Herrad Schmidt
Manfred Schwabl-Schmidt

AES und Rucksackverfahren

Theorie und Praxis mit AVR- und dsPIC-Mikrocontrollern

Springer Vieweg

AES und Rucksackverfahren

Lizenz zum Wissen.

Sichern Sie sich umfassendes Technikwissen mit Sofortzugriff auf tausende Fachbücher und Fachzeitschriften aus den Bereichen: Automobiltechnik, Maschinenbau, Energie + Umwelt, E-Technik, Informatik + IT und Bauwesen.

Exklusiv für Leser von Springer-Fachbüchern: Testen Sie Springer für Professionals 30 Tage unverbindlich. Nutzen Sie dazu im Bestellverlauf Ihren persönlichen Aktionscode **C0005406** auf *www.springerprofessional.de/buchaktion/*

Jetzt 30 Tage testen!

Springer für Professionals.
Digitale Fachbibliothek. Themen-Scout. Knowledge-Manager.

- Zugriff auf tausende von Fachbüchern und Fachzeitschriften
- Selektion, Komprimierung und Verknüpfung relevanter Themen durch Fachredaktionen
- Tools zur persönlichen Wissensorganisation und Vernetzung

www.entschieden-intelligenter.de

Springer für Professionals

Springer

Herrad Schmidt · Manfred Schwabl-Schmidt

AES und Rucksackverfahren

Theorie und Praxis mit AVR- und dsPIC-Mikrocontrollern

Springer Vieweg

Herrad Schmidt
Institut für Wirtschaftsinformatik
Universität Siegen
Siegen, Deutschland

Manfred Schwabl-Schmidt
Boppard, Deutschland

ISBN 978-3-658-19703-2 ISBN 978-3-658-19704-9 (eBook)
https://doi.org/10.1007/978-3-658-19704-9

Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.d-nb.de abrufbar.

Springer Vieweg

Gedruckt auf säurefreiem und chlorfrei gebleichtem Papier

Springer Vieweg ist Teil von Springer Nature
Die eingetragene Gesellschaft ist Springer Fachmedien Wiesbaden GmbH
Die Anschrift der Gesellschaft ist: Abraham-Lincoln-Str. 46, 65189 Wiesbaden, Germany

Vorwort

Wer das kryptographische Verfahren AES implementieren will oder soll, kann es sich sehr einfach machen, denn das Buch [DaRi] der beiden Erfinder des Verfahrens enthält eine komplette Realisierung. Soll die Implementierung jedoch für Mikrocontroller am unteren Ende des Leistungsspektrums durchgeführt werden, kommt man schnell in Schwierigkeiten, denn die Realisierung in [DaRi] ist ein C-Programm, das in die Mikrocontroller nicht hineinpasst.

Nun wäre es wohl möglich gewesen, an besagtem C-Programm so lange herumzubasteln, bis es doch passte, aber solch eine — kann man hier sagen — Strategie wäre doch sehr risikovoll gewesen: Änderungen an einem Programm vorzunehmen, dessen Funktion nicht genau bekannt ist war noch selten von Erfolg gekrönt. Der bessere Weg war sicherlich, vorab volles Verständnis des Verfahren zu erlangen und erst dann zu implementieren, und zwar direkt in die Assemblersprache der Mikrocontroller, um die speziellen Eigenschaften ihrer Befehlssätze ausnützen zu können und so auch Mikrocontroller mit geringen Ressourcen einsetzen zu können.

Allerdings verlangt ein „volles Verständnis des Verfahrens“ ein beträchtliches Mehr an Verständnis komplexer Mathematik als bei Programmierern von Mikrocontrollern gemeinhin vermutet werden kann. Aus diesem Grund hat das Buch in großen Teilen eine geschichtete Struktur. Und zwar sind drei Schichten vorhanden:

- Die obere Schicht enthält die Theorie des Verfahrens. Hier wird noch keine Rücksicht auf eine Implementierung genommen, im Gegenteil, das Verfahren wird in reiner abstrakter Form in mathematischer Terminologie dargestellt.
- Die mittlere Schicht stellt eine praktische Formulierung des Verfahrens dar, welche die Realisierung als Computerprogramm berücksichtigt. Beispielsweise werden Vektorraumhomomorphismen der oberen Schicht durch der Rechnung unmittelbar zugängliche Matrizen ersetzt.
- Die untere Schicht endlich enthält die Umsetzung in Programmcode mitsamt einer ausführlichen Erläuterung der gewählten Umsetzungsstrategien.

Die Umsetzung in Programmcode hängt natürlich stark davon ab, welcher Mikrocontroller das Ziel darstellt. Um diesen Effekt etwas zu mildern werden deshalb Programme für zwei Mikrocontroller vorgestellt, die sehr verschiedenen Welten angehören, nämlich AVR und dsPIC. Überraschenderweise ist der dsPIC, mit einer 16-Bit-CPU und einem großartigen Befehlssatz, dem 8-Bit-AVR-Controller mit seinem wenig Enthusiasmus hervorrufenden Befehlssatz nicht so weit überlegen wie man wohl annehmen konnte. Der Grund liegt darin, daß die beiden umgesetzten kryptographischen Verfahren auf der Maschinenbefehlsebene eine relativ einfache Struktur besitzen, bei der die Vorteile eines überlegenen Befehlssatzes nicht groß ausgespielt werden können.

Es sei an dieser Stelle vermerkt, daß die C-Programme in [DaRi] **keine** Grundlage für die Implementierung von AES gewesen sind.

Kein Thema des Buches sind die kryptographischen Aspekte von Verschlüsselungssystemen, also etwa ob es möglich ist, einen Geheimtext zu dechiffrieren, ohne den zur Chiffrierung verwendeten Schlüssel zu kennen, oder, falls es möglich sein sollte, welcher Aufwand dazu wohl nötig wäre. Allerdings reichen die Darstellung und Implementierung von AES für ein Buch mit anständigem Umfang nicht aus.

Hier bot sich an, zusätzlich ein Verfahren mit öffentlichem Schlüssel darzustellen und zu implementieren. Damit wäre dann auch allein auf der Mikrocontrollerebene die sichere Versendung von AES-Schlüsseln möglich.

Jedoch erfordern sowohl RSA und ähnliche Systeme als auch solche, die auf elliptischen Funktionen basieren, einen solch großen rechnerischen Aufwand, daß eine brauchbare Implementierung für die kleinen Mikrocontroller praktisch ausgeschlossen ist. Das bleibt selbst dann richtig, wenn die effizientesten Rechenverfahren eingesetzt werden, etwa die Durchführung von modularer Multiplikation mit Montgomery-Multiplikation. Das gilt nun nicht nur für die Verschlüsselung an sich, sondern beispielsweise auch für die Bereitstellung von Schlüsseln. So muß beispielsweise ein Paar von großen Primzahlen mit gewissen Eigenschaften erzeugt werden. Wie das Sprichwort es so schön sagt: Die Teufelchen verstecken sich in den Details.

Einige der Verfahren mit einem öffentlichen Schlüssel sind jedoch rechentechnisch noch praxisgeeignet zu bewältigen, und zwar sind das Verfahren, die auf dem Rucksackproblem basieren. Hier bereitet auch die Schlüsselerzeugung keine großen Probleme. Leider sind diese Chiffriersysteme nicht sicher (siehe z.B. [Sha]). Allerdings ist die Dechiffrierung eines Geheimtextes ohne den eingesetzten Schlüssel zu besitzen sehr aufwendig, und das gilt nicht nur für den Einsatz des Verfahrens, sondern auch für sein Verständnis. Zieht man daher den Einsatzzweck der im Buch behandelten Systeme in Betracht, nämlich die Verhinderung nicht autorisierten Gebrauches von Mikrocontrollersoftware, dann kann dieses Problem sicherlich ignoriert werden: Einem Raubkopierer von Mikrocontrollersoftware werden die benötigten Ressourcen wohl kaum zur Verfügung stehen.

Zum Schluss sei noch angemerkt, daß die Autoren bemüht waren, auch Querlesern einen leichten Zugang zu den verschiedenen Buchabteilungen zu ermöglichen, also etwa solchen Lesern, die an der theoretischen Darstellung nicht interessiert sind oder umgekehrt (wenn auch sehr unwahrscheinlich) nur an der Theorie interessiert sind. Das Buch enthält deshalb einiges an Redundanz, um das möglich zu machen.

Herrad Schmidt
Manfred Schwabl-Schmidt

Boppard, im Mai 2017

Inhaltsverzeichnis

1. Einleitung

Wie schon im Vorwort bemerkt wurde, sind nicht die kryptographischen Eigenschaften der behandelten Verschlüsselungssysteme das Thema des Buches, sondern ihre Darstellung und Implementierung. Allerdings ist dafür Sorge zu tragen, daß die Leser und die Autoren des Buches die kryptographische Begriffswelt in übereinstimmender Weise anwenden. Deshalb beginnt das Buch mit einer kurzen (aber wirklich kurzen) Einführung in die hier verwendeten Begriffe im Zusammenhang mit Verschlüsselungssystemen.

Weil diese Begriffe hauptsächlich in der theoretischen — weniger in der praktischen — Darstellung eine Rolle spielen, werden sie auf eine abstrakte, mathematische Weise eingeführt, was ebenfalls noch mit dazu beiträgt, Mißverständnisse zu vermeiden. Leser, die nur den praktischen Teil des Buches — also hauptsächlich die Implementierungen — zu nutzen gedenken, können die Einführung bedenkenlos übergehen.

Eine Ausnahme gibt es jedoch, nämlich das Beispiel des Verschlüsselungssystems von Hill in Abschnitt 2.2. Vom Anwender her gesehen besitzt es nämlich eine gewisse Ähnlichkeit mit dem System AES. Es ist allerdings weit weniger komplex und kann daher gut dazu dienen, sich auf AES einzustimmen, ohne mit seinen Schwierigkeiten konfrontiert zu werden. Das Hillsche System wird aber auch deshalb etwas ausführlicher behandelt, weil es sich durchaus in der Praxis einsetzen läßt. Es teilt mit AES zwar die unabdingbare Bedingung, den Schlüssel strikt geheim zu halten, bietet aber genug Sicherheit für Verschlüsselungen, die nur für eine kurze Lebensdauer gedacht sind.

