
Informatik-Seminar 2

Michael R. Albertin

Betreuer: Prof. Dr. P. Heinzmann
Cobetreuer: Prof. Dr. A. Rinkel

# 1 Inhalt

# 2 Ziel

Der vorliegende Bericht soll aufzuzeigen, wie komplexe oder umfangreiche Aufgaben, so zum Beispiel das im Folgenden beschriebene BruteForce-Cracking, auf mehreren Rechnern verteilt erledigt werden können. Auch soll er den aufmerksamen Leser anregen, sich Gedanken zu machen über die Möglichkeiten, die mit Distributed Computing and Cracking realisiert werden könnten, und auch über deren Folgen.

# 3 Einführung

## 3.1 Grundlagen

Die Computerbranche hat die Rechenleistung in den letzten Jahren enorm gesteigert. Viele PCs können heute unheimlich viel leisten, und doch gibt es Aufgaben aus den Bereichen Mathematik und Informatik, welche jeden Rechner an seine Grenzen bringen. Dazu gehören unter anderem das Finden von optimalen Golomb-Massstäben und das BruteForce-Cracking von langen Schlüsseln.

Wie schon im Vertiefungsfach Netzwerk-Sicherheit vermittelt wurde, gibt es keine absolute Sicherheit. Ein Verschlüsselungssystem ist nur so lange sicher, bis ein Weg gefunden wird, innert kurzer Zeit alle Schlüssel durchzuprobieren (BruteForce Cracking). Je länger ein Schlüssel ist, desto aufwendiger wird das Durchprobieren.

Die folgende Tabelle zeigt eine kleine Hochrechnung, wie lange das Durchprobieren der jeweiligen Schlüssellänge in Anspruch nehmen würde, in der Annahme, dass jede Schlüssellänge die selbe Rechenzeit beansprucht. Als Grundlage diente die durchschnittliche Leistung eines Teilnehmers vom RC5-64-Wettbewerb[1] am Stichtag 25.03.01.

| Schlüssel | 56bit | 64bit | 128bit | 512bit |
|---|---|---|---|---|
| Anzahl Bit | 56 | 64 | 128 | 512 |
| Anz. Möglichkeiten | 7.20576E+16 | 1.84467E+19 | 3.40282E+38 | 1.3408E+154 |
| Dauer in Jahren | 538 | 137878 | 2.5E+24 | 1E+140 |

**Tabelle 1: Hochrechnung für BruteForce-Cracking**

Um die Rechenzeit zu reduzieren, müssen entweder schnellere PCs verwendet oder die Aufgabe auf mehrere Rechner verteilt werden. Der Begriff "Distributed Computing and Cracking" umfasst dabei sowohl die Software- als auch die Hardware-Architektur um die Aufgabe auf mehreren Rechnern ausführen zu lassen.

Im Folgenden wird beschrieben, wie "Distributed Computing and Cracking" funktioniert und welche Aufgaben davon profitieren können.

[1] RC5-64 - Projektstatus: http://stats.distributed.net/rc5-64/

## 3.2 Grundvoraussetzungen für Distributed Computing and Cracking

Sehr wichtig für das Verständnis des verteilten Rechnens ist das Wissen, welche Aufgaben überhaupt aufgeteilt werden können. Dabei muss vor allem auf eine parallele Verarbeitung geachtet werden, also wiederkehrende Operationen, bei denen die einzelnen Ausführungen nicht von Ausgaberesultaten anderer Ausführungen abhängen dürfen. Zusätzlich darf die aufgewendete Zeit für das Verteilen und Managen der Ausführungen nicht grösser sein als mit der Parallelisierung gespart wird.

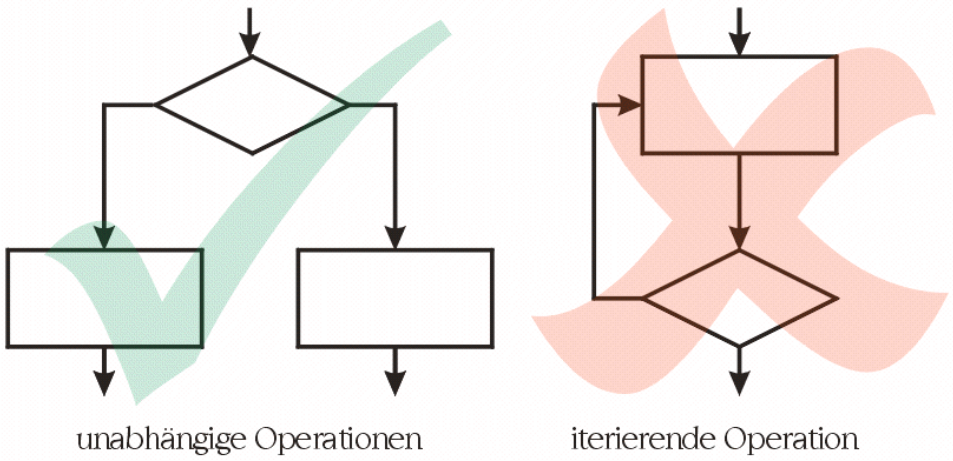


**Abbildung 1: Schematische Darstellung von Operationen**

## 3.3 Verteilung einer Crackaufgabe

Möchte man eine Aufgabe auf mehrere Rechner verteilen, müssen einige Punkte beachtet werden. Als Anschauungs-Beispiel habe ich das BruteForce-Cracking gewählt.

