Fakultät 4
Betriebssysteme und verteilte Systeme
1. Prüfer: Prof. Dr. Roland Wismüller

Fakultät 3
Institut für Wirtschaftsinformatik
2. Prüfer: Prof. Bernhard Esslinger
Betreuer: Dr. Nils Kopal

Masterarbeit:

Analyse symmetrischer Blockchiffren mittels differenzieller Kryptoanalyse in CrypTool 2

Universität: Universität Siegen

Studierender: Christian Bender
Matrikelnummer 1064298
Frankfurter Straße 151
57290 Neunkirchen
christian1.bender@student.uni-siegen.de

Studienfach: Master Informatik

Abgabe: 5.11.2019

Inhaltsverzeichnis

Zusammenfassung

Diese Masterarbeit ist im Bereich der Kryptologie angesiedelt und zum Abschluss des Informatik-Studiums verfasst worden. Gegenstand dieser Arbeit ist das Verfahren der differenziellen Kryptoanalyse (DKA), welche einer der bekannteren Angriffe auf moderne symmetrische Blockchiffren darstellt. Die Ergebnisse dieser Arbeit wurden in die Open-Source E-Learning-Software CrypTool 2 (CT2) eingebracht. CT2 enthält eine umfangreiche Auswahl an kryptographischen Verfahren und Angriffen. Allerdings existierte bisher weder in CT2 noch an anderer Stelle ein interaktives Angebot zu DKA.

Symmetrische Blockchiffren und deren Analysen bilden ein aktuelles Forschungsgebiet. Bekannt dafür sind beispielsweise die Ruhr-Universität Bochum und die Universität Haifa in Israel. Viele Beiträge präsentieren ihre Ergebnisse nur auf theoretischer Basis. Manche Evaluationen verlieren so an Transparenz und Nachvollziehbarkeit, insbesondere für unerfahrenere Leser. Des Weiteren existieren wenige bis keine Implementierungen, deren Quellcode öffentlich zugänglich ist. Darüber hinaus blieben die genauen Methoden und der Quellcode der praktisch erhobenen Ergebnisse oftmals unter Verschluss. Daher können Ergebnisse nicht reproduziert und validiert werden.

Erste Schritte zur Schließung dieser Lücke werden in dieser Arbeit anhand einer Implementierung in CT2 realisiert. Dafür ist ein interaktives Tutorial zur DKA mit drei unterschiedlichen Toy-Chiffren entwickelt worden. Es bietet erfahrenen und unerfahrenen Anwendern gleichermaßen sowohl einen theoretischen als auch einen praktischen Zugang zur DKA. Mittels Slides können sich Anwender in ihrem persönlichen Lerntempo die Grundlagen aneignen. Unterstützt wird die theoretische Basis durch eine ansprechende visuelle Darstellung. Über den hohen Grad an Interaktivität entsteht ein lückenloser Übergang zwischen Theorie und Praxis. Bei der praktischen Durchführung kann der Anwender den Schlüssel der zu brechenden Chiffre selbstständig festlegen und den Ablauf des Angriffs individuell steuern. Die Suche nach Charakteristiken (diese beschreiben einen Pfad durch eine Chiffre) kann auf dem eigenen Rechner durchgeführt werden. Um längere Rechenzeiten zu vermeiden, kann alternativ auf vorberechnete Daten zurückgegriffen werden. Gefundene Differenziale (eine Struktur, die Charakteristiken enthält) können in zwei unterschiedlichen Darstellungsformen (graphisch, tabellarisch) betrachtet werden. Bei der Wiederherstellung von Schlüsselbits wird der Erwartungswert und der empirisch erhaltene Wert der Treffer verglichen. Der Quellcode der Implementierung ist öffentlich zugänglich und kann von interessierten Anwendern nachvollzogen werden.

Einen weiteren Beitrag zur Schließung der oben genannten Lücke leisten die durchgeführten Experimente. Genaue Beschreibungen der Rahmenbedingungen sorgen für Transparenz und Nachvollziehbarkeit. Die Ergebnisse können mittels des verfügbaren Quellcodes reproduziert werden. Das entstandene Framework zur DKA kann leicht an weitere Substitutions-Permutations-Netzwerk-basierte Chiffren angepasst werden.

Im Zuge der Evaluationen konnte festgestellt werden, dass durchschnittlich 3,7 Nachrichten-Paare notwendig sind, um die erste Verschlüsselungsrunde einer Chiffre (bestehend aus zwei Rundenschlüsseln zu je 16 Bit) zu brechen. Beim Vergleich mit einem Brute-Force-Angriff mit 2^{31} Versuchen kann die Anzahl der zu testenden Schlüssel mittels DKA um den Faktor 2^{15} reduziert werden. Bei einer Chiffre mit einer Schlüssellänge von 96 Bit konnte die Anzahl der Schlüssel sogar um den Faktor 2^{79} verkleinert werden.

Abstract

This master thesis is about cryptology and was written at the end of the computer science studies. The subject of this thesis is the method of differential cryptanalysis (DCA), one of the better known attacks on modern symmetric block ciphers. The results of this work were incorporated into the open-source e-learning software CrypTool 2 (CT2). CT2 contains an extensive selection of cryptographic procedures and attacks. However, neither CT2 nor any other place offers an interactive tool on DCA.

Symmetric block ciphers and their analysis are a hot topic in current research. The Ruhr-Universität Bochum and the University of Haifa in Israel are known for this field. Many contributions present their results only on a theoretical basis. Some evaluations thus lose transparency and comprehensibility, especially for inexperienced readers. Furthermore, there are almost no meaningful implementations whose source code is publicly accessible. In addition, the exact methods and source code of the practically collected results often remained under lock from the public. Therefore, results cannot be reproduced and validated.

First steps to close this gap are realized in this work by means of an implementation in CT2. An interactive tutorial on the DCA with three different toy ciphers has been developed for this purpose. It offers experienced and inexperienced users both a theoretical and a practical approach to the DCA. Slides allow users to learn the basics at their own pace. The theoretical basis is supported by an appealing visual representation. The high degree of interactivity creates a seamless transition between theory and practice. During the practical execution, the user can independently determine the key of the cipher to be broken and individually control the course of the attack. The search for characteristics (these describe a path through a cipher) can be carried out on the user's own computer. In order to avoid longer computing times, pre-calculated data can be used as an alternative. Found differentials (a structure containing characteristics) can be viewed in two different ways (graphical, tabular). When key bits are restored, the expected value and the empirically obtained value of the hits are compared. The source code of the implementation is publicly accessible and can be traced by interested users.

A further contribution to closing the gap mentioned above is made by the experiments carried out. Exact descriptions of the framework conditions ensure transparency and traceability. The results can be reproduced using the available source code. The resulting framework for the DCA can easily be adapted to further substitution permutation network-based ciphers.

During the evaluations it was found that on average 3.7 message pairs are necessary to break the first encryption round of a cipher (consisting of two round keys of 16 bits each). Compared with a brute-force attack with 2^{31} attempts, the number of keys to be tested were reduced for a successful DCA by a factor 2^{15}. For a cipher with 96 bit key length the number of keys could even be reduced by the factor 2^{79}.

Danksagung

An dieser Stelle möchte ich mich bei den Personen bedanken, die mich während der Entstehung dieser Masterarbeit unterstützt haben.

Zunächst möchte ich mich bei Prof. Dr. Roland Wismüller und Prof. Bernhard Esslinger für die Themenstellung und die Betreuung bedanken und insbesondere bei Prof. Bernhard Esslinger für die vielen Anregungen und die anspruchsvolle, fordernde, aber auch führende Kritik. Darüber hinaus gilt ein besonderer Dank Dr. Nils Kopal für die sehr gute Betreuung, Unterstützung und die vielen hilfreichen Tipps und Ideen.

Ebenfalls möchte ich mich gerne bei Friedrich Wiemer und Phil Hebborn von der Ruhr-Universität in Bochum bedanken, die für fachliche Rückfragen zur differenziellen Kryptographie zur Verfügung standen.

Außerdem möchte ich Prof. Dr. Lars Knudsen (Technical University of Denmark) und Prof. Dr. Howard Heys (Memorial University of Newfoundland) meinen Dank aussprechen, die beide Fragen zu ihren Veröffentlichungen beantwortet haben.

1 Einleitung

In einer vernetzten Welt spielt die Sicherheit von Informationen eine stetig kritischer werdende Rolle. Ein Werkzeug zum sicheren Austausch von Informationen sind kryptographische Methoden. Die Wissenschaft der Kryptologie lässt sich in die Gebiete der Kryptographie und Kryptoanalyse einteilen. Während sich die Kryptographie mit dem Schutz von Daten beschäftigt, ist der Kern der Kryptoanalyse das Analysieren und Brechen von geheimen Nachrichten sowie der Analyse von kryptographischen Methoden [1, S. 2]. Analytische Angriffe versuchen, mathematische Schwächen von kryptographischen Verfahren zu finden und auszunutzen. Bis in die 1970er Jahre wurden kryptographische Methoden ausschließlich von Institutionen wie Militär und Regierungen verwendet [1, S. VII]. Mit zunehmender Entwicklung von Elektronik und Kommunikationsinfrastruktur sind weitere Anwendungen wie mobile Telefonie und im Bankenwesen hinzugekommen [1, S. IX]. Zusätzlich zur Verwendung von Kommunikationsgeräten im Alltag vergrößert sich der Anteil der Kommunikation zwischen Geräten verschiedenster Art. Im Jahr 2017 ermittelte der US-amerikanische Netzwerk-Ausstatter Cisco in einer Studie, dass die Anzahl der am Internet of Things (IoT) teilnehmenden Geräte bis 2021 von 17,1 auf 27,1 Milliarden ansteigen wird [2]. Klassische IoT-Anwendungen sind in den Bereichen Industrie 4.0, autonomem Fahren, Telemedizin oder Smart Grids zu finden.

Zum Schutz von Daten wurde eine Vielzahl von kryptographischen Methoden entwickelt. Im Bereich der symmetrischen Kryptographie existieren zwei Hauptklassen von Chiffren – Stromchiffren und Blockchiffren. Eine der älteren und bekannteren symmetrischen Blockchiffren der modernen Kryptographie ist der Data Encryption Standard (DES).

1.1 Motivation

Seit Mitte der 1970er Jahre beschäftigen sich viele Forscher von Universitäten und Geheimdiensten mit dem Aufdecken von Schwächen im DES [1, S. 85]. 1990 fanden die israelischen Forscher Eli Biham und Adi Shamir einen Angriff gegen den DES [3, S. 85], welcher als differenzielle Kryptoanalyse (DKA) bekannt und Teil der modernen Kryptoanalyse ist. Ziel des Verfahrens ist es, iterativ einzelne Schlüsselbits bis hin zu vollständigen Rundenschlüsseln wiederherzustellen. Durch sukzessive Anwendung ist es möglich, alle Rundenschlüssel zu erhalten und im Idealfall Nachrichten zu entschlüsseln. Es hat sich herausgestellt, dass dieser Angriff grundsätzlich auf alle symmetrischen Blockchiffren anwendbar ist [1, S. 85]. Biham und Shamir konnten zeigen,

dass die S-Boxen des DES besonders resistent gegen differenzielle Kryptoanalyse sind. Nach der Veröffentlichung des Angriffs erklärte ein damaliges Mitglied des DES-Design-Teams von IBM, dass zur Entwicklungszeit des DES dieser Angriff bereits bekannt war [1, S. 85]. Neben der historischen Bedeutung des Angriffs ist dieser mittlerweile ein wichtiges Werkzeug von Kryptoanalysten zur Analyse von neuen symmetrischen Blockchiffren [4].

In den letzten Jahren wurde das Verständnis der differenziellen Kryptoanalyse in vielen Bereichen vorangetrieben. So versuchen Kryptoanalytiker häufig, die Analyseergebnisse mit verschiedenen mathematischen Optimierungen zu verbessern. Beispielsweise verwendet man in diesem Kontext Mixed Integer Linear Programming (MILP) zur Lösung von zweierlei Problemen: einerseits zur Berechnung der minimalen Anzahl von aktiven S-Boxen und andererseits zur Suche der besten Charakteristik [5, S. 1]. Allerdings ist die Effizienz dieser Verfahren häufig nicht zufriedenstellend. Folglich wird zu der Analyse moderner Blockchiffren weiter aktiv geforscht. Beispielhaft seien an dieser Stelle die Arbeitsgruppe für Symmetrische Kryptographie der Ruhr-Universität Bochum [6] und das Department of Computer Science der Universität Haifa erwähnt [7], die regelmäßig wissenschaftliche Artikel publizieren. Differenzielle Kryptoanalyse stellt für etablierte kryptographische Verfahren zwar keine reale Gefahr für die Vertraulichkeit dar, allerdings spielt sie eine große Rolle bei der Analyse und Überprüfung der Verfahren.

Die E-Learning-Plattform CrypTool 2 (CT2) ist eine Software, die dem Anwender kryptographische Verfahren und kryptoanalytische Werkzeuge zur Verfügung stellt, mit dem Ziel, diese verstehen und verwenden zu können [8]. In CT2 können intuitiv per Drag & Drop verschiedene Verfahren und Werkzeuge miteinander kombiniert und so eigene Abläufe konstruiert werden. Der Nachfolger des bekannten CrypTool 1 umfasst bereits eine umfangreiche Auswahl an Werkzeugen zur Kryptoanalyse. Ein Werkzeug zur differenziellen Kryptoanalyse ist nicht enthalten.

1.2 Ziele der Arbeit

Im Rahmen dieser Masterarbeit soll das Verfahren der differenziellen Kryptoanalyse untersucht und implementiert werden. Die Implementierung soll in CT2 integriert werden und Tutorial-Charakter haben. Das bedeutet insbesondere, dass die Idee des Verfahrens unerfahrenen Anwendern verdeutlicht werden kann. Ausgangspunkt des Tutorials sollen verschiedene so genannte Toy-Chiffren sein. Der Verwendungszweck von Toy-Chiffren ist es, primär kryptographische Prinzipien darzustellen oder an ihnen verschiedene Kryptoanalysen durchzuführen. Diese eignen sich didaktisch besonders gut aufgrund

ihrer einfachen Konstruktion im Gegensatz zu real verwendeten kryptographischen Chiffren wie dem Advanced Encryption Standard (AES). Das Tutorial soll dem Anwender die Möglichkeit bieten, eine differenzielle Kryptoanalyse an den Chiffren durchzuführen. Die Toy-Chiffren werden so gewählt, dass diese einen steigenden Grad an Komplexität in ihren Parametern wie Schlüssellänge, Anzahl der Runden und in ihrer Konstruktion aufweisen. Ziel dabei ist es, das Verfahren der differenziellen Kryptoanalyse sukzessive durch verschiedene Techniken wie Differenzen, Charakteristiken, Differenziale und Filterung zu verfeinern und anwenderfreundlich zu präsentieren.

Des Weiteren soll untersucht werden, wie generisch die differenzielle Kryptoanalyse auf Blockchiffren angewandt werden kann. Dabei gilt es zu diskutieren, welche Parameter die Erfolgsaussichten des Angriffs beeinflussen. Mögliche Parameter sind beispielsweise die Anzahl an gewählten Nachrichten-Paaren, die Blocklänge oder die Struktur und Eigenschaften einer Blockchiffre. Ebenso soll das Verfahren mit Brute-Force verglichen werden.

Zusammenfassend sind folgende Ziele zu erreichen:

Z1: Untersuchung des Verfahrens der differenziellen Kryptoanalyse

Z2: Design und Aufbau eines Tutorials für differenzielle Kryptoanalyse mit drei Toy-Chiffren in CT2

Z3: Rekonstruktion der Rundenschlüssel der Toy-Chiffren mittels differenzieller Kryptoanalyse in CT2

Z4: Analyse und Evaluation der differenziellen Kryptoanalyse auf Basis der Implementierung

Z5: Vergleich der differenziellen Kryptoanalyse mit anderen Angriffen

1.3 Aufbau der Arbeit

Diese Masterarbeit ist in acht Kapitel eingeteilt. In Kapitel 2 werden die Grundlagen für die Arbeit beschrieben. Im darauf folgenden Kapitel 3 wird das Verfahren der differenziellen Kryptoanalyse vorgestellt. Kapitel 4 beschreibt das Konzept für das zu entwerfende Plugin in CT2, welches die differenzielle Kryptoanalyse für verschiedene Chiffren bereitstellt. In der Konzeptphase werden primär Überlegungen zur sinnvollen Strukturierung der Umsetzung angestellt. In Kapitel 5 werden das Design und die Implementierung der Anwendungen erläutert. Dabei wird das Programm zunächst als Konsolenanwendung projektiert und der resultierende Quellcode anschließend in

CT2 migriert. In Kapitel 6 werden die zu untersuchenden Toy-Chiffren mit der zuvor beschriebenen Implementierung der differenzielle Kryptoanalyse analysiert. Das vorletzte Kapitel gibt einen Überblick über verwandte Arbeiten. In Kapitel 8 werden die zuvor definierten Ziele auf Erfolg geprüft, die wichtigsten Ergebnisse zusammengefasst und ein Überblick über mögliche zukünftige Aktivitäten gegeben.

1.4 Verwendete Software

Diese Masterarbeit wurde mit der Textverarbeitungs-Software LaTeX verfasst. Abbildungen wurden mit der Software Visio 2016 erstellt. Alle Diagramme sind mit Matlab erzeugt worden. Die Implementierung wurde mit der Entwicklungsumgebung Visual Studio 2017 durchgeführt.

2 Grundlagen

In diesem Kapitel werden die für diese Arbeit notwendigen Grundlagen und Bezeichnungen eingeführt. Zunächst wird die E-Learning-Software CrypTool 2 ausführlich beschrieben. Dann werden das Gebiet der Kryptologie und dessen Einteilung dargelegt. Anschließend werden die Strukturen beschrieben, aus denen die in dieser Arbeit verwendeten Toy-Chiffren konstruiert sind. Als Letztes werden die Toy-Chiffren spezifiziert, an denen im Rahmen des zu entwickelnden Tutorials in CT2 die differenzielle Kryptoanalyse durchführt werden kann.

2.1 CrypTool 2[1]

Das Projekt „CrypTool“ (CT) wurde 1998 ins Leben gerufen. Seit 2003 ist es ein Open-Source-Projekt. Die maßgeblichen Resultate des Projekts sind die E-Learning-Programme CrypTool 1 mit den Nachfolgern CrypTool 2 (CT2) und JCrypTool (ab 2007/2008). Inzwischen zählen sie zu den weltweit am weitesten verbreiteten Lern-Programmen im Bereich Kryptografie und Kryptoanalyse und finden Anwendung sowohl in Lehre als auch in Aus- und Fortbildung. [10]

CT2 ist eine Windows-Anwendung und wird in der Programmiersprache C# [11] entwickelt. CT2 basiert auf dem derzeit aktuellen .NET-Framework 4.7.2 und der Windows Presentation Foundation WPF [12], die eine Vektorgrafikbasierte Oberfläche erlaubt. Die Architektur der Software ist modular, so dass sie ohne großen Aufwand um neue Funktionen ergänzt werden kann. [10] Die Programme werden pro Monat rund 10.000 mal von der CT-Webseite heruntergeladen.

Die Oberflächenelemente *Startcenter*, *Arbeitsbereich* und *Komponenten* werden vom Benutzer am häufigsten verwendet. Im Folgenden wird ihre Bedienung kurz beschrieben.

2.1.1 Das Startcenter

Beim Starten von CT2 öffnet sich das Startcenter (Abbildung 1). In diesem gibt es vier Bereiche: *Hauptfunktionen*, *Vorlagen*, *Neues* und *Zuletzt geöffnete Vorlagen*:

[1] Dieser Abschnitt ist größtenteils von [9] übernommen.

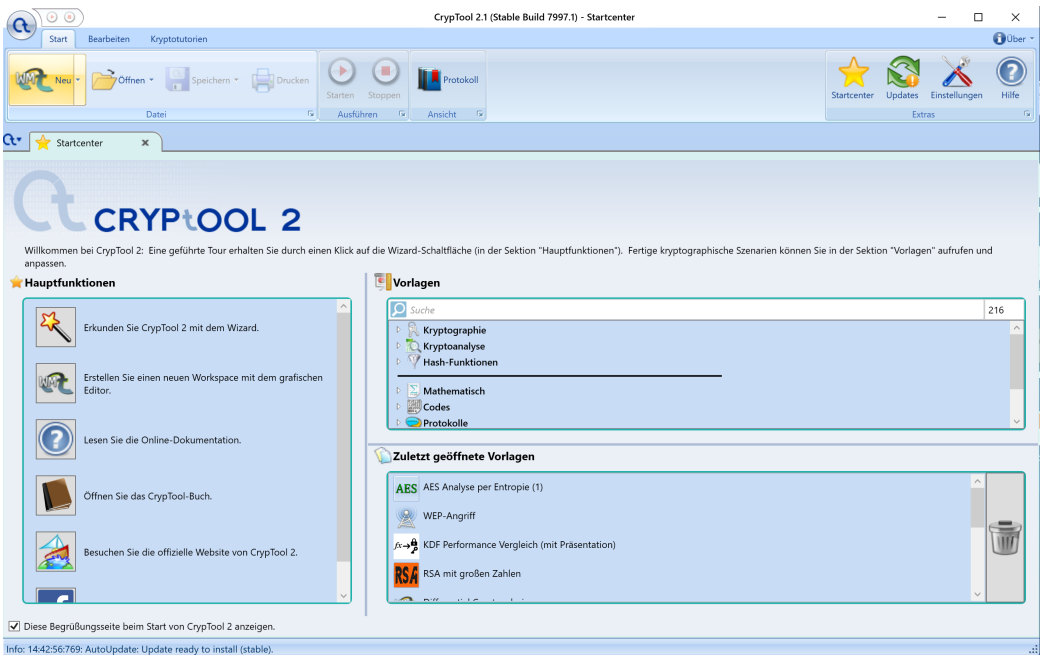


Abbildung 1: *CT2-Startcenter*

- Über die *Hauptfunktionen* kann man bspw. mit dem Wizard das Programm erkunden, einen neuen Arbeitsbereich (Workspace) mit dem eingebauten grafischen Editor erstellen, das CrypTool-Buch oder die Online-Dokumentation lesen sowie die offizielle Webseite oder die Facebook-Seite zu CrypTool 2 besuchen.
- Im Bereich *Neues* sieht man die aktuellsten Änderungen am Projekt.
- Eine Vorlage (Template) ist ein grafisches Programm, das ein ganzes Szenario enthält. Dazu nutzt es eine oder mehrere Komponenten, die miteinander verbunden sind. Vorlagen werden als Ganzes in den Arbeitsbereich geladen. Alle verfügbaren Vorlagen findet man über die eingebaute Vorlagen-Suchfunktion oben im Bereich *Vorlagen*.
- Bereits vom Benutzer geöffnete Vorlagen werden in eine Liste unter *Zuletzt geöffnete Vorlagen* geschrieben.

2.1.2 Der Arbeitsbereich

Wenn man eine Vorlage im Arbeitsbereich öffnet, zeigt CT2 **sechs** Bereiche, wie in Abbildung 2 dargestellt:

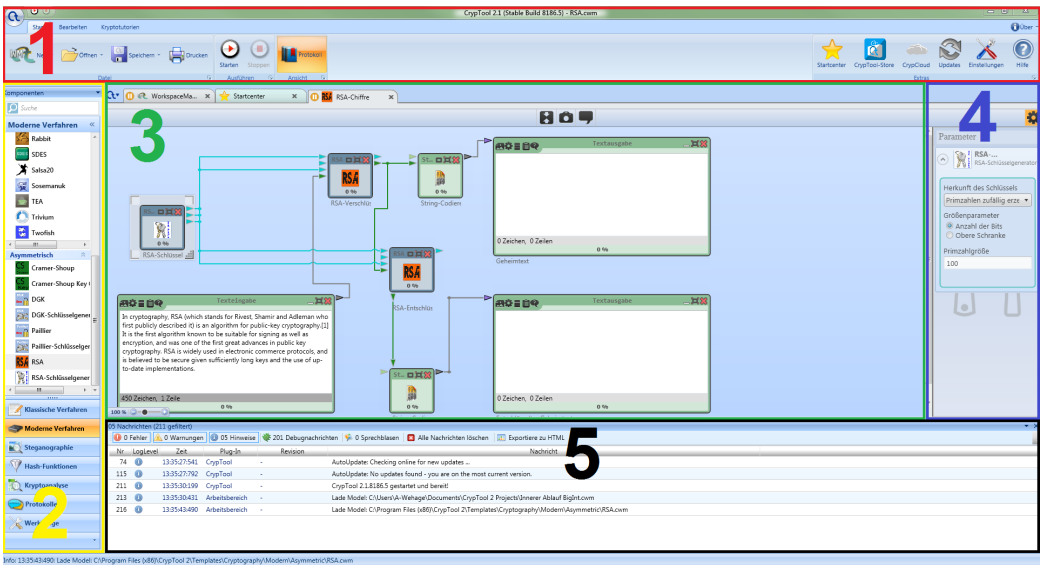


Abbildung 2: *CT2-Arbeitsbereich mit geöffneter RSA-Chiffre-Vorlage*

1. Im Bereich mit der Nummer 1 (oben, rot umrahmt) sind die Titelleiste, das Hauptmenü und die Ribbonbar zu sehen. Hier kann man unter anderem die geöffneten Programme starten oder stoppen.

2. Im zweiten Bereich (gelb umrahmt) sind alle verfügbaren Komponenten zu sehen, eingeteilt in Kategorien (Klassische Verfahren, Moderne Verfahren, Steganografie, Hash-Funktionen, usw.). Zusätzlich können diese über die Komponenten-Suchfunktion oben schnell gefunden werden. Die in dieser Masterarbeit neu zu implementierenden Komponenten werden in die Kategorie *Moderne Verfahren – Asymmetrisch* eingeordnet.

3. Der dritte Bereich (grün umrahmt) ist die Arbeitsfläche, in der fertige Vorlagen geladen und ausgeführt, oder eigene Workflows und Vorlagen erstellt werden können. Die Komponenten auf der Arbeitsfläche können über ihre Ein- und Ausgänge miteinander verbunden werden. Hier ist exemplarisch die RSA-Komponente zu sehen, die eine Nachricht als Texteingabe annimmt und verschlüsselt bzw. entschlüsselt (unter Zuhilfenahme folgender Parameter: des öffentlichen Schlüssels, eines Modulus und eines privaten Schlüssels).

4. Rechts daneben (blau umrahmt) liegt der vierte Bereich, in dem die Parameter angezeigt werden, die sich bei der selektierten Komponente einstellen lassen. Parameter können nur verändert werden, wenn das

Programm gestoppt ist. Die Parameterleiste einer Komponente wird mit Strg+I oder mit einem Klick auf das Zahnrad rechts oben geöffnet.

5. Im fünften Bereich (unten, schwarz umrahmt) sieht man die Nachrichtenkonsole (Log): In dieser werden alle Fehler, Warnungen Hinweise oder sonstige Nachrichten angezeigt, die während der Nutzung von CT2 entstehen.

6. Ganz unten ist die Statuszeile.

Mit F11 werden der gelbe und der schwarze Bereich (Nr. 2 und Nr. 5) ausgeblendet. Mit F12 werden die Ribbonbar mit ihren großen Ikonen (im roten Bereich, Nr. 1) und die Statuszeile ganz unten ausgeblendet. Beide Funktionstasten arbeiten alternierend. Damit kann man schnell mehr Platz für den Arbeitsbereich (Bereich Nr. 3) schaffen.

2.1.3 Komponenten

Entwickler, die Erweiterungen für CT2 schreiben, bauen ein Plugin. Ein Plugin ist eine .NET-Assembly, die eine oder mehrere Komponenten beinhaltet. [10]

Eine Komponente ist eine Funktion, die auf dem Arbeitsbereich liegt und nach dem Starten (Drücken des Starten-Buttons; im Englischen Play-Button) berechnet und ausgeführt wird. Im minimierten Zustand wird eine Komponente als Icon mit Ein- und Ausgängen dargestellt. Abbildung 3 zeigt eine typische Komponente im maximierten (aufgeklappten) Zustand, bei der im Präsentationsbereich die Einstellungen angezeigt werden.

Die Komponente in Abb. 3 wurde in vier Bereiche unterteilt:

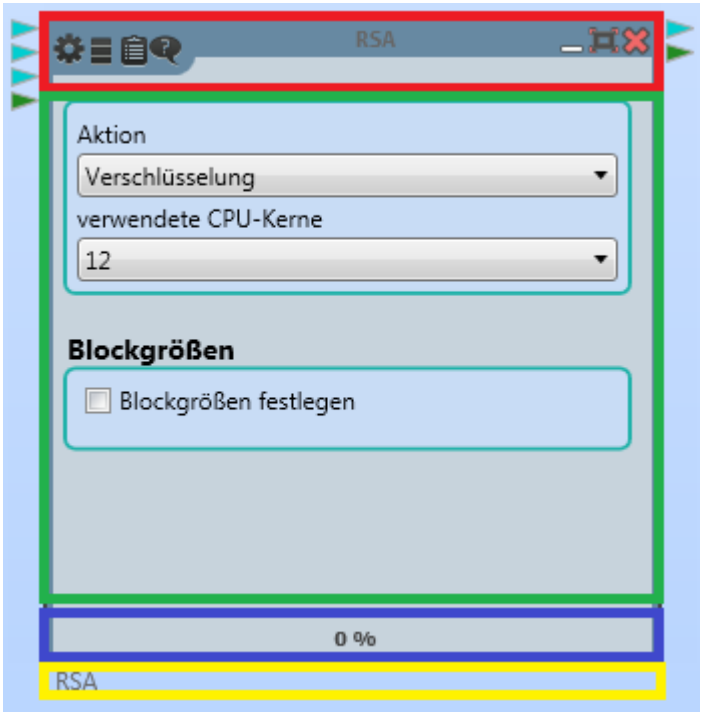


Abbildung 3: *RSA-Komponente mit Einstellungen*

- Im oberen, rot umrahmten Bereich sind links die drei Ikonen, mit denen man zwischen den Ansichten, die im Präsentationsbereich der Komponente dargestellt werden, wechseln kann. Mit der 4. Ikone kann man die zugehörige Onlinehilfe aufrufen. Mittig steht der Name der Komponente und rechts hat man die Möglichkeit, das Fenster zu minimieren, zu maximieren oder zu schließen.

- Im grün umrahmten Bereich (Präsentationsbereich) wird die eingestellte Ansicht angezeigt, in diesem Fall die Einstellungen. Parameter der Einstellungen können auch über die Eingänge einer Komponente gesetzt werden. Normalerweise hat eine Komponente Default-Werte für ihre Parameter. Gibt es einen Eingang dazu, überschreibt er den Defaultwert. Während der Ausführung einer Komponente können die Parameterwerte innerhalb der Komponente nicht geändert werden; sind sie jedoch mit Eingängen verbunden, können sie auch während der Ausführung geändert werden.

- Der blaue Bereich beinhaltet eine Fortschrittsanzeige.

- Unterhalb der Komponente, im gelben Bereich, steht eine kurze Beschreibung, die man frei wählen kann.

Über die ein- und ausgehenden Konnektoren (Dreiecke), links und rechts des roten Bereichs, kann die Komponente mit anderen Komponenten verbunden werden. Wie viele Ein- und Ausgänge eine Komponente hat, wird vom Programmierer festgelegt. Konnektoren können mit der Flag *mandatory* versehen werden, womit vom Programmierer festgelegt wird, dass ein Konnektor verbunden sein muss. Ohne diese Flag ist der Default, dass ein Konnektor nicht verbunden sein muss.

2.1.4 Der Lebenszyklus einer Komponente

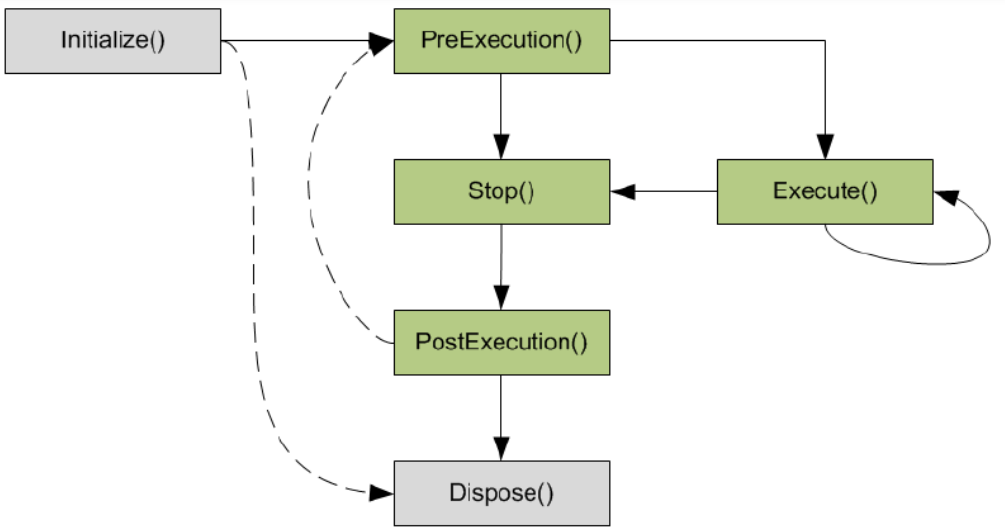


Abbildung 4: *Lebenszyklus einer Komponente [13]*

Der Lebenszyklus einer Komponente (siehe Abbildung 4) beginnt damit, dass sie – beim Laden in den Workspace – initialisiert wird. Dabei wird die Methode *Initialize()* aufgerufen. In dieser Methode können initiale Einstellungen vorgenommen werden, zum Beispiel das Laden einer Ansicht für eine Präsentation. Sobald man die Komponente mit einem Klick auf *Starten* (im Englischen *Play*) ausführt, startet CT2 alle Komponenten auf der Arbeitsfläche. Dabei wird als Erstes bei allen Komponenten auf der Arbeitsfläche die Methode *PreExecution()* ausgeführt. In dieser Methode können gewisse Vorarbeiten geleistet werden wie das Aufbauen einer Verbindung zu einem Server oder das Prüfen bestimmter Einstellungen.

Anschließend wird die Methode *Execute()* ausgeführt. Hier finden die eigentlichen Berechnungen der Komponente statt. Diese wird immer wieder neu aufgerufen, sobald sich einer der Eingänge verändert.

Mit Betätigung der Schaltfläche *Stoppen* wird die laufende Ausführung der Arbeitsfläche beendet. Dadurch wird in jeder Komponente auf der Arbeitsfläche die Methode *Stop()* ausgeführt, welche interne Berechnungen unterbricht. CT2 beendet alle laufenden Threads und ruft abschließend die Methode *PostExecution()* auf. Diese Methode ist das Äquivalent zu *PreExecution()*, hier kann die Komponente in ihren Ursprungszustand zurückversetzt werden.

Entfernt man eine Komponente oder schließt die Arbeitsfläche, wird die Methode *Dispose()* ausgeführt, in der noch verwendete Ressourcen freigegeben werden können. Damit ist der Lebenszyklus einer Komponente beendet. Der Prozess startet von Neuem, sobald eine Komponente neu auf die Arbeitsfläche gezogen wird.

2.1.5 CT2 als Programm

Das CT2-Projekt wird derzeit unterteilt in einen Nightly Build (NB) mit aktuellen Änderungen und einen Stable Build (SB) mit halbjährlich freigegebenen und geprüften Änderungen. Die SB-Version ist hierbei die offizielle Releaseversion. Beide Varianten sind als ZIP-Datei und als Setup-Executable verfügbar und beide verfügen über einen automatischen Update-Mechanismus. Die aktuelle Releaseversion ist als Setup-Datei [CrypTool 2.1 (Stable Build 8186.5) vom 25.6.2019] knapp 150 MB groß und enthält rund 220 Templates und ca. 180 verschiedene Komponenten.

Für diese Arbeit sind nur die grundlegenden Funktionen von CT2 von Bedeutung. Diese sind unter anderem die Online-Hilfe, Templates, Auswählen und Nutzen von Plugins mit ihren Einstellungen, Starten und Stoppen von Workspaces sowie die Präsentationsmodi von Komponenten. Auf Bestandteile wie den CT-Store (zum Nachladen weiterer Plugins oder großer Statistik-Dateien) oder auf die CrypCloud (zum verteilten Rechnen in Ad-hoc-Netzen) wird nicht weiter eingegangen.

2.2 Kryptologie

Bereits im 5. Jahrhundert v. Chr. verwendeten die Griechen eine auf Transposition basierende Chiffre namens Skytale [14, S. 105-106]. Als mathematische

Disziplin wird die Kryptologie unter anderem durch Zahlentheorie, Gruppentheorie, Kombinatorik, Statistik, Stochastik, Komplexitätstheorie und Informationstheorie beeinflusst [14, S. 2-3]. In [14, S. 2-8] ist beschrieben, dass zunächst Militär, Geheimdienste, staatliche Institutionen und Diplomaten primär Interesse an Kryptologie hatten. Durch schnelle Entwicklungen im Bereich der Mikroelektronik wuchs auch das Interesse der Informatik an Kryptologie [14, S. V]. Durch Anwendungen im zivilen Bereich vergrößerte sich zudem das kommerzielle und private Interesse an Kryptologie [14, S. 6].

Die Kryptologie lässt sich in die Gebiete Kryptographie und Kryptoanalyse (Abb. 5) aufteilen. Die Kryptographie beschäftigt sich mit dem Schutz von Daten. Dazu existieren verschiedene kryptographische Methoden, mit denen Schutzziele wie Vertraulichkeit oder Integrität erreicht werden können. Der Kryptographie steht das Gebiet der Kryptoanalyse gegenüber. Ziel ist das Analysieren und Brechen von kryptographischen Methoden und geheimen Nachrichten [1, S. 2].

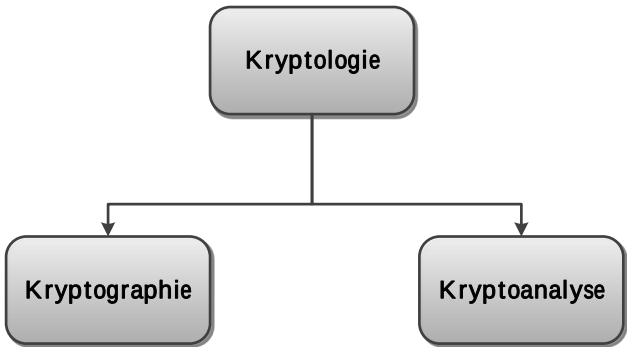


Abbildung 5: *Einteilung der Kryptologie, adaptiert von [1, S. 3]*

Häufig wird der Begriff Kryptographie als Synonym für Kryptologie verwendet [15, S. 1].

2.2.1 Kryptographie

Die Kryptographie lässt sich in die Bereiche klassische Kryptographie und moderne Kryptographie einteilen. Nach [15, S. 1], [16, S. 31] können alle bis 1950 entwickelten Verfahren zur klassischen Kryptographie gezählt werden. Der erste Schritt, der zum Übergang von der klassischen zur modernen Kryptographie geführt hat, war die Veröffentlichung einer Arbeit des amerikanischen Mathematikers Claude Elwood Shannon, der den Begriff der

informationstheoretischen Sicherheit prägte [15, S. 1]. In dieser Arbeit wird hauptsächlich der Bereich der modernen Kryptographie betrachtet.

Kryptographische Schutzziele

Es gibt verschiedene kryptographische Methoden, mit denen Informationen geschützt werden können. Diese dienen verschiedenen Schutzzielen. Nach [1, S. 301] sind die folgenden Schutzziele die wichtigsten beim Versand von Informationen:

- **Vertraulichkeit:** Informationen sind nur bestimmten Personen zugänglich.
- **Integrität:** Informationen wurden während des Versandes nicht verändert bzw. die Veränderung kann bemerkt werden.
- **Authentisierung:** Der Absender ist authentisch.
- **Nichtzurückweisbarkeit:** Der Absender kann den Versand nicht abstreiten.

Integrität, Authentisierung und Nichtzurückweisbarkeit können mit digitalen Signaturen und kryptographischen Prüfsummen realisiert werden [1, S. 302]. Vertraulichkeit von Informationen kann durch Anwendung von kryptographischen Verschlüsselungsverfahren erreicht werden [1, S. 302]. Verschlüsselungsverfahren werden häufig auch Verschlüsselungsalgorithmus oder Chiffre genannt. Eine Chiffre wird häufig in einem Protokoll als Methode zur Sicherstellung von Vertraulichkeit verwendet. Zu einem solchen Protokoll gehören häufig weitere Spezifikationen wie beispielsweise Blockgrößen, Betriebsmodi und Initialisierungsvektoren. Bei der Verschlüsselung werden Informationen durch eine Transformation so verändert, dass sich diese nur durch Kenntnis eines Geheimnisses zurück in den originalen Zustand transformieren lassen [17, Kapitel 5, S. 2]. Das Geheimnis kann grundsätzlich in zwei verschiedene Arten kategorisiert werden:

1. Die Chiffre selbst kann geheim gehalten werden, sodass diese nur einem bestimmten Personenkreis zur Verfügung steht. Dieses Prinzip wird *Sicherheit durch Verschleierung* („security by obscurity“) genannt [1, S. 12].
2. Die Chiffre verwendet einen Parameter, den so genannten *Schlüssel*, der als Geheimnis verwendet wird. Die Sicherheit beruht allein auf dem Schlüssel. Die Bestandteile der Chiffre und deren Funktionsweise sind

dabei öffentlich. Dieses Prinzip geht auf den niederländischen Kryptographen Auguste Kerckhoffs zurück und wird *Kerckhoffs'sches Prinzip* genannt [1, S. 12].

Die Vergangenheit hat gezeigt, dass Chiffren, deren Sicherheit auf der Geheimhaltung der Chiffre selbst basiert, häufig Schwächen aufweisen. Ein bekanntes Beispiel sind die „Mifare-Chipkarten“, die zum Bezahlen im öffentlichen Nahverkehr verwendet wurden [1, S. 12]. Diese nutzten ein geheimes Verschlüsselungsverfahren, welches durch Reverse-Engineering der Hardware und aufgrund von Schwachstellen leicht gebrochen werden konnte [1, S. 12]. Folglich sollten die Funktionsweise und Konstruktion einer Chiffre bekannt sein, sodass die Sicherheit öffentlich diskutiert und geprüft werden kann. Im Idealfall lässt sich beweisen, dass eine Chiffre ein bestimmtes Sicherheitsniveau aufweist – dies ist allerdings in noch nicht vielen Fällen gelungen [18, S. X].

Kryptographische Chiffren

Daten werden in Form von Nachrichten zwischen Sender und Empfänger ausgetauscht. Die unverschlüsselten Daten werden als *Klartext* bezeichnet. Ein Klartext m besteht aus Zeichen eines *Alphabets* Σ. Die Menge aller Klartexte über Σ wird mit M bezeichnet. Analog werden die verschlüsselten Daten als Geheimtext oder Chiffrat bezeichnet.
Moderne Chiffren arbeiten mit denselben Alphabeten für Klartexte und Geheimtexte. Diese Alphabete bestehen aus Bits oder Bytes. Ein Bit kann die zwei Zustände 0 oder 1 annehmen. Folglich wird ein Klartext $m \in M$ als eine Folge von Bits $b_i \in \{0, 1\}, i \in \mathbb{N}$ definiert. Eine Folge $(b_1, b_2, ..., b_n)$, die aus $n \in \mathbb{N}$ Bits besteht, wird als n-Bit Block bezeichnet. Ein Schlüssel $k \in K$ wird ebenfalls als Folge von Bits festgelegt. Eine Chiffre verschlüsselt einen Klartext m mit einem Schlüssel k_e zu einem *Geheimtext*. Die Verschlüsselung wird durch eine Abbildung $ENC_{k_e}(m)$ definiert. Die Menge aller Geheimtexte über dem Alphabet Σ wird mit C bezeichnet. Der Geheimtext $c \in C$ ist ebenfalls eine Folge von Bits. Die Entschlüsselung des Geheimtextes c mit einem Schlüssel k_d wird durch die Abbildung $DEC_{k_d}(c)$ beschrieben.

Das Prinzip der Verschlüsselung durch eine Chiffre wird durch Abbildung 6 beschrieben. Der Sender verschlüsselt (transformiert) einen Klartext m mit einem geheimen Schlüssel k_e. Der durch die Transformation entstehende Geheimtext c kann vom Empfänger durch einen geheimen Schlüssel k_d wieder in den Klartext entschlüsselt (transformiert) werden.

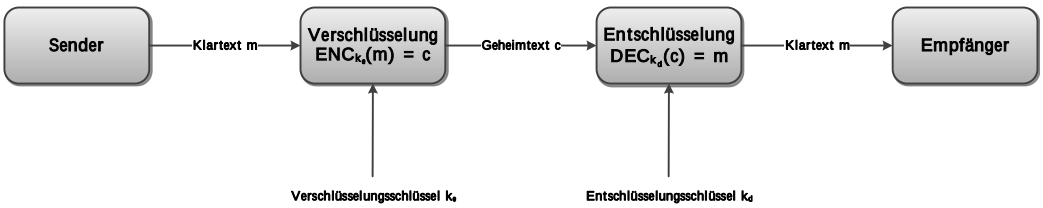


Abbildung 6: *Prinzip einer Chiffre, adaptiert von [17, Kapitel 5, S. 4]*

Innerhalb der Kryptographie werden zwei Klassen von Chiffren unterschieden (Abb. 7). Asymmetrische Chiffren verwenden zwei verschiedene Schlüssel zum Verschlüsseln (k_e in Abb. 6) und Entschlüsseln (k_d in Abb. 6) – es gilt also $k_e \neq k_d$. Eine der meistgebrauchten asymmetrischen Chiffren ist das RSA-Verfahren [1, S. 199]. Symmetrische Chiffren hingegen verwenden zum Ver- und Entschlüsseln denselben Schlüssel – hier gilt $k_e = k_d$. Bei symmetrischen Chiffren wird der Schlüssel daher mit k bezeichnet.

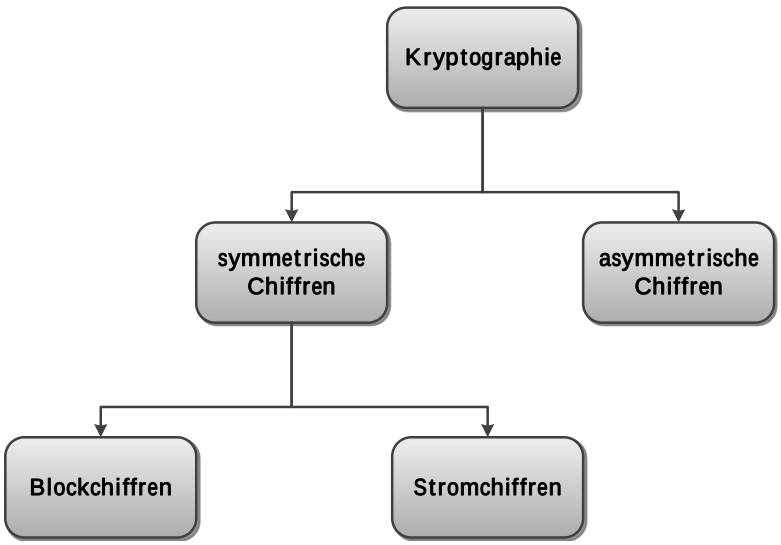


Abbildung 7: *Einteilung der Kryptographie, adaptiert von [1, S. 3]*

Bei den symmetrischen Chiffren wird zwischen Block- und Stromchiffren unterschieden. Stromchiffren verschlüsseln jedes einzelne Bit des Klartexts mit einem Schlüsselbit. Dazu wird ein Schlüsselstrom durch die Chiffre erzeugt [1, S. 33]. Bei Blockchiffren werden die zu verschlüsselnden Daten in Blöcke fester Länge eingeteilt und nacheinander mit demselben Schlüssel verschlüsselt [1, S. 34]. Gängige Blocklängen sind beispielsweise 64 Bit bei DES [19] oder 128 Bit bei AES [20]. Eingabe- und Ausgabeblock haben dieselbe Länge.

Ist ein Eingabeblock zu kurz, wird dieser durch ein *Paddingverfahren* auf die entsprechende Länge aufgefüllt. Nach [21, S. 1] sind Blockchiffren elementar für die heutige Kryptographie und das am meisten verwendete kryptographische Primitiv. Kryptographische Primitive sind die Bausteine, aus denen komplexere kryptographische Verfahren konstruiert sind. In Kapitel 2.3 wird der Aufbau von Blockchiffren weiter untersucht.

Betriebsmodi von Blockchiffren

Bei den Blockchiffren wird dabei zwischen verschiedenen *Betriebsarten* („Modes of Operation“) unterschieden. Im internationalen Standard ISO 10116 existieren fünf verschiedene Betriebsarten für Blockchiffren [17, Kapitel 8, S. 1]. Diese beeinflussen die Verarbeitungsweise der Blöcke durch die Chiffre – so können aufeinanderfolgende Blöcke miteinander verkettet und in Abhängigkeit gestellt werden. Zwei Betriebsarten sind im Folgenden beschrieben:

- **Electronic Code Book:** Im Electronic Code Book (ECB) Modus werden die Blöcke der Daten jeweils unabhängig voneinander verschlüsselt. Diese Betriebsart ist die einfachste und natürlichste Art, eine Blockchiffre zu nutzen [21, S. 66]. Die Ver- und Entschlüsselung lässt sich wie in Gleichung 1 formalisieren.

$$c_i = ENC_k(m_i) \text{ und } m_i = DEC_k(c_i) \text{ für } 1 \leq i \leq n \tag{1}$$

  Zwei gleiche Blöcke ergeben unter Nutzung desselben Schlüssels den gleichen Schlüsseltext – das ist insbesondere für lange Nachrichten problematisch, wo Redundanzen auftreten können [21, S. 66]. Es besteht das Risiko des Informationslecks [21, S. 66]. Da die Blöcke unabhängig voneinander ver- und entschlüsselt werden, können die Prozesse parallelisiert werden, um das Verfahren zu beschleunigen [21, S. 66]. Abbildung 8 illustriert den Ver- und Entschlüsselungsprozess.

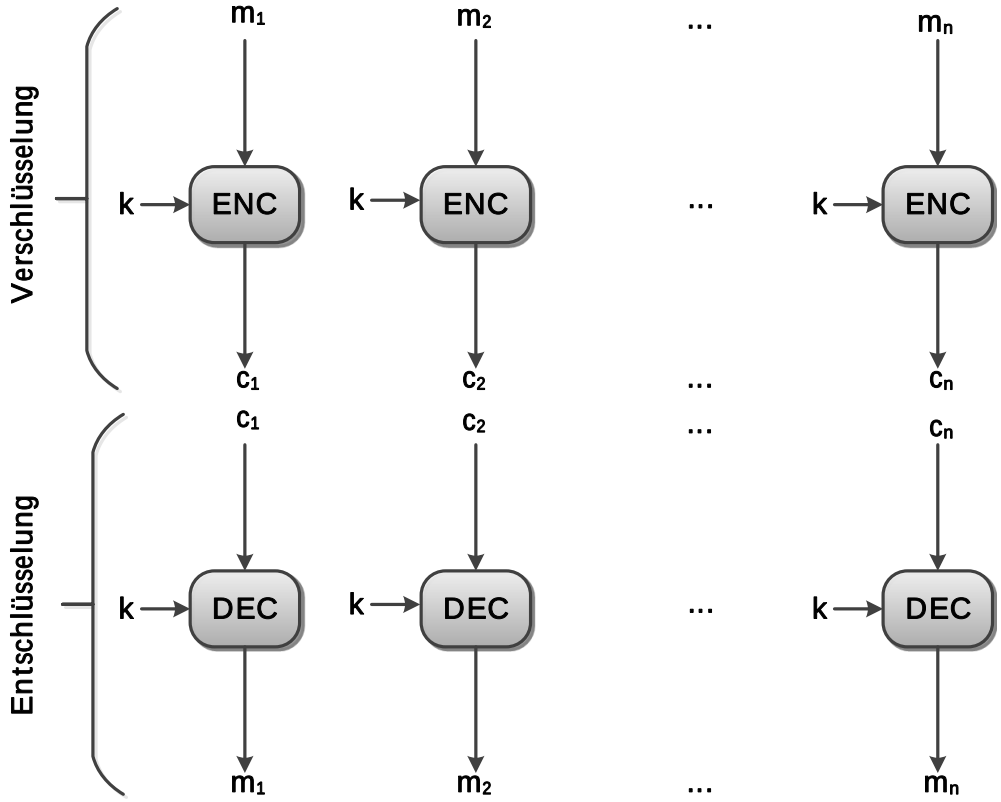


Abbildung 8: *ECB-Blockverschlüsselung, adaptiert von [21, S. 67]*

- **Cipher Block Chaining:** Im Cipher Block Chaining (CBC) Betriebsmodus werden aufeinander folgende Blöcke jeweils in Abhängigkeit gebracht – diese werden miteinander verknüpft. Der Verschlüsselungsprozess wird in Gleichung 2 beschrieben. Für den ersten Klartext-Block m_1 steht zur Verkettung kein Vorgängerblock zur Verfügung, daher wird ein Initialisierungsvektor (IV) gewählt.

  $$c_1 = ENC_k(m_1 \oplus IV) \text{ und } c_i = ENC_k(m_i \oplus c_{i-1}) \text{ für } 2 \leq i \leq n \quad (2)$$

  Analog dazu wird die Entschlüsselung wie in Gleichung (3) durchgeführt.

  $$m_1 = DEC_k(c_1) \oplus IV \text{ und } m_i = DEC_k(c_i) \oplus c_{i-1} \text{ für } 2 \leq i \leq n \quad (3)$$

  Durch die Verkettung der Blöcke ist sichergestellt, dass identische Klartext-Blöcke normalerweise auf verschiedene Geheimtext-Blöcke abgebildet werden. Allerdings ist das Verfahren nicht sinnvoll parallelisierbar [21, S. 68]. Ist ein Block inkorrekt übertragen worden oder verloren

gegangen, kann die Nachricht nicht mehr entschlüsselt werden. Abbildung 9 illustriert den Ver- und Entschlüsselungsprozess.

