Kryptologie – von einer Geheimwissenschaft zu einer Wissenschaft von den Geheimnissen

Dr. Jörg Vogel

WS 2006/07

Vorwort

Dieses Dokument wurde als Skript für die auf der Titelseite genannte Vorlesung erstellt und wird jetzt im Rahmen des Projekts „Vorlesungsskripte der Fakultät für Mathematik und Informatik“ weiter betreut. Das Dokument wurde nach bestem Wissen und Gewissen angefertigt. Dennoch garantiert weder der auf der Titelseite genannte Dozent, die Personen, die an dem Dokument mitgewirkt haben, noch die Mitglieder des Projekts für dessen Fehlerfreiheit. Für etwaige Fehler und dessen Folgen wird von keiner der genannten Personen eine Haftung übernommen. Es steht jeder Person frei, dieses Dokument zu lesen, zu verändern oder auf anderen Medien verfügbar zu machen, solange ein Verweis auf die Internetadresse des Projekts `http://uni-skripte.lug-jena.de/` *enthalten ist.*

Diese Ausgabe trägt die Versionsnummer 3658 und ist vom 29. März 2012. Eine neue Ausgabe könnte auf der Webseite des Projekts verfügbar sein.

Jeder ist dazu aufgerufen, Verbesserungen, Erweiterungen und Fehlerkorrekturen für das Skript einzureichen bzw. zu melden oder diese selbst einzupflegen – einfach eine E-Mail an die Mailingliste `<uni-skripte@lug-jena.de>` *senden. Weitere Informationen sind unter der oben genannten Internetadresse verfügbar.*

Hiermit möchten wir allen Personen, die an diesem Skript mitgewirkt haben, vielmals danken:

- *Jörg Sommer* `<joerg@alea.gnuu.de>` *(2006/07)*
- *Jens Kubieziel* `<jens@kubieziel.de>` *(2006)*
- *Michael Preiss (2006/07)*
- *Christine List (2007)*

Inhaltsverzeichnis

1. Einführung in die Kryptologie

Definition 1.1 (Kryptologie)
Kryptologie ist die Wissenschaft von der sicheren Übertragung (und Speicherung) von Nachrichten.

Dabei unterscheidet man zwei Teilgebiete:

- die **Kryptographie** als die Kunst des Verschlüsselns und
- die **Kryptoanalyse** als die Kunst des Codebrechens.

Die Übertragung der Nachricht erfolgt über **Kanäle**, wie z. B. Boten, Kupferleitungen, Glasfasern oder drahtlose Verbindungen. Dabei kann es zu verschiedenen Beeinträchtigungen kommen:

- zufällige Störungen – der Bote stürzt in eine Felsspalte.
- systematische (physikalisch bedingte) Störungen – im Wald werden alle Boten gefangen genommen.
- passive Beeinträchtigungen – während der König die Botschaft seinem Schreiber diktiert, hört der Spion hinterm Vorhang alles mit; Abhören von Telefongesprächen, Auslesen von Speichermedien.
- aktive Beeinträchtigungen – während der Bote sein Nickerchen macht, tauscht ein Spion die Botschaft aus; Fälschen von Nachrichten und Daten.

Mit den ersten beiden Punkten, also wie sichert man Nachrichten gegen Störeinflüsse, beschäftigt sich die Kodierungstheorie. Die passiven und aktiven Beeinträchtigungen sind Gegenstand der Kryptologie.

Aus diesen Beeinträchtigungen ergeben sich Anforderungen an kryptologische Verfahren:

- **Geheimhaltung/Vertraulichkeit** – Lesen der Nachricht für Unbefugte möglichst schwierig gestalten.
- **Authentifizierung** – Empfänger kann die **Identität** des Senders prüfen; **Authentisierung** – Sender kann seine Identität einem Empfänger gegenüber beweisen; Empfänger weiß, dass die Nachricht nicht von einem Dritten stammt.
- **Integrität** – Nachricht wurde während der Übertragung nicht verändert (Austauschen, Weglassen oder Hinzufügen von Teilen).

- **Verbindlichkeit** – Sender kann nicht leugnen, dass die Nachricht von ihm stammt.

Im Laufe der Zeit hat sich das Anwendungsgebiet für Kryptologie gewandelt. Früher wurde sie vorwiegend zur Geheimhaltung z. B. militärischer Anwendungen eingesetzt, während heute durch den enormen Zuwachs der Kommunikation in offenen Netzwerken Kryptologie auch verstärkt im zivilen Bereich eingesetzt wird.

1.1. Grundbegriffe

Der **Sender** – in der Kryptologie wird dieser mit **Alice**[24] bezeichnet – verschlüsselt seine **Nachricht**[1] mit Hilfe eines **Schlüssels** unter Verwendung eines **Verschlüsselungsverfahrens**[2] und erhält so einen **Geheimtext**[3] für die Nachricht. Diesen sendet er dem Empfänger – in der Kryptologie wird dieser mit **Bob** bezeichnet –, der durch Anwendung eines **Entschlüsselungsverfahrens**[4] wieder die Nachricht rekonstruieren kann.

Während der Übertragung könnte ein **passiver Angreifer** – in der Kryptologie wird dieser mit **Eve**[5] bezeichnet – aus dem Geheimtext die Nachricht oder Teile davon rekonstruieren und aus diesem Wissen Vorteile ziehen. Oder ein **aktiver Angreifer** – in der Kryptologie wird dieser mit **Mallory**[6] bezeichnet – könnte den Geheimtext so verändern, dass der Klartext eine andere Bedeutung bekommt.

Definition 1.2 (Kryptosystem)
Ein **Kryptosystem** $\mathcal{S}$ ist ein Fünftupel $\mathcal{S} = (\mathcal{M}, \mathcal{C}, \mathcal{K}, E, D)$ bestehend aus:

- einer Menge von Nachrichten, die als **Klartextraum** $\mathcal{M}$ ($\subseteq \Sigma^*$) bezeichnet wird; ein Element dieser Menge ist eine **Nachricht** $m \in \mathcal{M}$.
- dem **Geheimtextraum** $\mathcal{C}$ ($\subseteq \Gamma^*$), der Menge aller **Kryptogramme** $c \in \mathcal{C}$,
- dem **Schlüsselraum** $\mathcal{K}$, wobei jeder **Schlüssel** $k \in \mathcal{K}$ ein Paar $k = (k_e, k_d)$ ist, von dem k_e zur Verschlüsselung und k_d zur Entschlüsselung verwendet wird.
- einem **Verschlüsselungsalgorithmus** $E\colon \mathcal{M} \times \mathcal{K} \to \mathcal{C}$ für den gilt $E(m, k_e) = c$, wobei für zwei unterschiedliche Nachrichten m_1 und m_2 gilt: $E(m_1, k_e) \neq E(m_2, k_e)$ und
- einem **Entschlüsselungsalgorithmus** $D\colon \mathcal{C} \times \mathcal{K} \to \mathcal{M}$ mit $D(c, k_d) = m$.

Aus diesen Festlegungen ergeben sich folgende Konsequenzen für die Kommunikation zwischen Alice und Bob:

1. Vor der Übertragung müssen sich beide über das Verfahren einigen.

[1] Nachricht: auch **Klartext** oder **plain text** genannt
[2] Verschlüsselungsverfahren: auch als **Chiffrierung** oder **encryption** bezeichnet
[3] Geheimtext: auch **Chiffretext** oder **Kryptogramm** genannt
[4] Entschlüsselungsverfahren: auch als **Dechiffrierung** oder **decryption** bezeichnet
[5] Eve: vom englischen Wort *eavesdropper* für Horcher oder Lauscher
[6] Mallory: vom englischen Wort *malicious* für bösartig

2. Viele Schlüssel sind eine notwendige Voraussetzung (großer Schlüsselraum) für die Sicherheit eines Verfahrens.

3. Vor der Übertragung müssen sie sich auf ein Schlüsselpaar $k = (k_e, k_d)$ einigen.

4. Der Schlüssel k_d muss geheim gehalten werden. Bei symmetrischen Verfahren (siehe Abschnitt 1.3) ist $k_d = k_e$. Also müssen „beide“ Schlüssel geheim gehalten werden.

Da für den Austausch des Schlüssels zwischen Alice und Bob eine sichere Übertragung – sozusagen eine verschlüsselte Verbindung – notwendig ist, stellt sich die Frage, warum die beiden nicht gleich die Nachricht über diese sichere Verbindung übertragen.

- Im Allgemeinen ist die Nachricht wesentlich länger als der Schlüssel, d. h. der Schlüsselaustausch kann über andere (langsamere, aber sicherere) Kanäle als der Nachrichtenaustausch erfolgen. Alice lässt Bob das Kennwort durch einen Boten zukommen und der Austausch der Nachricht geschieht dann per E-Mail.

- Der Zeitpunkt der Schlüsselübergabe ist frei wählbar. Es können also bei einem persönlichen Treffen mehrere Schlüssel für spätere Verbindungen vereinbart werden.

- Mit demselben Schlüssel lassen sich mehrere Nachrichten verschlüsseln.

Definition 1.3 (Kerckhoffs’ Prinzip)
Das **Kerckhoffs’ Prinzip** besagt, dass die **Sicherheit** eines Kryptosystems nur von der Geheimhaltung des Schlüssels jedoch nicht von der Geheimhaltung des Algorithmus’ abhängt.

Dieses Prinzip wurde 1883 von AUGUSTE KERCKHOFFS in seiner Arbeit „La cryptographie militaire“[7] niedergeschrieben. Er formulierte das Prinzip in sechs Einzelpunkten.

Beispiel 1.1 (praktische Kryptographie)
Der **Freimaurercode** (zweite Zeile in Abbildung 1.1) arbeitet ohne Schlüssel. Jeder Buchstabe des Alphabets wird durch ein bestimmtes Geheimzeichen ersetzt. Damit ist der Freimaurercode kein Kryptosystem, da das Kerckhoffs’ Prinzip verletzt ist: Die Sicherheit des Kryptosystems hängt von der Geheimhaltung des Algorithmus’ ab.

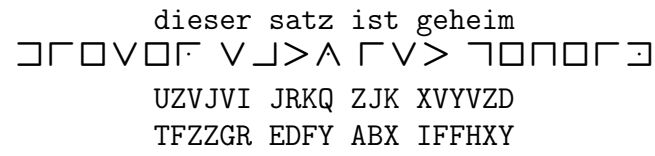

Abbildung 1.1.: Verschlüsselung der ersten Zeile mit dem Freimaurercode (2. Zeile), der Caesar-Chiffre (3. Zeile) und einem One-Time-Pad (4. Zeile)

Bei der **Caesar-Chiffre** (dritte Zeile in Abbildung 1.1) wird jeder Buchstabe des Alphabets für sich durch den Buchstaben, der k Positionen weiter hinten steht, ersetzt.

[7] Nachzulesen bei Fabien Petitcolas

$k \in \{1,2,\ldots,25\} = \mathcal{K}$ ist der Schlüssel dieses Kryptosystems – im obigen Beispiel ist $k = 17$. Jedoch ist diese Art der Verschlüsselung nicht sicher. Dazu später mehr.

Der **One-Time-Pad** (vierte Zeile in Abbildung 1.1) ist ähnlich dem Caesarcode, jedoch wird hierbei (unabhängig und gleichverteilt) für jedes einzelne Zeichen der Nachricht eine Zahl $k \in \{0,1,\ldots,25\}$ gewählt, um die das Zeichen verschoben wird. Der Schlüssel für eine Nachricht $m = (m_1, m_2, \ldots, m_n)$ ist also ein Tupel $k = (k_1, k_2, \ldots, k_n)$ der Länge n. Dieses Kryptosystem schützt sich nicht nur durch den großen Schlüsselraum 26^n, sondern auch dadurch, dass jeder beliebige Text mit einem geeigneten Schlüssel erzeugt werden kann.

Zur Einstufung einer **Kryptoanalyse** trifft man die folgenden qualitativen Unterscheidungen:

- **vollständiges Aufbrechen** (*engl.* total break) heißt, dass der Schlüssel k_d entdeckt wird, woraufhin jede Nachricht m, die mit k_e verschlüsselt wird, entschlüsselt werden kann.
- **globale Deduktion** (*engl.* global deduction) bedeutet, dass ohne Kenntnis von k_d ein zu $D(c, k_d)$ äquivalentes Verfahren $\widetilde{D}$ gefunden wird, so dass man jede Nachricht $m = D(c, k_d)$ durch $m = \widetilde{D}(c)$ rekonstruieren kann.
- **lokale Deduktion** (*engl.* instance or local deduction) bezeichnet das Finden eines Klartexts für einen einzelnen Chiffretext.
- **Informationsdeduktion** (*engl.* information deduction) bedeutet, dass einige Informationen über den Schlüssel oder den Klartext gewonnen werden können.

Ebenso gibt es eine Unterscheidung der verschiedenen Arten eines **Angriffs** auf ein Kryptosystem:

Cyphertext-Only-Angriff: Der Kryptoanalytiker verfügt über eine gewisse Menge von Geheimtexten.

Known-Plaintext-Angriff: Für einen gewissen Chiffretext ist der Klartext bekannt.

Chosen-Plaintext-Angriff: Für einen beliebigen Klartext ist es möglich, sich den zugehörigen Chiffretext zu besorgen. (Typisch für Public-Key-Verfahren)

Adaptive-Chosen-Plaintext-Angriff: Spezialfall des obigen Angriffszenarios. Hier wählt der Kryptoanalytiker einen Klartext basierend auf dem Ergebnis der vorigen Verschlüsselung.

Chosen-Ciphertext-Angriff: Verschiedene Chiffrate können entschlüsselt werden und es besteht Zugriff zum entschlüsselten Text.

Chosen-Key-Angriff: Man hat Wissen über die Beziehungen verschiedener Schlüssel untereinander (i. d. R. nicht praktisch relevant)

Rubber-Hose-Cryptanalysis: Angriff durch Gewalt/Erpressung/Entführung

Brute-Force-Angriff: Alle möglichen Schlüssel austesten. Dieses Verfahren sollte das effektivste sein.

Abhängig von der **Sicherheit** des Ver-/Entschlüsselungsverfahrens kann man Kryptosysteme in die Kategorien „uneingeschränkt sicher“ (z. B. One-Time-Pad), „praktisch sicher“ (in praktikabler Zeit keine Lösung; die Tageszeitung von Morgen erst nächste Woche entschlüsseln) und „unsicher“ (z. B. Caesarcode) einteilen.

Derzeit ist das **One-Time-Pad** das einzige bekannte, uneingeschränkt sichere Kryptoverfahren. In der Praxis findet es hauptsächlich im militärischen Bereich und in hoch sicheren Umgebungen Anwendung.

Alle weiteren Kryptosysteme sind mit einem **Ciphertext-Only-Angriff** verwundbar. Man probiert einfach jeden möglichen Schlüssel und prüft, ob der resultierende Klartext eine Bedeutung hat. Einen derartigen Angriff nennt man **Brute-Force-Angriff**, da diese nur mit roher Gewalt (Durchtesten aller Kombinationen) funktioniert.

1.2. Beispiel einer praktischen Kryptoanalyse

In der Regel kann man nicht von der Größe des Schlüsselraums eines Kryptosystems auf die Sicherheit des Systems schließen. Folgendes Beispiel soll dies verdeutlichen.

Ein Kryptosystem, das jedem Buchstaben des Klartexts ein Zeichen des Chiffretexts zuordnet, nennt man **monoalphabetisch**, d. h. der Schlüssel ist eine Permutation der Chiffretextbuchstaben. Es gibt also $26! = 403\,291\,461\,126\,605\,635\,584\,000\,000$ Schlüssel.

Klartextbuchstabe	a	b	c	d	...	x	y	z
Schlüssel	V	J	C	E	...	A	H	D

Trotz dieses großen Schlüsselraums ist es recht einfach die Verschlüsselung durch eine **Häufigkeitsanalyse** zu brechen. Aus der Größe des Schlüsselraums kann man also keine unmittelbaren Rückschlüsse auf die Sicherheit des Kryptosystems ziehen.

```
CJ UAFFC CHZ WAZZ CHZCZ CJCN, LCT JDUOZ NAZGC SAUTC LHC JACDVC
PZYCTLTOJJCZ QPT WPCUNC GCFTAGCZ UAFFC, LCJJCZ VTACRFC AXCT ZPZ QP
CZLC GHZGCZ, JO LAJJ CT QPT ATXCHF HWWCT PZFAPGNHDUCT IATL. LA LADUFC
LCT UCTT LATAZ, HUZ APJ LCW RPFFCT QP JDUARRCZ, AXCT LCT CJCN WCTVFC,
LAJJ VCHZ GPFCT IHZL ICUFC, NHCR ROTF PZL WADUFC JHDU APR LCZ ICG
ZADU XTCWCZ: LOTF, WCHZFC CT, VOCZZFC CT SA JFALFWPJHVAZF ICTLCZ.
```

Abbildung 1.2.: Ein Beispiel eines Geheimtexts, der sich leicht mit Häufigkeitsanalyse brechen lässt.

1. Schritt: Die Buchstaben im Geheimtext zählen. Für den Text in Abbildung 1.2 ergibt sich:

Bst.	Wsk.	Bst.	Wsk.	Bst.	Wsk.	Bst.	Wsk.
E	17,40 %	D	5,08 %	O	2,51 %	V	0,67 %
N	9,78 %	H	4,76 %	B	1,89 %	ß	0,31 %
I	7,55 %	U	4,35 %	W	1,89 %	J	0,27 %
S	7,27 %	L	3,44 %	F	1,66 %	Y	0,04 %
R	7,00 %	C	3,06 %	K	1,21 %	X	0,03 %
A	6,51 %	G	3,01 %	Z	1,13 %	Q	0,02 %
T	6,15 %	M	2,53 %	P	0,79 %		

Tabelle 1.1.: Häufigkeitsverteilung der Buchstaben der deutschen Sprache. Die Buchstaben ä, ö, ü wurden wie ae, oe, ue gezählt. *Quelle: Wikipedia*

Buchstabe:	C	Z	T	A	F	L	J	...
Häufigkeit:	62	31	29	27	22	20	19	...

2. Schritt: Für natürliche Sprachen wie Deutsch ergibt sich in großen Texten eine signifikante Verteilung der Buchstaben (Tabelle 1). Anhand der bekannten Verteilung der Buchstaben in der Sprache und der Häufigkeitsanalyse aus dem 1. Schritt, kann man die Klartextbuchstaben für einige Geheimtextzeichen bestimmen. So steht z. B. in einem deutschen Text das am häufigsten auftretende Zeichen für das E. Für den Text in Abbildung 1.2 ist also `e` durch `C` verschlüsselt.

Da die Zeichenfolge `ZZ` vorkommt, ist ein weiterer Ansatz, dass das `Z` dem `n` entspricht.

Aus den Teilen `CHZ` (`eHn`) und `CHZCZ` (`eHnen`) kann man schließen, dass das `H` dem `i` entspricht.

Auffällig ist die Häufung des Wortes `CT` (`eT`). Dies könnte `er` oder `es` sein.

Ebenso fällt das Wort `LCT` auf. Möglich: `der` oder `des`. Da „der“ häufiger in deutschen Texten vorkommt, verwenden wir `r` für `T` und `d` für `L`.

3. Schritt: Lücken schließen:

- `iUn` ist sehr wahrscheinlich `ihn`; `U` entspricht `h`.
- `eJ` ist nicht `er` oder `ei` (da `r` und `i` schon vergeben), also `es`; `J` entspricht `s`
- `dAJJ` ist `dass`; `A` ist `a`
- `haFFe` ist `hatte`; `F` ist `t`
- Erster Teilsatz: „`es hatte ein Wann einen eseN,`“; Aus dem Kontext folgt: `W` entspricht `m` und `N` ist `l`.
- weiter: „der sDhOn lanGe Sahre die saeDVe“; `D` entspricht `c`, `O` ist `o`, `G` ist `g` und `S` ist `j` und `V` ist `k`.

```
es hatte ein mann einen esel, der schon lange jahre die saecke
unverdrossen zur muehle getragen hatte, dessen kraefte aber nun zu
ende gingen, so dass er zur arbeit immer untauglicher ward. da dachte
der herr daran, ihn aus dem futter zu schaffen, aber der esel merkte,
dass kein guter wind wehte, lief fort und machte sich auf den weg
nach bremen: dort, meinte er, koennte er ja stadtmusikant werden.
```

Abbildung 1.3.: Entschlüsselung des Textes aus Abbildung 1.2

1.3. Einteilung der Verschlüsselungsverfahren

Man kann die Verschlüsselungsverfahren anhand verschiedener Merkmale unterteilen. So unterscheidet man anhand der Schlüssel k_e und k_d zwischen symmetrischer und asymmetrischer Verschlüsselung. Ein Kryptosystem, bei dem für die Entschlüsselung der gleiche (oder ein in Polynomialzeit konstruierbarer) Schlüssel wie für die Verschlüsselung verwendet wird, bezeichnet man als **symmetrisch**. Unterscheidet sich der Schlüssel für die Entschlüsselung von dem für die Verschlüsselung, bezeichnet man das Verfahren als **asymmetrisch**.

Weiterhin unterscheidet man die Verfahren anhand der eingesetzten Schlüssel. Bei einer **Stromchiffre** wird aus einem Hauptschlüssel ein Schlüsselstrom generiert, wobei je ein Element des Schlüsselstroms für die Verschlüsselung eines Klartextblocks verwendet wird. Bei einer **Blockchiffre** ist der Schlüssel für alle Blöcke gleich. In der Regel ist die Blocklänge bei Stromchiffren wesentlich kürzer (32 statt 256 Bit), wodurch sie vorwiegend zur on-the-fly-Verschlüsselung, z. B. einer Videokonferenz oder eines Telefonats, verwendet werden.

Ein drittes Unterscheidungsmerkmal ist die Art der Verschlüsselung. Die zwei Grundverfahren sind die Substitution und die Transposition. Wird jede Einheit durch einen bestimmten Wert (u. U. aus einem anderen Alphabet) ersetzt, ist dies eine **Substitution**. Bei einer **Transposition** hingegen wird der Geheimtext durch Vertauschung der Einheiten im Klartext erzeugt. Die Transposition ist also eine Permutation der Einheiten des Klartexts, wobei keine neuen Symbole hinzukommen, insbesondere bleibt die Häufigkeitsverteilung der Einheiten gleich.

Heutige Verfahren verwenden eine Kombination aus beiden Grundverfahren.

2. Substitionsverfahren

2.1. Monoalphabetische Substitutionen

Bei einer **monoalphabetischen Verschlüsselung** wird jeder Buchstabe des Klartextalphabets Σ immer durch denselben Buchstaben des Geheimtextalphabets Γ ersetzt. Sei o. B. d. A. $\Sigma = \Gamma = \{a, b, c, \ldots, x, y, z\}$ bzw. sei das Alphabet in der Form $X = \{0,1,2, \ldots, 23, 24, 25\}$ gegeben; es gilt: $|X| = 26$. Für Alphabete mit n Zeichen, sei die Darstellung $X = \{0,1,2, \ldots, n-2, n-1\}$.

Die Verschlüsselung ist also eine Permutation π: $X \to X$ des Alphabets, die gleichzeitig der Schlüssel ist. Aus einer Nachricht $m = m_1 m_2 \ldots m_k$ wird so der Geheimtext $c = \pi(m_1)\pi(m_2)\ldots\pi(m_k)$. Es gibt hier $n!$ verschiedene Schlüssel, aber wir wissen bereits aus Abschnitt 1.2, dass dieses Verfahren kryptographisch auf Grund der Häufigkeitsverteilung der Buchstaben auch bei **Ciphertext-Only-Angriffen** unsicher ist.

2.1.1. Verschiebechiffre (Caesar-Chiffre)

Die **Verschiebechiffre** – auch **Caesar-Chiffre** genannt – ist die einfachste Form der monoalphabetischen Substitution. Jeder Buchstabe wird durch seinen k-ten Nachfolger ersetzt, wobei man wieder bei A beginnt, wenn man Z überschreitet.

Anschaulich kann man sich eine Verschlüsselungsvorschrift so vorstellen, dass unter den Buchstaben A bis Z die Buchstaben beginnend mit dem k-ten Buchstaben bis Z und weiter von A bis zum $(k-1)$-ten Buchstaben stehen. Jeder Buchstabe des Klartextes wird durch den Buchstaben, der unter ihm in der zweiten Zeile steht, ersetzt.

Mathematisch ausgedruckt, kodiert man die Buchstaben A bis Z mit den Zahlen 0 bis 25 und ordnet jedem Buchstaben den durch die Abbildung $x \mapsto x + k \pmod{26}$ beschriebenen Buchstaben zu.

Der Schlüssel für die Verschlüsselung ist dabei $k \in \{0,1, \ldots,25\}$, die Anzahl der Positionen, um die die Buchstaben verschoben werden.

Das Verfahren hat jedoch den Nachteil, das nur wenige (n) Schlüssel verfügbar sind und das die Verschlüsselung gebrochen ist, sobald *ein* Buchstabe übersetzt ist. Dies lässt sich wie in Abschnitt 1.2 gesehen, sehr leicht mit der Häufigkeitsanalyse der Buchstaben bewerkstelligen.

2.1.2. Tauschchiffren

Tauschchiffren sind eine Verallgemeinerung der Verschiebechiffren. Die Buchstaben werden ebenfalls durch Zahlen kodiert, mit denen dann gerechnet wird. Jedes Zeichen $x \in \{0, \ldots, n-1\}$ wird durch die **affine Abbildung** $x \mapsto a \cdot x + b \pmod{n}$ in das

Geheimzeichen überführt. Der Schlüssel ist dabei das Paar $(a, b) \in \{0, \ldots, n-1\}^2$. Bei den Verschiebechiffren ist $a = 1$.

Damit die Entschlüsselung möglich ist, muss die Abbildung bijektiv sein. Dazu müssen a und n teilerfremd sein. Es gibt also $\varphi(n)$[1] Möglichkeiten a zu wählen und insgesamt $\varphi(n) \cdot n$ Schlüssel (a, b).

Ein kurzes Gegenbeispiel, um das Problem zu verdeutlichen: Wählt man $n = 26$, $a = 13$ und $b = 0$, so wird das Zeichen 2 als $E(2) = 13 \cdot 2 + 0 \pmod{26} = 0$ als auch das Zeichen 0 als $E(0) = 0 \cdot 13 + 0 \pmod{26}$ verschlüsselt. Eine Entschlüsselung des Zeichens 0 ist also nicht möglich.

Zur Entschlüsselung sei $y \in \{0, \ldots, n-1\}$ der Geheimtextbuchstabe. Dann ist $x = (y - b)a' \pmod{26}$ mit $aa' \pmod{26} \equiv 1$ das Ergebnis der Entschlüsselung.

Beispiel 2.1
Der Klartext „text“ wird mit dem Schlüssel (5,13) auf den Geheimtext „EHYE“ abgebildet.

Buchstabe	Codierung x	$5 \cdot x + 13 \pmod{26}$	Decodierung
t	19	4	E
e	4	7	H
x	23	24	Y
t	19	4	E

Die Nachteile der Tauschchiffren sind:

- Häufigkeitsverteilung der natürlichen Sprache bleibt erhalten und
- Verschlüsselung ist geknackt, wenn *zwei* Buchstaben übersetzt sind

Die Tauschchiffren werden in einer allgemeineren Form nochmal im Abschnitt 4.4 besprochen.

2.2. Homophone Substitutionen

Homophone Substitutionen versuchen die strukturelle Schwäche der monoalphabetischen Verschlüsselung, die durch die Häufigkeitsverteilung der Buchstaben bei natürlichen Sprachen gegeben ist, aufzuheben.

Dazu führt man ein neues und *größeres* Geheimtextalphabet Y und eine Abbildung $f\colon X \to \mathfrak{P}(Y)$ ein, wobei verschiedene Buchstaben disjunkten Teilmengen entsprechen. Für eine Nachricht $m = m_1 m_2 \ldots m_k$ ist der Geheimtext $c = c_1 c_2 \ldots c_k$, wobei c_i zufällig aus $f(m_i)$ gewählt wird. (Italien um 1400).

Beispiel 2.2
Für das Klartextalphabet $X = \{a, b, c\}$, wobei die Buchstaben mit den Wahrscheinlichkeiten 0,5 %, 0,35 % und 0,15 % vorkommen, verwenden wir das Geheimtextalphabet

[1] Eulersche φ-Funktion; siehe Abschnitt 4.3

	x_1	x_2	$\dots$	x_i	$\dots$	x_n
x_1	$y_{\frac{n^2}{9}}$	$y_{\frac{7n^2}{8}}$	$\dots$	$y_{\frac{n^2}{4}}$	$\dots$	$y_{\frac{n^2}{2}}$
x_2	y_{n^2}	$\ddots$				
$\vdots$						
x_j						
$\vdots$					$\ddots$	$\vdots$
x_n					$\dots$	$y_{\frac{n^2}{12}}$

Tabelle 2.1.: Zuordnung von n Zeichen des Klartextalphabet auf n^2 Zeichen des Geheimtextalphabet

$Y = \{0,1,\dots,7\}$. Dazu definieren wir die Abbildung f wie folgt

$$f(x) = \begin{cases} \{0,3,4,7\} & : x = a \\ \{1,5,6\} & : x = b \\ \{2\} & : x = c \end{cases}$$

Der Buchstabe b wird also durch die drei Zeichen 1, 5 und 6 codiert. Das Wort cbba könnte z. B. als 2514 oder 2663 verschlüsselt werden.

Der Vorteil einer solchen homophonen Substitution besteht darin, dass die Häufigkeitsverteilung der Klartextbuchstaben zerstört wird, wenn für alle $x \in X$, wobei $p(x)$ die relative Häufigkeit ist, mit der der Buchstabe x in Texten auftritt, $f(x)$ so gewählt wird, dass $\frac{p(x)}{|f(x)|}$ für alle x annähernd gleich ist!
Der Effekt des Ganzen ist, dass im Geheimtext alle Buchstaben des Geheimtextalphabets Y etwa gleichwahrscheinlich auftreten!

Lemma 2.1
Es gibt ein homophones Verschlüsselungsverfahren, bei dem es für jeden Geheimtext c mindestens zwei verschiedene Schlüssel gibt, die c in zwei verschiedene, sinnvolle Klartexte überführen.

BEWEIS:
Sei X mit $|X| = n$ das Klartextalphabet und Y mit $|Y| = n^2$ das Geheimtextalphabet. Der Schlüsselraum wird durch eine $n \times n$-Matrix beschrieben. Die Zeilen und Spalten werden mit den Buchstaben aus X indiziert und die Eintragungen sind die (zufällig angeordneten) Buchstaben von Y, wie in Tabelle 2.1. Wir definieren zwei Abbildungen:

$$f_1(x_j) = \{\, y \in Y \mid y \text{ steht in der } j. \text{ Zeile} \,\}$$
$$f_2(x_i) = \{\, y \in Y \mid y \text{ steht in der } i. \text{ Spalte} \,\}$$

Es sei $m = m_1 m_2 \dots m_k$ der zu verschlüsselnde Klartext und $l' = l'_1 l'_2 \dots l'_k$ ein weiterer sinnvoller Klartext gleicher Länge.

Als Verschlüsselung für m_i verwenden wir den Eintrag in der Matrix, der im Schnittpunkt von $m_i = x_r$ und $l_i = x_s$ (r-ter Zeile und s-ter Spalte) liegt.

$$m_i \mapsto c_i \in f_1(m_i) \cap f_2(l_i)$$ ■

Ein **Brute-force-Angriff** bleibt bei solch einer Verschlüsselung prinzipiell wirkungslos, weil jeder Geheimtext in zwei verschiedene, *sinnvolle* Klartexte übersetzt werden kann!

Jedoch bleibt weiterhin der Nachteil bestehen, dass für alle Buchstaben $x \in X$ die Anzahl der möglichen Verschlüsselungen $|f(x)|$ gleich groß ist. Damit ist die Forderung, dass $\frac{p(x)}{|f(x)|}$ für alle x annähernd gleich ist, verletzt. Aber die Idee ist, dass der Angreifer nicht sagen kann, welcher der beiden konkurrierenden Klartexte der richtige ist.

Beispiel 2.3
Das Klartextalphabet X sei das deutsche Alphabet und wir wählen als Geheimtextalphabet $Y = \{00, 01, \ldots, 10, 11, \ldots, 99\}$. Dann ist $|Y| = 100$. Die Zuordnung $f\colon X \to \mathfrak{P}(Y)$ wählen wir so, dass gilt: Für alle Buchstaben $x \in X$ ist $|f(x)| \approx p(x) \cdot 100$.

In Tabelle 1 mit Häufigkeiten der Buchstaben sehen wir, dass z. B. c mit der Wahrscheinlichkeit 3,06 % auftritt. Also wählen wir zufällig für c drei Zeichen aus Y, z. B. 07, 23, 42.

2.2.1. Ansatz zur Kryptoanalyse einer homophonen Verschlüsselung

Eine weitere Idee für einen Angriff ist andere Unregelmäßigkeiten – neben der Häufigkeitsverteilung der Buchstaben – in der deutschen Sprache zu finden. Dazu betrachten wir sogenannte **Digramme** oder **Bigramme** (Zweierfolgen von Buchstaben) in Texten. Dafür gibt es insgesamt 26^2 mögliche Paarungen. Beispiele für Digramme sind *en* und *er*, die mit einer Wahrscheinlichkeit von ca. 4 % in deutschen Texten auftreten. Das Digramm *ch* tritt mit einer Wahrscheinlichkeit von ca. 2,75 % und die Digramme *de*, *te*, *nd*, *ei*, *ie*, *es*, *in* treten mit der Wahrscheinlichkeit von ca. 2 % auf.

Damit haben wir bereits eine Anomalie gefunden. Während die Buchstaben *e* und *n* häufig auftreten und auch das Digramm aus beiden *en* häufig auftritt, ist es im Fall von *c*, *h* und *ch* nicht so. Die einzelnen Buchstaben sind recht selten (siehe Tabelle 1), das Digramm jedoch tritt sehr häufig auf.

Ausgangspunkt: Verschlüsselung von ‚c'. Es sei $|f(c)| = k$ (z. B. $k = 3$ in Beispiel 2.3) Dann müssen (etwa) folgende Relationen erfüllt sein: $|f(e)| \approx 6k, |f(n)| \approx 3k, |f(i)| \approx 2{,}5k, |f(r)| \approx 2{,}5k$.

Das Bigramm „en" wird auf $6k \cdot 3k = 18k^2$ verschiedene Weisen verschlüsselt und „ch" auf $k \cdot 1{,}5k = 1{,}5k^2$ verschiedene Weisen.

