Bernhard Esslinger

Kryptografie lernen und anwenden mit CrypTool und SageMath

lehmanns media

Kryptografie lernen und anwenden mit CrypTool und SageMath

Bernhard Esslinger

Das CrypTool-Buch:

Kryptografie lernen und anwenden mit CrypTool und SageMath

Kryptografie, Mathematik und mehr mit dem freien E-Learning-Programm CrypTool

Bernhard Esslinger

lehmanns media

Zum Referenzieren per Bib(La)TEX:

```
@book{Esslinger:ctb_2025_de,
  editor    = {Bernhard Esslinger},
  title     = {Das CrypTool-Buch: Kryptografie lernen
               und anwenden mit CrypTool und SageMath},
  publisher = {Lehmanns Media, Berlin},
  url       = {https://www.cryptool.org/de/ctbook/},
  year      = {2025},
}
```

Impressum:

ISBN 978-3-96543-517-9
Verlag: Lehmanns Media, Berlin (https://www.lehmanns.de)

Umschlag: Herbert Voß
Schriftsatz-Software: LuaLaTeX
Automations-Tool zum Erstellen von LaTeX-PDFs: arara
Versionsverwaltungs-Software: Git
Unterstützung bei der Übersetzung: DeepL-Übersetzer (DeepL.com) und ChatGPT 4 (openai.com)
Unterstützung beim Generieren von Tikz-Grafiken und LaTeX-Tabellen: Claude 3.5 Sonnet (claude.ai)

Teil I

Vorbemerkungen

Vorwort von Prof. Paar

Abb. 1: Prof. Dr. Christof Paar
Direktor am Max-Planck-Institut für Sicherheit und Privatsphäre

Das vorliegende Buch zeichnet sich durch eine durchdachte Zusammenstellung hoch relevanter Themen in der Kryptografie aus, die einen einfachen Zugang zu diesem extrem spannenden Fachgebiet ermöglicht. Der Autor, Prof. Bernhard Esslinger, kann auf jahrzehntelange Erfahrung sowohl im akademischen Umfeld als auch in der Industrie zurückblicken, was sich in der geschickten Themenmischung aus Theorie und Praxis widerspiegelt.

In dem Buch ist der seltene Brückenschlag zwischen historischen Chiffren, die für das Verstehen der modernen Kryptografie sehr hilfreich sind, und moderner praxisrelevanter Kryptografie hervorragend gelungen. Im Bereich der asymmetrischen (oder auch: public-key) Kryptografie wird nicht nur das ganze Spektrum der heutzutage eingesetzten Algorithmen eingeführt, sondern es werden auch die nächste Generation von Verfahren behandelt, die sogenannte post-quantum Kryptografie. En passant gelingt es Prof. Esslinger, auch wichtige mathematische Konzepte aus dem Bereich der Zahlentheorie elegant einzuführen.

Durch die gelungene Kombination von Themen richtet sich das Werk an ein breites Publikum, nicht nur an naturwissenschaftlich Vorgebildete. Besonders hervorzuheben ist die Möglichkeit, die vorgestellten Verfahren direkt anhand der in den Fußnoten angegebenen Links zu Open-Source-Programmen wie CrypTool, OpenSSL oder SageMath auszuprobieren. Dies wird durch zahlreiche Screenshots unterstützt, die das Interesse wecken und den Einstieg erleichtern.

Ich freue mich, dass die immer noch recht beschränkte Auswahl an echten Lehrbüchern im Bereich der Kryptografie mit diesem Werk deutlich erweitert und bereichert wird!

Bochum, Oktober 2024 Christof Paar

Kurzinhaltsverzeichnis

Langinhaltsverzeichnis

Überblick über den Inhalt des CrypTool-Buchs

Mit dem Erfolg des Internets wurden die damit verbundenen Technologien verstärkt erforscht, was auch im Bereich Kryptografie viele neue Erkenntnisse schaffte.

Dieses *Buch zu den CrypTool-Programmen* bietet einen Überblick über die klassische und moderne Kryptografie und leitet zum konkreten Ausprobieren an. Dazu werden die CrypTool-Programme benutzt und ebenso Beispielcode, geschrieben für das Computer-Algebra-System **SageMath** (siehe Anhang A.7).

Das erste Kapitel dieses Buchs beschreibt die Prinzipien der symmetrischen und asymmetrischen **Verschlüsselung** und diskutiert Definitionen und Beispiele für deren Widerstandsfähigkeit.

Im zweiten Kapitel wird – aus didaktischen Gründen – eine ausführliche Übersicht über **Papier- und Bleistiftverfahren** gegeben. Außerdem wird ein typisches Beispiel einer Vor-Computer Maschinen-Chiffre erläutert und beschrieben, wie weit fortgeschritten KI-basierte Angriffe auf diese Hagelin-Maschine sind.

Kapitel 3 gibt einen umfassenden Überblick über **Historische Kryptologie**, ein neu etabliertes Forschungsgebiet, das sich mit den praktischen Problemen bei der Entschlüsselung und Einordnung von verschlüsselten historischen Dokumenten beschäftigt.

Kapitel 4 widmet sich dem faszinierenden Thema der **Primzahlen**. Anschließend führt Kapitel 5 anhand vieler Beispiele in die **modulare Arithmetik** und die **elementare Zahlentheorie** ein. Hier bilden die Eigenschaften des **RSA-Verfahrens** einen Schwerpunkt.

Danach erhalten Sie Einblicke in die mathematischen Konzepte und Ideen hinter der **modernen asymmetrischen Kryptografie** (Kapitel 6) inkl. einer neuen geometrischen Veranschaulichung der Vorgänge bei der RSA-Verschlüsselung.

Kapitel 7 gibt einen knappen Überblick zum Stand der Attacken gegen Passworte und moderne **Hash-Algorithmen**, und widmet sich dann kurz der Authentifizierung in der Praxis und den **digitalen Signaturen und Public-Key-Infrastrukturen** – sie sind unverzichtbarer Bestandteil von E-Business-Anwendungen. Hier werden ganz aktuell die Unterschiede der verschiedenen 2FA- und MFA-Verfahren klar gegenüber gestellt und die Erwartung von NIS-2 dazu verdeutlicht.

Kapitel 8 stellt **elliptische Kurven** vor: Sie sind eine Alternative zu RSA und für die Implementierung auf Chipkarten besonders gut geeignet.

Kapitel 9 führt in die **moderne symmetrische Kryptografie** ein. Boolesche Algebra ist Grundlage der meisten modernen, symmetrischen Verschlüsselungsverfahren, da diese auf Bitströmen und Bitblöcken operieren. Prinzipielle Konstruktionsmethoden dieser Verfahren werden beschrieben und in SageMath implementiert. Dieses Kapitel ist im Vergleich relativ mathematisch.

Kapitel 10 stellt **homomorphe Kryptofunktionen** vor. Homomorphe Verschlüsselung ist ein modernes Forschungsgebiet, das im Zuge des Cloud-Computing sehr an Bedeutung gewinnt.

Kapitel 11 gibt eine sehr einfache **Einführung in die Gitterkryptografie**, ein Gebiet, das Quantencomputer-resistente Verfahren ermöglicht – verbunden mit einigen kryptografischen Rätseln.

Kapitel 12 beschreibt **Aktuelle Resultate zum Lösen diskreter Logarithmen und zur Faktorisierung** und gibt einen breiten Überblick über die zur Zeit besten Algorithmen für (a) das Berechnen diskreter Logarithmen in verschiedenen Gruppen, für (b) das Faktorisierungsproblem und für (c) elliptische Kurven. Dieser Überblick wurde zusammengestellt, nachdem ein provozierender Vortrag auf der Black Hat-Konferenz 2013 für Verunsicherung sorgte, weil er die Fortschritte bei endlichen Körpern mit kleiner Charakteristik fälschlicherweise auf Körper extrapolierte, die in der Realität verwendet werden.

Kapitel 13 **Zukünftige Kryptografie** diskutiert Bedrohungen für bestehende kryptografische Verfahren und stellt die Forschungsansätze (Post-Quantum-Kryptografie) für eine langfristige kryptografische Sicherheit vor incl. der ersten vom NIST 2024 verabschiedeten PQC-Standards.

Die einzelnen Hauptkapitel sind von verschiedenen **(Mit-)Autoren** verfasst (siehe Anhang B.3) und in sich abgeschlossen. Die behandelten Inhalte werden begleitet von zahlreichen Beispielen und SageMath-Listings. Am Ende der meisten Kapitel finden Sie Literaturangaben und Web-Links. Die Kapitel wurden reichlich mit *Fußnoten* versehen, in denen auch darauf verwiesen wird, wie man die beschriebenen Funktionen in den verschiedenen CrypTool-Programmen, in SageMath oder OpenSSL aufruft und ausprobiert.

Während die CrypTool-*E-Learning-Programme* eher den praktischen Umgang motivieren, dient das *Buch* auch dazu, dem an Kryptografie Interessierten ein tieferes Verständnis für die implementierten mathematischen Algorithmen zu vermitteln – und das möglichst gut nachvollziehbar.

Den besten Überblick, welche Funktionen in den CrypTool-Programmen insgesamt zur Verfügung stehen, liefert die ff. Webseite: `https://www.cryptool.org/de/functions/`. Siehe auch Anhang A.1 auf Seite 755.

In diesem Buch geben die Anhänge A.2, A.3, A.4 und A.5 einen Überblick über die vier verschiedenen CrypTool-Varianten via:

- der Funktionsliste und den Menüs von CrypTool 1 (CT1)
- der Funktionsliste und den CrypTool-2-Vorlagen (CT2)
- den JCrypTool-Funktionen (JCT)
- den CrypTool-Online-Anwendungen (CTO)

Zwei weitere Anhänge geben fundierte Einführungen in **SageMath** und **OpenSSL** mit vielen Beispielen. SageMath wird dabei in einen breiteren Kontext gestellt (LaTeX, Python, Jupyter).

Die Programme zu diesem Buch oder spezifische Ergänzungen finden Sie auf der CrypTool-Seite: `https://www.cryptool.org/de/ctbook/`. Zu den Ergänzungen gehört bspw. das 90-seitige Dokument „CUDA Tutorial – Cryptanalysis of Classical Ciphers Using Modern GPUs and CUDA“.[1]

Die Autoren dieses Buchs möchten sich an dieser Stelle bedanken bei den Kollegen in der jeweiligen Firma und an den Universitäten Bochum, Darmstadt, Frankfurt, Gießen, Karlsruhe, Lausanne, München, Paris und Siegen.

Wie auch bei dem E-Learning-Programm CrypTool wächst die Qualität des Buchs mit den Anregungen und Verbesserungsvorschlägen von Ihnen als Leser. Wir freuen uns auf Ihre Rückmeldungen.

Geschlechtsneutrale Formulierung: Aus Gründen der einfacheren Lesbarkeit wird auf geschlechtsspezifische Differenzierung verzichtet. Entsprechende Begriffe wie „Entwickler“ oder „Kryptologe“ gelten im Sinne der Gleichbehandlung grundsätzlich für alle Geschlechter.

[1] Es enthält eine praktische Einführung in das Schreiben von CUDA-Programmen unter Linux und Windows.

Einleitung zum CrypTool-Buch

Das CrypTool-Buch versucht, die wichtigsten Themen aus der Kryptologie genau und trotzdem möglichst verständlich darzustellen, und zum Ausprobieren anzuleiten. Dazu wurden die Themen sinnvoll in eigenständig lesbare Kapitel unterteilt, damit Entwickler/Autoren unabhängig voneinander mitarbeiten können. Natürlich gäbe es viel mehr Themen aus der Kryptografie, die man vertiefen könnte – deshalb ist diese Auswahl nicht erschöpfend.

In der anschließenden redaktionellen Arbeit wurden Querverweise ergänzt, Fußnoten hinzugefügt, Index-Einträge vereinheitlicht und sehr viele Korrekturen vorgenommen.

Dieses Buch wurde ab dem Jahr 2000 zusammen mit CrypTool 1 (CT1) ausgeliefert. Im Vergleich zur letzten deutschen Ausgabe in 2018 wurden hier die TeX-Sourcen des Dokuments komplett umstrukturiert, etliche Themen neu hinzugefügt, ergänzt, korrigiert, und alles auf den aktuellen Stand gebracht, z. B.:

- ein detaillierter Vergleich zwischen theoretischem und praktischem Schlüsselraum und ein Überblick über die besten bekannten Angriffe gegen verschiedene Chiffren (neu, Kap. 1.7 und 1.9),
- eine detaillierte Erklärung der C(X)-52, der am weitesten verbreiteten mechanischen (Hagelin) Chiffriermaschine, ihre Bedeutung und die Implementierung ihrer Varianten in CT2 (neu, Kap. 2.5),
- das Forschungsgebiet Historische Kryptologie (neu, Kapitel 3 auf Seite 145),
- generell das Primzahlen-Kapitel und die Liste der größten Primzahlen (Kap. 4.5),
- RSA-2D: eine neue geometrische Veranschaulichung der Vorgänge bei der RSA-Verschlüsselung (neu, Abschnitt 6.5 auf Seite 401),
- Auflistung, in welchen Filmen und Romanen Kryptografie eine wesentliche Rolle spielt (Anhang B.1),
- die Funktionsübersichten zu CrypTool 2 (CT2), zu JCrypTool (JCT) und zu CrypTool-Online (CTO) (siehe Anhang),
- weitere SageMath-Skripte zu Kryptoverfahren, die Einführung in das Computer-Algebra-System SageMath (siehe Anhang A.7) und die Anpassung aller Skripte an die SageMath-Versionen 9.x und 10.x,
- die Einführung in das Kommandozeilen-Tool OpenSSL (neu, Anhang A.8 auf Seite 817),
- die Abschnitte über die Goldbach-Vermutung (siehe 4.10.5), über Primzahl-Zwillinge (siehe 4.10.6), und über Primzahl-Lücken (siehe 4.10.8),
- der Abschnitt über gemeinsame Primzahlen in real verwendeten RSA-Moduli (siehe 5.12.5.4),
- die Grundlagen der modernen symmetrischen Verschlüsselung (stark überarbeitet, Kapitel 9 auf Seite 487),
- die Einführung in die Gitterkryptografie (neu, Kapitel 11 auf Seite 629),
- das Kapitel über Diskrete Logarithmen und Faktorisierung (Kapitel 12 auf Seite 715) und
- das Kapitel über die Zukünftige Kryptografie (Kapitel 13 auf Seite 741).

Dank

An dieser Stelle möchte ich explizit folgenden Personen danken, die bisher in ganz besonderer Weise zum CrypTool-Projekt beigetragen haben. Ohne ihre besonderen Fähigkeiten und ihr großes Engagement wäre CrypTool nicht, was es heute ist:

- Hr. Henrik Koy
- Hr. Jörg-Cornelius Schneider
- Hr. Florian Marchal
- Dr. Peer Wichmann
- Hr. Dominik Schadow
- Prof. Nils Kopal
- Mitarbeiter in den Teams von Prof. Johannes Buchmann, Prof. Claudia Eckert, Prof. Alexander May, Prof. Torben Weis und insbesondere Prof. Arno Wacker.

Zu erwähnen sind die Studenten, die durch ihre inzwischen weit über 200 Diplom-, Bachelor- und Masterarbeiten beitrugen. Auch allen hier nicht namentlich Genannten ganz herzlichen Dank für das (meist in der Freizeit) geleistete Engagement.

Danke auch an die Leser, die uns Feedback sandten, und insbesondere an Olaf Ostwald, Helmut Witten, Prof. Ralph-Hardo Schulz und Prof. Gottfried Barthel, die Teile dieser neuen Version konstruktiv gegengelesen haben. Und an Herbert Voß, der uns immer dann half, wenn es in LaTeX schwierig wurde.

Ein besonderes Dankeschön geht an Dr. Doris Behrendt, die die aufwändige Aufgabe übernahm, zwei Bücher mit je 500+ Seiten nach KOMA-Script zu bringen und die TeX-Sourcen aufzuräumen und zu modernisieren, die von unterschiedlichen Autoren über die Jahre erstellt wurden – und die dabei auch noch viele Inhalte kritisch las.

Und schließlich Dank an die Mitarbeiter des Verlags und Herbert Voß, die uns vor allem am Ende halfen, fokussiert zu bleiben.

Die Bereitschaft von Christof Paar, das Vorwort zu schreiben, ist für mich eine große Ehre, und ich möchte ihm an dieser Stelle nochmals danken.

Ich hoffe, dass viele Leser mit diesem Buch mehr Interesse an und Verständnis für dieses moderne und zugleich uralte Thema finden.

Bernhard Esslinger

Heilbronn/Siegen, Oktober 2024

Zusammenspiel von Buch und Programmen

Das CrypTool-Buch

Dieses Buch kann in Papierform und als eBook bei Lehmanns erworben werden. Später wird es auch als PDF auf der Webseite des CT-Portals kostenlos zum Download zur Verfügung stehen:
`https://www.cryptool.org/de/ctbook`

Die Kapitel dieses Buchs sind weitgehend in sich abgeschlossen und können auch unabhängig von den CrypTool-Programmen gelesen und verstanden werden.

Für das Verständnis der meisten Kapitel reicht Abiturwissen aus. Die Kapitel 6 (Moderne asymmetrische Kryptografie), 8 (Elliptische Kurven), 9 (Moderne symmetrische Kryptografie), 10 (Homomorphe Chiffren) und 12 (Resultate zum Lösen diskreter Logarithmen und zur Faktorisierung) erfordern tiefere mathematische Kenntnisse.

Die Autoren haben sich bemüht, Kryptografie für eine möglichst breite Leserschaft darzustellen – ohne mathematisch unkorrekt zu werden, aber mit vielen Hinweisen, Dinge praktisch auszuprobieren. Wir glauben, dass man auf diese Wiese die Awareness für die IT-Sicherheit und für den Einsatz standardisierter, moderner Kryptografie am besten fördern kann.

Die Programme CrypTool 1, CrypTool 2 und JCrypTool

CrypTool 1 (CT1) ist ein Lernprogramm für Windows, mit dem Sie unter einer einheitlichen Oberfläche kryptografische Verfahren anwenden und analysieren können. Die umfangreiche Onlinehilfe in CT1 enthält nicht nur Anleitungen zur Bedienung des Programms, sondern auch Informationen zu den Verfahren selbst (aber weniger ausführlich und anders strukturiert als im CT-Buch).

CrypTool 1 und die Nachfolgeversionen CrypTool 2 (CT2) und JCrypTool (JCT) werden weltweit in Schule, Lehre, Aus- und Fortbildung eingesetzt. CT2 ist ebenfalls für Windows und hat zur Kryptoanalyse inzwischen einen deutlich größeren Funktionsumfang als CT1. JCT läuft unter Windows, Linux und macOS, und enthält inzwischen ebenfalls Vieles, was nicht in CT1 enthalten ist.

Die Setups/Executables dieser eigenständigen Desktop-Programme werden allein von der CT-Webseite mehr als 10 000 Mal pro Monat heruntergeladen.

Die Programme auf CrypTool-Online (CTO)

Die Webseite CrypTool-Online (`http://www.cryptool-online.org`), auf der man im Browser vom PC, Tablet oder Smartphone aus kryptografische Verfahren ausprobieren und anwenden kann, gehört ebenfalls zum CT-Projekt. Sie wird momentan stark forciert.

Der Umfang von CTO ist noch nicht so groß wie der der Standalone-Programme CT1, CT2 und JCT. Da CTO mehr und mehr als Erstkontakt genutzt wird, fließt ein hoher Entwicklungsaufwand in CTO. Wir haben Backbone und Frontend mit moderner Webtechnologie neu designt, um ein schnelles, konsistentes und responsives Look&Feel anzubieten. CTO enthält auch mit WebAssembly gebaute Plugins wie eine Python-Entwicklungsumgebung, Msieve oder OpenSSL. Die Nutzung von WebAssembly (wasm) führt

dazu, dass diese Funktionalität im Browser fast so schnell läuft wie als native Anwendung. Eine weitere moderne Technologie, die angeboten wird, sind Modelle zur Kryptoanalyse, die mit Algorithmen aus dem maschinellen Lernen (deep learning, neuronale Netze, KI) trainiert wurden.

Neben den klassischen Chiffren werden in CTO die folgenden Plugins am meisten aufgerufen: „RSA Schritt-für-Schritt", „RSA visuell und mehr", „AES Schritt-für-Schritt", „AES-Animation" und der „Passwort-Messer".

MTW (früher MTC3)

MTW ist die Abkürzung für **M**ystery**Tw**ister (`https://www.mysterytwister.org`), ein internationaler Kryptografie-Wettbewerb (cipher contest by CrypTool), der vom Lehrstuhl von Prof. May in Bochum und vom CT-Projekt getragen wird. Auf der MysteryTwister-Seite findet man kryptografische Rätsel in vier verschiedenen Kategorien, eine High-Score-Liste und ein moderiertes Forum.

Stand Juli 2024 sind über 13 000 Teilnehmer dabei, und es gibt über 360 Aufgaben, von denen 307 von zumindest einem Teilnehmer gelöst wurden. Seit dem Update auf das neue MTW im April 2024 ist die Seite benutzerfreundlicher, kann komplett auf dem Smartphone bedient werden und nutzt Discord für die Forums-Diskussionen.

Das Computer-Algebra-Programm SageMath

SageMath ist ein umfangreiches Open-Source Computer-Algebra-System (CAS), mit dem sich die in diesem Buch erläuterten mathematischen Verfahren relativ leicht Schritt-für-Schritt programmieren lassen. Eine Besonderheit dieses CAS ist, dass als Skript-Sprache Python (Version 3 seit SageMath 9) benutzt wird. Dadurch stehen einem in Sage-Skripten nicht nur die Mathematik-Befehle aus SageMath zur Verfügung, sondern auch alle Funktionen der Sprache Python. SageMath wird mehr und mehr zum Standard-CAS an Hochschulen. SageMath 8 brachte auch eine Version für Windows, die in einer Bash-Shell lief – seit SageMath 10 läuft es direkt im Windows Subsystem für Linux (WSL).

Die Schülerkrypto-Kurse und learn.crypto

Die Schülerkrypto-Initiative bietet Ein- und Zwei-Tages-Kurse in Kryptologie für Schüler und Lehrer, um zu zeigen, wie attraktiv MINT-Fächer wie Mathematik und Informatik (und insbesondere Kryptologie) sind. Die Kursidee ist eine virtuelle Geheimagenten-Ausbildung. Diese Kurse finden seit mehreren Jahren regelmäßig in Deutschland in unterschiedlichen Städten statt (abgesehen von Corona-Unterbrechungen). Alle Kursunterlagen sind frei erhältlich auf: `https://www.cryptool.org/de/scr/downloads/`. Im Kurs wird freie Lern-Software eingesetzt (meist wird CT2 genutzt). Wir würden uns auch freuen, wenn jemand die Kursunterlagen auf Englisch übersetzt und einen entsprechenden Kurs in Englisch anbieten würde.

Es gibt konkrete Pläne, vollständiges Lehrmaterial genau passend zu den Lehrplänen der Bundesländer zu erstellen. Begonnen wurde mit dem Informatik-Lehrplan für die 11. Klasse an den Gymnasien in Bayern. Die Lerneinheiten zur Kryptologie werden unter `https://learn.cryptool.org/` frei verfügbar sein.

Teil II

Hauptteil

1 Verschlüsselungen und Angriffe dagegen

Jahrhundertelang wurden Klartext-Nachrichten vom Militär, von Diplomaten und von Alchemisten verschlüsselt – viel seltener von Geschäfts- und Privatleuten. Das Ziel war, die Privatheit zwischen Sender und Empfänger zu schützen. Seit den 1970er-Jahren kamen weitere Ziele hinzu, um im modernen Datenverkehr Integrität, Authentizität und Nichtabstreitbarkeit zu erreichen, aber auch um auf verschlüsselten Daten in

der Cloud zu rechnen oder Quantencomputer-Resistenz zu erreichen.

Die Wissenschaft, die sich mit Verschlüsselung beschäftigt, wird **Kryptologie** genannt – aufgeteilt in die Zweige Kryptografie (Entwerfen von sicheren Verfahren) und Kryptoanalyse (Brechen von Verfahren). Real hängen diese beiden Zweige eng zusammen, und die Begriffe Kryptografie und Kryptologie werden oft synonym (austauschbar/gleichbedeutend) verwendet. Deshalb wird Kryptologie heute eher unterteilt in Fachgebiete wie symmetrische oder Public-Key-Kryptografie, Hardware- und Embedded-Systems-Kryptografie, theoretische Kryptologie und Real-World-Krypto. [1] Angelehnt an das englische „cryptanalysis“ wird auch manchmal das Wort „Kryptanalyse“ statt Kryptoanalyse benutzt.

Die Bedeutung der Kryptologie nimmt weiterhin zu, da unsere Gesellschaft immer stärker von Informationstechnologie abhängt. Obwohl Kryptologie und Informationssicherheit interdisziplinäre Forschungsgebiete sind, spielt die mathematische Disziplin inzwischen die größte Rolle in der Kryptologie. Und schließlich kann das Lernen über Kryptologie Spaß machen und unterhaltsam sein.

Das Besondere an diesem Buch ist, dass Sie die Verfahren immer gleich ausprobieren können – durch die jeweils direkt in Fußnoten angegebenen Links auf die Programme vom CrypTool-Projekt, von OpenSSL oder von SageMath. Alle diese Programme sind Open-Source.

In diesem Buch werden die Grundlagen ausführlich behandelt, danach werden aus dem sehr umfangreichen Gebiet der Kryptologie bestimmte (aktuelle) Themen ausgewählt (wie RSA, ECC oder Gitter). Dadurch ist dieses Buch für ein breites Publikum zugänglich, nicht nur für Interessierte der naturwissenschaftlichen Fächer.

Kapitel 1 soll einen eher beschreibenden Einstieg bieten und die Grundlagen (fast) ohne Verwendung von Mathematik vermitteln. Dazu werden als Beispiele moderne Verfahren (RSA, AES) verwendet. Danach vertiefen wir bspw. die Eigenschaft, wie viele mögliche Schlüssel (Schlüsselraum) unterschiedliche Verfahren haben (Abschnitt 1.7 auf Seite 31) und was die besten Angriffe gegen bekannte Verfahren sind (Abschnitt 1.9 auf Seite 47). Abschnitt 1.8 auf Seite 38 erklärt Angriffsarten und Sicherheitsdefinitionen einschließlich des modernen Konzepts der Ununterscheidbarkeit und ordnet diese ein. Empfohlene Bücher werden in Abschnitt 1.11 auf Seite 54 vorgestellt. Im Anhang finden Sie Screenshots, wie man AES in verschiedenen Programmen benutzt (Abschnitt 1.12 auf Seite 55). Klassische Verfahren werden später in den Kapiteln 2 und 3 vorgestellt.

Sinn der **Verschlüsselung** ist es, Daten (Klartext-Nachrichten) so zu verändern, dass nur ein autorisierter Empfänger in der Lage ist, den Klartext zu rekonstruieren. Das hat den Vorteil, dass verschlüsselte Daten offen übertragen werden können und trotzdem keine Gefahr besteht, dass eine Angreiferin die Daten unberechtigterweise lesen kann. Der autorisierte Empfänger ist im Besitz einer geheimen Information, des sogenannten Schlüssels, die es ihm erlaubt, die Daten zu entschlüsseln, während sie jedem anderen verborgen bleiben. Eine Angreiferin hat noch mehr Möglichkeiten als nur das Brechen von Geheimtexten: Sie kann auch die Verbindung stören (z. B. durch einen Denial-of-Service-Angriff) oder Metadaten (wer wann mit wem kommuniziert) abgreifen.

Als „Klartext“ werden die Daten bezeichnet, die als Input vom Verschlüsselungsverfahren verarbeitet werden. Diese Daten können Texte sein, aber auch binäre Daten wie ein Bild oder eine ausführbare Datei. Das Verschlüsselungsverfahren heißt „Chiffre“. Der Output heißt „Geheimtext“ oder „Chiffrat“ (aber nicht „Chiffretext“, wie manchmal von automatischen Übersetzern erzeugt). Bei modernen Verfahren besteht der Geheimtext immer aus binären Daten. Abb. 1.1 auf der nächsten Seite zeigt diesen Prozess grafisch.

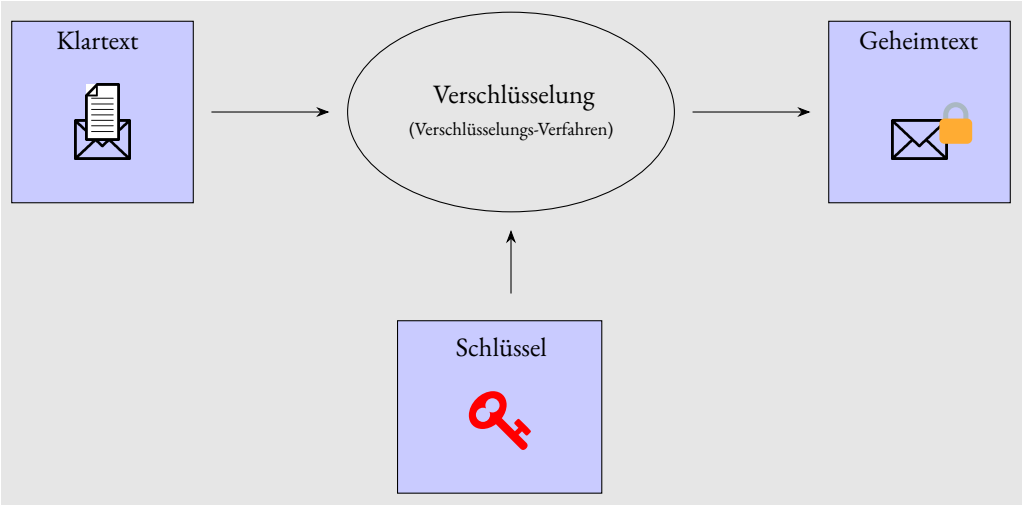


Abb. 1.1: Übliche Bezeichnungen bei der Verwendung von Verschlüsselungsverfahren

1.1 Bedeutung der Kryptologie

Durch die Nutzung des Internets und der drahtlosen Kommunikation werden Verschlüsselungstechnologien (meist transparent) von jedermann genutzt. Kryptografische Algorithmen sichern Geldautomaten und die Privatsphäre von Messengern, ermöglichen Anonymität für Wähler, helfen aber auch Kriminellen. Kryptografie hat einen doppelten Verwendungszweck – wie viele Innovationen des Menschen.

Verschlüsselungstechnologien sind jedoch nicht erst heute im Gebrauch, sondern werden schon seit Jahrhunderten von Regierungen, Militärs und Diplomaten eingesetzt. Welche Seite diese Technologie besser beherrschte, konnte mithilfe der Geheimdienste großen Einfluss auf die **Politik** und den Kriegsverlauf nehmen. Auf die Historie geht dieses Buch nur zweimal ein: wenn die früheren Verschlüsselungsverfahren in Kapitel 2 aus didaktischen Gründen eingeführt werden, und in Kapitel 3, wo die konkrete Anwendung früherer Verfahren vorgestellt wird. Einen Eindruck, welch entscheidende Bedeutung Kryptologie für die Mächtigen hatte und hat, kann man anhand der beiden folgenden Beispiele bekommen: dem Lehrfilm *Krieg der Buchstaben*[1] und der Debatte um die sogenannten Krypto-Wars[2].

Im Folgenden definieren wir, was ein Kryptosystem ausmacht und erläutern die Unterschiede zwischen symmetrischen (siehe Abschnitt 1.3) und asymmetrischen (siehe Abschnitt 1.4) Verfahren zur Verschlüsselung.

[1] Der BBC-Dokumentarfilm (dt. ca. 45 Min.) schildert vor dem Hintergrund der Weltpolitik von 1900-1945 die Entwicklung der Kryptologie und ihre Bedeutung für den Kriegsverlauf im ersten (Zimmermann-Depesche) und zweiten Weltkrieg. Ausführlich wird – aus anglo-amerikanischer Sicht – darauf eingegangen, wie bedeutsam die Kryptoanalyse (auf dem Atlantik gegen die Enigma und auf dem Pazifik gegen Purple) für den Verlauf des zweiten Weltkriegs war.
Siehe https://informatik.schule.de/krypto/Krieg_der_Buchstaben.pdf [2]

[2] Siehe https://de.wikipedia.org/wiki/Crypto_Wars.

1.2 Was ist ein Kryptosystem?

Beginnen wir mit einer mathematischen Notation, um zu definieren, was ein Kryptosystem (manchmal auch Verschlüsselungssystem genannt) ist. Diese Definition stammt aus [3, S. 15] und fasst die Hauptkomponenten sehr gut zusammen.

Definition 1.2.1. *Ein **Kryptosystem** ist ein Fünfertupel*[3] (P, C, K, E, D)*, bei dem die folgenden Bedingungen erfüllt sind:*

1. P *ist eine endliche Menge möglicher Klartexte* m*.*
2. C *ist eine endliche Menge möglicher Geheimtexte* c*.*
3. K*, der Schlüsselraum, ist eine endliche Menge möglicher Schlüssel.*
4. *Für jedes* $k \in K$ *gibt es eine Verschlüsselungsregel* $e_k \in E$ *und eine entsprechende Entschlüsselungsregel* $d_k \in D$*. Jede* $e_k : P \to C$ *und* $d_k : C \to P$ *sind Funktionen derart, dass* $d_k(e_k(m)) = m$ *für jedes Klartextelement* $m \in P$*.*

Die wichtigste Eigenschaft ist Eigenschaft 4. Sie besagt, dass, wenn ein Klartext x mit e_k verschlüsselt wird und der resultierende Geheimtext anschließend mit d_k entschlüsselt wird, der ursprüngliche Klartext x entsteht. Die Funktion kann also eindeutig invertiert werden.

Diese Definition gilt für klassische und moderne Chiffren und sowohl für symmetrische als auch für asymmetrische Chiffren. Klassische Chiffren arbeiten in der Regel mit Buchstaben als Elemente des Klartextes; moderne Chiffren verwenden Bits oder Zahlen als Elemente des Klartextes.

Ein wichtiger Punkt in jedem Kryptosystem ist der Zufall, der bei der Erzeugung der Schlüssel und manchmal auch im Verschlüsselungssystem selbst verwendet wird. Da Kryptosysteme Zufall benötigen, um sicher zu sein, benötigen sie eine Komponente, aus der sie den Zufall beziehen.

Bei der Entwicklung und Einführung eines Kryptosystems muss seine Sicherheit berücksichtigt werden. Traditionell war die Bedrohung, mit der sich die Kryptografie befasste, die eines Angreifers, der den verschlüsselten Text abfängt und versucht, ihn zu entschlüsseln. Es wird davon ausgegangen, dass der Angreifer nicht im Besitz des Schlüssels ist. Die Techniken, die der Angreifer anwendet, um zu versuchen, das Kryptosystem zu „brechen“ (manche sagen „knacken“), werden als Kryptoanalyse bezeichnet. Ein offensichtlicher erster Schritt bei der Entwicklung eines sicheren Kryptosystems besteht also darin, dass die Anzahl möglicher Schlüssel sehr groß ist, so viele, dass der Angreifer nicht in der Lage ist, sie alle in einer angemessenen Zeitspanne zu testen. Die Sicherheitsaspekte werden in den Abschnitten 1.7 und 1.8 vertieft.

1.3 Symmetrische Verschlüsselung

Bei der *symmetrischen* Verschlüsselung[4] müssen Sender und Empfänger über einen gemeinsamen (geheimen) Schlüssel verfügen, den sie vor Beginn der eigentlichen Kommunikation (über einen anderen Kanal, „out-

[3] Damit ist gemeint, dass fünf Komponenten gebraucht werden und die Reihenfolge ihrer Definition wesentlich ist.

