (siehe Abschnitt 2.2.5) aus der grössten Komponente entfernt. Diese CA-Schlüssel haben zusammen in dieser Komponente 4.256 Schlüssel signiert. Nach der Entfernung der CA-Schlüssel zerfiel die grösste Komponente in eine Reihe von starken Zusammen-

hangskomponenten: Eine grösste Komponente mit 42455 Schlüsseln und 1.058 sehr kleine Komponenten. Daraus ergibt sich, dass die CA-Schlüssel die grösste Komponente zwar nicht fundamental zusammenhalten. Für immerhin 2.497 Schlüssel in der grössten Zusammenhangskomponente ist aber das Vorhandensein und die Benutzung der CA-Schlüssel entscheidend, da sie andernfalls nicht erreichbar sind.

## 4.3 Eigenschaften einzelner Schlüssel, zeitliche Entwicklung

### 4.3.1 Zeitliche Entwicklung

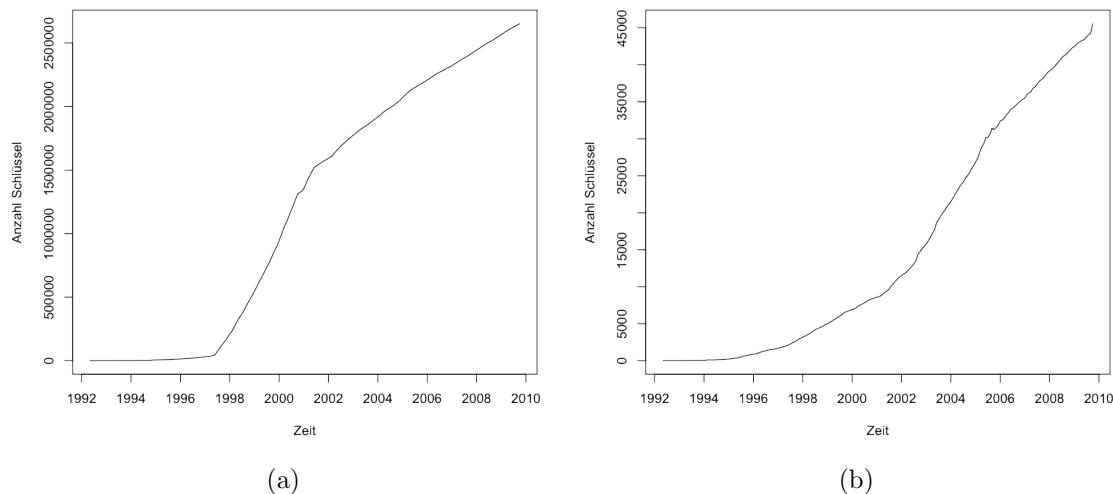


Abbildung 4.8: Zeitliche Entwicklung der Grösse des gesamten Schlüsselbestandes (a) und der grössten starken Zusammenhangskomponente (b)

Da die verwendete Datenbank alle jemals auf einen Keyserver geladenen Schlüssel enthält, ist es möglich, den Zustand des Schlüsselbestandes zu einem beliebigen Zeitpunkt zu berechnen. In Abbildung 4.8a wird die Entwicklung der Grösse des gesamten Schlüsselbestandes und in 4.8b die Entwicklung der Grösse der grössten starken Zusammenhangskomponente dargestellt. Als Beginn wurde das Jahr 1991 gewählt, da in diesem Jahr die erste Version von PGP vorgestellt wurde (siehe Abschnitt 2.2). Sowohl der gesamte Schlüsselbestand als auch die grösste starke Zusammenhangskomponente zeigen nennenswertes Wachstum erst etwa ab dem Jahr 1997. Dieser Zeitpunkt korreliert unter anderem mit der Gründung des Unternehmens *PGP Inc.*, der Veröffentlichung einer neuen Version 5 der PGP-Software und in Deutschland dem Start der bereits erwähnten “Krypto-Kampagne”. Gründe für das deutlich langsamere Wachstum des gesamten Schlüsselbestandes etwa ab dem Jahr 2001 und der grössten starken Zusammenhangskomponente mit einer Verzögerung etwa ab dem Jahr 2005 sind nicht bekannt.

Abbildung 4.8 zeigt die Rate neuer Schlüssel, abhängig von dem verwendeten Public-Key-Algorithmus. Die DSA-Schlüssel<sup>2</sup> machen zu jedem Zeitpunkt den grössten

<sup>2</sup>Mit DSA-Schlüssel sind hier öffentliche OpenPGP-Schlüssel gemeint, deren Public-Key-Paket DSA-Schlüsselmaterial enthält. Da DSA nicht zur Verschlüsselung benutzt werden kann, enthalten

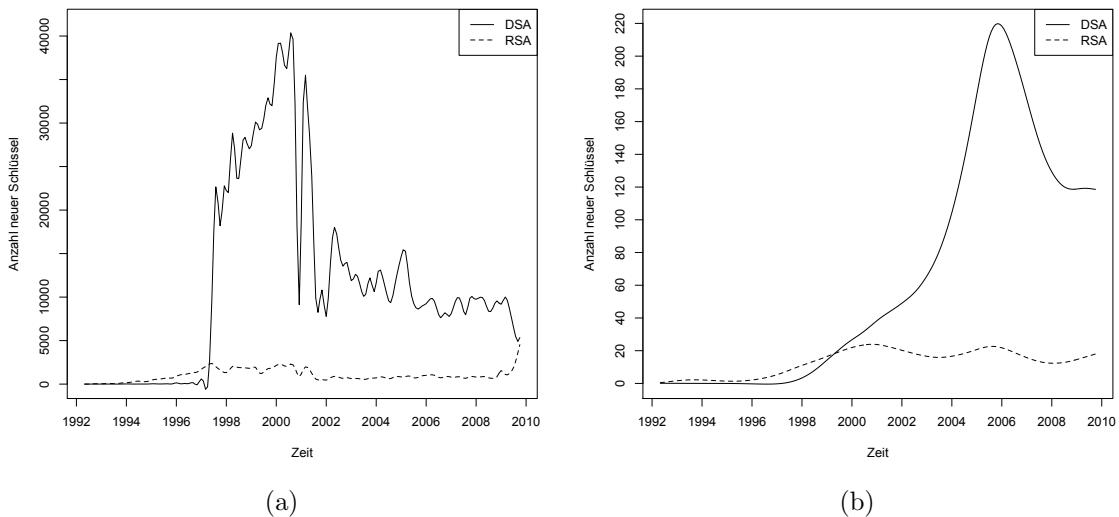


Abbildung 4.9: Rate neuer PGP-Schlüssel, die RSA oder DSA benutzen, für den gesamten Schlüsselbestand (a) und die grösste starke Zusammenhangskomponente (b) in monatlichen Abständen

Anteil aus. Ihre Entwicklung spiegelt im Wesentlichen die Größenentwicklung insgesamt aus Abbildung 4.8 wieder. Demgegenüber ist die Zuwachsrate von RSA-Schlüsseln bis zum Jahr 2008 konstant niedrig. Der Grund dafür ist vermutlich, dass die PGP-Implementierungen in diesem Zeitraum in der Standardeinstellung DSA-Schlüssel erzeugten, was wiederum unter anderem in dem bis ins Jahr 2000 bestehenden Patentschutz des RSA-Algorithmus begründet ist. Die Rate neuer DSA-Schlüssel sinkt ab dem Jahr 2009 deutlich ab, während die Rate neuer RSA-Schlüssel gleichzeitig deutlich ansteigt. Vermutlich ist dies darauf zurückzuführen, dass seit dem Jahr 2009 neue Schlüssel, die mit GnuPG erzeugt werden, in der Standardeinstellung RSA statt DSA verwenden. Diese Umstellung wurde in der Entwicklerversion von GnuPG am 17.05.2009<sup>3</sup> aufgrund von Berichten über verbesserte Angriffe auf die Hashfunktion SHA-1 [McHP09], auf der DSA basiert, vorgenommen. Warum diese Entwicklung nicht ebenfalls in der grössten starken Zusammenhangskomponente sichtbar ist, ist nicht klar. Möglicherweise liegt dies daran, dass für die Aufnahme in diese Teilmenge nicht nur das Erzeugen eines Schlüssels, sondern auch zusätzliche Aktivität in Form der Vernetzung mit anderen Schlüsseln notwendig ist. Neue Schlüssel werden also vermutlich im Allgemeinen nicht sofort in die Zusammenhangskomponente aufgenommen, sondern erst mit einiger Verzögerung. Die Zusammenhangskomponente dürfte also etwas “träger” auf Veränderungen reagieren.

### 4.3.2 Public-Key- und Hashalgorithmen

In Tabelle 4.1b sind schlussendlich noch einige Statistiken über Eigenschaften einzelner Schlüssel aufgeführt.

Die einzelnen “certification level” (siehe Abschnitt 2.2.3) werden durchaus verwendet (Tabelle ??). Zwar ist die Mehrheit der Signaturen mit dem Standardwert “Generic”

diese Schlüssel noch einen Unterschlüssel mit ElGamal-Schlüsselmaterial, der zum Verschlüsseln benutzt wird.

<sup>3</sup><http://lists.gnupg.org/pipermail/gnupg-devel/2009-May/025079.html>

Algorithmus	Anzahl	Algorithmus	Anzahl		
Signaturen gesamt	446325	Schlüssel gesamt	44952		
MD5	41700	RSA-512	203	Cert Level	Anzahl
SHA-1	398849	RSA-768	257	Generic	299518
RIPE-MD/160	122	RSA-1024	3903	Persona	904
SHA256	5031	RSA-2048	2408	Casual	28718
SHA512	2472	RSA-3072	96	Positive	119575
SHA224	532	RSA-4096	1198		
		DSA	36555		

Tabelle 4.1: Verwendung von Hashalgorithmen (a), Public-Key-Algorithmen (b) und Cert-Leveln ?? in der grössten starken Zusammenhangskomponente

versehen, der keine weitere Aussage macht. Allerdings verfügt über 25% der Signaturen über den höchsten Wert “Positive”. Da die Level keine im Standard festgelegte klare Semantik haben, ist die Aussagekraft dieser Werte natürlich begrenzt. Sie können aber zumindest dahingehend interpretiert werden, dass *mindestens* die Personen, die von “Generic” abweichende Werte vergeben, den Signaturprozess durchaus ernstnehmen.

Wie zu erwarten basiert die grosse Mehrheit der Signaturen in der grössten starken Zusammenhangskomponente auf SHA1. Neben einzelnen Signaturen mit Vertretern der SHA2-Familie findet sich aber auch noch eine überraschend hohe Anzahl von MD5-Signaturen: 41.700 Signaturen, d.h. 10% basieren auf MD5. Die Sicherheit von MD5 muss inzwischen als ernsthaft kompromittiert eingestuft werden. 2008 gelang es einer Gruppe, durch eine Chosen-Prefix-Kollision ein von einer kommerziellen CA signiertes “bösertiges” Certification-Authority-Zertifikat für SSL zu erhalten [SSAL<sup>+</sup>09]. GnuPG warnt zwar vor der Verwendung von MD5-Signaturen, erlaubt ihre Verwendung aber noch. Die Bedeutung der MD5-Signaturen für den Zusammenhalt der grössten Komponente ist nicht unwesentlich: Werden diese Signaturen entfernt, reduziert sich die Grösse der Komponente auf 37.599 Schlüssel, d.h. es werden ca. 8.000 Schlüssel abgetrennt.