Ein Digramm, dass „en" verschlüsselt, tritt mit einer Häufigkeit von $\frac{4\,\%}{18} \approx 0{,}25\,\%$ auf. Für „ch" ist die Häufigkeit $\frac{2{,}5\,\%}{1{,}5} \approx 1{,}6\,\%$.

Dies ist ein Ansatz zunächst durch eine **Häufigkeitsanalyse** die Verschlüsselung für *c* und *h* zu knacken. Voraussetzung für einen solchen Angriff ist aber ein wirklich langer Geheimtext!

	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
l	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	K
m	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
n	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
o	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
p	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
r	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
s	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
t	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
u	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
v	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
w	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
x	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
z	Z	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

Tabelle 2.2.: Das Vigenère-Quadrat.

2.3. Polyalphabetische Verschlüsselung

Polyalphabetische Verschlüsselung hat ebenfalls das Ziel, die Häufigkeit der Buchstaben zu verwischen. Ein prominentes Beispiel ist die **Vigenère-Chiffre** von dem französischen Diplomat BLAISE DE VIGENÈRE (1523–1596), die erst im 19. Jahrhundert gebrochen wurde.

Die Idee der Vigenère-Chiffre ist, dass jeder Buchstabe mit einer anderen Caesar-Chiffre verschlüsselt wird, aber nicht zufällig, sondern strukturiert mit Hilfe eines Schlüsselworts.

Beispiel 2.4
Der zu verschlüsselnde Klartext sei `emmentaler` und das Schlüsselwort „ALLGAEU“.

Klartext:	e	m	m	e	n	t	a	l	e	r				
Schlüsselwort:	A	L	L	G	A	E	U	A	L	L	G	A	E	U
Geheimtext:	E	X	X	K	N	X	U	L	P	C				

Die Verschlüsselung ist nicht wesentlich aufwendiger als die Caesar-Chiffre, aber es stellt sich eine neue Situation ein: „e“ wird zu „E“, „K“ und „P“ und umgekehrt entsteht „X“ aus „m“ und „t“. Dadurch werden die unterschiedlichen Häufigkeiten der Klartextbuchstaben im Geheimtext ausgeglichen.

Die **polyalphabetische Verschlüsselung** lässt sich allgemein auf folgenden Weise beschreiben: Es gibt ein Klartextalphabet Σ, d Geheimtextalphabete $\Gamma_0, \Gamma_1, \ldots, \Gamma_{d-1}$, d bijektive Abbildungen $f_i\colon \Sigma \to \Gamma_i$ $(i = 0, \ldots, d-1)$ und eine surjektive Abbildung $h\colon \mathbb{N} \to \{0,1,\ldots,d-1\}$. Den Schlüssel bilden dabei $h, f_0, f_1, \ldots, f_{d-1}$.

Ein Klartext $m = m_1 m_2 \ldots m_t$ mit $m_j \in \Sigma$ der Länge t wird in einen Geheimtext $c = c_1 c_2 \ldots c_t$ durch $c_j = f_{h(j)}(m_j)$ für $1 \leqslant j \leqslant t$ überführt.

Bei der **Vigenère-Chiffre** sind die Alphabete identisch, also $\Sigma = \Gamma_0 = \cdots = \Gamma_d = \{0,1,\ldots,25\}$. Das Schlüsselwort $k = k_0 k_1 \ldots k_d$ der Länge d beschreibt die einzelnen Verschlüsselungsfunktionen durch eine Verschiebechiffre $f_i(a) = a + k_i \pmod n$ (für $a \in X$). Als Funktion h wird einfach $h(x) = x \pmod d$ (für $x \in \mathbb{N}$) verwendet.

Beispiel 2.5
Das Alphabet sei $X = \{0,1,\ldots,25\}$ und das Schlüsselwort sei „KRYPTO“ $(d = 6)$, wobei dies als $10,17,24,15,19,14 = k_0, k_1, k_2, k_3, k_4, k_5$ dargestellt wird. Für den Klartext `kommemorgennicht` ergibt sich der Geheimtext `UFKBXAYIETGBSTFI`.

Schlüsselwort	K	R	Y	P	T	O
Kodierung k_j	10	17	24	15	19	14
Klartext	k	o	m	m	e	m
Kodierung x	10	14	12	12	04	12
$x + k_j \pmod{26}$	20	05	10	01	23	00
Geheimtext	U	F	K	B	X	A
Klartext	o	r	g	e	n	n
Kodierung x	14	17	06	04	13	13
$x + k_j \pmod{26}$	24	08	04	19	06	01
Geheimtext	Y	I	E	T	G	B
Klartext	i	c	h	t		
Kodierung x	08	02	07	19		
$x + k_j \pmod{26}$	18	19	05	08		
Geheimtext	S	T	F	I		

2.3.1. Analyse der polyalphabetischen Verschlüsselung

Der Angriffspunkt bei der polyalphabetischen Verschlüsselung ist die Periodizität. Bei einer Periodenlänge d bezeichnet man als einen Teiltext die Buchstabenfolge an den Positionen $k + nd$ für $1 \leqslant k \leqslant d$ und $n \in \mathbb{N}_0$. Wenn man den Text in Form einer Tabelle mit der Breite der Schlüsselwortlänge aufschreibt, dann ist jede Spalte von oben nach unten gelesen ein Teiltext.

Obwohl die Teiltexte keine sinnvollen Texte sind, stellt sich bei „langen“ Teiltexten die gleiche Häufigkeitsverteilung der Buchstaben wie bei einem richtigen Text ein. Innerhalb der Spalten findet man also eine Buchstabenhäufigkeit wie bei einem natürlichen Text.

Die einzelnen Teiltexte sind monoalphabetisch verschlüsselt. Man kann also im ersten Schritt die Schlüssellänge bestimmen und im zweiten Schritt die monoalphabetischen Teiltexte mit Hilfe der **Häufigkeitsanalyse** entschlüsseln.

Brute-force-Angriff auf polyalphabetische Verschlüsselung

Aus Erfahrung weiß man, dass die Häufigkeitsverteilung der Buchstaben im Klartext nivelliert ist, wenn ein Geheimtext, der durch polyalphabetische Verschlüsselung entstanden ist, mit einem Schlüssel der falschen Länge entschlüsselt wird. Man kann also systematisch $d = 1,2,3,\ldots$ durchprobieren bis man für den entschlüsselten Text eine natürliche Verteilung der Buchstaben gefunden hat.

Dies funktioniert nur für kleine d!

Kasiski-Test

CHARLES BABBAGE entdeckte 1854 die Möglichkeit, die Länge des Schlüsselworts für einen polyalphabetisch verschlüsselten Text zu bestimmen. Der Algorithmus wurde aber nach KASISKI benannt, der ihn das erste Mal veröffentlichte.

Wenn man die polyalphabetische Verschlüsselung betrachtet, fällt auf, dass Buchstabenfolgen, die sich im Abstand eines Vielfachen der Periodenlänge wiederholen, auf das gleiche Muster im Geheimtext abgebildet werden. Häufig auftretenden Buchstabenkombinationen werden also an mehreren Stellen im Text auf die gleiche Weise verschlüsselt.

Beispiele für solche Muster können die Artikel der, die, das, ein oder eine sein, die häufig in einem Text vorkommen und somit auch wiederholt in der gleichen Spalte bei der Verschlüsselung beginnen.

Findet man also in einem Geheimtext Muster, die länger als zwei Zeichen sind und mehrfach auftreten, so können dies Buchstabenfolgen sein, die auf die gleiche Weise verschlüsselt wurden. Die Länge des Schlüsselworts muss also ein Teiler des Abstands der Wiederholungen des Musters sein.

Man kann mit mehreren Mustern und anhand ihrer Struktur (längere Muster sind eher das Ergebnis eines Musters im Klartext als eine zufällige Wiederholung im Geheimtext) eine (oder ein paar) Vermutungen über die Schlüssellänge finden.

Friedman-Test

Von WILLIAM FRIEDMAN (1891–1969) stammt die Idee, die Periodenlänge mit Hilfe statistischer Methoden zu bestimmen. Die entscheidende statistische Größe ist der Koinzidenzindex $I(t)$ für einen Text t.

Für ein Alphabet $A = \{a_1, \ldots, a_n\}$ mit $|A| = n$ und einen Text t der Länge l ist der **Koinzidenzindex** $I(t)$ die Wahrscheinlichkeit dafür, dass an zwei zufällig gewählten Positionen in dem Text der selbe Buchstabe steht.

Es sei l_i die absolute Häufigkeit (Anzahl der Vorkommen) des Buchstabens a_i im Text, d. h. $l = \sum_{i=1}^{n} l_i$. Für einen Text der Länge l gibt es $\binom{l}{2}$ zufällig gewählte Zweiermengen von Positionen. Die Anzahl der Mengen von zwei zufällig gewählten Positionen mit gleichen Buchstaben beträgt

$$\sum_{i=1}^{n} \binom{l_i}{2} = \sum_{i=1}^{n} \frac{l_i(l_i - 1)}{2}$$

Damit ergibt sich die Wahrscheinlichkeit, dass an zwei zufällig gewählten Positionen zwei gleiche Buchstaben im Text t stehen:

$$\begin{aligned} I(t) &= \sum_{i=1}^{n} \frac{\binom{l_i}{2}}{\binom{l}{2}} = \sum_{i=1}^{n} \frac{l_i(l_i-1)}{l(l-1)} = \sum_{i=1}^{n} \frac{l_i^2}{l(l-1)} - \sum_{i=1}^{n} \frac{l_i}{l(l-1)} \\ &= \frac{l}{l-1} \sum_{i=1}^{n} \left(\frac{l_i}{l}\right)^2 - \frac{1}{l-1} \end{aligned} \tag{2.1}$$

Für einen hinreichend großen Text, d. h. $l \to \infty$, in deutscher Sprache t_d mit den Buchstabenhäufigkeiten aus Tabelle 1 ergibt sich ein **Koinzidenzindex** $I_d := I(t_d) \approx 0{,}0762$. In der englischen Sprache liegt der Index bei etwa 0,0667 und im französischen bei 0,0778.

Der Koinzidenzindex für einen hinreichend großen Zufallstext t_z der Länge l, bei dem alle Buchstaben gleich häufig auftreten $l_1 = l_2 = \cdots = l_n$, ist

$$I(t_z) = \frac{l}{l-1} \sum_{i=1}^{n} \left(\frac{1}{n}\right)^2 - \frac{1}{l-1} = \frac{l}{l-1}\frac{1}{n} - \frac{1}{l-1} \xrightarrow{l\to\infty} \frac{1}{n}$$

Legt man ein Alphabet mit 26 Buchstaben zugrunde, ergibt sich so ein **Koinzidenzindex** $I_z := \frac{1}{26} \approx 0{,}0385$.

Bemerkung 2.1
Der entscheidende Fakt für den Friedman-Test, der einen Angriff auf die Verschlüsselung ermöglicht, ist, dass die Wahrscheinlichkeit an zwei zufällig gewählten Positionen denselben Buchstaben zu finden, in deutschen Texten (und Texten anderer natürlicher Sprachen) etwa doppelt so groß wie in Zufallstexten ist.

In einem Geheimtext c der Länge l, der das Ergebnis einer polyalphabetischen Verschlüsselung mit Periode d ist, sind die Buchstaben in einem Teiltext $(k, d+k, 2d+k, \ldots)$ entsprechend der Buchstabenhäufigkeit der Sprache verteilt. Zwischen den Teiltexten (d. h. von einer Spalte zu einer anderen) sind die Buchstaben jedoch zufällig verteilt. (Eigenschaft der polyalphabetischen Verschlüsselung.)

Die Teiltexte haben die Länge $\frac{l}{d}$. Also gibt es

$$d \cdot \binom{\frac{l}{d}}{2} = \frac{l(l-d)}{2d}$$

zufällige Paare in ein und demselben Teiltext über alle Teiltexte. Für die zufälligen Paare in unterschiedlichen Teiltexten bleiben

$$\frac{1}{2} \cdot l \cdot (l - \frac{l}{d}) = \frac{l^2(d-1)}{2d}$$

Möglichkeiten. (l Möglichkeiten für die erste Position; $l - \frac{l}{d}$ Möglichkeiten für die zweite Position in den anderen Spalten; $\frac{1}{2}$ da es kein geordnetes Ziehen ist)

Damit ist die Wahrscheinlichkeit im gesamten Geheimtext c an zwei zufällig gewählten Positionen die gleichen Buchstaben zu finden

$$I(c) = \frac{\frac{l(l-d)}{2d} \cdot I_z + \frac{l^2(d-1)}{2d} \cdot I_d}{\frac{l(l-1)}{2}} \tag{2.2}$$

Da man den Koinzidenzindex $I(c)$ für den verschlüsselten Text mit der Gleichung 2.1 bestimmen kann, kann man so einen Schätzwert für die Periodenlänge d der Verschlüsselung aus Gleichung 2.2 ableiten.

$$d = \frac{l(I_d - I_z)}{I_d - lI_z + (l-1)I(c)} \tag{2.3}$$

Diese Schätzung kann man mit den Ergebnissen aus einem Kasiski-Test oder einem Brute-Force-Angriff vergleichen, um die Möglichkeiten für die wahre Periodenlänge genauer einzuschränken.

2.3.2. Möglichkeiten der Verteidigung gegen die Angriffe

Ausgangspunkt für eine Verteidigung ist der Umstand, dass die oben formulierten Tests nur für kleine Periodenlängen ($d \ll l$) funktionieren!

- Eine Abwehrstrategie wäre also die Periodenlänge zu vergrößern. Praktisch wurde dies bei der **Enigma** umgesetzt. Im deutschen Patent wurde 1920 mit drei Rotoren gearbeitet, die eine Periodenlänge von 26^3 ergaben. Im späteren Einsatz im Krieg wurde eine Maschine mit fünf Rotoren, die eine Periodenlänge von 26^5 hatte, gearbeitet.

- Ein anderer Weg wäre, gar keine Wiederholung zu verwenden, sprich das Schlüsselwort so lang wie den Text wählen. Dies kann eine Textstelle aus einem vorher gewählten Buch sein – Schlüssel ist dann (Seite, Zeile, Buchstabe). Aber dann funktioniert wieder die Häufigkeitsanalyse, weil der Roman die Charakteristik einer natürlichen Sprache hat. (In diese Kategorie fällt auch das anhängen des Klartextes an das Schlüsselwort.)

  Also müsste man für den „Schlüsselroman“ einen Zufallstext wählen, was zu dem absolut sicherem, aber unpraktikablen **One-Time-Pad** führt.

3. Transpositionsverfahren

Bei einer **Transpositionschiffre** werden die Buchstaben des Klartexts auf gezielte Art und Weise untereinander vertauscht, so dass der Text verschlüsselt wird. Das Klartextalphabet ist also gleich dem Geheimtextalphabet ($\Sigma = \Gamma$) und insbesondere bleibt die Häufigkeitsverteilung der Buchstaben gleich, so dass ein Angriff über eine **Häufigkeitsanalyse** wirkungslos bleibt.

Es gibt zwei verschiedene Ansätze der Verschlüsselung:

- Bei der **Blocktransposition** wird der Klartext m in Blöcke einer festen Länge k eingeteilt und jeder Block m_i wird derselben Permutation π unterworfen, wodurch sich der Geheimtextblock c_i ergibt.

  $$c_{i_{\pi(1)}} c_{i_{\pi(2)}} \dots c_{i_{\pi(k-1)}} c_{i_{\pi(k)}} = m_{i_1} m_{i_2} \dots m_{i_{k-1}} m_{i_k}$$

  Als Angriff auf einen durch Blocktransposition verschlüsselten Text erweist sich der folgende, etwas plumpe Ansatz als sehr gut: Rate die Blocklänge (ein Teiler der Geheimtextlänge) und errate innerhalb von Blöcken eine Permutation, so dass sich bekannte Anagramme ergeben.

- Die **Spaltentransposition** ist eine erweiterte Blocktransposition. Der Klartext wird wieder in Blöcke einer festen Länge k zerlegt und diese werden jeder für sich einer Permutation π unterworfen. Danach wird der Geheimtext gebildet, indem man erst alle ersten Buchstaben aller Blöcke, dann die zweiten usw. aufschreibt.

  Anschaulich kann man sich den Ablauf so vorstellen, dass der Text zeilenweise in eine Tabelle fester Breite eingetragen. Dann werden die kompletten Spalten vertauscht und der Text wird anschließend spaltenweise ausgelesen.

  Verwendet man als Permutation der Spalten die Identität, so erhält man die **Gartenzaunchiffre**: Die Zeichen werden zeilenweise eingetragen und spaltenweise wieder ausgelesen.

  Als Angriffsmöglichkeit ist wieder das Erraten der Blocklänge und anschließendes Suchen nach Anagrammen in jedem Block eine gute Variante.

Beispiel 3.1
Der Klartext lautet `kryptologie und datensicherheit`. Die Blockgröße k soll 6 sein und die Permutation $\pi = (1, 5, 3, 6, 4, 2)$. Wenn man den Text jetzt in ein Gitter einträgt, bleibt die letzte Zelle links unten leer. Diese füllt man einfach mit einem vorher vereinbarten Zeichen, bei uns `a`. siehe Abbildung 3.1. Wenn man dann den Text spaltenweise von oben nach unten ausliest, ergibt sich der Geheimtext `rodsh piaci tetht oueea klnnr ygdie`.

1.	2.	3.	4.	5.	6.		1.	2.	3.	4.	5.	6.
							$\pi(2)$	$\pi(4)$	$\pi(5)$	$\pi(6)$	$\pi(1)$	$\pi(3)$
k	r	y	p	t	o		r	p	t	o	k	y
l	o	g	i	e	u	$\rightarrow$	o	i	e	u	l	g
n	d	d	a	t	e		d	a	t	e	n	d
n	s	i	c	h	e		s	c	h	e	n	i
r	h	e	i	t	a		h	i	t	a	r	e

Abbildung 3.1.: Veranschaulichung der Spaltentransposition

Beispiel 3.2

IAEE␣KCWE␣NUCE␣TOEE␣KRAM␣DEZN␣EDNZ␣AIFA␣TRWN␣IRRS␣UBSN␣WKRE␣ELNT␣EURE
RKNS␣IKZI␣MIKF␣VNUH␣ANENK

Die Länge des Textes sind 77 Zeichen. Damit ergeben sich als mögliche Spaltenlängen 7 oder 11.

```
IEZRWEK
ATNWKRF
EOENRKV
EEDIENN
KENRESU
CKZRLIH
WRASNKA
EAIUTZN
NMFBEIE
UDASUMN
CETNRIK
```

```
IEEEANSLRIU
ANEZIINNKMH
EUKNFRWTNIA
ECREARKESKN
KEADTSRUIFE
CTMNRUERKVN
WODZWBEEZNK
```

Die erste Zeile für $k = 11$ enthält zu viele Konsonanten und die letzte Zeile zu viele Vokale für einen deutschen Text.

Idee: 1. und letzte Spalte liegen nebeneinander, da auf ein *c* sehr oft ein *k* oder ein *h* folgt.

Die sechste Spalte kommt vor die erste und dann können wir das Wort in der ersten Zeile erraten „zwei“: 3., 5., 1., 6., 2., 4. Insbesondere die zweite Zeile ist aufschlussreich. Als Wort springt hier „Kraft“ ins Auge. Wenn die Spalte entsprechend angeordnet werden, ergibt sich schnell der Klartext: „zwei kernkraftwerke von denen eines kuerzlich krank war sitzen auf einem baum und stricken“

4. Blockchiffren

4.1. Allgemeines

Wir bleiben zunächst noch bei den symmetrischen Verschlüsselungsverfahren und versuchen durch eine Vergrößerung des Schlüsselraums die Verfahren sicherer zu machen. Im Folgenden beschränken wir uns auf (rechnerfreundliche) Bitfolgen, d. h. wir arbeiten mit Teilmengen von $\mathbb{Z}_2^*$.

Definition 4.1 (Blockchiffre)
Bei einer **Blockchiffre** wird der Klartext in Blöcke einer frei wählbaren, aber festen Länge n (z. B. 32, 64 oder 128 Bit) zerlegt. Diese wird als **Blocklänge** bezeichnet. Eine Verschlüsselungsfunktion $f\colon \mathcal{M}\times\mathcal{K} \to \mathcal{C}$ erzeugt unter Verwendung eines Schlüssels $k \in \mathcal{K}$ Geheimtextblöcke $c = f(m,k)$ der *gleichen* Länge.

Damit eine Entschlüsselung des Textes möglich ist, muss die Verschlüsselung bijektiv sein und da sie von $\mathbb{Z}_2^n$ auf $\mathbb{Z}_2^n$ abbildet, ist sie eine Permutation der 2^n möglichen Blöcke. Es gibt also $2^n!$ mögliche Funktionen zur Verschlüsselung.

Ein kleines Gedankenspiel: Für die Blocklänge $n = 64$ gibt es $2^{64}!$ mögliche Verschlüsselungen und damit genau soviele Schlüssel. Um einen solchen Schlüssel abspeichern zu können, benötigt man $l = \log_2(2^{64}!)$ Bit. Mit Hilfe der Stirling-Formel kann man die Größe eines solchen Schlüssels abschätzen:

$$2^n! \approx \sqrt{2\pi \cdot 2^n} \cdot \left(\frac{2^n}{e}\right)^{2^n}$$

$$l \approx \log_2\left(\sqrt{2\pi} \cdot 2^{32} \cdot \left(\frac{2^{64}}{e}\right)^{2^{64}}\right) \leqslant 64 \cdot 2^{64} = 2^{6+64}$$

Es würden also zum Abspeichern *eines* Schlüssels für eine Nachricht $\frac{1}{8} \cdot 2^{20} \approx 10^5$ Pebibyte benötigt. In der Praxis kann man also nur mit „kleinen“ Teilmengen des Schlüsselraums arbeiten.

Für affine Chiffren werden beispielsweise $\{0,1\}$-Matrizen vom Typ 64×64 verwendet, deren Determinante 1 ist. Die Darstellung des Schlüssels ist also mit weniger als $64 \cdot 64 = 2^{12}$ Bit (im Vergleich zu 2^{70} Bit) möglich.

Können diese Eigenschaften nicht schon eher kommen? Im 1. Kapitel

Die Hintereinanderschaltung von *verschiedenen* Blockchiffren führt zu einer Erhöhung der Sicherheit. Oft realisiert man eine abwechselnde Folge von Substitutionen und Transpositionen mit dem Ziel die folgenden beiden **Eigenschaften der Verschlüsselung**, die auf CLAUDE SHANNON, den Begründer der Informationstheorie, zurückgehen, zu erreichen:

- **Diffusion**: Eine Änderung eines Klartextbits bewirkt die Änderung von vielen Bits des Geheimtexts.

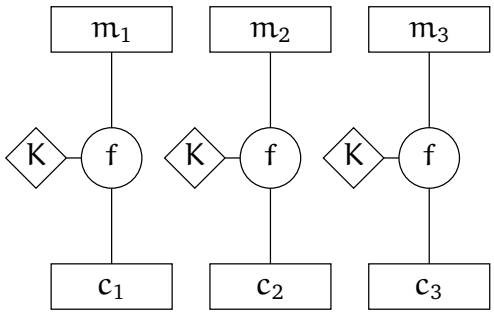

Abbildung 4.1.: Electronic Code Book

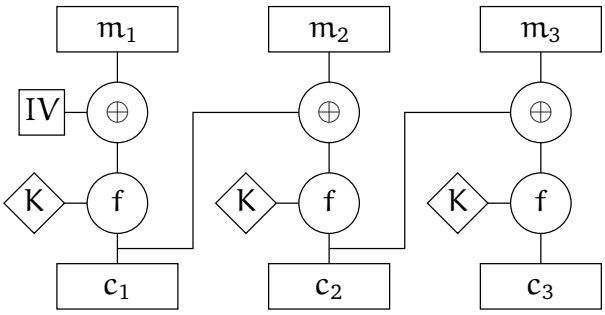

Abbildung 4.2.: Cipher Block Chaining

- **Konfusion**: Jedes Bit des Chiffretextes hängt von mehr als einem Bit des Schlüssels ab.

4.2. Blockverknüpfungsmodi

Bei der Verschlüsselung können gleiche Blöcke gleich verschlüsselt werden – in Analogie zur monoalphabetischen Verschlüsselung – oder sie werden in Abhängigkeit von ihrer Position im Klartext auf verschiedene Weise verschlüsselt – in Analogie zur polyalphabetischen Verschlüsselung.

Die erste Variante bezeichnet man als **Electronic Code Book Mode** (**ECB**). Jeder Block wird unabhängig von seiner Position mit Hilfe des Schlüssels durch die Funktion f in einen Geheimtextblock überführt. (siehe Abbildung 4.1)

$$c_i = E_K(m_i) \qquad\qquad m_i = D_K(c_i)$$

Dies hat den Vorteil, dass die Verschlüsselung aller Blöcke parallel ablaufen kann. Jedoch bleiben die Anomalien des Klartextes (häufigeres Auftreten bestimmter Blöcke) erhalten. Da sich Änderungen an einem Klartext- oder Chiffretextblock nur auf genau einen Klartext bzw. Chiffretextblock auswirken, könnte ein Angreifer gezielt Daten verändern, ohne dass es bemerkt wird. Jedoch bietet dieser Umstand auch die Möglichkeit einzelne Blöcke gezielt zu entschlüsseln oder zu verändern, ohne die ganze Nachricht entschlüsseln zu müssen. Für eine Dateiverschlüsselung z. B. könnte gezielt der neue Klartextblock verschlüsselt und in dem Geheimtext ersetzt werden.

Beim **Cipher Block Chaining Mode** (**CBC**) wird der Klartextblock vor der Verschlüsselung mit dem Ergebnis der Verschlüsselung des vorherigen Blocks verknüpft, um für gleiche Klartextblöcke unterschiedliche Geheimtextblöcke zu erzeugen. Dies bringt jedoch die Nachteile mit sich, dass die Verschlüsselung der Blöcke nicht mehr parallel ablaufen kann und dass bei einer Änderung eines Klartextblocks alle folgenden Geheimtextblöcke neu berechnet werden müssen. (siehe Abbildung 4.2)

$$c_i = E_K(m_i \oplus c_{i-1}) \qquad\qquad m_i = D_K(c_i) \oplus c_{i-1}$$

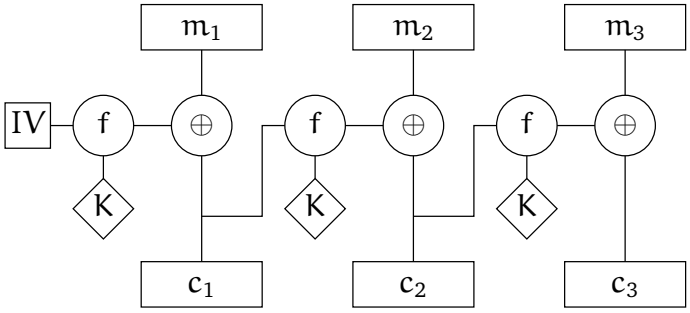

Abbildung 4.3.: Cipher Feedback

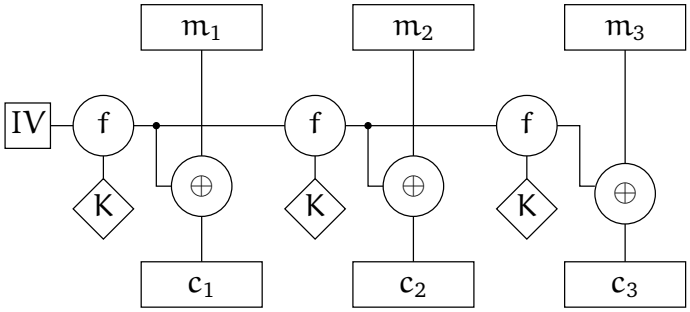

Abbildung 4.4.: Output Feedback

Eine Veränderung des r-ten Geheimtextblocks ändert nur den r-ten Block und den folgenden Klartextblock:

$$D_K(\widetilde{c_r}) \oplus c_{r-1} \neq m_r$$
$$D_K(c_{r+1}) \oplus \widetilde{c_r} \neq m_{r+1}$$
$$D_K(c_{r+2}) \oplus c_{r+1} = m_{r+2}$$

Cipher Feedback Mode (CFB) ist ähnlich zu CBC. Der vorherige Geheimtextblock wird verschlüsselt und dieses Ergebnis mit dem Klartextblock verknüpft, um Anomalien im Klartext auszugleichen. (siehe Abbildung 4.3)

$$c_i = E_K(c_{i-1}) \oplus m_i \qquad\qquad m_i = E_K(c_{i-1}) \oplus c_i$$

Die Verschlüsselungsfunktion f muss nicht umkehrbar sein, da man zur Entschlüsselung ebenfalls die Verschlüsselungsfunktion verwendet.

Für den **Output Feedback Mode (OFB)** wird aus einem Initialwert eine Folge von verschlüsselten Blöcken generiert, die mit den Klartextblöcken verknüpft, die Geheimtextblöcke ergeben. (siehe Abbildung 4.4)

$$c_i = m_i \oplus s_i \qquad\qquad m_i = c_i \oplus s_i \qquad\qquad s_i = E_K(s_{i-1})$$

Dies bringt zwei Vorteile mit sich: Zum Einen kann eine solche Folge schon vorab generiert werden, wodurch die eigentliche Verschlüsselung, bei Vorliegen des Klartextes,

mode	Schreibzugriff	Manipulation zerstört ...	Anomalien im Text
ECB	wahlfrei	diesen Block	unverändert
CBC	linear	aktuellen u. nächsten Block	verwischt
CFB	linear	aktuellen u. nächsten Block	verwischt
OFB	wahlfrei	diesen Block	verwischt

Tabelle 4.1.: Übersicht der Eigenschaften von ECB, CBC, CFB und OFB

sehr schnell geht. (Vorteilhaft z. B. bei Streaming) Zum Anderen kann ein Klartextblock direkt ausgetauscht werden, ohne dass man die folgenden Blöcke oder die verschlüsselte Folge neu berechnen muss: $\widetilde{c_i} = c_i \oplus m_i \oplus \widetilde{m_i}$.

4.3. Crashkurs über Restklassenringe und Matrizen darüber

Zwei Zahlen $a, b \in \mathbb{Z}$ sind zueinander **kongruent modulo** k, wenn sie bei der Division durch k den gleichen Rest ($0 \leqslant r < k$) lassen:

$$\exists r \in \mathbb{Z}\colon a = b + r \cdot k$$

Wir schreiben dafür $a \equiv_k b$ oder $a \equiv b \pmod{k}$.

Definition 4.2
Für $a \in \mathbb{Z}$ und $k \in \mathbb{N}$ heißt

$$[a]_k := a + k\mathbb{Z} = \{\, a + kz \mid z \in \mathbb{Z} \,\}$$

die **Restklasse** von a modulo k.

Die Menge dieser Restklassen $\{\, a + k\mathbb{Z} \mid a \in \mathbb{Z} \,\}$ bezeichnet mal als **Faktormenge** und schreibt dafür $\mathbb{Z}_k = \mathbb{Z}/k\mathbb{Z}$.

Auf dieses Restklassensystem kann man die Addition und Multiplikation auf folgende Weise übertragen:

$$[a]_k + [b]_k := [a + b]_k \qquad \text{Nullelement: } [0]_k$$
$$[a]_k \cdot [b]_k := [a \cdot b]_k \qquad \text{Einselement: } [1]_k$$

Für jedes $[a]_k$ existiert ein **inverses Element bzgl. Addition** $-[a]_k := [-a]_k = [k - a]_k$. Ein Element $[a]_k$ heißt genau dann **Einheit** des Restklassensystems, wenn ein $[b]_k$ existiert, so dass $[a]_k \cdot [b]_k = [1]_k$. In diesem Fall heißt $[b]_k$ **inverses Element bzgl. Multiplikation** zu $[a]_k$.

Lemma 4.1
Die Eulersche φ-Funktion $\varphi(k)$ ist die Anzahl der der positiven ganzen Zahlen $a \leqslant k$ an, die zu dieser teilerfremd sind:

$$\varphi(k) := |\{\, 1 \leqslant a \leqslant k \mid ggT(a, k) = 1 \,\}|$$

In $\mathbb{Z}_k$ gibt es genau $\varphi(k)$ Einheiten.

BEWEIS:
Sei $U(\mathbb{Z}_k) = \{ a \in \mathbb{Z}_k \mid a \text{ invertierbar} \}$ die Einheitsgruppe von $\mathbb{Z}_k$. Die Umformung ergibt:

$$\begin{aligned} U(\mathbb{Z}_k) &= \{ a \in \mathbb{Z}_k \mid a \text{ invertierbar} \} \\ &= \{ a + k\mathbb{Z} \mid \exists\, b \in \mathbb{Z}\colon (a + k\mathbb{Z})(b + k\mathbb{Z}) = 1 + k\mathbb{Z} \} \\ &= \{ a + k\mathbb{Z} \mid \exists\, b \in \mathbb{Z}\colon a \cdot b \equiv 1 \pmod{k} \} \\ &= \{ a + k\mathbb{Z} \mid \mathrm{ggT}(a,k) = 1 \} \end{aligned}$$

Den letzten Schritt in der Umformung kann man entweder einfach glauben oder u. a. im Skript zu Algebra 1 von Prof. Külshammer[1] nachlesen.

Damit ist klar, dass $|U(\mathbb{Z}_k)| = |\{ a + k\mathbb{Z} \mid \mathrm{ggT}(a,k) = 1 \}| = \varphi(k)$. ∎

Für das Produkt zweier Zahlen p und q, die zueinander teilerfremd sind, ist $\varphi(p \cdot q) = \varphi(p) \cdot \varphi(q)$, d. h. φ ist multiplikativ. Falls p eine Primzahl ist, dann ist $\varphi(p) = p - 1$. Es gibt also $\varphi(p) = p - 1$ Einheiten in $\mathbb{Z}_p$, womit $\mathbb{Z}_p$ ein Körper ist.

Die algebraische Struktur $\mathbb{Z}_k = [\mathbb{Z}_k, +, \cdot]$ mit dem Nullelement $[0]_k$ und dem Einselement $[1]_k$ bezeichnet mal als **kommutativen Ring**.

In der Kryptologie wird häufig mit dem Restklassenring modulo 26 gearbeitet: $\mathbb{Z}_{26} = \{0,1,\ldots,25\}$. Im Grunde sind alle endlichen Alphabete kommutative Ringe, was den Vorteil hat, dass man mit Buchstaben rechnen kann. Dies haben wir bereits bei der Caesar-Chiffre ausgenutzt.

