UNIVERSITÄT KASSEL
FACHBEREICH ELEKTROTECHNIK/INFORMATIK
FACHGEBIET ANGEWANDTE INFORMATIONSSICHERHEIT

Masterarbeit

Implementierung und Kryptoanalyse der Typex

Olaf Ralph Harland
Matrikelnummer: 31202204

UNIKASSEL
VERSITÄT
Fachgebiet Angewandte Informationssicherheit
Fachbereich Elektrotechnik/Informatik
Universität Kassel

10. Oktober 2017

Prüfer:
Prof. Dr. Arno Wacker
Prof. Dr. Sebastian Petersen
Betreuer:
M. Sc. Nils Kopal

Inhaltsverzeichnis

1 Einleitung **1**
- 1.1 Motivation . . . 1
- 1.2 Aufgabenstellung . . . 2
- 1.3 Ziele der Arbeit . . . 3
- 1.4 Aufbau der Arbeit . . . 4

2 Grundlagen **5**
- 2.1 Kryptographie . . . 6
  - 2.1.1 Substitution und Transposition . . . 8
  - 2.1.2 Symmetrische und Asymmetrische Schlüssel . . . 12
  - 2.1.3 Block- und Stromchiffre . . . 13
- 2.2 Rotor-Chiffriermaschinen . . . 13
- 2.3 Kryptoanalyse . . . 16
  - 2.3.1 Brute-Force Angriff . . . 17
  - 2.3.2 Ciphertext-Only Angriff . . . 18
  - 2.3.3 Known-Plaintext Angriff . . . 19
  - 2.3.4 Chosen-Plaintext Angriff . . . 20
  - 2.3.5 Heuristische Verfahren . . . 21

3 Typex **23**
- 3.1 Geschichte . . . 25
  - 3.1.1 Die Royal Air Force Enigma . . . 26
  - 3.1.2 Die Typex Produktion . . . 27
- 3.2 Funktionsweise . . . 29
  - 3.2.1 Rotoren . . . 29
  - 3.2.2 Druckeinheit . . . 30
  - 3.2.3 Modell-Reihe . . . 31
  - 3.2.4 Vergleich mit Enigma . . . 34
- 3.3 Schlüssel . . . 38
  - 3.3.1 Schlüsselraumgröße . . . 39
  - 3.3.2 Unizitätslänge . . . 41

4 Kryptoanalyse **43**
- 4.1 Schwächen . . . 43
- 4.2 Angriffe . . . 45
  - 4.2.1 Brute-Force Angriff . . . 46
  - 4.2.2 Ciphertext-Only Angriff . . . 46
  - 4.2.3 Known-Plaintext Angriff . . . 52

4.2.4 Hill-Climbing . . . 57
4.2.5 Analyse der Rotor-Verdrahtung nach Rejewski . . . 58
4.3 Statistische Analyse . . . 61

5 Design und Implementierung 67
5.1 Der Typex-Simulator . . . 70
5.1.1 Klasse – Typex . . . 72
5.1.2 Klasse – Program (Typex-Simulator) . . . 76
5.2 Das Typex-Analysewerkzeug . . . 78
5.2.1 Projekt – Kryptoanalyse . . . 78
5.2.2 Projekt – Evaluation . . . 94

6 Evaluation 99
6.1 Bewertung der Analysekomponente . . . 99
6.1.1 Metriken . . . 100
6.1.2 Testumgebungen . . . 103
6.1.3 Messungen . . . 104
6.2 Rätsel von Mystery Twister sowie Challenges vom FB 16 . . . 108

7 Zusammenfassung und Ausblick 113
7.1 Zusammenfassung . . . 113
7.1.1 (R-1) Implementierung der Typex-Chiffriermaschine als Simulator . . . 113
7.1.2 (R-2) Analyse, Konzeption, Entwicklung und Implementierung von Kryptoanalysewerkzeugen für die Analyse der Typex 114
7.1.3 (R-3) Evaluation der kryptographischen Stärke der Typex sowie Evaluation der entwickelten Kryptoanalyseverfahren . 115
7.1.4 (R-4) Vergleich der kryptographischen Stärke mit der deutschen Enigma . . . 115
7.1.5 (R-5) Lösen von Rätseln aus Mystery Twister (MTC3) sowie Brechen vom FB16 vorgegebene Geheimtexte . . . 116
7.2 Ausblick . . . 116

Literaturverzeichnis 119

Abbildungsverzeichnis

2.1 Allgemeines Kryptosystem 7
2.2 Funtionsschema – Rotor-Chiffriermaschinen 15
2.3 Allgemeines Kryptosystem unter einem Ciphertext-Only Angriff . . 18
2.4 Allgemeines Kryptosystem unter einem Known-Plaintext Angriff . . 19
2.5 Allgemeines Kryptosystem unter einem Known-Plaintext Angriff . . 20
2.6 Buchstabenhäufigkeit bei englischer und deutscher Sprache 21

3.1 Typex Ursprung 24
3.2 Ungefähre Chronologie der Maschinen-Entwicklung 24
3.3 Typex-Chiffriermaschine mit Druckfunktion 25
3.4 Rotorachse mit zwei Rotoren der Typex 27
3.5 Explosionsdiagramm der Rotorachse 28
3.6 Explosionsdiagramm – Montage der Rotorachse 28
3.7 Verzahnter Rotor mit drei Kerben entnommen 30

4.1 Menu Graph mit Zyklen 54
4.2 Menu Graph mit geschlossenen Zyklen 55
4.3 Buchstabenverteilung bei Entschlüsselung 62

5.1 Verschachelte Projekte 67
5.2 Ablaufdiagramm – Ciphertext-Only Angriffs 68
5.3 Klassendiagramm vom entwickelten Programm 69
5.4 Klassendiagramm des Projektes – Typex-Simulator 70
5.5 Klassendiagramm der Klasse – Typex 73
5.6 Klassendiagramm des Projektes – Kryptoanalyse 79
5.7 Klassendiagramm der Klasse – Kryptoanalyse (Analysekomponente) 80
5.8 Interval-Heap 83
5.9 Klassendiagramm der Klasse – DecodeRotors/DecodeStators (Anayl-sekomponente) 85
5.10 Parallelverarbeitung bei der Analysekomponente 87
5.11 Klassendiagramm der Klasse – DecodeRingsSequently (Anaylsekom-ponente) 90
5.12 Klassendiagramm der Klasse – FrequencyAnalysis (Anaylsekompo-nente) 91
5.13 Klassendiagramm der Klasse – FrequencyAnalysis (Anaylsekompo-nente) 94
5.14 Klassendiagramm des Projektes – Evaluation 95

6.1 Verteilung der Rotoranzahl 100

6.2 Angriffsgraph 101
6.3 Ausführungsdauer/Textlänge (1 und 2 Umkehrwalzen) – Server . . 106
6.4 Ausführungsdauer/Textlänge (1 und 2 Umkehrwalzen) – Desktop . 107
6.5 Erfolgsrate/Textlänge (1 und 2 Umkehrwalzen) – Server 108
6.6 Erfolgsrate/Textlänge (1 und 2 Umkehrwalzen) – Desktop 109

Tabellenverzeichnis

2.1 Permutation – Lat. Alphabet 9
2.2 Transpositions-Verschlüsselung 11
2.3 durchschnittliche Angriffsdauer nach Schlüsselraumgröße 18

3.1 Typex Modelle 32
3.2 Stecker-Kombinationen 37

4.1 Phase 1 – Schlüsselraum von den Rotoren 47
4.2 Phase 1 – Schlüsselraum von Statoren 47
4.3 Ergebnis nach Phase 1 49
4.4 Fortschritt bei schnellen Rotor 51
4.5 Fortschritt bei mittleren Rotor 51
4.6 Fortschritt bei langsamen Rotor 52
4.7 Graphenerstellung mit Klar-/Geheimtext (crib) 54
4.8 Menu-Ergebnis von Crib 56
4.9 Allgemeine Syntax der Variablendefinition 62
4.10 Koinzidenzindex-Wert für natürliche Sprachen 63
4.11 Sinkov Statistik – „DES“ 64
4.12 Wahrscheinlichkeiten für bestimmte Tetragramme 65

5.1 Rotorenverdrahtung bei Enigma I, M3, M4 71
5.2 Programmkonstanten des Simulators 72
5.3 Programmkonstanten und Ausführungsparameter für das Projekt Typex-Simulator 77
5.4 Array Index über die ganzen Zahlen $\mathbb{Z}$ 84
5.5 Messungen über un-/korrekte Schlüssel 89
5.6 Messungen über un-/korrekte Schlüssel ohne Statoren 89
5.7 Metrik bei Ringanalyse 90
5.8 Programmkonstanten und Ausführungsparameter für das Projekt Kryptoanalyse 93
5.9 Spalteninformation der Evaluation-Ausgabedatei 96
5.10 Programmkonstanten und Ausführungsparameter für das Projekt Evaluation 97

6.1 Pfad durch eine Levenshtein-Matrix 102
6.2 Testumgebungen 104

1 Einleitung

Diese Arbeit behandelt die Typex, eine Verschlüsselungsmaschine des britischen Militärs, die unter anderem im Zweiten Weltkrieg, eingesetzt wurde. Die Typex ist eine Rotor-Chiffriermaschine die der deutschen Chiffriermaschine Enigma nachempfunden ist. Das britische Militär nutzte die Typex nicht nur für die Verschlüsselung der eigenen Kommunikation, sondern auch für die Entschlüsselung der mit der Enigma verschlüsselten Nachrichten. Sie erreichte eine beträchtliche Stärke für die Geheimhaltung und war resistent gegenüber Angriffen. Sie galt als sicher während ihrer Einsatzjahre. Eine genaue Analyse der Typex soll diese angenommene Sicherheit widerlegen. Es wird daher im Rahmen dieser Arbeit ein Programm programmiert, welches die Vor- und Nachteile der Typex-Maschine hervorhebt.

1.1 Motivation

Klassische Verschlüsselungssysteme wurden in der Geschichte erfolgreich eingesetzt, wobei deren Einsatz heutzutage nicht mehr denkbar wäre. Zum einem kann eine Chiffriermaschine mit der heutigen Rechenleistung gelöst werden. Zum anderen gibt es Verfahren, die über mathematische Annahmen den Schlüsselraum dermaßen reduzieren, dass eine Schlüsselsuche innerhalb kurzer Zeit möglich ist. Nichtsdestotrotz kann eine Analyse nutzbare Informationen hervorbringen, die zur Lösung ausstehender Probleme beitragen können.

Der polnische Mathematiker Marian Rejewski legte die Grundlage zur Kryptoanalyse der deutschen Enigma [1]. Der britische Mathematiker Alan Turing war maßgeblich an der Entschlüsselung deutscher Funksprüche der Enigma während des Zweiten Weltkriegs beteiligt [2]. Dies beruhte mitunter auf spezifischen Klartexten, die in dem Geheimtext vermutet wurden. Später führte der amerikanische Informatiker James J. Gillogly einen reinen Geheimtextangriff auf die Enigma durch [3]. Abgesehen davon existieren noch immer ungelöste Geheimnachrichten, die mit einer Enigma verschlüsselt wurden. Das Lösen solcher Nachrichten hilft neue Erkenntnisse über bisher unbekannte historische Ereignisse zu erhalten.

Die Analyse von klassischen Chiffren kann zum besseren Verständnis der verwendeten Algorithmen und zur Entwicklung sicherer Systeme beitragen. Des Weiteren kann das Verhalten von statistischen Methoden beim Lösen einer Chiffre untersucht werden. Dies kann zu einer präziseren Konstruktion solcher Methoden führen. Der Fokus liegt im zuverlässigen Erkennen einer natürlichen Sprache. Die

gewonnenen Erkenntnisse aus dieser Arbeit können zukünftige Forschungen über ungelöste Chiffren, dieser oder verwandter Art, behilflich sein.

Es wurde noch kein Angriff gestartet, in der eine Typex nur über den bekannten Geheimtext lösbar war. Dieser Herausforderung haben sich Kryptoanalytiker bereits bei historischen Chiffriermaschinen, unter anderem bei der Enigma, gewidmet. Ferner wurde der Typex das Übersenden hochgradig sensibler Nachrichten anvertraut. Dies verdeutlicht die Gewissheit, dass die Typex zur damaligen Zeit sicher nicht lösbar sei. Diese angenommene Sicherheit bekräftigte die Motivation, sich mit der Typex zu beschäftigen. Es sollte möglich sein, Geheimtexte über eine bestimmte Länge mit einer akzeptablen Erfolgswahrscheinlichkeit entschlüsseln zu können.

1.2 Aufgabenstellung

Die ausgehändigte Aufgabenstellung wird im Folgendem wörtlich übernommen: *Die Typex ist eine britische Rotor-Chiffriermaschine, ähnlich der deutschen Chiffriermaschine Enigma. Sie wurde von den britischen Streitkräften während des Zweiten Weltkriegs sowohl für die Verschlüsselung der eigenen Kommunikation als auch für die Entschlüsselung deutscher, mittels der Enigma verschlüsselter, Nachrichten eingesetzt. Die einer Schreibmaschine ähnelnde Typex-Maschine besteht aus mehreren sogenannten Walzen. Innerhalb jeder dieser Walzen befindet sich eine eindeutige Verdrahtung, welche einen Stromfluss über alle Walzen hinweg, ermöglicht. Drückt man eine Taste auf der Tastatur, so drehen sich die Walzen unterschiedlich weiter und leiten den Strom auch entsprechend unterschiedlich weiter. Der Geheimtextbuchstabe leuchtet am Ende der Verkabelung entsprechend auf. Bei jedem Tastendruck wird der gleiche Buchstabe auf einen anderen Buchstaben verschlüsselt. Somit ermöglicht die Typex eine polyalphabetische Verschlüsselung. Wie die Enigma besitzt auch die Typex ein Steckerbrett. Allerdings ermöglicht das Typex Steckerbrett, im Gegensatz zu dem der Enigma, eine nicht involutorische Substitution einzelner Buchstaben. So ist es z.B. möglich, ein A durch ein B, ein B durch ein C und ein C durch ein A zu substituieren (zu „Steckern“). Dies erhöht die kryptographische Sicherheit der Maschine erheblich. Des Weiteren wurden zwischen 10, 120 und 240 unterschiedliche Rotore eingesetzt. Jeder dieser Rotore konnte sowohl „vorwärts“, als auch „rückwärts“ in die Maschine gesetzt werden. Jeder Rotor besaß fünf, sieben oder neun Fortschaltkerben. Außerdem gab es zwei nicht fortgeschaltete Ein-/Ausgangsstatoren die ebenfalls eingestellt werden konnten.*

Hauptgegenstand dieser Masterarbeit ist es, Analyseverfahren für die Kryptoanalyse der Typex Chiffriermaschine in zu entwickeln. Dafür muss zunächst die Chiffriermaschine als Simulator implementiert werden. Danach soll eine Analyse der Typex auf Grundlage der Kryptoanalyse der Enigma Chiffriermaschine, siehe [Gillogly etc], erfolgen. Hierfür soll eine Analysekomponente implementiert werden.

Ein weiteres Ziel der Arbeit ist es, die kryptographische Stärke der Typex abzuschätzen. Hierfür muss die Größe des Schlüsselraums, die Unizitätslänge der Chiffre, die Komplexität (z.B. Zeit und benötigte Länge des Geheimtextes) der entwickelten Angriffe sowie deren Mächtigkeit evaluiert werden.

Die Arbeit ist vollständig am Fachgebiet für Angewandte Informationssicherheit durchzuführen. Alle innerhalb der Arbeit getroffenen (Design-) Entscheidungen sollen in einer Ausarbeitung (der Masterarbeit) diskutiert werden. Neben der Ausarbeitung der Arbeit ist der Quellcode umfangreich zu dokumentieren. Alle in der Arbeit angefertigten Dokumente sowie der erstellte Quellcode sind in dem Fachgebiet einzureichen. Abschließend ist eine Präsentation der in der Arbeit erreichten Ziele innerhalb des „Master-Kolloquiums“ vorzubereiten und durchzuführen.

1.3 Ziele der Arbeit

Ziele dieser Masterarbeit sind zum einem die Recherche von Informationen über das Funktionsprinzip der Typex und zum anderen ein Konzept für den programmiertechnischen Teil zu erstellen. Die schriftliche Arbeit soll die Hintergründe zur Entwicklung, die Anforderungen und die Baueigenschaften im Bezug auf die kryptographische Stärke der Typex erläutern. Des Weiteren soll eine Analyse über die Schwächen und gewisse Angriffspunkte ausgearbeitet werden. Ein Programmteil soll die Simulation der Ver- und Entschlüsselung mit bestimmten Konfigurationen ermöglichen. Der zweite Programmteil wird zur Kryptoanalyse der Typex entwickelt. Die Analyse soll die Schlüsselsuche eines Geheimtextes mit dem Wissen der Rotoren *oder* Statoren durchführen können. Die Konzeption der Schlüsselsuche erfolgt mit Hilfe der gesammelten Informationen und mit einer Abschätzung der Komplexität bei Angriffen auf die Typex. Statistische Analysen zeigten, dass die Chiffriermaschinen, die es zu brechen galt, in viele Teilprobleme zerlegt wurden („divide and conquer“-Verfahren) [1–3]. Ausgehend davon soll das Programm folgende Anforderungen erfüllen:

(R-1) Implementierung der Typex-Chiffriermaschine als Simulator.

(R-2) Analyse, Konzeption, Entwicklung und Implementierung von Kryptoanalysewerkzeugen für die Analyse der Typex

(R-3) Evaluation der kryptographischen Stärke der Typex sowie Evaluation der entwickelten Kryptoanalyseverfahren

(R-4) Vergleich der kryptographischen Stärke mit der deutschen Enigma

(R-5) Lösen der MTC3 Known-Plaintext-Challenge sowie Brechen der vom Betreuer vorgegebenen Geheimtexte (Ciphertext-Only)

Die Komplikationen und Entschlüsse sollen in der Arbeit erwähnt werden. Die Lösungen sollen im adäquaten Umfang erläutert werden. Die Informationen werden aus historischer sowie aktueller Literatur bezogen. Die historische Beschreibung der

Typex befindet sich in einem eigens gewidmeten Kapitel, wobei sich die aktuelle Literatur über die ganze Arbeit sich streckt.

1.4 Aufbau der Arbeit

Die Masterarbeit ist in sieben Kapitel unterteilt. Im Kapitel 2 werden allgemeine Grundlagen der Kryptographie und benötigte Fachbegriffe im Zusammenhang mit den Chiffriermaschinen vermittelt. Kapitel 3 widmet sich dem historischen Verlauf der ersten Erwähnung bis zum Einsatz der Typex. Des Weiteren werden das Funktionsprinzip und die Eigenschaften der Baueinheiten erläutert, welche auf der eigenen Literaturrecherche basieren. Die unterschiedlichen Konzepte für die Kryptoanalyse sind in Kapitel 4 erklärt. Es zeigt die Angriffsmöglichkeiten und die Schwächen, die zur Reduzierung der kryptographischen Stärke führten. Vielmehr werden die statistischen Metriken zur Ermittlung eines Ergebnisses demonstriert. Die technische Implementierung erfolgt in Kapitel 5. Dort wird die programmiertechnische Umsetzung der Kryptoanalyse veranschaulicht. Die Erklärungen verwenden die Begrifflichkeiten und Strukturen aus dem erstellen Quellcode. Kapitel 6 konzentriert sich auf die Bewertung der Messungen. Die Ergebnisse der Kryptoanalyse werden evaluiert und in Summe reflektiert. Zusätzlich werden die Methoden zur Distanzermittlung zwischen den Ergebnissen verdeutlicht. Der erreichte Forschungsstand wird in Kapitel 7 gezeigt. Des Weiteren werden weitere Ideen für zukünftige Verbesserungen vorgestellt.

2 Grundlagen

Die Kryptologie ist die Wissenschaft der Ver- und Entschlüsselung (Kryptographie) von Texten. Sie dient der Geheimhaltung des Inhalts und dem Schutz vor unautorisiertem Entschlüsseln (Kryptoanalyse). Die Kryptographie beschäftigt sich mit der Verschlüsselung von Nachrichten, während die Kryptoanalyse die Entschlüsselung, ohne Kenntnis des Schlüssels, beabsichtigt [4]. Die Chiffre (Kryptosystem) benötigt einen geheimen Schlüssel, womit der Klartext verschlüsselt werden kann. Der Geheimtext erreicht den Empfänger über einen Kommunikationskanal und wird mit einem korrekten Schlüssel zu lesbarem Text entschlüsselt. Die Texte (Nachrichten) bestehen aus Zeichen über einem Alphabet. Die Kryptoanalyse platziert zur unautorisierten Entschlüsselung einen „Angriff“ auf das Kryptosystem [4]. Ein Kryptoanalytiker ist bemüht, auf professionelle und wissenschaftliche Weise eine Chiffre zu brechen (lösen).

Geheime Nachrichten von Regierungen konnten bis 1840 einfach abgefangen werden, denn die Post musste nur heimlich aus den Posttaschen geklaut werden. Somit ist die Geheimhaltung der Nachrichten mehr von der Sicherheit dieser Beutel abhängig gewesen, als von der Stärke der Chiffre. Im Telegraphen-Zeitalters konnte der Verkehr weiterhin leicht abgefangen werden und führte zwangsweise zu einer Verbesserung der Chiffresysteme. Die Telegraphie per Funk etablierte sich. Sie war schneller und flexibler, als die bisher verwendenden Kabelleitungen. Allerdings konnte der Datenverkehr ohne Mühe abgefangen werden. Die Regierungen verbesserten die traditionellen Chiffresysteme und entwickelten zur gleichen Zeit neue Chiffren, wie die Chiffriermaschinen [5].

Diese Mittel hielten die Kryptographie mit der Kryptoanalyse gleichauf. Allerdings war die Entwicklung durch die Regierungen mit der Tatsache gebremst worden, dass die Einfachheit der Kommunikation, wichtiger sei, als eine hohe Sicherheit der Nachrichten. Sogar Behörden statuierten, dass eigene Nachrichten schnell und korrekt zu erhalten, wichtiger sei, als die Verschleierung der Informationen vor dem Feind [5]. Eine komplexe Chiffre kann sehr sicher sein, wird aber auch die Übertragung von den Nachrichten verlangsamen. Sie kann sich zwar durch einen großen Schlüsselraum selbst verteidigen, führte aber auch zu höherer Fehlerrate bei der Übertragung. Es ist ein Kompromiss zwischen Sicherheit und Benutzbarkeit, sofern sie sensible Nachrichten über offene Kommunikationskanäle schicken. Kleinste Änderungen an einem Kryptosystem können sowohl die Geheimhaltung als auch die Leichtigkeit der Kommunikation beeinträchtigen. Chiffriermaschinen boten ein Mittel, um diese Gefahr zu reduzieren und gleichzeitig vorgenannte Anforderungen zu verbessern. Obwohl solche Maschinen für die Kabelkommunikation

wertvoll waren, sind sie im Wesentlichen aus den Anforderungen an die Chiffrestärke entstanden. Die britischen Entwicklungen weiterer Chiffren resultierte aus der Unzufriedenheit mit den bisher genutzten Verfahren. Leider unterschätzte Großbritannien immer wieder das Ausmaß der Telegraphie und die Erfordernis von neuen Instanzen bei der Sicherheit.

Großbritannien war trotz dessen unter den Ersten, die den potenziellen Wert von Chiffriermaschinen erkannten. Die britischen Kampfeinheiten interessierten sich für diese Technik, als sie anfingen, diese moderne Kommunikationsmethode in ganz anderen Dimensionen zu nutzen. Sie wussten, dass die Telegraphie-Kommunikation abhörbar war und dass „ein Experte, der über genügend Material und ausreichend Zeit verfügt"jede Chiffre knacken kann. Daher entschieden sie sich die Risiken zu minimieren, indem der Einsatz von Telegrafie per Funk nur in Ausnahmefällen erfolgte und Chiffriersysteme ständig wechselten [5].

In den folgenden Sektionen werden die Kryptographie und die Kryptoanalyse im Allgemeinen erklärt. Die Kryptographie und deren mathematische Eigenschaften werden in die Kategorien Substitution und Transposition, symmetrisch und asymmetrisch, Strom und Blockchiffre aufgeteilt. Die möglichen Angriffe eines Kryptoanalytikers sind im darauffolgenden Abschnitt unter dem Begriff „Kryptoanalyse“ erläutert. Diese Themenfelder sollen ein Fundament für die Rotormaschinen und deren Verletzbarkeit schaffen. Im Fokus steht die Wissensvermittlung von kryptografischen Techniken und funktionalen Eigenschaften der Typex-Rotormaschine.

2.1 Kryptographie

Kryptographie ist die Wissenschaft der Verschlüsselung von Informationen. Die Verschleierung von Informationen erfolgt mit einer Chiffre, auch kryptographisches System (Kryptosystem) genannt. Diese werden für den Datenschutz und die Authentifizierung in Kommunikationssystemen verwendet. Eine Chiffre benutzt einen Algorithmus für die Ver-/Entschlüsselung (siehe Formel 2.1 und 2.2). Ein Verschlüsselung-Algorithmus $encrypt()$ chiffriert den Klartext P mit einem Schlüssel K und überführt ihn in unverständliche Chiffretexte bzw. Geheimtexte C [6]. Der Geheimtext wird über einen unsicheren Kommunikationskanal, in der Regel einen öffentlichen Kanal, an den Empfänger gesendet. Es wird ein Entschlüsselung-Algorithmus $decrypt()$ für die Entschlüsselung verwendet, um die ursprünglichen Daten wiederherzustellen.

$$C = encrypt(P, K) \tag{2.1}$$

$$P = decrypt(C, K) \tag{2.2}$$

Dieser Ablauf ist in Abbildung 2.1 gezeigt. Die Geheimhaltung der Nachricht ist nur vom Schlüssel abhängig. Ein unautorisierter Empfänger kann das komplette Kryptosystem zwar kennen, nichtsdestotrotz bleibt ihm das Lesen der Nachricht

verwehrt, da er keinen passenden Schlüssel vorweisen kann [4]. Die Schlüssel bei Chiffrierung und Dechiffrierung müssen nicht identisch sein (symmetrische/asymmetrische Verfahren).

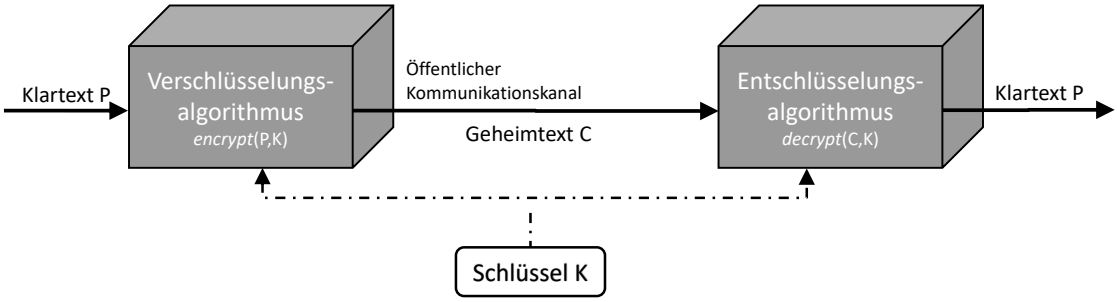


Abbildung 2.1: Allgemeines Kryptosystem

Der Schlüsselraum $\mathbb{K}$ definiert sich über die Menge aller möglichen Schlüssel K. Die Schlüsselraumgröße $|\mathbb{K}|$ ist die Anzahl aller möglichen Schlüssel aus $\mathbb{K}$. Ein Klartext P besteht aus einer begrenzten Menge von Zeichen, das über das Alphabet $\mathbb{A}_P$ definiert ist. Analog hierzu ist das Alphabet $\mathbb{A}_C$ über der endlichen Menge von Zeichen, die im Geheimtext C vorkommen, festgelegt. Die Zeichen im Alphabet $\mathbb{A}_P$ und $\mathbb{A}_C$ können aus Zahlen, Buchstaben oder Symbolen bestehen. Ein Schlüssel $K \in \mathbb{K}$ kann eine Zeichenfolge aus einer natürlichen Sprache sein oder gar eine Maschinenkonfiguration im Binärsystem repräsentieren.

Kryptographische Systeme sind allgemein in drei unabhängige Dimensionen klassifiziert [7]:

(a) Verwendete Operationsart für die Umwandlung von Klartext in Chiffretext. Alle Verschlüsselungsalgorithmen basieren auf zwei allgemeinen Prinzipien:

  (i) Substitution: jedes Element eines Klartextes (bit, Buchstaben, Bitfolgen beschreiben Buchstaben) wird auf ein anderes Element abgebildet

  (ii) Transposition: indem die Elemente im Geheimtext neu angeordnet werden

(b) Die Anzahl der genutzten Schlüssel:

  (i) Symmetrisches Kryptosystem, indem Sender und Empfänger denselben Schlüssel benutzen

  (ii) Asymmetrisches Kryptosystem, indem Sender und Empfänger unterschiedliche Schlüssel verwenden

(c) Die Art und Weise, in welches der Klartext verarbeitet wird:

  (i) Eine Blockchiffre verarbeitet jeweils einen Block der Eingabe und erzeugt jeweils einen passenden Ausgabeblock dazu.

  (ii) Die Stromchiffre verarbeitet kontinuierlich die Eingabe. Ein Element der Eingabe produziert also ein Element der Ausgabe.

Die genannten Punkte sind als Oberbegriffe innerhalb der Kryptographie zu verstehen und dienen zum besseren Verständnis der folgenden Kapitel. Die Operationen, Schlüssel und Verarbeitungen werden einleitend erklärt und in ihren Bestandteilen aufgelistet. Die Grundlagen aller Kryptosysteme werden vermittelt. Diese werden für nachfolgende Kapitel der Rotor-Chiffriermaschine im Zusammenhang mit deren Einsatzgebieten benötigt. Die Grundlagenkapitel legen ihren Fokus auf die Typex-Maschine und decken die Thematik von historischen Chiffren ab.

2.1.1 Substitution und Transposition

Substitution (Ersetzen) und Transposition (Vertauschen) sind unterschiedliche Operationen, die eine Chiffre auf einem gegebenen Text durchführt. Innerhalb der Substitution wird weiter zwischen der monoalphabetischen und polyalphabetischen Substitution unterschieden. Deren Ein- und Ausgabe ist entweder über ein konkretes Alphabet oder über mehrere Alphabete definiert.

2.1.1.1 Substitution

Beim Substitutionsverfahren wird jedes Zeichen des Klartextes durch ein Anderes ersetzt. Dieses Verfahren unterteilt sich in monoalphabetische und polyalphabetische Substitution, die anhand historischer Chiffren erklärt wird.

Monoalphabetische Substitution ist die einfachste Substitutions-Chiffre. Sie ersetzt die Zeichen eines Klartextes durch jeweils einzelne neue Zeichen. Diese hängen von Zeichen oder Gruppen aus dem Klartext ab. Es wird nur ein einziges fixes Alphabet zur Verschlüsselung/Substitution verwendet. Dieses Verfahren greift auf die Caesar-Chiffre zurück.

Im Allgemeinen lässt sich die Abbildungsfunktion nach Formel 2.3 definieren.

$$\begin{aligned} encrypt(P,K) &: \mathbb{A}_P \to \mathbb{A}_C \\ decrypt(C,K) &: \mathbb{A}_C \to \mathbb{A}_P \end{aligned} \tag{2.3}$$

Mit der von Caesar implementierten Funktion werden den Alphabeten die 26 Buchstaben des lateinischen Alphabets auf 25 Zahlen abgebildet, also $\mathbb{A}_P = \mathbb{A}_C = \{A = 0, \ldots, Z = 25\}$. Als Schlüssel dient eine Zahl $1 < K < |\mathbb{K}|$. Sie ver-/entschlüsselt jeden Buchstaben i des Klartextes P_i oder Geheimtextes C_i durch die Funktion 2.4.

$$\begin{aligned} encrypt(P_i,K) &= (P_i + K) \mod 26 \\ decrypt(C_i,K) &= (C_i - K) \mod 26 \end{aligned} \tag{2.4}$$

Nicht desto trotz gewährt diese Chiffre keine Sicherheit und ist einfach zu brechen. Es müssen nur höchstens 25 verschiedene Möglichkeiten ($K = \{1, \cdots, 25\}$)

durchprobiert werden [8]. Eine Häufigkeitsanalyse lässt sich nach Friedmann ideal anwenden [9]. Die Analyse zählt das Auftreten der einzelnen Buchstaben in $\mathbb{A}_C$ und vergleicht diese Werte mit den Häufigkeiten der natürlichen Sprache. Sofern der Kryptoanalytiker die verwendete Sprache weiß, kann er solch einen Angriff platzieren. In der deutschen Sprache ist das „E“ der Häufigste $(17,40\%)$ aller Buchstaben [8]. Der häufigste Buchstabe im Geheimtext lässt sich mit großer Wahrscheinlichkeit dem „E“ zuordnen.

Anstatt der gleichbleibenden Verschiebung des Textes kann auch eine individuelle Verschiebung auf jedem Buchstaben angewendet werden. Dies wird durch eine beliebige Permutation der Buchstaben des Alphabets erreicht und gewährt eine höhere Sicherheit. Hierzu wird eine bijektive Abbildung auf den Zeichen des verwendeten Alphabets definiert, gezeigt bei Permutation $\mathbb{A}_C$ *-Zuf.* in Tabelle 2.1. Eine Bijektion beschreibt eine Funktion, inder jedem Element aus $\mathbb{A}_P$ genau ein Element aus $\mathbb{A}_C$ zugeordnet ist.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
$\mathbb{A}_P$	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
$\mathbb{A}_C$ *-Zuf.*	M	X	L	E	R	P	N	S	V	U	B	W	D	G	J	T	C	Z	Q	A	F	I	H	K	Y	O
$\mathbb{A}_C$*-Wort*	T	Y	P	E	X	M	A	S	C	H	I	N	B	D	F	G	J	K	L	O	Q	R	U	V	W	Z

Tabelle 2.1: Permutation auf dem lateinischen Alphabet mit zufälliger und schlüsselabhängiger Anordnung

Damit die Permutation und somit der Schlüssel einfacher strukturiert wird, kann ein Schlüsselwort in die Abbildungsvorschrift aufgenommen werden. Dies ermöglicht es, bei verschiedenen Gelegenheiten nach dem gleichen Verfahren verschlüsseln zu können. In Tabelle 2.1 beginnt die Permutation $\mathbb{A}_C$*-Wort* mit dem Schlüsselwort „TYPEXMASCHINE“. Dabei sind alle mehrfachen Buchstaben nicht aufgelistet und die leeren Felder wurden mit den restlichen Buchstaben aufgefüllt [4].

Der Schlüsselraum lässt sich bei der monoalphabetischen Substitution über die Fakultät nach Formel 2.5 berechnen.

$$|\mathbb{K}_{Mono-Substitution}| = \prod_{i=1}^{n} i = n! \tag{2.5}$$

Über einem Alphabet $\mathbb{A}_P = \mathbb{A}_C$ mit $n = 26$ Buchstaben besteht bei der ersten Zuweisung eine Auswahl von 26 Buchstaben. Bei der zweiten Zuweisung sind es 25 $(= 26-1)$ Möglichkeiten und bei der Letzten nur noch *eine* $(= 26-25)$ Möglichkeit. Dieser Ablauf lässt sich mit dem Ausdruck $(n-1)\cdot(n-2)\cdot\dots\cdot 1 = n!$ erklären.

Polyalphabetische Substitution verwendet mehrere Verschlüsselungsalphabete. Im Generellen bestimmt der Schlüssel, welches Alphabet verwendet wird. Dabei wird jedes Klartextzeichen um unterschiedlich viele Zeichen verschoben. Die Häufigkeitsverteilung der Buchstaben ist dadurch viel gleichmäßiger, da ein Buchstabe aus $\mathbb{A}_P$ auf mehrere Buchstaben in $\mathbb{A}_{C,j}$ substituiert wird [8]. Dabei sind die Buchstaben P_i des Klartextes P durch die Buchstaben $C_{i,j}$ aus verschiedenen Alphabeten $\mathbb{A}_{C,j}$ ersetzt. Der Zähler i indiziert den Buchstaben im Klartext. Der Index j ($j <= n$) zeigt das zu verwendende Alphabet unter Berücksichtigung des Schlüssels an. n ist die Länge des Schlüsselwortes. Die Alphabete lassen sich gemäß dem Schlüssel in einer Tabelle nach fester Reihenfolge wiederholt anordnen. Dieses Prinzip ist im historischen Vigenère-Chiffre realisiert, wobei deren Alphabete $\mathbb{A}_{C,j}$ über die 26 lateinischen Buchstaben definiert sind. Die Vigenère-Chiffre ist lediglich eine Generalisierung der Cäsar-Chiffre, da nicht nur ein Schlüssel K (monoalphabetisch), sondern auch eine Schlüsselperiode von $K_1, K_2, \ldots, K_n$ (polyalphabetisch) benutzt wird. Für $n = 3$ ist eine Schlüsselperiode als $K_1, K_2, K_3, K_1, K_2, \cdots, K_3$ festgelegt.

Die Schlüsselraumgröße lässt sich für Alphabete mit jeweils 26 Buchstaben (Formel 2.6) berechnen [10]. Die Formel multipliziert lediglich n-Mal die Größe der Alphabete $\mathbb{A}_{C,j}$.

$$|\mathbb{K}_{Poly-Substitution}| = 26 \cdot 26 \cdot \cdots \cdot 26 = \prod_{i=1}^{n} 26 = 26^n \tag{2.6}$$

Die Angriffe auf solch ein Kryptosystem sind schwieriger durchzuführen, da eine einfache Häufigkeitsanalyse nicht zielführend wäre und komplexere Statistiken benötigt. Hilfreich ist zu wissen, wie viele Alphabete verwendet wurden. Der Kryptoanalytiker kann bestimmte wiederkehrende Kombinationen von Buchstaben erkennen und so zunächst auf die Anzahl der verwendenden Alphabete schließen. Sobald die Periode n identifiziert ist, kann eine Häufigkeitsanalyse, analog zum Angriff bei einer monoalphabetischen Substitution, durchgeführt werden. Jeweils der i-Buchstabe des Geheimtextes wird einer Häufigkeitsanalyse unterzogen. Dieses Verfahren heißt Kasiski-Test und sucht im Geheimtext nach wiederholenden Zeichenfolgen und vermutet, dass deren Distanz ein Vielfaches der Schlüsselwortlänge ist [8].

Die Schwachstelle der polyalphabetischen und besonders der monoalphabetischen Chiffrierung ist die Häufigkeitsverteilung. Deshalb sollte die Verteilung besser verschleiert werden, indem jedem Klartextbuchstaben mehrere Geheimtextzeichen zugeordnet sind. Die Geheimzeichen werden im Verhältnis zur natürlichen Häufigkeit des jeweiligen Buchstabens zugewiesen. Der Buchstabe „E“ wird die meisten Abbildungen aufweisen, da dieser am Häufigsten auftritt. Sobald eine Gleichverteilung im Geheimtext vorliegt, ist eine Häufigkeitsanalyse nicht einsetzbar. Solch ein System ist eine homophone Chiffre [4]. Die Schlüsselraumgröße der homophonen Chiffre lässt sich über Formel 2.7 berechnen [10].

$$|\mathbb{K}_{Homophone}| = \binom{n}{i} \cdot i! \cdot i^{n-1} \tag{2.7}$$

Die Variable n ist die Anzahl der Geheimtextzeichen und i ist die Zeichenanzahl des Klartextes.

2.1.1.2 Transposition

Transposition-Chiffre verschlüsseln den Inhalt eines Klartexts durch die Umstellung (Permutation) der einzelnen Zeichen. Die Buchstaben im Klartext werden lediglich in ihren Positionen vertauscht (umsortiert). Die Transposition ist zusammen mit der Substitution einer der ältesten bekannten Kryptosysteme. Die Skytale von Sparta verwendete solch ein Verfahren. Dort wurde ein Papier um ein Stab gewickelt und anschließend senkrecht mit der Nachricht beschriftet. Die Nachricht konnte erst gelesen werden, wenn ein Stab von gleichem Umfang verwendet wurde. Daher ist der Schlüssel K über den Umfang des Stabs definiert [4, 11].

Des Weiteren gibt es eine Transpositons-Chiffre, die den Text in mehrere Zeilen schreibt und die Spalten nach der Schlüsselreihenfolge vertauscht. Meist wird eine Zahlenfolge oder ein Schlüsselwort verwendet. Entweder werden die Buchstaben in den äquivalenten Zahlenpartner umgerechnet oder der Schlüssel besteht schlichtweg aus Buchstaben. Das Schlüsselwort „Typex“ wird in die Zahlenfolge „3,5,2,1,4“ überführt. Der Schlüssel wird über die Matrix geschrieben und bestimmt deren Spaltenanzahl. Der Klartext wird horizontal in jede Reihe geschrieben. Befinden sich leere Zellen in der Transpositionstabelle, spricht man von einer irregulären Spaltentransposition. Ist dies nicht der Fall, dann wird es als komplette Spaltentransposition bezeichnet. Die letzte Zeile kann mit Klartext oder Füllzeichen vervollständigt werden. Wichtig ist, dass diese Füllzeichen nicht besonders seltene Zeichen sind, da dies einen Angriff eines Kryptoanalytikers erleichtern würde [11]. Nach Tabelle 2.2 wird der Textkandidat „KryptoanalyseTypex“ zu „PYKTRANOLATEYYSTXPYE“ transformiert. Die grauen Zellen kennzeichnen die Füllzeichen.

Schlüssel	3	5	2	1	4
Matrix	K	R	Y	P	T
	O	A	N	A	L
	Y	S	E	T	Y
	P	E	X	K	R

→

1	2	3	4	5
P	Y	K	T	R
A	N	O	L	A
T	E	Y	Y	S
K	X	P	R	E

Tabelle 2.2: Transpositions-Verschlüsselung vor und nach dem Vertauschen

Mit den bisher bekannten Verfahren wird eine Häufigkeitsanalyse durchgeführt. Diese gibt eine Tendenz aus, ob eine Transposition-Chiffre vorliegt. Aus der Analyse bestimmter Zeichenfolgen ist dann die Reihenfolge und die Zahl der Spalten

leicht zu ermitteln. Das Wortpaar „CH“ aus dem Wort „Maschine“ lässt sich leicht detektieren. Auf Basis ähnlicher Annahmen aus den natürlichen Sprachen kann der Kryptoanalytiker die korrekte Reihenfolge erkunden und somit die Chiffre brechen. Diese Informationen im Chiffretext sind leicht ableitbar.

Die Schlüsselraumgröße einer Spaltentransposition wird mit der Formel 2.8 berechnet [10]. Über dem Schlüsselwort oder der maximalen Klartextlänge ist die Variable n definiert. Vor der Chiffrierung existieren n-Möglichkeiten das Zeichen im Geheimtext zu platzieren. Diese Möglichkeit reduziert sich um eine Stelle pro Verschlüsselungsschritt, bis der Klartext durchlaufen ist.

$$|\mathbb{K}_{Transposition}| = n \cdot (n-1) \cdot \ldots \cdot 1 = \prod_{i=1}^{n} i = n! \tag{2.8}$$

2.1.2 Symmetrische und Asymmetrische Schlüssel

Die bisherigen Chiffren (Skytale, Cäsar und Vigenère) basieren alle auf dem symmetrischen Schlüsselsystem. Der Unterschied zwischen symmetrischen und asymmetrischen Systemen ist lediglich die Schlüsselanzahl. Dieser gravierende Unterschied hat Auswirkungen auf die zugrundeliegenden Konzepte.

Symmetrische Verfahren verwenden sowohl für das Verschlüsseln als auch für das Entschlüsseln den gleichen Schlüssel. Der Nachteil am symmetrischen Kryptosystem ist der sichere Austausch des Schlüssels mit allen autorisierten Empfängern. Ein Kryptoanalytiker kann mit dem Schlüssel nicht nur die Daten lesen, sondern auch manipulierte Fehldaten an die Teilnehmer schicken. Die symmetrische Kryptographie zeichnet eine deutlich bessere Performance gegenüber den asymmetrischen System aus. Daher wird das Verfahren besonders bei der Verschlüsselung großer Datenmengen verwendet. Damit m Teilnehmer in einem Kommunikationsnetz sicher miteinander kommunizieren können, wird jedem Teilnehmerpaar ein Schlüssel ausgehändigt. Die Formel 2.9 berechnet, wie viele Schlüssel ausgetauscht werden.

$$\frac{m(m-1)}{2} \tag{2.9}$$

Asymmetrische Verfahren veranlassen, dass jeder Kommunikationspartner ein Schlüsselpaar besitzt. Es besteht aus einem persönlichen und einem geheimen Schlüssel. Die Verschlüsselung erfolgt mit dem öffentlichen Schlüssel und steht jedem Teilnehmer ohne vorherige Anfrage zur Verfügung. Im Gegensatz zum symmetrischen Verfahren müssen bei asymmetrischen Verfahren keine geheimen Schlüssel ausgetauscht werden, da der private Schlüssel nur der Person gehört, die ihn generiert hat. Die asymmetrischen Verfahren finden Anwendung bei modernen Verschlüsselungen und weniger bei historischen Chiffren.

2.1.3 Block- und Stromchiffre

Es wird zwischen Blockchiffren und Stromchiffren unterschieden. Unter einer **Blockchiffre** wird ein Verschlüsselungsverfahren verstanden, welches die Datenblöcke mit einer festen Länge verarbeitet und verschlüsselt. In der Regel weicht die Länge eines Textes von der erwarteten Textblocklänge ab. Deshalb wird durch das Verfahren entweder eine bestimmte Anzahl an Leerdaten an die Datenblöcke angehängt oder die Größe mit einer Fragmentierung der zu kurzen Datenblöcke reduziert.

Eine **Stromchiffre** ist meist über die kleinste darstellbare Dateneinheit (Byte, Zeichen, Wort) formatiert. Die Verschlüsselung besteht aus der sequentiellen Verschlüsselung der einzelnen Dateneinheiten. Des Weiteren können die späteren Verschlüsselungen von den vorhergehenden Daten abhängen. Bei diesem Verfahren werden keine Leerdaten benötigt, da sämtliche relevanten Daten in einzelne Dateneinheiten fragmentiert sind.

Das Kryptosystem ist deutlich angreifbarer, wenn die vorherige Verschlüsselung nicht von der aktuellen Verschlüsselung abhängt. Ohne diese Eigenschaft werden gleiche Dateneinheiten in denselben Geheimtext abgebildet. Des Weiteren können Geheimtexte mutwillig manipuliert werden, indem Fragmente aus abgehörten Daten gezielt ersetzt werden. Beabsichtigte Änderungen sind nur schwierig feststellbar.

Auch bei abhängigen Verschlüsselungsverfahren können solche Angriffsvarianten erfolgreich sein. Daher befindet sich meist eine redundante Information im Geheimtext. Nach der Entschlüsselung kann geprüft werden, ob die vorliegende Änderung autorisiert oder unbefugt durchgeführt wurde. Die Information kann ein Datum, eine Uhrzeit oder sogar ein inkrementierender Zähler sein, welche aber synchron auf allen Systemen installiert ist.

2.2 Rotor-Chiffriermaschinen

Eine Chiffriermaschine ist ein elektromechanisches Gerät, dessen kryptographische Stärke sogenannte Rotoren bilden. Ein Rotor ist ein Zylinder und ist auf beiden Seiten mit jeweils 26 Kontakten versehen. Jeder linke Kontakt ist mit einem Kontakt auf der rechten Seite paarweise elektronisch verbunden. Das Verdrahtungskonzept entspricht einer Permutation über einem Alphabet und bietet daher 26! Möglichkeiten zur Wahl. Das Prinzip der Rotor-Chiffriermaschine wurde in der Enigma implementiert, wobei hier 3 bis 4 Permutationen stattfanden.

