UNIVERSITÄT DUISBURG-ESSEN
FAKULTÄT FÜR INGENIEURWISSENSCHAFTEN
ABTEILUNG INFORMATIK UND ANGEWANDTE KOGNITIONSWISSENSCHAFT

Masterarbeit

Kryptoanalyse der SIGABA

B.Sc. Julian Weyers
Matrikelnummer: 2232616

UNIVERSITÄT DUISBURG ESSEN

Abteilung Informatik und Angewandte Kognitionswissenschaft
Fakultät für Ingenieurwissenschaften
Universität Duisburg-Essen

3. März 2014

Erstprüfer:
Prof. Dr. Arno Wacker
Zweitprüfer:
Prof. Dr.-Ing. Torben Weis
Betreuer:
M.Sc. Nils Kopal

Inhaltsverzeichnis

Abbildungsverzeichnis

Eidesstattliche Erklärung

Hiermit versichere ich an Eides statt, die vorliegende Arbeit selbstständig und unter ausschließlicher Verwendung der angegebenen Literatur und Hilfsmittel erstellt zu haben.

Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht.

Duisburg, 3. März 2014 _____

Unterschrift

1 Einleitung

1.1 Motivation

Rotorverschlüsselungsmaschinen nehmen im Bereich der Kryptografie einen besonderen Stellenwert ein. Trotz ihrer vergleichsweise simplen Funktionsweise besitzen ihre Chiffren einen für den Anfang des 20. Jahrhunderts sehr großen Schlüsselraum. Dieser Schlüsselraum ist, wie in dieser Arbeit noch gezeigt wird, auch von heutigen Personal Computern nicht in dem Maße trivial zu durchsuchen, wie es bei anderen klassischen (hier definiert als „nicht Computer gestützt“) Verschlüsselungstechniken, welche auf Substitution beruhen, der Fall ist. Innerhalb der Rotorverschlüsselungsmaschinen ist die SIGABA als wohl am weitesten entwickelt hervorzuheben. Desweiteren gibt es keine Belege, dass sie während der Zeit ihrer Verwendung jemals gebrochen worden wäre. Ziel dieser Arbeit ist es, aufzuzeigen, wie stark das Verfahren aus heutiger Sicht zu bewerten ist. Die Menge der möglichen Schlüssel muss daher durch logische Überlegungen und effiziente Programmierung der Analysewerkzeuge so stark wie möglich reduziert werden. Am Ende wird eine Approximation gegeben, wie viele Ressourcen zum Brechen einer Nachricht der SIGABA heute notwendig wären.

1.2 Aufgabenstellung

Der Original Wortlaut der vom Lehrstuhl gegebenen Aufgabenstellung ist hier noch einmal abgedruckt: Die „ECM Mark II“ war eine Chiffriermaschine, die von den Vereinigten Staaten von Amerika für die Verschlüsselung ihrer Kommunikation während des zweiten Weltkriegs und weiterhin bis in die 1950er Jahre benutzt wurde. Die Maschine war weiterhin bekannt als „SIGABA“, „Converter M-134” bei der US-Army, oder „CSP-888/889“ bei der US-Navy. Die einer Schreibmaschine ähnelnde Maschine bestand, ähnlich wie die deutsche Enigma, aus mehreren sogenannten Walzen. Innerhalb jeder dieser Walzen befand sich eine Verdrahtung, welche einen Stromfluss über alle Walzen hinweg weiterleitete. Drückte man eine Taste auf dem Tastenfeld, so drehten sich die Rotoren in einer

festgelegten Art und Weise und leiteten den Strom entsprechend unterschiedlich weiter. Im Gegensatz zu der Enigma bestand die SIGABA nicht aus drei, sondern aus insgesamt fünfzehn Walzen. Das aktuell ausgegebene Zeichen wurde auch nicht durch eine Lampe angezeigt, sondern mittels Papierstreifen ausgegeben. Eine Umkehrung des Stroms fand bei der SIGABA ebenfalls nicht statt (keine Umkehrwalze), im Gegensatz zu der Enigma, bei der dies eine ihrer Hauptschwächen war. Durch William Friedman wurde die Sicherheit der Maschine außerdem darüber hinaus erheblich verbessert. Er sorgte unter anderem dafür, dass die Rotoren unregelmäßig weiterschalteten. Aus diesen Gründen ist bis heute kein Angriff bekannt, welcher die SIGABA erfolgreich brechen konnte. Im Jahr 2007 untersuchte Wing On Chan innerhalb seines Masterprojekts [Cha07] „Cryptanalysis of SIGABA“ die Sicherheit der SIGABA.

CrypTool 2.0 – der Nachfolger der bekannten e-Learning Anwendung für Kryptographie und Kryptoanalyse CrypTool – ist ein Open-Source Projekt, welches Lernenden, Lehrenden und Entwicklern mit Interesse an der Kryptographie die Möglichkeiten eröffnet, selbst verschiedene kryptographische und kryptoanalytische Verfahren anzuwenden und auszuprobieren. Die moderne Benutzeroberfläche ermöglicht per einfachem und intuitiven Drag & Drop das Erstellen von einfachen bis hin zu sehr komplexen kryptographischen Algorithmen. Dies geschieht dabei mit einer graphischen Programmiersprache, welche speziell für diesen Zweck entwickelt wurde. Der Benutzer kann dabei ohne besondere Programmierkenntnisse die Algorithmen miteinander verbinden und so eigene neue Algorithmen und Abläufe erschaffen und testen. CrypTool 2.0 basiert auf modernsten Techniken wie z.B. das .net Framework (zurzeit 4.0 SP1) und der Windows Presentation Foundation (WPF). Darüber hinaus ist die Architektur von CrypTool 2.0 vollständig Plug-In-basiert und modular aufgebaut, wodurch die Entwicklung neuer Funktionalitäten stark vereinfacht wird. Im Rahmen dieses Projekts wurden bereits eine Mehrzahl von kryptographischen Algorithmen wie z.B. AES, SHA1 oder die Enigma als Komponenten entwickelt.

Hauptgegenstand dieser Masterarbeit ist die „Kryptoanalyse der SIGABA“ aufbauend auf der Arbeit von Wing On Chan. Hierbei sollen kryptographische Stärken und Schwächen evaluiert und darauf aufbauend eine Sicherheitseinschätzung (u.a. die Komplexität von „Angriffen“) erstellt werden. Praktische Ergebnisse („Angriffe“) sollen innerhalb einer Analysekomponente in CrypTool 2.0 umgesetzt werden. Daneben muss für eine „Simulation der SIGABA“ eine eigenständige Komponente erstellt werden, welche die vollständige Maschine simuliert. Diese Komponente soll das Ver- und Entschlüsseln von Texten ermöglichen. Hierfür muss ebenfalls eine geeignete Visualisierung (siehe Enigma) der Maschine für CrypTool 2.0 entworfen und umgesetzt werden. Nach der Fertigstellung müssen die CrypTool 2.0 Vorlagen sowie die für die Arbeit erstellten Komponenten in das bestehende Versionsverwaltungssystem von CrypTool 2.0 eingepflegt werden.

1.3 Ziele

Die Ziele dieser Arbeit können aus der Aufgabenstellung wie folgt definiert werden:

- Umsetzung einer Komponente zur Ver- und Entschlüsselung von Texten nach dem Verschlüsselungsprinzip der SIGABA
- Entwurf und Umsetzung einer Visualisierung der SIGABA in CrypTool 2.0
- Erstellung der Sicherheitseinschätzung unter Berücksichtigung der Komplexitäten aller vorgestellten Angriffe
- Implementierung eines Angriffs auf die SIGABA als Komponente in CrypTool 2.0

Ein zusätzliches Ziel soll die Lösung der SIGABA Challenge Teil I und Teil II auf MysteryTwister C3 sein.

1.4 Aufbau der Arbeit

Aus den Zielen ergibt sich der Aufbau dieser Arbeit. Dieser gestaltet sich wie folgt:

Das erste nachfolgende Kapitel beschäftigt sich mit den Grundlagen der Kryptografie und der Kryptoanalyse. Diese sind sowohl relevant für die Analyse, wie auch die Implementierung der, in dieser Arbeit erstellten, Komponenten. Es wird ein kurzer Überblick über das Feld der Kryptografie gegeben, um das Verfahren von Schlüssel-Rotor-Maschinen besser einordnen zu können und ein Verständnis für das didaktische Vorgehen in dieser Arbeit zu erzeugen. Desweiteren wird auf Grundlagen zum Verständnis der SIGABA selbst eingegangen, um die Maschine selbst besser analysieren zu können. Abschließend werden noch diverse kryptoanalytische Verfahren vorgestellt, von denen später noch Gebrauch gemacht wird.

Kapitel 3 beschäftigt sich mit der eigentlichen Maschine. Angefangen von ihrer Entstehung und der Verbreitung, über das Design und die Funktionsweise der Maschine, bis hin zur Bedienung. Am Ende des Kapitels wird bereits die Implementierung der SIGABA als Komponente in CrypTool 2.0 vorgestellt, um an dieser Stelle einen logischen Abschluss und eine Trennung zwischen Maschine- und Analyse-Teil der Arbeit zu erhalten.

In Kapitel 4 werden Ansätze zur Kryptoanalyse der SIGABA diskutiert. Exemplarisch wird zunächst ein Angriff ohne die Verwendung von Klartext illustriert, um abschätzen zu können, wie groß der Schlüsselraum der Maschine tatsächlich ist. Danach folgt die Vorstellung des Angriffs mit bekanntem Klartext von Chan aus dem Jahr 2007. Darauf

aufbauend wird im Folgenden ein neuer Angriff entwickelt und vorgestellt. Dabei wird der Begriff des Pseudorotors eingeführt und der Widerspruchsbeweis illustriert. Abschließend wird eine Approximation des Gesamtaufwands des neuen Angriffs gegeben.

Im 5. Kapitel wird gezeigt, wie die Analysekomponenten für CrypTool 2.0 implementiert wurden. Es wurden insgesamt zwei verschiedene Komponenten entwickelt. Die erste Komponente analysiert Texte nach dem Prinzip der erschöpfenden Schlüsselsuche. In der zweiten Komponente wird die Chiffre nach dem, wie in der Kryptoanalyse vorgestellten, neuen Verfahren analysiert. Als letztes wird kurz auf die SIGABA Challenges auf MysteryTwister eingegangen.

In Kapitel 6 werden die Ergebnisse der Kapitel Kryptoanalyse und Implementierung zusammengefasst. Es wird ein Fazit gezogen und ein Ausblick auf zukünftige Arbeiten gegeben.

2 Grundlagen

Für eine Kryptoanalyse der SIGABA, müssen zunächst die Begriffe Kryptografie und Kryptoanalyse geklärt werden.

2.1 Kryptografie

Das Wort Kryptografie (oder auch Kryptographie) setzt sich aus den altgriechischen Worten $\kappa\rho\upsilon\pi\tau\acute{o}\varsigma$ (kryptós), für „verdeckt“ oder „verborgen“ und $\gamma\rho\alpha\varphi\varepsilon\iota\nu$ (gráphein), für „schreiben“, was gemeinsam so viel wie „verdecktes schreiben“ bedeutet, zusammen. Der Begriff bezeichnet heute die Wissenschaft, welche sich mit der Widerstandsfähigkeit verschlüsselter Informationen gegen den Zugriff von Unbefugten beschäftigt.

Demgegenüber steht die Kryptoanalyse. Der Begriff Analyse kommt von dem altgriechischen Wort $\alpha\nu\alpha\lambda\upsilon\sigma\iota\varsigma$ (analysis) und bedeutet „Auflösung“. Als Kryptoanalyse werden sowohl die Anwendung, als auch die Untersuchung von Praktiken zur Reduzierung oder zur Auflösung der Widerstandsfähigkeit von kryptografischen Verfahren bezeichnet, wie auch der wissenschaftliche Nachweis der Widerstandsfähigkeit der Verfahren und deren Bewertung.

Die Kryptografie und die Kryptoanalyse bilden zusammen die Wissenschaft der Kryptologie. [Beu94]

2.2 Verschlüsselung

Eines der wichtigsten Teilgebiete der heutigen Kryptografie ist die Verschlüsselung von Nachrichten. Die Nachricht, also die zu verschlüsselnden Informationen, im Folgenden nur noch „Klartext“ genannt, werden zusammen mit dem „Schlüssel“, einer Kette von Zeichen beliebiger Länge, durch den Verschlüsselungsalgorithmus in die „Chiffre“, die verschlüsselten Informationen überführt. Der Sinn der Verfahrens ist, dass es nur mit

Hilfe des Schlüssels möglich ist, die Chiffre durch diesen Verschlüsselungsalgorithmus wieder in den Klartext zu überführen.

Sei (K, C, S, f_v, f_e) ein Tupel mit

- K = die Menge aller Klartexte
- C = die Menge aller Chiffren
- S = die Menge aller Schlüssel
- $f_v = K \times S \longrightarrow C$ ein Verschlüsselungsalgorithmus
- $f_e = C \times S \longrightarrow K$ ein Entschlüsselungsalgorithmus

Der Klartext $k \in K$ kann mit Hilfe des Schlüssels $s \in S$ in die Chiffre $c \in C$ überführt werden durch die injektive Funktion

$f_v(k, s) = c$

und durch die Umkehrfunktion

$f_e(f_v(k, s), s) = k$

wieder in den Klartext zurück überführt werden.

Es wird hier die Konvention vereinbart, Größen von Schlüsselräumen allgemein in Potenzen der Form 2^x anzugeben.

2.3 Kerckhoffs Prinzip

Als allgemeine Konvention der modernen Kryptografie gilt, dass die Sicherheit eines kryptografischen Verfahrens nur in der Geheimhaltung des Schlüssels gewährleistet werden darf, nicht aber in der Geheimhaltung des kryptografischen Verfahrens selbst. Dieses Prinzip wird Kerckhoffs Prinzip genannt, benannt nach dem Kryptologen und Begründer dieser Maxime Auguste Kerckhoff[Beu94]. Die Argumentation an diesem Prinzip festzuhalten, stützt sich vor allem darauf, dass es eher schwieriger ist, ein unsicheres Verfahren als einen Schlüssel auszutauschen und/oder geheim zuhalten. Desweiteren schafft die Transparenz eines Verfahrens bei dessen Nutzern Vertrauen.

Verfahren die ihre Sicherheit aus der Geheimhaltung ihrer Methodik beziehen, fallen heute in den Bereich der Steganografie, der Wissenschaft vom Maskieren von Informationen in anderen Informationen.

2.4 Substitution und Transposition

Um die SIGABA kryptoanalytisch bewerten zu können, ist es wichtig, sie im Gebiet der Kryptografie richtig einordnen zu können.

Die Verfahren zur Verschlüsselung in der Kryptografie teilen sich in zwei große Bereiche auf, die Transposition und die Substitution.

Verfahren der Transposition verschleiern den Inhalt des Klartextes durch Permutation der Positionen der Zeichen im Klartext.

Ein Beispiel:

- Klartext: AUTO
- AUTO $\longrightarrow$ TAOU (Schlüssel: 3142)
- Chiffre: TAOU

Die Zahlen im Schlüssel stellen in diesem Beispiel die Positionen der Buchstaben im Klartext dar.

Demgegenüber steht die Substitution, welche den Inhalt des Klartextes durch Ersetzen der Zeichen verschleiert.

Ein Beispiel:

- Klartext: AUTO
- AUTO $\longrightarrow$ BVUP
- Chiffre: BVUP

In dem Beispiel werden die Buchstaben des Klartextalphabetes, in diesem Fall des lateinischen, auf ein sogenanntes Substitutionsalphabet abgebildet, hier ebenfalls des lateinischen, wobei der Index der Buchstaben um $i = (i + 1)\ mod\ 26$ verschoben wurde. Diese Art der Substitution wird Caesar-Verschlüsselung genannt.

2.5 Monoalphabetisch und Polyalphabetisch

Es werden zwei Arten von Substitutionen unterschieden: Monoalphabetisch und Polyalphabetisch. Wird von dem Klartextalphabet auf nur ein einziges Substitutionsalphabet abgebildet, wird von einem monoalphabetischen Verfahren gesprochen. Ein Zeichen des Klartextalphabetes bildet dabei auf ein eindeutiges Zeichen des Substitutionsalphabetes ab.

Ein Beispiel:

- Klartextalphabet: ABCDE
- Substitutionsalphabet BCDEA (Caesar +1)
- Klartext: ABBA
- Chiffre: BCCB

Wird hingegen von einem Klartextalphabet auf mehrere Substitutionsalphabete abgebildet, wird von einem polyalphabetischen Verfahren gesprochen. Ein Zeichen des Klartextalphabete kann dabei, in Abhängigkeit vom Schlüssel, auf verschiedene Zeichen mehrerer Substitutionsalphabete abbilden.

Ein Beispiel:

- Klartextalphabet:
  - ABCDE
- Substitutionsalphabete:
  1. BCDEA
  2. CDEAB
  3. DEABC
- Schlüssel: 3213
- Klartext: ABBA
- Chiffre: DDCD

In diesem kleinen Beispiel wird bereits klar, dass polyalphabetische Substitutionen auf Grund ihrer uneindeutigen Abbildungsweise, im Allgemeinen, Angriffen gegenüber widerstandsfähiger sind als monoalphabetische. [Beu94]

2.6 Zufallszahlen

Wie im weiteren Verlauf gezeigt wird, bedient sich die SIGABA eines pseudozufälligen Prinzips. Um dies zu verstehen, muss in diesem Kapitel zunächst auf die Grundlagen der Zufälligkeit eingegangen werden.

Als Zufallszahlen werden die Ergebnisse eines echten Zufallsexperimentes bezeichnet. Es ist eine Folge von Zahlen mit den Eigenschaften, dass die Werte der Zahlen ohne erkennbares Muster statistisch gleich verteilt sind. Kriterien für ein echtes Zufallsexperiment sind, dass es sich um ein physisches Experiment handelt, dessen Parameter nicht vollständig überwacht werden können. In Folge dieser Bedingungen ergibt sich, dass die Vorhersage des nächsten Zeichens der Folge unmöglich und statistisch vollkommen unabhängig vom vorherigen ist. Beispiele für echte Zufallsexperimente sind weißes Rauschen, z.B. an einem Mikrofoneingang.

Neben den Zufallszahlen gibt es die Pseudozufallszahlen. Als Pseudozufallszahlen wird eine Folge von Zahlen bezeichnet, deren Ursprung kein echtes Zufallsexperiment ist. Eine Folge von Pseudozufallszahlen ist, von außen betrachtet, nicht von einer Folge echter Zufallszahlen zu unterscheiden, da die Werte der Zahlen dieselben Eigenschaften bezüglich des nicht erkennbaren Musters und der statistischen Gleichverteilung besitzen.

Pseudozufallszahlen können mit sogenannten Pseudozufallsgeneratoren erzeugt werden. Pseudozufallsgeneratoren sind deterministische Algorithmen, welche für eine Eingabe s, den sogenannten „Seed“, der Länge k, als Ausgabe eine Zeichenfolge der Länge $m >> k$ erzeugen. Es gilt zu beachten, dass es sich beim Seed um echt zufällige Zahlen handelt. Der Pseudozufallsgenerator liefert für einen bestimmten Seed stets dieselbe Ausgabe. Daher besteht die Gefahr, bei entsprechend aufwendiger Analyse der Folge, den Seed zu rekonstruieren. [Wac13]

2.7 One-Time-Pad

Das One-Time-Pad ist eines der ältesten Beispiele für ein perfektes Verschlüsselungssystem.

Das Verfahren gestaltet sich in der Form, dass der in Bit kodierte Klartext der Länge K mit einer (echt) zufälligen Folge von Bits z gleicher Länge, jeweils bitweise durch die Kontravalenz Operation in die Chiffre überführt wird.

Ein Beispiel:

Klartext:	101010101010101010
One-Time-Pad:	110101001001001011
Chiffre:	011111100011100001

Das One-Time-Pad ist nach dem Satz von Shannon Perfekt sicher.

In seinem Buch „Diskrete Mathematik“ schreibt Martin Aigner dazu:

Das System $(\mathcal{T}, \mathcal{C}, \mathcal{K})$ hat perfekte Sicherheit, falls für alle $T \in \mathcal{T}, C \in \mathcal{C}$

$p(T \mid C) = p(T)$

gilt, wobei $p(T \mid C)$ die bedingte Wahrscheinlichkeit ist.
Mit anderen Worten: Die Kenntnis des Kryptosystems C sagt absolut nichts darüber aus, welcher Text T gesendet wurde. Aus

$p(C \mid T) = \frac{p(C \wedge T)}{p(T)} = \frac{p(T|C)p(C)}{p(T)}$

folgt sofort die äquivalente Bedingung $p(C \mid T) = p(C)$ für alle $T \in \mathcal{T}, C \in \mathcal{C}$

[Aig04]

Für das One-Time-Pad bedeutet dies, gegeben ist ein Klartext $K = x_1, x_2, \ldots x_n$ aus $0, 1$ und eine echt zufällige Folge $S = z_1, z_2 \ldots z_n$ aus $0, 1$, wobei jede Zahl unabhängig und mit Wahrscheinlichkeit $\frac{1}{2}$ gezogen wird, S ist der Schlüssel. Die Chiffre C ist definiert durch

$C = c(T, z) = y_1, y_2 \ldots y_n$

mit $y_i = x_i \oplus z_i$ für alle i. Die Dekodierung ist dann $d(y_i) = y_i z_i$.

Alle möglichen 2^n Chiffren C sind gleich wahrscheinlich, also $p(C) = \frac{1}{2^n}$, und ebenso gilt $p(CT) = \frac{1}{2^n}$, da jedes Zeichen y_i in C mit Wahrscheinlichkeit $\frac{1}{2}$ aus x_i erzeugt wird.

2.8 Kryptoanalytische Verfahren

Kryptoanalytische Verfahren lassen sich innerhalb der Kryptoanalyse ihrer Methodik nach klassifizieren. Im Folgenden werden verschiedene Methoden vorgestellt, welche während der Kryptoanalyse der SIGABA zum Zuge kommen werden.

2.8.1 Verfahren mit bekanntem Klartext

Angriffe auf Chiffren, deren Methoden darauf angewiesen sind, dass der Klartext oder Teile davon bekannt sind, werden als Angriffe mit bekanntem Klartext, bzw. dem gebräuchlicheren englischen Begriff „Known-Plain-Text Attack“ bezeichnet. Gemein ist diesen Verfahren, dass der Klartext dazu benutzt wird, den Schlüsselraum einzugrenzen und auf diese Weise den Aufwand zu reduzieren. Wahrscheinlichkeit eines Erfolgs und Aufwand des Verfahrens hängt stark vom Verfahren selbst und von der Menge an benötigtem bekannten Klartext ab. Ziel eines Angriffs mit bekanntem Klartext besteht darin, die restliche Nachricht zu entschlüsseln, sowie den verwendeten Schlüssel zu ermitteln und/oder Aufschluss über die Funktionsweise von Teilen des Verschlüsselungsverfahrens zu gewinnen.[Beu94]

2.8.2 Nur-Chiffre Verfahren

Die Begriffe Nur-Chiffre, oder der gebräuchlichere englische Term „Cipher-Text-Only Attack“ bezeichnen Angriffe, welche die Chiffre ohne zusätzliche Informationen über den Klartext oder den Schlüssel analysieren. Das bekannteste und trivialste Verfahren ist dabei die sogenannte „erschöpfende Schlüsselsuche“, oder auch engl. „Bruteforce Attack“. [Beu94]

2.8.2.1 Erschöpfende Schlüsselsuche

Als Bruteforce oder erschöpfende Schlüsselsuche, werden jene Verfahren bezeichnet, deren Prinzip darauf beruhen, alle Varianten der Grundgesamtheit (z.B. Schlüssel) zu testen. Auf Grund der Tatsache, dass die korrekte Variante ebenfalls ein Teil der Grundgesamtheit ist, ist die Wahrscheinlichkeit des Erfolgs bei einer erschöpfenden Schlüsselsuche genau gleich 1,0. Die erschöpfende Schlüsselsuche stellt dabei aber auch per Definition stets die aufwendigste und ineffizienteste Art aller korrekten Verfahren dar. Durch diesen höchstmöglichen Aufwand, bzw. die sehr hohen Ressourcen die nötig sind, um diesen

bewältigen zu können, erklärt sich auch der Name Bruteforce, engl. für „Rohe Gewalt“. [KR11]

Beispiel: Ein Zahlenschloss mit drei Ringen, von denen jeder von 0-9 durchnummeriert ist, öffnet sich nur in der korrekten Stellung der drei Ringe. Eine erschöpfende Schlüsselsuche würde sich in der Form gestalten, alle Positionen der Ringe durchzuprobieren, bis sich das Schloss öffnet. Der Aufwand dieses Angriffs würde 3^{10} betragen. Der Angriff ist dabei mit einer Wahrscheinlichkeit von 1,0 erfolgreich. Es ist kein Verfahren denkbar, welches den richtigen Schlüssel langsamer findet und dabei keinen Schlüssel zwei mal testet.

2.8.3 Kostenfunktion

Viele kryptoanalytische Verfahren erzeugen nicht nur den einen Klartext, sondern auch teilweise sehr viele andere Texte, die durch ihre Eigenschaften und ohne weitere Informationen, vom Verfahren, nicht als Klartext ausgeschlossen werden können. In einer klassischen erschöpfenden Schlüsselsuche werden zum Beispiel, durch das Verfahren selbst, keine Ergebnis-Texte ausgeschlossen. Ist der Schlüsselraum demnach n groß, ist die Zahl der erzeugten Texte ebenfalls gleich n. Um diese Texte in eine Ordnung zu bringen, werden Kostenfunktionen verwendet.

Als Kostenfunktionen werden in der Kryptoanalyse Verfahren bezeichnet, welche Texte im Hinblick auf ein bestimmtes Kriterium hin analysieren und einen sogenannten Kostenwert für diese ermitteln.

In der Kryptoanalyse helfen diese Werte dabei, die Texte im Hinblick darauf zu bewerten, ob diese beispielsweise einer bestimmten natürlichen Sprache entstammen. Auf diese Weise kann eine Rangfolge der erzeugten Texte erstellt werden, in welcher jener Klartext einen sehr hohen Wert haben wird, der ebenfalls dieser natürlichen Sprache entstammt.

2.8.4 Teile und Herrsche

Als Teile und Herrsche bzw. der gebräuchlichere englische Begriff „Divide and Conquer“ werden Ansätze zur Lösung eines komplexen Problems bezeichnet, welche vorschlagen dieses in mehrere simplere Subprobleme zu unterteilen. Aus den Ergebnissen der Subprobleme kann dann das Ergebnis des Hauptproblems konstruiert werden. Die Motivation eines Teilen und Herrschen Ansatzes ist dabei stets, dass die Summe der Subprobleme

effizienter zu lösen ist, als das eigentliche Problem selbst. Es gibt viele Beispiele, wo dieser Ansatz zum Tragen kommt, zum Beispiel bei Sortierproblemen. [SKQ01]

3 Die SIGABA

Für eine vollständige Kryptoanalyse ist es wichtig, das Prinzip und die Funktionsweise der Verschlüsselung durch die SIGABA in voller Gänze verstanden zu haben. Dazu muss zu aller erst auf das technische Prinzip von Rotorschlüsselmaschinen im Allgemeinen, wie im Speziellen eingegangen werden.

3.1 Rotorverschlüsselungsmaschinen

Anfang des 20. Jahrhunderts entwickelten mehrere Erfinder gleichzeitig und unabhängig voneinander Maschinen, welche nach dem Prinzip der Rotorverschlüsselung arbeiten. Gründe dafür waren zum Einen das Aufkommen der Fernschreibertechnologie. Durch diese Technologie konnten Nachrichten schnell über große Distanzen elektronisch übertragen werden, was vorher mit manuellen Technologien nicht so ohne Weiteres möglich war. Diese Technik war jedoch äußerst anfällig gegenüber dem Mithören von Dritten, was schnell die Begehrlichkeit nach Verschlüsselung zur Geheimhaltung vertraulicher Nachrichten weckte. Da die Übertragung Zeichenweise erfolgte, war eine Verschlüsselung, welche ebenfalls zeichenweise funktionierte nur folgerichtig und logisch.