Der Ring $\mathbb{Z}_{26}$ hat genau 12 Einheiten, da $\varphi(|\mathbb{Z}_{26}|) = \varphi(2 \cdot 13) = \varphi(2) \cdot \varphi(13) = 1 \cdot 12$. Diese sind: 1,3,5,7,9,11,15,17,19,21,23,25.

Wir betrachten $n \times n$-Matrizen, deren Einträge Elemente aus einem beliebigen kommutativen Ring $[R, +\cdot]$ sind.

$$R^{n\times n} := \left\{ \begin{pmatrix} r_{11} & r_{12} & \ldots & r_{1n} \\ r_{21} & r_{22} & \ldots & r_{2n} \\ \vdots & \vdots & & \vdots \\ r_{n1} & r_{n2} & \ldots & r_{nn} \end{pmatrix} \;\middle|\; r_{ij} \in R, 1 \leqslant i,j \leqslant n \right\}$$

Die Matrizenaddition und -multiplikation ist dabei wie üblich.

In der Kryptologie betrachten wir speziell die Matrizen $\mathbb{Z}_k^{n\times n}$.

Als **Zeilen-** (Gleichung 4.1) bzw. **Spaltenvektoren** (Gleichung 4.2) bezeichnet man Matrizen der folgenden Form:

$$R^{1\times n} = \left\{ \begin{pmatrix} b_1 & b_2 & \ldots & b_n \end{pmatrix} \;\middle|\; b_i \in R, 1 \leqslant i \leqslant n \right\} \tag{4.1}$$

$$R^{n\times 1} = \left\{ \begin{pmatrix} b_1 & b_2 & \ldots & b_n \end{pmatrix}^T \;\middle|\; b_i \in R, 1 \leqslant i \leqslant n \right\} \tag{4.2}$$

[1] http://uni-skripte.lug-jena.de/skripte.html?id=kuelshammer-algebra1Skript zu Algebra 1

Als eine **Permutation** σ bezeichnet man eine Umordnung der Elemente eine Menge. Mit S_n bezeichnet man die Menge der Permutationen einer Menge mit n Elementen (**symmetrische Gruppe**).

$$S_n = \{\, \sigma \mid \sigma\colon \{1,\dots,n\} \to \{1,\dots,n\} \,\}$$

Für eine Permutation $\sigma \in S_n$ bezeichnet man ein Paar $(i,j) \in \{1,\dots,n\}^2$ als **Fehlstand**, wenn $i < j$ und $\sigma(i) > \sigma(j)$ gilt.

Die **Determinante** einer quadratischen Matrix $A \in R^{n\times n}$ mit $A = (a_{ij})_{n\times n}$ und $a_{ij} \in R$ kann man auf zwei Weisen berechnen.

Die Formel von LEIBNIZ für die Determinante einer Matrix lautet:

$$\det A := \sum_{\sigma\in S_n} \operatorname{sgn}(\sigma) \cdot \prod_{i=1}^{n} a_{i\sigma(i)}$$

$$\operatorname{sgn}(\sigma) = \begin{cases} +1 & : \text{die Anzahl der Fehlstände ist gerade} \\ -1 & : \text{die Anzahl der Fehlstände ist ungerade} \end{cases}$$

Beispiel 4.1
Die Determinante einer 1×1-Matrix $A = (a_{11})$ ist $\det A = a_{11}$, da die einzig mögliche Permutation $\sigma = \left(\begin{smallmatrix} 1 \\ 1 \end{smallmatrix}\right)$ das Vorzeichen $+1$ hat.

$$\det A = \sum_{\sigma\in S_1} \operatorname{sgn}(\sigma) \cdot \prod_{i=1}^{1} a_{i\sigma(i)} = \sum_{\sigma\in S_1} \operatorname{sgn}(\sigma) a_{11}$$

Für eine 2×2-Matrix A gibt es zwei Permutationen $\sigma_1, \sigma_2 \in S_2$:

$$\sigma_1 = \begin{pmatrix} 1 & 2 \\ 1 & 2 \end{pmatrix},\ \operatorname{sgn}(\sigma_1) = +1 \qquad \sigma_2 = \begin{pmatrix} 1 & 2 \\ 2 & 1 \end{pmatrix},\ \operatorname{sgn}(\sigma_2) = -1$$

Entsprechend ergibt sich die Determinante als

$$\begin{aligned} \det A &= \sum_{\sigma\in S_2} \operatorname{sgn}(\sigma) \cdot \prod_{i=1}^{2} a_{i\sigma(i)} = \sum_{\sigma\in\{\sigma_1,\sigma_2\}} \operatorname{sgn}(\sigma) \cdot a_{1\sigma(1)} \cdot a_{2\sigma(2)} \\ &= \operatorname{sgn}(\sigma_1) \cdot a_{1\sigma(1)} \cdot a_{2\sigma(2)} + \operatorname{sgn}(\sigma_2) \cdot a_{1\sigma(1)} \cdot a_{2\sigma(2)} = a_{11}a_{22} - a_{12}a_{21} \end{aligned}$$

Die andere Möglichkeit ist das Verfahren von LAPLACE, bei dem nach einer Zeile oder Spalte entwickelt wird:

$$\det A = \sum_{j=1}^{n} (-1)^{i+j} a_{ij} A_{ij}$$

wobei A_{ij} diejenige Matrix ist, die durch Streichen der i-ten Zeile und j-ten Spalte aus A entsteht.

Eine Matrix A besitzt eine zu ihr **inverse Matrix** A^{-1}, d. h. es gilt

$$A \cdot A^{-1} = A^{-1} \cdot A = E_n = \begin{pmatrix} 1 & & 0 \\ & \ddots & \\ 0 & & 1 \end{pmatrix}_{n \times n}$$

genau dann, wenn die Determinante $\det A$ ein inverses Element in $[R, +, \cdot]$ besitzt, also genau dann, wenn $\det A$ eine Einheit des Rings $[R, +, \cdot]$ ist.

Dabei gilt: $A^{-1} = \frac{B}{\det A}$, wobei $(b_{ij}) = (-1)^{i+j} \cdot \det A_{ji}$.

4.4. Affine Blockchiffren

Die Zeichen des Klartexts werden durch Elemente aus $\mathbb{Z}_k$ codiert und der Klartext wird in Blöcke der Länge n eingeteilt, die einzeln nach demselben Schema verschlüsselt werden.

Der Schlüssel ist ein Paar (A, b), wobei $A \in \mathbb{Z}_k^{n \times n}$ und $b \in \mathbb{Z}_k^{1 \times n}$ ist. Zur Verschlüsselung eines Blockes $m = (m_1 \quad \dots \quad m_n) \in \mathbb{Z}_k^{1 \times n}$ wird die lineare Abbildung

$$c = m \cdot A + b$$

verwendet. Diese Verschlüsselung ist weder eine Substitution (ein Eintrag c_i hängt im Allgemeinen von allen $m_1, \dots, m_n$ ab) noch keine Transposition.

Aber nicht alle Verschlüsselungen dieser Art sind auch eindeutig, d. h. eine Entschlüsselung ist möglich. Dafür muss zu der Matrix A die inverse Matrix A^{-1} existieren, was genau dann der Fall ist, wenn $\mathrm{ggT}(\det A, k) = 1$, also die Determinante von A eine Einheit von $\mathbb{Z}_k$ ist.

Die Entschlüsselung erfolgt dann mit der Abbildung

$$m = (c - b)A^{-1}$$

Der Spezialfall $b = 0$, also $c = m \cdot A$, ist die **Hill-Chiffre** (1929) und der Spezialfall $A = E_n$, also $c = m + b$, ist die **Vigenère-Chiffre** mit dem Schlüsselwort b.

Man kann auch eine **Blocktransposition** als eine affine Abbildung beschreiben. Sei σ die für die Blocktransposition genutzte Permutation von $\{1, \dots, n\}$. Mit $b = 0$ und der Matrix $P_\sigma = (p_{ij})_{n \times n}$ mit

$$p_{ij} = \begin{cases} 0 & \sigma(i) \neq j \\ 1 & \sigma(i) = j \end{cases}$$

ergibt sich der erwünschte Effekt, dass

$$\begin{pmatrix} m_1 & m_2 & \dots & m_n \end{pmatrix} \cdot P_\sigma = \begin{pmatrix} m_{\sigma(1)} & m_{\sigma(2)} & \dots & m_{\sigma(n)} \end{pmatrix}$$

und es gilt $P_\sigma^{-1} = P_{\sigma^{-1}}$.

4.4.1. Kryptoanalyse affiner Blockchiffren

Sind $n+1$ Klartextblöcke $m_0, \ldots, m_n$ und die dazugehörigen Geheimtextblöcke $c_0, \ldots, c_n$ bekannt – ein **Known-Plaintext-Angriff** –, kann man folgende zwei Matrizen definieren:

$$M = \begin{pmatrix} m_1 - m_0 \\ \vdots \\ m_n - m_0 \end{pmatrix}_{n \times n} \qquad C = \begin{pmatrix} c_1 - c_0 \\ \vdots \\ c_n - c_0 \end{pmatrix}_{n \times n}$$

Es gilt also

$$C = \begin{pmatrix} c_1 - c_0 \\ \vdots \\ c_n - c_0 \end{pmatrix} = \begin{pmatrix} (m_1 A + b) - (m_0 A + b) \\ \vdots \\ (m_n A + b) - (m_0 A + b) \end{pmatrix} = \begin{pmatrix} (m_1 - m_0)A \\ \vdots \\ (m_n - m_0)A \end{pmatrix} = M \cdot A$$

Wenn $\det M$ eine Einheit in $\mathbb{Z}_k$ ist, dann existiert auch M^{-1}. Hieraus folgt dann $A = M^{-1}C$ und weiter $b = c_0 - m_0 A$.

Bemerkung 4.1
Dies kommt nicht zu selten vor, dass $\det M$ eine Einheit ist, denn es gilt:

$$\limsup_{k \to \infty} \frac{\varphi(k)}{k} = 1 \qquad \text{und} \qquad \lim_{k \to \infty} \frac{\varphi(k)}{k^{1-\delta}} = \infty$$

für $\delta > 0$.

Für $\mathbb{Z}_{26}$ gibt es z. B. $\varphi(26) = 12$ Einheiten.

Bemerkung 4.2
In einer halbgeordneten[2] Menge kann man definieren:

$$\limsup_{k \to \infty} x_k = \inf\{\, \sup\{x_l \mid l \geqslant k\} \colon k \in \mathbb{N}\,\}$$

In vollständigen Verbänden existiert stets der Limes superior und Limes inferior und es gilt $\liminf_{k \to \infty} x_k \leqslant \limsup_{k \to \infty} x_k$.

Für **Hill-Chiffren** (d. h. $b = 0$) genügen n Paare (m_j, c_j), da die Matrix C nicht als $(c_i - c_0)$ definiert werden muss, um den Anteil von b zu eliminieren. Damit können auch für die Matrix M direkt die Klartextblöcke verwendet werden.

Beispiel 4.2
Es sei $k = 26$ und $n = 2$. Der Klartext sei „HERBST“ und der Geheimtext sei „SOMMER“. Die Aufgabe ist den Schlüssel (A, b) zu bestimmen.

Der Klartext „HERBST“ wird durch $\underbrace{7\,4}_{m_0}\ \underbrace{17\,1}_{m_1}\ \underbrace{18\,19}_{m_2}$ und der Geheimtext „SOMMER“ durch $\underbrace{18\,14}_{c_0}\ \underbrace{12\,12}_{c_1}\ \underbrace{4\,17}_{c_2}$ codiert.

[2] Eine Halbordnung ist eine transitive, reflexive und antisymmetrische Ordnungsrelation.

Damit kann man nun die Matrizen M und C bestimmen:

$$M = \begin{pmatrix} m_1 - m_0 \\ m_2 - m_0 \end{pmatrix} = \begin{pmatrix} 17-7 & 1-4 \\ 18-7 & 19-4 \end{pmatrix} = \begin{pmatrix} 10 & 23 \\ 11 & 15 \end{pmatrix}$$
$$C = \begin{pmatrix} c_1 - c_0 \\ c_2 - c_0 \end{pmatrix} = \begin{pmatrix} 12-18 & 12-14 \\ 4-18 & 17-14 \end{pmatrix} = \begin{pmatrix} 20 & 24 \\ 12 & 3 \end{pmatrix}$$

(Beachte: Die Elemente sind aus $\mathbb{Z}_{26}$, immer den Rest der Division durch 26 verwenden.)

Nun kann man die Determinante von M bestimmen und da diese eine Einheit ist, existiert M^{-1}.

$$\det M = \begin{pmatrix} 10 & 23 \\ 11 & 15 \end{pmatrix} = 10 \cdot 15 - 23 \cdot 11 = 1$$
$$M^{-1} = (\det M)^{-1} \cdot B = 1 \cdot \begin{pmatrix} 15 & 3 \\ 15 & 10 \end{pmatrix}$$

(Für die Bestimmung von B siehe Abschnitt 4.3.)

Damit lässt sich jetzt die Matrix A und der Vektor b bestimmen.

$$A = M^{-1} \cdot C = \begin{pmatrix} 15 & 3 \\ 15 & 10 \end{pmatrix} \cdot \begin{pmatrix} 20 & 24 \\ 12 & 3 \end{pmatrix} = \begin{pmatrix} 24 & 5 \\ 4 & 0 \end{pmatrix}$$
$$b = c_0 - m_0 = \begin{pmatrix} 18 & 14 \end{pmatrix} - \begin{pmatrix} 7 & 4 \end{pmatrix} \cdot \begin{pmatrix} 24 & 5 \\ 4 & 0 \end{pmatrix} = \begin{pmatrix} 18 & 14 \end{pmatrix} - \begin{pmatrix} 2 & 9 \end{pmatrix} = \begin{pmatrix} 16 & 5 \end{pmatrix}$$

Der einzige Schutz bei affinen Verschlüsselungen besteht darin, den Schlüsselraum möglichst groß zu machen, um die Berechnungen bei einem Angriff aufwendiger werden zu lassen.

4.5. Feistel-Chiffre

Die Idee zur Feistel-Chiffre wurde 1971 bei IBM in dem Projekt „Lucifer“ entwickelt und ist heute Grundlage für viele symmetrische Blockchiffren unter anderem dem heute am häufigsten eingesetzten Verschlüsselungsalgorithmus DES – dazu später mehr in Abschnitt 4.6. Als Erfinder der Feistel-Chiffre gilt HORST FEISTEL, ein Mitarbeiter des Projekts.

Bei der Feistel-Chiffre werden die Klartextblöcke durch eine alternierende Folge von Substitutionen und Transpositionen verschlüsselt. Die Substitutionen werden in sogenannten **S-Boxen** durchgeführt und dienen dazu, die Konfusionseigenschaft der Verschlüsselung zu sichern. Die Transpositionen werden in sogenannten **P-Boxen** (von Permutation) durchgeführt, um die Diffusionseigenschaft der Verschlüsselung zu erzeugen.[3]

Einen einzelnen Durchlauf von Substitution und Transposition bezeichnet man als **Runde**. Eine (Block-)Chiffre mit mehreren Runden bezeichnet man als **iterierte Blockchiffre.**

[3] Konfusion und Diffusion sind nach CLAUDE SHANNON zwei wichtige Eigenschaften einer Verschlüsselung.

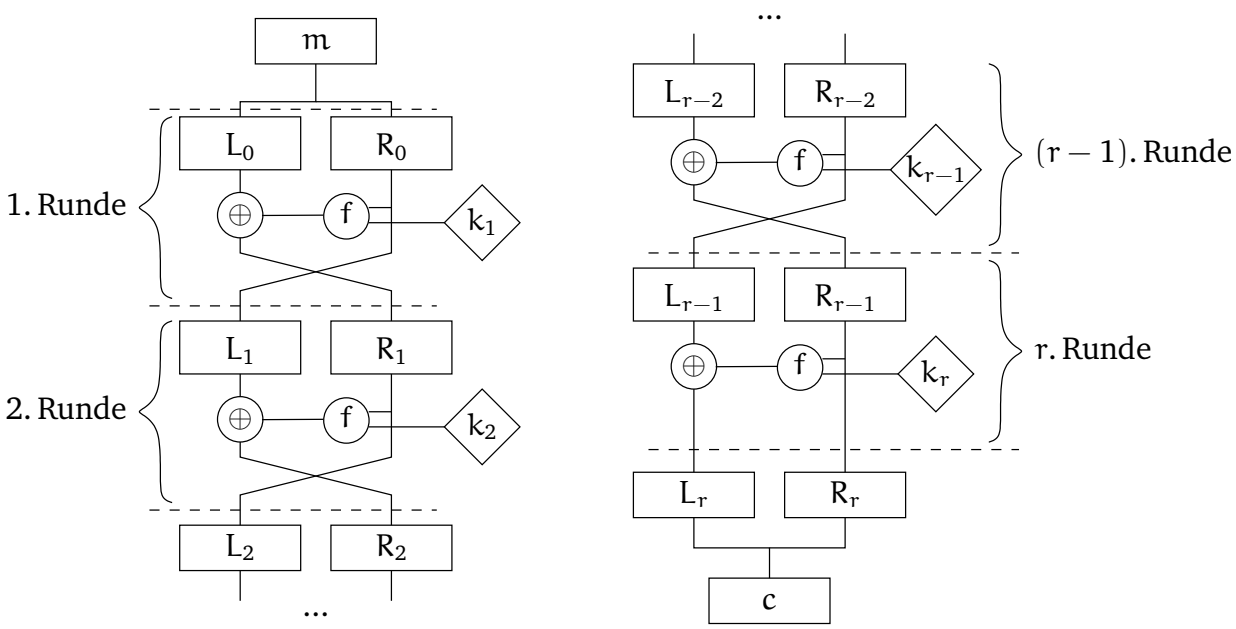

Abbildung 4.5.: Schematische Darstellung des Feistel-Netzwerks

Die einzelnen Schlüssel für jede Runde (**Rundenschlüssel** genannt) werden dabei aus einem (Haupt-)Schlüssel erzeugt.

Eine Nachricht $m \in \mathbb{Z}_2^n$ der Länge n (o. B. d. A. sei n gerade) wird in zwei gleichlange Teile L_0 und R_0 zerlegt. Aus diesen beiden wird unter Verwendung des Rundenschlüssels k_i durch die Zuordnung in Gleichung 4.3 ein neuer linker Teil L_1 und ein neuer rechter Teil R_1 gewonnen. Diese beiden Zeichenketten wiederum werden als Eingabe für die nächste Runde verwendet. Die **Rundenzahl** r bestimmt dabei, wie oft ein solcher Durchlauf stattfindet. In der letzten Runde wird die leicht modifizierte Vorschrift aus Gleichung 4.4 verwendet, bei der die Transposition entfällt.

$$\begin{aligned} L_i &= R_{i-1} \\ R_i &= L_{i-1} \oplus f_{k_i}(R_{i-1}) \end{aligned} \qquad \text{für } i = 1, \ldots, r-1 \tag{4.3}$$

$$\begin{aligned} L_r &= L_{r-1} \oplus f_{k_r}(R_{r-1}) \\ R_r &= R_{r-1} \end{aligned} \tag{4.4}$$

Dieses Schema bezeichnet man als **Feistel-Netzwerk** und eine Verschlüsselung, die danach arbeitet, nennt man eine **Feistel-Chiffre**.

$$(L_{i-1}, R_{i-1}) \xrightarrow{\text{Substitut.}} \left(L_{i-1} \oplus f_{k_i}(R_{i-1}), R_{i-1}\right) \xrightarrow{\text{Transposit.}} \left(R_{i-1}, L_{i-1} \oplus f_{k_i}(R_{i-1})\right)$$

Dabei bezeichnet $\oplus$ das Exclusive-Oder `xor` bzw. die Addition modulo 2.

Die Grafik ragt über die Ränder hinaus.

Für die Entschlüsselung einer Nachricht kann man den gleichen Algorithmus mit der umgekehrten Schlüsselfolge $k'_i := k_{r+1-i}$ (für $i = 1, \ldots, r$) auf $c = (L'_0, R'_0) = (L_r, R_r)$ anwenden und einfach die verschlüsselte Nachricht nochmals verschlüsseln. Dies hat zum Einen den Vorteil, dass man die gleiche Hard-/Software auf der Sender- und Empfängerseite einsetzen kann, und zum Anderen kann die Funktion f sehr kompliziert sein, da man nicht deren inverse Funktion benötigt.

Bei der Entschlüsselung gilt in allen Runden $i = 1, \ldots, r$ der Zusammenhang $(L'_i, R'_i) = (R_{r-i}, L_{r-i})$ und für die letzte Runde ergibt sich dann der Klartext der Nachricht als $(L'_r, R'_r) = (L_0, R_0)$, da die Ausgabe des letzten Schritts die vertauschte Ausgabe der normalen Runden ist.

$$
\begin{aligned}
(L'_1, R'_1) &= \big(R'_0,\ L'_0 \oplus f_{k'_1}(R'_0)\big) && \text{nach (4.3)}\\
&= \big(R_r,\ L_r \oplus f_{k_r}(R_r)\big) && \text{nach IV.}\\
&= \big(R_{r-1},\ \big(L_{r-1} \oplus f_{k_r}(R_{r-1})\big) \oplus f_{k_r}(R_{r-1})\big) && \text{nach (4.4)}\\
&= (R_{r-1}, L_{r-1}) && \text{da } a \oplus a = 0
\end{aligned}
$$

$$
\begin{aligned}
(L'_n, R'_n) &= \big(R'_{n-1},\ L_{n-1} \oplus f_{k'_n}(R'_{n-1})\big) && \text{nach (4.3)}\\
&= \big(L_{r-n+1},\ R_{r-n+1} \oplus f_{k_{r+1-n}}(L_{r-n+1})\big) && \text{nach IV.}\\
&= \big(R_{r-n},\ \big(L_{r-n} \oplus f_{k_{r-n+1}}(R_{r-n})\big) \oplus f_{k_{r+1-n}}(R_{r-n})\big) && \text{nach (4.3)}\\
&= (R_{r-n}, L_{r-n})
\end{aligned}
$$

Da die Schlüsselfolge $k_1, \ldots, k_r$ für die Ver- und Entschlüsselung aus dem gleichen (Haupt-)Schlüssel erzeugt wird, ist die Feistel-Chiffre eine symmetrische Verschlüsselung.

Die Sicherheit der gesamten Verschlüsselung beruht auf der Eigenschaft, dass die Substitution, genauer die Funktion f_{k_i}, eine nichtlineare Transformation ist. Wäre f_{k_i} eine lineare Transformation, so wäre auch die Verkettung von Substitution und Transposition[4] in jeder Runde eine lineare Transformation. Man könnte also den gesamten Verschlüsselungsvorgang mit einer einzigen linearen Gleichung beschreiben. Die Funktion $\oplus$ ist linear, da sich leicht zeigen lässt, dass $f(ax + by) = af(x) + bf(y)$ mit $f(x) = f(x_1, x_2) = x_1 \oplus x_2 = x_1 + x_2 \pmod 2$ für $x_1, x_2 \in \mathbb{F}_2$ folgt.

Ist die Funktion f eine kryptographisch sichere Pseudozufallsfunktion, so haben MICHAEL LUBY und CHARLES RACKOFF gezeigt[17], reichen drei Runden um eine Pseudozufallspermutation zu erreichen. Mit vier Runden bekommt man sogar eine starke Pseudozufallspermutation.[5]

4.6. DES – Data Encryption Standard

Der **Data Encryption Standard** (**DES**) wurde in den 1970ern von IBM für eine Ausschreibung des US National Bureau of Standards[6] entwickelt. Er basiert auf den vorangegan-

[4] $\begin{pmatrix} R & L \end{pmatrix} = \begin{pmatrix} L & R \end{pmatrix} \cdot \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$

[5] Geklaut aus der englischen Wikipedia, die selbst auf [17] verweist.

[6] wurde später in National Institute of Standards and Technology (NIST) umbenannt

genen Entwicklungen des Projekts **Lucifer**, das die Feistel-Chiffre entwickelt hatte. Am 23. November 1976 wurde DES als Standard für die Verschlüsselung in den amerikanischen Bundesbehörden anerkannt und im Januar 1977 erfolgte die Veröffentlichung als FIPS PUB 46[21]. Weitere Ergänzungen gab es in den folgenden Jahren.

Die International Organization for Standardization (ISO) übernahm den Algorithmus unter der Bezeichnung „Data Encipherment No. 1“ (DEA-1) in ihr Verzeichnis.

Die Kryptographen ELI BIHAM und ADI SHAMIR entwickelten gegen Anfang der 90er Jahre die **differentielle Kryptoanalyse** als Spezialform der Kryptoanalyse. Dabei wird untersucht, wie sich der Geheimtext bei unterschiedlichen Eingaben ändert. Das Verfahren wandten sie auch auf DES an.[19][20] Es zeigte sich, dass der DES einer differentiellen Kryptoanalyse stand hält, da man 2^{47} frei gewählte Klartexte dafür benötigt. Nach Angaben von IBM war dieser Angriff seit 1974 bekannt und es wurde entsprechende Gegenmaßnahmen bei der Entwicklung getroffen.[22]

Die fortschreitende Weiterentwicklung der Rechenkraft führte letztlich dazu, dass ein Brute-Force-Angriff gegen DES erfolgreich ist. Nach vielen theoretischen Überlegungen wurde 1997 das erste Mal eine verschlüsselte Nachricht gebrochen. Ein Jahr später gelingt es der Electronic Frontier Foundation (EFF) mit dem eigens dafür angefertigten Supercomputer „Deep Crack“, der 88 Milliarden Schlüssel pro Sekunde testen konnte, binnen 56 Stunden eine Verschlüsselung zu brechen. Ein halbes Jahr später, im Januar 1998, gelang es durch die Zusammenarbeit von Deep Crack und distributed.net einen DES-Schlüssel in 22 Stunden und 15 Minuten zu ermitteln – mehr als 245 Milliarden Schlüssel wurden pro Sekunde getestet.

Im Jahr 2004 wird die Absetzung von DES als Standard für die Verschlüsselung der amerikanischen Bundesbehörden widerrufen, was ein Jahr später realisiert wird. Seit 2002 ist der Advanced Encryption Standard (AES) als neuer Algorithmus in Kraft.

Der Data Encryption Standard ist eine symmetrische Blockchiffre und arbeitet mit einem Feistel-Netzwerk (Abbildung 4.5). Die Schlüssellänge für die Verschlüsselung beträgt 64 Bit, wobei ein Bit pro Byte als Prüfbit verwendet wird, was den effektiven Schlüsselraum $\mathcal{K} = \{0,1\}^{56}$ auf 56 Bit reduziert.

DES arbeitet mit 16 Runden. Aus dem Schlüssel $k \in \mathcal{K}$ werden 16 Teilschlüssel $k_1, k_2, \ldots, k_{16}$ der Länge 48 Bit generiert, die jeweils in einer Runde verwendet werden.

Die Nachricht wird in Blöcke m der Länge 64 aufgeteilt und nach einer initialen Permutation[7] (IP) in den 16 Runden transformiert und anschließend noch einmal mit IP^{-1} permutiert.

In der ersten Runde wird gemäß des **Feistel-Netzwerks** die Eingabe in eine linke und rechte Hälfte (L, R) geteilt. Die Ausgabe der Runde (L_i, R_i) wird durch $L_i = R_{i-1}$ und $R_i = L_{i-1} \oplus f_{k_i}(R_i)$ mit $i > 0$ erzeugt, wobei die Verschlüsselungsfunktion f wie folgt aufgebaut ist:

1. Diffusionsschritt: Aus den 32 Bit von R wird ein 48-Bit-Wort R_1 erzeugt.

[7] Die initiale Permutation hat keinen Einfluss auf die Sicherheit des Algorithmus'. Sie wurde damals eingeführt, um das Laden der Daten in den Chip zu vereinfachen. Da die Realisierung der IP in Software jedoch schwieriger ist, wird sie manchmal weggelassen. Eine derartige Implementierung darf jedoch nicht den Namen DES tragen.

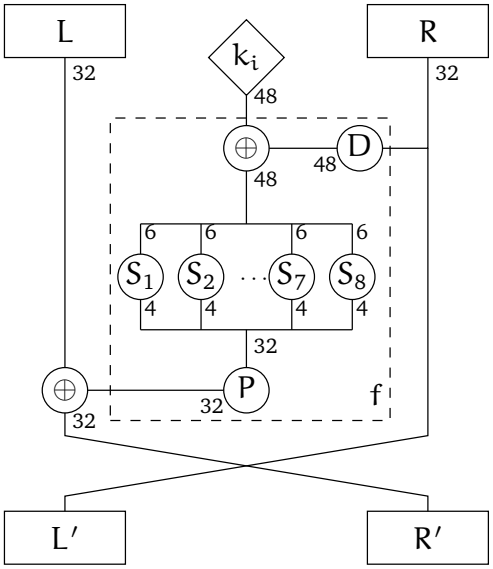

Abbildung 4.6.: Schematische Darstellung des Ablaufs in einer Runde beim DES

2. Verknüpfung mit dem Rundenschlüssel: $R_2 = R_1 \oplus k_i$.

3. Konfusionsschritt: aus R_2 wird durch eine nicht lineare Transformation S ein 32-Bit-Wort R_3 erzeugt. Dazu wird R_2 in 8 Worte der Länge 6 zerlegt und diese werden jeweils in einer eigenen S-Box S_i $(1 \leqslant i \leqslant 8)$ zu 8 Worten der Länge 4 verarbeitet. Diese aneinander gereiht ergeben das 32-Bit-Wort R_3.

4. Permutation: Die Ausgabe von f wird durch eine Blocktransposition (P-Box) aus R_3 erzeugt.

Der Ablauf ist in Abbildung 4.6 dargestellt. Für die genauen Vorschriften zur Berechnung der Diffusion, Konfusion und Permutation siehe FIPS PUB 46[21]. Leider ist deren mathematischer Hintergrund nur teilweise bekannt[22] und sie können nur als starre Operationen verstanden werden.

Aber es lässt sich zeigen, dass die beiden Eigenschaften einer Verschlüsselung nach CLAUDE SHANNON erfüllt sind: Die Änderung der Eingabe in einer S-Box in einem Bit ändert die Ausgabe in mindestens zwei Bit – **Diffusion** – und jedes Ausgabebit hängt nach fünf Runden von allen Schlüssel- und Klartextbits ab – **Konfusion**.

Die Abbildung $DES_k : \mathbb{Z}_2^{64} \to \mathbb{Z}_2^{64}$ sei die gesamte Transformation (vom IP bis zum IP^{-1}) des DES. Da der DES aufgrund seines kurzen Schlüssels nicht mehr als sicher gilt, aber im Laufe der Jahre sehr viele Implementationen des Algorithmus' in Hardware und Software geschaffen wurden, schaltet man einfach mehrere DES-Verschlüsselungen hintereinander, um die Sicherheit zu erhöhen.

Diese Kaskadierung von mehreren DES-Verschlüsselungen ist nur deshalb sinnvoll, weil DES keine **Gruppeneigenschaft** besitzt, d. h. für zwei Schlüssel k_1 und k_2 gibt es im Allgemeinen keinen Schlüssel k_3, so dass DES_{k_3} die gleiche Abbildung beschreibt

wie $DES_{k_1} \circ DES_{k_2}$. Wäre DES abgeschlossen bzgl. Verkettung (besäße also die Gruppeneigenschaft), so könnte man statt der Verkettung $DES_{k_1} \circ DES_{k_2}$ die einfache Verschlüsselung DES_{k_3} angreifen, der Schlüsselraum hätte sich also nicht vergrößert. Es gilt sogar, dass die Gruppe von Verschlüsselungen, die durch beliebige Verkettungen von DES-Verschlüsselungen erzeugt wird, mehr als 10^{2499} Verschlüsselungen umfasst.[4, S. 258][23]

- Eine Variante ist es, den Geheimtext noch einmal zu verschlüsseln: $(DES_{k_1} \circ DES_{k_2})$
  - Wenn $k_1 = k_2$ ist, bringt dies keine wesentliche Erhöhung der Sicherheit, da es einfach nur die Rundenzahl erhöht.
  - Der Fall $k_1 \neq k_2$ schafft eine echte Vergrößerung des Schlüsselraums auf 2^{112} Schlüssel. Jedoch kann man mit einem **Meet-in-the-Middle-Angriff** die maximale Anzahl der Versuche auf 2^{57} beschränken, benötigt dafür aber extrem viel Speicherplatz. Ist zu einem Klartext m der Geheimtext c bekannt (**Known-Plaintext-Angriff**), kann man damit das Paar (k_1, k_2) ermitteln.

    1. Berechne und speichere $DES_{k_1}(m)$ für alle Schlüssel k_1 (2^{56} Stück).
    2. Berechne und speichere $DES^{-1}_{k_2}(c)$ für alle Schlüssel k_2 (2^{56} Stück).
    3. Vergleiche beide Listen auf identische Einträge.

    Somit hat man bei n Schlüsseln nach $2n$ Berechnungen (anstelle von n^2) und $O(n \log n)$ Vergleichen das Schlüsselpaar (k_1, k_2) gefunden, für das gilt $DES_{k_2}(DES_{k_1}(m)) = c$.

    Unter Umständen gibt es mehrere Schlüsselpaare, die die Bedingung erfüllen. Dann benötigt man ein zweites Klartext-Chiffretext-Paar und muss damit das Verfahren nochmal wiederholen.

- Eine Verkettung der Form $DES_{k_1} \circ DES^{-1}_{k_2} \circ DES_{k_3}$ bezeichnet man als **Triple-DES** (**3DES**). Der Standard schlägt folgende Kombinationen von Schlüsseln vor:
  - $k_1 = k_2 = k_3$: Damit ergibt sich der einfache DES; Die Idee dieser Variante ist, dass man mit einer Triple-DES-Implementation weiterhin einen einfachen DES unterstützen kann.
  - $k_1 = k_3$ unabhängig von k_2: Diese Variante lässt sich leicht mit einem **Chosen-Plaintext-Angriff**, der ungefähr 2^{56} Schlüssel und 2^{56} Speicherplätze benötigt, brechen.
  - alle drei Schlüssel unabhängig voneinander: Damit erreicht man eine echte Vergrößerung des Schlüsselraums.