Die Rotoren sind drehbar einstellbar und können 26 Drehpositionen einnehmen. Eine bestimmte Anzahl von Rotoren wird axial angeordnet, so dass sich jeweils die Kontakte gegenüberliegender Seiten berühren. Die linke abschließende Seite ist von einer Umkehrwalze (Reflektor) besetzt. Die Umkehrwalze hat nur auf der

rechten, dem Gehäuse zugewandten Seite, 26 Kontakte hat. Diese Kontakte sind paarweise untereinander verbunden. Aufgrund der speziellen Verdrahtung wirkt dieser Rotor als Umkehrwalze.

Der Eintritt in das System kann über weitere Statoren erfolgen, die im rechten Teil des Gehäuses angeordnet sind. Ein Stator ist baugleich mit einem Rotor, nur dass sich der Stator nicht dreht und statisch in seiner Position verbleibt. Der Stator hat auch 26 Kontakte auf der linken Seite. Diese liegen elektronisch beim Rotorsatz an. Die rechten 26 Kontakte sind mit einer Tastatur verbunden. Die 26 Tasten wirken als Schalter, wobei jede Taste genau einem Buchstaben des Alphabets zugeordnet ist. Bei einem Tastendruck wird ein Stromkreis geschlossen. Der Stromkreis ist über die innere Verdrahtung der Rotoren, Umkehrwalze, Statoren und ihre Stellung zueinander definiert. Der Austrittspunkt am rechten Stator gibt das Ergebnis einer Chiffrierung aus. Der Geheimtext-Buchstabe wurde bei der Enigma mit einer Glühlampe in einem Lampenfeld angezeigt. Dieser Verlauf ist in Abbildung 2.2 zu erkennen. Bei jedem Tastendruck wird die Position der Rotoren durch eine Rotation um eine bzw. zwei Positionen verändert. Nach spätestens 26 Schritten (= eine volle Umdrehung) eines Rotors dreht sich dessen linker Nachbar um einen Schritt vor. Der mittlere Rotor veranlasst die Drehung des äußeren Nachbarn. Diese Drehungen werden durch einen regulären Mechanismus gesteuert [8, 11].

Jede mögliche Stellung der Rotoren zueinander bzw. zur Umkehrwalze sowie Ein-/Ausgangsstatoren bewirkt eine Permutation des Alphabets. Ein Rotor führt im Allgemeinen eine monoalphabetische Substitution durch. Im Verbund aller Permutationen wird eine polyalphabetische Substitution dargestellt [12]. Im Folgenden werden die Permutationen je Rotor R_j bezeichnet, wobei die j-Stelle die Position des Rotors bestimmt. Des Weiteren werden die Permutationen bei den Statoren als S_l benannt, wobei l die jeweilige Position festlegt. Beim äußerst rechten Stator fließt der Strom in den Mechanismus hinein. Der Strom durchläuft die Statoren $S_1, \cdots, S_l$ und die Rotoren $R_1, \cdots, R_j$ von rechts nach links. Die Umkehrwalze U (Reflektor) bewirkt das nochmalige Durchlaufen des Walzensatzes von links nach rechts. Der Strom durchläuft zuerst die Rotoren $R_j, \cdots, R_1$ und danach die Statoren $S_l, \cdots, S_1$.

Die Verschlüsselungsfunktion einer allgemeinen Chiffriermaschine lässt sich über den Stromkreis in Formel 2.10 überführen.

$$\begin{aligned} decrypt_{allg.Maschine}(K,P) = S_1^{-1}\ S_2^{-1}\ \ldots\ S_l^{-1}\ R_1^{-1}\ R_2^{-1}\ \ldots\ R_j^{-1} \\ U\ R_j\ \ldots\ R_2\ R_1\ S_l\ \ldots\ S_2\ S_1\ (K,P) \end{aligned} \tag{2.10}$$

Der äußerst linke Rotor R_1 ist der Erste und endet bei den Statoren S_j im rechten Bereich. Alle Permutationen der Rotoren und Statoren befinden sich in der Menge $\mathbb{P}$. Bei der Enigma wurden drei Rotoren und kein Stator ausgetauscht. Daher ist $|\mathbb{P}| = 3$ und $1 \leq j + l \leq |\mathbb{P}|$ mit $j = 3$ und $l = 0$. Die Umkehrwalze wird als U bezeichnet.

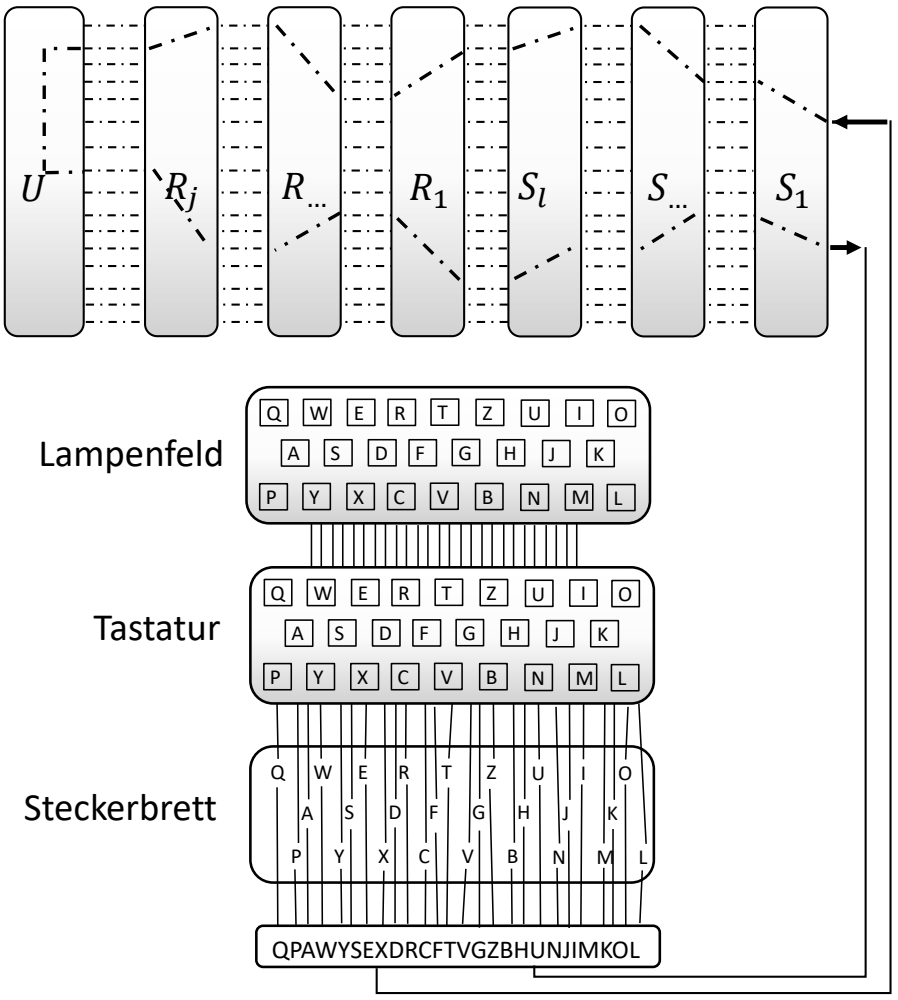


Abbildung 2.2: Allgemeines Funktionsschema einer Rotor-Chiffriermaschinen

Die Entschlüsselung erfolgt, indem der Chiffretext mit der gleichen Anfangsstellung erneut eingegeben wird. Jeder Zustand der Maschine hängt nur von einem Anfangszustand und der Zahl der vorausgegangenen Verschlüsselungen ab. Der Anfangszustand ist frei wählbar, sowohl durch Auswahl und Anordnung der Rotoren als auch durch Einstellung der Rotoren auf eine bestimmte Grundstellung.

Die Zuordnung der Kontakte eines Rotors zu den Buchstaben des Alphabets ist durch einen beschrifteten, drehbaren Ring festgelegt. Diese 26 möglichen Stellungen des Alphabetrings sind weitere frei wählbare Parameter. Durch sie wird der relative Zeitpunkt der Rotorbewegung bestimmt. Eine weitere Besonderheit bei einigen Chiffriermaschinen ist die Verwendung von frei wählbaren Statoren. So können zusätzliche Permutationen des Alphabets jeweils vor Eintritt und nach Austritt des Stromes aus der Maschine bewirkt werden.

Die **Enigma** verfügte zusätzlich über ein Steckerbrett *Sb*, welche das paarweise Vertauschen der Buchstaben vornahm. Bevor der Strom in das Rotorsystem eintrat, wurde der Strom durch eine Steckerverbindung mit einem anderen Buch-

staben vertauscht. Wenn kein Kabel gesteckt war, gelangte der Strom direkt zur ersten Permutation im Rotorsystem [13].

Die Enigma verhielt sich, bis auf die feste Anzahl von Rotoren, äquivalent zur allgemeinen Chiffriermaschine. Die drei Rotoren (R_1, R_2, R_3) und die Umkehrwalze U waren bei der Enigma austauschbar. Die Eintrittswalze (Stator) war nicht herausnehmbar. Durch die feste Position war die Eintrittswalze nicht Bestandteil des Schlüssels. Somit ist die Verdrahtung der Eintrittswalze dem Kryptoanalytiker bekannt (Kerckhoffs'sche Prinzip). Die Verschlüsselungsfunktion der Enigma lässt sich über den Stromkreis in Formel 2.11 definieren.

$$decrypt_{Enigma}(K,P) = \; Sb^{-1}\; R_1^{-1}\; R_2^{-1}\; R_3^{-1}\; U\; R_3\; R_2\; R_1\; Sb\; (K,P) \tag{2.11}$$

2.3 Kryptoanalyse

Dieser Abschnitt steht unter dem Gesichtspunkt der Kryptoanalyse und ihren Methoden. Die Kryptoanalyse ist die Wissenschaft des unautorisierten Entschlüsselns oder viel mehr als das Brechen von kryptographischen Verfahren bekannt. Sie ist genauso wichtig, wie die Kryptographie, denn ohne die Kryptographie wäre auch die Kryptoanalyse nicht notwendig.

Die Kryptoanalyse sorgt dafür, dass immer neue kryptographische Methoden entwickelt werden und sich somit die Kryptographie analog dazu weiter verändert. Des Weiteren sind kryptoanalytische Methoden notwendig, um zu überprüfen, wie sicher ein kryptographisches System ist, und um mögliche Schwächen zu korrigieren. Die verschiedenen Methoden in der Kryptoanalyse zur Berechnung eines Kryptosystems werden auch als Angriffe bezeichnet [4]. Einige dieser Angriffe sind allgemein, andere wiederum werden nur auf bestimmte kryptographische Verfahren angewendet. Durch bekannte Schlüsselteile kann ein Angriff schneller durchgeführt werden. Diese Informationen kann der Kryptoanalytiker durch das Abhören des Kommunikationskanals erhalten. Als Abhören bezeichnet man die Überwachung von Nachrichten durch einen Dritten, die durch Überwachung eines Kommunikationskanals erzielt wird. Der Kryptoanalytiker ist bestrebt, entweder einen neuen Geheimtext zu entschlüsseln oder im Idealfall den Schlüssel zu finden. Dafür können ihm unterschiedlich viele Informationen zur Verfügung stehen.

Kryptoanalytische Angriffe Eine Chiffre kann mit einem kryptoanalytische Angriff gebrochen werden. Unter anderem wurde dies für zahlreiche mono- und polyalphabetische Substitutionen gezeigt [4]. Die Wahl der richtigen Angriffsmethode hängt wesentlich von den Ressourcen und Fähigkeiten des Kryptoanalytikers ab. Diese Methoden lassen sich im Generellen in Ciphertext-Only, Known-Plaintext und Chosen-Plaintext Angriffe aufteilen. Dabei unterscheiden sich alle nur in der Vorabinformation über ein Kryptosystem. Der schlechteste Fall tritt ein,

wenn der Kryptoanalytiker nur abgehorchten Geheimtext auszuwerten hat (Brute-Force/Ciphertext-Only). Eventuell kann noch das Kryptosystem und die zugrundeliegende Sprache bekannt sein. Ein Kryptosystem, dass solch einem Angriff nicht stand hält, ist völlig unbrauchbar. Bei einem Known-Plaintext Angriff kennt der Kryptoanalytiker Texte mit korrespondierendem Klar- und Geheimtext.

Verfügt der Analytiker bei einem beliebigen Klartext über jeden Schlüssel mit dem zugehörigen Chiffretext, hat er zweifellos die meisten Informationen. Dies nennt sich ein Chosen-Plaintext Angriff. Er kann das System brechen, gar manipulieren. Die vorgenannten Angriffsarten werden in den folgenden Kapiteln allgemein eingeführt.

Dem Kryptoanalytiker ist der Ver- und Entschlüsselungsalgorithmus bei allen Angriffsarten bekannt. Diesen Informationen über einen Geheimtext werden bei den beschriebenen Angriffsarten in dieser Arbeit grundsätzlich als „angenommen“ vorausgesetzt. Das **Kerckhoffs'sche Prinzip** besagt, dass die Sicherheit eines Kryptosystems nur vom Schlüssel abhängig sein darf [4]. Wenn der Algorithmus geheim gehalten wird, ist die Sicherheit mit dem Schlüssel und dem Algorithmus verknüpft. Diese Verknüpfung verstößt gegen das Kerckhoffs'sche Prinzip.

2.3.1 Brute-Force Angriff

Der Kryptoanalytiker wählt seine Angriffsmethode in Abhängigkeit zu dem Verschlüsselungsschemas und den verfügbaren Informationen. Die einfachste Methode ist die vollständige Schlüsselsuche (engl.: „exhaustive key search“) ohne jeglichen Zugriff auf zusätzliche Informationen [7]. Dieser Angriff lässt sich immer auf jeden Verschlüsselungsalgorithmus anwenden. Es gibt keine spezielle Kryptovariante zur Verhinderung dieses Angriffs. Die Entwickler eines Kryptosystems zielen darauf ab, dass es keine bessere Angriffsvariante gibt [14].

Der Angriff durchsucht den ganzen Schlüsselraum nach dem korrekten Schlüssel. Die durchschnittliche Dauer für einen solchen Angriff auf ein System mit unterschiedlicher Schlüsselraumgrößen ist in Tabelle 2.3 aufgeführt. Die Schlüsselräume werden in Bits b angegeben. Es wurden moderne Schlüsselverfahren als Beispiel gewählt. Aufgelistet ist der DES mit 56 Bit [15], der Triple-DES mit 168 Bit [16], der AES mit 128 Bit [17] und ein Beispiel mit 32 Bit. Des Weiteren ist eine Substitutions-Chiffre über 26 Zeichen gezeigt, wobei jede der 26 Permutationen als Schlüssel dienen. Die Anzahl der unterschiedlichen Schlüssel wird mit 2^b berechnet. Im ungünstigsten Fall ist der korrekte Schlüssel an der letzten Stelle im Schlüsselraum und im durchschnittlichen Fall ist er bei der Hälfte zu erwarten. Die Schlüsselanzahl wird zur Ermittlung der Durchschnittswerte mit 2^{b-1} halbiert. Zuerst ist die Dauer mit 1 Entschlüsselung pro Mikrosekunde (μs) angegeben, dass 1.000.000 Entschlüsselungen pro Sekunde s entspricht. In der letzten Spalte ist die Dauer mit 10^{12} Entschlüsselungen pro s angegeben.

Schlüssel-raum (bits)	Anzahl unterschiedlicher Schlüssel	∅ Dauer bei 1 Entschlüsselung/μs	∅ Dauer bei 10^{12} Entschlüsselungen/s
32	$2^{32} = 4,3 \cdot 10^9$	$2^{31}\mu s = 3,58$ Minuten	2,15 Millisekunden
56	$2^{56} = 7,2 \cdot 10^{16}$	$2^{55}\mu s = 1145$ Jahre	10,01 Stunden
128	$2^{128} = 3,4 \cdot 10^{38}$	$2^{127}\mu s = 5,4 \cdot 10^{24}$ Jahre	$5,4 \cdot 10^{18}$ Jahre
168	$2^{168} = 3,7 \cdot 10^{50}$	$2^{167}\mu s = 5,9 \cdot 10^{36}$ Jahre	$5,9 \cdot 10^{30}$ Jahre
26 Zeichen (Permutation)	$26! = 4 \cdot 10^{26}$	$2 \cdot 10^{26}\mu s = 6,4 \cdot 10^{12}$ Jahre	$6,4 \cdot 10^6$ Jahre

Tabelle 2.3: ∅ Dauer eines Angriffs nach jeweiliger Schlüsselraumgröße

Die Brute-Force Suche kann sehr einfach parallelisiert werden, indem weitere Computerressourcen zu einem Netz zur schnelleren Schlüsselfindung verbunden werden. Ein Beispiel ist der Angriff auf den DES Algorithmus im Jahre 1999. Der Angriff konnte über das Internet verteilt werden und erreichte so einen Test von $92 \cdot 10^9$ DES Schlüssel pro Sekunde [14].

2.3.2 Ciphertext-Only Angriff

Bei einem Ciphertext-Only Angriff wird davon ausgegangen, dass der Kryptoanalytiker nur eine passive Fähigkeit hat, die verschlüsselte Kommunikation mitzuhören. In Abbildung 2.3 ist die Angriffsposition eines Kryptoanalytikers gezeigt.

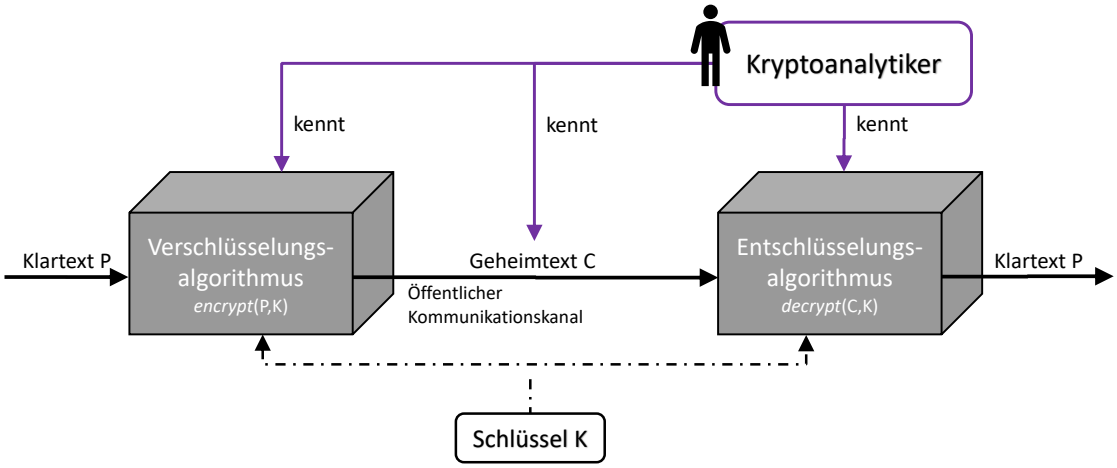


Abbildung 2.3: Allgemeines Kryptosystem unter einem Ciphertext-Only Angriff

Der Kryptoanalytiker kennt somit nur Geheimtext C_i mit $1 \leq i \leq |C|$, aber nicht die zugehörigen Klartexte P_i [4]. Bei diesen Angriffen steht dem Kryptoanalytiker nur der Geheimtext und die Chiffre (Kerckhoffs'sche Prinzip) zur Verfügung, um

den Klartext zu gewinnen. Dies kann geschehen, indem der Schlüssel K gefunden wird oder indem er einen Weg findet, um den Klartext auch ohne Schlüssel wiederherzustellen. Dieses Szenario stellt einen der stärksten Angriffe im Hinblick auf die Möglichkeiten des Kryptoanalytikers da, aber auch das Häufigste in der Realität.

Ein System, welches durch diesen Angriff erfolgreich analysiert werden kann, ist nach aller Erfahrung völlig unbrauchbar. Einfachste Beispiele hierfür sind die Cäsar-Substitution oder bei ausreichendem Geheimtext auch die polyalphabetische Substitution [8, 18]. Aber auch die Chiffriermaschinen des Zweiten Weltkrieges sind unter einem Ciphertext-Only Angriff lösbar [3, 19, 20].

2.3.3 Known-Plaintext Angriff

Sind dem Kryptoanalytiker darüber hinaus Teile korrespondierender Klar- und Chiffretexte bekannt, so wird von einem Known-Plaintext Angriff mit bekannten Klartext gesprochen [4, 7]. Die Abbildung 2.4 zeigt, dass der Kryptoanalytiker die Chiffre, den Geheimtext mit zugehörigen Klartext kennt.

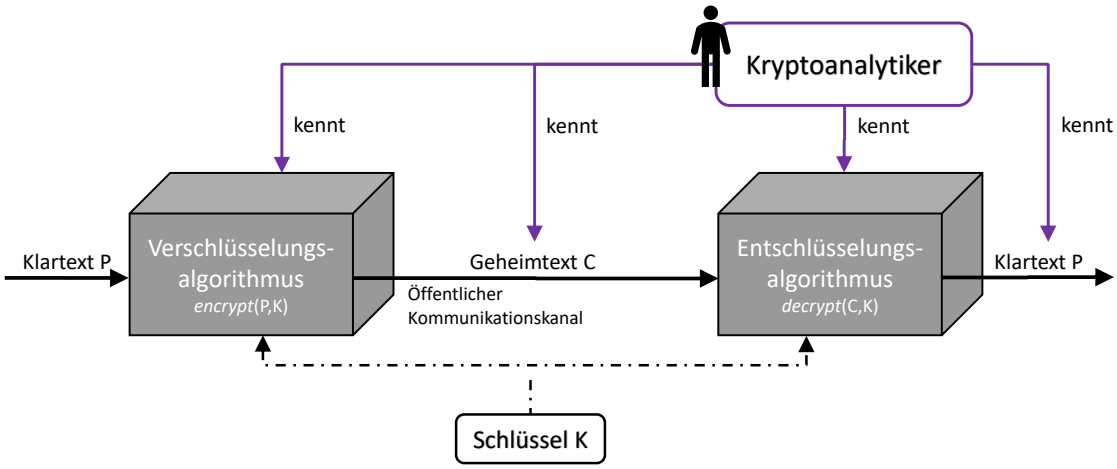


Abbildung 2.4: Allgemeines Kryptosystem unter einem Known-Plaintext Angriff

Die starre Struktur formaler Sprachen, wie z. B. häufige Funksprüche, liefern Kenntnisse von Klartext, die die Basis eines Known-Plaintext Angriff bilden können. Durch dies konnten die Briten die Funksprüche der Enigma lösen. Die Nachrichten enthielten eine bestimmte inhaltliche Ordnung. Dies führte zur Annahme von bestimmten Wörtern, die in gewissen Zeitfenstern vorkommen mussten. Der Zahlenkatalog „EINS“ von Alan Turing basierte auf dem technischen Nachteil, das Zahlen als Text (buchstabenweise) verschlüsselt wurden [21].

Ein Suchverfahren für diesen Angriff kann unter anderem sein, dass jeder Schlüssel ausprobiert wird, bis der Richtige gefunden wurde. Dabei wird der verschlüsselte Text mit Hilfe eines beliebigen Schlüssels dechiffriert und der entstandene Text

mit dem Originaltext verglichen. Stimmt der Text mit dem Originaltext überein, ist der richtige Schlüssel bestimmt. Ist das nicht der Fall, wird ein neuer Schlüssel ausgewählt. Dies geschieht solange, bis der richtige Originaltext ermittelt wurde.

2.3.4 Chosen-Plaintext Angriff

Bei Chosen-Plaintext Angriffen hat der Kryptoanalytiker zusätzlich die Möglichkeit, den Klartext, der verschlüsselt werden soll, selbst zu wählen [4].

Während die beiden vorherigen Situationen (Ciphertext-Only und Known-Plaintext) von einem passiven Angriff auf das Kryptosystem ausgehen, bei dem abgehörte Geheimtexte ausgewertet werden, kann der Chosen-Plaintext Angriff auch zu aktiven Eingriffen verwendet werden (Abbildung 2.5). Es birgt große Risiken, wenn viele Teilnehmer das gleiche Kryptosystem zur Kommunikations-Sicherung einsetzen. Beim Chosen-Plaintext Angriff kann der Kryptoanalytiker die verschlüsselte Version von ihm selbst beliebig gewählter Klartexte erhalten. Dafür steht ihm eine Black-Box Version des Entschlüsselungsverfahrens zur Verfügung. Der innere Ablauf einer Black-Box wird über die Eingaben und deren Ergebnisse erschlossen.

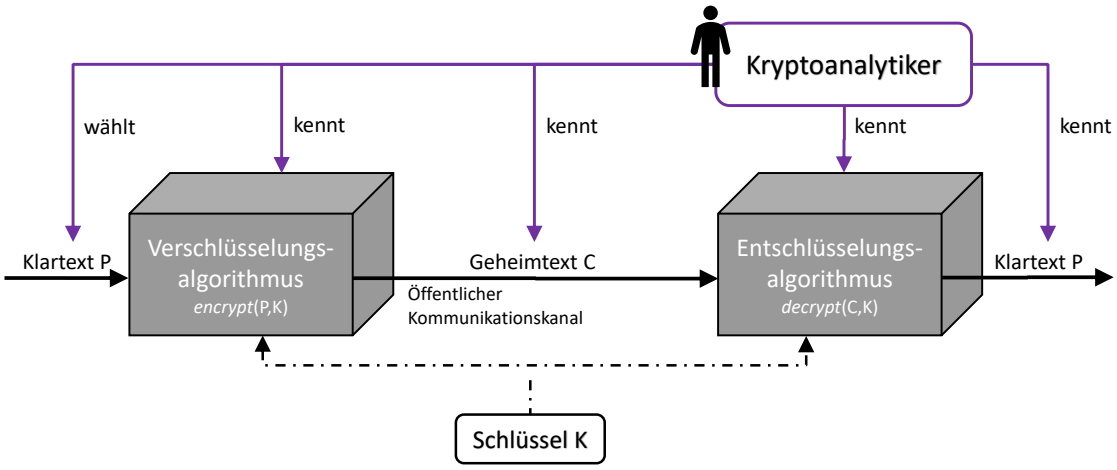


Abbildung 2.5: Allgemeines Kryptosystem unter einem Known-Plaintext Angriff

Die Bezeichnung Chosen-Plaintext Angriff erinnert daran, dass der Kryptoanalytiker den Klartext wählen und so ein Klar- und Geheimtext-Paar generieren kann. In der Praxis ist ein Chosen-Plaintext Angriff zur Bestimmung des Schlüssels schwer auszuführen, sofern ein konventionelles Kryptosystem benutzt wird. Ein Verschlüsselungsverfahren mit öffentlich bekanntem Schlüssel bietet ohne ein geeignetes Kommunikationsprotokoll jedoch jederzeit die Möglichkeit, zu beliebigem Klartext den Chiffretext zu bestimmen, selbst wenn der Verschlüsselungsalgorithmus an sich unangreifbar ist.

2.3.5 Heuristische Verfahren

Des Weiteren können auch die Buchstabenhäufigkeiten analysiert werden. Anhand der gezählten Häufigkeiten ist es möglich den Schlüssel zu finden. Diese Variante ist möglich, da keine Gleichverteilung von Buchstaben in den verschiedenen Sprachen auftritt. Wie in Abbildung 2.6 ersichtlich, ist die Verteilung der einzelnen Buchstaben in der englischen und deutschen Sprache an keine Gleichverteilung angelehnt.

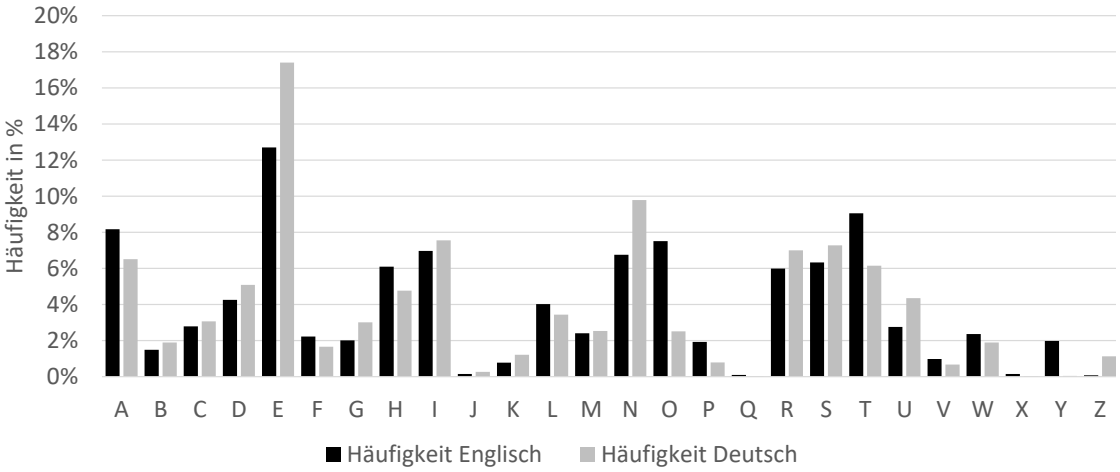


Abbildung 2.6: Buchstabenverteilung in deutscher und englischer Sprache [8, 22]

Sofern das Verschlüsselungssystem nicht die typischen Eigenschaften der Sprache aufhebt, kann das Analysieren des Geheimtextes Rückschlüsse auf den Klartext und sogar auf den Schlüssel zulassen. Diese Angriffe sind sehr effektiv bei langen Texten, denn die Worthäufigkeiten werden mit zunehmender Länge immer deutlicher und konvergieren zur Wahrscheinlichkeit der jeweiligen Sprache.

Es ist ebenfalls hilfreich das Auftreten von Buchstabenpaaren zu analysieren. Auch hierzu gibt es öffentliche Sammlungen über Buchstaben, die oft nebeneinander und zusammen auftreten.

3 Typex

Dieses Kapitel ist der Entstehung und Entwicklung der Typex gewidmet. Es werden historische Beschlüsse beleuchtet sowie die schrittweise Ableitung der Typex von der Enigma erklärt. Ihr Aussehen und die Funktionalität ähnelt der deutschen Enigma. Die Typex war eine von den Briten in der Zeit des Zweiten Weltkriegs entwickelte Rotor-Chiffriermaschine. Die Entwicklung von Rotormaschinen und der Typex begann deutlich vor dem Krieg. Die Funktion der Typex war umfangreich und wurde erstmals an unterschiedlichen Instituten erforscht.

Um 1920 wurde von den Briten die Schule der Kryptologie, genannt „Government Code and Cypher School“ (GC&CS), als übergreifende Organisation gegründet. Die Hauptaufgaben waren die Lehre über Kommunikationsmethoden von anderen Ländern und die Beratung von Regierungsorganisationen in sicherheitsrelevanten Fragen [5].

Im Jahr 1926 wurde ein Regierungsausschuss für die Suche eines effektiveren Kryptosystems berufen. Das neue Kryptosystem sollte zur Nachrichtenübertragung in Heer, Marine und Luftwaffe eingesetzt werden [23]. Alle verfügbaren Chiffriersysteme, wie Hebern, Kryha und die deutsche Enigma, sind bei der Neuwahl berücksichtigt worden.

Im Jahr 1928 erwarben die Briten zwei Enigmas vom deutschen Hersteller Chiffriermaschinen AG zur Mechanismusanalyse. Der Ausschuss sprach sich 1935 für die Herstellung einer gleichartigen Enigma mit zusätzlichen Installationen („Typex“) aus. Dies war der Ursprung der Typex und ihrer erfolgreichen Nutzung über viele Jahre beim Militär und anderen staatlichen Einrichtungen. Dieser Prototyp hatte drei verbundene Baueinheiten: das Chiffriermaschinen-Gehäuse mit Tastatur, das elektronische Verschlüsselungssystem (äquivalent zur Enigma) und die Typex-Installationen (Morsetechnik und Drucker) [24].

Heer und Luftwaffe haben die Typex schnellst möglich in den Dienst eingeführt. Im Jahr 1939 war sie bereits weitestgehend im Einsatz. Die Marine führte die Typex erst 1941 ein, als die bisherigen Chiffriersysteme als unsicher galten.

Die Entwicklung von Rotormaschinen basiert auf der Idee und dem Aussehen der kommerziellen Enigma. Sie beeinflusste die Entwicklung in anderen Ländern. Allerdings fiel recht schnell auf, dass die kommerzielle Enigma einen gravierenden Makel besaß, und zwar das seltene und regelmäßige Fortschreiten aller Rotoren. Diese Eigenschaft ließ Kryptoanalytiker passende Klartexte finden, die wiederum isomorphe Verbindungen aus dem Verschlüsselungsprozess aufzeigten. Die italienische Marine-Rotormaschine war identisch zur kommerziellen Enigma und konnte

daher mit den bisherigen Konzepten bis ins Jahr 1941 als die Maschine durch die Hagelin C-36 ersetzt werden [5]. Weitere Varianten der Enigma nach der kommerziellen Version dienten nur als Lösungsansatz für die isomorphen Schwächen.

Form und Aussehen der Typex zeigt nicht nur, dass die Briten die Maschine bis auf kleine Details kopierten (siehe Abbildung 3.3), sondern auch Bauteile aus dem Patent von Willi Korn entnahmen. Die Enigma wurde von Arthur Scherbius (Chiffriermaschinen AG) erfunden und einer seiner Mitarbeiter, Willi Korn, hat wesentlich zur Gestaltung dieser beigetragen. Diese zusätzlichen Baueinheiten aus dem Patent waren nicht in der serienmäßigen Produktion der Engima vorgesehen. In Abbildung 3.1 sind die Entwicklungen zu verschiedenen Ländern in einem Graph dargestellt. Die Abbildung 3.2 zeigt das ungefähre Entstehen auf einer chronologischen Zeitachse.

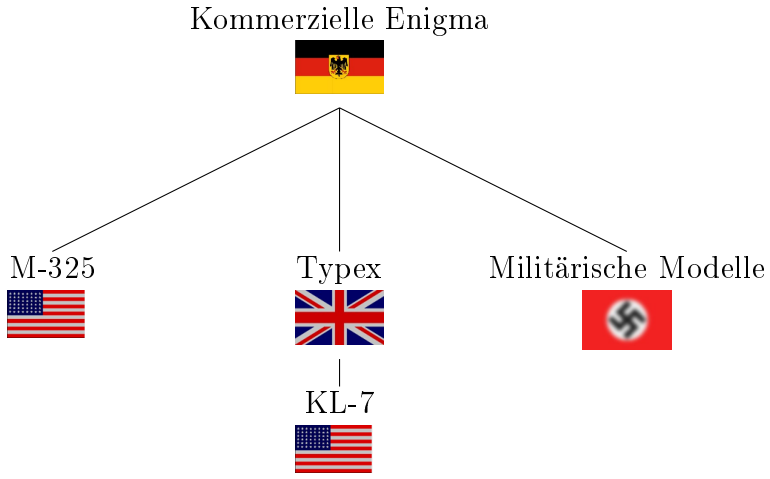


Abbildung 3.1: Typex Ursprung

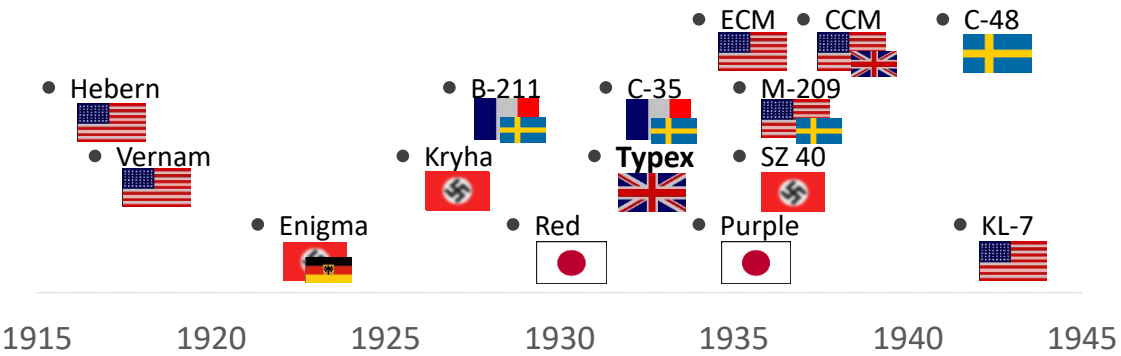


Abbildung 3.2: Ungefähre Chronologie der Maschinen-Entwicklung [24]

Die amerikanische M-325 war der Versuch von William Friedman eine Rotormaschine mit vergleichbaren Prinzipien der Enigma zu entwickeln [24]. Der Entwurf war bautechnisch unterschiedlich, aber kryptographisch ähnlich mit der Enigma. Die M-325 hatte drei Rotoren und eine Umkehrwalze (rechts eingesetzt, nicht wie bei Enigma und Typex). Nach der Bekanntgabe, dass die deutsche Enigma ein

Steckerbrett verwendet, wurde dieses auch bei der M-325 hinzugefügt. Die Version konnte bei richtiger Bedienung vergleichbare hohe Sicherheit bieten (wie die amerikanische Rotormaschine M-209 [25]). Allerdings war die Maschine langsam und schwierig zu bedienen, was sie untauglich für den Einsatz machte.

Die Wehrmacht nutze das Steckerbrett als Abwehr gegen isomorphe Angriffe. Diese Abwehrfunktion verlief mit mehr als fünf Steckern erfolgreich und machte damit die Enigma deutlich sicherer. Der Bediener wählte die Schlüsselpaare beim Steckerbrett nicht sorgfältig genug aus, denn diese Auswahl muss rein zufällig sein.

Die britische Typex hatte eine viel unregelmäßigere Drehmethode und andere Eigenschaften welche die Sicherheit erhöhten (z. B. Kerbenanzahl). Die Typex nutze viele Erfindungen aus dem deutschen Patent von Willi Korn, die nicht einmal bei der deutschen Enigma implementiert wurde. Es ist offensichtlich, dass der mechanische Aufbau direkt von der kommerziellen Version der Enigma übernommen wurde. Die Ähnlichkeit war so hoch, dass die Typex im Bletchley Park für die Kryptoanalyse der Enigma genutzt wurde [5].

Abbildung 3.3: Typex-Chiffriermaschine mit Druckfunktion [26]

3.1 Geschichte

Die Entwicklung für die Typex entstand durch die Ansprüche an Sicherheit und durch das Aufkommen des Funksystems. Das Funksystem erhöhte die Schwierigkeit bei der Geheimhaltung der zivilen und militärischen Kommunikation. Die Erfahrungen der Großmächte im Ersten Weltkrieg zeigten aber, dass kein Teilnehmer in einem großen Konflikt „Funkstille“ einhalten kann. Ferner zeigte sich, dass praktisch alle manuellen Methoden zur Verschlüsselung von Signalen viel zu

verwundbar waren, da ein erfahrener Gegner diese erfolgreich angreifen konnte. Viele Regierungen untersuchten daher schreibmaschinen-basierte Chiffriermaschinen in der Hoffnung, die Probleme der Kommunikationssicherheit damit zu lösen. Sie erkannten aber, dass größere Schwierigkeiten zu überwinden waren bis sie in den Einsatz gingen. Es war letztendlich ein langsamer Prozess während Friedenszeiten [27].

Wing Commander O. G. W. Lywood, in der Signals Division der Royal Air Force („RAF“, britische Luftwaffe), glaubte eine verbesserte Version der kommerziellen Enigma zu entwickeln, die mit Bauteilen aus dem Creed-Fernschreiber fähig wäre einen gedruckten Text zu produzieren. Der „Inter-Departmental“ Ausschuss weigerte sich solch eine Idee finanziell zu fördern. Die RAF hingegen sah es als lohnend mit der Entwicklung fortzufahren. Sie autorisierte Lywood daher im Jahr 1934 [24, 27].

3.1.1 Die Royal Air Force Enigma

Der erste Prototyp von Lywood wurde am 30. April 1935 an das Luftfahrtministerium (Air Ministry) geliefert. Es wurden bauliche Veränderungen und Einsatzerprobungen vorgenommen. Die neue Variante mit den Testergebnissen ging am 16. Mai 1936 an das Ministerium zurück [28, 29].

Im Januar 1935 wurde dem Luftfahrtministerium vom Inter-Departmental Cypher Komitee allerdings empfohlen, drei Maschinen als Weiterentwicklung der Enigma zu konstruieren. Dieses Gesprächsprotokoll wurde innerhalb des Nachrichtendienstes als „Type X“ benannt, woraus der Name für die britische Chiffriermaschine entstand [28].

Die zweite Erprobungsphase veranlasste das Luftfahrtministerium, die Typex Mk. I (Mark I) in Serie herzustellen. Die RAF fand heraus, dass die manuellen Kryptosysteme mit dem stetigen Anstieg bei der Datenübertragung überfordert waren. Die Blockanzahl beim Chiffrieren sprang von 6.000 auf 72.000 pro Monat und beschäftige 25 Personen. Bei Nutzung der Typex MK. I konnten 75.000 Blöcke mit 10 Arbeitern bewältigt werden [5].

Eine vorzeitige Teilentwicklung der Mk. II wurde im Februar 1937 initialisiert. Der finale Konstruktionsplan wurde 1938 umgesetzt und im Juni 1938 dem Inter-Departmental Cypher Komitee vorgestellt [27]. Das Komitee war vom Konzept überzeugt. Es wurden 350 Maschinen bestellt, wobei 30 Exemplare zur Einarbeitung beim Militär geliefert wurden.

Im Jahre 1939 war die Typex noch immer sehr ähnlich zur kommerziellen Enigma. Nichtsdestotrotz war die Typex Mk. II mit den Creed Telegraphen (Druckeinheit) viel größer als die Enigma. Die Enigma Größe maß 35 x 28 x 10 cm und war 12 kg schwer. Damit war sie deutlich kleiner als die Typex (75 x 55 x 35 cm, 55 kg). Anstelle eines Steckerbrettes gab es bei der Mk. II zwei Eintrittswalzen (wie stationäre Rotoren zum Einsetzen), welche sich während des Chiffriervorgangs nicht

drehten. Die beiden Statoren gab es zusätzlich zu den drei rotierenden Rotoren (siehe Abbildung 3.4 und 3.5). Die Typex-Rotation war unregelmäßiger verteilt mit Hilfe eines Pawl-Mechanismus von Smith [30]. Die Aufnahme von bis zu neun Kerben war auch unterschiedlich zur Enigma, da diese nur eine Kerbe auf den Rotoren I bis V und zwei Kerben auf den Rotoren VI bis VIII hatte. Für die Verlässlichkeit wurde jeder elektronische Kontakt bei den Rotoren doppelt verdrahtet. Der signifikanteste Unterschied zwischen der kommerziellen Enigma und Typex waren die zwei Druckeinheiten. Das Steckerbrett wurde erst spät im Krieg bei der Typex hinzugefügt [5].

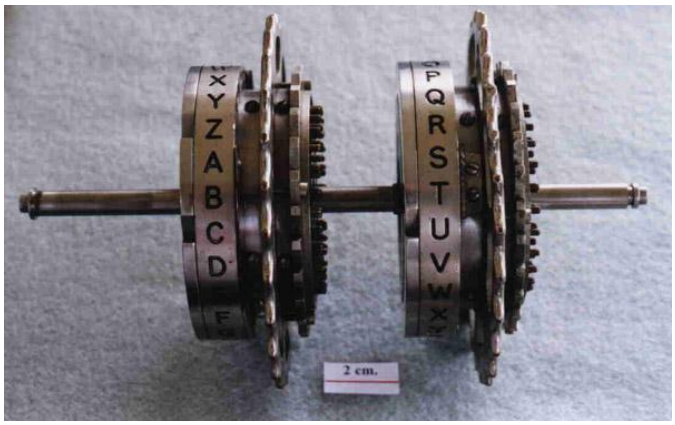


Abbildung 3.4: Rotorachse mit zwei Rotoren der Typex [31]

3.1.2 Die Typex Produktion

Die Nachfrage der Typex in allen Bereichen der Streitkräfte war enorm, aber leider konnte die Produktion dies nicht decken. Die Versorgung kam ins Stocken. Die Typex Produktion war langsam und nach der Errichtung einer zweiten Herstellungsstraße bei Creed in Wales wurde nur eine Produktion von ca. 2.400 Stück erreicht. Dies war immer noch eine Differenz von 1.800 Stück zur prognostizierten Herstellung. Dies erhöhte sich bis auf über 4.000 im Jahr 1944 [27].

Die Gründe für die langsame Produktion waren offensichtlich. Jede Art von Rotormaschinen basierte auf einer komplexen Mechanik. Die Abbildungen 3.5 und 3.6 zeigen die hohe Anzahl von Einzelteilen. Die britische Maschinenindustrie war überfordert mit der Komplexität der Maschine.

Trotz des hohen Interesses der Regierung an der Typex, dauerte es fast zwei Jahre, alle Werkzeuge für eine erfolgreiche Herstellung zu organisieren, damit eine annähernd reguläre Produktion hergestellt war. Nur ca. 2.300 Maschinen wurden bis Ende 1942 hergestellt, 4.000 Stück etwa 1943, 5.016 im Mai 1944. Über 8.200 Mk. II und eventuell 3.000 Mk. VI wurden bis August 1945 produziert. Die gesamte Herstellung aller Maschinenversionen war bis zum Ende des Krieges bei etwa 12.000 Stück [27].

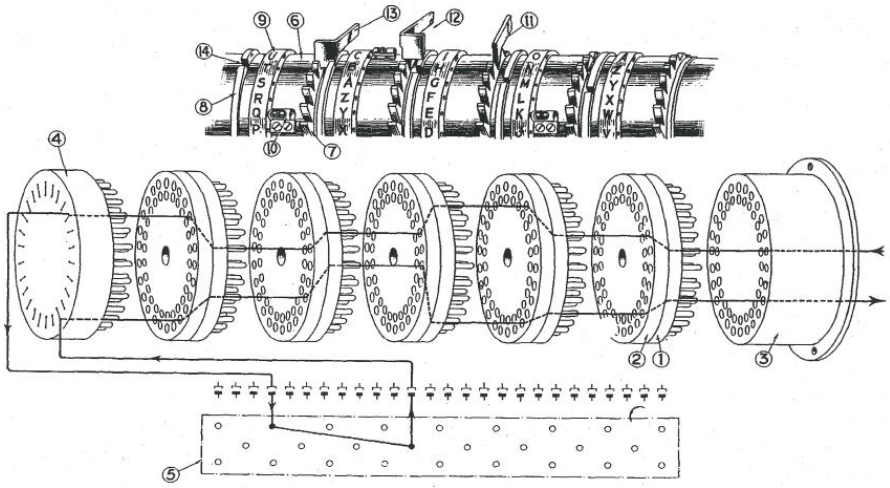


Abbildung 3.5: Explosionsdiagramm der Rotorachse [32]

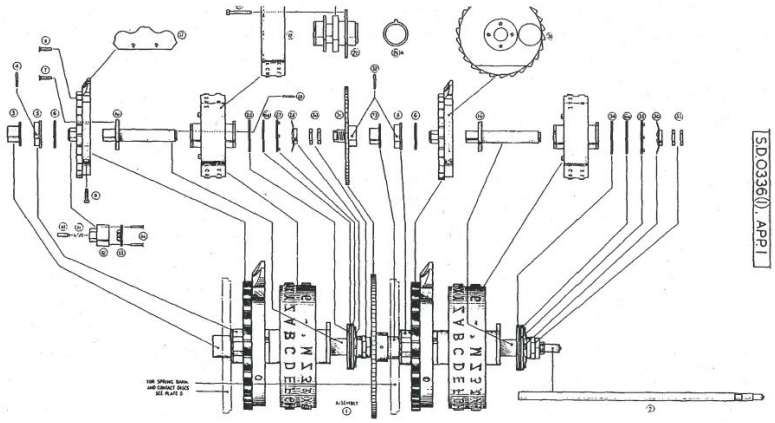


Abbildung 3.6: Explosionsdiagramm – Montage der Rotorachse [32]

Die Produktionsprobleme wurden verschlimmert durch eine stetig steigende Nachfrage. Die Kunden konnten ihren Bedarf nicht richtig einschätzen. Daher verdreifachte sich der geplante Bedarf von 2.550 auf 7.500 innerhalb eines Jahres [27]. Die Produktion der Mk. II stieg nach 1943 nicht großartig an. Zwei Produktionsstraßen konnten nicht mehr als 250 Maschinen pro Monat fertigen [29]. Umstrukturierungen und Lohnpolitiken sowie die hohe Anzahl von defekten Zulieferungen waren Gründe für die niedrige Produktionsrate. Die für die Produktion benötigten Materialien waren nicht immer verfügbar, um die Nachfrage der Streitkräfte nur etwas zu befriedigen. Es gab so wenige Typex-Maschinen, dass sogar die Kommunikation im Mittleren Osten 1941 bis 1942 ohne eine Typex stattfand [13].