RSA-Schlüssel mit einer Länge von höchstens 768 bit und weniger müssen ebenfalls als praktikabel brechbar betrachtet werden. Dies wurde durch die Faktorisierung der Zahl RSA-768 demonstriert [KAFL<sup>+</sup>10]. Kleinjung et al. schätzen den Zeitrahmen bis zur Faktorisierung einer 1024-bit-Zahl innerhalb eines *akademischen Rahmens* auf 5 bis 10 Jahre. Die Menge der in der grössten Komponente vorhandenen RSA-Schlüssel mit 512 bzw. 768 bit ist mit insgesamt 450 Schlüsseln gering. Ihre Entfernung trennt nur etwa weitere 300 Knoten von der Komponente ab. Die Anzahl von RSA-Schlüsseln mit 1024 bit ist aber deutlich grösser. Die Entfernung aller ca. 4.500 RSA-Schlüssel, deren Länge höchstens 1024 bit beträgt, reduziert die Grösse der Komponente um insgesamt ca. 8.500 Knoten, trennt also weitere 4000 Knoten vom Netz ab. Es ist anzunehmen, dass sich die durchschnittlichen Distanzen ebenfalls erhöhen. Da Standardisierungsorganisationen eine Abkehr von RSA-1024 ab dem Jahr 2010 empfehlen[BBBP<sup>+</sup>07] und praktikable Angriffe wohl nur eine Frage weniger Jahre sind, ist damit wie durch die MD5-Signaturen eine erhebliche Beeinträchtigung der grössten Komponente absehbar.

Es ist zu erwarten, dass sich durch die Umstellung der Standardeinstellung in GnuPG der Anteil von RSA-Schlüsseln mit einer Länge von mindestens 2048 bit in nächster Zeit deutlich erhöht.

## 4.4 Communities

Dieser Abschnitt analysiert die Community-Struktur der grössten starken Zusammenhangskomponente nach der in Abschnitt 3.3 beschriebenen Vorgehensweise.

Die Zerlegung des gerichteten Graphen mit dem Algorithmus von Rosval et al. lieferte keine verwertbaren Ergebnisse. Zwar wurde eine Zerlegung in 2869 Partitionen berechnet. Allerdings gilt für die grosse Mehrzahl dieser Knotenmengen, dass der durch sie induzierte Teilgraph überhaupt keine Kanten hat. Nur für ca. 160 Partitionen hat der jeweils induzierte Teilgraph Kanten, in allen Fällen allerdings sehr wenige (zwischen 1 und 104 für eine Partition der Grösse 682). Dass die Knoten einer Partition in allen Fällen so gut wie keine interne Vernetzung haben und damit der Definition von Communities absolut nicht entsprechen, ist ein klarer Hinweis, dass die berechnete Zerlegung die tatsächliche modulare Struktur des Graphen nicht sinnvoll wiedergibt. Dass der Graph in der Tat eine ausgeprägte modulare Struktur hat, zeigen die anhand des ungerichteten Graphen berechneten Zerlegungen, die auch unter inhaltlichen Gesichtspunkten Sinn ergeben. Diese Struktur kann sich auch nicht erst durch die Reduzierung des gerichteten auf einen ungerichteten Graphen ergeben. Die berechnete Zerlegung mit etlichen grossen Partitionen ohne interne Vernetzung ist in keinem Fall ein sinnvolles Ergebnis eines Community-Algorithmus und spricht daher für eine fehlerhafte Berechnung. Ob diese allerdings auf Fehler in der verwendeten Implementierung der Autoren<sup>4</sup> oder auf Fehler im Algorithmus zurückgeht, ist nicht bekannt.

Für die restlichen Methoden wurde der Graph auf einen ungerichteten Graphen reduziert. Dazu wurde jede einzelne gerichtete Kante in eine ungerichtete Kante umgewandelt. Aus dem gerichteten Graphen mit 44.6326 Kanten ergab sich so ein ungerichteter Graph mit 295.425 Kanten.

Für die COPRA-Berechnung wurden – wie in [Greg10] vorgeschlagen – Berechnungen mit unterschiedlichen Werten  $v$  durchgeführt, um den “besten” Wert im Sinne der Modularity zu ermitteln. Dabei ergaben sich mit steigender maximaler Überlappung  $v$  nur bis  $v \leq 3$  steigende Modularity-Werte, darüber hinaus fielen sie kontinuierlich ab. Deshalb wurde die Berechnung mit  $v = 3$  verwendet. Die dort berechneten Modularity-Werte hatten eine geringe Standardabweichung von 0,003. Dieses Ergebnis kommt unerwartet. In [Greg10] ergaben sich für ein PGP-Web-of-Trust-Netzwerk mit 10680 Knoten bis zu  $v = 9$  ansteigende Modularity-Werte. Das (lokale) Maximum bei geringem  $v$  für das hier verwendete Netzwerk könnte darauf hinweisen, dass seine Community-Struktur keine signifikante Überlappung aufweist. Ein weiterer Hinweis darauf ist, dass die Einführung der Überlappung mit  $v = 3$  gegenüber der nicht überlappenden Berechnung mit  $v = 1$  für die Modularity (Tabelle 4.2) keinen signifikanten Anstieg brachte. Auch die Zuordnung zu Gruppen bzw. Keysigning-Parties (Tabelle 4.3) und die Verteilung der Grösse der Communities (Abbildungen 4.10b und 4.10a) ähneln sich weitgehend. Alternativ könnte dieses Ergebnis allerdings auch in einem nicht-optimalen Verhalten des Algorithmus begründet liegen.

Für den Algorithmus von Blondel et al. zeigt sich zunächst ein deutlich höherer Modularity-Wert als für Fastmod. Dieses Ergebnis gibt also die reine Struktur des

---

<sup>4</sup><http://www.tp.umu.se/ rosval/code.html>

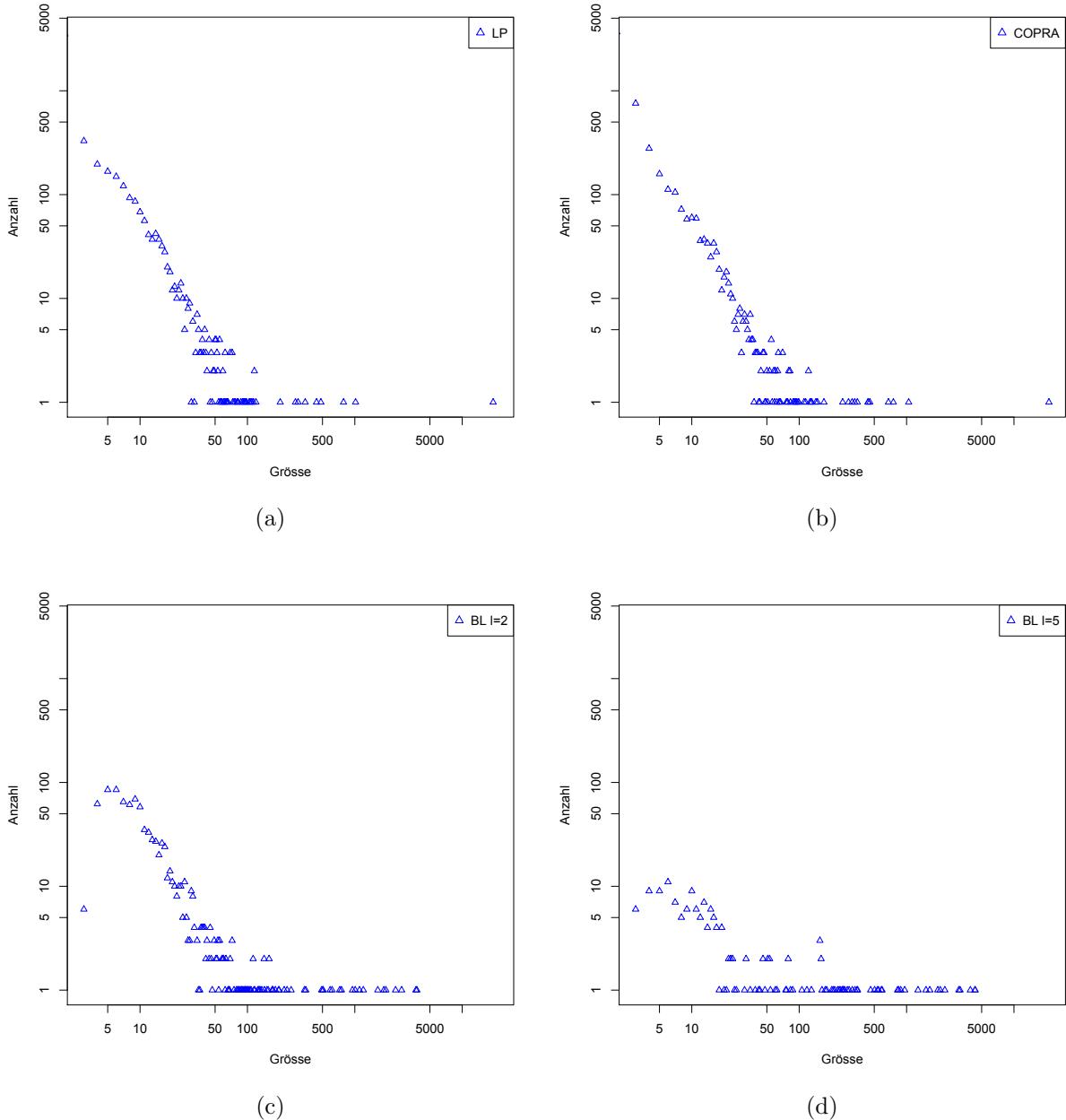


Abbildung 4.10: Größenverteilung der Communities

Graphen deutlich besser wieder. Wie in Abschnitt 2.3.5 beschrieben, verläuft der Algorithmus von Blondel et al. iterativ in Phasen, in denen jeweils zuerst Knoten einer benachbarten Community zugeordnet werden, wenn sich dadurch ein Modularity-Gewinn erzielen lässt, und dann die Knoten dieser so entstandenen Communities zu Knoten in einem neuen Netzwerk verschmolzen werden. Hier werden die Ergebnisse der Phasen  $l = 2$  und der letzten Phase  $l = 5$  betrachtet. Für  $l = 5$  ergab sich die höchste Modularity. Allerdings sticht hier die sehr kleine Anzahl von Communities heraus, die wiederum vergleichsweise gross sein müssen. Gegenüber dem Ergebnis der Phase 2 ergibt sich eine relativ kleine Steigerung der Modularity, die mit einem

erheblichen Verlust an *Auflösung* erkauft wird<sup>5</sup>. Aus Abbildung 4.10c und 4.10d geht hervor, dass diese Reduktion der Anzahl im Wesentlichen auf das Reduzieren sehr kleiner Communities zurückgeht. Da der Algorithmus von Blondel et al. genauso wie Fastmod mittels einer (lokalen) Optimierung der Modularity arbeitet, könnte dieses Phänomen auf das in der Literatur beschriebene Auflösungslimit der Modularity-Funktion (bspw. [FoBa07] [GodC09]) zurückzuführen sein. Es scheint fraglich, ob die in Relation zur Grösse des Graphen sehr geringe Zahl von 186 Communities die Struktur des Graphen (in Bezug auf die inhaltliche Analyse) adäquat wiedergibt.