### 3.3.1 Wie sieht ein traditioneller Crackvorgang aus?

Traditionelle Crackprogramme arbeiten auf einem Rechnersystem, evt. mit Multiprozessor-Betriebssystemen unterstützt. Dabei wird der gesamte Schlüsselraum durchprobiert, bis der richtige Schlüssel gefunden wird.

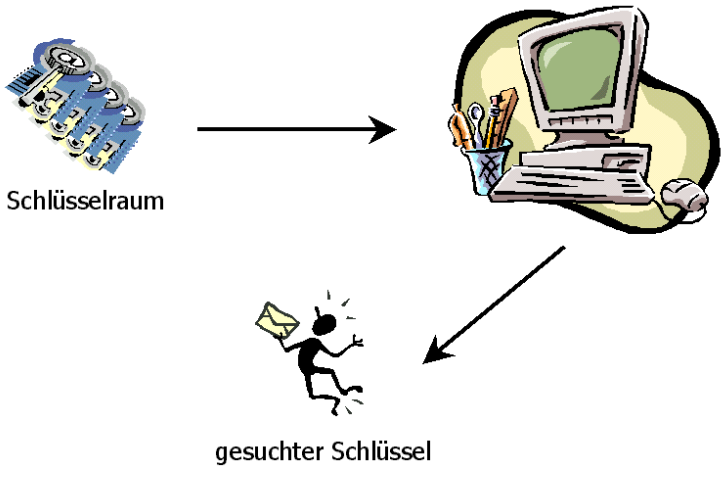


**Abbildung 2: Traditionelles Crackprogramm**

### 3.3.2 Wie kann die Aufgabe auf mehrere Clients aufgeteilt werden?

Da die BruteForce-Methode immer wiederkehrende Operationen verwendet, welche unabhängig von einander ausgeführt werden können, ist eine Arbeitsteilung relativ einfach. Der gesamte Aufgabenraum, d.h. die Anzahl aller möglichen Schlüssel wird in kleine Blöcke aufgeteilt und mit Bearbeitungsstatus, Erfolgsmeldung und Bearbeiter gespeichert.

### 3.3.3 Wie erhalten die Clients ihre Aufgabe?

Dazu muss ein Server entwickelt werden, welcher die Aufgabenblöcke verwaltet und auf Anfrage eine gewisse Anzahl Blöcke versendet. Die entsprechenden Clients müssen sich beim Server anmelden, die angeforderten Blöcke in Empfang nehmen und auf dem Clientsystem abspeichern. Ist nun Rechnerleistung verfügbar, berechnet der Client die Blöcke und gibt dem Server sein Feedback.

### 3.3.4 Wie kann sichergestellt werden, dass alle Blöcke berechnet werden, niemand aber doppelte Arbeit leistet?

Der Server speichert beim Versenden der Blöcke das Datum und die Uhrzeit. Wird nach einer bestimmten Zeit (ca. 2 Wochen) keine Antwort vom entsprechenden Client zurückgegeben, wird der Block wieder freigegeben und dem nächsten Client übermittelt. Auf der Clientseite muss der Benutzer selbst darauf achten, dass er nicht Blöcke auf seinem System speichert und berechnet, welche das Verfallsdatum überschritten haben.