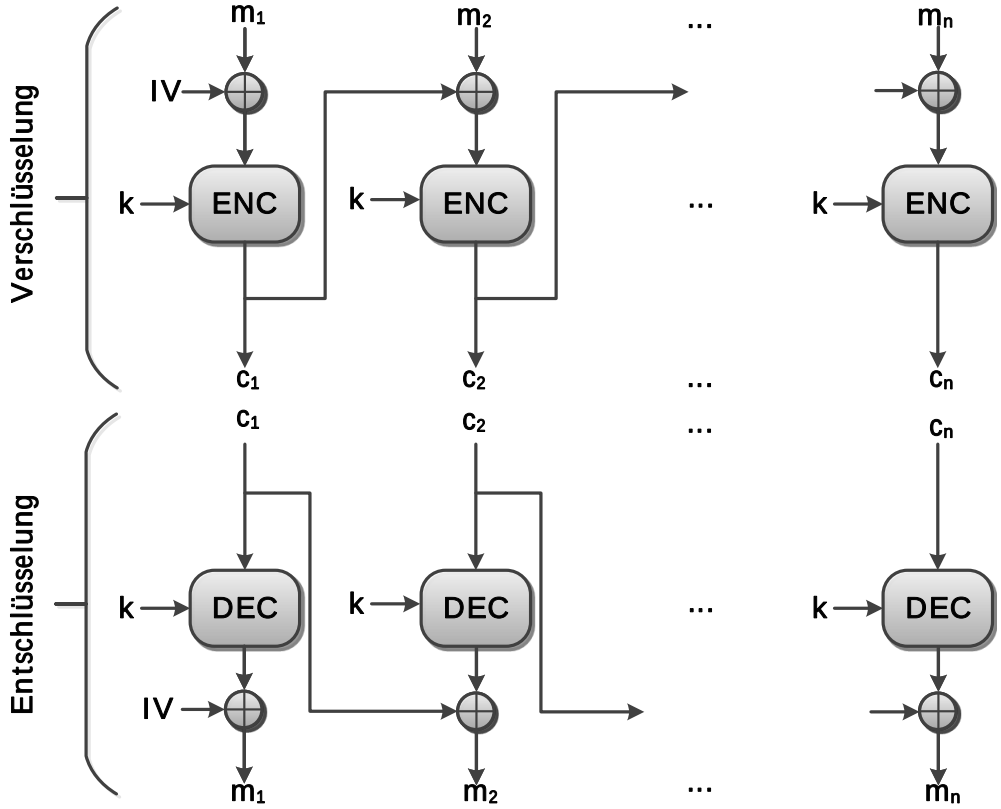


Abbildung 9: *CBC-Blockverschlüsselung, adaptiert von [21, S. 68]*

Kryptographische Prinzipien

Klassische kryptographische Verfahren arbeiten zeichenorientiert und nicht auf Bits [22, S. 11]. Dabei werden im Verschlüsselungsprozess Zeichen durch andere Zeichen ersetzt oder deren Position vertauscht [22, S. 11]. Methodisch werden dabei Substitution und Transposition der Zeichen eingesetzt [22, S. 11-13]. An dieser Stelle seien beispielhaft die Caesar- und Playfair-Chiffre als Vertreter der Substitutionschiffren, und die einfache Spaltentransposition als Vertreter der Transpositionschiffren genannt, während die ADFGVX-Chiffre beides kombiniert [22, S. 13-14] – all diese Chiffren sind schon in CT2 enthalten.
Allerdings hat die Geschichte gezeigt, dass der Verschlüsselungsprozess klassischer Verfahren häufig analysiert werden kann. Es bestehen daher oft Mög-

lichkeiten den Klartext ohne Kenntnis des Schlüssels zu rekonstruieren [23, S. 111]. Weitere Informationen zu diesem Thema sind in Kapitel 1 und 2 in [23] zu finden. Das Problem natürlicher Sprachen wie beispielsweise Deutsch oder Englisch ist, dass diese Redundanzen aufweisen und die Buchstabenhäufigkeiten sehr unterschiedlich sind [22, S. 274], [23, S. 10]. Die unterschiedlichen Häufigkeiten können mittels Häufigkeitsanalysen ausgenutzt werden, um einen Geheimtext zu entschlüsseln.

Für die Kryptoanalyse sind zwei Punkte wesentlich, die durch Shannon geprägt wurden [23, S. 111]. Dieser stellte in den 40er Jahren seine Arbeit [24] vor, welche die Grundlage der sogenannten Informations- und Kodierungstheorie bildet [23, S. 73]. Die folgenden Überlegungen gelten sowohl für klassische als auch für moderne Chiffren. Shannon beschreibt in seiner Arbeit die Sicherheit von Kryptosystemen durch ein präzises mathematisches Modell [22, S. 275]. Dabei werden zwei Prinzipien genannt, um Redundanzen von Klartext-Nachrichten zu verbergen – *Konfusion* und *Diffusion* [22, S. 277].

Durch Konfusion wird der Zusammenhang zwischen Klar- und Geheimtext verschleiert [22, S. 277]. Dies erschwert die Suche nach Redundanzen und statistischen Mustern bei der Analyse eines Geheimtexts. Die einfachste Möglichkeit für Konfusion sind Substitutionen. Bei modernen Verfahren kann beispielsweise ein ganzer Block durch einen anderen substituiert werden. Das Ergebnis ändert sich durch die Änderung eines einzelnen Bits oder des Schlüssels [22, S. 277]. Die Substitution allein ist allerdings nicht ausreichend für eine sichere Chiffre [22, S. 277].

Durch Diffusion werden Redundanzen des Klartexts über den Geheimtext verteilt [22, S. 278]. Folglich wird die Suche nach Redundanzen erschwert. Eine einfache Möglichkeit, um Diffusion zu erreichen, ist die Transposition oder Permutation [22, S. 278]. Moderne Verfahren mischen häufig alle Bitpositionen durch, sodass alle Zeichen durch Anwendung einer Permutation an einer anderen Stelle stehen.

Eine moderne und kurze Beschreibung der Konzepte von Konfusion und Diffusion von Massey ist in [21, S. 6] zu finden:

- **Konfusion:** Die Geheimtextstatistik sollte von der Klartextstatistik abhängig sein, und zwar in einer Weise, die zu kompliziert ist, um vom Kryptoanalytiker genutzt zu werden.
- **Diffusion:** Jede Stelle des Klartextes und jede Stelle des geheimen Schlüssels sollte viele Stellen des Chiffretextes beeinflussen.

Nach Shannon sollten Chiffren so konstruiert sein, dass diese einen hohen Grad an Konfusion und Diffusion aufweisen. Shannons Vorschlag zur Erreichung dieser Eigenschaften ist, dass zunächst eine endomorphe Substitutionschiffre als Ausgangspunkt verwendet wird [23, S. 111]. Endomorph sind alle Chiffren, deren Klartext- und Geheimtext-Blöcke aus der gleichen Menge stammen [23, S. 111]. Diese besitzen die Eigenschaft, dass sie mehrfach hintereinander ausgeführt werden können [23, S. 111]. Shannons Ziel ist es nun, Konfusion und Diffusion von Chiffren zu erhöhen, indem verschiedene Verschlüsselungsschritte mehrfach hintereinander ausgeführt werden, wobei sich Substitution und Permutation abwechseln sollen [23, S. 111].
Eine Chiffre, welche verschiedene Kombinationen aus Konfusion und Diffusion wiederholt anwendet, wird auch als *Produktchiffrierung* oder *Produktverschlüsselung* bezeichnet [22, S. 400]. Die Wiederholung von einzelnen Schritten im Ver- und Entschlüsselungsprozess einer Chiffre wird als Runde bezeichnet. Verwendet eine Chiffre in jeder Runde dieselben Schritte, wird eine solche Chiffre auch iterierte Chiffre genannt [21, S. 7].

Key Schedule

Für die einzelnen Verschlüsselungsrunden einer Chiffre werden Schlüssel benötigt. Der für die Chiffre spezifizierte Schlüssel wird häufig als *Schlüsselmaterial* bezeichnet. Nach [21, S. 7] ist es ein gutes Design-Prinzip von Chiffren, wenn innerhalb des Verschlüsselungsprozesses das spezifizierte Schlüsselmaterial so oft wie möglich verwendet wird. Ein Verfahren, welches eine Folge von so genannten Rundenschlüsseln auf Basis des Schlüsselmaterials generiert, wird als *Key Schedule* bezeichnet. Key Schedules können einfach, aber auch sehr komplex sein [21, S. 7]. Da die Sicherheit eines Verfahrens im Schlüssel liegt [22, S. 200], sollten gute Schlüssel Zufallsbitfolgen sein. Diese haben alle dieselbe Auftrittswahrscheinlichkeit [22, S. 203].

2.2.2 Kryptoanalyse

Die Kryptoanalyse beschäftigt sich mit der Analyse kryptographischer Verfahren und dem Brechen und Analysieren von geheimen Nachrichten [1, S. 2]. Darüber hinaus hilft Kryptoanalyse auch bei der Beurteilung der Sicherheit von kryptographischen Verfahren [1, S. 6]. Der Versuch einer Kryptoanalyse kann als Angriff bezeichnet werden [22, S. 6]. Personen, die Kryptoanalyse betreiben, werden Kryptoanalytiker oder Kryptoanalyst genannt [1, S. 2], [22, S. 5]. Das in Kapitel 2.2.1 vorgestellte Kerckhoffs'sche Prinzip spielt für die Kryptoanalyse eine besondere Rolle: Durch breite Diskussion eines Verfahrens in der Öffentlichkeit können leichter Sicherheitsprobleme gefun-

den werden. Die sichersten Chiffren sind aktuell die, die jahrelang öffentlich von Kryptoanalytikern analysiert und attackiert wurden [22, S. 8]. Hier sind beispielsweise RSA oder AES zu nennen.

Das Gebiet der Kryptoanalyse lässt sich in klassische und moderne Kryptoanalyse (Abb. 10) einteilen.

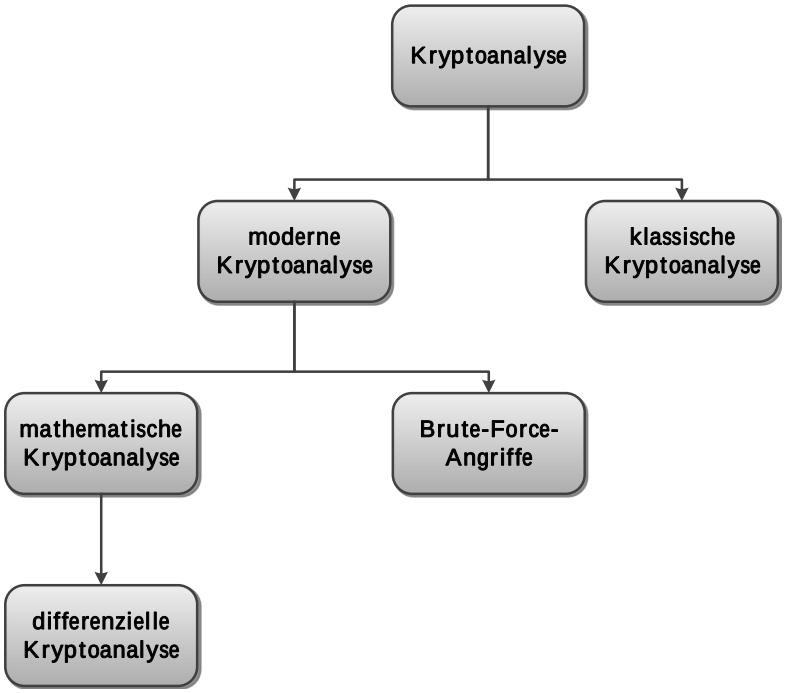


Abbildung 10: *Einteilung der Kryptoanalyse, adaptiert von [1, S. 11]*

Die klassische Kryptoanalyse beschäftigt sich mit klassischen kryptographischen Verfahren wie beispielsweise der antiken Skytale, dem Weltkrieg-1-Verfahren ADFGVX oder der Weltkrieg-2-Maschine Enigma. Die moderne Kryptoanalyse befasst sich mit der Analyse moderner Chiffren wie beispielsweise dem DES, AES oder RSA, welche auch Untersuchungsgegenstand dieser Arbeit ist. Dabei kann zwischen mathematischer Kryptoanalyse und Brute-Force-Angriffen unterschieden werden. Bei Brute-Force-Angriffen wird versucht, den Schlüssel durch systematisches Probieren aller Möglichkeiten zu ermitteln.

In der Kryptoanalyse werden verschiedene Angriffsmodelle unterschieden, welche die Rahmenbedingungen aus Sicht des Angreifers beschreiben. Einige ausgewählte Angriffsmodelle sind die folgenden [22, S. 6]:

- **Ciphertext-only attack:** Bei diesem Angriff verfügt der Angreifer über einen oder mehrere Geheimtexte und versucht die unbekannten Klartexte zu rekonstruieren oder den Schlüssel zu ermitteln, mit dem die Klartexte verschlüsselt wurden.
- **Known-plaintext attack:** Der Angreifer verfügt über mehrere Geheimtext-Paare und zugehörigem Klartext. Er versucht, den Schlüssel zu berechnen.
- **Chosen-plaintext attack:** Bei einem Chosen-plaintext-Angriff kann sich der Angreifer oder Kryptoanalytiker nahezu beliebig viele Paare von Klartext und zugehörigem Geheimtext erstelllen lassen. Oft kann er auch den zu verschlüsselnden Klartext selbst festlegen.

Darüber hinaus existieren noch weitere Angriffsmodelle. An dieser Stelle sei auf [17, Kapitel 5, S. 16] oder [22, S. 6] verwiesen. Beim Vergleich zwischen Chosen-plaintext- und Known-plaintext-Angriffen ist der Kryptoanalytiker beim Chosen-plaintext-Angriff in der besseren Situation, da er gezielt Klartext-Blöcke zur Verschlüsselung auswählen kann [22, S. 6]. Dagegen liegen beim Ciphertext-only-Angriff die wenigsten Informationen für den Kryptoanalytiker vor.

Beim Brechen einer Chiffre wird zwischen verschiedenen Kategorien der Vollständigkeit des Angriffserfolgs unterschieden [22, S. 9] :

- **Vollständiges Aufbrechen:** Beim vollständigen Aufbrechen erhält der Kryptoanalytiker den kompletten Schlüssel k, um Nachrichten zu entschlüsseln.
- **Globale Deduktion:** Der Kryptoanalytiker findet ohne Kenntnis des Schlüssels k ein alternatives Verfahren, welches äquivalent zur Ver- oder Entschlüsselungsfunktion (ENC_k bzw. DEC_k) ist.
- **Punktuelle (oder lokale) Deduktion:** Bei diesem Angriff kann der Kryptoanalytiker zu einem Geheimtext den passenden Klartext erhalten, ohne den Schlüssel zu ermitteln.
- **Informationsdeduktion:** Bei der Informationsdeduktion gelangt der Kryptoanalytiker an Informationen über den Schlüssel oder Klartext. Diese können aus Bits des Schlüssels oder Hinweisen zum Format des Klartextes o. ä. bestehen.

Nach [22, S. 9] gilt ein Algorithmus als *uneingeschränkt sicher*, wenn der Klartext auch dann nicht ermittelt werden kann, wenn Geheimtext in beliebigem Umfang vorhanden ist. Grundsätzlich lassen sich alle Chiffren (mit Ausnahme des One-Time-Pad) bei unbegrenzten Ressourcen brechen [22, S. 8]. Dies kann durch einen Ciphertext-only-Angriff und systematisches Durchprobieren aller möglichen Schlüssel erreicht werden – dieses Vorgehen heißt *Brute-Force-Angriff* [22, S. 9]. Aus diesem Grund ist eine Klassifizierung von kryptographischen Chiffren als *berechnungssicher* eingeführt worden [22, S. 9]. Ein Verfahren gilt als berechnungssicher, wenn es mit derzeit und auch zukünftig vorhandenen Ressourcen nicht gebrochen werden kann [22, S. 9].

Zur Bewertung der notwendigen Ressourcen wird ein Angriff mit verschiedenen Metriken gemessen. Die wichtigsten Metriken nach [22, S. 9] sind:

- **Datenkomplexität:** Die Datenkomplexität beschreibt die notwendige Anzahl an Eingabedaten, die zur Durchführung eines Angriffs notwendig sind.
- **Berechnungskomplexität:** Die für einen Angriff notwendige Zeit wird durch die Berechnungskomplexität beschrieben.
- **Speicheranforderungen:** Die Speicheranforderungen beschreiben den notwendigen Speicherplatz für einen Angriff.

Die Komplexität eines Brute-Force-Angriffs ist maximal. Aus diesem Grund sind Kryptoanalytiker daran interessiert, bessere Angriffsmethoden zu finden. Dies geschieht gelegentlich durch mathematisch-analytische Verfahren. Eines dieser Verfahren ist die differenzielle Kryptoanalyse (zur Einordnung siehe Abb. 10), auf die in Kapitel 3 ausführlich eingegangen wird. Unter Kryptographen gilt eine Chiffre als *gebrochen* im strengen Sinne, wenn ein Angriffsverfahren gefunden wird, das besser ist als der Brute-Force-Angriff [25, S. 2].

2.3 Substitutions-Permutations-Chiffren (SPN)

In Kapitel 2.2.1 wurden die grundlegenden Prinzipien von Konfusion und Diffusion und deren Realisierung beschrieben. Moderne Blockchiffren wenden diese Prinzipien oft wiederholend auf einen Klartext-Block zur Verschlüsselung an und folgen damit dem Vorschlag von Shannon. Solche Chiffren werden Substitutions-Permutations-Netzwerke (SPN) genannt.
Häufig werden zur Erreichung von Konfusion und Diffusion Schlüsseladdition, Substitution und Permutation verwendet. Im Folgenden werden diese Operationen zunächst näher charakterisiert.

2.3.1 Schlüsseladdition

Moderne Blockchiffren arbeiten auf Blöcken von Bits. Eine Standardoperation zur Verknüpfung von Bitblöcken ist die *exklusiv-ODER-Operation*. Häufig wird diese auch als Bit-Addition, kurz XOR, bezeichnet [23, S. 116]. Das Symbol der XOR-Operation ist $\oplus$. Die XOR-Verknüpfung zweier Bits ist wie in Gleichung 4 festgelegt:

$$0 \oplus 0 = 0,\ 0 \oplus 1 = 1,\ 1 \oplus 0 = 1,\ 1 \oplus 1 = 0, \tag{4}$$

Analog ist die XOR-Operation für zwei n-Bit Blöcke $a = (a_1, a_2, ..., a_n)$ und $b = (b_1, b_2, ..., b_n)$ wie in Gleichung 5 festgelegt:

$$(a_1, a_2, ..., a_n) \oplus (b_1, b_2, ..., b_n) = (a_1 \oplus b_1, a_2 \oplus b_2, ..., a_n \oplus b_n) \tag{5}$$

2.3.2 Substitution

Allgemein ist eine Substitution eine Abbildung eines m-Bit Eingabeblocks auf einen n-Bit Ausgabeblock. Häufig erfolgen Substitutionen in einer Chiffre mittels so genannter Substitutions-Boxen (kurz: S-Boxen) [22, S. 320]. S-Boxen sind nicht-linear und gewährleisten den größten Anteil der Sicherheit einer Chiffre [22, S. 321]. Sie sind der wichtigste Bestandteil einer Chiffre – alle anderen sind linear und lassen sich deutlich leichter analysieren [22, S. 321].

Eine Abbildung f heißt nichtlinear bezüglich der zuvor eingeführten $\oplus$-Operation, falls für alle Blöcke m_1, m_2 die in Gleichung 6 beschriebene Bedingung gilt [1, S. 75].

$$f(m_1 \oplus m_2) \neq f(m_1) \oplus f(m_2) \tag{6}$$

Nach [22, S. 403] sind S-Boxen umso sicherer, je größer sie sind. Die Wahl der S-Boxen spielt eine große Rolle bei der Widerstandsfähigkeit einer Chiffre gegenüber Angriffen mit beispielsweise differenzieller Kryptoanalyse. Dieser Aspekt wird in Kapitel 3 näher betrachtet.

In dieser Arbeit werden im Folgenden quadratische $n \times n$ S-Boxen betrachtet, kurz n-Bit S-Boxen. Sie ermöglichen eine bijektive Abbildung, bei der n Bits eines Eingabeblock $(e_0, e_1, ..., e_{n-1})$ auf n Bits eines Ausgabeblocks $(a_0, a_1, ..., a_{n-1})$ abgebildet werden. Die Umsetzung einer Substitutionsabbildung wird häufig durch eine Tabelle mit zwei Spalten und 2^n Zeilen beschrieben. Die eine Spalte enthält alle Werte von e, die andere Spalte alle Werte von a. Eine solche Tabelle legt also für jeden auftretenden Eingabeblock einen

Ausgabeblock fest. Tabelle 1 zeigt beispielhaft eine Substitutionstabelle für eine 4-Bit S-Box.

Eingabebits	**Ausgabebits**
0000	0110
0001	0100
0010	1100
0011	0101
0100	0000
0101	0111
0110	0010
0111	1110
1000	0001
1001	1111
1010	0011
1011	1101
1100	1000
1101	1010
1110	1001
1111	1011

Tabelle 1: *Beispielhafte Darstellung einer Substitutionstabelle. Die Tabelle basiert auf der von [21, S. 118]. Sie wird auch in Chiffre 1, 3 und 4 verwendet.*

Eine wichtige Eigenschaft von S-Boxen ist der *Lawineneffekt*, dieser stellt die Frage, wie viele Ausgabebits einer S-Box sich ändern, wenn ein Teil der Eingabebits geändert wird [22, S. 404]. Der Lawineneffekt war beispielsweise ein wichtiges Kriterium beim Design der S-Boxen des DES [26, S. 66]. Ein weiterer Faktor ist die Implementierung der S-Boxen in Hardware. Nach [23, S. 120] entschied man sich bei der Konstruktion des DES für 4-Bit S-Boxen, da die Hardwareimplementierung für größere Substitutionen zu Anfang der 70er Jahre nicht mit angemessenem Aufwand herstellbar war. Im Folgenden wird die Anwendung einer S-Box auf einen passend großen Block mit $S()$ notiert.

2.3.3 Permutation

Permutationen ändern die Reihenfolge der Bits eines n-Bit großen Eingabeblocks $(e_0, e_1, ..., e_{n-1})$ gemäß einer festen Vorschrift in einen n-Bit großen Ausgabeblock $(a_0, a_1, ..., a_{n-1})$. Diese kann in Form einer $n \times 2$ Matrix angegeben werden. Zeile 1 beschreibt dabei die Position der Eingabebits und

Zeile 2 die Position der Ausgabebits [23, S. 117]. In Gleichung 7 ist eine beispielhafte Permutation P beschrieben.

$$P = \begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 \\ 0 & 4 & 8 & 12 & 1 & 5 & 9 & 13 & 2 & 6 & 10 & 14 & 3 & 7 & 11 & 15 \end{pmatrix} \quad (7)$$

Oft wird die erste Zeile auch weggelassen, so dass die Permutation als Vektor wie in Gleichung 8 beschrieben wird.

$$P = \begin{pmatrix} 0 & 4 & 8 & 12 & 1 & 5 & 9 & 13 & 2 & 6 & 10 & 14 & 3 & 7 & 11 & 15 \end{pmatrix} \quad (8)$$

Alternativ zur Matrix kann die Permutation auch als Vektor beschrieben werden, wobei auf die erste Zeile der Matrix verzichtet wird. Die Bitverschiebung ergebt sich dann aus Position i der Komponente a_i des Vektors $(a_0, a_1, ..., a_{n-1})$. Bitpermutationen lassen sich in elektronischen Schaltungen nach [23, S. 118] leicht realisieren. So könnte ein 16-Bit Eingabeblock von einem Bus mit 16 Kabeln repräsentiert werden, wobei das i-te Kabel Strom führt, wenn für das i-te Bit des Eingabeblocks gilt: $e_i = 1$. Die Permutationsvorschrift aus Gleichung 7 könnte beispielhaft wie in Abbildung 11 realisiert werden.

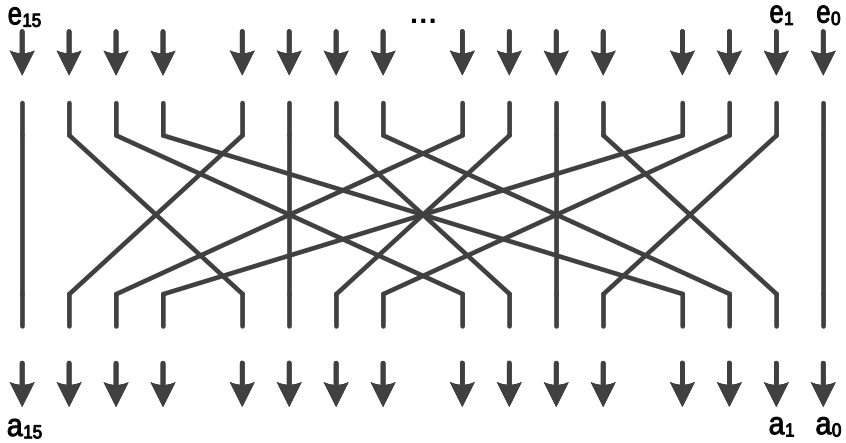


Abbildung 11: *Realisierung einer Permutation in Hardware, adaptiert von [21, S. 119]*

Durch Permutationen werden die Positionen der Bits des Eingabeblocks vertauscht, während bei Substitutionen der Inhalt des Eingabeblocks ersetzt

wird. Im Folgenden wird die Anwendung einer Permutation auf einen passend großen Block mit $P()$ bezeichnet.

2.3.4 Entwurf einer Blockchiffre

Grundsätzlich ist der Entwurf einer Blockchiffre aus den zuvor beschriebenen Elementen sehr einfach. S-Boxen fügen einer Chiffre Konfusion hinzu, Permutationen sorgen für Diffusion. Diese Elemente erschweren statistische Analysen. Die Sicherheit einer Chiffre basiert auf dem Schlüssel [22, S. 4], der durch den Key Schedule und Schlüsseladdition auf einen Block angewendet wird. Schwierig wird es allerdings, wenn an die Blockchiffre weitere Anforderungen wie bestimmte Sicherheitsanfoderungen, leichte Nachvollziehbarkeit und Implementierbarkeit hinzukommen [22, S. 405]. Darüber hinaus werden Chiffren häufig in Umgebungen verwendet, wo Ressourcen wie Rechen- und Speicherkapazität begrenzt sind. Daher ist nach [22, S. 406] die größte Herausforderung bei der Konstruktion von Blockchiffren in der Realität, das Entwerfen von Chiffren, die den kürzestmöglichen Schlüssel, die geringsten Speicheranforderungen und die geringste Laufzeit haben.

Substitutions-Permutations-Netzwerke sind Blockchiffren, die einen Klartext-Block in mehreren Stufen bzw. Runden verschlüsseln [22, S. 400]. Die einzelnen Runden zur Verschlüsselung eines Klartext-Blocks bestehen aus Schlüsseladdition, Substitution und Permutation. Bei vielen Blockchiffren wird der Substitutions-Schritt durch mehrere parallele S-Boxen realisiert. Dabei wird der Block in Teilblöcke aufgeteilt, jeder Teilblock wird für sich substituiert, und anschließend werden die substituierten Teilblöcke wieder konkateniert. Der primäre Grund für diese Arbeitsweise liegt im Bereich der Hardware-Realisierung. Abbildung 12 veranschaulicht den Aufbau eines SPN.

Die Ausgabe einer Verschlüsselungsrunde dient als Eingabe für die nächste Runde. In der letzten Runde muss als Abschluss eine Schlüsseladdition erfolgen, ansonsten würden die Substitution und Permuation der letzten Runde keine weitere Sicherheit hinzufügen [4, S. 205]. Der Grund dafür ist, dass die Konstruktion und Funktionsweise der einzelnen Elemente wie S-Boxen und Permutationen nach dem in Kapitel 2.2.1 beschriebenen Kerckhoffs'schen Prinzip öffentlich sind und damit auch natürlich invertiert werden können, wenn diese unabhängig vom Schlüssel verwendet werden. Für eine Chiffre mit r Runden werden folglich $r + 1$ Rundenschlüssel benötigt. Ein weiterer Bestandteil ist der Key Schedule. Dieser generiert die Schlüssel für die einzelnen Runden, die zur Verschlüsselung verwendet werden.

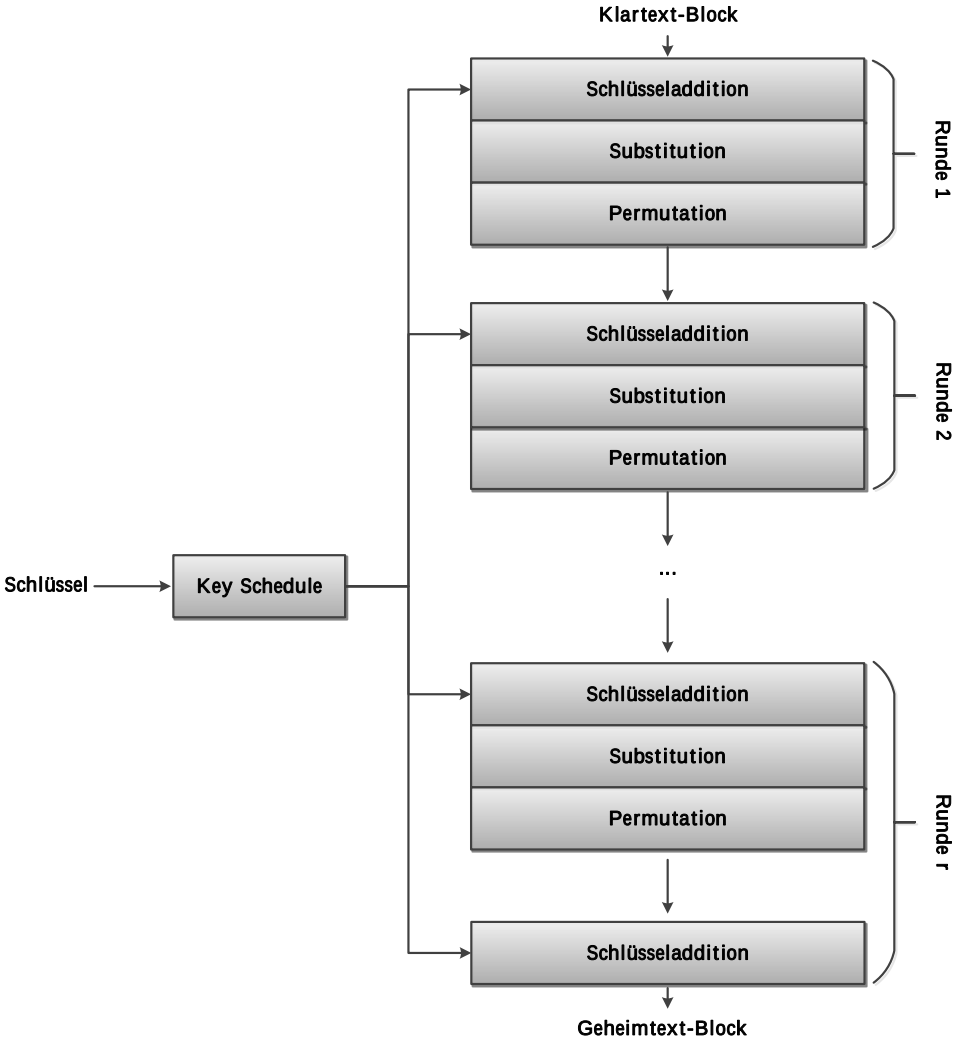


Abbildung 12: *Aufbau eines Substitutions-Permutations-Netzwerks (SPN), adaptiert von [21, S. 6]*

2.4 Verwendete Chiffren

In Kapitel 2.3 ist der allgemeine Aufbau von SPN erklärt worden. Im folgenden Kapitel werden die in dieser Arbeit verwendeten SPN beschrieben, an denen eine differenzielle Kryptoanalyse in CT2 durchführbar sein soll. Die Chiffren sind allesamt als Toy-Chiffren zu bezeichnen und sollten nicht zur Verschlüsselung von Daten außerhalb von Testzwecken verwendet werden.

2.4.1 Chiffre 1

Chiffre 1 arbeitet mit einer Blockbreite von 16 Bit und besteht insgesamt aus einer Verschlüsselungsrunde. Es wird ein Eingabeblock $e = (e_0, e_1, ..., e_{15})$ in einen Ausgabeblock $a = (a_0, a_1, ..., a_{15})$ transformiert. Die Verschlüsselung setzt sich aus Schlüsseladdition, Substitution und Schlüsseladdition zusammen. Der Schlüssel ist 32 Bit lang und wird durch einen einfachen Key-Schedule auf die zwei Schlüsseladditionen aufgeteilt. Die Aufteilung erfolgt in der Mitte des Schlüssels, sodass die jeweiligen Rundenschlüssel genau 16 Bit lang sind. Gleichung 9 beschreibt den Schlüssel k für Chiffre 1.

$$k = k_0 \,||\, k_1 \tag{9}$$

Die Substitution erfolgt durch vier parallele S-Boxen, die jeweils 4 Bit substituieren. Alle vier S-Boxen sind baugleich und substituieren auf dieselbe Weise einen 4-Bit-Block. Tabelle 1 illustriert die Funktionsweise dieser S-Box. Algorithmus 1 beschreibt den Verschlüsselungsprozess eines Klartext-Blocks durch Chiffre 1.

Eingabe: e, k
Ausgabe: a
Berechne $a = e \oplus k_0$;
Teile a in 4 Teilblöcke b_j zu je 4 Bit, sodass $a = b_0 \,||\, b_1 \,||\, b_2 \,||\, b_3$
Berechne $a = S(b_0) \,||\, S(b_1) \,||\, S(b_2) \,||\, S(b_3)$
Berechne $a = a \oplus k_1$

Algorithmus 1: *Verschlüsselung eines Klartext-Blocks mit Chiffre 1*

Die Chiffre basiert auf der in [21, S. 110] beschriebenen „CIPHERONE“. Die Konstruktion von Chiffre 1 ist in Abbildung 13 dargestellt. In Abbildungen, in denen Chiffren zu sehen sind, werden S-Boxen aus Platzgründen durch „SBox“ abgekürzt.

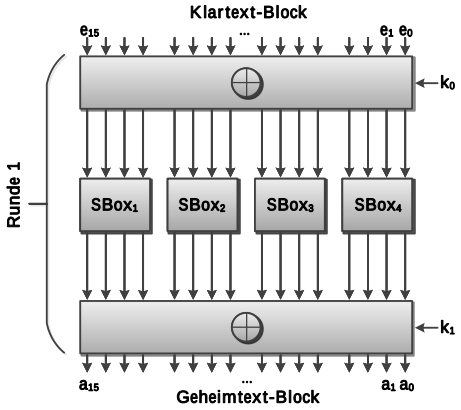


Abbildung 13: *Konstruktion Chiffre 1*

2.4.2 Chiffre 2

Chiffre 2 arbeitet mit einer Blockbreite von 16 Bit und besteht aus drei Verschlüsselungsrunden. Dabei wird ein Eingabeblock $e = (e_0, e_1, ..., e_{15})$ in einen Ausgabeblock $a = (a_0, a_1, ..., a_{15})$ überführt. Eine Verschlüsselungsrunde besteht aus Schlüsseladdition, Substitution und Permutation. In der letzten Runde wird auf die Permutation verzichtet, sodass diese aus Schlüsseladdition, Substitution und einer finalen Schlüsseladdition besteht. Damit entspricht diese Chiffre der in Kapitel 2.3 vorgestellten Struktur von SPN. Der Schlüssel ist 64 Bit groß und wird durch den Key Schedule auf vier Rundenschlüssel zu je 16 Bit aufgeteilt. Gleichung 10 beschreibt den Schlüssel k für Chiffre 2.

$$k = k_0 \,||\, k_1 \,||\, k_2 \,||\, k_3 \tag{10}$$

Die Substitution erfolgt wie bei Chiffre 1 durch vier parallele S-Boxen, welche jeweils 4 Bit substituieren. Die S-Boxen sind baugleich und substituieren durch dieselbe Vorschrift. Tabelle 2 beschreibt die Substitutionsvorschrift.
Die verwendete Permutation P_2 von Chiffre 2 ist in Gleichung 11 zu sehen. Zeile 1 beschreibt dabei die Position eines Bits im Eingabeblock und Zeile 2 die neue Position des Bits im Ausgabeblock.

Eingabebits	Ausgabebits
0000	1010
0001	0000
0010	1001
0011	1110
0100	0110
0101	0011
0110	1111
0111	0101
1000	0001
1001	1101
1010	1100
1011	0111
1100	1011
1101	0100
1110	0010
1111	1000

Tabelle 2: *Darstellung der Substitutionstabelle für Chiffre 2. Die Tabelle basiert auf der von [27]*

$$P_2 = \begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 \\ 0 & 13 & 10 & 7 & 4 & 1 & 14 & 11 & 8 & 5 & 2 & 15 & 12 & 9 & 6 & 3 \end{pmatrix} \quad (11)$$

Algorithmus 2 beschreibt den Verschlüsselungsprozess eines Klartext-Blocks durch Chiffre 2.

Eingabe: e, k
Ausgabe: a
`/* Runden 1 bis 2 */`
for *$i = 0$ to 1* **do**
  Berechne $e = e \oplus k_i$;
  Teile e in 4 Teilblöcke b_j zu je 4 Bit, sodass $e = b_0 \,||\, b_1 \,||\, b_2 \,||\, b_3$
  Berechne $e = S(b_0) \,||\, S(b_1) \,||\, S(b_2) \,||\, S(b_3)$
  Permutiere jedes Bit e_k, sodass $e = P_2(e)$
end
`/* Letzte Runde */`
Berechne $a = e \oplus k_2$;
Teile a in 4 Teilblöcke b_j zu je 4 Bit, sodass $a = b_0 \,||\, b_1 \,||\, b_2 \,||\, b_3$
Berechne $a = S(b_0) \,||\, S(b_1) \,||\, S(b_2) \,||\, S(b_3)$
Berechne $a = a \oplus k_3$

Algorithmus 2: *Verschlüsselung eines Klartext-Blocks mit Chiffre 2*

Chiffre 2 basiert auf der „3-Round SPN“ von [27]. Abbildung 14 illustriert die Konstruktion von Chiffre 2.

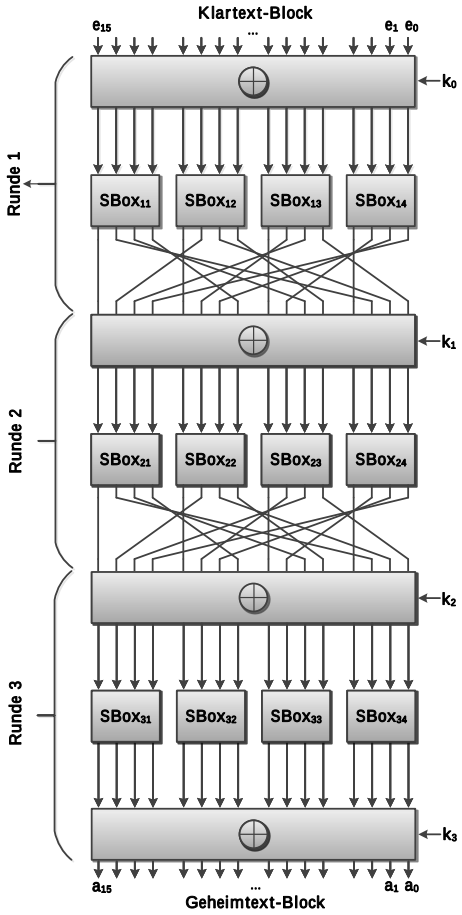


Abbildung 14: *Konstruktion Chiffre 2*

2.4.3 Chiffre 3

Chiffre 3 ist eine Blockchiffre, die 16 Bit große Blöcke verarbeitet. Es werden insgesamt fünf Verschlüsselungsrunden ausgeführt. Dabei wird ein Eingabeblock $e = (e_0, e_1, ..., e_{15})$ in einen Ausgabeblock $a = (a_0, a_1, ..., a_{15})$ überführt. Die Runden bestehen jeweils aus Schlüsseladdition, Substitution und Permutation. In der letzten Runde wird die Permutation nicht durchgeführt. Diese besteht aus Schlüsseladdition, Substitution und finaler Schlüsseladdition. Chiffre 3 entspricht damit ebenfalls der in Kapitel 2.3 vorgestellten SPN-Struktur. Der Schlüssel ist 96 Bit groß und wird durch den Key Schedule auf sechs Rundenschlüssel zu je 16 Bit aufgeteilt. In Gleichung 12 ist die Zusammensetzung des Schlüssels k von Chiffre 3 beschrieben.

$$k = k_0 \,||\, k_1 \,||\, k_2 \,||\, k_3 \,||\, k_4 \,||\, k_5 \tag{12}$$

Die Substitution erfolgt wie bei Chiffre 1 und Chiffre 2 durch vier parallele S-Boxen, die jeweils 4 Bit substituieren. Die S-Boxen in allen Runden sind identisch. Tabelle 1 beschreibt die Substitutionsvorschrift der S-Boxen von Chiffre 3. Die verwendete Permutation P von Chiffre 3 ist in Gleichung 7 zu sehen. Algorithmus 3 beschreibt den Verschlüsselungsprozess eines Klartext-Blocks durch Chiffre 3.

Eingabe: e, k
Ausgabe: a
`/* Runden 1 bis 4 */`
for *i = 0 to 3* **do**
  Berechne $e = e \oplus k_i$;
  Teile e in 4 Teilblöcke b_j zu je 4 Bit, sodass $e = b_0 \,||\, b_1 \,||\, b_2 \,||\, b_3$
  Berechne $e = S(b_0) \,||\, S(b_1) \,||\, S(b_2) \,||\, S(b_3)$
  Permutiere jedes Bit e_k, sodass $e = P(e)$
end
`/* Letzte Runde */`
Berechne $a = e \oplus k_4$;
Teile a in 4 Teilblöcke b_j zu je 4 Bit, sodass $a = b_0 \,||\, b_1 \,||\, b_2 \,||\, b_3$
Berechne $a = S(b_0) \,||\, S(b_1) \,||\, S(b_2) \,||\, S(b_3)$
Berechne $a = a \oplus k_5$

Algorithmus 3: *Verschlüsselung eines Klartext-Blocks mit Chiffre 3*

Die Chiffre basiert auf der in [21, S. 119] beschriebenen „CIPHERFOUR“. Die Konstruktion von Chiffre 3 ist in Abbildung 15 dargestellt.

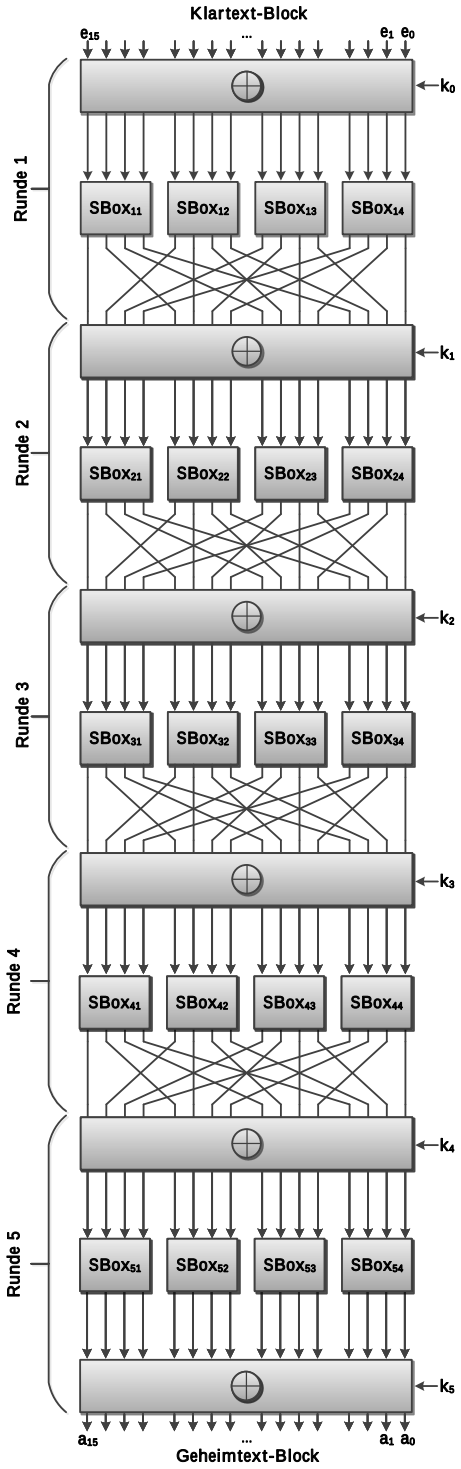


Abbildung 15: *Konstruktion Chiffre 3*

2.4.4 Chiffre 4

Chiffre 4 ist eine Blockchiffre, die auf 4 Bit großen Blöcken arbeitet. Es werden drei Verschlüsselungsrunden ausgeführt. Ein Eingabeblock $e = (e_0, e_1, e_2, e_3)$ wird dabei auf einen Ausgabeblock $a = (a_0, a_1, a_2, a_3)$ abgebildet. Die Runden bestehen bei Chiffre 4 jeweils aus Schlüsseladdition und Substitution. In der letzten Runde findet nach Schlüsseladdition und Substitution eine abschließende Schlüsseladdition statt. Der Schlüssel ist 16 Bit lang und wird durch den Key Schedule auf vier Rundenschlüssel aufgeteilt. In Gleichung 13 ist die Beschreibung des Schlüssels k beschrieben.

$$k = k_0 \,||\, k_1 \,||\, k_2 \,||\, k_3 \tag{13}$$

Die S-Boxen von Chiffre 4 sind alle identisch. Tabelle 1 zeigt die Abbildungsvorschrift der S-Boxen. Algorithmus 4 beschreibt den Verschlüsselungsprozess eines Klartext-Blocks durch Chiffre 4.

Eingabe: e, k
Ausgabe: a
`/* Runden 1 bis 2 */`
for *i = 0 to 1* **do**
  Berechne $e = e \oplus k_i$;
  Berechne $e = S(e)$
end
`/* Letzte Runde */`
Berechne $a = e \oplus k_2$;
Berechne $a = S(a)$
Berechne $a = a \oplus k_3$

Algorithmus 4: *Verschlüsselung eines Klartext-Blocks mit Chiffre 4*

Diese Chiffre basiert auf der in [21, S. 115] beschriebenen „CIPHERTHREE“. Die Konstruktion von Chiffre 4 ist in Abbildung 16 dargestellt.

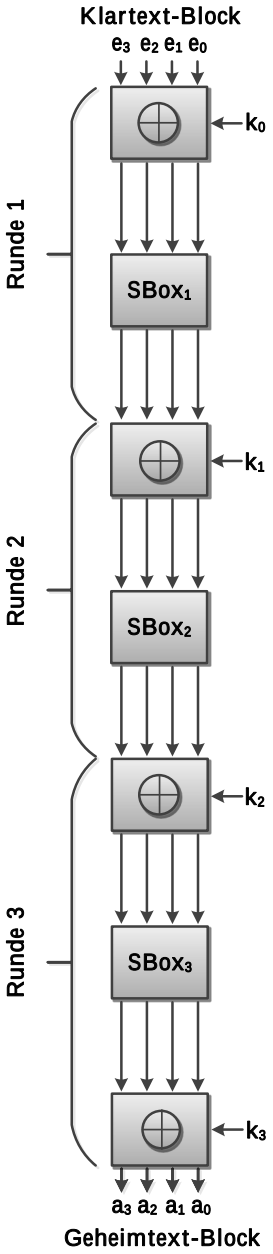


Abbildung 16: *Konstruktion Chiffre 4*

2.4.5 Übersicht der Chiffren

In diesem Abschnitt wird abschließend eine kompakte Übersicht über die vier zuvor beschriebenen Chiffren gegeben. In Tabelle 3 ist ein tabellarischer Vergleich der Chiffren zu sehen.

Name	Block-größe	Anzahl Runden	Schlüssel-länge	Anzahl der Schlüssel	Quelle der Chiffre
Chiffre 1	16 Bit	1	32 Bit	2^{32}	[21, S. 110]
Chiffre 2	16 Bit	3	64 Bit	2^{64}	[27]
Chiffre 3	16 Bit	5	96 Bit	2^{96}	[21, S. 119]
Chiffre 4	4 Bit	3	16 Bit	2^{16}	[21, S. 115]

Tabelle 3: *Vergleich der Chiffren 1 bis 4*

Die Chiffren 1, 2 und 3 arbeiten auf einer Blockgröße von 16 Bit. Chiffre 4 arbeitet auf 4 Bit großen Blöcken. Es werden verschiedene Anzahlen von Verschlüsselungsrunden durchgeführt. Daraus folgen unmittelbar auch die entsprechenden Schlüssellängen der Chiffren. Alle vorgestellten Chiffren verwenden Schlüsseladdition und Substitution. Chiffre 2 und 3 verwenden zusätzlich noch Permutationen. Die Chiffren unterscheiden sich durch die Substitutions- und Permutationsvorschriften. Diese sind in den Kapiteln 2.4.1 - 2.4.4 angegeben. Grundsätzlich ist die Struktur der Chiffren identisch, mit Ausnahme der Anzahl der Verschlüsselungsrunden.

Die Chiffren 1 - 3 sind so gewählt, dass man mit diesen ein in Schwierigkeits- und Komplexitätsgrad ansteigendes Tutorial aufbauen kann. Anhand von Chiffre 1 soll die grundlegende Idee der differenziellen Kryptoanalyse vermittelt werden. Aufgrund der übersichtlichen Konstruktion kann dieser Angriff auch leicht mit Stift und Papier durchgeführt und nachvollzogen werden. Darauf aufbauend wird die Idee erweitert und auf eine Chiffre mit drei Verschlüsselungsrunden übertragen, die aus den heute gebräuchlichen kryptographischen Primitiven für Chiffren besteht. An Chiffre 3 sollen dann als Letztes weitere Bestandteile einer differenziellen Kryptoanalyse vorgestellt werden, wobei diese Chiffre mit fünf Verschlüsselungsrunden und einem 96 Bit langen Schlüssel die komplexeste im Tutorial ist. Der genaue Aufbau des Tutorials wird in Kapitel 4.1 beschrieben. Bei der differenziellen Kryptoanalyse von Chiffre 4 treten Probleme auf, sodass der Angriff nicht in ausreichendem Maß erfolgreich durchgeführt werden kann. In Kapitel 6.2.1 wird dies näher diskutiert. Auf eine Integration von Chiffre 4 in die finale Implementierung in CT2 wird daher verzichtet.