Eine weitere Erfindung, welche die Entwickler maßgeblich inspirierte, war die der elektronischen Schreibmaschine. Erst das Prinzip der digitalen Kodierung durch Eingabe von Zeichen über die Tastatur machte eine elektronische Verschlüsselung durch die Rotoren praktikabel.

3.2 Rotoren

Der Kern einer jeden Rotorverschlüsselungsmaschine ist der sogenannte „Rotor“. Der physische Rotor besteht aus einer Deck- und einer Bodenplatte, mit jeweils 26 nach außen gerichteten, elektrischen Kontakten, einer respektive für jeden Buchstaben des Alphabets. Im Falle der SIGABA gab es auch kleinere Rotoren mit nur zehn Kontakten.

Jeweils ein Kontakt der Bodenplatte, ist mit einem Kontakt der Deckplatte verbunden. Welcher Kontakt mit welchem verbunden ist, hängt allein von dem Verdrahtungsschema des Rotors ab. Wird von außen eine Spannung an einen Kontakt des Rotoren angelegt, fließt der Strom zu dem verdrahteten Kontakt weiter. Kryptografisch gesehen führt ein Rotor somit eine monoalphabetische Substitution durch. Erst durch das Fortschalten, die Rotation des Rotors, wird eine polyalphabetische Substitution erreicht.

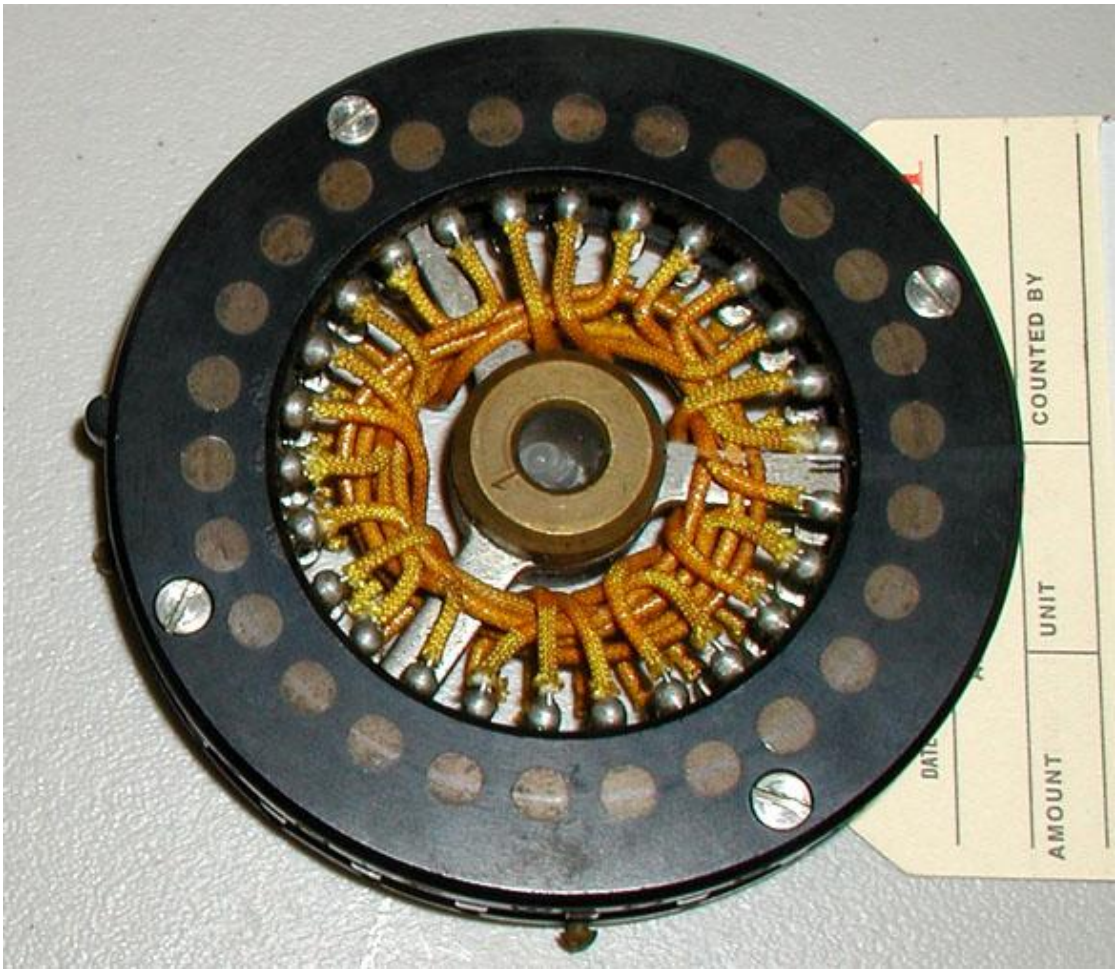

Abbildung 3.1: Darstellung eines orginal Rotors [Pek10]

Beim Fortschalten dreht sich der komplette Rotor mechanisch in seinem Gehäuse genau so, dass der Eingangskontakt, welcher vorher als 'A' interpretiert werden konnte, nun als 'B' interpretiert werden muss.
Da sich die Verdrahtung innerhalb des Rotors mitdreht, ändert sich auf diese Weise auch die Substitution. Mathematisch gesehen wird die Distanz zwischen den Interpretationen der Kontakte verschoben. Entscheidend für die Substitution ist also, neben der Verdrahtung, die „Stellung“ des Rotoren. Auf der Hülle des Rotors ist ein fortlaufendes Alphabet aufgebracht. Jeder Buchstabe bezeichnet dabei eine jener Stellungen. Diese kann durch

ein Sichtfenster in der Maschine abgelesen werden.

Ein Beispiel:

Der Rotor substituiert in Stellung 'A' den Buchstaben 'A' auf 'B', dann beträgt der Abstand zwischen den Kontakten der durch den Draht im Inneren überbrückt wird +1 und den Buchstaben 'B' auf 'D', also +2. Schaltet der Rotor nun fort auf Stellung 'B', substituiert der Rotor nicht mehr 'B' auf 'D', sondern 'B' auf 'C', da sich die Substitution von A nach B verschoben hat, also die +2 Substitution durch die +1 Substitution ersetzt wurde. In der nachfolgenden Grafik 3.2 ist dies noch einmal illustriert.

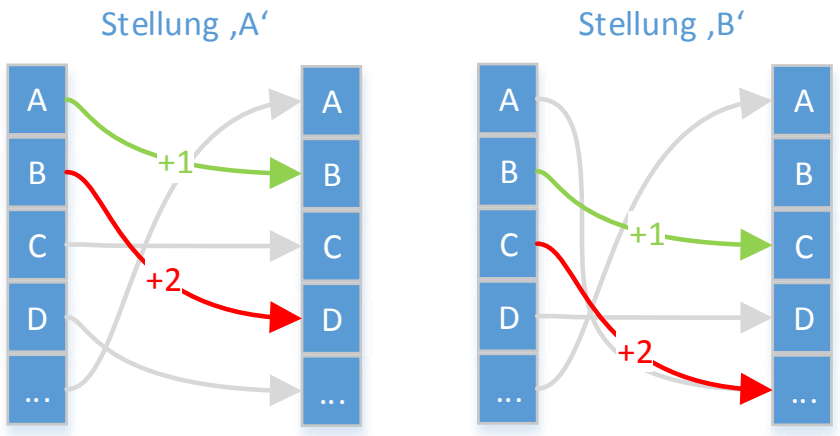

Abbildung 3.2: Schematische Darstellung der Substitution durch den Rotor

Mehrere Rotoren hintereinander geschaltet werden als Rotorbank bezeichnet. Über die Kontakte wird der Stromimpuls an den nächsten Rotor weitergeleitet. Die kryptografische Sicherheit dieses Verfahrens beruht auf der Vielzahl der möglichen Stellungen mehrerer Rotoren einer Bank zueinander. Die Summe aller Stellungen der Rotoren bilden demnach zusammen mit den korrekten Positionen der Rotoren innerhalb der Bank den Schlüssel.

Ein Rotor kann im speziellen Fall der SIGABA auch „falsch" herum, also „auf dem Kopf" oder „rückwärts" eingelegt werden, was zu einer genau entgegengesetzten Substitution durch den Rotor führt. Sprich 'A' substituiert nicht mehr auf 'B', sondern 'B' substituiert auf 'A'. Außerdem schaltet ein falsch herum eingelegter Rotor auch inkrementell, statt dekrementell. Also von der Stellung 'S' nach 'T', wie in dem folgenden Schema 3.3 veranschaulicht.

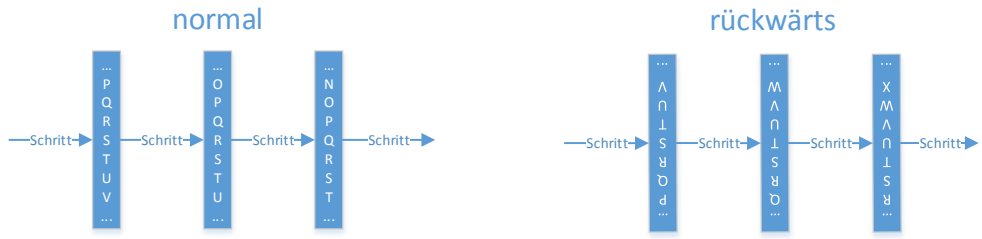

Abbildung 3.3: Schematische Darstellung der Fortschaltungen eines Rotoren

3.3 Die SIGABA

Die SIGABA Rotorverschlüsselungsmaschine war eine von den Amerikanern im zweiten Weltkrieg und noch darüber hinaus bis in die 50er Jahre benutzter Apparat zum Ver- und Entschlüsseln von Nachrichten. Eine Rotorverschlüsselungsmaschine verschlüsselt dabei, ausgehend von einer Starteinstellung, die in diesem Zusammenhang als Schlüssel definiert werden kann, jeden Buchstaben einzeln. Dazu wurde die verschlüsselte Nachricht mittels einer Tastatur, Buchstabe für Buchstabe, wie in einer Art Schreibmaschine, in die Maschine eingegeben. Die Ausgabe erfolgte bei der SIGABA über einen Papierstreifen. Die Maschine führt dabei für jeden Buchstaben eine Substitution durch, welche nach jeder Eingabe eines weiteren Buchstabens permutiert. Es handelt sich daher um ein polyalphabtisches Kryptografieverfahren. [Lee03]

3.4 Geschichte der SIGABA

Der Vorgänger der SIGABA war die „M-134“. Diese Maschine wurde von William Frederick Friedmann 1933 vorgestellt. Friedmann griff die Idee der Rotorverschlüsselung, wie sie Edward Hebern in seinem Patent aus dem Jahre 1921 beschrieb [Heb21]. Er erkannte jedoch die kryptografische Schwäche der deterministischen Fortschaltung der Rotoren. Er schlug in seinem Entwurf hingegen ein zufälliges Fortschalten der Rotoren vor. Dieses Fortschalten sollte bei der M-134 zunächst durch ein Lochband erreicht werden, welches in die Maschine eingespannt wurde und auf dem die Fortschaltungen durch Löcher festgehalten waren. Dieses Lochband erwies sich in der Praxis jedoch als unpraktisch. Es wurde daher in den nachfolgenden Modellen, der „M-134 A“ und „M-134 C“, durch das „M-229“ ersetzt, dem sogenannten „Stepping Maze“, welches auf Friedmanns Mitarbeiter Frank Byron Rowlett zurück geht. Dieses Bauteil sollte die Fortschaltung nach

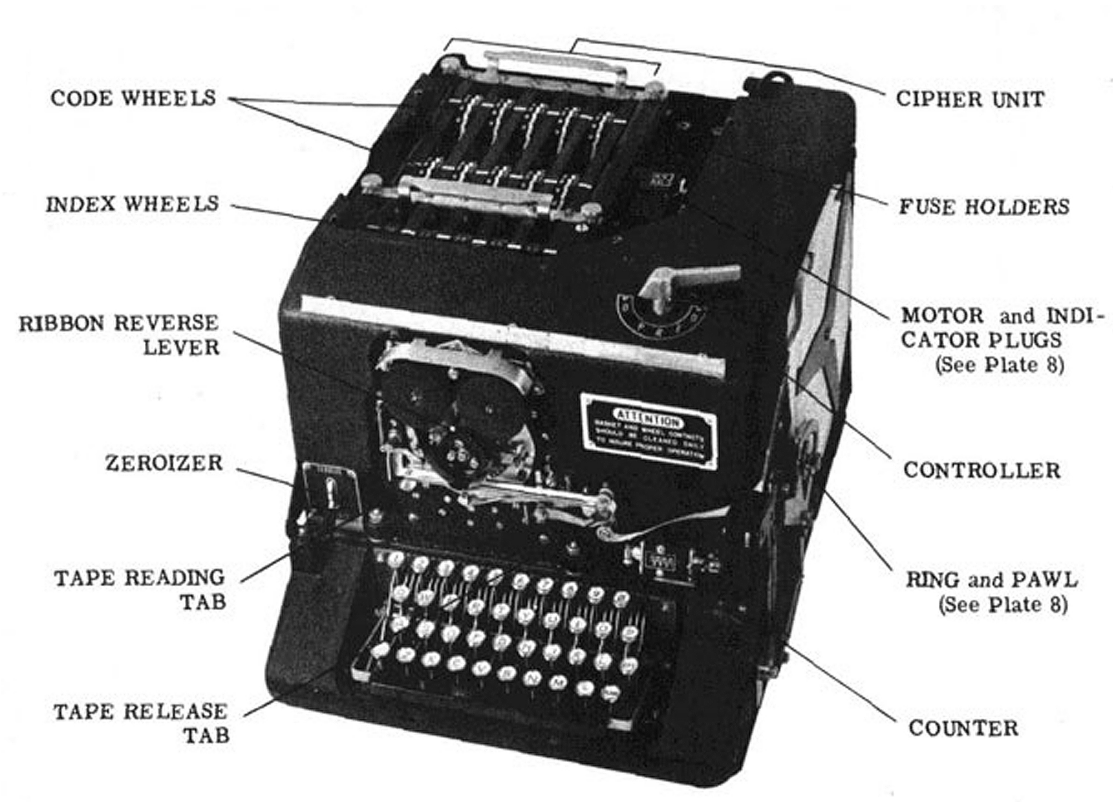

Abbildung 3.4: Darstellung der SIGABA [Pek10]

einem pseudozufälligen Prinzip steuern. Dieses Prinzip sollte durch eine weitere Rotorverschlüsselung erreicht werden, welche die nun wieder deterministische Fortschaltung maskieren sollte. [Goe12]

Auf Grund der erfolgreichen Brechung des Verschlüsselungssystems der japanischen Marine, erkannte der, damals noch US Navy Commander, Laurance F. Safford die Wichtigkeit eines sicheren Verschlüsselungssystems. Er erkannte im Winter 1936-1937 weiterhin das Potential der M-134, und machte sich für deren Einsetzung in der US Navy stark.

Ab diesem Zeitpunkt wurde die M-134 mit dem Stepping Maze von der US Navy unter Verschluss gehalten und unter dem Namen ECM (Electronic Cipher Machine) Mark II weiterentwickelt, vor allem in Bezug auf die Zuverlässigkeit der mechanischen Teile. 1940 wurden die Einzelheiten der ECM Mark II mit der US Army geteilt und ab dem 01.08.1940 von beiden Militärs genutzt. Die San Francisco Maritime National Park Association betont die Wichtigkeit eines gemeinsam genutzten, hoch sicheren Kryptografiesystems. So ist auf ihrer Internetseite zu lesen:

„The use of a common system was of great military value, particularly during the early stages of the war when the distribution of machines and codewheels was incomplete.

By 1943, over 10,000 machines were in use. The „Stepping Maze“ and use of electronic control were a generation ahead of the systems employed by other countries before and after WW II. No other country is known to have ever broken the ECM Mark II cryptographic system.“ [Pek10]

Die SIGABA war bei Außerbetriebnahme in den späten 50er Jahren zwar noch verhältnismäßig sicher, jedoch gab es mittlerweile Maschinen, welche simpler zu bedienen waren. Aus Sicherheitsgründen haben sowohl US Navy, als auch US Army versucht, alle verbleibenden Maschinen, Rotoren und Codebücher zu vernichten. Dies ist auch einer der Gründe, warum es sich heute schwierig gestaltet, die echten Verdrahtungsschemata der Rotoren in Erfahrung zu bringen. [SOC07]

3.5 Schwächen der Enigma

Um zu verstehen, warum die SIGABA im Gegensatz zu der Enigma so schwer zu brechen ist, muss im Folgenden auf die Schwächen der Enigma eingegangen werden, welche Friedmann erkannte und bewusst bei der Entwicklung der SIGABA vermied. Die beiden Hauptschwächen, welche auch zum Fall der ENIGMA führten, waren erstens die Umkehrwalze und zweitens die regelmäßige Fortschaltung der Rotoren.

3.5.1 Die Umkehrwalze

Die Umkehrwalze ist ein Stator (ein Rotor der nicht rotiert oder in einer anderen Stellung eingesetzt werden kann), welcher nur eine Kontaktplatte hat, deren Kontakte mit den eigenen Kontakten verdrahtet sind. Dies führt dazu, dass der Strom wieder an einem anderen Kontakt der selben Platte zurück geführt wird. Dadurch ist es zwar möglich, mit demselben Schlüssel zu entschlüsseln wie zu verschlüsseln. Allerdings ist eine Substitution eines Buchstabens des Klartextes auf den selben in der Chiffre unmöglich. Dies bedeutet im Umkehrschluss eine Einschränkung der möglichen Substitutionen für einen Buchstaben, von 26 auf 25. Dies klingt zunächst nicht viel, führte jedoch im weiteren Verlauf des Krieges zum Fall der Chiffre. [May03]

Auf Grund der Reflektion der Umkehrwalze an einem anderen Kontakt, war es möglich Kollisionsangriffe durchzuführen um die Chiffre zu brechen. Da ein Buchstabe nicht auf einen anderen abbilden kann, können so genannte „Vermutete Klartext Angriffe“ auf die Nachrichten gefahren werden. Dazu benötigt der Angreifer ein relativ langes Wort, das mit einer hohen Wahrscheinlichkeit in der Nachricht vorkommen wird. Im Verlaufe des Angriffs wird das Wort dann mit den Buchstaben der Chiffre zur Kollision gebracht. Das

Wort kann nur an der Stelle in der Chiffre liegen, wo es keine Übereinstimmungen von Positionen der Buchstaben des Wortes, mit denen der verschlüsselten Nachricht gibt. Hat man eine oder mehrere solcher Stellen gefunden, kann angefangen werden, Angriffe mit bekanntem Klartext auf die Nachricht anzuwenden. [Sha49]

3.5.2 Die regelmäßige Fortschaltung der Rotoren

Die Rotoren der ENIGMA schalten bei jedem Tastendruck weiter, wie die Anzeige in einem Hodometer. Der ganz rechte Rotor ist dabei der schnellste, er schaltet bei jedem Tastendruck weiter, der Mittlere schaltet bei einer Umdrehung des Rechten weiter und der Linke Schaltet wiederum erst bei einer vollständigen Umdrehung des Mittleren weiter. Diese Weiterschaltung findet in spezifischen Stellungen, ausgelöst durch sogenannte Übertragskerben des nächst schnelleren Rotors statt. Die ersten Rotoren hatten nur eine Übertragskerbe, was bedeutet, dass nur alle 26 Tastendrücke weitergeschaltet wurde. Diese langsame Fortschaltung des Mittleren und noch langsameren linken Rotoren, birgt eine entscheidende Schwäche. So können der Mittlere, der Linke und die Umkehrwalze als sogenannte Pseudoumkehrwalze vereinfacht werden, da sich deren Substitution nur alle 26 Tastendrücke verändert. Somit müssen nur noch die Verdrahtung der Pseudoumkehrwalze und die Stellung des schnellen Rotoren vom Angreifer herausgefunden werden. Dies ist auch bekannt als „Bâtons Methode“, siehe [Bec05].

3.6 Funktionsweise der SIGABA

Wie im vorherigen Kapitel gesehen war die Schwachstelle der ENIGMA die vorhersagbare Fortschaltung der Rotoren. Bei der Entwicklung der SIGABA wurde hingegen darauf geachtet, die Rotoren möglichst nicht-deterministisch fortschalten zu lassen. Frühe Versionen der SIGABA verwendeten dazu einen Papierstreifen, auf dem die Fortschaltung pseudo-zufällig feststand. Eine Nachricht zu entschlüsseln war nur dann möglich, wenn man das korrekte Lochband mit pseudo-zufalls Fortschaltungen hatte. Wäre jedes Band nur einmal verwendet worden, kann kryptologisch gesehen, das Verfahren auf das „One-Time-Pattern“ zurückgeführt werden und die SIGABA wäre ohne Kenntnis des Lochbandes nicht zu brechen gewesen. Denn selbst wenn, wie in der SIGABA, nur höchstens vier und mindestens ein Rotor nach jeder Eingabe fortschalten, ist die Summe der möglichen Kombinationen von Rotoren, die fortschalten können, mit 30 größer als 26, der abzubildenden Menge von Buchstaben. Vernachlässigen wir potentiell unsichere Stellungen der Rotoren, kann im Großteil der Fälle auf beliebige Nachrichten geschlossen werden.

Die Soldaten im Kampfeinsatz empfanden die Arbeit mit dem Lochband zur Fortschaltung jedoch als unpraktikabel, da das Band immer wieder riss. Die Entwickler reagierten auf diese Kritik und erweiterten die SIGABA durch das Stepping Maze.

Die SIGABA besitzt insgesamt 15 Rotoren, zehn „Cipher“(engl. für Chiffre) bzw. „Control“(engl. für Steuer) Rotoren und fünf „Index“ Rotoren. Logisch kann sie in zwei verschiedene Baugruppen unterteilt werden. Die erste ist die „Cipher Bank“(engl. für Chiffre-Bank). In dieser findet die eigentliche Ent- bzw. Verschlüsselung statt. Sie besteht aus den fünf in Reihe geschalteten Chiffre-Rotoren. Die zweite ist das sogenannte „Stepping Maze“ (engl. frei übersetzt „Fortschaltungs-Labyrinth“) bestehend aus der Steuer-Bank und der Index-Bank. Das Fortschaltungs-Labyrinth steuert lediglich die Fortschaltung der Chiffre-Rotoren. Während Chiffre- und Steuer-Rotoren baugleiche Rotoren mit 26 Kontakten sind und beliebig gegeneinander ausgetauscht werden können, sind die Index-Rotoren kleiner und haben nur zehn Kontakte. [SSD44]

In der Abbildung 3.5 ist die SIGABA als Schaltplan dargestellt. Die mit C1-C5 gekennzeichneten Elemente sind die fünf Chiffre-Rotoren, der Chiffre-Bank. Darunter sind die fünf Steuer-Rotoren dargestellt, beschriftet mit S1-S5. Unter den Steuer-Rotoren ist die Index-Bank zu sehen, mit den kleineren Index-Rotoren, beschriftet durch I1-I5.

3.7 Alphabet-Labyrinth

Die eigentliche Verschlüsselung des Textes findet in der Chiffre-Bank oder auch Alphabet-Labyrinth, im englischen „Alphabet Maze“, der SIGABA statt. Diese Bank besteht aus fünf der großen Chiffre-/Steuer-Rotoren mit 26 Kontakten. Die Eingabe eines Buchstabens über die Tastatur führt dazu, dass an dem entsprechenden Kontakt des ersten Rotor der Bank, eine Spannung angelegt wird. Der Rotor leitet durch seine Verdrahtung diese Spannung an den nächsten Rotor der Bank weiter, je nach Verdrahtung an einem anderen Kontakt. Die Weiterleitung der Spannung ist in der Abbildung 3.5 beispielhaft durch Linien innerhalb der Elemente, welche die Rotoren darstellen sollen, zu erkennen. Der letzte Rotor ist an eine Ausgabeeinheit angeschlossen, welche je nach Kontakt die entsprechende Interpretation des Kontaktes als Buchstaben auf Papier bringt.

Nach der Verschlüsselung eines Buchstabens beginnt die Fortschaltung der Rotoren. Maximal vier und minimal einer der Rotoren schalten dabei in die nächste Stellung. Hintergrund diesen Prinzips ist, sicherzustellen, dass maximal viele Kombinationen von Fortschaltungen möglich sind, während sich die gesamte Substitution der Bank zwingend verändert.

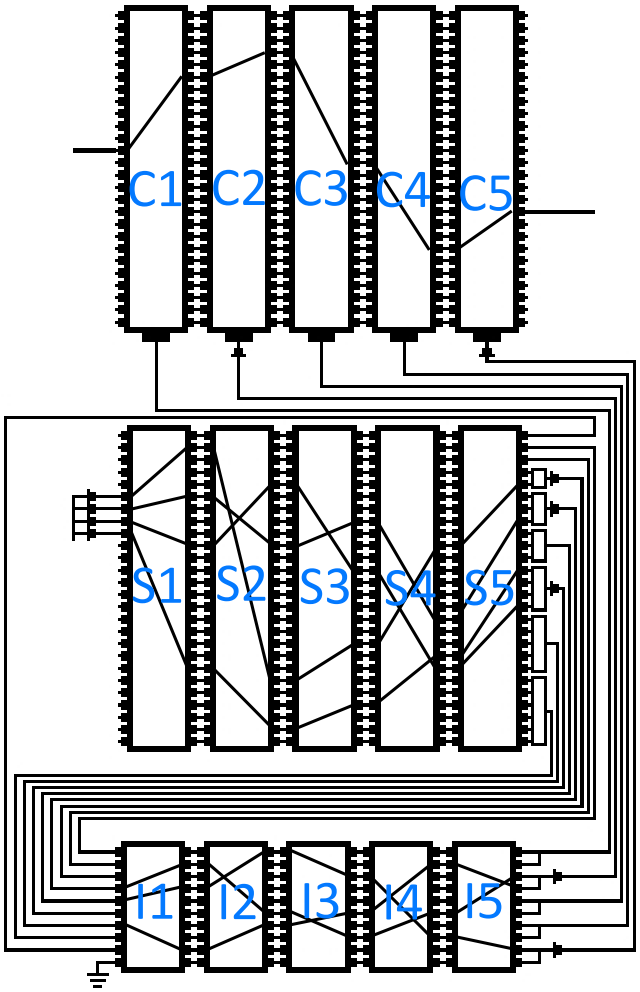

Abbildung 3.5: Darstellung der SIGABA als Schaltplan

Die Dechiffrierung erfolgt durch die Vertauschung von Ausgabe und Eingabe, bei gleich bleibendem Schlüssel und Fortschaltung aller Rotoren.

3.8 Fortschaltungs-Labyrinth

Das „Fortschaltungs-Labyrinth“ kann als eine Art Pseudozufallszahlengenerator verstanden werden, welcher die Fortschaltung der Chiffre-Rotoren steuert.

Das Fortschaltungs-Labyrinth besteht aus den fünf Steuer- und den fünf Index-Rotoren. Die fünf Steuer-Rotoren sind ähnlich als Bank in Reihe geschaltet, wie die Chiffre-Rotoren. Anstatt das jedoch Strom an dem Kontakt des zu verschlüsselnden Buchstaben

angelegt wird, wird stets Strom an den Kontakten 'F', 'G', 'H' und 'I' des ersten Rotoren der Bank gleichzeitig angelegt. Die Substitution der vier Stromimpulse erfolgt genau wie bei der Ent- bzw. Verschlüsselung durch die Chiffre-Bank.