[4] Neben vielen klassischen Verfahren finden sich etliche moderne, symmetrische Verschlüsselungsverfahren in den verschiedenen CrypTool-Programmen:
- Über `CTO (www.cryptool-online.org)` kann man sich im Browser AES in 2 Plugins ansehen: als „AES-Animation“ und per „AES (Schritt-für-Schritt)“. Außerdem bietet CTO im OpenSSL-Plugin alle symmetrischen Verfahren, die in OpenSSL 3.0 enthalten sind.
- Über `CT1 Ver-/Entschlüsseln ▷ Symmetrisch (modern)` können Sie folgende moderne symmetrische Verschlüsselungsverfahren ausführen: IDEA, RC2, RC4, DES (ECB), DES (CBC), Triple-DES (ECB), Triple-DES (CBC), MARS (AES-Kandidat), RC6 (AES-Kandidat), Serpent (AES-Kandidat), Twofish (AES-Kandidat), Rijndael (offizielles AES-Verfahren).
- Unter `CT2 Startcenter ▷ Vorlagen ▷ Kryptografie ▷ Modern ▷ Symmetrisch`: AES, DES, PRESENT, RC2, RC4, SDES, TEA, Triple-DES,

of-the-band“) ausgetauscht haben. Der Sender benutzt diesen Schlüssel, um die Nachricht zu verschlüsseln und der Empfänger, um diese zu entschlüsseln. Dies veranschaulicht Abb. 1.2.

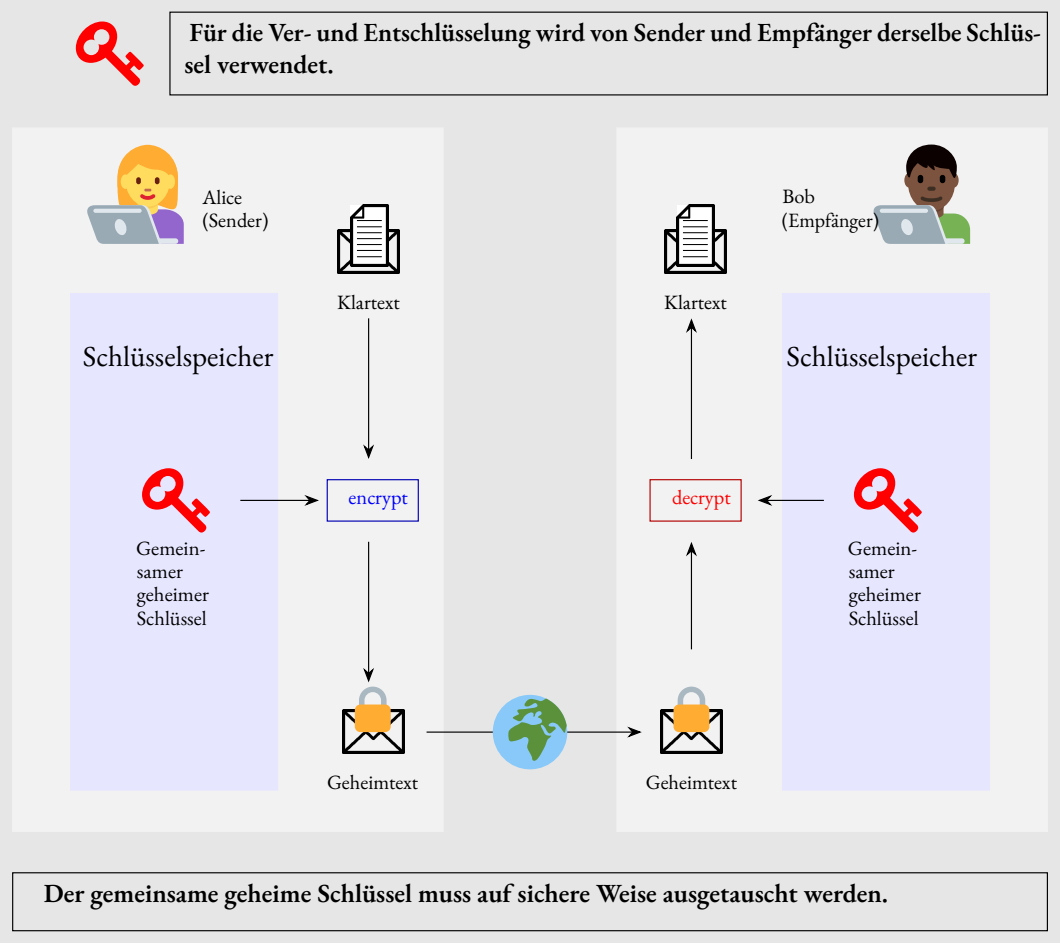


Abb. 1.2: Symmetrische oder Secret-Key-Verschlüsselung

Alle klassischen Chiffren gehören zum Typ symmetrisch. Beispiele dazu finden Sie in den CT-Programmen, im Kapitel 2 (P&B- und Vor-Computer-Chiffren)[5] in diesem Buch oder in [4]. In diesem Unterkapitel wollen wir jedoch nur die moderneren symmetrischen Verfahren betrachten.

Der Hauptvorteil von symmetrischen Algorithmen ist ihre hohe Geschwindigkeit, mit denen Daten ver- und entschlüsselt werden. Hauptnachteil ist der hohe Aufwand für die Schlüsselverteilung. Um miteinander vertraulich kommunizieren zu können, müssen Sender und Empfänger vor Beginn der eigentlichen Kommunikation über einen sicheren Kanal einen Schlüssel ausgetauscht haben. Spontane Kommunikation zwischen Personen, die sich vorher noch nie begegnet sind, scheint so nahezu unmöglich. Soll in einem Netz mit n Teilnehmern jeder mit jedem zu jeder Zeit spontan kommunizieren können, so muss jeder Teilnehmer

Blowfish, Twofish, Threefish, ChaCha, LAMBDA1, Format-erhaltende Verschlüsselung, visuelle Kryptografie und einige mehr.
- In der `JCT Algorithmen-Perspektive` stehen alle modernen Verschlüsselungsverfahren aus der Bouncy Castle Crypto API (siehe `www.bouncycastle.org`) zur Verfügung: AES, Rijndael, Camellia, DES, Dragon, IDEA, LFSR, MARS, Misty1, RC2, RC5, RC6, SAFER+, SAFER++, Serpent, Shacal, Shacal2, Twofish.

[5] P&B = Papier und Bleistift (englisch P&P = Paper-and-Pencil)

vorher mit jedem anderen der $n-1$ Teilnehmer einen Schlüssel ausgetauscht haben. Insgesamt müssen also $n(n-1)/2$ Schlüssel ausgetauscht werden.

Der aktuelle Standard unter den modernen symmetrischen Verfahren ist AES.

1.3.1 AES (Advanced Encryption Standard)[6]

Vor AES war der DES (Data Encryption Standard) das bekannteste moderne, symmetrische Verschlüsselungsverfahren. Der DES-Algorithmus war eine Entwicklung von IBM in Zusammenarbeit mit der National Security Agency (NSA). Er wurde 1975 als Standard veröffentlicht. Trotz seines relativ hohen Alters ist bis heute kein „effektiver" Angriff auf ihn gefunden worden (was „effektiv" genau bedeutet, hängt von der Definition von Sicherheit ab – siehe Abschnitt 1.8 auf Seite 38). Der effektivste Angriff gegen DES besteht aus dem Durchprobieren (fast) aller möglichen Schlüssel, bis der richtige gefunden wird (*Brute-Force-Angriff*). Aufgrund der relativ kurzen Schlüssellänge von effektiv 56 bit (64 bit, die allerdings 8 Paritätsbits enthalten)[7], sind in der Vergangenheit schon mehrfach mit dem DES verschlüsselte Nachrichten gebrochen worden, so dass er heute nicht mehr als sicher anzusehen ist. Alternativen zum DES sind zum Beispiel die Algorithmen Triple-DES (TDES, 3DES) und vor allem AES.

Standard unter den symmetrischen Verfahren ist heute AES: Der dazu gehörende Rijndael-Algorithmus wurde am 2. Oktober 2000 zum Gewinner der AES-Ausschreibung erklärt und ist damit Nachfolger des DES-Verfahrens. AES wurde ausgiebigen Untersuchungen unterzogen und widerstand bisher allen praktischen Angriffsversuchen.

Weitere Informationen zum AES finden sich in Abschnitt 9.2.14 auf Seite 559. Einen Einstieg und Informationen zu den AES-Kandidaten der letzten Runde finden Sie z. B. in der Online-Hilfe von CT1[8] oder in Wikipedia[9]. Im Anhang 1.12 zu diesem Kapitel wird vorgestellt, wie der AES in CTO animiert ist, und wie der AES ausgeführt wird: in CT2 und mit OpenSSL.

1.3.2 Aktueller Stand der Brute-Force-Angriffe auf symmetrische Verfahren

Anhand des Angriffs auf die Blockchiffre RC5-64 kann der aktuelle Stand von Brute-Force-Angriffen auf symmetrische Verschlüsselungsverfahren gut erläutert werden. Bei einer Schlüssellänge von 64 bit sind dies maximal 2^{64} = 18 446 744 073 709 551 616 oder rund 18 Trillionen (= $18 \cdot 10^{18}$) zu überprüfende Schlüssel.

Als Brute-Force-Angriff (exhaustive search, trial-and-error) bezeichnet man das vollständige Durchsuchen des Schlüsselraums: Dazu müssen keine besonderen Analysetechniken eingesetzt werden. Der Angreifer kennt nur den Geheimtext, er führt also einen Ciphertext-only-Angriff durch, was die schwächste aller möglichen Angriffsvoraussetzungen ist. Dazu wird der Geheimtext mit allen möglichen Schlüsseln des

[6] - In CTO kann man sich im Browser AES in 2 Plugins ansehen: als „AES-Animation" `https://www.cryptool.org/de/cto/aes-animation` und per „AES (Schritt-für-Schritt)" `https://www.cryptool.org/de/cto/aes-step-by-step`.
- Über `CT1 Einzelverfahren ▷ Visualisierung von Algorithmen ▷ AES` finden Sie 3 Visualisierungen dieses Verfahrens.
- Mithilfe des Suchstrings „AES" bei `CT2 Startcenter ▷ Vorlagen` können Sie ein Plugin finden, das den Algorithmus Schritt-für-Schritt visualisiert.

[7] Als Einheit in Formeln schreiben wir „bit" klein und ohne Plural-s. Siehe Anhang B.2 auf Seite 866.

[8] Online-Hilfe von CT1: Das Stichwort **AES** im Index führt zu den drei Hilfeseiten: **AES-Kandidaten**, **Der AES-Gewinner Rijndael** und **Der AES-Algorithmus Rijndael**.
Eine ausführliche und sehr ansprechende Beschreibung in deutsch von AES mit C-Code findet sich in [5].

[9] `https://de.wikipedia.org/wiki/Advanced_Encryption_Standard`

Schlüsselraums entschlüsselt[10] und geprüft, ob der resultierende Text einen sinnvollen Klartext ergibt[11]. Siehe Abschnitt 1.7 auf Seite 31 zu „Schlüsselraum“ und Abschnitt 1.8.6 zum Begriff „n-bit-Sicherheit“.

Um zu zeigen, welche Sicherheit bekannte symmetrische Verfahren wie DES, 3DES oder RC5 haben, veranstaltete z. B. RSA Security sogenannte Cipher-Challenges. Unter kontrollierten Bedingungen wurden Preise ausgelobt, um Geheimtexte (verschlüsselt mit verschiedenen Verfahren und verschiedenen Schlüssellängen) zu entschlüsseln und den symmetrischen Schlüssel zu ermitteln.[12]

Dass das „alte“ Standard-Verfahren DES mit der fixen Schlüssellänge von 56 bit nicht mehr sicher ist, wurde schon im Januar 1999 von der Electronic Frontier Foundation (EFF) demonstriert, als sie mit Deep Crack eine DES-verschlüsselte Nachricht in weniger als einem Tag knackte.[13]

Der aktuell bekannte Rekord für starke symmetrische Verfahren liegt bei 64 bit langen Schlüsseln. Dazu wurde das Verfahren RC5 benutzt, eine Blockchiffre mit variabler Schlüssellänge.

Die RC5-64 Challenge wurde im Juli 2002 nach 5 Jahren vom distributed.net-Team gelöst.[14] Insgesamt arbeiteten 331 252 Personen gemeinsam über das Internet zusammen.[15] Getestet wurden rund 15 Trillionen ($= 15 \cdot 10^{18}$) Schlüssel, bis der richtige Schlüssel gefunden wurde. Das waren rund 85 % des Suchraums.[16]

Somit sind symmetrische Verfahren mit 64 bit langen Schlüsseln (auch wenn sie keinerlei kryptografische Schwächen haben) keine geeigneten Verfahren mehr, um sensible Daten über einen längeren Zeitraum geheim zu halten.

> Das BSI fordert für nach 2024 eingesetzte moderne symmetrische Verfahren ein Sicherheitsniveau von 128 bit (siehe [6], Seite 20, Tab. 1.2).
> Das BSI-Dokument [6] empfiehlt nicht nur einen Algorithmus wie AES-128, sondern legt auch Details wie passende Blockmodi und Paddingsverfahren fest.

[10] - Über `CT1 Analyse ▷ Symmetrische Verschlüsselung (modern)` kann man Brute-Force-Analysen von modernen symmetrischen Verfahren durchführen.
- Mit `CT2 Vorlagen ▷ Kryptoanalyse ▷ Modern` können Sie ebenfalls Brute-Force-Analysen durchführen. Besonders mächtig ist die darin verwendete KeySearcher-Komponente, die die Ausführung auf mehrere Rechner verteilen kann.

[11] Ist der Klartext natürlich-sprachlich und wenigstens 100 B lang, kann diese Prüfung ebenfalls automatisiert durchgeführt werden. Um in einer sinnvollen Zeit auf einem Einzel-PC ein Ergebnis zu erhalten, dürfen nicht zu viele Bits des Schlüssels als unbekannt gekennzeichnet sein. Auf einem aktuellen PC in 2022 probiert CT1 für AES 24 bit in ca. 20 sec durch, aber 32 bit dauern schon 1:45 h. Vergleiche die Screenshots in Abschnitt 1.7 auf Seite 31.

[12] `https://web.archive.org/web/20170417095446/http://www.emc.com/emc-plus/rsa-labs/historical/the-rsa-laboratories-secret-key-challenge.htm`
Im Mai 2007 meldete RSA Inc leider, dass sie die Korrektheit der bis dahin nicht gelösten RC5-72 Challenge nicht bestätigen werden. Alternative: Ein weites Spektrum einfacher und komplexer, symmetrischer und asymmetrischer Rätsel-Aufgaben bietet der internationale Krypto-Wettbewerb **MysteryTwister**: `https://www.mysterytwister.org`.

[13] `https://web.archive.org/web/20170417095101/https://www.emc.com/emc-plus/rsa-labs/historical/des-challenge-iii.htm`

[14] `https://www.distributed.net/Pressroom_press-rc5-64`
`https://www.distributed.net/images/9/92/20020925_-_PR_-_64_bit_solved.pdf`

[15] Eine Übersicht über deren aktuelle Projekte (letzter Update war am 17.5.2020) zum verteilten Rechnen finden Sie unter: `https://distributedcomputing.info/`

[16] CT2 hat begonnen, mit einer allgemeinen Infrastruktur für verteiltes Rechnen zu experimentieren (CrypCloud, die sowohl Peer-to-Peer als auch zentralisiert eingesetzt werden kann). Damit wird CT2 in der Zukunft in die Lage versetzt, Berechnungen auf viele Computer zu verteilen. Was man erreichen kann, wenn die Komponenten für die Parallelisierung eingerichtet sind, zeigte ein Cluster zur verteilten Kryptoanalyse von DES und AES: Am 21. März 2016 wurde ein Brute-force-Angriff zur verteilten Schlüsselsuche gegen AES auf 50 i5-PCs verteilt, jeder mit 4 virtuellen CPU-Kernen. Diese 200 virtuellen *Worker Threads* konnten ca. 350 Millionen AES-Schlüssel/s testen („s“ ist Einheitszeichen für Sekunde). Die *Cloud* verarbeitete dabei insgesamt ca. 20 GB/s an Daten. CrypCloud ist eine Volunteering-Cloud, so dass CT2-Nutzer sich freiwillig bei verteilten Jobs anschließen können.

1.4 Asymmetrische Verschlüsselung

Bei der *asymmetrischen* Verschlüsselung (auch Public-Key-Verschlüsselung genannt) hat jeder Teilnehmer ein eigenes Schlüssel**paar**, das aus einem *geheimen* Schlüssel (genannt *privater* Schlüssel) und einem *öffentlichen* Schlüssel besteht. Der öffentliche Schlüssel wird, der Name deutet es an, öffentlich bekanntgemacht – zum Beispiel in einem sogenannten Zertifikat (siehe Abschnitt 7.4.2 auf Seite 460) oder in einem Schlüsselverzeichnis im Internet (diese Art von „Schwarzem Brett“ wird auch Directory oder öffentlicher Schlüsselring genannt).

Abbildung 1.3 veranschaulicht die asymmetrische Ver- und Entschlüsselung.

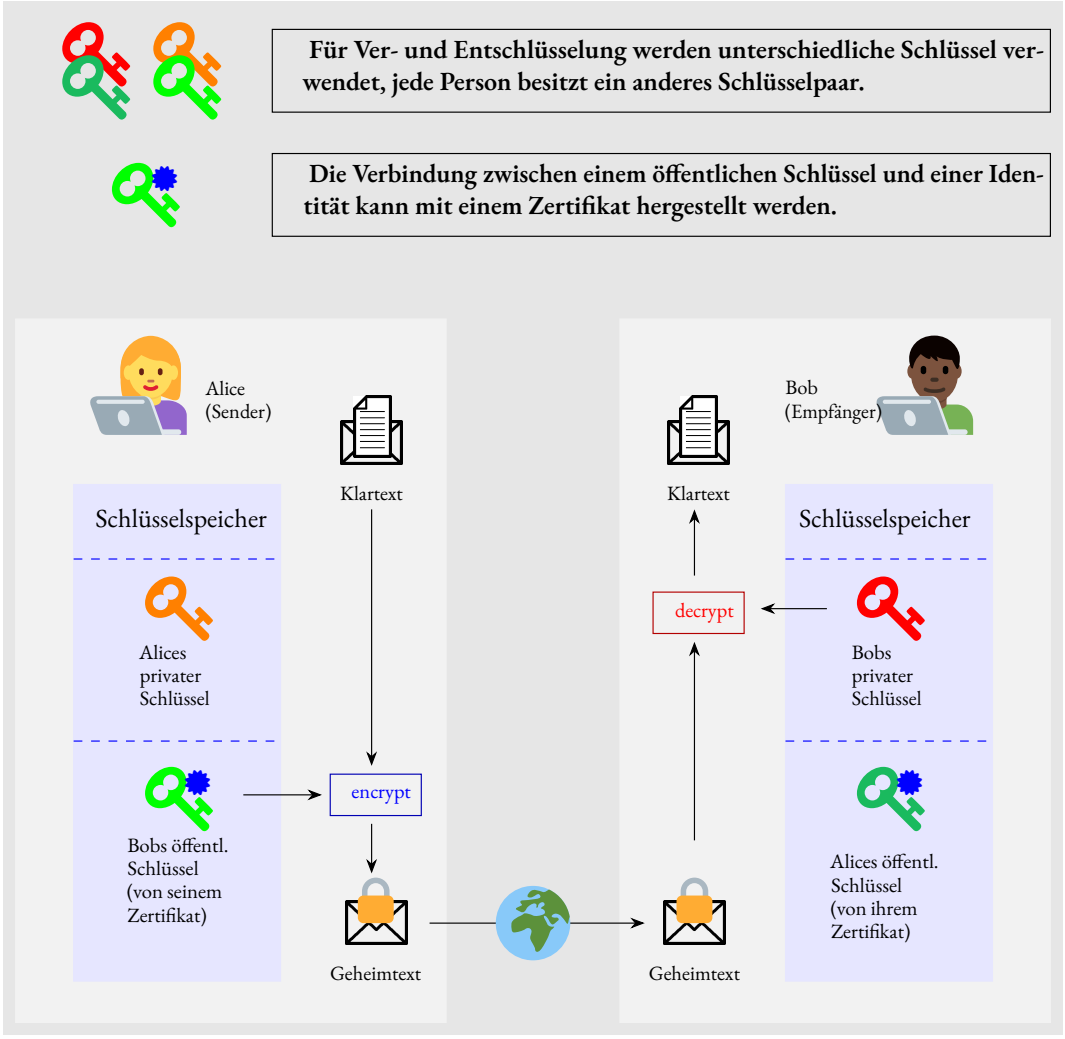


Abb. 1.3: Asymmetrische oder Public-Key-Verschlüsselung

Möchte Alice[17] mit Bob kommunizieren, sucht sie Bobs öffentlichen Schlüssel und benutzt diesen, um ihre

[17] Zur Beschreibung kryptografischer Protokolle werden den Teilnehmern oft Vornamen gegeben (vergleiche [7, S. 23]). Alice und Bob führen alle allgemeinen 2-Personen-Protokolle durch, wobei Alice diese initiiert und Bob antwortet. Die Angreifer werden als Eve (eavesdropper = passiver Lauscher) und Mallory (malicious active attacker = böswilliger, aktiver Angreifer) bezeichnet.

Nachricht (Klartext) an ihn zu verschlüsseln. Den verschlüsselten Text schickt sie dann an Bob, der mithilfe seines privaten Schlüssels den Text wieder entschlüsseln kann. Da einzig Bob Kenntnis von seinem privaten Schlüssel hat, ist auch nur er in der Lage, an ihn adressierte Nachrichten zu entschlüsseln. Selbst Alice als Absenderin der Nachricht kann aus der von ihr versandten (verschlüsselten) Nachricht den Klartext nicht wieder herstellen.[18] Asymmetrische Verfahren sind so designt, dass man aus dem öffentlichen Schlüssel nicht auf den geheimen Schlüssel schließen kann.

Veranschaulichen kann man sich ein solches Verfahren mit einer Reihe von einbruchssicheren Briefkästen. Wenn ich eine Nachricht verfasst habe, so suche ich den Briefkasten mit dem Namensschild des Empfängers und werfe den Brief dort ein. Danach kann ich die Nachricht selbst nicht mehr lesen oder verändern, da nur der legitime Empfänger im Besitz des Schlüssels für den Briefkasten ist.

Vorteil von asymmetrischen Verfahren ist das einfachere Schlüsselmanagement. Betrachten wir wieder ein Netz mit n Teilnehmern. Um sicherzustellen, dass jeder Teilnehmer jederzeit eine verschlüsselte Verbindung zu jedem anderen Teilnehmer aufbauen kann, muss jeder Teilnehmer ein Schlüsselpaar besitzen. Man braucht also $2n$ Schlüssel oder n Schlüsselpaare. Ferner ist im Vorfeld einer Übertragung kein sicherer Kanal notwendig, da alle Informationen, die zur Aufnahme einer vertraulichen Kommunikation notwendig sind, offen übertragen werden können. Hier ist „lediglich“ auf die Unverfälschtheit (Integrität und Authentizität) des öffentlichen Schlüssels zu achten. Nichtsdestotrotz sind die Anforderungen an die Schlüsselgenerierung nicht trivial.[19] Außerdem ist zu beachten, dass inzwischen auch die (Public-Key-)Infrastrukturen selbst Ziel von Cyber-Angriffen sind. **Nachteil**: Im Vergleich zu symmetrischen Verfahren sind reine asymmetrische Verfahren um ein Vielfaches langsamer (siehe Abschnitt 1.5 auf der nächsten Seite).

Das bekannteste asymmetrische Verfahren ist der RSA-Algorithmus[20,21], der nach seinen Entwicklern Ronald Rivest, Adi Shamir und Leonard Adleman benannt wurde. Der RSA-Algorithmus wurde 1978 veröffentlicht.[22] Das Konzept der asymmetrischen Verschlüsselung wurde erstmals von Whitfield Diffie und Martin Hellman in Jahre 1976 vorgestellt. Es ist erwähnenswert, dass sowohl das Konzept als auch das, was wir heute RSA-Verschlüsselung und Diffie-Hellman-Protokoll nennen, beim britischen Geheimdienst GCHQ schon 1973 bekannt war, bevor es unabhängig davon von Diffie, Hellman, Rivest, Shamir, Adleman und Merkle wiederentdeckt wurde.[23] Heute spielen auch die Verfahren nach ElGamal eine bedeutende Rolle, vor allem die Schnorr-Varianten im DSA (Digital Signature Algorithm).

[18] In der Praxis werden asymmetrische Verfahren nicht dazu benutzt, um die ganze Nachricht zu verschlüsseln, sondern nur einen Sessionkey (siehe Abschnitt 1.5 auf der nächsten Seite).

[19] Was schief gehen kann, wird z. B. in Abschnitt 5.12.5.4 erläutert.

[20] RSA wird in diesem Buch ab Abschnitt 5.10 auf Seite 303 ausführlich beschrieben. Die aktuellen Forschungsergebnisse im Umfeld von RSA werden in Abschnitt 5.12 auf Seite 307 beschrieben. In Abschnitt 6.5 auf Seite 401 wird der RSA-Algorithmus aus der Zahlentheorie heraus tiefer begründet: Die RSA-Ebene ist ein Modell, um die Vorgänge bei diesem Algorithmus an Bildern von Rechtecken zu veranschaulichen.

[21] Das RSA-Kryptosystem kann mit CrypTool in vielen Variationen nachvollzogen werden:
- CTO hat dazu 2 ausführliche Plugins:
Via „RSA (Schritt-für-Schritt)“ `https://www.cryptool.org/de/cto/rsa-step-by-step`.
Via „RSA-visuell und mehr“ `https://www.cryptool.org/de/cto/rsa-visual` kann man sich die Verschlüsselungs-Zuweisungen von RSA grafisch ansehen, Textbook-RSA auch mit großen Zahlen ausprobieren, und auch das in der Praxis implementierte RSA mit OAEP, Padding und Zertifikaten verwenden.
- Über `CT1 Einzelverfahren ▷ RSA-Kryptosystem ▷ RSA-Demo` können Blocklänge und Alphabet von Textbook-RSA variiert werden. Über `CT1 Ver-/Entschlüsseln ▷ Asymmetrisch` können Nachrichten schnell mit RSA ver- und entschlüsselt werden.
- Unter `CT2 Vorlagen ▷ Kryptografie ▷ Modern ▷ Asymmetrisch` finden Sie asymmetrische Verfahren wie RSA.
- Sowohl `JCT Standard-Perspektive ▷ Visualisierungen` als auch `JCT Algorithmen-Perspektive` bieten asymmetrische Verfahren wie RSA.

[22] Hinweise zur Geschichte von RSA und seiner Veröffentlichung, die nicht im Sinne der NSA war, finden sich in der Artikelserie *RSA & Co. in der Schule: Moderne Kryptologie, alte Mathematik, raffinierte Protokolle*. Siehe [8], S. 55 ff („Penible Lämmergeier“).

[23] Diese Information wurde 1997 freigegeben. Siehe die 1-seitige Beschreibung von Clifford Cocks aus 1973: „A note on 'non-secret encryption'“ (`https://web.archive.org/web/20080227001905/http://www.cesg.gov.uk/site/publications/media/notense.pdf`). Zur historischen Einordnung, siehe die Laudatio auf den englischen Mathematiker Clifford Cocks, gehalten von Nigel Smart 2008 (`https://www.bristol.ac.uk/alumni/our-alumni/honorary-degrees/honorary-graduates/2008/cocks.html`).

Das BSI fordert für über das Jahr 2024 hinaus eingesetzte Verfahren ein Sicherheitsniveau von 128 bit. Für ein gleichwertiges Sicherheitsniveau empfiehlt die entsprechende Technische Richtlinie (siehe [6], Seite 20, Tabelle 1.2) für RSA eine Schlüssellänge von 3000 bit.

Angriffe gegen asymmetrische Verfahren werden behandelt in

- Kapitel 5: Elementare Zahlentheorie,
- Kapitel 6: Moderne Kryptografie,
- Kapitel 8: Elliptische Kurven und
- Kapitel 12: Aktuelle Resultate zum Lösen diskreter Logarithmen und zur Faktorisierung.

1.5 Hybridverfahren[24]

Um die Vorteile von symmetrischen und asymmetrischen Techniken gemeinsam nutzen zu können, werden (zur Verschlüsselung) in der Praxis meist Hybridverfahren verwendet.

Hierbei werden die Mengen-Daten (bulk data) mittels symmetrischer Verfahren verschlüsselt: Der Schlüssel ist ein vom Absender zufällig[25] generierter, geheimer Sitzungsschlüssel (session key), der nur für diese Nachricht verwendet wird. Anschließend wird dieser Sitzungsschlüssel mithilfe des asymmetrischen Verfahrens verschlüsselt und zusammen mit der Nachricht an den Empfänger übertragen. Der Empfänger kann den Sitzungsschlüssel mithilfe seines geheimen Schlüssels bestimmen und mit diesem dann die Nachricht entschlüsseln.

Auf diese Weise profitiert man von dem handhabbaren Schlüsselmanagement asymmetrischer Verfahren (mit öffentlichem und privatem Schlüssel), und man profitiert von der Schnelligkeit symmetrischer Verfahren, um große Datenmengen zu verschlüsseln (mit den geheimen Schlüsseln).

Erkläre es mir, ich werde es vergessen.
Zeige es mir, ich werde es vielleicht behalten.
Lass es mich tun, und ich werde es können.

Zitat 1: Indisches Sprichwort

[24] - Über `CT1 Ver-/Entschlüsseln ▷ Hybrid` können Sie die einzelnen Schritte und ihre Abhängigkeiten mit konkreten Zahlen nachvollziehen. Die Variante mit RSA als asymmetrischem Verfahren ist grafisch visualisiert; die Variante mit ECC nutzt die Standard-Dialoge. In beiden Hybrid-Fällen wird AES als symmetrisches Verfahren eingesetzt.
- Unter `JCT Algorithmen-Perspektive ▷ Hybride Verschlüsselung` finden sich ebenfalls Hybrid-Verfahren.

[25] Die Erzeugung zufälliger Zahlen ist ein wichtiger Bestandteil von kryptografisch sicheren Verfahren.
- Über `CT1 Einzelverfahren ▷ Zufallsdaten erzeugen` können Sie verschiedene Zufallszahlengeneratoren (PRNGs) ausprobieren und damit Binärdateien erzeugen.
- Über `CT1 Analyse ▷ Zufallsanalyse` können Sie verschiedene Zufalls-Testverfahren auf Binärdateien anwenden.
- Unter `CT2 Startcenter ▷ Vorlagen` können Sie mit dem Suchstring „Zufall“ sowohl Zufallsgeneratoren (PRNGs) als auch Key-Derivation Functions (KDF) finden. Die PRNGs nutzen intern bspw. Keccak oder den Linear Congruential Generator (LCG). Das Ergebnis wird dann z. B. für Schlüsselgenerierung oder Dezimalisierung genutzt.
- Sowohl `JCT Standard-Perspektive ▷ Algorithmen` als auch die `JCT Algorithmen-Perspektive` bieten **Pseudo**-Zufallszahlengeneratoren an.

1.6 Kerckhoffs' Prinzip

Der holländische Kryptologe Auguste Kerckhoffs formulierte 1883 sechs Grundsätze zur Konstruktion von sicheren militärischen Verschlüsselungsverfahren. Der zweite Grundsatz, Kerckhoffs' Prinzip oder Kerckhoffs' Maxime, gilt heute als Grundsatz der modernen Kryptografie. Dieses Prinzip besagt, dass die Sicherheit eines Verschlüsselungsverfahrens auf der Geheimhaltung des Schlüssels beruht, aber nicht auf der Geheimhaltung des Verschlüsselungsalgorithmus. Dem Kerckhoffs' Prinzip wird oft die sogenannte „Security through Obscurity" gegenübergestellt: Sicherheit durch die zusätzliche Geheimhaltung des Verschlüsselungsalgorithmus.

Kerckhoffs' Prinzip wurde mehrfach abgewandelt. Bspw. formulierte Claude Shannon, dass man Verschlüsselungsssysteme unter der Annahme designen soll, dass ein Feind das System von Anfang an genau kennt (Shannons Maxime).

1.7 Schlüsselräume – theoretische und praktische

Bei den heutzutage verwendeten guten Verschlüsselungsverfahren ist der Zeitaufwand zum Brechen so hoch, dass sie praktisch nicht gebrochen werden können. Deshalb werden diese Verfahren als (praktisch) sicher angesehen – aus einer rein auf den Algorithmus bezogenen Sichtweise.[26]

Der Schlüsselraum einer Chiffre ist eine wichtige Kennzahl für die Sicherheit einer Chiffre. Bei einer monoalphabetischen Substitution (MASC; auch einfache Substitution genannt) über einem Alphabet der Länge k beträgt der Schlüsselraum z. B. $k!$. Bei AES-128 beträgt er 2^{128}. Ein (ausreichend) großer Schlüsselraum (ca. 2^{100}) ist eine notwendige Voraussetzung für eine sichere Chiffre, aber keine hinreichende Bedingung: Die gerade genannte MASC hat einen großen Schlüsselraum (bei einem Alphabet von 26 Zeichen hat der Schlüsselraum eine Größe von rund $2^{88{,}4}$, was der Anzahl möglicher Geheimtextalphabete entspricht), aber die MASC konnte schon jahrhundertelang mit Häufigkeitsanalysen geknackt werden.

Aus dem Schlüsselraum wird der Aufwand für einen Brute-Force-Angriff (BF-Angriff) hergeleitet, also für das systematische Durchprobieren aller möglichen Schlüssel. Ist der Schlüsselraum so klein, dass ein Angreifer einen kompletten BF-Angriff durchführen kann, ist das Verfahren nicht nur theoretisch, sondern auch praktisch gebrochen.

Beim BF-Angriff entschlüsselt der Angreifer den gegebenen Geheimtext (oder Teile davon) mit jedem möglichen Schlüssel (siehe Abschnitt 1.3.2 auf Seite 26). Dann wird der gefundene Klartext bewertet. Wie erstaunlich gut Fitness-Algorithmen korrekte natürliche Texte erkennen können, kann man in CT1[27] in

[26] Insbesondere seit den Informationen von Edward Snowden gab es viele Diskussionen, ob Verschlüsselung sicher ist. In [9] wird das Ergebnis einer Evaluierung vorgestellt, auf welche Kryptografie man sich verlassen kann – nach dem heutigen Kenntnisstand. Der Artikel untersucht: Welche Krypto-Verfahren können im Lichte der NSA-Enthüllungen noch als sicher gelten? Wo wurden Systeme gezielt geschwächt? Wie können wir die kryptografische Zukunft sicher gestalten? Wie unterscheiden sich Mathematik und Implementierung?

[27] `CT1 Analyse ▷ Symmetrische Verschlüsselung (modern) ▷ AES (CBC)`
Der Schlüssel wird in hex dargestellt. Jedes Hexzeichen entspricht 4 bit. Bei 7 unbekannten Hexzeichen (dargestellt als Sterne) sind 28 bit unbekannt und die vollständige Suche dauerte rund 4 Minuten. Bei 20 unbekannten bit ist es sofort fertig, bei 24 dauerte es ca. 1/4 min; bei einer 32-bit-Suche (also 8 Sterne) dauerte es rund 1 Stunde. Mit jedem weiteren unbekannten Hexzeichen kommen 4 bit hinzu und die Suchdauer versechzehnfacht sich ($2^4 = 16$).

Abb. 1.4 und 1.5 auf dieser Seite und auf der nächsten Seite; und in CT2[28] in Abb. 1.6 auf der nächsten Seite sehen. CT1 nutzt ähnliche Fitness-Funktionen wie die Solver und Analyzer in CT2.