3.2 Funktionsweise

Im Folgenden wird das technische Konzept als auch die notwendigen Baueinheiten der Typex erklärt. Solch eine Maschine besteht aus Rotoren und schreibmaschinenähnlichen Mechanismen. Die funktionale Analyse der Typex basiert zu großen Teilen auf den Bildern. Es gab verschiedene Modelle der Typex. Daher können mehrere Maschinen auf die folgenden Beschreibungen passen.

Die Typex war über eine standardisierte Elektroversorgung angeschlossen. Die Ausgabe erfolgte über Drucker im hinteren Teil. Die Maschine funktionierte mit fünf austauschbaren Rotoren/Statoren und einem sechsten Bauteil, der Umkehrwalze ganz links. Die Rotormechanik und das Gehäuse sind nahezu identisch mit dem der kommerziellen Enigma. Die tragbare Mark III hatte eine Kurbel auf der rechten Maschinenseite, womit ein Impulsgenerator für den Drucker angestoßen wurde.

3.2.1 Rotoren

Das Rotieren der Rotoren bei der Typex (und Enigma) war durch zwei Zahnräder je Rotor gewährleistet. Jeder Rotor hatte außen 26 Buchstaben aufgedruckt, sowie 26 Kontakte auf beiden Seiten. Mit dem linken Zahnrad wurde die Grundstellung modifiziert (Abbildung 3.7(a)), mit dem rechten Zahnrad wurde die Ringstellung mit den unregelmäßigen Kerbenpositionen beeinflusst (Abbildung 3.7(b)). Der Rotor drehte sich anfangs bei der Grundstellung bis das linke Zahnrad bei einer Kerbe des benachbarten Rotors einrastete und diesen mit dem führenden Rotor jeweils um einen Schritt rotieren ließ. Dieses mechanische Konzept implementierte eine Anomalie innerhalb der Rotation. Sobald der Rotor seinen Nachbar dreht, schreitet er selbst fort. Dies produzierte nicht die maximale Perioden von 26 Buchstaben, sondern deutlich kürze. Die exakte Periode hing von der Grundstellung auf dem Rotor ab. Die zur Enigma verwandte Hebern-Maschine hatte das Zahnrad für die Grundstellung vom Rotor losgelöst.

Es gab nur eine Kerbe pro Rotor in der Heeres-Version der Enigma. Ferner gab es zwei Kerben bei der späteren Marine-Version, wobei die Typex bis zu 9 Kerben auf einem Rotor besaß [5].

Eine besondere Eigenschaft bei der Typex waren die zwei Reihen mit Kontakten. Dies war ein Schutz vor Kontaktverlust, da die Rotoren durch die Drehung nicht immer bündig bei allen Kontakten anlagen. Daher wurden zwei Kontakte zur besseren Überlappung zweier Kontakte konstruiert (Abbildung 3.7).

Die zwei Eintrittswalzen, auch Statoren genannt, drehten sich nicht und sind mechanisch durch eine gegenseitige Sperrvorrichtung davon ausgeschlossen. Als die Enigma durch das Steckerbrett revolutioniert wurde, versuchten die Briten mit den beiden Statoren dasselbe Sicherheitskonzept zu verfolgen. Allerdings war das Steckerbrett involutorisch (selbstinvers), das heißt, wenn bei einer bestimmten

(a) Rotor mit Stiftkontakt

(b) Rotor mit Kerbkontakt

(c) Sammlung von Rotoren

Abbildung 3.7: Verzahnter Rotor mit drei Kerben entnommen [31]

Stellung der Rotoren ein U in ein X verschlüsselt wird, dann wird bei dieser Stellung auch ein X in ein U verschlüsselt. Des Weiteren bedeutet es, dass nun kein Buchstabe mehr in sich selbst verschlüsselt werden kann, denn der Strom kann in keinem Fall genau den Weg durch die Rotoren und Statoren wieder zurück laufen, den er gekommen ist. Dies würde letztlich in einem Kurzschluss resultieren.

Der rechte Rotor, auch schneller Rotor genannt, dreht den mittleren Rotor und dieser wiederum dreht den langsamen Rotor.

3.2.2 Druckeinheit

Das Drucken wird durch ein Rad mit geprägten Buchstaben ermöglicht. Dieses wird ständig von einer Kartusche eingefärbt und dann gegen einen einzelnen Streifen Papier gedrückt. Der Mechanismus drehte sich einmal für jede gedrückte Taste. Elektrische Energie wurde von einem Impulsgeber jeweils zu Beginn der Revolution, in einem Kondensator gespeichert und dann freigegeben. Wenn das entsprechende Zeichen den Druckkopf erreichte und ein kleines Relais den Druckkopf auslöste, drückte das Papier kurzzeitig gegen das Rad. Bedingt durch den Druck und die kleine Menge der Stromversorgung war es ein kritischer Vorgang.

- Verschlüsselungsmodus: Die linke Trommel druckte die Klartextversion, von wo getippt wurde, während die rechte Trommel die verschlüsselte Version zum Überprüfen der Eingabe druckte.
- Entschlüsselungsmodus: Dieser war das reine Gegenteil zum Verschlüsselungsmodus. Die linke Druckeinheit würde den chiffrierten Text drucken, während die rechte Trommel die entschlüsselte Nachricht im Nur-Text-Format druckt.

Mit anderen Worten, die linke Trommel übernahm immer die Eingabe von der Tastatur, während die Rechte die Ergebnisse der Chiffre (Verschlüsselung- oder Entschlüsselungsmodus) ausdruckte.

3.2.3 Modell-Reihe

Die üblichsten Modelle der Typex war die Mk II (Mark, dt: Typ, Modell) und die mobile Version Mk. VI mit dem Produktionsbeginn um November 1943 [29]. Der markante Unterschied der beiden Maschinen war die Stromversorgung, da die Mk. VI über eine 6V-Batterie anstatt einer stationären Stromzufuhr verfügte. Eine Handkurbel hatte eine passende Übersetzung, um die Druckeinheit bei der Typex zu betreiben. Tabelle 3.1 zeigt die Unterschiede zwischen den Modellen auf. In der aktuellen Literatur liegen keine umfangreichen Informationen über die Maschinen Mk. IV, V und VII vor. Dies könnte ein Indiz sein, dass diese Maschinen noch in der experimentellen Phase waren oder gar nur wenige in die Serienproduktion gelangten. Die Entwicklung hielt aber über den Krieg hinweg an und erfuhr weiteres Interesse nach dem Krieg. Während dieser Zeit wurde ein Steckerbrett bei gewissen Versionen ergänzt. Ebenfalls sollte ein modifizierbare Umkehrwalze mit steckbarer Verkabelung für die Typex Mk. 22, eine Weiterentwicklung der Mk. II, entwickelt werden. Die gewählte Verkabelung ist ein Schlüsselbestandteil. Sie war zum Ende des Krieges bei der Enigma mit der Umkehrwalze Dora zuerst aufgetreten.

Sieben Rotoren waren für eine akzeptable Sicherheit erforderlich, wobei im Jahr 1941 eine Typex mit 10 Rotoren ausgeliefert wurde. Die Rotoren waren zuerst ein gesamtes Fabrikatstück. Später wurde es zu einem stabilen Kranz mit einer herausnehmbaren Einlage, die die Verdrahtung beinhaltete. Die Mk. VI wurde mit 14 Einlagen (Verdrahtungen) ausgeliefert, wobei 5 davon in der Maschine gleichzeitig benutzt wurden. Die Einlagen konnten in beide Richtungen eingesetzt wrden. Somit wurde die Anzahl der möglichen Rotoren verdoppelt, da jede Richtung eine eigene Permutation repräsentierte. Zu Kriegsbeginn nutzten ranghöhere Einheiten der britischen Luftwaffe die Mk. I Rotoren und die restlichen Truppen die Mk. II Rotoren. Es gab bis zum Kriegsende 9 Rotor- bzw. Verdrahtung-Sätze sowie einen losgelösten Satz für die Typex der britischen Luftwaffe [30].

Die Idee, dass aus einer Schreibmaschine eine Chiffriermaschine entstehen kann, war bereits von der Enigma bekannt. Lywood führte eine weitere Ebene ein. Er machte die Typex-Verschlüsselung zur Sendeeinheit mit der Morsetechnik und

die Typex-Entschlüsselung zum Empfangsmodul durch den Drucker. Dies war der Grundstein zur automatischen Online-Kommunikation. Im Jahre 1936 erhielt das britische Luftfahrtministerium einen ersten Prototypen. Die erste Typex Mk I gab es so dann im Jahre 1937. Die staatliche Kommunikationsbehörde (GC&CS) sprach sich für die absolute Sicherheit aus, selbst wenn der Geheimtext und zahlreiche Klartexte bekannt waren.

Mark	Spannung	Drucker	Geschwindigkeit (Gruppen per Minute)	Bemerkungen
I	230V	Ja	8	Nur 29 hergestellt
IA	230V	Nein	8 (mit 2 Betreiber und 1 Schreiber	Alle Funktionen von Mk. II und III, außer drucken da Zeichen angezeigt wurden
IB	230V	Nein	8	
II	230V oder Batterie	Ja	40 bis 50	40 war ein bemerkenswerte hohe Geschwindigkeit
IIA	230V oder Batterie	Ja	40 bis 50	Gleichzeitig wurden 3 bis 4 Kopien vom chiffrierten Text auf zwei oder mehr Kanälen entsendet.
22	230V	Ja	40 bis 50	Einsetzbare Umkehrwalze. Weiterentwicklung der Mk. II. Eventuell eine Nachkriegsmaschine
III	Handbetrieb	Ja	16 bis 18	Der Drucker war durch einen Impulsgenerator mit einer Kurbel. Strom wurde im Kondensator gespeichert und 230V Antriebseinheit war auch vorhanden. Gewicht mit Gehäuse: 29kg
VI	6V Batterie oder Akku	Ja	12	Spezielle hohle Rotoren die nicht kompatibel mit anderen Modellen war. Größe: 50 x 30 x 23cm Gewicht: 13kg
VIII	230V oder Batterie	Ja	50	Mk. II mit Morse-Perforiermaschine. Nur 398 Maschinen bestellt am 10 Januar 1945

Tabelle 3.1: Typex Modelle [29]

Zeitnah entwickelte Lywood eine weitere Maschine, die Typex Mark II, welche eine der beiden Standartmodelle werden sollte. Mit jedem weiteren Entwicklungsschritt näherte sich die Typex der Enigma, letztlich ohne ein Detail auszulassen. Lywood entfernte die Typex-Installationen, die ihr einst den Namen gaben. Die Typex Mk. I war voluminös, komplex und kaum beweglich. Der Verzicht auf die Typex-Installationen machte die Typex Mk. II deutlich einfacher als den Vorgänger. Die Mk. II war eine bewegliche und robuste Offline-Chiffriermaschine mit breitem Einsatzgebiet im Büro oder auf dem Feld, da die Typex einfach auf einem Lastwagen verladen werden konnte. Die Mk. I konnte zur Übersee-Kommunikation in den Ministerien eingesetzt werden. Allerdings wurde eine Technologie für die zahllosen kleinen Kommunikationsnetzwerke gesucht. Letztendlich erhielt die Mk. II passende Anschlüsse, um extern mit den Druckern und der Morseeinheit zu kommunizieren. Damit konnte die Typex auf Wunsch online geschaltet werden. Das Team um Lywood erhöhte die Geschwindigkeit und demonstrierte 1938 den ersten Mk. II Prototyp. Dies löste viele Bestellungen aus. Nichtsdestotrotz wurde die Anzahl an Bestellungen der Typex Mk. III nicht erreicht, welche eine manuelle offline Maschine, ähnlich zur Enigma, war.

Die Briten hatten damit das schwerwiegende Problem der kryptografischen Kommunikation durch den Bau der Typex auf dem Konzept der Enigma elegant und schnell gelöst. Der Nachbau funktionierte so tadellos, dass zwanghaft jegliche Ähnlichkeit zum Enigma-Patent abgestritten wurde, um so gerichtliche Schritte fern zu halten [33]. Die Typex Mark II wurde als schwächere Maschine gegenüber der Enigma betitelt, da selbst die Rotorbewegung der Typex nicht die kryptographische Stärke des Steckerbretts schlägt. Spätere mathematische Lösungen konnten diese Annahme nicht halten. Weitere Verfeinerungen hob die Typex über die kryptografischen Standards hinaus. Die Enigma stand der Typex von Einführung bis hin zur Überholung nichts nach. Die Typex Mark I mit lokaler Stromversorgung und online Funktion hatte die höchste Sicherheit und Signalverarbeitung in den Jahren 1939 - 1942. Die amerikanische Rotormaschine Sigaba war durch die irreguläre Rotordrehung noch sicherer als die Typex und zuverlässiger im Einsatz, musste allerdings manuell bedient werden. Diesen Nachteil kompensierte die Typex Mk. X im Jahre 1950, indem der Rotationsmechanismus von Sigaba auf die Typex übernommen wurde. Friedman und kryptografische Ausschüsse beachteten die Typex als unterlegen zur Sigaba, besonders die Typex Mk III stand in der Kritik [34]. Die Rotormaschinen wurden in ihrer Qualität und Einsatzmöglichkeit gemessen. Dort führte bis 1939 die Enigma, 1940 die Typex, 1941 die Sigaba und ab 1942 allesamt abgelöst durch Online-Chiffriermaschinen.

Alle Versionen der Typex hatten gewisse Vorteile bei der Benutzerfreundlichkeit gegenüber der Enigma. Jedes Drücken eines Buchstaben auf der Tastatur löste das Aufleuchten eines Lämpchens bei dem ent-/verschlüsselten Partner aus. Dieses Signal wurde von einem zweiten Mitarbeiter auf einem Zettel notiert. Die Typex erforderte nur einen Bediener, welches im Jahr 1945 eine Personalkapazität von 12 000 Mitarbeitern erforderte [5]. Zugleich wurden auch Fehler ausgemerzt, da der Text automatisch gedruckt wurde.

Anders als bei der Enigma, waren alle Mk. I mit einem Drucker verknüpft. Die Mk. II war technisch dafür sogar geschaffen. Die welterste Online-Chiffriermaschine umfasste viele nützliche Eigenschaften und arbeitete deshalb sehr effizient. Bisherige Signale, wie etwa von Enigma oder Sigaba, mussten erstellt, verschlüsselt, übertragen, empfangen, entschlüsselt und auf Papier geschrieben werden. Eine Nachricht bei der Typex wurde in einem Schritt bei Eingabe automatisch verschlüsselt und versendet. Beim Empfangen wurde sie entschlüsselt und ausgedruckt (zweiter Schritt).

3.2.4 Vergleich mit Enigma

Die einzelnen Länder, wie Deutschland und Großbritannien, hatten eigene Chiffriermaschinen und vertrauten diesen rigoros [35, 36]. Die Deutschen nutzten die Enigma-Chiffriermaschine. Die militärische Version der Enigma war standardisiert. Es gab nahezu keine Abweichungen zwischen den drei Versionen der Wehrmacht. Selbst das Konzept der fünf identischen Rotoren war gleich geblieben. Nur die Marine mit der 4-rotorigen Enigma M4 besaß zusätzliche Rotoren (VI bis VIII) und zwei herausnehmbare Umkehrwalzen. Die Enigma Uhr, welche das Steckerbrett nicht involutorisch (selbstinvers) macht, und die Umkehrwalze Dora kamen verhältnismäßig spät zum Einsatz. Nicht alle Enigmas waren kompatibel dazu. Es existieren Vermutungen, dass das Oberkommando der Wehrmacht (OKW) eine eigene Version mit speziellen Rotorverdrahtungen verwendete, womit nur in der Führungsriege kommuniziert wurde. Die Enigma M4 mit dem 4. Rotor war nur geringfügig anders konstruiert. Die Typex wurde durch ihre kryptografische Stärke unterschiedlich positioniert. Es gab allein 120 verschiedene Möglichkeiten die Rotoren in die Typex einzulegen.

Vielmehr konnten fünf Rotoren aus einer Sammlung von 28 ($= 2 \cdot 14$, beide Laufrichtungen) bei einer Mk. VI gewählt werden. Nicht wie bei der Engima eine Wahl von 3 Rotoren aus 5 oder bei der 3-rotorigen Marine-Enigma M3 mit Wahl von 3 Rotoren aus 8 [24]. Ein einzelner Rotorsatz mit 14 Einlagen (Verdrahtungen) kann 7.687.680 ($= 28 \cdot 26 \cdot 24 \cdot 22 \cdot 20$) Möglichkeiten bei der Rotorauswahl kreieren. Im Gegenteil konnte die Enigma nur 60 ($= 5 \cdot 4 \cdot 3$) oder die M3 nur 336 ($= 8 \cdot 7 \cdot 6$) Möglichkeiten erzeugen [37].

Die Typex war eine deutlich verfeinerte Maschine als die Enigma. Die Wehrmachts-Enigma hatte nur zwei Rotorsätze, eine für die Landstreitkräfte und eines für die Marine-Version M3/M4. Große Ähnlichkeit muss bei den Enigmas vorhanden sein, da gemeinsame Rotoren übergreifend genutzt wurden. Die Typex verwendete 20 oder mehr komplett unterschiedliche Rotorsätze, wobei kein Rotorsatz dem anderen bei der Verdrahtung ähnelte. Zusätzlich wurde kein Rotor aus der Rotorsammlung für die übergreifende Kommunikation im Funknetz vom Herr, Marine oder Luftwaffe benutzt. Denn bei einem erfolgreichen Angriff auf eines der Netze war die Sicherheit der anderen Netzwerke weiterhin gewährleistet. Damit die Briten bei der Enigma mit hören konnten, galt es, fünf Rotor-Verdrahtungen zu finden.

Im Gegensatz hierzu hatten es die Deutschen mit 120 bis zu 252 Verdrahtungen zu tun. 120 entstanden aus den drei Rotorsätzen aus Herr und Luftwaffe mit je 10 Rotoren (beide Richtungen), 252 entstanden später aus der Ausstattung der britischen Luftwaffe [5].

Die Deutschen engagierten Kryptoanalytiker, die sich mit dem Angriff auf die Typex beschäftigten. Dies endete mit geringen Erfolgen. Die Typex galt weiterhin als unangreifbar.

Die folgende Auflistung zeigt die Unterschiede zwischen Typex und Enigma [38]:

- Zwei Statoren statt dem Steckerbrett
- Mehr Kerben zum Rotieren
- Die Rotoren und Statoren können in beide Richtungen eingesetzt werden

Das Steckerbrett und die enorme Anzahl an Rotoren machten die Typex sicher. Auch ohne Steckerbrett war die Typex durch die Vielfalt der Rotoren gegen Kryptoanalytiker geschützt. Es ist verwunderlich, dass die Deutschen nie wirklich versuchten die Typex zu brechen.

Die Deutschen kaperten in der Schlacht um Dunkirk 1940 eine Typex und eine zweimonatige Schlüsselliste für ein Kommunikationsnetzwerk, allerdings fehlten die zugehörigen Rotoren [5]. Die Ironie bei der britischen Typex-Entwicklung ist, dass basierend auf der Enigma entwickelt wurde. Rechtlich war sie durch Patente der „Chiffriermaschinen Aktiengesellschaft“ geschützt [39]. Das Patent erklärt im Detail die Beschaffenheiten einer Rotormaschine und ist sinngemäß auf die Typex offensichtlich übertragen. Die britische Regierung wäre durch die Zahlung von Lizenzgebühren berechtigt gewesen, die Konzepte zu nutzen. Allerdings konnten die Briten selbst zu Friedenszeiten keine Lizenzgebühren entrichten, da das Projekt Typex als höchst sicherheitsrelevant eingestuft war und somit eine Zahlung zu Kriegszeiten völlig ausgeschlossen war. Letztendlich wurden nie Zahlungen an den Patentinhaber autorisiert.

Die Entwicklung der Typex hatte die Briten nur wenig gekostet, da der Großteil der Forschung und Konstruktion indirekt von der Chiffriermaschinen AG zum Verkauf der kommerziellen Enigma bereits investiert wurde.

In den Nachkriegsjahren wurde die Typex kontinuierlich weiter entwickelt. Es wurde aber parallel nach einer Ausweichmaschine gesucht. Mit dem Wissen, dass Bletchley Park die deutsche Enigma unterwanderte, fühlte sich die damalige Regierung in Sicherheitsfragen verunsichert. Obwohl sich die Enigma mit der geringen Rotoranzahl in einer anderen Dimension befand, war die Ähnlichkeit der beiden Maschinen zu groß. Der Vergleich von Enigma und Typex lässt erkennen, dass niemals eine gebrochene Maschine als Standard für eine weitere Maschine gelten sollte.

3.2.4.1 Schlüsselraumgröße der Enigma

Das folgende Kapitel erläutert den Schlüsselraum der deutschen Enigma-Chiffriermaschine. Die Baueinheiten werden mit dem jeweiligen Schlüsselraum vorgestellt und final in den gesamten Schlüsselraum summiert. Der Schlüssel K für die Verschlüsselung ist identisch mit der aktuellen Drehposition der Rotoren beim ersten Buchstaben. Die erste Eingabe erfolgt anhand des Schlüssels mit der Konfiguration. Die folgenden Baueinheiten waren beim Chiffriervorgang integriert:

(a) Die Grundstellung bestand aus drei Rotorpositionen über 26 Buchstaben. Diese Eigenschaft brachte $26^3 = 17.576$ Möglichkeiten hervor. Sie lässt sich mit dem binären Logarithmus als $2^{14,1}$ umrechnen.

(b) Die Auswahl der Rotoren basierte auf der Anzahl zu wählender Rotoren und auf der Anzahl der verfügbaren Rotoren. Bei der Enigma wurden drei Rotoren aus einem fünfteiligen Walzensatz gewählt. Dies ermöglichte $5 \cdot 4 \cdot 3 = 60$ unterschiedliche Rotorreihenfolgen ($2^{5,9}$). Die Umkehrwalze war bei früheren Enigma-Versionen, unter anderem bei der kommerziellen Version, nicht austauschbar. Später konnte aus zwei Umkehrwalzen gewählt werden. Diese Faktoren bildeten $60 \cdot 2 = 120$ Kombinationen ($2^{6,9}$).

(c) Die Ringstellung verursachte eine Verschiebung des Alphabetes um eine bestimmte Anzahl von Zeichen. Alle drei Rotoren konnten mit einer Verschiebung versehen werden. Dies bedeutete $26^2 = 676$ Möglichkeiten ($2^{9,4}$).

(d) Das Steckerbrett konnte mit höchstens 13 Kabeln gesteckt werden. Jedes Kabel hatte zwei Stecker. Im gesteckten Zustand veranlasste es eine Vertauschung zwischen jeweils zwei Buchstaben. Bei einem Alphabet mit 26 Buchstaben waren maximal 13 Vertauschungen möglich. Es waren üblicherweise zehn Kabelverbindungen gesetzt. Die Funktion $F(p)$ repräsentierte die möglichen Kombinationen einer Steckeranzahl p. Die Werte in Tabelle 3.2 zeigt die berechneten Werte. Schlussfolgernd existierten $F(10) \approx 2^{47,1}$ unterschiedliche Kombinationen nach Formel 3.1.

$$F(p) = \frac{26!}{(26 - 2p)!\ p!\ (2^p)} \tag{3.1}$$

p	$F(p)$	$\log_2(F(p))$
0	1	$2^{0,000}$
1	325	$2^{8,344}$
2	44.850	$2^{15,453}$
3	3.453.450	$2^{21,720}$
4	164.038.875	$2^{27,289}$
5	5.019.589.575	$2^{32,225}$
6	100.391.791.500	$2^{36,547}$

p	$F(p)$	$\log_2(F(p))$
7	1.305.093.289.500	$2^{40,247}$
8	10.767.019.638.375	$2^{43,292}$
9	53.835.098.191.875	$2^{45,614}$
10	150.738.274.937.250	$2^{47,099}$
11	205.552.193.096.250	$2^{47,546}$
12	102.776.096.548.125	$2^{46,546}$
13	7.905.853.580.625	$2^{42,846}$

Tabelle 3.2: Stecker-Kombinationen

In der hier beschriebenen Konfiguration (Formel 3.2) verfügt die Enigma über ein Schlüsselraum von *77 Bit*.

$$2^{14,1} \cdot 2^{6,9} \cdot 2^{9,4} \cdot 2^{47,1} \approx 2^{77,5} \tag{3.2}$$

Schlüsselraumgröße bei unbekannter Verdrahtung Ist die Verdrahtung von Rotoren und Umkehrwalze nicht bekannt, also ein Teil des Schlüssels, sodann steigt der theoretische Schlüsselraum auf *378 Bit*. Die folgende Auflistung zeigt die Veränderung zum regulären Schlüsselraum:

(a) Eine Rotorverdrahtung hat 26! Möglichkeiten und verhält sich bei drei Rotoren gemäß Formel 3.3.

$$(26!)^3 \approx 2^{265,15} \tag{3.3}$$

(b) Die Umkehrwalze verhält sich äquivalent zu einem Steckerbrett mit 13 Kabeln (Verdrahtungen). Die Formel 3.1 ergibt für $F(13) \approx 2^{42,85}$ Möglichkeiten.

Sofern die Verdrahtung geheim ist, beläuft sich der Schlüsselraum der Enigma auf *378 Bit* (Formel 3.4).

$$2^{14,1} \cdot 2^{9,4} \cdot 2^{47,1} \cdot 2^{265,15} \cdot 2^{42,85} \approx 2^{378,6} \tag{3.4}$$

3.3 Schlüssel

In diesem Abschnitt wird die Schlüsselraumgröße und dessen Hintergrund erläutert. Die Geheimhaltung der Maschine wurde priorisiert, obwohl die übliche Sicherheit nur vom jeweiligen Schlüssel abhängig sein sollte. Die Typex-Maschine wird in ihren Schlüsselbestandteilen aufgelistet und mit entsprechenden Kombinationen für den Schlüssel versehen. Die tatsächliche Schlüsselraumgröße wird vorgestellt. Zur besseren Isolierung werden zunächst die kryptographischen Maßnahmen der Typex erläutert.

Im November 1939 kam bei der Militärführung die Frage auf, ob die Maschine noch sicher sei, wenn sie mit einem kompletten Rotorsatz in fremde Hände gerät. Die GC&CS (Government Code and Cypher School) antwortete, dass die Typex selbst dann sicher sei. Um solch einem Fall vorzubeugen sei es ratsam, einige Unterkategorien (unterschiedliche Rotorsätze) einzuführen, damit ein Satz austauschbar wird und nicht sämtliche Maschinen ausgetauscht werden müssen. Es wurde empfohlen, zehn Rotoren einzusetzen, wobei die eine Hälfte der Rotoren in der Maschine eingesetzt werden sollte und die andere Hälfte in einem Ort mit vergleichbarer Sicherheit verbleiben sollte [40]. Im Februar 1940, als das Heer und die Marine (Royal Navy) begannen, die Typex Mk. II zu verwenden, wurde mit GC&CS eine höhere Sicherheit vereinbart. Daraus entstand, dass für jede Militärsparte ein eigener Rotorsatz angefertigt wird und dass der jeweilige Bereich einen Rotorsatz für übergreifende Kommunikation erhält. Jeder Rotorsatz sollte mindestens sieben, bald 10 Rotoren beinhalten. Es sollten keine Informationen nach außen gelangen, z. B. welche Art von Rotoren für eine bestimmte Verschlüsselung verwendet wurde. Ein sicheres und standardisiertes System sollte implementiert werden, wobei die einzelnen Bereiche selbstverantwortlich die Rotorsätze häufig wechselten und der Kommunikationsaustausch nur im Bedarfsfall auftreten sollte [41].

Im Juni 1940 äußerte GC&CS den nachweislich korrekten Verdacht, dass eine militärische Typex gekapert wurde und dies den Datenverkehr gefährden könnte. Es wurde angewiesen, dass die Rotoren zwei Mal am Tag mit ihrer Initialstellung geändert werden. Diese sorgte für einen stark alternierenden Datenverkehr. Der Gegner könnte mit keiner Maschine die Geheimtexte nachvollziehen. Abgesehen davon müsste der Gegner die korrekte Initialstellung finden [42].

Nach diesem Vorfall wurden bald 5 Rotoren aus 10 verfügbaren gewählt. Diese Sicherheit war nicht vergleichbar mit der deutschen „Luftwaffe oder Marine“ Versionen, da die nur drei Rotoren aus 5 bzw. 8 aussuchten (60 bzw. 336 vers. Rotoreinstellungen). Die deutschen U-Boote wählten 1942 vier Rotore, drei aus 8 und eine aus drei Rotoren, welche 1.008 ($= 8 \cdot 7 \cdot 6 \cdot 3$) Möglichkeiten einräumten. Ab 1943 waren es 1.344 ($= 8 \cdot 7 \cdot 6 \cdot 2 \cdot 2$) Möglichkeiten. Die endgültige Version der Typex hatte 14 separate Rotoren, womit aus 7.687.680 ($= 28 \cdot 26 \cdot 24 \cdot 22 \cdot 20$) unterschiedlichen Rotorreihenfolgen gewählt werden konnte. Diese Sicherheit war möglich, da die Briten die Maschine mit Sorgfalt nutzten. Eine Fehlbedienung ließ den Sicherheitsraum deutlich verkleinern. Der britische Funkverkehr war ebenfalls

unterteilt und es war nur eine benötigte Anzahl von Maschinen im Umlauf. 1942 begann die Royal Air Force erst damit ihren Verkehr in zwei Bereiche zu unterteilen. 1942 wurden landesweit drei Hauptkategorien eingeführt.

Die Royal Air Force Maschinen Mk. I und II benutze bis 1941 die gleichen Schlüssel, als dann vier Schlüssel für den weltweiten Verkehr eingeführt wurden: „Allgemein“, „England“, „Mittlerer Osten“ und „Königshaus“. Ein Betreiber hatte den jeweiligen Schlüssel für seinen Bereich, plus den Allgemeinen. Wenn eine Nachricht in mehr als einen gezielten Bereich gesendet wurde, verwendete der Betreiber den allgemeinen Schlüssel. Später wurden weitere Schlüssel eingeführt, für den Mittleren Osten, Indien (2 Schl.), Australien (2 Schl.), Kanada und den Fährenverkehr. Ende 1944 hatte die britische Luftwaffe (Royal Air Force) mindestens 30 verschiedene Schlüssel für die Typex. Überraschend ist, dass diese 30 Schlüssel bei den neueren Maschinen nur 10 unterschiedliche Steckerbrettverbindungen hatten und manche Maschinen an dem gleichen Kommunikationsnetz noch nicht mal ein Steckbrett hatten. Die Netze wurden vom Operator oder später aus einer Liste bestimmt [2, 43]. Diese Sicherheitsstandards machten die Ent- und Verschlüsselung viel langsamer. Die gesendeten Daten waren immer öfters fehlerhaft, welche dann Korrekturen erforderten. Aber die Typex war dennoch schneller als die vorher genutzten Codebücher. Aber zum Beispiel dauerte es über 13 Stunden einen Bericht von Isaiah Berlin an das britische Auswärtige Amt über das Kriegsverhalten aus Washington zu entschlüsseln.

3.3.1 Schlüsselraumgröße

Die Schlüsselraumgröße bei der Typex basiert auf drei Teilen:

(a) Die Grundstellung bestand aus drei Rotor- und zwei Statorpositionen über 26 Buchstaben. Diese Eigenschaft stärkte die Typex mit $26^5 = 11.881.376$ Möglichkeiten. Der Ausdruck lässt sich mit dem binären Logarithmus als $2^{23,5}$ darstellen.

(b) Die Auswahl der Rotoren und Statoren erfolgte aus dem gleichen Walzensatz. Die bekannteste Typex wählte 5 Rotoren/Statoren aus zehn Verfügbaren, welche in beide Richtungen einsetzbar waren (20 Verdrahtungen). Formel 3.5 zeigt die Anzahl der unterschiedlichen Rotorreihenfolgen.

$$20 \cdot 18 \cdot 16 \cdot 14 \cdot 12 = 967.680 \approx 2^{19,88} \tag{3.5}$$

Es gab zwei Umkehrwalzen. Daher wird das Ergebnis aus Formel 3.5 verdoppelt. Formel 3.6 zeigt die tatsächliche Anzahl einschließlich der Wahl der Umkehrwalze.

$$967.680 \cdot 2 = 1.935.360 \approx 2^{20,88} \tag{3.6}$$

(c) Die Auswahl der gekerbten Ringe war separat auswählbar. Die Verdrahtungen wurden als Einlage mit dem Kerbenring verbunden. Die Ringe können zwischen einem und neun Kerben haben. Sie werden aus 26 möglichen Stellungen ausgesucht. Formel 3.7 berechnet die Auswirkung der Ringe.

$$\frac{26!}{9!\ (26-9!)} \approx 2^{21,58} \tag{3.7}$$

In der hier beschriebenen Konfiguration (Formel 3.8) verfügt die Typex über einen Schlüsselraum von *65 Bit*.

$$2^{23,5} \cdot 2^{20,88} \cdot 2^{21,58} \approx 2^{65,96} \tag{3.8}$$

Schlüsselraumgröße bei unbekannter Verdrahtung Es wurde versucht, die Rotorverdrahtung geheim zu halten. Sie war kein Bestandteil des Schlüssels, aber trotzdem wurde sie der Geheimhaltung unterzogen. Im Allgemeinen sollte keine Maschine in gegnerische Hände fallen. Sofern die Rotor-, Stator und Umkehrwalzenverdrahtung nicht bekannt waren, stieg die Schlüsselraumgröße signifikant an. Im Folgenden wird die Variante von Chang, Low und Stamp verfolgt da sie einen Schlüsselraum in Abhängigkeit zur Verdrahtung [38] berechnet. Unter dieser Annahme setzt sich der theoretische Schlüsselraum aus drei Komponenten zusammen:

(a) Die Auswahl der Rotoren und Statoren erfolgte abhängig zum Schlüssel. Die Berechnung umfasst alle möglichen Verdrahtungen, wobei jede Grundstellung sich einer Verdrahtung zuordnen lässt. Aus diesem Grund ist die Grundstellung bereits in der Menge aller Verdrahtungen bereits vorhanden. Es verbleibt die Anzahl aller Rotorreihenfolgen zu berechnen.

Die Rotoren erzeugen eine Permutation über 26 Buchstaben des Alphabets. Ein Rotor ist in einer der 26 möglichen Grundstellungen (A - Z) positioniert. Die Statoren verbleiben in ihrer Position und verhalten sich kryptographisch wie ein zusammengefasster Rotor [38]. Wenn ein A als Eingabe beim ersten Stator anliegt und die Ausgabe beim zweiten Stator T ist, dann wird diese Abbildung bei jedem $A \rightarrow T$ auftreten. Ist die Verdrahtung der Rotoren nicht bekannt, zeigt Formel 3.9 die möglichen Rotorreihenfolgen und deren Grundeinstellung.

$$(26!)^4 \approx 2^{353,52} \tag{3.9}$$

(b) Die Typex-Umkehrwalze verhält sich technisch äquivalent zur Enigma-Umkehrwalze. Sie verursacht eine Permutation über 26 Buchstaben mit der Eingrenzung, dass kein Buchstabe auf sich selbst abbilden kann, da es sonst gäbe es einen elektrischen Kurzschluss im Stromfluss gäbe. Diese Annahme macht die Umkehrwalze äquivalent zu einem Steckerbrett mit 13 Verdrahtungen. Aus Tabelle 3.2 lässt sich das Ergebnis für Formel 3.10 entnehmen.

$$F(13) \approx 2^{42,85} \tag{3.10}$$

(c) Die Auswahl der gekerbten Ringe beläuft sich auf $2^{21,58}$ Kombinationen (gemäß Formel 3.7).

In Summe verfügt die Typex (Formel 3.11) über einen theoretischen Schlüsselraum von *417 Bit*.

$$2^{353,52} \cdot 2^{42,85} \cdot 2^{21,58} \approx 2^{417,95} \tag{3.11}$$

3.3.2 Unizitätslänge

Die Unizitätslänge ist die theoretische Mindestlänge, die ein Geheimtext haben muss, damit eine eindeutige Dechiffrierung möglich wird. Die Unizitätslänge hängt von der Struktur des Klartextes und von der Schlüssellänge ab. Klartext und Geheimtext werden zueinander in Beziehung gesetzt, so dass sich die Frage nach der Längenkongruenz stellt. Die Unizitätslänge wird auf eine bestimmte Anzahl von Zeichen im Klartext bezogen, da sie sich oft auf die Beurteilung der Effektivität eines Brute-Force Angriffs bezieht [44]. Nach Shannon sind Kryptosysteme ohne Unizitätslänge ideal sicher [45]. In symmetrischen Kryptosystemen wird die Unizitätslänge U auch definiert als die *Entropie* $H(k)$ des Kryptosystems dividiert durch die Redundanz der Sprache D.

$H(k)$ ist der Logarithmus der möglichen Schlüsselanzahl beim Kryptosystem. D ist die Redundanz bei Klartexten in (Bits/Zeichen) [45]. Die Redundanz ist eine wichtige Eigenschaft einer Sprache, da es ein Maß der Kompressionsrate innerhalb einer Sprache darstellt. Diese beträgt laut Shannon bei der englischen Sprache einen Wert von 3,2 [45].

Für die Typex bedeutet dies, dass ein Ciphertext-Only Angriff unabhängig von seiner Effizienz nur dann sinnvoll sein kann (Shannon), wenn die Länge des bekannten verschlüsselten Textes größer als 21 ist. Die Berechnung lässt sich aus Formel 3.12 entnehmen.

$$U = \frac{H(k)}{D} = \frac{65\ Bits}{3,2\ \frac{Bits}{Zeichen}} = 20\ Zeichen \tag{3.12}$$

Die Formel 3.13 zeigt die Mindestlänge für einen potenziell erfolgreichen Angriff unter Berücksichtigung, dass die Verdrahtungen von Rotor, Stator und Umkehrwalze nicht bekannt sind.

$$U = \frac{H(k)}{D} = \frac{417\ Bits}{3,2\ \frac{Bits}{Zeichen}} = 130\ Zeichen \tag{3.13}$$

4 Kryptoanalyse

Dieses Kapitel beschreibt die Kryptoanalyse bei der Typex-Maschine. Die Maschine verwendet kryptographische Prinzipien (polyalphabetische Substitution) aus dem Grundlagen Kapitel. Diese haben deutliche Schwächen, unter anderem bei der Buchstabenverteilung. Diese Schwächen werden in den folgenden Kapiteln zuerst erläutert und später mit den genannten Kryptoanalyse-Methoden angegriffen. Ein Angriff benötigt ein Konzept, dass den Erfolgsstand bewertet. Dies geschieht mit statischen Metriken in unterschiedlichen Ansätzen.

Die Typex selbst wurde während den Einsatzjahren nicht gebrochen, aber es gab vereinzelt gute Möglichkeiten einen Angriff zu platzieren. Im August 1943 wurden deutsche Geheimdienstmitarbeiter verhaftet und behaupteten, dass Typex-Nachrichten in Tunis und Berlin während des tunesischen Feldzuges geknackt worden seien. Ferner seien manche Entschlüsselungen sogar innerhalb von 12 Stunden vollzogen worden. Angeblich hatten die Deutschen eine Typex-Maschine in Tobruk, (1942) gekapert und versuchten, unter größten Anstrengungen, einen mechanischen Angriff auszuführen. Obwohl diese Berichte lückenhaft waren, wurde dies den Deutschen zugetraut. Die Sicherheit der Typex wurde im generellen Sinne hinterfragt. Im Dezember zeigten umfassende Untersuchungen mit neuartigen Angriffsmethoden, dass die Typex bei Verwendung des gleichen Schlüssels mit langen Texten, „angreifbar“ war [46].

4.1 Schwächen

Die Schwächen von historischen Kryptosystemen wurden an die Enigma und Typex vererbt. Die Maschinen sind sehr anfällig gegen eine Häufigkeitsanalyse. Diese Eigenschaft lässt sich bei den Rotor-Chiffriermaschinen feststellen und letztlich erfolgreich zu einem Angriff führen. Die Schwächen reichten von den Rotoren bis zum Rotationsmechanismus.

Die Typex hatte drei Rotoren und zwei Statoren aus einer Sammlung von zehn zu wählen. Die starre Verdrahtung der Rotoren und die teilweise begrenzte Auswahl reduzierte dramatisch die Anzahl möglicher Verdrahtungen bei einem Rotor bzw. bei der Einlage für den Rotor.

Tatsächlich gelang polnischen Mathematikern die Rekonstruktion der drei Rotoren der Enigma. Dies passierte kurz nachdem ihnen der erste erfolgreiche Angriff auf die Enigma gelungen war. Das Suchen der Verdrahtungen bei den 10 Rotoren musste nur einmal erfolgreich verlaufen, da sich diese Werte durch die starre

Verdrahtung nicht änderten. Bei jedem zukünftigen Angriff kann dies als wichtige Information definiert werden, da sich der Suchraum um den Faktor 26! je Rotor reduziert. Des Weiteren reflektiert der umgedrehte Rotor nur eine inverse Verdrahtung und lässt sich daher auch berechnen. Es verbleiben 5 Rotoren in der korrekten Reihenfolge zu wählen.

Seit die Standart-Typex mit 5 Rotoren in einem Satz ausgeliefert wurde und der Bediener immer nur einen Rotor zur selben Zeit nutzen konnte, ließen sich Kombinationen wie „II-II-II-II-II“ aus der Auswahl eliminieren. Der Kryptoanalytiker muss nur die Auswahl und die Reihenfolge der Rotoren herausfinden. Es werden 3.840 ($= 10 \cdot 8 \cdot 6 \cdot 4 \cdot 2$) Möglichkeiten eingeräumt. Eine Enigma hatte drei aus fünf Rotoren mit 60 ($= 5 \cdot 4 \cdot 3$) Reihenfolgen. Die frühere Enigma-Version, welche die polnischen Mathematiker zwischen 1930 und 1932 angriffen, hatte nur drei Rotoren mit sechs ($= 3 \cdot 2 \cdot 1$) möglichen Reihenfolgen.

Diese zehn universellen Rotoren wurden mit Hilfe der zusätzlichen Umkehrwalze nochmals durchlaufen. Die Erfindung der steckbaren Umkehrwalze mit selbst definierbarer Verdrahtung entfaltete erst allein das volle Potenzial dieses Bauteils. Die Umkehrwalze konnte direkt am Arbeitsplatz, sofort neu verdrahtet werden. Diese Idee ließ zahlreiche neue Verdrahtungen entstehen und erhöhte gleichzeitig die möglichen Kombinationen um ein Vielfaches.

Die Typex hatte eine weitere Schwachstelle bei der Rotorarchitektur, welche keine Auswirkung auf die statistische Analyse hatte, aber die Maschine angreifbarer machte: Die reguläre Drehung der Rotoren. All diese Rotoren hatten eine voraussehbare Rotationsperiode. Sie drehten sich in nur eine Richtung und tätigten einen Schritt pro Eingabe, zwei Schritte bei einer vollen Drehung. Die reine Vorwärtsdrehung half den Briten beim regelmäßigen Lesen von Enigma-Telegrammen. Kannte der Kryptoanalytiker die Reihenfolge, konnte er durch Zählen der Buchstaben auf die Fortschritte der Rotoren schließen. Dieser reguläre Verlauf ließ den Kryptoanalytiker zwangsläufig jeden Teil des Geheimtextes lesen. Die höhere Kerbenanzahl in einem Rotor änderte das Muster des benachbarten Rotors. Mit den zusätzlichen Kerben bewegte sich der langsame Rotor deutlich häufiger, als nach nur 26 Buchstaben (volle Umdrehung). Aber auch dieser Lösungsansatz führte zu einem einzigartigen Muster innerhalb des physikalischen Mechanismus. Die vergleichbare Enigma hatte eine gewisse Begrenzung. Nach 26 Drehungen vom mittleren Rotor rotierten alle drei Rotoren beim nächsten Schritt gemeinsam. Der Schnelle rotierte bei einer Kerbe den mittleren Rotor an, der Mittlere stieß bei einer Kerbe den Langsamen an und beim folgenden Buchstaben drehten sich alle gemeinsam. Dieser doppelte Schritt schob den schnellen Rotor um zwei Buchstaben weiter anstatt nur um einen Buchstaben. Dies reduzierte die Periode einer vollen Drehung (17.576) um einen Schritt auf 16.900 Möglichkeiten. Die Verwendung von zwei Kerben bei zwei Rotoren lässt die Möglichkeiten auf 16.224 ($= 26 \cdot 25 \cdot 25 - 26$) sinken.

Des Weiteren war eine weitere Anomalie der Typex, ähnlich der Enigma, zu erkennen. Ein Buchstabe konnte nicht auf sich selbst abgebildet werden. Diese Eigenschaft sollte bei der Enigma das Ausgeben von Klartext im Geheimtext vermeiden,

aber verfehlte vollkommen das Ziel von Zufälligkeit beim Verschlüsselungsprozess. Allerdings konnten unter dieser Randbedingung die Standardkryptotechniken zur Schlüsselfindung angewendet werden. Sobald der Kryptoanalytiker über diese Information verfügte, konnten Regeln aufgestellt werden [2]. (z.B. wenn ein „e“ im Geheimtext auftauchte, konnte der Klartext an dieser Stelle kein „e“ enthalten). Dieses Wissen befähigte den Kryptoanalytiker die statistischen Methoden zu konkretisieren und anzuwenden. Damit wurden unkorrekte Rotorreihenfolgen ausgeschlossen, wenn die Kombination zu einer Selbstabbildung führte.

Die starren Verdrahtungen und die mechanischen Begrenzungen von der Maschine reduzierten den theoretischen Schlüsselraum um ein Vielfaches, wobei einige Thesen nicht in einem Ernstfall Anwendung fanden. Die möglichen Kombinationen wirkten zur damaligen Zeit abschreckend. Die Analyse zeigte aber, dass viele Einschränkungen den Schlüsselraum unbewusst reduzierten. Um so einschränkender wäre die fehlerhafte Bedienung der Maschine, wie das ständige Senden des Tagesschlüssels an derselben Stelle im Telegramm. Die Fehlbedienung erlaubte es den Briten das Enigma-System einfacher zu „knacken“ und führte zu größeren Sicherheitsbedenken als die Sicherheitslücken bei der Maschine.

4.2 Angriffe

Die Angriffe sind nach den verfügbaren Informationen über ein Kryptosystem in Kategorien aufgeteilt. Der Brute-Force Angriff benötigt keinerlei Informationen. Der Ciphertext-Only verwendet lediglich abgehörte Geheimtexte. Der Known-Plaintext Angriff benutzt abgehörten Geheimtext und korrespondierenden Klartext. Für jede Angriffsart wurde eine Methode entwickelt, die die Typex angreifbar macht. Der Ciphertext-Only Angriff wurde programmiertechnisch im Rahmen dieser Arbeit implementiert.