Im Gegensatz zu der Chiffre-Bank, schalten die Steuer-Rotoren ähnlich der Enigma nach dem deterministischen Hodometer-Prinzip weiter. Anders als bei der Enigma, ist der schnellste Rotor der Mittlere, in der Abbildung 3.5 als 'S3' gekennzeichnet. S3 schaltet bei der Verschlüsselung von jedem einzelnen Buchstaben weiter. Eine vollständige Umdrehung dieses Rotors bewirkt die Fortschaltung seines linken Nachbarn 'S2' und eine vollständige Umdrehung dieses wiederum die Fortschaltung 'S4', dem rechten Nachbarn des schnellen Rotors. Fortgeschalten wird dabei stets in der Stellung 'O' des auslösenden Rotors. Die Index-Rotoren schalteten nie weiter.

Die Übertragung der Stromimpulse auf die Index-Bank, welches auch als „Index Maze" bezeichnet wird, erfolgte nach folgendem Schema. Index-Rotoren unterscheiden sich baulich von den Chiffre- und Steuer-Rotoren. Während Chiffre- und Steuer-Rotoren baulich identisch sind und gegeneinander ausgetauscht werden können, besitzen Index-Rotoren lediglich zehn Kontakte. Daher müssen bei der Übertragung die 26 Kontakte der Steuer-Rotoren in zehn Gruppen abgebildet werden. Dies erfolgt in der SIGABA nach dem folgenden Schema 3.6. Es fällt auf, dass der 0 Kontakt nicht verbunden ist und somit niemals angesteuert wird.

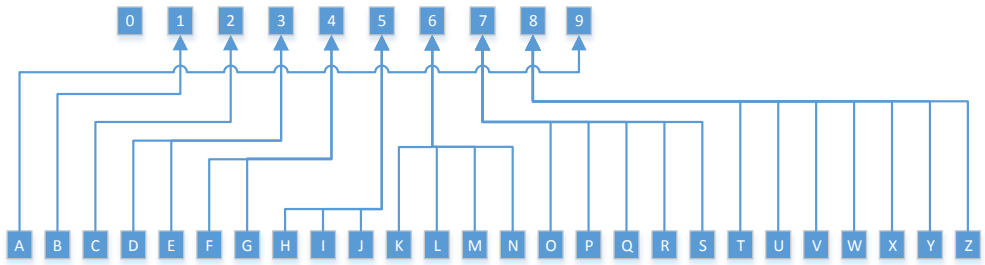

Abbildung 3.6: Schematische Darstellung des Übertrags von 26 Kontakten auf 10

Die verbleibenden Stromimpulse werden auf die Index-Bank übertragen, welche die Signale weiter substituiert und die Fortschaltung der Chiffre-Rotoren mit Hilfe von Magneten und Motoren bewirkt. Da Index-Rotoren zehn Kontakte haben, es jedoch fünf Chiffre-Rotoren gibt, werden die Stromimpulse wiederum zusammengefasst. Dies ist in dem folgenden Schema 3.7 festgehalten.

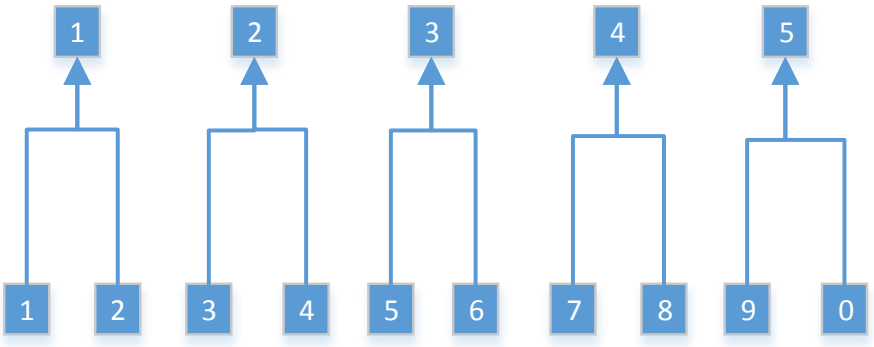

Abbildung 3.7: Schematische Darstellung des Übertrags von zehn Kontakten auf fünf

3.9 Bedienung der SIGABA

Die Bedienung der SIGABA erfolgt in sechs Schritten.[Red44]

3.9.1 Schritt Eins

Als erstes müssen Steuer-, Chiffre- und Index-Rotoren in der korrekten Position in die Maschine eingelegt werden. Die korrekten Positionen mussten aus einer Liste der sogenannten „Tagesschlüssel“ abgelesen werden. Diese Tagesschlüssel wurden vom Verteidigungsministerium ausgegeben und hatten nur für alle Nachrichten eines Tages Gültigkeit. Die Abbildung 3.8 zeigt einen solchen Tagesschlüssel.

Abbildung 3.8: Original Tagesschlüssel der SIGABA

Der Eintrag „Codewheel Arangement“ zeigt die Auswahl der Rotoren und ob diese normal oder rückwärts eingelegt werden sollen, jeweils für das Alphabet- und das Fortschaltungs-Labyrinth.

In dem hier gezeigten Beispiel sollen also ins „Alphabet Maze“, die Chiffre-Bank, von links nach rechts die Rotoren mit der Kennzeichnung 31, 40, 32, 39 und 37 eingelegt werden. Das 'R' hinter der 32 bedeutet, dass der Rotor rückwärts eingelegt werden soll.

Der Eintrag „Indexwheel Setting“ stellt die Positionen der Index-Rotoren dar. Es wird zwischen Nachrichten, die als „SECRET“ oder als „CONFIDENTIAL“ eingestuft wurden, unterschieden.

Da der erste Index 15 lautet, muss in diesem Beispiel, die Stellung des ersten Rotors auf 5 eingestellt werden, im Fall des zweiten Index 24, der zweite Rotor auf 4, usw. Die Buchstabenfolge hinter dem Schlüssel sind Kontrollbuchstaben.

Der letzte Teil des Tagesschlüssels, das „Initial Code Wheel Alignment“, gibt die Startstellung der Chiffre-Rotoren des Alphabet-Labyrinths an.

3.9.2 Schritt Zwei

Danach wird der sogenannte „26-30 letter check“ vom Bediener durchgeführt. Dazu wird die Maschine in den Modus „Encipher“ versetzt und 30 mal hintereinander der Buchstabe 'A' in die Maschine eingegeben. Die letzten fünf Buchstaben (vom 26. - 30.) werden dann mit der Kontrollbuchstabenfolge verglichen. Sind die Folgen identisch, ist die Maschine funktionstüchtig und einsatzbereit.

3.9.3 Schritt Drei

Als nächstes muss sich der Bediener eine fünfstellige Buchstabenkombination ausdenken, den sogenannten „Internal Indicator“. Diese Kombination darf jedoch nicht vollkommen willkürlich sein. Jede Nachricht eines Tages muss einen anderen Message Indicator haben. Außerdem dürfen nicht die Buchstaben „O“ oder „Z“ verwendet werden. Der Internal Indicator stellt die Startstellung der Steuer-Rotoren dar und wird unverschlüsselt mitgesendet.

3.9.4 Schritt Vier

Als erstes werden notwendige Daten vor die Nachricht geschrieben wie Datum und Informationen zur Identifikation. Zu Beginn des Textes wird der „External Indicator“ unver-

schlüsselt geschrieben. Dieser wird aus einer Schlüsselliste gesucht und abgehakt, damit er kein zweites Mal verwendet wird. Dann wird der Internal Indicator geschrieben.

3.9.5 Schritt Fünf

Die Maschine wird dann in den Operationsmodus „Encipher“ gestellt und die Nachricht wird dann in die Maschine eingetippt. Zahlen werden dabei ausgeschrieben.

Jede Art von Interpunktion wird durch ein „X“ repräsentiert. Die einzigen zulässigen Interpunktionen die ausgeschrieben werden dürfen sind „PAREN“, „PARA“, und „QUES“. Diese müssen aber von der Verwendung her auf ein absolutes Minimum gehalten werden.

3.9.6 Schritt Sechs

Der Text wird mit beliebigen Zeichen aufgefüllt, bis die Länge des Textes dem nächst größeren Vielfachen von fünf entspricht. Am Ende werden Internal Indicator und External Indicator zur Sicherheit als Klartext ein weiteres Mal in die Nachricht geschrieben.

Die Entschlüsselung erfolgt auf ähnliche Weise, nur dass in Schritt Sechs die Maschine auf „Decipher“ gestellt werden muss.

3.10 Implementierung

In diesem Abschnitt wird die Implementierung der SIGABA-Komponente beschrieben.

3.10.1 CrypTool 2.0

CrypTool 2.0 (CT2) ist ein Open-Source Projekt unter der Schirmherrschaft von Professor Bernhard Esslinger. Kern des Projektes ist eine komponentenbasierte Lern-Software. Diese bietet eine Benutzeroberfläche auf der verschiedene Komponenten in einer grafischen Programmiersprache in einem Arbeitsbereich arrangiert und ausgeführt werden können. Die Komponenten können dabei über Verbindungen Daten austauschen und an andere Komponenten weiterreichen. [Ess14a]

3.10.1.1 Aufbau von CrypTool 2.0

Die Oberfläche von CrypTool 2.0 kann in vier logische Bereiche eingeteilt werden. Diese Bereiche sind in Abbildung 3.10 zu erkennen.

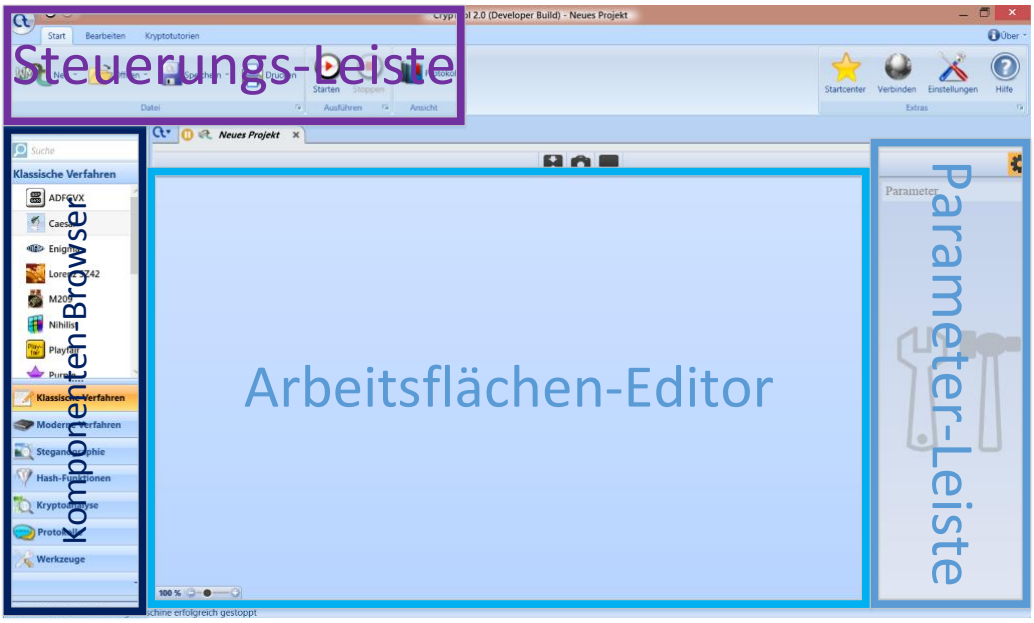

Abbildung 3.9: Darstellung der Oberfläche von CrypTool 2.0

3.10.1.2 Arbeitsflächen-Editor

Den größten Teil der Oberfläche nimmt der Arbeitsflächen-Editor (engl. Workspace Manager) ein. Der Arbeitsflächen-Editor ermöglicht es Arbeitsflächen zu editieren und strukturieren. Die Arbeitsfläche selbst kann als eine Art grafische Programmiersprache gesehen werden. Auf Arbeitsflächen können Komponenten hinzugefügt, entfernt und arrangiert werden. Über die Ein- und Ausgänge der Komponenten können diese miteinander verbunden werden. Der Arbeitsbereich kann mit den Schaltflächen Starten und Stoppen der Steuerungs-Leiste gesteuert werden.

Das Starten eines Arbeitsbereichs führt zur Ausführung aller Komponenten, welche über keinen Eingang verfügen. Nach erfolgreicher Ausführung geben diese dann Ausgangsdaten als Eingangsdaten an die verbundenen Komponenten weiter. Die Ausführung des Arbeitsbereichs ist abgeschlossen, wenn alle Komponenten vollständig ausgeführt wurden.

3.10.1.3 Parameter-Leiste

Rechts neben dem Arbeitsflächen-Editor befindet sich die Parameter-Leiste (engl. Parameter Bar). In der Parameter-Leiste werden alle Einstellungsmöglichkeiten der aktuell markierten Komponente gelistet. Die Einstellungen einer Komponente werden vom Entwickler der Komponente selbst definiert. Auch die Einstellungen integrierter Komponenten werden hier dargestellt.

3.10.1.4 Komponenten-Browser

Am linken Bildrand befindet sich der Komponenten-Browser. Der Komponenten-Browser ermöglicht es, durch die installierten Komponenten der CrypTool 2.0 Distribution zu stöbern und Komponenten für den aktuellen Arbeitsbereich auszuwählen. Die Komponenten sind in verschiedene Gruppen wie „klassische Verfahren“, „moderne Verfahren“ oder „Steganographie“ angeordnet.

3.10.2 Steuerungs-Leiste

In diesem Bereich sind administrative Einstellungsmöglichkeiten des Programms gelistet. Sie sind in die drei verschiedenen Ribbons „Start“, „Bearbeiten“ und „Kryptutorien“ eingeteilt. Insgesamt sechs verschiedene Schaltflächen sind in den zwei verschiedenen Kategorien „Datei“ und „Ausführen“ aufgeführt. In der Kategorie Datei sind vier Schaltflächen zur Verwaltung von Vorlagen und Arbeitsbereichen angeordnet. Die Schaltfläche „Neu“ erzeugt einen neuen Arbeitsbereich. Über die Schaltfläche „Öffnen“ kann eine neue Vorlage aus einer Datei geladen werden und über die Schaltfläche „Speichern“ in die Datei zurück gespeichert werden. Die Schaltfläche „Drucken“ exportiert den Arbeitsbereich als Bild. In der Kategorie „Ausführen“ befinden sich die Schaltflächen „Starten“ und „Stoppen“. Mit diesen Schaltflächen kann der Arbeitsbereich gesteuert werden.

3.10.3 Komponenten

Der Kern von CrypTool 2.0 sind die Komponenten. Komponenten sind elementare Werkzeuge zur Verarbeitung von Daten, innerhalb des Arbeitsbereichs. Zur Datenverarbeitung haben Komponenten sowohl Ein- wie auch Ausgänge, über die die Komponenten Daten empfangen, wie weitergeben können.

Die Oberfläche einer Komponente hat bis zu fünf verschiedene Ansichten, „Präsentation“, „Einstellungen“, „Protokoll“, „Daten“ und „Hilfe“.

3.10.3.1 Präsentation

Die Ansicht Präsentation stellt Raum zur Verfügung, das Verfahren der Komponente transparent zu illustrieren. Präsentationen können dabei so unterschiedlich ausfallen, wie es Verfahren und Anwendungsfälle gibt. Auch direkte Interaktionen sind möglich.

3.10.3.2 Einstellungen

Die Ansicht Einstellungen listet alle Einstellungen, die zur Konfiguration der Komponente möglich sind. Die Anzahl und Art der Einstellungen, hängen stark von der Komponente ab. Die gleichen Einstellungen werden in der Parameter-Leiste der CrypTool 2.0 Oberfläche gelistet, wenn die Komponente im Fokus ist. Auch die Einstellungen integrierter Komponenten werden hier gelistet.

3.10.3.3 Protokoll

In der Ansicht Protokoll werden Meldungen gelistet, die während der Ausführung der Komponente auftreten. Das können Fehlermeldungen, Warnungen oder Meldungen zur Defektlokalisierung sein.

3.10.3.4 Daten

In dieser Ansicht werden die Aus- und Eingangsdaten der Komponente dargestellt.

3.10.3.5 Hilfe

Ein Druck auf diese Schaltfläche öffnet die Seite der Dokumentation für diese Komponente.

3.10.4 Integrierte-Komponenten

Neben der Möglichkeit Komponenten über den Arbeitsbereich mit Datenverbindungen zu verbinden, gibt es in CrypTool 2.0 die Alternative, über die Integrierte-Komponenten-Schnittstelle andere Komponenten nach dem Baukastenprinzip einzubinden. Dieses Vorgehen verhindert Redundanzen bei der Programmierung einer Komponente, da durch die Schnittstelle, direkt auf Methoden der integrierten Komponente zugegriffen werden

kann, sofern dies in der Schnittstelle spezifiziert wurde. Ein weiterer Unterschied zwischen dieser Schnittstelle und der Verbindung durch Datenverbindungen ist, dass die Komponente, welche eine andere integriert, diese auch steuern kann.

Abbildung 3.10: Darstellung der Integrierte-Komponenten-Schnittstelle in CrypTool 2.0

Ist eine Integrierte-Komponenten-Schnittstelle innerhalb der Komponente spezifiziert worden, wird dies im Arbeitsbereich durch ein Blitzsymbol in der Komponente deutlich gemacht. Bei der Erstellung des Arbeitsbereichs muss hier die zu integrierende Komponente ausgewählt werden, wie in Abbildung 3.10 zu sehen.

3.10.5 Verwendete Technologien

Alle CrypTool 2.0 Komponenten in dieser Arbeit wurden in der objektorientierten Programmiersprache C# erstellt. Als Entwicklungsumgebung wurde Microsoft Visual Studio 2010 verwendet. Für Präsentationen und Benutzeroberflächen der Komponenten, wurde zusätzlich das Microsoft Windows Presentation Foundation (WPF) Framework verwendet. [Mic14b] [Mic14a]

3.10.6 Die SIGABA Komponente

Die Implementierung im Rahmen dieser Arbeit wird sich in ihrer Umsetzung in drei verschiedene Komponenten unterteilen. Mit der ersten Komponente ist es möglich, Texte

nach dem Prinzip der SIGABA zu ver- und entschlüsseln. Die zweite und dritte Komponente hingegen sind als Analysewerkzeuge zu verstehen. Um Redundanzen zwischen den Komponenten zu vermeiden, sollen die Analysewerkzeug dabei zum Testen von Schlüsseln, neben der SIGABA-Komponente, auch Gebrauch von weiteren Komponenten von CT2 machen, wie der Kostenfunktions-Komponente.

Als Eingabedaten erhält die SIGABA-Komponente den zu verschlüsselnden Text, als Ausgabe gibt sie wiederum den verschlüsselten Text aus.

Da, wie oben erwähnt, die SIGABA-Komponente später von der Analysekomponente verwendet wird, ist es bei der Implementierung der Komponente wichtig, die Methoden zur Entschlüsselung möglichst effizient zu gestalten, da diese von der Analysekomponente viele Male aufgerufen wird.

3.10.6.1 Implementierung der Methode zum Verschlüsseln von Nachrichten

Nummeriert man die Buchstaben des Alphabets von 0 bis 25 durch, lässt sich die Substitution durch einen Rotor im Quelltext durch den Datentyp „Feld“, engl. „Array“ abbilden. Dabei wird die Substitution des entsprechenden Indexes als Wert gespeichert. Das Problem ist, dass diese Verdrahtung nur in einer einzigen Stellung gültig ist. Für alle anderen Stellungen fallen zwei zusätzliche Rechnungen an, um von der aktuellen Rotorstellung auf die Ausgangsstellung zu schließen und zurück. Dies wird hier vermieden, in dem ein zweidimensionales Array vorgeschlagen wird, in dem alle Substitutionen aller Stellungen des Rotors bereits berechnet wurden. Auf diese Weise sollen Hin- und Rückrechenoperationen der Rotorstellung auf das Substitutionsalphabet in der Grundstellung vermieden werden. Dieses Array wird im folgenden als Verschlüsselungs-Matrix bezeichnet. In Tabelle 3.1 ist eine solche Verschlüsselungs-Matrix eines Index-Rotoren abgebildet. Als Beispiel ist hier eine +1 Substitution des Rotors rot markiert und zu erkennen, wie diese bei jeder Stellung um einen Index nach links verschoben wird.

Die Substitution des Textes durch die Komponente, erfolgt durch Rotor-Objekte. Bei der Initialisierung der SIGABA-Komponente, werden diese Rotor-Objekte erzeugt. Verwaltet werden diese in weiteren Arrays „ChiffreBank“, „SteuerBank“ und „IndexBank“. Die Substitution der Rotoren wird durch die Verschlüsselungs-Matrizen realisiert. Wird dieser Gedanke zu Ende gedacht, ergibt sich, dass verschiedene Matrizen für verschiedene Anwendungsfälle konzipiert werden müssen. Ein Rotor kann normal und rückwärts eingelegt werden. Außerdem braucht der Rotor verschiedene Matrizen, je nachdem ob die Maschine ver- oder entschlüsselt. Insgesamt ergeben sich daraus vier verschiedene Verschlüsselungs-Matrizen der Größe $26 \cdot 26$. Der Inhalt der Matrizen hängt von der Verdrahtung des Rotoren ab. Die Verdrahtungsschemata sind als Konstanten gespeichert.

Stellung /Index	0	1	2	3	4	5	6	7	8	9
0	6	4	9	7	1	3	5	2	8	0
1	3	8	6	0	2	4	1	7	9	5
2	7	5	9	1	3	0	6	8	4	2
3	4	8	0	2	9	5	7	3	1	6
4	7	9	1	8	4	6	2	0	5	3
5	8	0	7	3	5	1	9	4	2	6
6	9	6	2	4	0	8	3	1	5	7
7	5	1	3	9	7	2	0	4	6	8
8	0	2	8	6	1	9	3	5	7	4
9	1	7	5	0	8	2	4	6	3	9

Tabelle 3.1: Schema einer Rotormatrix eines Index-Rotor

In den Rotor-Objekten wird Stellung und Ausrichtung als Variable gespeichert, so dass die *public int Ciph(int input)* Methode sofort die korrekte Substitution diesen Rotors zurückgeben kann.

Die Substitution durch die Chiffre-Bank der Maschine gestaltet sich in Form einer einzigen Anweisung der „Aggregate-Methode“ des Arrays ChiffreBank, wie in Textauszug 3.1 zu sehen ist.

```
1 int temp = c;
2 if (_settings.Action == 0){
3  temp = ChiffreBank.Aggregate(temp, (current, rotor) =>
      rotor.Ciph(current));
4  }
5 if (_settings.Action == 1){
6  temp = ChiffreBank.Reverse().Aggregate(temp, (current,
      rotor) => rotor.DeCiph(current));
7  }
8 return temp;
9 }
```

Textauszug 3.1: Methode zur Substitution durch die Chiffre-Bank

Bei der Entschlüsselung, wird die Aggregate Methode hingegen auf das Inverse Objekt angewendet und die *DeCiph* der Rotor-Objekte aufgerufen, welche die Matrizen zum dechiffrieren verwendet.

Steuer-Bank und Index-Bank können analog dazu in gleicher Weise implementiert werden.

An diesem kleinen Beispiel wird klar, dass der meiste Rechenaufwand der Komponente

durch die Arrayaufrufe anfällt. Aus dieser Überlegung heraus ergibt sich die folgende Verbesserung:

Da die Index-Rotoren I1-I5 sich nicht drehen und C5 sich ebenfalls nicht dreht, können diese Rotor-Objekte für eine Einstellung durch ein einziges Array ersetzt werden. Dadurch werden sechs Arrayaufrufe auf einen einzigen Arrayaufruf reduziert. Zählt man die Substitutionen durch die beiden Statoren zwischen Steuer- und Index-Bank und den Stator zwischen Index-Bank und Chiffre-Bank noch hinzu, welche im ersten Ansatz auch durch Arrays abgebildet wurden, ergeben sich sogar acht Arrayaufrufe. Wird dann noch beachtet, dass im Fortschaltungs-Labyrinth vier Substitutionen gleichzeitig stattfinden, ergeben sich bis zu 32 Arrayaufrufe, welche durch vier ersetzt werden können. Daraus ergibt sich ein neues Array, in Anlehnung an die Kryptoanalyse im nachfolgenden Quelltextauszug 3.2 als das Array: $PseudorRotor[]$ bezeichnet.

Die Substitution des Fortschaltungs-Labyrinths kann in den in 3.2 dargestellten Anweisungen realisiert werden

```
1  public int[] Fortschaltungen()
2  {
3  int tempf = 5;
4  int tempg = 6;
5  int temph = 7;
6  int tempi = 8;
7
8  for (int i = 0; i < 4; i++)
9   {
10   tempf = SteuerRotor[i].DeCiph(tempf);
11   tempg = SteuerRotor[i].DeCiph(tempg);
12   temph = SteuerRotor[i].DeCiph(temph);
13   tempi = SteuerRotor[i].DeCiph(tempi);
14  }
15 return new int[] { PseudoRotor[tempf], PseudoRotor[tempg
     ], PseudoRotor[temph], PseudoRotor[tempi] };
16 }
```

Textauszug 3.2: Methode zur Erzeugung der Fortschaltungen

In diesem Quelltextauszug ist zu sehen, wie das Array mit den Fortschaltungsanweisungen, später in dieser Arbeit im Übrigen auch als Pfad-Punkte bezeichnet, erzeugt werden. Es wird die stets selbe Eingabe 'F','G','H' und 'I' in die Steuer-Bank eingegeben und durch die vier Steuer-Rotoren im $Steuer - Bank[]$ Array substituiert. Zurückgegeben wird dann am Ende die Substitution durch den Pseudorotor.

Die Methode zur Verschlüsselung eines Textes, gestaltet sich wie in 3.3 unter der Verwendung der zuvor vorgestellten Methoden.

```
Klartext k;
for (Länge der Chiffre c)
{
  k[i] = Chiffre(c[i])
  foreach (int i in Fortschaltungen())
  {
    ChiffreRotor[i].Fortschalten();
  }
  if (SteuerRotor[2].Stellung == 14)
  {
     if (SteuerRotor[1].Stellung == 14)
     {
       SteuerRotor[3].Fortschalten();
     }
     SteuerRotor[1].Fortschalten();
   }
   SteuerRotor[2].Fortschalten();
}
return k;
```

Textauszug 3.3: Methode zur Entschlüsselung einer Chiffre

Für alle Buchstaben der Chiffre wird als erstes der neue Buchstabe der Klartextes bestimmt. Danach werden die Chiffre-Rotoren fortgeschaltet, gemäß der errechneten Fortschaltungen des Fortschaltungs-Labyrinths. Zu allerletzt werden die Steuer-Rotoren fortgeschaltet. Wurde jeder Buchstabe der Chiffre substituiert und in den Klartext eingetragen, wird dieser von der Funktion zurückgeben.

3.10.6.2 Einstellungen

Die Einstellungsmöglichkeiten der Komponente gestalten sich in drei Kategorien. In den Haupteinstellungen kann zwischen Ver- und Entschlüsselung gewählt werden, genauso wie die Stellungen der Rotoren, welche bei der SIGABA den Schlüssel darstellt. Bauartbedingt können nur die 26 Buchstaben des lateinischen Alphabets von der Maschine verarbeitet werden. Daher bietet der Menüpunkt „Textoptionen“ die Möglichkeit einzustellen, wie mit weiteren Zeichen verfahren werden soll. Unbekannte Zeichen können dabei ignoriert werden, wobei sie an selbiger Stelle im Ausgabetext wieder eingefügt, entfernt oder durch den Buchstaben ’X’ ersetzt werden. Ebenfalls kann in dieser Rubrik

eingestellt werden, ob alle Buchstaben in Groß- oder Kleinbuchstaben umgewandelt werden sollen. In der letzten Kategorie können die Rotoren gewählt werden, jeweils unterteilt in die drei Rotorbänke. Mittels eines Kontrollkästchens kann außerdem entschieden werden, ob der Rotor „normal“ oder „rückwärts“ eingesetzt werden soll.

