Karlsruhe Institute of Technology

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
Betreuender Mitarbeiter:	Dipl.-Inform. Ralph Holz

Bearbeitungszeit: 08. Monat 2009 – 01. Monat 2010

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

Tübingen, den ??. ?????? 200?

Inhaltsverzeichnis

1 Einleitung **1**
1.1 Zielsetzung der Arbeit . . . 1
1.2 Gliederung der Arbeit . . . 1

2 Grundlagen **3**
2.1 Kryptographie mit öffentlichen Schlüsseln . . . 3
2.1.1 Prinzip . . . 3
2.1.2 Authentisierung von Schlüsseln . . . 3
2.1.2.1 Zentrale PKI . . . 3
2.1.2.2 Web of Trust . . . 3
2.2 PGP/GnuPG . . . 3
2.2.1 Geschichte von PGP und GnuPG . . . 3
2.2.2 Eigenschaften/Fähigkeiten der Implementierungen . . . 3
2.2.3 Das GnuPG-Vertrauensmodell . . . 3
2.2.4 Soziale Komponente des Web of Trust . . . 5
2.3 Der OpenPGP-Standard (RFC4880) . . . 5
2.3.1 Paketformat v4 . . . 5
2.3.2 Unterschiede v3 . . . 5
2.4 Graphentheorie allgemein . . . 5
2.5 Netzwerkanalyse . . . 5
2.5.1 Netzwerkstatistiken . . . 5
2.5.2 Communities . . . 5
2.6 Verwandte Arbeiten . . . 5
2.6.1 Analysen des OpenPGP-Web of Trust . . . 5
2.6.2 Community-Strukturen allgemein . . . 5

3 Methoden und Materialien **7**
3.1 Warum eigene Software? . . . 7
3.2 Design und Implementierung . . . 9
3.2.1 Der SKS Keyserver . . . 9
3.2.2 Eigene Software . . . 9
3.2.2.1 Datenextraktion . . . 10
3.2.2.2 Datenauswertung . . . 12
3.3 Community-Analyse . . . 13

4 Ergebnisse **15**
4.1 Allgemeine Merkmale des Netzwerkes . . . 15
4.2 Struktur der starken Zusammenhangskomponenten . . . 15
4.3 Zusammenhangskomponenten und Communities . . . 15

4.3.1 Communities in gerichteten Graphen 15

5 Diskussion FIXME **25**
5.1 Abschnitt 1 25

6 Zusammenfassung und Ausblick **27**

Literaturverzeichnis **29**

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...

1.1 Zielsetzung der Arbeit

Was ist die Aufgabe der Arbeit?

Bla fasel...

1.2 Gliederung der Arbeit

Was enthalten die weiteren Kapitel?

Bla fasel...

2. Grundlagen

2.1 Kryptographie mit öffentlichen Schlüsseln

2.1.1 Prinzip

Klassische Kryptographie basiert auf einem *Geheimnis* oder *privaten Schlüssel*, der beiden Kommunikationspartnern bekannt ist.

2.1.2 Authentisierung von Schlüsseln

2.1.2.1 Zentrale PKI

2.1.2.2 Web of Trust

2.2 PGP/GnuPG

2.2.1 Geschichte von PGP und GnuPG

2.2.2 Eigenschaften/Fähigkeiten der Implementierungen

2.2.3 Das GnuPG-Vertrauensmodell

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

Ein Schlüssel wird von GnuPG genau dann als gültig (*valid*) 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 *ultimate trust*) 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 (*marginally valid*) 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 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.

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 betrachtet.

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.