Bemerkung 4.3
Nach dem Prinzip der Feistel-Chiffre wäre die Entschlüsselung DES^{-1}_k eine Verschlüsselung DES_{k^R} mit dem umgekehrten Schlüssel k^R. Da aber der DES noch eine Erzeugung der Teilschlüssel aus k umfasst, ist k^R nicht der permutierte Schlüssel, der bei der Entschlüsselung DES^{-1}_k verwendet wird.

4.6.1. Ein Beispiel einer vereinfachten DES-Verschlüsselung

Zur Vereinfachung des ganzen, sei die Blocklänge $n = 12$ (statt 64) und der (Haupt-) Schlüssel k habe die Länge 9 (statt 64). Damit sind die beiden Teilblöcke L und R 6 Bit lang. Die initiale Permutation IP und die Permutation im 4. Schritt seien die Identität, sprich diese Schritte entfallen.

Die Rundenschlüssel $k_1, \ldots, k_r \in \mathbb{Z}_2^8$ werden aus dem Schlüssel $k = (\kappa_0 \quad \ldots \quad \kappa_8)$ nach folgender Vorschrift erzeugt:

$$k_{i+1} = \begin{pmatrix} \kappa_{i \pmod 9} & \kappa_{i+1 \pmod 9} & \ldots & \kappa_{i+7 \pmod 9} \end{pmatrix} \qquad 0 \leqslant i < r$$

Dadurch wiederholen sich die Rundenschlüssel zyklisch ab der 10. Runde; $k_i = k_{i+9} = k_{i+18} = \cdots$.

Die Diffusion $D\colon \mathbb{Z}_2^6 \to \mathbb{Z}_2^8$ für den ersten Schritt sei folgende lineare Abbildung

$$\begin{aligned} D(\alpha) = D(\begin{pmatrix} \alpha_1 & \alpha_2 & \ldots & \alpha_6 \end{pmatrix}) &= \begin{pmatrix} \alpha_1 & \alpha_2 & \alpha_4 & \alpha_3 & \alpha_4 & \alpha_3 & \alpha_5 & \alpha_6 \end{pmatrix} \\ &= \alpha \cdot \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \end{aligned}$$

Es soll auch nur zwei S-Boxen geben, die aus einer Eingabe von 4 Bit eine Ausgabe von 3 Bit erzeugen. In Anlehnung an den Standard seien diese in Form einer 2×8-Tabelle mit Einträgen aus $\{0, \ldots, 7\}$ gegeben.

$$S_1 = \begin{pmatrix} 5 & 2 & 1 & 6 & 3 & 4 & 7 & 0 \\ 1 & 4 & 6 & 2 & 0 & 7 & 5 & 3 \end{pmatrix}$$

$$S_2 = \begin{pmatrix} 4 & 0 & 6 & 5 & 7 & 1 & 3 & 2 \\ 5 & 3 & 0 & 7 & 6 & 2 & 1 & 4 \end{pmatrix}$$

Für eine Eingabe $\alpha = (\alpha_1 \quad \ldots \quad \alpha_4)$ ist das Ergebnis der Transformation in der $(\alpha_1 + 1)$-ten Zeile und in der $(2^2 \cdot \alpha_2 + 2 \cdot \alpha_3 + \alpha_4)$-ten Spalte abzulesen. Das erste Bit kodiert also die Zeile (0 für die erste, 1 für die zweite) und die verbleibenden drei Bits beschreiben als Binärzahl interpretiert die Spalte (000 die erste, 111 die achte).

Beispiel 4.3
Wie sieht nun der Ablauf in der 4. Runde aus, wenn der Schlüssel $k = (0100\,11001)$ lautet und die Eingabe $(001\,111\,101\,011)$ ist?

Der Rundenschlüssel k_4 lautet also $(0110\,0101)$. Für die Rundenfunktion $f(101\,011)$ ergeben sich folgende Teilergebnisse:

1. Diffusion: $R_1 = D(101\,011) = (1001\,0111)$
2. Addition des Rundenschlüssels: $R_2 = (1001\,0111) \oplus (0110\,0101) = (1111\,0010)$

3. Konfusion: Zerlegung von R_2 in zwei Teile und Ersetzung anhand der S-Boxen $S_1(1\,111) = 3 = (011)$ und $S_2(0\,010) = 6 = (110)$.

Das Ergebnis der 4. Runde lautet also $(101\,011\,011\,110)$.

4.6.2. Differentielle Kryptoanalyse des DES mit einer Runde

Die **differentielle Kryptoanalyse** wurde Ende der 1980er von ELI BAHIM und ADI SHAMIR entwickelt. Dabei wird die statistische Verteilung der Abstände der Ausgaben (vom gesamten oder von Teilen des Algorithmus') in Abhängigkeit von den Abständen der Eingaben untersucht. Treten einige Abstände häufiger als andere auf, so kann man darauf aufbauend, versuchen den Schlüssel zu ermitteln. Diese Methode des Angriffs war IBM beim Entwurf des DES bereits bekannt und so wurden entsprechende Gegenmaßnahmen ergriffen; siehe [22].

Um mit einer differentiellen Kryptoanalyse den Schlüssel $k = (\kappa_1 \dots \kappa_r)$ erlangen zu können, muss sich der Angreifer zu jedem beliebigen Klartext den Geheimtext beschaffen können (**Chosen-Plaintext-Angriff**). Außerdem muss er die Kenntnis über alle Teile des Algorithmus' haben. Die differentielle Kryptoanalyse des DES stützt sich auf zwei Punkte:

- Da der Diffusionsschritt unabhängig vom Rundenschlüssel k_i ist, kann man also das Ergebnis der Diffusion für einen beliebigen Klartext berechnen.

  Danach kommt die Addition des Rundenschlüssels. Dabei gilt aber, dass die Differenz $a \oplus b$ zwischen zwei Blöcken a und b sich nicht ändert, wenn zu jedem Block derselbe Wert (Rundenschlüssel) addiert wird: $a \oplus b = (a \oplus k_i) \oplus (b \oplus k_i)$.

  Für zwei Klartexte kann man also aus der Kenntnis über die Ergebnisse der Diffusion auf die Differenz der Ergebnisse der Schlüsseladdition schließen.

- Dieses Wissen über die Eingabe in die S-Box und die Ausgabe der S-Box, den Geheimtext (wobei man wieder die Berechnung der P-Box rückgängig machen kann), kann man ausnutzen, um die Menge der möglichen Eingabewerte in die S-Box einzugrenzen. Damit kann man dann die Menge der möglichen Schlüssel eingrenzen bzw. mit mehreren Klartext-Geheimtext-Paaren den Schlüssel bestimmen.

Für zwei Eingabeblöcke $a, \tilde{a} \in \mathbb{Z}_2^m$ und einen S-Box $S\colon \mathbb{Z}_2^m \to \mathbb{Z}_2^n$ des Konfusionsschritts heißt $a^* = a \oplus \tilde{a} \in \mathbb{Z}_2^m$ **Eingabedifferenz** (Input-Differenz) und $S_1(a) \oplus S_1(\tilde{a}) \in \mathbb{Z}_2^n$ **Ausgabedifferenz** (Output-Differenz). Beim DES ist $m = 6$ und $n = 4$, in dem Beispiel in Abschnitt 4.6.1 ist $m = 4$ und $n = 3$.

Für ein festes $a^* \neq 0 \in \mathbb{Z}_2^m$ gibt es genau 2^{m-1} Mengen[8] der Form $\{a, \tilde{a}\}$, für die $a \oplus \tilde{a} = a^*$ ist. Theoretisch sind 2^n verschiedene Ausgabedifferenzen $S(a) \oplus S(\tilde{a}) \in \mathbb{Z}_2^n$ möglich, jedoch sind für einige Eingabedifferenzen (insgesamt gibt es $2^m - 1$) die Ergebnisse einiger S-Boxen nicht gleichverteilt.

[8] Man hat 2^m Möglichkeiten für die Wahl von a. $\tilde{a}$ ist dann fest $\tilde{a} = a \oplus a^*$. Da die Menge ungeordnet ist, entfällt die Hälfte der Möglichkeiten.

Beispiel 4.4
Die S-Box sei die S_1 aus Abschnitt 4.6.1 und die Eingabedifferenz sei $a^* = (0011)$. Es ergeben sich damit folgende Mengen $\{a, \widetilde{a}\}$ mit ihren Ausgabedifferenzen. Die Ausgabedifferenz (011) tritt dabei in 6 von 8 Fällen auf.

a	$\widetilde{a}$	$S_1(a)$	$S_1(\widetilde{a})$	$S_1(a) \oplus S_1(\widetilde{a})$
(0000)	(0011)	(101)	(110)	(011)
(0001)	(0010)	(010)	(001)	(011)
(0100)	(0111)	(011)	(000)	(011)
(0101)	(0110)	(100)	(111)	(011)
(1000)	(1011)	(001)	(010)	(011)
(1001)	(1010)	(100)	(110)	(010)
(1100)	(1111)	(000)	(011)	(011)
(1101)	(1110)	(111)	(101)	(010)

Die Eingabe der i-ten Runde sei der Block (L_{i-1}, R_{i-1}), wobei für f_{k_i} nur R_{i-1} relevant ist. Der Ablauf ist nun wie folgt:

1. Diffusion: $d = D(R_{i-1})$
2. Schlüsselverknüpfung: $a = d \oplus k_i$
3. Konfusionsschritt: Zerlegung von a in Blöcke a^ℓ der Länge m und Transformation dieser in der jeweiligen S-Box S_ℓ.
4. (Permutationsschritt:) Dieser muss nicht betrachtet werden, da man diesen Schritt ausgehend vom Geheimtext einfach rückgängig machen kann.

Teilt man bereits das Ergebnis der Diffusion $D(R_{i-1})$ in Blöcke d^ℓ und den Rundenschlüssel k_i in Blöcke k_i^ℓ entsprechend der S-Box, in die das Ergebnis der Schlüsselverknüpfung $D(R_{i-1}) \oplus k_i$ gelangt, so kann man das Ergebnis der Rundenfunktion wie folgt darstellen:

$$\begin{aligned} f_{k_i}(R_{i-1}) &= \begin{pmatrix} S_1(a^1) & S_2(a^2) & \dots & S_\ell(a^\ell) & \dots \end{pmatrix} \\ &= \begin{pmatrix} S_1(d^1 \oplus k_i^1) & S_2(d^2 \oplus k_i^2) & \dots & S_\ell(d^\ell \oplus k_i^\ell) & \dots \end{pmatrix} \end{aligned}$$

Für zwei verschiedene Eingaben R_{i-1} und $\widetilde{R}_{i-1}$ in der i-ten Runde ist die Eingabedifferenz der ℓ-ten S-Box

$$a^\ell \oplus \widetilde{a}^\ell = (d^\ell \oplus k_i^\ell) \oplus (\widetilde{d}^\ell \oplus k_i^\ell) = d^\ell \oplus \widetilde{d}^\ell \tag{4.5}$$

Man kann also aus der Ausgabe des Diffusionsschritts (1. Schritt) Rückschlüsse auf die Eingabe des Konfusionsschritts (3. Schritt) ziehen. Die Ausgabedifferenz der ℓ-ten S-Box ist

$$S_\ell(a^\ell) \oplus S_\ell(\widetilde{a}^\ell) = S_\ell(d^\ell \oplus k_i^\ell) \oplus S_\ell(\widetilde{d}^\ell \oplus k_i^\ell)$$

Sind die Ausgaben des Diffusionsschritts d^ℓ und $\widetilde{d}^\ell$ und die Ausgabedifferenz des Konfusionsschritts $S_\ell(a^\ell) \oplus S_\ell(\widetilde{a}^\ell)$ bekannt, so kann man mit Hilfe dieser Größen über die bekannte Verteilung der Eingabe- und Ausgabedifferenzen für die S-Box S_ℓ die Menge der möglichen Eingaben $\{a^\ell, \widetilde{a}^\ell\}$ in die S-Box bestimmen. Über die Beziehung $a^\ell = d^\ell \oplus k_i^\ell$ ergibt sich so eine Einschränkung für den Teilrundenschlüssel k_i^ℓ.

Beispiel 4.5
Das Beispiel bezieht sich auf den vereinfachten DES aus Abschnitt 4.6.1. Es soll nur um die Betrachtung der S-Box S_1 ($\ell = 1$) gehen. Die Ausgaben des Diffusionsschritts seien $d^1 = (0101)$ und $\widetilde{d}^1 = (0110)$. Die Ausgabedifferenz der S-Box sei (010).

Die Eingabedifferenz der S-Box ist nach Gleichung 4.5 $d^1 \oplus \widetilde{d}^1 = (0011)$. Laut der Aufstellung in Beispiel 4.4 sind als Eingabe a^1 in die S-Box die Werte $\{(1001), (1010), (1101), (1110)\}$ möglich.

Daraus ergeben sich durch die Beziehung $k_i^1 = d^1 \oplus a^1$ als mögliche Teilrundenschlüssel $\{(1100), (1111), (1000), (1001)\}$.

Mit mehreren Klartextblöcken kann man so verschiedene Schlüsselmengen ermitteln. Der tatsächliche Schlüssel liegt im Durchschnitt aller Schlüsselmengen.

4.6.3. Differentielle Kryptoanalyse des DES mit drei Runden

Diese Überlegungen für eine Runde, lassen sich auch auf ein System mit drei Runden erweitern. (L_0, R_0) und $(\widetilde{L}_0, R_0)$ seien die Eingabe, wobei der rechte Teil bei beiden gleich ist, und $(L_3, R_3), (\widetilde{L}_3, \widetilde{R}_3)$ die zugehörigen Geheimtextblöcke. L_3 bestimmt sich laut Feistel-Netzwerk (Gleichung 4.3 und Gleichung 4.4) als

$$L_3 = L_2 \oplus f_{k_3}(R_2) = R_1 \oplus f_{k_3}(R_2) = L_0 \oplus f_{k_1}(R_0) \oplus f_{k_3}(R_2)$$

Analog gilt $\widetilde{L}_3 = \widetilde{L}_0 \oplus f_{k_1}(R_0) \oplus f_{k_3}(\widetilde{R}_2)$. Die Ausgabedifferenz der dritten Runde ist also

$$L_3^* = L_3 \oplus \widetilde{L}_3 = L_0 \oplus \widetilde{L}_0 \oplus f_{k_3}(R_2) \oplus f_{k_3}(\widetilde{R}_2)$$

Die Eingabedifferenz $L_0^* = L_0 \oplus \widetilde{L}_0$ ist bekannt. Da die Permutation (4. Schritt) eine lineare Abbildung ($g(a) + g(b) = g(a + b)$) ist, erhält man die Ausgabedifferenz des Konfusionsschritts

$$L_3^* \oplus L_0^* = f_{k_3}(R_2) \oplus f_{k_3}(\widetilde{R}_2) = P\big(S(a_3)\big) \oplus P\big(S(\widetilde{a}_3)\big) = P\big(S(a_3 \oplus \widetilde{a}_3)\big)$$

Da $R_3 = R_2$ (Gleichung 4.4) ist, kann man für $f_{k_3}(R_2)$ die Diffusion $D(R_2)$ bestimmen. Damit hat wieder wieder die Situation wie im Fall mit einer Runde erreicht: Man kennt die Ausgabe des Diffusionsschritts für $f_{k_3}(R_2)$ und $f_{k_3}(\widetilde{R}_2)$ und man kennt die Ausgabedifferenz des Konfusionsschritts $S(a) \oplus S(\widetilde{a})$. Man ist also in der Lage, den Schlüssel für die 3. Runde zu bestimmen. Mit diesem Schlüssel kann man dann $f_{k_3}(R_2)$ und damit $L_2 = L_3 \oplus f_{k_3}(R_2)$ bestimmen.

Für den Schlüssel der zweiten Runde kann man direkt die Ausgabedifferenz des Konfusionsschritts bestimmen, denn $R_2 = L_1 \oplus f_{k_2}(R_1) = R_0 \oplus f_{k_2}(R_1)$. Die Diffusion

$D(R_1) = D(L_2)$ lässt sich wieder berechnen und damit besteht wieder eine Situation wie im Fall mit einer Runde. Danach verbleibt nur noch eine Runde.

Beispiel:

$$(L_0, R_0) = (000\,111, 011\,011)$$
$$(L_0', R_0') = (101\,110, 011\,011)$$
$$(L_3, R_3) = (100\,101, 000\,011)$$
$$(L_3', R_3') = (011\,000, 100\,100)$$

Ziel: Bestimme den Schlüssel K (bzw. Einschränkung der Möglichkeiten)

$$R_2 = R_3 = (000\,011) \qquad D(R_2) = (0000, 0011) = (\widetilde{A}, \widetilde{B})$$
$$R_2' = R_3' = (100\,100) \qquad D(R_2') = (1010, 1000) = (\widetilde{A}', \widetilde{B}')$$

Eingabedifferenz in S-Box $S_1 = \widetilde{A} \oplus \widetilde{A}' = (1010)$
Ausgabedifferenz von S_1: linke Hälfte von $L_3^* \oplus L_0^*$
linke Hälfte von $L_3^* \oplus L_0^*$

$$L_3^* = (100\,101) \oplus (011\,000) = (111\,101)$$
$$L_0^* = (000\,111) \oplus (101\,110) = (101\,001)$$
$$L_3^* \oplus L_0^* = (010\,100)$$

Die Ausgabedifferenz von S_1 ist (010). kam in der Tabelle oben (siehe Beispiel 4.4) zweimal vor. $A \in \{(0011), (1001)\}$
$K_3^L = A \oplus \widetilde{A} = \{(0011), (1001)\}$ zwei Möglichkeiten.

$$S_2 = \begin{pmatrix} 4 & 0 & 6 & 5 & 7 & 1 & 3 & 2 \\ 5 & 3 & 0 & 7 & 6 & 2 & 1 & 4 \end{pmatrix} \qquad B^* = \begin{pmatrix} 1 & 0 & 1 & 1 \end{pmatrix}$$

Eingabedifferenz der S-Box S_2:

$$\widetilde{B} \oplus \widetilde{B}' = (0011) \oplus (1000) = (1011)$$

Ausgabedifferenz der S-Box S_2: rechte Hälfte von $L_3^* \oplus L_0^*$: (100).

liefert die Schlüssel $K_3^R \in \{(1111), (0100)\}$ zwei Möglichkeiten $\rightarrow$ vier Möglichkeiten für K_3 und damit acht Möglichkeiten für K statt 512.

$\widetilde{B}$	$\widetilde{B}'$	$S_2(\widetilde{B})$	$S_2(\widetilde{B}')$	$S_2(\widetilde{B}) \oplus S_2(\widetilde{B}')$	
(0000)	(1011)	(100)	(111)	(011)	
(0001)	(1010)	(000)	(000)	(000)	
(0010)	(1001)	(110)	(011)	(101)	
(0011)	(1000)	(101)	(101)	(000)	
(0100)	(1111)	(111)	(100)	(011)	
(0101)	(1110)	(001)	(001)	(000)	
(0110)	(1101)	(011)	(010)	(001)	
(0111)	(1100)	(010)	(110)	(100)	$(*)$

$(*)$ $B = \{(0111), (1100)\}$ $\qquad K_3^R = B \oplus \widetilde{B} = \{(0100), (1111)\}$

Brauchen ein zweites Klartextpaar: z. B. $(L_0, R_0) = (010\,111, 011\,011)$ und $(L_0', R_0') = (101\,110, 011\,011)$ mit $(L_3, R_3) = (001\,010, 001\,011)$ und $(L_3', R_3') = (011\,000, 100\,100)$.

Unsere Aufgabe ist es, K_3^L und K_3^R sowie anschließend K zu bestimmen. Dazu nehmen wir an, $K_3 = (x_1x_2x_3x_4x_5x_6x_7x_8)$. Dann hat K die Gestalt $K = (x_p x_1x_2x_3x_4x_5x_6x_7)$.

$$L_3 = L_2 \oplus f_{k_3}(R_2) = R_1 \oplus f_{k_3}(R_2)$$
$$L_0 \oplus f_{k_1}(R_0)$$

4.6.4. Differentielle Analyse des DES mit vier Runden

Ansatz: „Eigenschaft 2“: Es gibt einige Differenzen D, so dass es „viele“ Eingabepaare mit der Differenz D gibt, derart dass die Ausgabedifferenzen übereinstimmen.

Beispiel: S-Box S_1 liefert für $D = A^* = (0011)$ in 6 von 8 Fällen die Ausgabedifferenz (011). $S_1(A) \oplus S_1(A')$. Das heißt: Für $A^* = (0011)$ erwarten wir mit einer Wahrscheinlichkeit von $\frac{3}{4}$ die Ausgabedifferenz (011).

S-Box S_2 liefert für $D = B^* = (1100)$ in 4 von 8 Fällen die Ausgabedifferenz (010). Das heißt: Für $B^* = (1100)$ erwarten wir mit einer Wahrscheinlichkeit von $\frac{1}{2}$ die Ausgabedifferenz (010).

Idee: Für Paare von Klartextblöcken (L_0, R_0) wird (L_0', R_0') mit $L_0 \oplus L_0' = (0011)$ und $R_0 \oplus R_0' = (1100)$ erwarten wir (beim richtigen Schlüssel) mit Wahrscheinlichkeit $\frac{3}{4} - \frac{1}{2} = \frac{3}{8}$ die Ausgabedifferenz $(011\,010)$.

4.6.5. Analyse für vier Runden

Ansatz: Nichtgleichverteilung der Output-Differenzen.
Idee: Versuchen eine Situation nach einer Runde zu erreichen, so dass wir dann die 3-Rundenanalyse ansetzten können.

Dazu wählen wir Paare von Klartextblöcken (L_0, R_0) und (L_0', R_0') mit folgenden Eigenschaften:

1. $R_0^* = R_0 \oplus R_0' = (001\,100)$

Bei der Bildung von $f_{K_1}(R_0)$ und $f_{K_1}(R'_0)$ passiert folgendes:

$$D(R_0) \oplus D(R'_0) = (00111100)$$

Input für S_1 ist (0011) für S_2 ist es (1100).

für (0011) erwarten wir mit Wahrscheinlichkeit $\frac{3}{4}$ den Output (011) bei S_1 und für (1100) erwarten wir mit Wahrscheinlichkeit $\frac{1}{2}$ den Output (010) bei S_2.

Wir betrachten die Outputs der beiden S-Boxen als unabhängig und erwarten deshalb mit Wahrscheinlichkeit $\frac{3}{4} \cdot \frac{1}{2} = \frac{3}{8}$ eine gesamte Output-Differenz von $f_{K_1}(R_0) \oplus f_{K_1}(R'_0) = (011\,010)$

2. $L^*_0 = L_0 \oplus L'_0 = (011\,010)$

   $R_1 = L_0 \oplus f_{K_1}(R_0), R'_1 = L'_0 \oplus f_{K_1}(R'_0)$. Für den Fall $f_{K_1}(R_0) \oplus f_{K_1}(R'_0) = (011\,010)$ ergibt sich:

$$\begin{aligned} R_1 \oplus R'_1 &= (L_0 \oplus L'_0) \oplus (f_{K_1}(R_0) \oplus f_{K_1}(R_0)) \\ &= (011\,010) \oplus (011\,010) = (000\,000) \end{aligned}$$

   Da $L_1 = R_0$ und $L'_1 = R'_0$ gilt mit Wahrscheinlichkeit $\frac{3}{8}$, dass (L_1, R_1) und (L'_1, R'_1) die Eigenschaft $R_1 = R'_1$ haben und $(L_1 \oplus R_1) \oplus (L'_1, R'_1) = (R_0 \oplus R'_0, 0 \ldots 0) = (001\,100, 000000)$.

Strategie: Wähle Klartextpaare (L_0, R_0) und (L'_0, R'_0) mit der Summe $(011\,010, 001\,100)$ und bestimme dazu (L_4, R_4) und (L'_4, R'_4). Angenommen es ist so, dass $R_1 \oplus R'_1 = (000\,000)$ (das gilt in 3 von 8 Fällen), dann haben wir die Ausgangssituation zur Analyse der Runden 2, 3 und 4 als 3-Rundenanalyse.

Dort, wo die Annahme korrekt ist, erhalten wir eine Menge möglicher Schlüssel für K^L_4 bzw. K^R_4. Dort, wo die Annahme falsch ist (die Mehrzahl der Fälle), erhalten wir irgendwelche (zufälligen) Bitfolgen von acht Bit.

4.6.6. Die Sicherheit von DES

1. Die Größe des Schlüsselraums ist 2^{56} (es genügt aber 2^{55} Schlüssel zu testen), mit erschöpfender Suche war dies 1977 de facto nicht möglich.

   1997: setzt die RSA-Data-Security US-$ 10 000 für die Dechiffrierung eines DES-Textes aus. Nach 5 Monaten war der Schlüssel ermittelt von RONALD VESPER – durch organisiertes Parallelrechnen via Internet (25 % des Schlüsselraums wurde getestet.)

   1998: setzt die RSA-Data-Security wieder einen Preis aus und nach 39 Tagen war wieder der Schlüssel gefunden, obwohl 85 % des Schlüsselraums durchsucht wurden.

   1999: Electronic Frontier Foundation, 1536 parallel arbeitende Spezialchips benötigen im Schnitt $4\frac{1}{2}$ Tage um den kompletten Schlüsselraum zu durchsuchen.

Quelle: FAQ der EFF zum DES-Cracker und Cracking DES von der EFF

2006: Ein Team der Universitäten Kiel und Bochum entwickeln COPACOBANA. Mit Kosten von etwa US-$ 10 000 berechnet die Maschine 65 Milliarden DES-Schlüssel pro Sekunde.

2. Es gibt bei DES **schwache Schlüssel**, die für alle Runden den gleichen Rundenschlüssel erzeugen. Aber man kann diese auch einfach beim Einsatz vermeiden.

3. Differentielle Kryptoanalyse funktioniert nicht, wenn man die S-Boxen geeignet wählt. (Nicht-Gleichverteilung vermeiden.)

DES und differentielle Kryptoanalyse

Die differentielle Kryptoanalyse war den Designern des DES bekannt und deshalb folgte die Verteidigung. S-Boxen sind so konstruiert, dass für höchstens ein Viertel der Zweiermengen $\{A, A'\}$ die Ausgabedifferenzen $S(A) \oplus S(A')$ übereinstimmen. Mit der Folge, dass für einen Chosen-Plaintext-Angriff 2^{47} Klartextblöcke benötigt werden. Ein Known-Plaintext-Angriff benötigt 2^{55} Klartextblöcke. Dies ist nicht wesentlich weniger als für einen Brute-Force-Angriff benötigt werden: 2^{56}. Deshalb sind es 16 Runden!

DES und lineare Kryptoanalyse

Von M. MATSUI auf der Konferenz EUROCRYPT im Jahre 1993 vorgestellt ([18]). War den Designern von DES nicht bekannt.

Weder die S-Boxen noch die gesamte Verschlüsselung beschreiben eine lineare Abbildung. Damit sicher gegen solche Angriffe.

Die Idee der **linearen Kryptoanalyse** ist: Approximation der Verschlüsselung durch lineare Abbildungen. Ansatz: Bilde `xor` für einige Klartextbits und `xor` für einige Geheimtextbits und verknüpfe beide mit `xor`. Das Ergebnis ist ein einzelnes Bit, das als die `xor`-Verknüpfung einiger Schlüsselbits interpretiert wird. Das Ganze wieder für eine bestimmte Wahrscheinlichkeit p. Wenn $p \neq \frac{1}{2}$, Ausnutzung der Asymmetrie (insbesondere ist der fünfte S-Boxtyp des DES dafür anfällig) liefert eine Einschränkung der Schlüssel.

In der Arbeit von M. MATSUI wurde gezeigt, dass man für einen Known-Plaintext-Angriff nur 2^{43} Paare, d. h. sie ist besser als die **differentielle Kryptoanalyse**.

4.7. International Data Encryption Algorithm (IDEA)

Der **International Data Encryption Algorithm** (kurz **IDEA**) wurde zu Beginn der neunziger Jahre des letzten Jahrhunderts in einer Gemeinschaftsarbeit von JAMES MASSEY und XUEIJA LAI von der ETH Zürich und der Ascom Systec AG entwickelt. Die Ascom Systec AG hält bis 2010 bzw. 2011 das Patent für IDEA. Der Algorithmus kann jedoch für nichtkommerzielle Verwendung kostenlos benutzt werden. Daher wird er unter anderem im PGP-Verfahren eingesetzt.

IDEA arbeitet mit der Blocklänge von 64 Bit, einer Schlüssellänge von 128 Bit und 8 Runden zuzüglich einer Abschlussrunde. Durch den riesigen Schlüsselraum wird ein Brute-Force-Angriff gegenüber DES wesentlich erschwert.

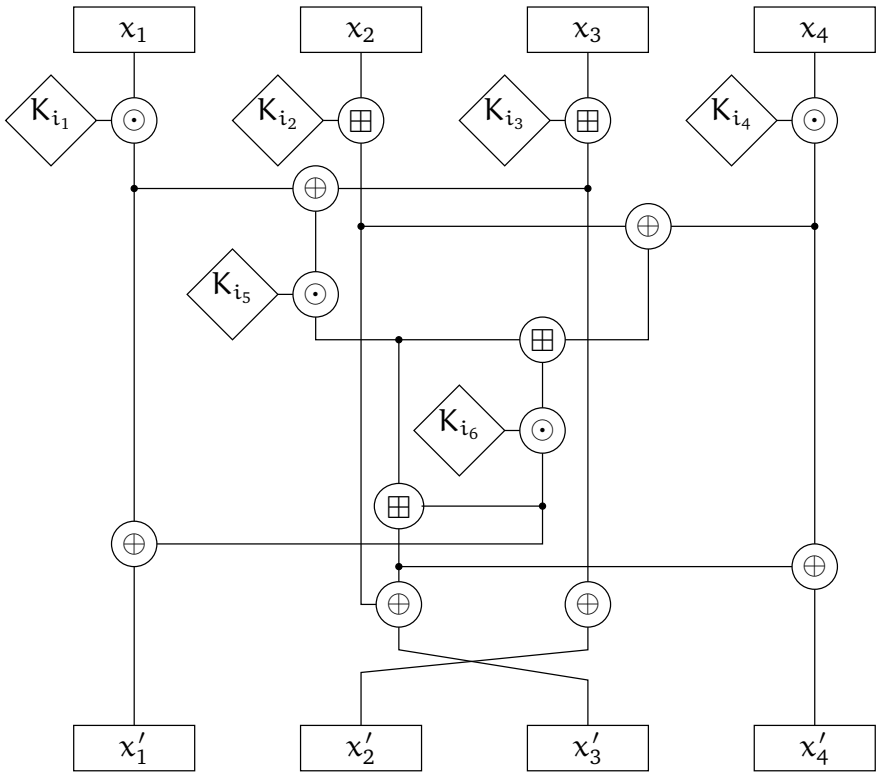

Abbildung 4.7.: Blockschema für den Ablauf der i-ten Runde der IDEA-Verschlüsselung

Der Ablauf jeder Runde ist in Abbildung 4.7 dargestellt. Jeder Klartextblock wird in vier Teilblöcke zu je 16 Bit zerlegt, die durch Verknüpfung mit 6 Rundenschlüsseln bzw. den Ergebnissen der Verknüpfungen in vier Teilblöcke von je 16 Bit transformiert werden. Als Verknüpfung werden die drei Operationen Exklusives-Oder ($\oplus$), Addition modulo 2^{16} ($\boxplus$) und Multiplikation modulo $2^{16}+1$ ($\odot$) eingesetzt.

Nach der 8. Runde folgt eine Schlussrunde:

$$x'_1 = x_1 \odot k_{9_1} \qquad x'_2 = x_2 \boxplus k_{9_2} \qquad x'_3 = x_3 \boxplus k_{9_3} \qquad x'_4 = x_4 \odot k_{9_4}$$

Da IDEA andere algebraische Operationen einsetzt, wird er nicht als Feistel-Chiffre bezeichnet. Dennoch ist er so konstruiert, dass die Verschlüsselungsfunktion auch als Entschlüsselungsfunktion verwendet werden kann.

Die Sicherheit von IDEA ist auf der Unverträglichkeit der drei unterschiedlichen arithmetischen Operationen begründet. Es ist zwar eine Klasse von **schwachen Schlüsseln** mit 2^{65} Elementen bekannt, dennoch ist der Algorithmus durch die komplexen Runden sicher gegen lineare und differentielle Kryptoanalyse.[16]

4.8. RC-Familie

In den 1990ern wurden von RONALD RIVEST von der RSA Data Security eine Reihe von Verschlüsselungsverfahren entwickelt, die als **RC-Familie** (Ron's Code oder Rivest's Cipher) zusammengefasst werden.[14, Abs. 3.6.2–4]

RC2 bezeichnet eine Feistelchiffre, die mit 18 Runden und einer Klartextblocklänge von 64 Bit arbeitet. Die Schlüssellänge muss ein Vielfaches von acht sein und kann zwischen einschließlich 8 und 128 Bit gewählt werden.

RC2 wurde 1987 als Ersatz für DES entwickelt und ist als solcher auch schneller und, bei geeigneter Wahl des Schlüssels, sicherer als DES. RC2 wurde, bis es 1996 von einem Unbekannten im Usenet veröffentlicht wurde, geheim gehalten.

RC5 wurde 1994 ebenfalls für die RSA Data Security entwickelt und weist ungewöhnlich viele Freiheiten bei der Wahl der Parameter auf. Die Blocklänge kann 32, 64 oder 128 Bit sein. Die Schlüssellänge kann von 0 bis 2040 Bit und die Rundenanzahl von 0 bis 255 gewählt werden.

Es gibt drei Operationen: Schlüsselerweiterung, Ver- und Entschlüsselung. Bei der Schlüsselerweiterung wird in Abhängigkeit von der Rundenanzahl eine Tabelle für die Ver-/Entschlüsselung erzeugt. Die Ver-/Entschlüsselung arbeitet mit Addition, exklusiven Oder und Bitrotation. Durch einen extrem einfachen Aufbau ist der RC5-Algorithmus leicht zu implementieren und zu verifizieren.

$$x_1' = \mathrm{rot}(x_1 \oplus x_2, x_2) \boxplus K_1 \qquad x_2' = \mathrm{rot}(x_2 \oplus x_1', x_1') \boxplus K_2$$

RC5 ist sicher gegen differentielle und lineare Kryptoanalyse.[15]

Um die Kriterien für die Ausschreibung des Advanced Encryption Standards zu erfüllen, wurde RC5 etwas verändert und als **RC6** eingereicht. Die Blockgröße, Schlüssellänge und Rundenanzahl können aus den selben Bereichen wie bei RC5 gewählt werden. RC6 wurde 1998 veröffentlicht und gelangte auch in die letzte Ausscheidungsrunde für den AES.