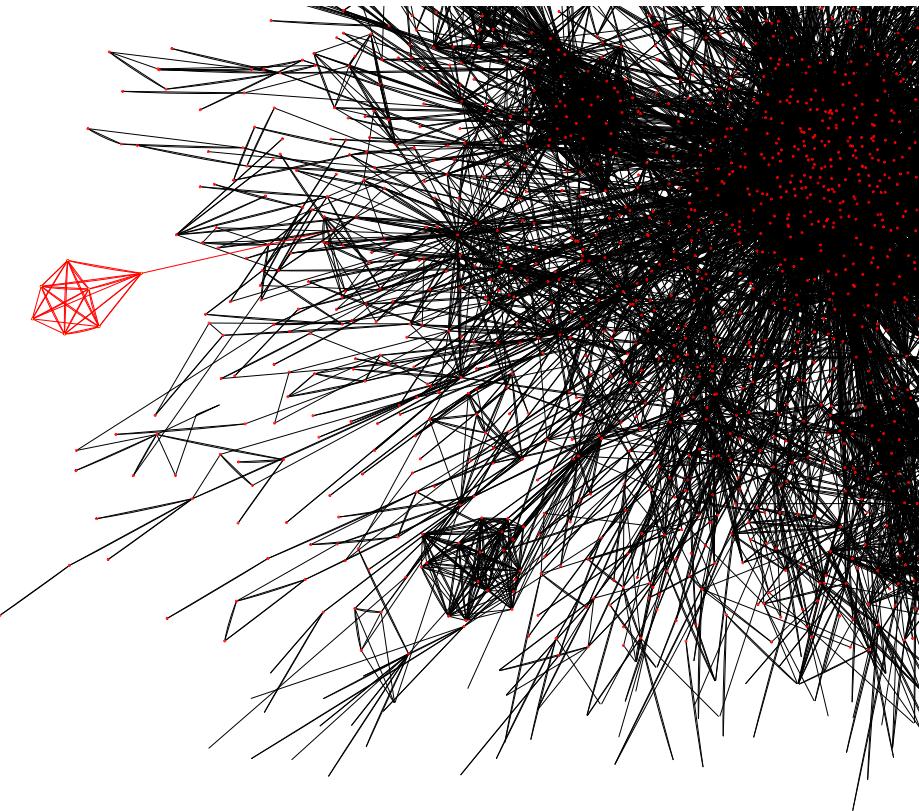


Abbildung 4.11: Community mit modularen Untereinheiten (Grösse 2271, Blondel ( $l = 5$ ), Force-directed Layout)

Abbildung 4.11 illustriert dies anhand einer Community aus der Zerlegung des Algorithmus von Blondel et al. mit  $l = 5$ , die klar abgegrenzte Untereinheiten enthält, die intuitiv wieder eigenständige Communities darstellen.

Abbildung 4.12 illustriert die begrenzte Qualität der Fast-Modularity-Zerlegung, die sich im vergleichweise niedrigen Modularity-Wert niederschlägt. In der Zeichnung des induzierten Teilgraphen einer grossen Community sind etliche dicht gezeichnete Zusammenballungen von Knoten zu sehen, die nur wenige Kanten nach aussen haben. Diese können als Community-artige Strukturen interpretiert werden. Zwar kann argumentiert werden, dass die Zeichnung die Struktur des Teilgraphen nicht notwendigerweise sinnvoll wiedergibt, dass also die dichten Teilebereiche in der Zeichnung nur Artefakte der Zeichenmethode darstellen. Allerdings zeigt Noack, dass starke Übereinstimmungen zwischen einer energieoptimalen Einbettung eines Graphen in

<sup>5</sup>Die Zerlegungen der Phasen 3 und 4 unterschieden sich nur geringfügig von der Zerlegung der Phase 5

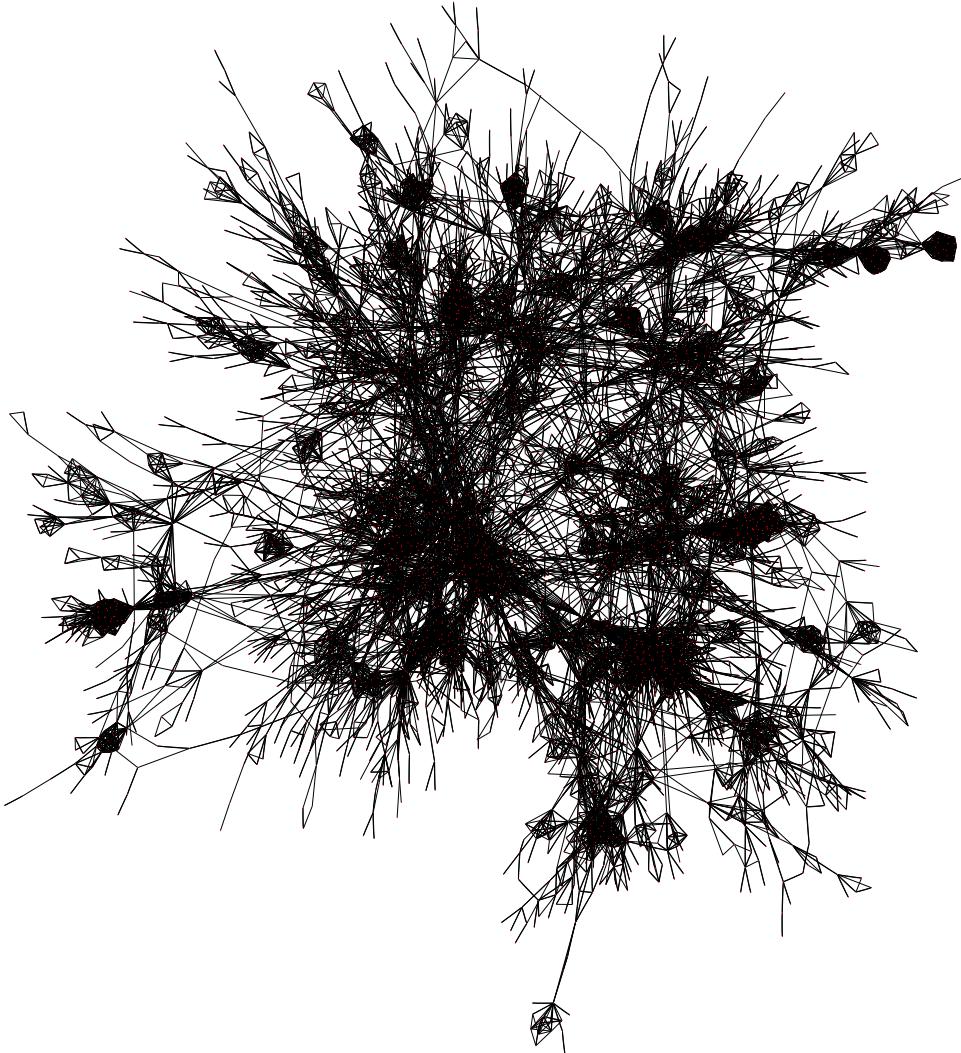


Abbildung 4.12: Grosse Community mit modularer Struktur (Berechnet mit Fast-Modularity, Force-directed layout)

die Ebene nach der Force-directed-Methode und einer Partitionierung eines Graphen mit optimaler Modularity bestehen [Noac09]. Es scheint daher gerechtfertigt, eine Zeichnung eines Graphen (Force-directed) mit markanten dichten Bereichen als Indiz für eine modulare Struktur dieses Graphen zu betrachten.

Die Verteilungen der Community-Größen aus allen Berechnungen sind in einer Hinsicht konsistent: Neben einer grossen Zahl von kleinen Communities gibt es eine signifikante Anzahl von grösseren bis sehr grossen Communities. Es scheint also keine charakteristische Grösse der Communities zu geben.

Aus Tabelle 4.2 geht hervor, dass sich – zumindest im Fall von Fastmod und Blondel et al. – fast alle Knoten des Graphen in einer Community wiederfinden, die mindestens die Grösse 4 hat. In Abschnitt 4.2 wurde gezeigt, dass ein erheblicher Anteil der Knoten einen sehr geringen Grad von 1 oder 2 hat. Da solche Knoten mangels Kanten nicht selbst eine dichte Vernetzung herstellen können, müssen sie Mitglied in Communities sein, in denen Knoten mit höherem Grad sowohl die interne Vernetzung der Community als auch die Vernetzung der Community mit anderen erzeugen. COPRA scheint hingegen dazu tendieren, deutlich mehr Knoten in Communities sehr

Algorithmus	Modularity	Communities	Enthaltene Knoten
FM	0,596	552	44611
BL ( $l = 2$ )	0,701	936	44934
BL ( $l = 5$ )	0,710	186	44934
COPRA ( $v = 1$ )	(0,780)	1421	42205
COPRA ( $v = 3$ )	(0,786)	1354	—

Tabelle 4.2: Modularity-Werte, Anzahl der Communities mit mehr als 3 Knoten sowie die Anzahl der insgesamt in diesen Communities ( $>3$ ) enthaltenen Knoten für die Algorithmen Fast-Modularity (FM), Label-Propagation (LP), Blondel et al. (BL) auf Level 2 und 5 sowie COPRA. Die Modularity-Werte für COPRA entsprechen der Modularity-Definition für überlappende Zerlegungen nach [NMCM09] und können mit den anderen Werten nicht verglichen werden.

kleiner Grösse mit 3 oder weniger unterzubringen. Dieses Verhalten könnte dazu führen, dass die Aussagekraft der Zerlegung in Bezug auf die realen Communities geschwächt wird, wenn viele Knoten auf diese Art abgespalten werden.

Gegenüber der COPRA-Zerlegung mit  $v = 1$  zeigt die Zerlegung mit  $v = 3$  eine deutlich grössere Anzahl von Communities der Grösse 3 und 4. Abgesehen davon ähneln sich die Grössenverteilungen aber weitgehend. Beide enthalten eine charakteristische sehr grosse Community mit ca. 19000 bzw. ca. 21000 Mitgliedern. Demgegenüber bietet die Blondel-Zerlegung mit  $l = 2$  eine deutlich geringere Anzahl solcher sehr kleiner Communities, dafür aber eine höhrere Zahl mittlerer (mehr als 100 Mitglieder) und insbesondere eine höhere Zahl sehr grosser (zwischen 500 und 5000 Mitglieder) Communities. Die beiden Algorithmen konvergieren nicht gegen ein gemeinsames Optimum, sondern liefern Ergebnisse, die sich qualitativ stark unterscheiden.

Der insgesamt recht hohe Modularity-Wert weist darauf hin, dass der Graph in der Tat eine ausgeprägte modulare Struktur hat, wie dies von einem sozialen Netzwerk erwartet wird. Newman et al. geben an, dass ein Modularity-Wert  $C \geq 0.3$  in der Praxis auf eine signifikante Community-Struktur schliessen lässt[ClNM04].

Die Abbildungen 4.13 und 4.14 zeigen die Struktur der Communities für die Zerlegungen COPRA ( $v = 1$ ) und Blondel ( $l = 2$ ). Auffällig ist hier insbesondere die teilweise sternförmige Struktur der COPRA-Zerlegung. Ein signifikanter Anteil der kleineren Communities ist rund um die zentrale Community der Grösse ca. 19000 angeordnet und im wesentlichen nur mit dieser vernetzt. Diese Struktur weist gewisse Ähnlichkeiten zur Struktur der starken Zusammenhangskomponenten auf (siehe Abschnitt 4.2.1). Diese sind ebenfalls sternförmig um eine zentrale Komponente angeordnet, die die Grösse der anderen um mehrere Grössenordnungen übertrifft. Allerdings sind im Fall der Community-Struktur noch eine Reihe mittelgrosser Communities vertreten, die dieses Muster etwas stören. Das sternförmige Muster der Zusammenhangskomponenten wiederholt sich also in etwa innerhalb der grössten Zusammenhangskomponente auf einem anderen Level. Es könnte daher spekuliert werden, ob das Netzwerk eine *Selbstähnlichkeit* aufweist, wie sie für eine Reihe von komplexen Netzwerken gezeigt wurde[SoHM05]. Dieser Frage wurde allerdings im Rahmen dieser Arbeit nicht weiter nachgegangen. Im Vergleich dazu zeigt sich in der Struktur der Blondel-Communities die etwas ausgewogenere Grössenverteilung. Zwar sind auch hier viele kleine Communities sternförmig um die zentralen Com-

munities angeordnet, verteilen sich hier aber auf mehrere solcher grossen zentralen Communities.