3 Differenzielle Kryptoanalyse

In diesem Abschnitt wird die Angriffsart der differenziellen Kryptoanalyse (DKA) beschrieben. Zunächst wird die Idee des Angriffs erläutert. Darauf aufbauend werden die in Kapitel 2.3 vorgestellten kryptographischen Primitive der verwendeten SPN untersucht. Anschließend werden verschiedene Strukturen der differenziellen Kryptoanalyse sowie die praktische Durchführung des Angriffs beschrieben. Abschließend werden Widerstandsfähigkeit gegen DKA, der notwendige Aufwand und weitere Optimierungsmöglichkeiten diskutiert.

Die differenzielle Kryptoanalyse ist auch etwa 30 Jahre nach der Publizierung immer noch eine der besten Analysemethoden für Blockchiffren. Das öffentlich verfügbare Wissen über Kryptologie war bis nach dem 2. Weltkrieg gering, da es primär Domäne des Militärs war [23, S. 130]. In den 1960er Jahren forschte das Unternehmen IBM im Bereich kryptographischer Techniken und als Resultat entstand der Vorgänger des auch heute noch verwendeten DES [23, S. 120]. Dieser wurde als Einreichung im Rahmen einer Ausschreibung des *National Bureau of Standards* (NBS) aus damals unbekannten Motiven durch die *National Security Agency* (NSA) modifiziert und am 15.01.1977 als DES in der *Federal Information Processing Standard* (FIPS) Publikation 46 veröffentlicht [23, S. 120-130]. Die Geschichte des DES ist detailliert in [23, S. 130-133] und [22, S. 309-314] beschrieben.

1990 präsentierten die israelischen Forscher Eli Biham und Adi Shamir in ihrer Arbeit [3] den Angriff der differenziellen Kryptoanalyse der Öffentlichkeit. Ihre Arbeit bezieht sich primär auf den DES, allerdings ist das Verfahren allgemein für iterierte Blockchiffren und Hash-Funktionen verwendbar [28, S. 6]. Die DKA ist beim DES zwar nicht wesentlich effizienter als ein Brute-Force-Angriff, allerdings konnten aufschlussreiche Resultate über innere Strukturen von Blockchiffren gewonnen werden – weshalb das Verfahren auch heute noch als eines der mächtigsten gilt [23, S. 149 ff.]. Der Grund für die hohe Resistenz des DES gegenüber DKA liegt im Design der Chiffre. Don Coppersmith, ein an der Entwicklung des DES beteiligter IBM-Mitarbeiter, schrieb 1994 in einem Statement, dass die NSA gemeinsam mit IBM entschieden hatte, die Entwurfsziele nicht zu veröffentlichen, da diese die Technik der differenziellen Kryptoanalyse offenlegen und so den Vorsprung der Vereinigten Staaten auf dem Gebiet der Kryptographie gegenüber anderen Ländern gefährden würden [23, S. 150-151].

DKA ist eine Methode zur Analyse einer Blockchiffre. Ziel ist es, iterativ die Rundenschlüssel einer Chiffre wiederherzustellen. Die Ergebnisse hängen

allerdings unter anderem vom Wert des Schlüssels, der Klartexte und der Anzahl der Klartexte ab [29, S. 2]. DKA ist ein Chosen-Plaintext-Angriff, dessen Hauptelemente Paare von Geheimtexten sind, deren Klartexte sich in einer bestimmten Differenz unterscheiden [28, S. 6]. Zur Bildung der Differenz Δd von Blöcken m_1, m_2 eignet sich der XOR-Operator besonders gut, da Zusammenhänge zwischen Permutation und Schlüsseladdition leicht zu beschreiben sind [23, S. 154]. Grundsätzlich kann eine Differenz zweier n-bit-Blöcke auch anders definiert sein, beispielsweise durch eine Addition modulo 2^n oder das logische „und“ [23, S. 154]. Gleichung 14 zeigt die Bildung der Differenz.

$$\Delta d = m_1 \oplus m_2 \tag{14}$$

Die Klartexte können zufällig gewählt werden, der konkrete Wert ist nicht relevant. Bei der Analyse wird der Effekt bestimmter Klartextdifferenzen auf den zugehörigen Geheimtext untersucht [28, S. 11]. Dabei wird versucht zu analysieren, welche Schlüsselbits in den S-Boxen verarbeitet werden [28, S. 21]. Auf Basis der Differenzen können Wahrscheinlichkeiten zu möglichen Rundenschlüsseln berechnet werden, um den wahrscheinlichsten Rundenschlüssel zu bestimmen [28, S. 11].

Biham und Shamir teilen die Methode in zwei Phasen ein [28, S. 7]:

1. **Collection Phase:** Dabei werden Paare von Klartexten, die sich in einer festgelegten Differenz unterscheiden, mit dem unbekannten Schlüssel verschlüsselt. Die resultierenden Geheimtexte stehen dem Angreifer dann zur Verfügung.

2. **Analyse Algorithm:** Die Geheimtexte werden mit dem Ziel analysiert, den korrekten Schlüssel zu finden.

Der Analyse-Teil verläuft aufgrund weniger und einfacherer Operationen schneller als die Collection Phase ab [28, S. 7].

3.1 Analyse der Komponenten einer Chiffre

Die meisten Blockchiffren bestehen aus Schlüsseladdition, Substitution und Permutation. Diese werden häufig über mehrere Runden wiederholend ausgeführt. In diesem Abschnitt werden die Auswirkungen der einzelnen Operationen betrachtet. Von Interesse ist die Differenz Δd zweier Blöcke m_1, m_2 vor einer Operation und wie diese sich durch Anwendung der Operation zu $\Delta d'$ verändert. Abbildung 17 beschreibt diesen Vorgang.

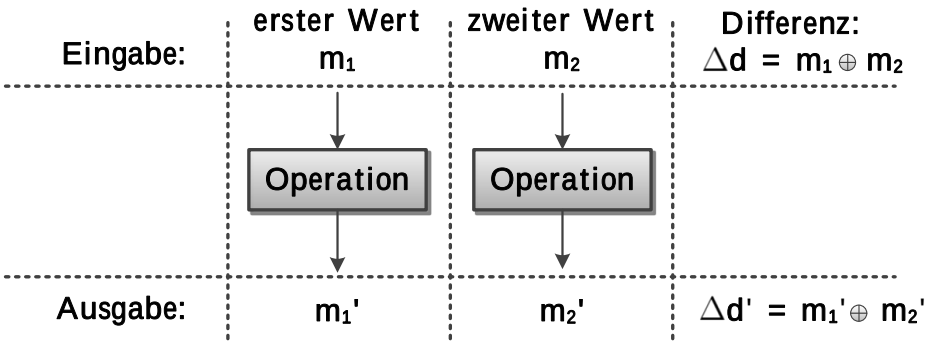


Abbildung 17: *Ein- und Ausgabedifferenzen bei Operationen adaptiert von [23, S. 155]*

3.1.1 Schlüsseladdition

Bei der Schlüsseladdition wird ein unbekannter Rundenschlüssel zusammen mit einem Klartext-Block mittels XOR-Operation verarbeitet. Die zentrale Beobachtung bei der Bildung von Differenzen bei der Verwendung von Schlüsseladdition ist, dass die Differenz von Blöcken durch den Schlüssel nicht verändert wird. An dieser Stelle seien m_1, m_2 n-Bit Blöcke und k ein n-Bit Rundenschlüssel. Wenn nun die Klartext-Blöcke mittels Schlüsseladdition mit k zusammen zu Geheimtext-Blöcken verarbeitet werden, gilt:

$$c_1 = m_1 \oplus k \text{ und } c_2 = m_2 \oplus k \tag{15}$$

Wird nun die Differenz $\Delta d'$ der Blöcke c_1 und c_2 berechnet, gilt:

$$\begin{aligned} \Delta d' &= c_1 \oplus c_2 \\ &= (m_1 \oplus k) \oplus (m_2 \oplus k) \\ &= (m_1 \oplus m_2) \oplus (k \oplus k) \\ &= m_1 \oplus m_2 = \Delta d \end{aligned} \tag{16}$$

Diese Beobachtung ist der wichtigste Punkt bei der differenziellen Kryptoanalyse nach [29, S. 2]. Sind zwei Blöcke m_1, m_2 mit Differenz Δd gegeben, bleibt diese bei Anwendung der Schlüsseladdition mittels unbekanntem Schlüssel k erhalten. Bei der Betrachtung der Paare wird versucht, die Veränderung der Differenzen durch die einzelnen Verschlüsselungsrunden einer Chiffre zu beobachten, um sukzessive Informationen über die einzelnen Rundenschlüssel zu erhalten.

3.1.2 Substitution

Der Zusammenhang von Eingabe- und Ausgabedifferenz bei S-Boxen ist der komplexeste Teil der Analyse, da sich diese i. d. R. nicht-linear verhalten [23, S. 155]. Die Anwendung einer S-Box auf einen Block m_1 wird mit $S(m_1)$ notiert. Die S-Boxen haben eine feste Bitbreite für Ein- und Ausgabe. Bei einer S-Box mit je n Ein- und Ausgabebits gibt es je 2^n verschiedene Ein- und Ausgabeblöcke. Aus 2^n verschiedenen Eingabeblöcken können $2^n \cdot 2^n = 2^{2n}$ Paare von Blöcken (m_1, m_2) erzeugt werden. Jede Eingabedifferenz Δd kann von 2^n verschiedenen Eingabepaaren (m_1, m_2) mit $m_1 \oplus m_2 = \Delta d$ erzeugt werden [23, S. 155]. Um das Verhalten der S-Box zu analysieren, werden für alle 2^n Paare je Eingabedifferenz Δd die Ausgabedifferenzen $\Delta d'$ berechnet [23, S. 155]. Für jede Eingabedifferenz Δd wird eine Tabelle mit $2^n + 3$ Zeilen und $2^n + 2$ Spalten erstellt. Jedes mögliche Paar erhält eine Zeile und enthält einen Punkt in den Spalten $\Delta d'$, um die zutreffende Ausgangsdifferenz zu markieren. Die letzte Zeile der Tabelle enthält eine Zusammenfassung, wie viele Eingabepaare zu der jeweiligen Ausgabedifferenz führen [23, S. 155]. Bei gegebenem m_1 gilt für $m_2 = m_1 \oplus \Delta d$. Diese Tabelle wird Differenzentabelle oder Differenzenverteilung genannt [23, S. 155].

An dieser Stelle wird die S-Box von Chiffre 3 aus Kapitel 2.4.3 mit dem zuvor beschriebenen Vorgehen analysiert. Es wird die Eingangsdifferenz $\Delta d = 0\text{x}3$ betrachtet. Die Darstellung der Zahlen ist hexadezimal. Tabelle 4 zeigt das Ergebnis der Analyse.

		$\Delta d' = S(m_1) \oplus S(m_2)$															
m_1	m_2	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	3				•												
1	2									•							
2	1									•							
3	0				•												
4	7															•	
5	6						•										
6	5						•										
7	4															•	
8	B													•			
9	A													•			
A	9													•			
B	8													•			
C	F				•												
D	E				•												
E	D				•												
F	C				•												
Differenzenverteilung:		0	0	0	6	0	2	0	0	2	0	0	0	4	0	2	0

Tabelle 4: *Darstellung der Differenzentabelle zur Eingangsdifferenz Δd = 0x3 der S-Box von Chiffre 3*

Zu jedem Eingabepaar (m_1, m_2) gibt es immer ein Eingabepaar (m_2, m_1), d. h. die Paare sind symmetrisch. Daraus folgt unmittelbar, dass es immer eine gerade Anzahl an Paaren gibt. Da S-Boxen (i. d. R.) deterministisch sind und die XOR-Operation kommutativ ist, gibt es nur gerade Häufigkeiten in der Differenzenverteilung [23, S. 156]. Bei der Betrachtung der letzten Zeile von Tabelle 4 fällt auf, dass die Differenzen nicht gleichverteilt sind. Es gibt Differenzen, die gar nicht auftreten, und andere, die häufiger auftreten als andere. So tritt die Ausgabedifferenz 0x3 beispielsweise sechsmal auf und die Ausgabedifferenz 0xE zweimal. Bei der Betrachtung aller Ausgabedifferenzen fällt auf, dass insgesamt fünf verschiedene Ausgabedifferenzen auftreten, obwohl 16 verschiedene Ausgabedifferenzen möglich sind.

Die Differenzentabelle muss zur Analyse des Verhaltens einer S-Box für jede mögliche Eingangsdifferenz Δd aufgestellt werden. Diese können zu der so genannten Differenzenverteilungstabelle zusammengefasst werden [23, S. 156]. Die Differenzenverteilungstabelle besteht im Wesentlichen aus der letzten Zeile der Differenzentabellen. Die Differenzenverteilungstabelle hat je Eingabedifferenz eine Zeile und je Ausgabedifferenz eine Spalte. Eine Zelle $(\Delta d, \Delta d')$ gibt die Anzahl an, wie oft ein Eingabepaar mit Differenz Δd durch ein Ausgabepaar mit Differenz $\Delta d'$ ersetzt wurde. Diese können dazu

verwendet werden, um die Wahrscheinlichkeit einer bestimmten Substitution zu berechnen. An dieser Stelle wird die Differenzenverteilungstabelle der S-Box von Chiffre 3 aus Kapitel 2.4.3 erstellt. Tabelle 5 zeigt die Differenzenverteilungstabelle. So ist beispielsweise die letzte Zeile von Tabelle 4 in der 4. Zeile von Tabelle 5 enthalten. Die Ein- und Ausgabedifferenzen (1. Spalte und 1. Zeile) der Tabelle sind hexadezimal, die Tabelleneinträge $(\Delta d, \Delta d')$ sind dezimal.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	16	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
1	-	-	6	-	-	-	-	2	-	2	-	-	2	-	4	-
2	-	6	6	-	-	-	-	-	-	2	2	-	-	-	-	-
3	-	-	-	6	-	2	-	-	2	-	-	-	4	-	2	-
4	-	-	-	2	-	2	4	-	-	2	2	2	-	-	2	-
5	-	2	2	-	4	-	-	4	2	-	-	2	-	-	-	-
6	-	-	2	-	4	-	-	2	2	-	2	2	2	-	-	-
7	-	-	-	-	-	4	4	-	2	2	2	2	-	-	-	-
8	-	-	-	-	-	2	-	2	4	-	-	4	-	2	-	2
9	-	2	-	-	-	2	2	2	-	4	2	-	-	-	-	2
A	-	-	-	-	2	2	-	-	-	4	4	-	2	2	-	-
B	-	-	-	2	2	-	2	2	2	-	-	4	-	-	2	-
C	-	4	-	2	-	2	-	-	2	-	-	-	-	-	6	-
D	-	-	-	-	-	-	2	2	-	-	-	-	6	2	-	4
E	-	2	-	4	2	-	-	-	-	-	2	-	-	-	-	6
F	-	-	-	-	2	-	2	-	-	-	-	-	-	10	-	2

Tabelle 5: *Darstellung der Differenzenverteilungstabelle der S-Box von Chiffre 3*

In der Differenzenverteilungstabelle der S-Box von Chiffre 3 fällt auf, dass die Differenzen nicht gleichverteilt sind. So gibt es beispielsweise kein Eingabepaar mit Differenz $\Delta d = 0x1$, das zu einem Ausgabepaar mit Differenz $\Delta d' = 0x1$ führt. Des Weiteren gibt es 10 Eingabepaare mit Differenz $\Delta d =$ 0xF die zu einem Ausgabepaar mit Differenz $\Delta d' =$ 0xD führen. Die textuelle Färbung dient zur leichteren Identifizierung der entsprechenden Einträge in Tabelle 5.

3.1.3 Permutation

Permutationen sind lineare Funktionen. Bei Bildung der Differenz Δd zweier n-Bit Klartext-Blöcke m_1, m_2 unter Verwendung einer Permutationen $P()$

gilt daher folgender Zusammenhang:

$$\Delta d' = P(m_1) \oplus P(m_2) = P(m_1 \oplus m_2) \tag{17}$$

Die Berechnung der Differenz von permutierten Blöcken hat demnach einen einfach nachzuvollziehenden Zusammenhang und verhält sich völlig deterministisch [29, S. 2]. Es spielt für die Differenzbildung keine Rolle, ob eine Permutation unabhängig voneinander auf zwei Blöcke m_1, m_2 angewendet wird oder ob die Permutation auf die Differenz der Blöcke angewendet wird. Für die Schlüsselwiederherstellung bedeutet dies, dass bei der Betrachtung der Differenzen von Blöcken die Permutation die Differenz der Blöcke nicht verändert, wenn diese beim Entschlüsseln eines einzelnen Blockes invertiert wird.

3.2 Charakteristik

Eine der zentralen Strukturen bei DKA sind *differenzielle Charakteristiken*. Eine differenzielle Charakteristik ist ein Paar von Differenzen (α, β). Es beschreibt, dass ein Paar von Blöcken (m_1, m_2) mit Differenz $\Delta d = \alpha$ unter der Operation $S()$ zu einem Paar von Blöcken mit Differenz $\Delta d' = \beta$ führt [21, S. 116]. Die Notation für den Übergang der Differenzen unter Anwendung der S-Box ist $\alpha \xrightarrow{S} \beta$ [21, S. 116]. Eine Charakteristik hat eine Wahrscheinlichkeit. Durch Erstellung der Differenzenverteilungstabelle können die Wahrscheinlichkeiten für Charakteristiken abgelesen werden. Beispielsweise kann Tabelle 5 entnommen werden, dass die Charakteristik 0x3 $\xrightarrow{S}$ 0x3 eine Wahrscheinlichkeit von $\frac{6}{16}$ hat. In der Literatur wird Charakteristik oft auch als *Rundendifferenzial* oder *Differenzenspur* bezeichnet [23, S. 164].

An dieser Stelle sei erwähnt, dass die Blöcke eines (Klartext-)Paares mit einer Differenz Δd einzeln und unabhängig voneinander durch eine Chiffre verschlüsselt werden. Die entstehenden Geheimtext-Blöcke unterscheiden sich lediglich mit einer bestimmten Wahrscheinlichkeit in einer Differenz $\Delta d'$.

3.2.1 Ein-Runden-Charakteristik

Das Konzept einer differenziellen Charakteristik soll im Folgenden auf eine Verschlüsselungsrunde einer SPN erweitert werden. Dazu wird Chiffre 3 aus Kapitel 2.4.3 betrachtet. Im vorigen Abschnitt wurde gezeigt, dass sich die Differenz durch Schlüsseladdition und Permutation nicht verändert. S-Boxen werden mit Differenzenverteilungstabellen (siehe Tabelle 5) analysiert, um die Veränderung von Differenzen mit Wahrscheinlichkeiten zu bestimmen. Chiffre 3 arbeitet mit 16-Bit Blöcken und verwendet zur Substitution jeweils

vier parallele S-Boxen. Prinzipiell könnte man die vier parallelen S-Boxen auch als eine einzige betrachten und für diese eine Differenzenverteilungstabelle erstellen. Alternativ bietet es sich auch an, für jede einzelne S-Box eine Differenzenverteilungstabelle aufzustellen (Tabelle 5). Der Übergang von Differenzen für diesen Vorgang wird durch $(\alpha_1, \alpha_2, \alpha_3, \alpha_4) \xrightarrow{S} (\beta_1, \beta_2, \beta_3, \beta_4)$ beschrieben, wobei α_i, β_i, $i = 1...4$ die Ein- und Ausgabedifferenz von $SBox_{ir}$ in der r-ten Runde beschreibt.

An dieser Stelle ist anzumerken, dass $\alpha_1 \,||\, \alpha_2 \,||\, \alpha_3 \,||\, \alpha_4$ die Eingabedifferenz und $\beta_1 \,||\, \beta_2 \,||\, \beta_3 \,||\, \beta_4$ die Ausgabedifferenz der Substitution einer Runde beschreibt. Da die Permutation linear ist, kann die Differenz nach der Permutation durch $(\beta_1, \beta_2, \beta_3, \beta_4) \xrightarrow{P} (\Gamma_1, \Gamma_2, \Gamma_3, \Gamma_4)$ angegeben werden [21, S. 118].

Eine differenzielle Charakteristik einer Verschlüsselungsrunde kann folglich durch $(\alpha_1, \alpha_2, \alpha_3, \alpha_4) \xrightarrow{\Re} (\Gamma_1, \Gamma_2, \Gamma_3, \Gamma_4)$ notiert werden. Diese wird als *Ein-Runden-Charakteristik* bezeichnet [21, S. 120]. Dabei wird zwischen *trivialer* und *nicht-trivialer* Ein-Runden-Charakteristik unterschieden. Die triviale Ein-Runden-Charakteristik hat eine Eingabedifferenz von 0, was zu einer Ausgabedifferenz von 0 mit einer Wahrscheinlichkeit von 1 führt. Allerdings ist es bei DKA nicht sinnvoll, denselben Block als Paar zu verwenden, da dieser auch zu identischem Geheimtext führt [21, S. 120]. Die nichttriviale Ein-Runden-Charakteristik hat eine Eingabedifferenz verschieden von 0 [21, S. 120]. Zur Berechnung der Wahrscheinlichkeit für eine Ein-Runden-Charakteristik werden die Wahrscheinlichkeiten der einzelnen Übergänge von Differenzen der S-Boxen multipliziert [21, S. 120], da diese unabhängig voneinander auftreten. Dabei wird angenommen, dass die Wahrscheinlichkeit bei der Berechnung der Permutation 1 ist [21, S. 120], da sich Differenzen aufgrund der Linearität der Permutation nicht verändern (siehe Kapitel 3.1.3).

3.2.2 n-Runden-Charakteristik

Mehrere Ein-Runden-Charakteristiken können miteinander verbunden werden unter der Voraussetzung, dass der Nachfolger mit der Eingabedifferenz beginnt, die der Ausgabedifferenz des Vorgängers entspricht [21, S. 120]. Auf diese Weise können Charakteristiken über n Runden erzeugt werden, die eine bestimmte Wahrscheinlichkeit haben [21, S. 120]. Unter der Annahme, dass die Wahrscheinlichkeiten der einzelnen Runden unabhängig voneinander auftreten, können diese miteinander multipliziert werden, um die Wahrscheinlichkeit der *n-Runden-Charakteristik* zu berechnen [21, S. 120].

Allerdings muss diese Annahme nicht zutreffend sein – es sind weitere detaillierte Analysen und Experimente notwendig [21, S. 120]. Ein Beispiel für eine 2-Runden-Charakteristik ist $(0,0,0,f) \xrightarrow{\Re} (1,1,0,1) \xrightarrow{\Re} (0,0,D,0)$. Es ist anzumerken, dass die Ausgabe- und Eingabedifferenz von Vorgänger und Nachfolger nicht doppelt notiert werden, da diese identisch sind. Eine Ein-Runden-Charakteristik, die dieselbe Ein- und Ausgabedifferenz hat, wird als *iterative Charakteristik* bezeichnet [21, S. 120]. Mit dieser kann über beliebig viele Runden eine Charakteristik erzeugt werden [21, S. 120]. Um den Rundenschlüssel k_r einer Chiffre anzugreifen, muss eine $(r-1)$-Runden-Charakteristik gefunden werden [21, S. 117]. In Chiffre 3 kann beispielsweise die folgende iterative 4-Runden-Charakteristik gefunden werden:

$$(0,0,2,0) \xrightarrow{\Re} (0,0,2,0) \xrightarrow{\Re} (0,0,2,0) \xrightarrow{\Re} (0,0,2,0) \xrightarrow{\Re} (0,0,2,0) \tag{18}$$

Abbildung 18 visualisiert exemplarisch die iterative 4-Runden-Charakteristik.

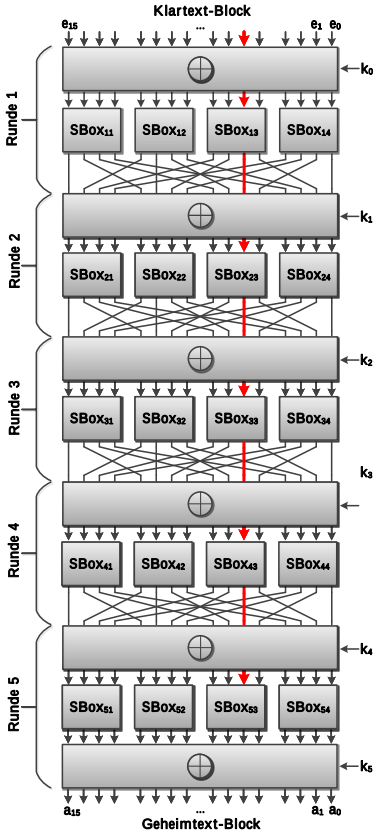


Abbildung 18: *Exemplarische Darstellung einer 4-Runden-Charakteristik für Chiffre 3*

Eine wichtige Frage bei Charakteristiken ist, wie hoch die Wahrscheinlichkeiten sein sollten, um einen Angriff mit DKA durchzuführen. Eine pauschale Antwort ist natürlich, dass diese möglichst hoch sein sollte. Allerdings kann die Suche sehr aufwändig sein. Für eine erfolgreiche DKA muss nicht zwangsweise die Charakteristik mit der höchsten Wahrscheinlichkeit gefunden werden. Die Bewertung der Wahrscheinlichkeit einer Charakteristik wird in Kapitel 3.6 näher betrachtet.

3.2.3 Suche nach Charakteristiken

Ein zentraler Bestandteil einer praktischen DKA ist die Suche nach Charakteristiken mit hohen Wahrscheinlichkeiten. In der Veröffentlichung [28, S. 29] von Shamir und Biham ist beschrieben, dass ein heuristisches Programm verwendet wurde, um die besten Charakteristiken zu finden. Die Autoren geben an, dass sie sich sicher seien, die beste Charakteristik gefunden zu haben, allerdings hätten sie keinen Beweis dafür. In [21, S. 120] beschreiben Knudsen und Robshaw eine Heuristik zur Suche von Charakteristiken: Man beginnt die Suche, indem man die höchste Ein-Runden-Charakteristik verwendet. Anschließend betrachtet man die Ausgabedifferenz dieser Charakteristik und sucht in der Differenzenverteilungstabelle der S-Boxen der nächsten Runde nach dem Eintrag, der mit dieser Eingangsdifferenz die höchste Wahrscheinlichkeit hat. Anschließend berechnet man die Charakteristik für diese Runde und setzt das Verfahren iterativ fort. Allerdings führt dieses Verfahren nicht immer zu der höchsten Charakteristik über mehrere Runden [21, S. 120]. Das liegt hauptsächlich daran, dass Komponenten einer Chiffre (wie Permutationen) die Suche erschweren [21, S. 120]. In der offenen Literatur über DKA gibt es wenige konkrete Verfahren oder Algorithmen zur Suche nach Charakteristiken. In Kapitel 7 wird darauf noch weiter eingegangen.

Eine vollständige Suche nach Charakteristiken kann sehr aufwändig sein. Wesentliche Gründe dafür sind die Blockgröße und die Rundenanzahl einer Chiffre. An dieser Stelle wird Chiffre 2 weiter betrachtet. Chiffre 2 hat eine Blockgröße von 16 Bit und drei Runden. Bei der experimentellen Untersuchung aller möglichen auftretenden Charakteristiken für Chiffre 2 wurden 67.937.484.003 Charakteristiken gefunden. Das entspricht ungefähr 2^{36} unterscheidbaren Charakteristiken. Dieser Wert ist kleiner als 2^{64}, was einer vollständigen Schlüsselsuche entsprechen würde. Es bleibt festzuhalten, dass für eine Implementierung einer DKA Suchverfahren implementiert werden müssen, die deutlich weniger Charakteristiken untersuchen müssen, um effizienter zu sein. Trotzdem stellt der zeitliche Aufwand weiterhin ein großes Problem dar. Die experimentelle Untersuchung hat etwa sechs Tage in An-

spruch genommen.

Abschließend ist zu erwähnen, dass die Suche nach Charakteristiken unabhängig vom Schlüssel ist und somit nicht immer für eine konkrete Chiffre wiederholt werden muss. Die Unabhängigkeit bei der Betrachtung von Differenzen folgt aus Gleichung 16 (die Rundenschlüssel heben sich bei der Schlüsseladdition gegenseitig auf). Daher kann die Suche nach Charakteristiken einmalig erfolgen. Die gewonnenen Daten können bei späteren Angriffen wiederverwendet werden.

3.3 Differenzial

Bei der Suche nach Charakteristiken kann es vorkommen, dass man sehr viele davon findet, die allerdings alle eine geringe Wahrscheinlichkeit aufweisen. Dies hat zur Folge, dass bei der Wiederherstellung von Schlüsselbits Fehler passieren. Des Weiteren kann bei einem Paar von Nachrichten nicht überprüft werden, ob es der Folge von Ein- und Ausgabedifferenzen einer Charakteristik stets folgt. Dies liegt daran, dass der Angreifer nicht in den Verschlüsselungsprozess einer Chiffre hineinschauen kann [21, S. 122]. Bei einem praktischen Angriff kann festgestellt werden, dass ein Angreifer lediglich die Eingabedifferenz des Nachrichten-Paares und die letzte Ausgabedifferenz der Charakteristik benötigt [21, S. 122]. An dieser Stelle wird Chiffre 3 beispielhaft betrachtet. Die Differenzen, die innerhalb der Chiffre auftreten, können nicht verifiziert werden. Aus diesem Grund sind für einen Angriff auf eine Chiffre Charakteristiken der Form

$$(\alpha_1, \alpha_2, \alpha_3, \alpha_4) \xrightarrow{\Re} (?, ?, ?, ?) \xrightarrow{\Re} ... \xrightarrow{\Re} (?, ?, ?, ?) \xrightarrow{\Re} (\Gamma_1, \Gamma_2, \Gamma_3, \Gamma_4) \qquad (19)$$

interessant, wobei ? bedeutet, dass eine Differenz unbekannt und irrelevant ist [21, S. 122]. Werden mehrere Charakteristiken gefunden, die dieser Struktur entsprechen, können diese zusammengefasst werden. Diese Struktur wird dann als *Differenzial* bezeichnet [21, S. 122]. Wichtig ist, dass die Eingabedifferenz α_i und Ausgabedifferenz Γ_i für alle i identisch sind. Die mindeste Wahrscheinlichkeit des so erhaltenen Differenzials kann durch Summenbildung der Wahrscheinlichkeiten der Charakteristiken berechnet werden [21, S. 122]. Ein Differenzial, bestehend aus mehreren Charakteristiken, ist folglich eine Struktur, die die Wahrscheinlichkeit auf Erfolg einer DKA erhöht.

3.4 Filterung

Bei der praktischen Durchführung der DKA werden Klartext-Nachrichten-Paare generiert, die alle eine bestimmte Differenz aufweisen. Die Differenz

kann von Differenzialen bzw. Charakteristiken erhalten werden. Diese beschreiben durch eine Folge von Differenzen, wie sich die Differenzen der Nachrichten-Paare durch die einzelnen Verschlüsselungsrunden einer Chiffre verändern. Oft kommt es vor, dass die Paare die Folge von Differenzen des Differenzials bzw. einer Charakteristik nicht erfüllen. Wenn ein Paar die erste Differenz eines Differenzials erfüllt, bedeutet es noch nicht, dass es auch die letzte Differenz erfüllt (Γ_i). Ein Paar, welches einer Charakteristik bzw. einem Differenzial folgt, wird als *richtiges Paar* bezeichnet [21, S. 122]. Dahingegen wird ein Paar, welches einer Charakteristik bzw. einem Differenzial nicht folgt, als *falsches Paar* bezeichnet [21, S. 122]. Falsche Paare können bei der Wiederherstellung des gesuchten Rundenschlüssels ignoriert werden, da diese die Identifizierung nur erschweren [23, S. 179]. Werden sie vor der Wiederherstellung entfernt, so kann der gesuchte Rundenschlüssel leichter gefunden werden [23, S. 179]. Falsche Paare können durch *Filterung* aller Paare identifiziert und entfernt werden [21, S. 124]. Oft kann dies bereits durch Betrachtung der zugehörigen Geheimtexte eines Klartext-Paares realisiert werden [21, S. 124].

An dieser Stelle wird Chiffre 3 aus Kapitel 2.4.3 betrachtet. Bei der Suche nach Charakteristiken für diese Chiffre kann beispielsweise die in 18 dargestellte 4-Runden-Charakteristik gefunden werden. Diese Charakteristik hat die Differenz 0x2 vor der letzten Runde. Der Differenzenverteilungstabelle der S-Box in Tabelle 5 kann entnommen werden, dass eine Eingabedifferenz von 0x2 die Ausgabedifferenzen 0x1, 0x2, 0x9 oder 0xA erzeugen kann. Folglich kann durch Bildung der Differenz der Geheimtexte bei einem potenziell guten Paar nur eine der zuvor genannten Ausgabedifferenzen auftreten. Erfüllt ein Paar diese Geheimtext-Differenz nicht, kann dies als falsches Paar identifiziert und entfernt werden [21, S. 124]. Des Weiteren werden oft Charakteristiken bzw. Differenziale verwendet, in denen nicht in alle S-Boxen eine Differenz $\neq 0$ aufweisen. Bei der Betrachtung der Charakteristik in (18) liegt beispielsweise an drei S-Boxen die Differenz 0 an (S-Box_{51}, S-Box_{52} und S-Box_{54}) [21, S. 125]. Folglich können diese S-Boxen ebenfalls zur Filterung verwendet werden, denn die Differenz der Geheimtexte muss an den Bits a_{15} bis a_8 und a_3 bis a_0 ebenfalls 0 sein [21, S. 125]. Nach [21, S. 125] ist eine gute Filterung sehr wichtig für den Erfolg bei einer DKA.

In [23, S. 180] wird beschrieben, dass Biham und Shamir ebenfalls effiziente Möglichkeiten zur Filterung nutzten. Die Idee dabei ist, dass ein gutes Paar im Durchschnitt eine größere Kandidatenmenge von Schlüsseln erzeugt als ein falsches Paar [23, S. 180]. So kann mit einer Gewichtsfunktion und einem experimentell ermittelten Schwellwert eine sehr große Menge von falschen

Paaren identifiziert werden [23, S. 180].

3.5 Schlüsselinformationen wiederherstellen

In diesem Abschnitt wird die Wiederherstellung des gesuchten Schlüssels beschrieben. Diese erfolgt sukzessive durch Wiederherstellung der einzelnen Rundenschlüssel einer Blockchiffre. Das Verfahren findet dabei unterschiedlich in Abhängigkeit der Verschlüsselungsrunde statt. Bei der Wiederherstellung des ersten und zweiten Rundenschlüssels ist der Ablauf verschieden von dem, welcher für eine beliebige Verschlüsselungsrunde verwendet wird. Daher wird in Kapitel 3.5.1 das Verfahren für eine Chiffre bestehend aus einer einzigen Verschlüsselungsrunde (wie beispielsweise Chiffre 1 in Kapitel 2.4.1) vorgestellt und in Kapitel 3.5.2 das Verfahren für Chiffren mit potenziell beliebig vielen Verschlüsselungsrunden (wie beispielsweise Chiffre 2 in Kapitel 2.4.2).

3.5.1 Eine Verschlüsselungsrunde

An dieser Stelle wird Chiffre 1 aus Kapitel 2.4.1 stellvertretend für Chiffren mit einer Verschlüsselungsrunde betrachtet. Chiffre 1 verwendet eine Blockgröße von 16 Bit und besteht aus Schlüsseladdition und Substitution. Die Verschlüsselung von zwei Klartext-Blöcken m_1 und m_2 erfolgt dabei, wie in den Gleichungen 20 zu sehen ist, in drei Schritten:

$$\begin{aligned} u_1 &= m_1 \oplus k_0 & u_2 &= m_2 \oplus k_0 \\ v_1 &= S(u_1) & v_2 &= S(u_2) \\ c_1 &= v_1 \oplus k_1 & c_2 &= v_2 \oplus k_1 \end{aligned} \tag{20}$$

Die Rundenschlüssel k_0 und k_1 sind unbekannt. Die Werte der internen Variablen u_i sind folglich unbekannt. Allerdings kann die Differenz der u_i wie in Gleichung 21 berechnet werden:

$$u_1 \oplus u_2 = (m_1 \oplus k_0) \oplus (m_2 \oplus k_0) = m_1 \oplus m_2 \tag{21}$$

Diese Differenz kann verwendet werden, um den Rundenschlüssel k_1 zu bestimmen [21, S. 111]. Für die Berechnung werden zwei Paare von Klartext- und Geheimtext-Blöcken (m_1, c_1), (m_2, c_2) benötigt – die Differenz der Klartext-Blöcke spielt hier keine Rolle. Man wählt alle Werte t von k_1 und mit diesen können die internen Variablen v_i, wie in Gleichung 22 zu sehen ist, berechnet werden [21, S. 111].

$$v_1 = c_1 \oplus t \qquad v_2 = c_2 \oplus t \tag{22}$$

Da die Funktionsweise der S-Boxen öffentlich zugänglich ist und die S-Boxen auch invertierbar sind, kann $S^{-1}(v_1)$ und $S^{-1}(v_2)$ berechnet werden. Diese Werte können nicht direkt mit den u_i verglichen werden, da diese unbekannt sind [21, S. 111]. Allerdings gilt bei einem korrekten Rateversuch t von k_1 die Gleichung 23:

$$u_1 \oplus u_2 = S^{-1}(v_1) \oplus S^{-1}(v_2) \tag{23}$$

Wie bereits ausgeführt, ist die Differenz der u_i leicht durch Berechnung der Differenz der Klartext-Blöcke zu erhalten. Ist Gleichung 23 erfüllt, so wird t als Schlüsselkandidat notiert. Diese Prozedur wird für alle möglichen Werte t von k_1 wiederholt. Dies ist eine vollständige Schlüsselsuche für einen Rundenschlüssel und nicht über den vollständigen Schlüsselraum (dieser ist wesentlich größer). Falls anschließend mehr als ein Kandidat für k_1 übrig bleibt, muss das Verfahren an neuen Klartext- und Geheimtext-Blöcken durchgeführt werden [21, S. 111]. Wenn k_1 korrekt wiederhergestellt wurde, kann k_0 durch Nutzung eines bekannten Klartext-Geheimtext-Block-Paares (m_1, c_1), wie in Gleichung 24 zu sehen ist, in drei Schritten berechnet werden:

$$\begin{aligned} v_1 &= c_1 \oplus k_1 \\ u_1 &= S^{-1}(v_1) \\ k_0 &= m_1 \oplus u_1 \end{aligned} \tag{24}$$

3.5.2 Beliebige Anzahl von Verschlüsselungsrunden

Das zuvor beschriebene Verfahren zur Wiederherstellung von Rundenschlüsseln kann bei einer Chiffre mit mehreren Verschlüsselungsrunden nicht verwendet werden. Deshalb wird nun Chiffre 2 aus Kapitel 2.4.2 betrachtet – als Beispiel für eine Chiffre mit einer „beliebigen“ Anzahl von Verschlüsselungsrunden.

Chiffre 2 arbeitet auf 16-Bit Blöcken und besteht aus drei Verschlüsselungsrunden mit vier Rundenschlüsseln. Jede Verschlüsselungsrunde (bis auf die letzte) besteht aus Schlüsseladdition, Substitution und Permutation. Der Verschlüsselungsprozess von einem Klartext-Block m_1 erfolgt, wie in den Gleichungen 25 zu sehen ist, in mehreren Schritten:

$$
\begin{aligned}
u_1 &= m_1 \oplus k_0 \\
v_1 &= S(u_1) \\
w_1 &= P(v_1) \\
\\
u_2 &= w_1 \oplus k_1 \\
v_2 &= S(u_2) \\
w_2 &= P(v_2) \\
\\
u_3 &= w_3 \oplus k_3 \\
v_3 &= S(u_3) \\
c_1 &= v_3 \oplus k_4
\end{aligned}
\tag{25}
$$

Um eine DKA an einer Chiffre mit r-Runden mit $r > 1$ durchzuführen, muss zunächst die Chiffre nach einer geeigneten Charakteristik bzw. einem Differenzial durchsucht werden. Bei einem Angriff werden die Schlüsselbits eines Rundenschlüssels angegriffen, für die die Eingabedifferenz der Runde $(r-1)$ ungleich 0 im Differenzial ist. Folglich müssen unter Umständen mehrere Charakteristiken bzw. Differenziale gesucht werden, um alle Schlüsselbits eines Rundenschlüssels anzugreifen [21, S. 126]. Nun soll für Chiffre 2 die folgende 2-Runden-Charakteristik betrachtet werden, welche die Autoren von [27] beispielsweise angeben:

$$(0, E, 0, 0) \xrightarrow{\Re} (0, 8, 0, 0) \xrightarrow{\Re} (4, 8, 1, 0) \tag{26}$$

Die in 26 dargestellte Charakteristik kann beispielsweise wie in Abbildung 19 visualisiert werden. Die roten Linien geben an, welche Bits auf 1 gesetzt sind. So kann schnell erkannt werden, wie sich die Differenzen des angegebenen Differenzials durch die Verschlüsselungsrunden verändern.

Die Wahrscheinlichkeit der Charakteristik beträgt $P_{Char} = \frac{48}{256} \approx 0.19$. Bei der Betrachtung der Eingabedifferenz in Runde 3 ist zu sehen, dass die Differenz bei drei S-Boxen von 0 verschieden ist (S-Box$_{31}$, S-Box$_{32}$ und S-Box$_{33}$). Mit der Charakteristik werden die Schlüsselbits 4-16 von k_3 angegriffen. Der nächste Schritt besteht darin, eine ausreichend große Anzahl an Nachrichten-Paaren zu generieren, die die Eingabedifferenz der Charakteristik erfüllen. In [27, S. 3] und [30, S. 28] wird die Anzahl der notwendigen Klartext-Paare N_D durch die in 27 gegebene Formel berechnet, wobei c eine kleine Konstante und P_{Char} die Wahrscheinlichkeit der Charakteristik ist.

$$N_D = \frac{c}{P_{Char}} \tag{27}$$

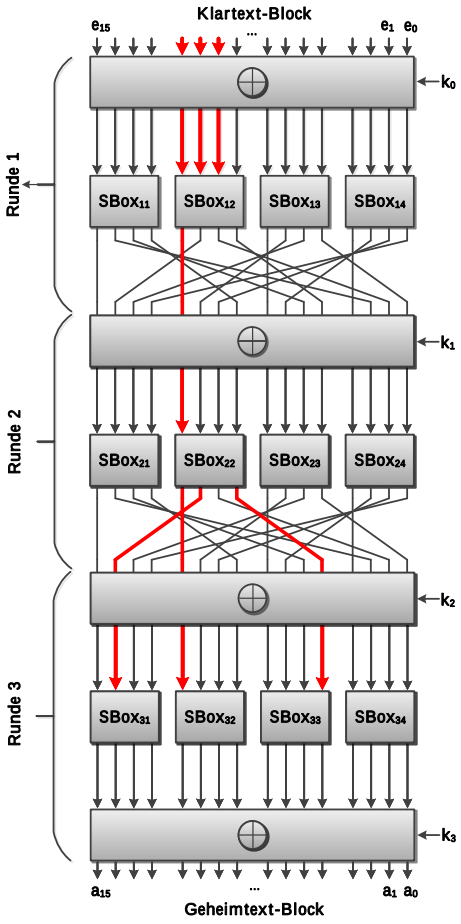


Abbildung 19: *Exemplarische Darstellung einer 2-Runden-Charakteristik für Chiffre 2*

In [27, S. 4] wird beispielsweise beim Angriff auf zwölf Schlüsselbits bei einer 3-Runden-SPN-Chiffre für $c = 10$ gewählt. In Kapitel 3.6 wird eine weitere Möglichkeit vorgestellt, mithilfe derer die notwendige Anzahl von Nachrichten bestimmt werden kann.

Vor der Wiederherstellung von Schlüsselbits sollten die falschen Nachrichten-Paare (siehe Kapitel 3.4) durch die vorgestellten Filterungstechniken entfernt werden. Zur Wiederherstellung von Schlüsselbits wird jeder mögliche Schlüssel t von k_r durchprobiert, wobei nur die Bitstellen in t betrachtet werden, die durch S-Boxen substituiert werden, also deren Differenz in der letzten Runde der Charakteristik von 0 verschieden ist. Für jedes t wird ein Zähler erstellt. Mit jedem möglichen t wird die r-te Verschlüsselungsrunde für alle Paare partiell entschlüsselt, d. h. bei den vorgestellten SPN-Chiffren werden Schlüsseladdition, Substitution und Permutation (falls in der Runde vorhanden) invertiert. Anschließend wird die Differenz der partiell entschlüsselten Blöcke gebildet und mit der Differenz der Charakteristik verglichen. Falls die Differenzen übereinstimmen, wird dies als Treffer vermerkt und der Zähler erhöht. Bei n Nachrichten-Paaren und einer Wahrscheinlichkeit P_{Char} der Charakteristik werden $P_{Char} \cdot n$ Treffer für ein korrektes t erwartet. Der Kandidat t mit dem höchsten Zähler kann als richtiger Teilschlüssel erkannt werden, falls die Wahrscheinlichkeit P_{Char} und die Anzahl der verwendeten Paare hoch genug waren [23, S. 174].

In der beispielhaft betrachteten Charakteristik aus 26 werden zwölf Schlüsselbits angegriffen, d. h. es werden 2^{12} verschiedene t durchprobiert. Für jedes t wird die dritte Verschlüsselungsrunde entschlüsselt, d. h. für ein Paar von Klartext-Blöcken (m_1, m_2) mit bekannten Geheimtext-Blöcken (c_1, c_2) wird $u_{3_{m_1}} = S^{-1}(c_1 \oplus t)$ und $u_{3_{m_2}} = S^{-1}(c_2 \oplus t)$ berechnet. Anschließend wird die Differenz der Blöcke bestimmt $\Delta d = u_{3_{m_1}} \oplus u_{3_{m_2}}$ und mit der Differenz der Charakteristik verglichen. Dies wird für jedes Paar von Klartext-Blöcken wiederholt. Am Ende des Verfahrens sollte der Zähler des richtigen Teilschlüssels am höchsten sein.

3.5.3 Widerstandsfähigkeit gegen differenzielle Kryptoanalyse

Die Widerstandskraft von Chiffren gegen differenzielle Kryptoanalyse hängt direkt von den S-Boxen ab [22, S. 403]. An dieser Stelle werden $m \times n$-Bit S-Boxen betrachtet, d. h. es gibt m Eingabebits und n Ausgabebits.

In der Vergangenheit sind verschiedene Chiffren veröffentlicht worden, die unterschiedliche Größen der S-Boxen verwenden. So nutzt beispielsweise Khafre eine einzige 8×32-Bit S-Box und LOKI eine 12×8-Bit S-Box [22, S. 403].

Nach [22, S. 403] sind S-Boxen umso sicherer, je größer sie sind. Die Schwierigkeit der Analyse bei größeren S-Boxen besteht darin, nützliche Statistiken für einen Angriff zu finden. Starke S-Boxen lassen sich leichter finden, wenn die S-Boxen größer sind [22, S. 404]. Bei der Größe ist der Faktor m wichtiger als der Faktor n [22, S. 404], durch eine Erhöhung von n wird die Effektivität differenzieller Kryptoanalyse vermindert, dadurch erhöht sich allerdings die der linearen Kryptoanalyse [22, S. 404].

Kryptologen schlugen für die Wahl von S-Boxen vor, für alle S-Boxen die gleiche Verteilungstabelle der Differenzen zu nutzen [22, S. 404]. Dadurch soll Schutz vor differenzieller Kryptoanalyse erreicht werden, da die Differenziale in jeder Runde geglättet würden [22, S. 404]. Laut [22, S. 404] wurde LOKI nach diesem Prinzip entworfen, allerdings erleichtert dieser Ansatz manchmal differenzielle Kryptoanalyse. Ein besserer Ansatz besteht darin, ein möglichst kleines maximales Differenzial zu gewährleisten [22, S. 404].

In [22, S. 404] finden sich vier Ansätze, um gute S-Boxen zu wählen. Diese sind im Folgenden beschrieben:

1. Die S-Boxen werden zufällig gewählt. Kleine zufällig gewählte S-Boxen sind unsicher, dahingegen könnten große zufällige S-Boxen gut sein. Zufällige S-Boxen ab acht Eingabebits werden als ziemlich stark bezeichnet. Verbessert werden können die S-Boxen, wenn diese vom Schlüssel abhängen. Beispielweise verwendet IDEA große und schlüsselabhängige S-Boxen.

2. Die S-Boxen werden gewählt und getestet. Es gibt Chiffren, welche S-Boxen zufällig wählen und sie dann auf gewünschte Eigenschaften überprüfen.

3. Die S-Boxen werden händisch konstruiert. Bei diesem Ansatz wird wenig Mathematik verwendet, die S-Boxen werden mit intuitiven Verfahren erzeugt.

4. Die S-Boxen werden auf Basis mathematischer Prinzipien erzeugt, sodass sie nachweisbaren Schutz vor differenzieller und lineare Kryptoanalyse bieten und gute Diffussionseigenschaften aufweisen.

Innerhalb der Forschung wird zwischen der Wahl von zufälligen S-Boxen und S-Boxen mit bestimmten Eigenschaften diskutiert [22, S. 405]. S-Boxen mit bestimmten Eigenschaften könnten optimalen Schutz vor differenzieller und linearer Kryptoanalyse bieten, allerdings kann schlecht abgeschätzt werden,

wie gut sie gegen unbekannte Angriffsmethoden schützen [22, S. 405]. Schneier empfiehlt beispielsweise, dass S-Boxen so groß wie möglich, zufällig und schlüsselabhängig sein sollten [22, S. 405].

3.6 Aufwandsanalyse

In der Literatur wird der Aufwand einer DKA für gewöhnlich durch die Anzahl der zu untersuchenden Paare beschrieben [23, S. 176 ff] [30, S. 28]. Die Suche nach einem Differenzial bzw. nach Charakteristiken wird dabei häufig nicht näher betrachtet – möglicherweise aufgrund der Tatsache, dass die Suche schlüsselunabhängig ist und daher nicht wiederholt werden muss. Im Folgenden werden zunächst drei Faktoren beschrieben, die die Anzahl der notwendigen Paare beeinflussen. Der vierte Abschnitt ist nach [28, S. 30] ein Hilfsmittel zur Beurteilung der praktischen Nutzbarkeit einer Charakteristik für eine DKA.

Wahrscheinlichkeit der Charakteristik

Der richtige Teilschlüssel wird mit einer bestimmten Wahrscheinlichkeit durch eine DKA gefunden. Je höher die Wahrscheinlichkeit eines Differenzials bzw. einer Charakteristik ist, desto häufiger wird der richtige Teilschlüssel bei der partiellen Entschlüsselung und Differenzbildung eines Paares identifiziert [23, S. 176]. Dies führt dazu, dass der korrekte Teilschlüssel durch weniger Paare gefunden werden kann [23, S. 176].

Anzahl der angegriffenen Schlüsselbits durch eine Charakteristik

In Abhängigkeit des verwendeten Differenzials bzw. der verwendeten Charakteristik werden unterschiedlich viele Schlüsselbits in einem Angriff versucht wiederherzustellen. Würden beispielsweise beim Angriff auf Chiffre 2 alle vier S-Boxen einer Runde angegriffen, müssten 2^{16} verschiedene Schlüssel getestet werden. Für jeden Schlüsselkandidat werden alle Paare partiell entschlüsselt und anschließend wird die Differenz gebildet. Aus praktischen Gründen kann es sinnvoller sein, weniger Schlüsselbits mit einem Differenzial bzw. Charakteristik anzugreifen [23, S. 176]. Gründe dafür sind eine kleinere Anzahl an Zählern [23, S. 176], eine höhere Wahrscheinlichkeit des Differenzials bzw. der Charakteristik und das S-Boxen, deren Eingabedifferenz gleich 0 ist, zur Filterung verwendet werden können [28, S. 30].

Filterung der Paare

Durch die Filterung können falsche Paare erkannt und entfernt werden, bevor der gesuchte Schlüssel wiederhergestellt wird [23, S. 177]. Folglich müssen weniger Paare partiell entschlüsselt werden und es werden weniger falsche Schlüssel gezählt [23, S. 177].

