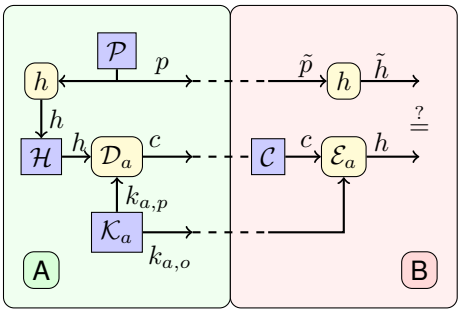
der Bundeswehr
Universität München

Dr. Robert Schmied



Professur für Mathematik
Fakultät für Elektrotechnik
und Informationstechnik
85577 Neubiberg
Tel.: 089/6004-3931
Fax: 089/6004-2615
robert.schmied@unibw.de

Kryptologie

FT 2023

31. März 2023

Inhaltsverzeichnis

1. Datensicherheit in Systemen

Der Austausch und die Ablage von Daten ist in der heute digital vernetzten Welt von großer Bedeutung. Die Daten bestehen aus Zeichen, die aus einem Zeichenvorrat, dem Alphabet, stammen und in der Regel nach gewissen syntaktischen Vorgaben zusammengesetzt werden dürfen. Erst in einem Bedeutungskontext werden aus Daten Informationen. Unter Daten verstehen wir damit Symbole, die noch nicht interpretiert sind.

Definition 1.1: Alphabet, Zeichen, Wort, Daten

Sei $\mathcal{A}$ eine beliebige nichtleere, total geordnete endliche Menge, genannt Alphabet. Jedes $a \in \mathcal{A}$ heißt Zeichen. Sei weiter $\mathcal{A}^k := \{(a_1, \ldots, a_k);\ a_i \in \mathcal{A}, i \in \{1, \ldots, k\}, k \in \mathbb{N}\}$. Jedes Tupel $(a_1, \ldots, a_k) \in \mathcal{A}^k$ heißt Wort der Länge k. $\mathcal{P}_\mathcal{A} \subseteq \mathcal{A}^* := \bigcup_{k=1}^{\infty} \mathcal{A}^k$ sei die Menge der zugelassenen Nachrichten. Jedes Element $p \in \mathcal{P}_\mathcal{A}$ wird als Nachricht bezeichnet.

Bemerkung.
(1) Ist das Alphabet klar, so kann statt $\mathcal{P}_\mathcal{A}$ auch $\mathcal{P}$ geschrieben werden.
(2) Im speziellen Kontext der Kommunikation werden Nachrichten auch Daten genannt.
(3) Das Alphabet enthält den für Daten zulässigen Zeichenvorrat. In manchen Anwendungen wird die maximale Länge von Daten vorgegeben. Zudem können bestimmte Zeichenkombinationen als unsinnig angesehen und vorab ausgeschlossen werden. Diese beiden Fälle haben zur Einführung der Menge $\mathcal{P}_\mathcal{A}$ geführt.
(4) Der Einfachheit halber werden wir Daten nicht immer in Tupelschreibweise darstellen, sondern durch Konkatenation der einzelnen Elemente aus dem jeweiligen Alphabet, etwa KRYPTOGRAPHIE anstelle von (K,R,Y,P,T,O,G,R,A,P,H,I,E). Zudem benutzen wir folgende Schreibweise, um Daten $p = (p_1, \ldots, p_l, p_{l+1}, \ldots, p_m) \in \mathcal{P}_\mathcal{A}$ in zwei Teile aufzuteilen:

$$\underbrace{p_1 \ldots p_l}_{L} \,||\, \underbrace{p_{l+1} \ldots p_m}_{R}.$$

Wir werden uns mit digitalen Daten beschäftigen. Um sie aufbewahren bzw. austauschen zu können, müssen dafür in technischen Systemen verschiedene Dienste angeboten werden. Eine Instanz ist ein an der Datenhaltung bzw. am Datenaustausch Beteiligter. Er nutzt die von Systemen angebotenen Dienste allein oder zusammen mit weiteren Instanzen.

1.1. Sicherheitsziele und Bedrohungen

Wir betrachten zwei Arten technischer Systeme. Ein Kommunikationssystem ist ein System, das den Austausch von Daten zwischen wenigstens zwei Instanzen ermöglicht. Ein Datenhaltungssystem ist ein System, das die Aufbewahrung von und den Zugriff auf Daten ermöglicht. Die beiden genannten technischen Systeme sollen gewährleisten, dass die

von ihnen bereitgestellten Dienste die jeweiligen Daten zur Zufriedenheit der nutzenden Instanzen verarbeiten.

Definition 1.2: Dienst

Ein Dienst ist eine technische Einheit eines Systems, die zusammenhängende Funktionalitäten bündelt und sie über eine klar definierte Schnittstelle bereitstellt.

Die Zufriedenheit einer Instanz wird durch die Sicherheit ihrer Daten erhöht. Unter Sicherheit stellen wir uns das Bewahren eines Systems vor Beeinträchtigung und Missbrauch vor. Das Thema Sicherheit von Daten betrifft dabei alle Ausprägungen der genannten Systeme. Bei der Datenhaltung möchte eine Instanz sicherstellen, dass keine unbefugte Instanz Zugriff auf die Daten hat, bei der Datenübertragung sollen die Daten ebenfalls unangetastet übertragen werden. Einen weiteren wichtigen Punkt stellt die Kommunikation selbst dar. Sie soll dritten Instanzen manchmal einseitig oder komplett verborgen bleiben. Ein Datenhaltungssystem ist auch ein Kommunikationssystem. Denn alles, was den wie auch immer gearteten Zugriff auf die Daten oder auf die Verwaltung der Daten abzielt, ist mit einer Kommunikation verbunden. Das Ablegen von instanzenbezogenen Daten erfordert deren vertrauliche Behandlung. Dieser Bereich wird in unseren Überlegungen auch durch die Betrachtung von Kommunikationssystemen abgedeckt. Wir werden uns deshalb auf das Thema Sicherheit in Kommunikationssystemen fokussieren. Um Sicherheit zu erreichen und gewährleisten, werden abstrakte Ziele formuliert.

Definition 1.3: Sicherheitsziel

Ein (technisches) Sicherheitsziel ist ein erwünschter Zustand eines technischen Systems, der die Sicherheit von Daten gewährleistet.

Wir unterscheiden folgende Sicherheitsziele:

- Vertraulichkeit
- Datenintegrität
- Authentizität und Kontrollierter Zugang
- Zurechenbarkeit und Nichtabstreitbarkeit
- Privatsphäre

Sollen nur berechtigte Instanzen Zugang zu (übertragenen) Daten des Systems haben, sprechen wir von Vertraulichkeit. Dabei kann es um die reinen Nutzdaten oder um die Identität des oder der beteiligten Instanzen gehen. Bei einer Kommunikation zweier Instanzen ist es für den Empfänger von Daten oft wichtig, eindeutig zu erkennen, ob die Daten durch Ändern, Ersetzen oder Löschen von Teilen von Daten verändert wurden. Wir nennen dies Datenintegrität. Dabei spielt es keine Rolle, ob die Daten vorsätzlich oder unbeabsichtigt geändert wurden. Soll eine Instanz seine eigene Identität einer anderen Instanz eindeutig nachweisen können oder soll überprüft werden, ob Daten von einer bestimmten Instanz stammen und sie unverändert sind, betrachten wir das Ziel der Authentizität. Sie schließt mit dem letztgenannten Punkt die Integrität mit ein. Weiter wird sie benötigt, wenn nur

autoriserte Instanzen Zugang zu bestimmten Diensten oder Daten haben sollen. Wir betrachten das Ziel des Kontrollierten Zugangs. Der Zugang kann auf bestimmte Zugriffsarten eingeschränkt sein. Bei der Nutzung eines Dienstes kann es ein Ziel sein, sie eindeutig einer Instanz zuzuordnen. Es liegt das Ziel der Zurechenbarkeit vor. Soll die Nutzung eines Dienstes nicht nachträglich durch eine beteiligte Instanz gegenüber dritten abgestritten werden, sprechen wir von Nichtabstreitbarkeit. Manchmal ist die völlige Geheimhaltung einer Dienstenutzung erwünscht. Es geht dann um das Ziel der Privatsphäre. Ein Kernpunkt dabei ist das Erreichen von Anonymität. Anonymität lässt sich in drei Arten unterteilen[1]: Anonymität der Sendeinstanz, der Empfängerinstanz und der gesamten Kommunikationsbeziehung. Es gibt weitere Sicherheitsziele, die wir jedoch im Kontext der Kryptologie nicht näher betrachten wollen. So bietet ein technisches System verschiedene Dienste an. Die Dienste sollen stets verfügbar sein und korrekt funktionieren. Wir sprechen vom Sicherheitsziel der Verfügbarkeit. Es gibt eine Vielzahl an Möglichkeiten, um das Erreichen der genannten Sicherheitsziele zu verhindern.

Definition 1.4: Bedrohung, Angriff

Eine Bedrohung ist ein potenzielles Ereignis beziehungsweise eine Reihe von Ereignissen, die zur Gefährdung eines oder mehrerer Sicherheitsziele führt. Die tatsächliche Realisierung einer Bedrohung nennt man einen Angriff.