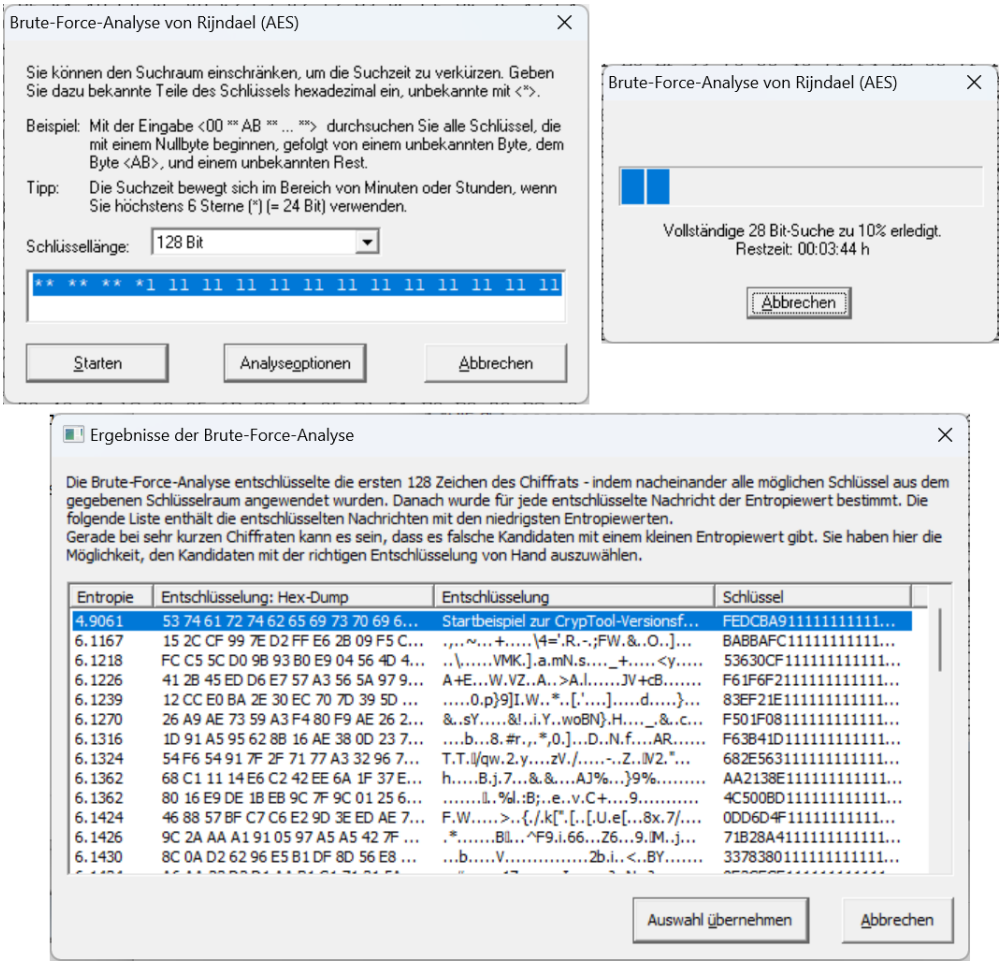


Abb. 1.4: Dialoge für die Brute-Force-Analyse von AES in CT1 bei teilweise bekanntem Schlüssel

Ob ein Angreifer wirklich den maximalen, theoretischen Schlüsselraum durchprobieren muss, ist zumindest bei den älteren Chiffren fraglich. Deshalb werden auch die Begriffe **praktischer Schlüsselraum** (eingeführt von Ralph Simpson für historische Verschlüsselungsgeräte) und **Angriffszeit** betrachtet. Die „Angriffszeit" hängt eng zusammen mit der n-bit-Sicherheit (siehe Abschnitt 1.8.6).

[28] `CT2 Vorlagen ▷ Kryptoanalyse ▷ Modern ▷ AES-Analyse per Entropie (2) – mit veränderbarem Klartext`
Während man sich in CT1 erst eine Datei verschlüsseln muss, ist bei CT2 der ganze Vorgang in den beigelegten Templates enthalten: CT2 hat 4 Vorlagen zu BF-Angriffen gegen AES. Während in CT1 als Fitness-Algorithmus immer die Entropie der ersten x Bytes genutzt wird, kann man in den Einstellungen des CT2-KeySearchers festlegen, welcher von den 3 Fitness-Algorithmen (Entropie, N-Gramme, Koinzidenzindex IoC) genutzt werden soll oder ob nach einem Crib (Klartext-Teil) gesucht werden soll. Das Crib kann man mittels regulärem Ausdruck bestimmen. Auch für Schlüssel kann man sein Vorwissen m. H. eines regulären Ausdrucks wie `00-01-02-03-04-**-**-**-[01]8-09-0A-0B-0C-0D-0E-0F` angeben.

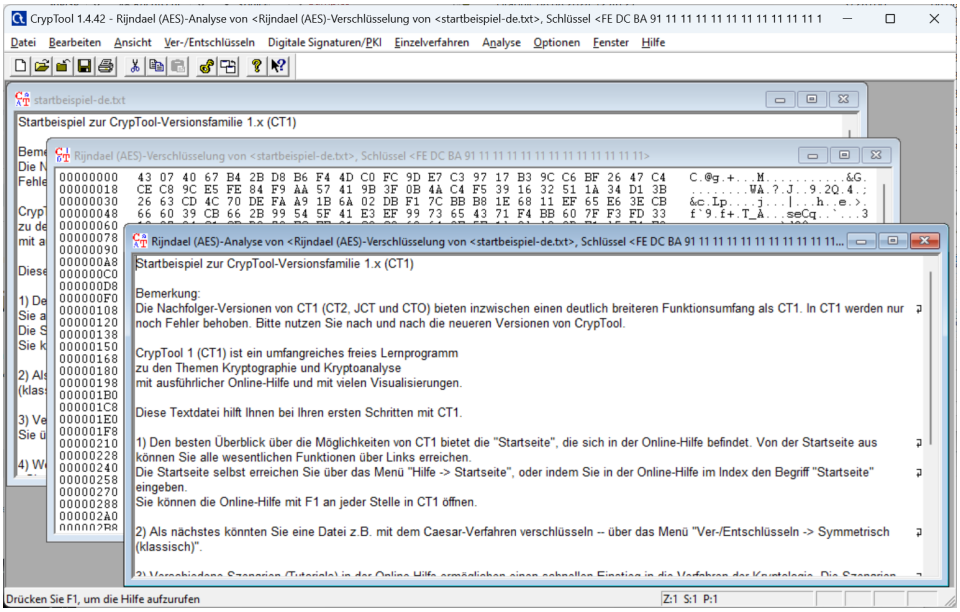


Abb. 1.5: Ergebnis im Hauptfenster von CT1 nach der Brute-Force-Analyse von AES

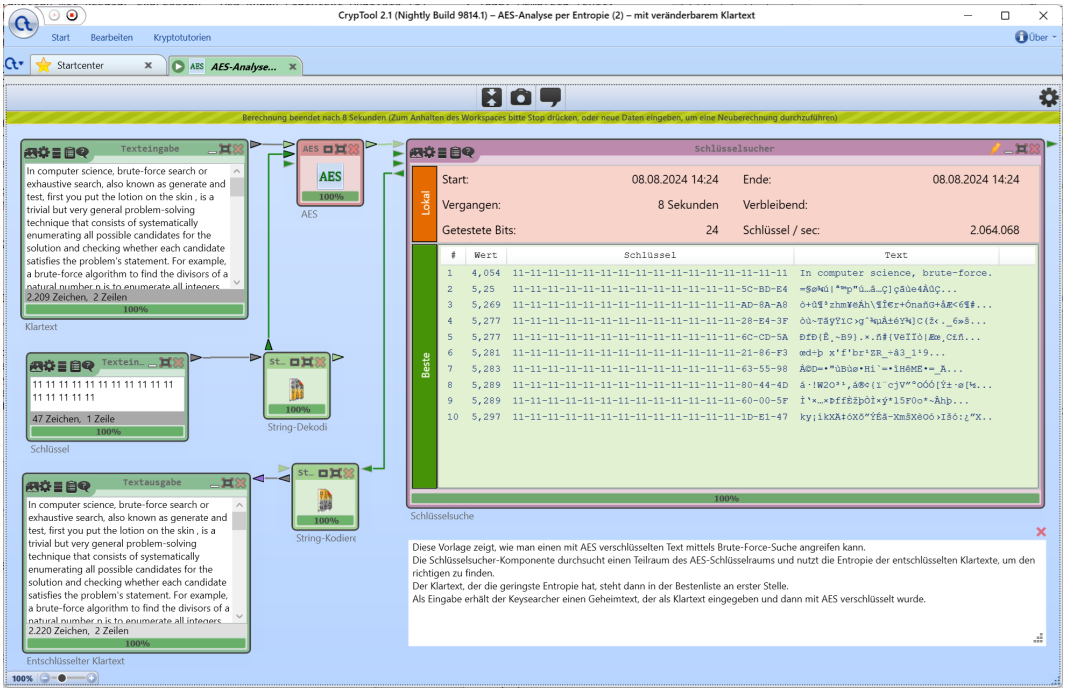


Abb. 1.6: Brute-Force-Analyse von AES in CT2 bei teilweise bekanntem Schlüssel

1.7.1 Schlüsselräume historischer Verschlüsselungsgeräte

Die Größe der Schlüsselräume historischer Verschlüsselungsgeräte wird in der populären Presse oft mit gigantischen Zahlen angegeben, um den Leser von der unglaublichen Stärke der Verschlüsselung zu überzeugen. Dies ist oft die Einleitung zu einer Geschichte über den erstaunlichen Einfallsreichtum der Codebrecher, die deren Verschlüsselung knackten.

Beispielsweise ist der Schlüsselraum der berüchtigten Enigma I größer als die Anzahl der Atome im Universum. Nach Tabelle 1.1 auf Seite 38 beträgt der theoretische Schlüsselraum der Enigma ca. $3 \cdot 10^{114}$, während die Anzahl der Atome im Universum ca. 10^{77} beträgt (nach Tabelle 4.15 auf Seite 256).

Es gibt zwei Hauptprobleme mit den Schlüsselräumen historischer Chiffriergeräte. Das erste Problem ist, dass der Schlüsselraum ein irreführendes Maß für die Stärke der Verschlüsselung sein kann. Der Grund für diese Verwirrung liegt darin, dass der Schlüsselraum eines modernen symmetrischen Chiffriersystems normalerweise ein gutes Maß für die Stärke der Verschlüsselung darstellt. Historische Geräte sind jedoch mechanisch oder elektromechanisch, was die Zufälligkeit der Verschlüsselung einschränkt. Dies bedeutet, dass Methoden entwickelt werden können, um diese Verschlüsselung zu brechen, ohne dass Brute-Force-Methoden erforderlich sind. Es sei daran erinnert, dass der Schlüsselraum nur ein Maß für die Brute-Force-Methoden ist, die zum Brechen einer Verschlüsselung erforderlich sind, ohne die Methoden zu berücksichtigen, die von Kryptoanalytikern verwendet werden, um (viele) Teile dieses Schlüsselraums zu verkürzen.

Das zweite Problem bei den Schlüsselräumen historischer Geräte sind die großen Abweichungen, die oft für ein und dasselbe Gerät angegeben werden. Diese Abweichungen sind in der Regel auf unterschiedliche Grundannahmen zurückzuführen, die jedoch nicht immer angegeben werden.

Ein weiterer Punkt, der bei Schlüsselräumen berücksichtigt werden muss, ist die Tatsache, dass die Methoden zur Kryptoanalyse einiger historischer Geräte erst viele Jahrzehnte oder sogar Jahrhunderte nach ihrer Erfindung entwickelt wurden. So wurde beispielsweise die 1466 erfundene Vigenère-Scheibe (Alberti-Scheibe) noch 1917 von der Zeitschrift Scientific American als unknackbar bezeichnet. Dieser Artikel erschien im selben Jahr, in dem Joseph Mauborgne, Chief Signal Officer der US-Armee, damit prahlte, dass seine Kryptografen die Vigenère-Scheibe schneller entschlüsseln könnten als der Feind seine eigenen Nachrichten.

Trotz der aufgezeigten Probleme ist die Untersuchung der Schlüsselräume historischer Chiffriergeräte ein nützliches Instrument, um die Denkweise der Erfinder, Anwender und Codebrecher besser zu verstehen. Mit modernen Methoden können wir den Wert der Schlüsselräume historischer Geräte „verunglimpfen“, aber das allein würde nicht dazu beitragen, um zu verstehen, warum historische Entscheidungen auf der Grundlage der (damaligen) Stärke der Verschlüsselung getroffen wurden, die diese großen Schlüsselräume implizierten.

1.7.2 Welche Annahmen sollte man bei der Berechnung der Schlüsselräume treffen

Nur wenn man einen gemeinsamen Satz von Annahmen definiert, kann man die Schlüsselräume der historischen Geräte transparent berechnen und vergleichen. Der am häufigsten zitierte Schlüsselraum über die Enigma stammt aus dem NSA-Dokument [10]. Die darin verwendeten Annahmen wurden benutzt, um die Tabelle der historischen Schlüsselräume (Tabelle 1.1 auf Seite 38) zu erstellen. Das NSA-Dokument wurde von Ray Miller verfasst und erstmals 1995 veröffentlicht. In diesem Dokument beschreibt Miller einen maximalen und einen „praktischen“ Schlüsselraum, aber leider hat er damals die verwendeten Annahmen nicht explizit definiert.

„Maximaler Schlüsselraum“ vs. „Praktischer Schlüsselraum“ vs. „Work-Faktor“ Miller verwendete den Begriff **maximaler Schlüsselraum** für die theoretisch maximale Anzahl von Einstellungen, die für einen Brute-Force-Angriff getestet werden müsste. Gemäß dem Kerckhoffs'schen Prinzip ging er davon aus, dass der Feind das Gerät erbeutet hat, aber alle vor Ort austauschbaren Teile unbekannt sind, wie z. B. die Rotoren und Reflektoren. Es müssten also alle möglichen Verdrahtungen von Rotoren und Reflektoren kryptoanalytisch untersucht werden, und es könnte eine beliebige Anzahl von Steckerbrett-Kabeln verwendet werden.

Der **praktische Schlüsselraum** ist ebenfalls eine theoretische Anzahl von Einstellungen, setzt aber voraus, dass die erbeutete Maschine und alle vor Ort austauschbaren Teile bekannt sind und verwendet werden. Das bedeutet, dass die Verdrahtung der Rotoren und des Reflektors bekannt ist, aber die Rotoren, die in die Maschine eingesetzt werden sollen, und deren Reihenfolge sind nicht bekannt. Dies bedeutet auch, dass der Reflektor keinerlei kryptografische Stärke bietet, da seine Verdrahtung bekannt ist. Außerdem sind alle dem Benutzer auferlegten Beschränkungen bekannt und werden ausgenutzt, wie z. B. die Tatsache, dass die Deutschen im Zweiten Weltkrieg meist 10 Steckerbrett-Kabel verwendeten. All diese Faktoren tragen dazu bei, dass der praktische Schlüsselraum meist deutlich geringer ist als der maximale Schlüsselraum.

Ein weiterer Begriff (nicht von Miller verwendet, aber eng mit dem Schlüsselraum verbunden) ist der **Work-Faktor**. Dies ist der Arbeitsaufwand, der tatsächlich erforderlich ist, um eine Verschlüsselung zu knacken. Diese Zahl ist in der Regel kleiner als der praktische Schlüsselraum, da alle bekannten Kryptoanalyse-Techniken als Abkürzungen verwendet werden. Bei der Enigma wurde bspw. die Gesamtzahl der zu prüfenden Einstellungen stark reduziert durch Rejewskis Methode, die Kryptoanalyse des Steckerbretts von der der Rotoren und dem Reflektor zu trennen. Einige dieser Kryptoanalyse-Techniken waren zum Zeitpunkt des Einsatzes nicht bekannt, oder sie waren den Benutzern dieser Chiffriergeräte nicht bekannt.

Der Work-Faktor ist ein Konzept, das vor allem für moderne Chiffriersysteme verwendet wird. Für die historischen Geräte gibt es nur sehr wenige Angaben zum Work-Faktor. Er hängt von der Größe der Nachricht oder der Anzahl der erfassten Nachrichten ab. Und er hängt vom Stand der kryptoanalytischen Techniken ab, die angewendet werden können. Ein Beispiel: Obwohl die Enigma-Maschine einen riesigen theoretischen Schlüsselraum hat, musste die Turing-Welchman-Bombe nur etwa 422 000 Einstellungen überprüfen, um die Enigma zu knacken.[29] Dieser Arbeitsfaktor wird in Tabelle 1.5 auf Seite 51 als „Angriffszeit“ bezeichnet, wenn man die besten Angriffe gegen moderne Chiffren vergleicht. Für DES ist der Work-Faktor drastisch kleiner (2^{43}) als der „praktische“ Schlüsselraum, für AES ist er nur etwa 2 Bit kleiner ($2^{254,4}$).

Definition der Annahmen für den Schlüsselraum Ziel ist es, einen gemeinsamen Satz von Annahmen zu haben, um alle historischen Chiffriergeräte vergleichen zu können, und die Annahmen zu verwenden, die anscheinend die größte Akzeptanz haben. Da Miller seine Annahmen nicht explizit dargelegt hat, mussten sie rekonstruiert werden. Eine sorgfältige Lektüre des NSA-Dokuments führt zu den folgenden Annahmen.

Der von Miller berechnete **maximale** Schlüsselraum beruht auf drei Annahmen:

1. Die Basismaschine wurde erbeutet und ist dem Feind bekannt (nach Kerckhoffs' Prinzip).

[29] Warum nur 422 000? Die britische „Bombe“ prüfte nur die Rotorreihenfolge und die Rotoreinstellungen; Ringeinstellungen und Steckerbrett-Einstellungen wurden dann manuell bestimmt. Bei 3 von 5 gewählten Rotoren gibt es $5 \cdot 4 \cdot 3 = 60$ mögliche Rotorreihenfolgen. Nach den deutschen Bedienungsvorgaben durfte jedoch keine 3-Rotor-Reihenfolge im selben Monat wiederholt werden, wodurch sich die 60 möglichen Reihenfolgen zu Beginn des Monats auf 30 am Ende des Monats reduzierten. Außerdem wurde vorgegeben, dass ein einzelner Rotor am nächsten Tag nicht auf derselben Position steht, wodurch sich die 60 möglichen Rotorreihenfolgen auf 32 reduzierten. Zusammengenommen reduzierten diese beiden Regeln die möglichen Reihenfolgen zu Beginn des Monats auf 32 und am Ende des Monats auf 16, d. h. im Durchschnitt auf 24 Rotorreihenfolgen. Diese durchschnittliche Rotorreihenfolge multipliziert mit den 26^3 Rotoreinstellungen ergab $24 \cdot 17576 = 421\,824$ Einstellungen, die von der Bombe für einen vollen Lauf getestet wurden.

2. Vor Ort austauschbare Teile können ausgetauscht werden, sind also nicht bekannt (z. B. Rotor- und Reflektorverkabelung).
3. Eine „Nachrichteneinstellung“ wird mit jeder Nachricht gesendet, unabhängig von der festen Maschineneinstellung.

Der **praktische** Schlüsselraum, wie er von Miller berechnet wurde, hat vier Annahmen:

1. Die Basismaschine wird erbeutet und ist dem Feind bekannt (nach Kerckhoffs' Prinzip).
2. Vor Ort austauschbare Teile sind ebenfalls erfasst und bekannt.
3. Benutzerbedingte Einschränkungen sind bekannt (z. B. immer 10 Steckerbrett-Kabel verwenden).
4. Eine „Nachrichteneinstellung“ wird mit jeder Nachricht gesendet, unabhängig von der festen Maschineneinstellung.

Erläuterung der NSA-Schlüsselraum-Annahmen Diese Annahmen scheinen vernünftig und einfach zu sein, mit Ausnahme der jeweils letzten Annahme: Eine „Nachrichteneinstellung“ wird mit der Nachricht gesendet, getrennt von der festen Maschineneinstellung. Die Bedeutung und Auswirkung dieser Annahme bedarf einer weiteren Erklärung.

Bei der Enigma sind alle möglichen Verdrahtungen der Rotoren im maximalen Schlüsselraum enthalten. Miller bezieht aber auch die Startpositionen der Rotoren mit ein. Die Aufnahme der Startposition des Rotors in den Schlüsselraum – abgesehen davon, dass sie bereits in allen möglichen Verdrahtungen enthalten ist – kann als redundant betrachtet werden.

Wenn sich ein Rotor beispielsweise in Position „A“ befindet und ein bestimmtes Verdrahtungsschema für korrekt befunden wird, könnte dasselbe Verdrahtungsschema um eine Position weitergeschaltet werden, und nun funktioniert dieses neue Verdrahtungsschema, wenn der Rotor in Position „B“ bewegt wird. Alle Schaltpläne sollten also 26 korrekte Lösungen ergeben, wenn man den Rotor durch die 26 Positionen dreht. Es scheint, dass man die Startposition des Rotors für die 3 Rotoren einfach ignorieren sollte. Aus diesem Grund haben viele andere den Enigma-Schlüsselraum ohne diesen Faktor angegeben.

Indem er die Rotoreinstellung in den Schlüsselraum einbezog, schuf Miller einen etwas größeren Schlüsselraum, was alle täglichen Nachrichten nach der Kryptoanalyse der ersten Nachricht bricht. Alle nachfolgenden Nachrichten, die dieselbe Maschineneinstellung verwenden, können dann in Echtzeit entschlüsselt werden, so wie der Feind seine eigene Nachricht entschlüsseln würde.

Wir gehen hier nicht näher auf die Enigma ein. Es gibt viele Bücher und Artikel über diese Rotormaschine und ihre Geschichte. Eine gute Zusammenfassung ihrer Konstruktion (und Konstruktionsfehler) und ein anderer Ansatz zur Berechnung ihres „relevanten“ Schlüsselraums findet sich in [11].

Millers Grundprinzip der Rotorposition gilt für alle historischen Chiffriergeräte auf Rotorbasis, einschließlich der mechanischen Geräte, wie der Hagelin M-209 (siehe Abschnitt 2.5.1 auf Seite 97). Bei dieser Maschine werden alle möglichen Stiftstellungen an jedem Rotor analysiert und in den Schlüsselraum aufgenommen. Es ist also nicht notwendig, die Drehposition des Rotors zu kennen, um eine Nachricht zu knacken. Die Stifteinstellungen sind Teil der Maschineneinstellung und für den Tag festgelegt, und die Rotoreinstellung ist Teil der Nachrichteneinstellung, die sich mit jeder Nachricht ändert. Wie bei der Enigma müssen die Rotorpositionen bekannt sein, um alle Nachrichten des Tages in Echtzeit zu entschlüsseln.[30]

[30] Moderne Analyzer für Enigma und M-209 finden Sie in CT2 in verschiedenen Vorlagen (Ciphertext-only- und Known-Plaintext-Angriffe) unter: `CT2 Vorlagen ▷ Kryptoanalyse ▷ Klassisch`

1.7.3 Zusammenfassung der Schlüsselräume historischer Verschlüsselungssysteme

Mit einem klar definierten Satz von Annahmen konnten die Schlüsselräume entsprechend berechnet werden.

Tabelle 1.1 listet 34 historische und 4 moderne Chiffriersysteme auf und zeigt die maximalen und praktischen Schlüsselräume für jedes System unter denselben Annahmen. „Historisch“ ist hier gefasst bis ca. 1960, also vor dem Aufkommen und der Nutzung von Computern. Das erste Mal wurde diese Tabelle der ICCH-Gruppe [12] von Ralph Simpson im Dezember 2022 vorgelegt. Die Schlüsselräume für einige dieser Geräte waren vorher noch nicht veröffentlicht worden, z. B. Hebern, die japanische Purple, NEMA, KL-7, Transvertex HC-9, die russische VIC und Hagelin CD-57. Für die meisten anderen historischen Chiffriergeräte waren neue Berechnungen erforderlich, um die oben aufgeführten maximalen und praktischen Annahmen zu erfüllen.

Es ist wichtig zu bedenken, dass diese Schlüsselräume immer noch kein guter alleiniger Indikator für die kryptografische Stärke der Verschlüsselungsmethode sind – Beispiele für diese Kritik sind die monoalphabetische Substitution (2^{88}), Enigma I (2^{75}) und Playfair (2^{79}). Die Verwendung einer gemeinsamen Reihe von Annahmen bei der Berechnung des Schlüsselraums wird jedoch zumindest eine gewisse Konsistenz zwischen all diesen unterschiedlichen Geräten schaffen.

Jahr	Chiffre	Maximaler Schlüsselraum		Praktischer Schlüsselraum	
600 v. Chr.	Monoalphabet. Substitution (MASC)	$4.03 \cdot 10^{26}$	2^{88}	$4.03 \cdot 10^{26}$	2^{88}
50 v. Chr.	Caesar	$2.50 \cdot 10^{1}$	2^{5}	$2.50 \cdot 10^{1}$	2^{5}
1466	Vigenère (repeating keyword – 15 char.)	$1.68 \cdot 10^{21}$	2^{71}	$1.68 \cdot 10^{21}$	2^{71}
1586	Vigenère (autokey – 314 char. message)	$2.00 \cdot 10^{444}$	2^{1476}	$2.00 \cdot 10^{444}$	2^{1476}
1854	Playfair	$6.20 \cdot 10^{23}$	2^{79}	$6.20 \cdot 10^{23}$	2^{79}
1860s	Wheatstone Cryptograph	$4.03 \cdot 10^{26}$	2^{88}	$4.03 \cdot 10^{26}$	2^{88}
1912	Lugagne Transpositeur	$1.30 \cdot 10^{532}$	2^{1768}	$1.32 \cdot 10^{13}$	2^{44}
1912	M-94 cylinder cipher	$3.45 \cdot 10^{666}$	2^{2214}	$3.88 \cdot 10^{26}$	2^{88}
1916	M-138A strip cipher	$3.69 \cdot 10^{799}$	2^{2656}	$1.95 \cdot 10^{59}$	2^{197}
1918	ADFGX	$4.19 \cdot 10^{47}$	2^{158}	$4.19 \cdot 10^{47}$	2^{158}
1918	ADFGVX	$1.01 \cdot 10^{64}$	2^{213}	$1.01 \cdot 10^{64}$	2^{213}
1922	Hebern 5-rotor	$1.27 \cdot 10^{140}$	2^{466}	$4.56 \cdot 10^{10}$	2^{35}
1924	Kryha	$2.02 \cdot 10^{53}$	2^{177}	$1.78 \cdot 10^{29}$	2^{97}
1926	Enigma Swiss K	$1.60 \cdot 10^{101}$	2^{336}	$1.85 \cdot 10^{9}$	2^{31}
1930	Lugagne Le Sphinx	$1.30 \cdot 10^{532}$	2^{1768}	$2.43 \cdot 10^{24}$	2^{81}
1931	Abwehr Enigma G	$7.17 \cdot 10^{121}$	2^{405}	$4.82 \cdot 10^{10}$	2^{35}
1932	Enigma I	$3.28 \cdot 10^{114}$	2^{380}	$4.31 \cdot 10^{22}$	2^{75}
1937	SIGABA	$1.82 \cdot 10^{285}$	2^{941}	$5.95 \cdot 10^{28}$	2^{96}
1939	Japanese Purple	$3.81 \cdot 10^{59}$	2^{198}	$1.45 \cdot 10^{31}$	2^{104}
1939	Japanese JN-25 codebook (100 words)	$1.00 \cdot 10^{12}$	2^{40}	$8.25 \cdot 10^{10}$	2^{36}
1941	Lorenz SZ40/SZ42	$1.05 \cdot 10^{170}$	2^{565}	$1.05 \cdot 10^{170}$	2^{565}
1941	SG-41 „Hitler Mill“	$4.24 \cdot 10^{51}$	2^{171}	$4.24 \cdot 10^{51}$	2^{171}
1942	M-209 pin & lug	$6.16 \cdot 10^{60}$	2^{202}	$6.02 \cdot 10^{58}$	2^{195}
1942	Enigma M4	$2.33 \cdot 10^{145}$	2^{483}	$3.13 \cdot 10^{25}$	2^{85}
1942	T-52d Geheimschreiber	$7.23 \cdot 10^{213}$	2^{710}	$8.11 \cdot 10^{23}$	2^{79}
1943	Typex Mark 22	$1.82 \cdot 10^{195}$	2^{649}	$5.51 \cdot 10^{54}$	2^{182}

... Fortsetzung nächste Seite ...

Jahr	Chiffre	Maximaler Schlüsselraum		Praktischer Schlüsselraum	
1947	NEMA	$5.99 \cdot 10^{164}$	2^{551}	$1.83 \cdot 10^{19}$	2^{64}
1952	Hagelin C-52	$1.68 \cdot 10^{117}$	2^{389}	$7.17 \cdot 10^{57}$	2^{192}
1952	Hagelin CX-52	$1.17 \cdot 10^{123}$	2^{409}	$1.10 \cdot 10^{104}$	2^{346}
1952	KL-7	$5.87 \cdot 10^{431}$	2^{1434}	$1.70 \cdot 10^{34}$	2^{114}
1950s	Transvertex HC-9	$2.96 \cdot 10^{71}$	2^{237}	$4.39 \cdot 10^{69}$	2^{231}
1953	VIC paper & pencil	$9.09 \cdot 10^{40}$	2^{136}	$1.00 \cdot 10^{27}$	2^{90}
1956	Fialka	$2.82 \cdot 10^{458}$	2^{1523}	$6.24 \cdot 10^{77}$	2^{258}
1957	Hagelin CD-57	$1.52 \cdot 10^{103}$	2^{343}	$1.49 \cdot 10^{60}$	2^{200}
1976	DES (56 bit)	$7.21 \cdot 10^{16}$	2^{56}	$7.21 \cdot 10^{16}$	2^{56}
1977	RSA-4096	$2.22 \cdot 10^{1225}$	2^{4071}	$2.22 \cdot 10^{1225}$	2^{4071}
1992	AT&T TSD 3600-E Clipper chip	$1.21 \cdot 10^{24}$	2^{80}	$1.21 \cdot 10^{24}$	2^{80}
2001	AES-256	$1.16 \cdot 10^{77}$	2^{256}	$1.16 \cdot 10^{77}$	2^{256}

Tab. 1.1: Schlüsselraum-Größen von 34 historischen und 4 „modernen“ Verschlüsselungssystemen (zusammengestellt von Ralph Simpson)

1.8 Angriffs-Typen, Sicherheits-Definitionen und n-bit-Sicherheit

Es ist nicht notwendig, diesen Abschnitt zu lesen, um die weiteren Kapitel zu verstehen – Sie können diesen Abschnitt auch einfach überspringen. Wenn Sie jedoch an den Definitionen und Konzepten interessiert sind, die in der modernen Kryptografie verwendet werden, werden diese hier mit so wenig Mathematik wie möglich erklärt. Außerdem wird die Beziehung zwischen den verschiedenen Definitionen erklärt – etwas, das in Kursen oft zu kurz kommt. Erst das Verständnis für die Unterschiede zwischen den verschiedenen Konzepten ermöglicht es den Studenten, die Ideen richtig zu verstehen, sie voneinander abzugrenzen und sie später richtig anzuwenden.

1.8.1 Sicherheits-Definitionen

Moderne Kryptografie basiert auf mathematischer Theorie und Computer-Praxis. Beim Design kryptografischer Algorithmen werden die Berechnungen so schwierig gemacht, dass sich solche Verfahren in der Praxis von einem Angreifer nur schwer brechen lassen.

Es gibt verschiedene Ansätze (Kategorien), die Sicherheit von Kryptosystemen zu definieren.

Für die formale Definition der Sicherheit eines Verschlüsselungsverfahrens werden in der Regel zwei grundlegende Ansätze verwendet: [13]

- Die erste ist die **semantische** Sicherheit, die voraussetzt, dass es für einen Angreifer undurchführbar ist, aus dem Geheimtext irgendwelche Informationen über den Klartext zu gewinnen.
- Die zweite Definition bestimmt Sicherheit als die Undurchführbarkeit, zwischen den Verschlüsselungen von zwei gegebenen Nachrichten (gleicher Länge) zu unterscheiden (**Ununterscheidbarkeit**).

In beiden Definitionen der Sicherheit wird der Begriff „undurchführbar“ und nicht „unmöglich“ verwendet. Dies ist darauf zurückzuführen, dass es **generische Angriffe** gegen fast jedes bekannte Verschlüsselungsverfahren (mit Ausnahme des One-Time-Pads) gibt. Einer dieser universellen Angriffe, nämlich ein

Brute-Force-Angriff, wird in Abschnitt 1.3.2 auf Seite 26 behandelt. Brute-Force-Angriffe können zu Time-Memory-Trade-Off (TMTO)-Angriffen erweitert werden, einer breiteren Klasse von Angriffen, die es in bestimmten Fällen ermöglichen, die Schlüssel-Wiederherstellungszeit durch Erhöhung der Speicherkosten zu verringern.

Eine andere Hauptkategorie in der Literatur definiert Sicherheit in Abhängigkeit von den Möglichkeiten des Angreifers (vgl. z. B. *Cryptography 101* [14, Kap. 1.2.2]):

- **Berechenbare, bedingte oder praktische Sicherheit**
  Ein Verschlüsselungsverfahren ist *berechenbar* sicher, wenn es (obwohl es theoretisch möglich ist, es zu brechen) selbst mit den besten bekannten Verfahren nicht gebrochen werden kann. Theoretische Fortschritte (z. B. Verbesserungen bei den Algorithmen zur Faktorisierung) und schnellere Computer erfordern, dass dies ständig angepasst wird.

  Selbst wenn man den besten bekannten Algorithmus zum Brechen benutzt, wird man so viele Ressourcen brauchen (z. B. 1 000 000 Jahre), dass das Kryptosystem sicher ist.

  Daher basiert dieses Konzept auf Annahmen über die begrenzte Rechenkraft des Angreifers und auf dem aktuellen Stand der Wissenschaft.

  Ein typisches Beispiel für ein pragmatisch sicheres Verfahren ist AES: Darauf ist kein praktisch durchführbarer Angriff bekannt. Trotzdem ist AES theoretisch „gebrochen“: Das bedeutet aber nur, dass das Brechen mit geringerem Aufwand als mit einem Brute-Force-Angriff möglich ist. Dieser Aufwand ist immer noch unrealistisch hoch. Vgl. Abschnitt 1.9 auf Seite 47.

- **Informationstheoretische oder unbedingte Sicherheit**
  Ein Verschlüsselungsverfahren wird als *unbedingt* sicher bezeichnet, wenn seine Sicherheit gewährleistet ist, völlig unabhängig davon, wie viele Ressourcen (Zeit, Speicher) der Angreifer hat. Selbst wenn der Angreifer unbegrenzt viele Ressourcen hat, kann er aus dem Chiffrat keine sinnvollen Informationen gewinnen.

  Die einzigen informationstheoretisch sicheren Verfahren, die beweisbar nicht gebrochen werden können, auch nicht mit unendlich viel Rechenkraft – sind das *One-Time-Pad* (OTP) oder Varianten davon.

Abb. 1.7 auf der nächsten Seite soll eine Idee davon vermitteln, dass es nicht möglich ist, bei einem OTP den Klartext zu bestimmen (sofern das OTP-Verfahren richtig angewendet wird und alle Schlüssel gleich wahrscheinlich sind). In dem Beispiel in der Abbildung ist als Geheimtext ein 8 Zeichen langes Wort gegeben: `11 1B 1E 18 00 04 0A 15`. Die Hexwerte entsprechen den ASCII-Werten der Buchstaben: Bspw. hat der Buchstabe `C` den Zahlenwert `67` (dezimal), was in Hexdarstellung `43` ist.