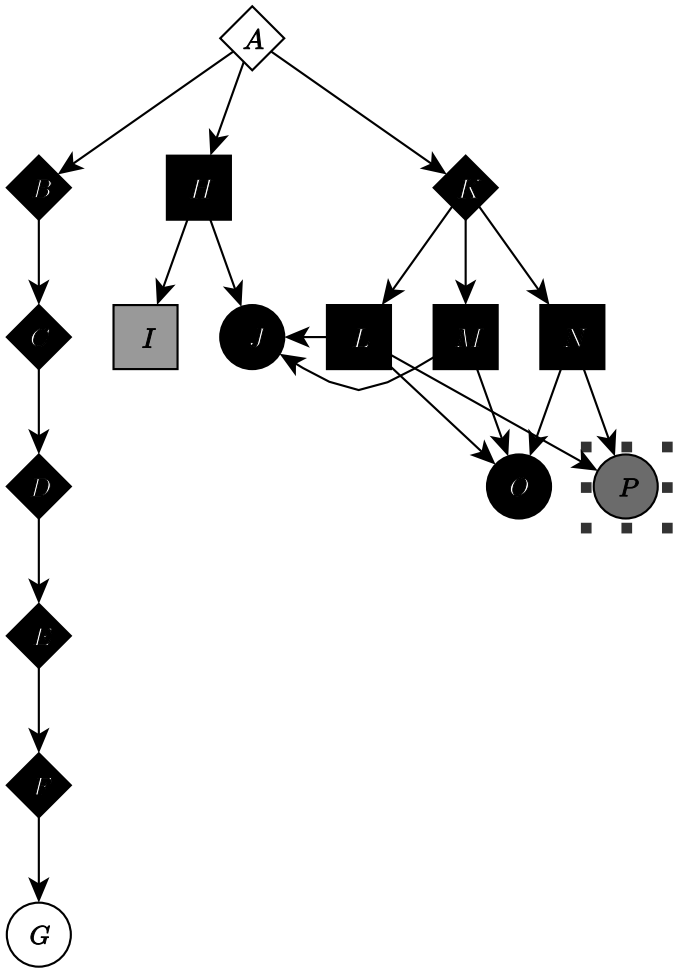

Abbildung 2.1: Beispiel für die Berechnung der Gültigkeit von Schlüsseln

2.2.4 Soziale Komponente des Web of Trust

2.3 Der OpenPGP-Standard (RFC4880)

2.3.1 Paketformat v4

2.3.2 Unterschiede v3

2.4 Graphentheorie allgemein

2.5 Netzwerkanalyse

2.5.1 Netzwerkstatistiken

2.5.2 Communities

2.6 Verwandte Arbeiten

2.6.1 Analysen des OpenPGP-Web of Trust

2.6.2 Community-Strukturen allgemein

3. Methoden und Materialien

3.1 Warum eigene Software?

Das bereits in Abschnitt 2.6 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 Analyses 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 es 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. 45100 Schlüssel, während der *wotsap*-Datensatz (MSCC-2) vom gleichen Tag nur ca. 42130 Schlüssel enthält. Diep 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 u.a. 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 fäschlicherweise 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 wiederrum bei *wotsap*, dass diese Schlüssel fälschlicherweise ausschliesst. Als Ursache dafür ergab sich in allen betrachteten Fällen der fehlerhafte Bestand des Keyservers *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. 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. fäschlicherweise 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 3.2.1 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[1] abgefragt werden.

Im Unterschied dazu kommunizieren SKS-Keyserver zur Synchronisation direkt (ohne Umwege über Mailserver) miteinander. Dazu wird ein binäres „Gossip"-Protokoll benutzt, dass 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 liesst 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 (PostgreSQL FIXME cite) abgelegt und können dort abgefragt werden. Der zweite Teil besteht aus einer Reihe von Werkzeugen zur Analyse dieser Daten entsprechend der Zielsetzung dieser Arbeit.

[1] 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 enhaltenen 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 geparst. 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 FIXME, den Zertifizierungslevel, das Erstellungsdatum der Signatur, falls vorhanden Ablauf- und Rückzugsdatum, der verwendete Signaturalgorithmus und der verwendete 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 „Oberschlüssels“ 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 müssten 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 auf einem beliebigen Recher 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.