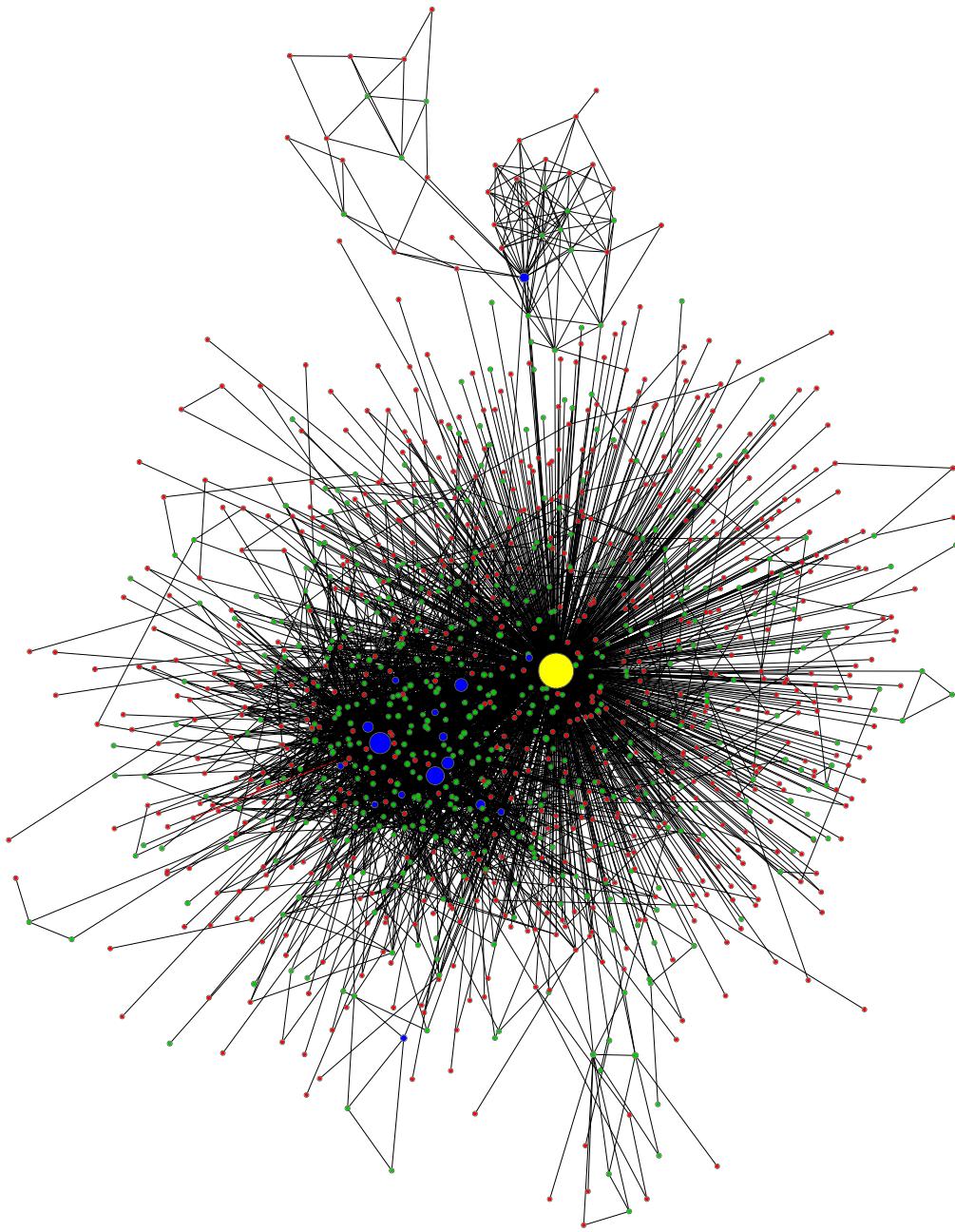


Abbildung 4.13: Struktur der Communities (COPRA ( $v = 1$ ) (rot: Grösse  $< 10$ , grün: Grösse  $< 100$ , blau: Grösse  $< 1000$ , gelb: Grösse  $> 1000$ ).

Aus Tabelle 4.3 geht hervor, dass für alle Zerlegungen fast alle Communities einer Top-Level-Domain zuordnenbar sind oder von einer TLD dominiert werden. Werden die generischen Top-Level-Domains (gTLD: com, org, net, info, ...) abgezogen, verbleiben im Fall von COPRA noch 519 (38%) Communities, die einer TLD (und damit einer *Country-Coded TLD (ccTLD)*) zugeordnet werden können und 315 (23%), die von einer ccTLD dominiert werden. Für die anderen Zerlegungen ergeben sich ähnliche Werte. Damit ist immerhin etwa die Hälfte der Communities einem Land zuordnenbar. Für die von einer gTLD dominierten Communities ergeben Stichpro-

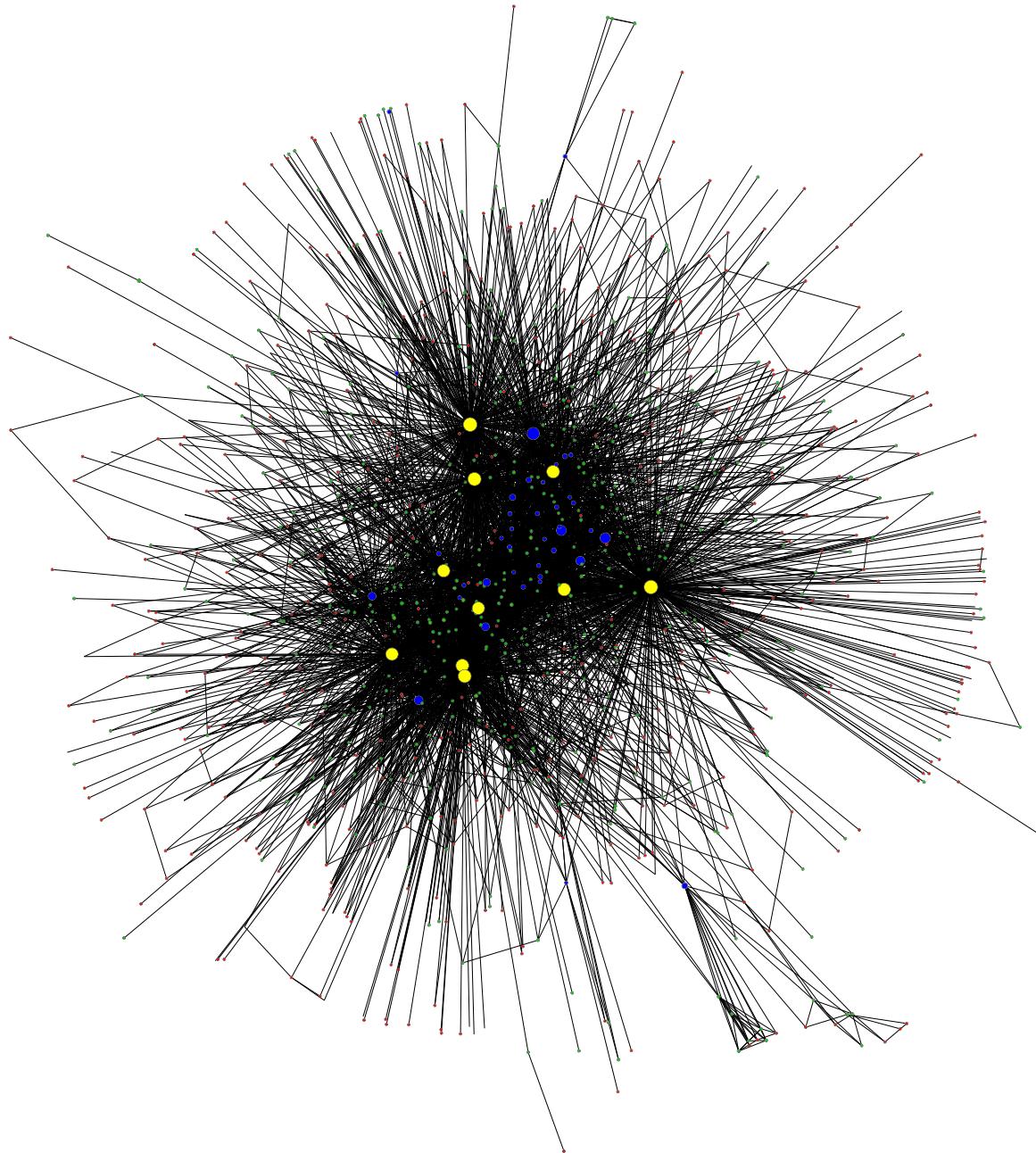


Abbildung 4.14: Struktur der Communities (Blondel ( $l = 5$ ) (Knotenfärbung wie in Abb.4.13)

ben, dass sich die Teilnehmer der meisten dieser Communities anhand der Namen zumindest einem Sprachraum zuweisen lassen.

Dieses Ergebnis ist naheliegend. Wie in Abschnitt 3.3 dargelegt, wird davon ausgegangen, dass ein wesentlicher Faktor, der das Zustandekommen von Signaturen beeinflusst, die sozialen Kontakte einer Person sind. Es ist weiterhin naheliegend, dass sich diese sozialen Kontakte unter anderem aufgrund von Sprachbarrieren und der räumlichen Nähe hauptsächlich im Rahmen eines Landes ergeben. Beispielsweise ist für eine aus Finnland stammende Person anzunehmen, dass die Mehrzahl ihrer Freunde, Bekannten, Geschäftspartner und sonstigen Kontakte wieder Finnen sind.

Algorithmus	TLD-S	TLD-M	SLD-S	SLD-M	SIG
FM	293 (53%)	247 (44%)	56 (10%)	154 (27%)	128 (23%)
BL ( $l = 2$ )	499 (53%)	417 (45%)	41 (4%)	254 (27%)	115 (12%)
BL ( $l = 5$ )	85 (47%)	85 (47%)	15 (8%)	38 (21%)	26 (14%)
COPRA ( $v = 1$ )	824 (58%)	564 (40%)	178 (13%)	429 (30%)	572 (40%)
COPRA ( $v = 3$ )	792 (58%)	525 (38%)	187 (14%)	425 (31%)	555 (41%)

Tabelle 4.3: SLD-TLD-Zuweisung, TIME-CORR

Zwar könnte argumentiert werden, dass das Internet die geographische bzw. nationale Beschränkung sozialer Kontakte aufhebt. Allerdings spielt die geographische Nähe eine wichtige Rolle im Signaturprozess. Die verbreitete Prozedur für das Signieren eines Schlüssels setzt ein direktes persönliches Treffen der Signierenden voraus (siehe Abschnitt 2.2.5). Es ist anzunehmen, dass die überwiegende Mehrheit der PGP-Benutzer die meiste Zeit in einem Land und damit in einem geographisch eingegrenzten Bereich verbleibt. Damit kommen für diese Mehrheit überwiegend nur Bewohner des gleichen Landes als Signaturpartner in Frage. Als Mechanismen für die internationale Vernetzung kommen dann etwa internationale Konferenzen in Frage. Nur ein sehr kleiner Teil der PGP-Benutzer dürfte die notwendige internationale Mobilität aufweisen, so dass sich seine Signaturen nicht überwiegend einem geographischen Bereich bzw. einem Land zuweisen lassen.