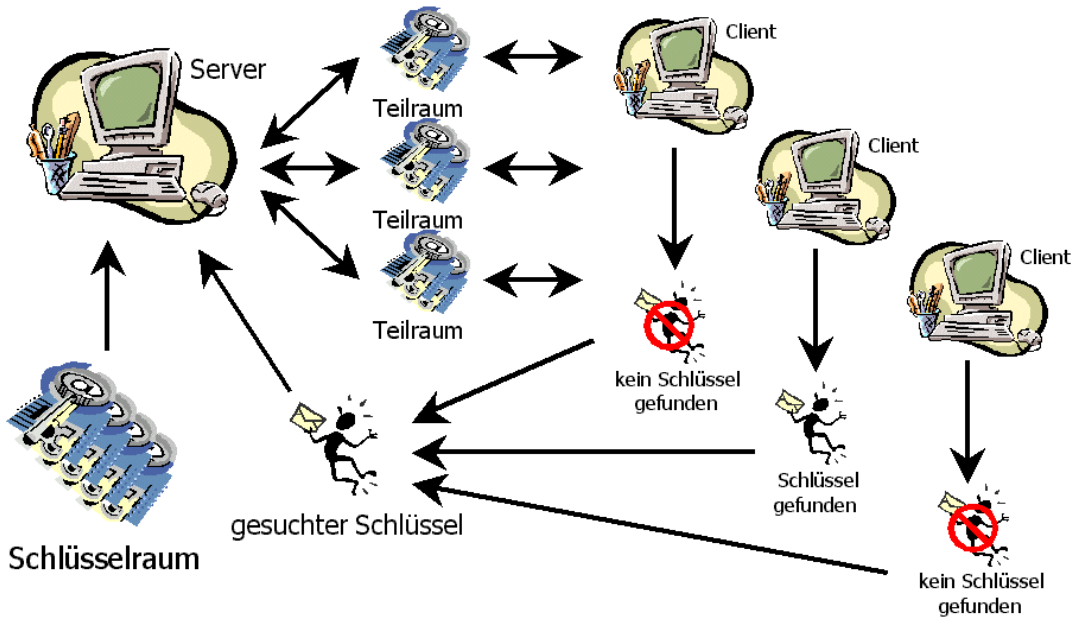


**Abbildung 3: Distributed Computing-Architektur**

### 3.3.5 Wie arbeiten die Clients, braucht es dafür einen speziellen Computer?

Die Clients laufen als Hintergrundprozess und brauchen die gesamte Restprozessorkapazität (jedoch mit niedrigster Priorität) oder arbeiten als Bildschirmschoner. So kann gearbeitet werden, ohne das System merklich zu verlangsamen. Optimale Ergebnisse erhält man natürlich nur, wenn der Client ungestört arbeiten kann.

Je nach Anbieter der Clients und in Abhängigkeit der Aufgabe gibt es solche Clients für fast jedes Betriebssystem. Der im nächsten Kapitel vorgestellte Anbieter distributed.net[2] bietet zum Beispiel Software für folgende Betriebssysteme an:

| | | | |
|---|---|---|---|
| **Acorn RISCOS** | **AmigaOS** | **AIX** | **BeOS** |
| **OpenBSD** | **NetBSD** | **FreeBSD** | **BSD/OS** |
| **DEC UNIX/OSF1** | **PC-DOS/MS-DOS** | **HPUX** | **Linux** |
| **Mac OS X** | **Mac OS** | **Novell NetWare** | **NeXT/OS** |
| **IBM OS/2** | **IBM OS/390** | **QNX** | **SCO Unix** |
| **SGI IRIX** | **Sequent DYNIX** | **SINIX** | **Solaris/SunOS** |
| **DEC VMS** | **Windows 95/98/NT** | **Windows 3.x** | |

**Tabelle 2: von distributed.net unterstützte Betriebssysteme**

[2] distributed.net – Clientdownload: http://www.distributed.net/download/

## 3.4 Beispiel-Anwendungen

Im Folgenden sind einige aktuell laufende Beispiel-Anwendungen und Projekte aufgeführt.

### 3.4.1 Seti@Home[3]

Dieses Projekt beschäftigt sich mit der Suche nach Aliens. Dabei werden mit dem Arecibo Radio Teleskop im Nordwesten von Puerto Rico alle verdächtigen Signale in einem bestimmten Frequenzbereich aufgenommen und diese auf den Clients mit aufwendigen Transformationen auf ausserirdische Nachrichten überprüft. Gestartet wurde das Projekt 1997 und bis jetzt wurde noch kein Aliensignal identifiziert.

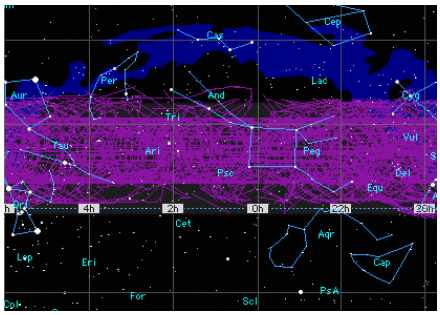

**Abbildung 4: Bereits durch Clients abgesuchtes Gebiet**

### 3.4.2 RC5-64 & ORG24+

Die beiden Projekte RC5-94 und OGR24+ werden gerade vom Team distributed.net bearbeitet. Mehr zu diesen beiden Anwendungen ist im Abschnitt distributed.net zu finden.

### 3.4.3 Weitere Projekte

Es gibt viele weitere Projekte: da sind Versuche, die Börsenkurse[4] der Zukunft zu berechnen, eine Gruppe hat Aids[5] den Kampf angesagt, die Nasa[6] möchte die Marskrater katalogisieren und selbst DDos-Attacken[7], im Seminar von Herrn J. Zehnder behandelt, gehören dazu.