Es gibt viele sinnvolle Worte aus 8 Buchstaben und zu jedem einen richtigen Schlüssel. Ein Angreifer kann damit allein nicht bestimmen, welches der richtige Schlüssel bzw. welches das richtige Klartext-Wort ist. Mit anderen Worten: Derselbe Geheimtext kann bei unterschiedlichen Schlüsseln zu unterschiedlichen und sinnvollen Klartexten führen und so kann in diesem Fall nicht unterschieden werden, welcher Klartext der richtige ist.[31]

Da das OTP ein informationstheoretisch sicheres Verschlüsselungsverfahren ist, leitet sich seine Sicherheit allein aus der Informationstheorie ab – und ist sicher, auch wenn der Angreifer unbegrenzte Rechenkapazitäten hat. Das OTP weist allerdings einige praktische Nachteile auf (der Schlüssel darf nur einmal verwendet werden, er muss zufällig gewählt werden und mindestens so lang sein wie die zu schützende Nachricht),

[31] Vertieft wird das OTP-Verfahren in Abschnitt 2.2.4 auf Seite 89 unter dem Punkt „One-Time-Pad“. Vergleiche auch Abb. 9.24 in Abschnitt 9.3.2, wo ein entsprechendes Text-Beispiel mit SageMath erstellt wird und das XOR-Verfahren genauer erklärt wird.
[32] Bildquelle: Kostenlose Bilder von `https://pixabay.com/`

Schlüssel (hex)	Gefundener Klartext (hex)	Gefundener Klartext (txt)
52 49 47 48 54 4B 45 59	43 52 59 50 54 4F 4F 4C	CRYPTOOL
41 5A 50 57 52 45 47 54	50 41 4E 4F 52 41 4D 41	PANORAMA
54 77 7B 68 68 65 64 61	45 4C 45 50 48 41 4E 54	ELEPHANT
50 55 5B 55 4F 4A 4F 46	41 4E 45 4D 4F 4E 45 53	ANEMONES
52 53 5F 55 50 4D 45 5B	43 48 41 4D 50 49 4F 4E	CHAMPION
42 58 5B 56 41 56 43 5A	53 43 45 4E 41 52 49 4F	SCENARIO

Abb. 1.7: Illustration für die informationstheoretische Sicherheit des OTP[32]

so dass es außer in geschlossenen Umgebungen, zum Beispiel beim heißen Draht zwischen Moskau und Washington, oder beim Agentenfunk kaum eine Rolle spielt.[33]

Manchmal werden auch zwei weitere Sicherheits-Konzepte verwendet:

- **Beweisbare Sicherheit** Dies bedeutet, dass das Brechen eines Kryptosystems mindestens so schwierig ist wie die Lösung eines bestimmten schwierigen Problems, z. B. die Berechnung des diskreten Logarithmus, die diskrete Quadratwurzel-Berechnung oder die Faktorisierung sehr großer Zahlen.

  Beispiel: Aktuell wissen wir, dass RSA höchstens so schwierig ist wie die Faktorisierung, aber wir können nicht beweisen, dass es genauso schwierig ist. Deshalb hat RSA keine beweisbare Mindest-Sicherheit. Oder in anderen Worten: Wir können nicht beweisen, dass wenn das Kryptosystem RSA gebrochen ist, dass dann auch die Faktorisierung (ein schwieriges mathematisches Problem) gelöst werden kann.

  Das Rabin-Kryptosystem war das erste Kryptosystem, für das sich beweisen ließ, dass es berechenbar äquivalent zu einem harten mathematischen Problem (Ganzzahl-Faktorisierung) ist.

- **Ad-hoc-Sicherheit** Ein kryptografisches System hat diese Sicherheit, wenn es sich nicht lohnt, es zu versuchen es zu brechen, weil der Aufwand dafür teurer ist als der Wert der Daten, die man durch das Brechen erhalten würde. Z. B. weil ein Angriff nicht in einer ausreichend kurzen Zeit erfolgen kann (vgl. [15]).

  Dies kann z. B. zutreffen, wenn börsenrelevante Daten sowieso am nächsten Tag veröffentlicht werden und man für deren Brechen ein Jahr brauchen würde.

1.8.2 Angriffs-Klassifizierungen

In der Kryptografie ist ein Sicherheitsparameter ein Maß dafür, wie schwer es für einen Angreifer ist, ein kryptografisches System zu brechen. Angriffsparameter beschreiben die Bedingungen, die dem Angreifer zur Verfügung stehen.

[33] Auch wenn OTPs theoretisch nicht brechbar sind, wurden Versuche, sie anzuwenden, praktisch gebrochen: siehe Fußnote 35 in Abschnitt 2.2.4 auf Seite 87.

Angriffsdefinition Bevor wir zur Diskussion der verschiedenen Angriffsarten kommen (siehe Abschnitt 1.8.2 auf der nächsten Seite), ist es wichtig, Angriffskonzepte gegen moderne Chiffren zu klären. Wir beginnen diese Erklärung mit dem Kerckhoffs'schen Prinzip (siehe Abschnitt 1.6 auf Seite 31). Dieses Prinzip besagt, dass ein Kryptosystem auch dann sicher sein sollte, wenn dem Angreifer alle Details des Systems, mit Ausnahme des geheimen Schlüssels, bekannt sind.

Das Prinzip bringt den Begriff „sicher" ins Spiel. Um eine Definition von Sicherheit zu formulieren, verwenden wir Ideen über die **Unmöglichkeit der Unterscheidung** (infeasibility of distinguishing) – siehe Abschnitte 1.8.1 und 1.8.3 auf Seite 38 und auf Seite 43. Kurz gesagt ist ein kryptografischer Angriff ein Algorithmus, der darauf abzielt, die mangelnde Sicherheit eines gegebenen Kryptosystems zu demonstrieren.

Angriffskosten Bei der Analyse, wie schwierig es ist, einen kryptografischen Angriff durchzuführen, wird die **rechnerische Komplexität** des entsprechenden Algorithmus bewertet. Diese Komplexität ist die Menge an Ressourcen, die zur Ausführung des Algorithmus benötigt wird. Dabei werden in der Regel drei Hauptressourcen berücksichtigt: Zeit, Speicher und Daten.

- Die Zeitkomplexität des Angriffs oder einfach die Angriffszeit ist eine geschätzte Obergrenze für die Anzahl der Operationen, die erforderlich sind, um eine Chiffre erfolgreich zu knacken. Zeit ist die wichtigste zu berücksichtigende Ressource. Wenn von **rechnerischer Komplexität** ohne nähere Angaben die Rede ist, ist in der Regel die **Zeitkomplexität** gemeint.
- Die Speicherkomplexität ist der Speicherplatz, der für die Ausführung des Angriffs benötigt wird.
- Die Datenkomplexität bezieht sich auf die Menge der Daten (Klartext, Geheimtext oder beides), auf die der Angreifer Zugriff haben muss, um den Angriff durchzuführen. (Vergleiche Abschnitt 1.8.6).

Die **Angriffszeit** wird im Allgemeinen durch die Anzahl der benötigten Entschlüsselungen (oft schreibt man auch einfach „**Ver**schlüsselungen") ausgedrückt. Dies geschieht, um zu zeigen, um welchen Faktor der entsprechende Angriff schneller ist als der Brute-Force-Angriff. Wie in Abschnitt 1.3.2 erläutert, steht die Größe des Schlüsselraums in direktem Zusammenhang mit der Angriffszeit des Brute-Force-Angriffs. Um jeden Schlüssel zu testen, muss der entsprechende Entschlüsselungsalgorithmus jeweils einmal komplett durchlaufen werden. Wenn also die Länge des Schlüssels (in binärer Darstellung) n ist und alle möglichen Varianten des Schlüssels zu unterschiedlichen Geheimtexten führen, dann ist die Größe des Schlüsselraums 2^n. Das heißt, um eine Chiffre sicher zu brechen, reichen immer 2^n Entschlüsselungen aus. Dies bestimmt die Angriffszeit der erschöpfenden Suche. Die „Angriffszeit" hängt eng zusammen mit der n-bit-Sicherheit (siehe Abschnitt 1.8.6).

Bei verschiedenen Angriffen kann es vorkommen, dass nicht der Entschlüsselungsalgorithmus selbst, sondern andere Rechenoperationen ausgeführt werden müssen. In diesem Fall wird geschätzt, wie viele dieser Operationen die Zeit benötigen, die der Zeit einer Entschlüsselung entspricht. Dann wird die Gesamtzahl der für den Angriff erforderlichen Operationen durch die Anzahl der Operationen geteilt, die einer einzigen Entschlüsselung entsprechen. Daraus ergibt sich die Zeitkomplexität für den aktuellen Angriff, gemessen in Entschlüsselungen.

Sicherheitsparameter Ein kryptografischer Angriff gilt als erfolgreich, wenn er weniger Kosten verursacht als der von den Entwicklern eines Kryptosystems festgelegte Sicherheitsparameter. Ein Sicherheitsparameter gibt an, wie schwierig es für einen Angreifer ist, ein kryptografisches System zu knacken. Er wird oft in Bits ausgedrückt. Man kann beispielsweise sagen, dass ein bestimmtes Verfahren n-bit-Sicherheit bietet, wenn die Angriffszeit $O(2^n)$ Verschlüsselungen benötigt. Die $O()$-Notation (auch Big-O-Notation oder Bachmann-Landau-Notation oder asymptotische Notation genannt) beschreibt eine obere Schranke für die Zeitkomplexität eines Algorithmus. Im Wesentlichen gibt sie das Worst-Case-Szenario an, wie die Laufzeit mit zunehmender Eingabegröße wächst. Vergleiche Abschnitt 1.8.6.

Im Zusammenhang mit symmetrischen Verschlüsselungsverfahren entspricht der Sicherheitsparameter in der Regel der Schlüsselgröße. Dies liegt daran, dass der Brute-Force-Angriff die Mindestgrenze für den Sicherheitsparameter festlegt. Der Sicherheitsparameter kann jedoch niedriger als die Schlüsselgröße sein, wenn ein Angriff, der schneller ist als der Brute-Force-Angriff, bereits bei der Entwicklung einer Chiffre bekannt ist. Dies ist eine häufige Situation bei Public-Key-Verschlüsselungsverfahren.

Ziele In der modernen Kryptologie gibt es verschiedene Klassifizierungen von kryptoanalytischen Angriffen. Nach dem Ziel des Angreifers unterscheidet man zwischen **Key-Recovery-** und **Distinguishing-Angriffen**. Die Key-Recovery-Angriffe zielen darauf ab, an den eigentlichen Verschlüsselungs- oder Entschlüsselungsschlüssel zu gelangen, wodurch die Sicherheit des kryptografischen Systems vollständig gebrochen wird. Im Gegensatz dazu konzentrieren sich Unterscheidungs-Angriffe auf die Fähigkeit, verschlüsselte Daten von echten Zufallsdaten zu unterscheiden, indem sie Abweichungen oder Schwachstellen in der Chiffre entdecken, die zu Schlüsselwiederherstellungs-Angriffen führen können.

Einzel-/Mehrfachschlüssel Kryptoanalytische Angriffe unterscheiden sich auch durch die Fähigkeit des Angreifers, die Durchführung einer Chiffre zu beobachten, wenn sie unterschiedliche Schlüssel benutzt. Bei Angriffen mit einem Schlüssel (**single-key attacks**) wird davon ausgegangen, dass der Angreifer Zugriff auf die mit demselben Schlüssel verschlüsselten Geheimtexte hat. Bei Angriffen mit unterschiedlichen Schlüsseln (**variable-key attacks**) wird davon ausgegangen, dass der Angreifer Zugriff auf Geheimtexte hat, die mit mehreren unbekannten Schlüsseln verschlüsselt sind. Dies spiegelt häufig reale Situationen wider, in denen der Benutzer einer Chiffre den Schlüssel nach einer bestimmten Anzahl von Verschlüsselungen ändern muss. Wenn ein Angreifer Zugang zu mehreren entsprechenden Geheimtexten erhält, kann er diese Informationen als Vorteil nutzen, um zu versuchen, eine der entsprechenden Verschlüsselungen zu knacken. **Related-Key-Angriffe** gehen davon aus, dass ein Angreifer eine bestimmte mathematische Beziehung zwischen verschiedenen geheimen Schlüsseln kennt und die entsprechenden Geheimtexte abgreifen kann. Obwohl ein solches Szenario auf den ersten Blick als zu unrealistisch erscheint, wurden in der realen Welt mehrere Kryptosysteme mithilfe von Related-Key-Angriffen geknackt, z. B. [16].

Zugriff auf Daten (Geheimtext-Klartext-Paare) Die kryptografischen Angriffe können in die folgenden vier Hauptkategorien unterteilt werden, je nachdem welchen Zugriff der Angreifer auf den Geheimtext und den Klartext hat (jeweils unter der Annahme, dass der Schlüssel unbekannt ist):

- Ciphertext-only-Angriffe (COA) setzen voraus, dass nur auf die Geheimtexte zugegriffen wird, ohne die zugehörigen Klartexte zu kennen.
- Bei Angriffen mit bekanntem Klartext (known-plaintext attacks, KPA) werden Paare von bekanntem Klartext und dem dazugehörigen Geheimtext verwendet, um den geheimen Schlüssel zu ermitteln.
- Chosen-Plaintext-Angriffe (CPA) erlauben es dem Angreifer, selbst beliebige Klartexte zu wählen und deren Geheimtexte zu erhalten, was mehr Flexibilität bei der Analyse des Verschlüsselungsalgorithmus bietet.
- Chosen-Ciphertext-Angriffe (CCA) ermöglichen es dem Angreifer, beliebige Geheimtexte zu wählen und deren Klartext zu erhalten, wobei er die Möglichkeit hat, die Geheimtexte während der Entschlüsselung zu manipulieren.

Darüber hinaus unterscheiden sich die Angriffe anhand spezifischer mathematischer Methoden, wie z. B. differentielle Kryptoanalyse (Analyse, wie sich Unterschiede in den Eingaben der Chiffren auf Unterschiede bei den Ausgaben auswirken), lineare Kryptoanalyse (Ausnutzung linearer Beziehungen im Verschlüsselungsprozess), Meet-in-the-Middle-, Biclique-, Integral-, Boomerang-, Cube- und andere Angriffe. Alle diese Methoden sind sehr speziell, so dass wir für eine umfassende Erläuterung auf die angegebenen Referenzen in den Tabellen 1.4 und 1.5 verweisen.

1.8.3 Sicherheitsdefinitionen mittels Ununterscheidbarkeit

Die diskutierten Angriffsarten CPA und CCA stehen in direkter Beziehung zu den kryptografischen Sicherheitsdefinitionen IND-CPA, IND-CCA1 und IND-CCA2. Diese Definitionen spielen eine entscheidende Rolle im Bereich der „beweisbaren Sicherheit" der Kryptografie. In diesem – sehr theoretischen – Bereich geht es darum, die Sicherheit von kryptografischen Verfahren mathematisch zu beweisen. Dies wird erreicht, indem gezeigt wird, dass das Brechen eines bestimmten Verfahrens die Lösung eines Problems erfordert, das allgemein als schwierig bekannt ist.

IND-CPA (Indistinguishability under chosen-plaintext attack): Bei diesem Modell kann ein Angreifer beliebige Klartexte wählen und sich die entsprechenden Geheimtexte so oft wie nötig vom Verschlüsselungsorakel erzeugen lassen. Dann wählt der Angreifer zwei unterschiedliche Nachrichten (Klartexte) aus und sendet sie an das Verschlüsselungsorakel, das einen Geheimtext von nur einer dieser Nachrichten, den sogenannten Challenge-Geheimtext, zurückgibt.

Danach kann der Angreifer eine beliebige Anzahl von zusätzlichen Berechnungen und Verschlüsselungen durchführen. Ein Verschlüsselungsverfahren gilt als sicher, wenn der Angreifer nicht erraten kann, auf welchen Klartext sich die Geheimtext-Herausforderung bezieht, und zwar mit einer Wahrscheinlichkeit von mehr als $|1/2+\eta|$, wobei η vernachlässigbar ist. Es ist klar, dass der Angreifer nicht die gleichen Nachrichten für die Herausforderung wählen kann, für die er die Geheimtexte vom Orakel erhält.

Diese Sicherheitsdefinition kann sowohl auf symmetrische als auch auf asymmetrische Verschlüsselungsverfahren angewendet werden, obwohl diese formal unterschiedlich beschrieben werden.[17] Bei *deterministischen* asymmetrischen Verschlüsselungsverfahren hat ein Angreifer jedoch Zugang zum öffentlichen Schlüssel, was bedeutet, dass er leicht unterscheiden kann, welcher Geheimtext durch welche Nachricht erzeugt wurde, indem er die Nachrichten selbst verschlüsselt. Daher wird die Definition nur auf *probabilistische* asymmetrische Verschlüsselungsverfahren angewandt, bei denen Zufall im Verschlüsselungsprozess verwendet wird. Dies bedeutet, dass wenn dieselbe Nachricht mehrmals mit einem probabilistischen Verschlüsselungsverfahren verschlüsselt wird, unterschiedliche Geheimtexte erzeugt werden.

IND-CCA1 (Indistinguishability under chosen-ciphertext attack, auch bekannt als non-adaptive or lunchtime attack): Diese Sicherheitsdefinition setzt ein höheres Sicherheitsniveau als IND-CPA voraus. In diesem Modell kann ein Angreifer wiederum die Klartexte wählen und ihre entsprechenden Geheimtexte vom Orakel erhalten, aber auch beliebige Geheimtexte entschlüsseln lassen, um die entsprechenden Klartexte zu erhalten. Das weitere Vorgehen ist ähnlich wie im IND-CPA-Fall. Im Falle von IND-CCA1 ist jedoch das Entschlüsselungsorakel nicht mehr verfügbar, nachdem der Angreifer die Geheimtext-Challenge erhalten hat.

IND-CCA2 (Indistinguishability under adaptive chosen-ciphertext attack): Dies ist die stärkste Definition, die den höchsten Grad an Sicherheit bietet. Sie erlaubt es dem Angreifer, mit dem Entschlüsselungsorakel weiter zu interagieren, auch nachdem er den Challenge-Geheimtext erhalten hat.

Bei der Betrachtung moderner kryptografischer Verschlüsselungsprimitive ist die Auswahl des besten Angriffs keine einfache Aufgabe. In Tabelle 1.5 auf Seite 51 haben wir die Informationen übersichtlich gehalten und Angriffe zur Wiederherstellung von Schlüsseln priorisiert, die nur minimale Berechnungen erfordern und schneller sind als Brute-Force-Angriffe, die eine universelle Angriffsmethode gegen jeden Verschlüsselungsalgorithmus darstellen. Durch diese Priorisierung haben wir andere Komplexitäten wie Daten- und Speicherkosten (z. B. die Anzahl der erforderlichen Klartext-Geheimtext-Paare) außer Acht gelassen.

In unserer Tabelle liegt der Fokus in der Regel auf Szenarien mit nur einem Schlüssel, mit zwei Ausnahmen: der Related-Key-Angriff gegen die Kasumi-Chiffre und der Variable-Key-Angriff gegen RC4. Wenn die vollständige Chiffre nicht kompromittiert ist, versuchen wir, Angriffe auszuwählen, die so viele Runden

wie möglich brechen. Bei MUGI und Enocoro beziehen wir uns nur auf Unterscheidungsangriffe, da uns keine veröffentlichten Angriffe zur Schlüsselwiederherstellung bekannt sind.

1.8.4 Shannon und perfekte Sicherheit

Vorläufer der Konzepte der Sicherheitsdefinitionen mittels Ununterscheidbarkeit (siehe Abschnitt 1.8.3) sind die Forschungen von Claude Shannon[34], veröffentlicht in [18], und die Definition der perfekten Geheimhaltung. Sein Paper [18] behandelt die Kryptografie aus der Sicht der Informationstheorie und ist eine der grundlegenden Abhandlungen der modernen Kryptografie. Es ist auch ein Beweis dafür, dass alle theoretisch unknackbaren Chiffren die gleichen Anforderungen wie das One-Time-Pad erfüllen müssen.

Die wichtigste Erkenntnis von Shannon war, dass in einem sicheren Verschlüsselungsverfahren der Geheimtext keine zusätzlichen Informationen über den Klartext preisgeben sollte. Wenn es also zum Beispiel für Eve a priori möglich war, den Klartext mit einer Wahrscheinlichkeit von $1/t$ zu erraten (z. B., weil es nur t Möglichkeiten dafür gab), dann sollte sie nicht in der Lage sein, ihn mit einer höheren Wahrscheinlichkeit zu erraten, nachdem sie den Geheimtext gesehen hat. Dies wird wie folgt formalisiert:

Definition 1.8.1. ***Perfekte Geheimhaltung (perfect secrecy):*** *Ein Kryptosystem (P, C, K, E, D) ist vollkommen geheim, wenn für jede Menge $P \subseteq \{0,1\}^\ell$ von Klartexten und für jede von Eve verwendete Strategie gilt: Wenn wir zufällig ein $m \in P$ und ein $k \in \{0,1\}^n$ (die Menge der Schlüssel) wählen, dann beträgt die Wahrscheinlichkeit, dass Eve m richtig errät, nachdem sie $c = E_k(m)$ gesehen hat, höchstens $1/|P|$.*

Insbesondere, wenn wir entweder „Ja“ oder „Nein“ mit einer Wahrscheinlichkeit von $1/2$ verschlüsseln, dann wird Eve nicht in der Lage sein, mit einer Wahrscheinlichkeit von mehr als der Hälfte zu erraten, welches von beiden es ist. Das ist der eigentliche Kern des Problems:

Ein Kryptosystem (P, C, K, E, D) ist perfekt geheim, wenn und nur wenn für jedes Paar verschiedener Klartexte $\{m_0, m_1\} \subseteq \{0,1\}^\ell$ und jede von Eve verwendete Strategie, wenn wir zufällig $b \in \{0,1\}$ und einen zufälligen Schlüssel $k \in \{0,1\}^n$ wählen, dann ist die Wahrscheinlichkeit, dass Eve m_b errät, nachdem sie $E_k(m_b)$ gesehen hat, höchstens $1/2$.

Eine andere äquivalente Bedingung für perfekte Geheimhaltung ist die folgende: (P, C, K, E, D) ist vollkommen geheim, wenn für alle Klartexte $m, m' \in \{0,1\}^\ell$ die beiden Zufallsvariablen $\{E_k(m)\}$ und $\{E_{k'}(m')\}$ (für zufällig und gleichmäßig gewählte Schlüssel k und k') genau die gleiche Verteilung haben.

Wie wir in Definition 1.8.1 gesehen haben, bedeutet perfekte Geheimhaltung, dass Eve durch Beobachtung des Geheimtextes keine Informationen über den Klartext erhalten kann. Eine andere Möglichkeit, diese Idee genau zu definieren, ist die Verwendung der „bedingten“ Wahrscheinlichkeit, ein Begriff der Wahrscheinlichkeitsverteilungen (die Definition ist angepasst aus [3, p 66]). Siehe die folgende Definition 1.8.2.

Definition 1.8.2. ***Perfekte Geheimhaltung (perfect secrecy) (definiert mit Wahrscheinlichkeiten):*** *Ein Kryptosystem hat eine perfekte Geheimhaltung, wenn $\Pr[m|c] = \Pr[m]$ für alle $m \in P$, $c \in C$. Das heißt, die a posteriori-Wahrscheinlichkeit, dass der Klartext m ist, wenn der Geheimtext c beobachtet wird, ist identisch mit der a priori-Wahrscheinlichkeit, dass der Klartext m ist.*

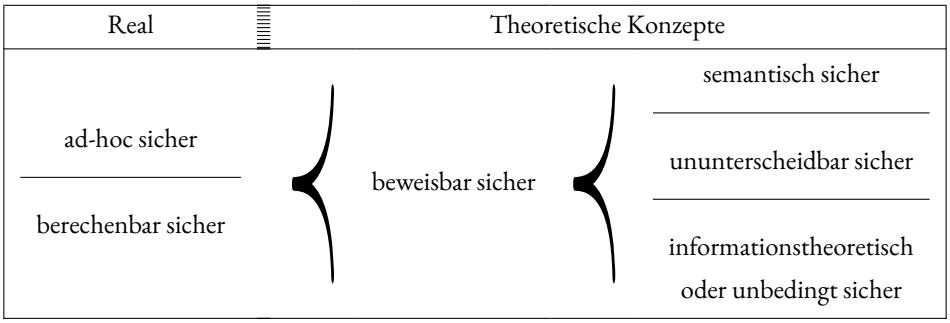
Perfekte Geheimhaltung bedeutet, dass der Geheimtext keine Informationen über den Inhalt des Klartextes vermittelt. Das bedeutet, dass der Geheimtext, egal wie viel man davon hat, nichts über den Inhalt des Klartextes und des Schlüssels vermittelt. In Bezug auf Wahrscheinlichkeiten bedeutet dies, dass die Wahrscheinlichkeitsverteilung der möglichen Klartexte unabhängig vom Geheimtext ist. In Bezug auf die

[34] Claude Elwood Shannon, amerikanischer Mathematiker und Elektrotechniker, 30.4.1916–24.2.2001.

Möglichkeiten des Angreifers ist eine solche Chiffre informationstheoretisch oder bedingungslos sicher, unabhängig davon, wie viele Ressourcen der Angreifer hat.

1.8.5 Tabellarischer Vergleich verschiedener Sicherheits-Definitionen

Tabelle 1.2 zeigt einen Vergleich der verschiedenen Kategorien für die Sicherheit von Kryptosystemen. In der Realität sind die meisten kryptografischen Verfahren, die wir heute verwenden, rechnerisch sicher und weisen eine gute Ad-hoc-Sicherheit auf. Wir könnten eine Reduktion auf ein mathematisches Problem finden, das als schwierig vermutet wird, d. h. unser Problem ist mindestens so schwierig zu lösen wie das schwierige Problem (dann ist es nachweislich oder beweisbar sicher). Wenn also das Problem tatsächlich schwer zu lösen ist, muss auch das Knacken unserer Verschlüsselung schwer sein. In der realen Welt sind wir jedoch weit von semantisch oder ununterscheidbar sicheren Systemen entfernt. Für das One-Time-Pad (OTP) bedeutet ununterscheidbar sicher, dass der Schlüssel mindestens so lang sein muss wie der Klartext. Das OTP und einige Varianten sind das einzige Kryptosystem mit theoretisch perfekter Geheimhaltung.



Tab. 1.2: Kategorien der Sicherheit von Kryptosystemen (geordnet nach Stärke)

1.8.6 Sicherheitslevel und n-bit-Sicherheit

In Abschnitt 1.3 haben wir bereits gesehen, dass Brute-Force-Angriffe eine erschöpfende Suche durch den gesamten Schlüsselraum durchführen. In Abschnitt 1.7 wurden für historische Chiffriermaschinen theoretische und praktische Schlüsselräume unterschieden. Für moderne symmetrische Chiffren ist der (maximale oder theoretische) Schlüsselraum (fast) derselbe wie der praktische Schlüsselraum. Um die Sicherheit einer kryptografischen Primitive (Grundfunktion) – wie eines Verschlüsselungs-Algorithmus oder einer Hash-Funktion – zu messen, wird häufig der Begriff „n-bit security“ verwendet, der sich auf den Schlüsselraum bezieht.

Laut NIST ist das Sicherheitslevel (oder die „Sicherheits-Stärke“) ein Maß für die Stärke, die eine kryptografische Primitive erreicht. Das Sicherheitslevel wird in der Regel als Anzahl von „Sicherheitsbits“ ausgedrückt [19, Glossar, Seite 17], wobei „n-bit-Sicherheit“ bedeutet, dass der Angreifer 2^n Operationen durchführen müsste, um es sicher zu knacken. Dieser n-Bit-Sicherheitswert ist leicht zu vergleichen und hat den allgemeinen Brute-Force-Angriff im Sinn. [20] Wie in Abschnitt 1.8 erörtert, sind auch andere Methoden vorgeschlagen worden, die die Kosten für einen Angreifer genauer modellieren. [21]

Die folgenden Absätze sind fast vollständig dem Wikipedia-Artikel Security Level und den referenzierten NIST-Dokumenten entnommen, da sie dieses Thema sehr gut aufbereitet haben. Weitere Vertiefungen finden sich in [22].

Das Maß n-bit-Sicherheit ermöglicht einen bequemen Vergleich zwischen Algorithmen und ist nützlich, wenn mehrere Primitive in einem hybriden System kombiniert werden, um zu wissen, welche Parameter eingestellt werden müssen, um kein eindeutiges schwächstes Glied zu haben. AES-128 (Schlüsselgröße 128 bit) ist beispielsweise so konzipiert, dass es ein Sicherheitslevel von 128 bit bietet, was in etwa einem RSA mit 3072-bit-Schlüssel entspricht (siehe Tabelle 1.3).

In diesem Zusammenhang bedeuten „Sicherheits-Anspruch“ (englisch security claim) oder „Sicherheits-Ziellevel“ (englisch target security level) das Sicherheitsniveau, das eine Primitive ursprünglich durch ihre Designer erreichen sollte. Werden Angriffe gefunden, die geringere Kosten verursachen als der Sicherheits-Anspruch, wird die Primitive als theoretisch gebrochen betrachtet. [23][24]

Symmetrische Algorithmen haben in der Regel einen streng definierten Sicherheits-Anspruch. Bei symmetrischen Chiffren ist dieser in der Regel gleich der Schlüsselgröße der Chiffre – was der Komplexität eines Brute-Force-Angriffs entspricht. [24][25] Kryptografische Hash-Funktionen mit einer Ausgabelänge von n bits haben in der Regel einen Sicherheitslevel von $n/2$ gegen Kollisions-Angriffe und einen von n gegen Preimage-Angriffe. Dies liegt daran, dass der allgemeine Geburtstagsangriff Kollisionen immer in $2^{n/2}$ Schritten finden kann. [15, Seite 336] SHA-256 bietet beispielsweise eine 128-bit-Kollisions-Resistenz und eine 256-bit-Preimage-Resistenz.

Der Entwurf der meisten asymmetrischen Algorithmen (Public-Key-Kryptografie) beruht auf bekannten mathematischen Problemen, die in einer Richtung effizient zu berechnen sind, aber vom Angreifer nur ineffizient umgekehrt werden können. Angriffe auf aktuelle Public-Key-Systeme sind jedoch immer schneller als die Brute-Force-Suche des Schlüsselraums. Ihr Sicherheitslevel ist nicht zur Entwurfszeit festgelegt, sondern stellt eine rechnerische Annahme zur Problemhärte dar, die so angepasst wird, dass sie dem besten derzeit bekannten Angriff entspricht. [25] Experten berechnen also den entsprechenden Wert der n-bit-Sicherheit.

Zur Einschätzung des Sicherheitslevels asymmetrischer Algorithmen wurden verschiedene Vorschläge veröffentlicht, die sich aufgrund unterschiedlicher Methoden leicht unterscheiden (vgl. z. B. [26, Tabelle 3.1 auf Seite 22]):

- Damit das RSA-Kryptosystem ein 128-bit-Sicherheitslevel erreicht, empfehlen NIST und ENISA die Verwendung von 3072-bit-RSA-Schlüsseln [19][26, Tabelle 3.5 auf Seite 32 und Tabelle 3.6 auf Seite 37] und IETF empfiehlt 3253 bit. [27][28] Die Umrechnung von der RSA-Schlüssellänge auf eine Schätzung des Sicherheitslevels basiert auf der Komplexität des GNFS. [29, §7.5]
- Der Diffie-Hellman-Schlüsselaustausch und DSA ähneln RSA in Bezug auf die Umwandlung der Schlüssellänge in eine Schätzung des Sicherheitslevels. [29, §7.5]
- Die Kryptografie mit elliptischen Kurven nutzt kürzere Schlüssel, daher lauten die Empfehlungen für 128-bit-Sicherheitslevel 256-383 bit (NIST), 256 bit (ENISA) und 242 bit (IETF). Die Umrechnung von der Schlüsselgröße f in das Sicherheitslevel ist ungefähr $f/2$: Das liegt daran, dass die Methode zum Brechen des Elliptic Curve Discrete Logarithm Problems (ECDLP), die Rho-Methode, mit 0,886 sqrt(2^f) Additionen auskommt. [30]

Tabelle 1.3 enthält Beispiele für typische Sicherheitslevel für verschiedene Algorithmen, wie sie in der sehr guten NIST-Veröffentlichung [19, Tabelle 2 auf Seite 54] zu finden sind. Vgl. auch die bereits 2013 erstellte Tabelle 12.2 auf Seite 731. Im Rahmen der NIST-Empfehlungen wurde „DES“ (DEA) bereits 2003 als veraltet eingestuft, und „3TDES“ wurde 2023 als veraltet bewertet (TDEA = Triple Data Encryption Algorithm). Für Finite Field/DLP ergeben sich die Parameter L, N aus L = Bitlänge von p^N. Für IFC und ECC warnt das NIST, dass ihre Schätzungen der Sicherheitsstärke erheblich beeinträchtigt werden, wenn Quantencomputer in der Praxis zum Einsatz kommen. Vergleiche auch die Sicherheitsempfehlungen in Abb. 13.2 auf Seite 744.

Sicherheits-Stärke	Symmetrische Chiffren	Endliche Körper Diskreter Logarithmus (DSA, DH, MQV)	Ganzzahl-Faktorisierung (RSA)	Elliptische Kurven (ECC) (ECDSA, EdDSA, ECDH, ECMQV)
80	2TDES	$L = 1024, N = 160$	$k = 1024$	$160 \leq f \leq 223$
112	3TDES	$L = 2048, N = 224$	$k = 2048$	$224 \leq f \leq 255$
128	AES-128	$L = 3072, N = 256$	$k = 3072$	$256 \leq f \leq 383$
192	AES-192	$L = 7680, N = 384$	$k = 7680$	$384 \leq f \leq 511$
256	AES-256	$L = 15360, N = 511$	$k = 15360$	$f \geq 512$

Tab. 1.3: Vergleichbare Sicherheitslevel für symmetrische und asymmetrische Chiffren

Bedeutung von „gebrochen“ Eine kryptografische Primitive gilt als (*theoretisch*) gebrochen, wenn sich herausstellt, dass ein Angriff weniger als das behauptete Sicherheitslevel aufweist. Allerdings sind nicht alle derartigen Angriffe praktikabel: Die meisten Angriffe, die derzeit praktisch demonstriert werden, benötigen weniger als 2^{40} Operationen, was auf einem durchschnittlichen PC ein paar Stunden ausmacht.

Aumasson zieht die Grenze zwischen praktischen und unpraktischen Angriffen bei 2^{80} Operationen. Er schlägt eine neue Terminologie vor: [31]

- Zu einer **„gebrochenen“** Primitive gibt es einen Angriff, der $\leq 2^{80}$ Operationen benötigt. Ein Angriff kann plausibel durchgeführt werden.
- Zu einer **„verwundeten“** Primitive gibt es einen Angriff, der zwischen 2^{80} und etwa 2^{100} Operationen benötigt. Ein Angriff ist im Moment nicht möglich, aber zukünftige Verbesserungen werden ihn wahrscheinlich möglich machen.
- Der beste Angriff auf eine **„angegriffene“** Primitive ist billiger als der Sicherheits-Anspruch, aber viel teurer als 2^{100}. Ein solcher Angriff ist zu weit davon entfernt, praktikabel zu sein.
- Zu einer **„analysierten“** Primitive gibt es keinen Angriff, der billiger ist als ihr Sicherheits-Anspruch.

Transparenz. Das ist das Höchste, was man sich in einer technologisch hoch entwickelten Gesellschaft erhoffen kann ... sonst wird man einfach nur manipuliert.

Zitat 2: Daniel Suarez[35]

1.9 Beste bekannte Angriffe auf konkrete Verschlüsselungsverfahren

Die Tabellen 1.4 und 1.5 enthalten für bekannte klassische und moderne Chiffren die besten heute bekannten Angriffe. Für moderne Verfahren wird in Tabelle 1.5 auf Seite 51 auch der Aufwand (Schrittzahl oder „attack time“) angegeben. Unseres Wissens ist das das erste Mal, dass so vollständige Tabellen erstellt wurden.

[35] Daniel Suarez, *Darknet*, rororo, (c) 2011, Kapitel 5, *Einsichten*, S. 69, Price.