Der Erfolg des Einsatzes statistischer Analyse-Verfahren hängt weitgehend von der Struktur des betrachteten Kryptosystems ab. Zur Verwendung eines Kryptosystems gehört aber nicht nur der eigentliche Verschlüsselungsalgorithmus, sondern ebenso die Art der Schlüsselerzeugung und -verteilung sowie die praktische Implementierung des Gesamtsystems. Die Bedeutung einer formalen Spezifikation all dieser Prozesse ist entscheidend für die Untersuchung der Sicherheit eines Systems.

Rotor-Chiffriermaschinen bildeten das Gerüst der geheimen militärischen Kommunikation während des Zweiten Weltkrieges. Die Konstrukteure waren damals überzeugt, nicht einmal der Besitz einer Maschine werde einem feindlichen Kryptoanalytiker zur Dechiffrierung des laufenden Nachrichtenverkehrs nützen. Es wird gezeigt, dass eine Rotor-Chiffriermaschine unter einem Known-Plaintext bzw. Ciphertext-Only Angriff unsicher ist.

4.2.1 Brute-Force Angriff

Diese Art des Angriffs wird im Allgemeinen vermieden, wenn bessere Alternativen für das Brechen einer Chiffre existieren. Er lässt sich aber immer anwenden, obwohl die Laufzeit bei großen Schlüsselräumen solch einen Angriff hinfällig macht (Kapitel 2.3.1). Abhängig zur Kostenfunktion ist nach der Terminierung ein meist korrektes Ergebnis verfügbar. Da für die Typex bessere Methoden als die Angriffe Ciphertext-Only und Known-Plaintext sich finden lassen, werden diese Angriffsformen nur auf einen reduzierten Schlüsselraum angewendet. Der Ciphertext-Only Angriff wird in Phase Eins und Zwei auf unterschiedliche, aber deutlich reduzierte Schlüsselräume ausgeführt. Diese Reduzierung erfolgt über mathematische Analysen und führt dazu, dass die Laufzeit des spezifischen Brute-Force Angriffs signifikant akzeptabler wird.

4.2.2 Ciphertext-Only Angriff

Die Angriffe zur damaligen Zeit basierten auf der Technik von Known-Plaintext (crib-Methode). Es gilt bei alten Chiffren, dass mit steigender Rechenleistung eine durchaus höherwertige Analyse durchführbar ist. Das Lesen verschlüsselter Nachrichten über einen Tag hinweg erfordert das Aufdecken der verwendeten Rotorstellung, Reihenfolge und Grundstellung. Die hier beschriebene Technik (mit abgefangenen Geheimtext auf den Schlüssel zu schlussfolgern) war bereits damals in der Theorie bekannt [9]. Die notwendige Rechenleistung war allerdings noch nicht gegeben.

Dieser Angriff wird hier theoretisch beschrieben und in Kapitel 5 technisch implementiert. Der Angriff wird in zwei Phasen strukturiert und anlehnend an den Artikel von James J. Gillogly [3] entworfen. Die erste Phase probiert alle möglichen Typex-Grundeinstellungen und beinhaltet die Auswahl der Rotoren, Grundeinstellung und Rotorrichtung. Jede Grundeinstellung, die zu einer Menge von semantisch und syntaktisch korrekten Texten (reelle Sprache) führt, wird Modell genannt und in einer Bestenliste gespeichert. Die restlichen Interpretationen, die den Automaten Typex nicht erfüllen, werden als „zufällig“ definiert und verworfen. Für jedes Modell aus der ersten Phase wird eine zweite Phase benötigt. Sie versucht alle möglichen Ringstellungen auf jedem Modell der ersten Phase aus. Es ist erforderlich, dass jedes mögliche Ergebnis mit Hilfe einer Spracherkennungsmetrik analysiert wird. Jedes Ergebnis einer Interpretation wird mit einer Metrik bewertet, welches die Distanz zum Klartext berechnet. Bei dieser Analyse wird nicht nur der oberste Kandidat herangezogen, sondern einige der besten Kandidaten. Diese Änderung erweitert den Fokus und erlaubt auch weiteren Kandidaten den Aufstieg in der Bestenliste.

Das Aufteilen des Angriffs in zwei Abläufe reduziert den Rechenaufwand. Die erste Phase gibt eine signifikante Tendenz aus und dient zugleich als deutlich reduzierte Eingabe für die zweite Phase. Die nicht benötigten Interpretationen werden

nach der ersten Phase verworfen. Diese Struktur ist deutlich effizienter als ein Brute-Force Angriff im Ganzen auf das System. Für beide Phasen werden folgende Annahmen getroffen:

- Es sind *entweder* komplett die Rotoren *oder* die Statoren mit Einstellung und Positionen im Detail gegeben.
- Es gibt 5 Rotoren, die in beide Richtungen einsetzbar sind (10 unabhängige Verdrahtungen).
- Die jeweilige Rotorverdrahtung ist bekannt analog zur Enigma mit jeweils einer Kerbe.
- Es gibt zwei Umkehrwalzen (Reflektoren).

Unter Berücksichtigung der o. g. Bedingung beläuft sich der Schlüsselraum beim *Suchlauf nach den Rotoren* auf:

$$48 \cdot 26^3 \cdot 26^3 \cdot 2 = 2^{34,79} \tag{4.1}$$

wobei Tabelle 4.1 gilt.

$48 = 6 \cdot 4 \cdot 2$	verbleibende Möglichkeiten abzüglich der zwei Statoren – reduziert sich in zweier Schritten, da ein Rotor in beide Richtungen einsetzbar ist
26^3	drei Rotoren mit je 26 Grundstellungen
26^3	drei Rotoren mit je 26 Ringstellungen
2	Auswahl zwischen zwei Umkehrwalzen mit fester Position

Tabelle 4.1: Phase 1 – Der Schlüsselraum beim Suchlauf nach den Rotoren

Der Schlüsselraum beim *Suchlauf nach den Statoren* verhält sich wie folgt:

$$8 \cdot 26^2 \cdot 26^2 \cdot 2 = 2^{22,80} \tag{4.2}$$

wobei Tabelle 4.2 gilt.

$8 = 4 \cdot 2$	verbleibende Möglichkeiten abzüglich der drei Rotoren – beide Richtungen möglich
26^2	zwei Statoren mit je 26 Grundstellungen
26^2	zwei Statoren mit je 26 Ringstellungen
2	Auswahl zwischen zwei Umkehrwalzen mit fester Position

Tabelle 4.2: Phase 1 – Der Schlüsselraum beim Suchlauf nach den Statoren

4.2.2.1 Erste Phase

Das Ziel in Phase Eins ist, alle zufälligen Interpretationen zu verwerfen und nur eine Liste mit Modellen zu erhalten. Die erste Phase probiert jede Kombination im Schlüsselraum aus. Dies beinhaltet die Rotorauswahl, Rotorreihenfolge und Grundstellung. Phase 1 wird isoliert zur zweiten Phase ausgeführt, indem Geheimtext für jede Schlüsselkombination eingegeben und der korrespondierende Klartext auf Konsistenz überprüft wird. Die Kombination, die korrekten Text dechiffriert, nennt sich Modell.

Die statistische Annäherung für die Rotor-Mechanismen zu verwenden, hat sich bewährt. Sie bedient sich der Eigenschaft, dass ein verschlüsselter Text mit nicht zufälliger Verteilung wieder in einen nicht zufällig verteilten Chiffriertext resultiert (siehe Abbildung 4.3). Die Häufigkeit der Buchstaben ist sogar beim Geheimtext von einem erkennbaren Verteilungsmuster geprägt.

Gillogly empfiehlt den Koinzidenzindex (Formel 4.3)für die aktuelle Phase [3]. Um diese Metrik zu berechnen, wird die Häufigkeit h_i jedes Buchstabens i über dem dechiffrierten Klartext der Länge n berechnet. Die Metrik setzt sich wie folgt zusammen:

$$IC = \sum_{i=A}^{Z} \frac{h_i(h_i - 1)}{n(n-1)} \tag{4.3}$$

Schlüsselraum der ersten Phase Die Menge der möglichen Schlüsselkombinationen, ohne Ringstellungen, definiert den Schlüsselraum der ersten Phase. Sie beinhaltet daher folgende Eigenschaften:

- Auswahl von 3 Rotoren von insgesamt 5 Rotoren
- Auswahl einer Umkehrwalze von 2 verfügbaren.
- Die Laufrichtung bestimmen (vorwärts oder rückwärts)
- Auswahl eines Buchstabens aus 26 möglichen auf jedem Rotor

Der zu durchsuchende Schlüsselraum von $2^{20,67}$ ergibt sich aus der Summe dieser Faktoren.

$$48 \cdot 26^3 \cdot 2 = 2^{20,67} \tag{4.4}$$

Iterieren durch die erste Phase In dieser Phase wird die Lage und Position der Walzen ermittelt. Dafür wird die Ringstellung auf „AAA“ fixiert. Dann wird für jede Einstellungsmöglichkeit der IC berechnet. Zwei zufällige Texte weisen einen IC von ungefähr $0,0385$ auf, während deutscher Text einen IC von etwa $0,0762$ und ein englischer Text einen IC von etwa $0,0667$ hat. Für jede Eingabe in die Maschine wird sich der schnelle Rotor um eine Position drehen. Beim Erreichen einer Kerbe dreht sich dessen linker Nachbar. Die Einstellungen für die höchsten

erzielten Werte bei Berechnung des IC werden gespeichert. Diese Einstellungen werden dann wiederum als Ausgangsbasis für die nächste Phase benutzt.

Friedman fand zusätzliche Möglichkeiten, die es ermöglichen, den zu erwartenden IC bei richtiger Lage und Position zu berechnen [9]. Sobald der Strom die Maschine durchlaufen hat können zwei Unterscheidungen getroffen werden: einige Buchstaben werden vertauscht, einige werden nicht vertauscht. Nichtsdestotrotz wird die Statistik des IC davon nicht beeinflusst. Gilloglys Annahme verspricht bemerkenswerten Erfolg mit der gewählten Metrik und bestätigt eine erfolgreiche Identifikation der korrekten Einstellung. Der Ablauf ist folgendermaßen vorgesehen:

(1) Die Ringstellung wird auf Initialwerte gesetzt. Dann werden alle möglichen Grundstellungen (z. B. „AAA“ - „ZZZ“ mit $26^3 = 17.576$ Möglichkeiten) getestet, indem permanent derselbe Geheimtext mit unterschiedlichen Einstellungen dechiffriert wird. Zusätzlich werden alle möglichen Rotorreihenfolgen ($6 \cdot 4 \cdot 2 = 48$ Möglichkeiten) sowie zwei Umkehrwalzen ausprobiert. Dieses Vorgehen ist keine kostspielige Funktion für ein Computerprogramm.

(2) Der IC wird für jede Schlüsselkombination berechnet und dem Wert nach geordnet. Mit dieser Methode lassen sich reelle Texte von zufälligen Texten abspalten. Somit erhält man auch eine separate Menge mit potenziellen Modellen.

(3) Die berechneten Kandidaten werden nun anhand ihrer Listenposition (Tabelle 4.3) beurteilt. Die negativen Werte in der Spalte „Reihenfolge“ bedeuten, dass der Rotor umgedreht eingesetzt ist. Der Pfeil indiziert das beste Ergebnis.

Reihenfolge			IC	Grundstellung			Umkehrwalze	
4	-2	1	0,01967	O	W	J	A	←
-1	-2	4	0,02383	Z	A	U	B	
-2	-4	2	0,02360	Y	S	H	B	
-1	4	-2	0,02379	N	M	O	A	
-1	-2	4	0,02363	L	J	J	B	

Tabelle 4.3: Ergebnis nach Phase 1

Mit diesem Vorgehen wurde die Anzahl der zufälligen Kandidaten von $2^{20,67}$ auf 5 entscheidend reduziert. Der Datensatz mit der Reihenfolge $4, -2, 1$ ist etwas höher als die Restlichen der Bestenliste. Die Kostenfunktion hat die gesuchte Einstellung bereits identifiziert. Das Ergebnis ist allerdings noch nicht zufriedenstellend. Dieser Schritt scheitert bei sehr kurzen Nachrichten, denn in diesen kurzen Phrasen ist eine geringe Redundanz vorhanden. Somit ist die Häufigkeitsanalyse mittels IC nicht zielorientiert. Da die Ringeinstellungen den Geheimtext noch erheblich verschieben und permutieren.

4.2.2.2 Zweite Phase

In Phase Eins wurden die Einstellungsmöglichkeiten um einige Faktoren eingeschränkt. Nun fehlt noch, die richtige Ringstellung zu ermitteln. Die zweite Phase probiert jede Kombination von Ringstellungen und verschlüsselt den jeweiligen Geheimtext. Ist die resultierende Zeichenkette äquivalent zum Klartext, so wurden die korrekten Ringstellungen gefunden. Liefert keine Einstellung konsistente Ergebnisse, so hat kein Modell einen ausreichenden Platz gemäß Kostenfunktion in der ersten Phase erhalten.

Schlüsselraum der zweiten Phase In dieser Phase müssen die Ringstellungen auf den drei Rotoren, in Abhängigkeit zur berechneten Grundstellung, ermittelt werden. Jeder Rotor kann über einen Ring, mit jeweils 26 Buchstaben aus dem lateinischen Alphabet, weiter modifiziert werden. Für Lösungssuche erschließen sich folgende Verfahren:

(a) Paralleler Suchlauf: Die unbekannten Ringstellungen werden zur selben Zeit gesucht.

  (i) Die drei unbekannten Ringstellungen werden gleichzeitig gesucht. Bei einer parallelen Suche der Ringstellungen würden $26^3 = 17.576$ verschiedene Ringstellungen für jede Schlüsselkombination ausprobiert werden.

  (ii) Nur zwei unbekannte Ringstellungen werden zugleich gesucht. Gillogly versichert, dass die Ringstellung des langsamen bzw. linken Rotors keine Auswirkung auf die Verschlüsselung hat [3]. Daraus reduziert sich der Suchraum um einen Faktor auf $26^2 = 676$ mögliche Ringstellungen.

(b) Sequentieller Suchlauf: Die unbekannten Ringstellungen werden nacheinander gesucht. Jede Unbekannte wird einzeln gesucht. Durch die sequentielle Abarbeitung lässt sich der Suchraum auf $26 + 26 + 26 = 78$ Möglichkeiten reduzieren.

Die Ringstellung des linken Rotors kann bei kurzen Texten vernachlässigt werden, da diese die Walzenfortbewegung nicht beeinflusst. Ist die Nachricht kürzer als 676 Buchstaben, dreht sich der dritte Rotor mit nur sehr geringer Wahrscheinlichkeit. Das hier beschriebene Verfahren verwendet die sequentielle Suche der Ringstellung, die sich vom schnellen Rotor zum langsamen Rotor vorarbeitet. Dieses Verfahren ist nochmals schneller als das mathematische Ausklammern des linken Rotors.

Iterieren durch die zweite Phase In Phase Eins wurden alle Ringstellungen auf den Ausgangswert „AAA“ gesetzt. Wenn die gewählte Ringeinstellung ähnlich ist, kann diese falsche Zuweisung durch das Rotieren des Ringes wieder behoben werden. Dies kann aber immer nur unter der Berücksichtigung geschehen, dass die Rotor-Grundeinstellung simultan angepasst wird.

Wenn die voreingestellte Ringstellung nicht gleich mit der Gesuchten ist, wird die Metrik keine akzeptablen Werte berechnen. War die Vermutung einer Ringstellung falsch, so wäre mindestens die Hälfte des Klartextes eine unlesbare Zeichenfolge. So wird die Dechiffrierung deutlich erschwert.

Der IC ist sehr effektiv in Phase Eins. Nichtsdestotrotz haben Messungen ergeben, dass die Suche mit verschiedenen Metriken erfolgreicher für die Ringstellung verläuft. Zum Auffinden der korrekten Ringstellung werden Tetragramme (Quadgramme), Sinkov Metrik und der IC verwendet.

Zuerst wird der rechte Rotor geprüft. Es werden für alle Ringstellungen neue Konsistenzen mit der Kostenfunktion berechnet, genau wie in der vorangegangenen Phase. Zu beachten ist, dass bei diesem Vorgang die Grundstellung ständig angepasst wird, damit die Verdrahtung des Rotors gleich bleibt. Der Pfeil zeigt das beste Ergebnis. In Tabelle 4.4 sind die Suchergebnisse für die Ringstellungen des schnellen Rotors gezeigt. Bei diesem Durchlauf wurde die Ringstellung für den schnellen Rotor gesucht. Die Positionen des langsamen und mittleren Rotors verbleiben unverändert.

Reihenfolge	Tetragramme	Grundstellung	Ringe	Umkehrwalze	
4 -2 1	7.726,27257	O W Q	A A H	A	←
4 -2 1	7.972,91353	O W P	A A G	A	
4 -2 1	8.164,58071	O W O	A A F	A	
4 -2 1	8.214,53135	O W N	A A E	A	
4 -2 1	8.331,40135	O W M	A A D	A	

Tabelle 4.4: Fortschritt bei schnellen Rotor

Die Ringstellung, die den besten Wert für die Kostenfunktion erzielt, wird für den rechten (schnellen) Rotor fixiert. Mit der gefundenen Position werden alle Ringstellungen für den mittleren Rotor ausprobiert. Daher variiert nur die Ring- und Grundstellung beim mittleren Rotor. Die Positionen beim langsamen und schnellen Rotor bleiben unverändert (sequentielle Suche). In Tabelle 4.5 sind die besten Ergebnisse für den mittleren Rotor zu sehen.

Reihenfolge	Sinkov	Grundstellung	Ringe	Umkehrwalze	
4 -2 1	84.819,21014	O Y Q	A C H	A	←
4 -2 1	84.837,05622	O Z Q	A D H	A	
4 -2 1	84.838,88000	O Z P	A D G	A	
4 -2 1	84.839,05480	O X Q	A B H	A	
4 -2 1	84.852,99116	O Y P	A C G	A	

Tabelle 4.5: Fortschritt bei mittleren Rotor

Danach wird wieder der beste Wert eines Kandidaten ausgewählt. Damit sind alle Parameter des Schlüssels bestimmt. Bei Texten, die länger als 676 Buchstaben sind, dreht sich der langsame Rotor (links außen). Im angeführten Beispiel wurde

eine Zeichenkette mit 400 Buchstaben zuverlässig entschlüsselt. In Tabelle 4.6 ist ersichtlich, dass jeder Eintrag dieselbe Konsistenz besitzt. Bei jedem Schlüsselsatz wurde die Nachricht komplett entschlüsselt, da nur der Abstand zwischen Ring- und Grundstellung relevant ist.

Reihenfolge	IC	Grundstellung	Ringe	Umkehrwalze	
4 -2 1	0,003429	O Y Q	A C H	A	←
4 -2 1	0,003429	D Y Q	P C H	A	←
4 -2 1	0,003429	P Y Q	B C H	A	←
4 -2 1	0,003429	F Y Q	R C H	A	←
4 -2 1	0,003429	Q Y Q	C C H	A	←

Tabelle 4.6: Fortschritt bei langsamen Rotor

Die Analyse hat ergeben, dass dieser Ablauf trotz verschobener Ringstellung stattfinden kann. Es wird die Ringstellung zur korrespondierenden Grundstellung geprüft. Zusätzlich wird noch ein Buchstabe vor und nach der aktuellen Position getestet. In den Ergebnistabellen der sequentiellen Suche hat die Grundstellung und die Ringstellung permanent denselben Abstand. Zum Beispiel in der Tabelle 4.6 tritt der Datensatz mit der Grundstellung Q und der Ringstellung H für den schnellen Rotor vermehrt auf. Diese Konfiguration hat einen Abstand von 9 Zeichen ($9 = 16 - 9$ mit $Q = 16, H = 7$). Der Datensatz mit der Grundstellung P und der Ringstellung G hat einen übereinstimmenden Abstand von 9 Zeichen ($9 = 15 - 6$ mit $P = 15, G = 6$). Dieses verdeutlicht, dass nur der Abstand der beiden Schlüsselteile relevant ist. Zur Fehlerdetektion bei einer geringen Verschiebung zwischen der Grundstellung und der korrespondierenden Ringstellung wird der Buchstabe vor und nach der gefundenen Grundstellung geprüft. Bei einem Zeichenabstand von 9 wird die Ringstellung mit dem Abstand von 8 und 10 Zeichen zusätzlich durchgeführt. Dieses Verfahren behebt einen Fehler bei einer einfachen Verschiebung.

4.2.3 Known-Plaintext Angriff

Bei dieser Angriffsart wird auf das Verfahren von Alan Turing zurückgegriffen. Da die Ähnlichkeit zur Enigma offensichtlich ist, kann solch ein Enigma-Angriff auch auf eine Typex-Maschine angewendet werden. In den folgenden Abschnitten wird erläutert, welche Benutzerfehler vorausgesetzt wurden, damit Schwachstellen erkennbar werden.

Bei der Enigma war das deutsche Oberkommando sehr akribisch. Wegen möglichen Problemen wurde auf die übergreifenden Sicherheitseinstellungen geachtet. Die meisten Bediener waren nicht so sorgfältig im Umgang mit der Maschine. Bletchley Park konnte einige „Tagesschlüssel“ erraten, da jeden Tag fester Funkverkehr bestand, wie etwa der tägliche Wetterbericht um die gleiche Uhrzeit. Die

Wetterverhältnisse waren bekannt und so entstand ein Angriff mit Zugriff auf den Klartext und den korrespondierenden Geheimtext (ein „Crib“).

Bei Bletchley Park wurden alle Anstrengungen unternommen, damit die deutsche Wehrmacht gewünschte Nachrichten mit bekanntem Klartext aussendet. Solche Cribs wurden in manchen Fällen durch die Royal Air Force erzwungen, die auf Anfrage von Bletchley Park eine bestimmte Zone verminte. Es war zu erwarten, dass der nachfolgende Funkverkehr über diese gegnerische Aktion berichten würde [47]. Es wurden deutsche Gefangene befragt, wie die Anweisungen zur Bedienung einer Enigma gewesen seien. Eine Anweisung war, dass Zahlen als Wort in den Geheimtext aufgenommen werden. Dies ließ sich nach Analyse des Funkverkehrs auch bestätigen.

Alan Turing fand heraus, dass die Zahl „eins“ die häufigste Zeichenfolge ist. Er kreierte aus dieser Annahme den Eins-Katalog. Dieser Katalog enthielt Wahrscheinlichkeiten für das Auftreten einer „eins“ an einer Stelle im Text unter Berücksichtigung der Rotorstellung, Grund- und Ringstellung.

4.2.3.1 Entschlüsseln einer vereinfachten Typex mit der Turing Bombe

Mit einer systematischen Suche fand die „Bombe“ den Schlüssel (Rotorreihenfolge und Grundstellung) für eine definierte Nachricht heraus. Die Maschine konnte als eine Reihe von vereinfachten Rotormaschinen verstanden werden, die einen potenziell korrekten Schlüssel als Ausgabe produzierte. Die Bombe konnte alle 17.576 Rotor-Grundstellungen der Enigma durchsuchen, Rotorreihenfolge und Rotorwahl mussten allerdings manuell getestet werden. Die Bombe nutzte Trommeln mit derselben Verdrahtung eines Rotors. Eine Trommel hatte keine Kerben und Ringe, da ein Rotor mit Ring- und Grundstellung A äquivalent zu derselben Einstellung mit B $(C, D, \ldots, Z)$ war. Die Turing Bombe bestand aus 36 Trommeln, die in drei Reihen und 12 Spalten angeordnet waren. Eine Trommelreihe funktionierte wie eine Enigma mit drei Rotoren oder wie eine Typex. Die Statoren waren dabei bekannt.

Der gesuchte Schlüssel konnte durch vollständiges Durchsuchen des Schlüsselraums gefunden werden. Die hierbei verwendete Methode basierte auf der Verwendung eines wahrscheinlichen Worts (engl.: crib), dessen Vorkommen im Text erwartet oder zumindest angenommen werden konnte. Ein Crib ist eine Zeichenfolge von Klartext, mit einer 1:1 Verknüpfung zu einem Teil des Geheimtextes.

Aufgrund der bekannten inneren Verdrahtung der Enigma-Schlüsselwalzen und ihrer möglichen Stellungen zueinander konnten die beobachteten oder angenommenen Zusammenhänge zwischen dem vorliegenden Geheimtext und dem wahrscheinlichen Wort des Klartextes nur unter ganz bestimmten Bedingungen und nur bei sehr wenigen Schlüsseln erfüllt sein. Mit Hilfe dieser Methode gelang es, die überwiegende Mehrzahl aller Schlüssel auszuschließen. Ferner wird sichergestellt, den von Funksprüchen korrekten Tagesschlüssel zu finden.

Die Bombe verglich die in der verschlüsselten Nachricht angenommene Textphrase mit dem entsprechenden Geheimtextfragment. Sie probiert mit allen möglichen Schlüsseleinstellungen und Walzenstellungen das Geheimtextfragment zu entschlüsseln. Passte das Ergebnis des Entschlüsselungsversuchs zum angenommenen Crib, entsprach die dazu benutzte Schlüsseleinstellung der Bombe möglicherweise dem gesuchten Tagesschlüssel der Enigma. Dabei noch auftretende „Fehltreffer“, die aufgrund der Kürze des Cribs durchaus möglich war, mussten durch probeweise Entschlüsselung des restlichen Geheimtextes erkannt und verworfen werden. War der Schlüssel gefunden, konnte der gesamte Geheimtext, wie vom befugten Empfänger, einfach entschlüsselt werden.

Damit der Dechiffriervorgang nachvollziehbar ist, bietet sich die Zeichnung eines Graphen gut an. Der Graph benötigt als Grundlage ein Crib. Der Faktor Mensch macht bei solch komplexen Maschinen einige Fehler. Schlussfolgernd ist anzunehmen, dass der Datenverkehr typische Merkmale beinhaltet, wie Begrüßungs-/Abschiedsfloskeln an den jeweiligen Positionen in einem Telegramm. Des Weiteren konnte die Typex, wegen ihrer Ähnlichkeit zur Enigma, keinen Buchstaben auf sich selbst abbilden. Anhand des Beispieltextes „Hallo Welt mit Typex“ und dem entsprechenden Geheimtext (Konfiguration: AAAAA für Rotor I-II-III und Stator IV-V, Umkehrwalze A) in Tabelle 4.7 wird der Aufbau des Graphen erklärt. Bei der Abfolge wurden Klartextbuchstaben in einen verschlüsselten Buchstaben überführt.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
H	A	L	L	O	W	E	L	T	M	I	T	T	Y	P	E	X
D	E	B	P	J	L	M	P	E	H	L	W	Y	K	Q	D	J

Tabelle 4.7: Graphenerstellung mit Klar-/Geheimtext (crib)

Diese konkreten Abbildungen werden in den Graphen pro Buchstabe einmalig aufgenommen. Daraus entsteht ein ungerichteter Graph, welcher im besten Falle einen Zyklus aufweist (Abbildung 4.1).

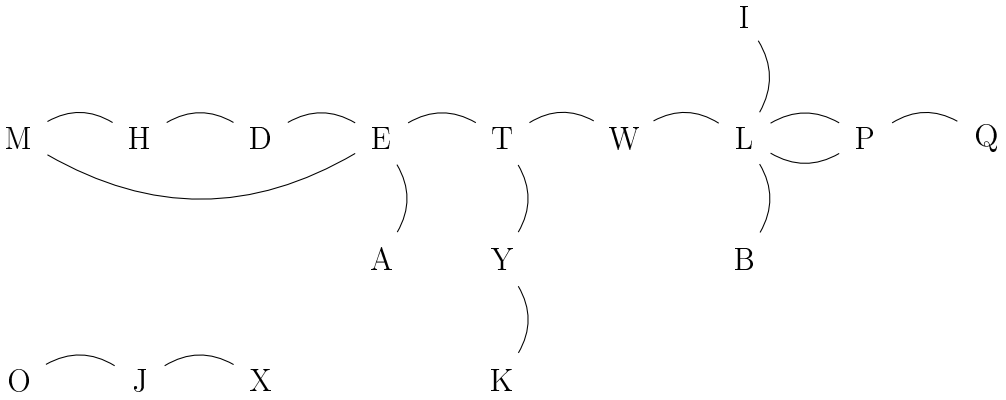


Abbildung 4.1: Menu Graph mit Zyklen

Ziel ist es, einen verbundenen Graphen mit so vielen Zyklen wie möglich abzuleiten (ein „Menu“). Ein Graph mit unzureichenden Buchstaben oder Zyklen wird den Algorithmus der Bombe beim Suchen nach dem Schlüssel sehr oft halten lassen. Dies bedeutet wieder weitere Arbeit zur richtigen Schlüsselbestimmung.

Das Ergebnis besteht nun aus zwei Graphen (Abbildung 4.1). Im Allgemeinen sind zusammenhängende Graphen besser zu verarbeiten. Aus diesem Grund werden die Buchstaben O, J, X, I und Q aus dem Graph entfernt. Die Anzahl der Knoten sollte nach oben begrenzt sein. Ferner sollte bei der Turing-Bombe nicht mehr als 12 Verknüpfungen vorhanden sein. Daher werden auch die Buchstaben I und Q entfernt, damit nur 12 Verknüpfungen im Graphen verbleiben. Die Turing-Bombe testet drei Rotoreinstellungen zur selben Zeit, wenn das zu testende Menu mit 12 Trommelreihen arbeitet.

Der Schlüssel kann aus vielen unterschiedlichen Kombinationen von Rotor- und Ringstellung definiert sein, da sich die Ringstellung „ABC“ und Grundstellung „HNT“ absolut gleich zur Ringstellung „ZZZ“ und Grundstellung „GLQ“ verhalten. Es kann weiterhin angenommen werden, dass die Rotoren auf „ZZZ“ gestellt sind. Aufgrund dieser Annahmen wurde der erste Buchstabe des Klartextes mit der Rotorstellung „ZZA“ verschlüsselt und der Zweite mit „ZZB“. Zur Verdeutlichung werden die Rotorstellungen für den jeweiligen Buchstaben in den Graphen geschrieben. Das beschriebene Verfahren ist in Abbildung 4.2 ersichtlich.

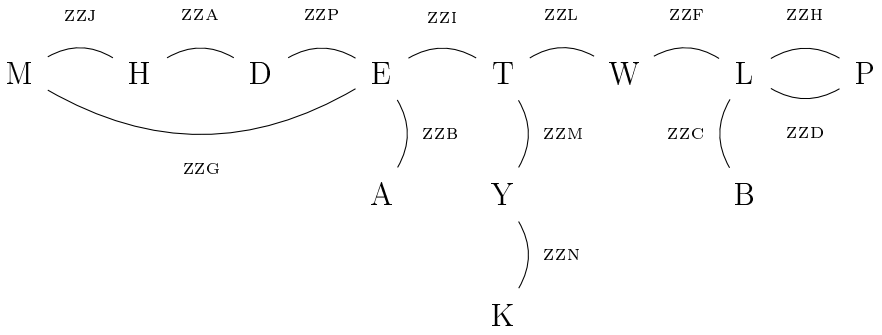


Abbildung 4.2: Menu Graph mit geschlossenen Zyklen und spezifischen Rotorstellung

Es wird eine Enigma-Trommelreihe zu jeweils einer Kante im Graphen verbunden. Es wird die Nummer der Enigma mit der angenommenen Grundstellung vermerkt. Es wird versucht, eine durchgängige Folge mit den Enigmas zu verknüpfen. Dieses Verfahren erleichtert die Steckerung (Verknüpfung) innerhalb einer Turing-Bombe, da dabei der Datenausgang einer Enigma direkt mit dem Dateneingang der nachfolgenden Enigma verbunden wird. Beim genannten Beispiel bietet sich folgende erste Route $K \rightarrow Y \rightarrow T \rightarrow E \rightarrow D \rightarrow H \rightarrow M \rightarrow E \rightarrow A$. Als zweite Route $B \rightarrow L \rightarrow P \rightarrow L \rightarrow W \rightarrow T \rightarrow E \rightarrow A$ und als Input bietet sich E an. Zusätzlich wird ein zentraler Buchstabe als Eingabe ausgewählt, denn hier wird der Testkandidat angesetzt.

Aus den definierten Zeichenfolgen entsteht eine Steckertabelle mit der dazugehörigen Belegung. Jeder Buchstabe im Graph wird mit der aktuellen Position zur jeweiligen Verbindung verknüpft. Der Buchstabe „E“ ist zum Beispiel an dem Ausgang 3 und an dem Eingang 4, des Weiteren noch am Eingang 15 verbunden. Buchstabe K ist nur am Eingang der ersten Trommelreihe verbunden. Ist ein Buchstabe am Ausgang und an einem Eingang mit verschiedenen Enigma-Trommelreihen verknüpft, so wird diese Verbindung in Klammern notiert. Diese können bei der Turing-Bombe mit einem speziellen Stecker (Bridge-Connector) verbunden werden. Zur Vervollständigung wird ein Buchstabe zum Testen ausgewählt, der weder der Eingabebuchstabe noch der direkte Nachbar von diesem ist. Hier bietet sich der Buchstabe H an und wird in Tabelle 4.8 vermerkt.

Position		Stellung		Route		Belegung
1	:	ZZN		K	:	1 in
2	:	ZZM		Y	:	(1 out, 2 in)
3	:	ZZI		T	:	(2 out, 3 in), (13 out, 14 in)
4	:	ZZP		E	:	(3 out, 4 in), (7 out, 8 in), (14 out, 15 in), input
5	:	ZZA		D	:	(4 out, 5 in)
6	:	ZZJ		H	:	(5 out, 6 in)
7	:	ZZG		M	:	(6 out, 7 in)
8	:	ZZB		A	:	8 out, 15 out
9	:	ZZC		B	:	9 in
10	:	ZZD		P	:	(10 out, 11 in)
11	:	ZZH		L	:	(11 out, 12 in), (9 out, 10 in)
12	:	ZZF		W	:	(12 out, 13 in)
13	:	ZZL				
14	:	ZZI				
15	:	ZZB				Aktueller Eingang bei H

Tabelle 4.8: Menu-Ergebnis von Crib aus Tabelle 4.7

Dieses Menu wird die Hypothese testen, ob H mit E auf dem Steckerbrett verbunden war. Durch diese Konstruktion, wird das korrekte Ergebnis geliefert, selbst wenn unsere Hypothese falsch oder richtig ist. Die Funktion hat folgende Definition:

(i) Rotorreihenfolge und Rotorwahl ist korrekt

  (a) Hypothese ist richtig → 25 von 26 Relays sind stromführend

  (b) Hypothese ist falsch → nur ein Relay ist stromführend

(ii) Rotorreihenfolge und Rotorwahl ist *nicht* korrekt

  (a) jede Hypothese ist falsch → alle Relays oder nur wenige zufällige sind stromführend

All diese Merkmale vervollständigen das Menu und helfen den Schlüssel bei der Typex zu finden, indem die bereits bekannte Turing-Bombe als Funktion definiert wird. Dies kann mit Hilfe eines Menus, welches an deinem Crib der Typex erstellt wurde, als Eingabe einer Bombe zur sicheren Schlüsselfindung benutzt werden. Wetterberichte, Begrüßungs- und Abschiedsfloskeln, Regulierungen und Wiederholungen wurden bei der Auswahl des Cribs und der Grundeinstellungen berücksichtigt. So könnten die Schlüssel in den unterschiedlichen Funknetzwerken aufgedeckt werden. Der Bombenablauf ist definiert. Er gilt bereits als implementiert und nutzbar für einen Known-Plaintext Angriff auf Basis eines Menus [48].

4.2.4 Hill-Climbing

Das Hill-Climbing ist ein heuristisches Suchverfahren, welches zwei grundlegende Funktionen benötigt:

- einen Algorithmus, der eine lokale Veränderung durchführt und die Lösung übernimmt, wenn der entstandene Lösungskandidat besser als der Vorherige geeignet ist, und
- eine Kostenfunktion, welche den Lösungskandidaten bewertet.

Es wird ein Zahlenwert berechnet. Mit diesem wird bewertet, ob ein Ergebnis besser als das Vorherige ist. Das Suchverfahren findet im Allgemeinen ein lokales Maximum, denn sobald das Maximum erreicht ist, kann kein besseres Ergebnis erreicht werden. Die Eingabeparameter sind mit Bedacht zu wählen, da jedes weitere Ergebnis auf den Eingabewerten basiert.

Das Hill-Climbing der Rotorverdrahtung, das Kelly Chang in seiner Master-Arbeit beschreibt [38], beläuft sich auf einen Suchraum von $2^{88,38}$ pro Rotor, da die Rotorverdrahtung 26! unterschiedliche Zustände annehmen kann. Wegen des großen Suchraums wird ein heuristisches Verfahren ausgewählt, dass innerhalb polynomieller Zeit mit einer Lösung terminiert. Als Kostenfunktion wird eine Wortpaar-Analyse (Bigramm) verwendet, welche die Wahrscheinlichkeit über das Auftreten zweier Buchstaben nebeneinander ausdrückt. Eine Bigramm-Analyse ist erfolgreich bei einer Substitutionschiffre und daher äquivalent zu der Rotorverdrahung eines Rotors [18].

Kelly Chang verwendete einen Tauschalgorithmus zum sukzessiven Durchlaufen der Kandidaten [49]. Bei einem Rotor mit r-Kontakten und einer potenziellen Verdrahtung von W_1 nach W_R wird im ersten Durchgang W_1 mit W_2 getauscht, danach W_2 mit W_3 bis W_R. Im zweiten Durchlauf tauscht W_1 mit W_3, dann W_2 mit W_4 und bei W_{R-1} wird W_R mit W_1 vertauscht. Im Ganzen werden $\binom{R}{2}$ Vertauschungen durchgeführt.

Chang zeigt, dass er einen unbekannten Rotor mit hoher Trefferrate nach einem Suchraum von 2^{26} entdeckt. Im Verhältnis zu einem Brute-Force Angriff mit

$\frac{26!}{2} \approx 2^{87}$ ist dieser Wert signifikant tiefer. Mit einem weiteren unbekannten Rotor verändert sich der Aufwand exponentiell. Hierfür hat Chang eine abgewandelte Form des Hill-Climbings entwickelt, womit er bei einem Aufwand von 2^{41} liegt. Der vergleichbare Brute-Force Angriff bei zwei unbekannten Rotoren basiert auf einem Suchraum von 2^{174}. Der spezifizierte Algorithmus ist schneller bei zwei Rotoren, als der vorherige Algorithmus für einen Rotor, da sich der Aufwand verdoppelt ($2^{41} \cdot 2^{41} = 2^{82}$).

Die Rotorverdrahtung kann durch Hill-Climbing erschlossen werden. Eine möglichst zufällige Rotorverdrahtung, die alle Möglichkeiten zulässt, kann einen Ciphertext-Only Angriff abwehren. Die Häufigkeit eines chiffrierten Textes muss besser verteilt sein, damit der IC näher an die Zufälligkeit von $0,38$ heranrückt.

4.2.5 Analyse der Rotor-Verdrahtung nach Rejewski

Marian Rejewski war ein polnischer Mathematiker, der im Jahr 1932 die interne Verdrahtung der Enigma Rotoren aufgedeckt hat. Sein Angriff nutzte die ersten sechs Buchstaben eines jeden Telegramms, welches den Tagesschlüssel in verschlüsselter zweifacher Form beinhaltete. Wie beim Known-Plaintext Angriff basiert der Angriff auf der Zyklen-Struktur innerhalb eines abgefangenen Geheimtextes, wo die ersten sechs Buchstaben immer der chiffrierte Tagesschlüssel war [19].

Die Enigma-Bediener waren angewiesen, jeden Tag einen neuen Schlüssel zu verwenden. Sie sollten selbst eine dreistellige Kombination zufällig wählen, welche später mit dem Tagesschlüssel chiffriert und an jedes Telegramm in den führenden drei Stellen mit zweifacher Wiederholung platziert wurde. Dieser Fehler hatte kritische Auswirkungen und befähigte Rejewski letztlich die Verdrahtung zu bestimmen [47]. Das Schema von Rejewski ist im Detail bekannt [19]. Es kann auch auf die Typex unter bestimmten Annahmen angewendet werden. Sofern das Bedienpersonal der Typex eine solche Schlüsselfrequenz vor jede Nachricht schrieb, wird auch dieser Angriff bei der Typex funktionieren.

Rejewski zeichnete zahlreichen Funkverkehr über einen Tag hinweg auf und verglich die ersten sechs Stellen untereinander. Die Zyklen traten immer zwischen der ersten und vierten, zweiten und fünften, dritten und sechsten Stelle auf. Er bezeichnete die Zyklen nach AD, BE, CF, welche aus der Position zum Buchstaben konvertiert wurden. Die Gleichungen in 4.5 stellen ein Beispiel für eine komplette Permutation in Zyklengruppen da:

$$\begin{aligned} AD &= (kxgzyodvpf)(nqlhteijmu)(bc)(rw)(a)(s) \\ BE &= (veoumblfq)(wizrnhjps)(xta)(gyc)(d)(k) \\ CF &= (jgfcqnyabvikt)(lxwpsmoduzreh) \end{aligned} \tag{4.5}$$

Der Erfolg lag darin, diese Gleichungen mit Zyklen zu faktorisieren und dadurch die Zerlegung in die individuellen Kombinationen (A, B, C, D, E, F) zu vollziehen.

Die Berechnungen erfolgten durch allgemein geltende Regeln in der Permutations-Theorie. Wichtig war, dass die gewählten Schlüssel vom Enigma-Personal nicht rein zufällig waren. Die wichtige Folgerung XY beim Faktorisieren gilt im Generellen: Wenn $XY = (a_1a_3)(a_2a_4)$ dann $X = (a_1a_4)(a_2a_3)$ und $Y = (a_1a_2)(a_3a_4)$ [19].

Gemäß dem Transponieren bei Permutationen lässt sich die Gleichung umformen. Im vorherigen Beispiel ist ersichtlich, dass (as) in A und D auftritt. Des Weiteren tritt entweder $(br)(cw)$ oder $(bw)(cr)$ in A auf, denn dann ist $(cr)(bw)$ oder $(cw)(br)$ in D vorhanden geltend zu den Elementen in A. Die letzte Zyklengruppe $(kxgzyodvpf)(nqlhteijmu)$ ist sehr aufwendig zu faktorisieren. Mit dem Wissen, dass das Enigma Personal die Schlüssel nicht zufällig wählte, sondern gewisse Muster erkennbar waren, konnte der Faktorraum dramatisch reduziert werden. Dies führte zu einzelnen Gleichungen in Formel 4.6, bestehend aus Paaren [19].

$$\begin{aligned}
A &= (as)(br)(cw)(di)(ev)(fh)(gn)(jo)(kl)(my)(pt)(qx)(uz)\\
B &= (ay)(bj)(ct)(dk)(ei)(fn)(gx)(hl)(mp)(ow)(qr)(su)(vz)\\
C &= (ax)(bl)(cm)(dg)(ei)(fo)(hv)(ju)(kr)(np)(qs)(tz)(wy)\\
D &= (as)(bw)(cr)(dj)(ep)(ft)(gq)(hk)(iv)(lx)(mo)(nz)(uy)\\
E &= (ac)(bp)(dk)(ez)(fh)(gt)(io)(jl)(ms)(nq)(rv)(uw)(xy)\\
F &= (aw)(bx)(co)(df)(ek)(gu)(hi)(jz)(lv)(mq)(ns)(py)(rt)
\end{aligned} \tag{4.6}$$

Die Fakturierungen befähigten Rejewski die Gleichungen, um einige Variablen zu reduzieren. Die Rotoren lassen ihren Nachbar spätestens nach einer vollen Umdrehung und frühestens beim Erreichen einer Kerbe drehen. Unter dieser Annahme wird sich nur der schnelle Rotor bewegen (bei der Eingabe der ersten sechs Stellen (Tagesschlüssel)). Daher konnte der mittlere R_m, R_m^{-1} und linke Rotor R_l, R_l^{-1} aus der Gleichung als Unbekannte gestrichen werden. Daraus ersetzte Rejewski die Teilgleichung von $R_mR_lTR_l^{-1}R_m^{-1}$ auf eine Variable Q. Die verbliebenen Variablen R_r, Q, S sind noch zu identifizieren mit P^i als Eingangspermutation (siehe Formel 4.7).

$$\begin{aligned}
A &= SP^1R_rP^{-1}QP^1R_r^{-1}P^{-1}S^{-1} \text{ mit } P^1 = (abcdefghijklmnopqrstuvwxyz)\\
B &= SP^2R_rP^{-2}QP^2R_r^{-1}P^{-2}S^{-1} \text{ mit } P^2 = (acegikmoqsuwy)(bdfhjlnprtvxz)\\
C &= SP^3R_rP^{-3}QP^3R_r^{-1}P^{-3}S^{-1} \text{ mit } P^3 = (adgjmpsvybehknqtwzcfilorux)\\
D &= SP^4R_rP^{-4}QP^4R_r^{-1}P^{-4}S^{-1} \text{ mit } P^4 = (aeimquycgkosw)(bfjnrzvdhlptx)\\
E &= SP^5R_rP^{-5}QP^5R_r^{-1}P^{-5}S^{-1} \text{ mit } P^5 = (afpuzejotydinsxchmrwbglqv)\\
F &= SP^6R_rP^{-6}QP^6R_r^{-1}P^{-6}S^{-1} \text{ mit } P^6 = (agmsyekqwciou)(bhntzflrxdjpv)
\end{aligned} \tag{4.7}$$

Die Franzosen kauften die Tagesschlüssel von einem Informanten, welche auch an die polnische Regierung weitergeleitet wurden. Diese Informationen reduzierten die Gleichung um eine weitere Variable, da S nun bekannt war [50]. Durch geltende Umformungsregeln kann S und P auf die linke Seite umgeschrieben und neu bezeichnet $(U - Z)$ werden [19]. Dies führte zur Formel 4.8.

$$
\begin{aligned}
U &\overset{S}{\equiv} S^{-1}AS = P^1R_rP^{-1}QP^1R_r^{-1}P^{-1} \overset{P}{\equiv} P^{-1}S^{-1}ASP^1 = R_rP^{-1}QP^1R_r^{-1} \\
V &\overset{S}{\equiv} S^{-1}BS = P^2R_rP^{-2}QP^2R_r^{-1}P^{-2} \overset{P}{\equiv} P^{-2}S^{-1}BSP^2 = R_rP^{-2}QP^2R_r^{-1} \\
W &\overset{S}{\equiv} S^{-1}CS = P^3R_rP^{-3}QP^3R_r^{-1}P^{-3} \overset{P}{\equiv} P^{-3}S^{-1}CSP^3 = R_rP^{-3}QP^3R_r^{-1} \\
X &\overset{S}{\equiv} S^{-1}DS = P^4R_rP^{-4}QP^4R_r^{-1}P^{-4} \overset{P}{\equiv} P^{-4}S^{-1}DSP^4 = R_rP^{-4}QP^4R_r^{-1} \\
Y &\overset{S}{\equiv} S^{-1}ES = P^5R_rP^{-5}QP^5R_r^{-1}P^{-5} \overset{P}{\equiv} P^{-5}S^{-1}ESP^5 = R_rP^{-5}QP^5R_r^{-1} \\
Z &\overset{S}{\equiv} S^{-1}FS = P^6R_rP^{-6}QP^6R_r^{-1}P^{-6} \overset{P}{\equiv} P^{-6}S^{-1}FSP^6 = R_rP^{-6}QP^6R_r^{-1}
\end{aligned}
\tag{4.8}
$$

Durch die Eingrenzung waren nur zwei Unbekannte zu bestimmen. So brauchte Rejewski nur vier Gleichungen (U, V, W und X) auf das Ergebnis UV, VW, WX zu faktorisieren. Schlussfolgernd stellte er fest, dass $QP^{-1}QP^1$ jeweils als Paar auftreten und das daraus die gleichen Zyklen entstehen. Durch Umformung der Gleichung (in Formel 4.9) konnten gemeinsame Teilformeln $QP^{-1}QP$ zwischen UV, VW und VW, WX eliminiert werden.

$$
\begin{aligned}
UV &= (R_rP^{-1}QP^1R_r^{-1})(R_rP^{-2}QP^2R_r^{-1}) = \\
&\quad R_rP^{-1}(QP^{-1}QP^1)P^1R_r^{-1} \\
VW &= (R_rP^{-2}QP^2R_r^{-1})(R_rP^{-3}QP^3R_r^{-1}) = \\
&\quad R_rP^{-2}(QP^{-1}QP^1)P^2R_r^{-1} = \\
&\quad (R_rP^{-1}R_r^{-1})^{-1}(UV)R_rP^1R_r^{-1} \\
WX &= (R_rP^{-3}QP^3R_r^{-1})(R_rP^{-4}QP^4R_r^{-1}) = \\
&\quad R_rP^{-3}(QP^{-1}QP^1)P^3R_r^{-1} = \\
&\quad (R_rP^{-1}R_r^{-1})^{-1}(VW)R_rP^1R_r^{-1}
\end{aligned}
\tag{4.9}
$$

Die Auswertung der Zyklen UV, VW und WX wurde in die Zyklenpaare $\overset{VW}{UV}$ und $\overset{WX}{VW}$ überführt. Die Gleichung wurde nach $R_rP^1R_r^{-1}$ gruppiert. Sofern P^1 weithin bekannt ist, wird in diesem finalen Schritt die Rotorverdrahtung des schnellen Rotors gefunden.