3.10.6.3 Präsentation

Aus didaktischen Gründen wird innerhalb dieser Arbeit eine Präsentation entwickelt, die dem Leser und dem späteren CrypTool 2.0-Benutzer der SIGABA, das Prinzip der Verschlüsselung möglichst detailgetreu illustrieren soll. Die Präsentation der Komponente gestaltet sich in Form einer grafischen Visualisierung, welche veranschaulicht, wie die elektrischen Signale hinsichtlich ihrer Interpretation durch die Maschine substituiert werden. In der folgenden Abbildung 3.11 ist die Chiffre-Bank zu sehen. Die Rotoren werden dabei durch die grün hinterlegten Kästen dargestellt. Die Rotoren befinden

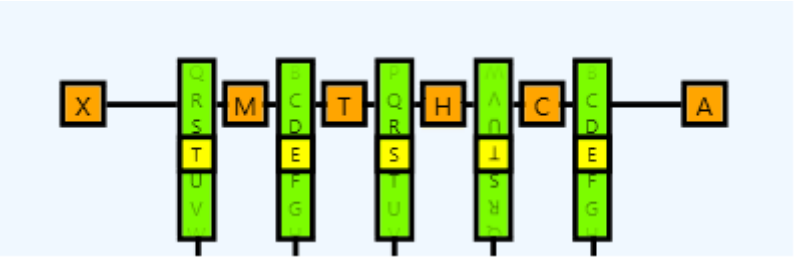

Abbildung 3.11: Darstellung der Chiffre-Rotoren in der Präsentation

sich in der Position „T“ „E“ „S“ „T“ „E“, angedeutet durch die Positionen der fortlaufenden Alphabete und dem Buchstaben in dem gelb hinterlegten Kästchen. Durch die Drehung um 180° des zweiten Rotors von rechts soll angedeutet werden, dass dieser „rückwärts“ eingelegt ist. Die orange hinterlegten Kästen hingegen symbolisieren die substituierten Buchstaben. Initial wird in dem konkreten Beispiel dieser Grafik von rechts der Buchstabe „A“ in die Maschine eingegeben. Der erste Rotor substituiert dieses ’A’ durch ein ’C’ und gibt es an den zweiten Rotoren weiter. Dieser substituiert das ’C’ wiederum durch ein ’H’ usw. Zum Verschlüsseln werden die Buchstaben von der anderen Seite in die Maschine eingegeben und in entgegengesetzter Richtung, sprich von links nach rechts, substituiert. Nach jeder Substitution werden die Positionen der Rotoren durch das Fortschaltungs-Labyrinth verändert. Die nächste Abbildung 3.12 zeigt die Darstellung des Fortschaltungs-Labyrinths in der Präsentation. Die Darstellung der Steuer-Rotoren ist analog zu denen der Chiffre-Rotoren. Die orangenen Kästchen symbolisieren auch in dieser Darstellung die Substitutionen der Signale durch die Rotoren. Da das Fortschaltungs-Labyrinth nicht nur mit einem elektrischen Signal, sondern mit

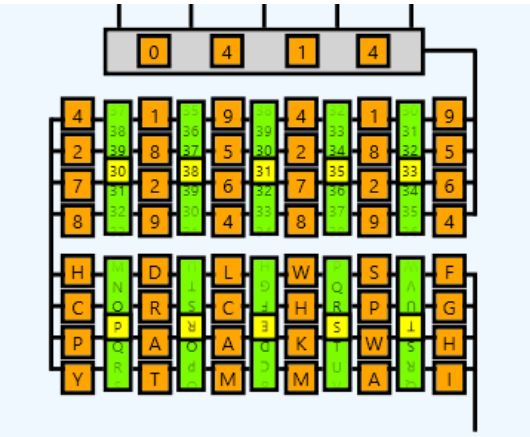

Abbildung 3.12: Darstellung des Fortschaltungs-Labyrinths in der Präsentation

vieren angesteuert wird, wird für jedes einzelne Signal eine Substitution dargestellt. Wie links zu sehen ist, ist der Übertrag auf die Index-Rotoren angedeutet. Die Index-Rotoren selber sind in dem gleichen Stil gehalten wie Chiffre- und Steuer-Rotoren, bis auf die Tatsache, dass sie als Positionsanzeige Zahlen statt Alphabete verwenden, wie es auch bei der original Maschine der Fall gewesen ist. An der rechten Seite der Index-Bank ist die Verbindung zur Steuer-Bank angedeutet. Die orange hinterlegten Kästchen geben an, welche Rotoren in dieser Runde durch das Fortschaltungs-Labyrinth rotiert werden sollen. Zeigen zwei Kästchen den gleichen Wert an, wird trotzdem nur einen Schritt rotiert. Nachdem Fortschalten durch das Fortschaltungs-Labyrinth, werden auch die entsprechenden Steuer-Rotoren nach dem Hodometer-Prinzip weitergeschaltet. Zu weiteren Illustrationszwecken können die Rotoren angeklickt werden, wodurch sich ein Schaubild öffnet, welches die Verdrahtung des angeklickten Rotors verdeutlichen soll, ähnlich der Abbildung 3.13.

Außerdem können Schaubilder der Übertrageinheiten mit einem Tastendruck auf die entsprechenden Schaltflächen aufgerufen werden.

Ist die Präsentation beim Start des Arbeitsfläche geöffnet, erfolgt eine Animation der Verschlüsselung. Dabei wird der zu verschlüsselnde Text in der linken unteren Ecke dargestellt und buchstabenweise in die Maschine eingelesen, dargestellt durch eine blaue Markierung des aktuellen Buchstaben. Im Falle des Beispiels in 3.14 ist der aktuelle Buchstabe das 'E'. Der Buchstabe wird dann in das blaue Eingangsfeld der Chiffre-Bank eingetragen. Die Substitution durch die Chiffre-Rotoren wird durch die orangenen Quadrate zwischen den Rotoren dargestellt, welche nun von links nach rechts die Substitution jedes Chiffre-Rotoren verdeutlichen. Im letzten Schritt wird die letzte Substitution im rechten blauen Quadrat als Ausgabe symbolisiert, im Falle des Beispiels 3.14 ein 'G'. Der Buchstabe wird dann in der Chiffre rechts unten verzeichnet.

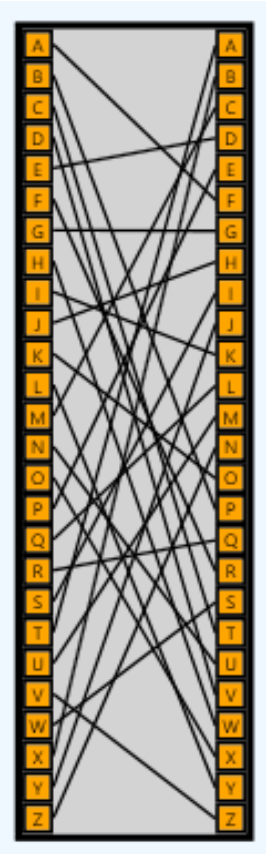

Abbildung 3.13: Schema der Verdrahtung eines Rotoren in der Präsentation

Nach der Verschlüsselung des Textes erfolgt die Fortschaltung der Chiffre-Rotoren. Dazu werden die Buchstaben 'F', 'G', 'H' und 'I' durch das Fortschaltungs-Labyrinth verschlüsselt. Die Darstellung dieser Verschlüsselung verläuft analog zu der der Chiffre-Rotoren. Die Ausgabe des Fortschaltungs-Labyrinths und die Fortschaltung der Chiffre-Rotoren wird durch grüne Rechtecke um die fortschaltenden Chiffre-Rotoren dargestellt.

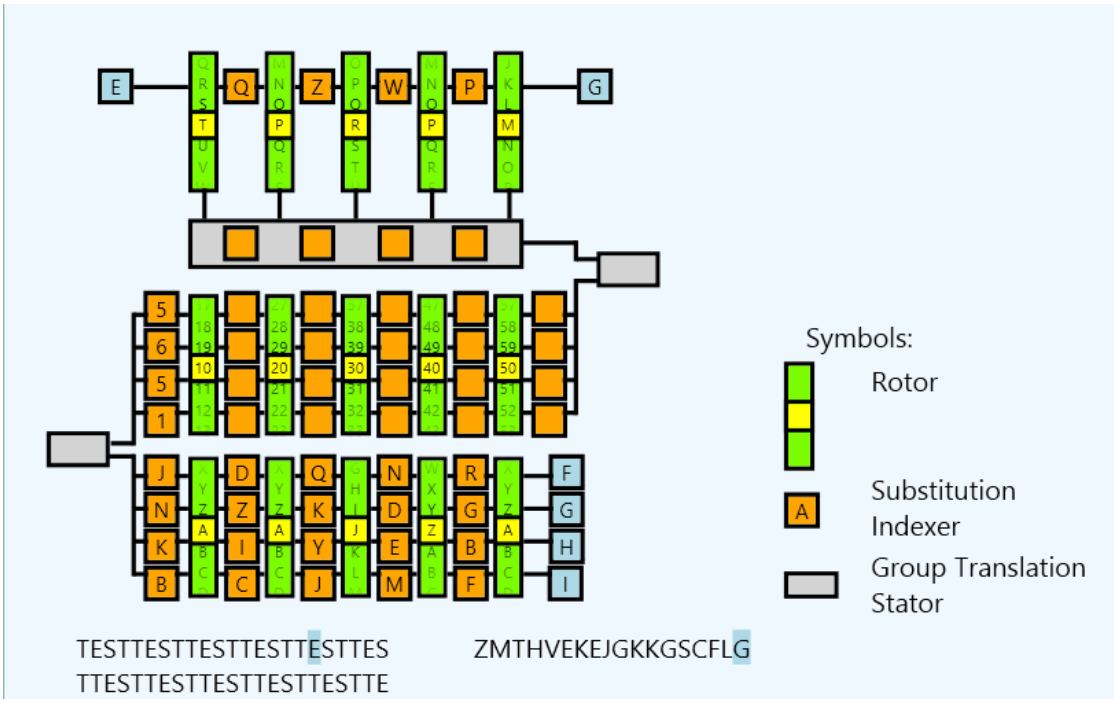

Abbildung 3.14: Ansicht einer Animation

4 Kryptoanalyse

In diesem Kapitel werden die Möglichkeiten der Kryptoanalyse der SIGABA Dechiffriermaschine betrachtet. Zunächst wird die Möglichkeit und Erfolgswahrscheinlichkeit eines Nur-Chiffre-Angriffs diskutiert. Danach wird auf den Angriff mit bekanntem Klartext von Chan/Stamp eingegangen.

4.1 Nur-Chiffre-Angriffe

Auf Grund der pseudozufälligen Fortschaltung der Chiffre-Rotoren, gestalten sich die gängigen Nur-Chiffre Angriffe auf die SIGABA schwierig.

Im Rahmen dieser Arbeit wird zunächst ein Angriff nach dem Verfahren der erschöpfenden Schlüsselsuche vorgestellt. Wie im Grundlagenkapitel bereits erwähnt, sind Angriffe nach dem Verfahren der erschöpfenden Schlüsselsuche immer möglich und haben eine Erfolgswahrscheinlichkeit von 1,0 bei einem maximalen Aufwand. Der Aufwand ist in diesem Fall so groß, wie der Schlüsselraum der SIGABA selbst. Im Folgenden wird nun also der Schlüsselraum ermittelt, woraus sich der Aufwand für diesen Angriff ergibt.

Für diesen Angriff wird

- Die Chiffre
- Verdrahtung der Rotoren

als gegeben vorausgesetzt. Gesucht hingegen ist bei diesem Angriff

- Der Klartext
- Stellung, Ausrichtung und Position aller 15 Rotoren

Um die Verdrahtung eines einzigen Rotors zu ermitteln, müssen alle möglichen Verdrahtungen durchgetestet werden. Die Fakultätsfunktion leistet genau dies. Ein Rotor mit n Kontakten hat dabei

$$1 \cdot 2 \cdot \dots \cdot n = \prod_{k=1}^{n} k = n!$$

Möglichkeiten der Verdrahtung.

Der Schlüsselraum ergibt sich aus der Auswahl der fünf Chiffre-Rotoren, der fünf Steuer-Rotoren und der fünf Index-Rotoren, was in der Theorie ohne nähere Kenntnis der Maschine einen Schlüsselraum von insgesamt

$$(26!)^5 \cdot (26!)^5 \cdot (10!)^5 \approx 2^{993}$$

ergibt.

Kerckhoffs Prinzip besagt, dass die Sicherheit eines kryptografischen Verfahrens allein von der Geheimhaltung des Schlüssels und nicht von der Geheimhaltung des Verfahrens abhängig sein darf. In der Praxis gab es einen Satz von zehn verschiedenen Chiffre-/Steuer-Rotoren und fünf Index-Rotoren. Die Rotoren und ihre Verdrahtung sind Teil der Maschine, demnach ein Teil des Verfahrens und nicht Teil des Schlüssels. Wird Kerckhoffs Prinzip zu Grunde gelegt, hängt die Sicherheit der SIGABA also nicht von der Geheimhaltung der Verdrahtung der Rotoren, sondern lediglich von deren Auswahl, Stellung und Position ab.

Die Kryptoanalyse wird weiterhin unter der Prämisse durchgeführt, dass die Verdrahtungen der Rotoren bekannt sind.

Für die echte Maschine steht ein Satz von zehn Chiffre-/Steuer-Rotoren und insgesamt fünf Index-Rotoren zur Verfügung. Weiterhin können Index-Rotoren nicht rückwärts eingesetzt werden. In diesem Fall ist 10! die Anzahl der Kombinationsmöglichkeiten zehn Chiffre- und Steuer-Rotoren einzusetzen, 2^{10} die Ausrichtung der Rotoren, 26^{10} die Stellungen der Rotoren. Für die Index-Bank ergeben sich Werte von 5! für die Positionen und 10^5 für die Stellungen. Die Ausrichtung kann hier vernachlässigt werden, da Index-Rotoren nicht rückwärts eingesetzt werden können. Unter diesen Einschränkungen ergibt sich ein Schlüsselraum von

$$10! \cdot 2^{10} \cdot 26^{10} \cdot 5! \cdot 10^5 \approx 2^{102}$$

für einen Angriff, bei dem die Verdrahtung aller Rotoren bekannt ist.

Da sich die Rotoren der Index-Bank nicht drehen, können diese als eine feste Verdrahtung von zehn Kontakten auf fünf interpretiert werden. Für eine beliebige Verdrahtung von zehn Kontakten an zehn Kontakte ergeben sich 10! Möglichkeiten. Diese Kontakte

werden nach der Index-Bank noch einmal zu Zweiergruppen zusammengefasst. Durch die Kombinatorik ergibt sich dadurch $2 \cdot 2 \cdot 2 \cdot 2 \cdot 2 = 2^5$ Kombinationsmöglichkeiten der Binärgruppen, auf die alle Verdrahtungen verteilt werden. Daraus ergeben sich

$$\frac{10!}{2^5} = 113.400$$

eindeutige Verdrahtungen. Da

$$\frac{10!}{2^5} = 113.400 < 5! \cdot 10^5 = 12.000.000$$

stellt dies eine Reduzierung des Gesamtschlüsselraums auf

$$10! \cdot 2^{10} \cdot 26^{10} \cdot \frac{10!}{2^5} \approx 2^{96}$$

dar. Werden Anwendungsfehler vernachlässigt und ohne weitere Informationen über den Schlüssel, ergibt sich für eine erschöpfende Schlüsselsuche ein Gesamtaufwand von 2^{96} Schlüsseln, die getestet werden müssten, bei einer Erfolgswahrscheinlichkeit von 1,0.

4.2 Angriff mit bekanntem Klartext von Stamp/Chan (2007)

In seiner Arbeit „CRYPTANALYSIS OF SIGABA“ aus dem Jahre 2007 beschreibt Chan einen Angriff mit bekanntem Klartext auf die SIGABA Dechiffriermaschine. Sein Angriff gestaltet sich in zwei Phasen. Chiffre-Bank und Fortschaltungs-Labyrinth werden in diesen Phasen einzeln angegriffen. Bei diesem Angriff handelt es sich also um einen Teilen und Herrschen Ansatz, da das Gesamtproblem in zwei kleinere Probleme zerlegt wurde. [Cha07]

Für diesen Angriff werden

- Die Chiffre
- Verdrahtung der Rotoren
- Klartext der Länge n und dessen Position innerhalb der Chiffre

als gegeben vorausgesetzt. Gesucht hingegen ist weiterhin

- Der vollständige Klartext
- Stellung, Ausrichtung und Position von allen 15 Rotoren

4.2.1 Phase I

In der ersten Phase werden für jede Auswahl von Rotoren alle möglichen Stellungen, Positionen und Ausrichtungen in der Chiffre-Bank durchgetestet, mit dem Ziel, jene Auswahl von Rotoren und ihre Positionen, Stellungen und Ausrichtungen zu finden, sowie die möglichen Fortschaltungen der Rotoren festzuhalten, für welche die Chiffre-Bank bei Eingabe der Chiffre den Klartext erzeugt. Ein Ergebnis dieser Tests, bestehend aus Startstellung, Position, Ausrichtung und einem „Pfad“ von Fortschaltungen der Chiffre-Rotoren, nennt Chan in diesem Zusammenhang einen „Survivor“, zu Deutsch „Überlebender“. Mit Fortschaltungen sind die Bewegungen der Chiffre-Rotoren gemeint, die sich aus der Ausgabe des Fortschaltungs-Labyrinths ergeben. Im Zuge dieser Arbeit werden die Fortschaltungen als „Punkte“ auf dem Pfad bezeichnet. Die absolute Anzahl aller Punkte eines Pfades ist die Pfadlänge.

Anmerkung: Zur Ermittlung aller Möglichkeiten einer Auswahl k aus einer Grundgesamtheit n wird in dieser Arbeit stets die Funktion des Binominialkoeffizienten verwendet

$$\frac{n!}{k! \cdot (n-k)!} = \binom{n}{k}$$

Chan greift an dieser Stelle eine Besonderheit der SIGABA auf: Da minimal einer und maximal vier Rotoren pro Eingabe rotieren, ergeben sich für jeden weiteren Buchstaben des Klartextes

$$\binom{5}{1} + \binom{5}{2} + \binom{5}{3} + \binom{5}{4} = 5 + 10 + 10 + 5 = 30$$

verschiedene Möglichkeiten, wie die Rotoren der Chiffre-Bank fortschalten können. Da die SIGABA allerdings ein Alphabet von nur 26 Zeichen verwendet, ergeben sich mit

$$\frac{30}{26} \approx 1,15$$

in jedem Schritt ein positives Wachstum der Möglichkeiten zur Fortschaltung. Konkret bedeutet dies, dass wenn n Klartext bekannt ist, sich durchschnittlich $1,15^n$ Survivor für eine Startstellung ergeben, welche in Phase II durchgetestet werden müssen. Die Menge aller Pfade aller Survivor einer Startstellung, Position und Ausrichtung können mit Hilfe eines Entscheidungsbaumes der durchschnittlichen Größe $1,15^n$ modelliert werden. Das bedeutet, je mehr Klartext bekannt ist, desto größer wird der Aufwand diesen Baum zu durchsuchen.

Ein Beispiel:

Das Durchtesten der Chiffre-Bank hat ergeben, dass für eine bestimmte Auswahl, Position und Ausrichtung der Chiffre-Rotoren, genau in der Stellung 'AAAAA' die Chiffre-Bank den ersten Buchstaben der Chiffre in den ersten Buchstaben des Klartextes substituiert. Darauf aufbauend werden nun im zweiten Schritt alle möglichen Fortschaltungen probiert und getestet, ob diese den zweiten Buchstaben der Chiffre in den zweiten Buchstaben des Klartextes überführen. Dies ist in dem Fall genau in den Stellungen 'BBABA' und 'ABABA' gegeben. Alle anderen Stellungen werden verworfen. Daraus ergibt sich der in Abbildung 4.1 dargestellte Baumgraph. Es ist wichtig zu erwähnen, dass die Wurzel des Baumes 'AAAAA' kein Teil der Pfad-Punkte ist. Die Wurzel wird hier stets als „Startstellung" bezeichnet. Dies wird so lange fortgeführt, bis entweder keine weitere Fortschaltung den nächsten Buchstaben der Chiffre in den Klartext überführt, von Chan „Random Case" (zu Deutsch: Zufall) genannt, oder keinen weiteren Klartext Buchstaben zum Testen zur Verfügung stehen, von Chan als „Casual Case" (zu Deutsch: Normalfall) bezeichnet. Im Falle eines Zufalls kann diese Initialstellung als möglicher Schlüssel verworfen werden, im Normalfall hingegen nicht.

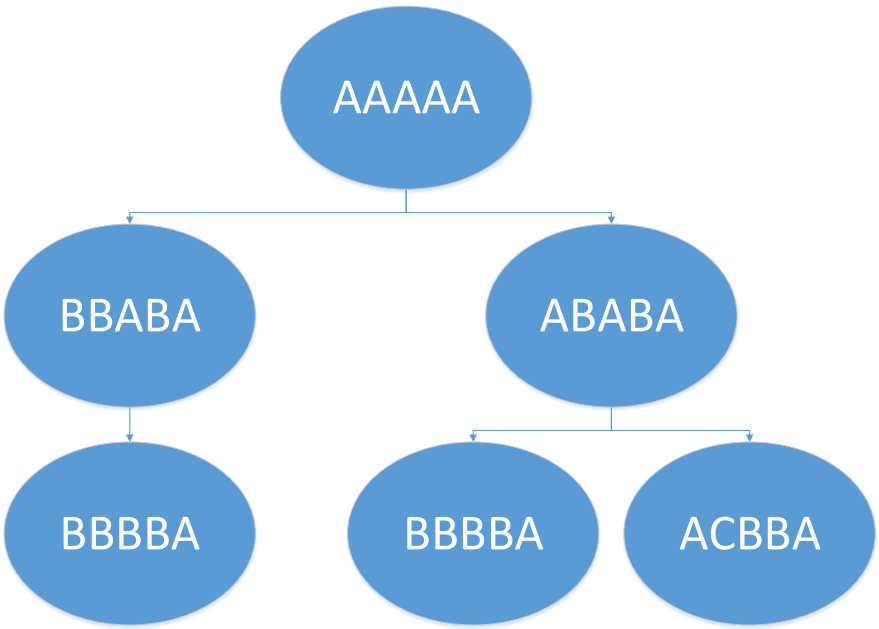

Abbildung 4.1: Darstellung der Fortschaltungen als Baum

Chan schlägt vor, dem Problem des anwachsenden Baumes dadurch zu begegnen, die Survivor vorher entsprechend Abbildung 4.2 zusammenzufassen, um die Anzahl der Pfade überschaubar zu halten. Führt der zusammengefasste Pfad zu einem Punkt, an dem keine weitere Fortschaltung möglich ist, gibt es für keinen der zusammengefassten Pfade einen Survivor.

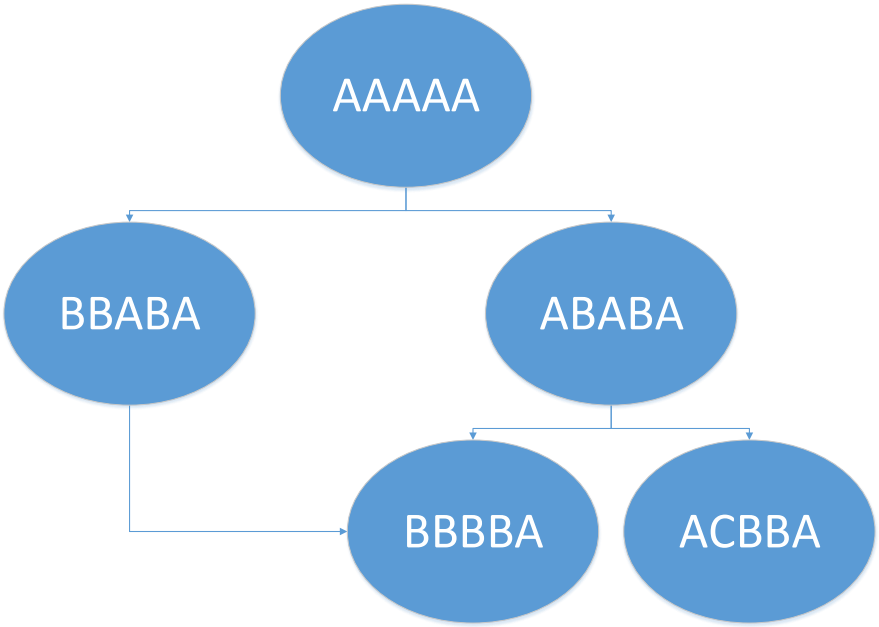

Abbildung 4.2: Darstellung der Zusammenfassung der Pfade

Dies bewirkt, dass die Anzahl der Pfade im Baum weniger schnell wächst, da die Summe der zusammengefassten Pfade kleiner ist, als die aller Pfade. Chan spricht davon, dass von 100.000 Startstellungen und 100 Klartext Buchstaben nur durchschnittlich 203 Normalfälle bei durchschnittlich 100 zusammengefügten Pfaden überhaupt übrig bleiben.

4.2.2 Phase II

In der zweiten Phase wird das Fortschaltungs-Labyrinth für jeden Survivor aus Phase I darauf hingehend überprüft, ob es eine Kombination aus Stellungen, Ausrichtungen und Positionen der verbleibenden Rotoren gibt, welche den Pfad des Survivors erzeugen. Den Aufwand hierfür gibt Chan zunächst mit allen Kombinationen der verbleibenden fünf Rotoren 5!, der Ausrichtung jener 2^5, der Position jener 26^5 und der Index-Bank als feste Verdrahtung von zehn Kontakten an fünf $\frac{10!}{2^5} = 113.400$. Daraus ergibt sich

$$5! \cdot 2^5 \cdot 26^5 \cdot 113.400 \approx 2^{52.2}$$

Auf Grund der unregelmäßigen Verteilung der Kontakte, wie in Abbildung 3.6 abgebildet ist, kann Chan weitere Einschränkungen hinsichtlich der Index-Bank vornehmen. So stellt Chan fest, dass die Chiffre-Rotoren auf Grund der ungleichen Verteilung der Kontakte in die neun Gruppen in unterschiedlichen Geschwindigkeiten fortschalten. Ergibt eine Analyse des Survivors, dass ein Rotor signifikant oft weiterschaltet, ist es wahrscheinlicher, dass dieser mit einer Gruppe von vielen Kontakten verbunden ist.

Buchstaben	Anzahl	Paare
1	3	(0,1) (0,2) (0,9)
2	4	(0,3) (1,2) (1,9) (2,9)
3	5	(0,4) (0,5) (1,3) (2,3) (3,9)
4	7	(0,6) (1,5) (2,5) (5,9) (1,4) (2,4) (4,9)
5	6	(0,7) (1,6) (2,6) (6,9) (3,4) (3,5)
6	6	(0,8) (1,7) (2,7) (7,9) (3,6) (4,5)
7	6	(1,8) (2,8) (8,9) (3,7) (4,6) (5,6)
8	3	(3,8) (4,7) (5,7)
9	3	(4,8) (5,8) (6,7)
10	1	(6,8)
11	1	(7,8)

Tabelle 4.1: Auflistung aller Paare nach Eingängen von klein bis hoch

Da zur Steuerung der Chiffre-Rotoren zwei Kontakte am Ende der Index-Bank zusammengefasst werden, können zwei Gruppen ebenfalls paarweise zusammengefasst werden. In Tabelle 4.1 ist eine Auflistung aller möglichen Paarungen aufgelistet. Die Kombination von fünf dieser Paarungen ergibt einen Pseudorotor der Index-Bank, so lange keine Ziffer doppelt vorkommt und die Gesamtanzahl an Eingängen genau 26 beträgt. In Tabelle 4.2 ist für einige Paare die Fortschaltungsgeschwindigkeit exemplarisch festgehalten. Chan argumentiert nun, dass auf Grundlage der Analyse der Fortschaltungen des Survivors, sich die Paarungen annäherungsweise zurück schließen lassen müssen.