4.9. Blowfish und Twofish

Blowfish (1994), Twofish (1998) von Bruce Schneier

Blowfish variable Schlüssellänge (bis 448 Bit), S-Boxen schlüsselabhängig, 16 Runden, Blocklänge 64. Trotz aufwendiger S-Boxen ein schnell zu implementierender Algorithmus (`xor` und Addition von 32-Bitwerten)

Twofish Runde der letzten 5 AES-Kandidaten. Schlüssellänge: 128, 192, 256, Blocklänge: 64, 8×8 S-Boxen, arithmetische Operationen in $\mathbb{F}_{2^8}$ endlicher Körper der Ordnung 2^8 (anstelle von $\mathbb{Z}_2 = \mathbb{F}_2$)

4.10. Sonstige Blockchiffren

SAFER 1995, Massey

CAST 1990, 1997 – wird bei PGP benutzt

Skipjack NSA, 1990

KASUMI Weiterentwicklung von MISTY, wird zur Verschlüsselung bei UMTS eingesetzt.

4.11. Exkurs über endliche Körper

4.11.1. Endliche Körper von Primzahlordnung

Wir wissen bereits aus Abschnitt 4.3, dass sich die ganzen Zahlen in Äquivalenzklassen modulo m zerlegen lassen:

$$\mathbb{Z}_m := \mathbb{Z}_{/\equiv(m)} = \{[0]_m, [1]_m, \ldots, [m-1]_m\} = \{0,1,\ldots,m-1\}$$

Damit ergibt sich für $m > 1$ ein Ring $[\mathbb{Z}_m, +, \cdot]$ mit dem Nullelement $0 := [0]_m$, dem Einselement $1 := [1]_m$ und insgesamt $|\mathbb{Z}_m| = m$ Elementen. Ein Element $a \in \mathbb{Z}_m$ heißt genau dann Einheit, wenn es ein inverses Element b mit $a \cdot b = 1$ gibt. Nach dem Lemma 4.1 gilt, a ist genau dann eine Einheit in $\mathbb{Z}_m$, wenn $\mathrm{ggT}(a, m) = 1$ ist.

Man bezeichnet $\mathbb{Z}_m^*$ als die **Menge der Einheiten** in $\mathbb{Z}_m$, also gilt: $|\mathbb{Z}_m^*| = \varphi(m)$ (Lemma 4.1). Wenn p eine Primzahl ist, dann ist $\mathbb{Z}_p^* = \{1, 2, \ldots, p-1\}$ wegen $\varphi(p) = p-1$ und $\mathbb{Z}_p$ ein endlicher Körper der Ordnung p.

4.11.2. Polynomringe

Es sei $[R, +, \cdot]$ ein (endlicher) Ring, z. B. $[\mathbb{Z}_m, +, \cdot]$, und x ist ein Symbol, das nicht im Ring vorkommt ($x \notin R$). Wir definieren **Polynome** vom Grad k, mit $r_i \in R$ für $i = 0, \ldots, k$, wobei $r_k \neq 0$ ist, wie folgt:

$$p(x) := r_k \cdot x^k + r_{k-1} \cdot x^{k-1} + \cdots + r_1 x^1 + r_0$$

$R[x]$ bezeichnet die **Menge aller Polynome** in x über R.

Die Addition und Multiplikation von zwei Polynomen erfolgt wie üblich komponentenweise. Damit bildet die Struktur $[R[x], +, \cdot]$ einen (abzählbar unendlichen) **Polynomring**.

Ein Polynom $p \in R[x]$ heißt genau dann **reduzibel**, wenn es zwei Polynome $r, s \in R[x]$ vom Grad größer 0 gibt, so dass man p als Produkt von r und s schreiben kann; $p(x) = r(x) \cdot s(x)$.

Ein Polynom $p \in R[x]$ heißt genau dann **irreduzibel**, wenn es nicht reduzibel ist, d. h. es keine zwei Polynome $r, s \in R[x]$ gibt, so dass $p = r \cdot s$ ist.

Man kann sich die Bedeutung von irreduzibel wie die von prim für ganze Zahlen vorstellen. Eine Zahl heißt prim, wenn sie keine (nichttrivialen) Teiler besitzt und analog bezeichnet man ein Polynom als irreduzibel, wenn es sich nicht als Produkt (nicht trivialer) Polynome darstellen lässt.

4.11.3. $\mathbb{F}_{p^k}$ endlicher Körper der Ordnung p^k

Analog zur Zerlegung der ganzen Zahlen in Restklassen anhand eines Moduls definiert man die Zerlegung der Menge aller Polynome (mit Koeffizienten aus $\mathbb{Z}_p$; p prim) in Restklassen, wobei man die Faktormenge (Menge der Restklassen) mit $\mathbb{F}_{p^k}$ identifiziert.

Dazu benötigt man ein irreduzibles Polynom $m \in \mathbb{Z}_p[x]$ vom Grad k, das als **Modulpolynom** dient. Zwei Polynome $s, t \in \mathbb{Z}_p[x]$ sind in der gleichen Restklasse, wenn sie bei der Division durch m den gleichen Rest $r \in \mathbb{Z}_p[x]$ vom Grad kleiner k lassen:

$$\begin{aligned} s(x) &= a(x) \cdot m(x) + r(x) \qquad \text{und} \qquad t(x) = b(x) \cdot m(x) + r(x) \\ [r]_m &= \{\, r(x) + l(x) \cdot m(x) \mid l \in \mathbb{Z}_p[x] \,\} \\ \mathbb{Z}_p[x]/_{m(x)} &= \left\{\, [r(x)]_{m(x)} \;\middle|\; r(x) \in \mathbb{Z}_p[x], r \text{ vom Grad} < \text{dem Grad von } m \,\right\} \end{aligned}$$

Das Polynom r heißt **Repräsentant** von s und t in $\mathbb{F}_{p^k}$ und hat die Form

$$r(x) = \rho_{k-1}x^{k-1} + \rho_{k-1}x^{k-2} + \cdots + \rho_1 x + \rho_0$$

mit den Koeffizienten $\rho_{k-1}, \rho_{k-2}, \ldots, \rho_1, \rho_0 \in \mathbb{Z}_p$. Es gibt also p^k verschieden Polynome dieser Form, also hat die Menge $\mathbb{Z}_p[x]/_{m(x)}$ die Mächtigkeit p^k.

$$\mathbb{F}_{p^k} = \mathbb{Z}_p[x]/_{m(x)}$$

Für das Rechnen mit diesen Restklassen gilt: Die Addition $\oplus$ und die Multiplikation $\odot$ erfolgen wie üblich bei Polynomen, nur dass bei der Multiplikation noch eine Division durch den Modul $m(x)$ angeschlossen wird, wobei der Rest das Ergebnis ist.

Die Multiplikation $\odot$ hängt also von dem Modulpolynom $m(x)$ ab. Jedoch sind die Körper $\mathbb{F}_{p^k}[m(x)]$ und $\mathbb{F}_{p^k}[m'(x)]$ isomorph zu einander. Es ist also völlig egal, welches Modulpolynom man wählt, die grundlegende mathematische Struktur bleibt die gleiche. Daher kann man auch allgemein vom $\mathbb{F}_{p^k}$ sprechen und muss nicht sagen, mit welchem Modulpolynom man konkret arbeitet.

Bei AES verwendet man $p = 2, k = 8$, d. h. wir betrachen $\mathbb{F}_{2^8} = \mathbb{F}_{256}$ mit Koeffizienten aus $\mathbb{Z}_2$, und das Modulpolynom

$$m(x) = x^8 + x^4 + x^3 + x + 1$$

Beispiel 4.6
Gegeben sind die beiden Polynome $p, q \in \mathbb{F}_{2^8}$ mit $p(x) = x^6 + x^4 + x^2 + x + 1$ und $q(x) = x^7 + x + 1$ und gesucht ist das Produkt $p \odot q$ gemäß obiger Definition.

Zuerst die übliche, komponentenweise Multiplikation von Polynomen:

$$\begin{aligned} &(x^6 + x^4 + x^2 + x + 1) \odot (x^7 + x + 1) \\ &\qquad = x^{13} + x^{11} + x^9 + x^8 + x^7 \\ &\qquad + \qquad\qquad + x^7 \qquad + x^5 \qquad + x^3 + x^2 + x \\ &\qquad + \qquad\qquad\qquad + x^6 \qquad + x^4 \qquad + x^2 + x + 1 \\ &\qquad = x^{13} + x^{11} + x^9 + x^8 \qquad + x^6 + x^5 + x^4 + x^3 \qquad + 1 \end{aligned}$$

(Hinweis: Die Koeffizienten x^7, x^2 und x entfallen, da $1+1 \pmod 2 \equiv 0$ ist.)
Jetzt muss dieses Polynom noch reduziert werden:

$$\begin{array}{rl} (x^{13} & +x^{11}+x^9+x^8+x^6+x^5+x^4+x^3+1\ ) : (x^8+x^4+x^3+x+1) = \underbrace{x^5+x^3}_{=q(x)} \\ \oplus\ (x^{13} & \quad +x^9+x^8+x^6+x^5 \qquad\qquad\ \ ) \\ \hline & x^{11} \qquad\qquad\quad +x^4+x^3+1 \\ & \oplus\ (x^{11}+x^7+x^6 \qquad\quad +x^4+x^3 \qquad) \\ \hline & x^7+x^6 \qquad\qquad\quad +1 \quad = r(x) \end{array}$$

Also ist

$$(x^6+x^4+x^2+x+1) \odot (x^7+x+1) = x^7+x^6+1$$

Da es für die Multiplikation nur 65 536 verschiedene Kombinationen gibt, kann man diese auch vorher berechnen und in einer 256×256-Tabelle abspeichern.

4.11.4. Darstellung der Elemente aus $\mathbb{F}_{256}$

1. Darstellung als Polynom

   $$p(x) = b_7x^7 + b_6x^6 + \cdots + b_1x + b_0$$

   mit $b_7, \ldots, b_0 \in \{0,1\}$.

2. Darstellung als eine Folge von Bits $(b_7, b_6, \ldots, b_0)$, wobei $b_7, \ldots, b_0$ die Koeffizienten aus der Polynomdarstellung sind.

   Für die Zahlen in Beispiel 4.6 lautet die Darstellung

   $$(0,1,0,1,0,1,1,1) \odot (1,0,0,0,0,0,1,1) = (1,1,0,0,0,0,0,1)$$

3. Darstellung als zweistellige Hexadezimalzahl der Bitfolgen

   In unserem Beispiel 4.6: $57 \odot 83 = C1$

4.11.5. Der erweiterte euklidische Algorithmus

Mit dem **erweiterten euklidischen Algorithmus** bestimmt man zusätzlich zum größten gemeinsamen Teiler zweier natürlicher Zahlen a und b zwei ganze Zahlen u und v, die folgende Gleichung erfüllen

$$ggT(a,b) = u \cdot a + v \cdot b \tag{4.6}$$

Ablauf des Algorithmus':

1. Setze $(x_0, y_0, z_0) := (1,0,a)$, $(x_1, y_1, z_y) := (0,1,b)$ und $i = 1$.
2. Falls $z_i = 0$ wird abgebrochen. Andernfalls zum nächsten Schritt.

3. Für $z_i \neq 0$ erhalten wir durch Division mit Rest die ganzen Zahlen q_i und r_i mit $z_{i-1} = q_i z_i + r_i$.
4. Setze $(x_{i+1}, y_{i+1}, z_{i+1}) \coloneqq (x_{i-1} - q_i x_i, y_{i-1} - q_i y_i, z_{i-1} - q_i z_i)$, erhöhe i um 1 und gehe zu 2.

```
Eingabe: a, b;
begin
    b[0] := a; b[1] := b;
    u[0] := 1; v[0] := 0;
    u[1] := 0; v[1] := 1;
    i := 1;
    while not teilt(b[i], b[i-1]) do
    begin
        q[i] := b[i-1] div b[i];
        b[i+1] := b[i-1] - q[i] * b[i];
        u[i+1] := u[i-1] - q[i] * u[i];
        v[i+1] := v[i-1] - q[i] * v[i];
        i := i + 1;
    end
end
Ausgabe: ggT := b[i], u := u[i], v := v[i];
```

Lemma 4.2
Die oben gemachte Aussage, dass der erweiterte euklidische Algorithmus zwei Zahlen u und v bestimmt, die die Gleichung $ggT(a, b) = u \cdot a + v \cdot b$ erfüllen, ist korrekt.

BEWEIS:
Per Induktion können wir zeigen, dass für alle Schritte i gilt: $b_0 \cdot u_i + b_1 \cdot v_i = b_i$.

Der Induktionsanfang für $i = 1$ ist klar. Also gilt die Aussage für den $(i-1)$-ten und i-ten Schritt. Für den $(i+1)$-ten Schritt folgt dann

$$\begin{aligned} b_0 u_{i+1} + b_1 v_{i+1} &= b_0(u_{i-1} - q_i u_i) + b_1(v_{i-1} - q_i v_i) \\ &= (b_0 u_{i-1} + b_1 v_{i-1}) - q_i(b_0 u_i + b_1 v_i) \\ &= b_{i-1} - q_i b_i = b_{i+1} \end{aligned}$$

■

Beispiel 4.7
Seien $a = 220 = b_0$ und $b = 26 = b_1$ (ganze Zahlen) gegeben.

i	q_i	b_{i+1}	u_{i+1}	v_{i+1}
1	8	$200 - 8 \cdot 26 = 12$	$1 - 8 \cdot 0 = 1$	$0 - 8 \cdot 1 = -8$
2	2	$26 - 2 \cdot 12 = 2$	$0 - 2 \cdot 1 = -2$	$1 - 2 \cdot (-8) = 17$

Abbruch im 3. Schritt, da $2 \mid 12$. Der größte gemeinsame Teiler von 220 und 26 ist also 2 und es gilt:

$$2 = -2 \cdot 220 + 17 \cdot 26 = -440 + 442$$

Beispiel 4.8
Seien $a = 2668 = b_0$ und $b = 157 = b_1$ gegeben.

i	q_i	b_{i+1}	u_{i+1}	v_{i+1}
1	16	$2668 - 17 \cdot 157 = 156$	$1 - 16 \cdot 0 = 1$	$0 - 16 \cdot 1 = -16$
2	1	$157 - 1 \cdot 156 = 1$	$0 - 1 \cdot 1 = -1$	$1 - 1 \cdot (-16) = 17$

Abbruch im 3. Schritt, da $1 \mid 156$. 2668 und 157 sind also teilerfremd und es gilt:

$$1 = -1 \cdot 2668 + 17 \cdot 157 = -2668 + 2669$$

Mit Hilfe des erweiterten euklidischen Algorithmus' lässt sich zu einem Element des $\mathbb{F}_{256}$ das **multiplikative Inverse** bestimmen. Dazu muss man den **erweiterten euklidischen Algorithmus** auf Polynome anwenden, was aber keine Änderungen am Algorithmus selbst erfordert. Man muss nur beachten, dass man für die einzelnen Operationen ($+$, $-$, $\cdot$ und div) die entsprechenden Operationen für den $\mathbb{F}_{256}$ ($\oplus$, $\oplus$[9], $\odot$ und Polynomdivision) verwendet.

Es sei $p(x) \in \mathbb{F}_{256}$ mit $p(x) \neq 0$. Da $m(x)$ irreduzibel ist, sind $p(x)$ und $m(x)$ teilerfremd zu einander, d. h. der größte gemeinsame Teiler von $p(x)$ und $m(x)$ ist 1. Also liefert der erweiterte euklidische Algorithmus nach Gleichung 4.6 zwei Polynome $u(x)$ und $v(x)$ mit der Eigenschaft, dass sich 1 als

$$1 = p(x)u(x) + m(x)v(x)$$

darstellen lässt. Daraus folgt: $p(x) \odot u(x) = 1$, u ist also das Inverse zu p bezüglich $\odot$.

Beispiel 4.9
Gegeben seien $a(x) = m(x) = x^8+x^4+x^3+x+1 = b_0$ und $b(x) = x^7+x^6+x^3+x+1 = b_1$. (Elemente des $\mathbb{F}_{256}$)

1. Schleifendurchlauf $(x^8 + x^4 + x^3 + x + 1) : (x^7 + x^6 + x^3 + x + 1) = x + 1$
$\oplus(x^8 + x^7 + x^4 + x^2 + x)$
—
$x^7 + x^3 + x^2 + 1$
$\oplus(x^7 + x^6 + x^3 + x + 1)$
—
$x^6 + x^2 + x$

Es ergibt sich also:

$$\begin{aligned}
q_1(x) &= x + 1 \\
b_2(x) &= b_0(x) - q_1(x) \cdot b_1(x) = x^8 + x^4 + x^3 + x + 1 - (x + 1) \\
&\qquad \cdot (x^7 + x^6 + x^3 + x + 1) = x^6 + x^2 + x \\
u_2(x) &= u_0(x) - q_1(x) \cdot u_1(x) = 1 - (x + 1) \cdot 0 = 1 \\
v_2(x) &= v_0(x) - q_1(x) \cdot v_1(x) = 0 - (x + 1) \cdot 1 = x + 1
\end{aligned}$$

[9] Wenn man sich die Verknüpfungstabellen für $\oplus$, $-$ und `xor` aufschreibt, sieht man, dass alle drei Operationen äquivalent sind.

2. Schleifendurchlauf $(x^7 + x^6 + x^3 + x + 1) : (x^6 + x^2 + x) = x + 1$
$\oplus(x^7 + x^3 + x^2)$
—
$x^6 + x^2 + x + 1$
$\oplus(x^6 + x^2 + x)$
—
1

Die Zwischenergebnisse:

$$\begin{aligned} q_2(x) &= x + 1 \\ b_3(x) &= b_1(x) - q_2(x) \cdot b_2(x) \\ &= (x^7 + x^6 + x^3 + x + 1) - (x + 1) \cdot (x^6 + x^2 + x) = 1 \\ u_3(x) &= u_1(x) - q_2(x) \cdot u_2(x) = 0 - (x + 1) \cdot 1 = x + 1 \\ v_3(x) &= v_1(x) - q_2(x) \cdot v_2(x) = 1 - (x + 1) \cdot (x + 1) = x^2 \end{aligned}$$

Es gilt also nach Gleichung 4.6

$$\begin{aligned} u(x) \cdot a(x) + v(x) \cdot b(x) &= (x + 1)(x^8 + x^4 + x^3 + x + 1) + x^2(x^7 + x^6 + x^3 + x + 1) \\ &= x^9 + x^5 + x^4 + x^2 + x + x^8 + x^4 + x^3 + x + 1 \\ &\quad + x^9 + x^8 + x^5 + x^3 + x^2 = 1 \end{aligned}$$

woraus folgt, dass $v(x)a(x) = u(x)m(x) + 1$. Also ist $v(x) \odot a(x) = 1$ und somit ist v das Inverse bzgl. $\odot$ zu a: $v(x) = a^{-1}(x)$.

Oder als Bitfolgen geschrieben: $a(x) = (1100\,1011)$ und $a^{-1}(x) = (0000\,0100)$

4.11.6. Polynome über dem Körper $\mathbb{F}_{256}$

Wir betrachten den Polynomring $\mathbb{F}_{256}[x]/(x^4+1)$. Die Elemente sind Polynome höchstens dritten Grades.

$$c(x) = c_3x^3 + c_2x^2 + c_1x + c_0$$

dabei sind die Koeffizienten $c_i \in \mathbb{F}_{256}$ und erlauben die folgende Darstellung:

Beispiel 4.10

1. als Polynom: $c(x) = 01x^3 + 03x^2 + A1x + 02$
2. Binär: $[0000\,0001, 0000\,0011, 1010\,0001, 0000\,0010]$
3. Hexadezimal: $[01, 03, A1, 02]$

Für das Rechnen mit den Elementen aus $\mathbb{F}_{256}[x]/(x^4+1)$ gilt wieder, dass die Addition $a \oplus b$ komponentenweise mit `xor` erfolgt

$$a(x) \oplus b(x) = (a_3 \oplus b_3)x^3 + (a_2 \oplus b_2)x^2 + (a_1 \oplus b_1)x + (a_0 \oplus b_0)$$

und die Multiplikation $a \otimes b$ in zwei Schritten abläuft

1. zuerst die Multiplikation mit einem Polynom $c(x) = a(x) \odot b(x) = c_6x^6 + \cdots + c_1x^1 + c_0$, wobei folgendes gilt

$$\begin{aligned}
c_6 &= (a_3 \odot b_3) \\
c_5 &= (a_3 \odot b_2) \oplus (a_2 \odot b_3) \\
c_4 &= (a_3 \odot b_1) \oplus (a_2 \odot b_2) \oplus (a_1 \odot b_3) \\
c_3 &= (a_3 \odot b_0) \oplus (a_2 \odot b_1) \oplus (a_1 \odot b_2) \oplus (a_0 \odot b_3) \\
c_2 &= (a_2 \odot b_0) \oplus (a_1 \odot b_1) \oplus (a_0 \odot b_2) \\
c_1 &= (a_1 \odot b_0) \oplus (a_0 \odot b_1) \\
c_0 &= (a_0 \odot b_0)
\end{aligned}$$

2. Anschließend muss noch eine Faktorisierung modulo $(x^4 + 1)$ durchgeführt werden. Entweder führt man die drei Schritte der Polynomdivision $c(x) : (x^4 + 1)$ durch oder man verwendet die Eigenschaft $x^i \pmod{x^4 + 1} = x^{i \pmod 4}$ und erhält so das modulare Produkt

$$\begin{aligned}
d(x) = a(x) \odot b(x) \pmod{x^4 + 1} &= c_6x^6 + \cdots + c_1x^1 + c_0 \pmod{x^4 + 1} \\
&= c_6x^2 + c_5x + c_4 + c_3x^3 + c_2x^2 + c_1x + c_0 \\
&= \underbrace{c_3}_{=d_3} x^3 + \underbrace{(c_6 \oplus c_2)}_{=d_2} x^2 + \underbrace{(c_5 \oplus c_1)}_{=d_1} x + \underbrace{(c_4 \oplus c_0)}_{=d_0}
\end{aligned}$$

mit den Koeffizienten

$$\begin{aligned}
d_3 &= (a_3 \odot b_0) \oplus (a_2 \odot b_1) \oplus (a_1 \odot b_2) \oplus (a_0 \odot b_3) \\
d_2 &= (a_2 \odot b_0) \oplus (a_1 \odot b_1) \oplus (a_0 \odot b_2) \oplus (a_3 \odot b_3) \\
d_1 &= (a_1 \odot b_0) \oplus (a_0 \odot b_1) \oplus (a_3 \odot b_2) \oplus (a_2 \odot b_3) \\
d_0 &= (a_0 \odot b_0) \oplus (a_3 \odot b_1) \oplus (a_2 \odot b_2) \oplus (a_1 \odot b_3)
\end{aligned}$$

Matrixschreibweise:

$$\begin{pmatrix} d_3 \\ d_2 \\ d_1 \\ d_0 \end{pmatrix} = \begin{pmatrix} a_0 & a_1 & a_2 & a_3 \\ a_3 & a_0 & a_1 & a_2 \\ a_2 & a_3 & a_0 & a_1 \\ a_1 & a_2 & a_3 & a_0 \end{pmatrix} \cdot \begin{pmatrix} b_3 \\ b_2 \\ b_1 \\ b_0 \end{pmatrix}$$

Beispiel 4.11

$$\begin{aligned}
[00,00,00,01] \otimes [47, 08, 1F, 2B] &= [47, 08, 1F, 2B] \\
[01,00,00,00] \otimes [2B, 1F, 08, 47] &= [47, 2B, 1F, 08]
\end{aligned}$$

Der Ring $R = \mathbb{F}_{256}[x]/(x^4+1)$ ist kein Körper, d. h. nicht jedes Element $c(x) \in R$ besitzt ein Inverses. AES benutzt

$$c(x) = 03x^3 + 01x^2 + 01x + 02$$

mit dem Inversen $c^{-1}(x) = 0Bx^3 + 0Dx^2 + 09x + 0E$

4.12. Der Advanced Encryption Standard – AES

4.12.1. Historischer Abriss

1997: macht die US-Standardisierungsbehörde eine *öffentliche* Ausschreibung für ein Verfahren als Nachfolger von DES. Forderungen an das Verfahren: öffentlich, lizenzfrei, weltweit verfügbar, schneller als Triple-DES. Vorgaben: Blocklänge min. 128 und Schlüssellänge von 128, 192, 256.

1998: Wurden 15 Verfahren eingereicht. (darunter auch MAGENTA von der deutschen Telekom, das schon in der Präsentation gebrochen wurde)

1999: 5 Kandidaten in der Endrunde: Twofish (Platz 2, weil $\mathbb{F}_{256}$ und nicht $\mathbb{F}_2$), RC6, MARS (IBM, David Coppersmith), Serpent (ähnlich DES, über 32 Runden, Platz 3), Rijndael

2000: Wird Rijndael als Sieger bekannt gegeben.

2002: Rijndael wird als AES (FIPS 197, siehe [25]) veröffentlicht.

Durch die Forderung, dass AES öffentlich sein soll, ist der mathematische Hintergrund für AES bekannt, während z. B. der Grund für die Wahl der S-Boxen bei DES unbekannt ist. – IBM kannte bereits seit den 1970ern die differentielle Kryptoanalyse. – Damit ist auch für jedermann nachprüfbar, ob AES korrekt arbeitet.

4.12.2. Beschreibung des Verfahrens von Rijndael

Das Verfahren von Rijndael ist eine iterierte Blockchiffre, deren Block- und Schlüssellänge unabhängig voneinander 128, 192 oder 256 Bit betragen können. Davon abhängig ergibt sich die Rundenzahl als 10, 12 oder 14 Runden. Das Verfahren ist keine Feistel-Chiffre, weil es eine ganz andere Struktur hat. Die Ein- und Ausgabe jeder Runde wird nicht nur in zwei, sondern min. 16 Blöcke geteilt, die alle auf die gleiche Weise transformiert werden.

Für AES wurden 128 Bit als Blocklänge festgelegt, woraus sich 10 Runden für einen 128 Bit Schlüssel, 12 Runden bei einem 192 Bit Schlüssel und 14 Runden für einen 256 Bit langen Schlüssel ergeben. Aus dem Schlüssel werden $r+1$ Rundenschlüssel von je 128 Bit Länge erzeugt, wobei r die Rundenzahl ist. Jede Runde bis auf die letzte besteht aus vier Schritten (Abbildung 4.8):

1. **AddRoundKey** – Verknüpfung mit dem Rundenschlüssel
2. **SubBytes** – nichtlineare Transformation als Schutz vor differenzieller und linearer Kryptoanalyse
3. **ShiftRows** – Diffusion der Bits innerhalb einer Zeile
4. **MixColumns** – Diffusion der Bits innerhalb einer Spalte

In der letzten Runde wird statt dem MixColumns-Schritt nochmal ein AddRoundKey-Schritt mit dem $(r+1)$-ten Schlüssel vorgenommen.

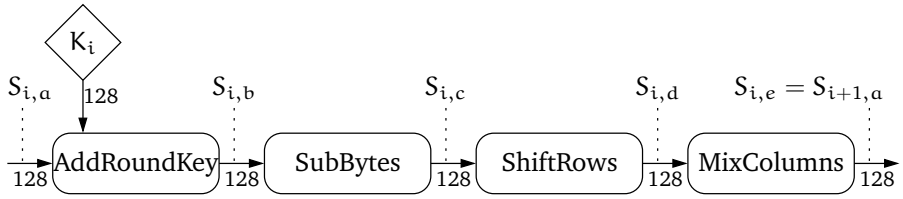

Abbildung 4.8.: Schema für die i-ten Runde des AES

4.12.3. Notationen für AES

Es werden Bytes von je 8 Bit verwendet, die somit als Elemente des Körpers $\mathbb{F}_{256}$ interpretiert werden können; $a = a_7 a_6 \dots a_0 \in \mathbb{F}_{256}$. Ein Eingabeblock S von 128 Bit Länge wird in eine Folge von 16 Bytes zerlegt $S = a^{(0)} a^{(1)} \dots a^{(14)} a^{(15)}$, die auf eine 4×4-Matrix spaltenweise übertragen werden:

$$S = \begin{pmatrix} a^{(0)} & a^{(4)} & a^{(8)} & a^{(12)} \\ a^{(1)} & a^{(5)} & \dots & \\ a^{(2)} & \vdots & \ddots & \\ a^{(3)} & a^{(7)} & a^{(11)} & a^{(15)} \end{pmatrix} = \begin{pmatrix} a^{(0,0)} & a^{(0,1)} & a^{(0,2)} & a^{(0,3)} \\ a^{(1,0)} & a^{(1,1)} & \dots & \\ a^{(2,0)} & \vdots & \ddots & \\ a^{(3,0)} & a^{(3,1)} & a^{(3,2)} & a^{(3,3)} \end{pmatrix} = (a^{z,s})_{4\times 4}$$

Die Eingabe des ersten Schritts (**AddRoundKey**) in der i-ten Runde wird mit $S_{i,a}$ bezeichnet und das Ergebnis des ersten Schritts wird als Eingabe des zweiten Schritts (**SubBytes**) verwendet und mit $S_{i,b}$ bezeichnet

$$S_{i,a} = \begin{pmatrix} a^{(0,0)} & a^{(0,1)} & a^{(0,2)} & a^{(0,3)} \\ \vdots & & & \vdots \\ a^{(3,0)} & \dots\dots\dots\dots & & a^{(3,3)} \end{pmatrix} \qquad S_{i,b} = \begin{pmatrix} b^{(0,0)} & b^{(0,1)} & b^{(0,2)} & b^{(0,3)} \\ \vdots & & & \vdots \\ b^{(3,0)} & \dots\dots\dots\dots & & b^{(3,3)} \end{pmatrix}$$

Das Ergebnis des zweiten Schritts wird als Eingabe des dritten Schritts (**ShiftRows**) verwendet und mit $S_{i,c}$ bezeichnet. Das Ergebnis des dritten Schritts wird als Eingabe für den vierten Schritt (**MixColumns**) verwendet und mit $S_{i,d}$ bezeichnet. Die Ausgabe des vierten Schritts ist gleichzeitig die Ausgabe der i-ten Runde und somit wieder die Eingabe $S_{i+1,a}$ für den ersten Schritt in der $(i+1)$-ten Runde. siehe Abbildung 4.8

Für die Elemente des Rings $\mathbb{F}_{256}[x]/x^4+1$ verwendet man die Polynomdarstellung oder die Hexadezimalschreibweise

$$a(x) = 03x^3 + 01x^2 + 01x + 02 \quad \text{bzw.} \quad a = [03, 01, 01, 02]$$

4.12.4. Beschreibung der einzelnen Schritte

Die Grundlagen aus Abschnitt 4.11 können jetzt dazu verwendet werden, die Arbeitsweise des AES auf mathematisch fundierten Regeln aufzubauen, um z. B. die Invertierbarkeit aller Operationen für den Entschlüsselungsalgorithmus zu sichern. Die Struktur, die durch den $\mathbb{F}_{256}$ repräsentiert wird, ist dabei die Grundlage, denn ihre Elemente können mit den Bytes, den Elementen in die jeder Eingabeblock zerlegt wird, assoziiert werden.

AddRoundKey

Die Verknüpfung mit dem Rundenschlüssel, um die Ausgabe vom Schlüssel abhängig zu machen, ist nicht weiter spektakulär:

$$S_{i,b} = S_{i,a} \oplus K_i$$

SubBytes

Für jedes Byte $b^{(z,s)} \in \mathbb{F}_{256}$ $(z,s \in \{0,\ldots,3\})$ der Eingabe $S_{i,b} = (b^{(z,s)})_{4\times4}$ werden zwei Transformationen ausgeführt:

1. Bestimme das multiplikative Inverse $(b^{(z,s)})^{-1} = \widetilde{b}^{(z,s)}\widetilde{b}_7\widetilde{b}_6\ldots\widetilde{b}_0$ zu $b^{(z,s)}$ in $\mathbb{F}_{256}$. Falls $b^{(z,s)} = 00$, dann ist $\widetilde{b}^{(z,s)} = 00$.

2. Bestimmte Operationen lassen sich mit gewissen Darstellungen einer Struktur besser (einfacher) beschreiben als mit anderen Darstellungen. Daher ist es manchmal günstig die Darstellung zu wechseln. Der Polynomring $\mathbb{F}_2[x]/x^8+1$ ist isomorph zu $\mathbb{F}_{256}$, d. h. beide beschreiben die gleiche Struktur. Man kann also die Elemente des Körpers $\mathbb{F}_{256}$ auch als Elemente des Polynomrings $\mathbb{F}_2[x]/x^8+1$ auffassen, d. h. $\widetilde{b}^{(z,s)} \in \mathbb{F}_2[x]/x^8+1$, und mit diesen weiter rechnen.

   Auf das Inverse $\widetilde{b}^{(z.s)}$ wendet man folgende Abbildung an:

$$c^{(z,s)} = [1,1,1,1,0,0,0,1] \otimes \widetilde{b}^{(z,s)} + [0,1,1,0,0,0,1,1]$$

   Analog zu den Rechenregeln, wie sie im Abschnitt 4.11.6 hergeleitet wurden, ergibt sich folgende affine Abbildung mit dem Ergebnis $c^{(z,s)} = c_7c_6\ldots c_0$

$$\begin{pmatrix} c_0\\c_1\\c_2\\c_3\\c_4\\c_5\\c_6\\c_7 \end{pmatrix} = \begin{pmatrix} 1&0&0&0&1&1&1&1\\ 1&1&0&0&0&1&1&1\\ 1&1&1&0&0&0&1&1\\ 1&1&1&1&0&0&0&1\\ 1&1&1&1&1&0&0&0\\ 0&1&1&1&1&1&0&0\\ 0&0&1&1&1&1&1&0\\ 0&0&0&1&1&1&1&1 \end{pmatrix} \cdot \begin{pmatrix} \widetilde{b}_0\\ \widetilde{b}_1\\ \widetilde{b}_2\\ \widetilde{b}_3\\ \widetilde{b}_4\\ \widetilde{b}_5\\ \widetilde{b}_6\\ \widetilde{b}_7 \end{pmatrix} + \begin{pmatrix} 1\\1\\0\\0\\0\\1\\1\\0 \end{pmatrix}$$

Die praktische Umsetzung der SubBytes-Transformation erfolgt durch eine **lookup-table**, d. h. man berechnet einmal für alle möglichen Eingaben die Ergebnisse und speichert sie in einer Tabelle im Programm ab. Das Programm schlägt dann nur noch in der Tabelle das Ergebnis nach und berechnet nichts mehr.