Interessanter für die Fragestellung ist aber die Erkennung von Keysigning-Parties und die Zuordnung zu Second-Level-Domains. Im Vergleich der Algorithmen (Tabelle 4.3) zeigt sich zunächst, dass die COPRA-Zerlegungen bessere Resultate liefern als die Blondel-Zerlegungen, da hier mehr Communities zugeordnet werden können. Zwar entfällt bei COPRA ein grosser Anteil dieser Zuordnungen auf sehr kleine Communities der Grösse 3 und 4. Die Aussagekraft solcher kleiner Zuordnungen ist selbstverständlich sehr begrenzt. Werden diese abgezogen, verbleibt aber immer noch eine grössere Zahl zuordnenbarer Communities nichttrivialer Grösse. Aus den absoluten Zahlen in Tabelle 4.3 und den Grössenverteilungen in Abbildungen 4.17, 4.18 und 4.19 geht hervor, dass die Überlappung keine wesentlichen Vorteile bringt. Im Vergleich der Blondel-Zerlegungen zeigt sich, dass das Zusammenpacken von Communities von Phase  $l = 2$  zu Phase  $l = 5$  und der damit verbundene Auflösungsverlust es erheblich erschwert, die Communities inhaltlich zuzuordnen. Eine Zerlegung mit höherer Modularity entspricht also im Allgemeinen nicht einer Zerlegung, die die Struktur realer Communities besser wiedergibt. Die Zerlegungen des Algorithmus von Blondel et al. scheinen insgesamt vom Auflösungslimit der Modularity-Maximierung betroffen zu sein und bieten daher nicht die notwendige Auflösung, um die Communities besser inhaltlich analysieren zu können.

Insgesamt ist eine tatsächliche Zuordnung anhand des 80%-Schwellwerts nur für sehr wenige Communities von geringer Grösse möglich. Für den 40%-Schwellwert und die zeitliche Korrelation ist die Anzahl zuordnenbarer Communities deutlich höher. Auch hier entfällt allerdings die Mehrzahl auf sehr kleine Communities. Dass so gut wie keine Communities mit einer Grösse über 100 zugeordnet werden können, spricht dafür, dass diese sich nicht anhand dieser einfachen Mechanismen bilden.

Die sehr grosse Community aus der COPRA-Zerlegung könnte vermuten lassen, dass hier relativ willkürlich Knoten zusammengepackt wurden. Allerdings hat diese Community eine Reihe von Eigenschaften, die nahelegen, dass diese Gruppierung durch-

Grösse	SLD	Anteil
436	apache.org	48%
130	nun.org	82%
97	cert.org	70%
81	redhat.org	68%
63	purdue.edu	65%
59	cisco.com	79%

Tabelle 4.4: foo

aus Sinn ergibt. Von 21.157 Knoten verfügen 8.879 (42%) über eine E-Mail-Adresse in der Top-Level-Domain *.de*, während andere – insbesondere nicht-deutschsprachige – ccTLDs deutlich schwächer vertreten sind. Das zeigt einerseits einen überraschend hohen Anteil von deutschen oder deutschsprachigen Benutzern am Gesamtnetzwerk. Die 8879 Schlüssel sind natürlich nur eine Untergrenze, da auch in einer Vielzahl weiterer Communities deutsche Teilnehmer vorkommen. Ein Grund für diese hohe Anzahl deutscher Schlüssel könnte in der in Abschnitt 2.2.5 erwähnten “Krypto-Kampagne” der Zeitschrift c’t liegen, die das Bewusstsein für die Notwendigkeit der Authentifizierung von Schlüsseln im deutschsprachigen Raum deutlich gesteigert haben dürfte. Andererseits lässt dies den Schluss zu, dass die enthaltenen Knoten eben nicht willkürlich zusammengepackt wurden, sondern neben ihrer topographischen Nähe auch eine “inhaltliche” Gemeinsamkeit aufweisen. Darauf weist auch hin, dass in dieser Community eine relativ zu anderen SLDs sehr grosse Anzahl von Schlüsseln von Mitgliedern des Debian-Projekts enthalten ist. Diese Community enthält nicht nur knapp die Hälfte der Schlüssel, sondern mit ca. 260.000 Kanten den Grossteil der überhaupt vorhandenen Kanten. Sie ist also anscheinend intern sehr dicht vernetzt. Daher scheint es fraglich, ob sie intern wieder eine schlecht aufgelöste modulare Struktur wie in Abbildung 4.12 hat, die sich in sinnvolle Untereinheiten zerlegen lässt.

In Tabelle 4.4 sind die 6 grössten Communities aus der COPRA-Zerlegung mit Zuordnung zu Second-Level-Domains aufgeführt.

Es stellt sich die Frage, ob die zeitliche Korrelation der Signaturen in einer Community ein geeignetes Mittel ist, um Keysigning-Parties zu erkennen. Neben dem zeitlichen Zusammenhang der Signaturen ist ein weiteres Merkmal einer Keysigning-Party wie in Ab. 2.2.5 beschrieben, dass jeder Teilnehmer mit (fast) jedem anderem Teilnehmer Signaturen austauscht, so dass sich eine fast vollständige Vermaschung – fast eine Clique – ergibt. Eine stichprobenartige Untersuchung in den Communities zeigt, dass die meisten Communities mit zeitlicher Korrelation zumindest teilweise dieses Merkmal zeigen. Abbildung 4.15 zeigt ein Beispiel einer solchen Community. Etwa die Hälfte der Knoten ist in einer Fast-Clique enthalten, deren Knoten mit (fast) allen anderen vernetzt sind. Dieser innere Teil des Teilgraphen entspricht genau dem Bild, das als Resultat einer Keysigning-Party erwartet wird. Auch wenn dies auf die meisten der Communities mit zeitlicher Korrelation zutrifft, gibt es auch Gegenbeispiele. Abbildung 4.16 zeigt eine Community, die zwar das Kriterium der zeitlichen Korrelation erfüllt, ansonsten aber nichts mit dem erwarteten Bild einer Keysigning-Party gemein hat. Allerdings erfüllt der Teilgraph trotzdem das Kriterium der Community, da die Knoten am Rand nur Signaturen zu dem zentralen Knoten haben und damit die interne Kantendichte deutlich höher ist als die ex-

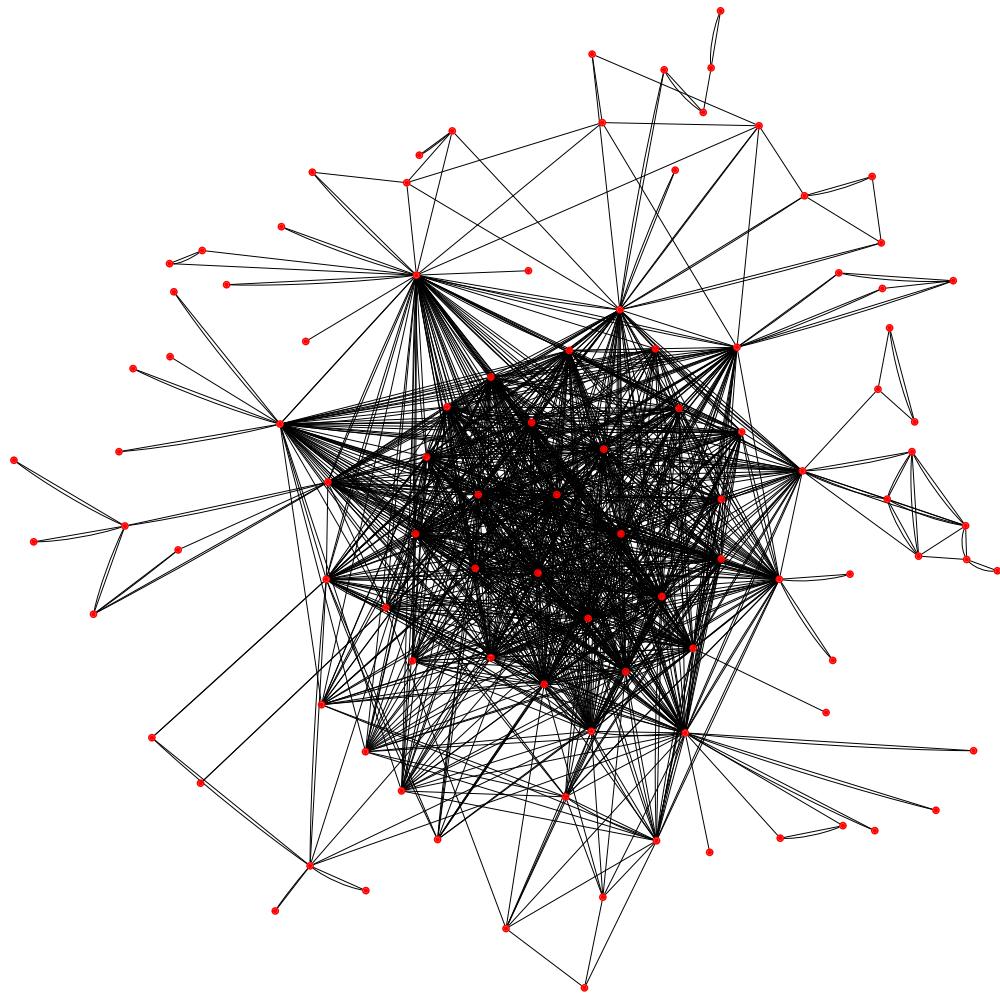


Abbildung 4.15: Community mit zeitlicher Korrelation der Signaturen (Blondel ( $l = 5$ ), 100 Knoten, 72% der Signaturen innerhalb eines Monats)

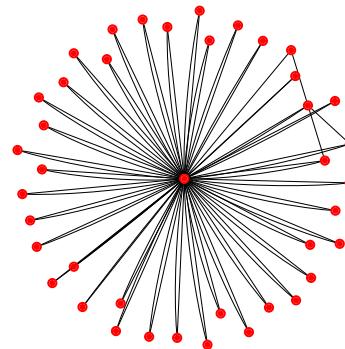


Abbildung 4.16: Community mit zeitlicher Korrelation der Signaturen (Blondel ( $l = 5$ ), 41 Knoten, 84 % der Signaturen innerhalb eines Monats)

terne. Um solche Extremfälle auszuschliessen, ist als zusätzliches Kriterium für die Erkennung von Keysigning-Parties ein hoher durchschnittlicher Grad der Knoten denkbar.