Gibt eine Instanz vor, die Identität einer anderen Instanz zu besitzen, wird die Bedrohung Maskerade genannt. Eine passive Bedrohung liegt beim Abhören von Daten vor. Dabei werden Daten von einer Instanz gelesen, die nicht für sie bestimmt sind. Nutzt eine Instanz Dienste oder Ressourcen, die sie nicht nutzen sollte, liegt eine Autorisierungsverletzung vor. Werden Daten verändert, eingeführt oder gelöscht, werden Sicherheitsziele durch Verlust oder Modifikation bedroht. Erzeugt eine Instanz neue Daten unter Benutzung der Identität einer anderen Instanz, liegt eine Fälschung von Daten vor. Leugnet eine Instanz die Beteiligung an einem bestimmten Ereignis, haben wir es mit dem Abstreiten von Ereignissen zu tun. Durch Aktionen, welche die Verfügbarkeit oder Zuverlässigkeit eines Systems beeinträchtigen sollen, wird Sabotage betrieben. Welches Sicherheitsziel durch welche Bedrohungen gefährdet ist, wird in Tabelle 1.1 zusammengefasst.

1.2. Schichtenmodelle für Kommunikationssysteme

Kommunikation in Netzen ist nicht so einfach, wie das oft suggeriert wird. Dahinter steckt eine Vielzahl an Überlegungen und deren Umsetzung.

Beispiel 1.5

Kaufen wir im Internet ein[a], muss durch das Kommunikationssystem folgendes geleistet werden:

- Verschiedene Web-Ressourcen sind zu transportieren (HTML, Bilder, . . .)
- Die Übertragung vertraulicher Daten muss gesichert werden

[1] vgl. [2], S.79

Tabelle 1.1.: Sicherheitsziele und Bedrohungen nach [7]

	Maskerade	Abhören	Autorisierungsverletzung	Daten einfügen, löschen oder verändern	Fälschen von Daten	Abstreiten von Ereignissen	Sabotage
Verfügbarkeit	✓		✓	✓			✓
Vertraulichkeit	✓	✓	✓				
Datenintegrität	✓		✓	✓	✓		
Kontrollierter Zugang	✓		✓		✓		
Zurechenbarkeit	✓		✓	✓		✓	
Privatsphäre		✓					

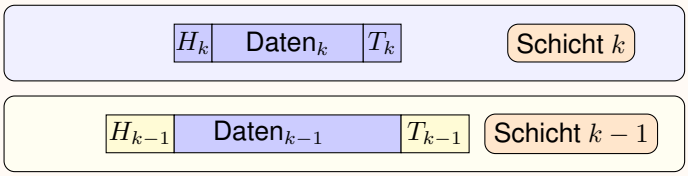
- Mehrere Kommunikationsströme (Web, E-Mail, Skype, . . .) sind zu trennen
- Datenpakete müssen durch das Internet das Ziel finden
- Datenpakete können im Netz verloren gehen und damit neu gesendet werden
- Das langsamere Internet darf nicht durch ein schnelles LAN überflutet werden
- In einem Funknetz senden nicht alle gleichzeitig, der Zugriff ist zu regeln
- Daten müssen physikalisch übertragen werden (Funk, DSL, Ethernet, . . .)
- Einzelne Bits können bei der physikalischen Übertragung verändert werden; dies muss erkannt werden

[a] Die Überlegungen sind aus [9] übernommen.

In Beispiel 1.5 sind sehr viele komplexe Aufgaben genannt, die je nach Kontext unterschiedlich zur Anwendung kommen. Bei einer E-Mail werden keine Web-Ressourcen benötigt, ansonsten werden alle anderen Items berücksichtigt. In Funknetzen werden andere Vorgehensweisen zur physikalischen Übermittlung benutzt als in drahtgebundenen Netzen. Deshalb werden Kommunikationssysteme in verschiedene Schichten eingeteilt. Jede Schicht übernimmt spezielle Teilfunktionen der Kommunikation. Damit ist die Komplexität jeder einzelnen Komponente geringer. Zudem kann je nach Situation eine Schicht weggelassen oder gegen eine andere Implementierung getauscht werden. Eine unmittelbar

unter einer Schicht k liegende Schicht $k-1$ kapselt sämtliche Daten jener Schicht, d.h. alles aus der Schicht k wird unverändert als Block übernommen wie in Abbildung 1.1 zu sehen ist. Diesem Block kann ein Header H voran- und ein Trailer T hintenangestellt werden. Der Header enthält Steuerinformationen, die von der entsprechenden Schicht beim Empfänger interpretiert werden können. Der Trailer enthält in der Regel Informationen zum Erkennen und Korrigieren von Übertragungsfehlern.

Abbildung 1.1.: Schichtenmodell



1.3. Kommunikationssicherheit

Ein technisches System muss für die gewünschten Sicherheitsziele entsprechende Dienste zur Verfügung stellen. Ein Sicherheitsdienst ist ein abstrakter Dienst, der zur Erreichung eines bestimmten Sicherheitszieles realisiert wird. Ein Sicherheitsdienst stellt über Schnittstellen Funktionalitäten zur Verfügung, die jeweils durch eine oder mehrere Funktionen implementiert werden. Solche Dienste ermöglichen eine Kommunikation und so den Austausch von Daten zwischen wenigstens zwei Instanzen. Die Kommunikation läuft oft nach gleichen Mustern und mit den gleichen Zielen ab. Es macht daher Sinn, diese nach vorher festgelegten Vorgaben durchzuführen.

Definition 1.6: Kommunikationsprotokoll

Ein Kommunikationsprotokoll ist eine Ablaufvorschrift für eine Reihe von Verarbeitungsschritten zum Zweck des Austauschs von Daten, an dem wenigstens zwei Instanzen beteiligt sind.

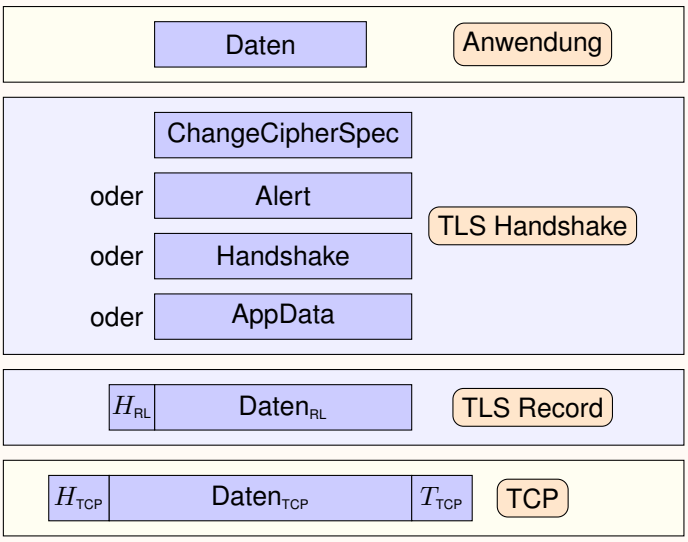
Es gibt Kommunikationsprotokolle, die bei der Ausführung wenigstens einen Sicherheitsdienst adressieren. Ein sehr wichtiges und in der Anwendung weit verbreitetes Beispiel eines solchen Kommunikationsprotokolls ist das TLS/SSL-Protokoll.

Beispiel 1.7: TLS/SSL-Protokoll

Mit dem TLS-Protokoll wird ein sicherer und synchroner Kommunikationskanal zwischen zwei Instanzen in einem TCP-Netzwerk aufgebaut. Die beiden beteiligten Instanzen sind hier näher als Client- und Serversystem spezifiziert. TLS ermöglicht die Authentisierung des jeweils anderen Kommunikationspartners. Vor der eigentlichen Kommunikation können sowohl die Identität des Clients als auch des Servers für die jeweils andere Instanz überprüft werden. Nach dieser Authentifizierung wird eine TLS-Sitzung, in der die Parameter der Kommunikation mitgeführt werden, zwi-

schen Client und Server etabliert. Weiter wird die Vertraulichkeit der Daten umgesetzt. Beim Aufbau der Sitzung kann eine Verschlüsselung der Daten festgelegt werden. Alle übermittelten Daten werden durch einen so genannten Nachrichtenprüfwert gesichert, um die Datenintegrität zu sichern. TLS ist die direkte Weiterentwicklung von SSL, das von Netscape Communications als Schutzmechanismus für das HTT-Protokoll geschaffen wurde, und stammt aus dem Jahre 1999. Aktuell wird Version 1.2 verwendet. TLS setzt auf TCP auf, bei dem ein Datenstrom vollständig und reihenfolgetreu übertragen wird. In Abbildung 1.2 ist das TLS-Protokoll präzisiert zu sehen. Oberhalb von TCP arbeitet das RecordLayer Protokoll. Es setzt die Verarbeitung der Daten, die Verschlüsselung und Authentifizierung um. Dies wird durch den Datenanteil des Protokolls durch eines von vier weiteren Protkollen einer darüberliegenden Schicht übernommen. Das AppData-Protokoll bildet die Schnittstelle zur Anwendung, von der die zu übertragenden Daten stammen. Von ihr ausgehende Daten werden von TLS übernommen und an die andere Instanz gesendet, eingehende Daten nach der Bearbeitung von TLS an die Anwendung weitergereicht. Das Handshake-Protokoll hat die Aufgabe, die im weiteren zu verwendenden Schlüssel beim Verbindungsaufbau auszuhandeln. Das ausgehandelte Schlüsselmaterial wird durch das ChangeCipherSpec-Protokoll aktiviert. Bei nicht vorgesehenen Zuständen wird das Alert-Protokoll für Status- und Fehlermeldungen benötigt.