Rauschverhältnis

Grundsätzlich hängt der Erfolg einer DKA von der Wahrscheinlichkeit des Differenzials (P_{Diff}) bzw. der Charakteristik (P_{Char}), der Größe des gesuchten Teilschlüssels und der Filterung der Paare ab [23, S. 179]. Um die Erfolgsaussichten zu beurteilen, kann das so genannte *Rauschverhältnis der Häufigkeitsverteilung* (S/N) (in der Literatur als „signal-to-noise ratio“ bezeichnet) verwendet werden. Es beschreibt das Verhältnis der Anzahl von Paaren, die dem Differenzial folgen, zu der durchschnittlichen Häufigkeit der falschen Teilschlüssel [23, S. 179]. Zur mathematischen Beschreibung des S/N sind die folgenden Bezeichnungen notwendig, welche [23, S. 179] entnommen sind:

- $\mathcal{K}$: Anzahl der Bits im gesuchten Teilschlüssel
- $2^{\mathcal{K}}$: Anzahl der möglichen verschiedenen Teilschlüssel
- $\mathcal{M}$: Anzahl der insgesamt untersuchten Paare
- α: Durchschnittliche Anzahl der Teilschlüsselkandidaten, die ein Paar liefert
- β: Anteil der Paare, die durch die Filterung entfernt wurden. Falls keine Paare durch einen Filter entfernt werden, gilt $\beta = 1$.

Nach [23, S. 179] kann davon ausgegangen werden, dass bei der Verwendung von hinreichend vielen Klartext-Paaren jeder Teilschlüssel im Durchschnitt $\frac{\mathcal{M} \cdot \alpha \cdot \beta}{2^{\mathcal{K}}}$ gezählt wird. Der korrekte Schlüssel wird $P_{Char} \cdot \mathcal{M}$-mal gezählt [23, S. 179]. Folglich wird das Rauschverhältnis der Häufigkeitsverteilung wie in Gleichung 28 definiert.

$$S/N = \frac{\mathcal{M} \cdot P_{Char}}{\mathcal{M} \cdot \alpha \cdot \frac{\beta}{2^{\mathcal{K}}}} = \frac{2^{\mathcal{K}} \cdot P_{Char}}{\alpha \cdot \beta} \tag{28}$$

[23, S. 180] kann entnommen werden, dass Biham und Shamir experimentell festgestellt haben, dass bei einem S/N-Wert von 1 bis 2 etwa 20 bis 40 richtige Paare notwendig sind, um den richtigen Teilschlüssel zu identifizieren. Falls

S/N wesentlich kleiner ist, ist der Angriff aufgrund der hohen Anzahl von benötigten Eingabepaaren nicht durchführbar [23, S. 180]. An dieser Stelle sei erwähnt, dass die Werte der Variablen α und β experimentell für jede Charakteristik bestimmt werden müssen.

3.7 Weitere Optimierungen

Die bisher eingeführten Elemente beschreiben alle essentiell notwendigen Bestandteile zur praktischen Durchführung einer differenziellen Kryptoanalyse einer Blockchiffre. In diesem Abschnitt wird eine Übersicht über weitere Varianten gegeben, die die Erfolgsaussichten unter Umständen verbessern können. Die Anwendbarkeit der Variationen ist nach [21, S. 154] beschränkt, aber falls sie anwendbar sind, können sie sehr effektiv sein. In dieser Masterarbeit werden die Variationen nicht detailliert diskutiert.

Truncated Differential Cryptanalysis

Bei der Suche nach Charakteristiken wird der Versuch unternommen, die Änderungen der Differenzen von Runde zu Runde vorherzusagen. Dabei kann festgestellt werden, dass bei fester Eingabedifferenz Δd einer Runde verschiedene Ausgabedifferenzen $\Delta d'$ möglich sind. Bei der Betrachtung der Unterschiede der verschiedenen Ausgabedifferenzen $\Delta d'_i$ auf Bitebene kann oftmals beobachtet werden, dass sich nicht alle Bits unterscheiden, sondern dass viele Bitstellen identisch sind. Für die Bitstellen, an denen sich die Ausgabedifferenzen unterscheiden, wird ein Platzhalter eingeführt [21, S. 155]. Ein *Truncated Differential* ist eine Sammlung von Charakteristiken ähnlich wie ein Differenzial. Mithilfe des Musters von Charakteristiken wird dann der Versuch unternommen, Schlüsselbits wiederherzustellen. An dieser Stelle sei auf [21, S. 154 ff] verwiesen.

Impossible differential cryptanalysis

Das Ziel der vorgestellten DKA ist es, statistische Unregelmäßigkeiten auszunutzen. Dabei ist die Suche nach Charakteristiken mit hoher Wahrscheinlichkeit ein wichtiger Bestandteil. Unter einer *impossible differential cryptanalysis* wird die Nicht-Existenz von Differenzen ausgenutzt, um Schlüsselbits wiederherzustellen [30, S. 30].

4 Konzept der CT2-Komponenten

In diesem Kapitel wird das Konzept der CT2-Komponenten beschrieben, welche zusammen eine differenzielle Kryptoanalyse für ausgewählte Chiffren ermöglichen. Dabei wird zunächst der Aufbau des Tutorials unter Berücksichtigung der Zielgruppe und des Zwecks von CT2 beschrieben. Anschließend wird dargestellt, wie das Verfahren der differenziellen Kryptoanalyse sinnvoll in CT2 integriert werden kann. Im letzten Abschnitt wird der Aufbau der Vorlage(n) beschrieben.

4.1 Aufbau des Tutorials

Als E-Learning-Plattform hat CT2 als Zielgruppe neben interessierten Anwendern (möglicherweise auch ohne Vorkenntnisse) auch Personenkreise, die im Bereich der (universitären) Lehre oder Fortbildung anzusiedeln sind. Für die Konzeption des Tutorials bedeutet dies, dass das Verfahren der differenziellen Kryptoanalyse ohne größere Vorkenntnisse durchführbar sein soll. Darüber hinaus soll es auch einem anspruchsvolleren Personenkreis gerecht werden. An dieser Stelle sei angemerkt, dass es sich bei der Implementierung in CT2 um einen Basis-Angriff handelt, der durch weitere Optimierungen verbessert werden kann. Der Fokus liegt dabei auf der Vermittlung von Wissen, einer hohen Anwenderfreundlichkeit und einer beispielhaft praktisch durchführbaren differenziellen Kryptoanalyse. Anwenderfreundlichkeit bedeutet insbesondere, dass die Benutzerschnittstelle die Inhalte ansprechend präsentiert. Attraktiv visualisierte Inhalte können den Verstehensprozess positiv beeinflussen.

In E-Learning-Software kann Wissen durch Interaktivität vermittelt werden. Dabei kann jeder Anwender individuell sein eigenes Tempo wählen. Da differenzielle Kryptoanalyse ein mathematisch-analytisches Verfahren ist, soll die grundsätzliche Idee ebenfalls vermittelt werden. Dies soll auf einem Niveau erfolgen, dem sowohl ein Anwender ohne Vorkenntnisse als auch ein anspruchsvoller Anwender folgen kann. Für die Vermittlung der theoretischen Grundlagen eignet sich eine Präsentation innerhalb einer Komponente. Diese kann beliebig kleinschrittig aufgebaut und in Abschnitte gegliedert werden. Durch die Gliederung kann ein anspruchsvoller Anwender beispielsweise einen Abschnitt überspringen, der für Anwender ohne Vorkenntnisse relevant sein könnte. Tiefere theoretische Kenntnisse über das Verfahren können in der Online-Hilfe der Komponente vermittelt werden, wo darüber hinaus Literaturverweise für anspruchsvolle Anwender gegeben werden können. Diese können dadurch motiviert werden, eigene Recherchen anzustellen.

Bei der praktischen Durchführung von differenzieller Kryptoanalyse soll das Verfahren dem Anwender durch einen hohen Grad an Interaktivität transparenter gemacht werden. Das hat den Vorteil, dass der Angriff dem Anwender „erfahrbar“ gemacht werden kann und dass jeder Schritt bewusster wahrgenommen wird. Dadurch können die verschiedenen Lerntypen angesprochen werden. Die differenzielle Kryptoanalyse besteht aus mehreren Schritten, bei denen unterschiedliche Entscheidungen getroffen werden können. Zu Beginn des Verfahrens ist der erste Schritt die Analyse einer Chiffre. Dabei muss festgelegt werden, welche S-Boxen in einer Verschlüsselungsrunde angegriffen werden sollen und mit welcher Suchstrategie nach Differenzialen gesucht wird. Die Analyseergebnisse bestehen aus einem Differenzial mit einer festen Wahrscheinlichkeit. Die enthaltenen Charakteristiken können graphisch sinnvoll und attraktiv durch den Anwender inspiziert werden. Die Erzeugung und Verschlüsselung der Nachrichten-Paare können ebenfalls als je zwei unabhängige Schritte voneinander realisiert werden. Ein wichtiger Faktor dabei ist die Anzahl der Nachrichten-Paare, die ebenfalls durch den Anwender spezifiziert werden könnte. Die Wiederherstellung von Schlüsselbits erfolgt probabilistisch, wobei mehrere Schlüsselkandidaten getestet werden. Die Auswertung der Schlüsselkandidaten erfolgt durch statistische Mittel. Alle Einstellungen und Parameter, die der Anwender aktiv beeinflussen kann, erfüllen den Zweck, dem Anwender eine größere Entscheidungsfreiheit bei der Durchführung der differenziellen Kryptoanalyse zu geben. Dadurch kann, je nach Interesse, mit den Parametern gearbeitet und erfahrbar werden, welche Faktoren in welchem Maß Einfluss auf den Erfolg einer differenziellen Kryptoanalyse haben.

Die Implementierung in CT2 soll drei Chiffren enthalten. Es ist sinnvoll, das Tutorial in mehrere Lerneinheiten aufzuteilen. Wie in Kapitel 2.4 ausgeführt, ist die Wahl der Chiffren so getroffen worden, dass die Lerneinheiten des Tutorials einen ansteigenden Schwierigkeits- und Komplexitätsgrad erlauben. Die Vermittlung eines Basis-Angriffs mittels differenzieller Kryptoanalyse kann sinnvoll auf drei Lerneinheiten aufgeteilt werden. Diese werden je als eigenständiges Tutorial bezeichnet.

In diesem Abschnitt werden zunächst die Gemeinsamkeiten der Tutorials beschrieben. Der Anwender soll die Freiheit haben, den wiederherzustellenden Schlüssel k der Chiffren selbst spezifizieren zu können. Alle Tutorials sollen neben der praktischen Durchführung des Verfahrens auch theoretisch notwendige Kenntnisse vermitteln. Es soll stets die Möglichkeit bestehen, Abschnitte des theoretischen Teils zu wiederholen, zu überspringen oder ganz auszulassen. Der Inhalt der einzelnen Tutorials wird im Folgenden beschrieben.

4.1.1 Tutorial 1

Im ersten Tutorial wird Chiffre 1 verwendet. Dies soll einen Überblick über die differenzielle Kryptoanalyse vermitteln. Dabei geht es im Kern um die zentrale Idee der Betrachtung von Differenzen. Zunächst sollen die Bestandteile von Chiffre 1 beschrieben werden. Wichtig ist die Beschreibung des bei der Schlüsseladdition verwendeten XOR-Operators. Dann soll der Verschlüsselungsprozess von Chiffre 1 kleinschrittig beschrieben werden, sodass dem Anwender die im Verschlüsselungsprozess auftretenden internen Zustände klar werden. Anschließend soll skizziert werden, wie die Wiederherstellung der Rundenschlüssel durchgeführt wird.

4.1.2 Tutorial 2

Das zweite Tutorial basiert auf Chiffre 2. Aufgrund der erhöhten Anzahl von Verschlüsselungsrunden ist das Vorgehen wie bei Chiffre 1 nicht möglich. Auf dieses Problem soll zunächst hingewiesen werden. Ähnlich wie bei Tutorial 1 soll der Verschlüsselungsprozess von Chiffre 2 illustriert werden, um auf die auftretenden internen Zustände während der Verschlüsselung hinzuweisen. Dann muss dem Anwender klar werden, weshalb der Angriff nicht wie bei Chiffre 1 mit einer Verschlüsselungsrunde durchgeführt werden kann. Darauf aufbauend sollen die S-Boxen analysiert und die Differenzenverteilungstabelle eingeführt werden. Für einen Angriff sind dann noch Differenziale und Charakteristiken zu beschreiben. Als Letztes kann illustriert werden, wie Schlüsselinformationen wiederhergestellt werden können.

4.1.3 Tutorial 3

Im letzten Tutorial wird Chiffre 3 mittels differenzieller Kryptoanalyse angegriffen. Durch Tutorial 2 ist der Anwender bereits in der Lage, auch Chiffre 3 erfolgreich zu brechen. In Tutorial 3 sollen weitere Bestandteile beschrieben werden. Dies sind zwei Filterungstechniken, um falsche Nachrichten-Paare zu identifizieren, und eine Berechnungsgrundlage, mit der die notwendige Anzahl an Nachrichten-Paare bestimmt werden kann. Als Letztes soll das Rauschverhältnis erwähnt werden. Da die konkrete Formel und deren Berechnung anspruchsvoll ist, soll dieser Aspekt nur angesprochen werden.

4.2 Aufteilung der Bestandteile der differenziellen Kryptoanalyse

Das Konzept der Tutorials sieht eine möglichst große Aufteilung der einzelnen Schritte vor, die bei einer differenziellen Kryptoanalyse durchgeführt werden

müssen. Diese Aufteilung kann durch die Implementierung mehrerer Komponenten in CT2 realisiert werden. Die zu implementierenden Komponenten und deren Funktion werden im Folgenden beschrieben. Tabelle 6 zeigt die CT2-Komponenten für differenzielle Kryptoanalyse.

Name der Komponente	Funktion
DKA-PfadFinder	Darstellung von Slides, Suche nach Differenzialen
DKA-PfadVisualisierer	visuelle Aufbereitung von Differenzialen
DKA-KeyRecovery	Schlüsselwiederherstellung
DKA-Orakel	Generierung von Klartext-Nachrichten-Paaren
DKA-ToyCipher	Verschlüsselung / Entschlüsselung von Nachrichten

Tabelle 6: *Übersicht über die CT2-Komponenten und deren Funktion*

4.2.1 DKA-PfadFinder

Jedes Tutorial und jeder Angriff wird durch die DKA-PfadFinder-Komponente gesteuert. Eine Präsentation soll die theoretischen Grundlagen zum Verständnis von differenzieller Kryptoanalyse vermitteln. Als zentrale Komponente der Steuerung des Angriffs soll diese Komponente alle notwendigen Parameter an die im Angriffsprozess nachgelagerten Komponenten übermitteln. Notwendige Parameter für die differenzielle Kryptoanalyse, die durch den Anwender zu spezifizieren sind, sind eine Suchstrategie, eine Abbruchstrategie bei der Suche nach Charakteristiken bzw. Differenzialen und die Anzahl der zu generierenden Nachrichten-Paare. Bei den Suchstrategien können eventuell notwendige Schwellwerte zum Abbruch der Suche festgelegt werden. Darüber hinaus gibt es Parameter, die das Verhalten der Komponente selbst steuern. Der Anwender kann auswählen, ob die Komponente im Tutorial-Modus oder im automatischen Modus arbeitet. Im Tutorial-Modus werden vor der praktischen Durchführung die Tutorial-Präsentationen gezeigt. Im automatischen Modus sind keine Eingaben durch den Anwender nötig – der Angriff läuft völlig automatisch ab. Ist weder der Tutorial- noch der automatische Modus angewählt, wird die Präsentation übersprungen und der Anwender kann direkt den Angriff durch Auswahl von S-Boxen in den Verschlüsselungsrunden steuern. Des Weiteren kann vor dem Start festgelegt werden, mit wie vielen Threads die Komponente rechnen kann. Neben der Anzeige der Präsentation berechnet die Komponente die zum Angriff notwendigen Charakteristiken. Diese können aus Gründen der Berechnungsdauer auch aus vorberechneten Daten geladen werden – so kann der Anwender Wartezeiten vermeiden.

DKA-PfadFinder: Suchstrategien

Die Suche nach Charakteristiken erfolgt bei allen drei zur Verfügung stehenden Suchstrategien nach einem ähnlichen Muster. Bei der Betrachtung von Differenzen zweier Blöcke neutralisieren sich die Schlüsseladditionen, und die Permutation kann aufgrund ihrer Linearität bei Differenzen invertiert werden. Die Suche beginnt immer in der aktuell anzugreifenden Verschlüsselungsrunde r in Abhängigkeit der gewählten S-Boxen(en). Die Suche nach Charakteristiken verläuft im Vergleich zur Verschlüsselung „rückwärts“. Es wird jede mögliche Eingangsdifferenz betrachtet und davon ausgehend kann die Schlüsseladdition (linear) übergangen und die Permutation invertiert werden. Dann muss bei der entsprechend anliegenden Ausgabedifferenz der S-Boxen der davorliegenden Verschlüsselungsrunde $(r-1)$ die Differenzenverteilungstabelle der S-Box(en) betrachtet werden. Die dazugehörigen Eingabedifferenzen der $(r-1)$-ten werden anschließend dazu verwendet, um die Suche in diesem Schema fortzusetzen. Dieses Vorgehen wird für jede Verschlüsselungsrunde wiederholt. Abbildung 20 beschreibt dieses allgemeine Vorgehen. Die Eingangsdifferenzen der r-ten Runde entsprechen der (den) erwarteten Differenz(en) bei der Schlüsselwiederherstellung.

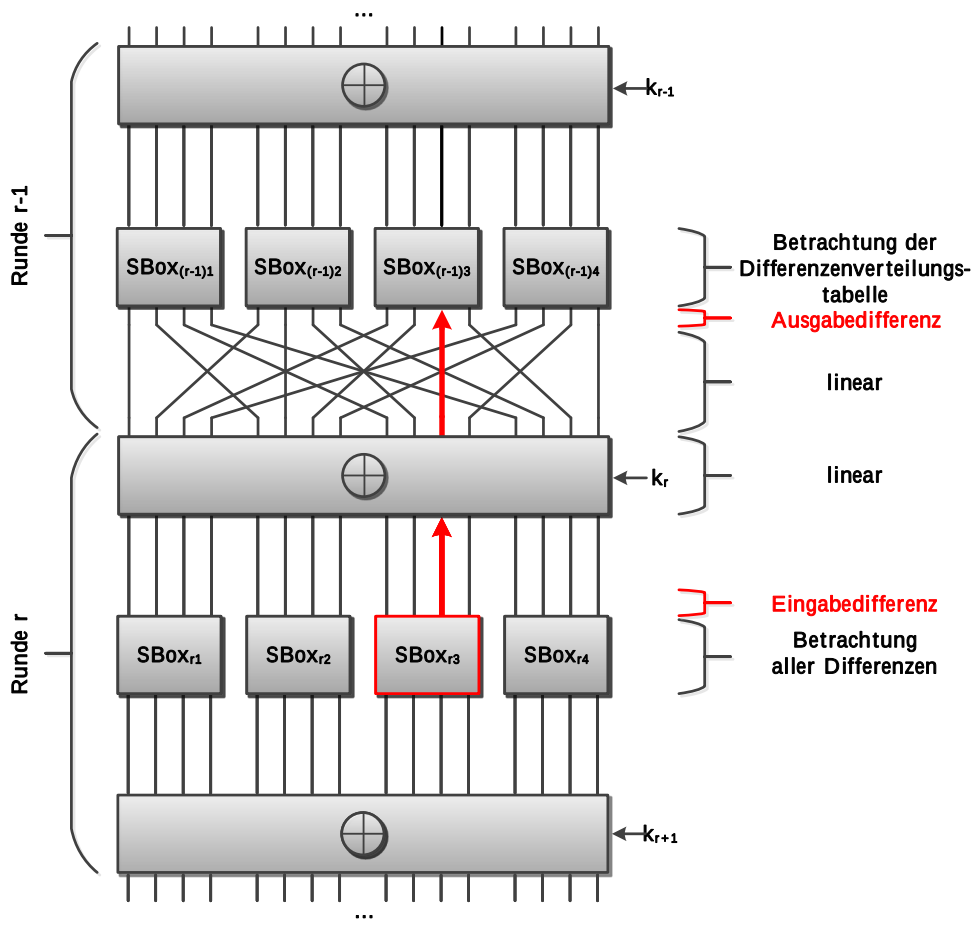


Abbildung 20: *Schematisches Vorgehen bei der Suche nach Charakteristiken*

Im Folgenden werden die vom Benutzer auswählbaren Suchstrategien beschrieben:

- **Beste Charakteristik (Heuristik), dann Differenzialsuche:** Bei der Suche nach Charakteristiken wird bei dieser Suchstrategie zunächst jede mögliche Eingabedifferenz der ausgewählten S-Box(en) der anzugreifenden Verschlüsselungsrunde betrachtet. Bei der Betrachtung der Differenzenverteilungstabelle der vorherigen Verschlüsselungsrunde wird für die Ausgabedifferenz diejenige Eingabedifferenz gewählt, die die höchste Wahrscheinlichkeit hat. Dieses Vorgehen wird für alle vorherigen Verschlüsselungsrunden wiederholt. Aus den resultierenden

Charakteristiken wird die mit der höchsten Wahrscheinlichkeit ausgewählt. Für die Charakteristik mit der höchsten Wahrscheinlichkeit wird anschließend mittels rekursiver Tiefensuche versucht, weitere Charakteristiken mit der spezifischen erwarteten Differenz zu finden. Bei der rekursiven Suche wird aufgrund langer Berechnungszeiten in jeder Rekursionsstufe geprüft, wie hoch die Wahrscheinlichkeit der Charakteristik ist – falls die Wahrscheinlichkeit unter einen zu definierenden Schwellwert sinkt, wird die Suche für diesen Teilbaum abgebrochen. Der Schwellwert kann durch den Anwender spezifiziert werden. Als Vorschlag ist eine Wahrscheinlichkeit von $0,0001$ festgelegt. Diese ist experimentell bestimmt worden und stellt einen Kompromiss aus Berechnungsdauer und sinnvoller Wahrscheinlichkeit für Charakteristiken dar.

- **Beste Charakteristik (Tiefensuche), dann Differenzialsuche:** Bei dieser Suchstrategie wird jede mögliche Eingabedifferenz der ausgewählten S-Box(en) der anzugreifenden Verschlüsselungsrunde betrachtet. Die weitere Suche erfolgt mittels rekursiver Tiefensuche, d. h. es wird jede mögliche Eingabedifferenz zu der spezifischen Ausgabedifferenz in der vorherigen Verschlüsselungsrunde betrachtet. Der Abbruch der Tiefensuche in einem Teilbaum erfolgt entsprechend der Auswahl der Abbruchstrategie. Als Abbruchstrategie ist ein globaler Schwellwert oder ein globales Maximum (in Form einer Charakteristik) wählbar. Der globale Schwellwert ist vom Anwender spezifizierbar. Als Vorschlag ist eine Wahrscheinlichkeit von $0,001$ festgelegt. Nach der Identifizierung der Charakteristik mit der höchsten Wahrscheinlichkeit erfolgt eine rekursive Tiefensuche, um ein Differenzial zu bilden. Diese Suche erfolgt wie in der zuvor beschriebenen Suchstrategie.

- **Alle Charakteristiken mittels Tiefensuche:** Bei dieser Suchstrategie werden zunächst in einem Suchvorgang alle möglichen Charakteristiken mittels Tiefensuche bestimmt. Die Suche erfolgt mittels rekursiver Tiefensuche, welche bei Unterschreitung einer festgelegten Wahrscheinlichkeit abgebrochen wird. Der Schwellwert ist vom Anwender spezifizierbar. Als Vorschlag ist eine Wahrscheinlichkeit von $0,0001$ festgelegt. Anschließend werden die gefundenen Charakteristiken zu Differenzialen strukturiert und als Ergebnis wird das Differenzial verwendet, welches die höchste Wahrscheinlichkeit hat. Diese Suchstrategie hat als Ziel, möglichst viele Charakteristiken zu finden, um ein Differenzial bilden zu können.

Der Grund für die unterschiedlichen Schwellwerte in den Suchstrategien ist,

dass bei der Suche nach der besten Charakteristik aufgrund des größeren Schwellwert früher abgebrochen werden kann. Wenn ein Differenzial gebildet werden soll, kann es sinnvoll sein, auch Charakteristiken zu finden, die eine geringere Wahrscheinlichkeit aufweisen. Durch die Summenbildung im Differenzial kann die daraus resultierende Wahrscheinlichkeit aufgrund der Charakteristiken höher ausfallen und somit ein gutes Ergebnis liefern. Abbildung 21 fasst die möglichen Kombinationen von Such- und Abbruchstrategien zusammen.

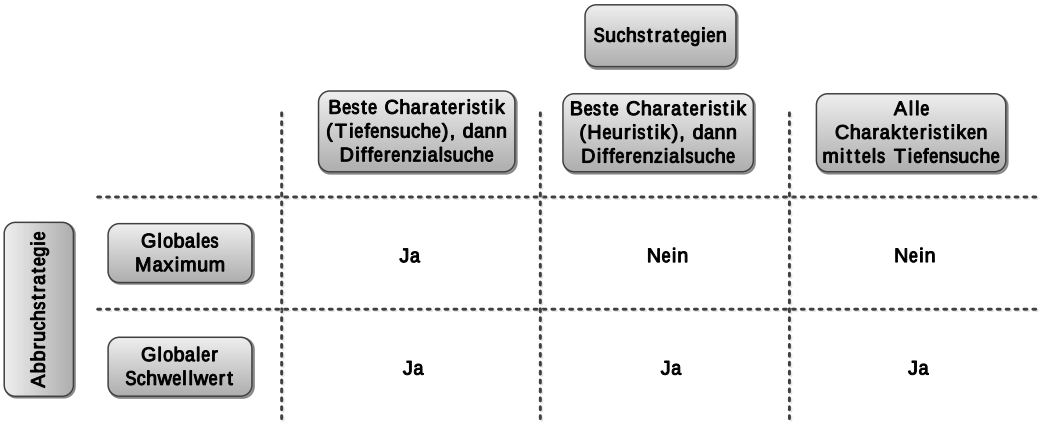


Abbildung 21: *Zusammenfassung der Kombinationsmöglichkeiten von Such- und Abbruchstrategien*

Das heuristische Verfahren basiert im Vergleich zu den zwei anderen Suchstrategien nicht auf rekursiver Tiefensuche. Dadurch wird der Suchraum für Charakteristiken kleiner, da bei der Betrachtung der Differenzenverteilungstabelle lediglich der Eintrag verwendet wird, der die höchste Wahrscheinlichkeit hat.

4.2.2 DKA-PfadVisualisierer

Diese Komponente bietet dem Anwender die Möglichkeit, die Suchergebnisse der DKA-PfadFinder-Komponente zu inspizieren. Dabei liegt der Fokus auf der Darstellung von einzelnen Charakteristiken, die in einem Differenzial enthalten sind. Dieser Schritt ist nicht notwendigerweise durchzuführen, unterstützt allerdings das Verständnis des Anwenders in einem hohen Maß.

4.2.3 DKA-KeyRecovery

In dieser Komponente findet die Wiederherstellung der einzelnen Rundenschlüssel statt. Dabei wird in Abhängigkeit der berechneten Konfiguration der DKA-PfadFinder-Komponente eine entsprechend große Anzahl an möglichen Schlüsselkandidaten getestet. Das Ergebnis der einzelnen Versuche kann durch den Benutzer betrachtet werden. Als Parameter kann der Anwender neben der Anzahl der Threads zur Wiederherstellung auch festlegen, ob die Benutzerschnittstelle in der Wiederherstellungsphase aktualisiert wird oder erst nach Fertigstellung der Berechnung. Die beiden Einstellungen beeinflussen die Dauer der Berechnung.

4.2.4 DKA-Orakel

Die Generierung der Klartext-Nachrichten-Paare wird in der DKA-Orakel-Komponente durchgeführt. Bei dieser Komponente muss die Blockbreite durch den Anwender spezifiziert werden. Ein Bestandteil des Paares wird zufällig gewählt, der andere wird entsprechend der Differenz berechnet. Eine Visualisierung ist für diese Komponente nicht notwendig.

4.2.5 DKA-ToyCipher

Die Verschlüsselung von Nachrichten wird durch die DKA-ToyCipher-Komponente realisiert. Diese Komponente enthält eine visuelle Darstellung der gewählten Chiffre und deren Bestandteile, sodass der Anwender den Verschlüsselungsprozess besser verstehen kann.

4.2.6 Interaktion der Komponenten

Ein wichtiger Aspekt bei der Realisierung ist die Synchronisation der Arbeitsphasen der Komponenten. Eine typische Implementierung eines Verfahrens in CT2 ist die Realisierung durch eine einzelne Komponente, die zustandslos arbeitet. Bei der vorliegenden Aufteilung wird das Verfahren auf verschiedene Komponenten verteilt, die in enger Abstimmung einen einzelnen Schritt durchführen. Darüber hinaus findet die Wiederherstellung der Rundenschlüssel iterativ statt. Dies erschwert die Realisierung durch einen erhöhten Kommunikations- und Synchronisationsbedarf, der bei der Realisierung durch eine einzelne Komponente vollständig entfällt. Konzeptuell wird für die Implementierung eine State-Machine benötigt, die den gesamten Zustand auf Ebene des Verfahrens realisiert. Innerhalb der Komponenten werden einzelne Schritte in Abhängigkeit des gesamten Fortschritts der Wiederherstellung

von Rundenschlüssel wiederholt. Dies muss ebenso auf Komponenten-Ebene durch eine State-Machine realisiert werden.

4.3 CT2-Vorlagen

Für jedes der drei Tutorials wird eine eigene Vorlage erstellt, in der für das Tutorial notwendigen Einstellungen festgesetzt sind. Das betrifft insbesondere die Auswahl des entsprechenden Tutorials bzw. der Chiffre in den Komponenten der Vorlagen. Die Komponenten selbst werden in der Vorlage miteinander durch Verbindungen der Konnektoren verknüpft. Über diese können die Komponenten miteinander kommunizieren und Daten austauschen.

Im Folgenden wird Vorlage schematisch beschrieben. Abbildung 22 illustriert die Vorlage zur differenziellen Kryptoanalyse. Im linken Teil der Abbildung ist durch den Anwender der Schlüssel zu spezifizieren, der mittels differenzieller Kryptoanalyse wiederhergestellt werden soll. Im mittleren Teil der Skizze sind die zuvor beschriebenen Komponenten zu sehen. Diese führen die einzelnen Schritte der differenziellen Kryptoanalyse durch. An den Verbindungen der Komponenten sind die Daten beschrieben, die unidirektional zwischen den Komponenten ausgetauscht werden. Im rechten Bereich der Abbildung befindet sich der Bereich der Ausgabe. An dieser Stelle werden die durch die differenzielle Kryptoanalyse wiederhergestellten Rundenschlüssel ausgegeben.

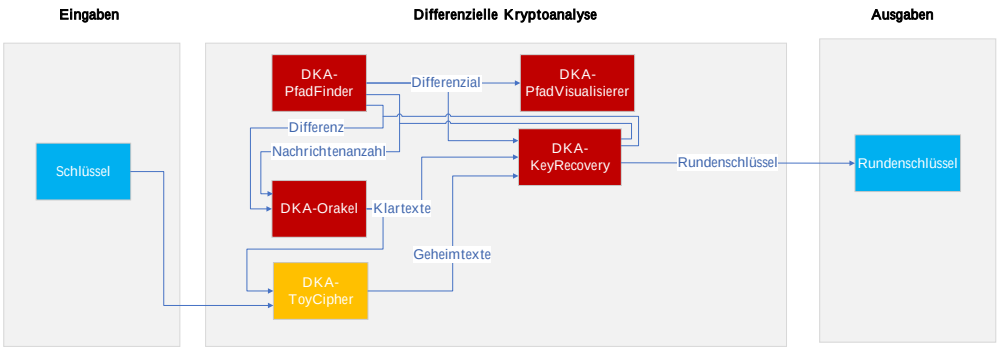


Abbildung 22: *Schematische Darstellung der Vorlage für differenzielle Kryptoanalyse*

In Kapitel 5.2 wird auf die praktische Umsetzung der hier schematisch beschriebenen Vorlage eingegangen.

5 Design und Implementierung der Anwendungen

In Kapitel 4 sind die zu entwickelnden Komponenten auf konzeptueller Ebene beschrieben. In diesem Kapitel wird die Implementierung beschrieben, die das Konzept umsetzen soll. Aufgrund der Komplexität des Vorhabens ist der Angriff der differenziellen Kryptoanalyse in einem ersten Schritt in einer Konsolenanwendung realisiert worden.

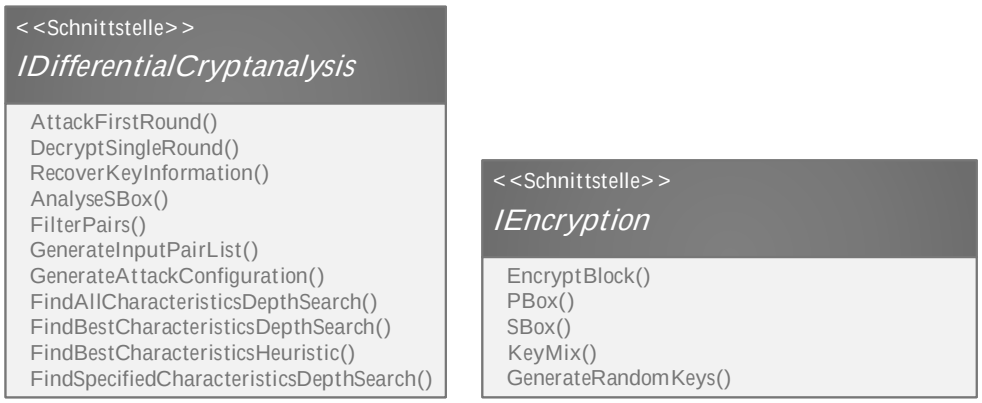
Bei der Entwicklung als prototypische Konsolenanwendung kann der Fokus zunächst vollständig auf eine Implementierung der differenziellen Kryptoanalyse gelegt werden. Durch die Ausblendung von (visuellen) design- und darstellungsspezifischen Anforderungen können große zeitliche Aufwände eingespart werden. Bei dieser Implementierung stehen die Funktionsfähigkeit und Geschwindigkeit im Vordergrund. Es können erste Erfahrungen in der Durchführung von differenzieller Kryptoanalyse gesammelt werden, welche nützlich bei einer Implementierung mit Fokus auf Benutzerfreundlichkeit sein können. Darüber hinaus soll die Softwarearchitektur einen hohen Grad an Wiederverwendung erlauben, sodass weitere Chiffren in weiteren Arbeiten implementiert werden können.

In einem zweiten Entwicklungsschritt wird der Code der Konsolenanwendung in die in Kapitel 4.2 beschriebenen Komponenten in CT2 migriert. Eine CT2-Implementierung hat als primäre Anforderung neben Benutzerfreundlichkeit die Umsetzung des beschriebenen Konzepts. Dabei ist es wichtig, dass die Komponenten visuell ansprechend und attraktiv sind. Die Geschwindigkeit der Berechnungen spielt in CT2 zwar auch eine Rolle, allerdings ist diese Anforderung für eine E-Learning-Software nicht wichtig. Der Wiederverwendungsgrad des Quellcodes sollte auch hier hoch sein, um weitere Chiffren implementieren zu können. Des Weiteren spielt hier die Synchronisation der Komponenten eine große Rolle: Jede der zu implementierenden Komponenten erfüllt eine Teilaufgabe bei der differenziellen Kryptoanalyse, wobei die Teilaufgaben allerdings nacheinander ablaufen müssen – folglich bestehen Abhängigkeiten, die durch eine korrekte Synchronisation sichergestellt werden müssen.

5.1 Konsolenanwendung

Bei der Implementierung der Konsolenanwendung (Kommandozeilenanwendung) spielen zwei Schnittstellen eine zentrale Rolle: IDifferentialCryptanalysis und IEncryption. Diese sind in Abbildung 23 dargestellt. In allen Dia-

grammen ist auf die vollständige Methoden-Signatur und die Beziehungen der Schnittstellen und Klassen untereinander verzichtet worden, um diese auf das Wesentlichste zu reduzieren. Darüber hinaus gibt es weitere (Hilfs-)Klassen und Methoden, die verschiedene unterstützende Funktionen haben. Ziel der Beschreibung ist die Darstellung aller Strukturen, die für die Analyse eine wichtige Rolle spielen.



(a) *IDifferentialCryptanalysis* ***(b)*** *IEncryption*

Abbildung 23: *Darstellung der Schnittstellen (Konsolenanwendung) mittels UML*

Zur Umsetzung einer differenziellen Kryptoanalyse müssen Klassen die Schnittstelle IDifferentialCryptanalysis implementieren. Die darin enthaltenen Methoden werden im Folgenden beschrieben:

- **AttackFirstRound():** Diese Methode realisiert den Angriff auf die ersten beiden Rundenschlüssel k_0 und k_1 einer Chiffre.
- **DecryptSingleRound():** Entschlüsselt eine spezifizierbare Verschlüsselungsrunde mittels gegebenem Rundenschlüssel
- **RecoverKeyInformationen():** Diese Methode realisiert einen Angriff auf eine Verschlüsselungsrunde $r > 1$.
- **AnalyseSBox():** Diese Methode erstellt die Differenzenverteilungstabelle einer S-Box.
- **FilterPairs():** Durch diese Methode können die Eingabe-Paare gefiltert werden.

- **GenerateInputPairList():** Erstellt eine Liste von Klartext-Paaren mit definierbarer Differenz.
- **GenerateAttackConfiguration():** Erstellt eine Angriffs-Konfiguration auf Basis der anzugreifenden Verschlüsselungsrunde und S-Box(en). Darüber hinaus muss eine Such- und (je nach Auswahl eine) Abbruchstrategie spezifiziert werden.
- **FindAllCharackteristicsDepthSearch():** Diese Methode realisiert die Suchstrategie „Alle Charakteristiken mittels Tiefensuche“.
- **FindBestCharacteristicsDepthSearch():** Diese Methode realisiert die Suchstrategie „Beste Charakteristik (Tiefensuche), dann Differenzialsuche“.
- **FindBestCharacteristicsHeuristic():** Diese Methode realisiert die Suchstrategie „Beste Charakteristik (Heuristik), dann Differenzialsuche“.
- **FindSpecifiedCharacteristicsDepthSearch():** Diese Methode sucht nach Charakteristiken mit einer gegebenen Eingabedifferenz für eine spezifische Verschlüsselungsrunde.

Für jede zu implementierende Chiffre muss die Schnittstelle IEncryption implementiert werden. Die darin enthaltenen Methoden sind:

- **EncryptBlock():** Diese Methode verschlüsselt einen Block entsprechend der Verschlüsselungsvorschrift einer Chiffre.
- **PBox():** Diese Methode wendet die Permutation auf einen Block an.
- **SBox():** Diese Methode wendet die S-Box(en) auf einen Block an.
- **KeyMix():** Diese Methode realisiert die Schlüsseladdition mittels XOR-Operation.
- **GenerateRandomKeys():** Diese Methode generiert zufällige Rundenschlüssel für die Chiffre.

Die Charakteristiken einer Chiffre werden durch die abstrakte Characteristics-Klasse (Abbildung 24) modelliert, welche die Schnittstelle IClonable implementiert. Diese enthält die abstrakte Methode *Clone()*, die durch die Subklassen überschrieben werden muss. Diese Methode wird bei der rekursiven Tiefensuche verwendet, um Kopien von Objekten erzeugen zu können.

Abbildung 24: *Darstellung der abstrakten Characteristics-Klassen (Konsolenanwendung) mittels UML*

Zur Verwaltung des Status eines Angriffs mittels differenzieller Kryptoanalyse werden weitere Klassen verwendet, die in Abbildung 25 zu sehen sind. Die Klasse *DifferentialKeyRecoveryAttack* verwaltet alle Informationen eines Angriffs wie beispielsweise die wiederhergestellten Rundenschlüssel. Die Klassen *DifferentialAttackRoundResult* und *DifferentialAttackLastRoundResult* speichern die wiederhergestellten (Teil-)Schlüsselbits eines Angriffs einer Verschlüsselungsrunde. Eine Angriffskonfiguration wird in der Klasse *DifferentialAttackRoundConfiguration* gespeichert – dies sind beispielsweise die anzugreifende(n) S-Box(en) und die Verschlüsselungsrunde.

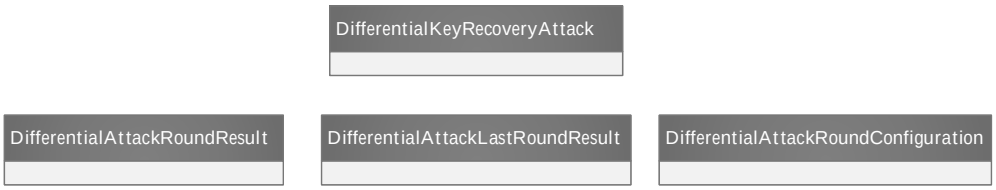


Abbildung 25: *Darstellung einiger Klassen (Konsolenanwendung) mittels UML*

Des Weiteren gibt es drei weitere Klasse (Abbildung 26), die bei der Implementierung notwendig sind.

- **SBoxCharacteristic:** Diese Klasse speichert für eine Ein- und Ausgabedifferenz einer S-Box die Wahrscheinlichkeit.
- **Pair:** Ein Nachrichten-Paar wird durch diese Klasse modelliert.
- **KeyProbability:** Diese Klasse speichert für einen Rundenschlüssel-Kandidat die Anzahl der Treffer durch die Nachrichten-Paare.

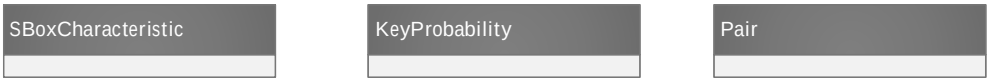


Abbildung 26: *Darstellung weiterer Klassen (Konsolenanwendung) mittels UML*

Die Such- und Abbruchstrategien sind mittels *Enumerationen* (Abbildung 27) realisiert worden. Für die verschiedenen Suchstrategien gibt es die Enumeration „SearchPolicy“ und für die Abbruchstrategien während der Suche die Enumeration „AbortingPolicy“.

Abbildung 27: *Darstellung zweier Enumerationen mittels UML*

Bei der Implementierung der Anwendung gibt es zwei Programmteile, die aufgrund ihrer Struktur und der notwendigen Rechendauer parallelisiert werden. Bei der Suche nach Charakteristiken werden in Abhängigkeiten der Anzahl der gewählten S-Boxen i mit $2^i - 1$ verschiedenen Eingabedifferenzen als Startdifferenz gesucht. Die $2^i - 1$ verschiedenen Eingabedifferenzen können als $2^i - 1$ verschiedene Bäume betrachtet werden, die parallel durchsucht werden können. Die Berechnungen sind disjunkt und damit ohne großen Aufwand parallelisierbar. Listing 1 zeigt eine beispielhafte Implementierung der Suche nach Charakteristiken mittels der Suchstrategie *Beste Charakteristik (Tiefensuche), dann Differenzialsuche*, wobei der Code-Teil die parallele Suche nach der besten Characteristik zeigt. Wichtig hierbei ist der synchronisierte Zugriff auf die Ergebnisliste.

```
1 Parallel.For(1, loopBorder, i =>
2 {
3   Characteristic inputObj = new CipherFourCharacteristic();
4
5   //expected difference
6   int expectedDifference =
7     GenerateValue(roundConfiguration.ActiveSBoxes, i);
```

```
8   int outputDifferencePreviousRound =
9     ReversePBoxBlock(expectedDifference);
10
11  inputObj.InputDifferentials[round] = expectedDifference;
12
13  //start depth-first search
14  Characteristic retVal =
15    FindBestCharacteristic(round, differentialsList,
          outputDifferencePreviousRound, inputObj);
16
17  //check if there is a result
18  if (retVal.Probability != -1)
19  {
20    _semaphoreSlim.Wait();
21    try
22    {
23      resultList.Add(retVal);
24    }
25    finally
26    {
27      _semaphoreSlim.Release();
28    }
29  }
30 });
```

Listing 1: *Beispielhafte Parallelisierung der Suche nach Charakteristiken*

Bei der Wiederherstellung von Schlüsselbits werden in Abhängigkeit der gewählten S-Boxen i genau 2^i verschiedene Schlüsselkandidaten getestet. Dieses Verfahren kann ebenfalls parallelisiert werden. Listing 2 zeigt die beispielhafte Implementierung der parallelen Wiederherstellung von Schlüsseln. Wichtig hierbei ist der synchronisierte Zugriff auf die Ergebnisliste.

```
1  Parallel.For(0, loopBorder, i =>
2  {
3    //guess key cadidate
4    int guessedKey =
5      GenerateValue(configuration.ActiveSBoxes, i);
6
7    //check if we need to permute the value
8    if (!configuration.IsLast)
9    {
10     guessedKey = ApplyPBoxToBlock(guessedKey);
11   }
12
13   KeyProbability curTry = new KeyProbability()
```

```
14      {
15        Counter = 0,
16        Key = guessedKey
17      };
18
19    foreach (var curPair in configuration.FilteredPairList)
20    {
21      Pair encryptedPair = new Pair()
22        {
23          LeftMember = encryption.EncryptBlock(curPair.
              LeftMember),
24          RightMember = encryption.EncryptBlock(curPair.
              RightMember)
25        };
26
27      encryptedPair.LeftMember =
28        PartialDecrypt(attack, encryptedPair.LeftMember);
29      encryptedPair.RightMember =
30        PartialDecrypt(attack, encryptedPair.RightMember);
31
32      //reverse round with the guessed key
33      int leftMemberSingleDecrypted =
34        DecryptSingleRound(encryptedPair.LeftMember, curTry.
            Key, configuration.IsBeforeLast, configuration.
            IsLast);
35      int rightMemberSingleDecrypted =
36        DecryptSingleRound(encryptedPair.RightMember, curTry.
            Key, configuration.IsBeforeLast, configuration.
            IsLast);
37
38      if (configuration.IsLast)
39      {
40        leftMemberSingleDecrypted =
41          ReverseSBoxBlock(leftMemberSingleDecrypted);
42        rightMemberSingleDecrypted =
43          ReverseSBoxBlock(rightMemberSingleDecrypted);
44      }
45      else
46      {
47        leftMemberSingleDecrypted =
48          ReversePBoxBlock(leftMemberSingleDecrypted);
49        leftMemberSingleDecrypted =
50          ReverseSBoxBlock(leftMemberSingleDecrypted);
51
52        rightMemberSingleDecrypted =
53          ReversePBoxBlock(rightMemberSingleDecrypted);
54        rightMemberSingleDecrypted =
55          ReverseSBoxBlock(rightMemberSingleDecrypted);
56      }
```

```
57
58      int differentialToCompare =
59        leftMemberSingleDecrypted ^
              rightMemberSingleDecrypted;
60
61      bool[] conditions = new bool[]{true, true, true, true
            };
62          BitArray diff = GetBitsOfInt(configuration.
                ExpectedDifference);
63          BitArray leftMemberBits = GetBitsOfInt(
                leftMemberSingleDecrypted);
64          BitArray rightMemberBits = GetBitsOfInt(
                rightMemberSingleDecrypted);
65
66          for (int j = 0; j < CipherFourConfiguration.
                SBOXNUM; j++)
67          {
68              if (configuration.ActiveSBoxes[j])
69              {
70                  int shift = 15 << (4 * j);
71                  int bit4Diff = configuration.
                        ExpectedDifference & shift;
72                  int bit4Left = leftMemberSingleDecrypted &
                        shift;
73                  int bit4Right = rightMemberSingleDecrypted
                        & shift;
74
75                  if ((bit4Left ^ bit4Right) == bit4Diff)
76                  {
77                      conditions[j] = true;
78                  }
79                  else
80                  {
81                      conditions[j] = false;
82                  }
83              }
84          }
85
86          if (conditions[0] && conditions[1] && conditions
                [2] && conditions[3])
87          {
88              curTry.Counter++;
89          }
90  }
91
92  //synchronize access to resultList
93  _semaphoreSlim.Wait();
94  try
95  {
```

```
96        roundResult.KeyCandidateProbabilities.Add(curTry);
97      }
98      finally
99      {
100       _semaphoreSlim.Release();
101     }
102 });
```

Listing 2: *Beispielhafte Parallelisierung der Wiederherstellung von Schlüsselbits*

5.2 Komponenten für CT2

In diesem Abschnitt wird die Implementierung der differenziellen Kryptoanalyse in CT2 beschrieben. Jede der zu entwickelnden Komponenten implementiert als CT2-Plugin die Schnittstelle *ICrypComponent*. ICryptComponent gibt u. a. die Lebenszyklus-Callbacks vor. In den folgenden Abschnitten wird jede Komponente beschrieben, die zur Realisierung einer differenziellen Kryptoanalyse unter Berücksichtigung des beschriebenen Konzept implementiert wurden. Im letzten Abschnitt werden die Vorlagen präsentiert.

Ein wichtiger Aspekt der Implementierung ist die Synchronisation und Abstimmung der Komponenten. Der Lebenszyklus der meisten Komponenten in CT2 ist zustandslos und führt ein Krypto-Verfahren durch Ausführung der Execute-Methode (siehe Kapitel 2.1.4) aus. Diese Methode wird ausgeführt, sobald an allen verbundenen Eingängen Daten anliegen. Bei den Eingängen wird zwischen notwendigen und nicht notwendigen Eingängen unterschieden. Die notwendigen Eingänge müssen zur Ausführung der Execute-Methode verbunden sein. Da bei der differenziellen Kryptoanalyse rundenbasiert vorgegangen wird, ist es notwendig, innerhalb der DKA-PfadFinder- und DKA-KeyRecovery-Komponenten den Zustand über State-Machines zu verwalten. Das Brechen einer einzelnen Verschlüsselungsrunde ist abhängig von der Anzahl an angegriffenen S-Boxen – diese Auswahl kann vom Nutzer vorgenommen werden. Folglich müssen die Komponenten den gesamten Angriffsverlauf zustandsbasiert abbilden. Die Komponenten DKA-PfadVisualisierer, DKA-Orakel und DKA-ToyCipher benötigen keine Zustandsverwaltung, diese reagieren zustandslos auf eingehende Daten.

5.2.1 DKA-PfadFinder

Die DKA-PfadFinder-Komponente steuert den gesamten Ablauf des Angriffs. Diese dient als Startpunkt für alle Aktivitäten. Im Tutorial-Modus zeigt die Komponente eine zum ausgewählten Tutorial passende Präsentation, welche

mittels WPF-Steuerelementen realisiert ist. Abbildung 28 zeigt die visuelle Darstellung der Komponente mit der ersten Seite der Präsentation.

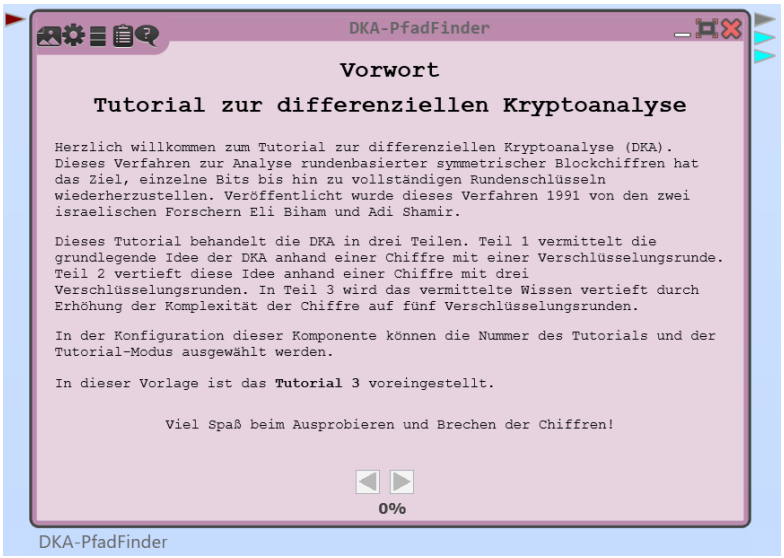


Abbildung 28: *Visuelle Darstellung der DKA-PfadFinder-Komponente*

Es gibt einen boolean-Eingang bei der DKA-PfadFinder-Komponente. Dieser zeigt an, wenn ein Berechnungsschritt der DKA-KeyRecovery-Komponente abgeschlossen ist. Als Ausgang gibt es ein Differenzial, eine Nachrichtenanzahl sowie die Differenz der Nachrichten. Abbildung 29 zeigt die konfigurierbaren Einstellungen dieser Komponente (diese Einstellungen wurden in Kapitel 4.2 erläutert). Mit „konfigurierbar“ ist hier gemeint, dass sowohl der Benutzer eine Auswahl treffen kann, aber auch, dass die Auswahl des Tutorials die folgenden Optionen passend verändert (so gibt es die Optionen zur Geschwindigkeit und zur DKA in Tutorial 2 und 3, aber nicht in Tutorial 1).