Diese Ergebnisse widerlegen nicht die Annahme, dass die Vernetzung im Web of Trust im wesentlichen von Keysigning-Parties und den sozialen Kontakten der Teil-

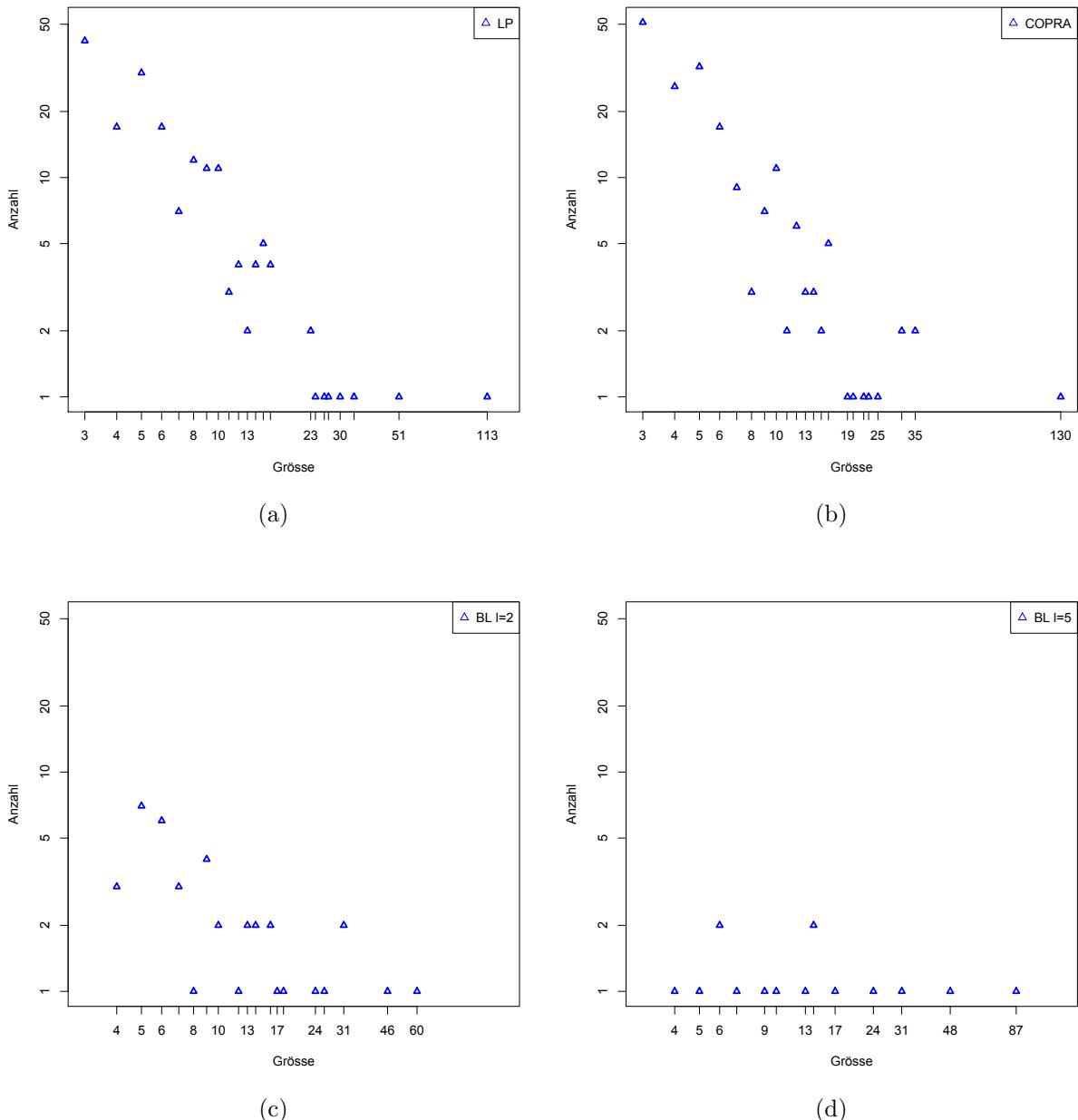


Abbildung 4.17: Zuweisung von Domains zu SLDs abhängig von der Community-Grösse

nehmer beeinflusst wird und damit unter anderem von Gruppenzugehörigkeit bestimmt wird. Allerdings muss festgehalten werden, dass die hier verwendeten Methoden nicht geeignet sind, um den Entstehungsmechanismus entlang dieser Annahme befriedigend zu erklären.

Ein Grund dafür ist sicherlich, dass die Struktur des Web of Trust nur zu einem bestimmten, festen Zeitpunkt betrachtet wurde. Um die Entstehungsdynamik zu verstehen, könnte alternativ die Entwicklung des Netzwerks tatsächlich über die Zeit nachvollzogen werden. Das Keyserver-Netzwerk speichert nicht nur den momentanen Zustand des Netzwerks, sondern implizit auch den Entstehungszeitpunkt jedes Schlüssels und jeder Signatur. Die im Rahmen dieser Arbeit entwickelte Software

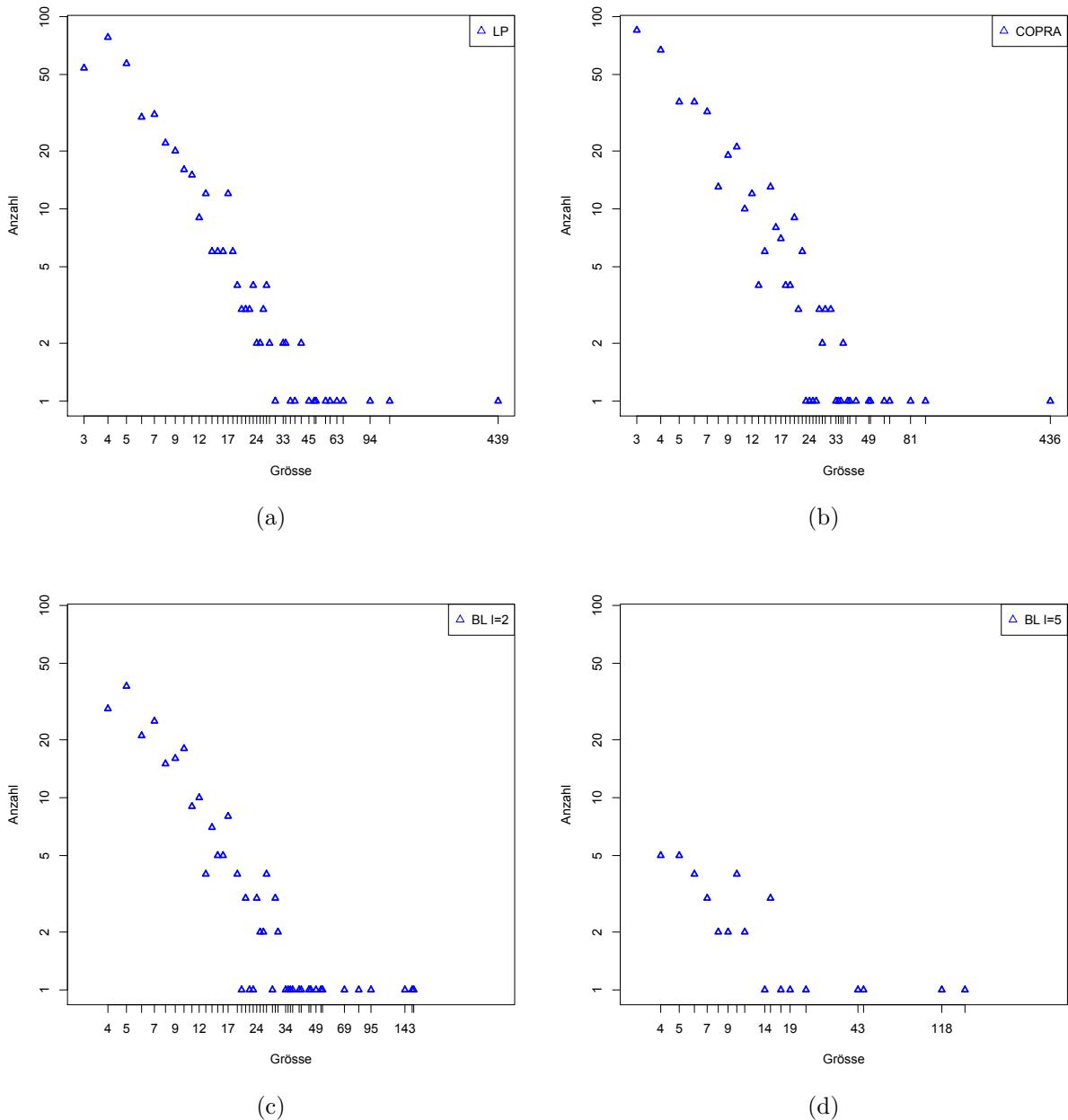


Abbildung 4.18: Verteilung der Grösse der von einer SLD dominierten Communities

speichert diese Zeitstempel in einer Datenbank und macht so die gesamte Entwicklungsgeschichte des Web of Trust verfügbar (siehe Abschnitt 3.2.2). Anhand dieser Daten könnte nachvollzogen werden, wie und in welcher Form die Communities tatsächlich entstehen und sich entwickeln.

Dass nur sehr wenige grössere Communities mit zeitlicher Korrelation der Signaturen gefunden wurden, obwohl regelmässig grössere Keysigning-Parties stattfinden (siehe Abschnitt 2.2.5) ist bei näherer Betrachtung durchaus plausibel. Dass eine Keysigning-Party sich in der Struktur markant abhebt, setzt voraus, dass ihre Teilnehmer ausser der Teilnahme an dieser Party keine wesentliche Signaturaktivität starten. Würden sie weiterhin an Signaturaktivitäten teilnehmen, würde die Struktur der Keysigning-Party mit der Zeit durch Signaturen von und zu Nicht-Teilnehmern

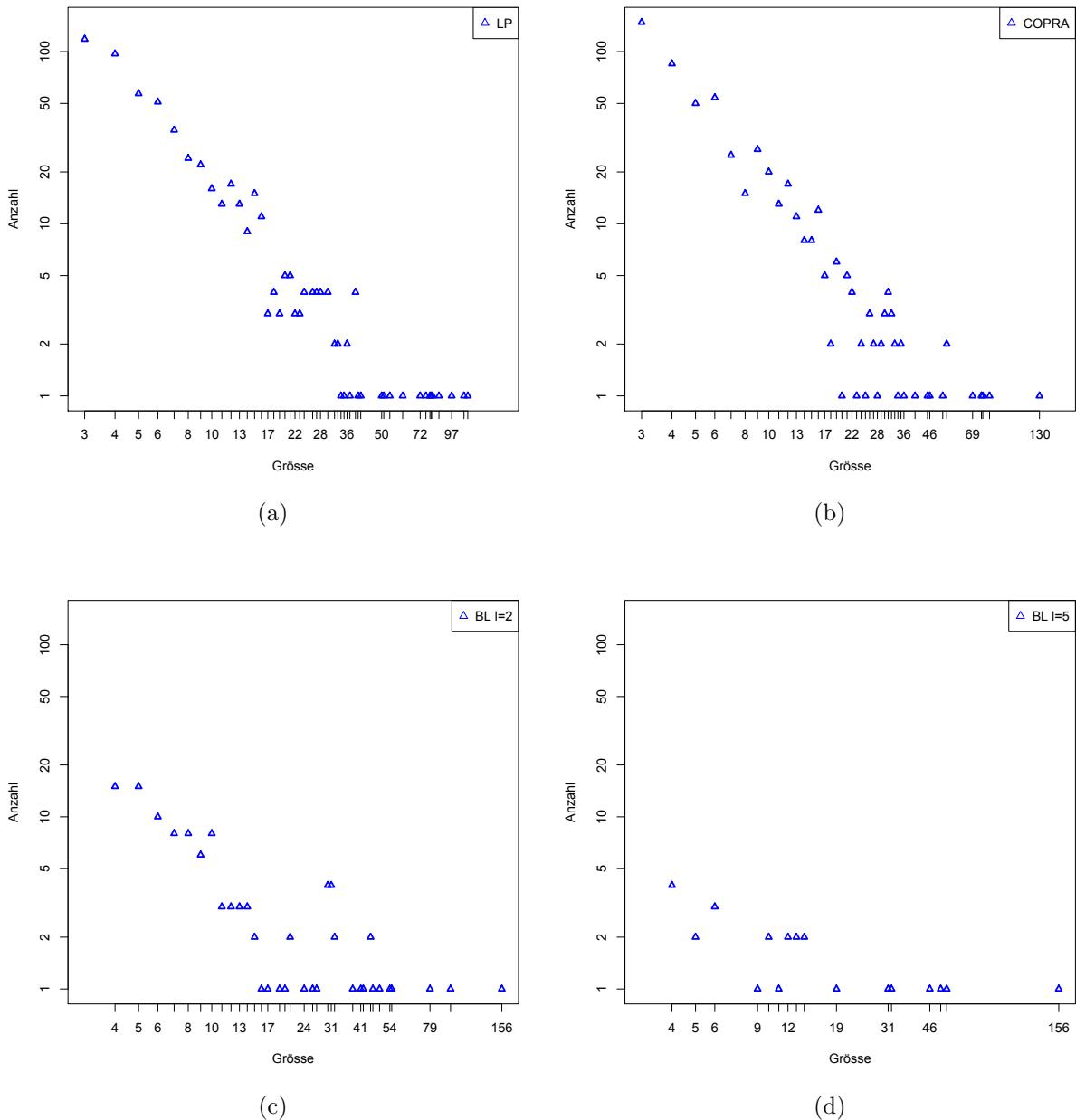


Abbildung 4.19: Verteilung der Grösse von Communities mit zeitlicher Korrelation der Signaturen