1.9.1 Beste bekannte Angriffe gegen klassische Chiffren

Chiffren	Angriffs-Voraussetzungen	(Beste) Kryptoanalyse-Methoden	Verweise
Substitutions-Chiffren			
Caesar	PCO	Brute force, frequency analysis	[32]
Monoalphabetic substitution	PCO	Hill climbing, frequency analysis	[32]
Homophonic substitution	PCO	Hill climbing / simulated annealing	[33]
Nomenclatures	PCO	Manual (deduced by context; or nomenclature available)	[34, 35]
Polyalphabetic substitution	PCO	Hill climbing / simulated annealing / (Friedman + Kasiski)	[32]
Playfair	PCO; crib	Simulated annealing	[36, 37]
Code books	PCO; crib/KP	Manual (deduced by context; availability of „similar“ code book)	[38]
Chao Cipher	PCO	Hill climbing / simulated annealing	[39]
Transpositions-Chiffren			
Scytale	PCO	Brute force	[32]
Columnar transposition	PCO	Brute force (short keys) / hill climbing / simulated annealing	[40]
Double columnar transposition	PCO	Hill climbing / simulated annealing; IDP attack	[41]
Zusammengesetzt			
ADFGVX	PCO	D&C + hill climbing / simulated annealing	[42]
Maschinen			
Enigma	PCO, crib	D&C; hill climbing / simulated annealing; Turing Bombe	[43, 44, 45]
Typex	PCO, crib	D&C; hill climbing / simulated annealing; (Turing Bombe)	[43, 44, 45]
SZ42	PCO, crib	„Testery methods“ + hill climbing	[46]
M209	PCO, crib	Simulated annealing / hill climbing	[47, 48]
SIGABA	KP	Meet in the middle; hill climbing / simulated annealing	[49, 50]

Tab. 1.4: Die bekanntesten Angriffe gegen 17 historische Chiffren (PCO = pure ciphertext-only; KP = known-plaintext; D&C = Divide&Conquer)

Die historischen Chiffren in Tabelle 1.4 repräsentieren verschiedene Perioden in der Geschichte der Kryptografie und reichen von einfachen Caesar-Chiffren bis zu komplexeren maschinenbasierten Systemen wie Enigma. Die Auswahl der Chiffren basiert auf deren historischer Bedeutung. Die in der Tabelle aufge-

führten Angriffsarten und -methoden sind die derzeit bekanntesten computergestützten Methoden zum Angriff auf diese Chiffren. Alle Handchiffren sind anfällig für Simulated Annealing und Hill Climbing. Zusammengesetzte Chiffren, in unserer Tabelle ADFGVX, benötigen anspruchsvollere Methoden. Bei ADFGVX kann ein Divide-and-Conquer-Angriff verwendet werden, um Substitution und Transposition unabhängig voneinander zu brechen. Ebenfalls erwähnenswert ist die SIGABA, da sie mit einem Meet-in-the-Middle-Angriff angegriffen werden kann. Darüber hinaus können alle gezeigten Handchiffren (Substitution, Transposition und zusammengesetzte Chiffren) heute in einem reinen Ciphertext-only-Szenario angegriffen werden. Eine Ausnahme bilden Nomenklatur-Chiffren, da die Nomenklatur-Elemente (Codewörter) oft nur entschlüsselt werden können, wenn man entweder den Originalschlüssel oder genügend Kontext hat, um sie abzuleiten. Außerdem sind die Chancen für einen erfolgreichen Angriff auf Chiffriermaschinen wie die Enigma und Typex größer, wenn ein Crib (ein teilweise bekannter Klartext) verfügbar ist. Nur Angriffe auf SIGABA erfordern immer noch den vollständigen Klartext, um erfolgreich zu sein.

1.9.2 Beste bekannte Angriffe gegen moderne Chiffren

Tabelle 1.5 zeigt eine Auswahl moderner Chiffren und die besten Angriffe gegen sie. Die Tabelle enthält **historisch bedeutsame** Chiffren wie DES und FEAL, **ISO-Standards** wie AES, Camellia und SNOW 2, **nationale Standards** wie GOST und SM4 sowie Chiffren, die aktiv in **industriellen Lösungen** verwendet wurden, wie KeeLoq und A5.1. Die Namen der Chiffren umfassen in der Regel eine Familie von Verschlüsselungsmethoden und beziehen sich nicht auf einen einzelnen Algorithmus. Diese Algorithmen unterscheiden sich in der Regel durch die Größe des verwendeten Schlüssels und, im Falle von Blockchiffren, durch die Größe des Datenblocks. Die besten Angriffe gegen verschiedene Versionen einer Chiffre können unterschiedlich sein. Der Kürze halber stellen wir je ein einziges Beispiel aus jeder Chiffre-Familie vor und präsentieren den erfolgreichsten Angriff gegen dieses Beispiel.

In der rechten Spalte von Tabelle 1.5 auf Seite 51 wird der Begriff „Angriffszeit“ verwendet. „Zeit“ ist ein gängiger Begriff in der modernen Kryptografie. Um zu verstehen, was die Angriffszeit – als Maß für die Widerstandsfähigkeit einer Chiffre – bedeutet, siehe Abschnitt 1.8, in dem Angriffskosten und verschiedene Angriffsarten vorgestellt werden.

Für symmetrische Verfahren ist der aus der Schlüssellänge abgeleitete Schlüsselraum eine wichtige Kennzahl (siehe Abschnitt 1.7 auf Seite 31). Aus dem Schlüsselraum wird der Aufwand für einen Brute-Force-Angriff (BF-Angriff) hergeleitet, der Maximalaufwand, den ein Angreifer haben kann.

Für AES-128 gilt beispielsweise Folgendes (siehe Tabelle 1.5): Die Schlüssellänge ist 128 bit. Der Schlüsselraum beträgt 2^{128} und damit auch die theoretische attack time. Der beste bekannte Angriff (Biclique-Angriff) verringert diesen Maximalaufwand auf $2^{126,1}$ Schritte. Dieser Unterschied von rund 2 im Exponenten bedeutet, dass der Angriff im Schnitt etwa um den Faktor 4 schneller ist als ein BF-Angriff. Damit zeigt er die prinzipielle Angreifbarkeit von AES, ist aber für die praktische Sicherheit überhaupt nicht relevant.

Chiffren	Angriffs-Arten	(Beste) Kryptoanalyse-Verfahren	Angriffs-Zeit
Block-Chiffren			
DES	Single key. KPA. Full	Linear [51, 52]	2^{43}
3DES (TDEA). 3-key version [53]	Single key. KPA. Full	Meet-in-the-middle [53]	2^{112}
AES-128 (Rijndael) [54]	Single key. CCA. Full	Biclique [55]	$2^{126.1}$

...Fortsetzung nächste Seite ...

Chiffren	Angriffs-Arten	(Beste) Kryptoanalyse-Verfahren	Angriffs-Zeit
Camellia-128 [56]	Single key. CPA. 11/18 rounds	Truncated differential [57]	$2^{121.3}$
MISTY1 [58]	Single key. CPA. Full	Integral [59, 60]	$2^{107.9}$
KASUMI [61]	Related-key. CCA. Full	Boomerang [16]	2^{32}
HIGHT [62]	Single key. CCA. Full	Biclique [63]	$2^{126.4}$
CAST-128 [64]	Single key. CPA. 9/16 rounds	Differential [65]	2^{73}
SEED-128 [66]	Single key. CPA. 8/16 rounds	Differential [67]	2^{122}
PRESENT [68]	Single key. CPA. 26/31 rounds	Truncated differential [69]	2^{70}
CLEFIA-128 [70]	Single key. CPA. 14/18 rounds	Truncated differential [57]	2^{108}
LEA-128 [71]	Single key. CPA. 13/24 rounds	Differential [72]	2^{127}
SM4 [73]	Single key. KPA. 24/32 rounds	Linear [74]	$2^{126.6}$
GOST 28147-89 [75] (Magma)	Single key. CPA. Full	Guess then truncated differential [76]	2^{179}
GOST R 34.12-2015 (Kuznechik) [77]	Single key. CCA. 5/10 rounds	Meet-in-the-middle [78]	2^{140}
KeeLoq [79]	Single key. KPA. Full	Slide and meet-in-the-middle [80]	$2^{44.5}$
Simon64/128 [81]	Single key. KPA. 31/44 rounds	Multidimensional linear [82]	2^{120}
Speck64/128 [81]	Single key. CPA. 20/27 rounds	Differential [83]	$2^{93.56}$
FEAL-32 [84]	Single key. CPA. 31/32 rounds	Differential [85]	2^{63}
Twofish-128 [86]	Single key. CPA. 7/16 rounds	Saturation [87]	2^{126}
Strom-Chiffren			
RC4	Variable-key. Plaintext recovery. COA	Statistical [88]	2^{31}
A5/1 [89]	Single key. KPA. Full	Time-memory-data trade-off [90]	2^{24}
A5/2 [91]	Single key. KPA. Full	Time-memory-data trade-off [90]	2^{16}
Chacha [92]	Single key. KPA. Chosen IV. 7/20 rounds	Differential [93]	2^{255}
Salsa20 [94]	Single key. KPA. Chosen IV. 8/20 rounds	Differential [93]	2^{255}
Crypto-1 [95]	Single key. KPA. Full	Algebraic [96]	2^{32}
Grain-128 [97]	Single key. KPA. Chosen IV. Full	Dynamic cube attack [98]	2^{74}
Trivium [99]	Single key. KPA. Chosen IV. 799/1152 rounds	Dynamic cube attack [100], Siehe auch Anmerkung 1	2^{62}
Rabbit [101]	Unbekannt	Siehe auch Anmerkung 2	
Enocoro 128v2 [102]	Distinguishing. KPA. Chosen IV. 22/96 rounds	Higher order differential [103]	2^{16}

...Fortsetzung nächste Seite ...

Chiffren	Angriffs-Arten	(Beste) Kryptoanalyse-Verfahren	Angriffs-Zeit
SNOW 2-128 [104]	Single key. KPA. Chosen IV. 14/32 rounds	Cube [105]	$2^{162.86}$
MUGI [106]	Distinguishing. KPA. Chosen IV. 21/32 rounds	Differential [107]	$2^{61.59}$
ZUC 1.6 [108]	Unbekannt	Siehe auch Anmerkung 3	
Public-Key-Verschlüsselung			
RSA[109]	Single key. COA. For RSA-250 (829-bit number)	Number field sieve [110, 111], Siehe auch Anmerkung 4	$2^{68.5}$
ElGamal [112]	Single key. CCA	Trivial algebraic	Instant
NTRUEncrypt [113]	Single key. COA	Hybrid [114] (Lattice reduction and combinatorial search)	PB, Siehe auch Anmerkung 5

Tab. 1.5: Beste bekannte Angriffe gegen 36 moderne Chiffren (PB = parameterbasiert)

Weitere Anmerkungen zu einigen der Chiffren aus Tabelle 1.5:

1. Ein weiterer Angriff, der behauptet, 855 Runden [115] von Trivium zu brechen, wurde in [116] in Frage gestellt.
2. Uns sind keine Angriffe auf Rabbit bekannt, die schneller als Brute-Force sind. Rabbit hat 4 Initialisierungsrunden. Die Werte innerhalb der Chiffre werden nach 2 Runden ausgeglichen [101], daher gibt es einen trivialen Unterscheidungsangriff gegen mindestens 1 Runde der Chiffre.
3. Es gibt Angriffe gegen frühere Versionen der ZUC-Chiffre. Die Kryptoanalyse der endgültigen Version, die von den Entwicklern erstellt wurde, ist nach unserem Wissen geheim.
4. Unsere Schätzung der oberen RSA-Schranke: In [111] wird die Angriffszeit mit 2700 Core-Jahren an Berechnungen mit einer Intel Xeon Gold 6130 CPU (jeweils 2,1 GHz) angegeben. Um diese Angriffszeit auf die RSA-250 „Verschlüsselungen“ umzurechnen, müssen wir wissen, wie viel Zeit im Durchschnitt benötigt wird, um eine Verschlüsselung auf dem genannten Prozessor anzuwenden. Für eine grobe Schätzung nahmen wir an, dass eine Verschlüsselung weniger Zeit benötigt als eine Integer-Operation, wie in [117] getestet.
5. Die tatsächliche Angriffszeit hängt von der Wahl der jeweiligen Parameter ab. Siehe [118].

1.9.3 Unizitätslängen

Tabelle 1.1 auf Seite 38 listet die Schlüsselräume für verschiedene Chiffren auf. Wir haben auch Angriffsmethoden gegen (klassische) Chiffren wie Brute-Force und Frequenzanalyse erörtert. Hier folgt ein weiteres Konzept dafür, wann Angriffe überhaupt erfolgreich sein können.

Claude Shannon definierte 1949 in [18] den Begriff „Unizitätslänge“ (auch: Eindeutigkeitsdistanz, engl. unicity distance). Die Unizitätslänge ist ein statistischer Wert, der die Mindestlänge eines Geheimtextes angibt, die ein Kryptoanalytiker benötigt, um die Chiffre durch Ausprobieren aller möglichen Schlüssel in einem Brute-Force-Angriff (oder einem anderen Angriff) zu knacken. Genauer gesagt handelt es sich um die Anzahl der Buchstaben im Klartext, die der Analytiker untersuchen muss, um sicher zu sein, dass es nur einen

richtigen Weg zur Entschlüsselung gibt. Im Gegensatz dazu kann die Analyse verschlüsselter Nachrichten mit einer Länge, die kleiner ist als die Unizitätslänge der Chiffre, zu mehreren gültigen Klartexten führen (und man kann nicht entscheiden, welches der richtige ist).

Der Wert der Unizitätslänge für eine Chiffre wird bestimmt, indem die Entropie H des Schlüsselraums der Chiffre durch die Redundanz der Klartextsprache geteilt wird. Die einfache monoalphabetische Substitutions-Chiffre hat beispielsweise eine (praktische) Schlüsselraumgröße von etwa 2^{88} (siehe Tabelle 1.1). Daher ist die Entropie H des Schlüsselraums $H(2^{88}) = 88$. Die Redundanz der englischen Sprache auf 3,2 geschätzt. Die Redundanz einer Sprache kann ermittelt werden, indem man z. B. die Entropie einer Sprache anhand der Komprimierung großer Textkorpora schätzt. Die Unizitätslänge U der einfachen monoalphabetischen Substitution kann also berechnet werden als $U = \lceil \frac{H(2^{88})}{3{,}2} \rceil = 28$. Da die Anzahl der Buchstaben in einer Nachricht eine ganze Zahl sein muss, wird die Unizitätslänge immer aufgerundet.

Tabelle 1.6 gibt einen Überblick über die Unizitätslänge für die englische Sprache – für alle Chiffren aus Tabelle 1.1.

Chiffre	**Unizitätslänge**	**Chiffre**	**Unizitätslänge**
Monoalphabet. Substitution (MASC)	28	Caesar	2
Vigenère (repeating keyword – 15 char.)	23	Vigenère (autokey – 314 char. message)	462
Playfair	25	Wheatstone Cryptograph	28
Lugagne Transpositeur	14	M-94 cylinder cipher	28
M-138A strip cipher	62	ADFGX	50
ADFGVX	67	Hebern 5-rotor	11
Kryha	31	Enigma Swiss K	10
Lugagne Le Sphinx	26	Abwehr Enigma G	11
Enigma I	24	SIGABA	30
Japanese Purple	33	Japanese JN-25 codebook (100 words)	12
Lorenz SZ40/SZ42	177	SG-41 „Hitler Mill“	54
M-209 pin & lug	61	Enigma M4	27
T-52d Geheimschreiber	25	Typex Mark 22	57
NEMA	20	Hagelin C-52	60
Hagelin CX-52	109	KL-7	36
Transvertex HC-9	73	VIC paper & pencil	29
Fialka	81	Hagelin CD-57	63
DES (56 bit)	18	RSA-4096	1273
AT&T TSD 3600-E Clipper chip	25	AES-256	80

Tab. 1.6: Unizitätslängen von 34 historischen und 4 „modernen“ Verschlüsselungssystemen

Man kann nicht nicht kommunizieren!

Zitat 3: Paul Watzlawick[36]

[36] Paul Watzlawick, Janet H. Beavin und Don D. Jackson, *Menschliche Kommunikation. Formen, Störungen, Paradoxien*, Huber, (c) 2007, Das erste der fünf pragmatischen Axiome ihrer Kommunikationstheorie.

1.10 Algorithmen-Typen und selbstgemachte Chiffren

Hier sollen kurz zwei Aspekte von Kryptoverfahren erwähnt werden, auf die oft nicht früh genug eingegangen wird: Arten von Algorithmen und das Sich-Ausdenken neuer Verfahren.

Arten von Algorithmen Man kann Algorithmen folgendermaßen kategorisieren:

- **Zufallsbasiert**
  Man kann Algorithmen aufteilen in *deterministische und heuristische* Verfahren. Oft machen sich Studenten nur deterministische Verfahren bewusst, bei denen die Ausgabe eindeutig durch die Eingabe vorgegeben ist. Bei heuristischen Verfahren werden Entscheidungen aufgrund zufälliger Werte getroffen und die Ergebnisse sind nur mit einer bestimmten Wahrscheinlichkeit richtig. Man kann noch genauer unterscheiden zwischen randomisierten Algorithmen, probabilistischen und heuristischen Verfahren, aber diese Feinheiten sind zum Verständnis des Gegensatzes zu deterministischen Verfahren nicht wichtig.

  In Kryptoverfahren spielt Zufall eine große Rolle. Immer müssen die Schlüssel zufällig gewählt werden, so dass zumindest bei der Schlüsselgenerierung „Zufall“ erzeugt werden muss. Zusätzlich sind manche Verfahren (vor allem aus der Kryptoanalyse) heuristisch.

- **Konstanten-basiert**
  Viele moderne Verfahren (insbesondere Hashverfahren und symmetrische Verschlüsselungsverfahren) benutzen numerische Konstanten. Diese sollten nachvollziehbar sein und keine Hintertüren ermöglichen. Zahlen, die das erfüllen, nennt man im Englischen „Nichts-im-Ärmel“-Zahlen: Nothing-up-my-sleeve numbers.[37]

Neue Algorithmen Es kommt immer wieder vor, dass sich jemand ohne tiefere Kenntnis der entsprechenden Design-Konzepte ein „neues“ Verschlüsselungs-Verfahren ausdenkt. Doch die Realität zeigt, dass dies keine gute Idee ist. Deshalb lernt man normalerweise früh, kein eigenes Kryptosystem zu entwerfen, wenn man hofft, dass einen die Tatsache schützt, dass es nicht bekannt ist. Hierfür gibt es viele Gründe, unter anderem: Es braucht nur einen verärgerten Mitarbeiter oder einen anderen böswilligen Akteur, um die Geheimnisse zu verraten, die das System sicher machen. Der Entwurf sicherer kryptografischer Verfahren ist extrem schwierig. Es ist unglaublich einfach, etwas zu entwickeln, das sicher aussieht, aber tatsächlich Informationen preisgibt.

Ein Preisgeld auszuloben und nur einzelne Geheimtexte anzubieten, ist unprofessionell – seriöse Forscher haben nur wenig Zeit und werden keine Mühe darauf verwenden (vielleicht geben sie es aus didaktischen Gründen an Studenten als Übung). Wenn Sie ein neues Verschlüsselungsverfahren entwickeln wollen, veröffentlichen Sie es zunächst mit einer ausführlichen Erklärung seiner Funktionsweise, seinen Vorteilen und allen Beweisen für seine Sicherheit. Dann können Sie sehen, ob jemand eine Schwachstelle findet. Dies ist kein schneller Prozess – Sie sollten damit rechnen, dass es Jahre dauern kann.[38]

[37] https://en.wikipedia.org/wiki/Nothing_up_my_sleeve_number

[38] Einige weitere Überlegungen dazu finden Sie z. B. unter https://www.quora.com/If-I-made-my-own-encryption-some-new-math-model-for-encryption-can-the-government-crack-this.

1.11 Weitere Informationsquellen / Empfohlene Bücher

Hier sind einige gute Kryptografie-Bücher mit nützlichem Hintergrundwissen zu verschiedenen Themen – geordnet von Anfängern (eher geschichtlich) über Wissende (eher angewandt) bis hin zu Fortgeschrittenen (mit Schwerpunkt auf der Theorie):

- David Kahn: The Codebreakers, 1995 [119]
- Elonka Dunin und Klaus Schmeh: Codebreaking: A Practical Guide, expanded ed, 2023 [120]
- Simon Singh: The Code Book, 2000 [121]
- Klaus Schmeh: Kryptographie – Verfahren, Protokolle, Infrastrukturen, 6. Auflage, 2016 [122]
- Bruce Schneier, Applied Cryptography, Protocols, Algorithms, and Source Code in C, 2. Aufl., 1996 [7]
- Christof Paar, Jan Pelzl und Tim Güneysu: Understanding Cryptography, 2. Auflage, 2024 [123]
- David Wong: Kryptografie in der Praxis, 2023 [124] (unser Favorit)
- Jean-Philippe Aumasson: Serious Cryptography, 2017 [125]
- Mike Rosulek: The Joy of Cryptography, 2021
- Niels Ferguson, Bruce Schneier und Tadayoshi Kohno: Cryptography Engineering, 2010
- Dan Boneh und Victor Shoup: A Graduate Course in Applied Cryptography, v0.6, 2023
- Mark Stamp und Richard M. Low: Applied Cryptanalysis, 2007 [126]
- Rolf Oppliger, Cryptography 101, 2021 [14]
- Jonathan Katz und Yehuda Lindell: Introduction to Modern Cryptography, 3. Aufl., 2020 [127]
- Douglas R. Stinson: Cryptography – Theory and Practice, 3. Auflage, 2006 [128]

Neben den Informationen in den oben genannten Büchern und in den folgenden Kapiteln gibt es auch eine große Anzahl von Webseiten und die Online-Hilfe der jeweiligen CrypTool-Varianten, die viele Details über Verschlüsselungsmethoden enthalten.

Einen leicht zugänglichen Überblick über die verschiedenen Verschlüsselungsverfahren bieten die Bücher von Bruce Schneier [7], Klaus Schmeh [122] und Paar/Pelzl/Güneysu [123]. Für einen etwas tieferen Einstieg können wir neben dem Buch von Rolf Oppliger [14] auch die Bücher von David Wong [124], Jean-Philippe Aumasson [125] und Douglas R. Stinson [128] empfehlen.

Für Software-Entwickler gibt es gute praxisnahe Checklisten zur Kryptografie auf den Seiten von **OWASP** (Open Worldwide Application Security Project). Manchmal sind die Informationen aber rudimentär oder sehr verteilt. Hier drei Beispiele aus den „Flagship Projects“ OWASP Cheat Sheet Series und OWASP Web Security Testing Guide: `https://cheatsheetseries.owasp.org/cheatsheets/Cryptographic_Storage_Cheat_Sheet.html`, `https://cheatsheetseries.owasp.org/cheatsheets/Password_Storage_Cheat_Sheet.html`, `https://owasp.org/www-project-web-security-testing-guide/stable/4-Web_Application_Security_Testing/09-Testing_for_Weak_Cryptography/README`.

1.12 Anhang: AES-Visualisierungen/-Implementierungen

AES ist inzwischen weltweit wahrscheinlich der am häufigsten eingesetzte moderne Verschlüsselungs-Algorithmus. AES ist ein sicheres, standardisiertes, symmetrisches Verfahren, das bspw. in Wifi- und Browser-Verbindungen Daten verschlüsselt. Die Varianten AES-192 und AES-256 sind in den USA für staatliche Dokumente mit höchstem Geheimhaltungsgrad zugelassen.

Im Folgenden wird zuerst eine AES-Animation in CTO vorgestellt; und danach wird AES direkt ausgeführt – einmal in CT2 und zweimal mit OpenSSL (einmal auf der Kommandozeile des Betriebssystems und einmal im OpenSSL-WebAssembly-Plugin in CTO).

1.12.1 AES-Animation in CTO[39]

Abb. 1.8 auf der nächsten Seite symbolisiert, dass der moderne Verschlüsselungsalgorithmus beide Eingaben (den Schlüssel und den Klartext) jeweils in binärer Form erhält, und dass er seinen Output ebenfalls in binärer Form erzeugt. Wie die meisten modernen (Block-)Chiffren enthält der Algorithmus einen „key-scheduling"-Teil, wo aus dem übergebenen Schlüssel (auch Sessionkey oder Hauptschlüssel genannt) die Rundenschlüssel erzeugt werden, und einen zweiten Teil, wo mit den erzeugten Rundenschlüsseln dann die eigentliche Verschlüsselung durchgeführt wird.

Die beiden Screenshots 1.8 und 1.9 sind aus der AES-Animation in CrypTool-Online (CTO). Screenshot 1.10 stammt aus CT1, das Bild ist aber auch Teil der Animation in CTO.

1.12.2 AES in CT2

Nach diesen Visualisierungen wollen wir im Folgenden an einem konkreten Beispiel einen Klartext der Länge 128 bit (1 Block) mit einem 128-bit Schlüssel mit AES im CBC-Modus verschlüsseln. Vom erhaltenen Geheimtext interessiert uns nur der erste Block (füllt der Klartext einen Block nicht auf, nutzen wir hier der Einfachheit halber Null-Padding).

Zur Veranschaulichung machen wir das einmal mit CT2 und zweimal mit OpenSSL[40].

Der Klartext `AESTEST1USINGCT2` wird nach Hex (41 45 53 54 45 53 54 31 55 53 49 4E 47 43 54 32) konvertiert. Damit und mit dem Schlüssel 3243F6A8885A308D313198A2E0370734 erzeugt die AES-Komponente dann den Geheimtext. Dieser lautet in Hex: B1 13 D6 47 DB 75 C6 D8 47 FD 8B 92 9A 29 DE 08

Abb. 1.11 zeigt die Verschlüsselung eines Blocks in **CT2**.[41]

[39] `https://www.cryptool.org/de/cto/aes-animation`

[40] OpenSSL ist eine für Webserver sehr verbreitete freie Open-Source-Kryptobibliothek, zu der auch das Kommandozeilentool `openssl` gehört, mit dem man die Funktionalität auf vielen Betriebssystemen direkt ausprobieren kann.
Eine Einführung in das CLI `openssl` finden Sie im Anhang A.8 auf Seite 817.

[41] Die gezeigte Vorlage ist ähnlich zur folgenden in CT2 ausgelieferten Vorlage: `CT2 Vorlagen ▷ Kryptografie ▷ Modern ▷ Symmetrisch ▷ AES-Chiffre (Texteingabe)`

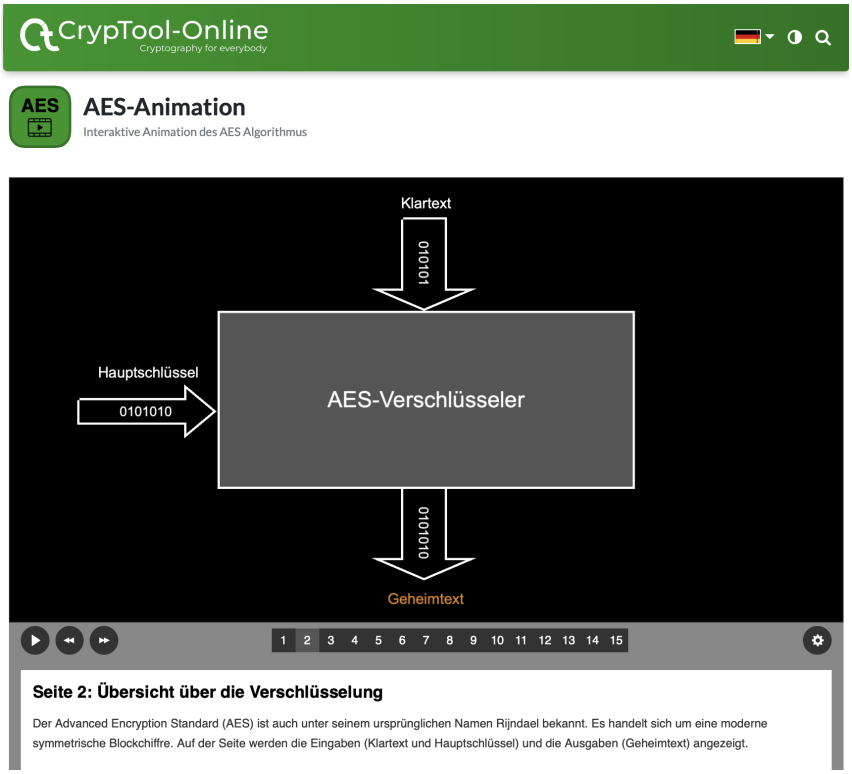


Abb. 1.8: AES-Visualisierung aus CTO (Teil 1)

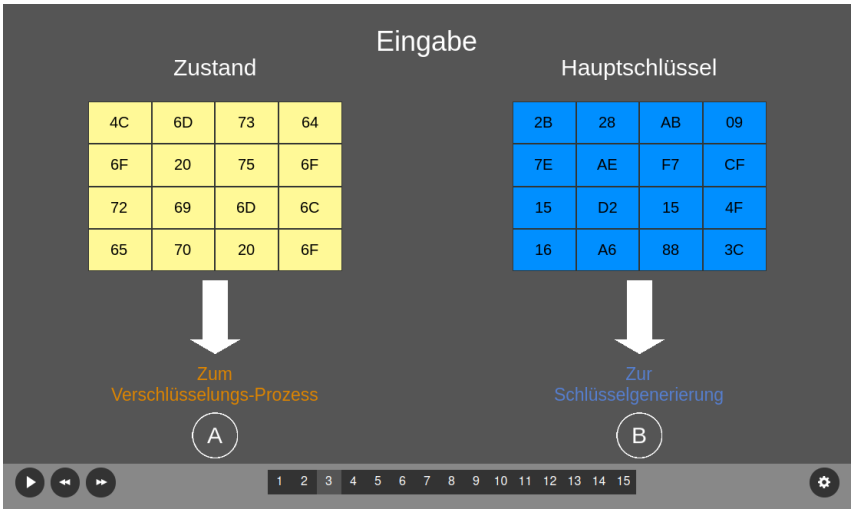


Abb. 1.9: AES-Visualisierung aus CTO (Teil 2)

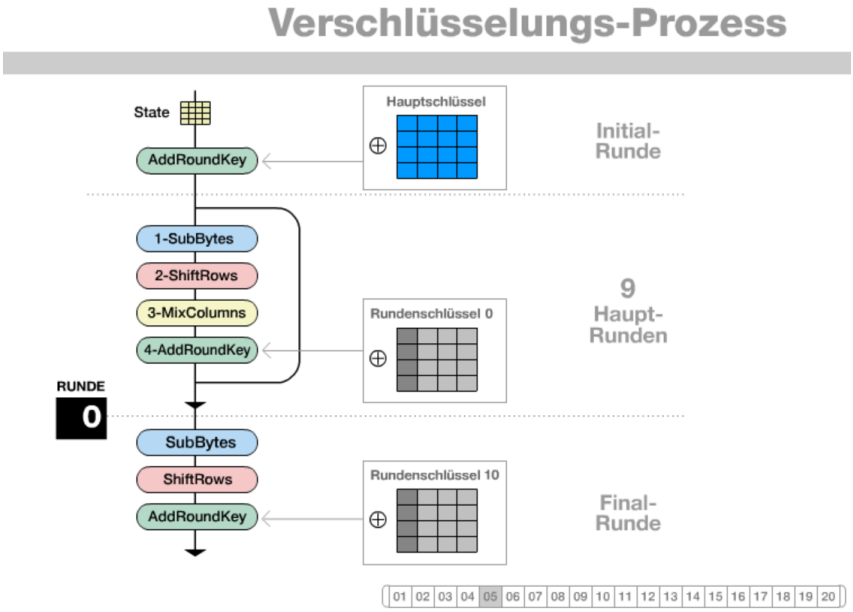


Abb. 1.10: AES-Visualisierung von Enrique Zabala aus CT1

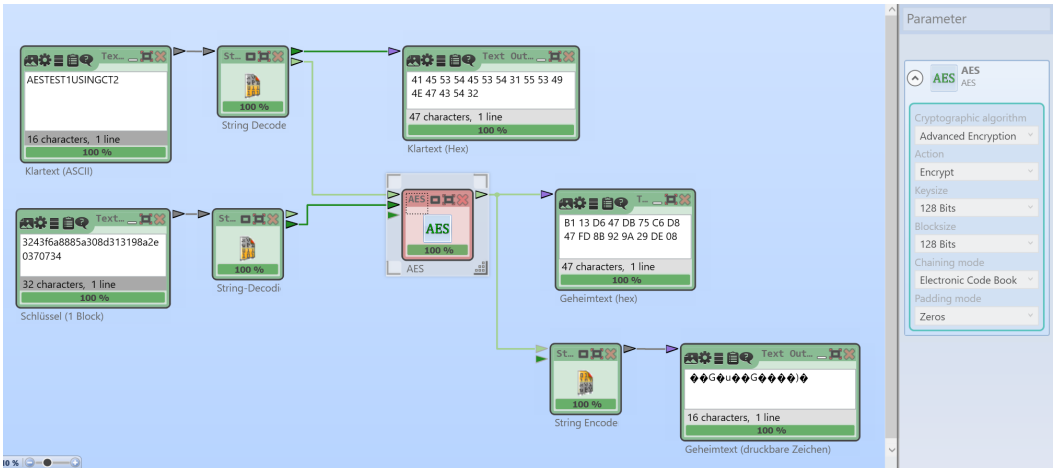


Abb. 1.11: AES-Verschlüsselung (hier genau 1 Block ohne Padding) in CT2

1.12.3 AES mit OpenSSL auf der Kommandozeile des Betriebssystems

Das OpenSSL-Beispiel 1.12.1 erzielt dasselbe Ergebnis wie CT2 mit **OpenSSL** auf der Kommandozeile (hier unter Windows).

OpenSSL-Beispiel 1.12.1: AES-Verschlüsselung (von genau einem Block ohne Padding)

```
>openssl enc -e -aes-128-cbc -K 3243F6A8885A308D313198A2E0370734 -iv 00000000000000000000000000000000 -in ▶
 ▶klartext-1.hex  -out klartext-1.hex.enc
>dir
06.07.2016  12:43                16 key.hex
20.07.2016  20:19                16 klartext-1.hex
20.07.2016  20:37                32 klartext-1.hex.enc
```

Anmerkung: Wie OpenSSL-Beispiel 1.12.2 zeigt, kann man mit etwas Mühe, Pipes und dem Tool `xxd` in der Bash-Shell dasselbe und auch ohne die Nutzung von Zwischen-Dateien erreichen:[42]

OpenSSL-Beispiel 1.12.2: AES-Verschlüsselung (ohne temporäre Dateien) mit Bash

```
$ echo 0: 41 45 53 54 45 53 54 31 55 53 49 4E 47 43 54 32 | xxd -r | openssl enc -e -aes-128-cbc -nopad -K 324▶
 ▶3F6A8885A308D313198A2E0370734 -iv 00000000000000000000000000000000  | xxd -p
b113d647db75c6d847fd8b929a29de08

$ echo -n AESTEST1USINGCT2 | openssl enc -e -aes-128-cbc -nopad -K 3243F6A8885A308D313198A2E0370734 -iv 000000▶
 ▶00000000000000000000000000  | xxd -p
b113d647db75c6d847fd8b929a29de08
```

1.12.4 AES mit OpenSSL in CTO[43]

Da CTO eine WebAssembly-basierte Version von OpenSSL integriert hat, kann man dies auch lokal im Browser ausführen – ohne die Notwendigkeit OpenSSL zu installieren. Während Linux-Systeme OpenSSL meist schon an Bord haben, ist das bei Windows-Systemen oder auf Smartphones nicht so. Für solche Systeme ist dieses CTO-Plugin nützlich.

Für das Beispiel in Abb. 1.12 auf der nächsten Seite speichern wir die Nachricht „AESTEST1USINGCT2“ in einer Datei namens „klartext-1.hex“. Dann laden wir diese Datei aus dem Dateisystem des Betriebssystems in ein virtuelles Dateisystem im Browser hoch: Dieser Upload wird im Tab „Dateien“ des OpenSSL-Plugins durchgeführt. Dann führt das OpenSSL-Plugin denselben `openssl`-Befehl wie zuvor im Terminal aus (siehe Abschnitt 1.12.3). Und wenn man die resultierende Datei „klartext-1.hex.enc“ herunterlädt und mit dem Ergebnis aus dem Terminal vergleicht, sieht man, dass beide identisch sind.

[42] Das Tool `xxd` erstellt einen Hex-Dump von einer gegebenen Datei oder von der Standardeingabe. Mit der Option „-r“ wandelt es Hex-Dump wieder in seine ursprüngliche Binärform um.