In Kapitel 3 wird das Verschlüsselungssystem AES auf eine abstrakte, d.h. mathematische Weise dargestellt. Hier sollte der Leser mit der Theorie endlicher Körper und ihrer Polynomringe vertraut sein. Leser, welche diese Kenntnisse nicht oder nur andeutungsweise besitzen, finden beispielsweise in [HSMS] eine recht ausführliche Darstellung dieser Theorie.

Die Darstellung orientiert sich nur wenig an [DaRi]. So wird dort z.B. die Berechnung der Rundenschlüssel in einer Weise dargestellt, die praktisch direkt von einer Möglichkeit der Implementierung abgeleitet ist. Die Darstellung in Abschnitt 3.5 vermeidet dagegen eine solche Abhängigkeit, sie verwendet die Schlüssel vielmehr in ihrer reinen oder abstrakten Gestalt als interner Text. In der abstrakten Darstellung läßt sich auch sehr schön ableiten, daß die Strukturen von Chiffrierung und Dechiffrierung so ähnlich sind, daß beide mit einem nahezu identischen Programm implementiert werden können, siehe dazu Abschnitt 3.7.

Höhere Anforderungen an die mathematischen Fähigkeiten der Leser stellt der Abschnitt 3.1 über die Herleitung der S-Box und ganz besonders der Abschnitt 3.4 über die Spaltenmischung. Hier sei noch einmal auf [HSMS] verwiesen. Bei Durchsicht dieser Abschnitte wird besonders deutlich, daß die Sicherheit von AES auf der hoch nichtlinearen Multiplikation des eingesetzen endlichen Körpers $\mathbb{K}_{2^8}$ beruht und nicht auf der Anwendung von Permutationen wie bei DES oder *Twofish*.

Es folgt in Kapitel 4 eine Umformulierung von AES in eine mehr praktische Gestalt, die zur Grundlage der Implementierung im nächsten Kapitel wird. Hier wird vom Leser mathematisch nicht mehr abverlangt als mit Matrizen mit Koeffizienten im Körper $\mathbb{K}_{2^8}$ umgehen zu können. Dazu kann Abschnitt B.2 im Anhang Verwendung finden.

H. Schmidt, M. Schwabl-Schmidt, *AES und Rucksackverfahren*, https://doi.org/10.1007/978-3-658-19704-9_1

Die nächsten beiden Kapitel bringen die Implementierungen von AES, und zwar Kapitel 5 die für AVR-Mikrocontroller und Kapitel 6 diejenige für dsPIC-Mikrocontroller. Der Leser wird feststellen, daß für AVR wesentlich mehr Aufwand betrieben wird, um die Schwächen des Befehlssatzes zu umgehen. Das hat zur Folge, daß sich die Implementierungen nicht allzusehr unterscheiden, was die Effizienz der erzeugten Programme betrifft.

Schließlich wird das Thema AES abgeschlossen mit den Kapiteln 7 und 8, in welchen Verfahren beschrieben werden, die eine Schwäche von AES beseitigen, die bei der Chiffrierung langer Klartexte auftreten kann. Die Methode in Kapitel 8 kann auch dazu dienen, die *Block Cipher* AES in eine *Stream Cipher* zu verwandeln. Beide Verfahren werden allerdings nur für AVR implementiert.

Das nächste Kapitel 9 ist einer Verschlüsselung auf der Basis des Rucksackproblems gewidmet. Der theoretische Teil in Abschnitt 9.1 verwendet im Gegensatz zu AES nur relativ einfache Mathematik. Hier kann die Theorie ohne Umformungen direkt in die Implementierungen für AVR und dsPIC umgesetzt werden. Um den Abschnitt 9.2 für AVR nicht ausufern zu lassen wird dort eine etwas vereinfachte Version realisiert. Die Realisierung für den dsPIC-Mikrocontroller in Abschnitt 9.3 ist dagegen frei von Beschränkungen, soweit das bei einem Mikrocontroller eben möglich ist.

Weil der Körper $\mathbb{K}_{2^8}$ an zentraler Stelle des AES-Verfahrens steht, ist seiner Implementierung ein eigenes Kapitel 10 gewidmet. Implementiert wird für beide Mikrocontroller. Auch hier ist die Realisierung für AVR viel aufwendiger, die notwendigen Optimierungsmaßnahmen haben Beschränkungen für das erzeugte Programm zur Folge.

Es folgt in Kapitel A ein (wenn auch sehr unvollständiger) mathematischer Anhang, in dem zumindest ein Teil der im Buch verwendeten Mathematik vorgestellt wird. Es ist allerdings kein Ersatz für [HSMS]. Immerhin kann sich der Leser hier mit der mathematischen Symbolik vertraut machen, die im Buch durchgehend verwendet wird. Das gilt besonders für die Teilerrestfunktion und die Polynome und Polynomringe. Auch wird kurz die Konstruktion eines endlichen Körpers wie $\mathbb{K}_{2^8}$ vorgeführt.

Das Buch endet mit Kapitel B im Anhang, das einiges unzusammenhängende Material enthält. Für den Leser wohl am wichtigsten ist der Abschnitt B.4, in dem häufig verwendete Symbole und Bezeichnungen kurz erläutert werden.

Zum Abschluss noch eine Bemerkung zu den Implementierungen. Die Verfahren wurden für AVR direkt in die Assemblersprache der Prozessoren übertragen. Die beträchtliche (schlechte) Erfahrung der Autoren mit der Programmierung von AVR-Prozessoren führte dazu, daß gar nicht erst der Versuch gemacht wurde, einen der zur Verfügung stehenden C-Compiler zu verwenden. Denn eines der Ziele war es schließlich, umfangreiche Programme in möglichst kleine Mikrocontroller zu packen, von der Ausführungsgeschwindigkeit ganz zu schweigen.

Bei dsPIC-Controllern wurden zunächst einige C-Programme geschrieben und der erzeugte Maschinencode untersucht. Wie recht eigentlich erwartet war der Maschinencode streckenweise unbeschreiblich schlecht. Die Vorteile des sehr guten Befehlssatzes wurden nur minimal genutzt, so wurden z.B. komplexe aber dennoch schnelle Befehle im Code nicht verwendet, sondern mit Folgen einfacher Befehle simuliert. Folglich wurden auch beim dsPIC die Verfahren direkt in Assemblercode übersetzt.

2. Eine kurze Einführung in Chiffriersysteme

2.1. Definition eines Chiffriersystems

Als Erstes sollte präzisiert werden, was genau unter einer Folge von Elementen einer Menge M zu verstehen ist, denn solche Folgen sind Basisobjekte von Chiffriersystemen.

> Eine Folge von Elementen einer beliebigen Menge M ist eine Abbildung $\boldsymbol{f}\colon N \longrightarrow M$, wobei $N \subset \mathbb{N}$. Statt $\boldsymbol{f}(n)$ für $n \in N$ wird gewöhnlich $\boldsymbol{f}_n$ geschrieben, oder die Folge wird direkt mit ihren Folgegliedern als $(f_n)_{n\in N}$ bezeichnet.
> Die Menge aller Folgen von Elementen einer Menge M wird mit $\boldsymbol{\mathcal{F}}_M$ bezeichnet, d.h.
>
> $$\boldsymbol{\mathcal{F}}_M = \left\{ (f_n)_{n\in N} \;\middle|\; N \subset \mathbb{N} \;\wedge \bigwedge_{n\in N} f_n \in M \right\}$$
>
> Die Menge N ist die *Indexmenge*, ihre Elemente sind die *Indizes* der Folge.

Bei einer endlichen Indexmenge $N = \{n_1, \ldots, n_k\}$ kann eine Folge auch so bezeichnet werden, daß man die Folgeglieder nebeneinander in einer Reihe anschreibt, zur besseren Abgrenzung oft in Klammern gesetzt,

$$(f_{n_1}, \ldots, f_{n_k})$$

speziell bei $N = \{a, b, c, d\}$ also als (f_a, f_b, f_c, f_d). Spielt die Indexmenge keine Rolle, kommt es also nur auf die Reihenfolge der Folgenglieder an, können die Indizes auch unterdrückt werden, etwa in (o, p, q, r). Das ist also so zu verstehen, daß es eine Indexmenge $N = \{n_1, n_2, n_3, n_4\} \subset \mathbb{N}$ und eine Abbildung $\boldsymbol{f}\colon N \longrightarrow \{o, p, q, r\}$ gibt mit $\boldsymbol{f}_{n_1} = o$, $\boldsymbol{f}_{n_2} = p$, $\boldsymbol{f}_{n_3} = q$ und $\boldsymbol{f}_{n_4} = r$. Man läßt oftmals sogar die Kommata weg, wie in $(o\ p\ q\ r)$.

In der Kryptographie betrachtet man gerne Folgen von Elementen von $\mathbb{Z}_{26}$, dabei werden die $n \in \mathbb{Z}_{26}$ mit den Großbuchstaben des Alphabetes bezeichnet, meistens A = 0, B = 1 und so fort bis Z = 25. Texte in Großbuchstaben können dann als Folgen von Elementen aus $\mathbb{Z}_{26}$ aufgefasst werden. Der besseren Lesbarkeit wegen empfiehlt es sich jedoch, zu $\mathbb{Z}_{27}$ überzugehen und mit dem hinzugekommenen Element das Leerzeichen zu benennen, also $\sqcup = 26$. So wird beispielsweise der Text (die Zeichenfolge)

$$\texttt{EIN}\sqcup\texttt{KLARTEXT}$$

zu der folgenden Folge von Elementen von $\mathbb{Z}_{27}$:

$$4\ 8\ 13\ 26\ 10\ 11\ 0\ 17\ 19\ 4\ 23\ 19$$

Diese Zahlenfolge repräsentiert eine unendliche Anzahl von endlichen Folgen der Länge 12 von Elementen von $\mathbb{Z}_{27}$. Eine dieser Folgen ist $\boldsymbol{f}\colon \{2, 3, \ldots, 12, 13\} \longrightarrow \mathbb{Z}_{27}$ mit $\boldsymbol{f}_2 = 4$, $\boldsymbol{f}_3 = 8$ usw. bis $\boldsymbol{f}_{13} = 19$.