Abbildung 1.2.: Übersicht über die TLS-Schicht



Aus dem Beispiel 1.7 ist zu erkennen, dass über ein Kommunikationsprotokoll die Möglichkeit besteht, auf klar strukturierte Weise verschiedene Sicherheitsdienste zu bedienen. Welche Funktionalitäten mit welchen Einzelfunktionen dabei jeweils verwendet werden, ist zunächst zweitrangig. Allerdings muss die Erreichbarkeit des Ziels nach noch vorzugebenden Maßstäben gewährleistet werden.

2. Grundbegriffe der Kryptologie

2.1. Kryptologie, Kryptographie und Kryptoanalyse

Kryptologie beschäftigt sich mit dem Schutz von Daten vor unbefugtem Zugriff. Aus dem alt-griechischen übersetzt erhalten wir die „Lehre vom Verbergen “[1]. Dies greift aus heutiger Sicht jedoch etwas zu kurz. Die Kryptologie wird in zwei Teilgebiete aufgeteilt. In seiner ursprünglichen Bedeutung beschäftigt sich die Kryptographie[2] mit dem Umformen von lesbaren Texten in nicht lesbare Texte und umgekehrt mit Hilfe eines geheimzuhaltenden Schlüssels. Das Umformen modellieren wir mit Hilfe einer Abbildung. Dazu müssen wir entsprechend die Definitionsmenge, die Bildmenge und die Abbildungsvorschrift festlegen.

Beispiel 2.1: Caesar-Chiffre

Bei der Caesar-Chiffre[a] nehmen wir als Definitions- und Bildmenge jeweils das Alphabet

$$\mathcal{A}_0 := \{A, B, C, D, E, F, \ldots, X, Y, Z\}$$

der Großbuchstaben zusammen mit einem Shift-Wert 3, der dem Schlüssel entspricht. So definieren wir die Abbildung

$$e : \mathcal{A}_0 \to \mathcal{A}_0,\ A \mapsto D, B \mapsto E, C \mapsto F, \ldots, X \mapsto A, Y \mapsto B, Z \mapsto C.$$

Jeder Buchstabe wird durch den drei Buchstaben weiter hinten stehenden Buchstaben ersetzt (Rechts-Shift). Dies stellt den Schlüssel des Verfahrens dar. Am Ende des Alphabets angelangt wird von vorne mit der Zählung fortgesetzt. Aus dem lesbaren Text „KRYPTOGRAPHIE“ wird dann der nicht lesbare Text “NUBSWRJUDSKLH“. Für ein längeres Beispiel nehmen wir den Anfang des Evangeliums nach Markus[b]:

ANFANGDESEVANGELIUMSVONJESUSCHRISTUSDEMSO
HNGOTTESESBEGANNWIEESBEIDEMPROPHETENJESAJ
ASTEHTICHSENDEMEINENBOTENVORDIRHERERSOLLD
ENWEGFUERDICHBAHNENEINESTIMMERUFTINDERWUE
STEBEREITETDEMHERRNDENWEGEBNETIHMDIESTRAS
SENSOTRATJOHANNESDERTAEUFERINDERWUESTEAUF
UNDVERKUENDIGTEUMKEHRUNDTAUFEZURVERGEBUNG
DERSUENDENGANZJUDAEAUNDALLEEINWOHNERJERUS
ALEMSZOGENZUIHMHINAUSSIEBEKANNTENIHRESUEN
DENUNDLIESSENSICHIMJORDANVONIHMTAUFENJOHA
NNESTRUGEINGEWANDAUSKAMELHAARENUNDEINENLE
DERNENGUERTELUMSEINEHUEFTENUNDERLEBTEVONH
EUSCHRECKENUNDWILDEMHONIGERVERKUENDETENAC

[1] ὁ λόγος, τὸ κρύπτειν
[2] τὸ γράφειν

HMIRKOMMTEINERDERISTSTAERKERALSICHICHBINE
SNICHTWERTMICHZUBUECKENUMIHMDIESCHUHEAUFZ
USCHNUERENICHHABEEUCHNURMITWASSERGETAUFTE
RABERWIRDEUCHMITDEMHEILIGENGEISTTAUFENDIE
TAUFEJESUINJENENTAGENKAMJESUSAUSNAZARETIN
GALILAEAUNDLIESSSICHVONJOHANNESIMJORDANTA
UFENUNDALSERAUSDEMWASSERSTIEGSAHERDASSDER
HIMMELSICHOEFFNETEUNDDERGEISTWIEEINETAUBE
AUFIHNHERABKAMUNDEINESTIMMEAUSDEMHIMMELSP
RACHDUBISTMEINGELIEBTERSOHNANDIRHABEICHGE
FALLENGEFUNDENDIEVERSUCHUNGJESUDANACHTRIE
BDERGEISTJESUSINDIEWUESTEDORTBLIEBJESUSVI
ERZIGTAGELANGUNDWURDEVOMSATANINVERSUCHUNG
GEFUEHRTERLEBTEBEIDENWILDENTIERENUNDDIEEN
GELDIENTENIHMERSTESAUFTRETENINGALILAEANAC
HDEMMANJOHANNESINSGEFAENGNISGEWORFENHATTE
GINGJESUSWIEDERNACHGALILAEAERVERKUENDETED
ASEVANGELIUMGOTTESUNDSPRACHDIEZEITISTERFU
ELLTDASREICHGOTTESISTNAHEKEHRTUMUNDGLAUBT
ANDASEVANGELIUMDIEBERUFUNGDERERSTENJUENGE
RALSJESUSAMSEEVONGALILAEAENTLANGGINGSAHER
SIMONUNDANDREASDENBRUDERDESSIMONDIEAUFDEM
SEEIHRNETZAUSWARFENSIEWARENNAEMLICHFISCHE
RDASAGTEERZUIHNENKOMMTHERFOLGTMIRNACHICHW
ERDEEUCHZUMENSCHENFISCHERNMACHENSOGLEICHL
IESSENSIEIHRENETZELIEGENUNDFOLGTENIHMALSE
REINSTUECKWEITERGINGSAHERJAKOBUSDENSOHNDE
SZEBEDAEUSUNDSEINENBRUDERJOHANNESSIEWAREN
IMBOOTUNDRICHTETENIHRENETZEHERSOFORTRIEFE
RSIEUNDSIELIESSENIHRENVATERZEBEDAEUSMITSE
INENTAGELOEHNERNIMBOOTZURUECKUNDFOLGTENJE
SUSNACH

Wir erhalten

DQIDQJGHVHYDQJHOLXPVYRQMHVXVFKULVWXVGHPVR
KQJRWWHVHVEHJDQQZLHHVEHLGHPSURSKHWHQMHVDM
DVWHKWLFKVHQGHPHLQHQERWHQYRUGLUKHUHUVROOG
HQZHJIXHUGLFKEDKQHQHLQHVWLPPHUXIWLQGHUZXH
VWHHUHLWHWGHPKHUUQGHQZHJHEQHWLKPGLHVWUDV
VHQVRWUDWMRKDQQHVGHUWDHXIHULQGHUZXHVWHDXI
XQGYHUNXHQGLJWHXPNHKUXQGWDXIHCXUYHUJHXQJ
GHUVXHQGHQJDQCMXGDHDXQGDOOHHLQZRKQHUMHUXV
DOHPVCRJHQCXLKPKLQDXVVLHHNDQQWHQLKUHVXHQ
GHQXQGOLHVVHQVLFKLPMRUGDQYRQLKPWDXIHQMRKD
QQHVWUXJHLQJHZDQGDXVNDPHOKDDUHQXQGHLQHQOH
GHUQHQJXHUWHOXPVHLQHKXHIWHQXQGHUOHEWHYRQK
HXVFKUHFNHQXQGZLOGHPKRQLJHUYHUNXHQGHWHQDF
KPLUNRPPWHLQHUGHULVWVWDHUNHUDOVLFHKLFHKELQH
VQLFKWZHUWPLFKCXEXHFNHQXPLKPGLHVFKXKHDXIC
XVFKQXHUHQLFHKDEHHXFKQXUPLWZDVVHUJHWDXIWH
UDEHUZLUGHXFKPLWGHPKHLOLJHQJHLVWWDXIHQGLH
WDXIHMHVXLQMHQHQWDJHQNDPMHVXVDXVQDCDUHWLQ