[43] https://www.cryptool.org/de/cto/openssl

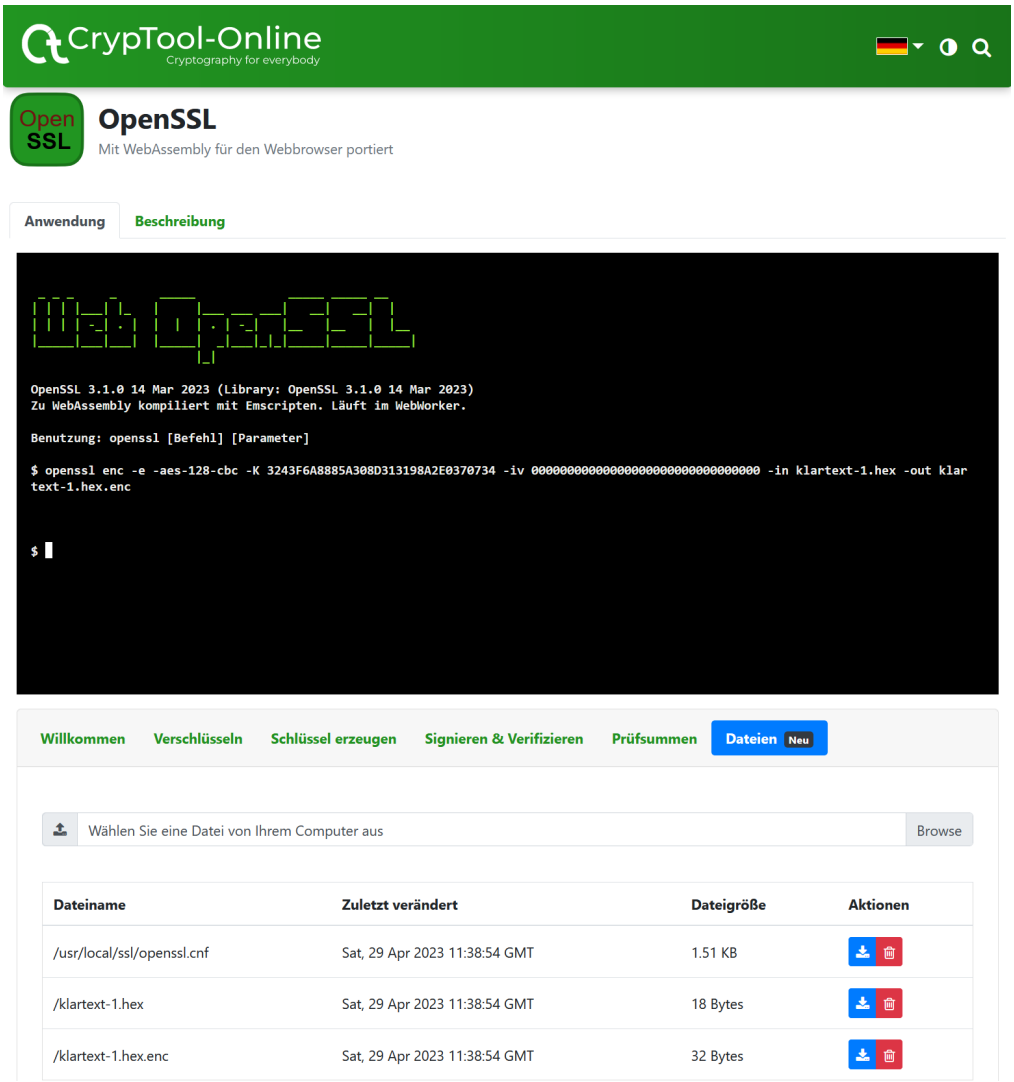


Abb. 1.12: AES-Verschlüsselung mit OpenSSL im Browser

1.13 Anhang: Didaktische Beispiele für symmetrische Chiffren mit SageMath

Dieser Anhang zeigt die SageMath-Implementierung einer für didaktische Zwecke abgestrippten Chiffre (Mini-AES). Im Anschluss werden weitere Publikationen in dieser Richtung aufgeführt.

1.13.1 Mini-AES

Das SageMath-Modul `crypto/block_cipher/miniaes.py` enthält den Mini-AES, mit dem Studenten die Funktionsweise moderner Blockchiffren untersuchen können.

Mini-AES, ursprünglich vorgestellt von [129], ist eine vereinfachte Variante des Advanced Encryption Standard (AES) für Ausbildungszwecke.

Hier ist eine kleine Gegenüberstellung wie der Mini-AES gegenüber dem AES vereinfacht wurde:

- AES hat eine Blocklänge von 128 bit, und unterstützt Schlüssellängen von 128, 192 und 256 bit. Die Rundenzahl beträgt 10, 12 oder 14, passend zur jeweiligen Schlüssellänge.
  Mini-AES hat eine Blocklänge von 16 bit, eine Schlüssellänge von 16 bit, und 2 Runden.
- Der 128-bit-Block von AES wird als Matrix von 4×4 Bytes dargestellt, wohingegen Mini-AES seinen 16-bit-Block als Matrix von 2×2 Nibbles (Halb-Bytes) darstellt.
- Die AES-Schlüsselerzeugung nimmt den 128-bit langen geheimen Schlüssel und stellt ihn als Gruppe von vier 32-bit-Worten dar.
  Die Mini-AES-Schlüsselerzeugung nimmt den 16-bit langen geheimen Schlüssel und stellt ihn als Gruppe von vier Nibbles (4-bit-Worten) dar.

Wie man Mini-AES benutzt, ist ausführlich auf der SageMath-Referenz-Seite beschrieben: `https://doc.sagemath.org/html/en/reference/cryptography/sage/crypto/block_cipher/miniaes.html`.

Das SageMath-Beispiel 1.13.1 stammt ursprünglich aus den Release-Notes von SageMath 4.1[44] und ruft den Mini-AES auf.

SageMath-Beispiel 1.13.1: Ver- und Entschlüsselung mit dem Mini-AES

```
print("\n# CHAP01 -- Sage-Script-SAMPLE 010: =========")

# (1) Encrypting a plaintext using Mini-AES
from sage.crypto.block_cipher.miniaes import MiniAES
maes = MiniAES()
K = FiniteField(16, "x")
MS = MatrixSpace(K, 2, 2)

P = MS([K("x^3 + x"), K("x^2 + 1"), K("x^2 + x"), K("x^3 + x^2")]); print("(1) P:\n",P, sep="")
key = MS([K("x^3 + x^2"), K("x^3 + x"), K("x^3 + x^2 + x"), K("x^2 + x + 1")]); print("key:\n",key, sep="")
C = maes.encrypt(P, key); print("C:\n",C, sep="")

# decryption process
plaintxt = maes.decrypt(C, key); print(plaintxt == P)

# (2) Working directly with binary strings
maes = MiniAES()
bin = BinaryStrings()
key = bin.encoding("KE"); print("\n(2) key:\n",key, sep="")

P = bin.encoding("Encrypt this secret message!"); print("P:\n",P,sep="")
C = maes(P, key, algorithm="encrypt"); print("C:\n",C,sep="")
plaintxt = maes(C, key, algorithm="decrypt"); print(plaintxt == P)

# (3) Or working with integers n such that 0 <= n <= 15:
maes = MiniAES()
P = [n for n in range(16)]; print("\n(3) P:\n",P, sep="")
key = [2, 3, 11, 0]; print("key:\n",key, sep="")

P = maes.integer_to_binary(P)
key = maes.integer_to_binary(key)
C = maes(P, key, algorithm="encrypt"); print("C:\n",C, sep="")
plaintxt = maes(C, key, algorithm="decrypt"); print(plaintxt == P)

#----------------------------------
# CHAP01 -- Sage-Script-SAMPLE 010: =========
# (1) P:
# [  x^3 + x   x^2 + 1]
```

[44] Siehe `https://mvngu.wordpress.com/2009/07/12/sage-4-1-released/`.
Weiterer Beispiel-Code zum Mini-AES findet sich in [130, Kap. 6.5 und Anhang D].

Fortsetzung SageMath-Beispiel 1.13.1

```
# [  x^2 + x x^3 + x^2]
# key:
# [    x^3 + x^2       x^3 + x]
# [x^3 + x^2 + x   x^2 + x + 1]
# C:
# [            x       x^2 + x]
# [x^3 + x^2 + x       x^3 + x]
# True
#
# (2) key:
# 0100101101000101
# P:
# 0100010101101110011000110111001001111001011100000111010000100000011101000110100001101001011100110010000001111
#  ▶0011011001010110001101110010011001010111010000100000011011010110010101110011011100110110000101100111011001 0▶
#  ▶100100001
# C:
# 1000100010100110111100000111100001001100111011010100011101101101010010011101111101010110011100111001000111011▶
#  ▶0010101010001010011111011001100101000100011101101101001000001100011000110000011100001110011010111100000000 1▶
#  ▶110001001
# True
#
# (3) P:
# [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]
# key:
# [2, 3, 11, 0]
# C:
# 1100100000100011111001010101010101011011100111111000100001110000 1
# True
```

Weitere Details zu in SageMath enthaltenen Kryptoverfahren (z. B. zum Simplified Data Encryption Standard SDES) finden sich bspw. in der Diplomarbeit von Minh Van Nguyen [131].

Ein altes Sprichwort, angeblich geprägt von der US National Security Agency (NSA), sagt:
„Angriffe werden immer besser, niemals schlechter."
"Attacks always get better; they never get worse."

Zitat 4: IETF[45]

1.13.2 Symmetrische Chiffren für Lehrzwecke

Verglichen mit den auf der Zahlentheorie beruhenden Public-Key-Verschlüsselungsverfahren, ist die Struktur von AES und den meisten anderen modernen symmetrischen Verschlüsselungsverfahren (wie DES, IDEA oder Present) sehr komplex und kann nicht so einfach wie RSA erklärt werden.

Deshalb wurden zu Lehrzwecken vereinfachte Varianten moderner symmetrischer Verfahren entwickelt, um Einsteigern die Möglichkeit zu geben, Ver- und Entschlüsselung von Hand zu lernen und ein besseres Verständnis zu gewinnen, wie die Algorithmen im Detail funktionieren. Diese vereinfachten Varianten helfen auch, die entsprechenden Kryptoanalyse-Methoden[46] zu verstehen und anzuwenden.

[45] https://datatracker.ietf.org/doc/html/rfc4270, 2005

[46] Ein sehr guter Start in die Kryptoanalyse ist das Buch von Mark Stamp [126]. Ebenfalls gut, aber sehr high-level und nur bezogen auf die Analyse symmetrischer Blockchiffren ist der Artikel von Bruce Schneier [132].
Einige der Cipher-Challenges bei *MysteryTwister* (https://www.mysterytwister.org) sind ebenfalls gut für Lehrzwecke einsetzbar.

Die bekanntesten Varianten sind SDES (Simplified DES)[47] und S-AES (Simplified-AES)[48] von Ed Schaefer und seinen Studenten [133], und Mini-AES (siehe Abschnitt 1.13.1 auf Seite 59):

- Edward F. Schaefer: *A Simplified Data Encryption Standard Algorithm* [134]
- Raphael Chung-Wei Phan: *Mini Advanced Encryption Standard (Mini-AES): A Testbed for Cryptanalysis Students* [129]
- Raphael Chung-Wei Phan: *Impossible differential cryptanalysis of Mini-AES* [135]
- Mohammad A. Musa, Edward F. Schaefer, Stephen Wedig: *A simplified AES algorithm and its linear and differential cryptanalyses* [136]
- Nick Hoffman: *A SIMPLIFIED IDEA ALGORITHM* [137]
- S. Davod. Mansoori, H. Khaleghei Bizaki: *On the vulnerability of Simplified AES Algorithm Against Linear Cryptanalysis* [138]

1.13.3 Weitere Krypto-Algorithmen in SageMath

Die Referenz zu SageMath führt als weitere symmetrische Funktionen u. a. auf:[49]

- PRESENT
- Linear feedback shift register (LFSR)
- S-Boxes

Literatur zu Kapitel 1

[1] International Association for Cryptologic Research (IACR). *IACR*. URL: https://www.iacr.org/ (besucht am 29. 04. 2023) (siehe S. 22).

[2] BBC documentary film. *War of the letters. German: „Krieg der Buchstaben“*. 1994. URL: https://www.youtube.com/watch?v=yfw1JLI8aWk (besucht am 10. 08. 2023) (siehe S. 23).

[3] Douglas Robert Stinson und Maura B. Paterson. *Cryptography: Theory and Practice*. 4. Aufl. Chapman und Hall/CRC, 2018. URL: https://doi.org/10.1201/9781315282497 (siehe S. 24, 44).

[4] Randall K. Nichols. *Classical Cryptography Course, Volume 1 and 2*. Techn. Ber. 12 Lektionen. Aegean Park Press 1996, 1996. URL: https://www.cryptogram.org/resource-area/crypto-lessons-tutorials-lanaki/ (siehe S. 25).

[5] Kristian Laurent Haan. *Advanced Encryption Standard (AES)*. 2008. URL: http://www.codeplanet.eu/tutorials/cpp/51-advanced-encryption-standard.html (siehe S. 26).

[47] Visualisierung: Macht man in CT2 einen Doppelklick auf den Titel der SDES-Komponente, ist in der Fullscreen-Ansicht zu sehen, wie die Bits der eingegebenen Daten durch den Algorithmus fließen. Ein Screenshot dazu: https://www.facebook.com/CrypTool2/photos/a.505204806238612.1073741827.243959195696509/597354423690316

[48] Siehe die Vorlage: CT2 Vorlagen ▷ Kryptografie ▷ Modern ▷ Symmetrisch ▷ S-AES

[49] Siehe https://doc.sagemath.org/html/en/reference/cryptography/index.html und https://doc.sagemath.org/html/en/reference/cryptography/sage/crypto/stream.html

[6] BSI. *Technische Richtlinie TR-02102-1, Kryptographische Verfahren: Empfehlungen und Schlüssellängen (Version 2024-01)*. Techn. Ber. 2024. URL: https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/TechnischeRichtlinien/TR02102/BSI-TR-02102.pdf (siehe S. 27, 30).

[7] Bruce Schneier. *Applied Cryptography, Protocols, Algorithms, and Source Code in C*. 2. Aufl. Wiley, 1996 (siehe S. 28, 54).

[8] Helmut Witten und Ralph-Hardo Schulz. „RSA & Co. in der Schule: Moderne Kryptologie, alte Mathematik, raffinierte Protokolle. NF Teil 2: RSA für große Zahlen". In: *LOG IN* 143 (2006), S. 50–58. URL: https://informatik.schule.de/krypto/ (siehe S. 29).

[9] B. Esslinger, J. Schneider und V. Simon. „Krypto + NSA = ? – Kryptografische Folgerungen aus der NSA-Affäre". In: *KES Zeitschrift für Informationssicherheit* 2014.1 (März 2014), S. 70–77. URL: https://www.cryptool.org/media/publications/journals/krypto_nsa.pdf (siehe S. 31).

[10] A. Ray Miller. *The Cryptographic Mathematics of Enigma*. Revised edition 2019. Center for Cryptologic History, National Security Agency, 1995. URL: https://www.nsa.gov/portals/75/documents/about/cryptologic-heritage/historical-figures-publications/publications/wwii/CryptoMathEnigma_Miller.pdf (siehe S. 34).

[11] Olaf Ostwald. *Cryptographic design flaws of early Enigma*. 2023. URL: https://cryptocellar.org/enigma/files/enigma-design-flaws.pdf (besucht am 09. 08. 2023) (siehe S. 36).

[12] International Conference on Cryptologic History (ICCH). *ICCH*. URL: https://www.cryptologichistory.org/ (besucht am 11. 06. 2023) (siehe S. 37).

[13] Oded Goldreich. *Foundations of cryptography: volume 2, basic applications*. Cambridge university press, 2009 (siehe S. 38).

[14] Rolf Oppliger. *Cryptography 101: From Theory to Practice*. 1. Aufl. Mit Slides zu allen 18 Kapiteln. Artech House, 2021. URL: https://rolf.esecurity.ch/?page_id=465 (siehe S. 39, 54).

[15] Alfred J. Menezes, Paul C. van Oorschot und Scott A. Vanstone. *Handbook of Applied Cryptography*. 5. Aufl. Series on Discrete Mathematics and Its Application. CRC Press, 2001. URL: https://cacr.uwaterloo.ca/hac/ (siehe S. 40, 46).

[16] Orr Dunkelman, Nathan Keller und Adi Shamir. „A practical-time related-key attack on the KASUMI cryptosystem used in GSM and 3G telephony". In: *Advances in Cryptology–CRYPTO 2010: 30th Annual Cryptology Conference, Santa Barbara, CA, USA, August 15-19, 2010. Proceedings 30*. Springer. 2010, S. 393–410 (siehe S. 42, 50).

[17] Mihir Bellare und Phillip Rogaway. *Introduction to modern cryptography*. 2005, S. 283 (siehe S. 43).

[18] Claude E. Shannon. „Communication Theory of Secrecy Systems". In: *The Bell System Technical Journal* 28.4 (1949), S. 656–715 (siehe S. 44, 51).

[19] Elaine Barker. *Recommendation for Key Management: Part 1 – General*. Special Publication (NIST SP), National Institute of Standards und Technology, 2020. URL: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-57pt1r5.pdf (siehe S. 45, 46).

[20] Arjen K. Lenstra. „Key Lengths: Contribution to The Handbook of Information Security". In: (2010). URL: https://infoscience.epfl.ch/record/164539/files/NPDF-32.pdf (siehe S. 45).

[21] Daniel J. Bernstein und Tanja Lange. „Non-uniform cracks in the concrete: the power of free precomputation". In: *Advances in Cryptology – ASIACRYPT 2013. Lecture Notes in Computer Science*. 2012, S. 321–340. DOI: 10.1007/978-3-642-42045-0_17. URL: https://cr.yp.to/nonuniform/nonuniform-20130914.pdf (siehe S. 45).

[22] Galbraith, Steven. *Security levels in cryptography*. 2020. URL: https://www.math.auckland.ac.nz/~sgal018/ACISP.pdf (besucht am 07. 07. 2024) (siehe S. 45).

[23] Jean-Philipp Aumasson. *Cryptanalysis vs. Reality – White paper for Black Hat Abu Dhabi*. 2011. URL: https://media.blackhat.com/bh-ad-11/Aumasson/bh-ad-11-Aumasson-CryptanalysisVSReality_WP.pdf (siehe S. 46).

[24] Daniel J. Bernstein. „Understanding brute force". In: *ECRYPT STVL. Workshop on Symmetric Key Encryption*. 2005. URL: https://cr.yp.to/snuffle/bruteforce-20050425.pdf (siehe S. 46).

[25] Arjen K. Lenstra. „Unbelievable Security: Matching AES Security Using Public Key Systems". In: *Advances in Cryptology –ASIACRYPT 2001. Lecture Notes in Computer Science 2248*. 2001, S. 67–86. DOI: https://doi.org/10.1007%2F3-540-45682-1_5. URL: https://www.iacr.org/archive/asiacrypt2001/22480067.pdf (siehe S. 46).

[26] European Union Agency for Cybersecurity und N. Smart. *Algorithms, key size and parameters – Report – 2014*. Hrsg. von N Smart. European Network und Information Security Agency (ENISA), 2014. URL: https://op.europa.eu/en/publication-detail/-/publication/e1a3ba61-f4b0-4a5b-bd75-a4d471a26b83 (siehe S. 46).

[27] Paul E. Hoffman und Hilarie Orman. *Determining Strengths For Public Keys Used For Exchanging Symmetric Keys*. RFC 3766. 2004. DOI: 10.17487/RFC3766. URL: https://datatracker.ietf.org/doc/html/rfc3766 (siehe S. 46).

[28] Damien Giry. *BlueKrypt: Cryptographic Key Length Recommendation*. Version 32.3. Mai 2020. URL: https://www.keylength.com/ (siehe S. 46).

[29] NIST und CCCS. *Implementation Guidance for FIPS 140-2 and the Cryptographic Module Validation Program (CMVP)*. U.S. Government's National Institute of Standards, Technology (NIST) und the Canadian Centre for Cyber Security (CCCS). 2023. URL: https://csrc.nist.gov/csrc/media/projects/cryptographic-module-validation-program/documents/fips140-2/fips1402ig.pdf (siehe S. 46).

[30] Daniel J. Bernstein und Tanja Lange. *SafeCurves: choosing safe curves for elliptic-curve cryptography. The rho method*. 2013. URL: https://safecurves.cr.yp.to/rho.html (besucht am 06. 07. 2024) (siehe S. 46).

[31] Jean-Philipp Aumasson. *Too Much Crypto – Real World Crypto Symposium*. 2020. URL: https://eprint.iacr.org/2019/1492.pdf (siehe S. 47).

[32] Nils Kopal. „Solving Classical Ciphers with CrypTool 2". In: *Proceedings of the 1st International Conference on Historical Cryptology*. 2018, S. 29–38 (siehe S. 48).

[33] Nils Kopal. „Cryptanalysis of Homophonic Substitution Ciphers using Simulated Annealing with Fixed Temperature". In: *Proceedings of the 2nd International Conference on Historical Cryptology*. 2019, S. 107–116 (siehe S. 48).

[34] George Lasry, Norbert Biermann und Satoshi Tomokiyo. „Deciphering Mary Stuart's lost letters from 1578-1584". In: *Cryptologia* 47.2 (2023), S. 101–202. URL: https://www.tandfonline.com/doi/full/10.1080/01611194.2022.2160677 (siehe S. 48).

[35] George Lasry, Beáta Megyesi und Nils Kopal. „Deciphering papal ciphers from the 16th to the 18th Century". In: *Cryptologia* 45.6 (2021), S. 479–540. URL: https://www.tandfonline.com/doi/full/10.1080/01611194.2020.1755915 (siehe S. 48).

[36] Elonka Dunin u. a. „How we set new world records in breaking Playfair ciphertexts". In: *Cryptologia* 46.4 (2022), S. 302–322 (siehe S. 48).

[37] George Lasry. „Solving a 40-letter Playfair Challenge with CrypTool 2". In: *Proceedings of the 2nd International Conference on Historical Cryptology*. 2019, S. 23–26 (siehe S. 48).

[38] George Lasry. „Deciphering German diplomatic and naval attaché messages from 1914-1915". In: *Proceedings of the 1st International Conference on Historical Cryptology*. 2018, S. 55–64 (siehe S. 48).

[39] George Lasry u. a. „Cryptanalysis of Chaocipher and solution of Exhibit 6“. In: *Cryptologia* 40.6 (2016), S. 487–514 (siehe S. 48).

[40] George Lasry, Nils Kopal und Arno Wacker. „Cryptanalysis of columnar transposition cipher with long keys“. In: *Cryptologia* 40.4 (2016), S. 374–398 (siehe S. 48).

[41] George Lasry, Nils Kopal und Arno Wacker. „Solving the Double Transposition Challenge with a Divide-and-Conquer Approach“. In: *Cryptologia* 38.3 (2014), S. 197–214 (siehe S. 48).

[42] George Lasry u. a. „Deciphering ADFGVX messages from the eastern front of World War I“. In: *Cryptologia* 41.2 (2017), S. 101–136 (siehe S. 48).

[43] James J Gillogly. „Ciphertext-only Cryptanalysis of Enigma“. In: *Cryptologia* 19.4 (1995), S. 405–413 (siehe S. 48).

[44] George Lasry, Nils Kopal und Arno Wacker. „Cryptanalysis of Enigma double indicators with hill climbing“. In: *Cryptologia* 43.4 (2019), S. 267–292 (siehe S. 48).

[45] Olaf Ostwald und Frode Weierud. „Modern breaking of Enigma ciphertexts“. In: *Cryptologia* 41.5 (2017), S. 395–421 (siehe S. 48).

[46] George Lasry. „Solving a Tunny Challenge with Computerized “Testery” Methods“. In: *Proceedings of the 3rd International Conference on Historical Cryptology*. 2020 (siehe S. 48).

[47] Lasry, George and Kopal, Nils and Wacker, Arno. „Automated known-plaintext cryptanalysis of short Hagelin M-209 messages“. In: *Cryptologia* 40.1 (2016), S. 49–69 (siehe S. 48).

[48] George Lasry, Nils Kopal und Arno Wacker. „Ciphertext-only cryptanalysis of short Hagelin M-209 ciphertexts“. In: *Cryptologia* 42.6 (2018), S. 485–513 (siehe S. 48).

[49] George Lasry. „A practical meet-in-the-middle attack on SIGABA“. In: *Proceedings of the 2nd International Conference on Historical Cryptology*. 2019, S. 23–26 (siehe S. 48).

[50] George Lasry. „Cracking SIGABA in less than 24 hours on a consumer PC“. In: *Cryptologia* 47.1 (2021), S. 1–37. URL: https://www.tandfonline.com/doi/full/10.1080/01611194.2021.1989522 (siehe S. 48).

[51] Mitsuru Matsui. „Linear cryptanalysis method for DES cipher“. In: *Advances in Cryptology—EUROCRYPT'93: Workshop on the Theory and Application of Cryptographic Techniques Lofthus, Norway, May 23–27, 1993 Proceedings 12*. Springer. 1994, S. 386–397 (siehe S. 49).

[52] Pascal Junod. „On the complexity of Matsui's attack“. In: *Selected Areas in Cryptography: 8th Annual International Workshop, SAC 2001 Toronto, Ontario, Canada, August 16–17, 2001 Revised Papers 8*. Springer. 2001, S. 199–211 (siehe S. 49).

[53] Ralph C Merkle und Martin E Hellman. „On the security of multiple encryption“. In: *Communications of the ACM* 24.7 (1981), S. 465–467 (siehe S. 49).

[54] Joan Daemen und Vincent Rijmen. *The design of Rijndael*. Bd. 2. Springer, 2002 (siehe S. 49).

[55] Andrey Bogdanov, Dmitry Khovratovich und Christian Rechberger. „Biclique cryptanalysis of the full AES“. In: *Advances in Cryptology–ASIACRYPT 2011: 17th International Conference on the Theory and Application of Cryptology and Information Security, Seoul, South Korea, December 4-8, 2011. Proceedings 17*. Springer. 2011, S. 344–371 (siehe S. 49).

[56] Kazumaro Aoki u. a. „Camellia: A 128-bit block cipher suitable for multiple platforms—design and analysis“. In: *Selected Areas in Cryptography: 7th Annual International Workshop, SAC 2000 Waterloo, Ontario, Canada, August 14–15, 2000 Proceedings 7*. Springer. 2001, S. 39–56 (siehe S. 50).

[57] Leibo Li u. a. „Meet-in-the-middle technique for truncated differential and its applications to CLEFIA and camellia“. In: *Fast Software Encryption: 22nd International Workshop, FSE 2015, Istanbul, Turkey, March 8-11, 2015, Revised Selected Papers*. Springer. 2015, S. 48–70 (siehe S. 50).

[58] Mitsuru Matsui. „New block encryption algorithm MISTY“. In: *Fast Software Encryption: 4th International Workshop, FSE'97 Haifa, Israel, Proceedings 4*. Springer. 1997, S. 54–68 (siehe S. 50).

[59] Achiya Bar-On und Nathan Keller. „A 2^{70} Attack on the Full MISTY1“. In: *Advances in Cryptology–CRYPTO 2016: 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part I*. Springer. 2016, S. 435–456 (siehe S. 50).

[60] Yosuke Todo. „Integral cryptanalysis on full MISTY1“. In: *Journal of Cryptology* 30.3 (2017), S. 920–959 (siehe S. 50).

[61] ETSI. *Universal Mobile Telecommunications System (UMTS); LTE; 3G Security; Specification of the 3GPP confidentiality and integrity algorithms; Document 2: Kasumi specification (3GPP TS 35.202 version 12.0.0 Release 12)*. https://www.etsi.org/deliver/etsi_ts/135200_135299/135202/07.00.00_60/ts_135202v070000p.pdf. Okt. 2014 (siehe S. 50).

[62] Deukjo Hong u. a. „HIGHT: A new block cipher suitable for low-resource device“. In: *Cryptographic Hardware and Embedded Systems – CHES 2006: 8th International Workshop, Yokohama, Japan. Proceedings 8*. Springer. 2006, S. 46–59 (siehe S. 50).

[63] Deukjo Hong, Bonwook Koo und Daesung Kwon. „Biclique attack on the full HIGHT“. In: *Information Security and Cryptology-ICISC 2011: 14th International Conference, Seoul, Korea, November 30-December 2, 2011. Revised Selected Papers 14*. Springer. 2012, S. 365–374 (siehe S. 50).

[64] Carlisle Adams. *RFC2144: The CAST-128 encryption algorithm*. https://www.rfc-editor.org/rfc/rfc2144. 1997 (siehe S. 50).

[65] Shaomei Wang, Tingting Cui und Meiqin Wang. „Improved Differential Cryptanalysis of CAST-128 and CAST-256“. In: *Information Security and Cryptology: 12th International Conference, Inscrypt 2016, Beijing, China, November 4-6, 2016, Revised Selected Papers*. Springer. 2017, S. 18–32 (siehe S. 50).

[66] HJ Lee u. a. *RFC4009:The SEED encryption algorithm*. https://www.rfc-editor.org/rfc/rfc4269. 2005 (siehe S. 50).

[67] Jaechul Sung. „Differential cryptanalysis of eight-round SEED“. In: *Information Processing Letters* 111.10 (2011), S. 474–478 (siehe S. 50).

[68] Andrey Bogdanov u. a. „PRESENT: An ultra-lightweight block cipher“. In: *Cryptographic Hardware and Embedded Systems-CHES 2007: 9th International Workshop, Vienna, Austria, September 10-13, 2007. Proceedings 9*. Springer. 2007, S. 450–466 (siehe S. 50).

[69] Céline Blondeau und Kaisa Nyberg. „Links between Truncated Differential and Multidimensional Linear Properties of Block Ciphers and Underlying Attack Complexities“. In: *Eurocrypt*. Bd. 14. Springer. 2014, S. 165–182 (siehe S. 50).

[70] Taizo Shirai u. a. *The 128-bit blockcipher CLEFIA, Fast Software Encryption-FSE 2007, LNCS 4593, pp: 181-195*. 2007. URL: https://dblp.uni-trier.de/rec/conf/fse/ShiraiSAMI07.html (siehe S. 50).

[71] Deukjo Hong u. a. „LEA: A 128-bit block cipher for fast encryption on common processors“. In: *Information Security Applications: 14th International Workshop, WISA 2013, Jeju Island, Korea, August 19-21, 2013, Revised Selected Papers 14*. Springer. 2014, S. 3–27 (siehe S. 50).

[72] Ashutosh Dhar Dwivedi und Gautam Srivastava. „Differential cryptanalysis of round-reduced LEA“. In: *IEEE Access* 6 (2018), S. 79105–79113 (siehe S. 50).

[73] Whitfield Diffie und George Ledin. „SMS4 encryption algorithm for wireless networks“. In: *Cryptology ePrint Archive* (2008) (siehe S. 50).

[74] Yu Liu u. a. „New linear cryptanalysis of Chinese commercial block cipher standard SM4“. In: *Security and Communication Networks* 2017 (2017) (siehe S. 50).

[75] IA Zabotin, GP Glazkov und VB Isaeva. „Cryptographic protection for information processing systems, government standard of the USSR, GOST 28147-89“. In: *Government Committee of the USSR for Standards* (1989) (siehe S. 50).

[76] Nicolas T Courtois. „An improved differential attack on full GOST“. In: *The New Codebreakers: Essays Dedicated to David Kahn on the Occasion of his 85th Birthday*. Springer, 2016, S. 282–303 (siehe S. 50).

[77] Federal Agency on Technical Regulation und Metrology (GOST). *GOST R 34.12-2015: Block Cipher „Kuznyechik“*. https://www.rfc-editor.org/rfc/rfc7801 (siehe S. 50).

[78] Riham AlTawy und Amr M Youssef. „A meet in the middle attack on reduced round Kuznyechik“. In: *IEICE TRANSACTIONS on Fundamentals of Electronics, Communications and Computer Sciences* 98.10 (2015), S. 2194–2198 (siehe S. 50).

[79] Frederick J Bruwer, Willem Smit und Gideon J Kuhn. *Microchips and remote control devices comprising same*. US Patent 5,517,187. Mai 1996 (siehe S. 50).

[80] Sebastiaan Indesteege u. a. „A practical attack on KeeLoq“. In: *Advances in Cryptology–EUROCRYPT 2008: 27th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Istanbul, Turkey, April 13-17, 2008. Proceedings 27*. Springer. 2008, S. 1–18 (siehe S. 50).

[81] Ray Beaulieu u. a. „The SIMON and SPECK lightweight block ciphers“. In: *Proceedings of the 52nd annual design automation conference*. 2015, S. 1–6 (siehe S. 50).

[82] Huaifeng Chen und Xiaoyun Wang. „Improved linear hull attack on round-reduced Simon with dynamic key-guessing techniques“. In: *Fast Software Encryption: 23rd International Conference, FSE 2016, Bochum, Germany, March 20-23, 2016, Revised Selected Papers 23*. Springer. 2016, S. 428–449 (siehe S. 50).

[83] Ling Song, Zhangjie Huang und Qianqian Yang. „Automatic differential analysis of ARX block ciphers with application to SPECK and LEA“. In: *Information Security and Privacy: 21st Australasian Conference, ACISP 2016, Melbourne, VIC, Australia, July 4-6, 2016, Proceedings, Part II*. Springer. 2016, S. 379–394 (siehe S. 50).

[84] Shoji Miyaguchi. „The FEAL cipher family“. In: *Advances in Cryptology-CRYPTO'90: Proceedings 10*. Springer. 1991, S. 628–638 (siehe S. 50).

[85] Eli Biham und Adi Shamir. „Differential cryptanalysis of Feal and N-hash“. In: *Advances in Cryptology—EUROCRYPT'91: Workshop on the Theory and Application of Cryptographic Techniques Brighton, UK, April 8–11, 1991 Proceedings 10*. Springer. 1991, S. 1–16 (siehe S. 50).

[86] Bruce Schneier u. a. „Twofish: A 128-bit block cipher“. In: *NIST AES Proposal* 15.1 (1998), S. 23–91 (siehe S. 50).

[87] Stefan Lucks. „The saturation attack—a bait for Twofish“. In: *Fast Software Encryption: 8th International Workshop, FSE 2001 Yokohama, Japan. Revised Papers*. Springer. 2002, S. 1–15 (siehe S. 50).

[88] Mathy Vanhoef und Frank Piessens. „All your biases belong to us: Breaking rc4 in wpa-tkip and TLS“. In: *24th USENIX Security Symposium (USENIX Security 15)*. 2015, S. 97–112 (siehe S. 50).

[89] Marc Briceno. „A pedagogical implementation of A5/1“. In: *http://www. scard. org* (1995) (siehe S. 50).

[90] Elad Barkan, Eli Biham und Nathan Keller. „Instant ciphertext-only cryptanalysis of GSM encrypted communication“. In: *Advances in Cryptology-CRYPTO 2003: 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003. Proceedings 23*. Springer. 2003, S. 600–616 (siehe S. 50).

[91] Marc Briceno, Ian Goldberg und David Wagner. „A pedagogical implementation of the GSM A5/1 and A5/2 "voice privacy" encryption algorithms". In: *Originally published at http://www. scard. org, mirror at http://cryptome. org/gsm-a512. htm* 26 (1999) (siehe S. 50).

[92] Daniel J Bernstein u. a. „ChaCha, a variant of Salsa20". In: *Workshop record of SASC*. Bd. 8. 2008, S. 3–5 (siehe S. 50).

[93] Jean-Philippe Aumasson u. a. „New features of Latin dances: analysis of Salsa, ChaCha, and Rumba". In: *Fast Software Encryption (FSE): 15th International Workshop, 2008, Revised Selected Papers 15*. Springer. 2008, S. 470–488 (siehe S. 50).

[94] Daniel J Bernstein. „The Salsa20 family of stream ciphers". In: *New stream cipher designs: the eSTREAM finalists* (2008), S. 84–97 (siehe S. 50).

[95] Karsten Nohl. „Mifare, little security, despite obscurity". In: *the 24th Congress of the Chaos Computer Club in Berlin, December 2007*. 2007 (siehe S. 50).

[96] Nicolas T. Courtois, Karsten Nohl und Sean O'Neil. „Algebraic Attacks on the Crypto-1 Stream Cipher in MiFare Classic and Oyster Cards". In: *Cryptology ePrint Archive* (2008). URL: https://eprint.iacr.org/2008/166.pdf (siehe S. 50).

[97] Martin Hell u. a. „A stream cipher proposal: Grain-128". In: *2006 IEEE International Symposium on Information Theory*. IEEE. 2006, S. 1614–1618 (siehe S. 50).

[98] Ximing Fu u. a. „Determining the Nonexistent Terms of Non-linear Multivariate Polynomials: How to Break Grain-128 More Efficiently". In: *IACR Cryptol. ePrint Arch.* 2017 (2017), S. 412 (siehe S. 50).