Heutzutage werden allerdings weniger alberne Botschaften wie $\texttt{ANGRIFF}\sqcup\texttt{IM}\sqcup\texttt{MORGENGRAUEN}$ verschlüsselt sondern unter Anderem auch ganze Computerdateien beliebigen Inhalts. Man hat es

H. Schmidt, M. Schwabl-Schmidt, *AES und Rucksackverfahren*, https://doi.org/10.1007/978-3-658-19704-9_2

dann mit Folgen der kleinsten Einheit einer solchen Datei zu tun, also etwa mit Bytes oder 16-Bit-Worten, die als Elemente von $\mathbb{Z}_2^8$ bzw. $\mathbb{Z}_2^{16}$ interpretiert werden können.

Moderne Verschlüsselungsverfahren arbeiten nicht mehr mit Folgen von als Zahlen interpretierten Zeichen (Buchstaben, Ziffern, Satzzeichen), sondern mit mathematischen Objekten. Dazu zählt natürlich AES, das mit 4×4-Matrizen mit Koeffizienten im endlichen Körper $\mathbb{K}_{2^8}$ arbeitet, andere Systeme arbeiten mit sehr großen natürlichen Zahlen. Diese systeminternen Klartexte und Geheimtexte werden zum praktischen Gebrauch in externe Texte verwandelt oder aus ihnen rückverwandelt.

Die Menge $\boldsymbol{\mathcal{F}}_M$ *aller* Folgen mit Elementen in M ist für praktische Zwecke viel zu groß, man beschränkt sich deshalb auf die Menge aller *endlichen* Folgen:

$$\boldsymbol{\mathcal{E}}_M = \left\{ \boldsymbol{f} \colon N \longrightarrow M \;\middle|\; N \subset \mathbb{N} \;\wedge\; \#(N) < \#(\mathbb{N}) \right\}$$

Es ist natürlich $\boldsymbol{\mathcal{E}}_M \subset \boldsymbol{\mathcal{F}}_M$. Schließlich sei noch für beliebige Mengen A und B

$$\mathbf{I}\,\langle A, B\rangle$$

die Menge aller injektiven Abbildungen $\varphi \colon A \longrightarrow B$, also solcher Abbildungen mit der Eigenschaft, daß $a = \tilde{a}$ aus $\varphi(a) = \varphi(\tilde{a})$ folgt.

Mit diesen Bezeichnungen kann nun ein Chiffriersystem präzise beschrieben werden. Mit der nachfolgenden Definition werden praktisch alle existierenden Chiffriersysteme erfasst. Und zwar besteht ein Chiffriersystem aus

($\boldsymbol{i}$) einem Klartextalphabet T
($\boldsymbol{ii}$) einem Klartextbereich $\boldsymbol{\mathcal{T}} \subset \boldsymbol{\mathcal{E}}_{\mathsf{T}}$
($\boldsymbol{iii}$) einem Geheimtextalphabet G
($\boldsymbol{iv}$) einem Geheimtextbereich $\boldsymbol{\mathcal{G}} \subset \boldsymbol{\mathcal{E}}_{\mathsf{G}}$
($\boldsymbol{v}$) einem Schlüsselbereich $\boldsymbol{\mathcal{K}}$
($\boldsymbol{vi}$) einer Abbildung $\boldsymbol{\Omega} \colon \boldsymbol{\mathcal{K}} \longrightarrow \mathbf{I}\,\langle \boldsymbol{\mathcal{T}}, \boldsymbol{\mathcal{G}}\rangle$

Jedem Schlüssel $\boldsymbol{k} \in \boldsymbol{\mathcal{K}}$ ist also eine injektive Abbildung $\boldsymbol{\Omega}(\boldsymbol{k})$ vom Klartextbereich $\boldsymbol{\mathcal{T}}$ in den Geheimtextbereich $\boldsymbol{\mathcal{G}}$ zugeordnet, nämlich die **Verschlüsselungsabbildung** oder **Chiffrierabbildung** des Schlüssels $\boldsymbol{k}$.
Die Umkehrfunktion von $\boldsymbol{\Omega}(\boldsymbol{k})$, also die Abbildung $\boldsymbol{\Omega}(\boldsymbol{k})^{-1} \colon \boldsymbol{\Omega}(\boldsymbol{k})[\boldsymbol{\mathcal{T}}] \longrightarrow \boldsymbol{\mathcal{T}}$, ist die **Entschlüsselungsabbildung** oder **Dechiffrierungsabbildung**.

Diese auf den ersten Blick möglicherweise etwas furchteinflößende Definition ist tatsächlich leicht zu verstehen, wie das folgende einfache Beispiel zeigt.

($\boldsymbol{i}$) $\mathsf{T} = \mathbb{Z}_{27}$
($\boldsymbol{ii}$) $\boldsymbol{\mathcal{T}} = \boldsymbol{\mathcal{E}}_{\mathbb{Z}_{27}}$
($\boldsymbol{iii}$) $\mathsf{G} = \mathbb{Z}_{27}$
($\boldsymbol{iv}$) $\boldsymbol{\mathcal{G}} = \boldsymbol{\mathcal{E}}_{\mathbb{Z}_{27}}$
($\boldsymbol{v}$) $\boldsymbol{\mathcal{K}} = \mathbb{Z}_{27}$
($\boldsymbol{vi}$) Die Abbildung $\boldsymbol{\Omega} \colon \mathbb{Z}_{27} \longrightarrow \mathbf{I}\,\langle \boldsymbol{\mathcal{E}}_{\mathbb{Z}_{27}}, \boldsymbol{\mathcal{E}}_{\mathbb{Z}_{27}}\rangle$ ist gegeben als $\boldsymbol{\Omega}(k)\big((\boldsymbol{t}_\nu)_{\nu\in N}\big) = (\boldsymbol{t}_\nu \oplus k)_{\nu\in N}$, wobei $N \subset \mathbb{N}$ endlich und $\boldsymbol{t} \colon N \longrightarrow \mathbb{Z}_{27}$ eine Folge von Elementen aus $\mathbb{Z}_{27}$ ist. Ferner ist $\oplus$ die Addition von $\mathbb{Z}_{27}$.

Das Verfahren ersetzt jedes Zeichen $\boldsymbol{t}_\nu$ einer Folge $(\boldsymbol{t}_\nu)_{\nu\in N}$ durch das Zeichen $\boldsymbol{t}_\nu \oplus k$, oder um es mit einem Fremdwort auszudrücken, $\boldsymbol{t}_\nu \oplus k$ *substituiert* $\boldsymbol{t}_\nu$. Dieses Chiffriersystem heißt deshalb

ein *Chiffriersystem mit Substitutionsschlüssel*.

Ganz konkret im Falle $k = 1$ wird also folgendermaßen substituiert: Jeder Buchstabe von `A` bis `Y` wird durch den rechten Nachbarbuchstaben im Alphabet ersetzt, d.h. `A` durch `B`, `B` durch `C` usw. bis schließlich `Y` durch `Z`. Weiter wird `Z` durch das Leerzeichen $\sqcup$ und endlich das Leerzeichen selbst durch den Buchstaben `A` ersetzt. Dieses Verschlüsselungssystem ist sehr alt, schon Julius Caesar soll es mit $k = 3$ eingesetzt haben.

Es ist allerdings noch nicht gezeigt worden, daß das beschriebene System tatsächlich ein Chiffriersystem ist. Dazu ist zu bestätigen, daß für jedes $k \in \mathbb{Z}_{27}$ die Abbildung $\boldsymbol{\Omega}(k)$ injektiv ist. Es seien also $\boldsymbol{s} = (\boldsymbol{s}_\mu)_{\mu\in M}$ und $\boldsymbol{t} = (\boldsymbol{t}_\nu)_{\nu\in N}$ Folgen mit

$$(\boldsymbol{s}_\nu \oplus k)_{\mu\in M} = \boldsymbol{\Omega}(k)\big((\boldsymbol{s}_\mu)_{\mu\in M}\big) = \boldsymbol{\Omega}(k)\big((\boldsymbol{t}_\nu)_{\nu\in N}\big) = (\boldsymbol{t}_\nu \oplus k)_{\nu\in N}$$

Zwei Abbildungen, d.h. hier die beiden Folgen $\boldsymbol{\Omega}(k)(\boldsymbol{s})\colon M \longrightarrow \mathbb{Z}_{27}$ und $\boldsymbol{\Omega}(k)(\boldsymbol{t})\colon N \longrightarrow \mathbb{Z}_{27}$, können nur dann gleich sein, wenn sie denselben Definitionsbereich besitzen. Aus obiger Annahme folgt deshalb $M = N$. Weiter sind zwei Abbildungen (mit demselben Definitionsbereich) genau dann gleich, wenn ihre Bilder gleich sind. Aus der Annahme folgt also weiter $\boldsymbol{s}_\nu \oplus k = \boldsymbol{t}_\nu \oplus k$. Die Addition von $-k = 27 - k$ bringt hier natürlich das Gewünschte, nämlich $\boldsymbol{s}_\nu = \boldsymbol{t}_\nu$.

Es bleibt noch zu bestimmen, wie in diesem System entschlüsselt wird, d.h. es sind die Umkehrfunktionen der $\boldsymbol{\Omega}(k)$ anzugeben. Aber die $\boldsymbol{\Omega}(k)$ sind natürlich surjektiv, denn zu gegebener Folge $(\boldsymbol{t}_\nu)_{\nu\in N}$ ist

$$\boldsymbol{\Omega}(k)\big((\boldsymbol{t}_\nu \ominus k)_{\nu\in N}\big) = (\boldsymbol{t}_\nu)_{\nu\in N}$$

womit auch die Umkehrabbildungen $\boldsymbol{\Omega}(k)^{-1}$ gegeben sind, nämlich als

$$\boldsymbol{\Omega}(k)^{-1}\big((\boldsymbol{t}_\nu)_{\nu\in N}\big) = (\boldsymbol{t}_\nu \ominus k)_{\nu\in N}$$

Dabei ist $u \ominus v$ eine Abkürzung von $u \oplus (-v)$.