JDOLODHDXQGOLHVVVLFKYRQMRKDQQHVLPMRUGDQWD
XIHQXQGDOVHUDXVGHPZDVVHUVWLHJVDKHUGDVVGHU
KLPPHOVLFKRHIIQHWHXQGGHUJHLVWZLHHLQHWDXEH
DXILKQKHUDENDPXQGHLQHVWLPPHDXVGHPKLPPHOVS
UDFKGXELVWPHLQJHOLEWHUVRKQDQGLUKDEHLFKJH
IDOOHQJHIXQGHQGLHYHUVXFKXQJMHVXGQDFKWULH
EGHUJHLVWMHVXVLQGLHZXHVWHGRUWEOLHEMHVXVYL
HUCLJWDJHODQJXQGZXUGHYRPVDWDQLQYHUVXFKXQJ
JHIXHKUWHUOHEWHEHLGHQZLOGHQWLHUHQXQGGLHHQ
JHOGLHQWHQLKPHUVWHVDXIWUHWHQLQJDOLODHDQDF
KGHPPDQMRKDQQHVLQVJHIDHQJQLVJHZRUIHQKDWWH
JLQJMHVXVZLHGHUQDFKJDOLODHDHUYHUNXHQGHWHG
DVHYDQJHOLXPJRWWHVXQGVSUDFKGLHCHLWLVWHUIX
HOOWGDVUHLFKJRWWHVLVWQDKHNHKUWXPXQGJODXEW
DQGDVHYDQJHOLXPGLHEHUIXIQJGHUHUVWHQMXHQJH
UDOVMHVXVDPVHHYRQJDOLODHDHQWODQJJLQJVDKHU
VLPRQXQGDQGUHDVGHQEUXGHUGHVVLPRQGLHDXIGHP
VHHLKUQHWCDXVZDUIHQVLHZDUHQQDHPOLFKILVFKH
UGDVDJWHHUCXLKQHQNRPPWKHUIROJWPLUQDFKLFKZ
HUGHHXFKCXPHQVFKHQILVFKHUQPDFKHQVRJOHLFKO
LHVVHQVLHLKUHQHWCHOLHJHQXQGIROJWHQLKPDOVH
UHLQVWXHFNZHLWHUJLQJVDKHUMDNREXVGHQVRKQGH
VCHEHGDHXVXQGVHLQHQEUXGHUMRKDQQHVVLHZDUHQ
LPERRWXQGULFKWHWHQLKUHQHWCHKHUVRIRUWULHIH
UVLHXQGVLHOHVVHQLKUHQYDWHUCHEHGDHXVPLWVH
LQHQWDJHORHKQHUQLPERRWCXUXHFNXQGIROJWHQMH
VXVQDFK

[a] Gaius Iulius Caesar, 100-44 v. Chr., um etwa 50 v. Chr.
[b] Markus 1, 1-20, Quelle: http://www.bibleserver.com/text/EU/Markus1

Dagegen bezeichnet die Kryptoanalyse[3] ursprünglich den Prozess aus verschlüsselten Texten ohne Kenntnis des Schlüssels wieder einen lesbaren Text zu erzeugen. Dabei setzen wir voraus, dass zwar die Verfahrensklasse, mit welcher der Text erzeugt wurde, bekannt ist, jedoch nicht der Schlüssel.

Beispiel 2.2

Um aus den nicht lesbaren Texten aus Beispiel 2.1 lesbare zu bekommen, gehen wir davon aus, dass eine Funktion mit Rechts-Shift auf das Großbuchstaben-Alphabet angewandt wurde. Es gibt dabei 26 verschiedene Möglichkeiten eines Rechts-Shifts, die alle ausprobiert werden können. Bei einem genügend langen Text ist es leicht, den richtigen Schlüssel zu finden. Eine weitere Möglichkeit besteht bei dieser Art der Umformung darin, auf Basis der in einer Sprache auftretenden Häufigkeiten der einzelnen Buchstaben Rückschlüsse vom nicht lesbaren Text zu erhalten. So tritt der Buchstabe E in deutschen Texten mit Abstand am Häufigsten auf (danach: N). In unserem Bibeltext, der aus 1811 Zeichen besteht, tritt das H 328 mal und somit mit Abstand am häufigsten auf, gefolgt vom Q (184). Wir vermuten, dass das E zum H

[3] τό αναλύειν

umgeformt wurde. Durch einen Links-Shift um drei Buchstaben (Shift-Wert -3),

$$d : \mathcal{A}_0 \to \mathcal{A}_0,\ A \mapsto X, B \mapsto Y, C \mapsto Z, \ldots, X \mapsto U, Y \mapsto V, Z \mapsto W,$$

bekommen wir den ursprünglichen Text zurück.

Die beiden Abbildungen e und d aus den Beispielen 2.1 und 2.2 dienen dem Sicherheitsdienst der Vertraulichkeit, indem sie die Funktionalitäten der Ver- und Entschlüsselung von Daten implementieren. Es gibt Abbildungen, die Funktionalitäten anderer Sicherheitsdienste abdecken. Alle solche Abbildungen erhalten eine spezielle Bezeichnung.

Definition 2.3: Kryptographische Funktion

Eine kryptographische Funktion ist eine (mathematische) Abbildung zur Implementierung der Funktionalität eines Sicherheitsdienstes.

Damit hat sich der Fokus der Aufgaben der Kryptologie deutlich erweitert und wir definieren die Begriffe in

Definition 2.4: Kryptologie, Kryptographie, Kryptoanalyse

Die Kryptologie ist die Wissenschaft vom Entwickeln (Kryptographie) und Analysieren (Kryptoanalyse) kryptographischer Funktionen.

Heute verstehen wir unter Kryptographie das Entwickeln kryptographischer Funktionen und unter Kryptoanalyse die Analyse kryptographischer Funktionen hinsichtlich ihrer implementierenden Funktionalität, d.h. es wird untersucht, ob eine kryptographische Funktion ihren Zweck der Gewährleistung eines Sicherheitsziels erfüllt. Beide Teilgebiete der Kryptologie gehören zusammen und dürfen nicht isoliert voneinander betrachtet werden.

2.2. Kryptographische Funktionalitäten

Die Funktionalitäten von Sicherheitsdiensten nennen wir kryptographische Funktionalitäten. Allen Funktionalitäten ist gemein, dass es um die Verarbeitung von Daten p geht. Die Daten stammen auf Basis eines Alphabets $\mathcal{A}$ aus einer Menge $\mathcal{P} = \mathcal{P}_{\mathcal{A}}$ der zugelassenen Nachrichten, die wir mit Klartextraum (engl. plain) bezeichnen. Wir unterscheiden fünf kryptographische Funktionalitäten - Chiffrierung, Prüfwerterzeugung, Signaturerzeugung, Schlüsselmanagement und Zufallszahlgenerierung, die wir in den folgenden Kapiteln untersuchen wollen. Einzelne kryptographische Funktionalitäten lassen sich weiter konkretisieren. Beispielsweise besteht die Chiffrierung aus der Ver- und Entschlüsselung von Daten oder das Schlüsselmanagement bedarf der Schlüsselerzeugung und des Schlüsselaustauschs. In Tabelle 2.2 ist zu sehen, welche Funktionalitäten für die jeweiligen Sicherheitsdienste notwendig (✓) sind bzw. hierfür unterstützend sein können (∼). Eine Kommunikation erfolgt zu einem bestimmten Zweck. Kryptographische Funktionalitäten sind daher in der Regel in einen auf diesen Zweck hin ausgerichteten festen Ablauf eingebettet. Der Ablauf einer Kommunikation, die kryptographische Funktionalitäten benutzt, unterliegt einem spezialisierten Kommunikationsprotokoll.

Tabelle 2.1.: Sicherheitsdienste und kryptographische Funktionalitäten in Anlehnung an Abbildung 4.1 in [9]

	Chiffrierung	Prüfwerterzeugung	Signaturerzeugung	Schlüsselmanagement	Zufallszahlengenerierung
Vertraulichkeit	✓			✓	∼
Datenintegrität	∼	✓	∼	∼	∼
Kontrollierter Zugang	∼	∼	∼	✓	∼
Zurechenbarkeit	✓	∼	✓	✓	∼
Privatsphäre	✓			✓	∼

Definition 2.5: Kryptographisches Protokoll

Ein kryptographisches Protokoll ist ein Kommunikationsprotokoll, dessen Ausführung wenigstens ein Sicherheitsziel gewährleisten soll.

2.2.1. Chiffrierung

Bei der Chiffrierung geht es einerseits um die Transformation von lesbaren Daten p (genannt Klartext) aus einem Klartextraum $\mathcal{P}$ mit einem zur Verfügung stehenden Schlüssel k_1 aus einem Schlüsselraum $\mathcal{K}$, in nicht lesbare Daten c (dem Geheimtext) aus einem Geheimtextraum $\mathcal{C}$ (engl. cipher), $e_{k_1} : \mathcal{P} \to \mathcal{C},\ p \mapsto c := e_{k_1}(p)$, genannt Verschlüsselungsfunktion (engl. encrypt). Andererseits soll ein auf diese Weise entstandener Geheimtext mit Hilfe eines Schlüssels $k_2 \in \mathcal{K}$ wieder in einen Klartext zurücktransformiert werden können, $d_{k_2} : \mathcal{C} \to \mathcal{P},\ c \mapsto p := d_{k_2}(c)$, genannt Entschlüsselungsfunktion (engl. decrypt). Klartext-, Schlüssel- und Geheimtextraum sind jeder für sich als Mengen von Symbolen vorstellbar. Jedes solche Symbol ist nach syntaktischer Vorgabe eine Aneinanderreihung von Zeichen eines Alphabets $\mathcal{A}$.