[99] Christophe De Canniere und Bart Preneel. „Trivium". In: *New Stream Cipher Designs: The eSTREAM Finalists* (2008), S. 244–266 (siehe S. 50).

[100] Pierre-Alain Fouque und Thomas Vannet. „Improving key recovery to 784 and 799 rounds of Trivium using optimized cube attacks". In: *Fast Software Encryption: 20th International Workshop, FSE 2013, Singapore, March 11-13, 2013. Revised Selected Papers*. Springer. 2014, S. 502–517 (siehe S. 50).

[101] Martin Boesgaard u. a. „Rabbit: A new high-performance stream cipher". In: *Fast Software Encryption: 10th International Workshop, FSE 2003, Lund, Sweden, February 24-26, 2003. Revised Papers 10*. Springer. 2003, S. 307–329 (siehe S. 50, 51).

[102] Dai Watanabe u. a. „Update on enocoro stream cipher". In: *2010 International Symposium On Information Theory & Its Applications*. IEEE. 2010, S. 778–783 (siehe S. 50).

[103] Naoki Shibayama und Yasutaka Igarashi. „A New Higher Order Differential of Enocoro-128v2". In: *2021 Ninth International Symposium on Computing and Networking Workshops (CANDARW)*. IEEE. 2021, S. 379–384 (siehe S. 50).

[104] Patrik Ekdahl und Thomas Johansson. „A new version of the stream cipher SNOW". In: *Selected Areas in Cryptography: 9th Annual International Workshop, SAC 2002 St. John's, Newfoundland, Canada, August 15–16, 2002 Revised Papers 9*. Springer. 2003, S. 47–61 (siehe S. 51).

[105] Yuki Funabiki u. a. „Several MILP-aided attacks against SNOW 2.0". In: *Cryptology and Network Security: 17th International Conference, CANS 2018, Naples, Italy, September 30–October 3, 2018, Proceedings*. Springer. 2018, S. 394–413 (siehe S. 51).

[106] Dai Watanabe u. a. „A new keystream generator MUGI". In: *Fast Software Encryption: 9th International Workshop, FSE 2002 Leuven, Belgium, February 4–6, 2002 Revised Papers 9*. Springer. 2002, S. 179–194 (siehe S. 51).

[107] Dai Watanabe u. a. *MUGI psuedorandom number generator, self evaluation (2001)*. Techn. Ber. Available at http://www.sdl.hitachi.co.jp/crypto/mugi/index-e.html. Hitachi, Ltd, 2001 (siehe S. 51).

[108] ETSI/SAGE Specification. *Specification of the 3GPP Confidentiality and Integrity Algorithms 128-EEA3 & 128-EIA3. Document 2: ZUC Specification, Version: 1.6, (2011)*. 2011 (siehe S. 51).

[109] Ronald L Rivest, Adi Shamir und Leonard Adleman. „A method for obtaining digital signatures and public-key cryptosystems". In: *Communications of the ACM* 21.2 (1978), S. 120–126 (siehe S. 51).

[110] Fabrice Boudot u. a. „Comparing the difficulty of factorization and discrete logarithm: a 240-digit experiment". In: *Advances in Cryptology–CRYPTO 2020: 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, August 17–21, 2020, Proceedings, Part II 40*. Springer. 2020, S. 62–91 (siehe S. 51).

[111] Fabrice Boudot u. a. „The State of the Art in Integer Factoring and Breaking Public-Key Cryptography". In: *IEEE Security & Privacy* 20.2 (2022), S. 80–86 (siehe S. 51).

[112] Taher ElGamal. „A public key cryptosystem and a signature scheme based on discrete logarithms". In: *IEEE transactions on information theory* 31.4 (1985), S. 469–472 (siehe S. 51).

[113] Jeff Hoffstein u. a. „Practical lattice-based cryptography: NTRUEncrypt and NTRUSign". In: *The LLL Algorithm: Survey and Applications*. Springer, 2009, S. 349–390 (siehe S. 51).

[114] Nick Howgrave-Graham. „A hybrid lattice-reduction and meet-in-the-middle attack against NTRU". In: *Advances in Cryptology-CRYPTO 2007: 27th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2007. Proceedings 27*. Springer. 2007, S. 150–169 (siehe S. 51).

[115] Ximing Fu u. a. „A key-recovery attack on 855-round Trivium". In: *Advances in Cryptology–CRYPTO 2018: 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19–23, 2018, Proceedings, Part II 38*. Springer. 2018, S. 160–184 (siehe S. 51).

[116] Yonglin Hao u. a. „Observations on the dynamic cube attack of 855-round TRIVIUM from Crypto'18". In: *Cryptology ePrint Archive* (2018) (siehe S. 51).

[117] PassMark Software. *CPU Benchmarks. Intel Xeon Gold 6130, 2.10GHz*. URL: https://www.cpubenchmark.net/cpu.php?cpu=Intel+Xeon+Gold+6130+%40+2.10GHz&id=3126 (besucht am 13. 06. 2023) (siehe S. 51).

[118] Jeff Hoffstein u. a. „Choosing parameters for NTRUEncrypt". In: *Topics in Cryptology–CT-RSA 2017: The Cryptographers' Track at the RSA Conference 2017, San Francisco, CA, USA, February 14–17, 2017, Proceedings*. Springer. 2017, S. 3–18 (siehe S. 51).

[119] David Kahn. *The Codebreakers: The Comprehensive History of Secret Communication from Ancient Times to the Internet*. Scribner, 1996 (siehe S. 54).

[120] Elonka Dunin und Klaus Schmeh. *Codebreaking: A Practical Guide*. Expanded edition. No Starch Press, 2023. URL: https://codebreaking-guide.com/ (siehe S. 54).

[121] Simon Singh. *The Code Book: The Science of Secrecy from Ancient Egypt to Quantum Cryptography*. Anchor, 1999 (siehe S. 54).

[122] Klaus Schmeh. *Kryptographie – Verfahren, Protokolle, Infrastrukturen*. 6. Aufl. Sehr gut lesbares, aktuelles und umfangreiches Buch über Kryptografie. Geht auch auf praktische Probleme (wie Standardisierung oder real existierende Software) ein. dpunkt.verlag, 2016 (siehe S. 54).

[123] Christof Paar, Jan Pelzl und Tim Güneysu. *Understanding Cryptography – From Established Symmetric and Asymmetric Ciphers to Post-Quantum Algorithms*. 2. Aufl. Springer, 2024. URL: https://www.cryptography-textbook.com/ (siehe S. 54).

[124] David Wong. *Kryptografie in der Praxis – Eine Einführung in die bewährten Tools, Frameworks und Protokolle*. dpunkt.verlag, 2023. URL: https://dpunkt.de/produkt/kryptografie-in-der-praxis/ (siehe S. 54).

[125] Jean-Philippe Aumasson. *Serious Cryptography: A Practical Introduction to Modern Encryption*. No Starch Press, 2017. URL: https://books.google.de/books?id=hLcrDwAAQBAJ (siehe S. 54).

[126] Mark Stamp und Richard M. Low. *Applied Cryptanalysis: Breaking Ciphers in the Real World*. Wiley-IEEE Press, 2007. URL: https://www.cs.sjsu.edu/~stamp/crypto/ (siehe S. 54, 61).

[127] Jonathan Katz und Yehuda Lindell. *Introduction to Modern Cryptography*. 3. Aufl. CRC, 2020. URL: https://www.routledge.com/Introduction-to-Modern-Cryptography/Katz-Lindell/p/book/9780815354369 (siehe S. 54).

[128] Douglas R. Stinson. *Cryptography – Theory and Practice*. 3. Aufl. Chapman & Hall/CRC, 2006 (siehe S. 54).

[129] Raphael Chung-Wei Phan. „Mini Advanced Encryption Standard (Mini-AES): A Testbed for Cryptanalysis Students". In: *Cryptologia* 26.4 (2002), S. 283–306 (siehe S. 60, 62).

[130] Minh Van Nguyen. *Number Theory and the RSA Public Key Cryptosystem – An introductory tutorial on using SageMath to study elementary number theory and public key cryptography*. 2009. URL: https://faculty.washington.edu/moishe/hanoiex/Number%20Theory%20Applications/numtheory-crypto.pdf (siehe S. 60).

[131] Minh Van Nguyen. *Exploring Cryptography Using the Sage Computer Algebra System*. 2009. URL: https://www.sagemath.org/files/thesis/nguyen-thesis-2009.pdf URL2: https://www.sagemath.org/library-publications.html (siehe S. 61).

[132] Bruce Schneier. „A Self-Study Course in Block-Cipher Cryptanalysis". In: *Cryptologia* 24 (2000), S. 18–34. URL: https://www.schneier.com/wp-content/uploads/2015/01/paper-self-study.pdf (siehe S. 61).

[133] Edward F. Schaefer. *Cryptography Research: Devising a Better Way to Teach and Learn the Advanced Encryption Standard*. 2011. URL: https://web.archive.org/web/20110829213229/http://www.scu.edu/cas/research/cryptography.cfm (siehe S. 62).

[134] Edward F. Schaefer. „A Simplified Data Encryption Standard Algorithm". In: *Cryptologia* 20.1 (1996), S. 77–84 (siehe S. 62).

[135] Raphael Chung-Wei Phan. „Impossible differential cryptanalysis of Mini-AES". In: *Cryptologia* (2003). URL: https://www.tandfonline.com/doi/abs/10.1080/0161-110391891964 (siehe S. 62).

[136] Mohammad A. Musa, Edward F. Schaefer und Stephen Wedig. *A simplified AES algorithm and its linear and differential cryptanalyses*. In: *Cryptologia* Bd. 17.2 (Apr. 2003), S. 148–177 (siehe S. 62).

[137] Nick Hoffman. *A SIMPLIFIED IDEA ALGORITHM*. 2006. URL: https://www.nku.edu/~christensen/simplified%20IDEA%20algorithm.pdf (siehe S. 62).

[138] S. Davod Mansoori und H. Khaleghei Bizaki. „On the vulnerability of Simplified AES Algorithm Against Linear Cryptanalysis". In: *IJCSNS International Journal of Computer Science and Network Security* 7.7 (2007), S. 257–263. URL: http://paper.ijcsns.org/07_book/200707/20070735.pdf (siehe S. 62).

2 Papier- und Bleistift-Chiffren und Vor-Computer-Verschlüsselungsverfahren

Unter dem Begriff Papier- und Bleistiftverfahren (P&B-Verfahren) lassen sich alle Verfahren zusammenfassen, die Menschen von Hand anwenden können, um Nachrichten zu ver- und entschlüsseln. Dazu gehören alle klassischen Chiffrier-Verfahren (im Gegensatz zu solchen, die Maschinen oder Computer erfordern), aber auch einige neuere, die bewusst mit dem Ziel entwickelt wurden, auch von Hand eine sehr hohe Sicherheit zu erreichen (was nicht immer gelang) – z. B. ElsieFour, Solitaire, Hutton oder Handycipher. Populär waren P&B-Verfahren auch bei Geheimdiensten, da ein Schreibblock und ein Stift unverdächtig sind.

Dieses Kapitel bietet einen recht vollständigen Überblick über die meisten dieser P&B-Verfahren, jeweils mit einem Beispiel und weiteren Referenzen.[1] Das Unterkapitel 2.5 stellt Hagelin-Rotormaschinen vor – als Beispiel für elektro-mechanische Verschlüsselungsmaschinen, die bis in die 1970er-Jahre Verwendung fanden.

Im Anhang finden Sie Informationen zu NCID, einem KI-Programm, mit dem man aus dem Geheimtext den Typ des Verschlüsselungsverfahrens bestimmen kann, sofern der Geheimtext mit einem klassischen oder einem Rotor-Verfahren verschlüsselt wurde.

Am Ende dieses Kapitels (Anhang 2.9) finden Sie zu einigen Verfahren (wie Caesar, Atbash, monoalphabetische Substitution, Vigenère, Hill, Spalten-Transposition) Beispielcode, geschrieben für das Computer-Algebra-System **SageMath**.

Während dieses Kapitel die Verfahren aus Sicht der modernen Kryptografie beschreibt (und damit liegt der Geheimtext schon korrekt vor), wird in Kapitel 3 auf Seite 145 die Sicht eines Historikers eingenommen, der ein gefundenes Dokument erst transkribieren muss, bevor er es kryptografisch untersuchen kann.

Die ersten Papier- und Bleistiftverfahren entstanden bereits vor rund 3500 Jahren in Mesopotamien. Bei allen Papier- und Bleistiftverfahren handelt es sich um symmetrische Verfahren. Selbst in den ältesten Verschlüsselungsmethoden steckten schon die grundsätzlichen Konstruktionsprinzipien wie Transposition, Substitution, Blockbildung und deren Kombination. Daher lohnt es sich vor allem aus didaktischen Gesichtspunkten, diese alten Verfahren genauer zu betrachten.[2,3]

Erfolgreiche Verfahren mussten die gleichen Merkmale erfüllen wie moderne Verfahren:

- Vollständige Beschreibung, klare Regeln, ja fast Standardisierung (inkl. Sonderfällen).
- Gute Balance zwischen Sicherheit und Benutzbarkeit (denn zu kompliziert zu bedienende Verfahren waren fehlerträchtig oder unangemessen langsam).

Konvention
Wenn das Alphabet nur die 26 Buchstaben verwendet, schreiben wir im Folgenden den Klartext in Kleinbuchstaben und den Geheimtext in Großbuchstaben.

[1] Die Fußnoten dieses Kapitels zeigen, wie man die kryptografischen Verfahren mit den Offline-Programmen CrypTool 1 (**CT1**), CrypTool 2 (**CT2**) und JCrypTool (**JCT**) ausführen kann. Vergleiche die Anhänge A.2, A.3 und A.4. Viele der Verfahren lassen sich auch online im Browser durchführen, z. B. auf der Seite von CrypTool-Online (**CTO**). Vergleiche dazu den Anhang A.5 in diesem Buch.
Während die CrypTool-Webseiten und -Programme sowohl klassische als auch moderne Chiffren anbieten, gibt es auch mehrere Seiten, die sich mit großer Detailtiefe nur auf klassische Chiffren fokussieren – diese stehen oft in Verbindung mit der American Cryptogram Association (ACA) [1]: z. B. die Seiten von Bion [2] und Pilcrow [3].

[2] Recht vollständig und aus eher mathematischer Sicht werden die klassischen Verfahren in dem Manuskript von Klaus Pommerening [4] beschrieben.

[3] Eine große, online verfügbare Sammlung verschlüsselter, historischer Dokumente findet sich in der Datenbank des internationalen und interdisziplinären Projekts DECRYPT (https://de-crypt.org). Im DECRYPT-Projekt werden reale historische verschlüsselte Dokumente gesammelt und Tools entwickelt, mit denen bspw. Historiker aus einem Scan eines Manuskripts eine Transkription erstellen können, die dann automatisch entschlüsselt wird. Insgesamt wurden bisher über 7900 (Stand Juli 2024) historische Geheimtexte und Schlüssel in die Datenbank aufgenommen. Siehe Kapitel 3 auf Seite 145.

Die Buchstaben des Geheimtextes schreiben wir meist – wie historisch üblich – in 5-er Blöcken gruppiert. Man könnte o.E.d.A. für die Ausgabe auch eine andere (konstante) Blocklänge oder gar keine Trennung durch Leerzeichen wählen.

> *Nur Wenige kann man glauben machen, dass es keine leichte Sache ist, eine Geheimschrift zu erdenken, die sich der Untersuchung widersetzt. Dennoch kann man rundheraus annehmen, dass menschlicher Scharfsinn keine Chiffre erdenken kann, die menschlicher Scharfsinn nicht lösen kann.*
>
> Zitat 5: Edgar Allan Poe: A Few Words on Secret Writing, 1841

2.1 Transpositionsverfahren

Bei der Verschlüsselung durch Transposition bleiben die ursprünglichen Zeichen der Nachricht erhalten, nur ihre Anordnung im Klartext wird geändert (Transposition = Vertauschung). Es gibt kein Geheimtextalphabet.

Manchmal wird auch der Begriff **Permutation** verwendet, um zu beschreiben, wie Buchstaben, Buchstabengruppen, Zahlen oder Spalten des Klartextes vertauscht werden, z. B. nach der Regel $(1, 2, 3, 4, 5) \Leftrightarrow (3, 4, 2, 1, 5)$.

2.1.1 Einführende Beispiele

- **Gartenzaun-Chiffre** [5]: Die Buchstaben des Klartextes werden abwechselnd in zwei (oder mehr) Zeilen geschrieben, so dass ein Zickzack-Muster entsteht. Dann werden die Zeichen zeilenweise nacheinander ausgelesen. Dieses Verfahren ist eher eine Kinderverschlüsselung.[4]

  Ein Beispiel findet sich in Tabelle 2.1.

  Klartext: ein beispiel zur transposition

i	b	i	p	e	z	r	r	n	p	s	t	o							
e	n	e	s	i	l	u	t	a	s	o	i	i	n						

Tab. 2.1: Gartenzaun-Verschlüsselung

  Geheimtext: IBIPE ZRRNP STOEN ESILU TASOI IN

[4] - In CTO kann man sich diese Chiffre im Plugin „Jägerzaun (Railfence/Zick-Zack)“ ansehen: `https://www.cryptool.org/de/cto/railfence`. Hierbei wird nicht nur das Ergebnis sondern auch die grafische Zickzack-Darstellung ausgegeben.
- Dieses Verfahren kann auch per `CT1 Ver-/Entschlüsseln ▷ Symmetrisch (klassisch) ▷ Skytale / Gartenzaun` abgebildet werden.
- Einen Gartenzaun mit 2 Zeilen und Offset 1 kann man als einfache Spaltentransposition simulieren, z. B. über `CT1 Ver-/Entschlüsseln ▷ Symmetrisch (klassisch) ▷ Permutation`, indem man als Schlüssel `(B,A)` eingibt und ansonsten die Standardeinstellungen (nur 1 Permutation, in der man zeilenweise ein- und spaltenweise ausliest) lässt. So wurde es in Tabelle 2.1 gemacht.
Mit dem Schlüssel `(A,B)` würde man das Zickzack-Muster in Tabelle 2.1 so beginnen, dass der erste Buchstabe in der ersten statt in der zweiten Zeile steht. Normalerweise kann man Gartenzaun-Verschlüsselungen nicht durch eine einfache Spaltentransposition simulieren.

- **Skytale von Sparta**[5] [5]: Dieses Verfahren wurde wahrscheinlich das erste Mal um 600 v.Chr. benutzt und es wurde von dem griechischen Schriftsteller und Philosophen Plutarch (50-120 v.Chr.) zuerst beschrieben.
  Um einen Holzstab wird ein Streifen Papier o.ä. gewickelt. Dann wird darauf zeilenweise der Klartext geschrieben. Nach dem Abwickeln steht auf dem Streifen der Geheimtext. Zum Entschlüsseln braucht der Empfänger einen zuvor verabredeten Holzstab mit gleichem Durchmesser.

- **Schablonen-Chiffre / Grilles** [6] Sender und Empfänger benutzen die gleiche Schablone. In deren Löcher werden zeilenweise die Klartextzeichen geschrieben, die dann spaltenweise ausgelesen werden. Bleibt Klartext übrig, wird der Vorgang wiederholt, unter Umständen mit einer anderen Ausrichtung der Schablone.[6]

- **Fleißner-Schablone** [7]: Die Fleißner-Schablone wurde im Ersten Weltkrieg von deutschen Soldaten benutzt, sie ist ein Spezialfall der Schablonen-Chiffre und wurde 1881 von Eduard Fleißner von Wostrowitz erfunden.[7] Ein quadratisches Gitter dient als Schablone, wobei ein Viertel der Felder Löcher hat. Der erste Teil des Klartextes wird zeichenweise durch die Löcher auf ein Blatt Papier geschrieben, dann wird die Schablone um 90 Grad gedreht, und der zweite Teil des Textes wird auf das Papier geschrieben, usw. Die Kunst besteht in der richtigen Wahl der Löcher: Kein Feld auf dem Papier darf frei bleiben, es darf aber auch keines doppelt beschriftet werden. Der Geheimtext wird zeilenweise ausgelesen.

  In die Beispiel-Fleißner-Schablone in Tabelle 2.2 können 4 Mal je 16 Zeichen des Klartextes auf ein Blatt geschrieben werden (jedes Zeichen in einen der Kreise, die ein gestanztes Loch symbolisieren sollen).

O	-	-	-	-	O	-	-
-	-	-	O	O	-	-	O
-	-	-	O	-	-	O	-
-	-	O	-	-	-	-	-
-	-	-	-	O	-	-	-
O	-	O	-	-	-	O	-
-	O	-	-	-	-	-	O
-	-	-	O	O	-	-	-

Tab. 2.2: 8x8-Fleißner-Schablone

[5] - In CTO kann man sich diese Chiffre im Plugin „Skytale“ ansehen: `https://www.cryptool.org/de/cto/scytale`. Hierbei wird auch die grafische Anordnung mit angezeigt.
- Dieses Verfahren kann auch mit dem Menüpunkt `CT1 Ver-/Entschlüsseln ▷ Symmetrisch (klassisch) ▷ Skytale / Gartenzaun` abgebildet werden. Da dieses Verschlüsselungsverfahrens ein Spezialfall der einfachen Spaltentransposition ist, kann man es per `CT1 Ver-/Entschlüsseln ▷ Symmetrisch (klassisch) ▷ Permutation` simulieren: Für die Skytale braucht man in der Dialogbox nur die erste Permutation. Darin gibt man bei z. B. 4 Kanten als Schlüssel `1,2,3,4` ein. Dies wäre so, als würde man den Text in 4-er Blöcken waagrecht in eine Tabelle schreiben und senkrecht auslesen. Weil der Schlüssel aufsteigend geordnet ist, bezeichnet man die Skytale auch als identische Permutation. Und weil das Schreiben und Auslesen nur einmal durchgeführt wird, als einfache (und nicht als doppelte) Permutation.
- Die Skytale findet sich auch in `CT2 Startcenter ▷ Vorlagen ▷ Kryptografie ▷ Klassisch`.

[6] Dieses Verfahren kann man nicht durch eine einfache Spaltentransposition darstellen.

[7] Unter `JCT Standard-Perspektive ▷ Analysen ▷ Fleißner-Grille-Analyse` können Texte nicht nur mit der Fleißner-Schablone ver- und entschlüsselt werden, sondern auch Geheimtexte angegriffen werden. Eine weitere gute Visualisierung findet sich archiviert unter `https://web.archive.org/web/20050922123752/http://www.turning-grille.com:80/`.

2.1.2 Spalten- und Zeilentransposition[8]

- **Einfache Spaltentransposition** [7]: Zunächst wird ein Schlüsselwort bestimmt, das über die Spalten eines Gitters geschrieben wird. Dann schreibt man den zu verschlüsselnden Text zeilenweise in dieses Gitter. Die Spalten werden entsprechend des Auftretens der Buchstaben des Schlüsselwortes im Alphabet durchnummeriert. In dieser Reihenfolge werden nun auch die Spalten ausgelesen und so der Geheimtext gebildet.[9]

  Siehe Tabelle 2.3.

  Klartext: ein beispiel zur transposition

K	E	Y
e	i	n
b	e	i
s	p	i
e	l	z
u	r	t
r	a	n
s	p	o
s	i	t
i	o	n

Tab. 2.3: Einfache Spaltentransposition

  Transpositionsschlüssel: K=2; E=1; Y=3.

  Geheimtext: IEPLR APIOE BSEUR SSINI IZTNO TN

- **AMSCO-Chiffre** [1]: Die Klartextzeichen werden abwechselnd in Einer- und Zweiergruppen in ein Gitter geschrieben. Dann erfolgt eine Vertauschung der Spalten, anschließend das Auslesen.
- **Doppelte Spaltentransposition (DST) / „Doppelwürfel“** [7]: Die doppelte Spaltentransposition wurde im Zweiten Weltkrieg und zu Zeiten des Kalten Krieges angewendet. Dabei werden zwei Spaltentranspositionen nacheinander durchgeführt; für die zweite Transposition wird ein neuer Schlüssel benutzt.[10]
  Werden zwei unterschiedliche und genügend lange Schlüssel (mindestens je 20 Zeichen) verwendet, dann ist das auch für heutige Computer noch eine Herausforderung.[11]

[8] Die meisten der folgenden Verfahren können auch in CT abgebildet werden:
- In CTO im Plugin „Einfache Spalten-Transposition“: `https://www.cryptool.org/de/cto/transposition`.
- Über `CT1` `Ver-/Entschlüsseln ▷ Symmetrisch (klassisch) ▷ Permutation`.
- Über `CT2` `Vorlagen ▷ Kryptografie ▷ Klassisch ▷ Transpositions-Chiffre`. Zusätzlich enthält CT2 einige Analyseverfahren für diese Chiffre.
- Über `JCT` `Standard-Perspektive ▷ Algorithmen ▷ Klassisch ▷ Transposition`.

[9] Darstellung mit CT1: Eingabe eines Schlüssels für die erste Permutation, zeilenweise einlesen, spaltenweise permutieren und auslesen. Die Komponente in CT2 animiert, wie der Text in die Matrix ein- und ausgelesen wird und wie die Spalten vertauscht werden.

[10] Darstellung mit CT1 oder CT2: Eingabe eines Schlüssels für die erste Permutation, zeilenweise einlesen, spaltenweise permutieren und spaltenweise auslesen. Eingabe eines (neuen) Schlüssels für die zweite Permutation, Ergebnis der erste Permutation zeilenweise einlesen, spaltenweise permutieren und spaltenweise auslesen.

[11] In MTW finden sich dazu Challenges (die mit dem Namensteil „reloaded“ erklären auch die Bedingungen an die Schlüssel genauer), bspw. `https://mysterytwister.org/challenges/level-x/doppelwuerfel` und `https://mysterytwister.org/challenges/level-3/doppelwuerfel-reloaded-teil-1`

- **Spaltentransposition, General Luigi Sacco** [7]: Die Spalten eines Gitters werden den Buchstaben des Schlüsselwortes entsprechend nummeriert. Der Klartext wird dann zeilenweise eingetragen, in der ersten Zeile bis zur Spalte mit der Nummer 1, in der zweiten Zeile bis zur Spalte mit der Nummer 2 usw. Das Auslesen erfolgt wiederum spaltenweise.

  Siehe Tabelle 2.4.

  Klartext: ein beispiel zur transposition

G	E	N	E	R	A	L
4	2	6	3	7	1	5
e	i	n	b	e	i	
s	p					
i	e	l	z			
u						
r	t	r	a	n	s	p
o	s	i				
t	i	o	n			

Tab. 2.4: Spaltentransposition nach General Luigi Sacco

  Geheimtext: ESIUR OTIPE TSINL RIOBZ ANENI SP

- **Spaltentransposition, Französische Armee im Ersten Weltkrieg** [7]: Nach Durchführung einer Spaltentransposition werden diagonale Reihen ausgelesen.
- **Zeilentransposition** [7]: Der Klartext wird in gleich lange Blöcke zerlegt, was auch mithilfe eines Gitters erfolgen kann. Dann wird die Reihenfolge der Buchstaben bzw. der Spalten vertauscht. Da das Auslesen zeilenweise erfolgt, wird nur jeweils innerhalb der Blöcke permutiert.[12]

2.1.3 Weitere Transpositionsverfahren

- **Geometrische Figuren** [6]: Einem bestimmten Muster folgend wird der Klartext in ein Gitter geschrieben (Schnecke, Rösselsprung o.ä.), einem zweiten Muster folgend wird der Geheimtext ausgelesen.
- **Union Route Cipher** [6]: Die Union Route Cipher hat ihren Ursprung im Amerikanischen Bürgerkrieg. Nicht die Buchstaben einer Nachricht werden umsortiert, sondern die Wörter. Für besonders prägnante Namen und Bezeichnungen gibt es Codewörter, die zusammen mit den Routen in einem Codebuch festgehalten werden. Eine Route bestimmt die Größe des Gitters, in das der Klartext eingetragen wird und das Muster, nach dem der Geheimtext ausgelesen wird. Zusätzlich gibt es eine Anzahl von Füllwörtern.
- **Nihilist-Transposition** [1]: Der Klartext wird in eine quadratische Matrix eingetragen, an deren Seiten jeweils der gleiche Schlüssel steht. Anhand dieses Schlüssels werden sowohl die Zeilen als auch die Spalten alphabetisch geordnet und der Inhalt der Matrix dementsprechend umsortiert. Der Geheimtext wird zeilenweise ausgelesen.

[12] Darstellung mit CT1: Eingabe eines Schlüssels für die erste Permutation, zeilenweise einlesen, spaltenweise permutieren und zeilenweise auslesen. Die Komponente in CT2 kann ebenfalls die zeilenweise Transposition visualisieren.

Siehe Tabelle 2.5: Die linke Matrix der Tabelle ist das Resultat nach dem Einlesen des Klartextes. Die rechte Matrix ist das Resultat nach dem Vertauschen von Zeilen und Spalten.

Klartext: `ein beispiel zur transposition`

	K	A	T	Z	E		A	E	K	T	Z
K	e	i	n	b	e	A	s	e	i	p	i
A	i	s	p	i	e	E	s	i	o	i	t
T	l	z	u	r	t	K	i	e	e	n	b
Z	r	a	n	s	p	T	z	t	l	u	r
E	o	s	i	t	i	Z	a	p	r	n	s

Tab. 2.5: Nihilist-Transposition

Geheimtext: `SEIPI SIOIT IEENB ZTLUR APRNS`

- **Cadenus-Chiffre** [1]: Cadenus ist eine Form einer Spaltentransposition, die zwei Schlüsselwörter verwendet. Cadenus kombiniert eine vollständige Spaltentransposition mit einer Schlüssel-Spaltenrotation. Das Verfahren – sowohl bei der Verschlüsselung als auch bei der Entschlüsselung – ist recht fehleranfällig.

  Nehmen wir an, das 1. Schlüsselwort (*KEY*) hat eine Länge von $n = 3$. Die maximale Länges des 2. Schlüsselworts ist die Länge des Alphabets. Es kann kürzer sein, muss aber die Buchstaben des 1. Schlüsselworts enthalten. Das 2. Schlüsselwort ist eine Permutation der verwendeten Elemente. Wir haben eine Tabelle mit einer ersten Spalte (*Vorspalte*) und drei Blöcken, die jeweils aus n Spalten bestehen. Das 1. Schlüsselwort wird in den Tabellenkopf jedes Blocks geschrieben und dazu verwendet, um später die Spalten des zweiten und dritten Blocks zu vertauschen. Das 2. Schlüsselwort wird verwendet, um die Vorspalte zu erstellen und so den Anfangsbuchstaben jeder Spalte des dritten Blocks zu definieren.

  Der Klartext wird im ersten Block zeilenweise eingetragen. Dann wird jede Spalte des ersten Blocks in den zweiten Block verschoben, transponiert in der Reihenfolge des 1. Schlüsselworts. Die Spalten des dritten Blocks werden erstellt, indem jede Spalte von Block 2 nach Block 3 kopiert und die Elemente innerhalb einer Spalte wrap-around so verschoben werden, dass sie mit dem Buchstaben beginnen, der in derselben Zeile steht wie der entsprechende Schlüsselbuchstabe des 1. Schlüsselwort innerhalb des zweiten Schlüsselworts. Der Geheimtext wird dann aus dem dritten Block zeilenweise ausgelesen.

  Siehe Tabelle 2.6 auf der nächsten Seite: Die Buchstaben des 1. Schlüsselwortes sind in der Vorspalte **grau** hinterlegt. Innerhalb des 2. Blocks werden diese Buchstaben **blau** hinterlegt und stehen dann in Block 3 oben in den Spalten (nach der wrap-around Rotation der entsprechenden Spalten). Das Beispiel hier ist eine leicht erweiterte Version von Cadenus, da die Länge des 2. Schlüsselworts nicht genau 25 betragen muss. Aber trotzdem muss die Klartextlänge (hier 48 mit dem Füller „xx“) ein Vielfaches der Länge des 2. Schlüsselwortes (hier 16) sein.

  Klartext: `ein laengeres beispiel zur transposition mit cadenusxx`

  1. Schlüsselwort: `KEY`
  2. Schlüsselwort: `ADXKCWNSYEDTUBRG`

	K	**E**	Y	**E**	K	Y	**E**	K	Y
A	e	i	n	i	e	n	**p**	**r**	**n**
D	l	a	e	a	l	e	i	b	o
X	n	g	e	g	n	e	o	s	t
K	r	e	s	e	**r**	s	i	e	n
C	b	e	i	e	b	i	a	u	t
W	s	p	i	p	s	i	n	r	d
N	e	l	z	l	e	z	x	s	u
S	u	r	t	r	u	t	i	s	x
Y	r	a	n	a	r	**n**	a	i	n
E	s	p	o	**p**	s	o	g	m	e
D	s	i	t	i	s	t	e	c	e
T	i	o	n	o	i	n	e	e	s
U	m	i	t	i	m	t	p	s	i
B	c	a	d	a	c	d	l	e	i
R	e	n	u	n	e	u	r	l	z
G	s	x	x	x	s	x	a	n	t

Tab. 2.6: Cadenus-Chiffre

Geheimtext: `PRNIB OOSTI ENAUT NRDXS UISXA INGME ECEEE SPSIL EIRLZ ANT`

2.2 Substitutionsverfahren

Substitutionsverfahren ordnen jedem Klartextobjekt (d. h. jedem Element des Klartextalphabets bzw. den Elementen der Nomenklatur) ein Geheimtextobjekt (d. h. ein Geheimtextzeichen oder den entsprechenden Nomenklatur-Code) zu.

Bei monoalphabetischen Substitutionsverfahren bleibt die einmal getroffene Zuordnung während des ganzen Verschlüsselungsprozesses fest – im Gegensatz zur **polyalphabetischen** Substitution. Behandelt man mehr als einen Buchstaben als ein Objekt, nennt man die Substitution **polygrafisch**. Kann man ein Objekt im Klartext-Alphabet auf mehr als ein Objekt im Geheimtext-Alphabet abbilden, nennt man sie **homophon**. Kann man ein Objekt im Geheimtext-Alphabet auf mehr als ein Objekt im Klartext-Alphabet abbilden, heißt sie **polyphon** – dies wurde in der Praxis kaum verwendet, weil dabei die Entschlüsselung nicht mehr eindeutig ist. Aber es ist ein von der American Cryptogram Association (ACA) benutzer Chiffre-Typ (dort als „Key-Phrase Cipher“ bezeichnet).

2.2.1 Monoalphabetische Substitution

Monoalphabetische Substitutionsverfahren (MASC) ordnen jedem Klartextzeichen ein Geheimtextzeichen fest zu, d. h. diese Zuordnung ist während des ganzen Verschlüsselungsprozesses dieselbe. In der Literatur wird das noch feiner unterschieden: Einfache MASC operieren auf Einzelbuchstaben (monografische Verschlüsselung), während MASC auch auf Gruppen operieren können.

- **Allgemeine monoalphabetische Substitution / Zufällige Buchstabenzuordnung**[13] [5]: Die Substitution erfolgt aufgrund einer festgelegten Zuordnung der Einzelbuchstaben. Das ist die übliche Beschreibung: Der Schlüssel ist dann eine Permutation[14] des Klartext-Alphabets. Diese Permutation wird als Geheimtext-Alphabet benutzt. Dieser allgemeine Fall hat mit 26! Möglichkeiten einen deutlich größeren Schlüsselraum als Caesar.
- **Atbash-Chiffre**[15] [5]: Der erste Buchstabe des Alphabets wird durch den letzten Buchstaben des Alphabets ersetzt, der zweite durch den vorletzten, usw. Atbash selbst ist eine feste Methode ohne Schlüssel. Sieht man es als MASC, wäre der Schlüssel eine Umkehrung des Alphabets.
- **Verschiebe-Chiffre / Caesar**[16] [5]: Das Geheimtextalphabet wird erzeugt, indem man das ganze Klartextalphabet um eine bestimmte Anzahl von Zeichen verschiebt. Der Verschiebungswert ist kleiner als die Länge des Alphabets. Bemerkung: Der römische Herrscher Caesar hat immer fix um 3 Stellen verschoben.