Buchstaben	Beispiel Paare	Schritte	Geschwindigkeit
1	(0,1)	2.300	0,1538462
2	(0,3)	4.324	0,2892308
3	(0,4)	6.095	0,4076923
4	(0,6)	7.635	0,5107023
5	(0,7)	8.965	0,5996656
6	(0,8)	10.105	0,6759197
7	(1,8)	11.074	0,7407358
8	(3,8)	11.890	0,7953177
9	(4,8)	12.570	0,8408027
10	(6,8)	13.130	0,8782609
11	(7,8)	13.585	0,9086957

Tabelle 4.2: Auflistung aller Paare nach Eingängen von klein bis hoch

Chan spricht davon, dass Tests ergeben haben, dass sich durch Ausnutzung dieser statistischen Ungleichverteilung, der Schlüsselraum der Index-Bank im Endeffekt auf 2^8 reduzieren lässt, bei einer Erfolgswahrscheinlichkeit von 0,82. Der Umfang des ganzen Fortschaltungs-Labyrinths reduziert sich damit laut Chan auf:

$2^8 \cdot 5! \cdot 2^5 \cdot 26^5 \approx 2^{43.3}$

4.3 Entwicklung eines eigenen Angriffs

Aufbauend auf den Ergebnissen von Chan/Stamp wird hier im Folgenden ein eigener Angriff entwickelt.

4.3.1 Ansatz zur Verbesserung von Phase II

Der Brite Alfred Dillwyn Knox stellte bei der Analyse der ENIGMA bereits fest, dass für die Verschlüsselung einer Rotorschlüsselmaschinen lediglich jene Rotoren von großer kryptografischer Relevanz sind, welche sich auch drehen. Alle anderen Rotoren und Statoren stellen lediglich monoalphabetische Substitutionen dar. Weiterhin hat Knox bereits gezeigt, dass zwei Rotoren, welche sich nicht oder nur langsam drehen zu sogenannten „Pseudorotoren“ zusammengefasst werden können. [Bec05]

Als Pseudorotoren werden in dieser Arbeit Rotoren bezeichnet, welche in der real existierenden Maschine, in dieser Definition aus Bauart und/oder Verdrahtung nicht existieren. Der Pseudorotor nimmt dabei die gleiche Substitution vor, wie der Rotor oder die Rotoren, für die er eingesetzt wird. Pseudorotoren sind daher Abstraktionen von Baugruppen und helfen, den wahren Schlüsselraum temporär oder total zu abstrahieren. Sie dienen hier lediglich als didaktisches Hilfsmittel zur Veranschaulichung der Idee zur Kryptoanalyse.

Bei genauer Analyse des Fortschaltungs-Labyrinth unter dieser Prämisse kann nun festgestellt werden, dass sich überhaupt nur drei Rotoren drehen, nämlich jene mittleren der Steuer-Bank, in Abbildung 3.5 mit S2, S3 und S4 bezeichnet. Weiterhin kann festgestellt werden, dass sich der Rotor an Position S4 bauartbedingt erst bei jedem 676 Schritt dreht, was dazu führt, dass dieser Rotor in $1 - \frac{1}{676} \approx 99,9\%$ der Fälle als Stator betrachtet werden kann.

Aus diesem Ansatz ergibt sich, dass das Fortschaltungs-Labyrinth zu einer Pseudo-Schlüsselmaschine reduziert werden kann, welche nur noch aus den beiden (schnell) drehenden Rotoren S2 und S3 besteht, Pseudorotor I in der Abbildung 4.3 blau dargestellt und Pseudorotor II in der Abbildung 4.3 rot dargestellt

Pseudorotor I stellt dabei eine beliebige Verdrahtung der beiden Rotoren mit F, G, H und I dar. Pseudorotor II hingegen eine beliebige Verdrahtung der Rotorenausgänge an die Steuerungseingänge der Chiffre-Bank.

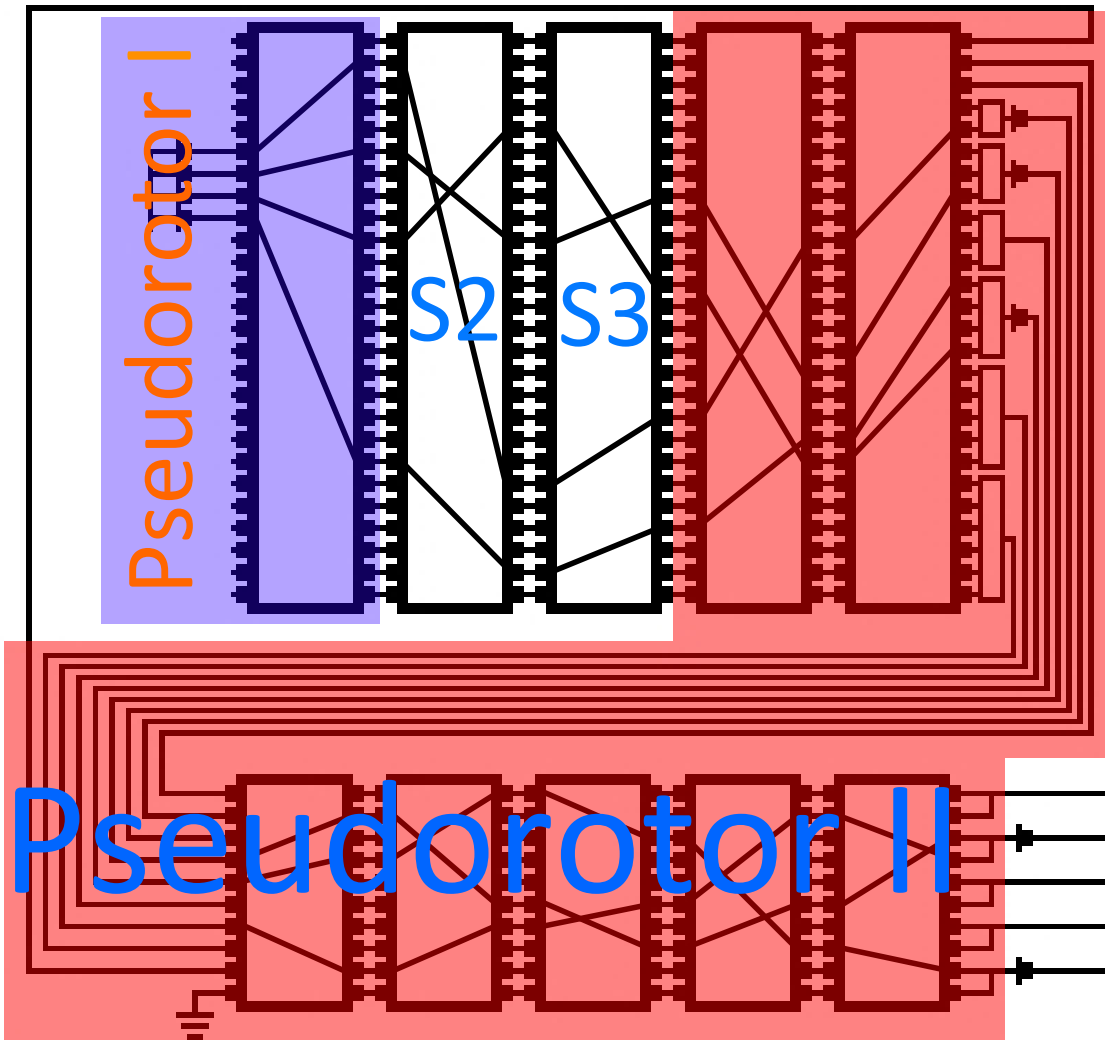

Abbildung 4.3: Darstellung von Pseudorotor I und II

Diese Maschine hätte somit eine Komplexität von

- Pseuorotor I: alle Kombinationen wie F, G, H und I durch den ersten Rotor permutiert werden können $\binom{26}{4} = 14.950$
- Pseudorotor II: $\frac{26!}{2^5} \approx 2^{83}$
- dem Durchtesten der zwei Rotoren $26^2 \cdot 2^2 \cdot 2! \cdot \binom{5}{2} = 54.080$

$14.950 \cdot 2^{83} \cdot 54.080 \approx 2^{113}$

Optimierung: Wenn die ersten drei Rotoren durchgetestet werden, verringert sich der Schlüsselraum auf

- Pseudorotor II: 2^{83}

- dem Durchtesten der ersten drei Rotoren $26^3 \cdot 3^2 \cdot 3! \cdot \binom{5}{3} = 9.491.040$

$2^{83} \cdot 9.491.040 \approx 2^{106}$

Diese Maschine liefert die gleichen Ergebnisse wie das echte Fortschaltungs-Labyrinth. Sie könnte auch beispielsweise für Nur-Chiffre-Angriffe verwendet werden, was allerdings auf Grund des großen Schlüsselraums des Pseudorotors II nicht effizient wäre, da wie gesehen alle Verdrahtungen des Pseudorotoren durchgetestet werden müsste. Da das Fortschalten des langsamen Rotoren von der Fortschaltung des Mittleren unmittelbar abhängig ist, können innerhalb des Pseudorotoren keine Veränderungen auftreten, die nicht von der Maschine selbst ausgelöst wurden. Veränderungen die wiederum von der Maschine ausgelöst werden, sprich das Fortschalten des langsamen Rotors, müssen bei der Analyse berücksichtigt und gegebenenfalls ausgeglichen werden. Dieser Ausgleich gestaltet sich am simpelsten in Form einer Neuinitialisierung des Pseudorotors.

Das Ersetzen der Rotoren mit bekannter Verdrahtung durch den Pseudorotor II mit unbekannter Verdrahtung hat also den Schlüsselraum der Maschine zunächst einmal vergrößert, statt ihn zu verkleinern. Das liegt daran, dass es für den Pseudorotor II zu viele Möglichkeiten der Verdrahtung gibt. Glücklicherweise können allerdings viele dieser Möglichkeiten durch den Pfad des Survivors ausgeschlossen werden. Ein Ansatz diese Möglichkeiten auszuschließen ist der Widerspruchsbeweis.

4.3.1.1 Der Widerspruchsbeweis

Ziel des Widerspruchsbeweises ist es mit Hilfe des Pfades des Survivors, die Ausgabe der drei Steuer-Rotoren abhängig von Stellung, Position und Ausrichtung zu einem Widerspruch zu führen und so zu beweisen, dass diese nicht für die Verschlüsselung der Chiffre verwendet worden sein konnten. Aufgrund der Annahme, dass diese Rotoren in exakt dieser Kombination aus Positionen, Stellungen und Ausrichtungen verwendet wurden, muss es eine eindeutige Verdrahtung zwischen den Rotoren und der Ausgabe geben.

Der Pseudorotor kann zunächst als leere Tabelle angesehen werden. Im Verlauf des Angriffs werden die Einträge dieser Tabelle gefüllt und somit die Verdrahtung des Pseudorotors konstruiert. Für jeden Klartext Buchstaben n wird im Verlaufe dieses Prozesses F, G, H und I durch die drei Steuer-Rotoren substituiert, wie es auch in der Maschine der Fall ist. Die Ausgabe der Rotoren werden als Positionen im Pseudorotor interpretiert und der aktuelle Wert des Pfades, des Survivors an jener Position eingetragen. Da ein Punkt eines Pfades aus 1-4 Anweisungen zum Fortschalten besteht, können diese auch als Mengen verstanden werden. Da in diesem Teil des Angriffs nicht unterschieden werden kann, welchen Ausgang der Steuer-Rotoren, welchen Teil des Pfad-Punktes verursacht,

werden alle Pfad-Punkte in jedem Eintrag verzeichnet. Ist bereits ein Eintrag an der entsprechenden Position im Pseudorotor vorhanden, wird die Schnittmenge aus Pfad-Punkt und Eintrag gebildet. In diesem Fall wird im Folgenden von einer „Kollision“ gesprochen. Ist die Schnittmenge nicht leer, wird sie als neuer Eintrag an dieser Position eingetragen. Sollte die Schnittmenge hingegen leer sein, ist der Widerspruchsbeweis für genau diese Kombination von Steuer-Rotoren und ihren Start-Positionen und -Ausrichtungen erbracht. Es gibt also keine Verdrahtung, die für genau diese Kombination aus Auswahl der Steuer-Rotoren, Position und Ausrichtung für F, G, H und I als Eingabe den Pfad des Survivors bilden kann. Ergo kann diese nicht zur Verschlüsselung der Chiffre verwendet worden sein. Bleibt der Pseudorotor hingegen über den kompletten Pfad hinweg widerspruchsfrei, wird im weiteren Verlauf über diese Kombination aus Stellung, Position und Ausrichtung der Steuer-Rotoren von einem „Kandidaten“ gesprochen.

Ein Beispiel:

Gegeben Survivor Pfad: {0,1,3}, {0,1,2}, {2,3,4}, {1,2,3}, {1,2}, {2,3,4}, {4}, {0,1,2,4}, {1,4}, {1,3,4}, {0,3,4},...

Ausgabe der Steuer-Rotoren:

{A,D,M,N},{C,H,X,Y},{B,E,J,K},{D,R,T,V}{C,H,X,Y},{B,E,J,K},{D,R,T,V}

Index/ n	1	2	3	4	5	6	7
A	{0,1,3}	{0,1,3}	{0,1,3}	{0,1,3}	{0,1,3}	{0,1,3}	{0,1,3}
B			{2,3,4}	{2,3,4}	{2,3,4}	{2,3,4}	{2,3,4}
C		{0,1,2}	{0,1,2}	{0,1,2}	{1,2}	{1,2}	{1,2}
D	{0,1,3}	{0,1,3}	{0,1,3}	{1,3}	{1,3}	{1,3}	∅
E			{2,3,4}	{2,3,4}	{2,3,4}	{2,3,4}	{2,3,4}
(...)							
Z							

Tabelle 4.3: Beispiel für den Verlauf der Einträge eines Pseudorotors

Tabelle 4.3 zeigt exemplarisch, wie sich die Einträge über den Angriff hinweg entwickeln. In der Spalte „Index“ sind die Bezeichnungen der 26 Zeilen zu sehen. Zunächst wird damit begonnen, die Einträge des Pseudorotors zu füllen. Dazu wird die Ausgabe der Steuer-Rotoren heran gezogen. Diese lautet im ersten Schritt {A,D,M,N}. Da für den Pseudorotor nicht entschieden werden kann, welche Teilmenge des Pfad-Punktes welchem Index zugeordnet werden kann, wird jedem Index der Steuerrotorausgabe der gesamte Pfad-Punkt zugeordnet. In der Tabelle wird der erste Eintrag durch die Verzeichnung von {0,1,3} an Position 1A und Position 1D veranschaulicht. Bei der Erstellung des

dritten Eintrags kommt es zu einer ersten Kollision und es wird für den Eintrag 3D der Schnitt aus beiden Mengen gebildet $\{0,1,3\} \cap \{1,2,3\} = \{1,3\}$. Da die Ergebnismenge jedoch nicht leer ist, ist der Widerspruchsbeweis noch nicht erbracht. Dies ist erst bei der Verzeichnung des siebten Eintrags der Fall. Da der Pseudorotor nicht sowohl auf 1 ODER 3 UND 4 abbilden kann, ergibt sich $\{1,3\} \cap \{4\} = \emptyset$.

Der Aufwand für den Widerspruchsbeweis errechnet sich aus dem Durchtesten der drei Steuerungsrotoren, die für den Beweis notwendig sind:

$26^3 \cdot 3^2 \cdot 3! \cdot \binom{5}{3} \approx 2^{23}$

Die Wahrscheinlichkeit eines erfolgreichen Widerspruchs hängt von der Menge erzeugter Kollisionen ab. Es wurde eine Stichprobe von 8 Millionen Tests evaluiert, deren Ergebnisse in Tabelle 4.4 aufgelistet sind.

# Tests	8.000.000
Ø Widerspruch	7,46350580
Ø Kollisionen	2,40018768
Koll. $>$ 2 max.	9
Ø Koll. # Eintrag = 1	1,71817646

Tabelle 4.4: Repräsentative Stichprobe über 8.000.000 Tests

Im Mittel ergibt sich ein Widerspruch nach ca. 7,5 Pfad-Punkten und dass ca. $2,4 \approx 3$ Kollisionen für einen Widerspruch nötig sind. Die Wahrscheinlichkeit wird hier nicht in einer Stichprobe evaluiert werden, da eine aussagekräftige Stichprobe von 1 Millionen Tests einen Aufwand von

$1.000.000 \cdot 2^{23} \cdot 100 = 2^{49}$

und daher zu aufwendig wäre. Stattdessen wird diese hier abgeschätzt:

Die Wahrscheinlichkeit einer einzigen Kollision auf einem bestimmten Eintrag ergibt sich aus der hypergeometrischen Verteilung

- $N = 26$ Menge aller möglichen Substitutionen
- $M = 4$ Menge der Substitutionen
- $n = 1$ Anzahl der günstigen Kollisionen
- $k = 1$ Elemente aus M die in n enthalten sind

$$\text{P(X=k)=} \frac{\binom{M}{k} \cdot \binom{N-M}{n-k}}{\binom{N}{k}} \approx 0,154$$

Es wird weiterhin die Wahrscheinlichkeit abgeschätzt, nach n Versuchen o Einträge mindestens 3 mal zur Kollision gebracht zu haben. Aus der Stichprobe ergibt sich, dass auf maximal $o = 9$ Einträgen mehr als 3 Kollisionen erzeugt wurden, bevor der Widerspruchsbeweis erbracht wurde. Dieser Wert wird hier als obere Grenze zum Abschätzen verwendet. Die Wahrscheinlichkeit ergibt sich demnach aus $P = 1 - P'$ wobei P' die Wahrscheinlichkeit ist, keine 3 Kollisionen nach n Versuchen auf o Einträgen erzeugt zu haben. P' wiederum lässt sich abschätzen durch die obere, kumulative Verteilungsfunktion der Bernoulli-Kette:

$P' \leq \sum_{i=\lfloor o \rfloor}^{26} \binom{26}{i} \cdot P''^{i} \cdot (1 - P'')^{26-i}$.

[BB06]

P'' letztendlich ist die Wahrscheinlichkeit, dass die Kollision eines bestimmten Eintrages weniger als 3 mal erzeugt wurde und lässt sich mit Hilfe der oberen, kumulativen Verteilungsfunktion der Bernoulli-Kette abschätzen:

- p = Die Wahrscheinlichkeit einer bestimmten Kollision
- n = Die Anzahl verfügbarer Pfad-Punkte (Versuche)
- $k = 2$ Die Anzahl der maximalen Kollisionen

$P''(X \leq k) = \sum_{i=0}^{\lfloor k \rfloor} \binom{n}{k} \cdot p^i \cdot (1 - p)^{n-i}$

Die Wahrscheinlichkeit eines Widerspruchs beträgt demnach die Gegenwahrscheinlichkeit zu P'

$P = 1 - P'$

In der Abbildung 4.5 ist zu erkennen, dass ab einer Größe von 30 Pfad-Punkten eine Wahrscheinlichkeit von P''<0,00000001 für keinen Widerspruch bei einem falschen Schlüssel besteht und weiter gegen 0 konvergiert. Das bedeutet, dass sich für 30 Pfad-Punkte mit

$2^{23} \cdot 0,0000001 \approx 0,839 < 1$

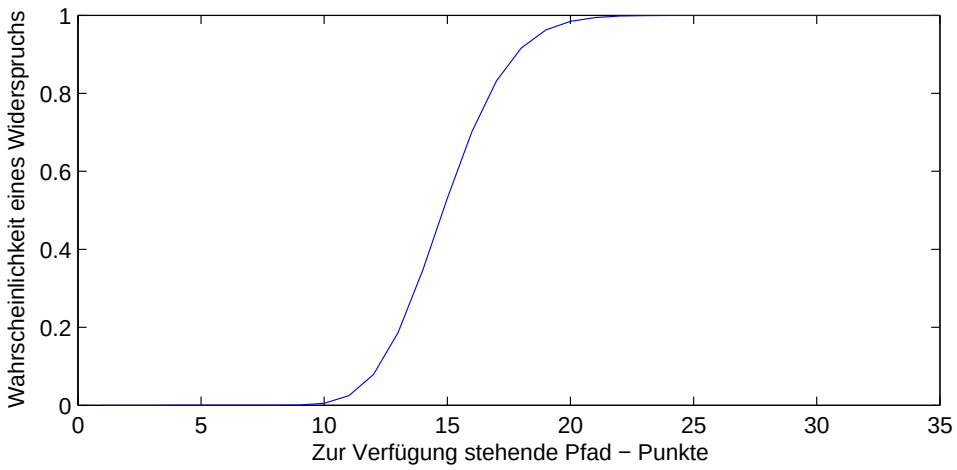

Abbildung 4.4: Darstellung von P

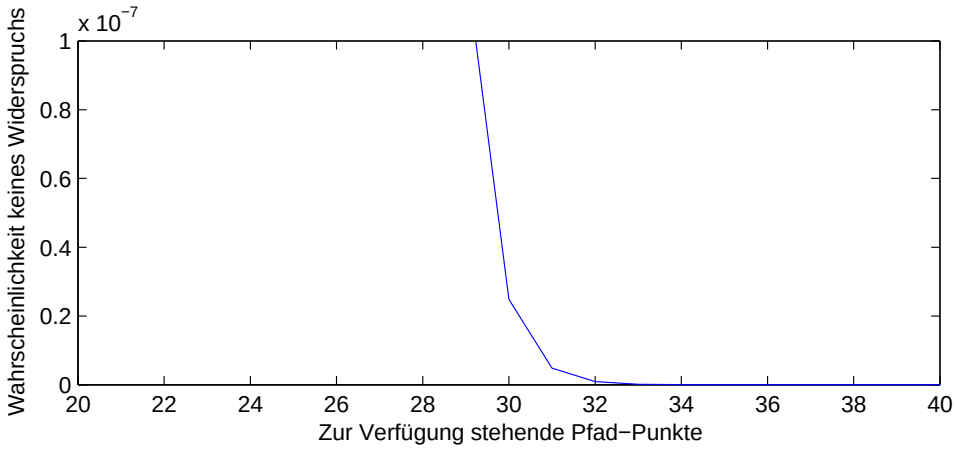

Abbildung 4.5: Darstellung von P'

weniger als ein einziger widerspruchsfreier Kandidat qualifiziert, ein Teil des Schlüssels zu sein. Zusammengefasst lässt sich festhalten, dass durch das Verfahren des Widerspruchsbeweises, für einen Survivor mit einer Pfadlänge > 30, sich weniger als ein potentieller Kandidat ergibt, welcher verifiziert werden muss, ob er Teil des Schlüssels ist.

Es kann festgestellt werden, dass durch den Widerspruchsbeweis falsche Schlüssel ausgeschlossen werden können. Widerspruchsfreie Ergebnisse hingegen müssen verifiziert und der restliche Schlüssel des Fortschaltungs-Labyrinths gefunden werden, um schließlich den Text dechiffrieren zu können.

4.3.1.2 Der Distanztest

Das erste Verifikationsverfahren, das hier vorgeschlagen wird, ist der Distanztest. Jeder Kontakt eines Rotors ist definiert durch eine Substitutionsdistanz. Diese Distanz ergibt sich aus dem Betrag der Differenz von Eingabe und Ausgabe eines Rotors. Die Menge aller Distanzen bildet für einen Rotor ein eindeutiges Muster. Gleichen sich die Muster zweier Rotoren, sind auch die Rotoren gleich in ihrer Substitution.

Die widerspruchsfreien Kandidaten können durch Einsetzen der verbleibenden beiden Chiffre-/Steuer-Rotoren in die Steuer-Bank an Position des Rotoren S4 verifiziert werden. Das Verfahren gestaltet sich dabei ähnlich dem des Widerspruchsbeweises. Die Eingabe F, G , H und I werden in diesem Verfahren durch alle vier Rotoren substituiert, in einen Pseudorotor eingetragen und auf Kollisionen, bis hin zu einem Widerspruch untersucht.

Auf Grund dieser eindeutigen Muster des zusätzlich eingesetzten Rotoren, ergeben sich mit einer gewissen Wahrscheinlichkeit weitere Widersprüche.

Der Aufwand des Distanztests ergibt sich aus der Überprüfung der restlichen zwei Chiffre-/Steuer-Rotoren.

$$26 \cdot 2^2 \cdot 2 = 2^{12}$$

Mit dem Distanztest können die Kandidaten nur verifiziert werden. Durch den Distanztest kann demnach nur ermittelt werden, ob der Rotor in der Maschine verwendet wurde und verifiziert damit die Verwendung der anderen drei Rotoren des Widerspruchsbeweises. Es können keine Aussagen über Stellung, Position und Ausrichtung der Rotoren getroffen werden. Dies liegt daran, dass lediglich die Distanzen überprüft werden. Verändert der Rotor seine Stellung, Position oder Ausrichtung, bleiben die Distanzen gleich. Der Pseudorotor fängt demnach an sich „mitzudrehen“, er erzeugt also insgesamt äquivalente Verdrahtungen.

Die Wahrscheinlichkeit eines Erfolgs des Distanztests konnte im Rahmen dieser Arbeit nicht mehr evaluiert werden und wird für zukünftige Arbeiten zur Diskussion gestellt. Auf Grund des kleineren Aufwand gegenüber dem Verfahren zur Schlüsselsuche, weisen erste nicht repräsentative Tests darauf hin, dass dieser für Pfadlängen < 30 effizient sein könnte.

4.3.1.3 Vollständiges Testen der Mehrdeutigkeiten im Pseudorotor

Durch die Verfahren des Widerspruchsbeweises und der Verifikation können nur falsche Schlüssel ausgeschlossen werden. Um jedoch den gesamten Schlüssel des Fortschaltungs-Labyrinths zu erhalten und die Chiffre entschlüsseln zu können, sind weitere Schritte notwendig.

Bei den hier vorgestellten Verfahren spielt der Pseudorotor, wie er im Widerspruchsbeweis erstellt wurde, weiterhin eine zentrale Rolle. Wie bereits erwähnt, stellt der Pseudorotor eine gleichwertige Verdrahtung dar, mit der es nicht nur möglich ist, Schlüssel zu überprüfen, in dem Fortschaltungen mit der Ausgabe zur Kollision gebracht werden, sondern ebenfalls, weitere Fortschaltungen zu generieren. Dafür muss der Pseudorotor jedoch vollständig sein. Vollständig heißt in diesem Fall, dass die Menge eines jeden Eintrags genau gleich 1 ist.

Es werden hier zwei Verfahren unterschieden:

- Das Verfahren zur Schlüsselsuche
- Das Verfahren zur schnellen Dechiffrierung

4.3.1.4 Das Verfahren zur Schlüsselsuche

Beim Verfahren zur Schlüsselsuche steht die Suche nach dem korrekten Schlüssel des Fortschaltungs-Labyrinths im Fokus. Zunächst werden alle mehrdeutigen Einträge des Pseudorotors sukzessive durchgetestet. Jeder Test besteht zunächst aus einem Zählen der Steuerrotorausgaben, im Weiteren Fortschaltungsanweisungen genannt. Aus der Spalte „Buchstabe“ der Tabelle 4.1 ergibt sich, dass jede Fortschaltungsanweisung mindestens ein mal vorkommen muss und höchstens elf mal im Pseudorotor vorkommen darf. Ist dies nicht der Fall, ist dieser spezielle Pseudorotor nicht gültig und es kann die nächste Kombination aus Einträgen getestet werden.