Die Tabellen sind wie folgt strukturiert: TODO

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.revoktime IS NULL OR keys.revoktime > $timestamp)
     AND (keys.exptime IS NULL OR keys.exptime > $timestamp)
     AND (sigs.revoktime IS NULL OR sigs.revoktime > $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.revoktime IS NULL OR keys.revoktime > $timestamp)
   AND (keys.exptime IS NULL OR keys.exptime > $timestamp)
   AND (sigs.revoktime IS NULL OR sigs.revoktime > $timestamp)
   AND (sigs.exptime IS NULL OR sigs.exptime > $timestamp))"
```

Abbildung 3.1: Abfrage aller zum Zeitpunkt $timestamp gültigen Signaturen

Als Beispiel gibt Abb. 3.1 ein SQL-Statement an, mit dem alle gültigen (d.h. nicht zurückgezogenen oder abgelaufenen) Signaturen zu einem bestimmten Zeitpunkt abgerufen werden können.

Wie beschrieben wird in einem Durchlauf der gesamte Datenbestand extrahiert. Aufgrund der Integration in SKS wäre es auch problemlos möglich, den Extraktionsteil nicht als eigenständigen Prozess zu implementieren, sondern den *db*-Prozess entsprechend zu erweitern, so dass jeder neue oder geänderte Schlüssel direkt in der SQL-Datenbank abgelegt wird. Auf diese Weise könnte ein jederzeit aktueller Datenbestand erhalten werden, ohne dass er ständig komplett neu berechnet werden muss.

3.2.2.2 Datenauswertung

Der Teil zur Datenauswertung besteht aus einer Reihe von unabhängigen Werkzeugen, die die jeweiligen Analyseaufgaben implementieren. Diese sind als Kom-

basic-properties-mpi	Verteilung von Eingangs- und Ausgangsgraden, Komponentengrössen, Nachbarschaften, Durchmesser, Radius, durschnittliche Pfadlängen (parallelisiert mittels MPI