“verwischt” werden. Aufgefunden wurden also vermutlich nur die Keysigning-Parties mit einer Mehrheit von insgesamt eher inaktiven Teilnehmern. Gleches kann für Teilnehmer angenommen werden, die sich anhand von Gruppenzugehörigkeit vernetzen. Auch hier würde die Struktur mit der Zeit “unschärfer” werden, wenn ihre Mitglieder weiterhin Signierungen vornehmen würden. Es kann also angenommen werden, dass die kleineren und insbesondere die klar zuordnenbaren Communities tendenziell Teilnehmer enthalten, die in Bezug auf die Teilnahme am Web of Trust eher inaktiv sind. Um diese Vermutung zu bestätigen oder zu widerlegen, könnte das Signaturverhalten der Teilnehmer über die Zeit mit ihrer Community-Mitgliedschaft verglichen und korreliert werden.

Es scheint unrealistisch, dass die Mitgliedschaft in einer Gruppe – etwa einer Firma, akademischen Einrichtung oder einem Open-Source-Projekt – die sozialen Kontakte einer Person vollständig charakterisiert. Daneben kann eine Person noch ein weites Netzwerk an Freunden oder Bekannten haben, die diese Mitgliedschaft nicht teilen. Trotzdem können sich zu ihnen Signaturen ergeben, die einen wesentlichen Teil der Signaturen dieser Person ausmachen können. Außerdem ist anzunehmen, dass eine Vielzahl solcher Gruppen nicht über eine gemeinsame Domain verfügen, und damit mit den hier vorliegenden Daten schlicht nicht sichtbar sind. Zwar wurde versucht, der annehmenden Komplexität der Gruppenzugehörigkeiten durch überlappende Communities zu begegnen. Allerdings konnte der verwendete Algorithmus keine signifikanten Überlappungseigenschaften liefern. Die Frage, inwiefern sich die einzelnen Communities sozialen Gruppen zuordnen lassen, konnte also nur unvollständig beantwortet werden.

Es zeigt sich, dass dem Web of Trust kein einzelner, einfacher Entstehungsmechanismus der hier angenommenen Form zugrundeliegt. Es kann nur unzureichend beschrieben werden als eine Ansammlung von einzelnen, einfach entstandenen Bausteinen, als die hier Communities angenommen wurden. Implizit vorausgesetzt wird dabei auch, dass diese zum Beispiel im Fall einer Keysigning-Party nach ihrer Entstehung keine wesentliche weitere Vernetzungsaktivität zeigen. Zwar trifft dies für einen Teil der Communities – gerade die, die eine zeitliche Korrelation der Signaturen zeigen oder einer Second-Level-Domain zugeordnet werden können – durchaus zu. Für einen erheblichen Anteil der Teilnehmer – beispielsweise die Mitglieder der sehr grossen Community aus der COPRA-Berechnung – scheint die Vernetzung jedoch so vielfältig zu sein, dass solche einfachen Mechanismen nicht mehr sichtbar sind und “in der Masse verschwinden”. Bei der COPRA-Zerlegung ist es etwa naheliegend, dass die sehr grosse zentrale Community die Mehrzahl der aktiveren Teilnehmer enthält. Hier ergibt sich eine Ähnlichkeit zur Struktur der Zusammenhangskomponenten, bei denen die grösste alle aktiven einigermassen aktiven Teilnehmer zu enthalten scheint (siehe Abschnitt 4.2.1).

Diese Punkte können anhand des Beispiels von Debian illustriert werden: Das Debian-Projekt als tatsächliche Community, in der PGP und das Web of Trust eine besonders wichtige Rolle spielen (siehe Abschnitt 2.2.5) findet sich in den Zerlegungen nicht als eigene abgrenzte Einheit wieder. Statt dessen finden sich sehr grosse Communities, in denen Debian-Entwickler einen signifikanten Anteil der Schlüssel stellen. Im Fall der COPRA-Zerlegung ist dies die bereits erwähnte Community mit ca. 21.000 Mitgliedern. Von diesen verfügen 1.247 über eine E-Mail-Adresse in der Domain [debian.org](http://debian.org) und sind damit offensichtlich Mitglieder des Debian-Projekts. Daneben finden sich Mitglieder des Debian-Projekts in einer Vielzahl von kleineren Communities aller Größen. Die Mitgliedschaft im Debian-Projekt scheint hier also für die Vernetzung keine so wesentliche Rolle zu spielen, dass sie die Struktur dieser real existierenden Community im Web of Trust wiederspiegelt. Selbst wenn viele Debian-Entwickler mit vielen anderen Projektmitgliedern vernetzt sind, so verfügen sie offensichtlich über genügend andere Kontakte, so dass diese Struktur nicht mehr auffindbar ist.



## 5. Zusammenfassung und Ausblick

- Datensatz: Unterschied zu anderen (Wotsap...)
- Datensatz wurde benutzt, um verschiedene Aspekte des Web of Trust zu untersuchen: Komponentenstruktur, Netzwerkstatistiken der MSCC, Community-Struktur, Verbindung mit inhaltlichen Informationen, Benutzung von Algorithmen. Wurde versucht, die Auswirkungen von Kryptoanalytischen Durchbrüchen abzuschätzen.
- viele Schluessel in Datenbank, aber nur wenige davon nehmen am Web of Trust teil.
- Grosse Mehrzahl der PGP-Benutzer (Leute mit Schluesseln, kann allerdings nicht vorausgesetzt werden, dass diese tatsaechlich benutzt werden) authentifiziert Schluessel nicht
- mesoscopic: MSCC hat ausgeprägte modulare Struktur wie viele andere soziale Netzwerke
- Communities stellen Bausteine dar, die das Gesamtnetz ergeben.
- Einige dieser Communities weisen Charakteristika auf, die eine Zuordnung zu einer sozialen Gruppe oder Keysigning-Party erlauben.
- Auflösungslimit hurts: Zusammenpacken in grosse Communities bringt zwar möglicherweise bessere Modularity, erschwert aber die inhaltliche Analyse.
- Allerdings sind diese Mechanismen nicht geeignet, um die Entstehungsdynamik insgesamt befriedigend zu erklaeren.
- Obwohl das nicht untersucht wurde, kann doch vermutet werden, dass die überwiegende Mehrheit der Schluessel in der MSCC von Personen stammt, die aus der Computersicherheit, Open-Source-Gemeinde stammen oder einen akademischen Background haben. Allerdings gibt es auch eine ganze Reihe Communities, die sich einzelnen Firmen zuordnen lassen (sony, cisco).

Schlüssel und Signaturen enthalten grundsätzlich den Zeitpunkt ihrer Erzeugung und im Keyserver-Netzwerk sind alle jemals öffentlich gemachten Schlüssel abrufbar. Im Unterschied zu vielen anderen sozialen Netzwerken ist hier die gesamte Entwicklungsgeschichte verfügbar. Der in dieser Arbeit berechnete Datensatz könnte damit ein interessanter Ansatzpunkt sein, um die Entstehungsdynamik eines komplexen sozialen Netzwerks im Detail zu untersuchen.

## A. Schema der Tabellen

FIXME Schema der Tabellen...

Die Zuordnung einzelner Schlüssel zu ihren starken Zusammenhangskomponenten erfolgt über die Tabelle *component\_ids*. Diese wird erst in einem separaten Schritt befüllt, nachdem die Komponentenstruktur berechnet wurde.

```
(SELECT signer, signee
FROM sigs INNER JOIN keys ON sigs.signer = keys.keyid
WHERE
    (keys.revoketime IS NULL OR keys.revoketime > $timestamp)
    AND (keys.exptime IS NULL OR keys.exptime > $timestamp)
    AND (sigs.revoketime IS NULL OR sigs.revoketime > $timestamp)
    AND (sigs.exptime IS NULL OR sigs.exptime > $timestamp))
INTERSECT
(SELECT signer, signee
FROM sigs INNER JOIN keys ON sigs.signee = keys.keyid
WHERE
    (keys.revoketime IS NULL OR keys.revoketime > $timestamp)
    AND (keys.exptime IS NULL OR keys.exptime > $timestamp)
    AND (sigs.revoketime IS NULL OR sigs.revoketime > $timestamp)
    AND (sigs.exptime IS NULL OR sigs.exptime > $timestamp))"
```

Abbildung A.1: Abfrage aller zum Zeitpunkt \$timestamp gültigen Signaturen

Als Beispiel gibt Abb. A.1 ein SQL-Statement an, mit dem alle gültigen (d.h. nicht zurückgezogenen oder abgelaufenen) Signaturen zu einem bestimmten Zeitpunkt abgerufen werden können.



## B. Analysewerkzeuge

basic-properties-mpi	Verteilung von Eingangs- und Ausgangsgraden, Komponentengrößen, Nachbarschaften, Durchmesser, Radius, durchschnittliche Pfadlängen (parallelisiert mittels MPI)
betweenness-mpi	Berechnung der Betweenness-Zentralität (parallelisiert mittels MPI)
clustering-coefficient-mpi	Berechnung des Clustering Coefficient (parallelisiert mittels MPI)
export	Export des Graphen in verschiedene Dateiformate (igraph, Cytoscape)
meta-graph	Zeichnung der Struktur der starken Zusammenhangskomponenten
db-scc-information	Befüllt die SQL-Datenbank mit der Zuordnung von Knoten zu Zusammenhangskomponenten
dump-sql	Legt die extrahierten Daten einmalig in der SQL-Datenbank ab
investigate-component	Untersucht einzelne Zusammenhangskomponente in Bezug auf Herkunft der UserIDs (Domains) und den Zeitraum der Entstehung der Signaturen
mscc-size	Entwicklung der Schlüsselanzahl der grössten starken Zusammenhangskomponente
simple-stats	Statistiken über die Verwendung bestimmter Schlüsseleigenschaften (Algorithmen, Schlüssellängen usw.)
time	Entwicklung der Verwendung von Signatur- und Public-Key-Algorithmen und deren Schlüssellängen