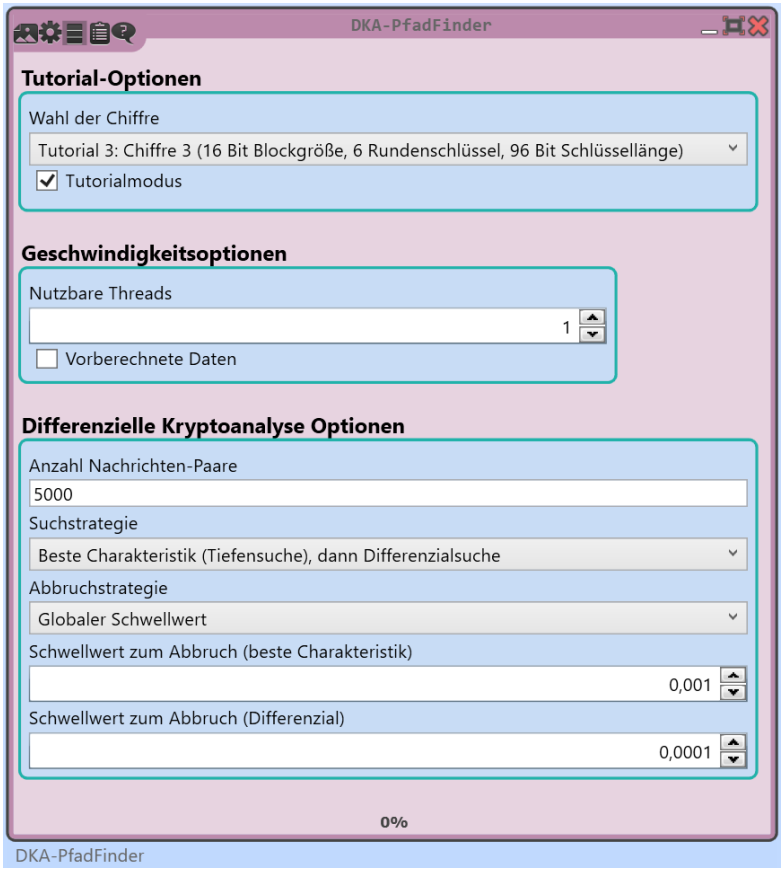


Abbildung 29: *Einstellungen der DKA-PfadFinder-Komponente*

Zur Realisierung der Suche nach Charakteristiken muss die Schnittstelle IPathFinder (Abbildung 30) für die verschiedenen Chiffren implementiert werden. Diese orientiert sich an der Schnittstelle ICryptAnalysis der Konsolenanwendung und enthält alle notwendigen Methoden, die durch das Plugin aufgerufen werden können. Die Methodennamen und deren Funktionalität entsprechen denen aus Kapitel 5.1.

Abbildung 30: *Darstellung der Schnittstelle IPathFinder (CT2-Komponente)*

Die State-Machine wird innerhalb der DKA-PfadFinder-Komponente durch einen separaten Thread ausgeführt, welcher durch mehrere Instanzen der Klasse *AutoResetEvent* gesteuert wird. Diese erlauben es, den ausführenden Thread an definierten Abschnitten zu blockieren und zu einem späteren Zeitpunkt wieder freizugeben. Der Zustand des Threads wird innerhalb der Methode *ExecuteDifferentialAttack()* in der CT2-Plugin-Klasse verwaltet. Beim Start der Komponente wird der Thread in der PreExecute-Methode vorbereitet und erstellt (Listing 3).

```
1  public void PreExecution()
2  {
3    //Code...
4
5    //prepare thread to run the state machine
6    ThreadStart tStart = new ThreadStart(
         executeDifferentialAttack);
7    _workerThread = new Thread(tStart);
8    _workerThread.Name = "DCA-PathFinder Workerthread";
9    _workerThread.IsBackground = true;
10 }
```

Listing 3: *Reduzierte Darstellung der PreExecute-Methode der DKA-PfadFinder-Komponente*

Sobald die Execute-Methode das erste Mal ausgeführt wird, wird der Thread der State-Machine gestartet. Je nach gewählter Chiffre und aktuellem Fortschritt des Angriffs werden dem Anwender entsprechende Benutzersteuerelemente angezeigt. Das Blockieren und weitere Ausführen des Threads wird durch Eingaben des Anwenders gesteuert. Den Zustand des Angriffs kapselt

eine Instanz einer von DifferentialKeyRecoveryAttack (siehe Kapitel 5.1) abgeleiteten Klasse. In Listing 4 ist die Implementierung der Execute-Methode in reduzierter Form dargestellt.

```
1  public void Execute()
2  {
3    //start thread with state machine if its unstarted
4    if ((_workerThread.ThreadState & ThreadState.Unstarted)
         == ThreadState.Unstarted)
5    {
6      workerThread.Start();
7    }
8
9    //prepare handles for pausing the state machine
10   WaitHandle[] waitHandles = new WaitHandle[]
11   {
12     _activePresentation.sendDataEvent
13   };
14
15   switch (settings.CurrentAlgorithm)
16   {
17     case Algorithms.Cipher1:
18     {
19       //Code...
20     }
21     break;
22     case Algorithms.Cipher2:
23     {
24       //Code...
25     }
26     break;
27     case Algorithms.Cipher3:
28     {
29       Cipher3DifferentialKeyRecoveryAttack c3Attack =
30         _differentialKeyRecoveryAttack as
              Cipher3DifferentialKeyRecoveryAttack;
31
32       //Attack Round 5
33       if (!c3Attack.recoveredSubkey5)
34       {
35
36       }
37       else if (!c3Attack.recoveredSubkey4)
38       {
39         //Code...
40       }
41       else if (!c3Attack.recoveredSubkey3)
42       {
```

```
43          //Code...
44        }
45        else if (!c3Attack.recoveredSubkey2)
46        {
47          //Code...
48        }
49        else
50        {
51          //Code...
52        }
53      }
54      break;
55    }
56  }
```

Listing 4: *Reduzierte Darstellung der Execute-Methode der DKA-PfadFinder-Komponente*

Die Implementierung der State-Machine ist in Listing 5 dargestellt. Sobald der Thread zur Ausführung freigegeben wurde (durch Benutzereingaben), wird entsprechend des Fortschritts und der durch den Anwender spezifizierten Einstellungen nach Charakteristiken gesucht. Diese werden serialisiert an die nachfolgenden Komponenten übertragen.

```
1  private void ExecuteDifferentialAttack()
2  {
3    switch (settings.CurrentAlgorithm)
4    {
5      case Algorithms.Cipher1:
6      {
7        //Code...
8      }
9      break;
10     case Algorithms.Cipher2:
11     {
12       //Code...
13     }
14     break;
15     case Algorithms.Cipher3:
16     {
17       if (!settings.AutomaticMode)
18       {
19         //break round 5
20         while (!c3Attack.recoveredSubkey5)
21         {
22           //Code...
```

```
23          }
24
25          //break round 4
26          while (!c3Attack.recoveredSubkey4)
27          {
28            //Code...
29          }
30
31          //break round 3
32          while (!c3Attack.recoveredSubkey3)
33          {
34            //Code...
35          }
36
37          while (!c3Attack.recoveredSubkey2)
38          {
39            //Code...
40          }
41
42          //break last round
43          //Code...
44        }
45        else
46        {
47          //break round 5 run sbox1
48          //Code...
49
50          //break round 5 run sbox2
51          //Code...
52
53          //break round 5 run sbox3
54          //Code...
55
56          //break round 5 run sbox4
57          //Code...
58
59          //break round 4 - 1
60          //Code...
61        }
62      }
63      break;
64    }
65 }
```

Listing 5: *Reduzierte Darstellung der ExecuteDifferentialAttack-Methode der DKA-PfadFinder-Komponente*

Bei Beendigung der Ausführung des Workspaces muss der Thread der State-

Machine korrekt beendet werden. Dazu wird eine boolean-Variable auf $false$ in der Stop-Methode (Listing 6) gesetzt – dieser Wert wird vor der Ausführung eines nächsten Schrittes in der State-Machine geprüft. Nachdem der Wert der Variable gesetzt wurde, werden die Instanzen des AutoResetEvents gesetzt, sodass der blockierte Thread fortgesetzt werden kann.

```
1  public void Stop()
2  {
3    _stop = true;
4
5    //Code...
6
7    _nextRound.Set();
8    _activePresentation.sendDataEvent.Set();
9    _activePresentation.workDataEvent.Set();
10
11   if ((_workerThread.ThreadState & ThreadState.Unstarted)
        != ThreadState.Unstarted)
12   {
13     _workerThread.Join();
14   }
15 }
```

Listing 6: *Reduzierte Darstellung der Stop-Methode der DKA-PfadFinder-Komponente*

Ein wichtiger Bestandteil der Präsentation (Abbildung 31) ist die anschauliche Visualisierung für den Anwender. Dieser kann individuell über Buttons in der Benutzerschnittstelle den Fortschritt festlegen. Einzelne Kapitel können bei Bedarf über Buttons wiederholt oder übersprungen werden.

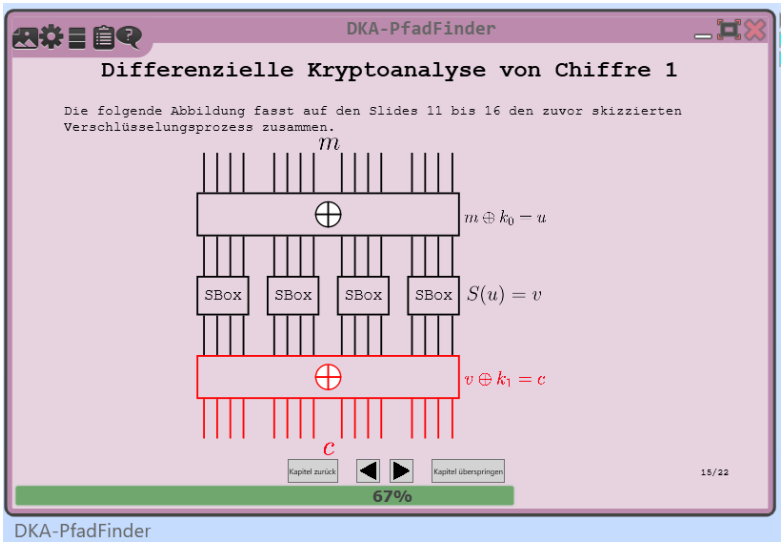


Abbildung 31: *Darstellung der Präsentation der DKA-PfadFinder-Komponente*

Der Angriff auf eine Chiffre kann durch den Anwender individuell gesteuert werden (siehe Abbildung 32). Dazu können in den Benutzersteuerelementen S-Boxen in der entsprechenden Verschlüsselungsrunde ausgewählt werden. Die aktuell anzugreifende(n) S-Box(en) können markiert werden, diese werden dann rot hervorgehoben. Ist der Angriff auf eine Verschlüsselungsrunde noch nicht abgeschlossen, werden die bereits zuvor angegriffenen S-Boxen grün markiert. S-Boxen, die noch nicht angegriffen wurden, werden in einem blassen Grau dargestellt.

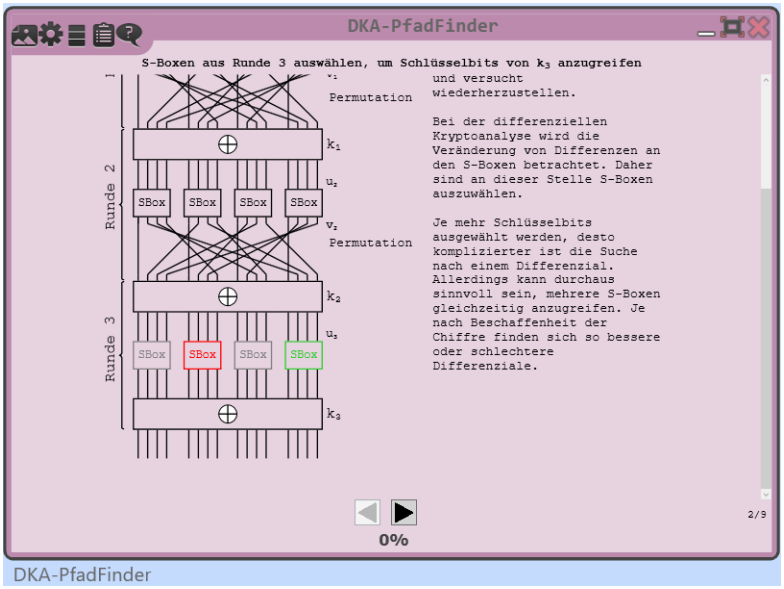


Abbildung 32: *Darstellung des Angriffs der DKA-PfadFinder-Komponente*

5.2.2 DKA-PfadVisualisierer

Die Darstellung eines Differenzials in der DKA-PfadVisualisierer-Komponente besteht aus zwei Teilen. In Abbildung 33 ist die Komponente dargestellt. Das WPF-Benutzersteuerelement enthält eine Navigation mittels Reiter. Auf dem Reiter „Charakteristik-Tabelle“ ist eine zusammenfassende Darstellung des anzuzeigenden Differenzials zu sehen. Auf dem Reiter „Charakteristik-Visualisierung“ kann eine einzelne Charakteristik durch den Anwender visuell inspiziert werden. Der DKA-PfadVisualisierer hat einen Eingang und keinen Ausgang. Als Eingang erhält die Komponente das anzuzeigende Differenzial.

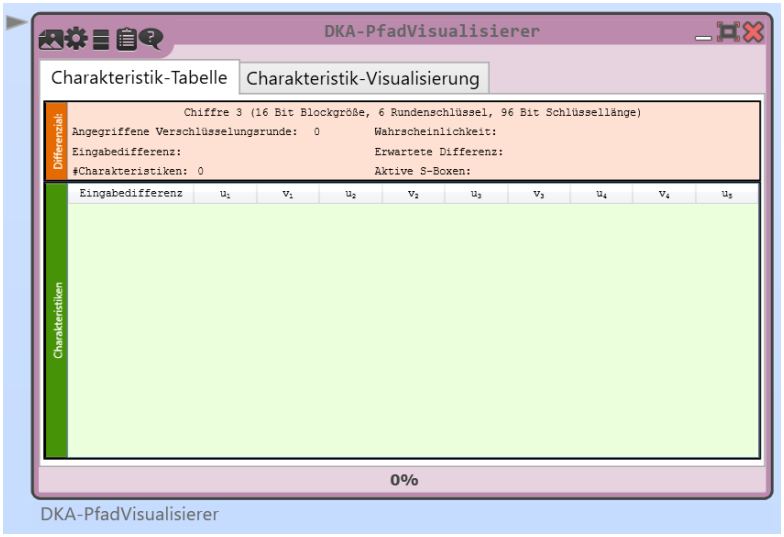


Abbildung 33: *Visuelle Darstellung der DKA-PfadVisualisierer-Komponente*

Abbildung 34 zeigt die Einstellungsmöglichkeiten der Komponente. Bei der DKA-PfadVisualiserer-Komponente kann die verwendete Chiffre ausgewählt werden.

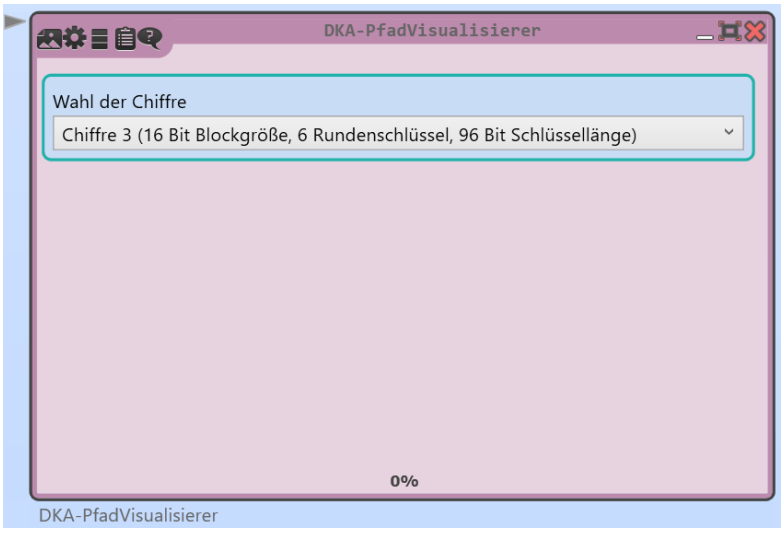


Abbildung 34: *Einstellungen der DKA-PfadVisualisierer-Komponente*

In Abbildung 35 ist ein Differenzial in der zusammenfassenden Darstellung zu sehen. Der obere Bereich zeigt unter anderem die Wahrscheinlichkeit, die Eingabedifferenz und die erwartete Differenz an.

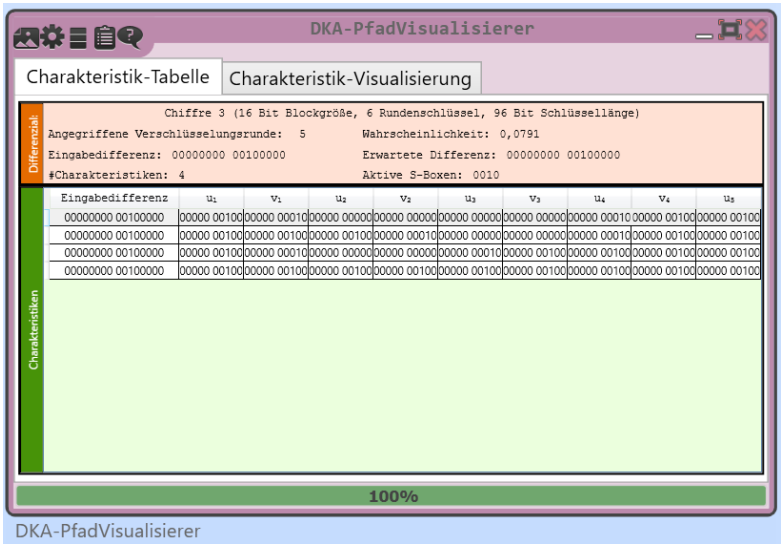


Abbildung 35: *Darstellung eines Differenzials in der DKA-PfadVisualisierer-Komponente*

Die einzelnen Charakteristiken, die in einem Differenzial enthalten sind, können auf dem Reiter „Charakteristik-Visualisierung“ graphisch betrachtet werden (Abbilding 36). Dazu muss die zu betrachtende Charakteristik markiert werden (einfacher Klick oder Doppelklick mittels Maus). Einzelne Bits der Charakteristik sind rot dargestellt. Die einzelnen Differenzen, die innerhalb der Charakteristik auftreten, sind im rechten Bereich der Ansicht binär abgebildet.

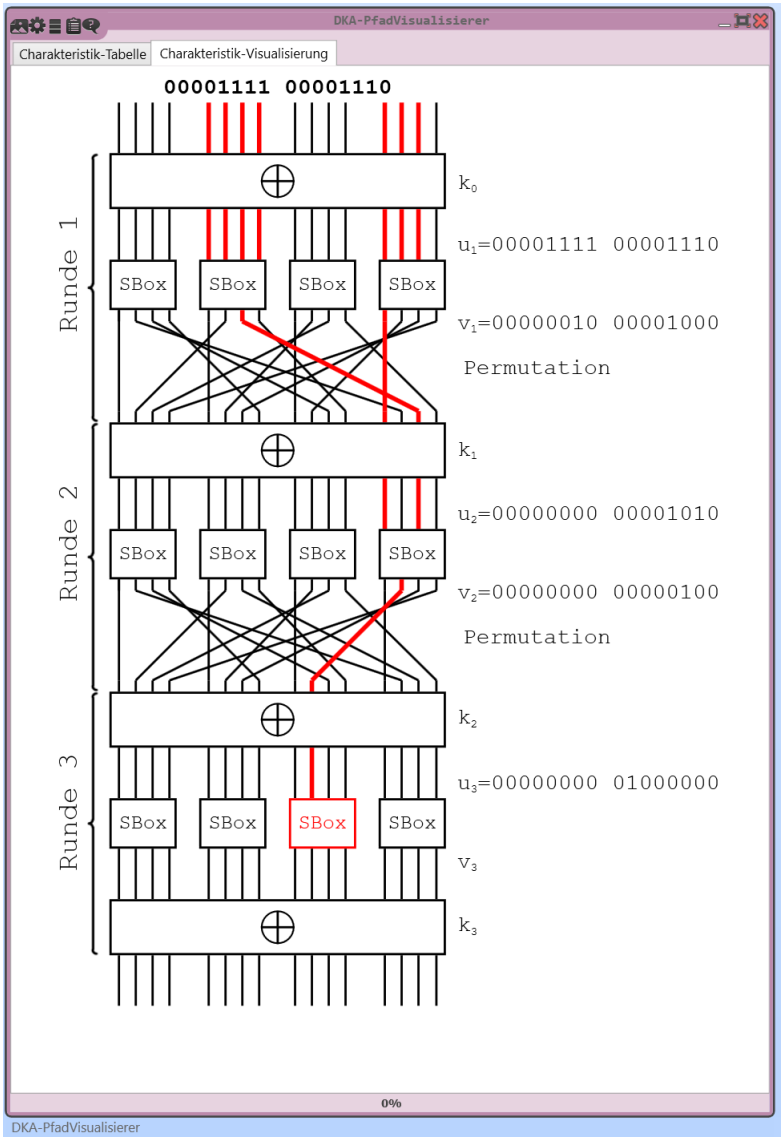


Abbildung 36: *Darstellung einer Charakteristik in der DKA-PfadVisualisierer-Komponente*

5.2.3 DKA-KeyRecovery

Die DKA-KeyRecovery-Komponente hat in der visuellen Darstellung eine Reiter-Navigation (Abbildung 37), über welche der Status der aktuell angegriffenen Verschlüsselungsrunde (Reiter „Rundenstatus“) und der Gesamtstatus des Angriffs dargestellt werden (Reiter „Gesamtstatus“). Als Eingang gibt es ein Differenzial, die Klartext- und Geheimtext-Nachrichten-Paare. Die

Komponente hat vier Ausgänge. Diese sind ein boolean (zeigt die Beendigung der Berechnung an), die Rundenschlüssel sowie eine Nachrichtenanzahl und Differenz – die letzten zwei Ausgänge dienen dem anfordern weiterer Nachrichten-Paare (falls nötig).

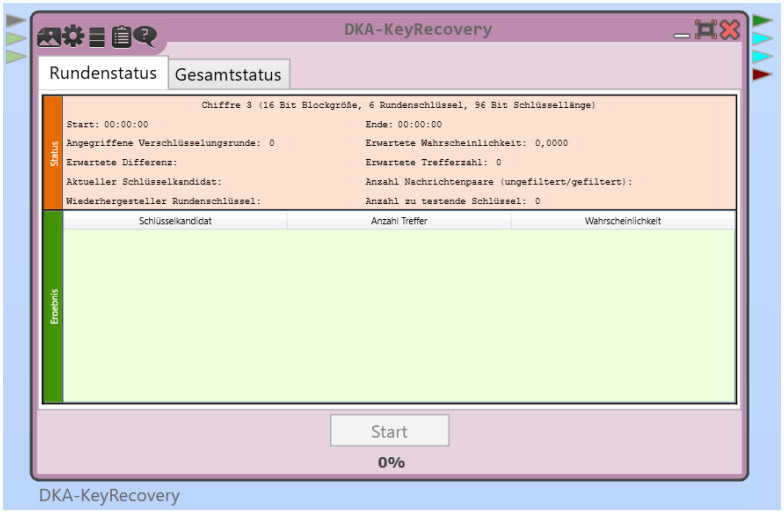


Abbildung 37: *Visuelle Darstellung der DKA-KeyRecovery-Komponente*

Die konfigurierbaren Einstellungen sind in Abbildung 38 dargestellt. Die Geschwindigkeitsoptionen dienen dem Anwendungskomfort. Es kann die Anzahl der nutzbaren Threads zum Rechnen, der automatische Modus und ob die Benutzerschnittstelle während Berechnungen aktualisiert werden darf konfiguriert werden. Insbesondere die letzte Einstellung beeinflusst die Rechenlast durch die Komponente, denn häufige Aktualisierungen der Benutzerschnittstelle unterbrechen die Durchführung des Angriffs. Ist diese Einstellung gesetzt, wird der Angriff ohne Unterbrechung durchgeführt und die Ergebnisse werden nach Abschluss dargestellt. Beim automatischen Modus muss der Benutzer keine Aktion mehr in der Komponente durchführen.

Abbildung 38: *Darstellung der Einstellungen der DKA-KeyRecovery-Komponente*

Für jede Chiffre muss die Schnittstelle IKeyRecovery implementiert werden. Diese orientiert sich ebenfalls an der Schnittstelle ICryptAnalysis der Konsolenanwendung. Die Methodennamen und deren Funktionalität entsprechen daher denen aus Kapitel 5.1.

Abbildung 39: *Darstellung der Schnittstelle IKeyRecovery (CT2-Plugin) mittels UML*

Strukturell ist die Implementierung dieser Komponente ähnlich zu der der DKA-PfadFinder-Komponente. Die State-Machine ist in der Methode *ExecuteDifferentialAttack()* implementiert, welche durch einen eigenen Thread ausgeführt wird. Die Synchronisation wird durch eine Instanz der Klasse *AutoResetEvent* gesteuert. Sobald die Execute-Methode des Plugins das erste

Mal ausgeführt wird, wird der Thread der State-Machine gestartet. Des Weiteren muss bei der Ausführung der Execute-Methode stets geprüft werden, ob an allen Eingängen der Komponente neue Daten anliegen. Bei Änderung eines Eingangs wird die Execute-Methode neu ausgeführt – dies ist allerdings nur sinnvoll, wenn sich alle Eingänge verändert haben. Ein Durchlauf der Execute-Methode macht nur Sinn, wenn neue Klartext-Paare, Geheimtext-Paare und eine neue Konfiguration an den Eingängen anliegen. In Listing 7 ist die Implementierung der Execute-Methode in reduzierter Form zu sehen.

```
1  public void Execute()
2  {
3    //Is the statemachine running?
4    if ((_workerThread.ThreadState & ThreadState.Unstarted)
         == ThreadState.Unstarted)
5    {
6      _workerThread.Start();
7    }
8
9    //check for new inputs
10   if (!_hasNewPlaintextPairs || !_hasNewCipherTextPairs)
11   {
12     return;
13   }
14   else
15   {
16     _hasNewPlaintextPairs = false;
17     _hasNewCipherTextPairs = false;
18
19     List<Pair> plainTextPairs = readPlaintextPairs();
20     List<Pair> cipherTextPairs = readCipherTextPairs();
21
22     switch (_currentAlgorithm)
23     {
24       case Algorithms.Cipher1:
25       {
26         //Code...
27       }
28       break;
29       case Algorithms.Cipher2:
30       {
31         //Code...
32       }
33       break;
34       case Algorithms.Cipher3:
35       {
36         Cipher3DifferentialKeyRecoveryAttack c3Attack =
               attack as Cipher3DifferentialKeyRecoveryAttack;
```

```
37
38          if (c3Attack.recoveredSubkey5 && c3Attack.
                recoveredSubkey4 &&
39            c3Attack.recoveredSubkey3 && c3Attack.
                 recoveredSubkey2)
40          {
41            //Prepare for last round
42          }
43          else
44          {
45            //Prepare for rounds r > 1
46          }
47        }
48        break;
49      }
50    }
51  }
```

Listing 7: *Reduzierte Darstellung der Execute-Methode der DKA-KeyRecovery-Komponente*

Die Implementierung der State-Machine ist in Listing 8 zu sehen. Den Status verwaltet eine Instanz einer von DifferentialKeyRecoveryAttack (siehe Kapitel 5.1) abgeleiteten Klasse. In Abhängigkeit der gewählten Chiffre werden sukzessive die einzelnen Rundenschlüssel wiederhergestellt.

```
1  private void ExecuteDifferentialAttack()
2  {
3    switch (_currentAlgorithm)
4    {
5      case Algorithms.Cipher1:
6      {
7        //Attack cipher 1
8      }
9      break;
10     case Algorithms.Cipher2:
11     {
12       //Attack cipher 2
13     }
14     break;
15     case Algorithms.Cipher3:
16     {
17       Cipher3DifferentialKeyRecoveryAttack c3Attack =
             attack as Cipher3DifferentialKeyRecoveryAttack;
18
19       //attack k5
20       while (!c3Attack.recoveredSubkey5)
```

```
21          {
22            //Code...
23          }
24
25          //attack k4
26          while (!c3Attack.recoveredSubkey4)
27          {
28            //Code...
29          }
30
31          //attack k3
32          while (!c3Attack.recoveredSubkey3)
33          {
34            //Code...
35          }
36
37          //attack k2
38          while (!c3Attack.recoveredSubkey2)
39          {
40            //Code...
41          }
42
43          //attack last round
44          //Code...
45        }
46        break;
47      }
48    }
```

Listing 8: *Reduzierte Darstellung der ExecuteDifferentialAttack-Methode der DKA-KeyRecovery-Komponente*

Ein wichtiger Bestandteil einer Komponente ist der geregelte Abbruch bei der Ausführung (Listing 9). Damit der ausführende Thread der State-Machine korrekt beendet werden kann, muss zunächst eine boolean-Variable auf *true* gesetzt werden. Diese Variable wird bei der Ausführung des Threads zyklisch geprüft, um festzustellen, ob Berechnungen beendet werden sollen. Anschließend wird der Thread, der zu diesem Zeitpunkt möglicherweise blockiert ist, zur Ausführung freigegeben. Als Letztes wird auf die Beendigung des Threads gewartet.

```
1 public void Stop()
2 {
3   _stop = true;
4
5   //Code...
```

```
6
7     _pres.StartClickEvent.Set();
8     _nextStep.Set();
9
10    if ((_workerThread.ThreadState & ThreadState.Unstarted)
          != ThreadState.Unstarted)
11    {
12      _workerThread.Join();
13    }
14  }
```

Listing 9: *Reduzierte Darstellung der Stop-Methode der DKA-KeyRecovery-Komponente*

5.2.4 DKA-Orakel

Die DKA-Orakel-Komponente erzeugt einen zufällig gewählten Block fester Größe und generiert entsprechend einer vorgegebenen Differenz einen zweiten Block. Die zufälligen Blöcke werden mittels einer Instanz der Klasse *Random* erzeugt. Diese Komponente verfügt über keine visuelle Ansicht, folglich wird diese in CT2 als Icon im Workspace dargestellt (Abbildung 40).

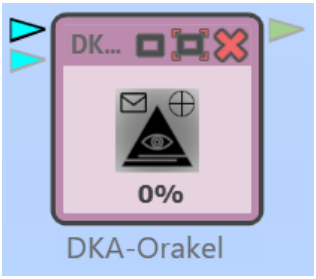


Abbildung 40: *Darstellung der ikonifizierten DKA-Orakel-Komponente*

Als Eingaben erhält die Komponente die Anzahl der Nachrichten-Paare sowie die Differenz der Nachrichten eines Paares. Als Ausgabe werden die Nachrichten-Paare ausgegeben. In den Einstellungen der Komponente kann die Blockgröße spezifiziert werden (Abbildung 41).

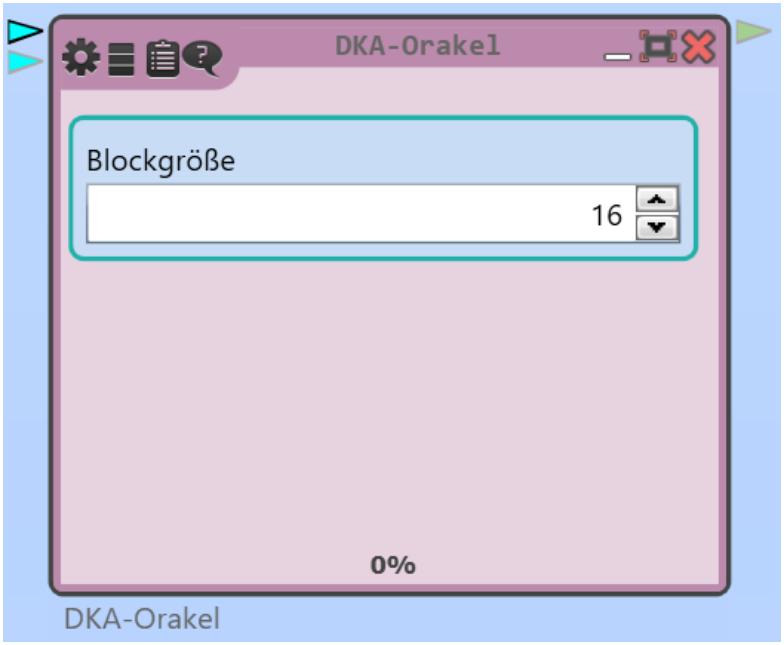


Abbildung 41: *Darstellung der Einstellungen der DKA-Orakel-Komponente*

5.2.5 DKA-ToyCipher

Die DKA-ToyCipher-Komponente kann Blöcke ver- und entschlüsseln. Es gibt eine Reiter-Navigation, die einerseits die Chiffre grafisch darstellt (Reiter „Chiffre“) und andererseits in dem Reiter „Details“ die einzelnen Elemente der Chiffre und deren Funktionsweise erläutert (Abbildung 42). Als Eingabe gibt es Klar- oder Geheimtext(e) und den Schlüssel. Als Ausgabe gibt es (entsprechend den Einstellungen) Klar- oder Geheimtext(e). Die DKA-ToyCipher-Komponente verschlüsselt Blöcke im CBC-Betriebsmodus.

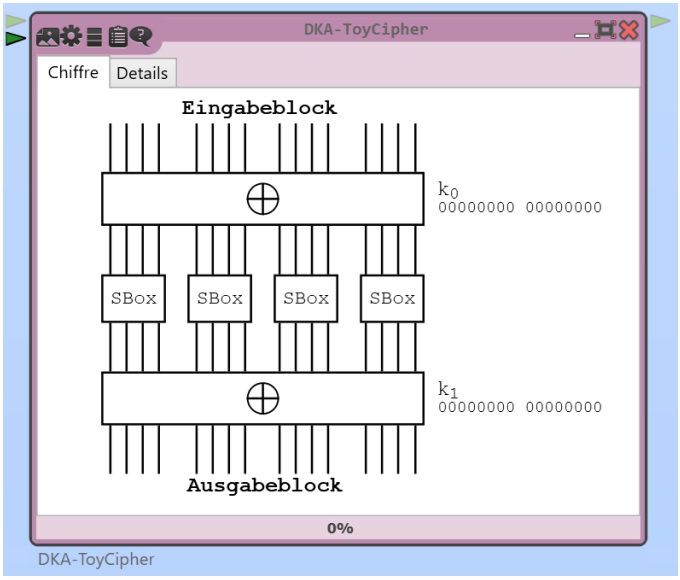


Abbildung 42: *Visuelle Darstellung der DKA-ToyCipher-Komponente*

Die Einstellungen der Komponente sind in Abbildung 43 zu sehen. Es können der Betriebsmodus (Ver- oder Entschlüsselung) und eine Chiffre ausgewählt werden.

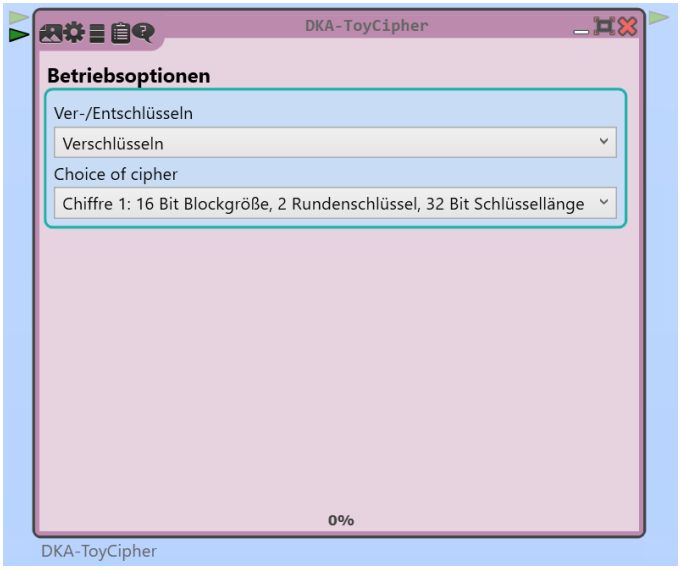


Abbildung 43: *Darstellung der Einstellungen der DKA-ToyCipher-Komponente*

5.2.6 DKA-Workspace

In Kapitel 4.3 ist das Konzept des Workspaces vorgestellt worden. In diesem Abschnitt wird die Umsetzung des Konzepts innerhalb des CT2 beschrieben. Es gibt für jedes Tutorial eine eigene Vorlage, deren Konzept und Implementierung bereits gezeigt wurden. Abbildung 44 zeigt die CT2-Implementierung der Vorlage.

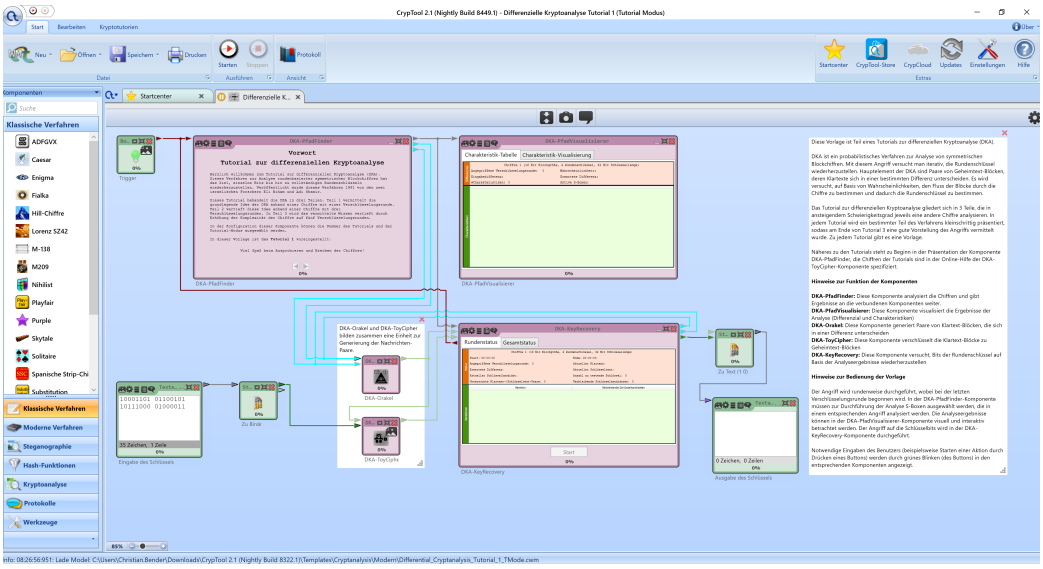


Abbildung 44: *Darstellung der Vorlage mit allen DKA-Komponenten*

Die Vorlagen zu den Tutorials unterscheiden sich nur durch die gesetzten Einstellungen in den Komponenten. Für jedes entsprechende Tutorial sind die Konfigurationen der Komponenten gesetzt (z. B. Tutorial 1). Des Weiteren enthält jede Vorlage im rechten Teil der Vorlage ein Textfeld, in dem einige Anmerkungen stehen. Der Anwender findet dort eine kurze Einführung zur differenziellen Kryptoanalyse, eine kurze Beschreibung der Komponenten und eine Bedienungsanleitung zur Vorlage. Im mittleren Teil der Vorlage ist um die DKA-Orakel und DKA-ToyCipher ebenfalls ein Rahmen in Form eines Textfeldes zu sehen. Dies unterstreicht visuell, dass die beiden Komponenten sehr eng miteinander zusammenarbeiten und man sich diese grundsätzlich als Einheit vorstellen kann. Aufgrund der Komplexität der differenziellen Kryptoanalyse sind die einzelnen Schritte in verschiedene Komponenten aufgeteilt worden, um diese kleinschrittiger darstellen zu können. Darüber hinaus gibt es zu jeder Chiffre eine weitere Vorlage, allerdings ist in dieser der Tutorialmodus nicht aktiviert, sodass der Anwender direkt mit dem Angriff beginnen

kann. Insgesamt gibt es also sechs Vorlagen.

Im Folgenden wird die Vorlage erläutert. Im linken unteren Bereich kann der Anwender den Schlüssel für die DKA-ToyCipher-Komponente spezifizieren, der mittels differenzieller Kryptoanalyse wiederhergestellt werden soll. Links oben befindet sich eine boolean-Komponente, die den initialen Startwert für die DKA-PfadFinder-Komponente bereitstellt – ohne diesen würde die Execute-Methode nicht zur Ausführung kommen. Ist der Workspace gestartet, kann mit der Präsentation in der DKA-PfadFinder-Komponente individuell begonnen werden. Zur Verfügung stehen vier Buttons, um einen Schritt vor- oder zurückzugehen oder um ein Kapitel zurückzugehen oder zu überspringen. Bei der Durchführung des Angriffs müssen in den Verschlüsselungsrunden $r > 1$ S-Boxen markiert werden, um Schlüsselbits anzugreifen. Ist die Suche nach Charakteristiken abgeschlossen, werden die Daten zur den verbundenen Komponenten übertragen.

In der DKA-PfadVisualisierer-Komponente kann der Anwender das Differenzial betrachten und die Charakteristiken visuell inspizieren. Dieser Schritt ist optional und dient primär dazu, die Struktur des Differenzials besser zu verstehen.

In der DKA-Orakel-Komponente werden entsprechend der gesetzten Anzahl Nachrichten-Paare generiert und einerseits zur DKA-KeyRecovery-Komponente und andererseits zur DKA-ToyCipher-Komponente weitergeleitet. Diese verschlüsselt mit dem durch den Anwender spezifizierten Schlüssel und leitet die Geheimtexte zur DKA-KeyRecovery-Komponente weiter.

Sind diese Schritte abgeschlossen, ist die DKA-KeyRecovery-Komponente bereit zur Ausführung. Diese führt nach Drücken des Start-Buttons die Wiederherstellung der Schlüsselbits entsprechend des Differenzials aus. Ist diese Berechnung abgeschlossen, wird die boolean-Ausgabe der Komponente gesetzt und signalisiert der DKA-PfadFinder-Komponente, dass der Angriff fortgeführt werden kann. In jeder Runde müssen alle S-Boxen einmal ausgewählt werden, bevor die nächste Verschlüsselungsrunde angegriffen werden kann. Sind alle Runden angegriffen worden, gibt die DKA-KeyRecovery-Komponente die wiederhergestellten Rundenschlüssel aus.

6 Analyse auf Basis der Implementierung

In diesem Kapitel wird die Implementierung des Verfahrens der differenziellen Kryptoanalyse untersucht. Die Analyse wird auf Basis der Chiffren 1 und 3 (siehe Kapitel 2.4.1 und 2.4.3) durchgeführt. Chiffre 1 wird stellvertretend für Chiffren mit einer Verschlüsselungsrunde evaluiert (Verfahren siehe Kapitel 3.5.1). Chiffre 3 ist aufgrund der sechs Verschlüsselungsrunden die komplexeste Chiffre und wird stellvertretend für Chiffren mit mehreren Verschlüsselungsrunden untersucht (Verfahren siehe Kapitel 3.5.2). Die Analysen umfassen den Versuchsaufbau sowie eine Auswertung der Daten.

Darüber hinaus wird bei Analyse von Chiffre 4 (siehe Kapitel 2.4.4) gezeigt, dass differenzielle Kryptoanalyse auch bei Chiffren mit kleiner Blockgröße erfolglos bleiben kann. In einem nächsten Abschnitt wird eine weitere Chiffre angegriffen, die im Rahmen dieser Masterarbeit von einem Betreuer konstruiert wurde. Abschließend werden die Ergebnisse tabellarisch zusammengefasst.

6.1 Evaluation des Verfahrens der DKA

In diesem Abschnitt werden die Chiffren 1 und 3 untersucht.

6.1.1 Analyse von Chiffre 1

Im folgenden Kapitel wird Chiffre 1 untersucht. Es wird die durchschnittliche Anzahl von Nachrichten-Paaren zum erfolgreichen Angriff bestimmt sowie ein Vergleich eines Brute-Force-Angriffs und DKA durchgeführt.

Anzahl von Nachrichten-Paaren

Beim Angriff auf Chiffre 1 mittels DKA wird eine kleine Anzahl von Nachrichten-Paaren benötigt, um die zwei Rundenschlüssel k_1 und k_0 wiederherzustellen. In dieser Analyse wird die verwendete Anzahl der Nachrichten-Paare untersucht, die notwendig ist, um Chiffre 1 zu brechen. Die Nachrichten-Paare sind mittels Zufallszahlengenerator (siehe [31]) erzeugt worden – diese werden aus dem Intervall $[0, 2^{16} - 1]$ ausgewählt. Der Angriff ist 1.000-mal wiederholt worden und verläuft in dieser Form immer erfolgreich. Die Ergebnisse der Analyse sind in Abbildung 45 zu sehen.

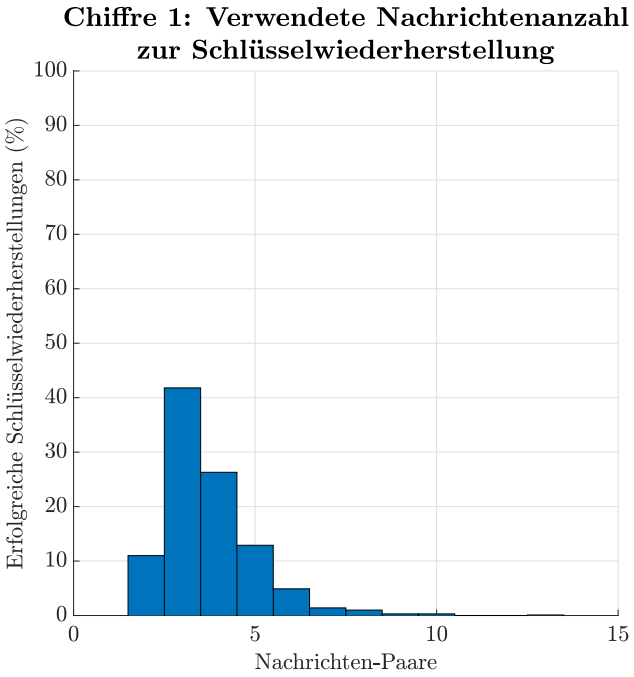


Abbildung 45: *Verwendete Nachrichten-Paare bei der Schlüsselwiederherstellung bei Chiffre 1*

Auf der X-Achse des Diagramms ist die Anzahl der verwendeten Nachrichten-Paare abgebildet, die zum Wiederherstellen der Rundenschlüssel notwendig waren. Auf der Y-Achse ist die Anzahl an erfolgreichen Schlüsselwiederherstellungen zu sehen. Das Minimum der Nachrichten-Paare beträgt 2, das Maximum sind 13. Am häufigsten mit 418 erfolgreichen Schlüsselwiederherstellungen sind 3 Nachrichten-Paare notwendig. Darauf folgen 263 erfolgreiche Schlüsselwiederherstellungen mit 4 Nachrichten-Paaren. Durchschnittlich sind 3,7 Nachrichten-Paare zur Schlüsselwiederherstellung nötig, die Standardabweichung beträgt 1,28. Für eine erfolgreiche Wiederherstellung der Rundenschlüssel sind also mindestens 2 Nachrichten-Paare notwendig. Nach der Durchführung der DKA mit dem ersten Nachrichten-Paar (Verfahren siehe Kapitel 3.5.1) bleibt in allen Fällen mehr als ein möglicher Schlüsselkandidat übrig. Folglich müssen noch weitere Nachrichten-Paare betrachtet werden, um die Menge der möglichen Schlüsselkandidaten weiter zu reduzieren. In den Fällen, in denen 6 oder mehr Nachrichten-Paare notwendig waren, könnte es ein, dass Nachrichten-Paare zufällig so gewählt wurden, sodass diese dieselbe Differenz aufgewiesen hatten. In diesem Fall würde durch das zweite Nachrichten-Paar keine Schlüsselkandidaten aus der Menge entfernt. Andererseits wäre es möglich, dass verschiedene Paare exakt dieselben

Schlüsselkandidaten aus der Menge entfernen, sodass weitere Nachrichten-Paare notwendig sind.

Vergleich mit Brute-Force

Bei einer Brute-Force-Attacke eines 32 Bit langen Schlüssels sind 2^{32} verschiedene Schlüssel zu untersuchen. Im Allgemeinen ist eine Brute-Force-Attacke nach 2^{n-1} getesteten Schlüsseln erfolgreich. Bei Chiffre 1 wären also 2^{31} Schlüssel zu testen. In dieser Analyse werden die durchschnittlich verwendeten Schlüssel bei der DKA von Chiffre 1 untersucht. Chiffre 1 wurde dabei 1.000-mal angegriffen und die Anzahl an getesteten Schlüsseln gezählt. Die Nachrichten zur Schlüsselwiederherstellung wurden, wie in der Untersuchung zuvor, mittels Zufallszahlengenerator erzeugt. Abbildung 46 zeigt den Vergleich zwischen einem Brute-Force-Angriff und DKA.

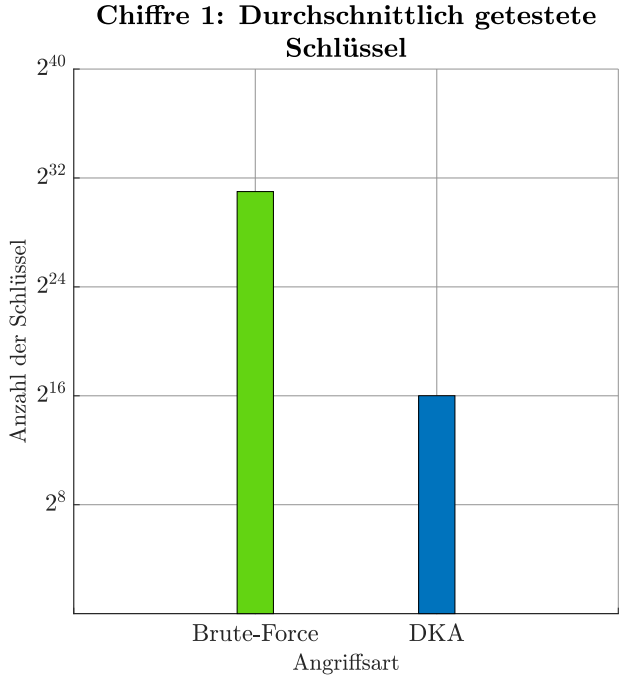


Abbildung 46: *Durchschnittlich durchsuchter Schlüsselraum bei der Schlüsselwiederherstellung bei Chiffre 1*

Auf der X-Achse sind die Angriffsarten (Brute-Force: grün, DKA: blau) abgebildet. Auf der Y-Achse ist die Anzahl der durchschnittlich getesteten Schlüssel als Zweierpotenz zu sehen. Durchschnittlich sind bei dieser Untersuchung 65.914 Schlüssel mittels DKA verwendet worden. Das entspricht im Mittel

2^{16} getesteten Schlüsseln. Im Vergleich zu Brute-Force ist die Anzahl der getesteten Schlüssel bei der DKA um den Faktor 2^{15} kleiner.

6.1.2 Analyse von Chiffre 3

Im folgenden Kapitel wird Chiffre 3 untersucht. Es werden die Wahrscheinlichkeiten der gefundenen Differenziale gezeigt und die verschiedenen Suchstrategien analysiert. Des Weiteren werden die Filterung und die Schlüsselwiederherstellung untersucht. Abschließend wird die DKA von Chiffre 3 mit einer Brute-Force-Attacke verglichen.

Gefundene Differenziale

Bei den verschiedenen Untersuchungen sind zum Teil, wie beispielsweise bei der Schlüsselwiederherstellung, Differenziale notwendig. Die Wahrscheinlichkeiten der gefundenen Differenziale bei Chiffre 3 sind in Abbildung 47 zu sehen. Diese sind mittels der Suchstrategie „Beste Charakteristik (Tiefensuche, globaler Schwellwert), dann Differenzialsuche“ ermittelt worden. Der Schwellwert zum Abbruch ist auf den in Kapitel 4.2.1 vorgeschlagenen Wert von $0,001$ festgelegt worden.