Wie viele Pfad-Punkte nötig sind, damit der Pseudorotor vollständig ist, kann mit der Formel des sogenannten Sammelbilderproblems abgeschätzt werden. [Kun10]

Wie in Tabelle 4.4 zu sehen ist, werden im Mittel 1,7 Kollisionen gebraucht, um einen Eintrag des Pseudorotors eindeutig zu bestimmen. Da für jeden Pfad-Punkt vier neue Einträge verzeichnet werden, ergibt sich durch

$\frac{4}{1{,}7} \approx 2$

im Mittel für jede zweite Verzeichnung ein eindeutiger Eintrag. Für den ersten Eintrag des Pseudorotors gibt es 26 günstige aus 26 möglichen Varianten, um einen neuen Eintrag zu verzeichnen. Der zweite Eintrag hat nun 26 - 1 (ein Eintrag ist bereits vorhanden) günstige aus 26 - 1(kein Eintrag kann in einem Durchgang doppelt erstellt werden) mögliche Varianten, um ebenfalls eindeutig bestimmt zu werden. Für den ersten Pfad-Punkt ergibt sich somit

$\frac{26}{26} + \frac{26-1}{26-1}$

für den zweiten

$\frac{26}{26-3} + \frac{26-1}{26-4}$

(...)

- $n = 26$ die Größe des Pseudorotors
- $p = 2$ Menge der Einträge, die im Mittel pro Pfad-Punkt einen eindeutigen Eintrag erzeugen
- $S =$ Pfad-Punkte die nötig sind für einen vollständigen Pseudorotor

$S = \sum_{i=0}^{n-1} \frac{n - (i \ mod \ p)}{n - i}$

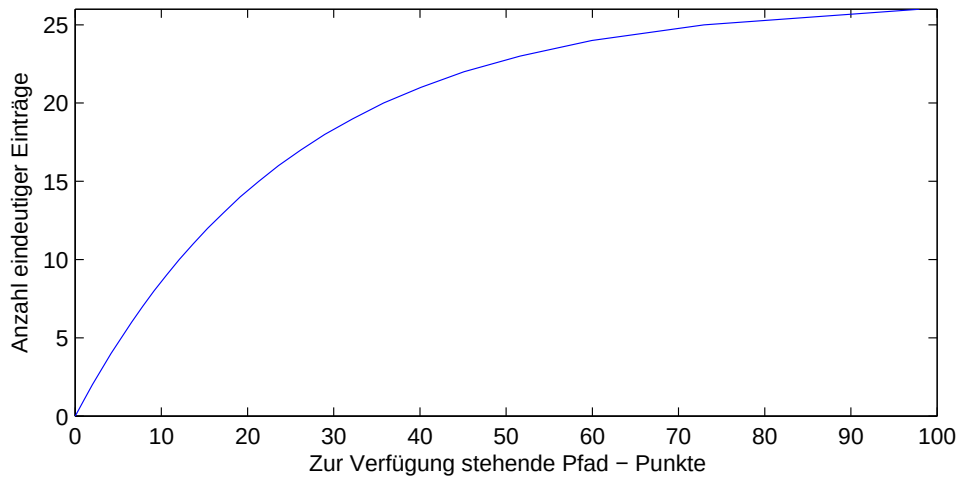

Abbildung 4.6: Darstellung wie sich S entwickelt

In der Abbildung 4.6 ist zu erkennen, dass für 95 Pfad-Punkte jeder Eintrag des Pseudorotors eindeutig ist.

Der Aufwand des Verfahrens S zur Schlüsselsuche ergibt sich folglich aus der Menge der mehrdeutigen Einträge $(26 - n)$, wobei n die Anzahl der eindeutigen Einträge ist.

Da ein mehrdeutiger Eintrag zwischen mindestens zwei und maximal vier Elementen, also durchschnittlich drei Einträge hat, ist die durchschnittliche Anzahl der zu testenden Pseudorotoren 3 potenziert mit der Anzahl der mehrdeutigen Einträge. Weiterhin ergibt sich aus dem Pseudorotor und dem Durchtesten der verbleibenden beiden Rotoren der Chiffre-Bank in Position 2!, Stellung 26^2 und Ausrichtung 2^2:

- n = Anzahl eindeutiger Einträge im Pseudorotor

$S(n) = 3^{(26-n)} \cdot 26^2 \cdot 2^2 \cdot 2!$

Das Index-Labyrinth muss ebenfalls durchgetestet werden. Dieser Test gestaltet sich in der Form, als dass statt der fünf Index-Rotoren, die äquivalenten 113.400 möglichen festen Verdrahtungen durchgetestet werden, da dies, wie bereits diskutiert, einen kleineren Aufwand darstellt. Weiterhin sind durch die ungleichmäßige Gruppenverdrahtung, siehe Abbildung 3.6, zwischen Index- und Steuer-Labyrinth nicht jede der 113.400 Verdrahtungen zulässig. Statt alle Verdrahtungen zu testen, werden die Häufigkeiten der Werte der Einträge im Pseudorotor gemessen und alle validen Verdrahtungen, die dieser Häufigkeit entsprechen, getestet.

Ein Beispiel:

Die Messung hat ergeben, dass der aktuelle Pseudorotor folgende Häufigkeiten an Einträgen aufweist:

Eintrag	1	2	3	4	5
Häufigkeit	7	7	4	4	4
Satz 1	(2,4)	(0,6)	(5,9)	(1,8)	(3,7)
Satz 2	(0,6)	(1,5)	(4,9)	(2,8)	(3,7)
Satz 3	(0,6)	(2,5)	(1,4)	(8,9)	(3,7)
Satz 4	(0,6)	(2,5)	(4,9)	(1,8)	(3,7)
Satz 5	(0,6)	(5,9)	(1,4)	(2,8)	(3,7)
Satz 6	(0,6)	(5,9)	(2,4)	(1,8)	(3,7)

Tabelle 4.5: Beispiel: Häufigkeitsanalyse der Ausgänge eines Pseudorotors

Das bedeutet für diesen Pseudorotor kann das Index-Labyrinth nur aus einem Satz von Paaren der Tabelle 4.5 bestehen, die diese Häufigkeit an Buchstaben aufweisen. Die Häufigkeit von 7 haben nur die Verdrahtungen (1,8) (2,8) (8,9) (3,7) (4,6) oder (5,6) und die Häufigkeit von 4 nur die Verdrahtungen (0,6) (1,5) (2,5) (5,9) (1,4) (2,4) und (4,9). Unter der Prämisse, dass keine Ziffer eines Paars zwei mal vorkommen darf, ergeben sich daraus sechs Möglichkeiten. Unklar ist jedoch, wie die Verdrahtung im Speziellen ist. Daher müssen alle Permutationen dieser Paarungen, also 5! ebenfalls getestet werden. Daraus ergeben sich insgesamt

$6 \cdot 5! = 720$

Möglichkeiten an Verdrahtungen für dieses Beispiel.

Die absolute Menge unterschiedlicher Häufigkeiten von Buchstaben beträgt genau 89. Insgesamt gibt es 945 einzigartige Sätze von Paarungen der Gruppen, unter den Prämissen, der Einzigartigkeit jeder Ziffer und der Summe von insgesamt genau 26 Buchstaben.

An dieser Stelle muss angemerkt werden, dass sich hier diese Arbeit von Chans unterscheidet. Die Zahl 945 wurde im Zuge dieser Arbeit durch einen Algorithmus evaluiert, welcher zunächst alle Paarungen errechnet und jene ohne doppelte Ziffern zählt. Chan spricht in seiner Arbeit hingegen von 2148 Gruppierungen, erwähnt an dieser Stelle jedoch ebenfalls keine Formel, um diese Zahl kombinatorisch zu ermitteln. Die weiteren Berechnungen stützen sich auf die evaluierten 945 Gruppierungen.

Im Durchschnitt hat jede gemessene Häufigkeit $\frac{945}{89} \approx 10,618$ Sätze von Paarungen, wobei alle Permutationen der Paarungen eines Satzes durchgetestet werden müssen. Das wiederum ergibt im Durchschnitt

$10,618 \cdot 5! \approx 1274,16$

Möglichkeiten an Verdrahtungen. Diese Verdrahtungen werden alle als mögliche Verdrahtungen des Index-Labyrinthes durchgetestet. Die Korrekte Verdrahtung muss ein Teil dieser Menge sein, es sei denn der Pseudorotor war falsch.

Die Verdrahtungen können zusammen mit den Steuer-Rotoren überprüft werden. Dazu werden wieder die Substitutionen der ersten drei Rotoren mit den beiden restlichen Rotoren und den Verdrahtungen durchgetestet. Die Ergebnisse müssen exakt den Pfad-Punkten des Survivors entsprechen. Ergeben sich hier Widersprüche, ist die Auswahl der ersten Rotoren oder der Pfad des Survivors falsch.

Es ergibt sich für das gesamte Fortschaltungs-Labyrinth ein Schlüsselraum S von

- n der Anzahl eindeutiger Einträge

$S(n) = 3^{(26-n)} \cdot 26^2 \cdot 2^2 \cdot 2! \cdot 1274,16$

für das Verfahren zur vollständigen Schlüsselsuche, bei einer Erfolgswahrscheinlichkeit von 1,0. Die Einstellungen des Index-Labyrinths können durch erschöpfendes Durchtesten der Index-Rotoren gefunden werden, wobei die erste Stellung akzeptiert werden kann, welche für jede Eingabe der Verdrahtung, dieselbe Ausgabe wie die Verdrahtung liefert.

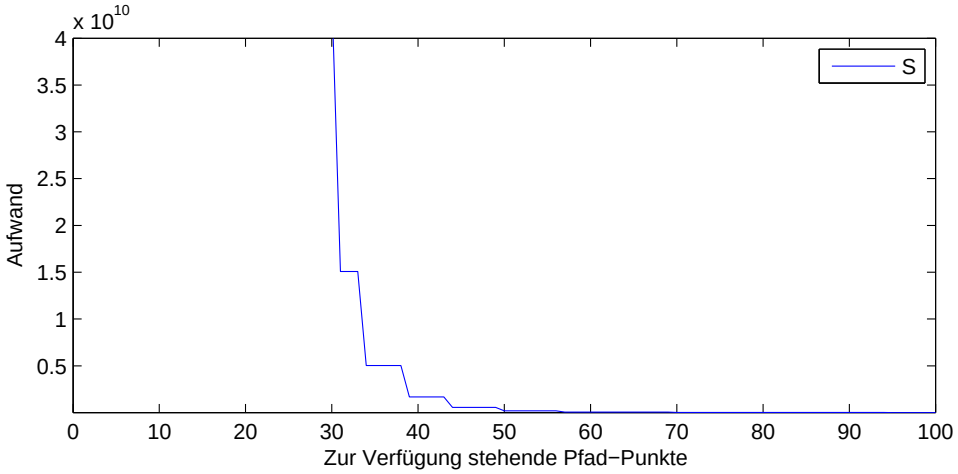

Abbildung 4.7: Darstellung wie sich S entwickelt

In Abbildung 4.7 ist zu erkennen, dass ab einer Pfadlänge von 30 ein Aufwand von $4 \cdot 10^9 = 2^{32}$ zum Durchtesten der Mehrdeutigkeiten erforderlich ist. Der Aufwand für dieses Verfahren konvergiert insgesamt für $n \to 26$ gegen

$1 \cdot 26^2 \cdot 2^2 \cdot 2! \cdot 1274,16 \approx 2^{23}$

4.3.1.5 Das Verfahren zur schnellen Dechiffrierung

Das nächste, hier vorgestellte Verfahren, verzichtet hingegen auf den Test zur Validierung der Verdrahtung. Statt dessen wird hierbei lediglich ein Rotor an Position S4 eingesetzt, ähnlich wie beim Distanztest. Danach wird der Pseudorotor ermittelt und abschließend die gesamte Chiffre für jede Mehrdeutigkeit des Pseudorotors entschlüsselt. Die Ausgabetexte werden dann wiederum von einer Kostenfunktion bewertet. Der Text mit dem besten Kostenwert wird als wahrscheinlichster Klartext angenommen.

Der Aufwand S für diesen Angriff errechnet sich aus dem Aufwand des eingesetzten Rotors, sowie der Vollständigkeit des Pseudorotors.

- n der Anzahl eindeutiger Einträge

$S(n) = 3^{(26-n)} \cdot 26 \cdot 2 \cdot 2$

Da wie im Distanztest bei einem positiven Ergebnis keine Texte ausgeschlossen werden können, ergeben sich genau so viele Texte wie die Größe des durchsuchten Raumes. Es kann aber garantiert werden, dass der Klartext mit einer Wahrscheinlichkeit von 1,0 in der Ergebnismenge enthalten ist. Zur Ermittlung des verwendeten Schlüssels sind

allerdings die Methoden notwendig, welche im Verfahren zur Schlüsselsuche beschrieben wurden.

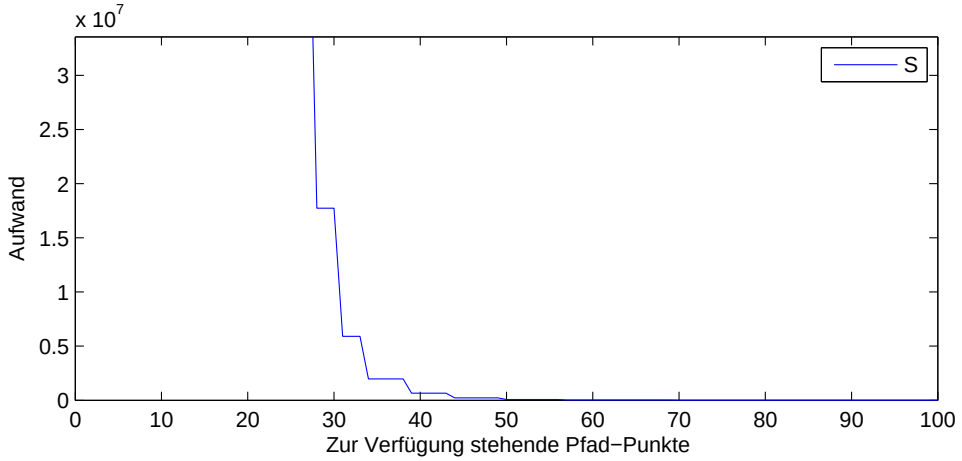

Abbildung 4.8: Darstellung wie sich S entwickelt

Folgerichtig konvergiert auch diese Funktion für $n \to 26$ gegen

$26 \cdot 2 \cdot 2 = 104$

Es ist in Abbildung 4.8 zu erkennen, dass durch den exponentiell fallenden Aufwand ab einer Pfadlänge von 30 ein Aufwand von $2 \cdot 10^7 = 2^{24}$ zum Durchtesten der Mehrdeutigkeiten erforderlich ist.

4.3.1.6 Schlüsselraum Phase II

Der Gesamtaufwand G für einen Survivor aus Phase I lässt sich abschätzen, aus der Wahrscheinlichkeit eines Widerspruch im Widerspruchsbeweis und einem der zuletzt vorgestellten Verfahren. Entweder dem Verfahren zur Schlüsselsuche oder dem zur schnellen Dechiffrierung. Dazu muss der Aufwand dieser Tests kombiniert werden.

Die Anzahl der Kandidaten wird hierbei für beide Verfahren durch die Multiplikation des Schlüsselraums des Widerspruchsbeweises mit der Wahrscheinlichkeit eines Widerspruchs P abgeschätzt durch

$A_1 \cdot P$

Der Gesamtaufwand G_1 für Phase II unter Verwendung des Verfahrens zur Schlüsselsuche, ergibt sich aus

- A_1 dem Aufwand für den Widerspruchsbeweis

- A_2 dem Aufwand für das Verfahren zur Schlüsselsuche
- p der Wahrscheinlichkeit eines Widerspruchs in Abhängigkeit von k
- n der Anzahl der eindeutigen Einträge im Pseudorotor in Abhängigkeit von k
- k der Anzahl der zur Verfügung stehende Pfad-Punkte

$G_1(k) = A_1 + A_2 \cdot(A_1 \cdot p(k))$

$G_1(k) = 2^{23} + 3^{(26-n(k))} \cdot 26^2 \cdot 2^2 \cdot 2! \cdot 1274,16 \cdot (2^{23} \cdot p(k) + 1)$

Es ist zu erkennen, dass der Gesamtaufwand unter Verwendung des Verfahrens zur Schlüsselsuche gegen

$2^{23} + 2^{23} = 2^{24}$

konvergiert.

Analog dazu ergibt sich der Gesamtaufwand G_2 für Phase II unter Verwendung des Verfahrens zur schnellen Dechiffrierung aus

- A_1 dem Aufwand für den Widerspruchsbeweis
- A_2 dem Aufwand für das Verfahren zur schnellen Dechiffrierung
- p der Wahrscheinlichkeit eines Widerspruchs in Abhängigkeit von k
- n der Anzahl der eindeutigen Einträge im Pseudorotor in Abhängigkeit von k
- k der Anzahl der zur Verfügung stehende Pfad-Punkte

$G_2(k) = A_1 + A_2 \cdot(A_1 \cdot p(k))$

$G_2(k) = 2^{23} + 3^{(26-n(k))} \cdot 26 \cdot 2 \cdot 2 \cdot (2^{23} \cdot p(k) + 1)$

Der Aufwand für das Verfahren zur schnellen Dechiffrierung konvergiert gegen

$2^{23} + 104 \approx 2^{23}$

In Abbildung 4.9 ist vergleichend dargestellt, wie sich der Aufwand der verschiedenen Verfahren gegeneinander entwickelt. Da Chan keine näheren Angaben macht, wie sich der Aufwand entwickelt, wird der von ihm erwähnte Aufwand von $2^{43,3}$ als Basiswert verwendet. Es ist deutlich zu erkennen, dass das Verfahren zur schnellen Dechiffrierung bereits ab einer Pfadlänge > 17 weniger aufwendig als das Verfahren von Chan ist. Auch das Verfahren zur Schlüsselsuche unterbietet den Ansatz von Chan ab einer Pfadlänge > 24.

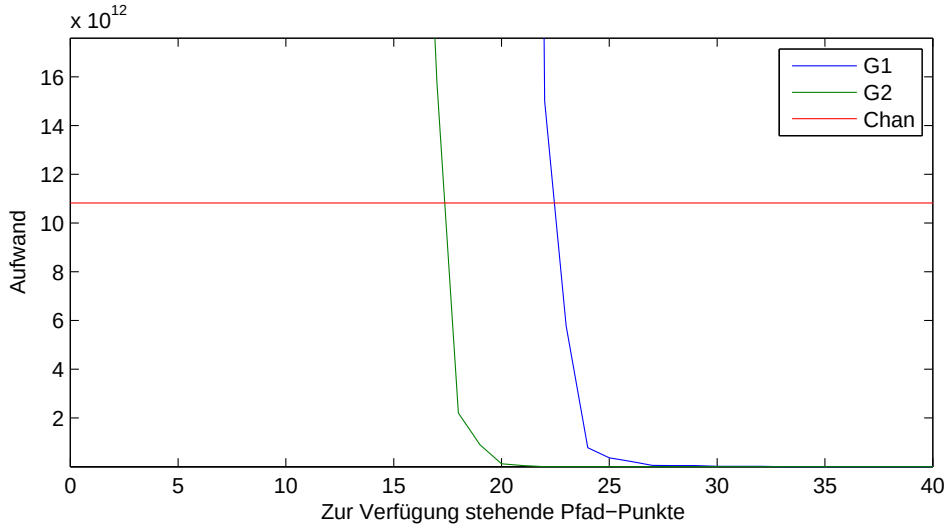

Abbildung 4.9: Vergleichende Darstellung der Verfahren

4.3.2 Ansatz zur Verbesserung von Phase I

Die Ergebnisse der Analyse der ersten Phase haben deutlich gemacht, wie stark der Schlüsselraum des Fortschaltungs-Labyrinths eingeschränkt werden kann, wenn genug Klartext vorhanden ist. Nach wie vor besteht jedoch das Problem, dass die Anzahl der Survivor aus Phase I wächst, je größer die Anzahl an verwendetem Klartext ist.

Aus dieser Überlegung heraus wird deutlich, dass die Trennung zwischen Phase I und II zu stark ist und hier kontraproduktiv wirkt. Hier wird daher vorgeschlagen die Phasen stärker miteinander zu verbinden, um Ergebnisse aus Verfahren von Phase II in Phase I verwenden zu können.

Die Vorgehensweise von Chan ist keine Lösung für die hier vorgestellten Verbesserungen von Phase II, da Chan Pfade zusammenfasst. Diese Zusammenfassung valider Pfade, hat laut Chan nur geringe Auswirkungen auf sein statistisches Verfahren. Für den Widerspruchsbeweis ist dies jedoch fatal, da jeder Pfad-Punkt wichtige Informationen für die Vollständigkeit des Pseudorotors und somit im Endeffekt für den Erfolg eines Widerspruchsbeweis darstellt. Dieser Ansatz ist daher für den Widerspruchsbeweis nicht tauglich.

Es wird hingegen statt einer klaren Trennung von Phase I und Phase II, eine Art von hybridem Ansatz vorgeschlagen. Wie bereits festgestellt, beträgt das Wachstum des Baumes w bei größer werdender Pfadlänge n

$$(\frac{30}{26})^n = w$$

was automatisch auch der Anzahl der Survivor entspricht. Es wird vorgeschlagen, statt zunächst alle Survivor zu ermitteln und diese erst in Phase II durchzutesten, den Widerspruchsbeweis schon in jedem neuen Pfad-Punkt des Baumes durchzuführen. Auf diese Weise können schon frühzeitig ganze Teilbäume als Survivor ausgeschlossen werden.

Ein Beispiel:

In Abbildung 4.10 ist ein Baum von Fortschaltungen vereinfacht dargestellt. Die grünen Ellipsen bedeuten es gab keinen Widerspruch der Pfad-Punkte bis dato, die roten hingegen, dass beim Testen dieses Pfad-Punktes ein Widerspruch auftrat. Tritt ein Widerspruch auf, können alle Pfad-Punkte nach diesem ebenfalls ausgeschlossen werden, ohne durch den Widerspruchsbeweis getestet worden zu sein. An diesem Beispiel wird klar, dass eine Zusammenfassung der Pfad-Punkte, wie sie Chan in seinem Ansatz vorgeschlagen hat, nicht funktionieren würde, da der Punkt 'BBBBB' als Unterpfad von 'BBABB' zwar gültig ist, für 'ABABB' gleichzeitig hingegen nicht. Eine Zusammenfassung würde bewirken, dass der Zustand nicht mehr eindeutig bestimmt werden könnte.

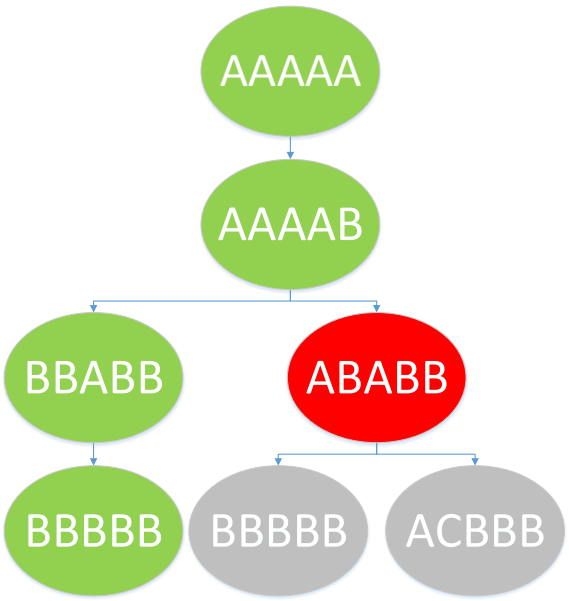

Abbildung 4.10: Darstellung des Widerspruchsbeweises für Phase I

Wie in Tabelle 4.4 gezeigt wurde, tritt ein Widerspruch bei einer durchschnittlichen Pfadlänge von 7,464 auf. Das ergibt für diesen Ansatz eine durchschnittliche Anzahl an

$$\frac{30}{26}^{7,464+1} \approx 3,263$$

Survivorn pro valider Initialstellung der Chiffre-Rotoren. Das bedeutet, der Aufwand für den Widerspruchsbeweis einer einzigen Initialstellung beträgt insgesamt

$2^{23} \cdot 3,263 \approx 2^{25}$

Die Anzahl valider Startstellungs-Kombinationen, ergibt sich aus der Anzahl aller potentieller Kombinationen, der Auswahl der fünf Chiffre-Rotoren, sowie deren Position, Ausrichtung und Stellung. Da für das Alphabet-Labyrinth eine Gleichverteilung der Substitution auf alle 26 Buchstaben angenommen werden kann, ist im Mittel jede 26. Kombination gültig. Dies ergibt durchschnittlich

$$\frac{\binom{10}{5} \cdot 5! \cdot 2^5 \cdot 26^5}{26} \approx 2^{38}$$

valide Startstellungen. In der Folge bedeutet dies, dass wenn der Widerspruchsbeweis in Phase I integriert wird, sich der Aufwand für den Widerspruchsbeweis auf

$2^{25} \cdot 2^{38} = 2^{63}$

erhöht.

4.4 Vergleich der Angriffe

Um den Gesamtaufwand G des hier vorgestellten Angriffs darzustellen, müssen der Aufwand für Phase I und der Aufwand für Phase II multipliziert werden. Der Gesamtaufwand wird hier zum einen für das Verfahren zur Schlüsselsuche und für das Verfahren zur schnellen Dechiffrierung dargestellt.

Analog zu den Berechnungen aus Phase II ergibt sich ein Gesamtaufwand G_1 für das Verfahren zur Schlüsselsuche aus

- A_1 dem Aufwand für Phase I
- A_2 dem Aufwand für das Verfahren zur Vollständigen Schlüsselsuche
- p der Wahrscheinlichkeit eines Widerspruchs in Abhängigkeit von k
- n der Anzahl der eindeutigen Einträge im Pseudorotor in Abhängigkeit von k
- k der Anzahl der zur Verfügung stehende Pfad-Punkte

$G_1(k) = A_1 + A_2 \cdot (A_1 \cdot p(k))$

$G_1(k) = 2^{63} + 3^{(26-n(k))} \cdot 26^2 \cdot 2^2 \cdot 2! \cdot 1274,16 \cdot (2^{63} \cdot p(k))$

Für das Verfahren zur schnellen Dechiffrierung ergibt sich ein Gesamtaufwand G_2 aus

- A_1 dem Aufwand für den Widerspruchsbeweis

- A_2 dem Aufwand für das Verfahren zur schnellen Dechiffrierung
- p der Wahrscheinlichkeit eines Widerspruchs in Abhängigkeit von k
- n der Anzahl der eindeutigen Einträge im Pseudorotor in Abhängigkeit von k
- k der Anzahl der zur Verfügung stehende Pfad-Punkte

$G_2(k) = A_1 + A_2 \cdot (A_1 \cdot p(k))$

$G_2(k) = 2^{63} + 3^{(26-n(k))} \cdot 26 \cdot 2 \cdot 2 \cdot (2^{63} \cdot p(k))$

Die Tabelle 4.4 stellt den Aufwand der verschiedenen Verfahren dar.

Angriff	Pfadlänge	Aufwand	Ws. Erfolg
Erschöpfende Suche	0	2^{96}	1,0
Chan	100	2^{84}	0,82
Schlüsselsuche	1	2^{126}	1,0
Schlüsselsuche	100	2^{63}	1,0
Schnelle Dechiffrierung	1	2^{115}	1,0
Schnelle Dechiffrierung	100	2^{63}	1,0

Tabelle 4.6: Vergleich des Aufwands der verschiedenen Angriffe

In Abbildung 4.11 ist der Aufwand für die Verfahren noch einmal über n dargestellt.