Mathematische Annahmen mit P^1 führen zu möglichen Einsetzungen und Umformungen. Die Werte für Rejewskis nächsten Analyseschritt führt die Gleichungen UV, VW und WX in zyklische Gruppen zusammen. Alle möglichen Formen von der ersten Gleichung mit IV, VW, und WX lassen viele Lösungen für $R_rP^1R_r^{-1}$ entstehen. Sofern P^1 weithin bekannt ist, wird in diesem finalen Schritt die Rotorverdrahtung des schnellen Rotors gefunden. Das Berechnen aller Formen für die zweite Gleichung ergibt dieselbe Anzahl von Lösungen. Alle Lösungen aus den Gleichungen müssen bei jeder Iteration ein gemeinsames Ergebnis haben. Es handelt sich um die Rotorverdrahtung für den Tagesschlüssel. Sofern kein Ergebnis

gefunden wurde, ist ein Fehler bei den Berechnungen unterlaufen oder eine Drehung des Rotors erfolgte.

Der Angriff funktioniert unter denselben Annahmen auch bei Typex-Rotoren. Die Annahme, dass der mittlere und der langsame Rotor nicht rotieren, tritt nur in wenigen Fällen ein. Denn sobald ein Rotor mit vielen Kerben verwendet wird, ist von einer Drehung bei den beiden Rotoren auszugehen. Eine Drehung weiterer Rotoren würde das Eliminieren der Variablen verhindern und den Angriff komplizierter gestalten [38].

4.3 Statistische Analyse

Beim Ausführen eines Brute-Force Angriffs ist es notwendig, die resultierenden Kandidaten zu bewerten. In der Kryptoanalyse werden die Texte auf ihre Ähnlichkeit zu einer natürlichen Sprache geprüft. Der genaue Angriffsverlauf ist im Kapitel 4.2 erläutert. Ein wichtiges Detail von einem erfolgreichen Angriff ist das Bewerten der Ergebnisse mit einer Kostenfunktion. Diese Kostenfunktionen verwenden kryptographische Metriken, wie die Häufigkeitsanalyse der Buchstaben. Die Buchstaben können alleine oder als Paare unterschiedliche Häufigkeiten aufweisen. Des Weiteren gibt es unterschiedliche Konzepte diese zu zählen oder gar zu berechnen. Diese Berechnungen sind mit Beispielen auf den folgenden Seiten aufgelistet.

Der Ciphertext-Only Angriff wurde konzeptionell erläutert. Damit das Programm einen Textkandidaten als Modell erkennt, werden die folgenden Metriken verwendet. Der generelle Ablauf wird zur Erläuterung der gewählten Metriken nochmals erwähnt. Der Verlauf umfasst zwei Phasen, wobei die Erste die Rotoren und die zweite Phase die Ringstellung analysiert. Die Metrik bewertet, welche Rotor- und Ringstellung ein gutes Ergebnis geliefert haben. In Phase 1 werden 843.648 ($= 48 \cdot 17.576$) Textkandidaten mit solch einer Metrik bewertet, ob ein korrekter Schlüssel vorliegt. In Abbildung 4.3 ist das schrittweise Auffinden der korrekten Grundeinstellung aus Phase 1 und Phase 2 ersichtlich. Der Abbildung 2.6 ist zu entnehmen, dass die Buchstaben „e“ und „a“ sehr dominant in der englischen Sprache sind. Dieses Verhalten lässt sich auf den Testkandidaten spiegeln. Es ist ersichtlich, dass sich diese Verteilung beim sukzessiven Durchlaufen auf die Ausgangswerte des Klartextes wieder einstellt.

Die Entropie liegt im Fokus eines Kryptoanalytikers. Entropie bedeutet „Je mehr Zeichen im Allgemeinen von einer Quelle empfangen werden, desto mehr Information erhält man und gleichzeitig sinkt die Unsicherheit über das, was hätte gesendet werden können“ [51]. Ein offensichtliches Beispiel von Shannon ist der Buchstabe u in der Kombination qu. Bei englischen Wörtern ist der Buchstabe q immer gefolgt vom Buchstaben u. Entropie entsteht gerade dort, wo die Regeln einer natürlichen Sprache angewandt werden. Eng verknüpft mit der Entropie ist der Begriff Redundanz. Darunter wird die Differenz, zwischen der maximalen Entropie und der

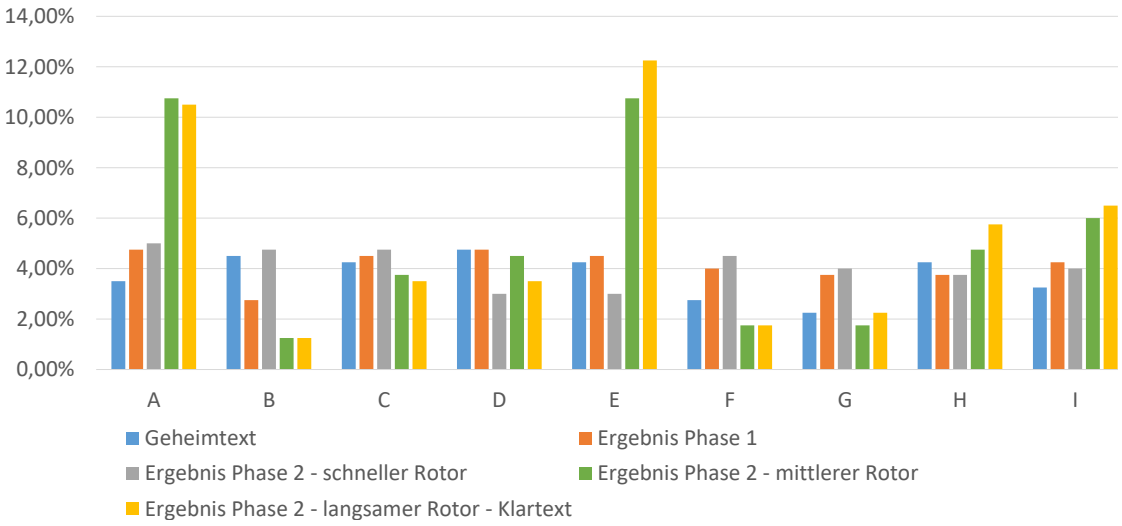


Abbildung 4.3: Buchstabenverteilung bei Entschlüsselung – Nachrichtenlänge von 400 Zeichen

Entropie des betrachteten Textes oder Sprache verstanden. Ein Zufallstext weist keinerlei Redundanz auf, während natürliche Sprachen redundant sind.

Diese Eigenschaften bilden die Grundlage für Analysen basierend auf der Buchstabenhäufigkeit (Bi- und Trigrammhäufigkeit). Die Typex ist ein polyalphabetische Chiffre. Daher wird versucht die unregelmäßige Buchstabenverteilung so zu ändern, dass diese im Chiffretext möglichst gleich verteilt ist. Beim Analysieren des Geheimtextes wird versucht das beste Ergebnis aus mehreren Entschlüsselungen zu wählen, welches am ähnlichsten zur Ausgangssprache ist. Die natürliche Spracherkennung mit einer Kostenfunktion ist wichtig, sobald ein Brute-Force Angriff auf einen beliebig definierten Schlüsselraum gestartet wird. Denn die Ergebnisse eines Kandidaten müssen mit einer Funktion auf Sprachgültigkeit bewertet werden. Die nachfolgenden Metriken wurden an Bedingungen aus dieser Arbeit geknüpft, um die Ähnlichkeit zu einer natürlichen Sprache zu berechnen. Im Folgenden wird die Variablendefinition aus Tabelle 4.9 verwendet.

Ausgangssprache ist Englisch mit den Buchstaben A bis Z ($\mathbb{A}_\mathbb{P} = \mathbb{A}_\mathbb{C} = 26$).

Ein **Textkandidat** ist das jeweilige Dechiffrierergebnis mit einem Schlüssel.

Variable	Definition
n	Textlänge des Textkandidaten
h_i	Häufigkeit vom Buchstaben „i“ im Textkandidat C
p_i	Häufigkeit des einzelnen Buchstaben „i“ in der Ausgangssprache
$\tilde{p}_i$	Häufigkeit des einzelnen Buchstaben „i“ im Textkandidat C

Tabelle 4.9: Allgemeine Syntax der Variablendefinition

(1) **Koinzidenzindex** IC eines Chiffretextes C gibt die Wahrscheinlichkeit an, dass zwei unabhängig voneinander gewählte Zeichen mit beliebigem Abstand des Chiffretextes C gleich sind. Der Wert wird aussagekräftiger bei steigender Textlänge. Zum einen lässt sich entscheiden, ob eine mono- oder polyalphabetische Chiffre vorliegt, zum anderen kann je nach Wert auch die Sprache festgestellt werden.

Für einen Chiffretext C, der aus einer Caesar-Chiffrierung hervorgegangen ist, hängt der Koinzidenzindex IC nicht vom Schlüssel K, sondern lediglich von den statistischen Eigenschaften der Klartextquelle ab. Die Durchschnittswerte in Tabelle 4.10 geben den IC für die jeweiligen Sprachen an.

$\approx IC$			
Englisch	0,0667	Französisch	0,0778
Deutsch	0,0762	Spanisch	0,0770
Italienisch	0,0738	Russisch	0,0529
Zufälliger Text		0,0385	

Tabelle 4.10: Koinzidenzindex-Wert für natürliche Sprachen [8, 52]

Ausgegangen wird vom lateinischen Alphabet mit 26 Zeichen, wobei die Häufigkeit der Buchstaben mit h_1 für A, h_2 für B bis h_{26} für Z angegeben wird. Über die Länge n des Textes ergibt sich dann die Formel 4.10.

$$\sum_{i=1}^{26} h_i \tag{4.10}$$

Die Gesamtanzahl aller Paare aus A's ist die Formel 4.11.

$$\frac{h_1(h_1-1)}{2} \tag{4.11}$$

Denn für die Auswahl des Ersten A's gibt es h_1 Möglichkeiten, für die Auswahl des zweiten nur noch $h_1 - 1$ Möglichkeiten. Also ergibt sich die Formel 4.12 für die Gesamtzahl A aller Paare, bei dem beide Buchstaben gleich sind.

$$A = \frac{h_1(h_1-1)}{2} + \frac{h_2(h_2-1)}{2} + \cdots + \frac{h_{26}(h_{26}-1)}{2} = \sum_{i=1}^{26} \frac{h_i(h_i-1)}{2} \tag{4.12}$$

Die Wahrscheinlichkeit IC (Formel 4.13) dafür, dass ein beliebiges Buchstabenpaar aus zwei gleichen Buchstaben besteht, berechnet sich demgemäß

$$IC = \sum_{i=1}^{26} \frac{h_i(h_i-1)}{h(h-1)} \tag{4.13}$$

(2) Der **Chi-Squared-Test** entspricht der Abweichung des Textes zu einer Sprache. Befindet sich der ermittelte Wert dieser Abweichung in einem gewissen, vorher festgelegten Spektrum, so kann davon ausgegangen werden, dass es sich tatsächlich um einen Text aus der natürlichen Sprache handelt. Leider lässt sich der Chi-Squared-Test nach oben nicht genau begrenzen. Erfahrungswerte zeigen jedoch, dass das Maximum bei 300 erreicht ist. Typischerweise werden Werte zwischen 0 und 30 als sprachkonform angesehen. Aufgrund der unbekannten statistischen Zahlenverteilung werden Zahlen bei der Betrachtung ignoriert. Später im Programm wurde es so realisiert, dass Groß- und Kleinbuchstaben dieselbe Auftrittswahrscheinlichkeit haben, um die Berechnungsform nicht unnötig zu verkomplizieren. Der Chi-Squared-Test (Formel 4.14) wird mit der Alphabetgröße i und einer Auftrittswahrscheinlichkeit $\tilde{p_i}$ eines Zeichens i und der Wahrscheinlichkeit eines Auftretens in der natürlichen Sprache p_i ermittelt.

$$CS = \sum_{i=1}^{26} \frac{(h_i \tilde{p_i} - h_i p_i)^2}{h_i p_i} \tag{4.14}$$

(3) **Sinkov Statistik** wählt den Datensatz, welcher sich am nächsten zum Klartext befindet. Diese Eigenschaft lässt sich durch Summieren der einzelnen Buchstabenhäufigkeiten über der Ausgangssprache finden. Der höchste Wert ist das beste Modell für den zugrundeliegenden Geheimtext. Wenn ein Datensatz mit einem Gewicht bewertet wird, ist es oft das multiplikative Produkt der Häufigkeiten. Diese Methode benötigt höheren Rechenaufwand als eine Addition. Daher wird das Produkt durch den Logarithmus und die Addition ersetzt [53]. Diese Statistik tritt gehäuft bei der Analyse von Kryptosystemen auf, die ähnlich zu der Vigenère-Chiffre sind. Für einen Textkandidaten $C = C_1, C_2, \cdots, C_n$ mit n Unigrammen bewertet die Sinkov Statistik S (Formel 4.15).

$$S = \sum_{i=1}^{26} h_i \ln p_i \tag{4.15}$$

Die ln-Funktion ist hier als der natürlicher Logarithmus ($\log_e$) definiert. Im Beispiel von Sherman für den Text „DES“ verhält sich die Statistik wie in Tabelle 4.11 gezeigt.

Text	Häufigkeit	ln
D	0,04253	-3,158
E	0,12702	-2,063
S	0,06327	-2,760
	Total	-7,981

Tabelle 4.11: Sinkov Statistik am Beispiel „DES“ [54]

(4) Das **Tetragramm** ist ein weiteres Maß, die Ähnlichkeit für einen gegebenen Text C mit der Ausgangssprache zu bewerten. Dies wird durch das Aufteilen der Texte in kleine Stücke der Länge 4 erzielt. Das Beispiel „HALLOWELT“ resultiert in den Stücken HALL, ALLO, LLOW, LOWE, OWEL, WELT. Die verschiedenen Längen der N-Gramme (Uni-, Bi-, Tri-, Tetraqramm) verfolgen denselben Zweck, wie die Tetragramm-Analyse. Verschiedene Testläufe mit Tetragrammen deuten auf eine bessere Erkennung bei Chiffretexten der Typex. Die Verwendung von N-Grammen zeigte, dass die Genauigkeit mit höherem N erfolgversprechender war. Allerdings stellte sich dies bei Tetragrammen dann ein.

Die Analyse benötigt zusätzliche Informationen über N-Gramm-Häufigkeiten der jeweiligen Sprache, die mit Hilfe einer externen Datei verfügbar werden. Hierfür wird ein langer Text (z.B. Bücher) eingelesen und deren Tetragramme gezählt. Diese Grundlage lässt sich für spezielle Themengebiete modifizieren, indem nur einschlägige Bücher eingelesen werden. Jede Häufigkeit wird durch die Länge l des eingelesenen Textes dividiert. Die Differenz enthält die Tetragramm-Häufigkeit. Das Buch „Alice im Wunderland“ (englische Titel) umfasst ohne Leerzeichen und Punkte 163.816 Zeichen ($\equiv$ Tetragramme). Die Wahrscheinlichkeit wird ideal übersetzt durch Einsetzen des Logarithmus $p_i = log\ _l\ (C_i)$. Tabelle 4.12 stellt spezielle Häufigkeiten mit den Extremen von keinem bis zu sehr häufigem Auftreten da. Textstücke, die kein Referenzwert haben, werden berücksichtigt $0 \leq x$ und nicht als $-\infty/\infty$ oder undefiniert übersetzt. Wenn ein Text das Stück „EINE“ beinhaltet, wird es als sehr Ähnlich zu Deutsch gehandelt.

Tetragramm	Anzahl	Wahrscheinlichkeit mit Log
ZZZV	1	0
QZYW	0	0
UEDO	539	0,490957592
JÖRN	4.280	0,652691544
EINE	3.692.344	1,180363198

Tabelle 4.12: Wahrscheinlichkeiten für bestimmte Tetragramme

Für die Tetragramm-Berechnung eines Textes, werden die Wahrscheinlichkeiten der einzelnen Stücke gemeinsam multipliziert. Für den Textkandidaten wird die Formel 4.16 berechnet.

$$\begin{aligned} p(HALLOWELT) \;=\; & p(HALL) \cdot p(ALLO) \cdot p(LLOW) \cdot \\ & p(LOWE) \cdot p(OWEL) \cdot p(WELT) \end{aligned} \tag{4.16}$$

$$\text{mit} \quad p(x) = \frac{p_i}{n} = \frac{Anzahl\ \ Tetragramm}{Anzahl\ \ aller\ \ Tetragramme}$$

Wenn eine große Anzahl kleiner Wahrscheinlichkeiten miteinander multipliziert werden, treten bei Gleitkommazahlen Ungenauigkeiten auf. Da es in

der Gleitkommadarstellung eine kleinste positive Zahl gibt, unterhalb derer kein Wert mehr dargestellt werden kann, wird ein Ergebnis in diesem Bereich meistens durch „0“ repräsentiert. So ist an diesem Punkt jede Information über das Ergebnis verloren gegangen. Abhilfe schafft hier der Logarithmus bei Formel 4.17, denn so wird nach dem Gesetz $\log(a \cdot b) = \log(a) + \log(b)$ eine Addition der Werte hergestellt.

$$\begin{aligned} \log(p(HALLOWELT)) \;=\; & \log(p(HALL)) + \log(p(ALLO)) + \\ & \log(p(LLOW)) + \log(p(LOWE)) + \\ & \log(p(OWEL)) + \log(p(WELT)) \end{aligned} \tag{4.17}$$

Die Logarithmus-Wahrscheinlichkeit dient als Metrik zur Erkennung von Texten und deren Ähnlichkeit zu einer natürlichen Sprache. Je höher der Wert, desto näher befindet er sich an der Ausgangssprache.

Es existieren natürlich viele Texte die den Tetragramm-Test „gut“ bestehen werden. Dies ist der Fall, wenn die Texte besonders kurz sind, wie 50 - 100 Zeichen. Dann sollten die Interpretationen des Ergebnisses per Hand durchsucht werden, denn eventuell befindet sich ein Modell darunter. Die Liste mit den zugrundeliegenden Tetragrammen lässt sich komplett ersetzen mit einer maßgefertigten Liste. Das heißt, Tetragramme, die überhaupt nicht im Inhalt eines Textes erwartet werden, können entfernt oder gar als schlechte Tendenz markiert werden. Texte wie „Die Katze bellt viel leise“ bestehen den Test und vergeuden Rechenleistung, da sie bis zum Ende als gute Interpretation weiter geführt werden.

5 Design und Implementierung

Dieses Kapitel erklärt die Implementierung des Simulators und der Typex-Analysekomponente. Dabei wird die Funktionsweise und Implementierung der verwendeten Techniken erläutert. Es wurden zwei verschiedene Komponenten implementiert: der Typex-Simulator und das Typex-Analysewerkzeug.

In Abbildung 5.1 ist die Programmstruktur gezeigt. Jedes Projekt (in der Entwicklungsumgebung „MS Visual Studio“) hat einen eigenen Einstiegspunkt mit der jeweiligen „Programm“ Klasse, wobei die verschachtelten Klassen die darunterliegende Struktur mehrmals zur Verarbeitung aufrufen.

Das Projekt Typex-Simulator wird zuerst erklärt, da es das tiefst gegliederte Projekt ist und für die weiteren Analyseschritte essentiell ist. Die Klasse *TypexMachine* umfasst dabei die meisten Funktionalitäten.

Das Projekt Kryptoanalyse beinhaltet weitaus mehr Klassen als der Simulator. Die Klassen spiegeln den Ciphertext-Only Angriff wieder und sind in die Baugruppen der Maschine unterteilt. Entweder wird die *DecodeRotors* oder die *DecodeStators* mit der nachgelagerten Klasse *DecodeRingsSequently* aufgerufen.

Das Projekt Evaluation generiert lediglich Textkandidaten, die an die Kryptoanalyse gereicht werden. Die Evaluation zeichnet die Erfolgsrate über viele Textkandiaten bei einer festen Textlänge auf.

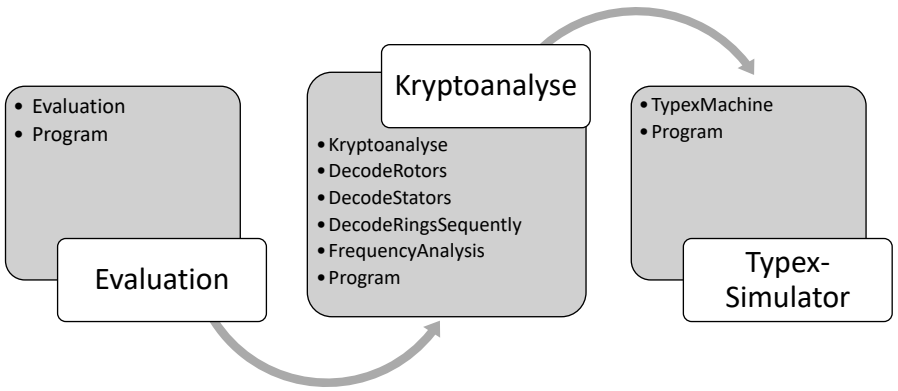


Abbildung 5.1: Projekt-Abhängigkeiten

Analog zu der genannten Auflistung strukturieren sich die folgenden Erläuterungen in den Kapiteln. Abbildung 5.2 zeigt den Programmfluss in einem Ablaufdiagramm von Programmeinstieg bis zur Ausgabe des Ergebnisses.

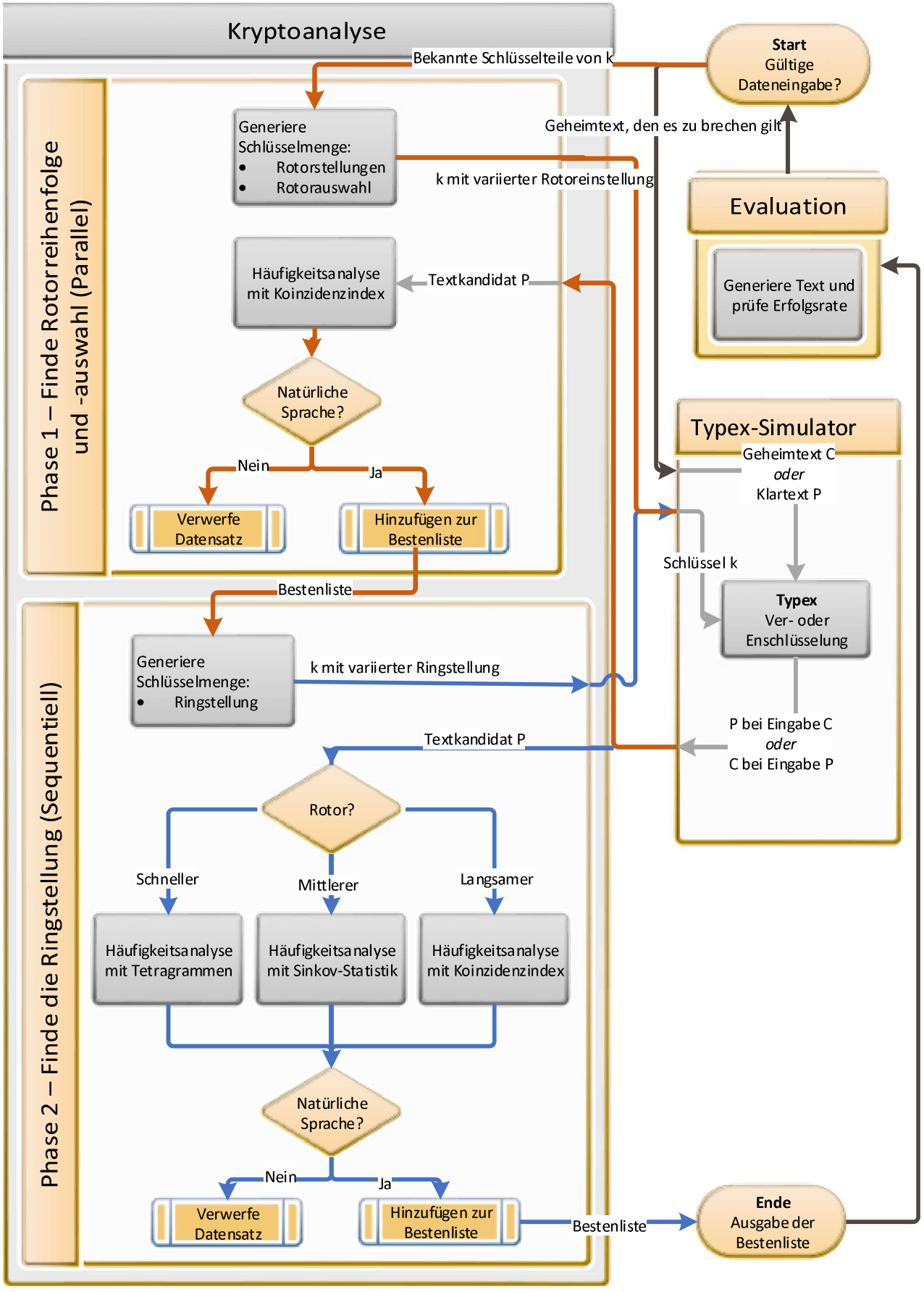


Abbildung 5.2: Ablaufdiagramm des implementierten Ciphertext-Only Angriffs

Klassendiagramm Im technischen Detail sind die Komponenten in einem Ausschnitt des Klassendiagramms ersichtlich. Dort sind die Klassen, innere Klassen, Methoden, Felder und Attribute für das jeweilige Projekt aufgelistet.

Projekte sorgen dafür, dass die Klassen gesammelt in einer Struktur geordnet sind. In der Entwicklungsumgebung MS Visual Studio wird der Name des Projektes auch als Name des Namespaces übertragen.

Eine Klasse ermöglicht das benutzerspezifische Definieren eines Typen (Objekts). Diese wird über eine Sammlung von Methoden und Variablen (Felder und Attribute) beschrieben.

Die Methoden beschreiben und implementieren das Verhalten von Datentypen in einer Klasse. Eine Variable repräsentiert einen abgelegten Wert in einer Speicherstelle, deren Inhalt zur Laufzeit jederzeit verändert werden kann.

Ein Feld ist eine Variable eines beliebigen Typs, die direkt in einer Klasse deklariert ist. Ein Attribut gewährleistet den sicheren Zugriff auf eine nicht-öffentliche Variable. Ein Attribut ist Teil einer Klasse und bietet einen flexiblen Mechanismus zum Lesen, Schreiben oder Berechnen des Werts eines privaten Feldes.

Abbildung 5.3 zeigt die abstrahierte Struktur des Programms und der darunterliegenden Komponenten. Die Abbildungen werden an den Stellen ihrer Beschreibung in dieser Arbeit detaillierter in einer gesonderten Abbildung gezeigt. Das Projekt Evaluation beinhaltet zwei Klassen „Evaluation“ und „Program“. Die innere Klasse„Evaluation Configuration“ ist im Quellcode der Klasse „Program“ eingegliedert. Das Projekt Kryptoanalyse hat neun Klassen und zwei innere Klassen. Diese Beschreibung ist auch auf das Projekt Typex-Simulator anwendbar. Es hat zwei Klassen und drei innere Klassen.

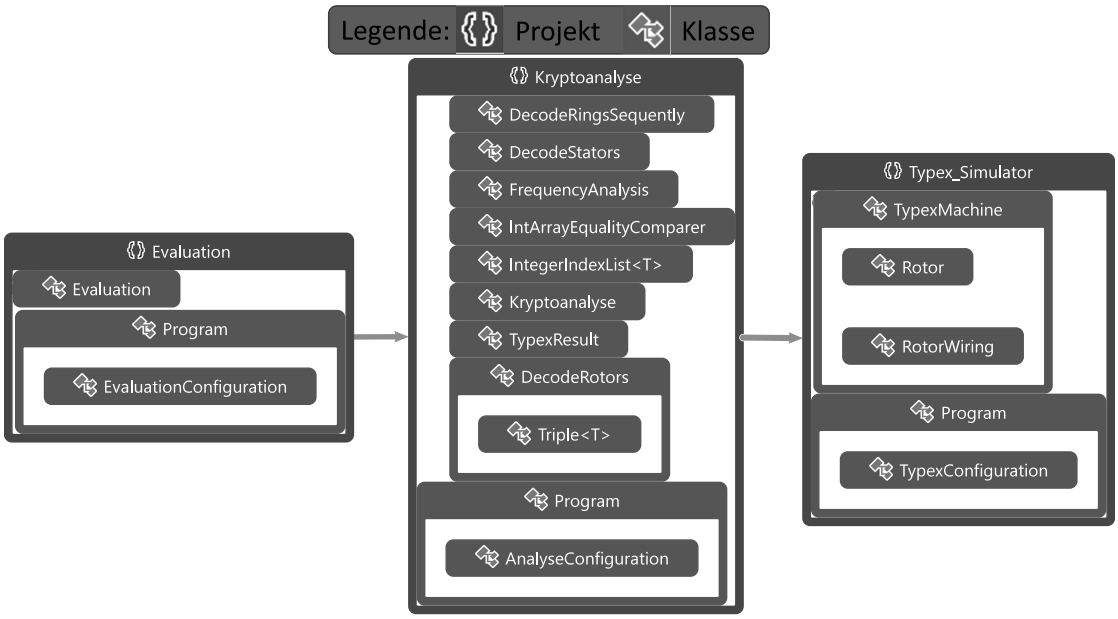


Abbildung 5.3: Klassendiagramm vom entwickelten Programm

Die folgenden Kapitel werden mit Hilfe des Klassendiagramms des jeweiligen Abschnitts visualisiert. Zuerst ist das Diagramm von den oberen Projektstruktur gezeigt. Die Komponenten des Typex-Simulators, der Analysekomponente und der Evaluation werden in gesonderten Kapiteln beschrieben. Die jeweilige Abbildung der Komponente zeigt die Struktur zwischen Projekt und Klassen.

Die Erklärung zu den bestimmten Klasse ist in Abschnitte untergliedert. Jeder Abschnitt zeigt die aktuelle Struktur der Klassen zueinander. Der Klasseninhalt wird in detaillierteren Klassendiagrammen über Klassen, Methoden, Felder und Attribute visualisiert.

5.1 Der Typex-Simulator

Der Simulator ver- und entschlüsselt einen Text mit einem gegebenen Schlüssel nach der Methode einer Typex-Maschine. Die Techniken und Methoden des Simulators werden im Folgenden erläutert. Die Bezeichnung der Methoden erfolgt analog zum Programmcode. Zunächst werden die mechanischen Bauteile in Programmcode überführt, anschließend in der korrekten Reihenfolge aufgerufen. Der Typex-Simulator umfasst zwei Klassen und deren untergliederte innere Klassen (Abbildung 5.4).

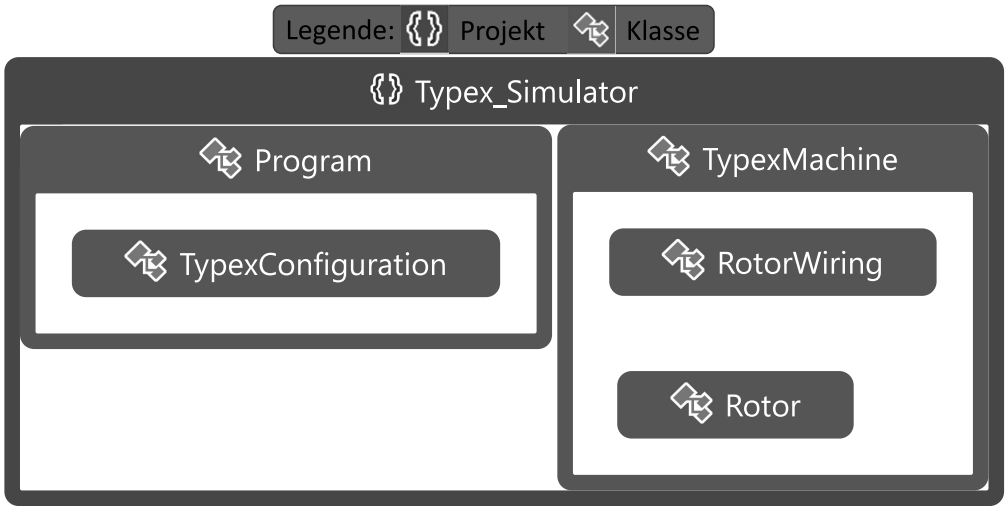


Abbildung 5.4: Klassendiagramm des Projektes – Typex-Simulator

Der Schlüssel für eine erfolgreiche Simulation der Typex-Maschine ist die interne Verkabelung der Rotoren (oder „Walzen“). Dies ist in Tabelle 5.1 zu erkennen. Die Rotoren sind von I bis VIII (1 - 8) gekennzeichnet. Die Buchstaben des Alphabets deuten den Eingangskanal, die Verkabelung für jeden Rotor darunter zeigt die Substitution für jeden Buchstaben an. Zum Beispiel, Rotor I transformiert A auf E und Z zu J.

Die Kerbenposition nennt die Stellung des Rings im Alphabet. Der erkennbare Buchstabe ist im Fenster der Rotorstellung für den Benutzer zu sehen und indiziert, wenn die Kerbe beim Fortbewegungsmechanismus eingerastet ist. Die Statoren verfügen über Verdrahtungsringe, jedoch nicht über Kerben und wirken somit wie Rotoren ohne Fortbewegung. „Dünne“ Umkehrwalzen B und C waren bei der Enigma als Beta- und Gamma-Rotoren nutzbar. Alle Angaben wurden in Ringstellung „A“ mit alphabetischer Reihenfolge und eingehendem Strom gemäß der allgemeinen Definition ermittelt [55]. Die Rotorverdrahtung wird mittels Enigma implementiert und gemäß Tabelle 5.1 übernommen.

Rotor Nr.	Innere Verdrahtung (Permutation)	Kerbenpos.
Alphabet	ABCDEFGHIJKLMNOPQRSTUVWXYZ	
I	EKMFLGDQVZNTOWYHXUSPAIBRCJ	Q
II	AJDKSIRUXBLHWTMCQGZNPYFVOE	E
III	BDFHJLCPRTXVZNYEIWGAKMUSQO	V
IV	ESOVPZJAYQUIRHXLNFTGKDCMWB	J
V	VZBRGITYUPSDNHLXAWMJQOFECK	Z
VI	JPGVOUMFYQBENHZRDKASXLICTW	Z,M
VII	NZJHGRCXMYSWBOUFAIVLPEKQDT	Z,M
VIII	FKQHTLXOCBJSPDZRAMEWNIUYGV	Z,M
b	LEYJVCNIXWPBQMDRTAKZGFUHOS	nur M-4
g	FSOKANUERHMBTIYCWLQPZXVGJD	nur M-4

Umkehrwalze	Innere Verdrahtung	Kerbenpos.
B	YRUHQSLDPXNGOKMIEBFZCWVJAT	
C	FVPJIAOYEDRZXWGCTKUQSBNMHL	
B „dünn“	ENKQAUYWJICOPBLMDXZVFTHRGS	nur M-4
C „dünn“	RDOBJNTKVEHMLFCWZAXGYIPSUQ	nur M-4

Tabelle 5.1: Rotorenverdrahtung bei Enigma I, M3, M4

Mit diesen Eigenschaften ist es möglich eine Chiffriermaschine zu entwickeln und diese in der Programmierumgebung „MS Visual Studio“ zu implementieren.

5.1.1 Klasse – Typex

Mit den zuvor genannten Prämissen können nun die Programmkonstanten des Simulators definiert werden. Diese sind in Tabelle 5.2 ersichtlich.

Klasse	Datentyp	Variablennamen	Erläuterung
	Rotor[3]	rotors	Enthält die ausgewählten Rotoren mit allen Attributen eines Rotors
	Rotor[2]	stators	Beinhaltet die Statoren ohne Kerben d.h. keine Rotation
	Rotor	reflector	Reflektor (Umkehrwalze) mit den typischen Attributen eines Rotors. Ein Reflektor verbleibt in seiner Position d. h. ohne Drehung
	List<int>	positiveRotors/ positiveReflectors	Liste von allen Verdrahtungen I bis VIII, RotorWiring beinhaltet die Verdrahtung und die Kerbenpositionen
	string	rotorIconf - rotorVIIIconf	Verdrahtung des Alphabets für den Rotor 1-8 (Abb.: 5.1)
	string	reflectorAconf - reflectorCconf	Reflektor A,B,C Verdrahtung
Rotor	char	outerChar	Aktueller Buchstabe des Rotors, welcher von außen ersichtlich ist für den Benutzer
Rotor	int	outerPosition	outerChar transformiert als Zahl gemäß ASCII-Tabelle
Rotor	string	wiring	gewählte Verdrahtung des aktuellen Rotors
Rotor	boolean	reversed	Rotorenrichtung (Umgedreht?)
Rotor	char[]	notches	Kerbenpositionen als Zeichenfolge – Bei welchen Buchstaben der Rotor sich dreht
Rotor	string	name	Rotorenname
Rotor	char	ring	Ringstellung in Abhängigkeit zur Kerbenposition und aktueller Position des Rotors
Rotor	int[26]	map	Distanz zwischen den Alphabeten und der Permutation für jede Buchstabenstelle
Rotor	int[26]	revMap	Gegenteil zu map, nur umgedreht.
Rotor	int[26]	map	Distanz zwischen den Alphabeten und der Permutation für jede Buchstabenstelle
Rotor	int[26]	revMap	Gegenteil zu map, nur umgedreht.

Tabelle 5.2: Programmkonstanten des Simulators

Die Typex-Maschine ist in die Klasse *Rotor* und *RotorWiring* (Rotor-Verdrahtung) aufgegliedert. Alle Klassen sind mit Methoden versehen, die im Folgenden erläutert

werden. Die Struktur dieser Methoden und Klassen ist in Abbildung 5.5 ersichtlich. Die Programmvariablen werden in dem Klassendiagramm nicht gezeigt, da sie schon in Tabelle 5.2 aufgelistet sind.

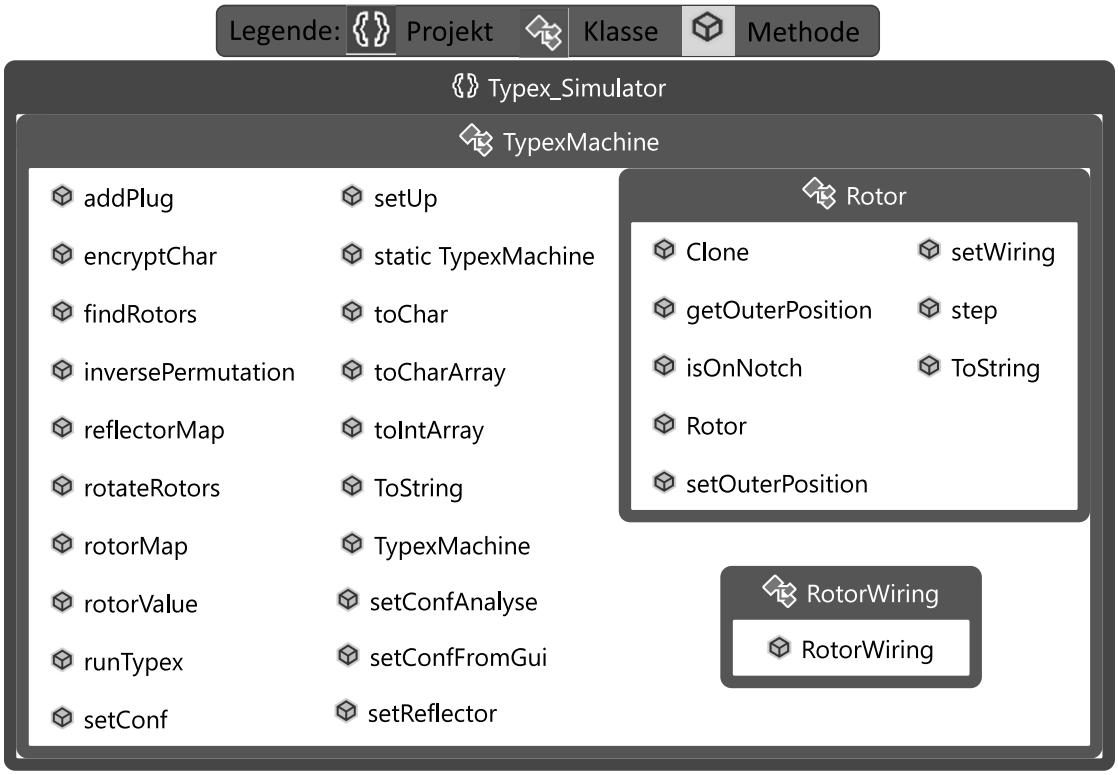


Abbildung 5.5: Klassendiagramm der Klasse – Typex

Die Implementierung der Rotorverdrahtung muss performant sein, denn sonst die Analyse auch darunter leidet. Der Rotor ist ein Mechanismus zur einfachen Substitution von Buchstaben, wo jeder Buchstabe durch einen anderen Buchstaben, der eine bestimmte Distanz im Alphabet entfernt liegt, ersetzt wird. So wird z. B. mit einer Distanz von 4 der Buchstabe A zu E, B zu F, C zu G substituiert.

Im Allgemeinen enthält ein Rotor zwei drehbare Scheiben. Es gibt eine innere und eine äußere Scheibe, beide mit 26 Positionen, die jeweils einen Buchstaben im Alphabet darstellen [11]. Durch Drehen der inneren Scheibe entgegen der äußeren Scheibe verschiebt sich die Distanz. Im Grunde wird jeder Buchstabe durch seinem $n - sten$ Nachbarn im Alphabet substituiert.

Der schwierigste Teil ist das Umkehren der Rotoren, da die Verschlüsselungen zweifach durch den Rotor verändert werden. Der reale Rotor wurde nur gedreht und konnte direkt in die Typex eingesetzt werden. Innerhalb des Programms bedeutet das eine Art Neuverdrahtung des Rotors. Dies wird erreicht durch die ASCII-Tabelle, Subtraktion und Modulo Berechnung.

Die Methode $setWiring()$ startet mit der Übersetzung der aktuellen Position (Zahl) eines Buchstabens ($A = 65, B = 66, \ldots, Z = 90$). Die For-Schleife be-

```
1  public void setWiring(string wiring)
2  {
3      outerChar = wiring.ToCharArray()[outerPosition];
4
5      for (int i = 0; i < 26; i++)
6      {
7          int match = ((int)wiring.ToCharArray()[i]) - 65;
8          map[i] = (26 + match - i) % 26;
9          revMap[match] = (26 + i - match) % 26;
10     }
11 }
```

Textauszug 5.1: Methode *setWiring*() - Distanzen der Verdrahung bestimmen

ginnt mit der Bestimmung der Variable *match* indem das ganze Array von Anfang durchlaufen (Variable i) und die Distanz zu jedem einzelnen Buchstaben berechnet wird (siehe Ausschnitt 5.1). Wenn A die aktuelle Position ist und laut Verdrahtungstabelle A auf E abgebildet wird, ist eine Distanz von 4 an der ersten Stelle des Array „*map*“ abgespeichert. Die Umrechnung von Zahl auf Buchstabe erfolgt über die ASCII-Tabelle. Buchstabe A hat die Position 65 in der ASCII-Tabelle und wird daher nach jeder Umrechnung abgezogen, damit die Äquivalenzklasse ($N = 26$) des Alphabets erreicht wird. Die Äquivalenzklasse ist immer Auslöser für modulo-26 Berechnungen, da sonst die Zahl außerhalb des Alphabets liegt.

Das Gegenstück ist das Array (*revMap*) mit gegenläufigen Werten, d. h. der Rotor wird umgedreht und die vorherige Ausgangsreihenfolge besteht nun aus Eingangswerten. Hierzu lässt sich auch eine Distanz ausrechnen. Der Strom fließt immer über einen anderen Buchstaben zurück. Daher werden auch diese umgedrehten Distanzen benötigt.

Die Variable *outerPos* repräsentiert die aktuelle Position als Zahl. Jede Rotation verlangt die Aktualisierung dieser Variable. Die Umrechnung erfolgt über die ASCII-Tabelle mit Addition oder Subtraktion der Zahl 65, welche garantiert, dass die Position weiterhin innerhalb des Alphabets liegt. Diese Referenzeigenschaft wird durch die Set-Methode (*setOuterPosition*()) der Variable gewährleistet.

Eine Drehung (*step*()) zum nächsten Buchstaben erfolgt bei jeder Buchstabeneingabe und über die Variable der aktuellen Position. Im Falle der Drehung wird dieser Zähler um eins inkrementiert und dessen Referenz aktualisiert.

Jeder Rotor ist mit mindestens einer Kerbe versehen, welche eine Drehung der benachbarten Rotoren veranlasst. Im Programm sind solche Kerben durch eine Vielzahl von *notches*-Positionen gekennzeichnet. Sie werden durch die Methode (*rotateRotors*()) bei einer Drehung abgefragt. Die Kerbenpositionen sind durch eine Zeichenfolge von Buchstaben bestimmt. Bei einer Drehung werden alle hinterlegten Kerben des Rotors durchlaufen, um eine erforderliche Drehung des Nach-

barrotors zu veranlassen. Stimmt die Kerbe mit der aktuellen Position überein, wird eine Drehung der bestimmten Rotoren eingeleitet.

Methoden ($rotorMap()$, $rotorValue()$) verantworten die Verschlüsselung eines Buchstaben im Bereich der Rotoren, der Umkehrwalze und der Statoren. Die Statoren rotieren nicht, sind aber die erste Komponente des Verfahrens. Der Ausgang der zwei Statoren liegt an den Kontakten der Rotoren. Drei Rotoren sind abgekoppelt, da diese einer mechanischen Drehvorrichtung nachgehen. Der Buchstabe, den es zu verschlüsseln gilt, durchläuft diese Methode zweimal, Hinweg und Rückweg des Chiffriervorgangs. Die Berechnung erfolgt auf dem Array mit den Distanzen bzw. Verdrahtungen.

Nach den Rotoren wird das Signal, innerhalb der Umkehrwalze, umgelenkt. Dieser Teil verbleibt auf der ursprünglichen Position und leitet lediglich das Signal durch die Rotoren und Statoren zurück. Die Methode der Rotoren und Statoren wird diesmal mit umgedrehter Reihenfolge aufgerufen.