Das soeben vorgestellte auf $\mathbb{Z}_{27}$ als Alphabet aufbauende Chiffriersystem mit Substitutionsschlüssel kann ganz allgemein formuliert werden. Es sei dazu für irgendeine Menge U

$$\mathbf{P}\,\langle U\rangle$$

die Menge aller bijektiven Abbildungen oder *Permutationen* $\xi\colon U \longrightarrow U$ von U. Damit wird durch das folgende Schema

- (*i*) $\mathsf{T} = A$ mit einer endlichen Menge A
- (*ii*) $\boldsymbol{\mathcal{T}} = \boldsymbol{\mathcal{E}}_A$
- (*iii*) $\mathsf{G} = A$
- (*iv*) $\boldsymbol{\mathcal{G}} = \boldsymbol{\mathcal{E}}_A$
- (*v*) $\boldsymbol{\mathcal{K}} = \mathbf{P}\,\langle A\rangle$
- (*vi*) Die Abbildung $\boldsymbol{\Omega}\colon \mathbf{P}\,\langle A\rangle \longrightarrow \mathbf{I}\,\langle \boldsymbol{\mathcal{E}}_A, \boldsymbol{\mathcal{E}}_A\rangle$ ist gegeben als

$$\boldsymbol{\Omega}(\xi)\big((\boldsymbol{a}_\nu)_{\nu\in N}\big) = \big(\xi(\boldsymbol{a}_\nu)\big)_{\nu\in N}$$

wobei $N \subset \mathbb{N}$ endlich und $\boldsymbol{a}\colon N \longrightarrow A$ eine Folge von Elementen aus A ist.

ein Chiffriersystem mit Substitutionsschlüssel definiert. Hier hat man im Wesentlichen zu zeigen, daß $\boldsymbol{a}_\nu = \boldsymbol{b}_\nu$ aus $\xi(\boldsymbol{a}_\nu) = \xi(\boldsymbol{b}_\nu)$ folgt, aber das ist wegen der Bijektivität von ξ klar. Und es gilt natürlich die Gleichung $\boldsymbol{\Omega}(\xi)^{-1} = \boldsymbol{\Omega}(\xi^{-1})$.

Offensichtlich wird durch $x \mapsto x \oplus k$ eine bijektive Abbildung $\mathbb{Z}_{27} \longrightarrow \mathbb{Z}_{27}$ definiert, d.h. das Chiffriersystem mit Substitutionsschlüssel ist tatsächlich eine Spezialisierung des eben vorgestellten Systems mit Permutationen. Es gibt noch viele solcher Spezialisierungen, wie es auch noch viele andere mit Raffinesse ausgedachte Verschlüsselungssysteme gibt, die heute allerdings aus Gründen mangelnder Sicherheit höchstens noch historische Bedeutung haben. Manche setzen mechanische Hilfsmittel ein, deren Spektrum sich vom simplen Streifen Papier bis zu ausgeklügelten Zahnradmechaniken erstreckt. Ein diesbezüglich leicht lesbares Buch mit vielen Illustrationen ist [ASMW].

Die bisher betrachteten Verschlüsselungssysteme chiffrieren und dechiffrieren die Zeichen so, wie sie der Reihe nach anfallen (daher *stream chipher* genannt). Manche Systeme fassen jedoch eine bestimmte Anzahl von Zeichen in einem Block zusammen und chiffrieren und dechiffrieren diesen Block als eine Einheit (daher *block chipher* genannt). Zu diesem Typ zählen auch die im Buch ausführlich vorgestellten Systeme, also AES und das Rucksacksystem.

Um nun auch diese Verschlüsselungssysteme in das obige Schema zu bringen wird aus der Menge $\boldsymbol{\mathcal{F}}_M$ *aller* Folgen mit Elementen in einer Menge M neben der Teilmenge aller endlichen Folgen eine weitere Teilmenge ausgesondert:

$$\boldsymbol{\mathcal{E}}_M^{16} = \left\{ \boldsymbol{f} : N \longrightarrow M \;\middle|\; N \subset \mathbb{N} \;\wedge\; \#(N) = 16 \right\}$$

Es ist also die Menge aller Folgen der Länge 16, und natürlich ist $\boldsymbol{\mathcal{E}}_M^{16} \subset \boldsymbol{\mathcal{E}}_M$. Weiterhin sei $\boldsymbol{\mathcal{M}}_M$ die Menge der 4×4-Matrizen mit Koeffizienten in der Menge M:

$$\boldsymbol{\mathcal{M}}_M = \left\{ \begin{pmatrix} m_{00} & m_{01} & m_{02} & m_{03} \\ m_{10} & m_{11} & m_{12} & m_{13} \\ m_{20} & m_{21} & m_{22} & m_{23} \\ m_{30} & m_{31} & m_{32} & m_{33} \end{pmatrix} \;\middle|\; m_{\nu\mu} \in M \right\}$$

Schließlich sei noch eine Abbildung $\boldsymbol{\Lambda} : \boldsymbol{\mathcal{E}}_M^{16} \longrightarrow \boldsymbol{\mathcal{M}}_M$ definiert durch

$$\boldsymbol{\Lambda}\big((\boldsymbol{f}_\nu)_{\nu\in N}\big) = \begin{pmatrix} \boldsymbol{f}_{\nu_0} & \boldsymbol{f}_{\nu_4} & \boldsymbol{f}_{\nu_8} & \boldsymbol{f}_{\nu_{12}} \\ \boldsymbol{f}_{\nu_1} & \boldsymbol{f}_{\nu_5} & \boldsymbol{f}_{\nu_9} & \boldsymbol{f}_{\nu_{13}} \\ \boldsymbol{f}_{\nu_2} & \boldsymbol{f}_{\nu_6} & \boldsymbol{f}_{\nu_{10}} & \boldsymbol{f}_{\nu_{14}} \\ \boldsymbol{f}_{\nu_3} & \boldsymbol{f}_{\nu_7} & \boldsymbol{f}_{\nu_{11}} & \boldsymbol{f}_{\nu_{15}} \end{pmatrix} \quad \text{mit } N = \{\nu_0, \nu_1, \ldots, \nu_{15}\} \text{ und } \#(N) = 16$$

Diese Abbildung ist offensichtlich eine Bijektion, die eine Folge der Länge 16 von Elementen aus M zu einer 4×4-Matrix zusammenfasst (siehe z.B. auch Abschnitt 4).

Bei den bisher vorgestellten Verschlüsselungssystemen ist der Schlüssel sowohl beim Sender als auch beim Empfänger eines Textes streng geheim zu halten. Vor dem Empfang eines Textes muß der Schlüssel also auf einem sicheren Wege zum Empfänger gelangen. Denn bei diesen Systemen ist bei gegebenem Schlüssel $\boldsymbol{k} \in \boldsymbol{\mathcal{K}}$ sowohl die Verschlüsselungsabbildung $\boldsymbol{\Omega}(\boldsymbol{k}) : \boldsymbol{\mathcal{T}} \longrightarrow \boldsymbol{\mathcal{G}}$ als auch deren Umkehrung, die Entschlüsselungsabbildung $\boldsymbol{\Omega}(\boldsymbol{k})^{-1} : \boldsymbol{\mathcal{G}} \longrightarrow \boldsymbol{\mathcal{T}}$ bekannt (hier wird die Bijektivität von $\boldsymbol{\Omega}(\boldsymbol{k})$ angenommen). Ein verschlüsselter Text $\mathbf{g} = \boldsymbol{\Omega}(\boldsymbol{k})(\mathbf{t})$ kann also von jedem, der den Schlüssel $\mathbf{k}$ kennt, sofort als $\mathbf{t} = \boldsymbol{\Omega}(\boldsymbol{k})^{-1}(\mathbf{g})$ entschlüsselt werden.

Betrachtet man jedoch die Verschlüsselungssysteme genauer, dann fällt auf, daß der Zusammenhang zwischen $\boldsymbol{\Omega}(\boldsymbol{k})$ und $\boldsymbol{\Omega}(\boldsymbol{k})^{-1}$ wie folgt beschrieben werden kann: Es gibt eine Abbildung $\zeta : \boldsymbol{\mathcal{K}} \longrightarrow \boldsymbol{\mathcal{K}}$ so, daß

$$\boldsymbol{\Omega}(\boldsymbol{k})^{-1} = \boldsymbol{\Omega}\big(\zeta(\boldsymbol{k})\big)$$

gilt. Im Chiffriersystem mit $\boldsymbol{\mathcal{K}} = \mathbb{Z}_{27}$ ist beispielsweise $\zeta(k) = -k$, im System mit $\boldsymbol{\mathcal{K}} = \mathbf{P}\langle A\rangle$ ist $\zeta(\xi) = \xi^{-1}$, und (etwas vorgegriffen) im HILLschen System ist $\zeta(\mathbf{K}) = \mathbf{K}^{-1}$. In diesen Beispielen ist $\zeta(\mathbf{k})$ aus $\mathbf{k}$ leicht zu berechnen.

Aber einmal angenommen, es wäre unmöglich oder zumindest praktisch unmöglich, $\tilde{\mathbf{k}} = \zeta(\mathbf{k})$ aus $\mathbf{k}$ zu bestimmen, ohne die Abbildung ζ zu kennen. Dann könnte doch die Chiffrierabbildung $\boldsymbol{\Omega}(\boldsymbol{k})$ öffentlich so zugänglich gemacht werden, etwa in einer Datenbank, daß **jeder** damit einen Text verschlüsseln könnte. Entschlüsseln könnte diesen Text nur derjenige, der die Abbildung ζ und damit die Entschlüsselungsabbildung $\boldsymbol{\Omega}(\boldsymbol{k})^{-1} = \boldsymbol{\Omega}\big(\zeta(\boldsymbol{k})\big)$ kennt. Bei Einsatz einer solchen Abbildung ζ ist kein gesicherter Schlüsseltransfer vom Empfänger an den Sender notwendig.

Nun hält die Mathematik eine Reihe von Problemen bereit, die bei geeigneter Parameterwahl praktisch unlösbar sind. Das bekannteste Problem ist wohl die Zerlegung einer natürlichen Zahl in ihre Primzahlkomponenten, die bei derzeitiger Rechentechnik bei einer geeignet gewählten Zahl unmöglich zu berechnen ist.

Problematisch bei vielen dieser Probleme ist allerdings, daß nur *vermutet* wird, daß sie praktisch unlösbar sind. Ein Verschlüsselungssystem, das auf der angenommenen (praktischen) Unmöglichkeit der Lösung eines Problems beruht, wird über Nacht wertlos, wenn doch eine Lösungsmöglichkeit gefunden wird. Ein Problem jedoch, das schon viele Jahrhunderte über traktiert wurde, dürfte für die nähere Zukunft als sicher gelten.