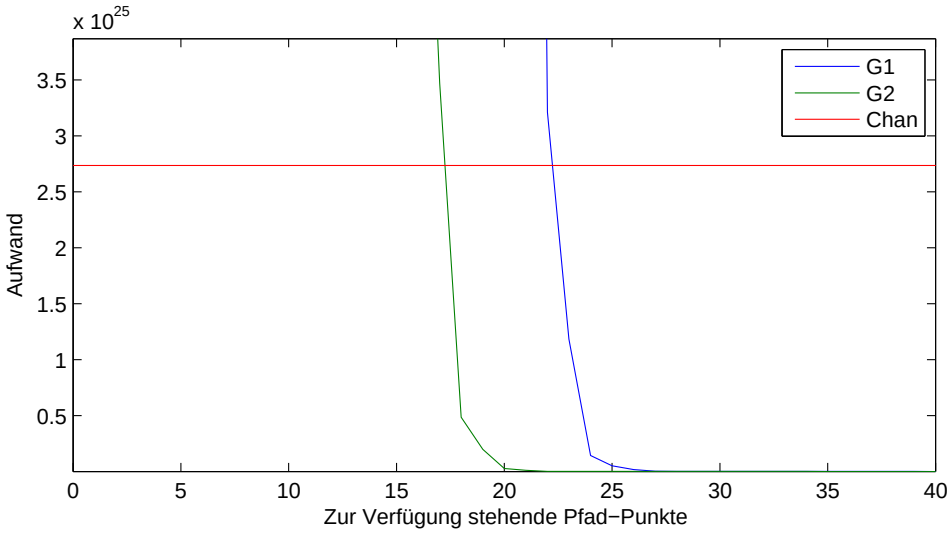

Abbildung 4.11: Darstellung wie sich der Aufwand über k entwickelt

Aus der Tabelle 4.4 ist klar zu erkennen, dass die Verfahren zur Schlüsselsuche und schnellen Dechiffrierung, welche auf dem Widerspruchsbeweis beruhen, erheblich weniger

Aufwand bei genug zur Verfügung stehenden Pfad-Punkten erfordern, als die erschöpfende Schlüsselsuche. Auch gegenüber dem von Chan vorgeschlagenen Angriff erweisen sich die hier vorgestellten Verfahren als günstiger, da sie sowohl weniger Klartext erfordern, als auch höhere Wahrscheinlichkeiten eines Erfolges aufweisen. In Abbildung 4.11 ist zu erkennen, dass das Verfahren zur Schlüsselsuche bereits ab einer Anzahl von 22 Pfad-Punkten effizienter ist, das Verfahren zur schnellen Dechiffrierung ab 17.

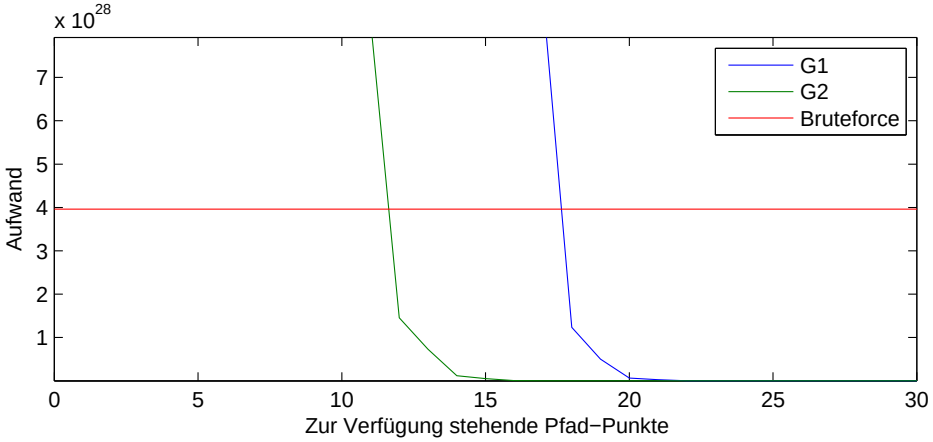

Abbildung 4.12: Darstellung wie sich der Aufwand über k entwickelt

Die hohen Werte im Aufwand für kleine Pfadlängen der hier vorgestellten Verfahren unter Verwendung des Widerspruchsbeweises, erklären sich durch den sehr großen Schlüsselraum des Pseudorotors. Für wenig Klartext, sind die hier vorgestellten Verfahren demnach ineffizient. Wie in Abbildung 4.12 sind die hier vorgestellten Verfahren erst ab einer Pfadlänge > 11 für das Verfahren zur schnellen Dechiffrierung und > 18 für das Verfahren zur Schlüsselsuche weniger aufwendig, als die erschöpfende Schlüsselsuche.

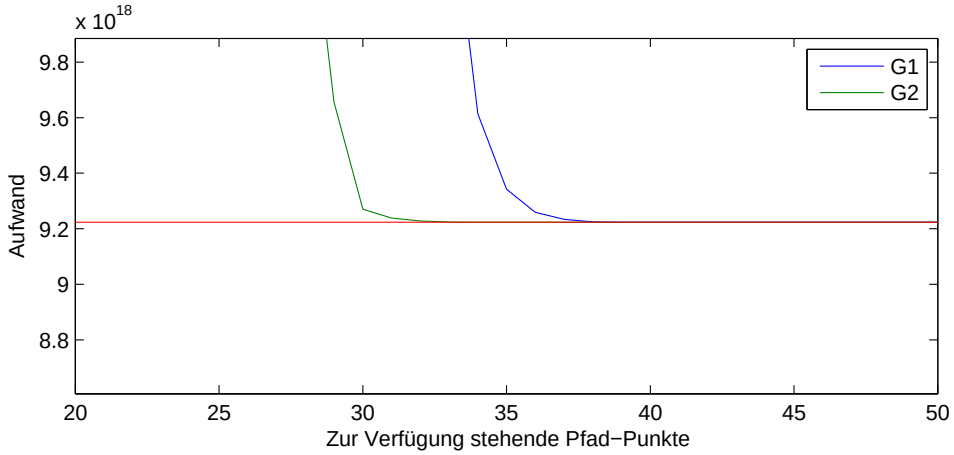

Abbildung 4.13: Darstellung wie sich der Aufwand über k entwickelt

In Abbildung 4.13 ist zu erkennen, dass ab 34 Pfad-Punkten der Schlüsselraum des

Verfahrens zur Schlüsselsuche die $9,8 \cdot 10^{18} \approx 2^{63,1}$ Marke an Aufwand unterschreitet, während das Verfahren zur schnellen Dechiffrierung dies ab 29 Pfad-Punkten tut.

5 Implementierung

Die im vorherigen Kapitel dargestellten Ansätze wurden im Zuge dieser Arbeit als Komponenten für CrypTool 2.0 implementiert. Insgesamt wurden zwei verschiedene Komponenten erstellt:

- Komponente SIGABA-Schlüsselsucher
- Komponente SIGABA-Widerspruchsbeweis

5.1 Komponente SIGABA-Schlüsselsucher

Ziel dieser Komponente ist es, eine gegebene Chiffre nach dem Verfahren der erschöpfenden Schlüsselsuche zu analysieren und den wahrscheinlichsten Klartext für die Chiffre auszugeben. Die Komponente bekommt dabei als Eingabedaten die Chiffre als Text, als Ausgabedaten wiederum liefert sie den Klartext. Eine weitere Anforderung ist, dass der Benutzer den Schlüsselraum über die Einstellungen manuell eingrenzen kann. Da die erschöpfende Suche, wie schon erwähnt, die aufwendigste Art der Analyse darstellt, kann diese Komponente als Maßstab für andere Verfahren, wie zum Beispiel dem Widerspruchsbeweis, verwendet werden.

5.1.1 Einstellungen

Auf Grund des komplexen Aufbaus der SIGABA und der Vielzahl an Einstellungsmöglichkeiten der Bauteile, sind die Einstellungsmöglichkeiten für diese Komponente ebenfalls umfangreich. Da diese Komponente anderen Analyseverfahren als Referenz dienen soll, müssen die Einstellungen jedoch möglichst Präzise den Schlüsselraum eingrenzen können.

Grob sind die Einstellungsmöglichkeiten in vier logische Gruppen unterteilt:

- Schlüsselraumberechnung

- Sichtbarkeit von Einstellungen
- Stellungen
- Auswahl nach Position

Die Gruppe „Schlüsselraumberechnung“ hat nur eine einzige Einstellung. In dieser Rubrik kann der Schlüsselraum der aktuellen Konfiguration durch Druck auf eine Schaltfläche als Zweierpotenz errechnet werden.

In der Rubrik „Stellung“ kann eingestellt werden, in welchem Raum von Stellungen jeder Rotor durchsucht werden soll, beispielsweise ’Chiffre-Rotor 1’ von der Stellung ’A’ bis zur Stellung ’F’, oder ’Index-Rotor 3’ von der Stellung ’3’ bis ’8’. Außerdem können die zu analysierenden Ausrichtungen der Rotoren in Abhängigkeit ihrer Position gewählt werden, also für jeden Rotor, ob seine ’normal Ausrichtung’, ’rückwärts Ausrichtung’ oder beide Ausrichtungen in den Schlüsselraum mit einfließen sollen.

In der letzten Rubrik findet sich für jede Rotor-Position innerhalb der Maschine eine weitere Gruppe, in der mit Hilfe von Kontrollkästchen vom Benutzer ausgewählt werden kann, welche Rotoren für diese Position innerhalb der Maschine in Betracht kommen. Kommt für Position 4 der Chiffre-Bank beispielsweise nur die Rotoren ’III’ und ’IV’ in Frage, würde der Benutzer nur die Kontrollkästchen dieser Rotoren markieren. Da diese Einstellungen recht umfangreich sind und die Einstellungen durch das Anzeigen dieser Einstellungen sehr unübersichtlich werden können, kann die Sichtbarkeit dieser Gruppen in der Rubrik „Sichtbarkeit der Einstellungen“ gesteuert werden.

5.1.2 Algorithmus

Die Implementierung eines Ansatzes zur erschöpfenden Suche kann durch eine Schleife gelöst werden, deren Schleifenvariante jede Form des Schlüssels annimmt. Innerhalb der Schleife wird dann versucht, die Chiffre mit Hilfe der Implementierung der Maschine und der Schleifenvariante in den Klartext zu überführen.

Beispiel Fahrradschloss:

Eine Methode, die das Fahrradschloss nach dem Prinzip der erschöpfenden Schlüsselsuche löst, sieht zum Beispiel aus wie in Textauszug 5.1.

Die Methode $testeSchlüssel(i)$ testet in diesem Beispiel den Schlüssel, versucht quasi das Schloss zu öffnen und gibt „true“ zurück, wenn der Schlüssel korrekt ist und „false“, wenn der Schlüssel falsch war. Die Methode $burteForceLock()$ gibt den richtigen Schlüssel zurück.

```
1  public int bruteForceTesteSchlüssel()
2   {
3    int value=-1;
4    for(int i= 0;i<1000;i++)
5     {
6       if(testeSchlüssel(i))
7       value = i;
8      }
9    return value;
10  }
```

Textauszug 5.1: Methode zur erschöpfenden Schlüsselsuche eines Fahrradschloss

Nach diesem Prinzip wurde auch diese Komponente implementiert, mit zwei Einschränkungen. Das Ergebnis des Tests innerhalb der Schleife ist ein Klartext, welcher mit einer Kostenfunktion bewertet werden muss. Zum anderen ist die SIGABA natürlich wesentlich komplexer aufgebaut als ein Fahrradschloss.

Die Implementierung hat daher nicht eine einzige Schleife. Stattdessen wird um jede Variable innerhalb der Maschine, sprich Stellung, Ausrichtung und Position der Rotoren eine Schleife konstruiert. Die Invarianten dieser Schleifen ergeben sich dabei aus den zuvor thematisierten Einstellungen des Benutzers. Auf diese Weise wird sichergestellt, dass der Schlüssel sich stets innerhalb dieser Grenzen befindet.

Insgesamt gibt es demnach 18 Schleifen, welche in der Reihenfolge von Textauszug 5.2 konstruiert wurden.

Der Schlüssel wird dann aus den Schleifenvarianten zusammengesetzt.

Die SIGABA-Komponente ist über die Integrierte-Komponente-Schnittstelle in diese Komponente integriert. So ist es möglich hier die Entschlüsselungsmethode der SIGABA Komponente zu verwenden. Ebenfalls über dieser Schnittstelle integriert ist die Kostenfunktion, mit der der Kostenwert für den Klartext ermittelt werden kann. Die Komponente führt eine Bestenliste, in welcher diese stets die Klartexte mit den zehn besten Kostenwerten speichert und verwaltet. Ist der Kostenwert des neuen Klartextes besser, als der letzte der Bestenliste, wird dieser der Bestenliste hinzugefügt. Die Bestenliste sortiert sich dann selbst und entfernt den schlechtesten Wert.

Optimierung:

Grundsätzlich ist die Verschachtelung der Schleifen äquivalent. Bei diesem Verfahren wird jedoch klar, dass die Entschlüsselungsmethode der SIGABA-Komponente der Flaschenhals ist, da diese bei jedem Schleifendurchlauf ausgeführt wird. Auf Grund der

```
for(Positionen der Index-Rotoren){
 for(Stellung Index-Rotor 1){
  for(Stellung Index-Rotor 2){
          (...)
    for(Stellung Index-Rotor 5){
     for(Positionen der Chiffre-/Steuer-Rotoren){
      for(Ausrichtungen der Chiffre-/Steuer-Rotoren){
       for(Stellung Chiffre-/Steuer-Rotor 1){
        for(Stellung Chiffre-/Steuer-Rotor  2){
          (...)
          for(Stellung Chiffre-/Steuer-Rotor  10){
              Schlüssel s = alle Schleifenvarianten;
              Klartext k = SIGABA(s, Chiffre);
              Wert w = Kostenfunktion(k);
              if(w > Bestenliste.Last){
                Bestenliste.add(w,k);
              }
            }
        }
       }
      }
     }
    }
  }
 }
```

Textauszug 5.2: Aufbau der Hauptmethode der erschöpfenden Schlüsselsuche

Tatsache, dass Steuer-Rotor 4 und 5 sowie die Index-Bank durch ein einziges Array ersetzt wurden und die Neuinitialisierung auf Grund von Änderungen der ersetzten Rotoren auf ein Minimum reduziert werden muss, sind die Schleifen, welche diese Rotoren steuern, weiter außen als die der anderen. Auch ist ein häufiger Wechsel der Positionen der Rotoren eher zu vermeiden, daher sind diese Schleifen ebenfalls eher außen.

5.1.3 Präsentation

Die Präsentation der Komponente besteht aus einer grafischen Darstellung der Bestenliste, wie sie in Abbildung 5.1 zu sehen ist. Die Implementierung der Oberfläche wurde im Zuge dieser Arbeit nicht selbst entwickelt, sondern beruht ursprünglich auf der Implementierung der Präsentation der Schlüsselsucher-Komponente in CrypTool 2.0.

Die Präsentation teilt sich in zwei Bereiche, „Top Ten“ und „Main“.

In Main werden verschiedene Statusinformationen der Komponente angezeigt. Darunter der Zeitpunkt des Starts mit Datum, die erwartete Endzeit, ebenfalls mit Datum, wie viel Zeit seit dem Start der Komponente vergangen ist, wie lange das Durchsuchen des restlichen Schlüsselraums aller Voraussicht nach noch dauern wird und wie viele Schlüssel pro Sekunde im Moment getestet werden.

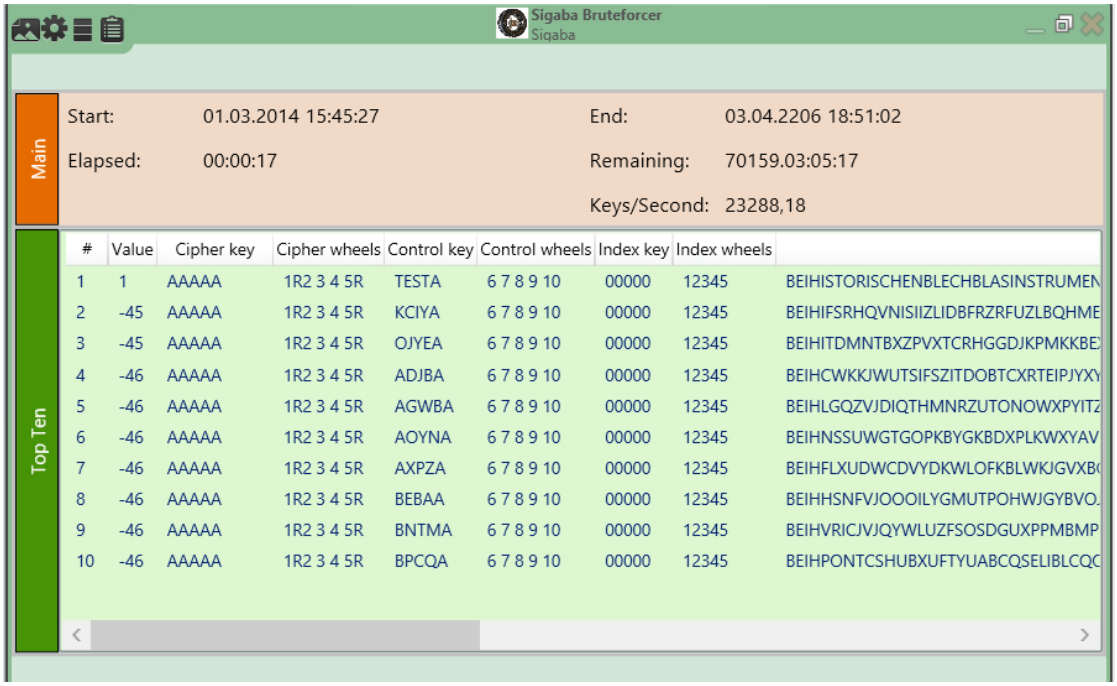

Abbildung 5.1: Darstellung der Präsentation

Der Bereich Top Ten nimmt die größere Fläche der Präsentation ein. In der Abbildung ist zu sehen, dass die Klartexte ihrem Kostenwert nach geordnet aufgeführt sind. Danach gibt es sechs Schlüsselkategorien. In den Spalten welche das Wort „key“ enthalten, sind die Stellungen der jeweiligen Rotoren „Chiffre“, „Steuer“ oder „Index“ gelistet. Die Spalten mit dem Wort „Wheels“ zeigen die Anordnungen der jeweiligen Rotoren. Der Zusatz ’R’ hinter einer Anordnung zeigt an, dass dieser Rotor „rückwärts“ eingelegt wurde.

Die letzte Rubrik zeigt den Klartext in Gänze an. Ein Doppelklick auf einen Eintrag der Liste, verwendet jenen Klartext als Ausgabedaten der Komponente.

5.2 Komponente SIGABA-Widerspruchsbeweis

In dieser Komponente, wurde der in der Kryptoanalyse vorgestellte Ansatz, unter Ausnutzung des Widerspruchsbeweises, des Distanztestes und dem Verfahren zur Schlüsselsuche, sowie der schnellen Dechiffrierung, vollständig als Komponente implementiert.

Die Komponente SIGABA-Widerspruchsbeweis erhält als Eingabedaten neben der Chiffre einen Klartext, beides als Text. Als Ausgabedaten gibt sie den Klartext zurück.

5.2.1 Algorithmus

Der Ansatz wurde in einem Algorithmus implementiert. In Abbildung 5.2 ist dieser Algorithmus als Flussdiagramm grafisch dargestellt. Der Algorithmus beginnt mit dem Durchtesten der Chiffre-Bank. Wie die Komponente zur erschöpfenden Suche, wurde dies, algorithmisch durch mehrere Schleifen gelöst, welche alle Stellungen der Chiffre-Bank durchtestet. Gemäß dem ursprünglichen Ansatz von Chan wird dabei für jede Stellung der erste Buchstabe der Chiffre durch die Chiffre-Bank entschlüsselt und mit dem ersten Buchstaben des Klartextes verglichen. Handelt es sich um denselben Buchstaben, wird die sogenannte Baumliste berechnet. Handelt es sich um einen anderen Buchstaben, wird der nächste Schlüssel ausprobiert. Die Setze Chiffre-Bank Methode ist hier in Textauszug 5.3 noch einmal als Pseudocode dargestellt.

```
1  for(Auswahl der Chiffre-Rotoren){
2   for(Positionen der Chiffre-Rotoren){
3    for(Ausrichtungen der Chiffre-Rotren){
4     for(Stellung Chiffre-Rotor 1){
5      for(Stellung Chiffre-Rotor 2){
6       (...)
7       for(Stellung Chiffre-Rotor 5){
8        Schlüssel s = Schleifenvariablen;
9        Liste<Survivor> SurvivorListe;
10       if(SIGABA.ChiffreBank(s,chiffre[0])==klartext[0])
11        SurvivorListe = SuchBaum(Klartext,
             Schleifenvariablen);
12
13    }
14   }
15  }
16 }
```

Textauszug 5.3: Die Setze Chiffre-Bank Methode in vereinfachtem Quelltext

Die Baumliste ist ein zweidimensionales Listenobjekt, in dem die Baumstruktur der Pfad-Punkte als Arrays festgehalten sind. Dies geschieht durch einen rekursiven Algorithmus, welcher alle Fortschaltungen der Rotoren durchtestet und die validen Pfad-Punkte in die Liste einträgt. Die Methode zum Erstellen des Baumlisten-Objektes wurde ähnlich umgesetzt, wie der Rekursionsalgorithmus zur Berechnung der Rückgabeliste in Abbildung 5.3. Beide Methoden hätten auch zusammengefasst werden können, davon wurde allerdings aus Überlegungen der Performanz und Übersichtlichkeit Abstand genommen.

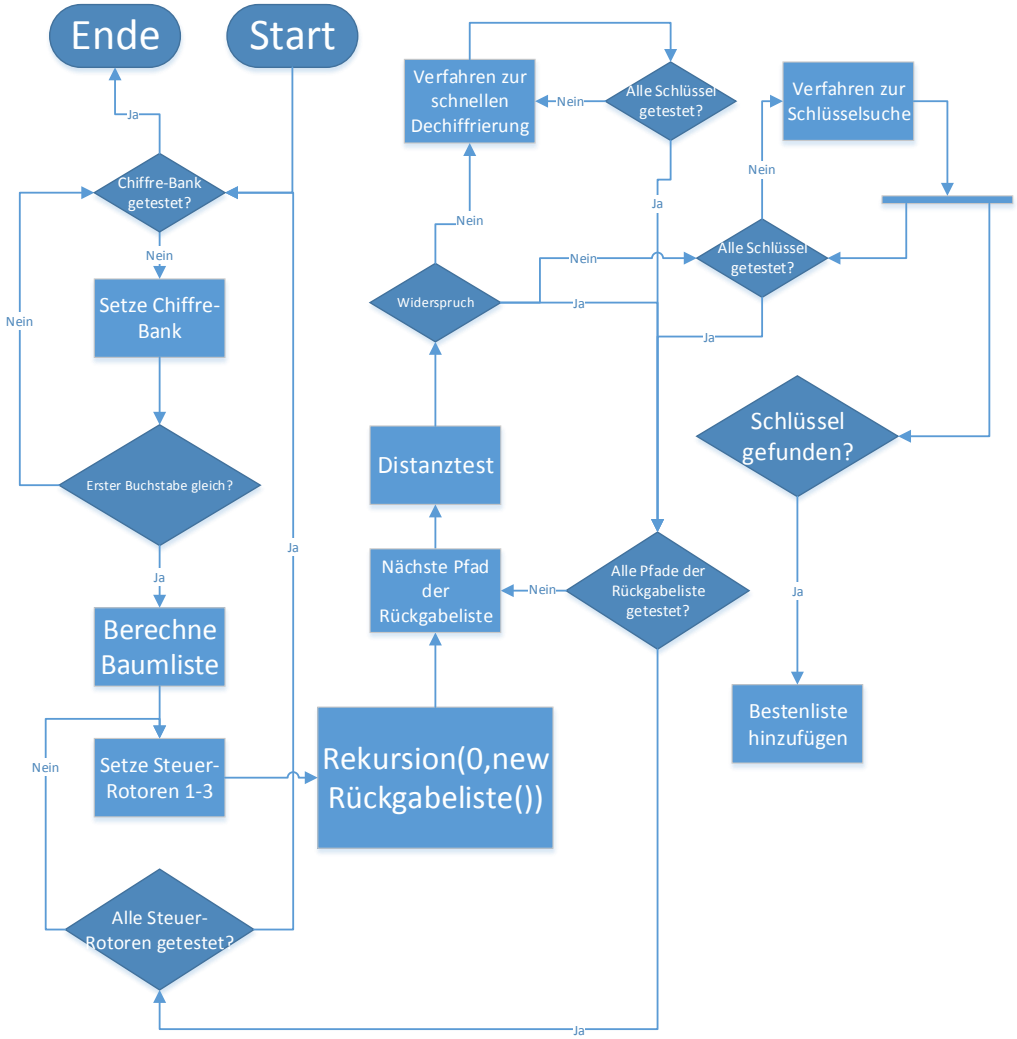

Abbildung 5.2: Darstellung des Algorithmus als Flussdiagramm

Als nächstes werden im Schritt „Setze Steuer-Rotoren 1-3“ die drei Steuer-Rotoren für den Widerspruchsbeweis gesetzt.

Zunächst wird jede Auswahl von drei Steuer-Rotoren aus den fünf übrigen getroffen

und dann alle Kombinationen ihrer Ausrichtungen, Positionen und Stellungen mit Hilfe weiterer Schleifen durchgetestet. Für jede Stellung wird dann im nächsten Schritt der Widerspruchsbeweis durchgeführt. Der Widerspruchsbeweis ist durch einen rekursiven Algorithmus implementiert, wie in Abbildung 5.3 zu sehen ist. Der rekursive Algorithmus „Rekursion“ wird initialisiert mit den Parametern 0 und einer Liste zur Speicherung der Rückgabewerte, der *Rückgabeliste*() welche durch das Schlüsselwort *new* erzeugt wird.

Der Rekursionsalgorithmus erzeugt zunächst eine Schleifenvariable 'j' zum Testen aller Äste, sprich aller Fortschaltungen, der aktuellen Baumtiefe 'i'. Ist j kleiner als die Anzahl der Äste in der Baumtiefe i wird der aktuelle Eintrag durch den Pseudorotor getestet. Wird ein Widerspruch erzeugt, wird die Schleifenvariable j inkrementiert. Bleibt der Test widerspruchsfrei, wird überprüft, ob i der Baumtiefe bzw., der Länge der Baumliste entspricht. Entspricht i der Baumtiefe, ist dieser Pfad widerspruchsfrei geblieben und somit valide. Er wird dann der Rückgabeliste hinzugefügt und der Algorithmus terminiert. Ist i hingegen kleiner als die Baumtiefe, wird durch Selbstaufruf die Rekursionstiefe erhöht und i inkrementiert. Kehrt der Selbstaufruf zurück, terminiert der Algorithmus ebenfalls.

Die Pfade der Rückgabeliste des Rekursionsalgorithmus werden im nächsten Schritt durch den Distanztest validiert. Ergibt sich ein Widerspruch wird der nächste Pfad getestet. Ergibt sich kein Widerspruch im Distanztest, wird auf diesen Pfad das Verfahren zur Schlüsselsuche, oder das Verfahren zur schnellen Dechiffrierung angewendet. Gefundene Schlüssel werden einer Bestenliste hinzugefügt. Wurden alle Schlüssel des Verfahrens zur Schlüsselsuche getestet und alle Pfade der Rückgabeliste getestet, wird die nächste Kombination der Steuer-Rotoren getestet. Wurden alle Kombinationen der Steuer-Rotoren probiert, wird für die Chiffre-Bank eine neue Einstellung gewählt. Sind alle Einstellungen für die Chiffre-Bank getestet worden terminiert dieser Algorithmus.

5.2.2 Widerspruchsbeweis Algorithmus

Der Widerspruchsbeweis lässt sich wie in Textauszug 5.4 durch einen Algorithmus implementieren. Der Pseudorotor ist durch ein zweidimensionales Array repräsentiert worden, dessen Einträge alle null sind. Es werden vier Variablen initialisiert für die 4 Eingänge 'F', 'G', 'H' und 'I', hier tempf, tempg, temph und tempi. Dann findet die Substitution durch die Steuer-Rotoren 1 - 3 statt.