Tabelle B.1: Foobar

# Abbildungsverzeichnis

2.1	Beispiel für die Berechnung der Gültigkeit von Schlüsseln . . . . .	11
4.1	Größenverteilung der starken Zusammenhangskomponenten . . . . .	34
4.2	Sternförmige Struktur der starken Zusammenhangskomponenten bis zur Grösse 6 (rot = Grösse 6-10, grün = Grösse 11-20, blau = Grösse 21-30, violett = Grösse $\geq 31$ , gelb = MSCC) . . . . .	35
4.3	Verteilung der eingehenden (a) und ausgehenden (b) Knotengrade in der grössten starken Zusammenhangskomponente . . . . .	37
4.4	Kumulative Verteilungsfunktion von $\frac{d^+(u)}{d^-(u)}$ für alle Knoten $u$ in der MSCC (a). Verteilung von $k_{nn}$ über alle ausgehenden Grade (b). . . . .	39
4.5	Entwicklung der Grösse der grössten starken Zusammenhangskomponente, wenn Knoten zufällig oder nach absteigender Grösse entfernt werden . . . . .	40
4.6	Verteilung der (gerundeten) durchschnittlichen Distanzen (a) und der Eccentricity (b) in der grössten starken Zusammenhangskomponente .	43
4.7	Kumulative Verteilungsfunktion der Nachbarschaftsgrössen in der grössten starken Zusammenhangskomponente . . . . .	44
4.8	Zeitliche Entwicklung der Grösse des gesamten Schlüsselbestandes (a) und der grössten starken Zusammenhangskomponente (b) . . . . .	45
4.9	Rate neuer PGP-Schlüssel, die RSA oder DSA benutzen, für den gesamten Schlüsselbestand (a) und die grösste starke Zusammenhangskomponente (b) in monatlichen Abständen . . . . .	46
4.10	Größenverteilung der Communities . . . . .	49
4.11	Community mit modularen Untereinheiten (Grösse 2271, Blondel ( $l = 5$ ), Force-directed Layout) . . . . .	50
4.12	Grosse Community mit modularer Struktur (Berechnet mit Fast-Modularity, Force-directed layout) . . . . .	51
4.13	Struktur der Communities (COPRA ( $v = 1$ ) (rot: Grösse $< 10$ , grün: Grösse $< 100$ , blau: Grösse $< 1000$ , gelb: Grösse $> 1000$ )) . . . . .	53
4.14	Struktur der Communities (Blondel ( $l = 5$ ) (Knotenfärbung wie in Abb.4.13)) . . . . .	54

4.15 Community mit zeitlicher Korrelation der Signaturen (Blondel ( $l = 5$ ), 100 Knoten, 72% der Signaturen innerhalb eines Monats) . . . . .	57
4.16 Community mit zeitlicher Korrelation der Signaturen (Blondel ( $l = 5$ ), 41 Knoten, 84 % der Signaturen innerhalb eines Monats) . . . . .	57
4.17 Zuweisung von Domains zu SLDs abhängig von der Community-Grösse	58
4.18 Verteilung der Grösse der von einer SLD dominierten Communities . .	59
4.19 Verteilung der Grösse von Communities mit zeitlicher Korrelation der Signaturen . . . . .	60
A.1 Abfrage aller zum Zeitpunkt \$timestamp gültigen Signaturen . . . . .	65

# Literaturverzeichnis

- [AlJB00] R. Albert, H. Jeong und A.-L. Barabasi. Error and attack tolerance of complex networks. *Nature* 406(6794), Juli 2000, S. 378–382.
- [BaAl99] A.-L. Barabasi und R. Albert. Emergence of Scaling in Random Networks. *Science* 286(5439), 1999, S. 509–512.
- [BBBP<sup>+</sup>07] E. Barker, W. Barker, W. Burr, W. Polk und M. Smid. NIST Special Publication 800-57 Part 1, Recommendation for Key Management - Part 1: General (Revised). [http://csrc.nist.gov/groups/ST/toolkit/key\\_management.html](http://csrc.nist.gov/groups/ST/toolkit/key_management.html), 2007.
- [BnPSDGA04] M. Boguñá, R. Pastor-Satorras, A. Díaz-Guilera und A. Arenas. Models of social networks based on social distance attachment. *Phys. Rev. E* 70(5), Nov 2004, S. 056122.
- [Bren08] V. A. Brennen. The Keysigning Party HOWTO. [http://www.cryptnet.net/fdp/crypto/keysigning\\_party/en/keysigning-party.html](http://www.cryptnet.net/fdp/crypto/keysigning_party/en/keysigning-party.html), 2008.
- [BrEr04] U. Brandes und T. Erlebach. Fundamentals. In *Network Analysis* [BrEr05], S. 7–15.
- [BrEr05] U. Brandes und T. Erlebach (Hrsg.). *Network Analysis: Methodological Foundations [outcome of a Dagstuhl seminar, 13-16 April 2004]*. Band 3418 der *Lecture Notes in Computer Science*, Springer, 2005.
- [BrSc04] M. Brinkmeier und T. Schank. Network Statistics. In Brandes und Erlebach [BrEr05], S. 293–317.
- [BrSc06] D. Brondsema und A. Schamp. Konfidi: Trust Networks Using PGP and RDF. In T. Finin, L. Kagel und D. Olmedilla (Hrsg.), *MTW*, Band 190 der *CEUR Workshop Proceedings*. CEUR-WS.org, 2006.
- [CaBH02] S. Capkun, L. Buttyán und J.-P. Hubaux. Small worlds in security systems: an analysis of the PGP certificate graph. In *NSPW '02: Proceedings of the 2002 workshop on New security paradigms*, New York, NY, USA, 2002. ACM, S. 28–35.
- [CaSW05] P. J. Carrington, J. Scott und S. Wasserman. *Models and Methods in Social Network Analysis*: Structural analysis in the social sciences. Cambridge University Press. 2005.

- [CDFS<sup>+</sup>07] J. Callas, L. Donnerhacke, H. Finney, D. Shaw und R. Thayer. OpenPGP Message Format. RFC 4880 (Proposed Standard), November 2007. Updated by RFC 5581.
- [CDFT98] J. Callas, L. Donnerhacke, H. Finney und R. Thayer. OpenPGP Message Format. RFC 2440 (Proposed Standard), November 1998. Obsoleted by RFC 4880.
- [ClNM04] A. Clauset, M. E. J. Newman und C. Moore. Finding community structure in very large networks. *Phys. Rev. E* 70(6), Dec 2004, S. 066111.
- [ClSN09] A. Clauset, C. R. Shalizi und M. E. J. Newman. Power-Law Distributions in Empirical Data. *SIAM Review* 51(4), 2009, S. 661–703.
- [CsNe06] G. Csárdi und T. Nepusz. The igraph software package for complex network research. *InterJournal Complex Systems*, 2006. 1695.
- [Dell07] M. Dell’Amico. Mapping Small Worlds. In M. Hauswirth, A. Wierzbicki, K. Wehrle, A. Montresor und N. Shahmehri (Hrsg.), *Peer-to-Peer Computing*. IEEE Computer Society, 2007, S. 219–228.
- [FoBa07] S. Fortunato und M. Barthelemy. Resolution limit in community detection. *Proceedings of the National Academy of Sciences* 104(1), 2007, S. 36–41.
- [Fort10] S. Fortunato. Community detection in graphs. *Physics Reports* 486(3-5), 2010, S. 75 – 174.
- [GiNe02] M. Girvan und M. E. J. Newman. Community structure in social and biological networks. *Proc Natl Acad Sci U S A* 99(12), Jun 2002, S. 7821–7826.
- [Gnu10] The GNU Privacy Guard. <http://www.gnupg.org>, May 2010.
- [GodC09] B. H. Good, Y.-A. de Montjoye und A. Clauset. The performance of modularity maximization in practical contexts, 2009.
- [Greg10] S. Gregory. Finding overlapping communities in networks by label propagation. Arxiv preprint arXiv:0910.5516, February 2010.
- [HeGu09] J. Heikkilä und A. Gurto. Filtering SPAM in P2PSIP Communities with Web of Trust. In A. U. Schmidt und S. Lian (Hrsg.), *MobiSec*, Band 17 der *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*. Springer, 2009, S. 110–121.
- [KAFL<sup>+</sup>10] T. Kleinjung, K. Aoki, J. Franke, A. Lenstra, E. Thomé, J. Bos, P. Gaudry, A. Kruppa, P. Montgomery, D. A. Osvik, H. te Riele, A. Timofeev und P. Zimmermann. Factorization of a 768-bit RSA modulus. Cryptology ePrint Archive, Report 2010/006, 2010. <http://eprint.iacr.org/2010/006>.

- [LADW05] L. Li, D. Alderson, J. C. Doyle und W. Willinger. Towards a Theory of Scale-Free Graphs: Definition, Properties, and Implications. *Internet Mathematics* 2(4), March 2005, S. 431–523.
- [LaFo09] A. Lancichinetti und S. Fortunato. Community detection algorithms: A comparative analysis. *Phys. Rev. E* 80(5), Nov 2009, S. 056117.
- [LeNe08] E. A. Leicht und M. E. J. Newman. Community Structure in Directed Networks. *Phys. Rev. Lett.* 100(11), Mar 2008, S. 118703.
- [McHP09] C. McDonald, P. Hawkes und J. Pieprzyk. Differential Path for SHA-1 with complexity  $O(2^{52})$ . Cryptology ePrint Archive: Report 2009/259 <http://eprint.iacr.org/2009/259>, June 2009.
- [MeOV96] A. J. Menezes, P. C. van Oorschot und S. A. Vanstone. *Handbook of Applied Cryptography*. CRC Press. 1996.
- [MiTZ03] Y. Minsky, A. Trachtenberg und R. Zippel. Set reconciliation with nearly optimal communication complexity. *IEEE Transactions on Information Theory* 49(9), 2003, S. 2213–2218.
- [NePa03] M. E. J. Newman und J. Park. Why social networks are different from other types of networks. *Phys. Rev. E* 68(3), Sep 2003, S. 036122.
- [Newm02] M. E. J. Newman. Assortative Mixing in Networks. *Phys. Rev. Lett.* 89(20), Oct 2002, S. 208701.
- [Newm03] M. E. J. Newman. The Structure and Function of Complex Networks. *SIAM Review* 45(2), 2003, S. 167–256.
- [NMCM09] V. Nicosia, G. Mangioni, V. Carchiolo und M. Malgeri. Extending the definition of modularity to directed graphs with overlapping communities. *Journal of Statistical Mechanics: Theory and Experiment* 2009(03), 2009, S. P03024.
- [Noac09] A. Noack. Modularity clustering is force-directed layout. *Phys. Rev. E* 79(2), Feb 2009, S. 026102.
- [PSVV01] R. Pastor-Satorras, A. Vázquez und A. Vespignani. Dynamical and Correlation Properties of the Internet. *Phys. Rev. Lett.* 87(25), Nov 2001, S. 258701.
- [RoBe08] M. Rosvall und C. T. Bergstrom. Maps of random walks on complex networks reveal community structure. *Proceedings of the National Academy of Sciences* 105(4), 2008, S. 1118–1123.
- [SKS] The SKS keyserver pool. <http://www.sks-keyservers.net/status/>. [Online; Stand 18. Mai 2010].
- [SoHM05] C. Song, S. Havlin und H. A. Makse. Self-similarity of complex networks. *Nature* 433(7024), Januar 2005, S. 392–395.

- [SSAL<sup>+</sup>09] M. Stevens, A. Sotirov, J. Appelbaum, A. K. Lenstra, D. Molnar, D. A. Osvik und B. de Weger. Short Chosen-Prefix Collisions for MD5 and the Creation of a Rogue CA Certificate. In *CRYPTO*, 2009, S. 55–69.
- [Wiki10] Wikipedia. Pretty Good Privacy — Wikipedia, The Free Encyclopedia. [http://en.wikipedia.org/w/index.php?title=Pretty\\_Good\\_Privacy&oldid=353992410](http://en.wikipedia.org/w/index.php?title=Pretty_Good_Privacy&oldid=353992410), 2010. [Online; Stand 5. April 2010].