Eine bijektive Abbildung $f : A \longrightarrow B$, mit der $b = f(a)$ einfach zu berechnen ist, mit der es jedoch unmöglich oder doch sehr schwer ist, $a = f^{-1}(b)$ zu berechnen, wird oft aus offensichtlichen Gründen mit dem Namen **Falltürfunktion** bezeichnet. Hier wäre also $\boldsymbol{\Omega}(\boldsymbol{k})$ die Falltürfunktion, deren Inverse $\boldsymbol{\Omega}\big(\zeta(\boldsymbol{k})\big)$ ohne die Kenntnis der Abbildung ζ unmöglich zu bestimmen ist.

Ein bekanntes Beispiel ist die diskrete Exponentialfunktion. Nach Kapitel 10 gibt es zu jeder Primzahlpotenz $q = p^k$ ein $a \in \{1, \ldots, q-1\}$ so, daß es zu jedem $u \in \{1, \ldots, q-1\}$ **genau ein** $\nu \in \{0, \ldots, q-2\}$ gibt mit $u = a^\nu$. Durch

$$\exp_a(\nu) = a^\nu$$

wird daher eine **bijektive** Abbildung $\exp_a : \{0, \ldots, q-2\} \longrightarrow \{1, \ldots, q-1\}$ definiert, die diskrete Exponentialfunktion (zum primitiven Element a).

Die Bezeichnung *Exponentialfunktion* darf allerdings nicht darüber hinwegtäuschen, daß $\exp_a$ nur eine Approximation der reellen Exponentialfunktion darstellt. Man kann nicht erwarten, daß die diskrete Exponentialfunktion alle Eigenschaften der reellen Funktion getreulich nachahmt. Aus der Eigenschaft $\exp_a(x)\exp_a(y) = \exp_a(x+y)$ der reellen Exponentialfunktion wird nämlich

$$\exp_a(\nu)\exp_a(\mu) = \exp_a\big(\boldsymbol{\varrho}_{q-1}(\nu+\mu)\big)$$

Natürlich gilt für das Potenzenprodukt auch $a^\nu a^\mu = a^{\nu+\mu}$, für die Exponentialfunktion ist jedoch $\nu \in \{0, \ldots, q-2\}$ gefordert, um $\exp_a(\nu)$ bilden zu können, d.h. der Exponent ist modulo $q-1$ zu reduzieren. Jedenfalls ist $\exp_a$ eine bijektive Abbildung, die eine Umkehrabbildung

$$\log_a : \{1, \ldots, q-1\} \longrightarrow \{0, \ldots, q-2\}$$

besitzt, die entsprechend diskreter Logarithmus genannt wird. Auch die bekannte Eigenschaft $\log_a(xy) = \log_a(x) + \log_a(y)$ der reellen Logarithmusfunktion kann nur in approximativer Gestalt erwartet werden:

$$\log_a(\nu\mu) = \boldsymbol{\varrho}_{q-1}\big(\log_a(\nu) + \log_a(\mu)\big)$$

In diesem Zusammenhang ist es allerdings bedeutender, daß die diskrete Exponentialfunktion eine Falltürfunktion ist. Die Berechnung von $\exp_a(\nu)$ ist sehr einfach auszuführen für kleine $\nu \in \{0, \ldots, q-2\}$ und es gibt schnelle und sehr schnelle Algorithmen für große ν. Niemand kennt jedoch einen schnellen Algorithmus zur Berechnung von $\log_a(\mu)$ für große $\mu \in \{1, \ldots, q-1\}$. In der Praxis ist q eine Primzahl mit mindestens 100 Dezimalziffern, es ist also unmöglich, alle $\nu \in \{0, \ldots, q-2\}$ zu durchlaufen und zu prüfen, ob $\exp_a(\nu) = \mu$ gilt.

Diese Eigenschaft der diskreten Logarithmusfunktion, sehr schwierig berechenbar zu sein, wird auch tatsächlich in einem Chiffriersystem genutzt, und zwar in dem System von ElGamal.

Es folgen noch zwei Beispiele von Verschlüsselungssytemen. Das erste, das System von Hill, ist vom Anwender her gesehen dem System AES ein wenig ähnlich, ist aber sehr viel einfacher strukturiert und kann so als eine leicht verdauliche Einstimmung auf AES dienen. Das zweite System ist RSA, das bekannteste aller Falltürsysteme. Es ist vom Konzept her nicht sonderlich schwierig und als ein Beispiel für ein Falltürsystem gut geeignet.

2.2. Das Chiffriersystem von HILL

In diesem Abschnitt wird eine Abart des Chiffriersystems von HILL vorgestellt, die möglichst dem System AES angenähert ist, jedenfalls was das Äußere, also die Schnittstelle zum Anwender, betrifft. Es ist zwar schon etwas in die Jahre gekommen — es wurde am Anfang des vorigen Jahrhunderts entwickelt — kann aber mit etwas Vorsicht durchaus noch heute eingesetzt werden, wenn es nicht auf die allerhöchste Geheimhaltungsstufe ankommt.

Die Klartexte und Geheimtexte des Chiffriersystems sind Elemente $\mathbf{T}$ der Menge $\boldsymbol{\mathcal{M}}$ der 4×4-Matrizen mit Koeffizienten im Körper $\mathbb{K}_{2^8} = \mathbb{K}_{256}$:

$$\mathbf{T} = \begin{pmatrix} \mathbf{t}_{00} & \mathbf{t}_{01} & \mathbf{t}_{02} & \mathbf{t}_{03} \\ \mathbf{t}_{10} & \mathbf{t}_{11} & \mathbf{t}_{12} & \mathbf{t}_{13} \\ \mathbf{t}_{20} & \mathbf{t}_{21} & \mathbf{t}_{22} & \mathbf{t}_{23} \\ \mathbf{t}_{30} & \mathbf{t}_{31} & \mathbf{t}_{32} & \mathbf{t}_{33} \end{pmatrix}$$

Die Schlüssel $\mathbf{S}$ des Systems sind ebenfalls Matrizen aus $\boldsymbol{\mathcal{M}}$, sie müssen jedoch aus der Teilmenge $\boldsymbol{\mathcal{S}} \subset \boldsymbol{\mathcal{M}}$ der **regulären** Matrizen gewählt werden. Das bedeutet also, daß es zu jedem Schlüssel $\mathbf{S}$ einen Schlüssel $\mathbf{S}^{-1}$ gibt mit

$$\mathbf{S}\mathbf{S}^{-1} = \mathbf{S}^{-1}\mathbf{S} = \mathbf{I}$$

dabei ist $\mathbf{I}$ die 4×4-Einheitsmatrix mit Einsen (also $\mathtt{01}_{16}$) in der Hauptdiagonalen und überall sonst Nullen (also $\mathtt{00}_{16}$). Die Multiplikation der Matrizen $\mathbf{S}$ und $\mathbf{S}^{-1}$ ist natürlich mit der Addition und Multiplikation des Körpers $\mathbb{K}_{2^8}$ auszuführen (hier weicht das vorgestellte Verschlüsselungssystem vom ursprünglichen HILLschen System ab). Und schließlich sind die zu einem Schlüssel $\mathbf{S}$ gehörigen Chiffrierabbildungen

$$\boldsymbol{\Psi}_{\mathbf{S}} : \boldsymbol{\mathcal{M}} \longrightarrow \boldsymbol{\mathcal{M}} \qquad \boldsymbol{\Psi}_{\mathbf{S}}^{-1} : \boldsymbol{\mathcal{M}} \longrightarrow \boldsymbol{\mathcal{M}}$$

definiert als die folgenden Matrizenprodukte:

$$\boldsymbol{\Psi}_{\mathbf{S}}(\mathbf{T}) = \mathbf{S}\mathbf{T} \qquad \boldsymbol{\Psi}_{\mathbf{S}}^{-1}(\mathbf{T}) = \mathbf{S}^{-1}\mathbf{T}$$

Hier besteht also der einfache Zusammenhang $\boldsymbol{\Psi}_{\mathbf{S}}^{-1} = \boldsymbol{\Psi}_{\mathbf{S}^{-1}}$, und $\boldsymbol{\Psi}_{\mathbf{S}}$ und $\boldsymbol{\Psi}_{\mathbf{S}}^{-1}$ sind offensichtlich invers zueinander:

$$\boldsymbol{\Psi}_{\mathbf{S}}^{-1}\big(\boldsymbol{\Psi}_{\mathbf{S}}(\mathbf{T})\big) = \mathbf{S}^{-1}\mathbf{S}\mathbf{T} = \mathbf{I}\mathbf{T} = \mathbf{T}$$

Ausführlich geschrieben wird die Chiffrierfunktion (wie auch ihre Umkehrung) wie folgt gebildet:

$$\mathbf{S}\mathbf{T} = \begin{pmatrix} \mathbf{s}_{00} & \mathbf{s}_{01} & \mathbf{s}_{02} & \mathbf{s}_{03} \\ \mathbf{s}_{10} & \mathbf{s}_{11} & \mathbf{s}_{12} & \mathbf{s}_{13} \\ \mathbf{s}_{20} & \mathbf{s}_{21} & \mathbf{s}_{22} & \mathbf{s}_{23} \\ \mathbf{s}_{30} & \mathbf{s}_{31} & \mathbf{s}_{32} & \mathbf{s}_{33} \end{pmatrix} \begin{pmatrix} \mathbf{t}_{00} & \mathbf{t}_{01} & \mathbf{t}_{02} & \mathbf{t}_{03} \\ \mathbf{t}_{10} & \mathbf{t}_{11} & \mathbf{t}_{12} & \mathbf{t}_{13} \\ \mathbf{t}_{20} & \mathbf{t}_{21} & \mathbf{t}_{22} & \mathbf{t}_{23} \\ \mathbf{t}_{30} & \mathbf{t}_{31} & \mathbf{t}_{32} & \mathbf{t}_{33} \end{pmatrix} = \begin{pmatrix} \mathbf{g}_{00} & \mathbf{g}_{01} & \mathbf{g}_{02} & \mathbf{g}_{03} \\ \mathbf{g}_{10} & \mathbf{g}_{11} & \mathbf{g}_{12} & \mathbf{g}_{13} \\ \mathbf{g}_{20} & \mathbf{g}_{21} & \mathbf{g}_{22} & \mathbf{g}_{23} \\ \mathbf{g}_{30} & \mathbf{g}_{31} & \mathbf{g}_{32} & \mathbf{g}_{33} \end{pmatrix} = \mathbf{G}$$

Die Koeffizienten $\mathbf{g}_{\mu\nu}$ der Matrix $\mathbf{G}$ sind gegeben als

$$\mathbf{g}_{\mu\nu} = \mathbf{s}_{\mu 0}\mathbf{t}_{0\nu} + \mathbf{s}_{\mu 1}\mathbf{t}_{1\nu} + \mathbf{s}_{\mu 2}\mathbf{t}_{2\nu} + \mathbf{s}_{\mu 3}\mathbf{t}_{3\nu} \qquad \mu, \nu \in \{0, 1, 2, 3\}$$

Die Additionen und Multiplikationen sind dabei die des Körpers $\mathbb{K}_{2^8}$. Eine solche Multiplikation wird mit Hilfe von Polynommultiplikationen und Polynomdivisionen ausgeführt, die beide hoch

nichtlinear sind. Die Chiffrierabbildung besitzt deshalb einen gut ausgebildeten Verschleierungseffekt, womit auf die von SHANNON geforderte Diffusion angespielt ist.