**Abbildung 5: Mars Clickworkers, FightAids@Home und MoneyBee**

[3] Seti@Home Deutsche Übersetzung: http://www.alien.de/seti/index.html
[4] MoneyBee: http://www.moneybee.de/
[5] FightAids@Home: http://www.fightaidsathome.org/
[6] Mars Clickworkers: http://clickworkers.arc.nasa.gov/
[7] DDos - Einige Programme dazu: http://packetstorm.securify.com/index.shtml

# 4 distributed.net "You've lost your key? - We'll find it."

Um nun detaillierter zu werden, beschränkt sich dieser Bericht auf das Team und die Webpage von distributed.net. Diese Gruppe hat sich zum Ziel gesetzt, die Entwicklung und die Förderung des verteilten Rechnens zu unterstützen.

Dazu bieten sie die Möglichkeit mit jeglicher Hardware und Betriebssystemen an den gemeinsamen Projekten teilzunehmen. Diese Projekte sind oft Wettbewerbe, welche bestimmte Firmen ausgeschrieben haben und deren Lösung mit einem finanziellen Gewinn fördern.

Was die Anhänger von distributed.net denken, lässt sich anhand der Slogan-Seite[8] erkennen:

- *There is a place where Linux, Microsoft and Apple work happily together - distributed.net*
- *Feel the Brute Force!*
- *....und wir knacken ihn doch*
- *Faster than NSA!*
- *distributed.net "Lets share the Power."*
- *You've lost your key? - We'll find it.*
- *Gotta' Crack 'em All!*
- *...because cpu time is a terrible thing to waste.*

## 4.1 Projekt "RC5-64"

Das aktuell wichtigste Projekt bei distributed.net ist der von den RSA Labs[9] ausgeschriebene RC5-64-Wettbewerb, der Versuch, einen 64bit-Schlüssel zu knacken. distributed.net geht hierbei nach der BruteForce-Methode vor.

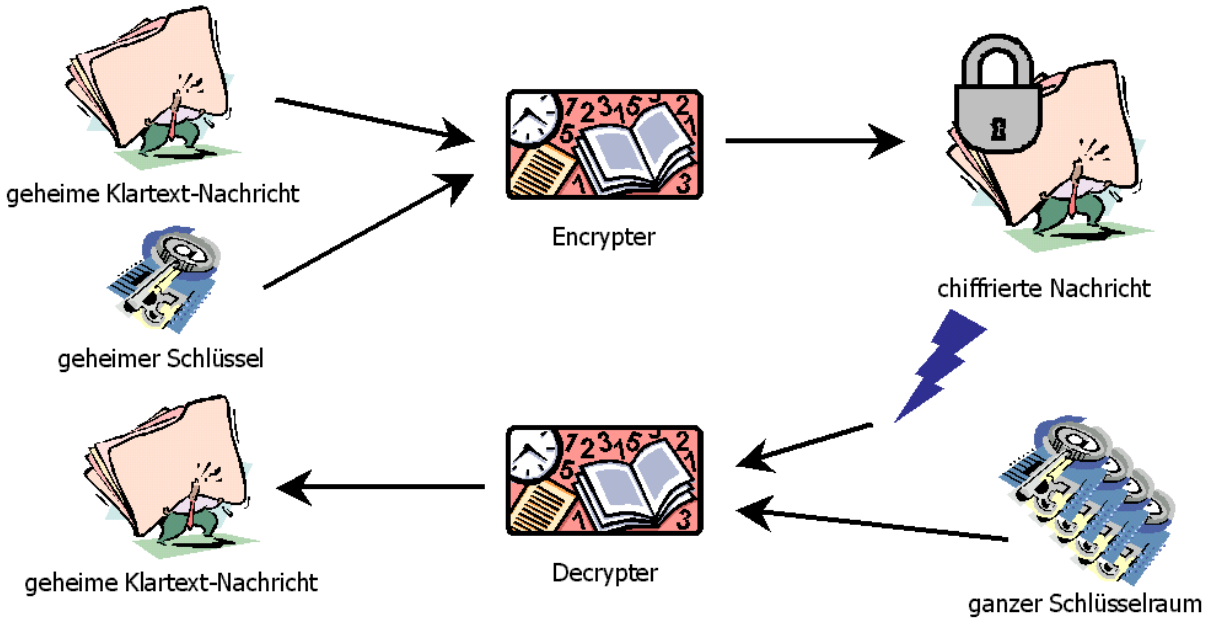


**Abbildung 6: schematische Darstellung einer BruteForce-Attacke**

[8] distributed.net – Slogan-Seite: http://www.distributed.net/slogans/
[9] RSA Data Security Secret Key Challenge von RSA Labs: http://www.rsa.com/rsalabs/97challenge