betweeness-mpi	Berechnung der Betweeness-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 3.1: Foobar

mandozeilenprogramme in eigenen Prozessen realisiert. Die Aufgaben der einzelnen Werkzeuge sind in Tab. 3.1 aufgeführt.

3.3 Community-Analyse

4. Ergebnisse

4.1 Allgemeine Merkmale des Netzwerkes

4.2 Struktur der starken Zusammenhangskomponenten

4.3 Zusammenhangskomponenten und Communities

Algorithmus	Modularity	Anzahl Communities (Grösse > 3)	Anzahl enthaltener Knoten
fastmod	0,596	552	foo
labelprop	0,658	1834	foo
cliquemod-3000	0,670	811	foo
cliquemod-500	0,677	451	foo
copra	—	1354	foo

Tabelle 4.1: Modulairty, Anzahl communities, Menge enthaltener Knoten (> 3)

Algo	TLD-S	TLD-M	SLD-S	SLD-M	SIG
fastmod	293 (53%)	247 (44%)	66 (11%)	194 (35%)	128 (23%)
labelprop	1048 (57%)	755 (41%)	277 (15%)	720 (39%)	553 (30%)
cliquemod-3000	460 (56%)	332 (40%)	104 (12%)	300 (36%)	230 (28%)
cliquemod-500	230 (50%)	209 (46%)	65 (14%)	141 (31%)	116 (25%)
copra	792 (58%)	525 (38%)	259 (19%)	533 (39%)	555 (40%)

Tabelle 4.2: SLD-TLD-Zuweisung, TIME-CORR

Modularity ist auf ung

4.3.1 Communities in gerichteten Graphen

erichteten Graphen definiert. Ebenso

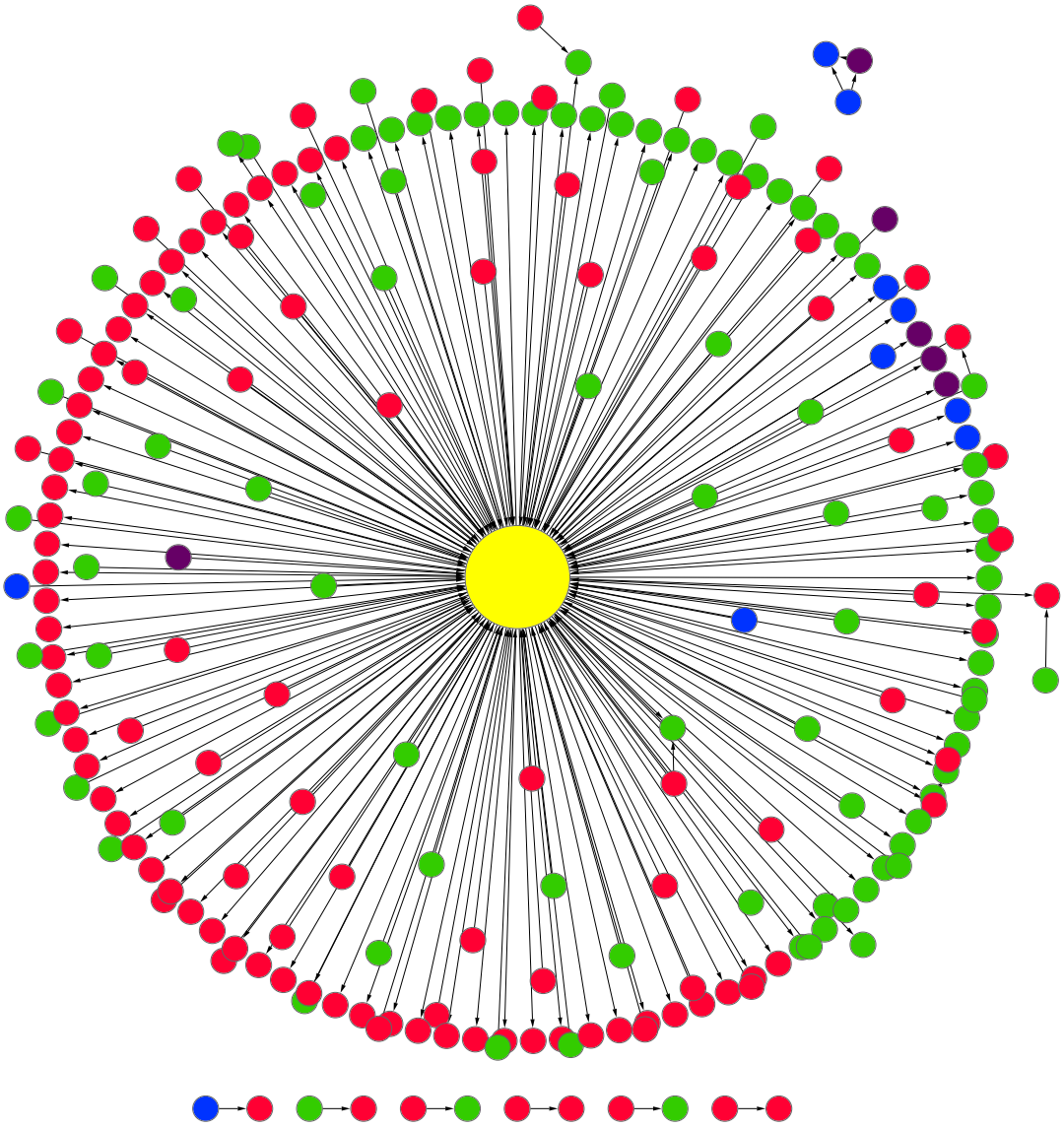

Abbildung 4.1: Struktur der starken Zusammenhangskomponenten bis zur Grösse 6 (rot = Grösse 6-10, grün = Grösse 11-20, blau = Grösse 21-30, violett = Grösse >= 31, gelb = MSCC)

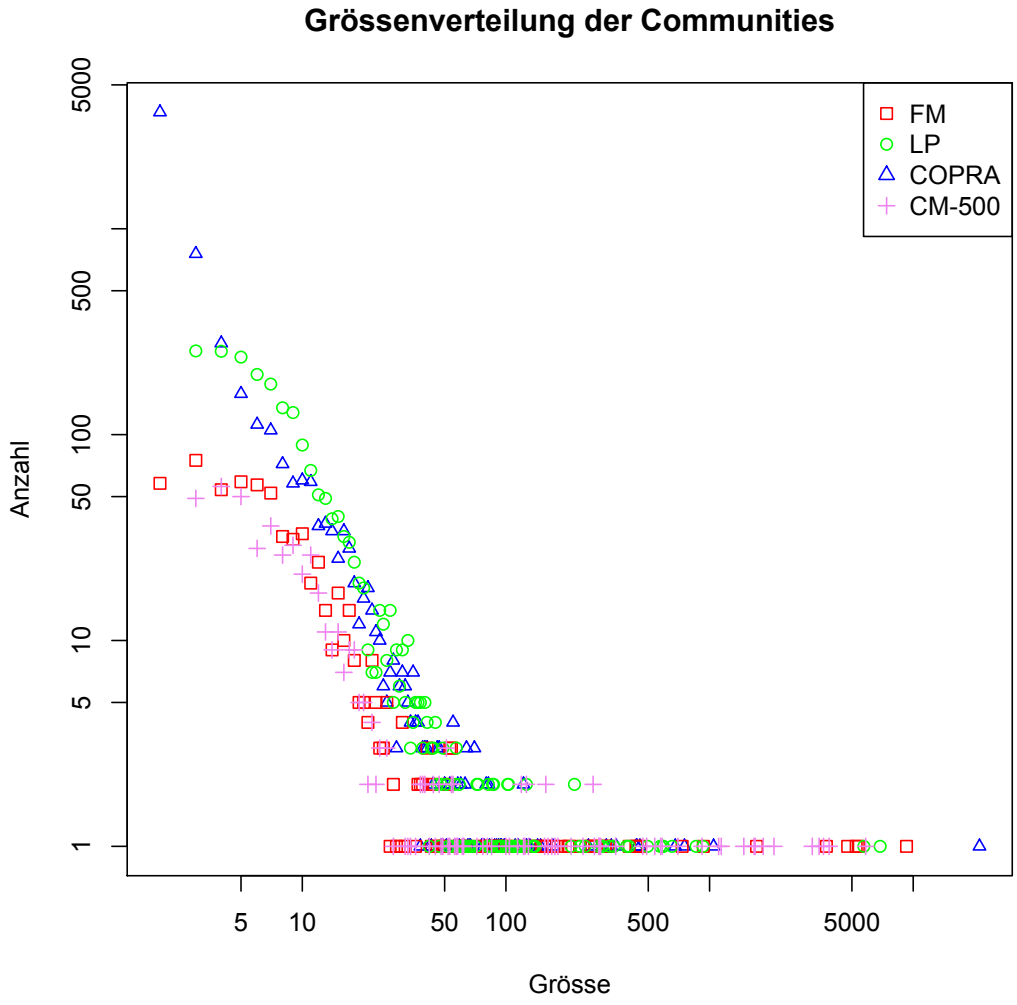

Abbildung 4.2: Grössenverteilung der von Fast-Modularity (FASTMOD), Label-Propagation (LABELPROP) und COPRA berechneten Communities

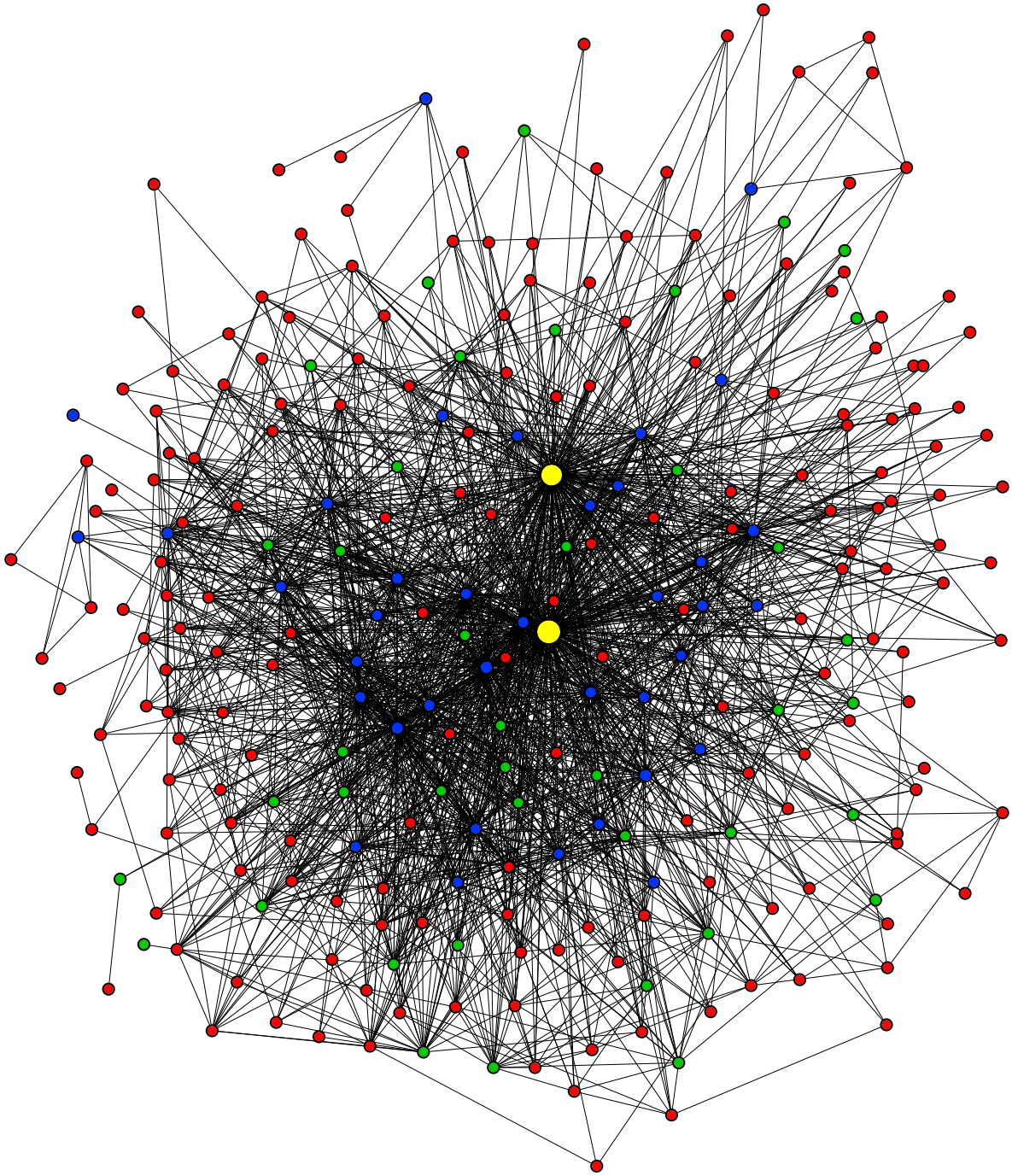

Abbildung 4.3: Struktur der Communities (Label-Propagation) (rot: Grösse < 50, grün: Grösse < 100, blau: Grösse < 1000, gelb: Grösse > 1000).