Die Initialposition der Rotoren werden über eine überlagerte Methode ($setConf()$) im Datenmodell abgelegt. Die Verschlüsselung bzw. Entschlüsselung wird mit der gewünschten Nachricht (Zeichenfolge) angestoßen und kann ebenfalls auf einzelnen Buchstaben basieren.

Die Methode $setConf()$ übernimmt das physikalische Einsetzen eines Rotors in die Maschine. Die Reihenfolge und Rotorkennziffer wird in einer Zeichenfolge angegeben und anhand des Bindestriches identifiziert. Die Angabe erfolgt im römischen Zahlensystem und wird hier mit einer Hilfsklasse ($RomanNumeral$) in eine natürliche Zahl überführt. Der Rotor kann auch umgekehrt eingesetzt werden, welches mit einem nachstehenden „R“ Buchstaben gekennzeichnet ist. Das „R“ wird aus dem Eingabestring entfernt und als solches im Datenmodell hinterlegt. Sofern Reihenfolge und der Typ des Rotors identifiziert sind, kann ein neues Objekt hierfür erzeugt werden. Das Programm erstellt nur für die aktuellen selektierten Rotoren eine Rotor-Verdrahtung ($findRotors()$). Dasselbe trifft auf die umgekehrten Rotoren zu. Die Hilfsmethode $inversePermutation()$ berechnet das Inverse des Rotors, denn dies entspricht einer physikalischen 180°-Grad-Drehung. Im Ausschnitt 5.2 wird die Inverse (iw) von der ursprünglichen Permutation (w) berechnet.

Die implementierte Methode invertiert die Permutation, mit Eingabe aus Buchstaben. Diese wird nach bekanntem Verfahren mit der ASCII-Tabelle in eine natürliche Zahl überführt und schließt mit der Inversen-Berechnung der Permutation ab. Die Zahlenwerte können wieder auf einen Buchstaben abgebildet werden. Somit ist die Berechnung der Inversen auf einer beliebigen Zeichenfolge über dem lateinischen Alphabet möglich.

Die Hilfsklasse $RomanNumeral$ konvertiert eine römische Zahl in eine natürliche Zahl. Diese Berechnung wird benötigt, um die Rotornummern, welche in römischen Zahlen durchnummeriert sind, effizient und ohne weitere Schleifen festzustellen.

```
1  public static int[] inversePermutation(int[] w)
2  {
3      int[] iw = new int[w.Length];
4
5      for (int k = 0; k < w.Length; k++)
6      {
7          iw[w[k]] = k;
8      }
9      return iw;
10 }
```

Textauszug 5.2: Methode *inversePermutation*() – Inverse einer Permutation berechnen

5.1.2 Klasse – Program (Typex-Simulator)

Die Programm-Datei bereitet die vom User übergebenen Werte auf und übergibt diese zum Ver- und Entschlüsseln an die Typex-Klasse. Der Hauptfokus liegt darin, die Maschine mit validen Dateneingaben aufzurufen, damit keine Fehleingabe zum Absturz des Programms führt.

Es wird ein Alphabet von 26 Buchstaben (A-Z) angenommen und allgemein als Alphabet vom Simulator definiert. Generell ist ein kryptographisches System über ein Alphabet mit allen möglichen Zeichen definiert, mit dem die Klartexte verschlüsselt werden. Ein 26-Buchstaben-Alphabet ist natürlich nicht sehr nützlich für ein modernes System, aber innerhalb des Programmcodes trägt es dazu bei, das lateinische Alphabet zur ASCII-Tabelle zu referenzieren und mit dem Zahlenrepräsentanten zu rechnen.

Im verkürzten Zeichensatz der Buchstaben bei allen Rotoren sollen auch Kleinbuchstaben berücksichtigt werden. Einfachheitshalber werden diese zum identischen Partner im Großbuchstabenalphabet überführt. Zu beachten ist, dass dies der einzige Informationsverlust innerhalb der Nachricht ist, ohne zu wissen, ob ein Groß- oder Kleinbuchstabe vorliegt. Dieser Verlust ist aber harmlos, denn Leser können einen syntaktisch korrekten Text ohne Kleinbuchstaben verstehen.

Die *main*-Funktion ruft die Klasse der Maschine *TypexMachine* selbst sowie die dazugehörige Konfiguration *TypexConfiguration* auf und initialisiert eine Instanz mit Standartwerten der jeweiligen Klasse. Die Methode *readSettings*() bietet dem User einige Einstellmöglichkeiten für den anstehenden Chiffriervorgang. Sie liest die übergebenen Argumente ein und übergibt diese an die Methode *convertSettings*(). Diese konvertiert und validiert die Werte. Sofern die eingelesenen Werte nicht valide sind, wird der Benutzer auf diese falsche Konstellation der Eingabe aufmerksam gemacht.

Nachdem die Konfiguration selektiert wurde, kann die zu ver-/entschlüsselnde Nachricht verarbeitet werden. Nach obiger Konvention wird der eingegebene Text

in Großbuchstaben überführt. Dieser modifizierte Text wird mit der Chiffriermethode *runTypex*() von Maschinenklassen aufgerufen. Die Methode speichert das Ergebnis der Berechnung in einer Variable ab und hält diese für weitere Berechnungen vor. Nachdem die Maschine wieder auf den Initialzustand zurückgesetzt wurde, ruft das Programm die Chiffriermethode erneut auf und erzeugt damit eine Klartextversion. Sofern die eingegebene Zeichenfolge und die Klartextversion übereinstimmen, ist das zu verschlüsselnde Ergebnis nach Konvention korrekt und kann daher ohne Probleme an ein Zielsystem übertragen werden.

Die Konfigurationsmethode *convertSettings*() wird ausgeführt, wenn der Benutzer die Grundkonfiguration selbst definieren möchte. Die weiteren Abfragen validieren die Ringstellung der einzelnen Verdrahtungen. Die Initialposition des Rotors bestimmt, welche Rotoren und welche Umkehrwalze eingesetzt werden. Die Rotoren werden mit dem jeweiligen Nummer eingegeben und mit Kommas separiert, wobei das „-“ (negative Nummer) einen umgedrehten Rotor markiert. Der Benutzer bestimmt die Umkehrwalze durch die entsprechende Kennziffer. Tabelle 5.3 zeigt die Variablen, die zur Validierung und Ausführung benötigt werden. Die Parameter sind als Variablen im Programmcode angelegt.

Parameter	Argument	Erklärung	Beispiel
-i (rings)	5 Buchstaben	-i definiert die Ringstellung. Sie besteht aus fünf Buchstaben (A-Z)	AAAAA, TYPEX
-b (rotorbegin)	5 Buchstaben	-b gibt die Grundstellung an. Es sind fünf Buchstaben (A-Z) anzugeben	AAAAA, TYPEX
-r (rotororder)	3 Zahlen	-r definiert die Rotorwahl und -reihenfolge. Sie besteht aus drei Zahlen über den verfügbaren Rotoren von -8 bis 8 (Walzensatz)	-1,2,-3
-s (statororder)	2 Zahlen	-s gibt die Statorwahl und -reihenfolge an. Dies sind zwei Zahlen aus dem Walzensatz von -8 bis 8	4,-5
-u (reflector)	1 Buchstabe	-u bestimmt die Umkehrwalze. Es ist ein Buchstabe (A-C) anzugeben	B
-m (message)	Buchstaben	-m übergibt die Geheimtextnachricht. Dies kann eine beliebige Buchstabenfolge (A-Z) sein.	HALL...

Tabelle 5.3: Programmkonstanten und Ausführungsparameter für das Projekt Typex-Simulator

Sofern der Benutzer keine individuelle Modifikation an der Initialposition vornehmen möchte, kann dies über die spezifische Eingabe „-d“ umgangen werden. Damit werden die Standardeinstellungen des Programms geladen. Die Ringstellung bzw. Verdrahtung befindet sich auf dem Buchstaben „A“, die Rotoren werden auf die Grundstellung gesetzt und die Kerbenpositionen wird aus der Vorlage übernommen. Die eingebettete Konfigurationsklasse $TypexConfiguration$ speichert

die gewählten Attribute der Chiffriermaschine. Entweder werden die benutzerspezifischen Eingaben in den Variablen (Ringe, Grundstellung, Reihenfolge Rotor und Stator, Umkehrwalze) abgespeichert oder mit den Standartwerten des Simulators ($setDefauflt()$) überschrieben.

Das Programm Typex-Simulator lässt sich mit dem folgenden Beispielaufruf mit den gegebenen Parametern problemlos ausführen:

```
Typex-Simulator.exe -i AAAAA -b TYPEX -r -1,2,-3 -s 4,-5 -u B
                    -m HALLOTYPEX
```

Die Datei gibt eine Meldung für gültige Werte aus. Der Benutzer wird bei ungültigen Werten darauf hingewiesen. Die Konsoleneingabe 5.3 zeigt den Typex-Simulator beim Ausführen des oben genannten Befehls.

```
C:\Release>Typex-Simulator.exe -i AAAAA -b TYPEX -r -1,2,-3 -s
   4,-5 -u B -m HALLOTYPEX
==========Typex-Machine Simulator==========
Input data is valid

Input text:     HALLOTYPEX
Encrypted:      FFPZGSTUJE
Decrypted:      HALLOTYPEX
```

Textauszug 5.3: Konsolen-Ausführung des Projektes Typex-Simulator

5.2 Das Typex-Analysewerkzeug

Dieses Kapitel erklärt die Umsetzung und die Implementierung der Typex-Analysekomponente. Das Analysewerkzeug besteht aus zwei Projekten und ist strukturell ineinander verschachtelt. Das Kryptoanalyse-Projekt versucht die gegebene Zeichenkette zu dechiffrieren und dessen Schlüssel (Grundstellung) anzuzeigen. Das Evaluation-Projekt ist nur für die Erstellung dieser Arbeit notwendig, da es im Allgemeinen nur Texte mit unterschiedlichen Längen für eine Erfolgsrate generiert.

5.2.1 Projekt – Kryptoanalyse

Im Folgenden wird das Analysewerkzeug aus einer technischen Perspektive beschrieben. Es dient als Leitfaden für den Programmcode. Der Ablauf korrespondiert mit den technischen Klassen-/Variablennamen und erklärt deren Funktionalität und Bestandteil im Programm. Die Grundlage für die Programmierung wurde bereits im theoretischen Ansatz in Kapitel 4.2.2 erläutert. Diese allgemeine Gliederung reflektiert sich entsprechend auf die Programmstruktur. Abbildung 5.6 zeigt

die beinhalteten Klassen, welche ein detailliertes Diagramm zu den Methoden und Variablen in den folgenden Abschnitten erhalten.

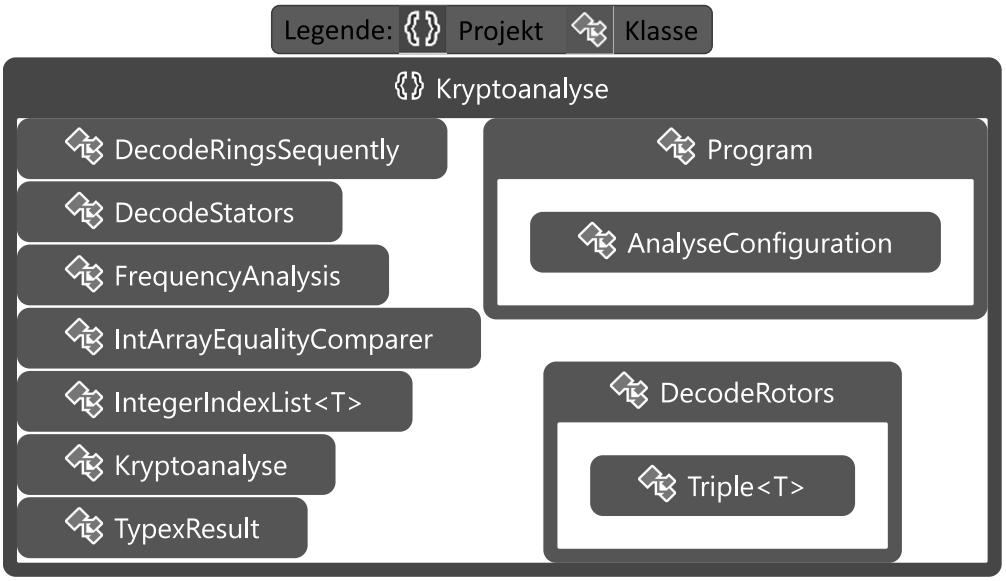


Abbildung 5.6: Klassendiagramm des Projektes – Kryptoanalyse

Das Kryptoanalyse-Projekt beinhaltet verschiedene Strategien, für den sukzessiven Suchlauf nach dem Schlüssel für ein gegebenen Zeichenstring. Das Projekt erfordert einige Rahmenbedingungen, wie das Festlegen auf einen Suchmodus und das Definieren der bekannten Verdrahtungen. Entweder sind die Rotoren bekannt und die Statoren werden gesucht oder die Statoren sind bekannt und die Rotoren stellen die Suchbedingung dar. Die Rotoren oder Statoren werden mit der Grund- und Ringstellung im Programm definiert, denn nur so ist in akzeptabler Zeit ein Ergebnis zu erwarten.

5.2.1.1 Klasse – Kryptoanalyse

Die Kryptoanalyse-Datei ist der zentrale Ausführungspunkt für die nachgelagerten Algorithmen beim Stator- oder Rotorsuchlauf. Abbildung 5.7 zeigt die entwickelten Methoden und deren benötigte Variablen.

Die Details der bekannten Schlüsselteile werden an oberster Stelle als Klassenvariablen $fixedStators$ oder $fixedRotors$ definiert. Die Struktur ist in einem dreiteiligen Array organisiert, welches in der Reihenfolge Rotor, Grundstellung, Ringstellung befüllt wird.

Gemäß dem theoretischen Konzept werden zwei nachgelagerte Instanzen von Phase 1 und 2 im Programm gestartet. Jede einzelne Instanz resultiert in einer Bestenliste, deren Größe vom Benutzer definiert und als Kandidatengrundlage an die nachgelagerte Instanz übergeben wird. Als Standard-Größe bei den Bestenlisten

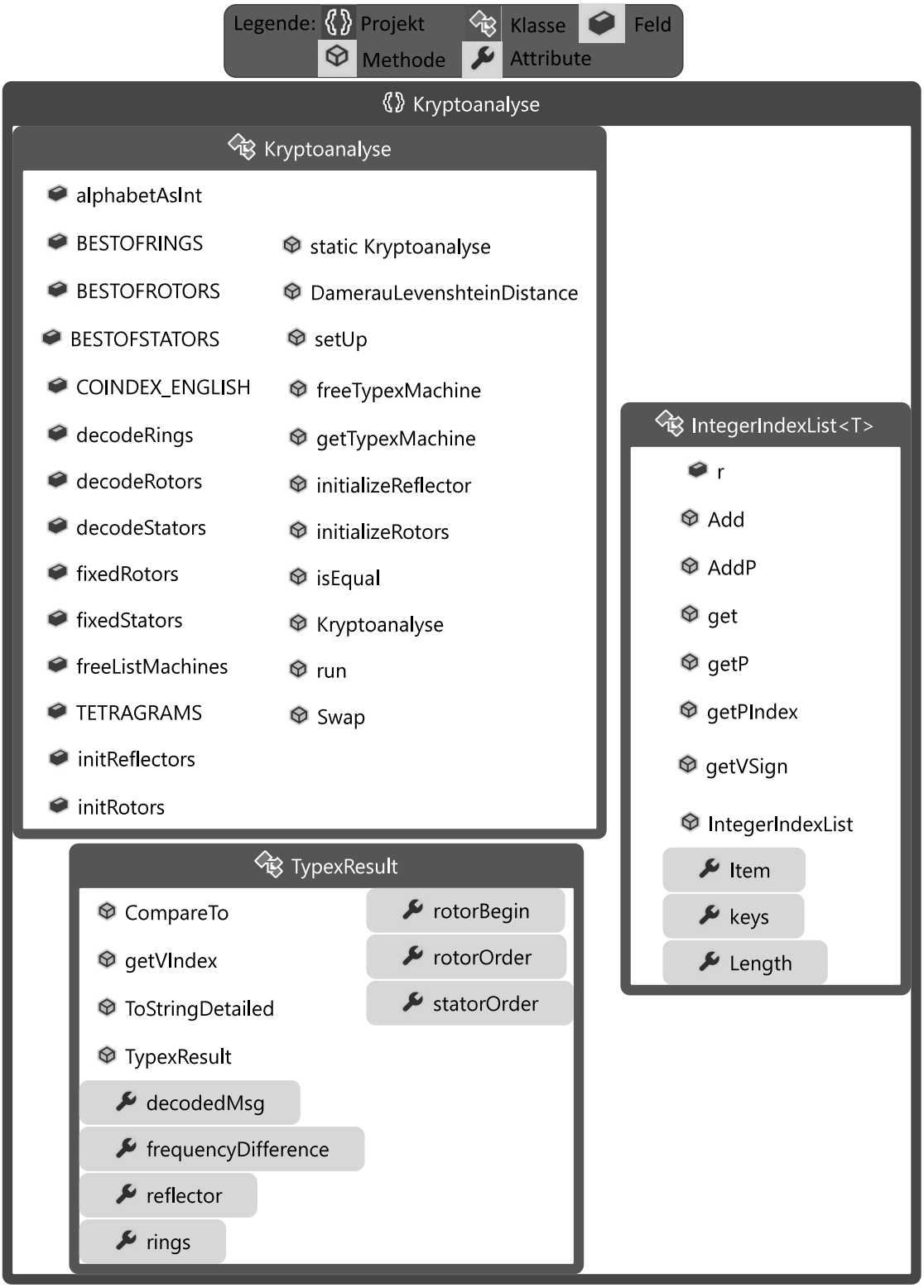


Abbildung 5.7: Klassendiagramm der Klasse – Kryptoanalyse (Analysekomponente)

für Rotoren, Statoren und Ringstellungen ist die „3“ hinterlegt. Dieser Wert harmonisiert zwischen einer guten Trefferrate und einer akzeptablen Laufzeit. Mit steigender Größe verlängert sich die Laufzeit, da eine komplette Ringsuche auf jedem Kandidaten aus der Bestenliste durchlaufen wird.

Die weiteren Klassenvariablen sind Instanzen zur Entschlüsselung der Rotoren, Statoren und Ringe. Die Instanzen sind zentral in der führenden Datei „Kryptoanalyse“ abgelegt, denn so ist eine generelle Verbindung übergreifend hergestellt. Zur allgemeinen Information ist das lateinische Alphabet mit 26 Buchstaben als Zahlenliste hinterlegt. Auf dieser Liste basieren viele Berechnungen.

Die Hauptmethode *run*() stößt den gesamten Kryptoanalyse-Prozess an. Es erfordert zum einen den Geheimtext als Eingabe und zum einen die Information, ob die Statoren oder Rotoren entschlüsselt werden. Der Ciphertext-Only Angriff wird mit dem Geheimtext und den bekannten Variablen ausgeführt. Zuerst werden die Ressourcen für die anstehende Berechnung mit der Methode *setup*() angefordert. Sie umfasst Typex-Maschine, Rotoren, Statoren mit Chiffrierfunktion. Rotoren und Statoren sind baulich gleich und vollständig mit dem Verschiebungsabstand initialisiert. Diese werden mit unterschiedlichen Postionen in die Typex eingesetzt. Dieselbe Konfiguration wird bei den Umkehrwalzen angewendet und garantiert damit eine höhere Performance. Die Methode *initializeRotors*() übernimmt diese Erstellung der Rotorverdrahtung und Kerbenpositionierung. Der Prozess generiert im Standard alle Rotoren (vorwärts und rückwärts) und kann optional auf die Erstellung der Rotoren mit Vorwärtsrichtung eingeschränkt werden (Tabelle 5.4).

Die Methode *isEqual*() prüft gemäß Tabelle 5.4 ob ein Rotorpaar vorliegt. Jede gerade Indexzahl in der Rotorliste repräsentiert die Rotoren in Vorwärtsrichtung. Jede ungerade Zahl korrespondiert zu den umgedrehten Rotoren in Rückwärtsrichtung.

Die Parallelverarbeitung beschleunigt das Programm so, dass immer gleiche Algorithmen mit kleinen Eingabeänderungen schneller durchläuft. Damit nicht bei jeder Dechiffrierung eine neue Typex instanziiert und nach Ablauf verworfen wird, gibt es eine Sammlung von Typex, deren Maschinen von einem Prozess angefordert und wieder freigegeben werden. Hierzu wird sich der Klasse „ConcurrentQueue“ mit dem „first-in, first-out (FIFO)“-Prinzip bedient, da diese als „thread“ sicher in den „.Net-Bibliotheken“ definiert sind. Diese Eigenschaft ist wichtig, da sonst verschiedene Prozesse eine gleiche Maschine zugewiesen bekommen und somit die Dechiffrierung deutlich verfälscht. Die Einreihung und Entnahme aus der Liste werden mit den Methoden $getTypexMachine()$ und $freeTypexMachine()$ realisiert. Beim Einreihen sind die Konfigurationen zurückzusetzen.

Interval-Heap ist eine Erweiterung des Min- und Max Heap-Prinzips und erlaubt das Einfügen und Löschen von Elementen in $O(\log(n))$, wobei n die Anzahl der Elemente im geordneten Binärbaum ist. Wenn die Anzahl n eine gerade Zahl ist, besitzt jeder Knoten genau zwei Nachfolger. Bei ungerader Anzahl ist jeder

Knoten ausbalanciert bis auf den letzten Knoten, der nur einen Nachfolger hat. In Abbildung 5.8(a) wird ein Baum mit 11 Elementen gezeigt, worin der graue Knoten das letzte Element repräsentiert. Jeder Knoten, bis auf den letzten, hat zwei Nachfolger und ergibt eine Summe von 6 Knoten. Das linke Ende eines Knotens definiert den min-Heap und die rechte Seite zeigt den max-Heap. Diese Eigenschaft kreiert folgende Funktionen auf der Datenstruktur:

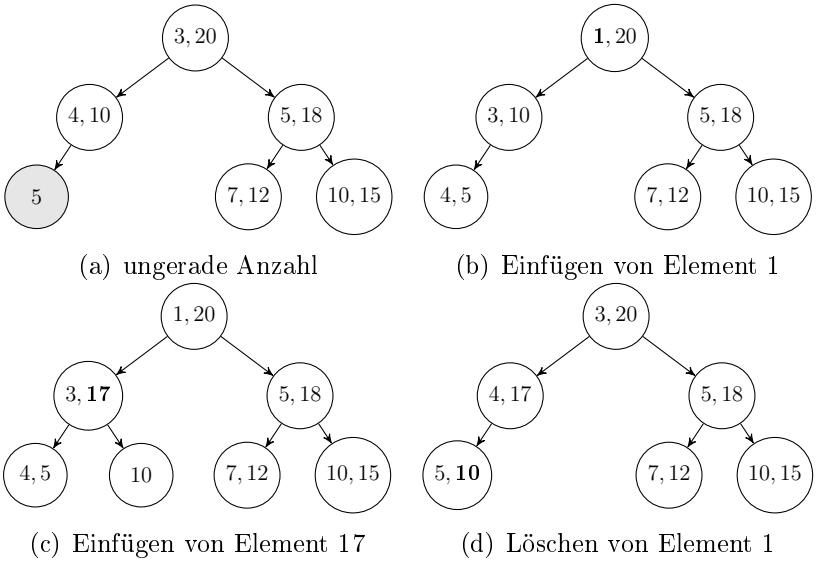
- *Add*()
- $FindMin()$ und $FindMax()$
- $DeleteMin()$ und $DeleteMax()$

Es wird angenommen, dass ein Ergebnis aus der Typex mit dem Wert 1 für einen Schlüsselkandidaten in den Interval-Heap eingegliedert. Hierfür wird die Methode *Add*() aufgerufen und das Element beim alleinigen Konten 5 eingefügt. Damit die Eigenschaften des Interval-Heaps (Min-/Maxheap) eingehalten sind, wird der Pfad vom Element bis zum Ausgangsknoten durchlaufen. Somit drängt das neue Element zuerst die 4 und dann die 3 um eine Ebene herab und resultiert in Abbildung 5.8(b). Das Hinzufügen des Ergebnisses 17 erstellt einen Nachfolger und reorganisiert den Graphen innerhalb des Pfades (Abbildung 5.8(c)). Das Hinzufügen eines Elementes lässt sich in drei Fälle aufteilen: Sofern das neue Element im Intervall des Vorgängerknotens passt, verändert sich die Struktur nicht. Wenn das neue Element nicht im Intervall liegt, wird entweder das Konzept des Minheaps auf die linke Seite angewendet oder das vom Maxheap auf die rechte Seite. Dies wird entschieden nach dem nicht konformen Intervallende. Diese Eigenschaft reduziert den Vorgang auf $O(\log(n))$ und eignet sich zur Sortierung der Bestenliste beim Entschlüsseln der Rotoren oder Statoren. Jedes Ergebnis eines Schlüsselkandidaten wird einmalig hinzugefügt. Bei der Laufzeit wird entschieden, ob dieser Wert in der Bestenliste weiterhin gehalten wird.

Zum Reduzieren des Speichers darf die Bestenliste mit dem Interval-Heap nur bis zu einer definierten Konstante anwachsen. Die nicht relevanten Ergebnisse werden verworfen. Das Entfernen des minimalen oder maximalen Ergebnisses reduziert den Baum um den Knoten mit aktuell einem Wert. Der Wert im Ausgangsknoten wird durch den gelöschten Knotenwert ersetzt und anschließend reorganisiert. Da das Element 1 als Minimalwert gespeichert war, wird das Minheap-Konzept durchlaufen, bis die Konstellation $10, 5$ wegen der Prämisse $10 \not< 5$ kollidiert. Dies erzwingt das Durchlaufen desselben Pfades mit dem Maxheap-Konzept. Es entsteht Abbildung 5.8(d) in $O(\log(n))$.

Das Auffinden des minimalen und maximalen Wertes innerhalb der Datensammlung erfolgt durch direktes Abrufen des Ausgangsknotens in $O(c)$.

Das gezeigte Verhalten eines Interval-Heaps ist ideal zur performanten Erstellung einer Bestenliste und garantiert das tatsächlich kleinste Ergebnis stets auszugeben, welches im generellen Fall die korrekte Grundstellung für den gesuchten Klartext beinhaltet.



(a) ungerade Anzahl (b) Einfügen von Element 1

(c) Einfügen von Element 17 (d) Löschen von Element 1

Abbildung 5.8: Interval-Heap

TypexResult ist eine Datenstruktur als Hilfsklasse angelegt. Sie erbt die Funktionalitäten der Vergleichsklasse *IComparable*, welche die Implementierung einer Vergleichsmethode erfordert. Der Vergleich erfolgt auf Basis der Kostenfunktion, welche eine statistische Analyse über den aktuellen Textkandidaten berechnet und sodann aufsteigend sortiert. Die Hilfsklasse speichert alle relevanten Informationen (entschlüsselte Nachricht, Häufigkeitsmaß, Ringstellung, Grundstellung, Rotorreihenfolge, Umkehrwalze) aus der jeweiligen Entschlüsselungsphase zu einem Ergebnis ab.

IntegerIndexList ist eine Liste im Bereich der ganzen Zahlen $\mathbb{Z}$. Sie hat den Vorteil, dass der Index nicht von null startet, sondern den negativen Zahlenbereich einschließt. Hierfür wird der ursprüngliche Index auf einen geraden/ungeraden Index transformiert, welche als virtueller Index $-\infty \geq x \leq +\infty$ und physikalischer Index $0 \leq x \leq +\infty$ definiert ist. Die Transformation vom virtuellen zum physikalischen Index erfolgt mit $p(v)$ in Formel 5.1.

$$p(v) = |v \cdot 2| - \ if\ v > 0\ then\ 1\ else\ 0 \tag{5.1}$$

Das umgekehrte Konvertieren vom physikalischen zum virtuellen Index geschieht mit $v(p)$ in Formel 5.2.

$$v(p) = (\ if\ p \mod 2 \equiv 1\ then\ 1\ else\ -1) \cdot \frac{2p+3}{4} \tag{5.2}$$

Diese Relation entsteht durch die Bijektion zwischen den Zahlenmengen, gezeigt in der Tabelle 5.4.

Diese Herangehensweise ist durch den wesentlichen Array-Datentyp einfach strukturiert, allerdings nicht für die Abfrage von großen Datenmengen geeignet. Der

$p(v)$	$\leftrightarrow$	$v(p)$
0	$\leftrightarrow$	0
1	$\leftrightarrow$	1
2	$\leftrightarrow$	-1
3	$\leftrightarrow$	2
4	$\leftrightarrow$	-2
5	$\leftrightarrow$	3
6	$\leftrightarrow$	-3
$\cdots$		$\cdots$

Tabelle 5.4: Array Index über die ganzen Zahlen $\mathbb{Z}$

Gebrauch bei der Typex begrenzt sich auf die Anzahl der Rotoren. Die Rotoren der Enigma (I, II, III, usw.) in Vorwärtsrichtung werden auf die positiven Zahlen abgebildet, dieselbe Verdrahtung in Rückwärtsrichtung ist dem negativen Bereich zugeordnet. Position 0 erhält keine Zuordnung von einem Rotor. Das Hinzufügen und Entfernen von Rotoren geschieht im Typex-Simulator (Tabelle 5.2). Er definiert die technische Größe der IntegerIndexList.

5.2.1.2 Klasse – DecodeRotors und DecodeStators

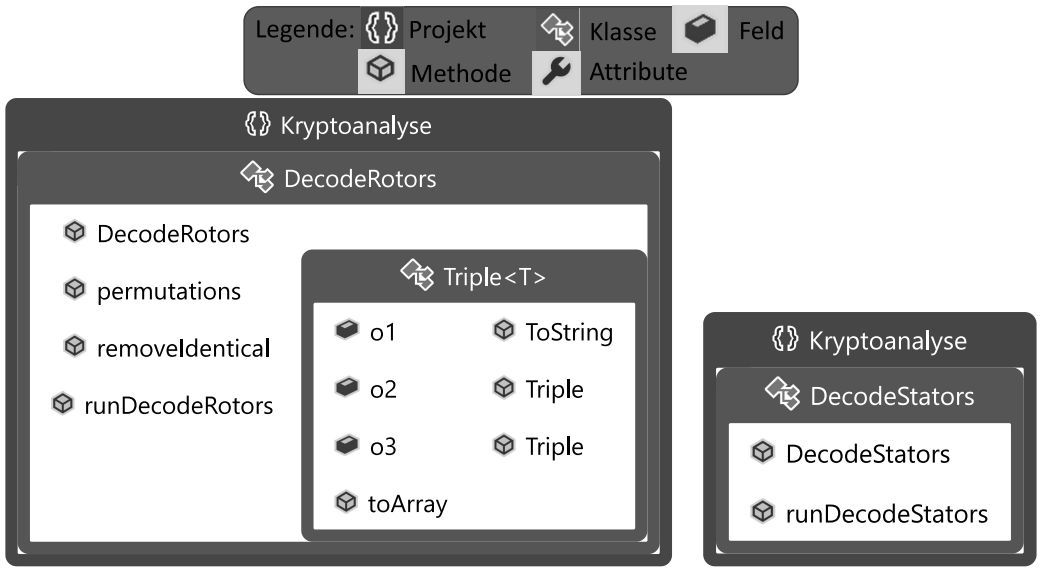
Das Ziel der beiden Klassen ist es, die jeweils richtige Rotorauswahl und Grundstellung der drei Rotoren bzw. zwei Statoren zu finden (analog zu Kapitel 4.2.2.1). Hierin liegt der generelle Unterschied der Analyse. Die Programmstruktur von *DecodeRotors* und *DecodeStators* ist sehr ähnlich. Daher werden beide Klassen in der Erklärung zusammengefasst.

Die strukturelle Perspektive auf die Klassen ist in Abbildung 5.9 ersichtlich. Die innere Klasse *Triple* wird von beiden Suchläufen (Rotor und Stator) benötigt. Die Methoden *permutations*() und *removeIdentical*() werden auch von der Klasse *DecodeStators* verwendet. Der Programmcode wurde zentral in der Klasse *DecodeRotors* abgelegt.

Das Alternieren der Permutationen, ausgelöst durch die Rotordrehung, beeinflusst den Grad der Verschlüsselung. Bei den feststehenden Statoren beim Annäherungsprozess auf die korrekte Schlüsselstellung schon früh eine deutliche Tendenz in den Messwerten zu erkennen, gezeigt in Tabellen 4.3 - 4.6.

Der beträchtliche Unterschied der beiden Klassen liegt in der Anzahl der möglichen Verdrahtungen, denn drei Rotoren bedeuten einen deutlich größeren Suchraum als zwei Statoren. Eine deutliche Verkürzung der Laufzeit ist erkenntbar und zeigt den Mehraufwand beim Hinzufügen eines Rotors/Stators.

Die Klasse beginnt mit dem Anlegen eines Interval-Heaps zur späteren Priorisierung des Typex-Ergebnisses (*TypexResult*) in der Bestenliste, welches als generelle Rückgabewert nach dem Durchsuchen des Schlüsselraumes dient. Zur Generie-



(a) Diagramm – DecodeRotors

(b) Diagramm – DecodeStators

Abbildung 5.9: Klassendiagramm der DecodeRotors und DecodeStators (Anaylsekomponente)

rung des verbliebenen Schlüsselraums wird die Methode *permutations*() verwendet. Es werden alle möglichen Kombinationen (Reihenfolgen) über die Rotoren und Grundstellungen erzeugt. Diese werden in der Warteschlange für die Verarbeitung eingereiht. Über einen Parameter wird definiert, ob Duplikate von Elementen innerhalb einer Kombination auftreten dürfen. Bei den Rotoren ist es nicht möglich, den gleichen Rotor zweifach einzusetzen. Einmal verwendet, verschwindet dieser aus der Auswahl der möglichen Rotoren. Allerdings ist das Auftreten von Duplikaten bei der Grund- und Ringstellung erwünscht. Daher kann die Kombination $I - I - II$ bei den Rotoren nicht auftreten, aber bei der Grundstellung ist ein AAB möglich. Die Grundstellung mit Duplikaten umfasst $26^3 = 17.576$ Möglichkeiten. Das mehrmals Verwenden eines der 5 Rotoren (vorwärts/rückwärts) ergibt $(5 \cdot 2)^3 = 1.000$ Möglichkeiten. Nach Durchlaufen der Methode wurden alle Duplikate bei den Rotorkombinationen auf $10 \cdot 9 \cdot 8 = 720$ Möglichkeiten reduziert.

Durch die generelle Annahme, dass die Statoren bei der Rotorsuche oder die Rotoren bei der Statorsuche bekannt sind, sind diese in der Klasse Kryptoanalyse definiert und werden nach der Generierung in die Typex-Maschine eingesetzt. Unter der physikalischen Annahme, dass ein Rotor und dessen inverser Partner nur einmal eingesetzt werden können, werden die Kombinationen mit Rotoren und dem inversen Partner aus der Auftragswarteschlange entfernt. Das Entfernen erfolgt mit der Methode *removeIdentical*(). Es prüft wie häufig Kombinationen mit identischen Partnern vorkommen (Tabelle 5.4). Die Kombination $I - I^{-1} - II$ ist ungültig und wird aus der Liste entfernt. Dieser weitere Schritt vermindert die Möglichkeiten auf $6 \cdot 4 \cdot 2 = 48$ Kombinationen, worin z. B. $I - II^{-1} - III$ beinhaltet ist.

Die vorherigen Methoden werden auch beim Suchlauf nach den Statoren durchlaufen. Sie resultieren in einen kleineren Suchraum. Bei den Statoren beläuft sich die Kombination der Grundstellung mit Duplikaten auf $26^2 = 676$ Möglichkeiten. Der gleiche Algorithmus von *permutation*() ergibt bei den Rotoren ohne Duplikate eine Menge von $10 \cdot 9 = 90$ Möglichkeiten. Die Methode *removeIdentical*() minimiert den finalen Suchraum von $4 \cdot 2 = 8$ Möglichkeiten. Bei der Statoranalyse ist der Ablauf auf Grund des geringen Suchraums ohne eine Parallelverarbeitung realisiert. Er bietet aber trotz dessen eine kurze Laufzeit mit korrektem Ergebnis.

Die Häufigkeitsanalyse mit Tetragrammen wird instanziiert und zur späteren Analyse, nach Dechiffrieren des Geheimtextkandidaten, verfügbar gemacht. Sofern die Tetragramm-Bibliothek noch nicht eingelesen ist, erfolgt dies und wird beim Programm zum schnellen Abruf im Arbeitsspeicher bereit gehalten.

Die Suche durchläuft die Auftragswarteschlange, die aus drei Schleifen besteht: Rotorkombination (48), Umkehrwalzen (2) und Grundstellung (17.576). Die Verarbeitung läuft nach dem Konzept der Parallelverarbeitung und wird durch die Klasse „Task“ realisiert, welche die Laufzeit des Angriffs zu verringert. Die Task-Bibliothek basiert auf dem Konzept einer Aufgabe, die einen asynchronen Vorgang darstellt. Der Begriff Aufgabenparallelität bezeichnet eine oder mehrere eigenständige Aufgaben, die gleichzeitig ausgeführt werden. Der „ThreadPool“ erhält die Aufgaben im Hintergrund und bestimmt die Anzahl der Threads. Zusätzlich wird ein Lastenausgleich durchgeführt, der den Durchsatz maximiert. Dies ist lediglich für den Ciphertext-Only Angriff interessant, da die Analyse der Statoren bereits kurze Laufzeiten aufweist. Darüber hinaus verfügt die Aufgabe jeweils über die öffentliche Task<TypexResult>.Result-Eigenschaft, die das Ergebnis der Berechnung enthält. Die Aufgaben werden asynchron ausgeführt und können in einer beliebigen Reihenfolge abgeschlossen werden.

Die Aufrufe in Phase 1 wurden parallelisiert, indem der Suchlauf für verschiedene Grundstellungen auf unterschiedliche Prozessoren aufteilt wurde. Die Algorithmen laufen autonom und jeder Task generiert eine eigene Bestenliste, in der die besten Schlüssel für den Grundstellungsbereich gespeichert sind. Jeweils ein Task wird mit der Suche der Grundstellung für eine feste Umkehrwalze und eine Rotorkombination verknüpft. Die verbleibenden 17.576 Möglichkeiten werden auf die beste Auswahl vermindert und in der Aufgabenergebnisliste final gespeichert. Nach dem Terminieren aller Tasks werden alle Bestenlisten zu einer einzigen Liste, die die besten Schlüssel für alle Grundstellungen enthält, zusammengefügt. Der Zugriff auf externe Variablen ist nicht erforderlich, da sonst Synchronisationsprobleme auftreten können. Dadurch kann die Laufzeit des Algorithmusfaktors durch die Anzahl der eingesetzten Prozessoren verringert werden.

Abbildung 5.10 zeigt die Effizienzsteigerung der Parallelisierung. Die untere graue Gerade $P_{mit}(n)$ repräsentiert den zeitlichen Aufwand (y-Achse) bei gegebener Textlänge (x-Achse) unter Nutzung der Parallelisierung. Die obere schwarze Gerade $P_{ohne}(n)$ nutzt keine Parallelisierung. Die Ausführungszeiten wurden bei einer Typex über eine Rotorauswahl von 5 mit einer Umkehrwalze erhoben. Der linearisier-

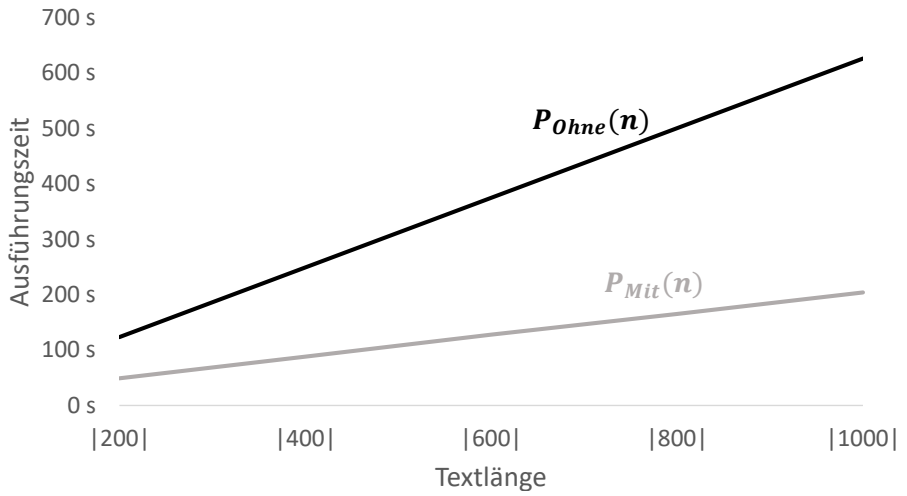


Abbildung 5.10: Parallelverarbeitung bei der Analysekomponente

te Verlauf der Parallelisierung kann als $P_{mit}(n) = 0,2 \cdot n$ und die iterative Verarbeitung als $P_{ohne}(n) = 0,62 \cdot n$ definiert werden. Die Variable n stellt die Textlänge als Eingabewert dar. Der Zeitgewinn $d(n)$ lässt sich über $d(n) = P_{ohne}(n) - P_{mit}(n)$ berechnen. Die Parallelverarbeitung ist bei der großen Datenmenge bis zu drei Mal schneller als der iterative Ansatz.

Final wird eine Typex aus der Klasse Kryptoanalyse (getTypexMachine() und freeTypexMachine()) angefordert und mit der aktuellen Schlüsselkombination bestückt. Der Geheimtext wird entschlüsselt und das Ergebnis mit der Häufigkeitsanalyse bewertet. Nach dem Terminieren von jedes Tasks wird das Gesamtergebnis konstruiert und an die aufrufende Klasse zurückgegeben.

Suche nach den Statoren *oder* Rotoren Tabelle 5.5 zeigt die Veränderung der Häufigkeitsanalyse bei Manipulation von Schlüsselteilen. Die Metriken aus Kapitel 4.3 wurden immer auf den gleichen Geheimtextkandidaten angewendet, allerdings die Schlüsselbestandteile (Rotorposition, Statorposition, Rotor-/Statorgrundstellung) verändert. Der ideale Schlüssel mit den Rotoren $II - I - III$, Statoren $IV - V$ und Grundstellung $HARLA$ entschlüsselte den Geheimtext korrekt und wandelte diesen in natürlichen Text um. Die Metrikanalyse ist in den grauen Feldern zu sehen. Die Bibliotheken für die Metrik Bi-, Tri-, Tetra- und Quintgramme haben sich bereits bei einigen Fällen als erfolgversprechend bewiesen [56].

Die benachbarte Spalte ist als Schlüssel (Rotor $II - I - III$, Grundstellung $HARLA$) bis auf die vertauschten Statoren ($V - IV$) unverändert geblieben. Die Werte verändern sich deutlich, obwohl kein Stator ersetzt wurde, sondern lediglich die Position der zwei vorherigen Statoren vertauscht wurde. Schon hier ist ersichtlich, selbst wenn die größten Teile des Schlüssels bekannt sind, lässt dies immer noch keine signifikante Annäherung auf den korrekten Schlüssel zu. Die nächsten zwei Spalten zeigen die Entschlüsselung mit den Rotoren $III - I - II$, Statoren $IV - V/V - IV$ und der Grundstellung $HARLA$. Diese reflektieren den Fall, dass

nur die Grundstellung des Schlüssels korrekt angegeben ist. In einer der letzten Zeilen und Spalten ist der Wert 0,026494 bei Rotor $III-I-II$, Stator $V-IV$ und Grundstellung $AAAAA$ ersichtlich, welcher den absolut unkorrekten Schlüsselfall repräsentiert. Dieser Fall wird laut Metrik sogar als besserer Kandidat in der Bestenliste geführt als der fast korrekte Fall mit vertauschten Statoren (0,028791). Diese Gegenüberstellung zeigt, dass selbst das kleine Fehlen eines Schlüsselteils einen signifikanten Einfluss auf die Häufigkeitsanalyse darstellt. Dies führte zur Entscheidung, dass die jeweils Suche nach den Statoren und Rotoren ein eigenes Konzept und eine Implementation in unterschiedlichen Klassen benötigt.

In Tabelle 5.6 sind Werte enthalten, die nur mit den Rotoren entschlüsselt und bewertet wurden. Diese Herangehensweise kam auf, als das Problem der Kryptoanalyse der Typex in kleine Teilprobleme untergliedert wurde.

Die Strategie bestand darin die Rotor- und Statoranalyse als gekapselte Probleme zu behandeln und so sukzessive die Rotoren ohne Statorverschlüsselung zu suchen. Allerdings wurde mit der Tabelle bewiesen, dass eine Abkapselung nicht zielführend ist, da sich schon bei einer Entschlüsselung mit korrektem Teilschlüssel (Rotor: $II-I-III$, Grundstellung HAR) mit dem Wert 0,029005 keine korrekte Tendenz herausdeutet. Die Entschlüsselung mit einem total unkorrekten Schlüssel (Rotor: $III-I-II$, Grundstellung AAA) mit dem Wert 0,028480 würde ebenso vorteilhafter in der Bestenliste bewertet werden.

5.2.1.3 Klasse – DecodeRingsSequently

Diese Klasse resultiert im Fund der korrekten Ringstellung der Rotorauswahl (Phase 2 in Kapitel 4.2.2.2). Die Statoren benötigen keine Angabe der korrekten Ringstellung. Diese ist bereits bei der Grundstellung mit einer entsprechenden Verschiebung berücksichtigt. Die Ringstellung jedes Rotors/Stators wird standardmäßig auf A gesetzt. War die Grundstellung E und Ringstellung D der korrekte Schlüssel, so wird nach Durchlaufen der Klasse DecodeStator eine Grundstellung von B mit Ringstellung A gefunden. Diese entschlüsselt trotz alledem den Geheimtext korrekt zu Klartext.

Die Rotoren benötigen in bestimmten Etappen eine andere Häufigkeitsanalyse als die Statoren. Der Großteil des Schlüsselraums wurde durchsucht, die wenigen Einträge aus der Bestenliste von Phase 1 als Suchgrundlage verbleiben. Die Analyse der Ringe wurde in einen sequentiellen Ablauf unterteilt und durchsucht so eine Ringstellung pro Rotor nacheinander. Dies ergibt einen Suchraum von $26+26+26=78$ Möglichkeiten. Diese Suche ist deutlich performanter, als die Suche aller Ringstellungen zur gleichen Zeit mit $26 \cdot 26 \cdot 26 = 17.576$ Möglichkeiten.

Die Struktur im Klassendiagramm (Abbildung 5.11) ist ähnlich der Suche nach Rotoren/Statoren. Zuerst wird die Klasse mit den benötigten Werten bestückt und danach wird die Methode $run\ldots()$ ausgeführt.