Der BruteForce-Crack basiert auf der Idee, dass mit dem richtigen Schlüssel die chiffrierte Nachricht wieder in den Klartext entschlüsselt werden kann. Also muss einfach der gesamte gültige Schlüsselraum durchprobiert werden. Da die ersten Worte der Klartextmitteilung bekannt sind ("The unknown message is:"), lässt sich leicht erkennen, wann der richtige Schlüssel gefunden worden ist. Man spricht in diesem Zusammenhang auch von der "Known-plaintext-Attacke". Hätte man keinen Anhaltspunkt auf den Inhalt der Klartextmitteilung, müsste man jeden Output auf Worte oder Silben in der entsprechenden Sprache prüfen und so entscheiden, ob es sich um die gesucht Mitteilung handelt.

Da es sich um einen riesigen Schlüsselraum handelt, ist die Gruppe auf viele Clients angewiesen, welche sich an der Suche nach dem richtigen Schlüssel beteiligen. Als Anreiz für die Teilnahme hat RSA Labs einige Dollar Preisgeld ausgeschrieben. Falls distributed.net gewinnen sollte, wird das Preisgeld wie folgt aufgeteilt:

- $1000 für den Finder
- $1000 für das Team des Finders - diese gehen direkt an den Finder, wenn er nicht in einem Team ist
- $6000 an eine gemeinnützige Gesellschaft, bestimmt durch die Teilnehmer
- $2000 an distributed.net für den Aufbau des Netzwerks und das Bereitstellen der Clients

Im Oktober 1997 hatte distributed.net das Vorgänger-Projekt '56-bit secret-key challenge' mit Hilfe der BruteForce-Methode nach nur 250 Tagen gewonnen.

Am Stichtag, dem 25.03.01, waren 290'272 Teilnehmer bei RC5-64 registriert, wovon 12% aktiv an der Suche beteiligt waren. Damals war das Projekt schon 1250 Tage in Arbeit und dauerte, bei der damaligen Tagesrate (146 GKeys/s), maximal noch 816 Tage um den ganzen Schlüsselraum abzusuchen.

Die folgende Tabelle zeigt eine kleine Hochrechnung, wie lange das Durchprobieren der jeweiligen Schlüssellänge in Anspruch nehmen würde, in der Annahme, dass jede Schlüssellänge die selbe Rechenzeit beansprucht und die Durchschnitts-Tagesrate konstant bleibt. Als Grundlage diente die durchschnittliche Leistung eines Teilnehmers vom RC5-64-Wettbewerb[10] am Stichtag 25.03.01.

| Schlüssel | 56bit | 64bit | 128bit | 512bit |
|---|---|---|---|---|
| Anzahl Bit | 56 | 64 | 128 | 512 |
| Anzahl Möglichkeiten | 7.20576E+16 | 1.84467E+19 | 3.40282E+38 | 1.3408E+154 |
| Dauer für Einzelrechner in Jahren | 538 | 137878 | 2.5E+24 | 1E+140 |
| Dauer bei 34595 Clients in Jahren | 5.7 Tage | 4 | 7.35E+19 | 2.9E+135 |

**Tabelle 3: Hochrechnung für Distributed-BruteForce-Cracking**

[10] RC5-64 - Projektstatus: http://stats.distributed.net/rc5-64/

## 4.2 Projekt "OGR24+"

Ein weiteres Projekt bei distributed.net ist das Suchen von optimalen Golomb-Massstäben[11]. Dabei handelt es sich um ein mathematisches Problem, also ist dieses Projekt für einmal ein nützlicher Dienst für die Wissenschaft.

Ein Golomb-Massstab ist ein Satz von positiven Zahlen, bei denen kein Paar der Zahlen die gleiche Differenz zueinander aufweist.

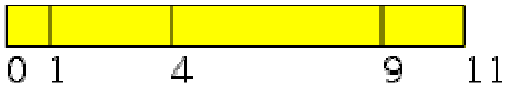


**Abbildung 7: Golomb-Massstab mit Länge 11 und 5 Markierungen**

| Markierung 1 | Markierung 2 | Abstand |
|---|---|---|
| 0 | 1 | 1 |
| 0 | 4 | 4 |
| 0 | 9 | 9 |
| 0 | 11 | 11 |
| 1 | 4 | 3 |
| 1 | 9 | 8 |
| 1 | 11 | 10 |
| 4 | 9 | 5 |
| 4 | 11 | 7 |
| 9 | 11 | 2 |

**Tabelle 4: Abstände im Golomb-Massstab**

Ein optimaler Golomb-Massstab ist der kürzeste Massstab bei einer gegebenen Anzahl von Markierungen. Dabei gibt es oft mehr als einen optimalen Massstab, wobei aber die jeweils gespiegelte Variante weggelassen wird.