Die Tabelle ist die S-Box[10], die zweidimensional mit 16 Zeilen und 16 Spalten dargestellt wird. Die niederwertigen Bits b_3, b_2, b_1 und b_0 der Eingabe bestimmen die Spalte in der Tabelle und die Bits b_7, b_6, b_5 und b_4 die Zeile. (siehe Tabelle 4.2)

[10] da sie je 8 Bit verarbeitet, wird sie mit S_8 bezeichnet

S_8	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
10	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
20	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
30	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
40	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
50	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
60	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
70	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
80	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
90	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
A0	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
B0	E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
C0	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
D0	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
E0	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
F0	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

Tabelle 4.2.: Die Substitutionsbox S_8 für den SubBytes-Schritt im AES; Quelle: Seite 16 in [25]

Beispiel: $S_8(1011\,0101)$, Zeile 11 und Spalte 5. Tabelle 4.2 liefert D5. $S_8(1011\,0101) = (1101\,0101)$.

$$c^{(z,s)} = S_8(b^{(z,s)})$$

ShiftRows

Rechnen in $\mathbb{F}_{256}[x]/x^4+1$ zeilenweise:

$$\begin{aligned}
[d^{(0,0)}, d^{(0,1)}, d^{(0,2)}, d^{(0,3)}] &= [c^{(0,0)}, c^{(0,1)}, c^{(0,2)}, c^{(0,3)}] \otimes 01x^4 \\
[d^{(1,0)}, d^{(1,1)}, d^{(1,2)}, d^{(1,3)}] &= [c^{(1,0)}, c^{(1,1)}, c^{(1,2)}, c^{(1,3)}] \otimes 01x^3 \\
[d^{(2,0)}, d^{(2,1)}, d^{(2,2)}, d^{(2,3)}] &= [c^{(2,0)}, c^{(2,1)}, c^{(2,2)}, c^{(2,3)}] \otimes 01x^2 \\
[d^{(3,0)}, d^{(3,1)}, d^{(3,2)}, d^{(3,3)}] &= [c^{(3,0)}, c^{(3,1)}, c^{(3,2)}, c^{(3,3)}] \otimes 01x
\end{aligned}$$

Praktisch bedeutet das, dass die Elemente in der k-ten Eingabezeile um k Plätze nach links rotiert werden.

$$\begin{pmatrix} d^{(0,0)} & d^{(0,1)} & d^{(0,2)} & d^{(0,3)} \\ d^{(1,0)} & d^{(1,1)} & d^{(1,2)} & d^{(1,3)} \\ d^{(2,0)} & d^{(2,1)} & d^{(2,2)} & d^{(2,3)} \\ d^{(3,0)} & d^{(3,1)} & d^{(3,2)} & d^{(3,3)} \end{pmatrix} = \begin{pmatrix} c^{(0,0)} & c^{(0,1)} & c^{(0,2)} & c^{(0,3)} \\ c^{(1,1)} & c^{(1,2)} & c^{(1,3)} & c^{(1,0)} \\ c^{(2,2)} & c^{(2,3)} & c^{(2,0)} & c^{(2,1)} \\ c^{(3,3)} & c^{(3,0)} & c^{(3,1)} & c^{(3,2)} \end{pmatrix}$$

MixColumns

Auf die Eingabe $S_{i,d}$ wird spaltenweise die Abbildung S_{32}[11] angewendet.

$$\begin{pmatrix} e^{(0,s)} \\ e^{(1,s)} \\ e^{(2,s)} \\ e^{(3,s)} \end{pmatrix} = S_{32}\left(\begin{pmatrix} d^{(0,s)} \\ d^{(1,s)} \\ d^{(2,s)} \\ d^{(3,s)} \end{pmatrix}\right)$$

Diese entspricht folgender Berechnung im $\mathbb{F}_{256}[x]/x^4+1$:

$$[e^{(0,s)}, e^{(1,s)}, e^{(2,s)}, e^{(3,s)}] = [03, 01, 01, 02] \otimes [d^{(0,s)}, d^{(1,s)}, d^{(2,s)}, d^{(3,s)}]$$

Die Matrixschreibweise für die Operation $\otimes$ (Abschnitt 4.11.6) ergibt

$$\begin{aligned} \begin{pmatrix} e^{(0,s)} \\ e^{(1,s)} \\ e^{(2,s)} \\ e^{(3,s)} \end{pmatrix} &= \begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix} \cdot \begin{pmatrix} d^{(0,s)} \\ d^{(1,s)} \\ d^{(2,s)} \\ d^{(3,s)} \end{pmatrix} \\ &= \begin{pmatrix} 02 \odot (d^{(0,s)} \oplus d^{(1,s)}) \oplus d^{(1,s)} \oplus d^{(2,s)} \oplus d^{(3,s)} \\ 02 \odot (d^{(1,s)} \oplus d^{(2,s)}) \oplus d^{(0,s)} \oplus d^{(2,s)} \oplus d^{(3,s)} \\ 02 \odot (d^{(2,s)} \oplus d^{(3,s)}) \oplus d^{(0,s)} \oplus d^{(1,s)} \oplus d^{(3,s)} \\ 02 \odot (d^{(3,s)} \oplus d^{(0,s)}) \oplus d^{(0,s)} \oplus d^{(1,s)} \oplus d^{(2,s)} \end{pmatrix} \end{aligned} \tag{4.7}$$

Die praktische Umsetzung mit einer lookup-table ist nicht möglich, da für die 2^{32} möglichen Eingabewerte je 32 Bit Speicherplatz benötigt werden, was 127 Gigabit entspricht, was sich praktisch nur schwer umsetzen lässt.

$$\text{xtime}(p_7 p_6 \dots p_0) = \begin{cases} \pi_6\pi_5\pi_4\pi_3\ \pi_2\pi_1\pi_0 0 & : \pi_7 = 0 \\ (\pi_6\pi_5\pi_4\pi_3\ \pi_2\pi_1\pi_0 0) \oplus (0001\ 1011) & : \pi_7 = 1 \end{cases} \tag{4.8}$$

Aber durch eine geschickte Umordnung der Operanden, wie es in Gleichung 4.7 geschehen ist, kann man eine sehr symmetrische Struktur (Abbildung 4.9) erreichen. Die Funktion xtime: $\mathbb{F}_{256} \to \mathbb{F}_{256}$ (Gleichung 4.8) berechnet dabei für ein $p \in \mathbb{F}_{256}$ das Produkt

$$\begin{aligned} 02 \odot p = x \odot p(x) &= x \cdot (\pi_7 x^7 + \dots + \pi_1 x + \pi_0) \pmod{m(x)} \\ &= \pi_7 x^8 + \pi_6 x^7 + \dots + \pi_1 x^2 + \pi_0 x \pmod{x^8 + x^4 + x^3 + x^1 + 1} \end{aligned}$$

Die Polynomdivision ergibt (nach einem Schritt) das Restpolynom

$$\begin{aligned} &\pi_6 x^7 + \pi_5 x^6 + \pi_4 x^5 + (\pi_3 \oplus \pi_7) x^4 + (\pi_2 \oplus \pi_7) x^3 + \pi_1 x^2 + (\pi_0 \oplus \pi_7) x + \pi_7 \\ &\qquad = [\pi_6, \pi_5, \pi_4, (\pi_3 \oplus \pi_7), (\pi_2 \oplus \pi_7), \pi_1, (\pi_0 \oplus \pi_7), \pi_7] \end{aligned}$$

[11] S_{32}, weil die Abbildung 32 Bit verarbeitet

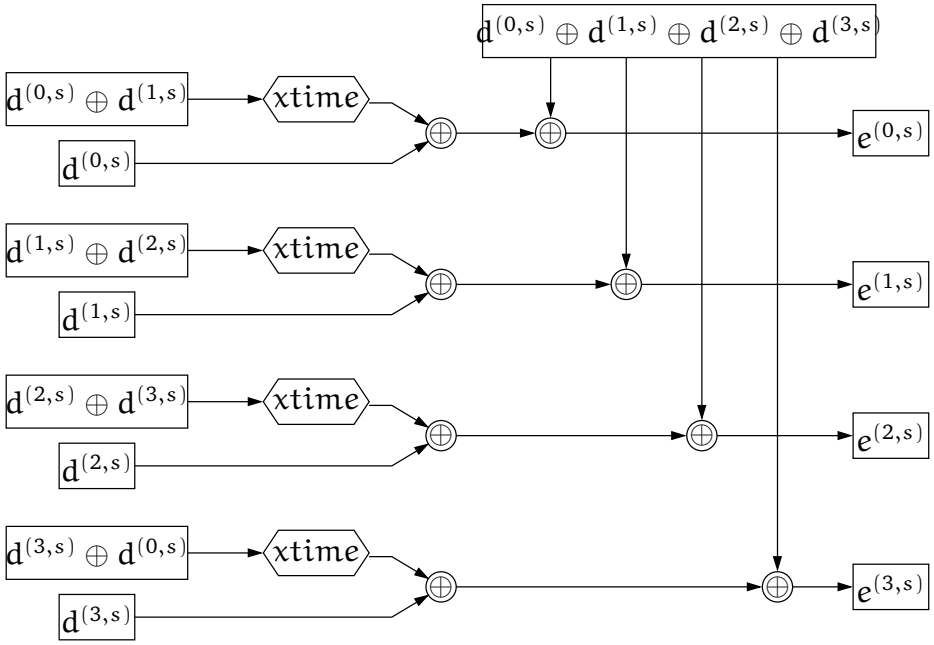

Abbildung 4.9.: Schematische Darstellung der S-Box S_{32} für MixColumns im AES

4.12.5. Struktur des Entschlüsselungsalgorithmus

Da jeder Verschlüsselungsschritt einen Umkehrschritt besitzt ist die Entschlüsselung der Nachricht gesichert:

1. AddRoundKey^{-1} =AddRoundKey ist einfach ein `xor` mit demselben Schlüssel.
2. SubBytes^{-1} existiert, da die affine Abbildung eine Umkehrabbildung besitzt und das Inverse des Inversen das ursprüngliche Element im Körper liefert.
3. ShiftRows^{-1} ist einfach eine Rotation nach rechts.
4. MixColumns^{-1} existiert, da $a(x) = 02 + 01x + 01x^2 + 03x^3$ im Ring $\mathbb{F}_{256}[x]/x^4+1$ das inverse Polynom $a^{-1}(x)$ hat.

Bemerkung 4.4
In der Beschreibung des Algorithmus' fehlt der Key-Schedule. Denn neben den eigentlichen Runden wird pro Runde ein Schlüssel errechnet, der dann mittels AddRoundKey benutzt wird.

5. Public-Key-Kryptosysteme

Die bisher behandelten Verschlüsselungsverfahren basierten alle darauf, dass Alice und Bob über den gleichen Schlüssel (genauer: zwei quasi-gleiche Schlüssel) verfügen. Dieser Schlüssel musste von beiden geheim gehalten werden, was auch zu dem Problem führte, dass beide vorher auf einem sicheren Weg den geheimen Schlüssel verabreden mussten.

Dies ist oft nicht praktikabel, denn Schlüssel sollten auch nicht mehrfach verwendet werden und vor jeder Kommunikation ein persönliches Treffen abzuhalten, ist aufwendig.

Bei der **asymmetrischen Verschlüsselung** besitzt Bob, der Empfänger, ein Paar von Schlüsseln. Der **öffentliche Schlüssel** ist für alle Leute zugänglich und wird zum Verschlüsseln von Nachrichten, deren Empfänger Bob ist, verwendet. Der **private Schlüssel** wird von Bob *geheim* gehalten, denn damit kann er Nachrichten, die mit seinem öffentlichen Schlüssel verschlüsselt wurden, entschlüsseln.

Der entscheidende Unterschied zu den symmetrischen Verschlüsselungsverfahren ist der, dass die Kenntnis des öffentlichen Schlüssels keine Rückschlüsse auf den privaten Schlüssel erlaubt. Eine Nachricht, die also mit einem öffentlichen Schlüssel chiffriert wurde, kann nur mit dem dazu passenden geheimen Schlüssel dechiffriert werden.

5.1. Das Problem des Tauschens geheimer Schlüssel

Alice und Bob kommunizieren über einen unsicheren Kanal und wollen daher ihre Nachrichten verschlüsseln. Dabei stehen sie vor dem Problem, dass sie ein Paar geheim zuhaltender Schlüssel über diesen Kanal austauschen müssen. (*engl. secret key agreement problem*)

Im Jahr 1976 stellten DIFFIE und HELLMAN in ihrer Arbeit „New directions in cryptography“ (siehe [12]) ein Verfahren für dieses augenscheinlich unlösbare Problem vor.

Definition 5.1
Eine Zahl a mit $1 \leqslant a < n$ heißt genau dann **primitive Wurzel** von n, wenn a die folgenden Eigenschaften erfüllt

- a und n sind teilerfremd ($ggT(a, n) = 1$) und
- der kleinste Exponent d, der $a^d \equiv 1 \pmod{n}$ erfüllt, ist $\varphi(n)$.

Man bezeichnet d als **Ordnung** von a.

Beispiel 5.1
Für $n = 5$ kommen als primitive Wurzeln die Zahlen $\mathbb{Z}_5^* = \{1,2,3,4\}$ in Frage. Davon entfallen 1 (aus offensichtlichen Gründen) und 4, da $4^2 \equiv 1 \pmod{5}$. Es bleiben also 2 und 3 als primitive Wurzeln von 5.

Für $n = 11$ gibt es die Kandidaten $\mathbb{Z}_{11}^* = \{1,2,\ldots,10\}$, von denen nur 2, 6, 7 und 8 primitive Wurzeln sind.

Für $n = 6$ sind $\mathbb{Z}_6^* = \{1,5\}$ mögliche primitive Wurzeln, von denen nur 5 die Bedingung $5^d \not\equiv 1 \pmod 6$ für alle $d = 1 < \varphi(6) = \varphi(3 \cdot 2) = 1 \cdot 2$ erfüllt.

a	a^2	a^3	a^4
2	4	3	1
3	4	2	1
4	1	4	1

a	a^2	a^3	a^4	a^5	a^6	a^7	a^8	a^9	a^{10}
2	4	8	5	10	9	7	3	6	1
3	9	5	4	1	3	9	5	4	1
4	5	9	3	1	4	5	9	3	1
5	3	4	9	1	5	3	4	9	1
6	3	7	9	10	5	8	4	2	1
7	5	2	3	10	4	6	9	8	1
8	9	6	4	10	3	2	5	7	1
9	4	3	5	1	9	4	3	5	1
10	1	10	1	10	1	10	1	10	1

Satz 5.1
CARL FRIEDRICH GAUSS hat gezeigt, dass

- n genau dann primitive Wurzeln besitzt, wenn n entweder $1, 2, 4$ ist oder n einfache oder doppelte Potenz einer Primzahl $p > 2$ ist ($n = p^k$ oder $n = 2 \cdot p^k$).
- Falls n primitive Wurzeln besitzt, dann sind es $\varphi(\varphi(n))$ Stück.

Bemerkung 5.1
- 8 und 12 sind die kleinsten Zahlen, die keine primitiven Wurzeln haben
- $n = 5$ hat $\varphi(\varphi(5)) = \varphi(4) = 2$ primitive Wurzeln.

Lemma 5.1
Damit lässt sich der **diskrete Logarithmus** $f_{a,n}^{-1}$ exakt als Funktion definieren, falls a eine primitive Wurzel von n ist.

$$f_{a,n}^{-1}(y) := \log_a y \pmod n$$

Dabei ist der Wert $\log_a x$ eindeutig als diejenige Zahl m bestimmt, für die $a^m \pmod n = y$ gilt.

5.1.1. Das Protokoll von Diffie und Hellman

Alice und Bob verabreden eine große Primzahl p und eine primitive Wurzel g, die beide öffentlich bekannt sein können. Danach wählt jeder für sich eine zufällige, große Zahl a bzw. b. Alice berechnet daraus $\alpha = g^a \pmod p$ und Bob berechnet $\beta = g^b \pmod p$. Diese beiden Ergebnisse tauschen dann beide wieder aus, wobei die Zahlen öffentlich

Schritt	Alice	Kanal	Bob
1.	Beide einigen sich auf eine große Primzahl p und eine primitive Wurzel g; g und p sind öffentlich		
2.	Wählt zufällig eine große Zahl a, berechnet $\alpha = g^a \pmod p$; a ist privat		Wählt zufällig eine große Zahl b, berechnet $\beta = g^b \pmod p$; b ist privat
3.		$\xleftarrow{\beta}\ \xrightarrow{\alpha}$	
4.	$K = \beta^a \pmod p$		$K = \alpha^b \pmod p$

Tabelle 5.1.: Austausch eines geheimen Schlüssels in der Öffentlichkeit nach Diffie und Hellman

Schritt	Alice	Kanal	Bob
1.	Wählt zufällig 2 große Zahlen x, y, berechnet $x \star y$; x privat, $y, x \star y$ öffentlich		
2.		$\xrightarrow{y, x\star y}$	
3.			Wählt zufällig eine große Zahl z, berechnet $y \star z$; z privat, $y \star z$ öffentlich
4.		$\xleftarrow{y\star z}$	
5.	Berechnet $K_A = x \star (y \star z)$		Berechnet $K_B = (x \star y) \star z$

Tabelle 5.2.: Austausch eines geheimen Schlüssels in der Öffentlichkeit nach Rivest und Sherman

zugänglich seien können. Jeder berechnet dann für sich das gemeinsame Geheimnis k. Wie Tabelle 5.1 zeigt, berechnet Alice $k_A = \beta^a \pmod p$ und Bob $k_B = \alpha^b \pmod p$.

$$k_A \equiv \beta^a \equiv (g^b)^a \equiv (g^a)^b = \alpha^b = k_B \pmod p$$

Aus den öffentlichen Größen α, β, p und g kann man nicht das Geheimnis k ermitteln, weil man dafür eine der beiden geheimen Zahlen a oder b benötigt, um α^b bzw. β^a berechnen zu können. Könnte man eine der Zahlen a oder b ermitteln, so wäre dies ein Weg, den diskreten Logarithmus $a = \log_g \alpha \pmod p$ bzw. $b = \log_g \beta \pmod p$ zu berechnen, was nach Abschnitt 5.2.1 (in Polynomialzeit) nicht möglich ist.

5.1.2. Das Protokoll von Rivest und Sherman

Alice wählt zufällig zwei große Zahlen x und y und berechnet daraus $x \star y$, wobei $\star$ eine zweistellige und assoziative Operation ist, die stark „nicht invertierbar“ ist, d. h. die Kenntnis von $x \star y$ und eines Operanden x oder y erlaubt (in Polynomialzeit) keine

Rückschlüsse auf den anderen Operanden. Die Zahl x bleibt privat und die Zahlen y und $x \star y$ übermittelt sie an Bob, wobei die Zahlen öffentlich zugänglich seien können. Bob wählt sich dann zufällig eine große Zahl z, die privat bleibt, und berechnet $y \star z$. Dieses Ergebnis teilt er Alice mit und auch hier kommt es nicht auf die Geheimhaltung an. Somit haben beide jetzt das gemeinsame Geheimnis $k = x \star y \star z$. Die Tabelle 5.2 zeigt schematisch den Ablauf.

$$k_A = x \star (y \star z) = (x \star y) \star z = k_B$$

Damit Eve (der mitlauschende Angreifer) an das Geheimnis k gelangt, muss sie entweder an die Zahl x oder z gelangen. Da aber eine der Forderungen an die Operation $\star$ ist, dass man mit der Kenntnis von $x \star y$ und y nicht auf x schließen kann, ist es für Eve nicht möglich, x aus $x \star y$ bzw. z aus $y \star z$ zu berechnen. Würde Eve also an das Geheimnis kommen, würde $\star$ nicht die gestellten Anforderungen erfüllen.

5.2. Das Konzept der Einwegfunktion

Die Frage ist nun, ob es denn überhaupt eine solche Operation $\star$ gibt und wenn ja, wie sieht diese aus? In der Komplexitätstheorie gibt es das Konzept der Einwegfunktion, das Funktionen beschreibt, die einfach zu berechnen, aber „schwer“ zu invertieren sind.

Definition 5.2
Eine Funktion $f\colon \mathbb{N} \to \mathbb{N}$ ist genau dann eine **Einwegfunktion**, wenn sie die folgenden Eigenschaften besitzt

- f ist im worst case in Polynomialzeit zu berechnen, d. h. $f \in$ FP und
- für fast alle $y \in W_f$ gibt es keinen probabilistischen Algorithmus, der im average case ein zugehöriges Urbild x mit $f(x) = y$ in Polynomialzeit berechnet, d. h. $f^{-1} \notin$ FP.

Bemerkung 5.2
Aus der Komplexitätstheorie ist bekannt, dass eine solche Einwegfunktion nur dann existiert, wenn P $\neq$ NP ist.

5.2.1. Modulare Exponentiation mit fester Basis und festem Modul

Im Folgenden sei $f_{a,n}\colon \mathbb{Z}_n \to \mathbb{Z}_n$ wie folgt definiert:

$$f_{a,n}(m) := a^m \pmod{n}$$

Satz 5.2
Die Zahlenfunktion $f_{a,n}$ (wie sie oben definiert ist) gehört zur Klasse FP der in Polynomialzeit berechenbaren Funktionen.

BEWEIS:
Folgende rekursive Prozedur, **square-and-multiply-Algorithmus** genannt, berechnet $f_{a,n}(m)$.

```
function (a,m,n)
begin
    if m = 0
    then return 1
    else
    begin
        if m gerade
        then return (function(a, m/2, n) ** 2) mod n
        else return (function(a, m-1, n) * a) mod n
    end
end
```

Da in mindestens jedem zweiten Schritt der Exponent m halbiert wird, erfordert dieser Algorithmus zwischen $\log_2 m$ und $2 \cdot \log_2 m$ Durchläufe und hat somit eine Laufzeit von $O(\log_2 m) = O(|m|)$. Da ist einzelnen Operationen mit $O(1)$ angenommen werden, ist dies ein Polynomialzeitalgorithmus. ■

Definition 5.3
Als **diskreten Logarithmus** bezeichnet man die inverse Abbildung $f_{a,n}^{-1}$, die zu einem gegebenen Wert y einen Wert m bestimmt, so dass $a^m \pmod n = y$, falls ein solches m existiert.

Diese Abbildung $f_{a,n}^{-1}$ ist jedoch *keine* Funktion, da $f_{a,n}$ nicht injektiv ist. Zum Beispiel ist das Ergebnis von $f_{5,21}$ für $m_1 = 4$ und $m_2 = 10$ gleich.

$$f_{5,21}(5) = 5^4 \pmod{21} = 16 = 5^{10} \pmod{21} = f_{5,21}(10)$$

Dogma
Wir glauben, dass die Funktion $f_{a,n}$ nicht FP-invertierbar und damit ein Kandidat für eine Einwegfunktion ist.

Könnten wir beweisen, dass $f_{a,n}$ nicht FP-invertierbar ist, dann wäre damit eine echte Einwegfunktion gefunden und man könnte nach Bemerkung 5.2 zeigen, dass P $\neq$ NP ist.

Bemerkung 5.3
Wenn f eine Einwegfunktion ist, dann kann auch der *legitimierte* Empfänger aus dem Kryptogramm $c = f(m)$ die Botschaft m nicht in Polynomialzeit zurückrechnen.

5.2.2. Falltür-Einwegfunktionen

Definition 5.4
Eine Zahlenfunktion $f\colon X \to Y$ heißt genau dann **Falltür-Einwegfunktion** (engl. trap door one way function), wenn sie die folgenden Bedingungen erfüllt:

- Die Funktion ist in Polynomialzeit berechenbar; $f \in$ FP

- f ist mit der **Falltürinformation** in Polynomialzeit umkehrbar;

$$\exists g \in \mathrm{FP}\ \exists\, \text{Falltürinformation } T\ \forall y \in W_f \colon f\big(g(y,T)\big) = y$$

- f ist ohne die Falltürinformation nicht in Polynomialzeit umkehrbar; f ist nicht FP-invertierbar

Diese Definition geht auf DIFFIE und HELLMAN zurück.

Bemerkung 5.4
Eine solche Falltür-Einwegfunktion ist keine Einwegfunktion, denn es gibt einen Polynomialzeitalgorithmus, der die FP-Invertierung ermöglicht.

5.2.3. Modulare Exponentation mit festem Exponenten und festem Modul

Analog zur Definition von $f_{a,n}$ im Abschnitt 5.2.1 sei im Folgenden $g_{n,m} \colon \mathbb{Z}_n \to \mathbb{Z}_n$ so definiert

$$g_{n,m}(a) := a^m \pmod n$$

Satz 5.3
Die oben definierte Funktion $g_{n,m}$ ist in Polynomialzeit berechenbar, d. h. $g_{n,m} \in \mathrm{FP}$.

BEWEIS:
Der square-and-multiply-Algorithmus aus dem Beweis von Satz 5.2 kann ebenfalls zur Berechnung der Funktion $g_{n,m}$ verwendet werden. ∎

Definition 5.5
Die inverse Abbildung $g_{n,m}^{-1} \colon \mathbb{Z}_n \to \mathbb{Z}_n$ bezeichnet man als **m-te Wurzel**. $g_{n,m}^{-1}$ berechnet für ein gegebenes y, das a, so dass $a^m \pmod n = y$ gilt, falls ein solches a existiert.

Hier gibt es das gleiche Problem wie beim diskreten Logarithmus, dass $g_{n,m}$ nicht eindeutig ist, d. h. die m-te Wurzel ist keine Funktion. Beispielsweise ist die 4. Wurzel von 16 für $n = 21$ gleich

$$g_{21,4}(5) = 5^4 \pmod{21} = 16 = 2^4 \pmod{21} = g_{21,4}(2)$$

Doch im Gegensatz zu $f_{a,n}$ lässt sich $g_{n,m}$ in Polynomialzeit invertieren, wenn die Faktorisierung von n als Falltürinformation zur Verfügung steht. (Dies wird später in Satz 5.5 gezeigt.) Dazu ein Exkurs in die Zahlentheorie.

Exkurs in die Zahlentheorie

Ist das gleich der Menge der Einheiten in 4.10.1?

Ansatzpunkt:

$$\mathbb{Z}_n^* := \{\, i \mid 1 \leqslant i < n, \mathrm{ggT}(i,n) = 1 \,\}$$

ist eine multiplikative Gruppe der Ordnung $\varphi(n)$.

Satz 5.4
Von EULER stammt die folgende Erkenntnis: Es gilt für alle x mit $1 \leqslant x < n$ und $ggT(x, n) = 1$ folgende Kongruenz:

$$x^{\varphi(n)} \equiv 1 \pmod{n}$$

oder anders ausgedrückt: ein beliebiges Element einer Gruppe hoch der Ordnung der Gruppe ergibt das Einselement der Gruppe.

Ein Spezialfall ist der kleine Satz von FERMAT: Für alle Primzahlen p und alle x mit $1 \leqslant x < p$ gilt:

$$x^{p-1} \equiv 1 \pmod{p}$$

Lemma 5.2
Der **erweiterte euklidische Algorithmus** lässt sich in Polynomialzeit berechnen; $ggT \in FP$.

BEWEIS:
Der Satz von Lemé (1985) zeigt, dass die Anzahl der Schleifendurchläufe für zwei aufeinander folgende Fibonacci-Zahlen f_{k-1} und f_k am Größten ist. Dann werden höchstens

$$\lfloor \log_{\frac{1}{2}(1+\sqrt{5})}(\sqrt{5}N) \rfloor - 2$$

Schleifendurchläufe benötigt, wobei $N > f_k$ ist. ■

Lemma 5.3
Seien $x, u > 0$. Für alle Primzahlen p und q, die voneinander verschieden sind, gilt:

$$x^u \equiv x \pmod{p} \quad \text{und} \quad x^u \equiv x \pmod{q} \Rightarrow x^u \equiv x \pmod{p \cdot q}$$

BEWEIS:
$x^u \equiv x \pmod{p}$ bedeutet $x^u - x$ ist durch p teilbar und entsprechend $x^u - x$ ist durch q teilbar. Da p und q teilerfremd sind, ist $x^u - x$ durch $p \cdot q$ teilbar. ■

Satz 5.5
Es gibt einen Polynomialzeitalgorithmus, der bei Eingabe von drei natürlichen Zahlen n, m und y, wobei $ggT(m, \varphi(n)) = 1$ und n das Produkt zwei bekannter Primzahlen p und q ist, eine Zahl a mit der Eigenschaft $g_{n,m}(a) = a^m \pmod{n} = y$ bestimmt, d. h. der Algorithmus berechnet die m-te Wurzel $a = g^{-1}_{n,m}(y)$. Die Faktorisierung von n in $p \cdot q$ ist die **Falltürinformation**.

BEWEIS:
Der Algorithmus besteht aus zwei Teilschritten:

1. Da m und $\varphi(n)$ teilerfremd zueinander sind, bestimmt der erweiterte euklidische Algorithmus in Polynomialzeit (Lemma 5.2) zwei Zahlen u und v (Lemma 4.2), so dass $1 = ggT(\varphi(n), m) = u \cdot \varphi(n) + v \cdot m$.

2. Mit dem Inversen v von m in $\mathbb{Z}_{\varphi(n)}$ lässt sich a nach Satz 5.2 mit dem square-and-multiply-Algorithmus in Polynomialzeit durch $a := f_{y,n}(v) = y^v \pmod n$ bestimmen.

Die Behauptung ist nun, dass $y \equiv a^m \pmod n$ gilt. Dazu machen wir die Annahme, dass $y \equiv x^m \pmod n$ für irgendein x ist und zeigen $x = a$.

Aus der Annahme $y \equiv x^m \pmod n$ folgt $y = x^m - s \cdot n$ (für geeignetes s) und mit $a \equiv y^v \pmod n$ ergibt sich

$$a \equiv (x^m - sn)^v \pmod n$$

Mithilfe des binomischen Satz' lässt sich der Klammerausdruck umschreiben zu

$$\equiv \sum_{k=0}^{v} \binom{v}{k} (x^m)^{v-k}(-sn)^k \pmod n$$
$$\equiv x^{m\cdot v} + \sum_{k=1}^{v} \binom{v}{k} x^{m(v-k)}(-s)^k n^k \pmod n$$

Da innerhalb des Summenzeichens Vielfache von n auftreten, entfällt dieser Teil

$$\equiv x^{m\cdot v} \pmod n$$

Da u und v so bestimmt sind, dass $u \cdot \varphi(n) + v \cdot m = 1$ gilt, ergibt sich mit $\tilde{u} = -u$

$$\equiv x^{\tilde{u}\cdot\varphi(n)+1} \pmod n$$

1. Fall Für $x \equiv 0 \pmod p$ gilt die Kongruenz trivialer Weise.

$$a \equiv 0^{\tilde{u}\cdot\varphi(n)+1} \equiv 0 \equiv x \pmod p$$

2. Fall Mit $x \not\equiv 0 \pmod p$ sind die Bedingungen für den kleinen Satz von Fermat (Satz 5.4) erfüllt und es gilt $x^{p-1} \equiv 1 \pmod p$, d. h. $x^{p-1} = r \cdot p + 1$ für ein passendes r. Also ergibt sich für die Kongruenz:

$$a \equiv x^{\tilde{u}\cdot\varphi(n)+1} = x^{\tilde{u}\cdot(p-1)\cdot(q-1)} \cdot x = (x^{p-1})^{\tilde{u}\cdot(q-1)} \cdot x \equiv 1^{\tilde{u}\cdot(q-1)} \cdot x \equiv x \pmod p$$

Für alle x ist also $a \equiv x \pmod p$.

Genauso kann man $a \equiv x \pmod q$ zeigen. Nach Lemma 5.3 gilt also $a \equiv x \pmod n$. ∎

Dogma
Wir glauben, dass die **Faktorisierung** (Zerlegung in Primfaktoren) einer Zahl nicht in Polynomialzeit berechenbar ist. Die Falltürinformation $p \cdot q = n$ ist also nicht FP-invertierbar.

Bemerkung 5.5
Der Algorithmus für Satz 5.5 verwendet im ersten Schritt n und $\varphi(n)$. Ist die Primfaktorzerlegung für n (die Falltürinformation) nicht gegeben, so ist die Berechnung von $\varphi(n)$ genauso schwer wie die Zerlegung von n in Primfaktoren. (Lemma 5.4) Damit ist der Algorithmus laut unserem Dogma ohne die Falltürinformation nicht in Polynomialzeit berechenbar.

Damit ist die Funktion $g_{n,m}$ ein Kandidat für eine Falltür-Einwegfunktion.

Sollte man das mit der vorherigen Bemerkung zusammenlegen?

Lemma 5.4
Die Berechnung von $\varphi(n)$, ohne Kenntnis der Primfaktorzerlegung von n, ist genauso schwer wie die **Faktorisierung** von n, d. h. die Zerlegung von n in ihre Primfaktoren.

BEWEIS:

„⇒“ Falls die Faktorisierung $n = p \cdot q$ bekannt ist, lässt sich $\varphi(n)$ leicht berechnen, da φ multiplikativ ist. (Abschnitt 4.3)

$$\varphi(n) = \varphi(p \cdot q) = \varphi(p) \cdot \varphi(q) = (p-1) \cdot (q-1)$$

Es gilt also: φ-Funktion $\leqslant_m^p$ Faktorisierung.

„⇐“ Falls $\varphi(n)$ bekannt ist (und n das Produkt zweier Primzahlen ist), dann sind p und q, so dass $n = pq$, leicht berechenbar.

$$\varphi(n) = (p-1) \cdot (q-1) = pq - p - q + 1 = n - (p+q) + 1$$

Also lässt sich die Summe der beiden Primzahlen anhand der gegebenen Größen bestimmen:

$$p + q = n - \varphi(n) + 1$$

Die Differenz $p - q$ lässt sich aus den gegebenen Größen auf die folgenden Weise bestimmen:

$$(p-q)^2 = p^2 - 2pq + q^2 = p^2 + 2pq + q^2 - 4pq = (p+q)^2 - 4n$$
$$p - q = \sqrt{(p+q)^2 - 4n}$$

Die gesamte Berechnung erfordert $O(1)$ viele Schritte, womit beide Probleme gleich schwer sind. Es gilt also: Faktorisierung $\leqslant_m^p$ φ-Funktion. ■

Beispiel 5.2
Gegen seien $n = 943 = 23 \cdot 41$ und $\varphi(n) = 22 \cdot 40 = 880$. Damit ergibt sich $p + q = 943 - 880 + 1 = 64$ und $p - q = \sqrt{64^2 - 4 \cdot 943} = 18$. Aus $(p+q) + (p-q)$ ergibt sich wie erwartet so $p = 41$ und aus $(p+q) - (p-q)$ ergibt sich $q = 23$.