Rotoren	*II-I-III*			*III-I-II*	
Statoren	*IV-V*	*V-IV*	**Metriken**	*IV-V*	*V-IV*
Grundstellung *HARLA*	0,002362	0,028791	$\lim_{x\to 0} IC(x)$	0,026786	0,028051
	-934,16	-1.234,05	$\lim_{x\to 0} Sinkov(x)$	-1245,95	-1.274,14
	2.134,92	1.284,33	$\lim_{x\to \infty} Chi\ Squared(x)$	1.216,63	1.121,48
	3.507,53	2.586,72	$\lim_{x\to \infty} Bigram(x)$	2.502,43	2.441,02
	2.793,16	944,69	$\lim_{x\to \infty} Trigram(x)$	777,01	710,49
	2.206,17	285,98	$\lim_{x\to \infty} Quadgram(x)$	490,63	495,46
	1.712,89	424,12	$\lim_{x\to \infty} Quintgram(x)$	394,49	360,36
	-330,53	-377,46	$\lim_{x\to 0} CT\ Trigram(x)$	-410,92	-385,49
	-389,87	-415,54	$\lim_{x\to 0} CT\ Quadgram(x)$	-477,80	-473,67
HARAA	0,028207	0,026980		0,027545	0,027740
	-1.227,61	-1.219,63		-1.302,86	-1.209,95
	1.259,16	1.222,94		1.142,70	1.242,23
	2.548,07	2.567,16		2.383,55	2.627,59
	887,91	927,48		685,89	978,61
	278,76	419,08		466,02	337,84
	349,67	362,10		393,23	273,82
	-462,38	-341,97		-357,01	-349,35
	-711,34	-465,60		-465,60	-454,60
AAAAA	0,028285	0,029122		0,028149	0,026494
	-1.244,49	-1.205,90		-1.228,87	-1.206,58
	1.285,60	1.277,79		1.288,38	1.277,11
	2.515,99	2.594,80		2.536,61	2.616,48
	914,60	964,50		871,55	997,37
	262,27	255,87		413,54	297,24
	397,25	434,57		324,62	313,28
	-385,49	-343,64		-374,40	-341,97
	-473,67	-389,89		-467,82	-465,60

Tabelle 5.5: Messungen über un-/korrekte Schlüssel

Rotoren	*II-I-III*	*III-I-II*		*II-I-III*	*III-I-II*		*II-I-III*	*III-I-II*
Grundstellung *HAR*	0,029005	0,028674	*AAR*	0,027506	0,027058	*AAA*	0,028674	0,028480
	-1.244,06	-1.222,27		-1.231,15	-1.205,33		-1.245,27	-1.279,91
	1.263,48	1.307,10		1.330,47	1.263,13		1.187,56	1.202,85
	2.507,58	2.577,92		2.621,47	2.583,16		2.520,86	2.437,83
	858,45	941,85		1.091,40	883,33		847,33	791,11
	470,33	310,13		207,84	433,65		444,19	368,90
	430,90	414,37		444,41	428,06		413,64	357,12
	-355,30	-410,92		-385,49	-385,49		-347,53	-341,06
	-457,82	-477,80		-473,67	-473,67		-415,23	-456,67

Tabelle 5.6: Messungen über un-/korrekte Schlüssel ohne Statoren

Abbildung 5.11: Klassendiagramm der Klasse – DecodeRingsSequently (Anaylsekomponente)

Die Klasse erhält die Bestenliste aus Phase 1 und initialisiert eine Typex-Maschine mit den gegebenen Einstellungen, wobei die Ringstellungen initial auf A stehen. Bei Aufruf der Klasse wird zuerst der schnelle Rotor analysiert und dessen Ringstellung sukzessive erhöht. Das Resultat aus der Entschlüsselung wird mit dem Schlüssel bewertet. Danach wird das gleiche Verfahren auf die verbleibenden Rotoren angewendet, wobei die Metrik sich pro Etappe ändert, gezeigt in Tabelle 5.7.

Rotorring		Metrik
Schneller	1	Tetragramm
Mittlerer	2	Sinkov
Langsamer	3	IC

Tabelle 5.7: Metrik bei Ringanalyse

Das Resultat der Analyse kann von der jeweiligen Grund- und Ringstellung abweichen, da nur der Abstand zwischen den beiden Schlüsselteilen relevant ist. Ist der Schlüssel mit Ringstellung UVW und Grundstellung XYZ korrekt, wäre ein möglicher Schlüssel mit Ringstellung ABC und Grundstellung DEF trotz dessen passend für den Geheimtext.

5.2.1.4 Klasse – FrequencyAnalysis

Die Häufigkeitsanalyse ist eine Kernfunktionalität, die essentiell zur erfolgreichen Ergebnisfindung ist. Die statistischen Metriken sind in Kapitel 4.3 theoretisch beschrieben. Sie wurden in der Klasse *FrequencyAnalysis* technisch implementiert. Die notwendigen Methoden und Variablen sind in Abbildung 5.12 gezeigt.

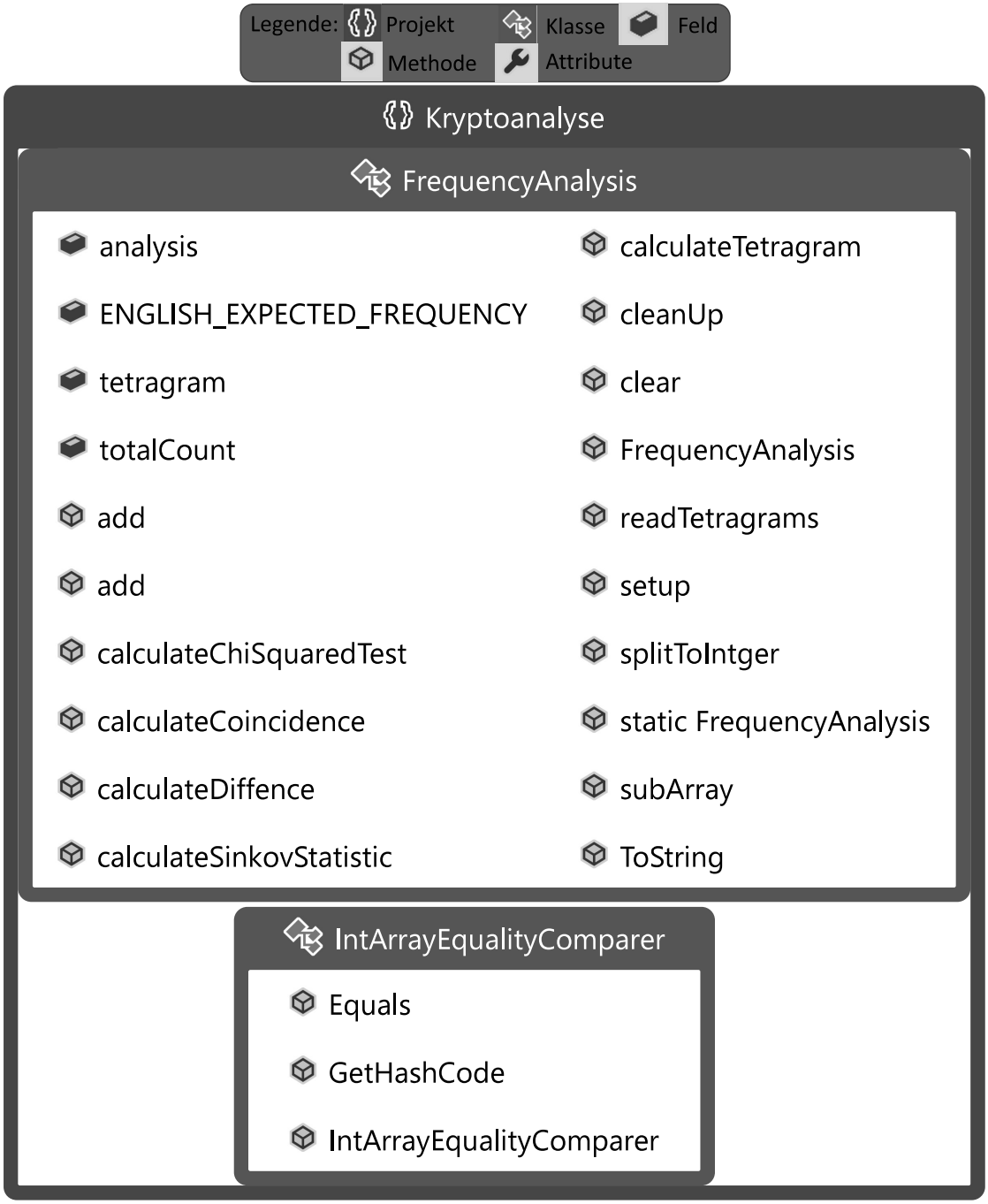


Abbildung 5.12: Klassendiagramm der Klasse – FrequencyAnalysis (Anaylsekomponente)

Die Häufigkeiten der Buchstaben aus der englischen Sprache sind als Konstanten aufgenommen. In Abbildung 2.6 sind diese visualisiert. Grundsätzlich werden die Zeichen aus dem Klartextkandidaten einzeln in die Bewertung hinzugefügt und mit dem jeweiligen Gewicht versehen. Nach Einlesen der Zeichenfolge wird die Berechnung der Metrik vorgenommen, die an die aufrufende Klasse als einzelner Zahlenwert zurückgegeben wird. Beim ersten Anlegen einer Klasseninstanz werden

allgemein geltende Daten eingelesen, wie etwa die Tetragramm-Bibliothek.

Die Methode *add*() dient zum Zählen der Buchstaben $A - Z$ und dem Bestimmen der Gesamtlänge des Textkandidaten. Die Daten können mit dem Aufruf von *clear*() verworfen werden.

Im Folgenden sind die Methoden *calculateCoincidence*(), *calculateSinkovStatistic*(), *calculateChiSquaredTest*() und *calculateTetragram*() analog zur theoretischen Beschreibung in Kapitel 4.3 implementiert. Die Ergebnisse werden so manipuliert, dass Werte nahe 0 die besten Resultate liefern.

Das Tetragramm-Konzept benötigt ferner die Initialmethode *readTetragrams*() für die Bereitstellung der Bibliothek, basierend auf eingelesenen Büchern. Das Einlesen der Bibliothek erfolgt einmal pro Programmaufruf und wird sonst dauerhaft im Arbeitsspeicher bereitgehalten. Eine weitere Hilfsmethode *splitToIntger*() ist für die Konvertierung von Text zu Zahlen zuständig.

Das ganze Projekt basiert auf Zahlen ($A = 0, B = 1, \dots$), so werden die Laufzeiten dramatisch verkürzt. Aus diesem Grund befinden sich die Tetragramm-Fragmente als Zahlenrepräsentant in der Heuristik. Die Tetragramm-Einträge bestehen aus ganzen Zahlen und setzen sich aus vier Werten zusammen. Der Eintrag $19, 8, 14, 13$ mit $-13.168.375$ ist das häufigste Tetragramm in der englischen Sammlung und spiegelt das Fragment T, I, O, N wieder. Sofern die Länge des Fragments nicht stimmt, wird der Einlesevorgang abgebrochen und der Nutzer auf eine falsche Eingabedatei hingewiesen. Die Hilfsklasse *IntArrayEqualityComparer* detektiert Duplikate bei Tetragrammen, denn eine Tetragramm-Konstellation mit entsprechenden Gewicht kann nur einmal auftreten.

5.2.1.5 Klasse – Program (Kryptoanalyse)

Diese Klasse ist der generelle Einstiegspunkt für die Ausführung einer Kryptoanalyse. Initial wird ein Objekt von der Kryptoanalyse angelegt und die gewünschte Zeichenkette mit Geheimtext, den es zu entschlüsseln gilt, wird definiert. Die Verarbeitung mit dem Geheimtext und dem Wahlparameter für die Rotor- oder Statorsuche wird über die Methode *run*() von der Kryptoanalyse-Klasse aufgerufen. Der erwartete Rückgabewert ist das beste Ergebnis. Obwohl in der Verarbeitung eine Reihe von optimalen Ergebnissen zur Verfügung steht, wird nur das idealste Ergebnis als einzige Lösung ausgegeben.

Die Variablen der Ausführungsklasse sind in Tabelle 5.8 ersichtlich. Die Variablen beinhalten die Argumente aus dem Programmaufruf des Benutzers. Das Programm wird beim Aufruf mit Argumenten versehen. Es umfasst Schlüsselteile und den Geheimtext, den es gilt zu brechen. Diese Werte wurden über die Methode *readSettings*() in das Programm eingelesen, indem auf das Komma-Trennzeichen geachtet wurde. Die Methode *convertSettings*() prüft die Zulässigkeit des Wertes. Entweder er entspricht den Eingabebedingungen für eine reibungslose Ausführung

oder er führt zu einem entsprechenden Fehlerhinweis durch eine nicht valide Eingabe. Diese Bedingungen sind ebenfalls in der Tabelle ersichtlich.

Parameter	Argument	Erklärung	Beispiel
-d (decodemode)	r / s	-d entscheidet, ob Rotoren (r) oder Statoren (s) gesucht werden	-d r oder -d s
-i (rings)	2/3 Buchstaben	-i definiert die Ringstellung. Sie besteht aus zwei Buchstaben (A-Z) beim Rotorsuchlauf (-d r) und aus drei Buchstaben beim Statorsuchlauf. (-d s)	AA, TY oder AAA, TYP
-b (rotorbegin)	2/3 Buchstaben	-b gibt die Grundstellung an. Beim Rotorsuchlauf sind es zwei Buchstaben, beim Statorsuchlauf sind es drei Buchstaben.	AA, TY oder AAA, TYP
-r (rotororder)	3 Zahlen	-r wird nur beim Statorsuchlauf angegeben. Es besteht aus drei Zahlen über den verfügbaren Rotoren von -8 bis 8 (Walzensatz).	-1,2,-3
-s (statororder)	2 Zahlen	-s ist beim Rotorsuchlauf notwendig. Es sind zwei Zahlen aus dem Walzensatz von -8 bis 8.	4,-5
-m (message)	Buchstaben	-m übergibt die Geheimtextnachricht. Dies kann eine beliebe Buchstabenfolge (A-Z) sein.	HFIF...
-p (plaintext)	Buchstaben	-p (optional) speichert den korrespondierenden Klartext. Sofern verfügbar wird die Distanz aus dem Klartext und dem gefundenen Text berechnet.	THEG...
-n (maxrotors)	1 Zahl	-n definiert die Größe des Walzensatzes. (minimal: 5 Rotore)	5

Tabelle 5.8: Programmkonstanten und Ausführungsparameter für das Projekt Kryptoanalyse

Sind alle Argumente für eine Rotorsuche mit einem Wert versehen sieht ein Programmaufruf wie folgt aus:

```
Kryptoanalyse.exe -d r -i AA -b AA -s 1,2 -n 5 -m HFIFJDXHTBKUN
                  -p THEGRASSWOULD
```

Die Datei gibt eine Meldung für gültige Werte aus. Sofern einige Werte ungültig sind, wird der Benutzer darauf hingewiesen. Die Konsolen-Ausführung 5.4 zeigt die

Kryptoanalyse beim Ausführen des oben genannten Befehls. Allerdings wurden die Parameter für den Klartext $-p$ und Geheimtext $-m$ durch längere Werte ersetzt.

```
C:\Release>Kryptoanalyse.exe -d r -i AA -b AA -s 1,2 -n 5 -m
   YNVCVLHEEADOMKMMFKEPCEQEYLZDNPDXLHSNCRYPEGGLGZTQHONYVVBIBQWEF

==========Typex-Machine Kryptoanalyse==========
Input data is valid
Processing ...
RESULT - SETTINGS:
Rings: AAAAA Initial state: PQZAA RotorOrder: 5,3,-4
   StatorOrder: 1,2 Reflector: B
FREQUENCY SCORE (smaller is better): 0,00124061809467493
MESSAGE:
THENAVYANDAIRMINISTERTOPRESIDENTOFTHEBASQUEGOVERNMENTWEARECONSID
```

Textauszug 5.4: Konsolen-Ausführung des Projektes Kryptoanalyse

5.2.2 Projekt – Evaluation

Zur Auswertung der Effektivität wird eine Komponente benötigt, die viele unterschiedliche Texte verschlüsselt, diese in variierter Länge generiert und verschlüsselt und letztendlich der Kryptoanalyse-Komponente zur Decodierung übergibt. Das Projekt Evaluation bedient diese Anforderung. Es zeigt detailliert auf, wann das Ergebnis zufriedenstellend oder nicht zielführend war. Das Projekt hat eine einseitige Abhängigkeit zum Projekt „Kryptoanalyse“.

Einen generellen Überblick auf die Programmstruktur verschafft Abbildung 5.13. Das Projekt *Evaluation* gliedert sich in die wichtige Klasse *Evaluation* und in die Ausführungsklasse *Programm* mit einer inneren Klasse *EvaluationConfiguration*.

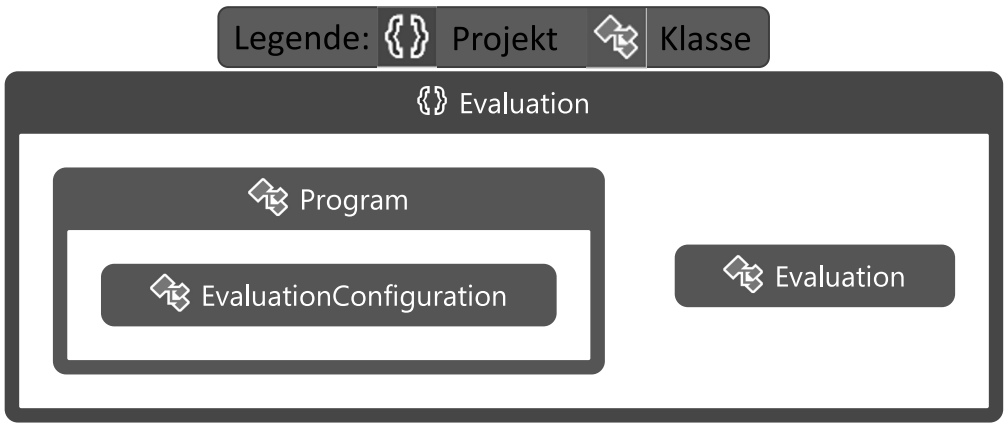


Abbildung 5.13: Klassendiagramm der Klasse – FrequencyAnalysis (Anaylsekomponente)

5.2.2.1 Klasse – Evaluation

Zum größten Teil werden in der Methode *run*() die technischen Gegenstände, die Bestandteile des Schlüssels sind, initialisiert. Ringe, Grundstellung, Rotor-/Statorauswahl und Umkehrwalze werden zufällig gezogen und aus der Datenstruktur abgerufen. Die Ringe und Grundstellung umfassen lediglich das Generieren von je fünf Zahlen zwischen 0 und 26. Die zufällige Rotor-/Statorwahl erfolgt über das Array *initRotors*, welches bereits alle initialisierten Rotoren/Statoren mit den dazugehörigen Verdrahtungsplänen beinhaltet. Das Abrufen wird über die vielfachen Verarbeitungsintervalle vereinfacht. Die Konfiguration wird bei der Maschine hinterlegt, welches sich zum physikalischen Einsetzen der Komponenten (Rotor/Stator) in die Typex assoziieren lässt. Die technische Positionierung der Variablen und Methoden ist in Abbildung 5.14 ersichtlich.

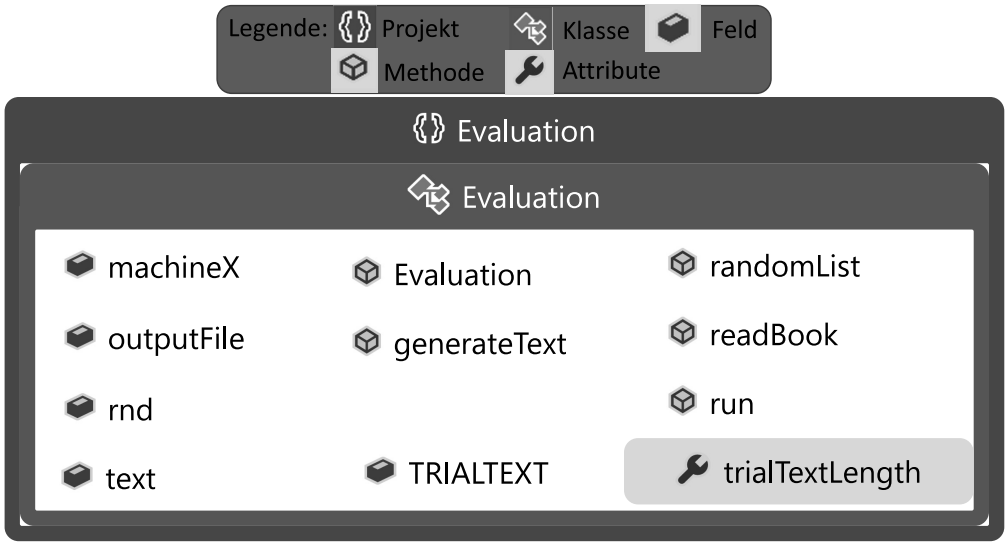


Abbildung 5.14: Klassendiagramm des Projektes – Evaluation

Die Methode *randomList*() generiert eine Sammlung von zufälligen Zahlen in einem gegebenen Intervall. Bei Aufruf werden das maximale Intervallende und die Anzahl der gewünschten Zufallszahlen definiert. Ein Array wird konform zu den Eingangsparametern generiert. Das Erzeugen der Rotor-/Statorauswahl wird gesondert behandelt, da keine Duplikate und kein Auftreten desselben Partners (umgedrehter Rotor) in der Zufallsliste vorkommt.

Zum Generieren von Textkandidaten werden Zeichenfolgen von verschiedener Länge aus einem Buch entnommen. Das Buch sollte eine hohe Textlänge und eine hohe Alternierung des Vokabulars einer spezifischen Sprache widerspiegeln. Die Maschine Typex hat favorisiert Texte aus der englischen Sprache verschlüsselt. Es bietet sich daher an, das Buch „Alice in Wonderland / Alice im Wunderland“ der englischen Fassung als Grundlage für die Textgenerierung zu wählen. Das Buch umfasst 29.708 Wörter und 132.107 Zeichen, wobei die 27.973 Leerzeichen, konform zur Eingabebedingung, entfernt sind. Des Weiteren treten nur Zeichen von

$A - Z$ und Großbuchstaben im Buchinhalt auf. Lustige Wortspiele, fantasievolle Wortneuschöpfungen und bizarre Gedankenkonstruktionen von Lewis Carroll [57] führen zu einer guten Wortverteilung und einem allgemeinen Themenfeld bei der Generierung von Texten. Der Textkandidat wird willkürlich über die definierte Länge aus dem Buch gewählt und mit dem zufälligen Schlüssel chiffriert.

Der gewonnene Geheimtext wird ohne jegliche Schlüsselinformation der Kryptoanalyse-Komponente über Rotor *oder* Stator zur Analyse übergeben. Das Resultat mit entschlüsseltem Text wird mit Hilfe der Levenshtein-Distanz aus Kapitel 6.1.1.2 auf Ähnlichkeit mit dem generierten Text geprüft. Der korrekte und der jeweils gefundene Schlüssel werden in eine Textausgabe umgelenkt. Darüber hinaus werden die Laufzeit, der Klartext und das Levenshtein-Maß als Datensatz ausgegeben. Die Spalten aus Tabelle 5.9 sind in einer *.csv-Datei nachvollziehbar. Die letzten beiden Spalten beinhalten Kennziffern über die Genauigkeit des Ergebnisses. Bei Dis_1 ist ein Wert nahe der 0 ein Anzeichen für ein gutes oder korrektes Ergebnis. Bei Dis_2 zeigt die obere Schranke 1 ein optimales oder korrektes Ergebnis an.

Spalteninformation	Ursprung
Länge	
Laufzeit	
Ringe	zufällig generiert
Grundstellung	
Rotorreihenfolge	
Statorreihenfolge	
Reflektor (Umkehrwalze)	
Metrik (IC)	
Text	
Levenstein-Distanz	
Ringe	gefunden / Analyseergebnis
Grundstellung	
Rotorreihenfolge	
Statorreihenfolge	
Reflektor (Umkehrwalze)	
Metrik (IC)	
Entschlüsselter Text	
$Dis_1 = \frac{LevensteinDistanz}{Textlänge}$	
$Dis_2 = \frac{Metrik(IC)\ Zufallstext}{Metrik(IC)\ Klartext}$	

Tabelle 5.9: Spalteninformation der Evaluation-Ausgabedatei

Die Ausgabedatei ist die Grundlage für die Evaluation der Anwendung und bietet gleichzeitig eine Basis zur Bewertung. Das Dateiformat lässt sich einfach in ein

Diagramm oder gar in eine Eingabedatei für anschließende Verarbeitungen überführen.

5.2.2.2 Klasse – Program (Evaluation)

Die Ausführungsdatei des Projektes Evaluation lässt sich über zwei Parameter starten. Sie übernimmt das hierarchische Durchlaufen der Komponenten Evaluation, Kryptoanalyse und Typex-Simulator. Die Parameter sind die Initiallänge des Textkandidaten und ein Schrittwert für das sukzessive Erhöhen der Textlänge. Das Programm wird viele Male ($100 < x$) mit der gleichen Textlänge ausgeführt. Es wird gemessen, ob das Ergebnis gefunden wurde oder nicht. Über die Messergebnisse mit gleicher Textlänge wird der Mittelwert errechnet, um den Grad der korrekten Ergebnisse bei gegebener Textlänge zu bestimmen.

Tabelle 5.10 zeigt die benötigten Parameter für eine erfolgreiche Ausführung des Projektes Evaluation. Die Eingabewerte werden auf ihre Gültigkeit validiert. Jedes Argument benötigt einen verknüpften Wert, da sonst eine Fehlerinformation ausgegeben wird. Die Variablen nehmen eine zentrale Referenz im Programmcode ein.

Parameter	Argument	Erklärung	Beispiel
-f (outputfile)	Dateipfad	-f legt den Dateipfad für die Datenausgabe fest. Die angezeigten Werte sind in dieser Datei gespeichert.	E:\eva.csv
-l (length)	Zahlen	-l bestimmt die Länge des Textkandidaten.	200, 300, 500
-s (steps)	Zahlen	-s definiert die zukünftige Länge des Textkandidaten. Um diese Länge wird -l erhöht.	20, 150, 400
-i (iterations)	Zahlen	-i gibt die Wiederholung für -l an bis die Variable um -s erhöht wird.	50,100
-n (maxrotors)	1 Zahl	-n definiert die Größe des Walzensatzes (minimal: 5).	5, 7

Tabelle 5.10: Programmkonstanten und Ausführungsparameter für das Projekt Evaluation

Ein korrekter Programmaufruf der Evaluation kann mit folgendem Befehl erfolgen:

```
Evaluation.exe -f E:\eva.csv -l 200 -s 100 -i 50 -n 5
```

Die Datei überprüft die Eingabe auf Gültigkeit und informiert den Benutzer über Fortschritte. Das Programm generiert, abhängig zur Eingabe, Textkandidaten, die anschließend mit ihrem Ergebnis präsentiert werden. Solch ein Ablauf ist im Konsolenausschnitt 5.5 gezeigt. Diese Anwendung terminiert nicht und muss daher manuell gestoppt werden. In der Ausgabedatei sind die Ergebnisse für eine nachträgliche Untersuchung bereit gehalten.

```
C:\Release>Evaluation.exe -f E:\eva.csv -l 200 -s 100 -i 50 -n
   5
==========Typex-Machine Evaluation==========
Input data ist valid
Created file: E:\eva.csv

Original Length: 300
SETTINGS:
Rings: EQBSW Initial state: DLYLC RotorOrder: 1,-2,-3
   StatorOrder: -4,5 Reflector: B
FREQUENCY SCORE (smaller is better): 0,00065763656633222
MESSAGE:
EREVENINTRODUCEDTOALOBSTERALICEBEGANTOSAYIONCETASTEDBUTCHECKEDHE


levensteinDistance: 0  Execution time: 124
SETTINGS:
Rings: AKBSW Initial state: ZFYLC RotorOrder: 1,-2,-3
   StatorOrder: -4,5 Reflector: B
FREQUENCY SCORE (smaller is better): 0,00065763656633222
MESSAGE:
EREVENINTRODUCEDTOALOBSTERALICEBEGANTOSAYIONCETASTEDBUTCHECKEDHE


==============================
Original Length: 300
SETTINGS:
Rings: OKDXE Initial state: CXPMO RotorOrder: 3,1,2 StatorOrder
   : 4,5 Reflector: B
FREQUENCY SCORE (smaller is better): 0,00103690078037905
MESSAGE:
EICANGOBACKBYRAILWAYSHESAIDTOHERSELFALICEHADBEENTOTHESEASIDEONCE


Processing...
```

Textauszug 5.5: Konsolen-Ausführung des Projekts Evaluation

6 Evaluation

Dieses Kapitel beschäftigt sich mit der Evaluation der implementierten Algorithmen. Zuerst wird die Funktion der Algorithmen diskutiert und im Anschluss die Qualität des Quellcodes zur Erreichung des Ergebnisses bewertet. Dazu wird geprüft, ob die für die Typex chiffrierten Nachrichten mit Hilfe der implementierten Algorithmen gelöst werden können.

6.1 Bewertung der Analysekomponente

Der Angriff im Kapitel 4.2.2 ist eine Brute-Force-Methode. Unter gewissen Annahmen zur Schlüsselraumreduzierung ist durch die heutige Rechenleistung eine Brute-Force-Methode auf die Typex möglich. Einen reinen Brute-Force Angriff auf den theoretischen Schlüsselraum (Kapitel 3.3.1) ist nicht möglich.

Der Rechenaufwand erhöht sich signifikant, zumindest vierfach, sobald weitere Rotoren in die Auswahl aufgenommen werden. Das Hinzufügen eines Rotors bei der Anzahl r vervierfacht die Auswahl von Rotoren (Kombinationen). Bei festen Statoren ($fixS$) und auswählbaren Rotoren wird die Anzahl r um zwei Statoren reduziert, da diese bereits aus der Auswahl entfernt wurden. Ein Rotor kann gedreht eingesetzt werden (2 Verdrahtungen = 1 Rotor). Daher berechnet sich die verbleibende Anzahl durch Formel 6.1.

$$count_{fixS}(r) = r - 2 \tag{6.1}$$

Die Möglichkeit eine Kombination aus den verbleibenden Rotoren zu wählen, wird in Formel 6.2 berechnet. Die Berechnung der Ergebnisse ist in Abbildung 6.1 ersichtlich.

$$comb_{fixS}(r) = \prod_{i=0}^{2} (count_{fixS}(r) - i) \cdot 2 \tag{6.2}$$

Rotoranzahl r	Kombinationen $comb_{fixS}(r)$
5	48
6	192
7	480
10	2.688
100	7.300.608
1.000	7.928.207.808

Abbildung 6.1: Hinzufügen weiterer Rotoren

6.1.1 Metriken

Ziel der Evaluation ist es, die Effizienz der implementierten Angriffe zu bewerten. Dies kann nach dem jeweiligen Angriff unterschiedlich gemessen werden. Merkmale für die Effektivität des Ciphertext-Only Angriffs sind sowohl die Dauer eines Brute-Force Angriffs auf den reduzierten Schlüsselraum bevor der Algorithmus terminiert als auch die Wahrscheinlichkeit, mit Hilfe des Angriffs den richtigen Schlüssel zu finden. Anhand der Dauer lässt sich abschätzen, wie lange der Algorithmus läuft, bis er terminiert. Dazu soll gemessen werden, wie viele Modelle der Algorithmus für gegebene Texte findet. Die Erfolgswahrscheinlichkeit des Algorithmus lässt sich angeben, indem man für verschiedene Textlängen die Wahrscheinlichkeit misst, den korrekten Schlüssel zu finden. Zu erwarten ist, dass die Wahrscheinlichkeit den richtigen Schlüssel zu finden, mit zunehmender Textlänge bis zu einem Schwellenwert steigt. Dies ist dadurch begründet, dass die Kostenfunktion mit steigender Textlänge mehr konsistente Daten verarbeitet, und somit weniger von Abnormitäten beeinflusst wird.

6.1.1.1 Ausführungsdauer

Diese Metrik wird zur Messung der Effizienz beim Entschlüsseln genutzt. Sie wird u. a. über die Dauer der Ausführung definiert. Die Messungen lassen das Ergebnis in Abhängigkeit zu den Eingabeparametern, wie die Schlüsselkomplexität, Satzlänge und Textkorpus kategorisieren. Jede Textlänge wird in vielen Kombinationen immer wieder ausgeführt. Die Verarbeitungen werden aufgezeichnet, um einen Mittelwert bei der Ausführungsdauer mit einer festen Textlänge zu ermitteln.

Die Angriffsdauer ist die Zeit, die benötigt wird, einen Angriff auf die Typex mit einem gegebenen Prozessor auszuführen. Die vorgeschlagene Metrik zeigt, wie sicher und resistent ein System sein. Ein Angriff wird berechnet und in einem gerichteten Graphen modelliert. Ein Angriffs-Graph repräsentiert einen Angriffsfall, da jeder Angriff einen eigenen Graphen erzeugt. Ein Graph wird in zwei Arten von Verzweigungen dargestellt nach Abbildung 6.2: *UND* bei vertikaler und *ODER* bei

horizontaler Verbindung. Die dicke Linie indiziert den Pfad zur korrekten Grundeinstellung. Im Angriffsgraph befindet sich ein Angriffsursprung (die Geheimtexteingabe), jeder weitere Fortschritt wird als Schritt definiert.

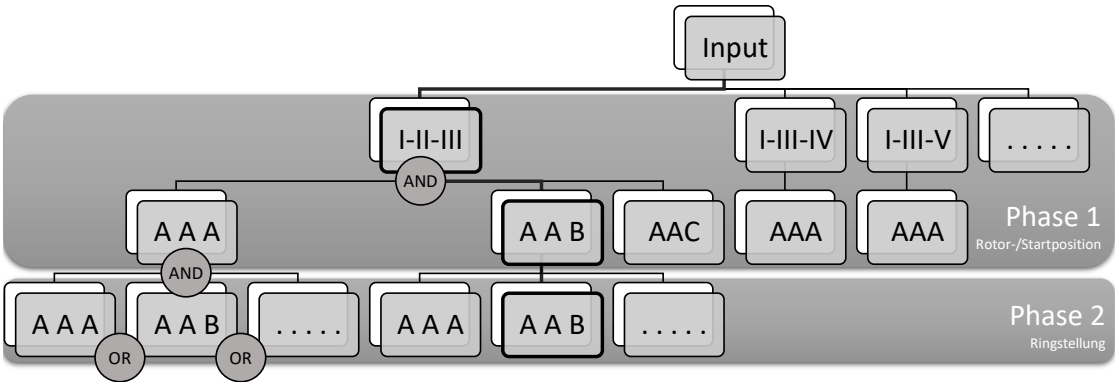


Abbildung 6.2: Graph von einem Angriff

Das Design des Angriffs-Graphen resultiert aus der gewählten Rotormaschine. Daher können unterschiedliche Konfigurationen ähnliche Graphen generieren. Wenn zum Beispiel die Rotorauswahl eingegrenzt wird, dann entstehen Graphen mit derselben Grundstellung. Die Dauer eines Angriffs in einem System kann die Systemsicherheit im Ganzen beeinflussen, da der Sicherheitsstatus eines Systems während des Angriffs stark schwanken kann. Je länger die Angriffsdauer auf das System, desto weniger effizient ist der Angriffsalgorithmus mit seiner Methode.

6.1.1.2 Levenshtein-Distanz

Die Levenshtein-Distanz zwischen zwei Zeichenketten ist die minimale Anzahl von Einfüge-, Lösch- und Ersetzungsoperationen, um die erste in die zweite Zeichenkette umzuwandeln.

Zur Berechnung der Levenshtein-Distanz wird eine Matrix aufgebaut. Mit Hilfe des kürzesten Wegs durch die Matrix ergeben sich dann die Operationen, die nötig sind, um aus Wort x, Wort y zu erzeugen. Im Beispiel weiter unten wird der Weg durch die Matrix genauer beschrieben.

Der Algorithmus ist nach [58] durch folgende Matrizen spezifiziert, wobei x und y die beiden Eingabe-Zeichenketten bezeichnen:

$$LEV[i,j] = lev(x[1\ldots i], y[1\ldots j]) \text{ mit } 0 \leq i \leq m, 0 \leq j \leq n, \quad m = |x|, n = |y| \tag{6.3}$$

Die Werte der Matrix $LEV[i,j]$ können mit folgenden Formeln berechnet werden:

$$LEV[0,j] = j \text{ für } 0 \leq j \leq n, \quad LEV[0,i] = i \text{ für } 0 \leq i \leq m \tag{6.4}$$

$$LEV[i,j] = min \begin{cases} LEV[i-1,j]+1 \\ LEV[i,j-1]+1 \\ LEV[i-1,j-1]+\mu(x[i],y[j]) \end{cases} \quad \mu(a,b) \begin{cases} 0 \text{ falls } a=b \\ 1 \text{ sonst} \end{cases} \tag{6.5}$$

Die berechnete Matrix für $x = typex$ und $y = typical$ zeigt eine Distanz von 4 ($lev(x,y) = 4$). Die Levenshtein-Distanz kann man in der Matrix an Stelle $LEV[i,j]$, (das ist die Zahl in der rechten unteren Ecke), ablesen. In diesem Fall ist die Distanz 4, d. h. es sind maximal vier Einfüge-, Lösch- oder Ersetzungsoperationen nötig, um aus x y zu erhalten. Des Weiteren ist in Tabelle 6.1 der kürzeste Pfad durch die Matrix eingezeichnet. Der Pfad beginnt immer im Punkt $LEV[0,0]$ und endet in $LEV[i,j]$. Das Zeichen ϵ repräsentiert das leere Wort. Diagonale Kanten in der Matrix stellen eine Ersetzungsoperation oder das Auslassen einer Operation dar. Horizontale Kanten zeigen Einfügeoperationen, vertikale Kanten hingegen Löschoperationen.

	ϵ	t	y	p	i	c	a	l
ϵ	0	1	2	3	4	5	6	7
t	1	0	1	2	3	4	5	6
y	2	1	0	1	2	3	4	5
p	3	2	1	0	1	2	3	4
e	4	4	3	2	1	2	3	4
x	5	4	3	2	2	2	3	**4**

Tabelle 6.1: Pfad durch eine Levenshtein-Matrix

Eine Transformation von „typex“ in „typical“ sieht folgendermaßen aus:

- S deutet an, dass eine Substitution erforderlich ist,
- G, dass der Buchstabe gleich bleibt und
- E, dass ein Buchstabe eingefügt werden muss.

Eine Löschoperation L ist in diesem Beispiel nicht erforderlich.

$$typex \rightarrow t^G ypex \rightarrow ty^G pex \rightarrow typ^G ex \rightarrow typi^S x \rightarrow typic^S \ldots typical^E \tag{6.6}$$

Die naive Implementierung des Levenshtein-Distanz-Algorithmus hat eine Laufzeitkomplexität von $O(mn)$ und eine Speicherkomplexität von $O(mn)$. m und n stellen die Wortlängen dar. Dan Hirschberg [59] überarbeitete den Algorithmus, sodass eine lineare Speicherkomplexität von $O(\min[m,n])$ möglich ist.

6.1.1.3 Textkorpus

Zum Evaluieren des Chiffrieralgorithmus werden Textkandiaten aus syntaktisch und semantisch korrekten Sätzen einer natürlichen Sprache benötigt. Diese Daten werden mit einem Textkorpus bereitgestellt. Unter einem Korpus wird eine Sammlung von geschriebenen Texten verstanden, die als Grundlage für eine Untersuchung mit Hilfe des Computers verwendet werden kann [60]. Die Elemente eines Korpus gehören in ihrer Gesamtheit bestimmten Kategorien an, die im Korpusdesign von vornherein festgelegt sind.

In diesem Kapitel wird unter einem Textkorpus ein maschinenlesbarer Text verstanden, der von einer Typex chiffriert werden kann. Das Resultat kann im Analyseprogramm auf die einzelnen Bestandteile hin untersucht werden. Sinnvollerweise ist es ein Text, der sehr groß ist und nicht von einer Person in kurzer Zeit effektiver bearbeitet werden kann.

Ein Textkorpus bietet die Möglichkeit, das Muster einer Sprache auf Basis tatsächlich geäußerte Sprachsequenzen zu untersuchen. Dem Korpus werden Stichproben entnommen und anhand dessen allgemeine Aussagen getroffen. Als Grundlage wird die englische Sprache gewählt. Ein englischer Satz besteht aus einer Folge von Buchstaben und die Wörter sind durch Leerzeichen getrennt. Das Alphabet basiert auf dem Unicode-Zeichensatz, sodass alle Texte nur Buchstaben von „A“ - „Z“ beinhalten. Das Korpus wird aus dem englischen Buch „Alice im Wunderland“ [57] des Gutenberg Projekt generiert.

Es wurden folgende Anforderungen an den Korpus gestellt:

- Korrekte Wortposition in Texten
- Auftreten eines Wortes mit bestimmten anderen Wörtern
- Worthäufigkeiten und Wortverteilungen in Texten (nicht zufällig)
- Satz- und Wortlängen analog zu englischen Texten

6.1.2 Testumgebungen

Unter einer Testumgebung ist die Gesamtheit aller Hardware- und Softwarekomponenten zu verstehen, die notwendig sind, um Testfälle durchführen zu können. Dazu gehören insbesondere die entsprechende Hardware und Betriebssysteme sowie eine geeignete Test-Software. Für die Evaluation wird eine Testumgebung eingerichtet, um verschiedene Ergebnisse über Kryptoanalyse zu sammeln.

Die Testumgebung muss so beschaffen sein, dass sie eine funktional gleichwertige Installation von Hard- und Software gemäß den Zielanforderungen erlaubt. Zum Testen von Programmverlauf und Eingabeparametern der Kryptoanalyse-Anwendung genügt meist ein technisch sparsam ausgestattetes System. Dies ist in Tabelle 6.2 – Spalte 1 als „Desktop“ zu erkennen. Nichtsdestotrotz wurde eine

technisch moderne Testumgebung für einen Vergleich herangezogen, aufgelistet in Tabelle 6.2 – Spalte 2 als „Server“.

	Desktop	Server
Betriebssystem	MS Windows 7 Pro	MS Windows 8.1 Pro
Version	6.1.7601 Build 7601	6.3.9600 Build 9600
Systemmodell	7661VKX	i440FX + PIIX, 1996
Systemtyp	x64-basierter PC	x64-basierter PC
Prozessor	Intel Core 2 Duo CPU T7100	Intel Xeon CPU E5-4620
Leistung	1,80 GHz	2,20 GHz
Kerne	2	16
Arbeitsspeicher	4 GB	64 GB

Tabelle 6.2: Testumgebungen

Es ist offensichtlich, dass Wechselwirkungen zwischen Hardwareumgebung und Anwendung erkennbar sind.

6.1.3 Messungen

Zur Durchführung der Messungen wurde ein Evaluationsprojekt mit den implementierten Algorithmen erstellt. Innerhalb dieses Projekts wurde eine Testumgebung aufgebaut. Hierzu wurde das Korpus in Abschnitt 6.1.1.3 in englischer Sprache genutzt, um zufällige Textkandidaten zu erstellen. Mit diesen Textkandidaten wurde anschließend der Algorithmus getestet.

Zuerst wurde der Ciphertext-Only Algorithmus auf seine Laufzeit getestet. Der Test des Algorithmus fand auf der Desktop- und Serverumgebung laut Abschnitt 6.1.2 statt. Die Messungen wurden mindestens 60 mal und maximal 200 mal für eine Textlänge wiederholt. Die Textlängen wurden in den Abständen von 20 bis 100 Zeichen erhöht. Anschließend wurde deren durchschnittliche Laufzeit berechnet.

Zum Testen der Qualität des Ciphertext-Only Angriffs wurde gemessen, wie wahrscheinlich es ist, dass die Analyse den richtigen Schlüssel findet. Dies wurde für Textlängen zwischen 0 und 1.500 Zeichen getestet. Ein Textkandidat wurde jeweils mit einem zufällig generierten Schlüssel verschlüsselt. Anschließend wurde der Ciphertext-Only Angriff gestartet, wobei die korrekten Statoren bereits übergeben wurden. Zusätzlich wurde dem Algorithmus der dechiffrierte Text übergeben, sodass nach der Terminierung, die Distanz zwischen Klartext und Text, der mit dem gefundenen Schlüssel dechiffriert wurde, verglichen werden konnte. Zum Schluss wurde gezählt, wie oft die Analyse den korrekten Schlüssel finden konnte. Ein Schlüssel wurde als richtig definiert, sobald der durch ihn entstandene Klartext zu mindestens 75% mit dem originalen Klartext übereinstimmte.

Des Weiteren kann der Algorithmus mit der Laufdauer sowie der daraus resultierenden Schlüssel bewertet werden. Beim verwendeten Algorithmus ist vor allem

die Anzahl der korrekten Suchläufe maßgebend. Die Erfolgsrate wird zunächst als erstes Kriterium genommen, da eine viel zu lange Laufzeit für die Implementation nicht zur Anwendung kommen sollte.

Die Anwendung wurde auf beiden Testumgebungen aus Tabelle 6.2 installiert und ausgeführt. Die folgenden Messungen sind anhand der Reihenfolge aus Kapitel 6.1.1 durchgeführt. Sie sind jeweils in die Ergebnisse von Desktop und Server unterteilt.

6.1.3.1 Ausführungsdauer

Diese Messung prüft das Verhalten des Programms nach der Implementierung aller Module. Es wird getestet, ob das Programm mit den maximal vorgesehenen Belastungen zurecht kommt. Das bedeutet die Prüfung, ob das Programm bei einer maximal Nutzung zuverlässig und auch mit einer angemessenen Geschwindigkeit arbeitet. Bei einigen Auffälligkeiten wurde entsprechend eine Optimierung oder Anpassung vorgenommen, damit das System bei einer Überlastung (z. B. Speicherleck) rechtzeitig entlastet wird. Das Durchführen eines Leistungstests ist wichtig, da es teilweise zu Problemen bei der Implementierung in einer anderen Testumgebung kommen kann. Selbst bei äquivalenten Systembedingungen kann eine schlechte Performance bei einer maximal Auslastung auftreten. Das erbringt nicht die geforderte Stabilität.

Die Ausführungsdauer wurde bei der Server- und Desktop-Testumgebung mit jeweils einem und zwei Umkehrwalzen über 5 Rotoren (10 Verdrahtungen) ausgeführt. Innerhalb dieser Bedingungen ist ein spezifisches Verhalten bei der Anwendung zu erkennen. Die Abbildung 6.3 zeigt den Unterschied bei den Umkehrwalzen auf der Server-Testumgebung. Abbildung 6.4 bezieht sich auf die Desktop-Testumgebung.

Die Ausführungsdauer steht im linearen Verhältnis zur Länge des Textkandidaten, denn desto länger der Textkandidat um so stetig steigender ist die Ausführungsdauer. Das Verhältnis lässt sich durch die Linearisierung in Formel 6.7 für die Server-Testumgebung mit einer Umkehrwalze überführen. Dabei ist $0 < n$ die Länge, $runtime_{Server\text{–}1U}(n)$ die Ausführungszeit in Sekunden. Formel 6.8 zeigt die Funktion $runtime_{Server\text{–}2U}(n)$ für zwei Umkehrwalzen auf der Server-Testumgebung.

$$runtime_{Server\text{–}1U}\ (n) = 0,03\ n + 5,7 \tag{6.7}$$

$$runtime_{Server\text{–}2U}\ (n) = 0,07\ n + 8,5 \tag{6.8}$$

Ein Durchlauf entschlüsselt jeden Textkandidaten pro Buchstabe. Daher ist mit steigender Textlänge ein Mehraufwand bei größeren Texten zu erkennen. Das Verhalten der Anwendung war erwartet, da die vorherigen Berechnungen der Schlüsselräume diesen Indiz untermauerten. Die Maschine terminiert immer nach einer

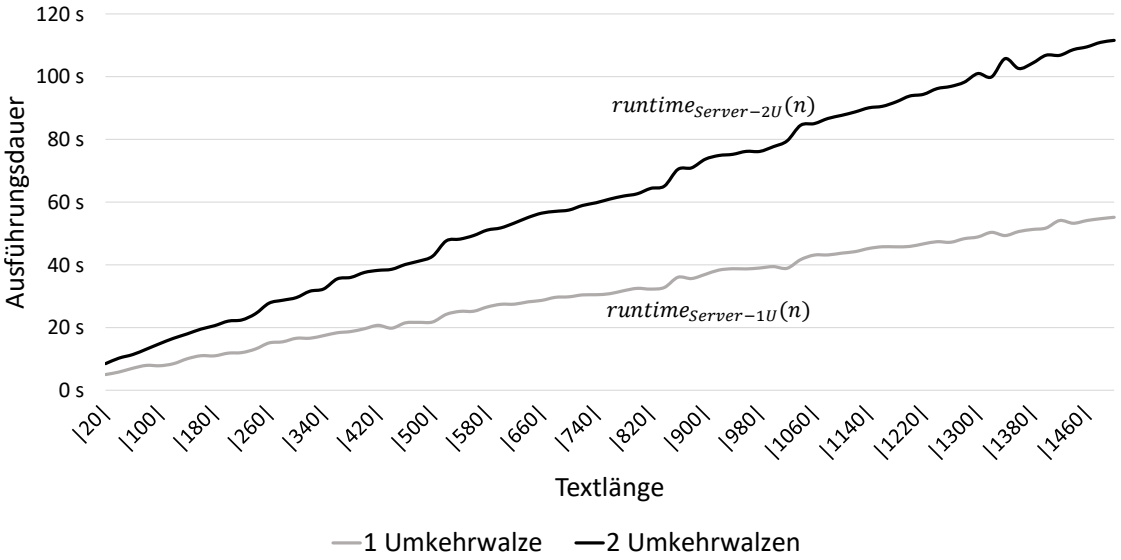


Abbildung 6.3: Ausführungsdauer zur Textlänge mit einer/zwei Umkehrwalzen (Server-Testumgebung)

berechenbaren Dauer. Es sind keine gleichbleibenden oder fallenden Kurvenabschnitte für die Server-Textumgebung vorhanden.