Definition 2.6: Chiffriersystem

Es seien $\mathcal{P}$ ein Klartextraum, $\mathcal{K}$ ein Schlüsselraum und $\mathcal{C}$ ein Geheimtextraum. Mit einer Familie

$$\mathcal{E} := \{e_k : \mathcal{P} \to \mathcal{C};\ k \in \mathcal{K}\} \tag{2.1}$$

von Verschlüsselungsfunktionen und einer Familie

$$\mathcal{D} := \{d_k : \mathcal{C} \to \mathcal{P};\ k \in \mathcal{K}\} \tag{2.2}$$

von Entschlüsselungsfunktionen, jeweils indiziert mit den Schlüsseln aus dem Schlüsselraum $\mathcal{K}$, heißt $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ Chiffriersystem, wenn es für alle $p \in \mathcal{P}$ und zu jedem $k_1 \in \mathcal{K}$ ein $k_2 \in \mathcal{K}$ gibt mit

$$p = d_{k_2}(e_{k_1}(p)). \tag{2.3}$$

Es gibt drei Kategorien von Chiffriersystemen. Bei einem symmetrischen Chiffriersystem gilt stets $k_1 = k_2$ und k_1 wird identisch wie k_2 benutzt (→ Abschnitt 3). Ist $k_1 \neq k_2$ sprechen wir von einem asymmetrischen Chiffriersystem (→ Abschnitt 4). Eine Sonderform stellt das hybride Chiffriersystem dar. Dabei wird ein symmetrisches mit einem asymmetrischen Chiffriersystem vereint (→ Abschnitt 6.4). Chiffriersysteme dienen vornehmlich dem Ziel der Vertraulichkeit der Daten. Aber auch private Kommunikationsverbindungen können unter Zuhilfenahme von Chiffriersystemen erzeugt werden. Bevor wir erste Beispiele für Chiffriersysteme kennenlernen, benötigen wir ein paar Grundlagen aus dem Bereich der Zahlentheorie. Es geht um die modulare Arithmetik, das Rechnen mit Resten auf endlichen Mengen.

Operationen modulo n
Mit Hilfe von Additions- und Multiplikationstabellen, z.B. [4]

$+_3$	0	1	2
0	0	1	2
1	1	2	0
2	2	0	1

$\cdot_3$	0	1	2
0	0	0	0
1	0	1	2
2	0	2	1

lässt sich eine Arithmetik auf endlichen Mengen definieren. Die damit benutzte Arithmetik können wir allgemein fassen, denn sie für jedes n durch derartige Tabellen aufzuschreiben ist zu aufwändig. Dazu benötigen wir Überlegungen zur ganzzahligen Division mit Rest. Eine Division einer ganzen durch eine natürliche Zahl geht oftmals nicht auf. Der übrig bleibende Rest ist dabei eindeutig bestimmt.

Satz und Definition 2.7: Division mit Rest in $\mathbb{Z}$

Seien $a \in \mathbb{Z}$ und $n \in \mathbb{N}$. Dann gibt es eindeutig bestimmte Zahlen $q \in \mathbb{Z}$ und $r \in \{0, 1, \ldots, n-1\}$ so, dass

$$a = q \cdot n + r$$

ist. q heißt der Quotient, r der Rest von a bei Division durch n.

Beweis.

Existenz: Setze $q := \lfloor a/n \rfloor$ und $r := a - q \cdot n$. Mit $q \leq a/n < q + 1$ ist dann $0 \leq r < n$ und $a = q \cdot n + r$.
Eindeutigkeit: Es ist $0 \leq r/n = a/n - q < 1$ und damit $a/n - 1 < q \leq a/n$, also ist q

[4] gemäß der Beispiele A.5 und A.8

eindeutig bestimmt.

□

Damit sind die Reste r der Division von a durch n (Sprechweise: Reste modulo n) eindeutig festgelegt und werden mit $r = a \bmod n$ bezeichnet. Gilt $r = 0$, so ist $a = q \cdot n$.

Definition 2.8: Teiler

Seien $a \in \mathbb{Z}$ und $n \in \mathbb{N}$. Wir schreiben

$$a \mid n \ :\Longleftrightarrow\ \exists q \in \mathbb{Z} : n = q \cdot a.$$

und sagen „a teilt n“. a heißt Teiler von n.

Wir fassen alle Teiler von n in den natürlichen Zahlen zur Menge

$$D_n := \{a \in \mathbb{N};\ a \mid n\}$$

zusammen. Es gilt $1 \in D_n$, da für jedes beliebige $n \in \mathbb{N}$ gilt $n = n \cdot 1$, also $1 \mid n$. Weiter teilt wegen $n = 1 \cdot n$ jede natürliche Zahl sich selbst, so dass D_n wenigstens die beiden Elemente 1 und n enthält.

Definition 2.9: Primzahl

Eine natürliche Zahl $n \in \mathbb{N}$ heißt Primzahl bzw. prim, falls $|D_n| = 2$.

Bemerkung.
(1) 1 ist keine Primzahl, da $D_1 = \{1\}$ und $|D_1| = 1 \neq 2$.
(2) Ein Teiler von n, der zugleich eine Primzahl ist, wird Primteiler genannt.
(3) Für die Elemente $a \in D_n$ gilt $1 \leq a \leq n$.

Beispiel 2.10

Für $n = 120$ bestimmen wir $D_{120} = \{1, 2, 3, 4, 5, 6, 8, 10, 12, 15, 20, 24, 30, 40, 60, 120\}$. Dabei sind 2,3 und 5 Primteiler. Andererseits ist $D_7 = \{1, 7\}$ und damit ist 7 eine Primzahl.

Betrachten wir zwei oder mehr ganze und von der Null verschiedene Zahlen $n_1, \ldots, n_k \in \mathbb{Z}\backslash\{0\}$, so kann das Teiler-Konzept ohne Probleme auf $\mathbb{Z}\backslash\{0\}$ erweitert werden. Die Zahlen besitzen wenigstens einen gemeinsamen Teiler: die Eins. Die Menge der gemeinsamen Teiler ist

$$D_{n_1} \cap D_{n_2} \cap \ldots \cap D_{n_k}.$$

Da die Menge aller Teiler jeder Zahl $n_1, \ldots, n_k$ endlich ist, gibt es in der Menge der gemeinsamen Teiler stets ein größtes Element.

Definition 2.11: Größter gemeinsamer Teiler

Seien $n_1, n_2, \ldots, n_k \in \mathbb{Z} \setminus \{0\}$. Dann heißt

$$\text{ggT}(n_1, n_2, \ldots, n_k) := \max D_{n_1} \cap D_{n_2} \cap \ldots \cap D_{n_k}$$

der größte gemeinsame Teiler der Zahlen $n_1, n_2, \ldots, n_k$.

Bemerkung.
Ist n eine Primzahl, so gilt für alle $a \in \{0, \ldots, n-1\}$ stets $\text{ggT}(a, n) = 1$.

Es gibt Zahlen in $\mathbb{Z}$, die bei Division durch n den gleichen Rest lassen. Diese Relation zwischen den Zahlen liefert

Definition 2.12: Kongruenz modulo n

Seien $a, b \in \mathbb{Z},\ n \in \mathbb{N}$ mit $a = q_1 n + r_1$ und $b = q_2 n + r_2$. a heißt kongruent b modulo n,

$$a \equiv_n b \text{ bzw. } a \equiv b \ (\text{mod } n),$$

wenn $r_1 = r_2$ gilt.

Bemerkung.
Die Definition ist äquivalent dazu, dass n die Zahl $a - b$ teilt, $n \mid (a - b)$, und dass a und b den selben Rest modulo n lassen (d.h. a mod n = b mod n)[5].

Auf der Menge $\mathbb{Z}_n := \{0, \ldots, n-1\}$ der Reste modulo n kann eine Addition und Multiplikation definiert werden.

Definition 2.13: mod-n-Addition und -Multiplikation

Für ein $n \in \mathbb{N}$ sei $\mathbb{Z}_n := \{0, \ldots, n-1\}$ die Menge der Reste modulo n. Auf $\mathbb{Z}_n$ definieren wir für $a, b \in \mathbb{Z}_n$ zwei innere zweistellige Verknüpfungen $+_n : \mathbb{Z}_n \times \mathbb{Z}_n \to \mathbb{Z}_n$ und $\cdot_n : \mathbb{Z}_n \times \mathbb{Z}_n \to \mathbb{Z}_n$ durch

$$\begin{aligned} a +_n b &:= (a + b) \text{ mod n}, \text{ (Addition modulo n)}, \\ a \cdot_n b &:= (a \cdot b) \text{ mod n}, \text{ (Multiplikation modulo n)}. \end{aligned}$$

Bemerkung.
(1) Wir schreiben a^2 mod $n := a \cdot_n a$ und entsprechend[6] für ein $k \in \mathbb{N}$

$$a^k \text{ mod } n := \underbrace{a \cdot_n \ldots \cdot_n a}_{k-\text{mal}}.$$

(2) Die Menge $\mathbb{Z}_2 = \{0, 1\}$ wird oft mit $\mathbb{F}_2$ bezeichnet. $\mathbb{F}_2^*$ enthält Binärfolgen beliebiger Länge.

[5] vgl. die Sätze A.31 und A.32
[6] vgl. Satz und Definition A.10

Einfache Chiffriersysteme
Mit diesen Vorbereitungen können wir zwei Beispiele für Chiffriersysteme betrachten. Dabei benutzen wir die Tatsache, dass Berechnungen von $\mathbb{Z}_n$ auf $\mathbb{Z}$ ausgelagert werden können, d.h. Berechnungen können in $\mathbb{Z}$ durchgeführt, müssen jedoch durch eine modulo Operation abgeschlossen werden. Dass dies funktioniert, wird in Satz A.36 gezeigt.

Beispiel 2.14

Es seien $\mathcal{P} = \mathcal{C} = \mathcal{K} := \mathbb{Z}_{26}$ und

$$e_k : \mathcal{P} \to \mathcal{C},\ p \mapsto e_k(p) := p + k \bmod 26,$$
$$d_k : \mathcal{C} \to \mathcal{P},\ c \mapsto d_k(c) := c - k \bmod 26.$$

Dann gilt $d_k(e_k(p)) = ((p + k) - k) \bmod 26 = p$ und damit liegt ein symmetrisches Chiffriersystem vor. Es entspricht der Caesar-Chiffre der Beispiele 2.1 und 2.2, indem jeder Buchstabe reihenfolgetreu durch eine Zahl kodiert wird.