Nun sind Matrizen mit Koeffizienten in $\mathbb{K}_{2^8}$ für Anwender des Chiffrierverfahrens sicherlich ungewohnte wenn nicht sogar befremdliche Klartexte und Geheimtexte. Um eine angenehmere Anwendungsschnittstelle zu bekommen, wird wie folgt vorgegangen:

- Es wird von der speziellen Struktur der Matrixkoeffizienten abgelassen, sie werden nur noch als eben strukturlose Bytes oder Bitoktetts $t_{\mu\nu}$ angesehen.
- Die Matrixspalten werden zu einem Vektor $\boldsymbol{t}$ von 16 Bytes zusammengesetzt.

Es ist eigentlich gleichgültig, auf welche Weise der Vektor $\boldsymbol{t}$ aus den Spalten der Matrix gebildet wird. Hier geschieht das folgendermaßen:

$$\boldsymbol{t} = \begin{pmatrix} t_0 & t_1 & t_2 & t_3 & t_4 & \cdots & t_{15} \end{pmatrix} = \begin{pmatrix} \mathbf{t}_{00} & \mathbf{t}_{10} & \mathbf{t}_{20} & \mathbf{t}_{30} & \mathbf{t}_{01} & \mathbf{t}_{11} & \mathbf{t}_{21} & \cdots & \mathbf{t}_{23} & \mathbf{t}_{33} \end{pmatrix}$$

Als ein Beispiel soll die bekannte Textzeile `Habe␣nun,␣ach!␣␣` verschlüsselt werden. Verwendet man für die Zeichen des Textes den ASCII-Code, so erhält man den Anwenderklartext

$$\boldsymbol{t} = (48_{16}\ 61_{16}\ 62_{16}\ 65_{16}\ 20_{16}\ 6E_{16}\ 75_{16}\ 6E_{16}\ 2C_{16}\ 20_{16}\ 61_{16}\ 63_{16}\ 68_{16}\ 21_{16}\ 20_{16}\ 20_{16})$$

und daraus den Klartext des Chiffriersystems als

$$\mathbf{T} = \begin{pmatrix} 48_{16} & 20_{16} & 2C_{16} & 68_{16} \\ 61_{16} & 6E_{16} & 20_{16} & 21_{16} \\ 62_{16} & 75_{16} & 61_{16} & 20_{16} \\ 65_{16} & 6E_{16} & 63_{16} & 20_{16} \end{pmatrix}$$

Wie man sich Schlüssel verschaffen kann wird weiter unten ausführlich diskutiert. In diesem Beispiel wird der Schlüssel

$$\mathbf{S} = \begin{pmatrix} 01_{16} & 23_{16} & 71_{16} & 15_{16} \\ 01_{16} & 7F_{16} & F1_{16} & 40_{16} \\ 01_{16} & B1_{16} & 7B_{16} & 08_{16} \\ 01_{16} & EA_{16} & EE_{16} & 61_{16} \end{pmatrix} \qquad \mathbf{S}^{-1} = \begin{pmatrix} 54_{16} & 16_{16} & A0_{16} & E3_{16} \\ 91_{16} & 8B_{16} & 33_{16} & 29_{16} \\ F6_{16} & BF_{16} & 83_{16} & CA_{16} \\ E0_{16} & 6B_{16} & 39_{16} & B2_{16} \end{pmatrix}$$

eingesetzt. Zur Not kann die Chiffrierung auch mit den Tabellen in Kapitel 10 durchgeführt werden. Wie es auch gemacht wird, der Geheimtext ist jedenfalls

$$\mathbf{ST} = \begin{pmatrix} 01_{16} & 23_{16} & 71_{16} & 15_{16} \\ 01_{16} & 7F_{16} & F1_{16} & 40_{16} \\ 01_{16} & B1_{16} & 7B_{16} & 08_{16} \\ 01_{16} & EA_{16} & EE_{16} & 61_{16} \end{pmatrix} \begin{pmatrix} 48_{16} & 20_{16} & 2C_{16} & 68_{16} \\ 61_{16} & 6E_{16} & 20_{16} & 21_{16} \\ 62_{16} & 75_{16} & 61_{16} & 20_{16} \\ 65_{16} & 6E_{16} & 63_{16} & 20_{16} \end{pmatrix} = \begin{pmatrix} D0_{16} & CA_{16} & 52_{16} & 43_{16} \\ 80_{16} & EB_{16} & BE_{16} & EF_{16} \\ 88_{16} & F6_{16} & BB_{16} & BC_{16} \\ DF_{16} & 62_{16} & 3D_{16} & BE_{16} \end{pmatrix} = \mathbf{G}$$

was zu folgendem Anwendergeheimtext führt:

$$\boldsymbol{g} = (D0_{16}\ 80_{16}\ 88_{16}\ DF_{16}\ CA_{16}\ EB_{16}\ F6_{16}\ 62_{16}\ 52_{16}\ BE_{16}\ BB_{16}\ 3D_{16}\ 43_{16}\ EF_{16}\ BC_{16}\ BE_{16})$$

Wie ein Vergleich von $\boldsymbol{t}$ und $\boldsymbol{g}$ zeigt, werden mehrfach vorkommende Buchstaben (also die Buchstaben `a` und `n`) verschieden verschlüsselt. Mit statistischen Untersuchungen über die Häufigkeit gewisser Buchstaben in einem Text einer Sprache kann man dieses Chiffriersystem daher nicht überlisten. Das gelingt jedoch unter gewissen Voraussetzungen auf einfache algebraische Weise,

wie weiter unten gezeigt wird.

Zunächst aber ist noch zu klären, auf welchen Wegen man sich Schlüssel verschaffen kann: Gesucht sind reguläre (invertierbare) 4×4-Matrizen mit Koeffizienten aus dem Körper $\mathbb{K}_{2^8}$.

Die sich hier sofort stellende Gretchenfrage ist natürlich: Wie kann man die Regularität (oder entgegengesetzt die Singularität) einer Matrix aus $\mathcal{M}$ erkennen? Es gibt den klassischen Weg über die Determinante, doch ist deren orthodoxe Berechnung über die Adjungierten der Matrix rechentechnisch ein Alptraum und sollte vermieden werden. Es gelingt auf viel einfacherem Wege, indem die Matrix in eine obere Dreiecksmatrix umgeformt wird: Ist die Hauptdiagonale der Dreiecksmatrix voll besetzt, dann ist die Matrix regulär.

Man kann so vorgehen, daß man Matrizen mit Zufallszahlen aufbaut, die dann auf Regularität getestet werden. Allerdings ist die Wahrscheinlichkeit, auf diese Weise eine reguläre Matrix zu finden, nicht sehr groß. Hier sind beispielsweise alle 2×2-Matrizen mit Koeffizienten aus dem Körper $\mathbb{K}_2$:

$$\begin{pmatrix}0&0\\0&0\end{pmatrix}\ \begin{pmatrix}1&0\\0&0\end{pmatrix}\ \begin{pmatrix}0&1\\0&0\end{pmatrix}\ \begin{pmatrix}0&0\\0&1\end{pmatrix}\ \begin{pmatrix}0&0\\1&0\end{pmatrix}\ \begin{pmatrix}1&1\\0&0\end{pmatrix}\ \begin{pmatrix}0&0\\1&1\end{pmatrix}\ \begin{pmatrix}1&0\\1&0\end{pmatrix}$$
$$\begin{pmatrix}0&1\\0&1\end{pmatrix}\ \begin{pmatrix}1&0\\0&1\end{pmatrix}\ \begin{pmatrix}0&1\\1&0\end{pmatrix}\ \begin{pmatrix}1&1\\1&0\end{pmatrix}\ \begin{pmatrix}1&1\\0&1\end{pmatrix}\ \begin{pmatrix}0&1\\1&1\end{pmatrix}\ \begin{pmatrix}1&0\\1&1\end{pmatrix}\ \begin{pmatrix}1&1\\1&1\end{pmatrix}$$

Wie unschwer zu erkennen ist (die Matrixspalten müssen linear unabhängig sein), sind sechs dieser 16 Matrizen regulär, nämlich

$$\begin{pmatrix}1&0\\0&1\end{pmatrix}\ \begin{pmatrix}0&1\\1&0\end{pmatrix}\ \begin{pmatrix}1&1\\1&0\end{pmatrix}\ \begin{pmatrix}1&1\\0&1\end{pmatrix}\ \begin{pmatrix}0&1\\1&1\end{pmatrix}\ \begin{pmatrix}1&0\\1&1\end{pmatrix}$$

folglich ist die Wahrscheinlichkeit, daß solch eine mit Zufallszahlen erzeugte Matrix regulär ist, $\frac{3}{8}$ oder 37,5%. Es ist aber sehr einfach, eine große Zahl von regulären Matrizen direkt anzugeben. Sind nämlich $\mathbf{a},\mathbf{b},\mathbf{c},\mathbf{d}\in\mathbb{K}_{2^8}$, dann ist die damit gebildete Matrix von VANDERMONDE

$$\mathbf{V}\langle\mathbf{a},\mathbf{b},\mathbf{c},\mathbf{d}\rangle=\begin{pmatrix}\mathbf{1}&\mathbf{a}&\mathbf{a}^2&\mathbf{a}^3\\\mathbf{1}&\mathbf{b}&\mathbf{b}^2&\mathbf{b}^3\\\mathbf{1}&\mathbf{c}&\mathbf{c}^2&\mathbf{c}^3\\\mathbf{1}&\mathbf{d}&\mathbf{d}^2&\mathbf{d}^3\end{pmatrix}$$

genau dann regulär, wenn die vier Parameter verschieden sind. Man erhält auf diese Weise daher $256\cdot255\cdot254\cdot253=4195023360$ reguläre Matrizen. Die obige Beispielsmatrix ist eine solche reguläre VANDERMONDEsche Matrix:

$$\begin{pmatrix}01_{16}&23_{16}&71_{16}&15_{16}\\01_{16}&7F_{16}&F1_{16}&40_{16}\\01_{16}&B1_{16}&7B_{16}&08_{16}\\01_{16}&EA_{16}&EE_{16}&61_{16}\end{pmatrix}=\mathbf{V}\langle23_{16},7F_{16},B1_{16},EA_{16}\rangle$$

Wenn keine Software zur Invertierung von Matrizen mit Koeffizienten in $\mathbb{K}_{2^8}$ zur Verfügung steht, kann die inverse Schlüsselmatrix auch mit einem AVR-Mikrocontroller berechnet werden, siehe dazu [HSMS] **5.2.**.