Bis zu 800 Zeichen wurden in 100 Tests pro gewählter Textlänge durchgeführt. Für die höheren Textlängen sind aufgrund der höheren Laufzeit weniger Testfälle vorhanden. Die unterschiedliche Anzahl an Testfälle erklärt die abrupte Kurvenänderung bei ca. 800 Zeichen in Abbildung 6.4. Je mehr Ergebnisse vorhanden sind, um so präziser ist die Funktionsdefinition möglich. Die Annäherung verläuft deutlich genauer.

Die Formeln 6.9 und 6.10 definieren die Funktionen $runtime_{Desktop\text{–}1U}$ und $runtime_{Desktop\text{–}2U}$ für die Desktop-Testumgebung, wobei $0 < n$ gilt.

$$runtime_{Desktop\text{–}1U}\ (n) = 0,03\ n + 23,8 \tag{6.9}$$

$$runtime_{Desktop\text{–}2U}\ (n) = 0,07\ n + 11,6 \tag{6.10}$$

Bei den Formeln 6.7 und 6.9 ist die gleiche Steigung von $0,3$ erwähnenswert. Die Formeln 6.8 und 6.10 mit einer Steigung von $0,07$ verhalten sich ebenso gleich. Nur die Verschiebung der Ausführungszeit ist bei diesen Formeln unterschiedlich. Dies liegt an der unterschiedlichen Rechenleistung ausgelöst. Unabhängig zur Rechenleistung verhält sich das Programm auf beiden Testumgebungen identisch.

Folglich sind die deterministischen Anforderungen an die Anwendung erfüllt. Ein deterministischer Ablauf garantiert gleiche Ergebnisse bei absolut identischen Eingabewerten.

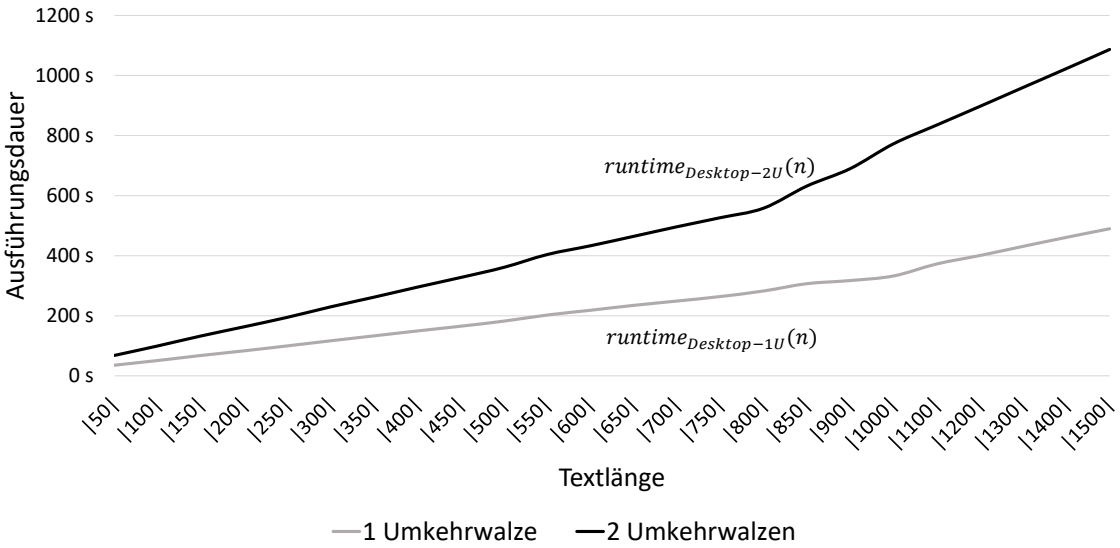


Abbildung 6.4: Ausführungsdauer zur Textlänge mit einer/zwei Umkehrwalzen (Desktop-Testumgebung)

6.1.3.2 Levenshtein-Distanz

Der Unterschied zwischen dem Textkandidaten und dem entschlüsselten Text wird mit der Levenshtein-Distanz aus Kapitel 6.1.1.2 berechnet. Je geringer die Distanz ist, desto eindeutiger wurde der korrekte Schlüssel für den Geheimtext gefunden.

Der Ciphertext-Only Angriff innerhalb der Textlänge 200 bis 1.000 Zeichen verläuft mit einer Auswahl an 5 Rotoren (10 Verdrahtungen) und zwei Umkehrwalzen sehr erfolgversprechend. Danach ist eine langsame Reduzierung der Erfolgsrate in Abbildung 6.5 zu erkennen. Die beiden Umkehrwalzen verdoppeln den Schlüsselraum von $48 \cdot 26^3 = 2^{19,69}$ auf $2^{20,67}$ (siehe Abschnitt 4.2.2.1) und umfassen die Schlüsselteile: Rotorauswahl, Grundeinstellung und Umkehrwalzen-Wahl.

Es bestätigt sich die Annahme, dass die Erfolgsrate des Algorithmus mit zunehmender Textlänge zuerst steigt. Mit steigender Textlänge zwischen 200 und 1.000 Zeichen ist die Wahrscheinlichkeit höher, dass die Buchstabenverteilung mit der Kostenfunktion besser detektiert wird und um so mehr der Verteilung einer natürlichen Sprache ähnelt. Es liegt also kein zufälliger Schlüssel vor, der eine gleichmäßige Buchstabenverteilung über dem entschlüsselten Text hätte.

Im Längenintervall 200 - 400 ist eine Erfolgsrate über 60% und im Längenintervall 400 - 1.000 sind über 80% möglich, einen abgefangenen Geheimtext aus den Einsatzjahren der Typex zu entschlüsseln. Das Zeichenintervall von 200 bis 1.000 ist der absolute Hauptfokus der Anwendung, da das Bedienpersonal die Anweisung besaß, Telegramme möglichst kurz zu halten. Telegramme wurden daher mit der Höchstlänge von 300 Zeichen erstellt wie bei der Enigma [11]. Bei einem und zwei Umkehrwalzen sinkt die Erfolgsrate ab ca. 1.000 Zeichen langsam über 2% ab. Bei einer Zeichenlänge unter 80 ist die Erfolgsrate nahezu bei 0%. Analystisch lässt

sich die Funktion $success_{U1/2}(n)$ für beide Graphen annäherungsweise über die Formel 6.11 definieren.

$$success_{U1/2}(n) = \begin{cases} -0,0002\ n^2 + 0,27\ n & 0 \leq n \leq 1000 \\ -0,02\ n + 96 & 1000 < n \end{cases} \tag{6.11}$$

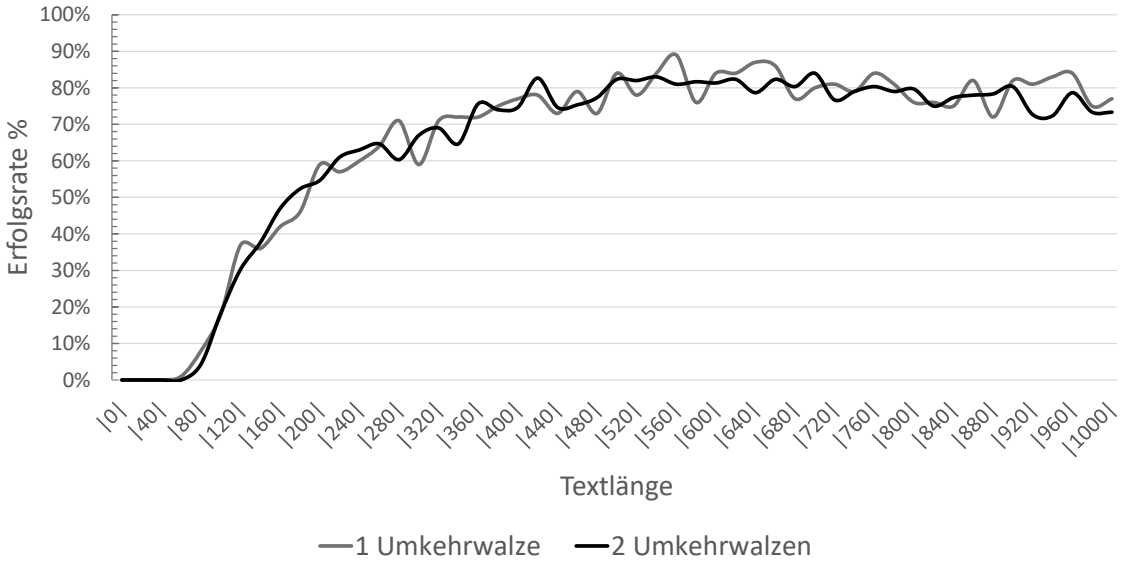


Abbildung 6.5: Erfolgsrate in % zur Textlänge mit einer und zwei Umkehrwalzen (Server-Testumgebung)

Abbildung 6.6 visualisiert die Erfolgsrate für die Desktop-Testumgebung mit jeweils einem und zwei Umkehrwalzen. Die Erfolgsrate weist keine Differenz zur Server-Testumgebung auf. Der Kurvenanstieg ab 80 Zeichen und der Kurvenabfall ab 1.000 Zeichen verhält sich identisch zu Formel 6.11.

Das Programm verläuft stabil und effizient auf unterschiedlichen Umgebungen. Abgesehen von der Rechenleistung sind die Graphen, welche aus den Testdaten hervorgingen, absolut identisch. Die Analyse ist mit einer hohen Erfolgsrate ausführbar.

6.2 Rätsel von Mystery Twister sowie Challenges vom FB 16

Die Anforderung R-5 in Kapitel 1.3 umfasst das Lösen der MysteryTwister Rätsel (MTC3) sowie das Brechen der vom Betreuer (FB 16) vorgegebenen Geheimtexte mit einem Ciphertext-Only Angriff. Die Rätsel MTC3 wurden auf ihre Programmfunktionalität geprüft. Des Weiteren wurden drei Geheimtexte vom Fachgebiet „FB16 – Angewandte Informationssicherheit“ erzeugt, die es mit dem entwickelten Kryptoanalyseprogramm zu brechen galt.

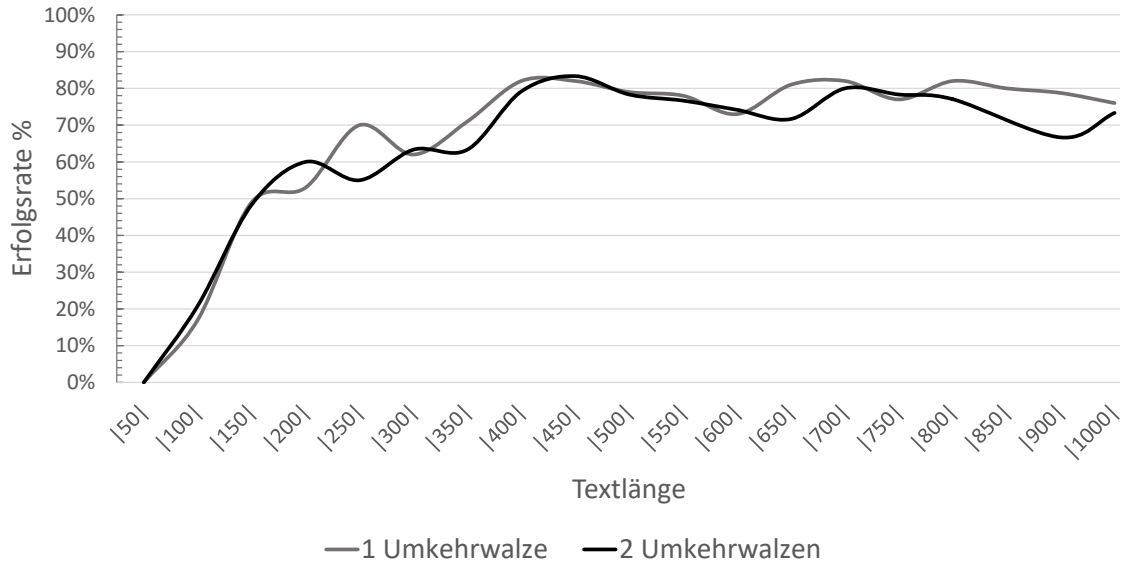


Abbildung 6.6: Erfolgsrate in % zur Textlänge mit einer und zwei Umkehrwalzen (Desktop-Testumgebung)

Rätsel von MysteryTwister (MTC3) Insgesamt haben die Kryptologen K. Chang, R. M. Low, und M. Stamp zwei verschiedene Rätsel zur Typex entworfen, die auf der MTC3-Website veröffentlicht wurden [61]: einen Known-Plaintext Angriff und die Suche nach der Rotorverdrahtung.

Das Auffinden der korrekten Rotorverdrahtung wurde in den Abschnitten 4.2.4 und 4.2.5 beschrieben. Das Anwenden des Algorithmus sollte für einen Rotor die Verdrahtung aufdecken können. Beim Rätsel handelt es sich um das Finden einer spezifischen Permutation. Einer der beiden Varianten sollte erfolgreich sein.

Das zweite Rätsel weist mehr Ähnlichkeit zum implementierten Ciphertext-Only auf. Es wurden zwei Lösungsansätze in dieser Arbeit behandelt, wobei beide die Statoren als bekannt voraussetzen:

(i) Mit Hilfe der Turing-Bombe wurde ein Known-Plaintext Angriff auf die Typex erklärt (siehe Abschnitt 4.2.3). Das Crib muss zu einem Menu aufbereitet werden. Anschließend kann es als Eingabe für die Turing-Bombe dienen.

(ii) Der implementierte Ciphertext-Only Angriff kann entweder korrekte Statoren oder Rotoren auffinden. Das eine setzt das andere als bekannt voraus. Diese beiden Verfahren lassen sich auch hintereinander ausführen, sodass die Laufzeit um ein Vielfaches verlängert wird. Es können alle möglichen Statoren mit allen Rotoreinstellungen geprüft werden. Der gesuchte Text hat eine Länge von 170 Zeichen. Bereits bei dieser Länge ist ein erfolgversprechendes Ergebnis zu erwarten.

    Dadurch dass hier ein Crib vorliegt, können mehr Information genutzt werden. Der Algorithmus kann von einem Ciphertext-Only Angriff auf einen Known-Plaintext Angriff geändert werden. Er kann um die Kostenfunktion

reduziert werden. Anstelle der Kostenfunktion wird ein Vergleich zum Klartext im Crib durchgeführt. Dieses Verfahren lässt den korrekten Schlüssel eindeutiger finden. Denn eine Kostenfunktion gibt generell nur eine Tendenz zu einem korrekten Ergebnis aus.

Rätsel vom FB16 – Angewandte Informationssicherheit Der Betreuer aus dem Fachbereich „Angewandte Informationssicherheit“ generierte mit dem Typex-Simulator aus dieser Arbeit drei Geheimtexte, die es zu brechen galt. Jeder Geheimtext hatte eine Länge von 254 Zeichen, die mit unterschiedlichen Schlüsseln chiffriert wurden. Es waren lediglich die verwendeten Statoren ohne Grundstellung bekannt. Die Aufgabenstellung gliederte sich in bekannte und zu suchende (unbekannte) Schlüsselteile. Für die drei Geheimtexte wurden folgende bekannte Informationen übergeben:

- Ringstellung: bekannt als AA für die Statoren
- Statoren: bekannt als Stator 1 und 2

Folgende Schlüsselteile galt es herauszufinden:

- Rotoren: unbekannt
- Umkehrwalze: unbekannt
- Grundstellung: unbekannt

Mit dem entwickelten Kryptoanalyseprogramm wurden die drei Geheimtexte analysiert, dessen Ergebnisse im Folgenden gezeigt werden. Es wird der Klartext mit gefundenem Schlüssel angegeben. Der gefundene Klartext benötigte kleinere Ausbesserungen. Diese sind grau markiert und hinsichtlich des aktuellen Wortkontextes verbessert.

(i) Der vorgegebene Geheimtext für **Rätsel 1** lautet:

```
YNVCVLHEEADOMKMMFKEPCEQEYLZDNPDXLHSNCRYPEGGLGZTQHONYVVBIBQWEFMUY
THWSWXUSZKMGAIALXUKHOFUGIHMVLSXNXCTYNOMHHTEUXSKGUKUEQWJAZHHCNBHJ
GTFCAFGTTDXUMWJOUOIYDJMXRFQNVHDUJYMHFZABPIFOXZHKPGEIIQCQUMMQNTMK
GUENGJYSMTICIHFQQGUNJVFSHJPXKRLXFIXDBMYELORMSSLBUWVSZQFFSBZYNY
```

Der korrespondierende Klartext lautet:

```
THE NAVY AND AIR MINISTER TO PRESIDENT OF THE BASQUE GOVERNMENT
WE ARE CONSIDERING THE IDEA TO ESTABLISH IN BILBAO AN AIRCRAFT
FACTORY WHERE THE MACHINE MARTIN BOMBERS AND THE FOKKER CAN BE
BUILT WELL I REQUEST YOU TO CALL THE AERONAUTICAL ENGINEER IN
LAMIACO CERRO SERVICE AND EXPLAINING HIM THIS IDEA GIV . . .
```

Der folgende Schlüssel wurde für die Dechiffrierung verwendet:

- Rotoren: 5, 3, -4
- Umkehrwalze: B
- Grundstellung: PQZAA
- Ringstellung der Rotoren: AAA

(ii) Das **Rätsel 2** hat folgenden Geheimtext:

```
AASESESHH-DHIXSTASJWTTVTMZRHIVHXQRVDAAZVKHNMWBGWALEJZSRGVUKJFVVG
UILKOPPFQCMWGWLKUWAFEBCTUOYDGXUPJMPBGAQXEADJCLZEHHBLVAPPPRUSSHLC
KVIEVRYIKWKXFKCUGKEKZHJQJGXLZKPMJPTMKWHLDRZQAXLSHUCYYQUFKIFKGJTM
UELWXNPGVFIGGJRAHYGCDLWLDHLHXKPRPYUQGQBFNCUOZFNAZUSZAURWBGISIMW
```

Der Klartext zum Rätsel lautet:

```
WILLIAM FREDERICK FRIEDMAN WAS A US ARMY CRYPTOGRAPHER WHO RAN
THE RESEARCH DIVISION OF THE ARMYS SIGNAL INTELLIGENCE SERVICE IN
THE THIRTIES AND PARTS OF ITS FOLLOW ON SERVICES INTO THE
FIFTIES. IN NINETY-FORTY SUBORDINATES OF HIS LED BY FRANK ROWLETT
BROKE JAPANS PURPLE CIPHER THUS DISCLOSING JAPANE
```

Der gefundene Klartext konnte mit dem folgenden Schlüssel erklärt werden:

- Rotoren: -3, 4, 5
- Umkehrwalze: A
- Grundstellung: NIKAA
- Ringstellung der Rotoren: AAZ

(iii) Der verschlüsselte Text für **Rätsel 3** lautet:

```
QYFWDCZSRCQZHCOGIMSQQXLVRLSYKAIZDSIIVYBZXGVMYOVEAURWOMNJZIZYHMVA
ZFXAWYMDQDJQQFEYOCLGAZHBOZATGHACVGIMOLVTYQTVSZVQSZRHQDTIXBAWFQVN
EMWMHOXSPFZXSANTKDZORJHRJSGWTBZEIMEVQDPTHCSQYHTWIXGLFOOZOHVEAWQN
BZAWEARXYKEBAPGKZHZPXNQXDMHOXHVEFNOLYIEQJJCMJFIOLMMUNVPEWMUZQS
```

Der gefundene Klartext ist wie folgt:

```
CRYPTOGRAPHY OR CRYPTOLOGY FROM GREEK KRYPTS HIDDEN SECRET AND
GRAPHE IN WRITING OR LOGIA STUDY RESPECTIVELY IS THE PRACTICE AND
STUDY OF TECHNIQUES FOR SECURE COMMUNICATION IN THE PRESENCE OF
THIRD PARTIE SCALLED ADVERSARIES MORE GENERALLY CRYPTOGRAPHY IS
ABOUT CONSTRUCTING AND ANALYZING PROTOC
```

Folgende Schlüsselteile erzielten die erfolgreiche Suche:

- Rotoren: -4, -3, -5
- Umkehrwalze: A
- Grundstellung: AISAA
- Ringstellung der Rotoren: AAA

Es wurden nur kleine Änderungen beim Klartext des Rätsels 2 vorgenommen. Abgesehen davon ist der Klartext von Rätsel 1 und 3 komplett ohne manuellen Eingriff gelöst worden. Aus diesem Grund konnten alle Geheimtexte erfolgreich entschlüsselt werden. Diese Lösungen wurden nur von der Kryptoanalyse-Komponente gefunden.

7 Zusammenfassung und Ausblick

Das Schlusskapitel fasst die Ergebnisse dieser Arbeit zusammen. Der Weg zu diesen Ergebnissen wurde im Hauptteil ausführlich dargelegt. Es wird überprüft, ob die gesteckten Ziele erreicht werden konnten. Des Weiteren werden Ideen, die während der Implementierung entstanden, zusammengefasst. Diese werden mit potenziellen Lösungsansätzen vorgestellt und vor dem Hintergrund einer einer zukünftigen Implementation diskutiert.

Die Typex-Maschine wurde kryptoanalysiert und mit verschiedenen Angriffsstrategien angegriffen. Dabei deckten sich viele Eigenschaften der deutschen Enigma mit der Typex, da diese ein praktikabler Nachbau der deutschen Version ist.

7.1 Zusammenfassung

In diesem Abschnitt werden die vordefinierten Ziele mit den erreichten Lösungen verglichen. Ferner wird argumentiert, ob ein Ziel erfolgreich erreicht oder nicht sinnvoll gelöst wurde. Hierbei soll bewertet werden, inwiefern die an die Arbeit gestellten Anforderungen erfüllt wurden.

7.1.1 (R-1) Implementierung der Typex-Chiffriermaschine als Simulator

Das Ziel bestand darin, beliebige Texte unter Angabe eines Schlüssels zu ver- und entschlüsseln. Dieser Vorgang sollte in einem Programm auf Basis der Typex realisiert werden. Mit Hilfe des Typex-Simulators ist dies gelungen. Dieser kann mit der Angabe eines Klartextes und eines Schlüssels den korrespondierenden Geheimtext ausgeben. Umgekehrt ist es unter Angabe des Geheimtextes und dem korrekten Schlüssel möglich, die entschlüsselte Nachricht zu erhalten. Der Simulator wurde möglichst abstrakt konzeptioniert, sodass weitere Rotoren einfach hinzugefügt und die Algorithmus-Änderungen unkompliziert eingebettet werden können. Eine Kommandozeilen-Oberfläche wurde zur gesteuerten Ein- und Ausgabe erstellt. Die Anforderung R-1 wurde also erfüllt.

7.1.2 (R-2) Analyse, Konzeption, Entwicklung und Implementierung von Kryptoanalysewerkzeugen für die Analyse der Typex

Es sollte ein Ciphertext-Only Angriff implementiert werden. James Gillogly führte erfolgreich einen Ciphertext-Only Angriff auf die Enigma aus und zeigte, dass die Kryptoanalyse auf abgefangenem Verkehr funktioniert. Analog hierzu wurde ein Ciphertext-Only Angriff (Kapitel 4.2.2) mit Randbedingungen erfolgreich auf die Typex implementiert. Diese Angriffsweise ist deutlich schwieriger als die Known-Plaintext Angriffe, die im Zweiten Weltkrieg favorisiert wurden.

Der Ciphertext-Only Angriff ist in zwei Phasen strukturiert, wobei die Rotoren oder Statoren komplett bekannt sind. Die erste Phase testet alle Typex-Schlüsselkombinationen (Auswahl von Rotoren, Grundeinstellungen und Rotorrichtung). Jede Schlüsselkombinationen, die zu einem korrekten Text führt, wird in einer Positiv-Liste gespeichert. Die inkorrekten Ergebnisse werden als zufällig definiert und verworfen. Für jedes Modell aus Phase 1 wird eine Phase 2 benötigt. Die zweite Phase probiert alle möglichen Ringstellungen auf jedem Modell aus der ersten Phase aus. Sie wird mit einer statistischen Metrik bewertet. Bei dieser Analyse werden mehrere der besten Kandidaten übernommen und der zweiten Phase zugeführt. Das Aufteilen des Angriffs in zwei Abläufe ist effektiv, da die erste Phase eine Tendenz ausgibt und als reduzierte Eingabe für Phase 2 dient.

Ein Known-Plaintext Angriff (Kapitel 4.2.3) konnte mit dem Verfahren von Alan Turing (Bletchley Park) auf die Typex angewendet werden. Mit einem Crib, den Rotor- oder Statoreinstellungen lässt sich die Eingabekonfiguration für die Turing-Bombe berechnen. Sie findet den größten Schlüsselteil von 17.576 Möglichkeiten für die Grundstellung der Rotoren oder Statoren heraus. Der geringe Schlüsselteil (Rotorreihenfolge und -auswahl) wird mit manuellen Taktiken durchgeführt. Der Known-Plaintext Angriff anhand der britischen Methode ist in dieser Arbeit theoretisch beschrieben und setzt die Turing-Bombe als definierte Funktion mit Ein- und Ausgabeparametern voraus.

Das Hillclimbing der Rotorverdrahtung von Kelly Chang [38] wird mit einem heuristischen Verfahren ausgeführt, dass innerhalb polynomieller Zeit mit einer Lösung terminiert. Als Kostenfunktion wird die Bigramm-Analyse verwendet. Sie garantiert gute Ergebnisse bei einer Substitutionschiffre, die äquivalent zu der Rotorverdrahtung eines Rotors ist. Marian Rejewski deckte im Jahre 1932 die interne Verdrahtung der Enigma Rotoren auf. Sein Angriff nutzte die ersten sechs Buchstaben eines jeden Telegramms, welches den Tagesschlüssel in verschlüsselter zweifacher Form beinhaltete. Wie im Known-Plaintext Angriff basiert es auf der Zyklen-Struktur innerhalb eines abgefangenen Geheimtextes. Die Enigma-Bediener waren jeden Tag angewiesen einen neuen Schlüssel zu verwenden und sich selbst eine dreistellige Kombination zufällig zu wählen, welche später mit dem Tagesschlüssel chiffriert und an jedes Telegramm in den führenden drei Stellen mit zweifacher Wiederholung platziert wurde. Dieser Fehler hatte kritische Auswirkungen und

befähigte Rejewski die Verdrahtung zu bestimmen. Das Schema von Rejewski, ist im Detail bekannt und kann unter bestimmten Annahmen auch auf der Typex starten. Sofern das Bedienpersonal bei der Typex ebenfalls eine solche Schlüsselfrequenz an jedes Telegramm heranstellt, wird auch dieser Angriff bei der Typex funktionieren.

Die vorgestellten Angriffe wurden konzeptionell beschrieben und dienen zur Entwicklung neuer Programme. Eine Angriffsart wurde programmiertechnisch realisiert. Auch der Ciphertext-Only Angriff ist dem Benutzer in einer Konsolenanwendung visuell zugänglich. Daher wurde Anforderung R-2 ebenfalls erfüllt.

7.1.3 (R-3) Evaluation der kryptographischen Stärke der Typex sowie Evaluation der entwickelten Kryptoanalyseverfahren

Im Ergebnis sind viele Texte mit der korrekten Grundeinstellung gefunden worden. Es ist offensichtlich, dass der Angriff auf kurze Texte mit dem Koinzidenzindex nicht erfolgreich ist. Grund ist, dass die Buchstabenverteilung sehr zufällig ausfällt und keine Tendenz bei solch kurzen Texten ablesbar ist. Vermutlich lässt die Kostenfunktion noch Entwicklungsbedarf zu. Das Programm ist objektorientiert entwickelt, so dass nur die statistische Metrik (Kostenfunktion) mit einer zukünftig besseren Methode ersetzt werden kann.

In Kapitel ist 6 ersichtlich, dass sobald die Textlänge unter 150 Zeichen fällt, die Sicherheit beim Chiffresystem drastisch steigt. Vorher lag die Erfolgsrate bei 80 % für eine korrekte Grundeinstellung über 250 Zeichen, für 500 Zeichen bei hingegen 90 %. Diese Erfolgsrate bestätigt, dass keine langen Texte mit dem gleichen Schlüssel bei der Typex transportiert werden sollten. Die Evaluation ist auf den implementierten Ciphertext-Only Angriff fokussiert. Anforderung R-3 wurde demnach erfüllt.

7.1.4 (R-4) Vergleich der kryptographischen Stärke mit der deutschen Enigma

Ziel ist es die Parallelen zwischen der Enigma und der Typex aufzuzeigen. Die Schlüsselräume der beiden Maschinen ähneln sich bis auf bestimmte Baueinheiten. Die Enigma verfügt über 77 Bit (Abschnitt 3.2.4.1) und die Typex über 65 Bit (Abschnitt 3.3.1).

Letztendlich hat sich das Typex-Chiffriersystem für viele Jahre, während und nach dem Krieg, als sicher erwiesen. Es erging der Typex ähnlich den klassischen Chriffresystemen, insbesondere wie der Enigma. Eine Analyse vom Geheimtext führt zu einem erfolgreichen Angriff, somit bricht es gleichartige Maschinen. Abgesehen davon, dass sich solch ein Angriff platzieren lässt, ist das Chiffresystem nicht anfällig

auf einen Angriff mit einer Textlänge unter 20 Zeichen, die den Koinzidenzindex als statistische Methode nutzt. Nach Shannons-Berechnung wird für einen praktikablen Angriff auf die Typex mindestens 20 Zeichen benötigt. Die Enigma wird in dem Kontext dieser Arbeit stetig erwähnt, da beide Maschinen täuschend ähnlich sind. Anforderung R-4 wurde ebenfalls erfüllt.

7.1.5 (R-5) Lösen von Rätseln aus Mystery Twister (MTC3) sowie Brechen vom FB16 vorgegebene Geheimtexte

Die zwei Rätsel von MTC3 sind: das Finden einer Rotor-Verdrahtung sowie ein Known-Plaintext Angriff mit einem Crib. Die vorangegangene Aufgabe war, einen Ciphertext-Only Angriff zu entwickeln. Dieser lässt sich nur mit Umbaumaßnahmen auf das Known-Plaintext-Rätsel anwenden. Ferner ist er gar nicht zur Aufdeckung der Rotor-Verdrahtung geeignet. Die theoretisch beschriebenen Methoden in dieser Arbeit sind Lösungsansätze für die beiden Rätsel.

Die Rätsel des Betreuers umfassten drei Geheimtexte, die es zu lösen galt. Mit der entwickelten Kryptoanalyse für Ciphertext-Only Angriffe ließ sich jeder Geheimtext mit dem zugehörigen Schlüssel lösen. Dies lässt auf eine generell hohe Erfolgsrate und Zuverlässigkeit schlussfolgern.

Die theoretischen Lösungsansätze und die Ergebnisse für die Rätsel werden in Kapitel 6.2 beschrieben. Die Anforderung R-5 ist als gelöst zu bewerten.

7.2 Ausblick

Der Known-Plaintext Angriff setzt die britische Methode von Alan Turing als definiert voraus. Die Turing-Bombe wurde bereits praktisch mit verschiedenen Programmiersprachen implementiert und könnte für die Typex, speziell mit zugehörige CrypTool2 Plugin, entwickelt werden. Sofern die Turing-Bombe nicht effizient genug ist, könnte ein Angriff mit stetigen Vergleichen zwischen Klartext und entschlüsseltem Textkandidaten auf verschiedenen Grundeinstellungen zielführender sein. Der Benutzer sollte den Beginn des Klartextes (crib) in dem abgefangenen Geheimtext auswählen können. Bisher wird davon ausgegangen, dass der Benutzer mit Hilfe des Crib-Verfahrens die Eingabe für die Turing-Maschine vorbereitet. Weiterhin sollte es möglich sein, nach bestimmten Wörtern im Geheimtext zu suchen. Sofern das Vorkommen eines gewisses Wortes im Geheimtext bekannt ist, sollte ein Partially Known-Plaintext Angriff möglich sein. Dies könnte für jede mögliche Position des Klartextes mit einem Known- Plaintext Angriff durchgeführt werden. Zusätzlich könnte ein Wörterbuch als Crib-Korpus eingebunden werden. Damit könnte eine Suche über alle Wörter, die mindestens eine gewählte Länge besitzen, konstruiert werden. Wird für einen Fall ein funktionierender Schlüssel gefunden, so wird dieser auf den gesamten Geheimtext angewendet.

Da der Ciphertext-Only Angriff bei Texten mit weniger als 150 Zeichen noch nicht zuverlässig den richtigen Schlüssel findet, besteht dort noch Entwicklungsbedarf. Die Verbesserung der verwendeten Kostenfunktion sollte bedacht werden. Bei der Analyse könnten auch Bigramme für kurze Texte einbezogen werden. Außerdem ist es vorstellbar, dass die Gewichtung der einzelnen Buchstaben noch optimiert wird. Es könnten ausgewählte Klartexte mit unterschiedlichen Metriken bewertet werden. Die Klartexte würden künstlich manipuliert und deren Verhalten mit der jeweiligen Buchstabenverteilung und Wahrscheinlichkeit über wechselnde Metriken analysiert.

Ab einer Textlänge von 1.000 Zeichen sinkt die Erfolgsrate beim Ciphertext-Only Angriff langsam ab. Da es unter Kriegsbedingungen unwahrscheinlich war, dass Telegramme mit mehr als 300 Zeichen gesendet wurden, lässt sich trotz dessen eine solche Nachricht angreifen. Ein Text mit 2.000 Zeichen könnte in vier Teile aufgeteilt und jeweils mit der Typex-Kryptoanalyse gestartet werden. Da die Erfolgsrate bei solch einer Textlänge deutlich höher ist als bei langen Texten, wird im Idealfall vier gleiche Ergebnisse entstehen oder ein Ergebnis wird mindestens zweimal vorkommen. Diese Aufteilung kann mit dem entwickelten Programm schon manuell vorgenommen werden und wäre zukünftig mit einer technischen Implementation realisierbar.

Des Weiteren werden bei Phase 1 im Ciphertext-Only alle Rotorpositionen ausprobiert. Der decodierte Text wird mit der statischen Methode bewertet. Diese Bewertung könnte auch vor der Decodierung platziert werden, indem eine Rotor-Verdrahtung vorher analysiert wird und ein passender Rotor, der zum Beispiel mit hoher Wahrscheinlichkeit ein „E“ auf einen häufigen Buchstaben im Geheimtext verdrahtet, in die Auswahl aufgenommen wird.

Von den mathematischen Ideen zur Schlüsselraumreduzierung abgesehen, werden die Technologien und Parallelverarbeitungen immer schneller und effektiver, sodass der Angriff um die gleichzeitige Suche für Rotorreihenfolge, Ring- und Grundstellung konzipiert werden kann. Das Verfahren wird umso effizienter, desto mehr der Bereich der möglichen Schlüssel eingeschränkt wird. Es ist logisch, dass je schneller die Computer werden und sich die Technologie verbessert, desto schneller wird die Schlüsselsuche. Eine Möglichkeit, die Suche effizienter zu gestalten, ist das Beteiligen weiterer Computer. Dabei wird der Bereich des Schlüssels aufgeteilt. Jeder Computer durchsucht nur noch seinen Teilbereich. Die allumfassende Schlüsselsuche ist ein wirksamer Angriff auf kryptographische Verfahren.

Der Typex-Simulator ist ebenfalls noch optimierbar. Es könnte eine gesonderte Behandlung von bisher unbekannten Zeichen berücksichtigt werden. Generell könnte der Eingabebereich auf deutlich größere Zeichenbereiche erweitert werden. Anstatt unbekannte Zeichen zu löschen oder zu ignorieren, könnten die neuen Zeichen mehrfach abgebildet werden oder gar eine eigene Abbildung im Zielalphabet erhalten. Mit der Vergrößerung des Alphabets wächst auch die Permutation eines Rotors.

Eine Implementation im CrypTool 2 (CT2) – Programm sollte angestrebt werden. Dies würde sowohl visuell als auch funktionell eine deutliche Bereicherung bedeuten. Dem Benutzer könnten die hier entwickelten Konzepte als Plugin-Erweiterung angeboten werden. Dies würde sich in der Benutzerfreundlichkeit, gegenüber der aktuellen Konsoleneingabe, signifikant auszeichnen. Des Weiteren gibt es in CT2 bestimmte Funktionalitäten zur Aufgabenverteilung bei großen Schlüsselräumen. Die Berechnung in einem verteilten Computernetzwerk ist zielführender und würde solche Angriffe zeitlich reduzieren.

Literaturverzeichnis

[1] M. Rejewski, "Mathematical solution of the enigma cipher," *Cryptologia*, vol. 6, no. 1, pp. 1–18, 1982.

[2] D. Kahn, *Seizing the enigma: the race to break the German U-boat codes, 1939-1943*. Barnes & Noble Publishing, 1991.

[3] J. J. Gillogly, "Ciphertext-only cryptanalysis of enigma," *Cryptologia*, vol. 19, no. 4, pp. 405–413, 1995.

[4] K. Schmeh, *Kryptografie: Verfahren, Protokolle, Infrastrukturen*. iX-Edition, dpunkt-Verlag, 2009.

[5] J. Ferris, *Intelligence and Strategy: Selected Essays*. Studies in Intelligence, Taylor & Francis, 2007.

[6] F. Ayoub and K. Singh, "Cryptographic techniques and network security," *Communications, Radar and Signal Processing, IEE Proceedings F*, vol. 131, pp. 684–694, December 1984.

[7] W. Stallings, *Cryptography and network security: principles and tices*. Pearson Education India, 2006.

[8] A. Beutelspacher, *Kryptologie*, vol. 7. Springer, 1996.

[9] W. F. Friedman, *The index of coincidence and its applications in cryptanalysis*. Aegean Park Press California, 1987.

[10] N. Kopal, *Secure Volunteer Computing for Distributed Cryptanalysis*. PhD thesis, Universität Kassel, Elektrotechnik/Informatik (FB16), 2017.

[11] F. L. Bauer, *Entzifferte geheimnisse: methoden und maximen der kryptologie*. Springer-Verlag, 2013.

[12] S. Spitz, M. Pramateftakis, and J. Swoboda, *Kryptographie und IT-Sicherheit: Grundlagen und Anwendungen*. Vieweg Teubner Verlag, 2011.

[13] J. Ferris, "The british army, signals and security in the desert campaign, 1940–42," *Intelligence and National Security*, vol. 5, no. 2, pp. 255–291, 1990.

[14] L. Knudsen and M. Robshaw, *The Block Cipher Companion*. Information Security and Cryptography, Springer Berlin Heidelberg, 2011.

[15] N. F. PUB, "46-3. data encryption standard," *Federal Information Processing Standards, National Bureau of Standards, US Department of Commerce*, 1977.

[16] W. C. Barker and E. B. Barker, "Sp 800-67 rev. 1. recommendation for the triple data encryption algorithm (tdea) block cipher," 2012.

[17] N. F. Pub, "197: Advanced encryption standard (aes)," *Federal information processing standards publication*, vol. 197, no. 441, p. 0311, 2001.

[18] T. Jakobsen, "A fast method for cryptanalysis of substitution ciphers," *Cryptologia*, vol. 19, no. 3, pp. 265–274, 1995.

[19] C. Christensen, "Polish mathematicians finding patterns in enigma messages," *Mathematics Magazine*, vol. 80, no. 4, pp. 247–273, 2007.

[20] A. J. Bagnall, G. P. McKeown, and V. J. Rayward-Smith, "The cryptanalysis of a three rotor machine using a genetic algorithm.," in *ICGA*, pp. 712–718, 1997.

[21] G. Welchman, *The hut six story: breaking the enigma codes*. McGraw-Hill Companies, 1982.

[22] R. Lewand, "Cryptological mathematics, the mathematical association of america, 2000."

[23] G. Ciano and M. Muggeridge, *Ciano's Diary: 1939-1943*. Heinemann, 1947.

[24] C. A. Deavours and L. Kruh, *Machine cryptography and modern cryptanalysis*. Artech House, Inc., 1985.

[25] R. Morris, "The hagelin cipher machine (m-209) reconstruction of the internal settings," *Cryptologia*, vol. 2, no. 3, pp. 267–289, 1978.

[26] B. Bevan, "The wireless-set-no19 group."

[27] J. Ferris, "The british enigma: Britain, signals security and cipher machines, 1906-1946 1," *Defense Analysis*, vol. 3, no. 2, pp. 153–163, 1987.

[28] J. Terraine, *The Right of the Line: The Royal Air Force in The European War, 1939-45*. Hodder & Stroughton, 1985.

[29] J. Rusbridger and E. Nave, *Betrayal at Pearl Harbor: How Churchill Lured Roosevelt into World War II*. Touchstone Books, 1992.

[30] L. Kruh and C. A. Deavours, "The typex cryptograph," *Cryptologia*, vol. 7, no. 2, pp. 145–166, 1983.

[31] J. Proc, "Betrayal at pearl harbor," 03.03.2017.

[32] P. Reuvers and M. Simons, "Operation of typex," Nov 2016.

[33] G. Waddington, "Hassgegner: German views of great britain in the later 1930s," *History*, vol. 81, no. 261, pp. 22–39, 1996.

[34] R. Mallett, *The Italian Navy and fascist expansionism, 1935-1940*. Routledge, 2013.

[35] D. Kahn, "Codebreaking in world wars i and ii: the major successes and failures, their causes and their effects," *The Historical Journal*, vol. 23, no. 03, pp. 617–639, 1980.

[36] D. Kahn, *Kahn on Codes: Secrets of the New Cryptology*. MacMillan Publishing Company, 1983.

[37] R. Erskine and F. Weierud, "Naval enigma: M4 and its rotors," *Cryptologia*, vol. 11, no. 4, pp. 235–244, 1987.

[38] K. Chang, R. M. Low, and M. Stamp, "Cryptanalysis of typex," *Cryptologia*, vol. 38, no. 2, pp. 116–132, 2014.

[39] "Patent 267472 - improvement in and relating to ciphering machines; convention date (germany) march 10, 1926," 1927.

[40] R. Szapranski, "A theory of information warfare; preparing for 2020," tech. rep., DTIC Document, 1995.

[41] A. Cebrowski, *Speech to the Heritage Foundation*. Office of Force Transformation, Department of Defense, 2003.

[42] D. A. Fulghum, "Fast forward the pentagon's force-transformation director takes an early swipe at what worked and what didn't in iraq," *Aviation Week and Space Technology*, vol. 158, no. 17, p. 34, 2003.

[43] R. Erskine, "Naval enigma: A missing link," *International Journal of Intelligence and Counter Intelligence*, vol. 3, no. 4, pp. 493–508, 1989.

[44] C. Deavours, "Unicity points in cryptanalysis," *Cryptologia*, vol. 1, no. 1, pp. 46–68, 1977.

[45] C. E. Shannon, "Communication theory of secrecy systems," *Bell Labs Technical Journal*, vol. 28, no. 4, pp. 656–715, 1949.

[46] J. Ferris, "The usual source: Signals intelligence and planning for the eighth army crusader offensive, 1941," *Intelligence and National Security*, vol. 14, no. 1, pp. 84–118, 1999.

[47] S. Singh, *The code book: the science of secrecy from ancient Egypt to quantum cryptography*. Anchor, 2000.

[48] M. Ekhall and F. Hallenberg, "Turing bombe simulator," *Turing Bombe Simulator*, 2012.

[49] A. Dhavare, R. M. Low, and M. Stamp, "Efficient cryptanalysis of homophonic substitution ciphers," *Cryptologia*, vol. 37, no. 3, pp. 250–281, 2013.

[50] J. Lawrence, "A study of rejewski's equations," *Cryptologia*, vol. 29, no. 3, pp. 233–247, 2005.

[51] D. Kahn, "The codebreakers: The story of secret writing, revised ed," *New York: Scribner*, 1996.

[52] W. F. Friedman and L. D. Callimahos, "Military cryptanalytics, part i," *Laguna Hills CA: Aegean Park Press*, 1985.

[53] A. Sinkov, "Elementary cryptanalysis: A mathematical approach, mathematical association of america, 1966," *Additional Reading*, 1966.

[54] R. Ganesan and A. T. Sherman, "Statistical techniques for language recognition: An introduction and guide for cryptanalysts," *Cryptologia*, vol. 17, no. 4, pp. 321–366, 1993.

[55] Wikipedia, "Enigma rotor details — wikipedia, the free encyclopedia," 2017. [Online; accessed 19-March-2017].

[56] J. Lyons, "Practical Cryptography," 2012.

[57] L. Carroll, A. Zimmermann, and J. Tenniel, *Alice Abenteuer im Wunderland*. Courier Corporation, 1869.

[58] G. FG Wissensverarbeitung (KDE), University of Kassel, "Internetsuchmaschinen."

[59] D. S. Hirschberg, *Pattern Matching Algorithms*. Oxford, UK: Oxford University Press, 1997.

[60] G. Kennedy, *An introduction to corpus linguistics*. Routledge, 2014.

[61] R. L. Kelly Chang, Mark Stamp, "Mystery twister c3 - the crypto challenge contest."

Versicherung an Eides Statt

Ich, Olaf Harland, Matrikelnummer 31202204, wohnhaft in 63571 Gelnhausen, versichere an Eides statt durch meine Unterschrift, dass ich die vorstehende Arbeit selbständig und ohne fremde Hilfe angefertigt und alle Stellen, die ich wörtlich oder annähernd wörtlich aus Veröffentlichungen übernommen habe, als solche kenntlich gemacht habe, mich auch keiner anderen als der angegebenen Literatur oder sonstiger Hilfsmittel bedient habe.

Ich versichere an Eides statt, dass ich die vorgenannten Angaben nach bestem Wissen und Gewissen gemacht habe und dass die Angaben der Wahrheit entsprechen und ich nichts verschwiegen habe.

Die Strafbarkeit einer falschen eidesstattlichen Versicherung ist mir bekannt, namentlich die Strafandrohung gemäß § 156 StGB bis zu drei Jahren Freiheitsstrafe oder Geldstrafe bei vorsätzlicher Begehung der Tat bzw. gemäß § 161 StGB bis zu einem Jahr Freiheitsstrafe oder Geldstrafe bei fahrlässiger Begehung.

Kassel, 10. Oktober 2017

Olaf Harland