Da mit steigender Anzahl der Markierungen der Aufwand für das Finden und Beweisen von OGR exponentiell ansteigt, ist das Problem für Distributed Computing sehr gut geeignet. Der Aufwand für die Berechnungen ist mit dem "Np-vollständig"-Problem vergleichbar, welches im Seminar von Herrn S. Eckert behandelt wird.

Wichtig dabei ist, dass immer der ganze Möglichkeitsraum durchgerechnet werden muss, da immer eine kürzere Variante gefunden werden könnte. Das bedeutet, dass nie eine Meldung "Du hast die optimale Lösung gefunden" auf dem Bildschirm erscheinen wird. Auch gibt es bei diesem Projekt keinen finanziellen Anreiz.

Aktuell wird bei distributed.net an der Suche nach OGRs mit 24 und 25 Markierungen gearbeitet. Am Stichtag, dem 25.03.01, waren 46'078 Teilnehmer für OGR25[12] registriert, wobei ca. 10% aktiv an der Arbeit waren.

[11] OGR24+: http://www.distributed.net/ogr/index.html.de
[12] OGR25 – Projektstatus: http://stats.distributed.net/ogr-25/

# 5 Demo

## 5.1 Very-Simple-Java-Crack

Zur Darstellung der Wirkung des verteilten Rechnens habe ich ein kleines Java-Programm geschrieben. Es wandelt mittels einer mathematischen Transformation eine Geheimzahl in eine verschlüsselte Zahl um. Für den Demoeffekt und um grosse Zahlen zu vermeiden, habe ich eine Warteschlaufe in der Transformation eingefügt, welche das System voll auslastet.

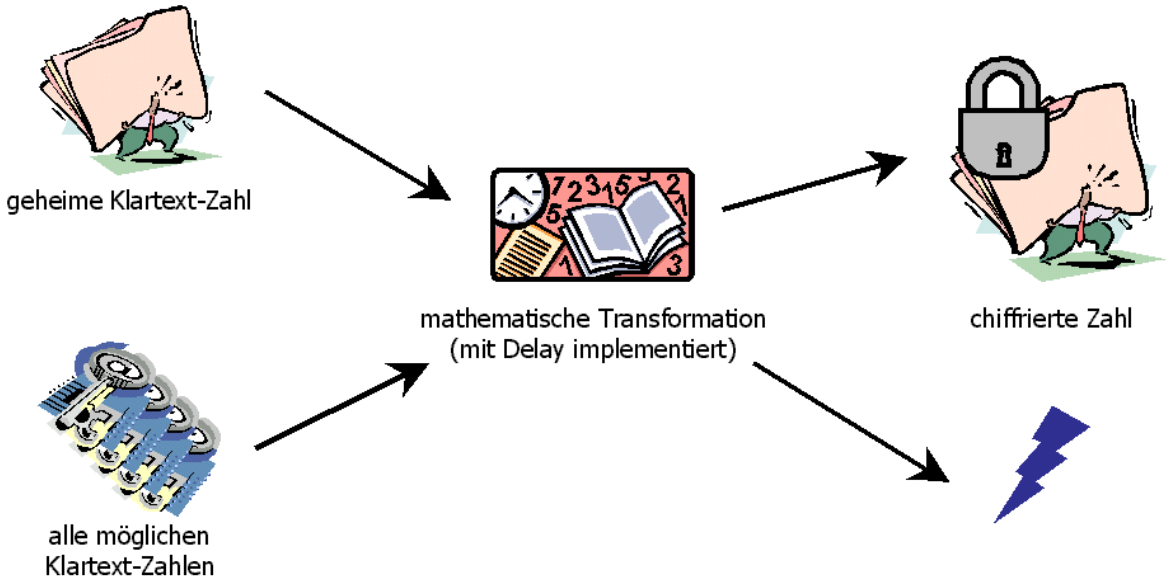


**Abbildung 8: Very-Simple-Java-Crack-Demo**

Mit diesem Demoprogramm möchte ich auf zwei wichtige Punkte des verteilten Rechnens hinweisen:

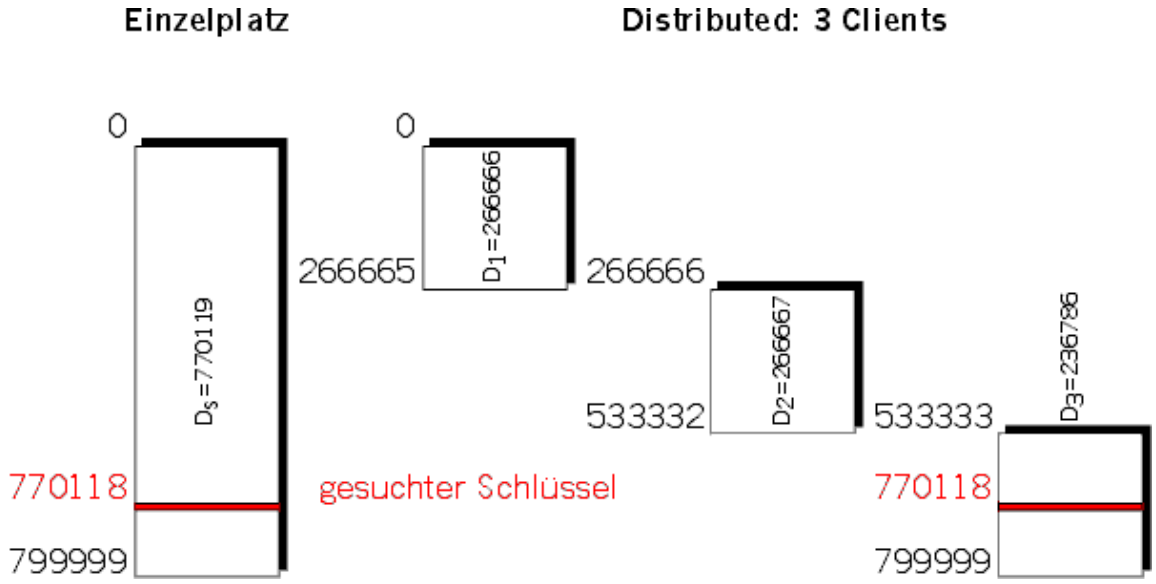


**Abbildung 9: Teilung von Aufgabenraum und Suchdistanz**

- Durchschnittliche Suchdistanz
  Mit dem Wissen, dass der gesuchte Schlüssel am Anfang oder ganz am Ende des Aufgabenraums liegen kann, ist die durchschnittliche Suchdistanz $D_{DS}$ gleich ½*n, wobei n für die Anzahl Schlüssel im Aufgabenraum steht.

- Zeitgewinn durch Aufteilung des Aufgabenraums
  Auf einem Einzelplatz-Rechner muss ein Schlüssel nach dem anderen durchprobiert werden. Je mehr Clients sich an der Suche beteiligen, desto mehr Schlüssel können gleichzeitig abgearbeitet werden und die durchschnittliche verteilte Suchdistanz $D_{DV}$ verringert sich dementsprechend auf ½*n/m, wobei m für die Anzahl parallel arbeitender Rechner steht.

## 5.2 DKBF (Distributed L0phtcrack)

Stöbert man ein Bisschen im Netz herum, findet man auch einen Distributed L0phtCrack mit dem Namen Dkbf. Das Programm arbeitet genau gleich wie der distributed.net-Client (Dkbf verweist im ReadMe-File sogar auf distributed.net), indem er den Schlüsselraum aufteilt und die Teile an die Clients weiterleitet.

Auf der Dkbf-Homepage[13] findet sich ein Formular zum Berechnen der Kosten für einen eigenen Cluster und dessen Leistung. Das Programm setzt ein Linux-Betriebssystem voraus.

Die folgende Tabelle zeigt eine Hochrechnung gemäss Dkbf-Homepage für einen Einzelrechner sowie einen Cluster mit 7 Clientrechnern und einem Server. Zur Vereinfachung wird eine lineare Rechnung mit einer Geschwindigkeitsverbesserung um den Faktor 7 angenommen.

| Bereich | Anz. Schlüssel | 7C1S nach Dkbf in Stunden | Single-PC nach Dkbf in Stunden |
|---|---|---|---|
| A-Z | 8'353'082'582 | 1 | 7 |
| A-Z, 0-9 | 80'603'140'212 | 9 | 63 |
| A-Z, 0-9, 33 Symbole | 7'555'858'447'479 | 35 Tage | 245 Tage |

**Tabelle 5: Zeitaufwand für D-L0phtCrack nach Dkbf (7C1S = 7 Clients à PIII 800 MHz & 1 Server Linux)**

Da die Zahlen auf der Dkbf-Homepage nicht gerade aktuell waren, habe ich eigene Werte gesammelt. Dazu habe ich L0phtCrack in jeder Stufe längere Zeit rechnen lassen und die Restdauer hochgerechnet. Wieder ist die Skalierung auf den 7-Client-Cluster linear vorgenommen worden.

| Bereich | Anz. Schlüssel | 7C1S nach MiA in Stunden | Single-PC nach MiA in Stunden |
|---|---|---|---|
| A-Z | 8'353'082'582 | 0 | 2 |
| A-Z, 0-9 | 80'603'140'212 | 3 | 18 |
| A-Z, 0-9, 33 Symbole | 7'555'858'447'479 | 18 Tage | 125 Tage |

**Tabelle 6: Zeitaufwand für D-L0phtCrack nach MiA (7C1S = 7 Clients à PIII 667 MHz & 1 Server Windows)**

Beide Tabellen zeigen eindrücklich, dass mit 7 Clients jedes Windows-Passwort innert kurzer Zeit geknackt werden kann. Dabei darf man aber nicht vergessen, dass zuerst das entsprechende SAM-File aus dem gewünschten Windows-System verfügbar sein muss.