Die Implementierung dieses Chiffriersystems ist simpel genug, es gilt nur, das Produkt zwei-

er 4×4-Matrizen zu berechnen. Allerdings gehört die Berechnung solcher Produkte auch zum Chiffriersystem AES, weshalb sich hier eine Probeimplementierung für AVR empfiehlt.

Unterprogramm `mcK284x4Mul`

Es wird das Produkt $\mathbf{G} = \mathbf{ST}$ zweier 4×4-Matrizen $\mathbf{S}$ und $\mathbf{T}$ mit Koeffizienten im Körper $\mathbb{K}_{2^8}$ berechnet.

Input

$\mathbf{r_{29:r28}}$	Die Adresse σ einer 4×4-Matrix $\mathbf{S}$
$\mathbf{r_{31:r30}}$	Die Adresse τ einer 4×4-Matrix $\mathbf{T}$
$\mathbf{r_{27:r26}}$	Die Adresse γ einer 4×4-Matrix $\mathbf{G}$

Die Koeffizienten der Matrizen werden zeilenweise gespeichert.
Das Unterprogramm kann keine Fehler erzeugen.

```
1  mcK284x4Mul:   push5  r16,r17,r18,r19,r20   5×2
2                 ldi    r20,1                1    r20 ← k = 1, d.h. i = 0
3  mcK284x4Mul05:ld     r16,Y                2    r16 ← s_i0
4                 ld     r17,Z                2    r17 ← t_00
5                 rcall  mcK28Mul             3+   r18 ← s_i0 t_00
6                 mov    r19,r18              1    r19 ← h = s_i0 t_00
7                 ldd    r16,Y+1              2    r16 ← s_i1
8                 ldd    r17,Z+4              2    r17 ← t_10
9                 rcall  mcK28Mul             3+   r18 ← s_i1 t_10
10                eor    r19,r18              1    r19 ← h + s_i1 t_10
11                ldd    r16,Y+2              2    r16 ← s_i2
12                ldd    r17,Z+8              2    r17 ← t_20
13                rcall  mcK28Mul             2+   r18 ← s_i2 t_20
14                eor    r19,r18              1    r19 ← h + s_i2 t_20
15                ldd    r16,Y+3              2    r16 ← s_i3
16                ldd    r17,Z+12             2    r17 ← t_30
17                rcall  mcK28Mul             3+   r18 ← s_i3 t_30
18                eor    r19,r18              1    r19 ← h + s_i3 t_30
19                st     X+,r19               2    g_i0 ← h
20                ld     r16,Y                2    r16 ← s_i0
21                ldd    r17,Z+1              2    r17 ← t_01
22                rcall  mcK28Mul             3+   r18 ← s_i0 t_01
23                mov    r19,r18              1    r19 ← h = s_i0 t_01
24                ldd    r16,Y+1              2    r16 ← s_i1
25                ldd    r17,Z+5              2    r17 ← t_11
26                rcall  mcK28Mul             3+   r18 ← s_i1 t_11
27                eor    r19,r18              1    r19 ← h + s_i1 t_11
28                ldd    r16,Y+2              2    r16 ← s_i2
29                ldd    r17,Z+9              2    r17 ← t_21
30                rcall  mcK28Mul             3+   r18 ← s_i2 t_21
31                eor    r19,r18              1    r19 ← h + s_i2 t_21
32                ldd    r16,Y+3              2    r16 ← s_i3
33                ldd    r17,Z+13             2    r17 ← t_31
34                rcall  mcK28Mul             3+   r18 ← s_i3 t_31
35                eor    r19,r18              1    r19 ← h + s_i3 t_31
```

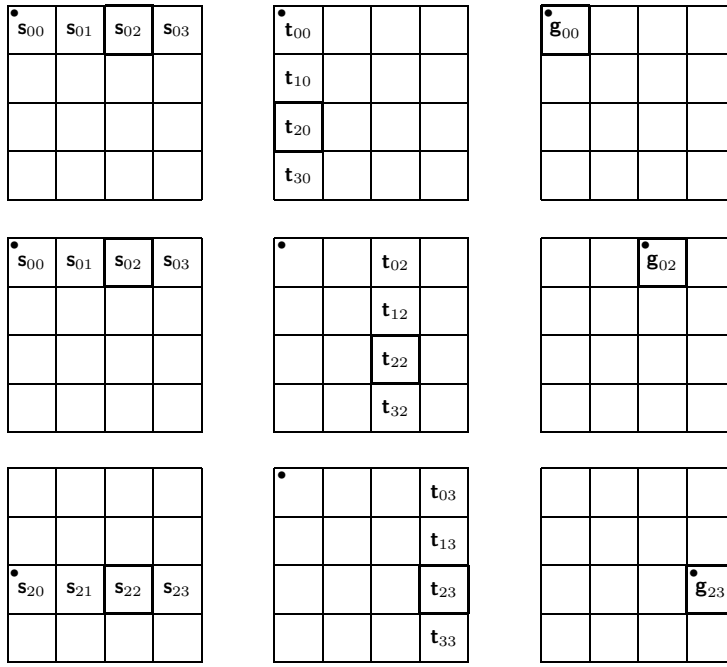
```
36                st      X+,r19                  2    g_{i1} ← h
37                ld      r16,Y                   2    r16 ← s_{i0}
38                ldd     r17,Z+2                 2    r17 ← t_{02}
39                rcall   mcK28Mul                3+   r18 ← s_{i0}t_{02}
40                mov     r19,r18                 1    r19 ← h = s_{i0}t_{02}
41                ldd     r16,Y+1                 2    r16 ← s_{i1}
42                ldd     r17,Z+6                 2    r17 ← t_{12}
43                rcall   mcK28Mul                3+   r18 ← s_{i1}t_{12}
44                eor     r19,r18                 1    r19 ← h + s_{i1}t_{12}
45                ldd     r16,Y+2                 2    r16 ← s_{i2}
46                ldd     r17,Z+10                2    r17 ← t_{22}
47                rcall   mcK28Mul                3+   r18 ← s_{i2}t_{22}
48                eor     r19,r18                 1    r19 ← h + s_{i2}t_{22}
49                ldd     r16,Y+3                 2    r16 ← s_{i3}
50                ldd     r17,Z+14                2    r17 ← t_{32}
51                rcall   mcK28Mul                3+   r18 ← s_{i3}t_{32}
52                eor     r19,r18                 1    r19 ← h + s_{i3}t_{32}
53                st      X+,r19                  2    g_{i2} ← h
54                ld      r16,Y+                  2    r16 ← s_{i0}, r29:28 ← A(s_{i1})
55                ldd     r17,Z+3                 2    r17 ← t_{03}
56                rcall   mcK28Mul                3+   r18 ← s_{i0}t_{03}
57                mov     r19,r18                 1    r19 ← h = s_{i0}t_{03}
58                ld      r16,Y+                  2    r16 ← s_{i1}, r29:28 ← A(s_{i2})
59                ldd     r17,Z+7                 2    r17 ← t_{13}
60                rcall   mcK28Mul                3+   r18 ← s_{i1}t_{13}
61                eor     r19,r18                 1    r19 ← h + s_{i1}t_{13}
62                ld      r16,Y+                  2    r16 ← s_{i2}, r29:28 ← A(s_{i3})
63                ldd     r17,Z+11                2    r17 ← t_{23}
64                rcall   mcK28Mul                3+   r18 ← s_{i2}t_{23}
65                eor     r19,r18                 1    r19 ← h + s_{i2}t_{23}
66                ld      r16,Y+                  2    r16 ← s_{i3}, r29:28 ← A(s_{i+1,3})
67                ldd     r17,Z+15                2    r17 ← t_{33}
68                rcall   mcK28Mul                3+   r18 ← s_{i3}t_{33}
69                eor     r19,r18                 1    r19 ← h + s_{i3}t_{33}
70                st      X+,r19                  2    g_{i3} ← h
71                lsl     r20                     1    k ← 2k
72                sbrs    r20,4                   1/2  Falls k < 8:
73                rjmp    mcK284x4Mul05           2       Zur i + 1-ten Zeile von S
74                sbiw    r27:r26,16              2    r27:r26 restaurieren
75                sbiw    r29:r28,16              2    r29:r28 restaurieren
76                pop5    r20,r19,r18,r17,r16     5×2
77                ret                             4
```

Die Koeffizienten der Matrix $\mathbf{G}$ werden zeilenweise berechnet. Daraus folgt unmittelbar, in welcher Reihenfolge die Zeilen von $\mathbf{S}$ mit den Spalten von $\mathbf{T}$ kombiniert werden müssen. Denn aus

$$\mathbf{g}_{\mu\nu} = \mathbf{s}_{\mu 0}\mathbf{t}_{0\nu} + \mathbf{s}_{\mu 1}\mathbf{t}_{1\nu} + \mathbf{s}_{\mu 2}\mathbf{t}_{2\nu} + \mathbf{s}_{\mu 3}\mathbf{t}_{3\nu} \qquad \mu, \nu \in \{0, 1, 2, 3\}$$

folgt, daß zuerst die erste Zeile von **S** der Reihe nach mit den vier Spalten von **T** zu kombinieren ist, dann die zweite Zeile von **S** der Reihe nach mit den vier Spalten von **T**, und so fort. Die vier Zeilen von **S** werden in einer Schleife durchlaufen, die Kombination mit den vier Spalten von **T** gemäß obiger Formel wird ohne Schleife direkt ausgeführt.