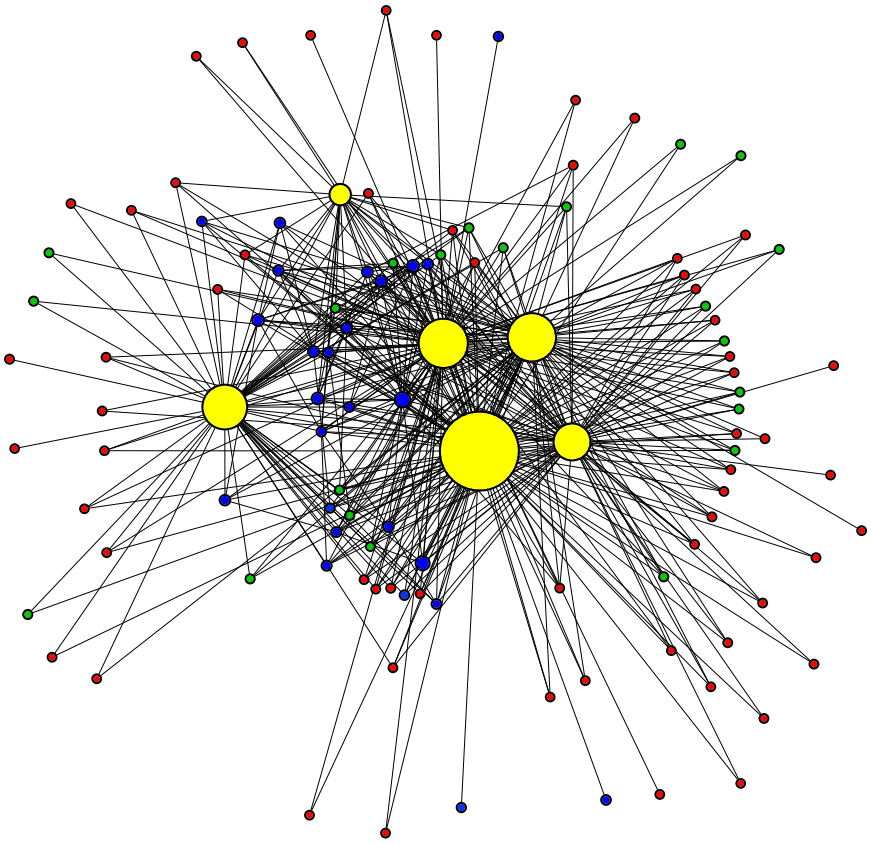

Abbildung 4.4: Struktu der Communities (Fast Modularity) (Farbkodierung wie in Abb.4.3)

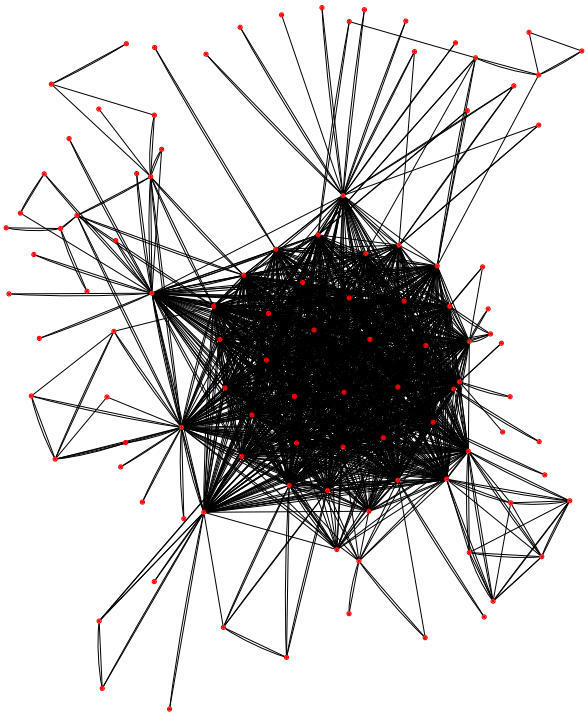

Abbildung 4.5: foo

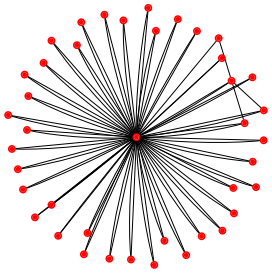

Abbildung 4.6: foo

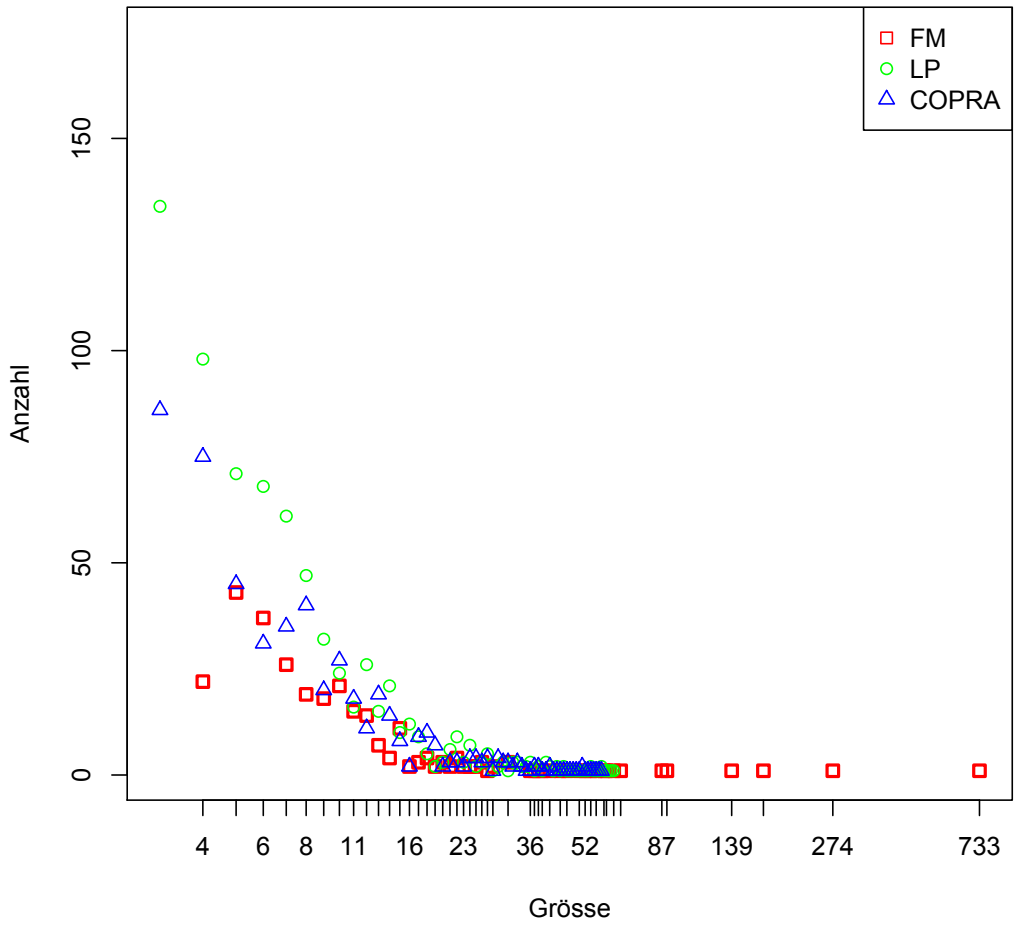

Abbildung 4.7: Zuweisung von Domains zu TLDs abhängig von der Community-Grösse

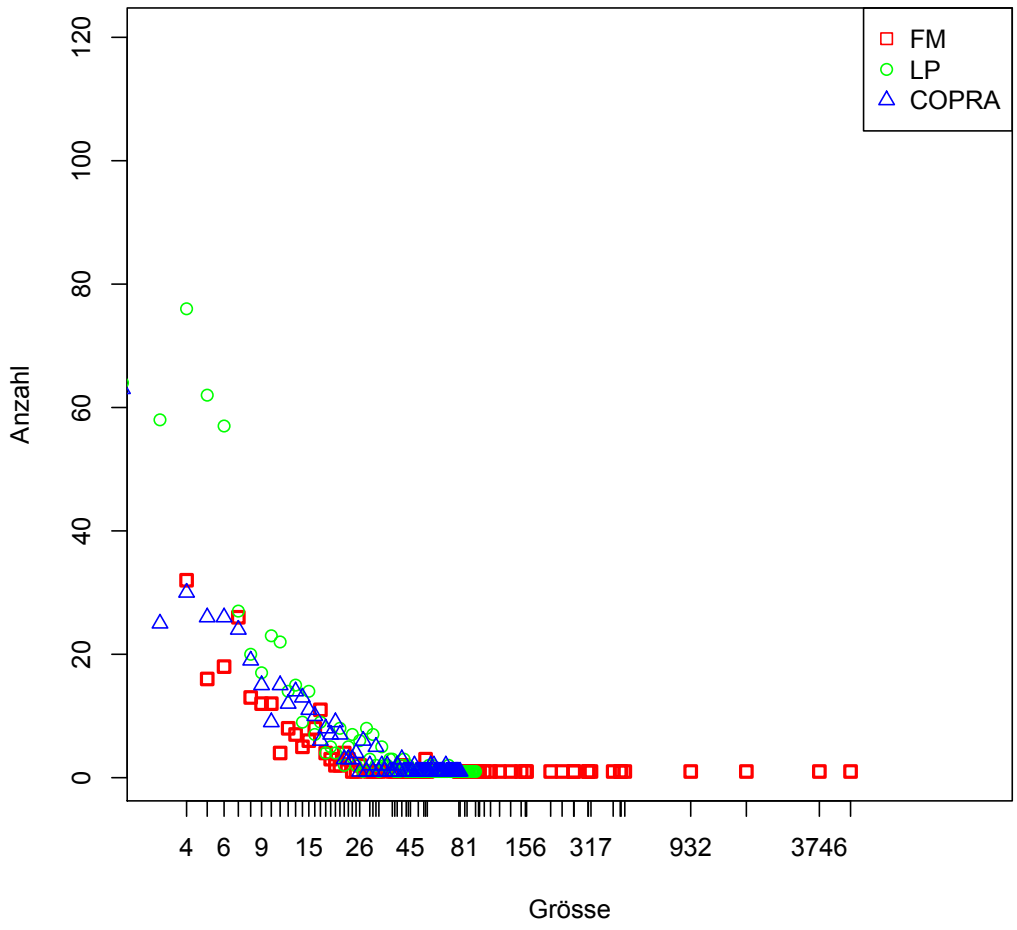

Abbildung 4.8: Verteilung der Grösse der von einer TLD dominierten Communities

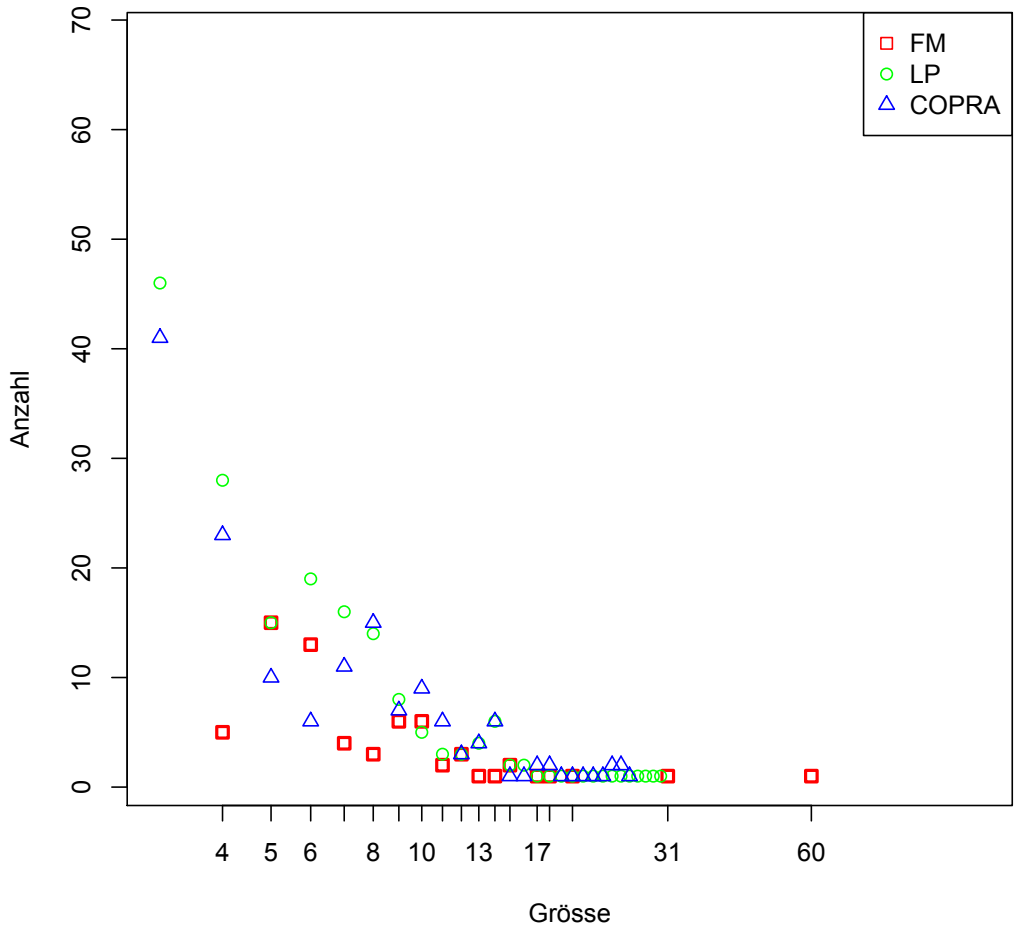

Abbildung 4.9: foo

Abbildung 4.10: Zuweisung von Domains zu SLDs abhängig von der Community-Grösse

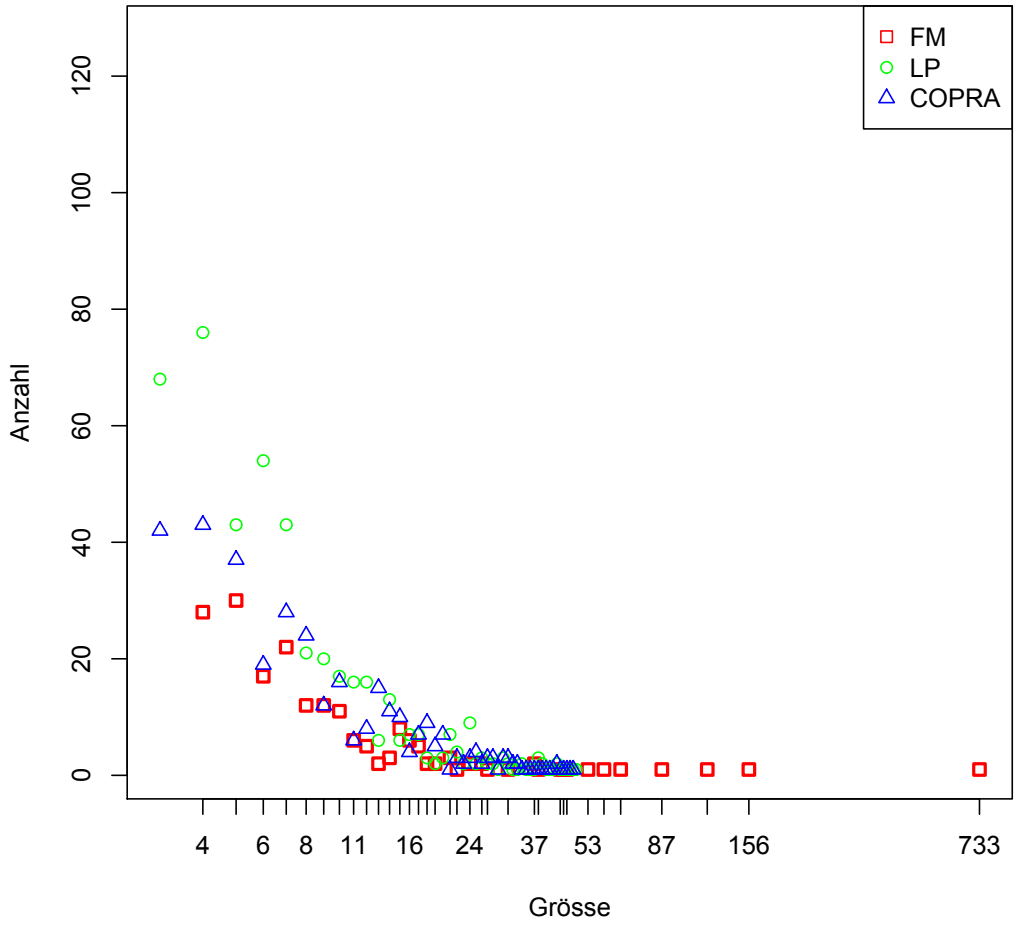

Abbildung 4.11: Verteilung der Grösse der von einer TLD dominierten Communities

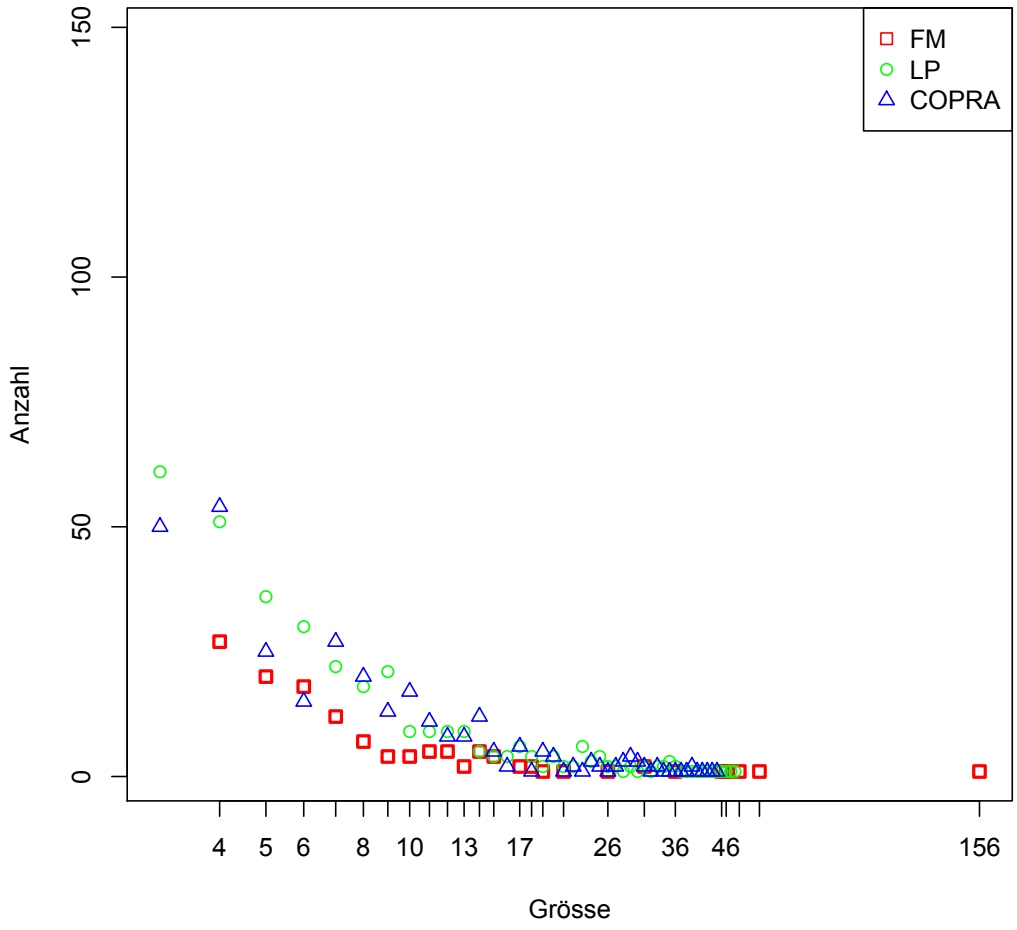

Abbildung 4.12: Verteilung der Grösse der Communities mit zeitlicher Korrelation

5. Diskussion FIXME

Die Grundlagen müssen soweit beschrieben werden, dass ein Leser das Problem und die Problemlösung versteht.Um nicht zuviel zu beschreiben, kann man das auch erst gegen Ende der Arbeit schreiben.

Bla fasel...

5.1 Abschnitt 1

Bla fasel...

6. Zusammenfassung und Ausblick

Bla fasel...

(Keine Untergliederung mehr!)

Literaturverzeichnis

[BrSc06] D. Brondsema und A. Schamp. Konfidi: Trust Networks Using PGP and RDF. In T. Finin, L. Kagal und D. Olmedilla (Hrsg.), *MTW*, Band 190 der *CEUR Workshop Proceedings*. CEUR-WS.org, 2006.

[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.

[HeGu09] J. Heikkilä und A. Gurtov. 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.

[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.