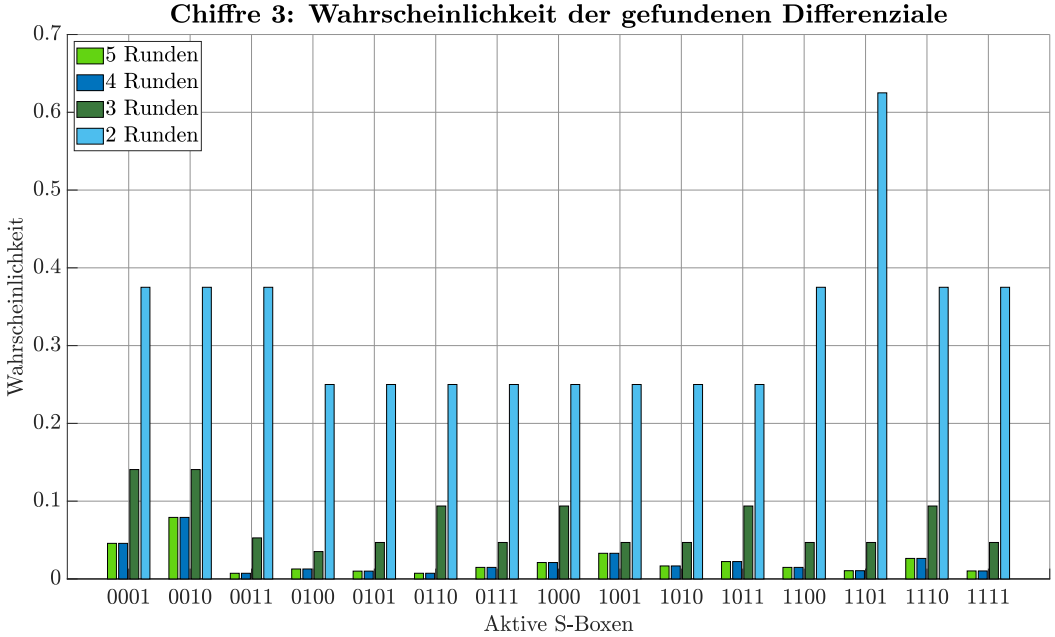


Abbildung 47: *Gefundene Differenziale Chiffre 3*

Auf der X-Achse sind die aktiven S-Boxen für die Suche nach Differenzialen abgebildet. Eine S-Box gilt als aktiv, wenn diese eine Eingangsdifferenz $\neq 0$ aufweist. Die S-Boxen werden beim Angriff mittels DKA vom Benutzer im DKA-PfadFinder manuell bestimmt. Die Codierung der aktiven S-Boxen ist binär angegeben: Eine 0 zeigt an, dass die S-Box nicht aktiv ist, eine 1, dass sie aktiv ist. Die S-Box mit den Eingabebits 1 bis 4 des Eingabeblocks entspricht dabei dem *least significant bit* (LSB), die S-Box mit den Eingabebits 12 bis 16 des Eingabeblocks entspricht dem *most significant bit* (MSB) in der Codierung. Alle folgenden Untersuchungen und Plots, in denen aktive S-Boxen verwendet werden, nutzen diese Codierung. Auf der Y-Achse ist die Wahrscheinlichkeit der gefundenen Differenziale abgebildet. Die Legende zeigt an, für welche Rundenzahl das entsprechende Differenzial gefunden wurde. So kennzeichnen die hellgrünen Balken Differenziale nach der fünften Runde und die dunkelblauen Balken die Differenziale nach der vierten Runde.

Bei der Betrachtung der aktiven S-Boxen für verschiedene Rundenanzahlen ist festzustellen, dass die Wahrscheinlichkeiten anwachsen, je geringer die Rundenanzahl ist. Bei der Betrachtung einer 4-Runden-Charakteristik könnte eine 3-Runden-Charakteristik beispielsweise dadurch erhalten werden, indem man die S-Boxen der letzten Runde wegstreicht – dadurch steigt die Wahrscheinlichkeit an. Für einen Angriff auf eine Verschlüsselungsrunde muss jede S-Box dieser Runde mindestens einmal aktiv sein. In der fünften Runde eignen sich für einen Angriff beispielsweise die aktiven S-Box-Kombinationen 0001, 0010, 1001 und 1110. Diese weisen die Wahrscheinlichkeiten 0,045; 0,079; 0,032 und 0,026 auf. Im Vergleich zu den Wahrscheinlichkeiten der anderen Differenziale sind diese die höchsten. Schlüsselbits werden bei dieser Wahl von Differenzialen mehrfach durchsucht, da die aktiven S-Boxen in der Auswahl zum Teil mehrfach vorkommen. Dieses Vorgehen kann sinnvoll sein, wenn für bestimmte S-Boxen kein Differenzial mit hoher Wahrscheinlichkeit gefunden werden kann. Beim mehrfachen Durchsuchen von Schlüsselbits wächst die Dauer der Schlüsselwiederherstellung, da Teile des Rundenschlüssels mehrfach angegriffen werden. Es ist auch zu bedenken, dass der Suchraum durch die Anzahl von aktiven S-Boxen exponentiell anwächst. Alternativ zu dieser Wahl von aktiven S-Box-Kombinationen könnten die S-Boxen 1000 und 0111 mit den Wahrscheinlichkeiten 0,045 und 0,026 verwendet werden. Dabei werden keine Schlüsselbits mehrfach durchsucht.

Der Faktor Zeit kann bei der Schlüsselwiederherstellung gegenüber höheren Wahrscheinlichkeiten abgewogen werden. Wenn der Fokus auf dem Erfolg des Angriffs liegt, können natürlich Differenziale mit höheren Wahrscheinlichkei-

ten verwendet werden, obwohl diese möglicherweise aktive S-Boxen mehrfach enthalten. Tabelle 7 fasst die gefundenen Differenziale zusammen.

Aktive S-Boxen	Klartext-Differenz	Erwartete Differenz	Wahrscheinlichkeit
0001	0000 0000 0000 0010	0000 0000 0000 0010	0,045
0010	0000 0000 0010 0000	0000 0000 0010 0000	0,079
0011	0000 0000 0010 0010	0000 0000 0010 0010	0,007
0100	0000 0010 0000 0000	0000 0010 0000 0000	0,012
0101	0010 0000 0000 0000	0000 0010 0000 0010	0,010
0110	0000 0010 0010 0000	0000 0010 0010 0000	0.007
0111	0000 0000 0010 0000	0000 0010 0010 0010	0,014
1000	0010 0000 0000 0000	0010 0000 0000 0000	0,021
1001	0000 0000 0010 0000	0010 0000 0000 0010	0,032
1010	0000 0000 0000 0010	0010 0000 0010 0000	0,016
1011	0010 0000 0000 0000	0010 0000 0010 0010	0,022
1100	0000 0000 0010 0000	0010 0010 0000 0000	0,014
1101	0010 0000 0000 0000	0010 0010 0000 0010	0,010
1110	0000 0000 0010 0000	0010 0010 0010 0000	0,026
1111	0010 0000 0000 0000	0010 0010 0010 0010	0,010

Tabelle 7: *Übersicht über die gefundenen Differenziale in Runde 5*

Untersuchung der Suchstrategien

Für die Suche nach Charakteristiken bzw. Differenzialen stehen dem Anwender in der CT2-Implementierung vier verschiedene Strategien zur Auswahl. In der folgenden Analyse wird die Laufzeit der Suchstrategien untersucht. Die Laufzeit ist dabei von der Hardware abhängig, auf der die Analyse ausgeführt wird. Diese wurde auf einer virtuellen Maschine (VM) mit 8 virtuellen Prozessoren (2,9 GHz) und 8 GB Arbeitsspeicher ausgeführt. Die VM läuft auf einem Virtualisierungs-Cluster mit 2 Knoten. Die Knoten sind an ein Storage Area Network (SAN) angebunden, welches Solid State Drives (SSD) enthält. Aufgrund der zum Teil langen Laufzeiten ist diese Berechnung einmalig ausgeführt worden. Abbildung 48 zeigt die Ergebnisse zur Untersuchung der Laufzeit der Suche nach Charakteristiken der fünften Runde in Chiffre 3.

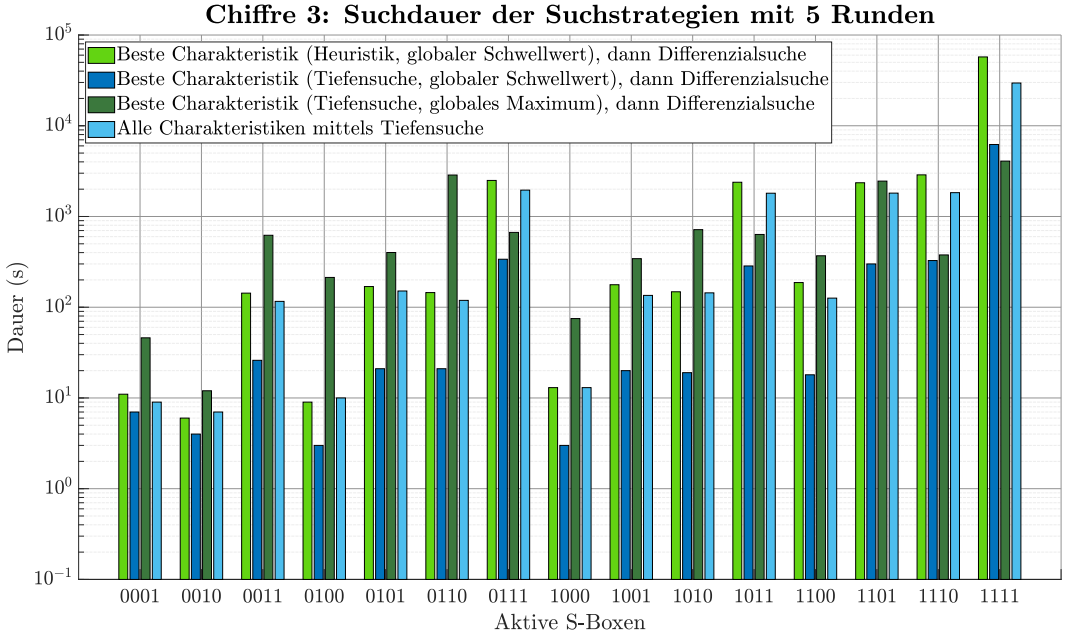


Abbildung 48: *Dauer der Suchstrategien mit 5 Runden bei Chiffre 3*

Auf der X-Achse sind die aktiven S-Boxen angegeben. Auf der Y-Achse ist die Suchdauer in Sekunden abgebildet. Die Y-Achse ist logarithmisch skaliert. Die Suchstrategien werden zur kürzeren Bezeichnung mit den folgenden Abkürzungen bezeichnet:

HGS: Beste Charakteristik (Heuristik, globaler Schwellwert), dann Differenzialsuche

TGS: Beste Charakteristik (Tiefensuche, globaler Schwellwert), dann Differenzialsuche

TGM: Beste Charakteristik (Tiefensuche, globales Maximum), dann Differenzialsuche

ACT: Alle Charakteristiken mittels Tiefensuche

Diese Abkürzungen werden in den nachfolgenden Untersuchungen ebenfalls verwendet.

Es lässt sich feststellen, dass die Suchdauer bei allen Suchstrategien mit zunehmender Anzahl an aktiven S-Boxen länger wird. Die Suchdauer bei HGS mit den aktiven S-Boxen 0001 und 1000 beträgt in etwa 12 Sekunden, bei der Betrachtung der aktiven S-Boxen 0011 und 0110 kann eine Suchdauer

von etwa 143 Sekunden abgelesen werden. Dahingegen beträgt die Suchdauer, wenn alle S-Boxen aktiv sind, ungefähr 16 Stunden. Im Wesentlichen liegt das daran, dass der Suchraum durch Hinzunahme von weiteren aktiven S-Boxen exponentiell anwächst. Diese Beobachtung kann auf alle 4 Suchstrategien übertragen werden. Allerdings sei darauf hingewiesen, dass die Suche ein einmaliger Aufwand ist. Die gewonnenen Resultate sind unabhängig vom Wert des Schlüssels und können bei Angriffen mittels DKA wiederverwendet werden. Bei den Resultaten fällt auf, dass die Suchstrategie TGM bei höchstens zwei aktiven S-Box stets die längste Dauer aufweist. Bei drei oder vier aktiven S-Boxen ist diese Suchstrategie nicht mehr am langsamsten. Das kann dadurch erklärt werden, dass zunächst ein Teilbaum vollständig durchsucht werden muss, bis ein Wert für das Maximum gefunden ist – während die Suche läuft, muss der erste gefundene Wert nicht das Maximum sein. Allerdings kann zu diesem Zeitpunkt früher abgebrochen werden, da ein Vergleichswert zur Verfügung steht.

Die durchschnittliche Suchdauer bei allen 15 aktiven S-Box-Kombinationen mit HGS beträgt 1 Stunde und 16 Minuten. Dahingegen beträgt die durchschnittliche Suchdauer bei TGS 8 Minuten und 27 Sekunden, bei TGM 15 Minuten und 24 Sekunden und bei ACT 42 Minuten und 1 Sekunde. Bei vier aktiven S-Boxen beträgt die Suchdauer für HGS 57.350 s, für TGS 6.217 s, für TGM 4.079 s und für ACT 29.596 s. Tabelle 8 fasst die durchschnittliche Dauer der verschiedenen Suchstrategien bei einer, zwei und drei aktiven S-Boxen zusammen.

Suchstrategie	**1 aktive S-Box**	**2 aktive S-Boxen**	**3 aktive S-Boxen**
HGS	9,75 s	161,50 s	2528,00 s
TGS	4,25 s	20,83 s	312,50 s
TGM	86,50 s	885,83 s	1033,50 s
ACT	9,75 s	131,83 s	1849,75 s

Tabelle 8: *Durchschnittliche Suchdauer der Suchstrategien mit 5 Runden bei Chiffre 3*

Bei der Betrachtung von Tabelle 8 fällt auf, dass die Suchstrategie TGS durchschnittlich die geringsten Suchzeiten aufweist. Diese Suchstrategie bricht bei der Suche nach der besten Charakteristik im ersten Teil der Suchstrategie aufgrund des Schwellwerts früh ab und ist im Vergleich mit den anderen Suchstrategien die schnellste. Daher ist diese Suchstrategie bei Betrachtung der Dauer die beste Wahl. Des Weiteren ist es sinnvoll, im Hinblick auf die Dauer der Suche, höchstens zwei S-Boxen gleichzeitig anzugreifen, da sonst

die Suchdauer wesentlich länger ist. In dieser Untersuchung sind Wahrscheinlichkeiten der gefundenen Differenziale nicht berücksichtigt worden. Diese unterscheiden sich oftmals nur geringfügig. In Tabelle 9 sind beispielhaft die gefundenen Differenziale der Suchstrategien für die aktiven S-Boxen 0011 abgebildet. Die Ergebnisse der Suchstrategien HGS, TGS und ACT sind identisch, lediglich mit TGM wurde ein anderes Differenzial gefunden. Unterschiede können dadurch auftreten, dass das gefundene globale Maximum als einzelne Charakteristik nicht so gut geeignet ist, um ein Differenzial zu bilden.

Suchstrategie	**Anz. gefundene Char.**	P_{Diff}
HGS	6	0,0073
TGS	6	0,0073
TGM	3	0,0036
ACT	6	0,0073

Tabelle 9: *Wahrscheinlichkeit der gefundenen Differenziale bei den aktiven S-Boxen 0011 der verschiedenen Suchstrategien mit 5 Runden bei Chiffre 3*

Abschließend ist in Anbetracht der Suchdauer und der Betrachtung der Differenziale die Suchstrategie TGS mit höchstens zwei aktiven S-Boxen zu empfehlen.

Untersuchung der Filterung

Durch Filterung der Nachrichten-Paare wird das Ergebnis der Schlüsselwiederherstellung bei DKA verbessert. In der folgenden Untersuchung wird die durchschnittliche Auswirkung der Filterung auf alle 2^{16} unterscheidbaren Paaren bei Chiffre 3 in der fünften Runde untersucht. Zur Filterung werden die in Kapitel 3.4 vorgestellten Techniken eingesetzt. Es werden die in Abbildung 47 notierten Differenziale verwendet. Bei den Nachrichten-Paaren sind keine Duplikate erlaubt. Für jedes Differenzial werden 2^{16} verschiedene Nachrichten-Paare generiert, anschließend werden diese gefiltert. Die Anzahl verbliebener Nachrichten-Paare wird notiert. Diese Berechnung wird 1.000-mal wiederholt. Die Filterung ist nicht unabhängig vom Schlüssel. Deshalb werden in jeder Iteration die Nachrichten-Paare mit einem neuen, durch einen Zufallszahlengenerator generierten, Schlüssel verschlüsselt. Von den Ergebnissen wird je Differenzial der Durchschnitt berechnet. Vergleichend dazu wird die erwartete Anzahl von Treffern bei der Schlüsselwiederherstellung mittels der Wahrscheinlichkeit des Differenzials berechnet. Bei der Filterung der Paare ist zu erwarten, dass mehr Paare verbleiben als Treffer bei der

Schlüsselwiederherstellung erwartet werden können. In Abbildung 49 ist das Ergebnis der Untersuchung zu sehen.

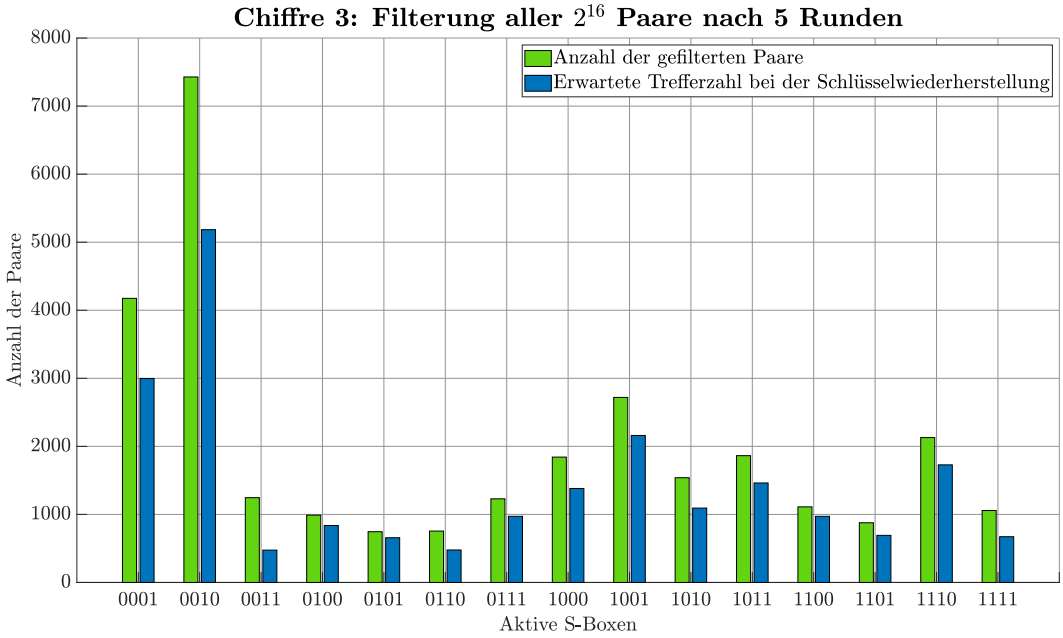
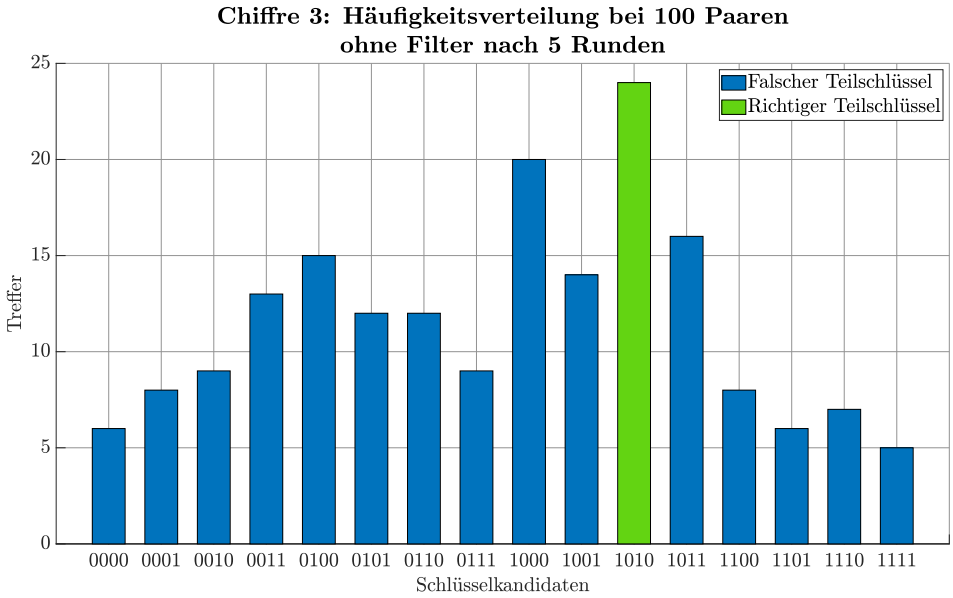


Abbildung 49: *Durchschnittlich ausgefilterte Nachrichten-Paare bei Chiffre 3*

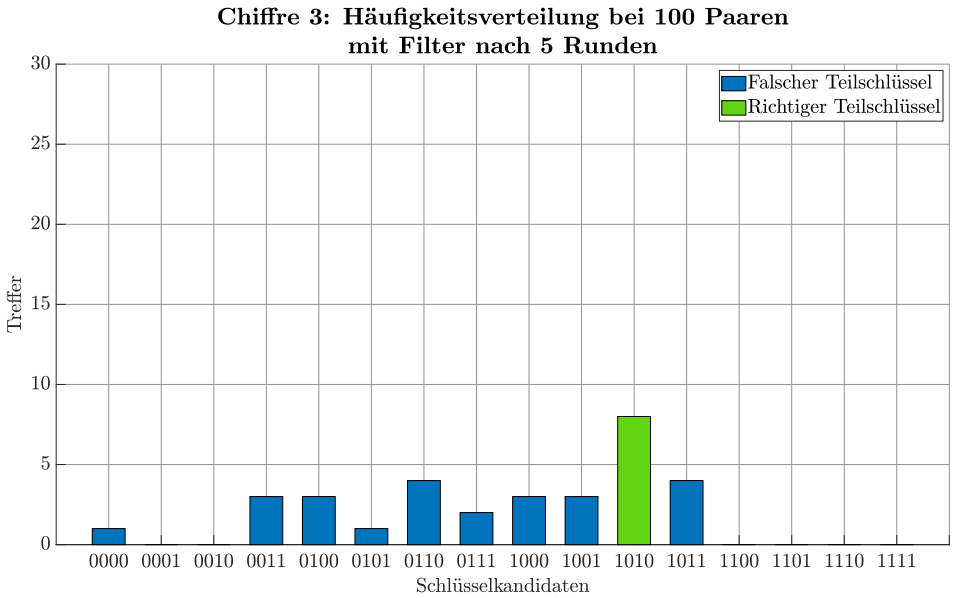
Auf der X-Achse sind die aktiven S-Boxen zu sehen, auf der Y-Achse ist die Anzahl der Nachrichten-Paare abgebildet. Die Anzahl an verbleibenden Nachrichten-Paaren ist grün, die erwartete Trefferzahl bei der Schlüsselwiederherstellung ist blau. Es kann festgestellt werden, dass die verbleibende Anzahl von Nachrichten-Paaren bei jedem Differenzial höher als die erwartete Trefferzahl ist. Bei der Betrachtung der Wahrscheinlichkeiten der Differenziale (Abbildung 47) ist zu sehen, dass die Anzahl an Nachrichten-Paaren, die nach der Filterung übrig bleiben, ansteigt, je höher die Wahrscheinlichkeit ist. So verbleiben bei der Betrachtung der aktiven S-Boxen 0010 nach der Filterung von 2^{16} Nachrichten-Paaren noch 7.428 Paare. Bei der Schlüsselwiederherstellung ist bei dem verwendeten Differenzial mit einer Wahrscheinlichkeit von 0,079 eine Trefferzahl von 5.184 zu erwarten. Insgesamt bestätigt diese Untersuchung die Erwartung, dass durch die Filterung immer eine höhere Anzahl von Nachrichten-Paaren verbleibt als die erwartete Trefferzahl bei der Schlüsselwiederherstellung.

In der nächsten Untersuchung wird der Einfluss der Filterung der Nachrichten-Paare auf die Schlüsselwiederherstellung untersucht. Es werden wieder die in

Abbildung 47 beschriebenen Differenziale verwendet und eine Anzahl von Nachrichten-Paaren generiert (hier 100), diese werden durch einen Zufallszahlengenerator erzeugt. Es sind keine Duplikate oder symmetrische Paare erlaubt. Zu einem Paar (a, b) ist (b, a) ein symmetrisches Paar. Mit den Nachrichten-Paaren wird einmal die Schlüsselwiederherstellung ohne Filterung (Abbildung 50a) und einmal mit Filterung (50b) durchgeführt. Die Liste mit den Nachrichten-Paaren ist initial für diese Untersuchung erstellt worden. Im Anhang A.2.2 sind weitere Untersuchungen mit höheren Anzahlen von Nachrichten-Paaren zu finden. Die Untersuchungen verwenden alle exakt dieselbe Nachrichten-Paar-Liste, allerdings nur die entsprechend angegebene große Teilmenge. Es wird jeweils der letzte Rundenschlüssel von Chiffre 3 mit den aktiven S-Boxen 0010 angegriffen – das heißt, es werden die Bits 4-8 von k_5 angegriffen. Abbildung 50 zeigt die Ergebnisse für 100 Nachrichten-Paare.



(a) *Häufigkeitsverteilung ohne Filterung*



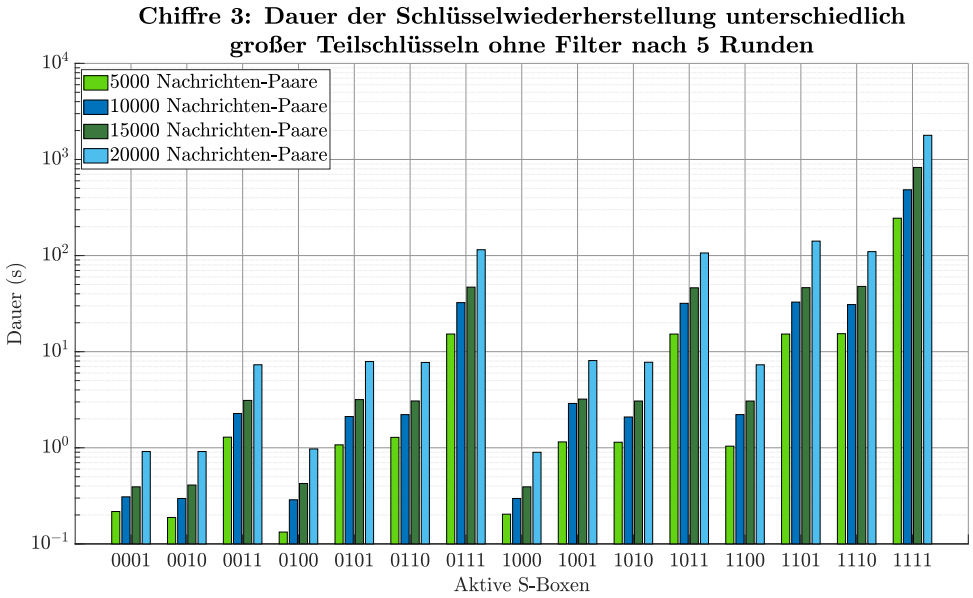
(b) *Häufigkeitsverteilung mit Filterung*

Abbildung 50: *Häufigkeitsverteilung bei der Schlüsselwiederherstellung mit 100 Nachrichten-Paaren bei Chiffre 3*

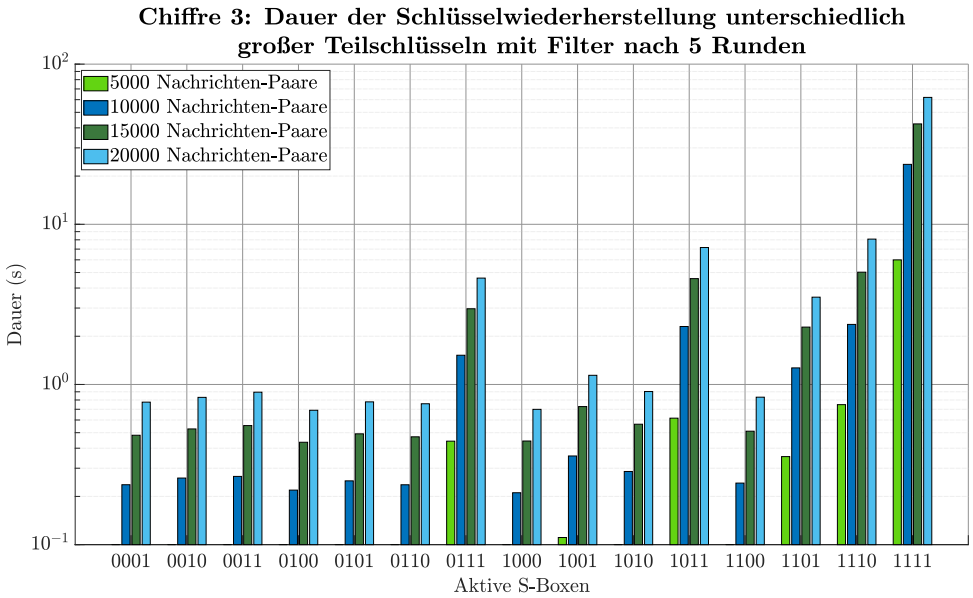
Auf der X-Achse sind die 16 verschiedenen Schlüsselkandidaten in binärer Codierung von k_5 abgebildet, auf der Y-Achse ist die Anzahl der Treffer für den jeweiligen Schlüsselkandidat zu sehen. Der grüne Balken steht für den

richtigen Schlüsselkandidat, der mittels Schlüsselwiederherstellung gefunden werden soll. Bei der Untersuchung mit und ohne Filter ist der richtige Schlüsselkandidat jeweils der mit den meisten Treffern. Ohne Filterung hat der richtige Schlüsselkandidat 24 Treffer und mit Filterung 8 Treffer. Die unterschiedlichen Anzahlen lassen sich dadurch erklären, dass durch die Filterung weniger Paare für die Schlüsselwiederherstellung zur Verfügung stehen und auch falsche Paare die erwartete Differenz erfüllen können. In Abbildung 50a ist zu sehen, dass für jeden Schlüsselkandidaten mindestens ein Treffer vorhanden ist. Dahingegen ist in Abbildung 50b nicht für jeden Schlüsselkandidaten ein Treffer vorhanden – manche Kandidaten, wie beispielsweise 1100, haben gar keinen Treffer. Bei beiden Abbildungen beträgt die Differenz zwischen dem richtigen Schlüsselkandidat und dem falschen mit der höchsten Trefferzahl 4. Ohne Filterung gibt es insgesamt 184 Treffer, wobei 160 Treffer dabei auf die falschen Schlüsselkandidaten fallen. Im Durchschnitt erhält jeder Schlüsselkandidat 11,5 Treffer und die falschen Schlüsselkandidaten 10,7 Treffer. Mit Filter gibt es insgesamt 32 Treffer, wobei sich 24 Treffer auf die falschen Schlüsselkandidaten verteilen. Im Durchschnitt erhält jeder Schlüsselkandidat damit 2 Treffer und die falschen Schlüsselkandidaten 1,6 Treffer.

In der nächsten Untersuchung wird die Dauer der Schlüsselwiederherstellung bei unterschiedlich großen Teilschlüsseln mit und ohne Filterung untersucht. Für jede aktive S-Box müssen 4 Schlüsselbits durchsucht werden. Bei i aktiven S-Boxen müssen 2^i verschiedene Schlüssel für jedes Nachrichten-Paar untersucht werden. Die Nachrichten-Paare werden mittels Zufallszahlengenerator erzeugt, es sind keine Duplikate oder symmetrische Paare erlaubt. Die Zeitmessung beginnt erst bei der Schlüsselwiederherstellung, alle vorherigen Aktivitäten wie die Differenzialsuche oder das Erzeugen der Nachrichten werden zeitlich nicht erfasst. Bei der Variante mit Filterung beginnt die Zeitmessung bereits bei der Filterung. Die verwendeten Differenziale sind in Tabelle 7 zu finden. Abbildung 51a zeigt die Untersuchung ohne Filterung, Abbildung 51b stellt die Untersuchung mit Filterung dar. Die Untersuchung wurde 10-mal durchgeführt. Die gemessene Zeit entspricht dem Durchschnitt der wiederholten Untersuchungen. Zwecks Vergleichbarkeit ist das Experiment auf der zuvor spezifizierten VM durchgeführt worden.



(a) *Dauer ohne Filterung*



(b) *Dauer mit Filterung*

Abbildung 51: *Dauer der Schlüsselwiederherstellung bei unterschiedlich großen Teilschlüsseln bei Chiffre 3*

Auf der X-Achse sind die aktiven S-Boxen zu sehen, die Y-Achse zeigt die Dauer der Schlüsselwiederherstellung in Sekunden. Die Y-Achse ist logarithmisch skaliert. In beiden Untersuchungen ist festzustellen, dass die Schlüssel-

wiederherstellung umso länger dauert, je mehr S-Boxen aktiv sind. Das liegt daran, dass der Schlüsselraum mit der Anzahl an aktiven S-Boxen exponentiell wächst. Beispielsweise dauert die Schlüsselwiederherstellung ohne Filterung bei einer aktiven S-Box mit 20.000 Nachrichten durchschnittlich 0,92 Sekunden und bei zwei aktiven S-Boxen durchschnittlich 7,68 Sekunden. Bei drei aktiven S-Boxen dauert die Schlüsselwiederherstellung durchschnittlich 118,19 Sekunden und bei vier aktiven S-Boxen 1.784 Sekunden. Vergleicht man diese Werte jeweils mit der gefilterten Variante, lässt sich eine geringere Dauer feststellen.

Anzahl Nachrichten-Paare	**1 aktive S-Box**	**2 aktive S-Boxen**	**3 aktive S-Boxen**
5.000	0,18 s	1,16 s	15,29 s
10.000	0,29 s	2,29 s	31,99 s
15.000	0,40 s	3,11 s	46,82 s
20.000	0,92 s	7,68 s	118,19 s

Tabelle 10: *Durchschnittliche Dauer der Schlüsselwiederherstellung für unterschiedlich große Teilschlüssel bei 5 Runden ohne Filterung bei Chiffre 3*

Anzahl Nachrichten-Paare	**1 aktive S-Box**	**2 aktive S-Boxen**	**3 aktive S-Boxen**
5.000	0,08 s	0,08 s	0,54 s
10.000	0,23 s	0,27 s	1,86 s
15.000	0,47 s	0,55 s	3,71 s
20.000	0,74 s	0,88 s	5,84 s

Tabelle 11: *Durchschnittliche Dauer der Schlüsselwiederherstellung für unterschiedlich große Teilschlüssel bei 5 Runden mit Filterung bei Chiffre 3*

Die Tabellen 10 (ohne Filterung) und 11 (mit Filterung) zeigen die durchschnittliche Dauer der Schlüsselwiederherstellung für unterschiedliche Anzahlen von aktiven S-Boxen und Nachrichten-Paaren. Es lässt sich ablesen, dass die Schlüsselwiederherstellung mit Filterung ab zwei aktiven S-Boxen wesentlich schneller durchgeführt werden kann als ohne. Das lässt sich dadurch erklären, dass durch die Filterung viele Paare entfernt werden, die im Anschluss bei der Schlüsselwiederherstellung nicht mehr verarbeitet werden müssen. Der Filterungsprozess ist zwar ein zusätzlicher Schritt bei der DKA, allerdings lohnt sich dieser bei der Betrachtung der Laufzeit. Bei 20.000 Nachrichten-Paaren und 1 aktiven S-Box beträgt der Faktor zwischen mit

und ohne Filterung etwa 1,25; bei 2 aktiven S-Boxen 8,72 und bei 3 aktiven S-Boxen 20,23. Bei Betrachtung der Faktoren ist festzustellen, dass diese nicht linear anwachsen. Dies ist vermutlich darauf zurückzuführen, dass die Anzahl der zu betrachtenden Schlüsselkandidaten exponentiell mit der Anzahl der aktiven S-Boxen anwächst und dementsprechend auch entsprechend schnell anwächst, je mehr Nachrichten-Paare betrachtet werden müssen.

Die nächste Messung zeigt die Dauer der Schlüsselwiederherstellung in Abhängigkeit der Anzahl von Nachrichten-Paaren. Untersucht wird einmal mit und einmal ohne Filterung. Unter den Paaren sind keine Duplikate erlaubt. Die Zeitmessung beginnt erst bei der Filterung bzw. bei der Schlüsselwiederherstellung. Alle vorherigen Aktivitäten sind in der Zeitmessung nicht enthalten. Ausgeführt wurde diese Untersuchung auf der bereits spezifizierten VM. Das Experiment wurde 10-mal wiederholt. Da in dieser Untersuchung hauptsächlich die Geschwindigkeit von Interesse ist, sind in allen anzugreifenden Runden die aktiven S-Box-Kombinationen 1000, 0100, 0010, 0001 verwendet worden. Die angegebene Anzahl von Nachrichten-Paaren ist für jedes verwendete Differenzial generiert worden. In Abbildung 52 sind die Durchschnittswerte dargestellt.

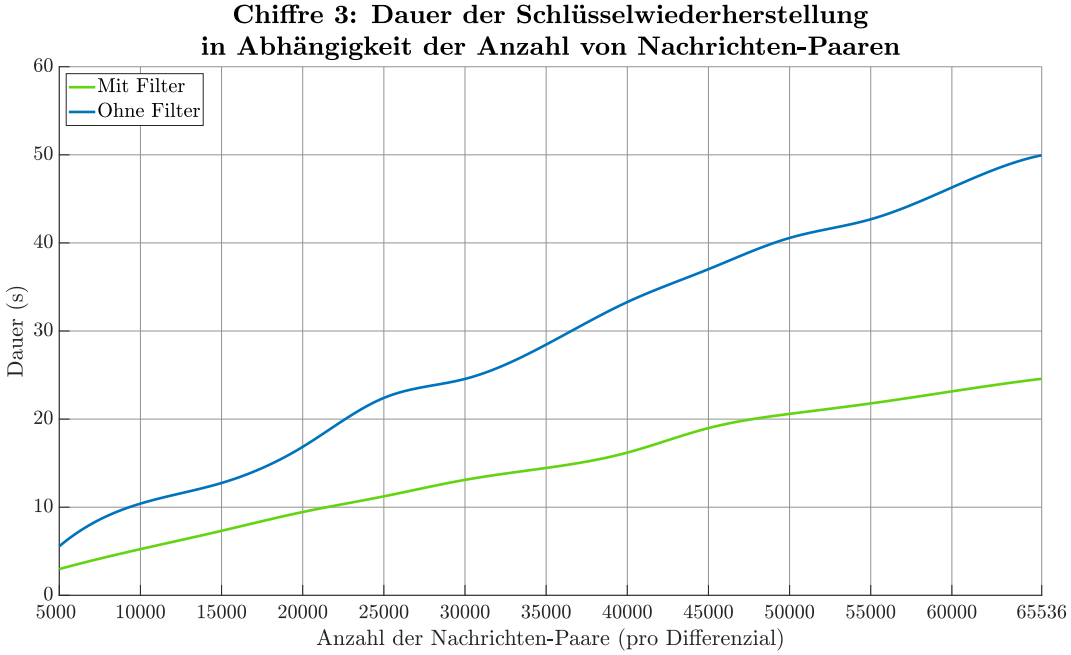


Abbildung 52: *Durchschnittliche Dauer der Schlüsselwiederherstellung mit und ohne Filterung bei Chiffre 3*

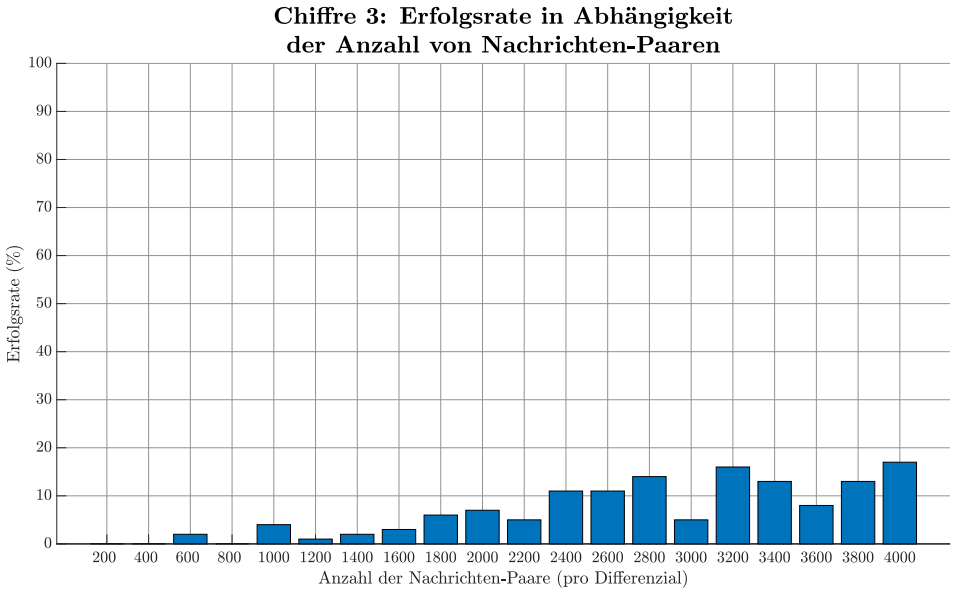
Auf der X-Achse ist die Anzahl von Nachrichten-Paaren zu sehen und auf der Y-Achse ist die Zeit der Schlüsselwiederherstellung in Sekunden angegeben. Der Verlauf der beiden Graphen ist bis auf wenige Abweichungen linear, welche durch unterschiedliche Laufzeiten (z. B. bedingt durch das Scheduling des Betriebssystems) zu erklären sind. Für beide Varianten gilt: Mit steigender Anzahl der Nachrichten-Paare steigt auch die Dauer der Schlüsselwiederherstellung. Dies liegt daran, dass für jeden möglichen Schlüsselkandidat die Nachrichten-Paare verarbeitet werden müssen. Vergleicht man die beiden Varianten miteinander, weist die Schlüsselwiederherstellung mit Filter stets eine kürzere Laufzeit auf. Beispielsweise beträgt die Differenz bei 5.000 Nachrichten-Paaren etwa 2 Sekunden, bei 35.000 ungefähr 14 Sekunden und bei 65.536 Paaren ca. 25 Sekunden. Daran lässt sich erkennen, dass die Filterung bei steigender Anzahl an Nachrichten-Paaren an Effektivität gewinnt.

Insgesamt hat sich durch die Untersuchungen zur Filterung gezeigt, dass diese ein sinnvolles Instrument zur Senkung der Laufzeit bei der Schlüsselwiederherstellung ist. Des Weiteren ist festzuhalten, dass durch den Einsatz von Filterung das Ergebnis der Schlüsselwiederherstellung qualitativ verbessert wird (siehe Abbildung 50).

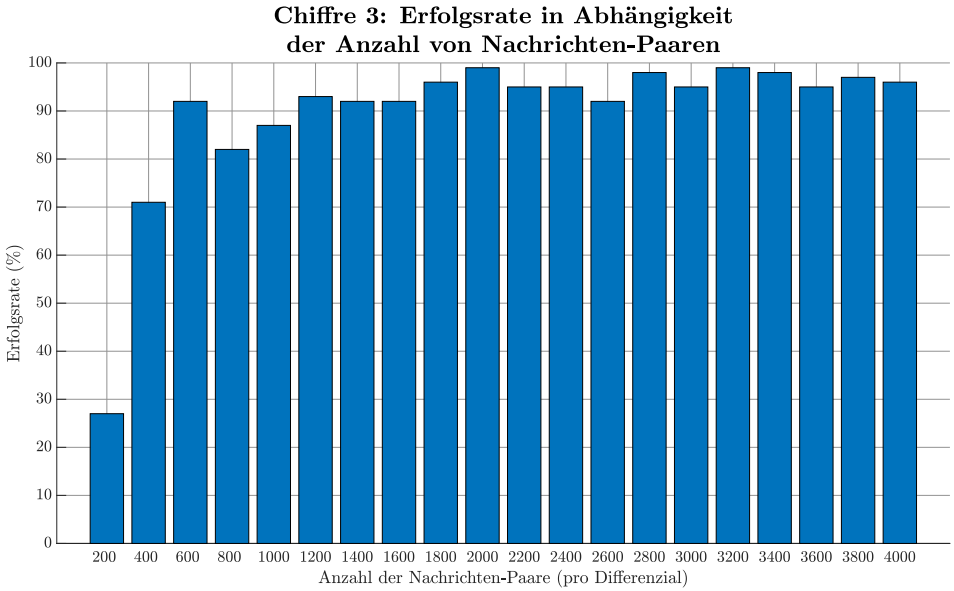
Untersuchung der Schlüsselwiederherstellung

Das Ziel einer differenziellen Kryptoanalyse ist das Wiederherstellen der Rundenschlüssel einer Chiffre. In diesem Abschnitt wird die Erfolgsrate untersucht. Dazu wird beispielhaft Chiffre 3 angegriffen. In Abhängigkeit der Anzahl der Nachrichten wird die Erfolgsrate der DKA untersucht. Es werden zwei verschiedene Arten der Nachrichten-Paar-Generierung unterschieden. Zum einen werden die Nachrichten-Paare zufällig generiert, zum anderen werden diese mit inkrementierten Werten erzeugt. Letzteres genanntes bedeutet, dass bei Betrachtung von n Nachrichten-Paaren $(a_1, b_1), (a_2, b_2), ..., (a_n, b_n)$ gilt: $a_j = (a_i + 1)$ mit $j = i + 1$ wobei $i, j \in \mathbb{N}$. Für alle b_i gilt, dass $a_i \oplus b_i = \Delta d$ für festes Δd. Bei den Untersuchungen sind keine Dubletten oder symmetrische Paare erlaubt. In beiden Fällen werden die erzeugten Nachrichten-Paare gefiltert. In jeder Runde r wurden die folgenden aktiven S-Boxen verwendet: Für die Schlüsselbits von S-Box$_{r4}$ wurde die aktive S-Box 0001 gewählt, für die Schlüsselbits von S-Box$_{r3}$ die aktive S-Box 0010, für die Schlüsselbits von S-Box$_{r1}$ die aktiven S-Boxen 1001 und für die Schlüsselbits von S-Box$_{r2}$ die aktiven S-Boxen 1110 verwendet (mit $r = 2, ..., 5$). Die Differenziale wurden mit der Suchstrategie TGS ermittelt. Die Erfolgsrate gibt einen prozentualen Durchschnittswert in Abhängigkeit einer festgelegten Anzahl von Nachrichten-Paaren an. Abbildung 53 zeigt die Erfolgsrate bei 1000

Durchführungen, wobei Abbildung 53a die Erfolgsrate mit inkrementierten Nachrichten-Paaren und Abbildung 53b mit zufälligen Nachrichten-Paaren darstellt. In jeder Wiederholung wurden die Schlüssel neu durch einen Zufallszahlengenerator erzeugt. Die angegebene Anzahl von Nachrichten-Paaren ist für jedes verwendete Differenzial generiert worden.



(a) *Erfolgsrate mit inkrementierten Nachrichten-Paaren*



(b) *Erfolgsrate mit zufälligen Nachrichten-Paaren*

Abbildung 53: *Erfolgsrate der differenziellen Kryptoanalyse bei Chiffre 3*

In den Diagrammen ist auf der X-Achse jeweils die Anzahl der Nachrichten-Paare und auf der Y-Achse jeweils die zugehörige prozentuale Erfolgsrate angegeben. Es wurden 200 bis 4.000 Nachrichten-Paare untersucht, wobei sich die Anzahl in den Untersuchungen immer um 200 erhöht. Bei der Va-

riante mit den inkrementierten Nachrichten-Paaren ist eine sehr geringe Erfolgsrate über alle Untersuchungen zu beobachten. Beispielhaft sind die Untersuchungen mit 200 und 400 Nachrichten-Paaren zu nennen, diese hatten keine erfolgreiche Wiederherstellung. Bis 2.400 Nachrichten-Paare steigt die Erfolgsrate langsam an, liegt aber unterhalb von 10 %. Bis 4.000 Nachrichten-Paare steigt diese bis ungefähr 18 % an. Der steigende Trend der Erfolgsrate ist bei 3.000 und 3.600 Paaren nicht zu beobachten, bei diesen Anzahlen ist die Erfolgsrate niedriger geworden. Es kann angenommen werden, dass die Erfolgsrate bei Erhöhung der Paare weiterhin wächst.

Bei der Variante mit den zufälligen Nachrichten-Paaren ist die Erfolgsrate höher als bei den inkrementierten Paaren. Bei 200 Nachrichten-Paaren beträgt diese ungefähr 28 %, bei 400 ungefähr 72 % und bei 600 Nachrichten-Paaren etwas über 90 %. Es ist ein schneller Anstieg der Erfolgsrate zu beobachten. Bei weiterer Erhöhung der Paare steigt die Erfolgsrate weiter an und bewegt sich im Bereich von 95 %. Beim Vergleich zwischen inkrementierten und zufälligen Nachrichten-Paaren fällt auf, dass die Erfolgsrate bei den zufälligen Nachrichten-Paaren wesentlich höher ist. Dies kann darauf zurückzuführen sein, dass die inkrementierten Nachrichten-Paare durch die aufsteigenden Werte der Nachrichten ein sehr ähnliches Bitmuster aufweisen und sich innerhalb der Chiffre sehr ähnlich verhalten.

Aufwand der DKA

Um den Aufwand (siehe Kapitel 3.6) des Angriffs mittels DKA zu bestimmen, muss die Anzahl der verwendeten Nachrichten-Paare berechnet werden. Bei 2.000 zufällig gewählten Nachrichten-Paaren (pro Differenzial) beträgt die Erfolgswahrscheinlichkeit in etwa 95 %. Pro Verschlüsselungsrunde werden drei Differenziale verwendet, um die Schlüsselbits zu erhalten. Folglich beträgt die Anzahl an gewählten Nachrichten-Paaren bei vier Verschlüsselungsrunden $3 \cdot 4 \cdot 2.000 = 24.000$. Für die letzte Verschlüsselungsrunde kommen (aufgerundet) 4 Nachrichten-Paare (siehe Abbildung 45) hinzu. Insgesamt werden für den Angriff also 24.004 Nachrichten-Paare verwendet, das entspricht einem Aufwand von ungefähr 2^{15}.

Vergleich mit Brute-Force

Bei einer Brute-Force-Attacke auf Chiffre 3 sind 2^{96} verschiedene Schlüssel zu untersuchen. Diese ist durchschnittlich nach 2^{95} getesteten Schlüsseln erfolgreich. In diesem Experiment werden die durchschnittlich getesteten Schlüssel bei der DKA von Chiffre 3 untersucht. Chiffre 3 wurde dabei 1.000-mal angegriffen und die Anzahl an getesteten Schlüsseln gezählt. Die Nachrichten zur

Schlüsselwiederherstellung wurden, wie in der Untersuchung zuvor, mittels Zufallszahlengenerator erzeugt. Abbildung 54 zeigt den Vergleich zwischen einem Brute-Force-Angriff und DKA.

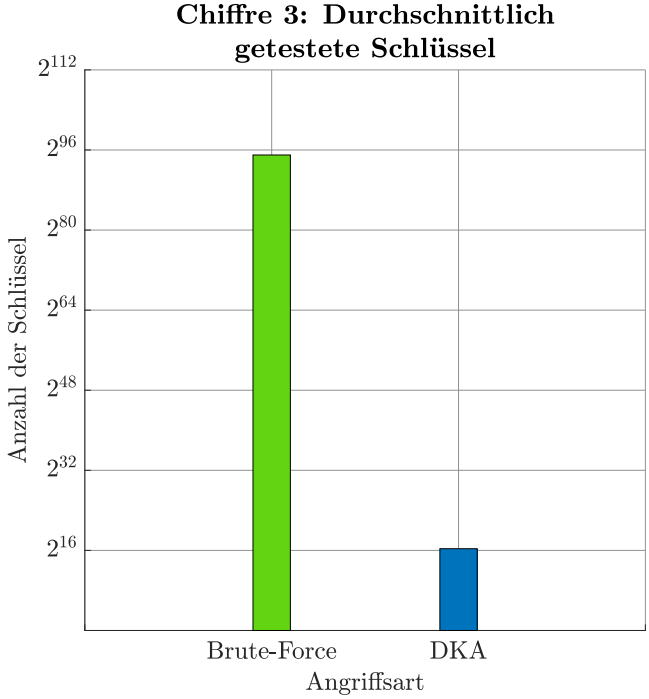


Abbildung 54: *Durchschnittlich durchsuchter Schlüsselraum bei der Schlüsselwiederherstellung bei Chiffre 3*

Auf der X-Achse sind die Angriffsarten (Brute-Force: grün, DKA: blau) abgebildet. Auf der Y-Achse ist die Anzahl der durchschnittlich getesteten Schlüssel als Zweierpotenz zu sehen. Durchschnittlich sind bei dieser Untersuchung 83.144 Schlüssel mittels DKA verwendet worden. Das entspricht in etwa 2^{16} Schlüsseln und ist um den Faktor 2^{79} kleiner als bei einem Brute-Force-Angriff. Der größte Anteil der durchschnittlich zu untersuchenden Schlüssel bei DKA entsteht durch die Wiederersterstellung der Schlüssel k_1 und k_0 (siehe Kapitel 6.1.1).