Es bleibt nur noch zu klären, wie die Adressenregister genutzt werden. Die nachfolgenden Skizzen erläutern die Vorgehensweise. Die erste Reihe zeigt die Berechnung von $\mathbf{g}_{00}$, die zweite Reihe die Berechnung von $\mathbf{g}_{02}$ und die dritte Reihe die Berechnung von $\mathbf{g}_{23}$.



Die Punkte zeigen an, welche Adresse das Adressregister bei der Berechnung von $\mathbf{g}_{\mu\nu}$ enthält. Es wird wie folgt verfahren:

- Register **X** enthält die Adresse des $\mathbf{g}_{\mu\nu}$, das gerade berechnet wird.
- Register **Y** enthält stets die Adresse von $\mathbf{t}_{00}$.
- Bei der Kombination der μ-ten Zeile von **S** mit den vier Spalten von **T** enthält Register **Z** die Adresse von $\mathbf{g}_{\mu 0}$.

Damit liegen auch die relativen Adressen (*offsets*) relativ zu den in den Adressregistern enthaltenen Adressen fest, mit welchen auf die Matrixkoeffizienten zugegriffen wird. Beispielsweise hat $\mathbf{s}_{22}$ bezüglich **Y** die relative Adresse 2 und $\mathbf{t}_{23}$ hat bezüglich Register **Z** die relative Adresse 11.

Kennt man den Schlüssel **S** nicht, aber auch **keinen** Klartext **T**, kann man den verwendeten Schlüssel nur durch Ausprobieren aller Schlüssel und Klartexte herausbekommen. Das ist bei der großen Zahl der möglichen Matrizen allerdings ein aufwendiges Unterfangen. Hat man jedoch Zugang zu einem Klartext **T** und einem Geheimtext **G**, dann hat man in

$$\mathbf{ST} = \mathbf{G}$$

ein System linearer Gleichungen für die Koeffizienten $\mathsf{s}_{\mu\nu}$ von $\mathbf{S}$. Ist man vom Glück begünstigt und ist der Klartext $\mathbf{T}$ sogar regulär, dann ergibt sich der Schlüssel direkt als

$$\mathbf{S} = \mathbf{G}\mathbf{T}^{-1}$$

Ist das nicht der Fall, dann lassen sich aus den Gleichungen einige der $\mathsf{s}_{\mu\nu}$ berechnen, für die Übrigen erhält man immerhin einige lineare Gleichungen. Wenn nur wenige Koeffizienten unbestimmt bleiben, kann man hier mit Durchprobieren zum Ziel kommen.

Ist zwar $\mathbf{S}$ unbekannt, hat man jedoch Zugang zum Verschlüsselungsprozess, etwa durch einen Mikroprozessor, der mit dem Verfahren programmiert ist, kann man also den Klartext selbst bestimmen, dann wählt man als Klartext einfach die identische Matrix $\mathbf{I}$ und erhält damit

$$\mathbf{S} = \mathbf{S}\mathbf{I} = \mathbf{G}$$

Das bedeutet also, daß bei diesem Verfahren sowohl Schlüssel als auch alle Klartexte im Verborgenen bleiben müssen.

Zum Schluss wird noch das Chiffrierverfahren von HILL in das oben gegebene Chiffrierschema eingeordnet. Man kann wie folgt vorgehen.

($\boldsymbol{i}$) $\mathsf{T} = \mathbb{K}_{2^8}$
($\boldsymbol{ii}$) $\boldsymbol{\mathcal{T}} = \boldsymbol{\mathcal{E}}^{16}_{\mathbb{K}_{2^8}}$
($\boldsymbol{iii}$) $\mathsf{G} = \mathbb{K}_{2^8}$
($\boldsymbol{iv}$) $\boldsymbol{\mathcal{G}} = \boldsymbol{\mathcal{E}}^{16}_{\mathbb{K}_{2^8}}$
($\boldsymbol{v}$) $\boldsymbol{\mathcal{K}} = \big\{\, \mathbf{K} \in \boldsymbol{\mathcal{M}}_{\mathbb{K}_{2^8}} \;\big|\; \mathrm{Rang}(\mathbf{K}) = 4 \,\big\}$, d.h. die regulären Matrizen aus $\boldsymbol{\mathcal{M}}_{\mathbb{K}_{2^8}}$
($\boldsymbol{vi}$) Die Abbildung $\boldsymbol{\Omega} \colon \boldsymbol{\mathcal{K}} \longrightarrow \mathbf{P}\big\langle \boldsymbol{\mathcal{E}}^{16}_{\mathbb{K}_{2^8}} \big\rangle$ ist gegeben als

$$\boldsymbol{\Omega}(\mathbf{K}) = \boldsymbol{\Lambda}^{-1} \circ \boldsymbol{\Psi}_{\mathbf{K}} \circ \boldsymbol{\Lambda} \quad \text{d.h.} \quad \boldsymbol{\Omega}(\mathbf{K})\big((\boldsymbol{f}_\nu)_{\nu\in N}\big) = \boldsymbol{\Lambda}^{-1}\Big(\boldsymbol{\Lambda}\big((\boldsymbol{f}_\nu)_{\nu\in N}\big)\mathbf{K}\Big)$$

mit der Chiffrierabbildung $\boldsymbol{\Psi}_{\mathbf{K}}$ des HILLschen Systems.

Der Schlüsselbereich besteht aus allen regulären, d.h. invertierbaren, 4×4-Matrizen mit Koeffizienten aus dem Körper $\mathbb{K}_{2^8}$, die Chiffrierabbildungen $\boldsymbol{\Psi}_{\mathbf{K}}$ sind daher bijektiv und damit schließlich auch die Chiffrierabbildungen $\boldsymbol{\Omega}(\mathbf{K})$.

In der Praxis besteht das Alphabet natürlich nicht aus $\mathbb{K}_{2^8}$, sondern aus $\mathbb{K}_2^8$. Also nicht aus Elementen des endlichen Körpers $\mathbb{K}_{2^8}$, der wohl nur sehr wenigen Anwendern des Verfahrens geläufig ist, sondern aus schlichten Bytes oder Bitoktetts. Es ist lediglich eine Angelegenheit der Interpretation, die selbstverständlich auch formalisiert werden könnte. Ein klein wenig problematisch ist die Angelegenheit nur deshalb, weil die Polynome, aus welchen die Elemente des Körpers $\mathbb{K}_{2^8}$ hier nach Konstruktion bestehen (siehe Abschnitt A.5), abkürzend (und von ihrer wahren Natur dabei ablenkend) mit Bitoktetts oder noch kompakter mit zwei Hexadezimalziffern bezeichnet werden. Würde tatsächlich Polynomnotation verwendet, dann wären noch Übergangsabbildungen von $\mathbb{K}_{2^8}$ nach $\mathbb{K}_2^8$ und zurück einzuführen.

2.3. RSA

Im Jahre 1978 erlebte die Kryptographie eine Revolution, als *Ronald Rivest*, *Adi Shamir* und *Leonard Adleman* das erste Verschlüsselungssystem mit öffentlich bekanntem Schlüssel vorstellten. Man übertreibt sicher nicht, wenn man das System für einen der wichtigsten Beiträge zur Kryptographie überhaupt ansieht.

Das RSA-Chiffriersystem ist leicht zu beschreiben, es ist wie folgt aufgebaut. Es seien p und q zwei (große) Primzahlen, und es sei

$$n = pq \quad m = (p-1)(q-1)$$

Die internen Klartexte t und Geheimtexte g des Verfahrens sind die Elemente des Ringes $\mathbb{Z}_n$, d.h. es gilt $0 \leq t < n$ und $0 \leq g < n$. Weiterhin sei $e \in \mathbb{Z}_m^\star = \mathbb{Z}_m \setminus \{0\}$ so gewählt, daß

$$\mathrm{ggT}(e, m) = 1$$

gilt. D.h. e ist ein **invertierbares Element** des Ringes $\mathbb{Z}_m$, das mit m keine echten Teiler gemeinsam hat. Die Zahl e ist der Schlüssel des Verfahrens. Die Verschlüsselungsfunktion $\Psi_e : \mathbb{Z}_n \longrightarrow \mathbb{Z}_n$ ist gegeben als

$$\Psi_e(u) = u^e$$

dabei ist die Potenzierung natürlich im Ring $\mathbb{Z}_n$ auszuführen. Wenn also die Multiplikation von $\mathbb{Z}_n$ mit $\otimes$ bezeichnet wird, dann ist

$$\Psi_e(u) = \underbrace{u \otimes u \otimes \cdots \otimes u}_{e\text{-mal}} = \varrho_n(u^e)$$

Die Entschlüsselungsfunktion $\Psi_e^{-1} : \mathbb{Z}_n \longrightarrow \mathbb{Z}_n$ schließlich ist gegeben durch

$$\Psi_e^{-1}(u) = u^{\zeta_{p,q}(e)} = \Psi_{\zeta_{p,q}(e)}(u)$$

wobei auch diese Potenzierung natürlich mit der Arithmetik des Ringes $\mathbb{Z}_n$ durchzuführen ist:

$$\Psi_e^{-1}(u) = \underbrace{u \otimes u \otimes \cdots \otimes u}_{\zeta_{p,q}(e)\text{-mal}} = \varrho_n(u^{\zeta_{p,q}(e)})$$

Darin ist die Abbildung $\zeta_{p,q} : \mathbb{Z}_m^\star \longrightarrow \mathbb{Z}_m^\star$ gegeben als

$$\zeta_{p,q}(e) = e^{-1}$$

wobei e^{-1} das zu e inverse Element **im Ring** $\mathbb{Z}_m$ ist, d.h. wenn die Multiplikation von $\mathbb{Z}_m$ mit $\odot$ bezeichnet wird, dann gilt

$$e \odot e^{-1} = 1$$

Ohne die Kenntnis von p und q und damit auch von m ist der Ring $\mathbb{Z}_m$ nicht bekannt, in dem das zu e inverse Element e^{-1} zu berechnen ist. Um aber p und q zu bestimmen ist die Zahl n in ihre beiden Primfaktoren p und q zu zerlegen, und in der Praxis werden die beiden Primzahlen so gewählt, daß diese Zerlegung unmöglich ist, jedenfalls mit den heutigen rechnerischen Mitteln. Beispielsweise sollen p und q Zahlen mit mindestens 100 Dezimalziffern sein, doch ist die Größe