Wir haben jetzt zwei Funktionen, von denen wir annehmen, dass sie nicht FP-invertierbar sind: der diskrete Logarithmus und die Faktorisierung von Zahlen. Die Public-Key-Verfahren beruhen auf diesen Funktionen, wobei RSA auf der Faktorisierung von Zahlen und Elgamal auf dem diskreten Logarithmus basiert.

Schritt	Alice	Kanal	Bob
1.			Wählt große Primzahlen p, q, bestimmt $n = p \cdot q$
2.			Berechnet $\varphi(n) = (p-1)(q-1)$.
3.			Wählt große Zahl e mit $ggT(e, \varphi(n)) = 1$
4.			Berechnet Zahl d mit $ed \equiv 1 \pmod{\varphi(n)}$
5.		$\xleftarrow{n,e}$	
6.	Chiffriert die Botschaft m durch $c = m^e \pmod n$		
7.		$\xrightarrow{c}$	
8.			Dechiffriert $m = c^d \pmod n$

Tabelle 5.3.: Das RSA-Verfahren

5.3. Das RSA-Verfahren

Das **RSA-Verfahren** wurde 1978 von RON RIVEST, ADI SHAMIR und LEONARD ADLEMAN entwickelt.

Damit Bob verschlüsselte Nachrichten empfangen kann, muss er folgende Schritte durchführen:

1. Wähle zwei Primzahlen p und q und bilde das Produkt $n = p \cdot q$.
2. Berechne $\varphi(n) = \varphi(p \cdot q) = \varphi(p) \cdot \varphi(q) = (p-1)(q-1)$.
3. Wähle eine natürliche Zahl e mit $1 < e < \varphi(n)$ und $ggT(e, \varphi(n)) = 1$.
4. Berechne $ed \equiv 1 \pmod{\varphi(n)}$ mit Hilfe des erweiterten euklidischen Algorithmus' (siehe Abschnitt 4.11.5).

Das Paar (n, e) ist der **öffentliche Schlüssel**, d ist der **geheime** oder **private Schlüssel** von Bob. Die anderen Größen werden im Folgenden nicht mehr benötigt.

Wenn Alice ihm eine Botschaft $m \leqslant n-1$ zukommen lassen will, muss sie diese als $c = m^e \pmod n$ verschlüsseln und das Kryptogramm c an Bob senden. Bob kann daraus die Botschaft mit $m = c^d \pmod n$ berechnen. Das Verfahren ist in Tabelle 5.3 dargestellt. Diese beiden Berechnungen sind nach Satz 5.2 in Polynomialzeit durchführbar.

Der Verschlüsselungsschritt entspricht der Berechnung der Funktion $g_{n,e}(m) = c$ aus Abschnitt 5.2.3 und der Entschlüsselungsschritt ist der Algorithmus aus dem Beweis von Satz 5.5, wobei $m = f_{c,n}(d)$ ist. Das RSA-Verfahren ist also so konstruiert, dass

$$m \equiv c^d \equiv (m^e)^d \equiv m^{ed} \equiv m^{k \cdot \varphi(n)+1} \equiv m^1 \pmod n$$

Bemerkung 5.6

- RSA ist etwa 100 mal langsamer als AES. Daher verwendet man **Hybridverfahren**: Die Daten werden mit einem symmetrischen Blockalgorithmus verschlüsselt. Der Schlüssel für den symmetrischen Algorithmus wird dann per RSA verschlüsselt und ausgetauscht.
- Die Primzahlen p und q sollten in der Größenordnung von 10^{100} oder größer liegen, damit man n nicht leicht faktorisieren kann. Es gibt unterschiedliche, probabilistische Verfahren, um eine Zahl auf prim zu testen; z. B. von SOLOVAY und STRASSEN – ca. 170 Zahlen testen – oder von RABIN und MILLER – ca. 100 Zahlen testen.
- Die Zahl d für den privaten Schlüssel sollte möglichst groß sein, d. h. etwa in der Nähe von n liegen. Wenn sie nur ein Viertel der Bitgröße von n oder kleiner ist, gibt es einen effizienten Algorithmus zur Berechnung von d. (siehe [26] und [4], Seite 313)
- Die Zahl e kann klein gewählt werden. Dadurch wird der Algorithmus effizienter. Eine ungünstige Wahl von e ermöglicht Angriffe. Dies kann durch Anhängen eines zufälligen Strings an den Klartext (**Salting**) umgangen werden. In der Praxis wird oft $e = 3$ oder $e = 2^{16} + 1 = 65537$ gewählt. Im ersten Fall muss einmal modular quadriert und multipliziert werden, im zweiten Fall sechszehnmal quadriert und einmal multipliziert werden. Die Verschlüsselungsoperation ist in beiden Fällen sehr schnell.
- c und m sind nach Satz 5.2 ebenfalls in Polynomialzeit berechenbar.
- Die Zahlen p und q mit $p \cdot q > 10^{200}$ sollten um einige Dezimalstellen differieren. Falls $p - q$ klein ist, gilt $p \approx \sqrt{n}$ und kann evtl. leicht ermittelt werden.
- $p - 1$ und $q - 1$ sollten große Primfaktoren besitzen und $p - 1$ und $q - 1$ sollten möglichst teilerfremd sein, d. h. $\mathrm{ggT}(p - 1, q - 1)$ sollte möglichst klein sein. siehe Abschnitt 5.3.1
- Ein direkter Angriff würde bedeuten, da $c = m^e \pmod n$ ist, aus c die e-te Wurzel modulo n zu berechnen. Dies ist nach Bemerkung 5.5 praktisch nicht machbar.
- Die Nachricht muss in Blöcke passender Länge zerlegt werden.

Beispiel 5.3

Bob wählt die zwei Primzahlen $p = 47$ und $q = 59$. Dann sind $n = pq = 2773$ und $\varphi(n) = (p-1)(q-1) = 46 \cdot 58 = 2668$. Als seinen geheimen Schlüssel $d \in \mathbb{Z}^*_{\varphi(n)}$ wählt er $d = 157$. Laut Beispiel 4.8 ist das Inverse davon $e = 17$. Als Blocklänge ergibt sich so 2, da bei 26 Buchstaben die Kodierung für „zz“ 2626 $\leqslant$ 2773 ist.

Alice möchte Bob die Nachricht „`dieser satz ist geheim`“ schicken. Diese kodiert sie durch ihre Position im Alphabet, wobei die Leerzeichen mit 00 kodiert werden.

0409 0519 0518 0019 0120 2600 0919 2000 0705 0805 0913

Man kann die Berechnung auch auf Quadrierungen und Multiplikationen zurückführen, wenn man e als $e = 17 = 2 \cdot 2 \cdot 2 \cdot 2 + 1$ schreibt. Der erste Block $m_1^2 = 920$ wird dann als

$$409^{17} \pmod{2773} = \left(\left((409^2)^2\right)^2\right)^2 \cdot 490 \pmod{2773} = 2510$$

verschlüsselt. Damit die Operanden nicht zu groß werden, kann man nach jedem Quadrieren bzw. Multiplizieren modulo 2773 rechnen. Insgesamt ergibt sich als Geheimtext

$$2510\ 1493\ 1787\ 1436\ 0505\ 0499\ 2244\ 0317\ 1692\ 0542\ 0180$$

Die Entschlüsselung erfolgt analog mit $d = 157$ für den letzten Block wie folgt

$$\begin{aligned} 180^{157} \pmod{2773} &= \left(\left(\left(\left(((180^2)^2)^2 \cdot 180\right)^2 \cdot 180\right)^2 \cdot 180\right)^2\right)^2 \cdot 180 \pmod{2773} \\ &= 913 \end{aligned}$$

Bemerkung 5.7
Die Sicherheit von RSA beruht auf zwei Säulen:

- Die Nachricht m lässt sich als die e-te Wurzel aus c berechnen. Die Berechnung der e-te Wurzel ist aber ohne die Kenntnis der Faktorisierung von n nicht in Polynomialzeit durchführbar. (siehe Bemerkung 5.5)
- Die Nachricht m ließe sich auch durch den normalen Entschlüsselungsalgorithmus mit d bestimmen. Die Berechnung von d aus den öffentlichen Größen ist aber nur mit Kenntnis von $\varphi(n)$ möglich, was wiederum nur bestimmt werden kann, wenn die Faktorisierung von n bekannt ist. (siehe Lemma 5.4)

5.3.1. Angriffe auf RSA

Ein Angriff auf das **Faktorisierungsproblem** ist die **Pollard-$(p-1)$-Methode**[1]. Die Idee ist, dass man den Primfaktor p von $n = p \cdot q$ durch $p = \mathrm{ggT}(n, k \cdot p)$ ermitteln kann, sofern n kein Teiler von $k \cdot p$ ist.

Sei ν ein Vielfaches von $p-1$ und a eine beliebige Zahl, die teilerfremd zu n ist. Damit ist a auch teilerfremd zu p und nach dem kleinen Satz von FERMAT (Satz 5.4) gilt dann, dass $a^\nu \equiv (a^{p-1})^l \equiv 1 \pmod{p}$ ist. Also ist $a^\nu - 1$ ein Vielfaches von p.

Als Kandidat für ν kommen Produkte von Primzahlpotenzen, die durch eine Schranke S nach oben begrenzt sind, in Frage.

$$\nu = \prod_{\substack{q \text{ ist prim} \\ q^k \leqslant S}} q^k$$

Falls hier ein q Teiler von $p-1$ dabei ist, dann ist ν ein Vielfaches von $p-1$. Falls $\mathrm{ggT}(n, a^\nu - 1) = 1$ ist, muss man die Schranke vergrößern.

Daher sollen p und q im RSA-Verfahren so gewählt werden, dass die Primfaktoren von $p-1$ und $q-1$ möglichst groß sind.

[1] Nach dem britischen Mathematiker JOHN M. POLLARD

Beispiel 5.4
Die zu faktorisierende Zahl sei $n = 1\,878\,551$. Die kleinste Zahl, die teilerfremd zu n ist, ist $a = 2$.

Der erste Ansatz sei $S_1 = 5$. Damit ist $\nu_1 = 2^2 \cdot 3 \cdot 5 = 60$ das Produkt aller größten Potenzen von Primzahlen, die kleiner sind als S_1. Als $a^{\nu_1} - 1 \pmod n$ ergibt sich so $1\,862\,178$. Leider sind die beiden Zahlen teilerfremd.

Also ein neuer Versuch mit $S_2 = 13$. Damit ist $\nu_2 = 2^3 \cdot 3^2 \cdot 5 \cdot 7 \cdot 11 \cdot 13 = 360\,360$. Damit ergibt sich $a^{\nu_2} - 1 \pmod n = 1\,637\,610$. Der größte gemeinsame Teiler hiervon und n ist 17.

Da n das Produkt zweier Primzahlen ist, ist somit die Faktorisierung von n gebrochen. Die beiden Primfaktoren sind 17 und 110 503.

Eine andere Möglichkeit, die **Faktorisierung** einer Zahl n zu bestimmen, ist das **quadratische Sieb**. Dafür werden Zahlen a und b so ermittelt, dass gilt:

$$a^2 \equiv b^2 \pmod n \qquad \text{und} \qquad a \not\equiv \pm b \pmod n$$

Das heißt n teilt $a^2 - b^2 = (a+b)(a-b)$, aber weder $a+b$ noch $a-b$. Damit ist der größer gemeinsame Teiler von $a^2 - b^2$ und n ein nichttrivialer Faktor von n.

5.3.2. Abwandlungen von RSA

RSA kann nicht als Primzahltest verwendet werden, da es auch mit Modulen funktioniert, die nicht das Produkt zweier Primzahlen sind!

hier stimmt was mit dem Text nicht. Der erste Punkt steht schon oben.

Definition 5.6
Eine **Carmichael-Zahl** c ist das Produkt von Primzahlen, so dass für alle Primfaktoren q von c gilt: $q - 1$ teilt $c - 1$.

- $q - 1$ teilt $c - 1$ und
- für alle zu q teilerfremden Zahlen a gilt $a^{q-1} \equiv 1 \pmod q$. (diese Eigenschaft ist genau dann aus der ersten ableitbar, wenn (wegen der ersten Eigenschaft) $a^{c-1} \equiv 1 \pmod c$ und wegen dem chinesischen Restsatz genau dann, wenn $a^{c-1} \equiv 1 \pmod c$.)

Beispiel 5.5
$561 = 3 \cdot 11 \cdot 17$ ist eine Carmichael-Zahl, denn 2, 10 und 16 sind Teiler von 560. Falls also eine Zahl a teilerfremd zu 3, 11 und 17 ist, dann ist $a^{560} \equiv 1 \pmod{561}$, z. B. $14^{560} \equiv 1 \pmod{561}$, aber z. B. gilt $9^{560} \equiv 375 \pmod{561}$.

Lemma 5.5
Das RSA-Verfahren funktioniert auch mit Modulen n, die nicht das Produkt zweier Primzahlen p und q sind.

BEWEIS:
Wir betrachten folgenden Modul $n = p \cdot c$, wobei $c = \prod q_i$ eine Carmichael-Zahl ist. In dem Fall gilt nicht, dass $\varphi(n) = (p-1)\cdot(c-1)$ ist. Verwendet man jedoch diese Annahme weiter und wählt ein d mit $ggT(d,(p-1)(c-1)) = 1$ und bestimmt dazu ein e mit $e \cdot d \equiv 1 \pmod{(p-1)(c-1)}$, so gilt dennoch $a^{ed} \equiv a \pmod{n}$.

Fall 1: Wir betrachten den Primteiler p von n.

Fall 1 a: Ist p ein Teiler von a, dann ist $a^{ed} \equiv 0 \equiv a \pmod{p}$.

Fall 1 b: Ist p kein Teiler von a, dann sind p und a teilerfremd zu einander, da p prim ist. Es ist also zulässig den kleinen Satz von Fermat (Satz 5.4) anzuwenden. Aus $e\cdot d \equiv 1 \pmod{(p-1)(c-1)}$ folgt $e\cdot d = k(p-1)(c-1)+1$ für geeignetes k und es ergibt sich

$$a^{ed} \equiv a^{k(p-1)(c-1)} \cdot a \equiv (a^{p-1})^{k(c-1)} \cdot a \equiv 1 \cdot a \equiv a \pmod{p}$$

Fall 2: Wir betrachten nun alle Primteiler q_i von n.

Fall 2 a: Wenn q_i ein Teiler von a ist, dann ist $a^{ed} \equiv 0 \equiv a \pmod{q_i}$.

Fall 2 b: Ist q_i kein Teiler von a, so darf wieder der kleine Satz von Fermat angewendet werden. Da c eine Carmichael-Zahl ist, lässt sich $c-1$ auch als $l_i \cdot (q_i - 1)$ schreiben und es gilt

$$a^{ed} \equiv a^{k(p-1)(c-1)} \cdot a \equiv (a^{q_i-1})^{k\cdot l_i\cdot(p-1)} \cdot a \equiv a \pmod{q_i}$$

Es gilt also $a^{ed} \equiv a \pmod{p}$ und $a^{ed} \equiv a \pmod{q_i}$ für alle q_i. Nach dem chinesischen Restsatz folgt daraus $a^{ed} \equiv a \pmod{n}$. ∎

Der Vorteil erschließt sich mir noch nicht. Wenn ich d und e aus dem Ring $\mathbb{Z}_{kgV(p-1,q-1)}$ wähle, verschenke ich doch mögliche Schlüsselpaare und der Gewinn ist im Vergleich zum späteren Aufwand bei der Ver-/Entschlüsselung zu vernachlässigen.

„Optimieren“ lässt sich der RSA-Algorithmus noch im folgendem Sinne:

Lemma 5.6
Es ist für das RSA-Verfahren nicht notwendig, den privaten Schlüssel d so zu bestimmen, dass dieser teilerfremd zu $\varphi(n)$ ist. Es ist ausreichend, wenn dieser teilerfremd zu $kgV(p-1,q-1)$ ist. Der Vorteil zeigt sich bei der Berechnung des öffentlichen Schlüssels e, da hierfür das Modul kleiner geworden ist. Für $p = 11$ und $q = 13$ beispielsweise ist $(p-1)(q-1) = 120$, aber $kgV(p-1,q-1) = 60$.

BEWEIS:
Der Beweis erfolgt analog zum Beweis von Satz 5.5, nur das zu beachten ist, dass d und e Einheiten des $\mathbb{Z}_{kgV(p-1,q-1)}$ sind. Bis zu der Kongruenz

$$a \equiv x^{u\cdot\varphi(n)+1} \pmod{n}$$

verläuft die Rechnung genauso. Für die Fallunterscheidung muss der Fall $x \not\equiv 0 \pmod{p}$ an die veränderte Situation angepasst werden:

Schritt	Alice	Kanal	Bob
1.			Wählt große Primzahl n und dazu eine primitive Wurzel g
2.			Wählt zufällig $b \in \mathbb{Z}_{n-1}^*$ und berechnet $\beta \equiv g^b \pmod{n}$
3.		$\xleftarrow{(n,g,\beta)}$	
4.	Wählt zufällig $a \in \mathbb{Z}_{n-1}^*$ und berechnet $\alpha \equiv g^a \pmod{n}$		
5.	$c \equiv m \cdot \beta^a \pmod{n}$		
5.		$\xrightarrow{c,\alpha}$	
6.			$m \equiv c \cdot \alpha^{n-1-b} \pmod{n}$

Tabelle 5.4.: Ablauf des Elgamal-Verfahrens

Nach dem kleinen Satz von Fermat (Satz 5.4) gilt $x^{p-1} = r \cdot p + 1$ (für passendes r) und das kleinste gemeinsame Vielfache lässt sich schreiben als $\text{kgV}(p-1, q-1) = b \cdot (p-1)$ Also ergibt sich für die Kongruenz $a \equiv x \pmod{p}$ für alle x.

$$a \equiv x^{u \cdot \text{kgV}(p-1,q-1)+1} = x^{u \cdot b \cdot (p-1)} \cdot x = (x^{p-1})^{u \cdot b} \cdot x \equiv 1^{u \cdot b} \cdot x \equiv x \pmod{p}$$

Analog kann man wiederum zeigen, dass $a \equiv x \pmod{q}$ gilt, und nach Lemma 5.3 folgt daraus $a \equiv x \pmod{n}$. ■

So richtig habe ich nicht verstanden warum.

Es ist sinnvoll diese Optimierung in einem hybriden Verfahren anzuwenden.

5.4. Das Verfahren von Elgamal

Das **Elgamal-Verfahren** wurde 1985 von TAHER ELGAMAL entwickelt und beruht auf der Annahme, dass der diskrete Logarithmus (Abschnitt 5.2.1) eine Einwegfunktion ist. Da man für den diskreten Logarithmus keine Falltürinformation wie für die m-te Wurzel benötigt, beruht die Sicherheit von Elgamal im Wesentlichen auf dem Dogma, dass der diskrete Logarithmus praktisch nicht berechenbar ist.

Damit Alice an Bob eine verschlüsselte Nachricht senden kann, muss dieser sich eine große Primzahl n, dazu eine primitive Wurzel g (d. h. n und g sind teilerfremd und für alle $d < n-1$ gilt $g^d \not\equiv 1 \pmod{n}$; Definition 5.1) und eine Zahl $b \in \mathbb{Z}_{n-1}^*$. Aus dieser berechnet er dann $\beta \equiv g^b \pmod{n}$. Das Tupel (n, g, β) bildet den öffentlichen Schlüssel von Bob, die Zahl b ist sein geheimer Schlüssel.

Wenn Alice nun Bob eine verschlüsselte Nachricht $m \leqslant n-1$ schicken will, muss sie sich ebenfalls eine Zahl $a \in \mathbb{Z}_{n-1}^*$ wählen und damit $\alpha \equiv g^a \pmod{n}$ berechnen. Ihre Nachricht m verschlüsselt sie durch $c \equiv m \cdot \beta^a \pmod{n}$ und sendet Bob das Tupel (α, c). Die Zahl a muss geheim bleiben. Beim Empfang kann Bob die Nachricht durch $m \equiv c \cdot \alpha^{n-1-b} \pmod{n}$ entschlüsseln. Tabelle 5.4

Satz 5.6
Der Elgamal-Algorithmus ist korrekt.

BEWEIS:
Es sind sowohl Alice bei der Verschlüsselung als auch Bob bei der Entschlüsselung alle Größen bekannt, die benötigt werden. Außerdem gilt die folgenden Kongruenz:

$$\begin{aligned} m &\equiv c \cdot \alpha^{n-1-b} \equiv m \cdot \beta^a \cdot (g^a)^{n-1-b} \equiv m \cdot g^{ab} \cdot (g^{p-1})^a \cdot g^{-ab} \\ &\equiv m \cdot g^{ab-ab} \cdot 1^a \pmod{n} \end{aligned}$$

■

Bemerkung 5.8
Es gibt zwei Möglichkeiten, an die Botschaft m zu gelangen.

- Entweder man kennt β^a und bestimmt dazu das Inverse γ im $\mathbb{Z}_n$, womit man $m \equiv c \cdot \gamma \pmod{n}$ bestimmen kann. Dafür muss man aber a kennen, wofür man den diskreten Logarithmus $a = \log_g \alpha \pmod{n}$ bestimmen müsste. Dieser ist aber nicht FP-berechenbar. Abschnitt 5.2.1

- Oder man kennt b und kann damit die normale Entschlüsselung vornehmen. An die Kenntnis von b gelangt man aber nur, wenn man den diskreten Logarithmus $b = \log_g \beta \pmod{p}$ bestimmen kann, was wiederum nicht in Polynomialzeit machbar ist.

- Insgesamt lässt sich folgendes feststellen: Wenn man den diskreten Logarithmus in $\mathbb{Z}_n^*$ effizient lösen könnte, so könnte man auch a aus $\alpha \equiv g^a \pmod{n}$ bestimmen und dann m aus der Kongruenz $c \equiv m\beta^a \pmod{n}$ berechnen.

5.5. Weitere Public-Key-Verfahren

Es soll hier nur kurz die Idee des **Rabin-Verfahrens** beschrieben werden: Wähle zwei große Primzahlen p und q mit $p \equiv 3 \pmod 4$ und $q \equiv 3 \pmod 4$, also $\frac{p+1}{4}, \frac{q+1}{4} \in \mathbb{N}$; p, q sind geheim, $n = p \cdot q$ ist öffentlich.

Eine Nachricht $m < n$ wird mit $c = m^2 \pmod{n}$ verschlüsselt. Für die Entschlüsselung werden mit dem erweiterten euklidischen Algorithmus die Zahlen u und v so bestimmt, dass $u \cdot p + v \cdot q = 1$. Dann werden mit den Hilfsgrößen m_p und m_q die vier Quadratwurzeln r_i bestimmt.

$$\begin{aligned} m_p = c^{\frac{p+1}{4}} \pmod{n} \qquad m_q = c^{\frac{q+1}{4}} \pmod{n} \\ r_1 = u \cdot p \cdot m_q + v \cdot q \cdot m_p \pmod{n} \\ r_2 = n - r_1 \\ r_3 = u \cdot p \cdot m_q - v \cdot q \cdot m_p \pmod{n} \\ r_4 = n - r_3 \end{aligned}$$

Schritt	Alice	Kanal	Bob
1.			Wählt große Primzahl n und dazu eine primitive Wurzel g
2.			Wählt zufällig $b \in \mathbb{Z}_{n-1}^*$ und berechnet $\beta \equiv g^b \pmod{n}$
3.			Wählt zufällig $r \in \mathbb{Z}_{n-1}^*$ und berechnet $\rho = g^r \pmod{n}$
4.			Bestimmt $s' \cdot r \equiv 1 \ (n-1)$ und berechnet $s \equiv s'(m - b \cdot \rho) \ (n-1)$; $sig_{Bob}(m) = (\rho, s)$
5.		$\xleftarrow{(n,g,\beta),m,(\rho,s)}$	
6.	Prüft $g^m = \beta^\rho \cdot \rho^s \ (n)$		

Tabelle 5.5.: Das Verfahren von Elgamal zur digitalen Signatur

Das Problem dabei ist, dass die Quadratwurzel nicht eindeutig ist. Man muss sich also noch darüber verständigen, welche der vier möglichen Wurzeln der gemeinte Klartext ist.

Die Sicherheit des Verfahrens beruht darauf, dass Wurzelziehen nur möglich ist, wenn p und q bekannt sind, da die m-te Wurzel nicht FP-berechenbar.

5.6. Digitale Signaturen

Das Anliegen besteht in der **Authentifizierung** des Absenders. Grundidee: Wir nutzen die Sicherheit der Protokolle der Public-Key-Verfahren in modifizierter Form für die Authentifizierung.

5.6.1. Elgamal

Damit Bob an Alice eine Nachricht m mit Echtheitsnachweis senden kann, muss er zusätzlich zum Elgamal-Verschlüsselungsverfahren (Tabelle 5.4) mit dem öffentlichen Schlüssel (n, g, β) sich eine Zahl $r \in \mathbb{Z}_{p-1}^*$ wählen und dafür $\rho = g^r \pmod{n}$ bestimmen. Da $r \in \mathbb{Z}_{p-1}^*$, d. h. insbesondere ist r teilerfremd zu $p-1$, existiert ein s' mit $r \cdot s' \equiv 1 \pmod{n-1}$. Damit bestimmt er dann $s \equiv s'(m - b \cdot \rho) \pmod{n-1}$. Das Paar (ρ, s) bildet die Signatur der Nachricht m. Alice kann diese durch die Kongruenz $g^m \equiv \beta^s \cdot \rho \pmod{n}$ überprüfen. Tabelle 5.5

Wegen $r \cdot s \equiv r \cdot s'(m - b \cdot \rho) \equiv m - b \cdot \rho \pmod{n-1}$ gilt $b \cdot \rho + r \cdot s \pmod{n-1}$ und damit $m = b \cdot \rho + r \cdot s + k \cdot (n-1)$ für geeignetes k und mit Hilfe des kleinen Satz von Fermat (Satz 5.4) ergibt sich

$$g^m = g^{b\cdot\rho + r\cdot s + k\cdot(n-1)} = g^{b\cdot\rho + r\cdot s} \cdot (g^{n-1})^k \equiv g^{b\cdot\rho + r\cdot s} \cdot 1^k = \beta^\rho \cdot \rho^s \pmod{p}$$

Schritt	Alice	Kanal	Bob
1.			Wählt große Primzahlen p, q, bestimmt $n = p \cdot q$
2.			Wählt große Zahl d mit $ggT(d, \varphi(n)) = 1$
3.			Berechnet das Inverse e in $\mathbb{Z}^*_{\varphi(n)}$
4.			Erstellt $sig_{Bob}(m) \equiv m^d \pmod n$
5.		$\xleftarrow{(n,e),m,sig_{Bob}(m)}$	
6.	Prüft $m \equiv (sig_{Bob}(m))^e \pmod n$		

Tabelle 5.6.: RSA-Verfahren für digitale Signaturen

Dies entspricht der Kongruenz, die Alice zur Verifikation prüft:

$$g^m \equiv g^{b\cdot\rho+r\cdot s} = \beta^\rho \cdot \rho^s \pmod p$$

5.6.2. RSA

Die Erstellung einer Signatur mit Hilfe des RSA-Verfahrens Tabelle 5.3 ist recht simpel. Wenn Bob eine Nachricht signieren will, tut er dies, indem er die Nachricht mit seinem geheimen Schlüssel d verschlüsselt: $sig_{Bob}(m) = m^d \pmod n$. Alice kann die Signatur dann prüfen, indem sie mit dem öffentlichen Schlüssel e von Bob die Kongruenz $m \equiv (sig_{Bob}(m))^e \pmod n$ berechnet.

Der Beweis für die Korrektheit erfolgt analog zum Beweis für die Korrektheit von RSA mit

$$sig_{Bob}(m)^e \equiv m^{d\cdot e} \equiv m \pmod n$$

5.7. Das Shamir-ohne-Schlüssel-Protokoll

Analog zum Diffie-Hellman-Schlüsseltausch (Tabelle 5.1) funktioniert das Shamir-ohne-Schlüssel-Protokoll ohne ein gemeinsames Geheimnis von Alice und Bob. Jeder hat ein Geheimnis für sich, aber es gibt kein gemeinsames Geheimnis.

Einer von beiden bestimmt eine große Primzahl p, die er dem anderen mitteilt. Daraufhin wählt jeder sein Geheimnis, als eine Einheit im Ring $\mathbb{Z}^*_{p-1}$. Alice berechnet ein Paar

Schritt	Alice	Kanal	Bob
1.	Wählt große Primzahl p		
2.		$\xrightarrow{p}$	
3.	Wählt Einheit a in $\mathbb{Z}_p^*$		Wählt Einheit b in $\mathbb{Z}_p^*$
4.	Berechnet $x = m^a \pmod p$ für eine Nachricht m		
5.		$\xrightarrow{p}$	
6.			$y = x^b \pmod p$
7.		$\xleftarrow{y}$	
8.	$z = y^{a^{-1}} \pmod p$		
9.		$\xrightarrow{z}$	
10.			$m = z^{b^{-1}} \pmod p$

Tabelle 5.7.: Das Protokoll Shamir-ohne-Schlüssel

(a, a^{-1}) mit der Eigenschaft $a \cdot a^{-1} \equiv 1 \pmod{p-1}$ und entsprechend berechnet Bob ein Paar (b, b^{-1}) mit der Eigenschaft, dass $b \cdot b^{-1} \equiv 1 \pmod{p-1}$ ist.

Alice verschlüsselt die Nachricht m mit ihrem Geheimnis a durch $x \equiv m^a \pmod p$. Dieses x sendet sie an Bob der die Nachricht ein zweites Mal durch $y \equiv x^b \pmod p$ verschlüsselt. Dieses y übergibt er wieder an Alice, die dann ihr „Schloss“ a durch $z \equiv y^{a^{-1}} \pmod p$ wieder entfernt. Daraufhin kann Bob sein „Schloss“ b entfernen und erhält die Nachricht $m \equiv z^{b^{-1}} \pmod p$. Es gilt also

$$\begin{aligned} m &\equiv z^{b^{-1}} \equiv y^{a^{-1} \cdot b^{-1}} \equiv x^{b \cdot a^{-1} \cdot b^{-1}} \equiv m^{a \cdot b \cdot a^{-1} \cdot b^{-1}} \pmod p \\ &\equiv (m^{b \cdot b^{-1}})^{l_a(p-1)+1} \equiv 1^{l_a} \cdot m^{l_b(p-1)+1} \equiv 1^{l_b} \cdot m \pmod p \end{aligned}$$

6. Kryptographische Hashfunktionen

Definition 6.1
Eine **Hashfunktion** ist eine Abbildung der Form $H\colon \Sigma^* \to \Sigma^n$ für ein festes $n \in \mathbb{N}$.

Beispiel 6.1
Die **Parität** ist eine Hashfunktion. Für $\Sigma = \{0,1\}$ ist die Parität für einen Wert $w = w_1 w_2 \dots w_k \in \Sigma^k$ genau dann 1, wenn die Anzahl der Einsen in w ungerade ist.

$$H\colon \Sigma^* \to \Sigma^1, \quad H(w) = w_1 \oplus w_2 \oplus \dots \oplus w_k$$

Genauso ist die **Quersumme** eine Hashfunktion. Für $\Sigma = \{0,1,\dots,9\}$ ist die Quersumme von $w \in \Sigma^k$

$$H(w) = \sum_{i=1}^{k} w_i$$

Wie leicht ersichtlich ist, kann eine Hashfunktion nicht injektiv sein, da sie immer von einer mächtigeren Menge in eine schmächtigere Menge abbildet.

In der Praxis werden Hashfunktionen häufig durch Kompressionsfunktionen erzeugt. Ein **Kompressionsfunktion** ist eine Abbildung der Form $K\colon \Sigma^m \to \Sigma^n$ für $m > n$. Es werden zwei Forderungen an die Funktion gestellt:

- H (und K) müssen schnell berechenbar sein
- $H(w) = H(w')$ für $w \neq w'$ sollte nicht zu häufig auftreten! Eine solche Situation nennt man **Kollision**.

Die beiden Beispiele erfüllen die erste Forderung auf alle Fälle. Jedoch gibt es bei der Parität sehr viele Kollisionen, bei der Quersumme sind es weniger, aber sie treten immer noch häufig auf.

Für kryptographische Hashfunktionen sind diese Forderungen jedoch nicht ausreichend, denn es besteht folgende Angriffsmöglichkeit: Wenn Alice neben der Nachricht m ihren Identitätsnachweis $\mathrm{sig}_{\mathrm{Alice}}(m)$, also insgesamt $(m, \mathrm{sig}_{\mathrm{Alice}}(m))$, schickt, ist die Signatur genauso lang wie die Nachricht, die zu übertragenden Daten verdoppeln sich also. Daher verwendet man den Hashwert einer Nachricht für eine **digitale Signatur**.

Aber nach der obigen Forderung ist der Hashwert $H(m)$ leicht zu berechnen. Mallory, der Angreifer, kann also leicht eine Nachricht m' mit $H(m') = H(m)$ finden und sich dann als Alice ausgeben, denn er besitzt für seine Nachricht m' eine gütige Signatur von Alice

$$(m', \mathrm{sig}_{\mathrm{Alice}}(H(m))) = (m', \mathrm{sig}_{\mathrm{Alice}}(H(m')))$$

Um genau das zu vermeiden, formulieren wir folgende Eigenschaften für eine **kryptographische Hashfunktion** H:

- H ist eine Einwegfunktion.
- H ist (schwach) **kollisionsresistent**; d. h. für gegebenes m gibt es keinen Polynomialzeitalgorithmus, der ein $m' \neq m$ mit der Eigenschaft $H(m') = H(m)$ berechnet.
- H ist (stark) kollisionsresistent; d. h. es gibt keinen Polynomialzeitalgorithmus, der überhaupt eine Kollision berechnet.

Bemerkung 6.1
Ein Invertierungsalgorithmus A für H liefert für ein gegebenes m ein Element $m' = A(H(m))$. Da die Anzahl der Urbilder von $H(m)$ im Mittel größer als eins ist, ist die Wahrscheinlichkeit eine Kollision m' zu berechnen im Mittel größer als $\frac{1}{2}$.