  Klartext:
  `bei der caesarchiffre wird um n stellen verschoben`

  Geheimtext:
  `EHL GHU FDHVDUFKLIIUH ZLUG XP q VWHOOHQ YHUVFKREHQ`
- **Affine Chiffre**[17]: Dies ist eine Verallgemeinerung der Verschiebe-Chiffre. Jeder Klartext-Buchstabe wird erst durch einen anderen ersetzt und das Ergebnis wird dann mit der Verschiebe-Chiffre verschlüsselt. Die Bezeichnung *affine Chiffre* kommt daher, dass man die Ver- und Entschlüsselung als affine bzw. lineare Funktion beschreiben kann.
- **Substitution mit Symbolen, z. B. Freimaurerchiffre** [5]: Ein Buchstabe wird durch ein Symbol ersetzt.
- **Varianten**: Füller, absichtliche Fehler [5].
- **Nihilist-Substitution**[18] [1]: Das Alphabet wird in eine 5x5-Matrix eingetragen und bei der Verarbeitung wird jedem Klartextbuchstaben die aus Zeilen- und Spaltennummer gebildete Zahl zugeordnet.

[13] Die monoalphabetische Substitution (MASC) findet sich in CT per:
- CTO-Plugin „Monoalphabetische Substitution“: `https://www.cryptool.org/de/cto/monoalpha`
- `CT1 Ver-/Entschlüsseln ▷ Symmetrisch (klassisch) ▷ Substitution / Atbash`
- `CT2 Vorlagen ▷ Kryptografie ▷ Klassisch`. In CT2 kann man einem Element auch mehr als **einen** Einzelbuchstaben zuordnen (Homophone). Entsprechende Analyzer finden sich unter `CT2 Vorlagen ▷ Kryptoanalyse ▷ Klassisch`.

[14] Da die MASC eine bijektive Abbildung zwischen den Objekten des Klartextalphabets und des Geheimtext-Alphabets ist, können wir im mathematischen Sinne den Begriff **Permutation** auch hier verwenden. Der Unterschied zu den Transpositions-Chiffren besteht darin, dass dort die Klartextbuchstaben selbst gemischt werden und kein Geheimtext-Alphabet existiert.

[15] Atbash findet sich in CT per:
- CTO-Plugin „ATBASH“: `https://www.cryptool.org/de/cto/atbash`
- `CT1 Ver-/Entschlüsseln ▷ Symmetrisch (klassisch) ▷ Substitution / Atbash`

[16] Caesar und der Spezialfall Rot13 (manchmal auch Rot-13) finden sich in CT per:
- CTO-Plugin „Caesar / Rot13“: `https://www.cryptool.org/de/cto/caesar`
- `CT1 Ver-/Entschlüsseln ▷ Symmetrisch (klassisch) ▷ Substitution / Caesar / ROT13`
- `CT1 Analyse ▷ Symmetrische Verschlüsselung (klassisch) ▷ Ciphertext only ▷ Caesar`
- `CT1 Einzelverfahren ▷ Visualisierung von Algorithmen ▷ Caesar`
- `CT2 Vorlagen ▷ Kryptografie ▷ Klassisch ▷ Caesar-Chiffre`.

Entsprechende Analyzer finden sich unter `CT2 Vorlagen ▷ Kryptoanalyse ▷ Klassisch`.

[17] - CTO-Plugin „Affin / Multiplikativ“: `https://www.cryptool.org/de/cto/multiplicative`
- Weitere Details zur affinen Chiffre und eine SageMath-Implementierung finden Sie in Abschnitt 2.9.2.3 auf Seite 123.
- In MTW findet sich dazu eine Challenge für Kinder: `https://mysterytwister.org/challenges/level-1/affine-codes--rechnen-modulo-n--erweiterter-euklid`.

[18] - Eine Animation zu diesem Nihilist-Verfahren: `CT1 Einzelverfahren ▷ Visualisierung von Algorithmen ▷ Nihilist`
- `CT2 Vorlagen ▷ Kryptografie ▷ Klassisch ▷ Nihilist-Chiffre`

Gebildet wird das Alphabet aus einem ersten Schlüsselwort (`SCHLUESSEL ==> SCHLUE`) und den restlichen Buchstaben des Alphabets in der normalen Ordnung. Dann wird ein zweites Schlüsselwort (`KEY`) gewählt und in die Kopfzeile einer zweiten Tabelle geschrieben. In diese Tabelle wird der Klartext zeilenweise eingetragen. Unter jedes Klartextzeichen wird der Geheimtext-Zahlenwert geschrieben – dieser Zahlenwert ist die Summe aus der Zahl des Klartextbuchstabens und der Zahl des Schlüsselwortbuchstabens. Bei Zahlen zwischen 100 und 110 wird die führende „1“ ignoriert, so dass jeder Buchstabe durch eine zweistellige Zahl repräsentiert wird.

Siehe Tabelle 2.7: Hier ist eine Beispielberechnung für den ersten Buchstaben im Klartext markiert: Da $E = 21$ und $K = 33$, wird e zu $33 + 21 = 54$.

Klartext: ein beispiel zur substitution

	1	2	3	4	5
1	S	C	H	L	U
2	**E**	A	B	D	F
3	G	I	**K**	M	N
4	O	P	Q	R	T
5	V	W	X	Y	Z

(a) Matrix

K	E	Y
(33)	(21)	(54)
e	i	n
(54)	(53)	(89)
b	e	i
(56)	(42)	(86)
s	p	i
(44)	(63)	(86)
e	l	z
(54)	(35)	(109)
u	r	s
(48)	(65)	(65)
u	b	s
(48)	(44)	(65)
t	i	t
(78)	(53)	(99)
u	t	i
(48)	(66)	(86)
o	n	
(74)	(56)	

(b) Tabelle

Tab. 2.7: Nihilist-Substitution

Geheimtext: 54 53 89 56 42 86 44 63 86 54 35 09 48 65 65 48 44 65 78 53 99 48 66 86 74 56

- **Codes** [5]: Im Laufe der Geschichte wurden immer wieder Codebücher verwendet. In diesen Büchern wird jedem möglichen **Wort** eines Klartextes ein Codewort, ein Symbol oder eine Zahl zugeordnet. Voraussetzung für eine erfolgreiche geheime Kommunikation ist, dass Sender und Empfänger exakt das gleiche Codebuch besitzen und die Zuordnung der Codewörter zu den Klartextwörtern nicht offengelegt wird.
- **Nomenklatur** [5]: Als Nomenklatur werden Techniken bezeichnet, die ein Verschlüsselungsverfahren mit einem Codebuch kombinieren. Meist basiert das Verschlüsselungssystem auf einem Geheimtextalphabet, mit dem ein Großteil der Nachricht chiffriert (via Substitution) wird. Für besonders häufig auftretende oder geheim zu haltende Wörter existieren eine begrenzte Anzahl von Codewörtern.

- **Landkarten-Chiffre** [8]: Diese Methode stellt eine Kombination aus Substitution und Steganographie[19] dar. Klartextzeichen werden durch Symbole ersetzt, diese werden nach bestimmten Regeln in Landkarten angeordnet.
- **Straddling Checkerboard**[20] [6]: Eine 3x10-Matrix wird mit den 26 Buchstaben des Alphabets und zwei beliebigen Sonderzeichen (oder Zahlen) gefüllt, indem zunächst die voneinander verschiedenen Zeichen eines Schlüsselwortes und anschließend die restlichen Buchstaben des Alphabets eingefügt werden. Die Spalten der Matrix werden mit den Ziffern 0 bis 9, die zweite und dritte Zeile der Matrix mit den Ziffern 1 und 2 nummeriert. Jedes Zeichen des Geheimtextes wird durch die entsprechende Ziffer bzw. das entsprechende Ziffernpaar ersetzt. Da die „1“ und die „2“ die ersten Ziffern der möglichen Ziffernkombinationen sind, werden sie nicht als einzelne Ziffern verwendet. Das Besondere, dass die Geheimtextzeichen mal 1-stellig und mal 2-stellig sind: Der Buchstabe „S“ wird zu „0“,„C“ wird zu „3“, aber „D“ wird zu „10“. Siehe Tabelle 2.8.

  Klartext: substitution bedeutet ersetzung

	0	1	2	3	4	5	6	7	8	9
	S	-	-	C	H	L	U	E	A	B
1	D	F	G	I	J	K	M	N	O	P
2	Q	R	T	V	W	X	Y	Z	.	/

Tab. 2.8: Straddling Checkerboard mit Passwort „Schluessel“

  Geheimtext: 06902 21322 23221 31817 97107 62272 27210 72227 61712

  Sehr auffällig ist die Häufigkeit der Ziffern „1“ und „2“, dies wird jedoch durch die folgende Variante behoben. Bemerkung: Nicht-Alphabetzeichen, wie das Leerzeichen, werden vor der Verschlüsselung herausgefiltert.

- **Straddling Checkerboard, Variante**[21] [6]: Diese Form des Straddling Checkerboards wurde von sowjetischen Spionen im Zweiten Weltkrieg entwickelt. Angeblich haben auch Ernesto (Ché) Guevara und Fidel Castro diese Chiffre zur geheimen Kommunikation benutzt.[22] Das Alphabet wird in ein

[19] Pure **Steganographie** versucht, die Existenz der Nachricht zu verbergen statt eine Nachricht zu verschlüsseln.
In CT2 sind verschiedene steganographische Techniken zu finden: z. B. Bild-Steganografie (mit BPCS und mit LSB), Text-Steganografie [mit Großbuchstaben (allein oder im Binärmodus), mit Buchstabenmarkierung, mit Leerzeichen von Nullbreite] oder Wasserzeichen (unsichtbar , robust oder sichtbar). Siehe z. B. `CT2 Vorlagen ▷ Steganografie ▷ Bild-Steganografie mit BPCS`

[20] `CT2 Vorlagen ▷ Kryptografie ▷ Klassisch ▷ Straddling Checkerboard`
Um das Beispiel in Tabelle 2.8 in CT2 auszuführen, ist in den Eingabe-Komponenten Folgendes einzugeben (Schlüssel und Alphabet definieren die Matrix des Checkerboards):
- Klartext: substitution bedeutet ersetzung
- Zeilen und Spalten: 12 0123456789
- Alphabet: abcdefghijklmnopqrstuvwxyz
- Schlüssel: schluessel [Die Komponente „Alphabet-Permutator“ erzeugt aus Alphabet und Schlüssel die Reihenfolge für die Matrix, die in Tabelle 2.8 zu sehen ist.]

[21] `CT2 Vorlagen ▷ Kryptografie ▷ Klassisch ▷ Straddling Checkerboard`. CT2 unterstützt alle drei Straddling Checkerboard-Varianten. Um das Beispiel in Tabelle 2.9 auf der nächsten Seite in CT2 auszuführen, ist in den Eingabe-Komponenten Folgendes einzugeben (wobei der Schlüssel „schluessel“ nötig war für den Aufbau des Gitters; das Matrixalphabet ergibt sich durch das Auslesen des Gitters):
- Klartext: substitution bedeutet ersetzung
- Zeilen und Spalten: 37 0123456789
- Matrixalphabet: sardtneijqycbkzhm.lfv/ugowpx [Matrixalphabet direkt als Eingabe statt Alphabet und Schlüssel wie im Beispiel zuvor]

[22] Zusätzlich benutzte Ché Guevara für seine Kommunikation mit Fidel Castro ein One-Time-Pad. Siehe Teil 3 aus der Artikelserie *RSA & Co. in der Schule: Moderne Kryptologie, alte Mathematik, raffinierte Protokolle*: [9], S. 52.

Gitter eingetragen (Spaltenanzahl = Länge des Schlüsselwortes). Das zusätzliche Gitter dient nur dazu, dass beim Eintragen in die Matrix die alphabetische Reihenfolge der Buchstaben verlorengeht. Dann werden zwei beliebige Ziffern als „reserviert“ festgelegt, die später die zweite und dritte Zeile einer 3x10-Matrix bezeichnen (in unserem Beispiel 3 und 7). Die acht häufigsten Buchstaben (ENIRSATD für die deutsche Sprache) bekommen zur schnelleren Chiffrierung die Ziffern 0 bis 9 zugewiesen, dabei werden die reservierten Ziffern nicht vergeben. Die übrigen Buchstaben werden in dem Gitter spaltenweise ausgelesen und zeilenweise in die Matrix übertragen.

Siehe Tabelle 2.9.

Klartext: substitution bedeutet ersetzung

Gitter

S	C	H	L	U	**E**
A	B	**D**	F	G	**I**
J	K	M	**N**	O	P
Q	**R**	**T**	V	W	X
Y	Z	.	/		

Matrix

	0	1	2	3	4	5	6	7	8	9
	S	**A**	**R**	-	**D**	**T**	**N**	-	**E**	**I**
3	J	Q	Y	C	B	K	Z	H	M	.
7	L	F	V	/	U	G	O	W	P	X

Tab. 2.9: Variante des Straddling Checkerboards

Geheimtext: 07434 05957 45976 63484 87458 58208 53674 675

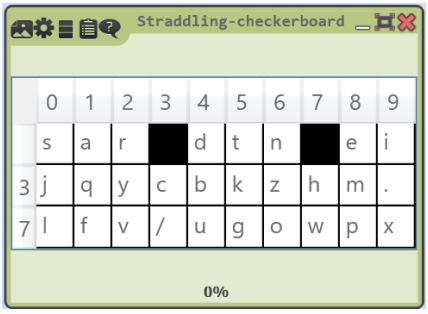


Abb. 2.1: CT2-Komponente für das Straddling-Checkerboard

– **Ché Guevara-Chiffre**[23]: Mit hoher Wahrscheinlichkeit benutzte Ché Guevara einen Spezialfall der obigen Checkerboard-Variante (mit einem zusätzlichen Substitutionsschritt und einem leicht modifizierten Checkerboard):
  * Die sieben häufigsten Buchstaben im Spanischen werden auf die erste Zeile verteilt.
  * Es werden vier statt drei Zeilen benutzt.
  * Damit konnte man $10 \cdot 4 - 3 = 37$ verschiedene Zeichen verschlüsseln.

[23] CT2 Vorlagen ▷ Kryptografie ▷ Klassisch ▷ Ché Guevara-Chiffre

- **Tri-Digital-Chiffre** [1]: Aus einem Schlüsselwort der Länge 10 wird ein numerischer Schlüssel gebildet, indem die Buchstaben entsprechend ihres Auftretens im Alphabet durchnummeriert werden. Dieser Schlüssel wird über ein Gitter mit zehn Spalten geschrieben. In dieses Gitter wird unter Verwendung eines Schlüsselwortes zeilenweise das Alphabet eingetragen, wobei die letzte Spalte frei bleibt. Die Klartextzeichen werden durch die Zahl über der entsprechenden Spalte substituiert, die Zahl über der freien Spalte dient als Trennzeichen zwischen den einzelnen Wörtern.

- **Bacon-Chiffre**[24] [1]: Dies ist eigentlich ein Code und keine Chiffre (es gibt keinen Schlüssel). Jedem Buchstaben des Alphabets und 6 Zahlen oder Sonderzeichen wird ein fünfstelliger Binärcode zugeordnet (z. B. 00000 = A, 00001 = B, usw.). Die Zeichen der Nachricht werden entsprechend ersetzt. Nun benutzt man eine zweite, unverdächtige Nachricht, um darin den Geheimtext zu verbergen. Dies kann z. B. durch Klein- und Großschreibung oder kursiv gesetzte Buchstaben geschehen: Man schreibt z. B. alle Buchstaben in der unverdächtigen Nachricht groß, die unter einer „1“ stehen. Insgesamt fällt das sicher auf.

  Siehe Tabelle 2.10.

Klartext/Nachricht	H	I	L	F	E
Zwischen-Geheimtext	`00111`	`01000`	`01011`	`00101`	`00100`
Unverdächtige Nachricht	`esist`	`warmu`	`nddie`	`sonne`	`scheint`
Geheimtext	`esIST`	`wArmu`	`nDdIE`	`soNnE`	`scHeint`

Tab. 2.10: Bacon-Chiffre

2.2.2 Homophone Substitution

Homophone Verfahren stellen eine Sonderform der monoalphabetischen Substitution dar. Einigen oder allen Klartextzeichen werden mehr als ein Geheimtextzeichen zugeordnet. Dieses Methodik war in vielen Abwandlungen in der historischen Wirklichkeit weit verbreitet, z. B. im Codex Copiale[25], einer verschlüsselten Handschrift aus dem 18. Jahrhundert (siehe Abb. 2.2 auf der nächsten Seite).

- **Homophone monoalphabetische Substitution**[26] [5]: Jede natürliche Sprache hat eine typische Häufigkeitsverteilung der Buchstaben im Alphabet. Um diese Verteilung zu verschleiern, werden einem Klartextbuchstaben mehrere Geheimtextzeichen fest zugeordnet. Die Anzahl der zugeordneten Zeichen richtet sich gewöhnlich nach der Häufigkeit des zu verschlüsselnden Buchstabens.
- **Buch-Chiffre**: Die Wörter eines Klartextes werden durch Zahlentripel der Form „Seite-Zeile-Position“ ersetzt. Diese Methode setzt eine genaue Absprache des verwendeten Buches voraus, so muss es sich insbesondere um die gleiche Ausgabe handeln (Layout, Fehlerkorrekturen, etc.).
- **Beale-Chiffre** [5]: Die Beale-Chiffre ist eine Buchchiffre, bei der die Wörter eines Schlüsseltextes durchnummeriert werden. Diese Zahlen ersetzen die Buchstaben des Klartextes durch die Anfangsbuchstaben der Wörter.

[24] `CT2 Vorlagen ▷ Kryptografie ▷ Klassisch ▷ Bacon-Chiffre`

[25] Siehe `https://en.wikipedia.org/wiki/Copiale_cipher`

[26] - Mittels `https://www.cryptool.org/de/cto/homophonic-substitution-analyzer/` kann das Verfahren in CTO angegriffen werden.
- `CT1 Ver-/Entschlüsseln ▷ Symmetrisch (klassisch) ▷ Homophone`
- `CT2 Vorlagen ▷ Kryptografie ▷ Klassisch ▷ Homophone Substitutions-Chiffre und Nomenklatur`. Zusätzlich enthält CT2 mächtige Analyseverfahren für diese Chiffre.

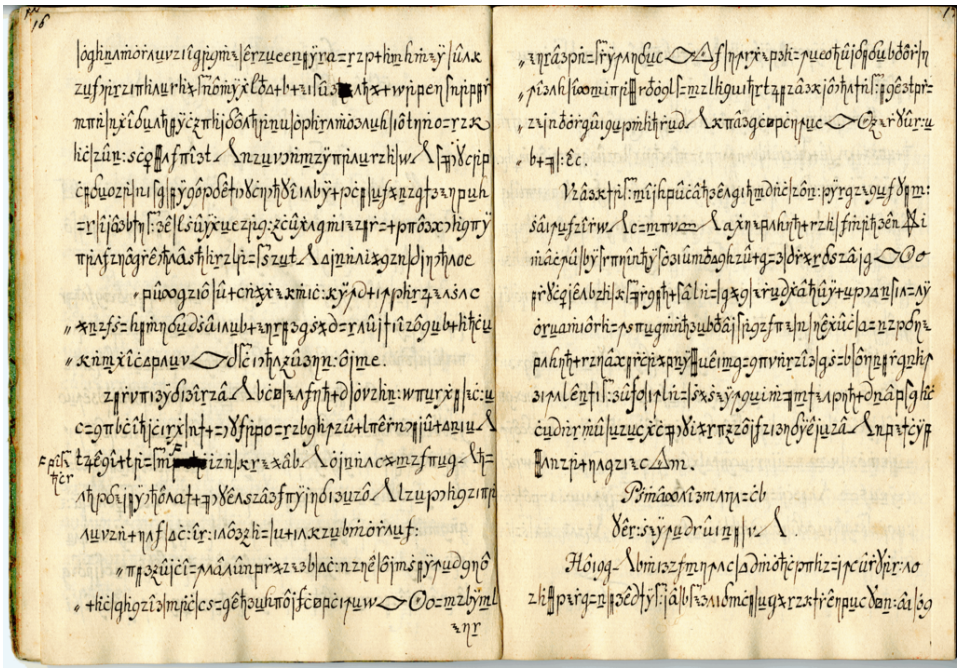


Abb. 2.2: Codex Copiale, skalierte Seite 16 und 17 (Quelle: [10])

- **Grandpré-Chiffre** [7]: Eine 10x10-Matrix (auch andere Größen sind möglich) wird mit zehn Wörter mit je zehn Buchstaben gefüllt, so dass die Anfangsbuchstaben ein elftes Wort ergeben. Da die Spalten und Zeilen mit den Ziffern 0 bis 9 durchnummeriert werden, lässt sich jeder Buchstabe durch ein Ziffernpaar darstellen. Es ist offensichtlich, dass bei 100 Feldern die meisten Buchstaben durch mehr als ein Ziffernpaar ersetzt werden können. Wichtig ist, dass die zehn Wörter alle Buchstaben des Alphabets enthalten.
- **Spanische Streifen-Chiffre (SSC)** [11]: Diese homophone Substitutions-Chiffre war die offizielle Verschlüsselungs-Methode der Spanischen Ministerien im späten 19. Jahrhundert und das Verfahren, das meistens im spanischen Bürgerkrieg (1936-1939) von beiden Seiten, den Republikanern und den Nationalisten, verwendet wurde.[27]

2.2.3 Polygrafische Substitution

Bei der polygrafischen Substitution werden nicht (nur) einzelne Buchstaben ersetzt, sondern (auch) Buchstabengruppen. Dabei kann es sich um Digramme, Trigramme, Silben, etc. handeln.[28]

[27] - CT2 Vorlagen ▷ Kryptografie ▷ Klassisch ▷ Spanische Streifen-Chiffre. Siehe Abb. 2.3 auf der nächsten Seite. CT2 kann das Verfahren mit dem Homophonen-Analyzer brechen.
- MTW: bspw. https://mysterytwister.org/challenges/level-2/spanische-strip-chiffre-teil-1
- Das Kopal-Video erklärt genau, wie die Schlüssel erzeugt werden und wie man dieses Verfahren mit CT2 bricht: https://www.youtube.com/watch?v=-C-hgnrUMKo

[28] Für die in der Häufigkeitsanalyse eingesetzten Buchstabenfolgen definierter Länge werden unterschiedliche Alternativbezeichnungen benutzt: 2-Gramm, Digramm, Bigramm und Digraph bedeuten alle daselbe. Wegen der besseren Lesbarkeit und der praktischen Vergleichbarkeit verschiedener Längen setzt sich langsam die Schreibweise n-Gramm durch.

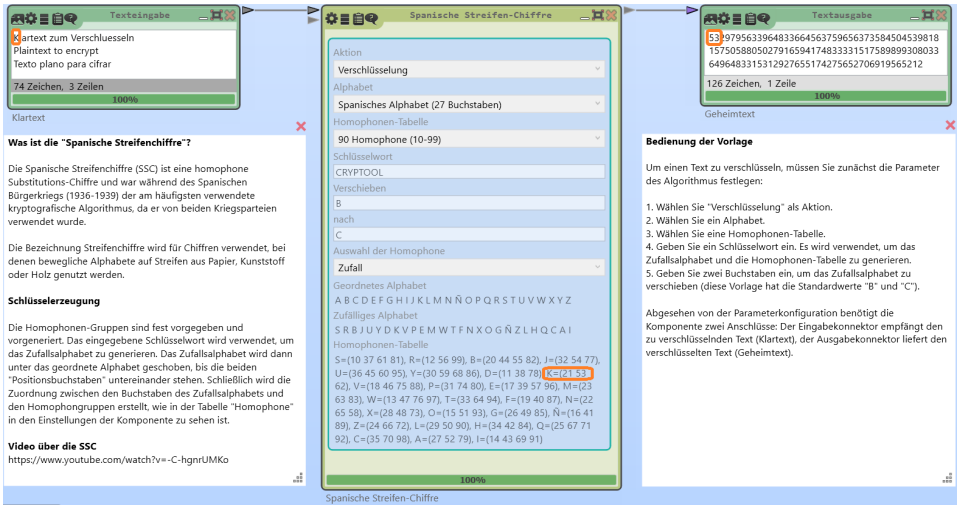


Abb. 2.3: CT2-Vorlage zur Spanischen Strip-Chiffre

- **Große Chiffre** [5]: Die Große Chiffre wurde von Ludwig XIV. im 17. Jahrhundert verwendet und erst kurz vor Beginn des 20. Jahrhunderts gebrochen. Die Kryptogramme enthielten 587 verschiedene Zahlen, jede Zahl repräsentierte eine Silbe. Die Erfinder dieser Chiffre (Rossignol, Vater und Sohn) hatten zusätzliche Fallen eingebaut, um die Sicherheit der Methode zu erhöhen. Eine Zahl konnte beispielsweise die vorangehende löschen oder ihr eine andere Bedeutung zuweisen. Siehe die „operationalen Code-Elemente“ in Kapitel 3 auf Seite 145.
- **Playfair-Chiffre**[29] [5]: Eine 5x5-Matrix wird mit dem Klartext-Alphabet (z. B. dem lateinischen Alphabet ohne das „J“) auf folgende Weise gefüllt: erst mit den verschiedenen Zeichen eines Schlüsselworts, dann mit den restlichen Buchstaben des Alphabets (ähnlich wie beim Straddling Checkerboard). Der Klartext (KT) wird in Digramme unterteilt, die nach den folgenden Regeln verschlüsselt werden:
  1. Befinden sich die Buchstaben in derselben Spalte, werden sie durch die Buchstaben ersetzt, die direkt darunter stehen.
  2. Befinden sich die Buchstaben in derselben Zeile, nimmt man jeweils den Buchstaben rechts vom Klartextzeichen.
  3. Befinden sich die Buchstaben in unterschiedlichen Spalten und Zeilen, nimmt man jeweils den Buchstaben, der zwar in derselben Zeile, aber in der Spalte des anderen Buchstabens steht.
  4. Für Doppelbuchstaben (falls sie in einem Digramm vorkommen) gelten Sonderregelungen, wie z. B. die Trennung durch einen Füller.

  Siehe Tabelle 2.11 auf der nächsten Seite.

  Unformatierter KT: `buchstaben werden paarweise verschluesselt`
  1. formatierter KT: `bu ch st ab en we rd en pa ar we is ev er sc hl ue ss elt`
  2. formatierter KT: `bu ch st ab en we rd en pa ar we is ev er sc hl ue sx se lt`

[29] - CT1 Ver-/Entschlüsseln ▷ Symmetrisch (klassisch) ▷ Playfair
- CT2 Vorlagen ▷ Kryptografie ▷ Klassisch ▷ Playfair-Chiffre. Zusätzlich enthält CT2 mächtige Analyseverfahren dafür.
- JCT Standard-Perspektive ▷ Algorithmen ▷ Klassisch ▷ Playfair

S	C	H	L	U
E	A	B	D	F
G	I	K	M	N
O	P	Q	R	T
V	W	X	Y	Z

Tab. 2.11: 5x5-Playfair-Matrix mit dem Passwort „Schluessel“

Geheimtext: FHHLU OBDFG VAYMF GWIDP VAGCG SDOCH LUSFH VEGUR

- **Playfair für Trigramme** [7]: Zunächst füllt man eine 5x5-Matrix mit dem Alphabet und teilt den Klartext in Trigramme auf. Für die Verschlüsselung gelten folgende Regeln:

  1. Drei gleiche Zeichen werden durch drei gleiche Zeichen ersetzt. Es wird der Buchstabe verwendet, der rechts unter dem ursprünglichen Buchstaben steht (Beispiel anhand von Tabelle 2.11: `BBB` ⇒ `MMM`).
  2. Bei zwei gleichen Buchstaben in einem Trigramm gelten die Regeln der Original-Playfair-Verschlüsselung.
  3. Bei drei unterschiedlichen Buchstaben kommen relativ komplizierte Regeln zur Anwendung, mehr dazu unter [7].

- **Ersetzung von Digrammen durch Symbole** [7]: Giovanni Battista della Porta, 15. Jahrhundert. Er benutzte eine 20x20-Matrix, in die er für jede mögliche Buchstabenkombination (das verwendete Alphabet bestand aus nur zwanzig Zeichen) ein Symbol eintrug.

- **Four Square-Chiffre** [7]: Diese Methode ähnelt Playfair, denn es handelt sich um ein Koordinatensystem, dessen vier Quadranten jeweils mit dem Alphabet gefüllt werden, wobei die Anordnung des Alphabets von Quadrant zu Quadrant unterschiedlich sein kann. Um eine Botschaft zu verschlüsseln, geht man wie folgt vor: Man sucht den ersten Klartextbuchstaben im ersten Quadranten und den zweiten Klartextbuchstaben im dritten Quadranten. Denkt man sich ein Rechteck mit den beiden Klartextbuchstaben als gegenüberliegende Eckpunkte, erhält man im zweiten und vierten Quadranten die zugehörigen Geheimtextzeichen.

  Siehe Tabelle 2.12: Hier sind jeweils die ersten beiden Klartext- und Geheimtextbuchstaben markiert.

  Klartext: buchstaben werden paarweise verschluesselt

d	w	x	y	m	E	P	T	O	L
r	q	e	k	i	C	V	I	Q	Z
u	v	h	p	s	R	M	A	G	U
a	l	**b**	z	n	F	W	**Y**	H	S
g	c	o	f	t	B	N	D	X	K
Q	T	B	L	E	v	q	i	p	g
Z	H	**N**	D	X	s	t	**u**	o	h
P	M	I	Y	C	n	r	d	x	y
V	S	K	W	O	b	l	w	m	f
U	A	F	R	G	c	z	k	a	e

Tab. 2.12: Four Square-Chiffre

Geheimtext: YNKHM XFVCI LAIPC IGRWP LACXC BVIRG MKUUR XVKT

- **Two Square-Chiffre / Doppelkasten** [7]: Die Vorgehensweise gleicht der der Four Square-Chiffre, allerdings enthält die Matrix nur zwei Quadranten. Befinden sich die beiden zu ersetzenden Buchstaben in der gleichen Reihe, werden sie nur vertauscht. Andernfalls werden die beiden Klartextzeichen als gegenüberliegende Eckpunkte eines Rechtecks betrachtet und durch die anderen Eckpunkte ersetzt. Die Anordnung der beiden Quadranten ist horizontal und vertikal möglich. Der Doppelkasten wurde von der Wehrmacht unter dem Namen „Truppenschlüssel“ verwendet.
- **Tri Square-Chiffre** [1]: Drei Quadranten werden jeweils mit dem Alphabet gefüllt. Der erste Klartextbuchstabe wird im ersten Quadranten gesucht und kann mit jedem Zeichen derselben Spalte verschlüsselt werden. Der zweite Klartextbuchstabe wird im zweiten Quadranten (diagonal gegenüberliegend) gesucht und kann mit jedem Buchstaben derselben Zeile verschlüsselt werden. Zwischen diese Geheimtextzeichen wird der Buchstabe des Schnittpunktes gesetzt.
- **Dockyard-Chiffre / Werftschlüssel** [7]: Angewendet von der deutschen Marine im Zweiten Weltkrieg.

2.2.4 Polyalphabetische Substitution

Bei der polyalphabetischen Substitution ist die Zuordnung Klartext-/Geheimtextzeichen nicht fest, sondern variabel (meist abhängig vom Schlüssel). Polyalphabetische Chiffren sind historisch weit bedeutender als polygrafische Chiffren.

Wenn man den Schlüssel wie einen Schlüsselstrom aus vorhandenen Klartext- oder Geheimtext-Teilen generiert, spricht man von Autokey-Verfahren.[30] Dabei macht ein fehlerhafter Buchstabe an einer beliebigen Stelle den gesamten nachfolgenden Text unbrauchbar. Solche Fehler können sowohl beim Verschlüsseler, bei der Übertragung, beim Empfang oder beim Entschlüsseler entstehen (also auch bei den Parteien, die den richtigen Schlüssel kennen).

- **Vigenère**[31] [5]: Entsprechend den Zeichen eines Schlüsselwortes wird jedes Klartextzeichen mit einem anderen Geheimtextalphabet verschlüsselt (als Hilfsmittel dient das sog. Vigenère-Tableau). Ist der Klartext länger als der Schlüssel, wird der Schlüssel (zyklisch) wiederholt.

  Siehe Tabelle 2.13 auf der nächsten Seite: Der erste Buchstabe des Klartextes und des Geheimtextes ist hier markiert.

  Vigenère-Varianten:

  – **Unterbrochener Schlüssel**: Der Schlüssel wird nicht fortlaufend wiederholt, sondern beginnt mit jedem neuen Klartextwort von vorne.

[30] https://en.wikipedia.org/wiki/Autokey_cipher

[31] - CTO in den Plugins „Vigenère“, „Autokey“, „Beaufort“, „Porta“ und „Trithemius“. Zusätzlich enthält CTO noch eine interaktive Analyse zur Bestimmung der Länge des Schlüssels und des kleinsten Autokorrelationswertes: https://www.cryptool.org/de/cto/vigenerebreak. Für die automatisierte Analyse wird die Autokorrelation verwendet: https://www.cryptool.org/de/cto/autocorrelation.
- CT1 Ver-/Entschlüsseln ▷ Symmetrisch (klassisch) ▷ Vigenère
Visualisiert in CT1 Einzelverfahren ▷ Visualisierung von Algorithmen ▷ Vigenère...
- CT2 Vorlagen ▷ Kryptografie ▷ Klassisch ▷ Vigenère. Zusätzlich enthält CT2 Analyseverfahren dafür.
- JCT Standard-Perspektive ▷ Algorithmen ▷ Klassisch ▷ Vigenère

Klartext	das	alphabet	wechselt	staendig
Schlüssel	KEY	KEYKEYKE	YKEYKEYK	EYKEYKEY
Geheimtext	NEQ	KPNREZOX	UOGFCIJD	WRKILNME

-	A	B	C	**D**	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	K	L	M	**N**	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
...	...	...																								

Tab. 2.13: Vigenère-Tableau

– **Autokey-Variante**[32] [7]: Nachdem der vereinbarte Schlüssel abgearbeitet wurde, geht man dazu über, die Zeichen der Nachricht als Schlüssel zu benutzen.

  Siehe Tabelle 2.14.

Klartext	das	alphabet	wechselt	staendig
Schlüssel	KEY	DASALPHA	BETWECHS	ELTSTAEN
Geheimtext	NEQ	DLHHLQLA	XIVDWGSL	WETWGDMT

Tab. 2.14: Autokey-Variante von Vigenère

– **Progressive-Key-Variante** [7]: Der Schlüssel ändert sich im Laufe der Chiffrierung, indem er das Alphabet durchläuft. So wird aus KEY LFZ.

– **Gronsfeld** [7]: Vigenère-Variante, die einen Zahlenschlüssel verwendet.

– **Beaufort**[33] [7]: Vigenère-Variante. Die Geheimtext-Alphabete werden rückwärts geschrieben (nicht das Klartext-Alphabet), dadurch wird Beaufort involutorisch.[34]

– **Porta** [1]: Vigenère-Variante, die nur 13 Alphabete verwendet. Das bedeutet, dass jeweils zwei Schlüsselbuchstaben dasselbe Geheimtextalphabet zugeordnet wird, und die erste und zweite Hälfte des Alphabets reziprok sind.

– **Slidefair** [1]: Kann als Vigenère-, Gronsfeld- oder Beaufort-Variante verwendet werden. Dieses Verfahren verschlüsselt Digramme. Den ersten Buchstaben sucht man im Klartextalphabet über dem Tableau, den zweiten in der Zeile, die dem Schlüsselbuchstaben entspricht. Diese beiden Punkte bilden gegenüberliegende Punkte eines gedachten Rechtecks, die verbleibenden Ecken bilden die Geheimtextzeichen.

[32] Autokey ist in den CT-Varianten meist auf dem gleichen Weg zu finden wie Vigenère.

[33] - In CTO im Plugin „Vigenère und Varianten“: https://www.cryptool.org/de/cto/vigenere
- In CT2: Teil der Vigenère-Komponente und -Vorlage (inklusive der Autokey-Variante)

[34] Eine Verschlüsselungsmethode ist involutorisch, wenn für das Ver- und Entschlüsseln das identische Verfahren verwendet werden kann. Beispiele: Rot13, Beaufort, XOR, Enigma.