Das symmetrische Chiffriersystem im letzten Beispiel 2.14 nutzt die Addition modulo n. Folgendes Beispiel zeigt ein asymmetrisches Chiffriersystem, welches die Multiplikation modulo n benutzt. Es ist so konstruiert, dass n das Produkt der beiden Primzahlen 3 und 11 ist.

Beispiel 2.15: RSA-light

Es seien $\mathcal{P} = \mathcal{C} := \mathbb{Z}_{33}$, $\mathcal{K} := \{33\} \times \{3, 7, 9, 11, 13, 17, 19, 21, 23, 27, 29, 31\}$ und mit $(33, k_1), (33, k_2) \in \mathcal{K}$

$$e_{(33,k_1)} : \mathcal{P} \to \mathcal{C},\ p \mapsto e_{(33,k_1)}(p) := p^{k_1} \bmod 33,$$
$$d_{(33,k_2)} : \mathcal{C} \to \mathcal{P},\ c \mapsto d_{(33,k_2)}(c) := c^{k_2} \bmod 33.$$

Wir werden in Abschnitt 4 zeigen, dass in dieser Situation für gewisse Kombinationen von k_1 und k_2

$$d_{(33,k_2)}\left(e_{(33,k_1)}(p)\right) = p^{k_1 \cdot k_2} \bmod 33 = p$$

ist. Damit liegt aufgrund verschiedener Schlüssel ein asymmetrisches Chiffriersystem vor. Der zur Verschlüsselung benötigte Schlüsselanteil $(33, k_1)$ ist der öffentliche Schlüssel, zur Entschlüsselung wird der private Schlüssel $(33, k_2)$ benutzt. Das vorliegende System entspricht den Ideen des RSA-Verfahrens.

2.2.2. Prüfwerterzeugung

Bei der Datenintegrität geht es um die Vollständigkeit und Unverfälschtheit der Daten. Sie dürfen nicht unzulässig oder unautorisiert verändert oder gelöscht worden sein. Die Integrität eines Objekts lässt sich mit Hilfe eines Prüfwerts nachweisen. Dabei wird ein gegebener Wert mit einem mit Hilfe des Objektes erzeugten Wert verglichen.

Definition 2.16: Prüfwertabbildung, Prüfwert

Sei $\mathcal{A}$ ein beliebiges Alphabet. Eine Prüfwertabbildung ist eine Abbildung $h : \mathcal{P} \to \mathcal{A}^n$, die Daten beliebiger Länge Daten mit einer festgelegten Länge $n \in \mathbb{N}$, den Prüfwert, zuordnet.

Die Länge eines Prüfwerts ist in der Regel kürzer als die Länge der Daten. Kleine Änderungen an den Daten wie etwa Vertauschungen zweier oder Änderungen eines Zeichens sollen völlig andere Prüfwerte erzeugen.

ISBN-10-Code
Ein Buch lässt sich durch eine Folge

$$z_1 - z_2z_3z_4z_5 - z_6z_7z_8z_9 - z_{10},$$

der so genannten internationalen Standardbuchnummer identifizieren. Dabei gilt $z_i \in \mathbb{Z}_{11}$ für $i \in \{1, \ldots, 10\}$ und anstelle von 10 wird X geschrieben. z_1 steht für das Land, $z_2z_3z_4z_5$ für den Verlag, $z_6z_7z_8z_9$ für die Titelnummer und z_{10} ist eine Prüfziffer (Prüfwert). Die Prüfziffer ist durch die ersten neun Ziffern eindeutig festgelegt durch die Prüfwertabbildung

$$z_{10} := \sum_{i=1}^{9} i \cdot z_i \bmod 11.$$

Beispiel 2.17

Das Buch [3] besitzt die ISBN-10 3-540-41283-2. Wegen

$$2 \equiv_{11} (\underbrace{1 \cdot 3 + 2 \cdot 5 + 3 \cdot 4 + 4 \cdot 0 + 5 \cdot 4 + 6 \cdot 1 + 7 \cdot 2 + 8 \cdot 8 + 9 \cdot 3}_{156})$$

gilt $z_{10} = 2$ und die Prüfziffer ist korrekt.

Sei $c := (z_1, \ldots, z_{10}) \in \mathbb{Z}_{11}^{10}$ und $h : \mathbb{Z}_{11}^{10} \to \mathbb{Z}_{11}$, $c \mapsto h(c) := \sum_{i=1}^{10} i \cdot z_i \bmod 11$. Dann gilt

$$\begin{aligned} h(c) &= \sum_{i=1}^{10} i \cdot z_i \bmod 11 \\ &= \left(\sum_{i=1}^{9} i \cdot z_i + 10 \cdot z_{10} \right) \bmod 11 \\ &= 11 \cdot z_{10} \bmod 11 = 0. \end{aligned}$$

Der ISBN-10-Code wird nun definiert als die Menge C aller 10-Tupel $c \in \mathbb{Z}_{11}^{10}$ mit $h(c) = 0$. Was passiert nun, wenn die ISBN-10 durch Vertauschung oder Änderung einer Ziffer verändert wird?

Satz 2.18

Seien C der ISBN-Code, $z \in C$ und $x, y \in \mathbb{Z}_{11}^{10}$ mit $x \neq z$ und $y \neq z$. Angenommen, x

geht aus z durch Ändern von genau einer Ziffer bzw. y geht aus z durch Vertauschen zweier benachbarter Ziffern hervor. Dann erfüllen x und y die Gleichungen $h(x) = 0$ bzw. $h(y) = 0$ nicht und sind keine Elemente von C.

Beweis.

Seien x und z an der i-ten Stelle verschieden und ansonsten gleich. Dann folgt

$$h(x) = (h(x) - h(z)) \bmod 11 = i \cdot (x_i - z_i) \bmod 11 \neq 0,$$

da mit $i \neq 0$ und $x_i - z_i \neq 0$ auch das Produkt wegen der Körpereigenschaft von $\mathbb{Z}_{11}$[a] nicht Null sein kann. Damit ist x kein ISBN-10-Code. Sei nun $y_i = z_{i+1}$ und $y_{i+1} = z_i$. Dann gilt

$$\begin{aligned} h(y) &= (h(y) - h(z)) \bmod 11 \\ &= (i(y_i - z_i) + (i+1)(y_{i+1} - z_{i+1})) \bmod 11 \\ &= (i(z_{i+1} - z_i) + (i+1)(z_i - z_{i+1})) \bmod 11 \\ &= (z_i - z_{i+1}) \bmod 11 \\ &= (z_i - y_i) \bmod 11 \neq 0, \end{aligned}$$

da $y_i \neq z_i$ ist. Somit ist y ebenfalls kein ISBN-10-Code.

□

[a] $\cdot_{11}$ ist abgeschlossen in $\mathbb{Z}_{11}$, für die Körpereigenschaft siehe Satz A.25

Da die ersten neun Ziffern frei wählbar sind, gibt es genau 11^9 verschiedene ISBN-10-Codes. Die Prüfwertabbildung $z_{10} : \mathbb{Z}_{11}^9 \to \mathbb{Z}_{11}$ ist nicht injektiv. Eine Vielzahl an ISBN-10-Codes kollidieren bzgl. der Prüfziffer. So hat etwa auch das Buch [1] die Prüfziffer 2 (ISBN-10: 0-387-27934-2) wie [3] aus Beispiel 2.17. Im Hinblick auf die Integrität einer ISBN-10 ist das nicht wünschenswert. Denn hier lässt sich sehr einfach eine falsche und dennoch sinnvolle ISBN-10 mit korrekter Prüfziffer erzeugen.

MD5-Prüfsumme in NextCloud

Das Beispiel der ISBN-10-Codes zeigt das grundsätzliche Prinzip der Prüfwerterzeugung. Während kleine Änderungen an der ISBN-10 günstigerweise zu keinen gültigen ISBN-10-Codes führen, weist die Prüfwertbildung auf der anderen Seite jedoch gravierende Nachteile auf. Zum einen ist für die Prüfwerterzeugung ein Eingabetupel fester Länge vorgesehen. Zum anderen ist der Bildbereich der Abbildung zu klein, so dass viele Bücher dieselbe Prüfsumme besitzen. Am Beispiel der Prüfwertbildung des Cloud-Dienstes NextCloud sollen diese Schwierigkeiten zumindest abgeschwächt werden. NextCloud ist ein File-Sharing- und File-Synchronisationsdienst. Zu jeder abgelegten Datei wird ein Prüfwert, die MD5-Prüfsumme, erzeugt (siehe Abbildung 2.1, im roten Rahmen). Wird nun eine Datei heruntergeladen, so kann von ihrem Bytecode die MD5-Prüfsumme erzeugt und mit der auf dem NextCloud-Server vorliegenden MD5-Prüfsumme verglichen werden. Ergeben sich Unterschiede, so muss es beim Herunterladen irgendwelche Schwierigkeiten gegeben haben und die Datei sollte noch einmal heruntergeladen werden.

Abbildung 2.1.: Prüfsumme einer Datei in NextCloud.