Im nächsten Schritt wird überprüft, ob der Pseudorotor für den Index der Substitution von F noch „leer“, also „==null“ ist. Ist dem so, wird der aktuelle Pfad-Punkt

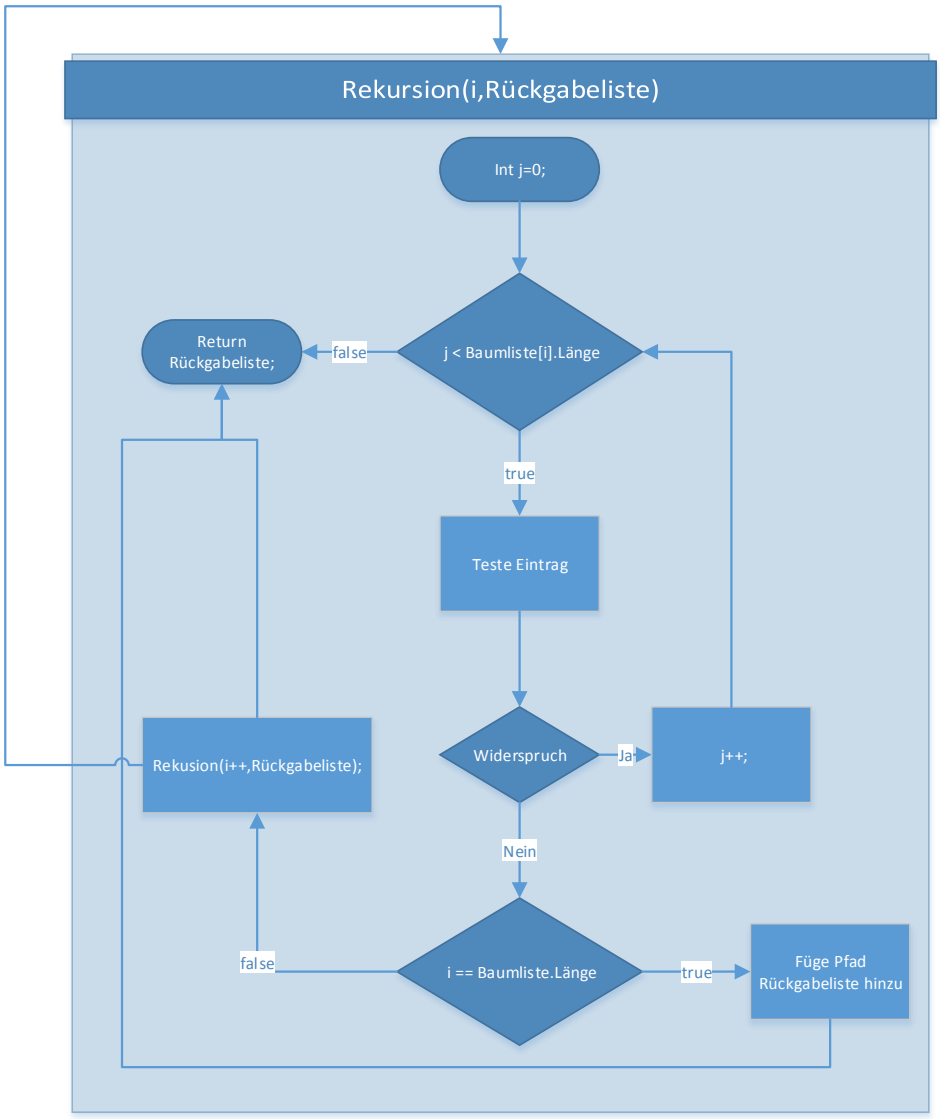

Abbildung 5.3: Darstellung der Funktion Rekursion als Flussdiagramm

dort gespeichert. Andernfalls ergibt sich für diesen Index durch die „ARRAY.Intersect“-Methode die Schnittmenge des Arrays $pseudo[tempf]$ und dem Pfad-Punkt. Ist die Länge dieses Arrays genau gleich 0, wurde der Widerspruchsbeweis erbracht. Dieser Test wird ebenfalls für $tempg$,$temph$ und $tempi$ in jedem Pfad-Punkt durchgeführt.

```
int tempf = '5';
int tempg = '6';
int temph = '7';
int tempi = '8';

for (int i = 1; i < 4; i++)
{
  tempf = Steuer-Rotor[i].[tempf];
  tempg = Steuer-Rotor[i].[tempg];
  temph = Steuer-Rotor[i].[temph];
  tempi = Steuer-Rotor[i].[tempi];
}

  if (pseudo[tempf] == null){
   pseudo[tempf] = Pfad-Punkt;
   }
  else{
     pseudo[tempf] = pseudo[tempf].Intersect(Pfad-Punkt)
        .ToArray();
     if (pseudo[tempf].Count() == 0){
         return false;
       }
    }
    if (pseudo[tempg] == null)
                 (...)
```

Textauszug 5.4: Der Widerspruchsbeweis als Methode in vereinfachtem Quelltext

5.2.3 Verfahren zur Schlüsselsuche Algorithmus

Das Durchtesten der mehrdeutigen Einträge des Pseudorotors und der verbleibenden beiden Steuer-Rotoren geschieht wieder durch eine Schleife.

Wie in Kapitel 4.2 gesehen, lassen sich aus dem Pseudorotor durch Analyse der Häufigkeiten die Paarungen der Gruppen ermitteln. Die 945 möglichen Gruppierungen werden in einer Lookup-Tabelle als Konstante gespeichert, geordnet nach den 89 absoluten Häufigkeiten der Buchstaben.

Der Algorithmus gestaltet sich so, dass zunächst die Häufigkeitsanalyse des Pseudorotors durchgeführt wird. Diese besteht darin, alle Fortschaltungsanweisungen in jedem Eintrag zu zählen. Hat eine Fortschaltungsanweisung eine größere Häufigkeit als 11 oder eine kleinere als 1, wird der nächste potentielle Pseudorotor, getestet.

Die Häufigkeit wird darauf hingehend überprüft, ob eine solche Häufigkeit schon getestet wurde. Da im Folgenden alle Permutationen der Häufigkeit getestet werden, muss jede Häufigkeit nur einmal getestet werden.

Im nächsten Schritt werden in der LookupTabelle alle Sätze von Paarungen ermittelt, welche der Häufigkeit des Pseudorotors entsprechen. Es werden alle Sätze getestet. Für jeden Satz müssen dann auch noch alle Permutationen probiert werden. Der Algorithmus ist in 5.5 noch einmal dargestellt.

```
1  for(alle validen Pseudorotoren p){
2  Häufigkeit h = Häufigkeitsanalyse(p);
3
4  if(!markiereAlsGetestet(h))
5   Index-Labyrinth il= LookUpTabelle(h);
6  else
7   nächsterValiderPseudorotor();
8
9  markierAlsGetestet(h);
10 for(alle Sätze s von il){
11  for(alle Permutationen von s){
12   for(Steuer-Rotor 4){
13    for(Steuer-Rotor 5){
14     Test;
15    }
16   }
17  }
18 }
```

Textauszug 5.5: Methode zum Durchtesten aller mehrdeutigen Einträge im Pseudorotor

Der eigentliche Test geschieht durch den folgenden Quelltext in Textauszug 5.6. Die übergebenen Substitutionen werden durch Steuer-Rotor 4 und 5 weiter substituiert. Das Array Transform macht die Substitution des Stators von 26 Kontakte auf die 9 Restgruppen des Index-Labyrinths, wie in Abbildung 3.6 veranschaulicht. Das Array Index-Labyrinth enthält die Substitutionen der äquivalenten Verdrahtungen des Index-Labyrinths.

Der Test, ob es sich im Endeffekt um den korrekten Schlüssel handelt gestaltet sich in der Form, dass getestet wird, ob der Pfad-Punkt die Substitutionen tempf, tempg, temph und tempi enthält. Ist eines davon nicht der Fall, ist der Test für diese Konfiguration der Maschine gescheitert. Die Steuer-Rotoren konnten in dieser Konfiguration also nicht zur Erzeugung der Chiffre verwendet worden sein. Um Redundanzen zu vermeiden, wird dem

```
int tempf = temp[0];
int tempg = temp[1];
int temph = temp[2];
int tempi = temp[3];

for (int i = 0; i < 2; i++)
{
  tempf = Steuer-Rotor[i].[tempf];
  tempg = Steuer-Rotor[i].[tempg];
  temph = Steuer-Rotor[i].[temph];
  tempi = Steuer-Rotor[i].[tempi];
}

tempf = Transform[tempf];
tempg = Transform[tempg];
temph = Transform[temph];
tempi = Transform[tempi];

tempf = Index-Labyrinth[tempf];
tempg = Index-Labyrinth[tempg];
temph = Index-Labyrinth[temph];
tempi = Index-Labyrinth[tempi];

if (!Pfad-Punkt.Contains(tempf) ||
  !Pfad-Punkt.Contains(tempg) ||
  !Pfad-Punkt.Contains(temph) ||
  !Pfad-Punkt.Contains(tempi))
  break;
```

Textauszug 5.6: Methode zum Testen des Pseudorotors

Verfahren zur Schlüsselsuche nur noch die Substitutionen von 'F', 'G' , 'H' und 'I' durch die ersten Steuer-Rotoren als Array *temp* übergeben, da diese bereits im Distanztest erzeugt wurden.

5.2.4 Verfahren zur schnellen Dechiffrierung

Die Implementierung eines Algorithmus nach dem Verfahren zur schnellen Dechiffrierung gestaltet sich wesentlich simpler, als der des Verfahrens zur Schlüsselsuche. Es müssen lediglich alle Mehrdeutigkeiten des Pseudorotors und den Steuer-Rotoren-4 mit Schleifen durchgetestet werden. Es muss ein neuer Pseudorotor erzeugt werden, welcher den Steuer-Rotor-4 nicht mit einschließt.

Der neu erzeugte Pseudorotor muss für die SIGABA-Komponente geladen werden und der Text von der Komponente entschlüsselt werden. Die Klartexte werden mit der Kostenfunktion bewertet und in die Bestenliste eingetragen.

```
for(Steuer-Rotor 4){
  erzeugePseudorotor();
    for(valide Mehrdeutigkeiten Pseudorotor){
      SIGABA.load(Pseudorotor)
      Schlüssel s = alle Schleifenvarianten +
      die aktuelle Chiffre-Bank Einstellung;
      Klartext k = SIGABA(s, Chiffre);
      Wert w = Kostenfunktion(k);
      if(w > Bestenliste.Last)
      {
        Bestenliste.add(w,k);
      }
  }
 }
}
```

Textauszug 5.7: Das Verfahren zur schnellen Dechiffrierung in vereinfachtem Quelltext

Der Schlüssel ergibt sich aus dem durchgetesteten Steuer-Rotor-4 und dem Schlüssel im Widerspruchsbeweis. Für die durch den Pseuodorotor ersetzten Rotoren, sind keine Einstellungen und auch kein Schlüssel nötig. Der Algorithmus für diesen Test ist in Textauszug 5.7 dargestellt.

5.2.5 Einstellungen

Als Einstellungen kann zwischen dem Verfahren zur schnellen Dechiffrierung und der Schlüsselsuche gewählt werden.

5.2.6 Präsentaion

Die Ergebnisse dieser Komponente werden in ähnlicher Weise dargestellt, wie in der Komponente zur erschöpfenden Schlüsselsuche. Die Komponente verwaltet eine Liste der Ergebnis-Klartexte.

Da bedingt durch die Verfahren nicht der vollständige Schlüssel ermittelt wird, kann auch nicht der vollständige Schlüssel in der Präsentationsliste angezeigt werden. Durch

einen Druck auf eine entsprechende Schaltfläche innerhalb der Präsentation kann der vollständige Schlüssel für diesen speziellen Klartext noch zusätzlich bestimmt werden.

5.3 Vergleich der Komponenten nach Durchsatz

Die Effizienz der Implementierung des Widerspruchsbeweises wird am deutlichsten in dem folgenden Beispiel: Angenommen, die Stellungen, Ausrichtungen und Positionen der Chiffre-Rotoren ist gegeben und über das Fortschaltungs-Labyrinth ist Nichts bekannt. Außerdem stehen 35 Buchstaben an Klartext zur Verfügung. Für diesen Test wurde ein Intel(R) Core(TM) i5 mit 2,3 GHZ verwendet. Die Komponente SIGABA-Schlüsselsucher bräuchte für diesen Schlüsselraum 2^{48} auf dem Testsystem Messungen zu Folge ca. 3.560 Tage. Die Komponente SIGABA-Widerspruchsbeweis braucht für den Widerspruchsbeweis aller Survivor $2^{23} \cdot 10 \approx 2^{26}$ hingegen lediglich ca. 3 Minuten, um alle falschen Stellungen, Ausrichtungen und Positionen der Steuer-Rotoren S1-S3 auszuschließen.

5.4 MysteryTwister C3 - The Crypto Challenge Contest

„MysteryTwister C3 - The Crypto Challenge Contest“ ist ein Online Portal für Kryptografie interessierte Menschen. Hier werden Aufgaben in Form sogenannter Challenges eingestellt, welche dann von den Mitgliedern des Portals gelöst werden können. Diese Aufgaben bestehen in der Analyse verschiedener kryptografischer Verfahren zum Beispiel darin, eine Nachricht zu entschlüsseln oder einen Schlüssel zu finden. Für die Lösung einer Challenge werden Punkte im persönlichen Profil des Mitglieds ausgeschüttet. Über die angeschlossene Forums-Funktion können sich die Mitglieder über die Challenges austauschen und sich gegenseitig helfen. [Ess14b]

Mark Stamp hat in diesem Forum zwei Challenges erstellt, welche mit der SIGABA verschlüsselt wurden. Im Zuge dieser Arbeit wurden diese Challenges versucht mit Hilfe der vorgestellten Verfahren in CrypTool 2.0 zu lösen.

5.4.1 SIGABA - Teil I

In der ersten Challenge sind Teile des Schlüssels gegeben, sowie 7 Buchstaben des Klartextes. Die Teile des Schlüssel sind die Startstellung der Chiffre-Bank: „ABCDE“ und der Steuer-Bank: „ZYXWV“. Desweiteren ist die Auswahl und Reihenfolge der Index-Rotoren 0,1,2,3 und 4, sowie die Startstellung von Index-Rotor I1. Die Verdrahtung der Rotoren ist ebenfalls gegeben. Unklar ist hingegen, welcher Rotor an welcher Position und in welcher Ausrichtung verwendet wurde.

Insgesamt ergibt sich für diese Challenge also ein Schlüsselraum von

$10! \cdot 2^{10} \cdot 133.400 \approx 2^{48}$

unter der Bedingung, dass die Index-Bank, wie üblich, übergangen wird.

Durch die kleine Menge an Klartext und den umfangreichen Informationen über das Fortschaltungs-Labyrinth, lohnt sich das Verfahren zur Schlüsselsuche nicht, da, wie in der Analysephase gesehen, der Schlüsselraum für den Pseudorotor viel zu hoch ist.

Es musste daher eine neue Komponente entwickelt werden. Der Algorithmus dieser Komponente unterscheidet sich hauptsächlich darin, dass statt nur die ersten drei Steuer-Rotoren S1-S3 im Widerspruchsbeweis zu testen, direkt die gesamten Steuer-Bank getestet wird. Da alle anderen Chiffre- und Steuer-Rotoren bereits in Verwendung sind und die Stellung der gesamten Steuer-Bank bekannt ist, bleiben für die zwei zusätzlich zu testenden Rotoren nur noch vier Kombinationen der Ausrichtung übrig. Tritt ein Widerspruch ein, wird die nächste Konfiguration der Steuer-Bank getestet.

Tritt kein Widerspruch auf, was bei so wenig Klartext wahrscheinlich ist, werden alle 113.400 Verdrahtungen der Index-Bank getestet, welche vorher als Array im Speicher abgelegt wurden, um schnell verfügbar zu sein.

Um alle Kerne des Testsystems nutzen können, musste der Schlüsselraum außerdem noch sinnvoll aufgeteilt werden. Der Schlüsselraum wurde anhand des Schlüsselraumes der Konfigurationen der Chiffre-Bank aufgeteilt. So gibt es für $\binom{10}{5} = 252$ Möglichkeiten zum Einsetzen der zehn Chiffre-/Steuer-Rotoren in die Chiffre-Bank. In Abhängigkeit der Anzahl der Kerne, im Fall des zur Verfügung stehenden Testsystems 64, wurden jedem Kern $\frac{252}{64}$ also 3 bis 4 Startstellungen zugeteilt.

Das Testsystem brauchte für diesen Angriff ca. 4-5 Tage.

Zum Vergleich: Unter Verwendung der Komponente erschöpfende Schlüsselsuche, hätte das System Schätzungen zu Folge sechs Wochen gebraucht.

Der Angriff blieb leider ohne Ergebnis. Untersuchungen haben ergeben, dass die Implementierung der SIGABA, welche Stamp zur Erstellung der Challenge benutzt hat, nachweislich fehlerhaft ist[1]. Abgesehen davon, dass die Rotoren substituieren als wären sie rückwärts eingelegt, aber trotzdem dekrementell fortschalten und nicht in der Position 'O' rotieren, sondern nach jeder 14. Schaltung, gibt es weitere Fehler. So werden, sobald ein Rotor rückwärts eingelegt wird, die Substitution der anderen Rotoren scheinbar ignoriert.

Während der Bearbeitung dieser Arbeit konnte das spezielle Verhalten dieser Implementierung auch nicht imitiert werden.

5.4.2 SIGABA - Teil II

Die zweite SIGABA Challenge ist ebenfalls von Stamp in dem Portal erstellt worden. In dieser Challenge sind keine Informationen über den Schlüssel gegeben, dafür ein sehr langer Klartext über 1135 Zeichen. Auf Grund der von Stamp verwendeten Implementierung hat es innerhalb dieser Arbeit keinen Sinn gemacht, den Angriff zu versuchen.

Messungen haben ergeben, dass einer der Kerne im Schnitt ≈ 3 Minuten zum Ausschließen einer einzigen Startstellung braucht, was dem Testen von 300.000 Schlüsseln pro Sekunde entspricht. Für den gesamten Schlüsselraum bedeutet dies einen geschätzten Zeitaufwand von ungefähr

$$\frac{3\,Minuten \cdot 2^{38}}{64} \approx 24.500\,Jahre$$

für das Testsystem. Um diese Challenge mit Hilfe des Verfahrens zur Schlüsselsuche in einem Monat zu lösen, würde eine Rechenkapazität von 200.000 Prozessoren der gleichen Performanz wie der Testrechner gebraucht werden.

[1] `https://www.mysterytwisterc3.org/de/forum/viewtopic.php/f,5/t,46/sid,48cdc3a181223a1c36527ea77e6d909c/start,10/`

6 Zusammenfassung und Ausblick

Die Ergebnisse dieser Arbeit werden hier noch mal nach Zielen geordnet zusammengefasst.

- Umsetzung einer Komponente zur Ver- und Entschlüsselung von Texten nach dem Verschlüsselungsprinzip der SIGABA

  In Kapitel 3 wurde ausführlich dargestellt, wie die SIGABA also Komponente umgesetzt wurde. Dabei wurden bereits Optimierungen des Algorithmus vorgestellt, um die Verschlüsselung, unter Berücksichtigung der Wiederverwendbarkeit der Implementierung durch andere Komponenten, effizienter zu gestalten.

- Entwurf und Umsetzung einer Visualisierung der SIGABA in CrypTool 2.0

  Ebenfalls wurde in Kapitel 3 gezeigt, wie die Verschlüsselung der SIGABA transparent visualisiert werden kann. Die Visualisierung stellt die Substitution jedes Bauteils der SIGABA dar und erhöht durch eine Animation des Prozesses die Nachvollziehbarkeit der Funktionsweise.

- Erstellung der Sicherheitseinschätzung unter Berücksichtigung der Komplexitäten aller vorgestellten Angriffe

  Es kann festgestellt werden, dass die SIGABA Dechiffriermaschine mit einem Schlüsselraum von 2^{96} trotz der schnell voranschreitenden Computertechnik, keinesfalls trivial zu brechen ist. Selbst unter der Verwendung leistungsstarker Computer, ist es heute nicht möglich, mit dem Ansatz zur erschöpfenden Schlüsselsuche eine Nachricht, welche durch die SIGABA verschlüsselt wurde, ohne Wissen über den verwendeten Schlüssel in absehbarer Zeit zu entschlüsseln. Die Suche nach effizienteren Nur-Chiffre-Verfahren wird an dieser Stelle für zukünftige Arbeiten in Aussicht gestellt.

  Die Ergebnisse der Kryptoanalyse haben gezeigt, dass die SIGABA jedoch sehr viel anfälliger gegenüber Angriffen mit bekanntem Klartext ist, als bisher vermutet. In der Kryptoanalyse wurde gezeigt, dass der Schlüsselraum der SIGABA für einen bekannten Klartext einer Länge größer als 30 auf unter $2^{63,1}$ eingeschränkt

werden kann. Im Verlauf der Kryptoanalyse hat sich der Widerspruchsbeweis dabei als zentrales Werkzeug für den Angriff herausgestellt. Desweiteren konnte gezeigt werden, dass durch den Widerspruchsbeweis wesentlich weniger Klartext nötig ist, als bei vergleichbaren Ansätzen. Und das bei einer absoluten Erfolgswahrscheinlichkeit von 1,0. Die These, zur Effizienzsteigerung des Angriffs diverse Rotoren durch einen Pseudorotor zu ersetzen, konnte dabei ebenfalls hinreichend belegt werden. Im Vergleich zu dem von Chan vorgeschlagenen Angriff konnte gezeigt werden, dass die starre Trennung von Phase I und II jedoch nicht effizient ist. Die Untersuchungen haben aber auch gezeigt, dass der Widerspruchsbeweis, bei kleinen Mengen an Klartext, keine Lösung darstellt und unpraktikabel wird.

- Implementierung eines Angriffs auf die SIGABA als Komponente in CrypTool 2.0

  Es wurde in Kapitel 5 gezeigt, wie die verschiedenen Angriffe als Komponenten für CrypTool 2.0 umgesetzt wurden.

  Ausführlich wurde dargestellt wie der Angriff zur erschöpfenden Schlüsselsuche, im Allgemeinen wie im Speziellen, als Algorithmus umgesetzt werden kann.

  Weiterhin zeigte sich, dass der Pseudorotor als Datenstruktur implementiert werden kann und sich im Algorithmus des Widerspruchsbeweises verwenden lässt.

  Das Testen der Implementierung macht dabei deutlich, wie stark das Durchsuchen des Fortschaltungs-Labyrinths durch die Verwendung des Widerspruchsbeweises im Vergleich zur erschöpfenden Schlüsselsuche beschleunigt werden kann.

- MysteryTwister C3 SIGABA Challenge Teil I und II

  In Kapitel 5 wurde gezeigt, dass Teil I der Challenge mit den vorgestellten Modifizierungen der Komponente innerhalb einer Woche lösbar wäre.

  Aus den Überlegungen zum zweiten Teil der Challenge geht hervor, dass die Komponente nicht effizient genug ist, um diese in einem privaten Kontext lösen zu können.

6.1 Ausblick

Zukünftige Arbeiten müssen sich vor allem auf die Steigerung der Performanz der Implementierungen richten, zum Beispiel durch die Verwendung von anderen Programmiersprachen wie C/C++, die Implementierung des Algorithmus auf einer Grafikkarte, oder eine effizientere Verteilung des Algorithmus auf mehrere Recheneinheiten.

Weiterhin haben Stichproben ergeben, dass die Survivor über den Schlüsselraum nicht gleich verteilt sind. Es konnte festgestellt werden, dass es sehr viele Stellungen ohne einen einzigen Survivor und dafür wenig Stellungen mit überproportional vielen gibt. Zukünftige Arbeiten müssen zum Gegenstand ihrer Untersuchungen machen, ob dies Rückschlüsse auf die korrekte Startstellung der Chiffre-Bank zulässt. Eine Vorabsortierung der zu testenden Schlüssel, oder aber auch ein Angriff nach dem Hill-Climbing-Verfahren erscheinen hier realistisch. Für den Moment wird hier das meiste Potential zur Effizienzsteigerung gesehen.

Literaturverzeichnis

[Aig04] Aigner, Martin: *Diskrete Mathematik*. Aufbaukurs Mathematik. Vieweg Verlag, März 2004.

[BB06] Behnke, Joachim und Nathalie Behnke: *Die Binomialverteilung*. Grundlagen der statistischen Datenanalyse: Eine Einführung für Politikwissenschaftler, Seiten 229–247, 2006.

[Bec05] Beckh, Joachim: *Blitz & Anker*, Band 2. BoD–Books on Demand, 2005.

[Beu94] Beutelspacher, Albrecht: *Cryptology*. MAA, 1994.

[Cha07] Chan, Wing: *Cryptanalysis of SIGABA*. Technischer Bericht, San Jose State University, 2007.

[Ess14a] Esslinger, Bernhard: *CrypTool 2.0*. `https://www.cryptool.org/de/cryptool2`, 2014. [Online; zugegriffen am 1. März 2014].

[Ess14b] Esslinger, Bernhard: *MysteryTwister C3 - The Crypto Challenge Contest*. `https://www.mysterytwisterc3.org`, 2014. [Online; zugegriffen am 1. März 2014].

[Goe12] Goebel, Greg: *US Codebreakers In World War II*. `http://www.vectorsite.net/ttcode_07.html`, November 2012. [Online; zugegriffen am 1. März 2014].

[Heb21] Hebern, Edward H: *Electric coding machine*. Patent, March 1921. United States Patent 1510441 A.

[KR11] Knudsen, Lars R und Matthew JB Robshaw: *Brute force attacks*. In: *The Block Cipher Companion*, Seiten 95–108. Springer, 2011.

[Kun10] Kunze, Simone: *Das Sammelbilderproblem*. Technischer Bericht, Universität Potsdam, 2010.

[Lee03] Lee, Michael: *Cryptanalysis of the SIGABA*. Technischer Bericht, San Jose State University, 2003.

[May03] Mayr, Thomas: *Das Enigma System*. 2003.

[Mic14a] Microsoft: *Visual C#*. `http://msdn.microsoft.com/de-de/library/kx37x362.aspx`, Februar 2014. [Online; zugegriffen am 1. März 2014].

[Mic14b] Microsoft: *Windows Presentation Foundation*. `http://msdn.microsoft.com/de-de/library/ms754130(v=vs.110).aspx`, Februar 2014. [Online; zugegriffen am 1. März 2014].

[Pek10] Pekelney, Richard: *Electronic Cipher Machine (ECM) Mark II*. `http://www.maritime.org/tech/ecm2.htm`, 2010. [Online; zugegriffen am 1. März 2014].

[Red44] Redman, Joseph R.: *OPERATING INSTRUCTIONS FOR ECM MARK 2 AND CCM MARK 1*. Historic Naval Ships Association, may 1944.

[Sha49] Shannon, Claude E: *Communication Theory of Secrecy Systems**. Bell system technical journal, 28(4):656–715, 1949.

[SKQ01] Schindler, Werner, François Koeune und Jean-Jacques Quisquater: *Improving divide and conquer attacks against cryptosystems by better error detection/correction strategies*. In: *Cryptography and Coding*, Seiten 245–267. Springer, 2001.

[SOC07] Stamp, Mark und Wing On Chan: *SIGABA: Cryptanalysis of the Full Keyspace*. Cryptologia, 31(3):201–222, 2007.

[SSD44] Safford, Laurance F. (Washington DC), Seiler und Donald W. (Washington DC): *Control circuits for electric coding machines*. Patent, December 1944. United States Patent 6175625.

[Wac13] Wacker, Arno: *Applied Cryptology*. Vorlesung, 2013.