Konsequenz: H ist eine Einwegfunktion, deswegen probabilistische Algorithmen zur Erzeugung von Kollisionen.

Satz 6.1
Aus einer stark kollisionsresistenten Kompressionsfunktion K der Form $K\colon \mathbb{Z}_2^m \to \mathbb{Z}_2^n$ $(m > n)$ lässt sich eine stark kollisionsresistente Hashfunktion $H\colon \mathbb{Z}_2^m \to \Sigma_2^n$ konstruieren. (von Merkle)

Da nicht bewiesen ist, dass es kryptographische Einwegfunktionen gibt, verwendet man nur solche Hashfunktionen, deren Kollisionsresistenz (bisher) nicht widerlegt ist. Gute Kandidaten dafür sind Verschlüsselungsverfahren, die als sicher gelten.

Konkret wird ein Verschlüsselungsverfahren gebraucht, für das die Klartext- und Geheimtextmenge $\mathbb{Z}_2^n$ ist und die Verschlüsselungsfunktion E_k immer wieder auf diese Menge abbildet, wobei der zugehörige Schlüssel $k \in \mathbb{Z}_2^n$ ebenfalls aus der Menge stammt. Damit lässt sich eine **Hashfunktion** $H\colon \mathbb{Z}_2^* \to \mathbb{Z}_2^n$ wie folgt konstruieren:

Es sei $x = x_1 x_2 \dots x_l \in \mathbb{Z}_2^*$ eine Nachricht, die in Blöcke der Länge n zerlegt wird, d. h. $x_i \in \mathbb{Z}_2^n$, – der letzte Block wird einfach bis zur Länge n aufgefüllt.

Der Hashwert einer Nachricht $H(x)$ ist der Wert k_l nach der l-ten Iteration der Vorschrift $k_i = f(x_i, k_{i-1})$ für $i = 1, \dots, l$. Die Funktion $f\colon \mathbb{Z}_2^n \times \mathbb{Z}_2^n \to \mathbb{Z}_2^n$ wird durch die Verschlüsselungsfunktion auf folgende Weise definiert, wobei man vier verschieden Typen unterscheidet:

$$f(x,k) = \begin{cases} E_k(x) \oplus x \\ E_k(x) \oplus x \oplus k \\ E_k(x \oplus k) \oplus x \\ E_k(x \oplus k) \oplus x \oplus k \end{cases}$$

Es bleibt die Frage: „Wie groß soll n sein?“ Es gibt zwei sich widersprechende Bestrebungen: n sollte nicht zu groß (eine Hashfunktion soll komprimieren), aber auch nicht zu klein sein, da es sonst zu viele Kollisionen gibt.

Aus der Stochastik ist das Geburtstagsparadoxon mit dem verblüffenden Resultat bekannt, dass sich unter 23 Personen mit der Wahrscheinlichkeit von $\frac{1}{2}$ mindestens zwei Personen befinden, die am gleichen Tag Geburtstag haben. Dessen Grundlage soll jetzt zur Klärung der Frage genutzt werden.

Satz 6.2
Sei $H\colon \Sigma^k \to \Sigma^n$ eine Hashfunktion, die für jede Nachricht genau einen Hashwert erzeugt, wobei es k verschiedene Nachrichten und n verschiedene Hashwerte gibt.

Die Wahrscheinlichkeit dafür, dass zwei Nachrichten m und m' den gleichen Hashwert $H(m) = H(m')$ haben, ist $\geqslant \frac{1}{2}$, falls

$$k \geqslant \frac{\sqrt{1 + 8 \cdot n \cdot \ln 2} + 1}{2} \approx 1{,}18\sqrt{n}$$

BEWEIS:
Ein Elementarereignis $(g_1, g_2, \ldots, g_k) \in (\Sigma^n)^k$ ist die Zuordnung von k Hashwerten auf k Nachrichten, d. h. Nachricht i bekommt den Hashwert g_i zugeordnet. Da die Zuordnung der Hashwerte gleichverteilt ist, ergibt sich die Wahrscheinlichkeit für ein Elementarereignis als $\left(\frac{1}{n}\right)^k = \frac{1}{n^k}$.

Die Wahrscheinlichkeit p, dass es zwei (oder mehr) Nachrichten i, j gibt, die den gleichen Hashwert $g_i = g_j$ haben, entspricht der Wahrscheinlichkeit $q = 1 - p$ der dualen Aufgabe, dass alle g_i paarweise verschieden sind.

$$q = \frac{\operatorname{card}\{(g_1, \ldots, g_k)\colon \text{alle } g_i \text{ sind verschieden}\}}{n^k} = \frac{1}{n^k} \prod_{i=0}^{k-1} n - i = \prod_{i=0}^{k-1} 1 - \frac{i}{n}$$

Die Faktoren $1 - \frac{i}{n}$ lassen sich nach oben abschätzen durch $e^{-\frac{i}{n}}$, denn $1 + x \leqslant e^x$ für alle $x \in \mathbb{R}$.

$$q \leqslant \prod_{i=0}^{k-1} e^{-\frac{i}{n}} = e^{-\frac{1}{n}\sum_{i=0}^{k-1} i} = e^{-\frac{k(k-1)}{2n}}$$

Für $p \geqslant \frac{1}{2}$ bzw. $q \leqslant \frac{1}{2}$ ergibt sich also

$$\begin{aligned} e^{-\frac{k(k-1)}{2n}} \leqslant \frac{1}{2} &\Leftrightarrow -\frac{k(k-1)}{2n} \leqslant -\ln 2 \\ &\Leftrightarrow k^2 - k - 2n\ln 2 \geqslant 0 \\ &\Leftrightarrow k \geqslant \frac{1}{2} \pm \sqrt{\frac{1}{4} + 2n\ln 2} \end{aligned}$$

∎

Für eine Hashfunktion $H\colon \mathbb{Z}_2^* \to \mathbb{Z}_2^n$ hat die „Geburtstagsangriff“ folgende Konsequenz: Es genügen $\sqrt{2^n}$ Hashwerten, um mit einer Wahrscheinlichkeit $\geqslant \frac{1}{2}$ eine Kollision zu erzeugen. In der Praxis bedeutet dies, dass $n = 64$ zu kurz und damit zu unsicher ist, da nur $2^{32} \approx 4 \cdot 10^9$ Hashwerte für eine Kollision berechnet werden müssen. Daher

werden in der Regel Hashfunktionen mit $n \geqslant 128$ in der Praxis verwendet, bei digitalen Signaturen gilt sogar $n > 168$.

Illustration an einem Beispiel „Ein unehrenhafter Vertrag“: Alice erarbeitet zwei Verträge: Der eine begünstigt Bob, der andere benachteiligt ihn und erzeugt für jeden Vertrag durch marginale Änderungen (z. B. hinzufügen/weglassen von Leerzeichen) 2^{32} Varianten, berechnet für jede Variante den Hashwert und wählt die zwei Fassungen, deren Hashwerte gleich sind. Bob signiert den Hashwert und akzeptiert damit auch den für ihn nachteiligen Vertrag.

In der Praxis haben sich zur Berechnung von Hashfunktionen (neben der Erzeugung mit Hilfe von Verschlüsselungsfunktionen) spezielle Algorithmen etabliert:

MD4 Erfunden von RON RIVEST, erzeugt einen Hashwert von 128 Bit Länge, Kollisionen wurden gefunden und vom Einsatz wird abgeraten.

MD5 Weiterentwicklung von MD4 aus dem Jahr 1991, fünf Jahre später fand der deutsche Kryptologe HANS DOBERTIN eine Kollision zweier speziell präparierter Nachrichten, mittels einer speziellen Angriffstechnik (Preimage-Angriff) können Kollisionen erzeugt werden, von einer Verwendung wird abgeraten.

SHA-Familie von der NSA, SHA-1 hat ebenfalls Schwächen, aber SHA-2 ist derzeit sicher.

Die NIST hat ähnlich wie bei AES einen öffentlichen Wettbewerb für einen neuen, sicheren Hash-Algorithmus ausgeschrieben. Ergebnisse werden für das Jahr 2012 erwartet.

Für die **digitale Signatur** ergibt sich also folgendes Fazit: Alice sendet Bob eine Nachricht und fügt ihre Signatur an: $(m, \mathrm{sig}_{\mathrm{Alice}}(m))$. Bob verifiziert mit dem öffentlichen Schlüssel von Alice die Signatur: $\mathrm{ver}_{\mathrm{Alice}}(\mathrm{sig}_{\mathrm{Alice}}(H(M))) = H(m)$. Mallory wäre dabei nur erfolgreich, wenn er

1. für seine Nachricht m' die Signatur von Alice $\mathrm{sig}_{\mathrm{Alice}}$ berechnen könnte oder
2. zur Nachricht m eine weitere Nachricht m' mit $H(m) = H(m')$ erzeugen könnte.

Aber Punkt 1 ist ihm nicht möglich, weil er aus der Kenntnis von $\mathrm{ver}_{\mathrm{Alice}}$ nicht auf $\mathrm{sig}_{\mathrm{Alice}}$ schließen kann, da $\mathrm{ver}_{\mathrm{Alice}}$ eine Einwegfunktion ist, und Punkt 2 ist für ihn nicht möglich, da H kollisionsresistent ist.

7. Zero-knowledge-Protokolle

Zero-knowledge-Protokolle haben das Anliegen, seinem Gegenüber zu zeigen, dass man im Besitz eines Geheimnisses ist, ohne dieses Geheimnis preiszugeben. Damit ist es Alice möglich, Bob einen **Identifizierungsnachweis** zu liefern, ohne diesen zu verraten und ihn so vor Eve und Mallory zu schützen.

In der Komplexitätstheorie ist das sogenannte Arthur-Merlin-Protokoll[1] bekannt. Dabei versucht der übermächtige Zauberer Merlin den König Arthur von der Richtigkeit seines Beweises zu überzeugen. Da aber König Arthur nicht so gut rechnen kann, wie Merlin[2], prüft er nur Teile des Beweises und akzeptiert, wenn er darin keinen Fehler findet.

An diese Idee angelehnt, versucht Alice, durch die Kenntnis der Lösung für ein schwer zu lösendes Problem (ein Problem in NP), Bob davon zu überzeugen, dass sie die Lösung kennt. Ein mögliches Problem ist das **Graphisomorphieproblem**: Gegeben seien zwei einfache Graphen $G_1 = (V_1, E_2)$ und $G_2 = (V_2, E_2)$. Die Frage ist, ob die Graphen zueinander **isomorph** ($G_1 \cong G_2$) sind, d. h. existiert eine Umbenennung/Permutation π der Kanten in G_1, so dass $G_2 = \pi(G_1)$. Für das Graphisomorphieproblem ist bekannt, dass es in NP liegt. Es ist noch offen, ob es in P liegt oder ob es NP-vollständig ist.

Damit Merlin sich später einmal identifizieren kann, wählt er einen Graphen G_0 und eine Permutation π und berechnet dazu einen isomorphen Graphen G_1. Die Permutation stellt das persönliche Geheimnis von Merlin dar, G_0 und G_1 sind öffentlich.

Wenn Arthur von Merlin eine Identifikation verlangt, wählt sich Merlin einer weiter Permutation ρ und ein Bit $a \in \mathbb{Z}_2$ und berechnet den Graphen $H = \rho(G_a)$. Diesen sendet er an Arthur, der seinerseits ein Bit b wählt und von Merlin einen Isomorphismus σ verlangt, so dass $H = \sigma(G_b)$ ist. Es können nun drei Fälle auftreten:

- $a = b$: Dann gilt $H = \rho(G_a) = \rho(G_b)$, also ist $\sigma = \rho$.
- $a = 1$ und $b = 0$: Dann gilt $H = \rho(G_1) = \rho\big(\pi(G_0)\big)$, also ist $\sigma = \pi \circ \rho$.
- $a = 0$ und $b = 1$: Dann gilt $H = \rho(G_0) = \rho\big(\pi^{-1}(G_1)\big)$, also ist $\sigma = \pi^{-1} \circ \rho$.

Dieses σ sendet Merlin an Arthur, der daraufhin $\sigma(G_b) = H$ prüfen kann. Tabelle 7.1

[1] siehe Eintrag bei der englischen Wikipedia zum Arthur-Merlin-Protokoll

[2] Arthur ist ein Computer mit einer Zufallszahleinheit und Merlin ein Orakel mit unbegrenzter Rechenkraft, welches nicht immer die Wahrheit spricht.

Schritt	Merlin	Kanal	Arthur
1.	Wählt Permutation ρ, $a \in \mathbb{Z}_2$ und berechnet $H = \rho(G_a)$		
2.		$\xrightarrow{H}$	
3.			Wählt $b \in \mathbb{Z}_2$ und verlangt σ, so dass $\sigma(H) = G_b$
4.		$\xleftarrow{b}$	
5.	Berechnet $\sigma = \rho$, falls $a = b$; $\sigma = \pi \circ \rho$, falls $a = 1, b = 0$; $\sigma = \pi^{-1} \circ \rho$, falls $a = 0, b = 1$		
6.		$\xrightarrow{\sigma}$	
7.			Verifiziert $\sigma(G_b) = H$

Tabelle 7.1.: Zero-Knowledge-Protokoll auf Basis des Graphisomorphieproblems

Literaturverzeichnis

[1] Bruce Schneier: „Angewandte Kryptographie“, Pearson Studium, 1996

[2] Jörg Rothe: „Complexity theory and cryptology—an introduction to cryptocomplexity“, Springer, 2005

[3] J. Buchmann: „Einführung in die Kryptographie“, Springer, 1999

[4] Alfred J. Menezes, Paul C. van Oorschot, Scott A. Vanstone: „Handbook of Applied Cryptography“, CRC Press, 1997, http://www.cacr.math.uwaterloo.ca/hac/

[5] Trappe, Washington: „Introduction to cryptography by coding theory“, Prantice Hall, 2002

[6] D. Wätjen: „Kryptographie – Grundlagen, Algorithmen, Protokolle“, Spektrum, 2004

[7] Douglas Stinson: „Cryptography—Theory and praxis“, CRC Press, 1995

[8] F. Bauer: „Entzifferte Geheimnisse“, Springer, 2000

[9] A. Beutelspacher, J. Schwenk, K.-D. Wolfenstetter: „Moderne Verfahren der Kryptographie“, 6. Auflage, Vieweg, 2006

[10] H. Delfs, H. Knebl: „Introduction to cryptography“, Springer, 2002

[11] O. Goldreich: „Fundamentals of cryptography“, Cambridge, 2001

[12] W. Diffie, M. Hellman: „New directions in cryptography“, IEEE Transaction on information theory 22, Seiten 644–654, 1976

[13] L. Adleman, R. Rivest, A. Shamir: „On digital signatures and public key cryptosystems“, MIT Press, 1977

[14] RSA Laboratories: „RSA Laboratories’ Frequently Asked Questions About Today’s Cryptography, Version 4.1“, RSA Security Inc., 2000, ftp://ftp.rsasecurity.com/pub/labsfaq/rsalabs_faq41.pdf

[15] Y. L. Yin: „The RC5 encryption algorithm: two years on“, CryptoBytes (3) 2, 1997

[16] J. Daemen, R. Govaerts, J. Vandewalle: „Weak keys for IDEA“, Advances in Cryptology—CRYPTO ’93, Springer-Verlag, 1994

[17] M. Luby, C. Rackoff: „How to Construct Pseudorandom Permutations and Pseudorandom Functions.“, SIAM Journal on Computing, vol. 17, 1988, S. 373–386, http://citeseer.ist.psu.edu/naor96construction.html (Überarbeitete Fassung)

[18] Mitsuru Matsui: „Linear cryptanalysis method for DES cipher“, Advances in Cryptology—EUROCRYPT '93 Proceedings, Springer-Verlag, 1994, S. 386–397

[19] Eli Biham, Adi Shamir: „Differential Cryptanalysis of DES-like Cryptosystems“, CRYPTO'90 & Journal of Cryptology, vol. 4, No. 1, 1991, http://www.cs.technion.ac.il/~biham/Reports/Weizmann/cs90-16.ps.gz

[20] Eli Biham, Adi Shamir: „Differential Cryptanalysis of the Full 16-Round DES“, Proceedings of CRYPTO '92, vol. 740, December 1991, http://www.cs.technion.ac.il/users/wwwb/cgi-bin/tr-info.cgi?1991/CS/CS0708

[21] „Data Encryption Standard, FIPS PUB 46-3“, National Institute of Standards and Technology, 1999, http://csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf

[22] David Coppersmith: „The Data Encryption Standard (DES) and its strength against attacks“, IBM Journal of Research and Development, Band 38, 1994, http://www.research.ibm.com/journal/rd/383/coppersmith.pdf

[23] Keith W. Campbell, Michael J. Wiener: „DES is not a group“, Advances in cryptology - Crypto '92, pages 512-520, Springer-Verlag, 1993, http://dsns.csie.nctu.edu.tw/research/crypto/HTML/PDF/C92/512.PDF

[24] Ron Rivest: „A method for obtaining digital signatures and public-key cryptosystems“, Communications of the ACM, 21(2):120-126, February 1978. http://people.csail.mit.edu/rivest/""RivestShamirAdleman-AMethodForObtainingDigitalSignaturesAndPu pdf

[25] „Advanced Encryption Standard, FIPS PUB 197“, National Institute of Standards and Technology, 2001, http://www.csrc.nist.gov/publications/fips/fips197/fips-197.pdf

[26] Michael J. Wiener: „Cryptanalysis of short RSA secret exponents“, IEEE Transactions on Information Theory, 36 (1990), 553–558

B. Übungsaufgaben

B.1. Blatt 1

Der folgende mit Vigenère-Chiffrierung deutschsprachige Text ist zu entschlüsseln. (Die Blockstruktur dient lediglich der besseren Lesbarkeit.)

Aufgabe 1

Bestimmen Sie eine mögliche Periodenlänge d

a) mit Hilfe des Kasiski-Tests.

b) mit Hilfe des Friedman-Tests.

c) Formulieren Sie eine Hypothese für d.

Aufgabe 2

Bestimmen Sie gemäß d eine Häufigkeitsverteilung und leiten Sie daraus ein Schlüsselwort ab.

Aufgabe 3

Bestimmen Sie den Klartext.

```
FSGEXV  EVIISA  MGYFNX  EJTMUR  MPNYME  FMPSIH  EFIXUE  HQFOOU
PGIAVI  KJSWLT  IIZJIJ  ELXVOT  YBKMEC  GYUELW  RHEHOR  ONIFVS
EHKCJS  WLFEEL  JIBNTS  VTIMGY  JSNECT  IBRQVE  HXJDHF  YVTSYP
EEIYWX  JLNRRU  UYVCJC  BELDHZ  YVSKFE  IUERXV  TGCFKF  IHIZOF
UGYFSP  IIGABV  VOGYRL  FGYRUM  AHKVOK  FEIUER  XRVFTY  XFHIII
JGEIZU  ZOIZOE  LFVLAH  RKFNMT  IBCBIQ  VUHXVS  SOGYFN  ILEFSY
MEFSSR  KBXORU  TEGEEU  IEDLCE  NVRDHN  IE
```

B.2. Blatt 2

Aufgabe 1

1. Verschlüsseln Sie die Nachricht $m = \text{WASSER}$ mit Hilfe der Hill-Chiffre und verwenden Sie die Matrix
$$K = \begin{pmatrix} 7 & 3 \\ 5 & 8 \end{pmatrix}$$
2. Bestimmen Sie die entsprechende Matrix zur Entschlüsselung und wenden Sie diese auf den in a) erhaltenen Geheimtext an.

Aufgabe 2

Bestimmen Sie eine 2×2-Matrix M, durch die die Nachricht „hund“ mit der Hill-Chiffre in den Geheimtext „AFFE“ umgewandelt wird.

Aufgabe 3

Wir definieren die Abbildung $\Phi_{a,b} \colon \mathbb{Z}_{26} \to \mathbb{Z}_{26}$ durch

$$\Phi_{a,b}(x) := (ax + b) \pmod{26}$$

Bestimmen Sie diejenigen ganzen Zahlen a, b, für die die Abbildung $\Phi_{a,b}$ bijektiv ist.

Bemerkung: Diese Abbildung kann zum Verschlüsseln verwendet werden (affine Chiffre)

B.3. Blatt 3

Der folgende Text wurde mit der Vigenère-Chiffrierung verschlüsselt. Die Blockstruktur dient lediglich der besseren Lesbarkeit.

Aufgabe 1

Bestimmen Sie eine mögliche Periodenlänge d

1. mit Hilfe des Kasiski-Tests.
2. mit Hilfe des Friedman-Tests.
3. Formulieren Sie eine Hypothese für d.

Aufgabe 2

Bestimmen Sie gemäß d eine Häufigkeitsverteilung und leiten Sie darauf ein Schlüsselwort ab.

Aufgabe 3

Bestimmen Sie den Klartext.

```
UEQPC  VCKAH  VNRZU  RNLAO  KIRVG  JTDVR  VRICV  IDLMY
IYSBC  COJQS  ZNYMB  VDLOK  FSLMW  EFRZA  VIQMF  JTDIH
CIFPS  EBXMF  FTDMH  ZGNMW  KAXAU  VUHJH  NUULS  VSJIP
JCKTI  VSVMZ  JENZS  KAHZS  UIHQV  IBXMF  FIPLC  XEQXO
CAVBV  RTWMB  LNGNI  VRLPF  VTDMH  ZGNMW  KRXVR  QEKVR
LKDBS  EIPUC  EAWJS  BAPMB  VSZCF  UEGIT  LEUOS  JOUOH
UAVAG  ZEZIS  YRHVR  ZHUMF  RREMW  KNLKV  KGHAH  FEUBK
LRGMB  JIHLI  IFWMB  ZHUMP  LEUWG  RBHZO  LCKVW  THWDS
ILDAG  VNEMJ  FRVQS  VIQMU  VSWMZ  CTHII  WGDJS  XEOWS
JTKIH  KEQ
```

C. Lösungen

C.1. Blatt 1

Aufgabe 1

a) Für den Kasiski-Test muss man Wiederholungen von Zeichenketten finden, die länger sind als 2 Zeichen, Beispiele:

Zeichenkette	Auftreten	Abstand
KFEIUERX	172, 222	$50 = 2 \cdot 5^2$
YMEF	28, 288	$260 = 2^2 \cdot 5 \cdot 13$
GYFN	14, 279	$265 = 5 \cdot 53$
JSWL	56, 101	$45 = 3^2 \cdot 5$
GRY	207, 212	5
TIB	126, 264	$138 = 2 \cdot 3 \cdot 23$

b) Für den Friedman-Test muss die Anzahl l_i des Auftretens der einzelnen Buchstaben gezählt werden, was wesentlich einfacher mit einem Programm geht. Daraus kann man mit Gleichung 2.1 den Koinzidenzindex für den Text bestimmen – $I(c) \approx 0{,}0483$ – woraus sich mit Gleichung 2.3 eine Periodenlänge von $d \approx 3{,}93$ ergibt.

c) Da vor allem 5 als Faktor in den längeren Wiederholungen beim Kasiski-Test auftritt und $d \approx 3{,}93$ aus dem Friedman-Test näher an 5 als an 2 liegt, ist eine erste Schätzung für die Periodenlänge $d = 5$.

TIB scheint eine zufällige Wiederholung zu sein, die nicht auf eine gleiche Verschlüsselung von gleichen Zeichenketten zurückzuführen ist.

Aufgabe 2

Für die Bestimmung des Schlüsselworts muss für jeden Textteil (jede Spalte) die Häufigkeitsverteilung der Buchstaben ermittelt werden. Dies geht am Leichtesten mit einem Programm. Im 1., 2., 4. und 5. Textteil ist die Zuordnung des Buchstabens E eindeutig. Hier treten die Buchstaben F, E, I resp. V am Häufigsten auf. In der dritten Spalte gibt es zwei Möglichkeiten: H oder Y, beide treten acht Mal auf. Jedoch liefert nur die Zuordnung von H auf E einen sinnvollen Text. Das Schlüsselwort ist also BAUER.

Spalte	häuf. Buchst.	Verschiebung	Verschl. v. a
1.	F	1	B
2.	E	0	A
3a	H	3	D
3b	Y	20	U
4.	I	4	E
5.	V	17	R

Aufgabe 3

Es mag ueberraschen, dass man von einem vorgelegten monoalphabetisch chiffrierten Text leichter sagen kann, ob er englisch oder franzoesisch ist als ihn zu entschluesseln. Dies gilt natuerlich auch fuer Klartext. Es gibt ein einfaches Verfahren genuegend langen Klartext auf Zugehoerigkeit zu einer bekannten Sprache zu untersuchen ohne seine Syntax und Semantik zu betrachten.

C.2. Blatt 2

Aufgabe 1

1. Die Verschlüsselungsmatrix K arbeitet mit Blöcken der Länge 2. Also muss der Klartext `wasser` erst einmal kodiert werden (a $\to$ 0) und dann in Blöcke geteilt werden. Es ergeben sich also die drei Blöcke

   $$v_1 = \begin{pmatrix}22 & 0\end{pmatrix} \qquad v_2 = \begin{pmatrix}18 & 18\end{pmatrix} \qquad v_3 = \begin{pmatrix}4 & 17\end{pmatrix}$$

   Diese werden durch $w = v \cdot K$ verschlüsselt und wieder als Text kodiert (0 $\to$ A): `YOIJS`. Man beachte, dass die Einträge in den Matrizen/Vektoren Elemente aus $\mathbb{Z}_{26}$ sind.

   $$w = \begin{pmatrix}v_1\\v_2\\v_3\end{pmatrix} \cdot K = \begin{pmatrix}22 & 0\\18 & 18\\4 & 17\end{pmatrix} \cdot \begin{pmatrix}7 & 3\\5 & 8\end{pmatrix} = \begin{pmatrix}24 & 14\\8 & 16\\9 & 18\end{pmatrix}$$

2. Die inverse Matrix zu K kann man entweder mit der Formel aus Abschnitt 4.3 oder durch simultane Umformung einer Einheitsmatrix bei der Umformung von K in eine Einheitsmatrix bestimmen. Mit dieser Matrix K^{-1} kann man dann den Geheimtext auf die gleiche Weise entschlüsseln, wie man den Klartext verschlüsselt hat: $v_i = w_i \cdot K^{-1}$. Nach der Kodierung durch Buchstaben (0 $\to$ a) ergibt sich dann auch wieder der Klartext `wasser`.

   $$v = \begin{pmatrix}24 & 14\\8 & 16\\9 & 18\end{pmatrix} \cdot \begin{pmatrix}4 & 5\\17 & 23\end{pmatrix} = \begin{pmatrix}22 & 0\\18 & 18\\4 & 17\end{pmatrix}$$

$$\begin{pmatrix}7 & 3\\ 5 & 8\end{pmatrix} \quad \begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix} \quad | \cdot 15$$
$$\begin{pmatrix}1 & 19\\ 5 & 8\end{pmatrix} \quad \begin{pmatrix}15 & 0\\ 0 & 1\end{pmatrix} \quad \overset{21}{\underset{+}{\leftarrow\!\!\rightarrow}}$$
$$\begin{pmatrix}1 & 19\\ 0 & 17\end{pmatrix} \quad \begin{pmatrix}15 & 0\\ 3 & 1\end{pmatrix} \quad | \cdot 23$$
$$\begin{pmatrix}1 & 19\\ 0 & 1\end{pmatrix} \quad \begin{pmatrix}15 & 0\\ 17 & 23\end{pmatrix} \quad \overset{+}{\underset{7}{\leftarrow\!\!\rightarrow}}$$
$$\begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix} \quad \begin{pmatrix}4 & 5\\ 17 & 23\end{pmatrix}$$

Mit etwas Glück fällt auf, dass 7 eine Einheit von $\mathbb{Z}_{26}$ ist und man so leicht in der oberen linken Ecke eine 1 erzeugen kann. Das Inverse zu 7 ist 15, daher wird die erste Zeile mit 15 durchmultipliziert.
Dann ist es leicht in der unteren Zeile aus dem linken Element eine 0 zu machen: $0 = 21 \cdot 1 + 5 \pmod{26}$.
17 ist wieder eine Einheit, so dass eine 1 durch Multiplikation mit 23 erzeugt werden kann: $23 \cdot 17 \pmod{26} = 1$.
Aus der 19 lässt sich jetzt wiederum sehr leicht eine 0 erzeugen: $0 = 1 \cdot 7 + 19 \pmod{26}$.

Aufgabe 2

Die Kodierung der Nachricht `hund` ist $v \in \mathbb{Z}_{26}^{2\times 2}$ und die des Geheimtexts `AFFE` ist $w \in \mathbb{Z}_{26}^{2\times 2}$, wobei `a` im Klartext bzw. `A` im Geheimtext durch 0 kodiert wird.

$$v = \begin{pmatrix}7 & 20\\ 13 & 3\end{pmatrix} \qquad w = \begin{pmatrix}0 & 5\\ 5 & 4\end{pmatrix}$$

Es ist jetzt nach einer Matrix M gesucht, die $w = v \cdot M$ erfüllt, d. h. $M = v^{-1} \cdot w$. Mit dem Verfahren aus Aufgabe 1 ergibt sich v^{-1} in 4 Schritten (1: $v_1 = 15 \cdot v_1$, 2: $v_2 = 13 \cdot v_1 + v_2$, 3: $v_2 = 9 \cdot v_2$, 4: $v_1 = 12 \cdot v_2 + v_1$).

$$M = v^{-1} \cdot w = \begin{pmatrix}15 & 4\\ 13 & 9\end{pmatrix} \cdot \begin{pmatrix}0 & 5\\ 5 & 4\end{pmatrix} = \begin{pmatrix}20 & 13\\ 19 & 23\end{pmatrix}$$

Aufgabe 3

Diese Aufgabe war eigentlich nur ein Test, ob man in der Vorlesung aufgepasst hat, weil die Antwort bereits in Abschnitt 2.1.2 gegeben wurde.

Ob die Abbildung bijektiv ist oder nicht, hängt nicht von b ab. Es hängt allein von a ab, ob zwei Zeichen x_1, x_2 durch das gleiche Zeichen ($E(x_1) = E(x_2)$) verschlüsselt werden.

$$ax_1 + b \equiv_k ax_2 + b \Leftrightarrow ax_1 \equiv_k ax_2$$

Haben k und a einen gemeinsamen Teiler $ggT(k, a) \neq 1$, so ist die Entschlüsselung der Null beispielsweise nicht mehr eindeutig, denn $E(0) = 0 \cdot a = 0$ und

$$E\Big(\frac{k}{ggT(k,a)}\Big) \equiv \frac{k}{ggT(k,a)} \cdot a \equiv k \cdot \underbrace{\frac{a}{ggT(k,a)}}_{\in \mathbb{Z}_k,\, >1} \equiv 0 \pmod{k}$$

Also muss a teilerfremd zu k sein, damit alle $\Phi_{a,b}$ bijektiv sind.

Der Versuch einer Erklärung: Es gilt $ax_1 \equiv_k ax_2 \Leftrightarrow a(x_1 - x_2) = l \cdot k$. Da a teilerfremd zu k, muss l ein Vielfaches von a sein, also $l = \lambda a$. Es gilt demnach $a(x_1 - x_2) = \lambda ak \Rightarrow x_1 - x_2 = \lambda k$. Da $x_1, x_2 \in \mathbb{Z}_k$, ist $-k < x_1 - x_2 < k$. Das einzige λ, das die Gleichung erfüllt, ist $\lambda = 0$. Also ist $x_1 - x_2 = 0$ und damit gilt die Kongruenz, wenn a und k teilerfremd sind, nur für gleiche x. Für unterschiedliche x ergeben sich unterschiedliche Verschlüsselungen, die $\Phi_{a,b}$ ist also bijektiv.

C.3. Blatt 3

Aufgabe 1

a) Für den Kasiski-Test muss man Wiederholungen von Zeichenketten finden, die länger sind als 2 Zeichen, Beispiele:

Zeichenkette	Auftreten	Abstand
`TDMHZGNMWK`	92, 182	$90 = 2 \cdot 3^2 \cdot 5$
`BXMFF`	87, 147	$60 = 2^2 \cdot 3 \cdot 5$
`ZHUM`	256, 296	$40 = 2^3 \cdot 5$
`VIQM`	71, 336	$265 = 5 \cdot 53$
`KAH`	8, 136	$128 = 2^7$
`JTD`	26, 76	$50 = 2 \cdot 5^2$

b) Für den Friedman-Test muss die Anzahl l_i des Auftretens der einzelnen Buchstaben gezählt werden, was wesentlich einfacher mit einem Programm geht. Daraus kann man mit Gleichung 2.1 den Koinzidenzindex für den Text bestimmen – $I(c) \approx 0{,}0412$ – woraus sich mit Gleichung 2.3 eine Periodenlänge von $d \approx 13{,}38$ ergibt.

Stimmt das $d = 13{,}38$?

c) Da vor allem 2 und 5 als Faktoren in den längeren Wiederholungen beim Kasiski-Test auftritt und $d \approx 13{,}38$ aus dem Friedman-Test auf 10 deuten, ist eine erste Schätzung für die Periodenlänge $d = 10$.

`KAH` scheint eine zufällige Wiederholung zu sein, die nicht auf eine gleiche Verschlüsselung von gleichen Zeichenketten zurückzuführen ist.

Aufgabe 2

Mit der Periodenlänge $d = 10$ ergeben sich vier mögliche Schlüsselwörter `HNZIORADIO`, `RNZIORADIO`, `HNZIORAHIO` und `RNZIORAHIO`, die alle keinen sinnvollen Klartext liefern.

Also korrigieren wir die Schätzung auf $d = 5$. Damit ergibt sich das Schlüsselwort `RADIO`, womit sich auch ein sinnvoller deutscher Klartext ergibt.

Spalte	häuf. Buchst.	Verschiebung	Verschl. v. a
1.	V	17	R
2.	E	0	A
3.	H	4	D
4.	M	8	I
5.	S	14	O

Aufgabe 3

Den hoechsten Organisationsstand erfuhr die Kryptologie in Venedig, wo sie in Form einer staatlichen Buerotaetigkeit ausgeuebt wurde. Es gab Schluesselsekretaere die ihr Buero im Dogenpalast hatten und fuer ihre Taetigkeit rund zehn Dukaten im Monat bekamen. Es wurde dafuer gesorgt, dass sie waehrend ihrer Arbeit nicht gestoert wurden. Sie durften ihre Bueros aber auch nicht verlassen, bevor sie eine gestellte Aufgabe geloest hatten.

Index