Weitere Experimente

Weitere Analysen zu Chiffre 3 sind im Anhang A.2 zu finden.

6.2 Analyseergebnisse weiterer Chiffren

In diesem Kapitel werden die Grenzen der Implementierung der DKA anhand von Chiffre 4 gezeigt. Darüber hinaus wird eine weitere, neu konstruierte und bisher nicht analysierte Chiffre mit DKA angegriffen.

6.2.1 Grenzen des Verfahrens am Beispiel von Chiffre 4

In CT2 wurden die Chiffre 1, 2 und 3 implementiert. Bis zu diesem Zeitpunkt konnten Differenziale, Filterungstechnik und die Anzahl der Nachrichten-Paare als Faktor des Erfolgs einer DKA ausgemacht werden. Bei Chiffre 4 hat sich gezeigt, dass es noch weitere Aspekte gibt, die den Erfolg einer DKA beeinflussen. Bei der Analyse des Erfolgs bei der Schlüsselwiederherstellung von Chiffre 4 zeigt sich, dass diese nur mit geringem Erfolg durchführbar ist. In der folgenden Untersuchung steht die Erfolgsrate in Abhängigkeit zu 8 Nachrichten-Paaren. Bei einer 4-Bit Blockchiffre sind $2^4 = 16$ unterscheidbare Nachrichten möglich. Abbildung 55 zeigt das Ergebnis der Untersuchung. Die Untersuchung ist 1.000-mal wiederholt worden, je Wiederholung ist der Schlüssel durch einen Zufallszahlengenerator generiert worden.

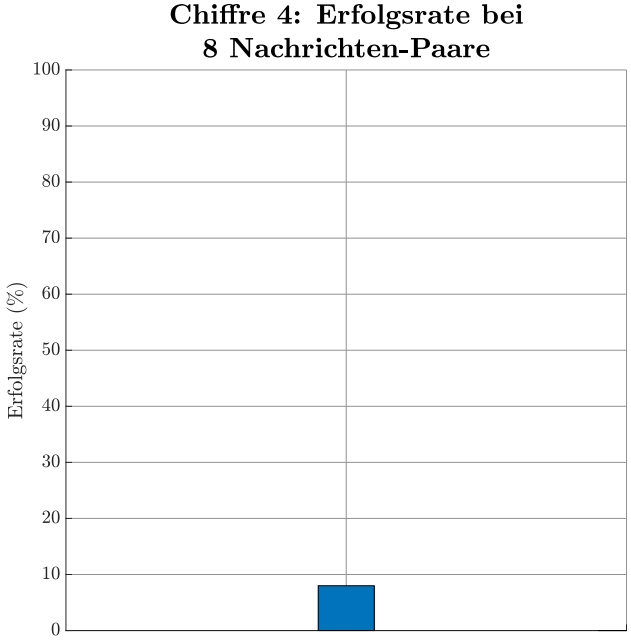


Abbildung 55: *Erfolgsrate der differenziellen Kryptoanalyse bei Chiffre 4*

Auf der Y-Achse ist die Erfolgsrate in Prozent angegeben. Die Abbildung zeigt, dass die DKA von Chiffre 4 in 8 % der Schlüsselwiederherstellungen

erfolgreich verlaufen ist. Da es nicht mehr als 16 verschiedene Nachrichten gibt, kann die Nachrichtenanzahl nicht erhöht werden, um die Erfolgsrate möglicherweise zu verbessern. Möglicherweise ist DKA bei Chiffren mit sehr kleiner Blockgröße nicht sinnvoll anwendbar.

6.2.2 EasyCipher One - eine bisher nicht analysierte Chiffre

Im Zuge der Evaluierung der differenziellen Kryptoanalyse ist vom Betreuer eine Familie von Toy-Chiffren mit dem Namen EasyCipher One erstellt worden. Diese enthält verschiedene SPN-basierte Chiffren mit unterschiedlichen Blockgrößen, Rundenzahlen, S-Boxen und Permutationen. In diesem Abschnitt ist der Angriff auf eine dieser Chiffren mittels differenzieller Kryptoanalyse beschrieben. Die Chiffre heißt Easy Cipher One 16 Bit 16 (EC1) und basiert auf einer Blockbreite von 16 Bit. Ein Block wird dabei in drei Runden verschlüsselt. Die Verschlüsselungsrunden bestehen aus Schlüsseladdition, Substitution und Permutation. In der letzten Runde wird auf die Permutation verzichtet, sodass die Runde aus Schlüsseladdition, Substitution und einer finalen Schlüsseladdition besteht. Diese Chiffre entspricht der in Kapitel 2.3 vorgestellten Struktur von SPN. Der Schlüssel ist 64 Bit groß und wird durch den Key Schedule auf 4 Rundenschlüssel zu je 16 Bit aufgeteilt. Gleichung 29 beschreibt den Schlüssel k für EC1.

$$k = k_0 \,||\, k_1 \,||\, k_2 \,||\, k_3 \tag{29}$$

Die verwendete Permutation P_3 von EC1 ist in Gleichung 30 zu sehen. Zeile 1 beschreibt dabei die Position eines Bits im Eingabeblock und Zeile 2 die neue Position des Bits im Ausgabeblock.

$$P_3 = \begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 \\ 6 & 13 & 2 & 11 & 9 & 10 & 7 & 4 & 8 & 14 & 12 & 3 & 15 & 1 & 0 & 5 \end{pmatrix} \tag{30}$$

Die Substitution erfolgt durch vier parallele S-Boxen, welche jeweils 4 Bit substituieren. Die S-Boxen sind baugleich und substituieren durch dieselbe Vorschrift. Tabelle 12 beschreibt die Substitutionsvorschrift.

Eingabebits	Ausgabebits
0000	1010
0001	0101
0010	1111
0011	1000
0100	1011
0101	0000
0110	0011
0111	0111
1000	0001
1001	1101
1010	1001
1011	1100
1100	0110
1101	1110
1110	0010
1111	0100

Tabelle 12: *Darstellung der Substitutionstabelle für EC1*

Algorithmus 5 beschreibt den Verschlüsselungsprozess eines Klartext-Blocks durch EC1 mit 3 Runden.

Eingabe: e, k
Ausgabe: a
`/* Runden 1 bis 2 */`
for *i = 0 to 1* **do**
  Berechne $e = e \oplus k_i$;
  Teile e in 4 Teilblöcke b_j zu je 4 Bit, sodass $e = b_0 \,||\, b_1 \,||\, b_2 \,||\, b_3$
  Berechne $e = S(b_0) \,||\, S(b_1) \,||\, S(b_2) \,||\, S(b_3)$
  Permutiere jedes Bit e_k, sodass $e = P_3(e)$
end
`/* Letzte Runde */`
Berechne $a = e \oplus k_2$;
Teile a in 4 Teilblöcke b_j zu je 4 Bit, sodass $a = b_0 \,||\, b_1 \,||\, b_2 \,||\, b_3$
Berechne $a = S(b_0) \,||\, S(b_1) \,||\, S(b_2) \,||\, S(b_3)$
Berechne $a = a \oplus k_3$

Algorithmus 5: *Verschlüsselung eines Klartext-Blocks mit EC1 mit 3 Runden*

Abbildung 56 illustriert die Konstruktion von EC1 mit 3 Runden.

Untersuchung der Schlüsselwiederherstellung

In diesem Abschnitt wird die Erfolgsrate einer DKA von EC1 mit 3 Runden untersucht. In Abhängigkeit der Nachrichten-Paar-Anzahl wird der prozentuale Erfolg angegeben. Es werden inkrementierte Nachrichten-Paare und zufällig gewählte Nachrichten-Paare unterschieden. Es sind keine Dubletten oder symmetrische Paare bei dieser Untersuchung erlaubt. Als Differenziale sind die aktiven S-Box-Kombinationen 0001, 0010, 0100 und 1000 verwendet worden. Die angegebene Anzahl von Nachrichten-Paaren ist für jedes verwendete Differenzial generiert worden. Das Resultat zeigt Abbildung 57.

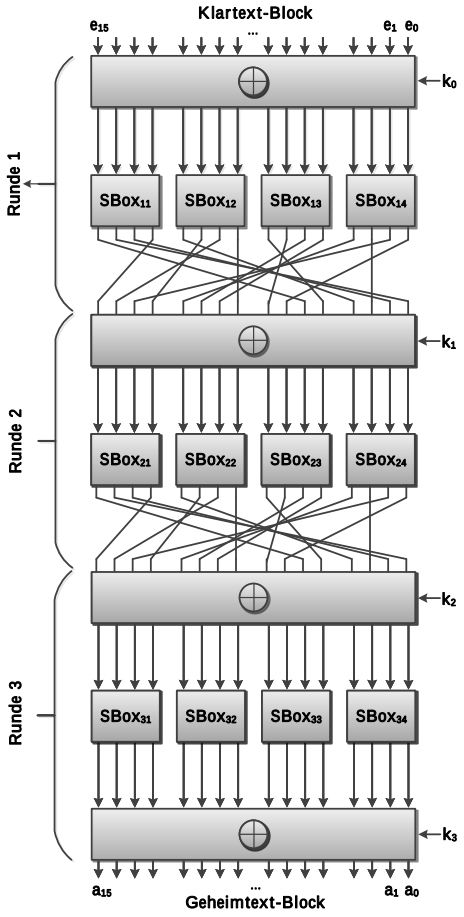
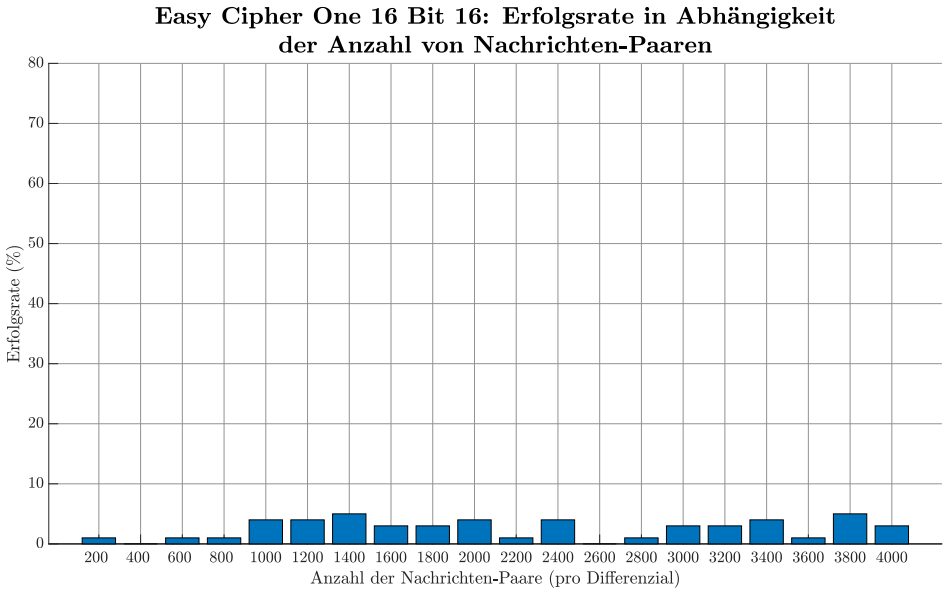
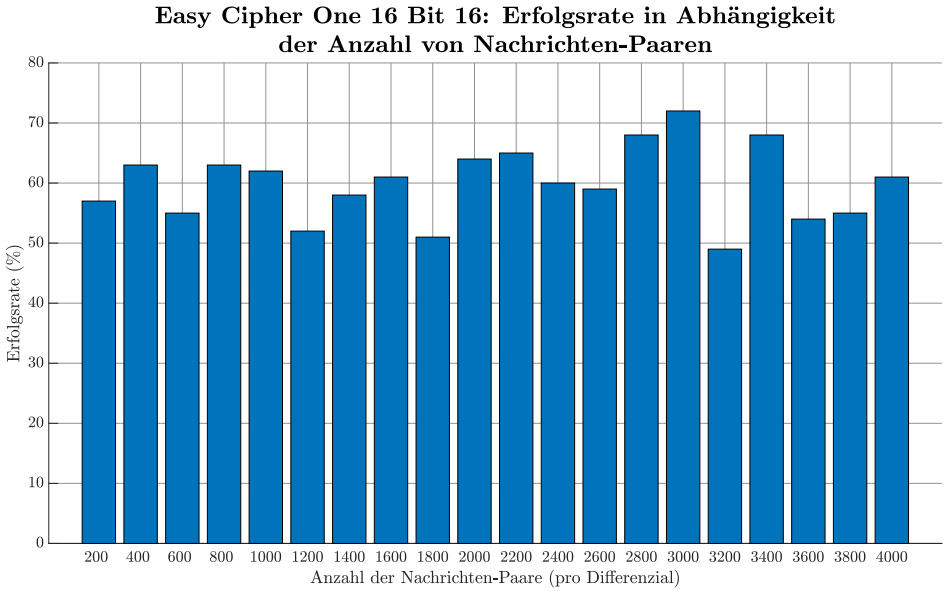


Abbildung 56: *Konstruktion der EC1 mit 3 Runden*

In den Diagrammen ist auf der X-Achse jeweils die Anzahl der Nachrichten-Paare und auf der Y-Achse jeweils die zugehörige prozentuale Erfolgsrate angegeben. Es wurden 200 bis 4.000 Nachrichten-Paare untersucht, wobei sich die Anzahl in den Untersuchungen immer um 200 erhöht. Die Erfolgsrate bei inkrementierten Nachrichten-Paaren ist in Abbildung 57a zu sehen. Die höchste Erfolgsrate in der Untersuchung liegt bei 5 % und ist damit sehr gering ausgefallen. Zwischen 200 und 1.400 Nachrichten-Paaren steigt die Erfolgsrate an und fällt und steigt mit wachsender Anzahl von Paaren immer wieder. Bei zufällig gewählten Nachrichten-Paaren (Abbildung 57b) liegt die höchste Erfolgswahrscheinlichkeit bei 72 %. Zwischen 200 und 3.000 Nachrichten-Paaren wächst die Erfolgswahrscheinlichkeit von etwa 56 % auf bis zu 72 % an. Im Vergleich zu Chiffre 3 nähert sich die Erfolgswahrschein-



(a) *Erfolgsrate mit inkrementierten Nachrichten-Paaren*



(b) *Erfolgsrate mit zufälligen Nachrichten-Paaren*

Abbildung 57: *Erfolgsrate der differenziellen Kryptoanalyse bei EC1*

lichkeit nicht den 100 % an, allerdings könnte in weiteren Untersuchungen die Anzahl der Paare weiter sukzessive gesteigert werden. Sehr wahrscheinlich steigt die Erfolgswahrscheinlichkeit dadurch noch weiter an.

Um den Aufwand des Angriffs mittels DKA zu bestimmen, muss die Anzahl der verwendeten Nachrichten-Paare bestimmt werden. Bei 400 zufällig gewählten Nachrichten-Paaren (pro Differenzial) beträgt die Erfolgswahrscheinlichkeit in etwa 62 %. Bis 4.000 gewählte Paare steigt diese nicht mehr signifikant an. Pro Verschlüsselungsrunde werden 4 Differenziale verwendet, um die Schlüsselbits zu erhalten. Folglich beträgt die Anzahl an gewählten Nachrichten-Paaren bei 2 Verschlüsselungsrunden $4 \cdot 2 \cdot 400 = 3.200$. Für die letzte Verschlüsselungsrunde kommen (aufgerundet) 4 Nachrichten-Paare (siehe Abbildung 45) hinzu. Insgesamt werden für den Angriff also 3.204 Nachrichten-Paare verwendet, das entspricht einem Aufwand von ungefähr 2^{12}.

Bei dieser Chiffre zeigt sich, dass das implementierte Verfahren auch bisher nicht untersuchte Chiffren erfolgreich angreifen kann. Diese Chiffre ist also ebenfalls mittels DKA angreifbar und sollte nicht zur Verschlüsselung von Daten verwendet werden.

6.3 Diskussion der Ergebnisse

In diesem Abschnitt werden die Ergebnisse zusammengefasst und diskutiert. Tabelle 13 illustriert die Angreifbarkeit der Chiffren. Dabei zeigen die zweite und vierte Spalte, wie viele Schlüssel bei dem jeweils angegebenen Angriff (DKA, Brute-Force) durchschnittlich untersucht werden müssen. In der dritten Spalte ist die durchschnittliche Erfolgsrate der DKA zu sehen. Die fünfte Spalte gibt an, ob ein Angriff mittels DKA in Anbetracht der erzielten Ergebnisse sinnvoll ist.

Chiffre	**DKA: Anz. Schlüssel**	**DKA: Erfolgsrate**	**Brute-Force: Anz. Schlüssel**	**DKA sinnvoll?**
Chiffre 1	2^{16}	100%	2^{31}	Ja
Chiffre 2	2^{16}	100%	2^{63}	Ja
Chiffre 3	2^{16}	$\approx 90\%$	2^{95}	Ja
Chiffre 4	2^{6}	8%	2^{15}	Nein
EC1	2^{16}	$\approx 60\%$	2^{63}	Ja

Tabelle 13: *Angreifbarkeit der Chiffren mittels DKA*

Tabelle 13 zeigt, dass die Chiffren 1, 2, 3 und EC1 sinnvoll mittels DKA angreifbar sind. Die Anzahl der zu untersuchenden Schlüssel ist wesentlich kleiner als bei einem Angriff mittels Brute-Force. Des Weiteren ist in den Evaluationen festgestellt worden, dass die Erfolgsraten bei den Chiffren 2 und 3 sehr hoch sind (bei über 90 %) – bei EC1 liegt die Erfolgsrate zwischen 50 % und 72 %. Bei Chiffre 1 konnte der Schlüssel immer berechnet werden. Diese Chiffren sind daher sinnvoll mittels DKA angreifbar. Bei Chiffre 4 liegt die Erfolgsrate bei 8 % mit 8 Nachrichten-Paaren. Bei einer Blockgröße von 4 Bit ist die Anzahl der Nachrichten-Paare auf 8 Paare beschränkt, wenn keine Dubletten erlaubt sind. Aufgrund des kleinen Schlüsselraums wäre hier ein Angriff mittels Brute-Force sinnvoller, da dieser schnell durchführbar wäre und eine höhere Erfolgswahrscheinlichkeit aufweisen würde.

Im Folgenden werden weitere Ergebnisse diskutiert.

Anhand von Chiffre 1 ist festgestellt worden, dass zur Wiederherstellung des ersten und zweiten Rundenschlüssels dieser Chiffre im Durchschnitt 3,7 Nachrichten-Paare notwendig waren. Hierbei spielt die Differenz der gewählten Paare eine Rolle. Offensichtlich ist die durch ein Paar reduzierte Kandidatenmenge in keiner der Wiederholungen eindeutig gewesen. Aus diesem Grund sind immer mindestens zwei Paare notwendig. Da die Paare immer zufällig gewählt wurden, können durch eventuell identische Differenzen der Paare Abweichungen auftreten. Möglicherweise bringen zwei unterschiedliche Paare auch dieselben Schlüsselkandidaten hervor, wodurch auch weitere Paare notwendig werden.

Beim Vergleich mit einer Brute-Force-Attacke sind für Chiffre 1 bei der DKA im Mittel um den Faktor 2^{15} weniger Schlüssel zu prüfen, um zum Erfolg zu kommen. Das liegt hauptsächlich daran, dass bei der DKA nur ein kleiner Teil des gesamten Schlüsselraums durchsucht werden muss, um den Rundenschlüssel k_1 zu erhalten. Wenn k_1 erfolgreich wiederhergestellt ist, kann anschließend der Rundenschlüssel k_0 sofort algebraisch berechnet werden, sodass nur etwas mehr als 2^{16} verschiedene Schlüssel betrachtet werden müssen.

Chiffre 3 ist stellvertretend für SPN-basierte Chiffren mit beliebig vielen Verschlüsselungsrunden untersucht worden. Bei Differenzialen ist festgestellt worden, dass die Wahrscheinlichkeiten steigen, je weniger Verschlüsselungsrunden betrachtet werden. Das liegt daran, dass die Folge der Charakteristiken weniger Elemente und dementsprechend das Produkt der Wahrscheinlichkeiten weniger Faktoren enthält und somit größer wird.

Bei den implementierten Suchstrategien ist festgestellt worden, dass diese

aufgrund ihrer Arbeitsweise unterschiedliche Laufzeiten aufweisen. Grundsätzlich lässt sich die Aussage treffen, dass die Laufzeiten abhängig von der Anzahl an aktiven S-Boxen sind. Dies ist auf den exponentiell anwachsenden Suchraum zurückzuführen. Ein weiterer Faktor ist die Anzahl an Verschlüsselungsrunden. Bei der Suche (auf einem einzelnen Rechner) sollten in diesem interaktiven Tutorial aufgrund der Laufzeiten nicht mehr als zwei S-Boxen gleichzeitig angegriffen werden. Die Ergebnisse der Suchstrategien in Bezug auf deren Wahrscheinlichkeiten sind sehr ähnlich. Bei der Ausführung auf einem leistungsfähigem Cluster und einer dafür optimierten Implementierung können sicherlich auch mehr S-Boxen parallel angegriffen werden.

Bei der Filterung zeigt sich, dass diese einen Einfluss auf die Geschwindigkeit und Qualität der Schlüsselwiederherstellung hat. Das ist darauf zurückzuführen, dass durch die Filterung falsche Nachrichten-Paare entfernt werden, sodass diese bei der Schlüsselwiederherstellung nicht mehr verwendet werden müssen. Falsche Nachrichten-Paare erzielen häufiger Treffer bei falschen Schlüsselkandidaten – eine Entfernung vor der Schlüsselwiederherstellung verbessert damit das Ergebnis.

Bei der Schlüsselwiederherstellung ist die Dauer in Abhängigkeit der Anzahl der Nachrichten-Paare untersucht worden. Je weniger aktive S-Boxen gesetzt sind, desto schneller verläuft die Schlüsselwiederherstellung. Sind bei der Schlüsselwiederherstellung drei statt einer S-Boxen aktiv, dauert es ca. den Faktor 20 länger. Das liegt daran, dass die Anzahl der Schlüsselkandidaten mit der Anzahl der aktiven S-Boxen exponentiell wächst.

Die Erfolgsrate der DKA ist bei zufällig gewählten Paaren höher als bei inkrementierten Paaren. Möglicherweise liegt dies darin begründet, dass inkrementierte Paare ein ähnliches Bitmuster aufweisen und sich daher innerhalb der Verschlüsselungsrunden der Chiffre vergleichbar verhalten. Eventuell verhält sich das Ergebnis anders in Abhängigkeit eines bestimmten Startwertes für die Nachrichten-Paare (hier wurde immer ab dem Wert „1“ begonnen). Darüber hinaus lässt sich die Erfolgsrate durch die Anzahl der Nachrichten-Paare verbessern. Bei Chiffre 2 und 3 steigert sich der Erfolg kontinuierlich bis zu 100 %, bei EC1 konnte das nicht festgestellt werden. Möglicherweise kann dies auf die verschiedenen S-Boxen, Permutationen und Anzahl der Verschlüsselungsrunden zurückgeführt werden. Diese Einflussfaktoren könnten in zukünftigen Arbeiten untersucht werden.

Für eine erfolgreiche Brute-Force-Attacke sind bei Chiffre 3 im Mittel 2^{95} Schlüssel zu testen, für eine erfolgreiche DKA sind im Mittel etwas mehr als 2^{16} Schlüssel analysiert worden. Damit werden durch die DKA um den

Faktor 2^{79} weniger Schlüssel verwendet als bei einem Brute-Force-Angriff. Bei Chiffre 3 war der Vorteil der DKA gegenüber einer Brute-Force-Analyse am deutlichsten.

Die Grenzen der DKA-Implementierung konnten an Chiffre 4 festgestellt werden. Diese Chiffre war mittels DKA nicht sehr erfolgreich angreifbar, der Erfolg lag bei 8 %. Möglicherweise liegt das an der Blockgröße von 4 Bit; alle anderen Chiffren haben eine Blockgröße von 16 Bit. Für Chiffre 4 ist die DKA ungeeignet, aber hier ist die Brute-Force-Analyse auch völlig ausreichend aufgrund der kurzen Schlüssellänge. Welche konkreten Grenzen nach oben die DKA hat, wenn bei größeren Chiffren die Brute-Force-Analyse keinen Erfolg mehr hat, müsste noch im Detail in zukünftigen Arbeiten untersucht werden.

Abschließend wird der Aufwand der Angriffe auf die verschiedenen Chiffren mittels DKA verglichen (Tabelle 14).

Chiffre	**DKA: Aufwand**
Chiffre 1	2^2
Chiffre 2	2^8
Chiffre 3	2^{15}
Chiffre 4	2^3
EC1	2^{12}

Tabelle 14: *Vergleich des Aufwands der Angriffe auf die verschiedenen Chiffren mittels DKA (benötigte Nachrichten-Paare)*

Bei der Betrachtung von Tabelle 14 ist zu sehen, dass der Aufwand der DKA bei Chiffre 1 am geringsten ist. Für diese Chiffre werden im Durchschnitt 4 Nachrichten-Paare benötigt, um die zwei Rundenschlüssek k_1 und k_0 wiederherzustellen. Für Chiffre 2 sind pro angegriffener Verschlüsselungsrunde $r > 1$ zwei Differenziale und 50 Nachrichten-Paare verwendet worden. Beim Angriff auf die letzte Verschlüsselungsrunde kommen im Mittel 4 weitere Nachrichten-Paare hinzu. Die Erfolgsrate lag dabei bei 100 %. Daraus ergibt sich ein Aufwand von 2^8 für die DKA bei Chiffre 2. Bei der DKA von Chiffre 3 sind ungefähr 2^{15} Nachrichten-Paare notwendig gewesen, um die Rundenschlüssel mit einer Erfolgswahrscheinlichkeit von etwa 95 % zu erhalten. Der Aufwand bei Chiffre 4 beträgt 2^3 und bei Chiffre EC1 2^{12}.

7 Verwandte Arbeiten

In diesem Kapitel wird ein Überblick über die verwendete Literatur gegeben. Als Standardwerk für symmetrische Blockchiffren ist das Buch „The Block Cipher Companion“ ([21]) von Lars Knudsen und Matthew Robshaw zu nennen. Dieses Buch gibt einen sehr umfangreichen und grundlegenden Überblick über symmetrische Blockchiffren und verschiedene Angriffe. Der Angriff der differenziellen Kryptoanalyse wird grundlegend und für unerfahrene Leser dieses Gebiets beschrieben. Es gibt Messwerte und Ergebnisse zu einzelnen praktischen Experimenten. Die in dieser Masterarbeit verwendeten Chiffren 3 und 4 entstammen diesem Buch. Chiffre 1 hingegen entstammt nicht diesem Buch, ist allerdings vergleichbar mit „CIPHERONE“ aus diesem Buch. Fig. 6.10 auf Seite 126 des Buches zeigt ein Experiment über die Erfolgsrate zu der Wiederherstellung der Schlüsselbits 4-8 von k_5 der Chiffre, die in dieser Masterarbeit Chiffre 3 heißt. Die dort benannten Werte konnten im Rahmen dieser Masterarbeit nicht vergleichbar erfüllt werden. Nach Kontaktaufnahme mit dem Autor Lars Knudsen hat dieser per E-Mail bestätigt, dass die in dieser Arbeit erreichten Ergebnisse ebenfalls valide sind. Knudsen konnte den konkreten Aufbau des damaligen Experiments nicht mehr mitteilen, allerdings stellte er die Vermutung an, dass die Generierung der Nachrichten-Paare mit anderen Zufallszahlengeneratoren durchgeführt wurde. Darüber hinaus konnte kein Quellcode von Knudsen geteilt werden.

Ein bekanntes Tutorial im Bereich differenzieller Kryptoanalyse ist die Arbeit „A Tutorial on Linear and Differential Cryptanalysis“ ([30]) von Howard Heys. Dieses illustriert grundlegend und praktisch, wie differenzielle Kryptoanalyse durchführbar ist. Innerhalb des Papers gibt es ein Experiment zur Wiederherstellung des letzten Rundenschlüssels der dort verwendeten Chiffre. Diese konnten in ersten Experimenten in dieser Masterarbeit ebenfalls erreicht werden. Darüber hinaus konnte in einem Kontakt mit dem Autor kein Quellcode zum Vergleich erhalten werden.

Als dritte Arbeit ist das Paper „DIFFERENTIAL CRYPTANALYSIS FOR A 3-ROUND SPN“ ([27]) zu nennen. Die dort beschriebene Chiffre ist in dieser Arbeit als Chiffre 2 enthalten. Auch in dieser Arbeit ist kein Quellcode veröffentlicht worden.

Jon King betreibt eine Website mit verschiedenen Themen aus den Bereichen Kryptographie und Kryptoanalyse ([32]). Dieser hat eine Reihe von Artikeln über differenzielle Kryptoanalyse verfasst, in denen der Angriff auf zwei Toy-Chiffren und FEAL beschrieben ist. Neben der textuellen Beschreibung des Angriffs gibt es C-Quellcode für Konsolenanwendungen zum Angriff auf die

Chiffren. Die Programme haben keine graphische Benutzerschnittstelle und sind nicht interaktiv. Der Angriff verwendet fest definierte Charakteristiken im Quellcode und sucht nicht nach diesen. Darüber hinaus ist der Quellcode nicht kommentiert und es gibt keine Literaturverweise auf der Website. Zudem sind keine umfangreichen Evaluationen zu finden.

Resümierend ist zusammenzufassen, dass in der öffentlichen Literatur wenige Werke zu finden sind, die das Verfahren der differenziellen Kryptoanalyse grundlegend und praktisch demonstrieren. Zudem ist es schwierig, Quellen und Dokumentationen von Quellcode von Experimenten zu erhalten. Ein interaktives Tutorial oder Programm mit graphischer Oberfläche konnte nicht gefunden werden.

8 Fazit und Ausblick

In diesem Kapitel wird das Fazit gezogen und ein Ausblick auf mögliche zukünftige Arbeiten gegeben.

8.1 Fazit

Die in Kapitel 1.2 formulierten Ziele werden im Folgenden überprüft. Es sollte das Verfahren der differenziellen Kryptoanalyse (DKA) untersucht werden (**Z1**). Auf Basis von Z1 sollte ein Tutorial geplant und in CT2 erstellt werden (**Z2**). Die mit Z2 zu erarbeitende Implementierung sollte in der Lage sein, alle Rundenschlüssel der Chiffren des Tutorials wiederherzustellen (**Z3**). Weiterhin sollte auf Basis der Implementierung das Verfahren der DKA evaluiert werden (**Z4**). Zuletzt sollte die DKA mit dem Brute-Force-Angriff verglichen werden (**Z5**).

Z1: Untersuchung des Verfahrens der differenziellen Kryptoanalyse

Die DKA wurde als mathematisch-analytischer Angriff untersucht (siehe Kapitel 3). Dabei sind die grundlegenden Strukturen von symmetrischen Blockchiffren in Hinblick auf DKA analysiert worden. Diese sind Schlüsseladdition, Substitution und Permutation. Das Verfahren dient zur Wiederherstellung von einzelnen Bits bis zu den vollständigen Rundenschlüsseln. Dazu werden Charakteristiken bzw. Differenziale gesucht, um auf deren Basis Nachrichten-Paare zu generieren, deren Veränderung der Differenzen durch die einzelnen Verschlüsselungsrunden einer Chiffre mit einer Wahrscheinlichkeit vorausgesagt werden kann. Bei ausreichend hohen Wahrscheinlichkeiten und vielen Nachrichten-Paaren, können die Rundenschlüssel sukzessive wiederhergestellt werden. Darüber hinaus wurden der Aufwand, Faktoren des Erfolgs und weitere Optimierungsmöglichkeiten des Verfahrens untersucht.

Z2: Design und Aufbau eines Tutorials für differenzielle Kryptoanalyse mit drei Toy-Chiffren in CT2

Bisher gab es keine Software, die DKA interaktiv und Step-by-Step präsentiert. In dieser Masterarbeit ist ein Tutorial zur DKA mit drei unterschiedlichen Toy-Chiffren erstellt worden. Dieses vermittelt die Grundlagen des Verfahrens mittels Slides kleinschrittig und interaktiv (siehe Kapitel 4). Die einzelnen Schritte einer DKA sind auf vier einzelne Komponenten aufgeteilt worden, um Anwendern den Einstieg durch eine didaktische Reduktion zu erleichtern.

Z3: Rekonstruktion der Rundenschlüssel der Toy-Chiffren mittels differenzieller Kryptoanalyse in CT2

Die Implementierung ist zunächst als Konsolen-Anwendung geplant und realisiert worden. Auf Basis dieser Implementierung ist die CT2-Implementierung umgesetzt worden. Die Implementierungen sind in der Lage, alle Rundenschlüssel der drei Toy-Chiffren wiederherzustellen (siehe Kapitel 5). Sollte beispielsweise aufgrund zu weniger Nachrichten-Paare die Schlüsselwiederherstellung nicht vollständig funktionieren, wird der Anwender darüber informiert. Dieser kann anschließend die DKA mit neuen Parametern (beispielsweise mit erhöhter Nachrichten-Paar-Anzahl) neu starten.

Z4: Analyse und Evaluation der differenziellen Kryptoanalyse auf Basis der Implementierung

Auf Basis der Implementierung sind verschiedene Aspekte der DKA untersucht worden (siehe Kapitel 6). Bei Chiffre 1 konnte festgestellt werden, dass zur Schlüsselwiederherstellung im Durchschnitt 3,7 Nachrichten-Paare notwendig sind. Die implementierten Suchverfahren für Charakteristiken bzw. Differenziale haben unterschiedliche Laufzeiten. Die Suchstrategie TGS weist für eine, zwei und drei aktive S-Boxen mit 4,25 s, 20,83 s und 5 min und 12 s jeweils die besten Laufzeiten auf. Für eine und zwei aktive S-Boxen schneidet die Suchstrategie TGM mit 1 min und 26 s und 14 min und 45 s am schlechtesten ab. Mit 42 min und 7 s ist die Suchstrategie HGS bei drei aktiven S-Boxen am langsamsten. Bei vier aktiven S-Boxen ist die Suchstrategie TGM mit 67 min und 58 s am schnellsten und HGS mit 15 h, 55 min und 50 s am langsamsten. Die Wahrscheinlichkeiten der gefundenen Charakteristiken bzw. Differenziale unterscheiden sich lediglich geringfügig. Allgemein lässt sich aus den Messergebnissen ableiten: Je mehr S-Boxen aktiv sind, desto größer ist der Suchraum für Charakteristiken und in Folge dessen dauert die Suche länger. Die Filterung der Nachrichten-Paare ist ein großer Faktor bei der Dauer der Schlüsselwiederherstellung. In Abhängigkeit der Anzahl der aktiven S-Boxen kann die Dauer um einen Faktor zwischen 1 und 20 reduziert werden. Darüber hinaus verbessert die Filterung das Ergebnis der Schlüsselwiederherstellung qualitativ. Beispielsweise haben manche falschen Schlüsselkandidaten keine Treffer. Ohne Filter hatten die falschen Schlüsselkandidaten 10,7 Treffer und mit Filter lediglich 1,7 Treffer. Die Erfolgsaussicht bei der Schlüsselwiederherstellung ist abhängig von der Anzahl der Nachrichten-Paare. Bei Chiffre 3 ist die Erfolgsaussicht bei 200 Nachrichten-Paaren bei etwa 25 %, bei 400 Paaren bei etwa 70 % und steigt mit wachsender Anzahl auf bis zu 95 %. Resümierend bleibt festzuhalten, dass Chiffre

1, 2 und 3 mittels DKA sehr gut angreifbar sind. Bei Chiffre 4 funktioniert die DKA nicht zuverlässig. Die Chiffre EC1 konnte mittels DKA gebrochen werden, allerdings waren die Erfolgsraten im Vergleich zu den Chiffren 1-3 nicht so hoch. Faktoren des Erfolgs sind die S-Boxen, die Permutation und die Rundenanzahl.

Z5: Vergleich der differenziellen Kryptoanalyse mit anderen Angriffen

Im Rahmen der Evaluation (siehe Kapitel 6) wurde das Verfahren der DKA mit einem Brute-Force-Angriff verglichen. Bei der Wiederherstellung der Rundenschlüssel von Chiffre 1 mit je 16 Bit sind bei einem Brute-Force-Angriff im Mittel 2^{31} Schlüssel zu prüfen. Bei der DKA ist diese Anzahl um den Faktor 2^{15} auf 2^{16} reduziert worden. Chiffre 3 hat eine Gesamt-Schlüssellänge von 96 Bit. Beim Angriff mit Brute-Force sind im Mittel 2^{95} Schlüssel zu testen. Damit ist die Anzahl der verwendeten Schlüssel durch DKA um den Faktor 2^{79} verringert worden.

In dieser Arbeit wurden die folgenden praktischen Ergebnisse erreicht:

- Implementierung von Suchverfahren (basierend auf Tiefensuche und einer Heuristik) für Charakteristiken (Kapitel 4.2)
- Implementierung einer Konsolenanwendung für DKA (Kapitel 5.1)
- Implementierung DKA-PfadFinder-Komponente in CT2 (Kapitel 5.2)
- Implementierung DKA-PfadVisualisierer-Komponente in CT2 (Kapitel 5.2)
- Implementierung DKA-KeyRecovery-Komponente in CT2 (Kapitel 5.2)
- Implementierung DKA-Orakel-Komponente in CT2 (Kapitel 5.2)
- Implementierung DKA-ToyCipher-Komponente in CT2 (Kapitel 5.2)

In dieser Arbeit wurden die folgenden theoretischen Ergebnisse erreicht:

- Design und Konzept eines Tutorials für DKA in CT2 (Kapitel 5.2)
- Evaluation des Angriffs der DKA(Kapitel 3)

- Evaluation der Anzahl der Nachrichten-Paare zur Wiederherstellung der Rundenschlüssel der ersten Verschlüsselungsrunde (Kapitel 6)
- Evaluation der implementierten Suchverfahren (Kapitel 6)
- Evaluation der Filterung (Kapitel 6)
- Evaluation der Dauer der Schlüsselwiederherstellung (Kapitel 6)
- Evaluation der Erfolgsrate der Schlüsselwiederherstellung (Kapitel 6)
- Vergleich von DKA und Brute-Force (Kapitel 6)

8.2 Ausblick

Mithilfe der im Rahmen dieser Masterarbeit entstandenen Implementierung können Anwender den Angriff der differenziellen Kryptoanalyse kennen lernen. Das Tutorial ist so aufgebaut, dass Anwender mit und ohne Vorwissen gleichermaßen die Komponenten verwenden können. Darüber hinaus wurde die Architektur so konzeptioniert, dass weitere Chiffren leicht integriert werden können. Aktuell enthält die Implementierung drei Chiffren.

Auf Basis der Implementierung könnten weitere Analysen angestrebt werden, um ein besseres Verständnis der Parameter zu erlangen, die den Erfolg einer DKA beeinflussen. Es könnte untersucht werden, welche Faktoren die Erfolgsaussichten bei den vorliegenden Chiffren verändern. Dabei könnte analysiert werden, wieso beispielsweise Chiffre 4 sehr schlecht angreifbar ist und die EC1 im Vergleich zu Chiffre 3 auch schlechtere Erfolgsaussichten hat. Darüber hinaus könnte in Betracht gezogen werden, die Implementierung in Forschung und Lehre einzusetzen. In Benutzerstudien könnte man untersuchen, inwieweit das Tutorial Anwender beim Verständnis von DKA unterstützt. Konkret könnte so eine Studie evaluieren, ob die Aufteilung des Angriffs auf verschiedene CT2-Komponenten sinnvoll ist.

Die vorliegende Implementierung ist als Framework für differenzielle Kryptoanalyse entwickelt worden. In weiteren Arbeiten könnten auf Basis dieser Entwicklung weitere Chiffren integriert werden. Aktuell sind SPN-basierte Chiffren enthalten, zukünftig könnten aber beispielsweise auch Feistel-Netzwerk-basierte Chiffren integriert werden. So wäre es interessant, wenn zum Beispiel die bereits in CT2 enthaltene FEAL-Implementierung mittels DKA angegriffen werden würde. FEAL ist eine bekannte symmetrische Blockchiffre, die in der Realität bereits eingesetzt wurde, allerdings hochgradig anfällig für DKA ist.

Literatur

[1] Christof Paar und Jan Pelzl. *Kryptografie verständlich – Ein Lehrbuch für Studierende und Anwender*. 1. Aufl. 2016. Berlin Heidelberg New York: Springer-Verlag, 2016. ISBN: 978-3-662-49297-0.

[2] Cisco Systems Inc. *IoT applications will represent more than 50 Percent of global devices and connections by 2021*. Letzter Zugriff: 22 Mai 2019. URL: https://newsroom.cisco.com/press-release-content?type=webcontent&articleId=1853168.

[3] Adi Biham Eli und Shamir. „Differential cryptanalysis of DES-like cryptosystems“. In: *Journal of Cryptology* 4.1 (Jan. 1991), S. 3–72. ISSN: 1432-1378. DOI: 10.1007/BF00630563. URL: https://doi.org/10.1007/BF00630563.

[4] Jonathan Katz und Yehuda Lindell. *Introduction to Modern Cryptography, Second Edition*. 2. Aufl. Boca Raton, Fla: CRC Press, 2014. ISBN: 978-1-466-57027-6.

[5] Chunning Zhou u. a. „Improving the MILP-based Security Evaluation Algorithms against Differential Cryptanalysis Using Divide-and-Conquer Approach“. In: *IACR Cryptology ePrint Archive* 2019 (2019), S. 19.

[6] Workgroup for Symmetric Cryptography. *Symmetric Cryptography*. Letzter Zugriff: 22 Juni 2019. URL: http://www.crypto.rub.de/resgroups/symcrypt/index.html.en.

[7] University of Haifa, Department Of Computer Science. *Faculty*. Letzter Zugriff: 22 Juni 2019. URL: https://cs.hevra.haifa.ac.il/index.php/en/people-en/faculty-en.

[8] Nils Kopal u. a. „CrypTool 2.0“. In: *Datenschutz und Datensicherheit – DuD* 38.10 (Okt. 2014), S. 701–708. ISSN: 1862-2607. DOI: 10.1007/s11623-014-0274-7. URL: https://doi.org/10.1007/s11623-014-0274-7.

[9] Alexander Hirsch. *Implementierung und Visualisierung von Formaterhaltenden Verschlüsselungsverfahren in CrypTool 2*. 2018.

[10] Olaf Versteeg. *Visualisierung und Implementierung von Betriebsmodi von Blockchiffren für CrypTool 2*. 2018.

[11] Thomas Theis. *Einstieg in C# mit Visual Studio 2017 – Ideal für Programmiereinsteiger*. Bonn: Rheinwerk Verlag, 2017. ISBN: 978-3-836-24495-4.

[12] Thomas Claudius Huber. *Windows Presentation Foundation – Das umfassende Handbuch zur WPF, aktuell zu .NET 4.6/5.0 und Visual Studio 2015*. 4. Aufl. Bonn: Rheinwerk, 2015. ISBN: 978-3-836-23756-7.

[13] CrypTool 2. *CrypTool 2 Entwicklungstipps*. Letzter Zugriff: 9 Juni 2019. URL: `https://www.cryptool.org/trac/CrypTool2/wiki/IPluginHints`.

[14] Friedrich L. Bauer. *Entzifferte Geheimnisse – Methoden und Maximen der Kryptologie*. Berlin Heidelberg New York: Springer-Verlag, 2013. ISBN: 978-3-642-97972-9.

[15] Ralf Küsters und Thomas Wilke. *Moderne Kryptographie – Eine Einführung*. 1. Aufl. Berlin Heidelberg New York: Springer-Verlag, 2011. ISBN: 978-3-834-88288-2.

[16] Wolfgang Ertel und Ekkehard Löhmann. *Angewandte Kryptographie*. Carl Hanser Verlag GmbH Co KG, 2018. ISBN: 978-3-446-45704-1.

[17] Karl Christoph Ruland. *Vorlesung Kryptographische Verfahren und Anwendungen WS 2015/2016*. 2015.

[18] Albrecht Beutelspacher. *Kryptologie – Eine Einführung in die Wissenschaft vom Verschlüsseln, Verbergen und Verheimlichen*. 10. Aufl. Berlin Heidelberg New York: Springer-Verlag, 2014. ISBN: 978-3-658-05976-7.

[19] U.S. NIST. *DATA ENCRYPTION STANDARD (DES)*. Letzter Zugriff: 25 Juni 2019. URL: `https://csrc.nist.gov/csrc/media/publications/fips/46/3/archive/1999-10-25/documents/fips46-3.pdf`.

[20] U.S. NIST. *Announcing the ADVANCED ENCRYPTION STANDARD (AES)*. Letzter Zugriff: 25 Juni 2019. URL: `https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197.pdf`.

[21] Lars R. Knudsen und Matthew Robshaw. *The Block Cipher Companion*. 2011. Berlin Heidelberg: Springer Science & Business Media, 2011. ISBN: 978-3-642-17342-4.

[22] Bruce Schneier. *Angewandte Kryptographie – Protokolle, Algorithmen und Sourcecode in C*. München: Pearson Studium, 2006. ISBN: 978-3-827-37228-4.

[23] Michael Miller. *Symmetrische Verschlüsselungsverfahren: Design, Entwicklung und Kryptoanalyse klassischer und moderner Chiffren*. Teubner, 2003. ISBN: 3-519-02399-7.

[24] Claude E. Shannon. „Communication Theory of Secrecy Systems“. In: *Bell System Technical Journal, Vol 28, pp. 656–715* (1949).

[25] Bruce Schneier. „A Self-Study Course in Block-Cipher Cryptanalysis“. In: *Cryptologia* 24.1 (Jan. 2000), S. 18–33. ISSN: 0161-1194. DOI: 10.1080/0161-110091888754. URL: http://dx.doi.org/10.1080/0161-110091888754.

[26] Wolfgang Ertel und Löhmann Ekkehard. *Angewandte Kryptographie*. Carl Hanser Verlag GmbH Co KG, 2018. ISBN: 978-3-446-45704-1.

[27] Muharrem Tolga Sakallı u. a. *DIFFERENTIAL CRYPTANALYSIS FOR A 3-ROUND SPN*. URL: https://pdfs.semanticscholar.org/dee2/23a06f16f6d6939b363d56b6af5a2c9edbee.pdf.

[28] Eli Biham und Adi Shamir. *Differential Cryptanalysis of the Data Encryption Standard*. 2009. URL: http://www.cs.technion.ac.il/~biham/Reports/differential-cryptanalysis-of-the-data-encryption-standard-biham-shamir-authors-latex-version.pdf.

[29] Maciej Misztal. *The signal to noise ratio in the differential cryptanalysis of 9 rounds of data encryption standard*. 2006. URL: https://pdfs.semanticscholar.org/a109/b45f4c07a40c86e8f110ec5023a90bb6c389.pdf.

[30] Howard M. Heys. „A Tutorial on Linear and Differential Cryptanalysis“. In: *Cryptologia* 26.3 (Juli 2002), S. 189–221. ISSN: 0161-1194. DOI: 10.1080/0161-110291890885. URL: http://dx.doi.org/10.1080/0161-110291890885.

[31] Microsoft. *Random.Next Method*. Letzter Zugriff: 25 September 2019. URL: https://docs.microsoft.com/de-de/dotnet/api/system.random.next?view=netframework-4.8.

[32] Jon King. *Differential Cryptanalysis Tutorial*. Letzter Zugriff: 16 Oktober 2019. URL: http://theamazingking.com/crypto-diff.php.

[33] CrypTool 2. *CrypTool 2 Codeverwaltung*. Letzter Zugriff: 4 November 2019. URL: https://www.cryptool.org/svn/CrypTool2.