Zur Erzeugung der MD5-Prüfsumme wird eine Bitfolge beliebiger Länge als Eingabe benötigt. Als Ausgabe ergibt sich eine Folge von 128 Bits,

$$\text{md5} : \mathbb{F}_2^* \to \mathbb{F}_2^{128}.$$

Auch diese Prüfwerterzeugung ist nicht injektiv, d.h. es gibt Dateien, welche dieselbe MD5-Prüfsumme besitzen. Obwohl der Algorithmus der Erzeugung der MD5-Prüfsumme offen und bekannt ist, ist es ungleich schwerer von einer Prüfsumme ausgehend eine Datei zu erzeugen, welche diese Prüfsumme liefert. Dennoch ist es gelungen zu zeigen, dass die MD5-Prüfwerterzeugung angreifbar ist und es wird empfohlen, diese nicht mehr zu verwenden[7].

Prüfwerte wie die MD5-Prüfsumme werden Hashwerte (→ Abschnitt 8.1) genannt. Es gibt weiter die Möglichkeit, Prüfsummen auf Basis symmetrischer Chiffriersysteme zu erzeugen. Solche Prüfwerte heißen Message Authentication Codes (→ Abschnitt 8.2). Eine Kombination beider Prüfwerteerzeugungen führt zu einer hybriden Form, den Hashbasierenden Message Authentication Codes (→ Abschnitt 8.2).

2.2.3. Signaturerzeugung

Eine (digitale) Signatur ist ein schlüsselbasierter Prüfwert, mit dem die Urheberschaft von Daten belegt werden kann. Mit einem nur der Quelle der Daten bekannten Schlüssel wird eine Signatur erzeugt. Mit einem öffentlichen Schlüssel kann die Signatur auf ihre Echtheit hin überprüft werden. Die Daten können dem Urheber eindeutig zugewiesen werden. Umgekehrt kann der Urheber die Herkunft der Daten nicht abstreiten.

Wir unterscheiden zwei Arten von Signaturverfahren, die sich allerdings vom Grundprinzip nicht unterscheiden. Bei beiden werden sowohl die Daten als auch das, was wir als Signatur bezeichnen, übermittelt. Bei der digitalen Signatur mit Nachrichtenrückgewinnung (→ Abschnitt 9.1) besitzt die Signatur dieselbe Länge wie die Daten und es werden während des Überprüfungsvorganges die Daten zurückgewonnen. Stimmen diese überein, wird die Echtheit akzeptiert. Beim zweiten Verfahren ist die Signatur ein hashbasierter Prüfwert, dessen Länge in aller Regel bedeutend kürzer als die Datenlänge ist (→ Abschnitt 9.2). Aus der erzeugten Signatur lassen sich die Daten nicht zurückgewinnen, sondern lediglich der Hashwert. Dieser wird dann mit dem Hashwert der empfangenen Daten verglichen.

Beispiel 2.19

Betrachten wir das asymmetrische Chiffriersystem aus Beispiel 2.15. Wird anstelle des öffentlichen Schlüssels der private, den nur eine einzige Instanz kennt, zur „Verschlüsselung“ verwendet, so ist nach der „Entschlüsselung“ und Überprüfung davon

[7] vgl. http://www.mscs.dal.ca/~selinger/md5collision/

auszugehen, dass die Daten von der Instanz mit dem privaten Schlüssel stammen. Sie sind eindeutig zurechenbar.

2.2.4. Schlüsselmanagement und Zufallszahlengenerierung

Wir haben bereits gesehen, dass für die meisten genannten Funktionalitäten Schlüssel verwendet werden. Diese Schlüssel müssen erzeugt, verteilt, gespeichert, manchmal wiederhergestellt, für ungültig erklärt und endgültig gelöscht werden. Bei der Erzeugung ist darauf zu achten, dass kein rein deterministischer Vorgang erfolgt, der sonst für jedermann nachvollziehbar ist. Eine pseudozufallsgesteuerte Schlüsselerzeugung liegt vor, wenn die Schlüssel zwar deterministisch erzeugt, alle möglichen Schlüssel aber gleichwahrscheinlich sind. Die Verfahren nutzen meist einen zufälligen Initialwert. Bei symmetrischen Chiffriersystemen können Zufallszahlen unmittelbar benutzt werden, während es bei asymmetrischen Chiffriersystemen weiterer Bearbeitungsschritte bedarf, etwa um große Primzahlen zu erzeugen. Mit der Generierung von (Pseudo-)Zufallszahlen beschäftigen wir uns in Abschnitt 7. Sind Schlüssel erzeugt, müssen sie entsprechend an die an der Kommunikation beteiligten Instanzen verteilt werden. Die Verteilung kann wiederum über eine Kommunikation erfolgen. Da Schlüssel für symmetrische Chiffriersysteme vertraulich zu behandeln sind, kann dafür eine verschlüsselte Kommunikation mit einem weiteren Schlüssel erforderlich sein. Dieser Schlüssel ist jenem Schlüssel, der übertragen werden soll, als übergeordnet anzusehen. Dagegen können öffentliche Schlüssel für asymmetrische Chiffriersysteme frei zugänglich sein. Das Schlüsselmanagement beschäftigt uns in Kapitel 5.

2.3. Kryptoanalyse

Eine der Hauptaufgaben der Kryptoanalyse besteht darin, aus Geheimtexten den Klartext zu gewinnen. Der Kryptographie liegt das Prinzip von Kerckhoff[8], seinem zweiten von sechs Grundsätzen, zugrunde:

> „Die Sicherheit eines Kryptosystems darf nicht von der Geheimhaltung des Algorithmus abhängen. Die Sicherheit gründet sich nur auf die Geheimhaltung des Schlüssels."

Unter der Annahme, dass eine kryptographische Funktion bekannt ist, liegt es an der Güte des Schlüssels, ob aus dem Geheimtext der Klartext rekonstruiert werden kann. Je einfacher das klappt, desto schwächer ist die kryptographische Funktion einzuordnen.

Brute-Force-Angriff
Ein erstes Kriterium der Güte einer kryptographischen Funktion besteht in der Mächtigkeit des Schlüsselraums. Bei einem Brute-Force-Angriff werden alle Schlüssel nacheinander ausprobiert. Mit den heutigen Rechenleistungen ist das bei der Caesar-Chiffre kein Problem, der Klartext ist sofort wiederhergestellt. Liefert ein Schlüssel ein sinnvolles Resultat, so kann davon ausgegangen werden, dass der richtige Schlüssel gefunden wurde.
Eine Verallgemeinerung der Caesar-Chiffre aus Beispiel 2.14 ist die Verschiebungschiffre, bei der nicht um den festen Shift-Wert 3, sondern um einen beliebigen Wert $k \in \mathbb{Z}_{26}$

[8] A. Kerckhoff, 1835-1903, in La cryptographie militaire, Journal des sciences militaires. Bd. 9, S. 5–38 (Jan. 1883), S. 161–191 (Feb. 1883).

verschoben wird. Soll nun der Geheimtext „YPCLY“ ohne Kenntnis des Schlüssels entschlüsselt werden, können wir alle möglichen Klartextkandidaten aufzählen:

„YPCLY“, „XOBKX“, „WNAJW“, „VMZIV“,
„ULYHU“, „TKXGT“, „SJWFS“, „**RIVER**“,
„QHUDQ“, „PGTCP“, „OFSBO“, „NERAN“,
„MDQZM“, „LCPYL“, „KBOXK“, „JANWJ“,
„IZMVI“, „HYLUH“, „GXKTG“, „FWJSF“,
„EVIRE“, „DUHQD“, „CTGPC“, „BSFOB“,
„**ARENA**“, „ZQDMZ“.

Davon sind zwei sinnvolle Kandidaten: „RIVER“ und „ARENA“. Da nur genau ein Kandidat die richtige Lösung ist, ist einer der beiden Schlüssel falsch. Solche Schlüssel werden spurious key genannt. Sei $\mathcal{K}$ der endliche Schlüsselraum. Wir betrachten zunächst ein Zufallsexperiment, das die mittlere Anzahl an Versuchen modelliert. Dabei bezeichnet $\mathbb{P}(\{v\})$ die Wahrscheinlichkeit dafür, genau v Versuche durchzuführen, d.h. die ersten $v-1$ Versuche waren Fehlversuche. Jedes $v \in \{1,\ldots,|\mathcal{K}|\}$ sollte gleichwahrscheinlich sein. Um das zu sehen, betrachten wir die Wahrscheinlichkeit, exakt im Schritt v nach $v-1$ Fehlversuchen den richtigen Schlüssel zu erhalten, wobei die Wahrscheinlichkeit im ersten Versuch Erfolg zu haben gleich

$$p_1 = \frac{1}{|\mathcal{K}|} = \frac{1}{|\mathcal{K}|+1-1} \tag{2.4}$$

sei und der Wert im Nenner in jedem Schritt um Eins verkleinert werde. Im Schritt v haben wir dann

$$p_v = \frac{1}{|\mathcal{K}|+1-v}.$$

Die Wahrscheinlichkeit, exakt im Schritt v nach $v-1$ Fehlversuchen den richtigen Schlüssel zu erhalten ist dann

$$\begin{aligned}
\mathbb{P}(\{v\}) &= (1-p_1)\cdot(1-p_2)\cdot\ldots\cdot(1-p_{v-1})\cdot p_v \\
&= \left(1-\frac{1}{|\mathcal{K}|}\right)\cdot\left(1-\frac{1}{|\mathcal{K}|-1}\right)\cdot\ldots\cdot\left(1-\frac{1}{|\mathcal{K}|+1-(v-1)}\right) \\
&\quad\cdot\frac{1}{|\mathcal{K}|+1-v} \\
&= \left(\frac{|\mathcal{K}|-1}{|\mathcal{K}|}\right)\cdot\left(\frac{|\mathcal{K}|-2}{|\mathcal{K}|-1}\right)\cdot\ldots\cdot\left(\frac{|\mathcal{K}|+1-v}{|\mathcal{K}|+2-v}\right)\cdot\frac{1}{|\mathcal{K}|+1-v} \\
&= \frac{1}{|\mathcal{K}|}.
\end{aligned} \tag{2.5}$$

Der erwartete Wert für V der dazugehörigen Zufallsvariablen V mit Werten in $\{1,\ldots,|\mathcal{K}|\}$ beantwortet die Frage, wie viele Versuche im Mittel notwendig sind. Mit $\mathbb{P}(\{v\}) = \frac{1}{|\mathcal{K}|}$ für $v \in \{1,\ldots,|\mathcal{K}|\}$ ergibt sich der Erwartungswert zu

$$\mathbb{E}[V] = \sum_{v=1}^{|\mathcal{K}|} v\cdot\mathbb{P}(\{v\}) = \frac{|\mathcal{K}|(|\mathcal{K}|+1)}{2|\mathcal{K}|} = \frac{|\mathcal{K}|+1}{2}.$$

Damit kann festgehalten werden, dass eine große Anzahl verschiedener Schlüssel grundsätzlich sehr nützlich ist. Jedoch genügt das nicht allein, um Sicherheit zu gewährleisten.