[13] Dkbf-Homepage: http://dkbf.sourceforge.net/

# 6 Zusammenfassung

Es wurde aufgezeigt, wie einfach eine Aufgabe auf mehrere Rechner verteilt werden kann, wobei je nach Aufgabe diese innerhalb kürzester Zeit erledigt werden kann.

| Schlüssel | 56bit | RC5-64 | NT-Passwort |
|---|---|---|---|
| Dauer bei 34595 Clients | **5.7 Tage** | **4 Jahre** | |
| Dauer bei 7 Clients | | | **max. 18 Tage** |

**Tabelle 7: Zusammenfassung**

Als logische Konsequenz muss aus diesen Werten gefolgert werden, dass Passwörter mindestens einen BruteForce-Aufwand von RC5-64 oder mehr haben sollten, um als sicher eingestuft zu werden. Da sich aber niemand solche lange Passwörter merken kann, sind dringend Vorschläge gefragt.

Als aufmerksamer Leser liegt es nun an Ihnen, ob Sie in Zukunft nur noch Sonderzeichen in Ihren Passwörter verwenden, oder ob Sie sich einen 7fach-Linux-Cluster zulegen: In Zukunft kommen Sie nicht mehr um das verteilte Rechnen und dessen Folgen herum.

# 7 Anhang 1 - Begriffe

| Begriffe | Bedeutung |
|---|---|
| **BruteForce-Attacke** | Das Durchprobieren des gesamten Schlüsselraumes um an die geheime Mitteilung zu gelangen. |
| **Client** | hier: Software, um den eigenen Rechner an Distributed Computing-Projekte anzuschliessen. Sie übernimmt die Paketübermittlung und deren Berechnung. |
| **Cracking** | Einbruchversuch in ein System oder eine Verschlüsselung. |
| **Distributed Computing and Cracking** | Das Konzept, zeitaufwendige Berechnungen auf mehreren Rechnern verteilt durchzuführen. |
| **L0phtCrack** | Crack-Programm, um Windows-NT-Passworte nach der Wort-Listen- und BruteForce-Methode zu finden. |
| **OGR24+** | Optimaler Golomb-Massstab mit 24 und mehr Markierungen |
| **RC5-64** | Stream Cipher Algorithmus mit 64bit-Schlüssel |
| **Wort-Listen-Methode** | Viele bekannte Wörter, wie z.B. Jaguar oder Porsche, sind schon als Passwörter gefunden worden. Diese werden in einer Wortliste gesammelt und als mögliche Eingaben beim Knacken durchprobiert. Dabei wird oft auch etwas mit der Permutation dieser Wörter gespielt. |
| **Cluster** | Gruppe von Rechnern, welche durch einen Server-Prozessor verwaltet und synchronisiert werden. |
| **Known-plaintext Attack** | Der Angreifer kennt oder errät Teile der Nachricht und kann mit Hilfe dieses Wissens Rückschlüsse auf den Schlüssel ziehen. |
| **Chosen-plaintext Attack** | Der Angreifer hat die Möglichkeit, beliebige Texte mit dem (unbekannten) Schlüssel zu kodieren. Er kann sich so Vergleichsmaterial beschaffen und den Schlüssel brechen. |

**Tabelle 8: verwendete Begriffe und ihre Bedeutungen**

# 8 Anhang 2 - Internet-Verweise

## 8.1 Distributed.net - allgemein

- distributed.net – Slogan-Seite
  http://www.distributed.net/slogans/
- distributed.net – Clientdownload
  http://www.distributed.net/download/

## 8.2 Distributed.net - RC5-64

- RC5-64 - Projektstatus
  http://stats.distributed.net/rc5-64/
- RSA Data Security Secret Key Challenge von RSA Labs
  http://www.rsa.com/rsalabs/97challenge

## 8.3 Distributed.net - OGR24+

- OGR24+
  http://www.distributed.net/ogr/index.html.de
- OGR25 – Projektstatus
  http://stats.distributed.net/ogr-25/

## 8.4 Distributed Computing-Projekte

- Seti@Home Deutsche Übersetzung
  http://www.alien.de/seti/index.html
- MoneyBee
  http://www.moneybee.de/
- FightAids@Home
  http://www.fightaidsathome.org/
- Mars Clickworkers
  http://clickworkers.arc.nasa.gov/

## 8.5 Distributed Cracking-Projekte

- Dkbf-Homepage (Distributed L0phtcrack)
  http://dkbf.sourceforge.net/
- DDos - Einige Programme dazu
  http://packetstorm.securify.com/index.shtml