Abkürzungsverzeichnis

ACT Alle Charakteristiken mittels Tiefensuche

AES Advanced Encryption Standard

CBC Cipher Block Chaining

CT2 CrypTool 2

CT CrypTool

DCA Differential Cryptanalysis

DES Data Encryption Standard

DKA Differenzielle Kryptoanalyse

EC1 Easy Cipher One 16 Bit 16

ECB Electronic Code Book

FIPS Federal Information Processing Standard

HGS Beste Characteristik (Heuristik, globaler Schwellwert), dann Differenzialsuche

IV Initialisierungsvektor

IoT Internet of Things

LSB least significant bit

MILP Mixed Integer Linear Programming

MSB most significant bit

NBS National Bureau of Standard

NB Nightly Build

NSA National Security Agency

S-Box Substitutions-Box

SAN Storage Area Network

SB Stable Build

SPN Substitutions-Permutations-Netzwerk

SSD Solid State Drive

TGM Beste Characteristik (Tiefensuche, globales Maximum), dann Differenzialsuche

TGS Beste Characteristik (Tiefensuche, globaler Schwellwert), dann Differenzialsuche

VM Virtuelle Maschine

XOR Exklusive-Oder

Notationsverzeichnis

m Klartext

c Geheimtext

k Schlüssel

r Runde

ENC() Verschlüsselung

DEC() Entschlüsselung

f() Abbildung

S() S-Box

P() Permutation

P_{Char} Wahrscheinlichkeit einer Charakteristik

$(e_1, e_2, ..., e_n)$ Eingabeblock

$(a_1, a_2, ..., a_n)$ Ausgabeblock

Abbildungsverzeichnis

Liste der Algorithmen

Listings

Tabellenverzeichnis

Anhang

A Weitere Analysen

A.1 Analysen zu Chiffre 2

A.1.1 Gefundene Differenziale

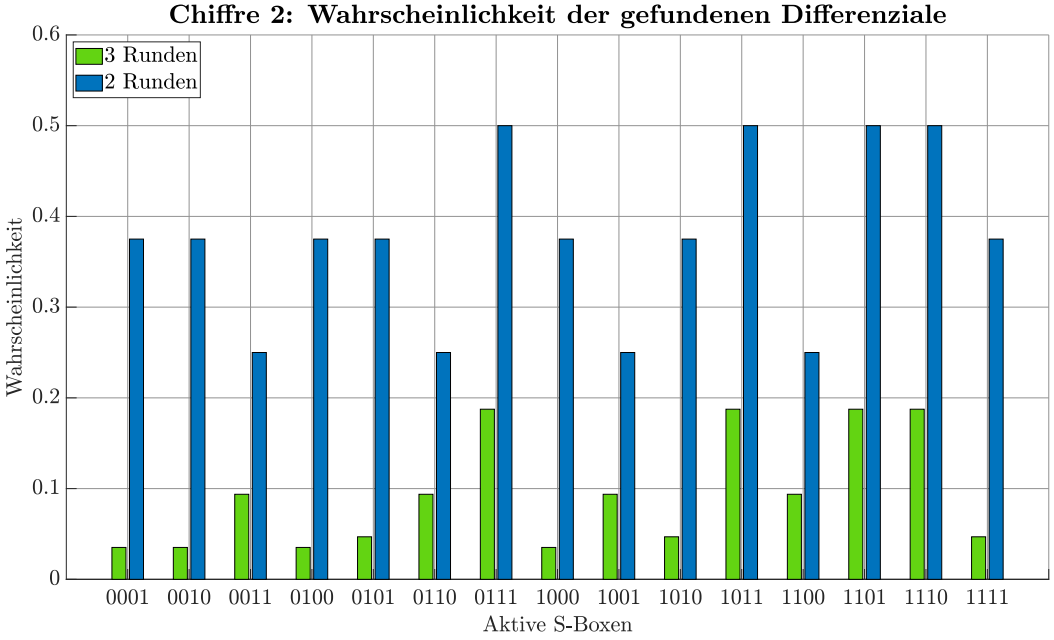


Abbildung 58: *Wahrscheinlichkeiten der gefundenen Differenziale nach Runde 3 bei Chiffre 2*

A.1.2 Untersuchung der Suchstrategien

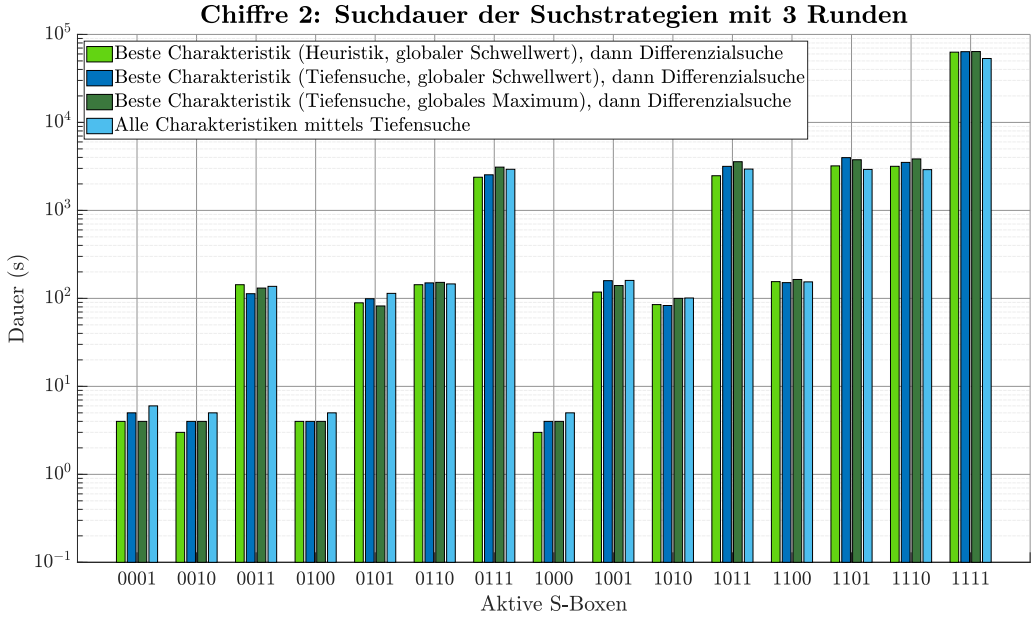


Abbildung 59: *Dauer der Suchstrategien mit 3 Runden bei Chiffre 2*

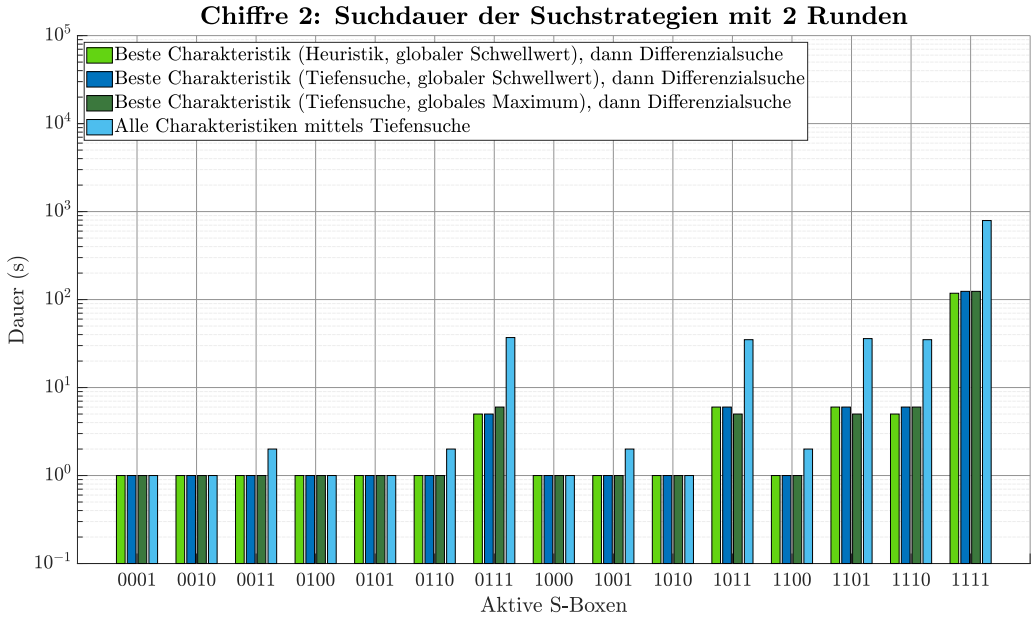


Abbildung 60: *Dauer der Suchstrategien mit 2 Runden bei Chiffre 2*

A.1.3 Untersuchung der Filterung

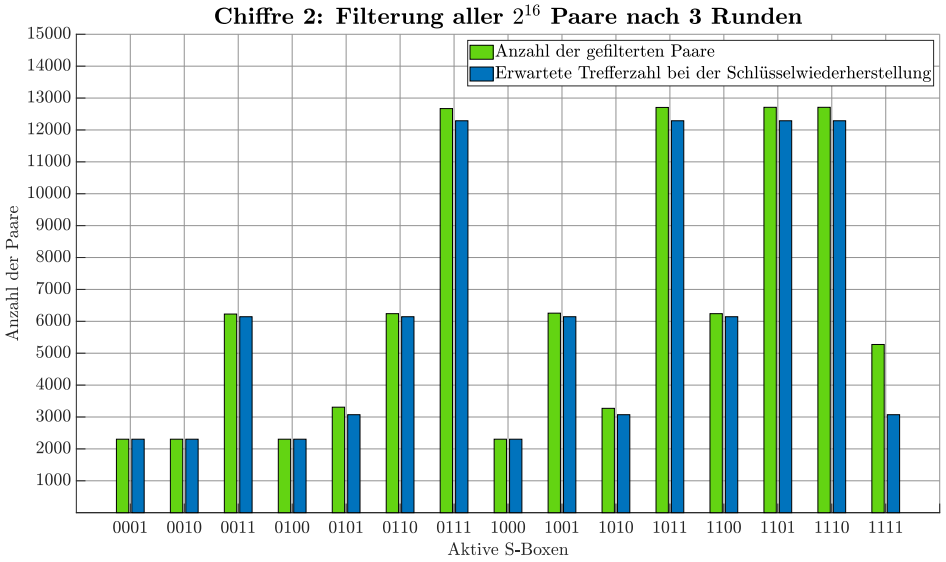
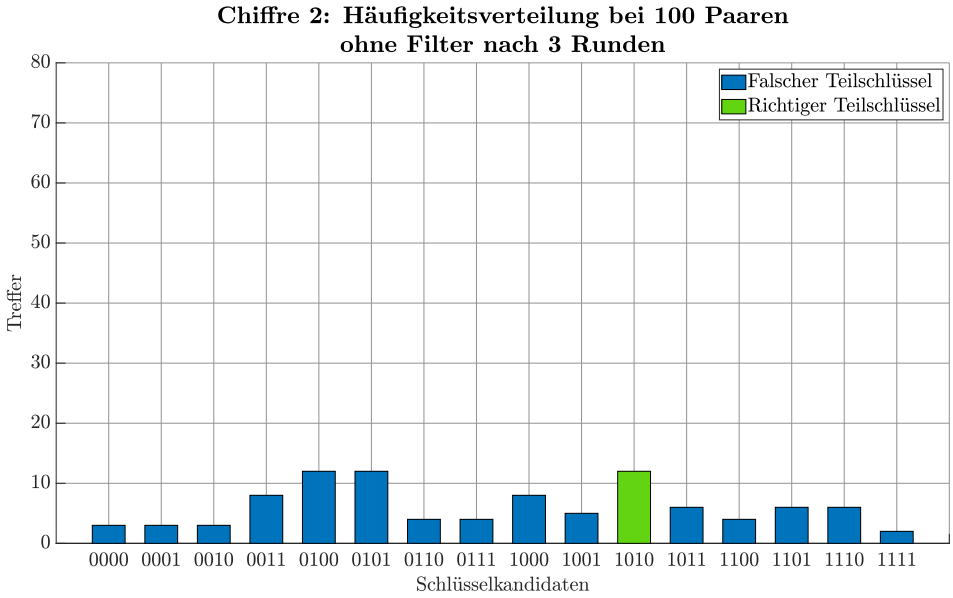
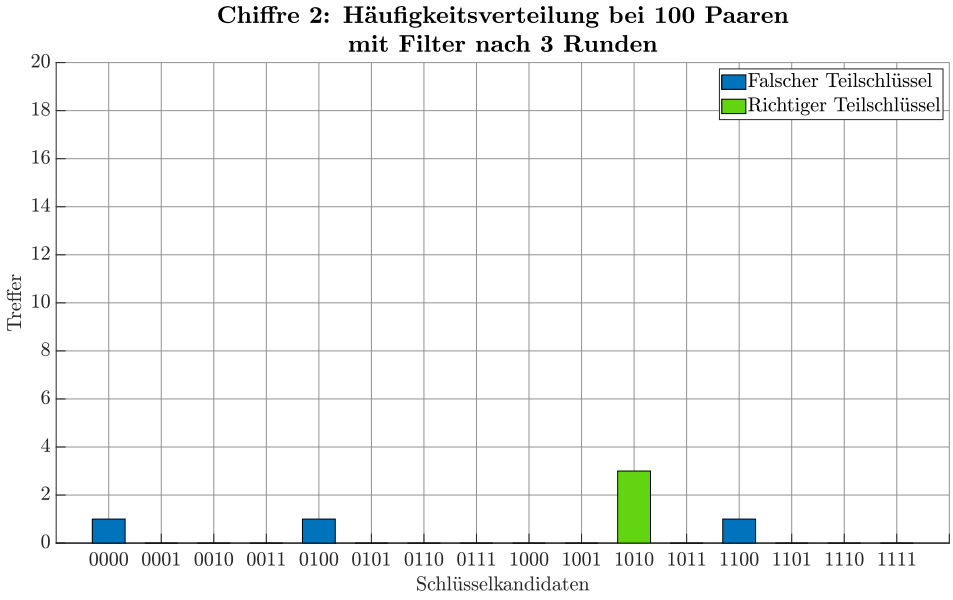


Abbildung 61: *Durchschnittlich gefilterte Nachrichten-Paare mit 3 Runden bei Chiffre 2*

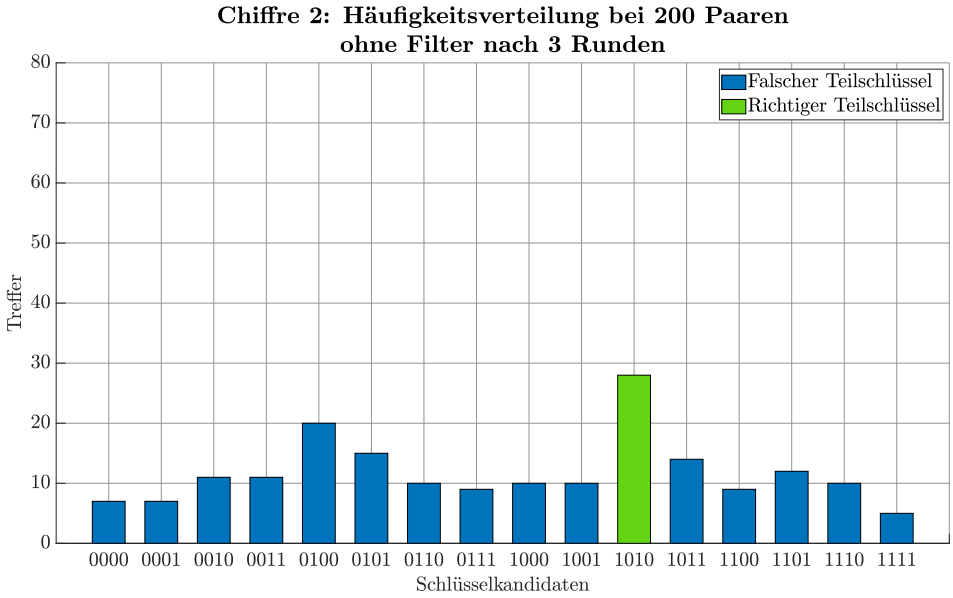


(a) Häufigkeitsverteilung ohne Filterung

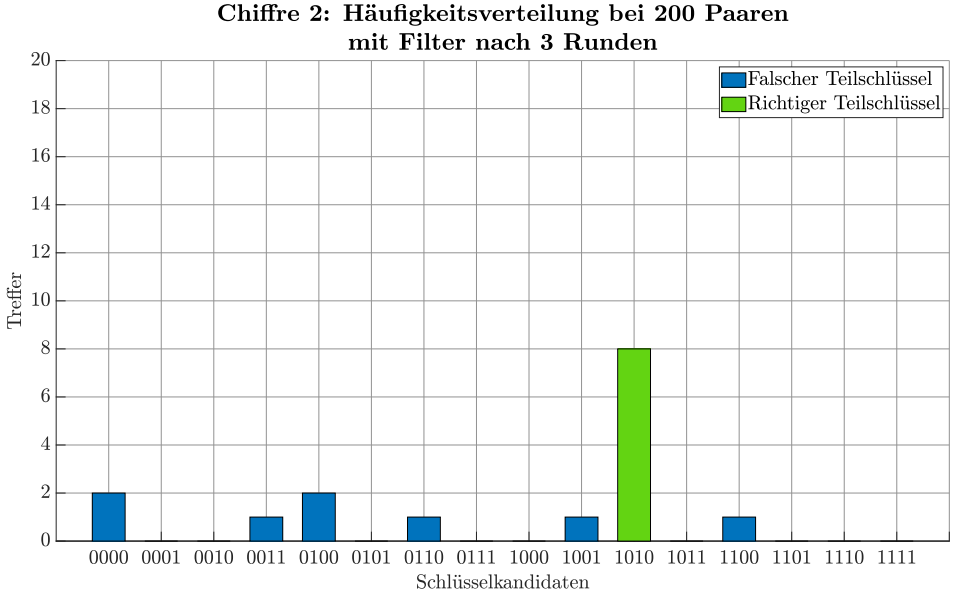


(b) Häufigkeitsverteilung mit Filterung

Abbildung 62: *Häufigkeitsverteilung bei der Schlüsselwiederherstellung mit 100 Nachrichten-Paaren nach Runde 3 bei Chiffre 2*

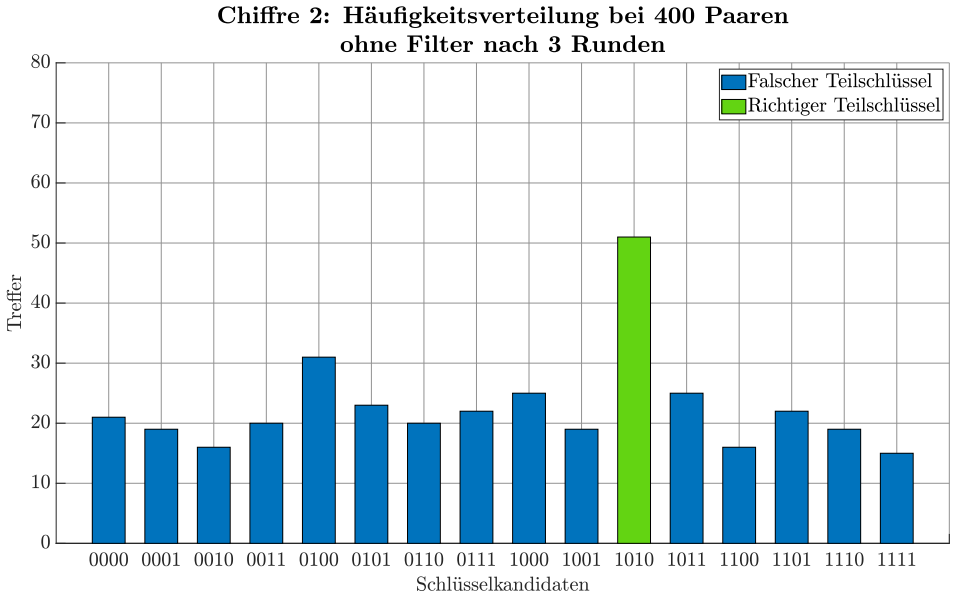


(a) Häufigkeitsverteilung ohne Filterung

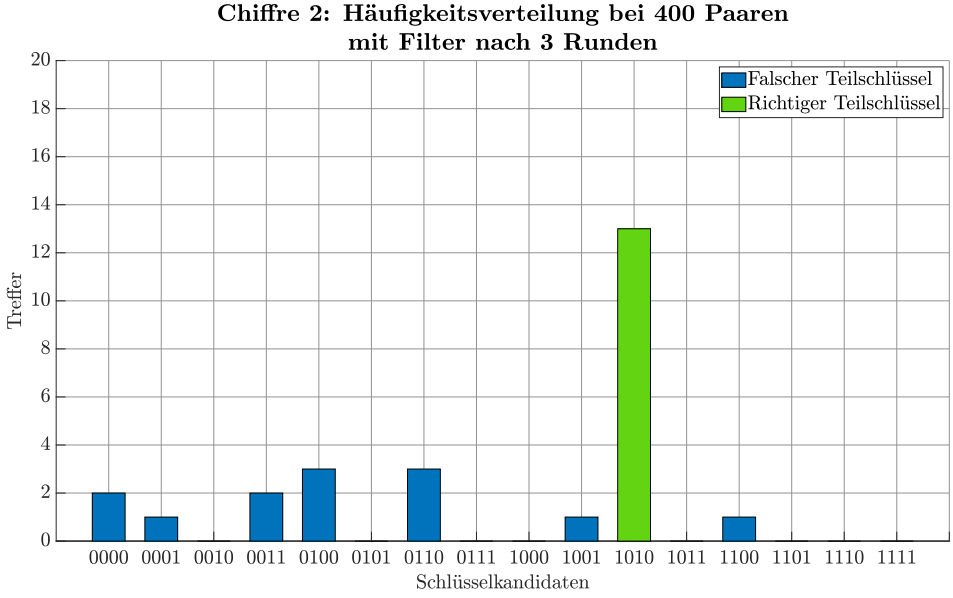


(b) Häufigkeitsverteilung mit Filterung

Abbildung 63: *Häufigkeitsverteilung bei der Schlüsselwiederherstellung mit 200 Nachrichten-Paaren nach Runde 3 bei Chiffre 2*

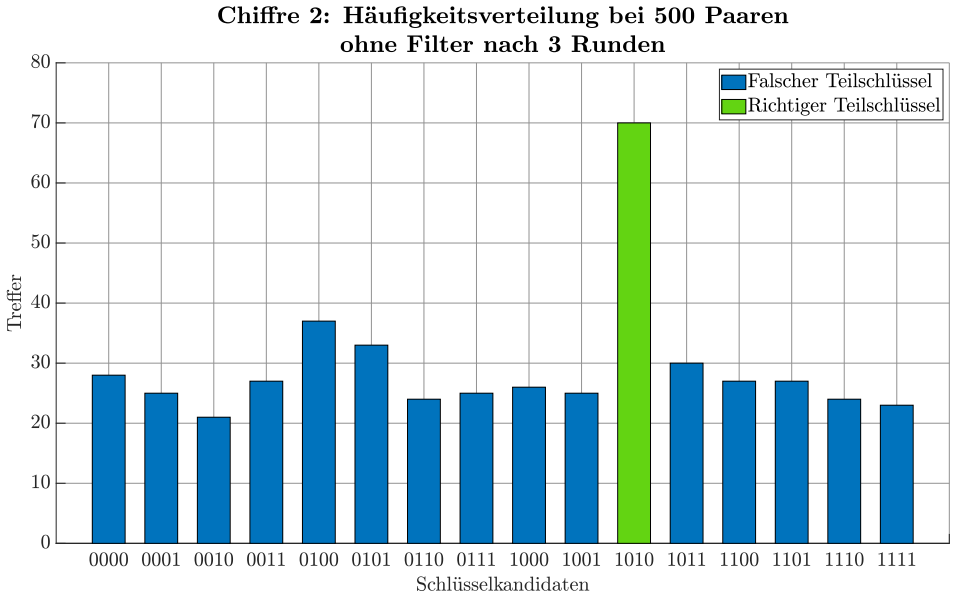


(a) Häufigkeitsverteilung ohne Filterung

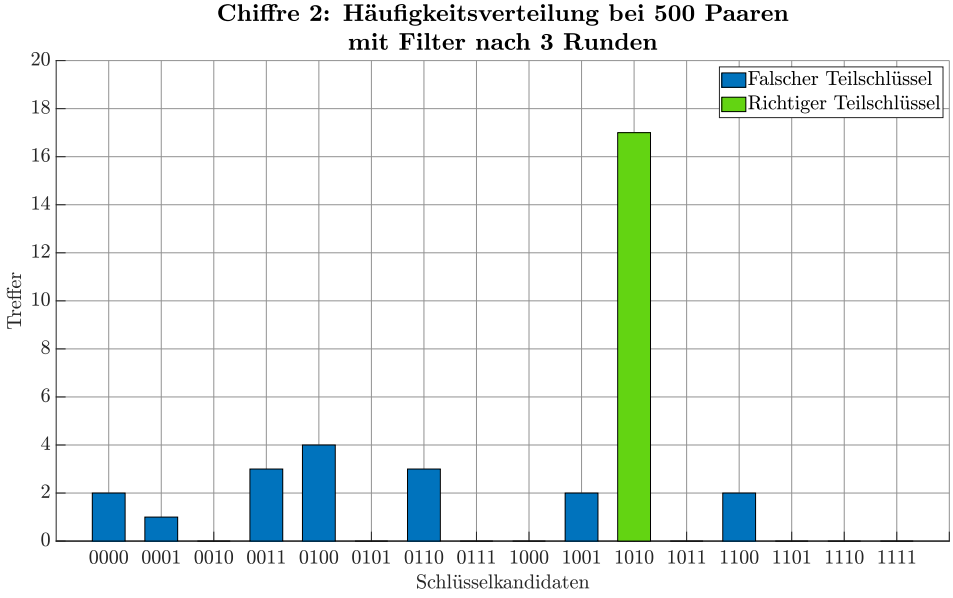


(b) Häufigkeitsverteilung mit Filterung

Abbildung 64: *Häufigkeitsverteilung bei der Schlüsselwiederherstellung mit 400 Nachrichten-Paaren nach Runde 3 bei Chiffre 2*

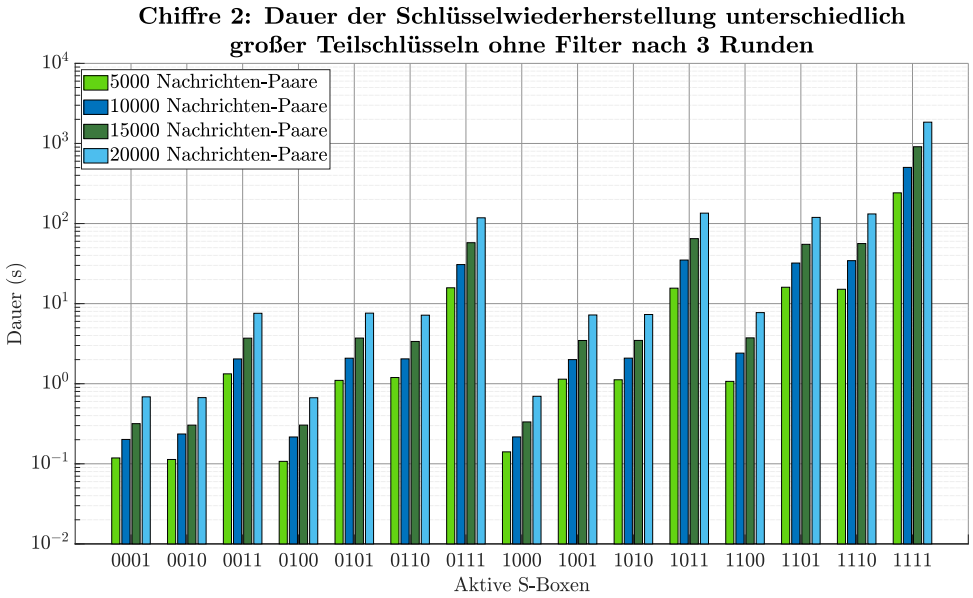


(a) Häufigkeitsverteilung ohne Filterung

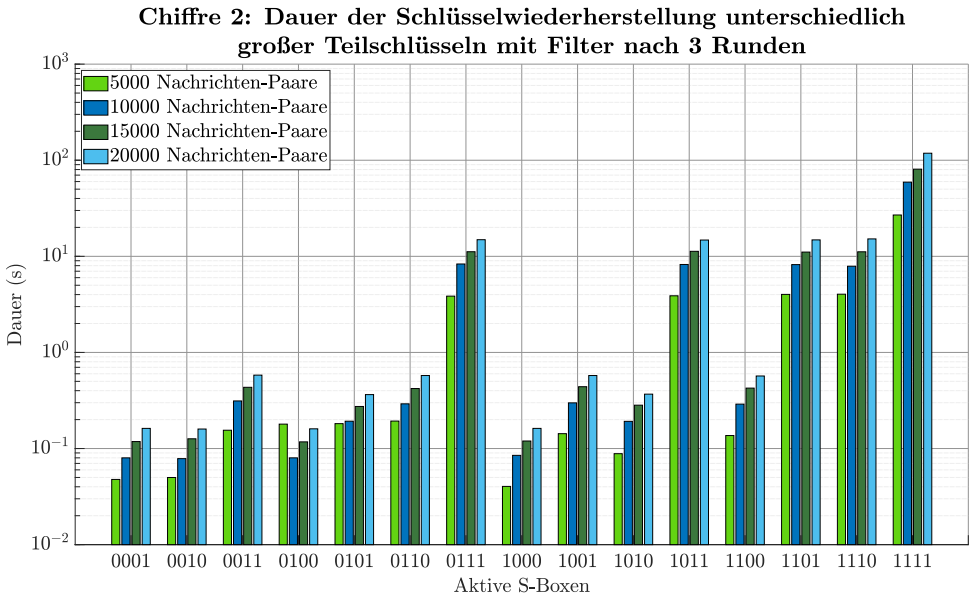


(b) Häufigkeitsverteilung mit Filterung

Abbildung 65: *Häufigkeitsverteilung bei der Schlüsselwiederherstellung mit 500 Nachrichten-Paaren nach Runde 3 bei Chiffre 2*



(a) *Dauer ohne Filterung*



(b) *Dauer mit Filterung*

Abbildung 66: *Dauer der Schlüsselwiederherstellung bei unterschiedlich großen Teilschlüsseln bei Chiffre 2*

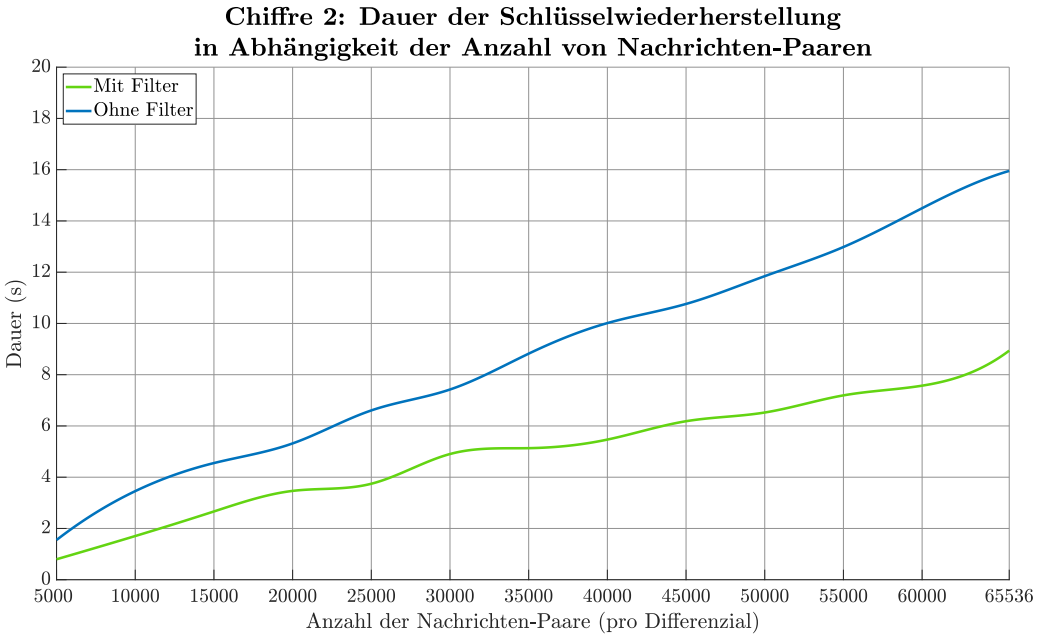
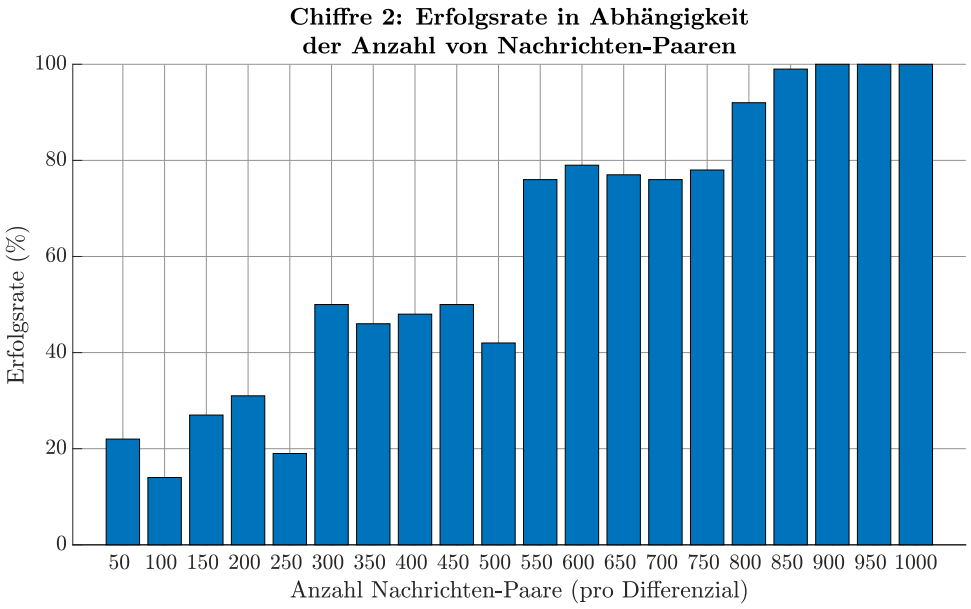
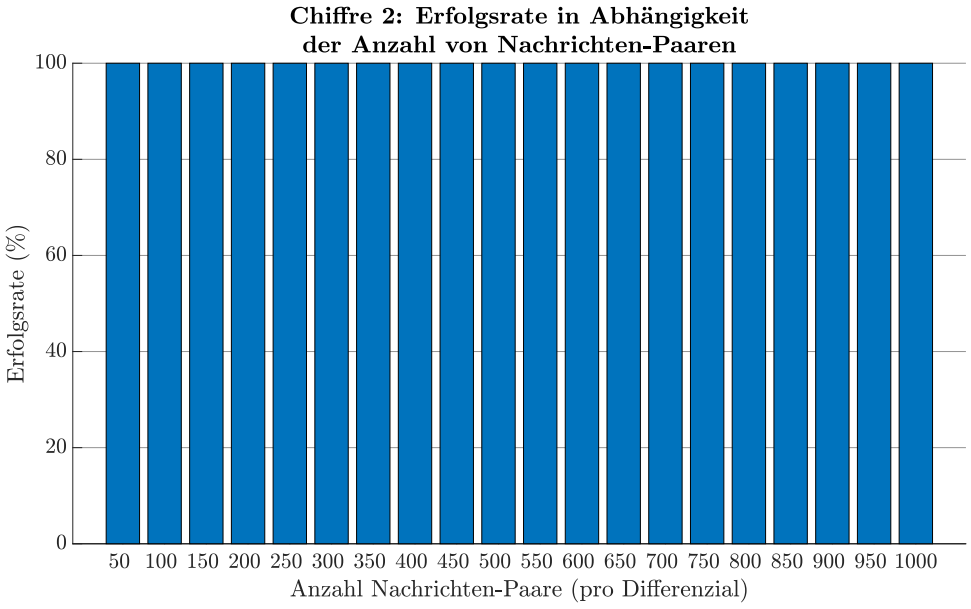


Abbildung 67: *Dauer der vollständigen Schlüsselwiederherstellung mit und ohne Filterung bei Chiffre 2*

A.1.4 Untersuchung der Schlüsselwiederherstellung



(a) *Erfolgsrate mit inkrementierten Nachrichten-Paaren*



(b) *Erfolgsrate mit zufälligen Nachrichten-Paaren*

Abbildung 68: *Erfolgsrate der differenziellen Kryptoanalyse bei Chiffre 2*

A.1.5 Vergleich mit Brute-Force

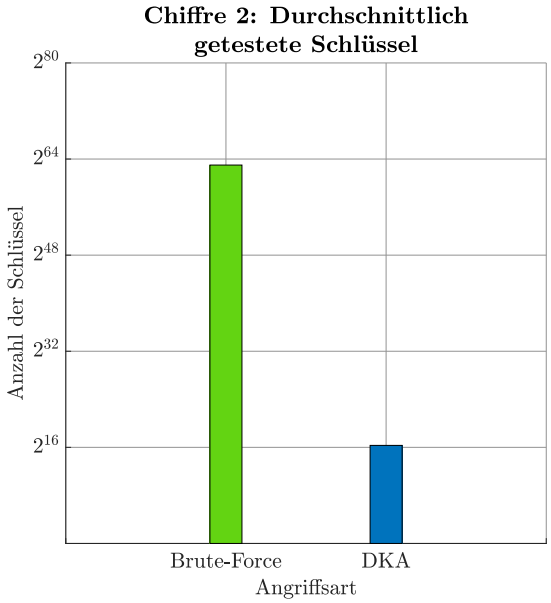


Abbildung 69: *Durchschnittlich mit der DKA durchsuchter Schlüsselraum bei der Schlüsselwiederherstellung bei Chiffre 2*

A.2 Analysen zu Chiffre 3

A.2.1 Untersuchung der Suchstrategien

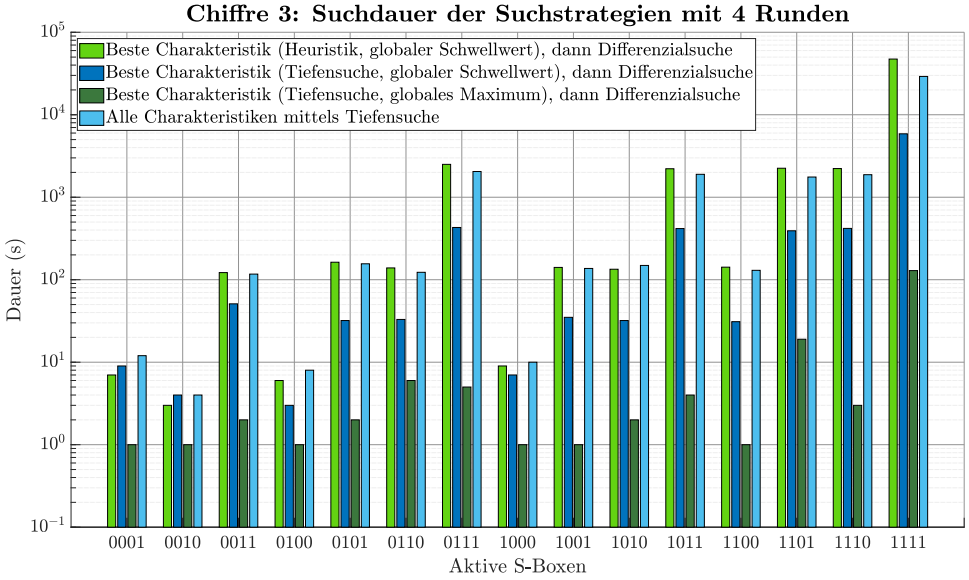


Abbildung 70: *Dauer der Suchstrategien mit 4 Runden bei Chiffre 3*

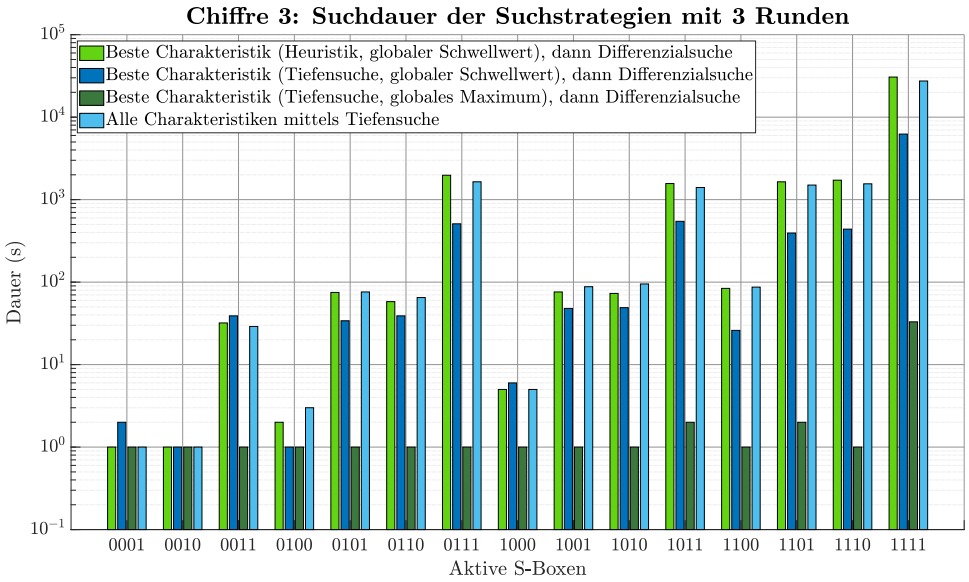


Abbildung 71: *Dauer der Suchstrategien mit 3 Runden bei Chiffre 3*

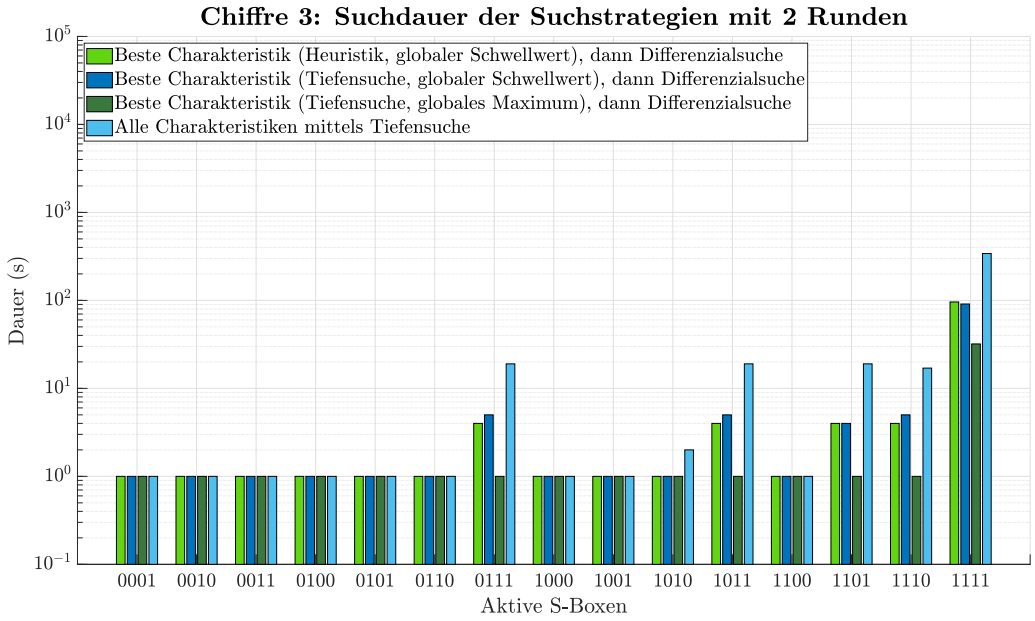
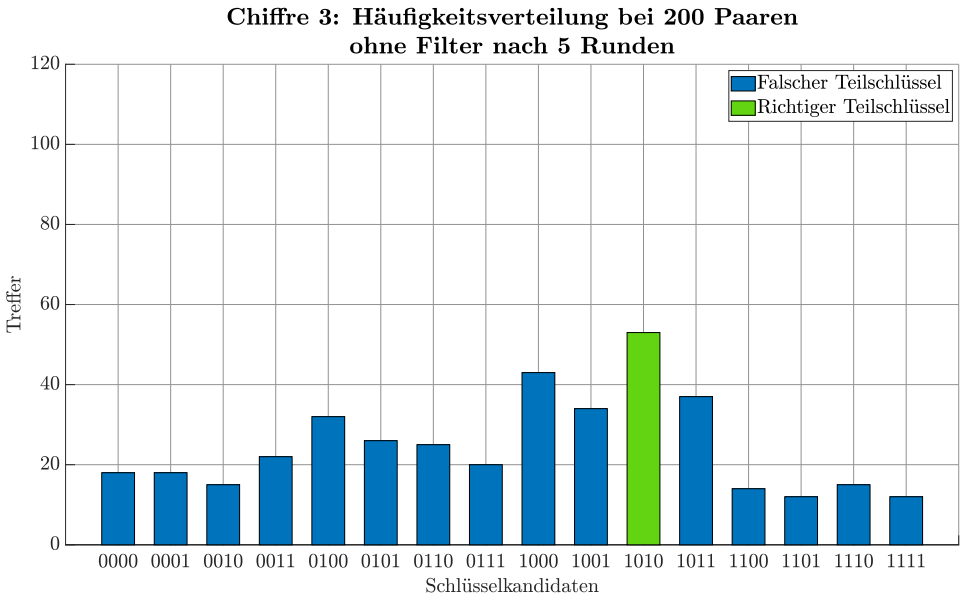
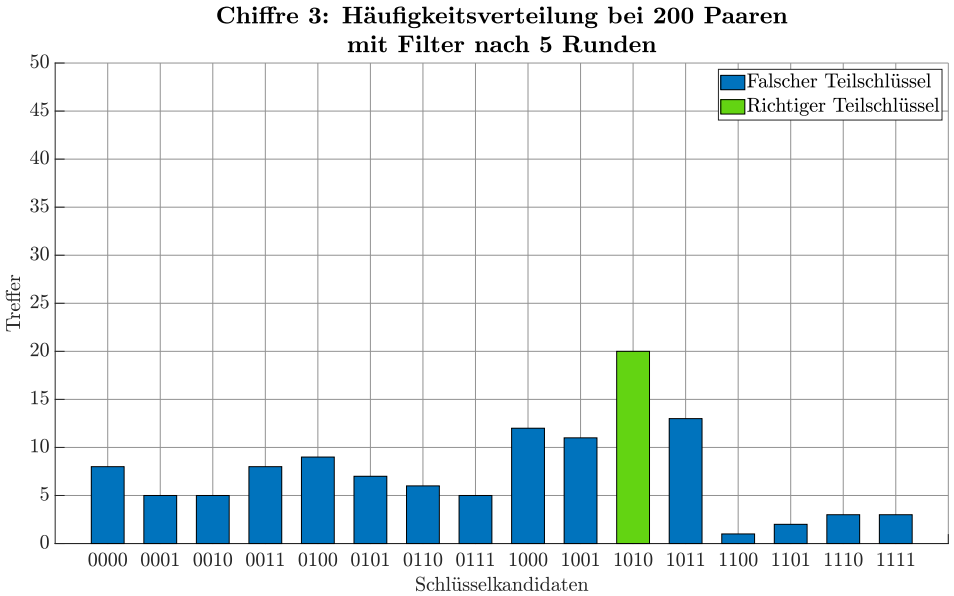


Abbildung 72: *Dauer der Suchstrategien mit 2 Runden bei Chiffre 3*

A.2.2 Untersuchung der Filterung

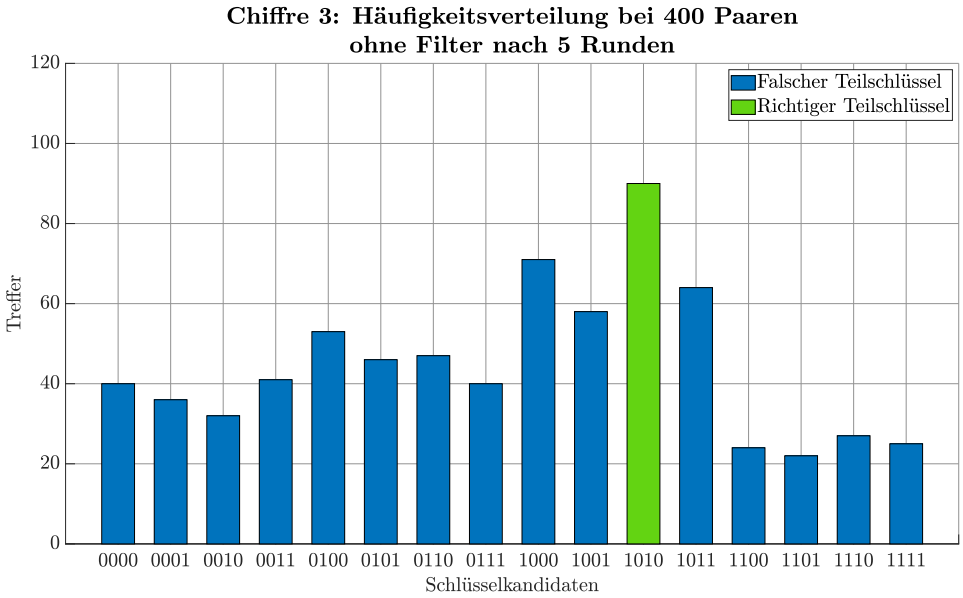


(a) Häufigkeitsverteilung ohne Filterung

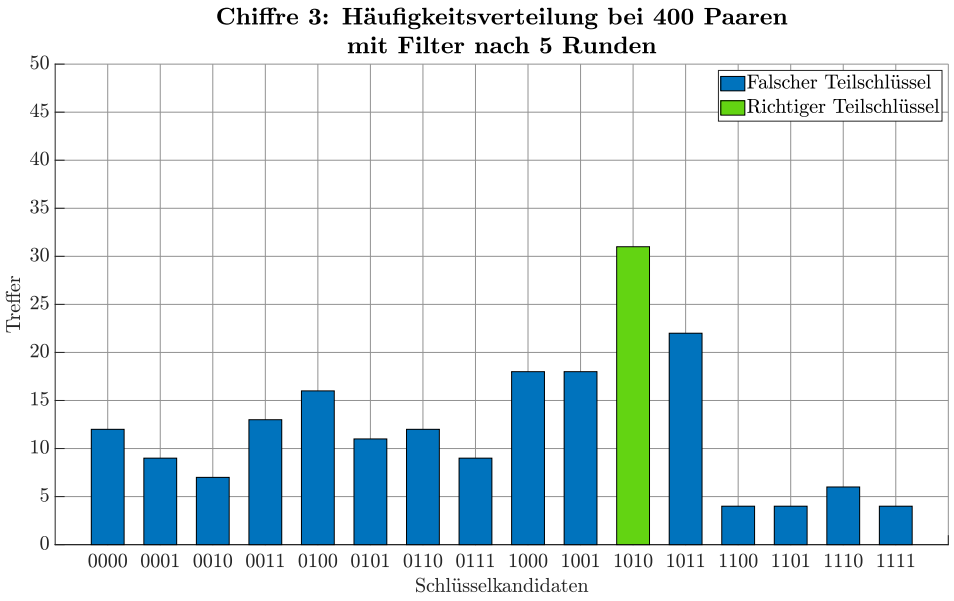


(b) Häufigkeitsverteilung mit Filterung

Abbildung 73: *Häufigkeitsverteilung bei der Schlüsselwiederherstellung mit 200 Nachrichten-Paaren nach Runde 5 bei Chiffre 3*

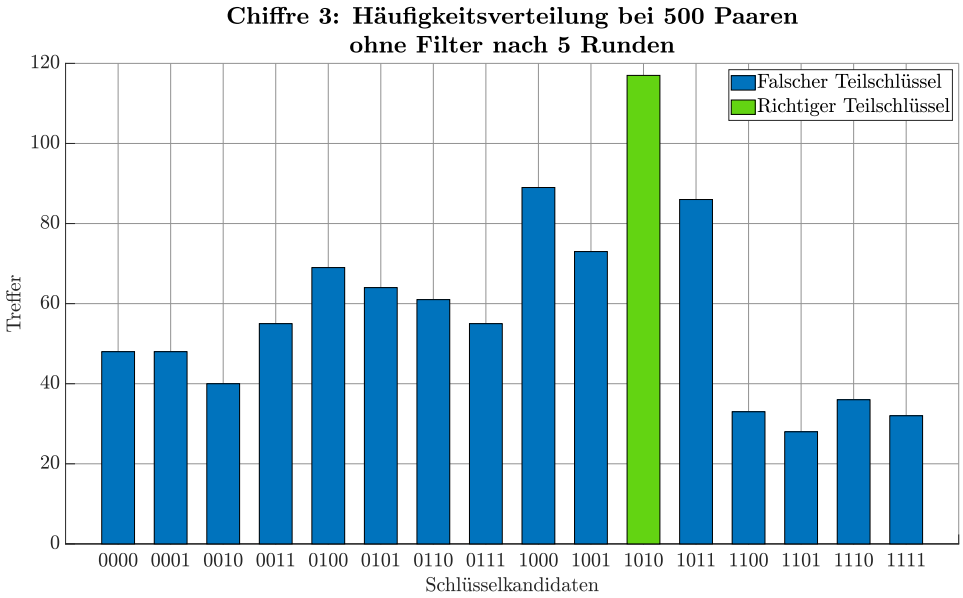


(a) Häufigkeitsverteilung ohne Filterung

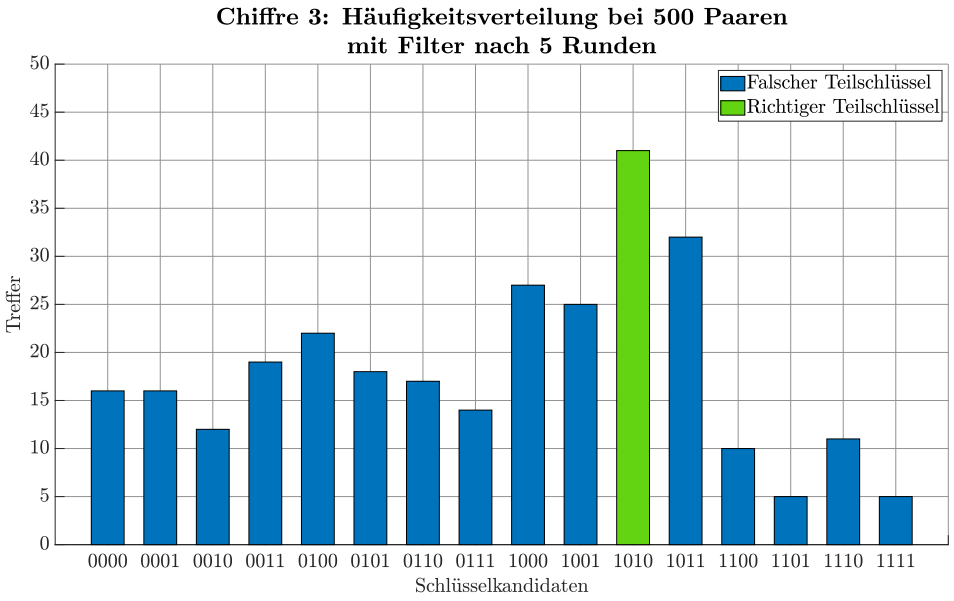


(b) Häufigkeitsverteilung mit Filterung

Abbildung 74: *Häufigkeitsverteilung bei der Schlüsselwiederherstellung mit 400 Nachrichten-Paaren nach Runde 5 bei Chiffre 3*



(a) Häufigkeitsverteilung ohne Filterung



(b) Häufigkeitsverteilung mit Filterung

Abbildung 75: *Häufigkeitsverteilung bei der Schlüsselwiederherstellung mit 500 Nachrichten-Paaren nach Runde 5 bei Chiffre 3*

B SVN

Der Quellcode der entwickelten CT2-Plugins wurde am 04.11.2019 im SVN-Repository eingecheckt und ist unter [33] zu finden.

C DVD

Auf der DVD befindet sich die Masterarbeit in PDF-Form, der Latex-Quellcode der Masterarbeit, der Matlab-Quellcode zur Erstellung der Diagramme, der letzte Stand des Quellcodes der CT2-Developer-Edition und der Quellcode der entwickelten Konsolenanwendung (siehe 5.1).