Known-Ciphertext-Angriff
Durch geschicktes Ausprobieren von Schlüsseln kann die Anzahl Versuche bei der Klartextsuche reduziert werden. So haben wir durch die Häufigkeitsanalyse bei der Caesar-Chiffre in Beispiel 2.2 bereits nach einem Versuch den richtigen Schlüssel gefunden und konnten die Bibelstelle rekonstruieren. Bei Known-Ciphertext-Angriffen wird angenommen, dass verschiedene Klartexte $p_1, p_2, \ldots$ mit dem selben Schlüssel k zu entsprechenden Geheimtexten $c_1 = e_k(p_1), c_2 = e_k(p_2), \ldots$ verschlüsselt wurden. Zudem wird vermutet, dass sich Informationen aus den Klartexten auch in den Geheimtexten finden lassen (wie die Buchstabenhäufigkeiten).

Known-Plaintext-Angriff
Manchmal sind einem Angreifer nicht nur Geheimtexte sondern einige Klartext-Geheimtext-paare $(p_1, c_1),\ (p_2, c_2), \ldots, (p_n, c_n)$, die mit dem gleichen Schlüssel erzeugt wurden, bekannt und er möchte aus diesen den Schlüssel oder einen Klartext p_{n+1} aus einem Geheimtext c_{n+1} gewinnen. Ein wichtiger Vertreter eines solchen Known-Plaintext-Angriffs ist die lineare Kryptoanalyse, mit der wir uns bei der Betrachtung der kryptographische Funktion in einer affin-linearen Blockchiffre beschäftigen.

Chosen-Plaintext-Angriff
Bei einem Chosen-Plaintext-Angriff stehen dem Angreifer nicht nur Klartext-Geheimtext-Paare wie beim Known-Plaintext-Angriff zur Verfügung, sondern es ist darüber hinaus möglich, Klartexte selber zu wählen und die dazu von einem Schlüssel $k \in \mathcal{K}$ erzeugten Geheimtexte bestimmen zu lassen. Dadurch ist es möglich in den kryptographischen Funktionen nach bestimmten Mustern zu suchen und diese in die Klartexte zu übertragen. Diese Idee wird bei der differentiellen Kryptoanalyse angewandt. Erfolgreiche Angriffe gab es auf die Prüfwerteverfahren von MD4 bzw. MD5, aber auch beim symmetrischen Verfahren FEAL[9].

Chosen-Ciphertext-Angriff
Im Unterschied zum Chosen-Plaintext-Angriff werden bei der Klasse der Chosen-Ciphertext-Angriffe nicht Klartexte sondern Geheimtexte gewählt und die entsprechenden Klartexte müssen erst beschafft werden, was sicher mehr Aufwand bedeutet als im umgekehrten Fall. Auch hier können wiederum Muster in den Geheimtexten benutzt werden.

Related-Key-Angriff
Lassen sich Muster bei der Wahl der Schlüssel ausnutzen, um aus Geheimtexten Klartexte bzw. den gesuchten Schlüssel zu identifizieren, wird das Related-Key-Angriff genannt. Ein bekanntes Beispiel für einen solchen Angriff ist der auf das vor einigen Jahren noch vielfach eingesetzte WLAN-Standardprotokoll WEP. Darin wird die kryptographische Funktion RC4[10] benutzt, die mit dieser Methode erfolgreich angegriffen wurde.

Kryptoanalyse asymmetrischer Verfahren
Die bisher vorgestellten Ideen der Kryptoanalyse befassen sich mit kryptographischen Funktionen, die im wesentlichen auf symmetrischen Verfahren basieren. Asymmetrische kryptographische Funktionen müssen sich dagegen auf die verwendeten mathematischen

[9] vgl. [10], S.170 ff.
[10] vgl. [10], S.103 ff.

Funktionen, die sich aus zahlentheoretischen Überlegungen ergeben (so genannte Einwegfunktionen), befassen. Während bei den bisherigen Angriffen stets ein einzelner Schlüssel betrachtet wurde, steht hierbei das gesamte Verfahren auf dem Prüfstand.

Aufwand kryptographischer Analysen
Rechnergestützte Kryptoanalyse zu betreiben ist mit Kosten verbunden. Sowohl Speicherplatz als auch Rechenleistung werden benötigt. Bei einer vorgegebenen Rechenleistung kann abgeschätzt werden, wie lange ein Angriff in der schlechtesten Variante eines Brute-Force-Angriffs dauert. Die Tabelle 2.3 zeigt hierzu eine Übersicht. Die Anzahl Entschlüsselungen je Mikrosekunde beziehen sich auf das DES-Verfahren.[11] Zum Vergleich: Es gibt

Tabelle 2.2.: Durchschnittlicher Zeitaufwand eines Brute-Force-Angriffs auf DES

Schlüssellänge [bit]	Anzahl Schlüssel	Zeitbedarf bei 1 Entschlüsselung je µs	Zeitbedarf bei 10^6 Entschlüsselungen je µs	Zeitbedarf bei 10^9 Entschlüsselungen je µs
32	$4.3 \cdot 10^9$	35.8 min	2.1 ms	2.1 µs
56	$7.2 \cdot 10^{16}$	1142 a	10 h	36 s
64	$1.8 \cdot 10^{19}$	$2.9 \cdot 10^5$ a	107 d	2.5 h
128	$3.4 \cdot 10^{38}$	$5.4 \cdot 10^{24}$ a	$5.4 \cdot 10^{18}$ a	$5.4 \cdot 10^{15}$ a
256	$1.2 \cdot 10^{77}$	$1.8 \cdot 10^{63}$ a	$1.8 \cdot 10^{57}$ a	$1.8 \cdot 10^{54}$ a
512	$1.3 \cdot 10^{154}$	$2.1 \cdot 10^{140}$ a	$2.1 \cdot 10^{134}$ a	$2.1 \cdot 10^{131}$ a

schätzungsweise $8.4 \cdot 10^{77}$ Elektronen im Universum bzw. seit der Entstehung des Sonnensystems sind schätzungsweise $2 \cdot 10^{17}$ Sekunden vergangen. Die Zeit für das Finden eines 128 bit langen Schlüssels übersteigt mit dieser Rechnung bereits das Alter unseres Sonnensystems. in dieser Hinsicht kann ein Brute-Force-Angriff als nicht praktikabel angesehen werden.

[11] vgl. [7], S. 25f., DES wird in Abschnitt 3.3 beschrieben

2.4. Kodierung von Daten

Die Übertragung von Daten erfolgt bei einer Kommunikation in Form von Bits $b \in \mathbb{F}_2 := \{0, 1\}$. Kryptographische Funktionen arbeiten mit binären, hexadezimalen oder dezimalen Zahlen - allgemein Zahlenfolgen. Sämtliche Daten, die nicht als Zahlenfolge vorliegen, wie etwa der Bibeltext aus Beispiel 2.1, müssen zunächst transformiert werden. Dies wird Kodierung genannt. Dazu muss es für jedes einzelne Zeichen $a \in \mathcal{A}$ eines Alphabets $\mathcal{A}$ eine eindeutige Zuordnung, hier beispielsweise binär

$$\mathcal{A} \to \mathbb{F}_2^n \text{ für ein geeignetes } n \in \mathbb{N},$$

zu einer entsprechenden Zahl geben.

2.4.1. Kodierung durch Zahlenalphabete

Um Kryptographie betreiben zu können, werden wir Daten, die nicht als Zahlenfolge dargestellt sind, durch Zahlen kodieren. Wir brauchen deshalb eine injektive Abbildung f zwischen zwei Alphabeten (das zweite Alphabet besteht aus Zahlen). Besteht das Bild von f aus Dezimalzahlen, wird dafür meist die Menge $\mathbb{Z}_n = \{0, \ldots, n-1\}$, oft mit $n := |\mathcal{A}|$, zugrunde gelegt.

Beispiel 2.20

Wir betrachten folgende Nachricht:

„SICHERHEITISTDASEINZIGEZIELDERKRYPTOGRAPHIE“

Das Alphabet für den Klartext sei das Alphabet der Großbuchstaben

$$\mathcal{A}_0 := \{A, B, C, D, E, F, G, H, I \ldots, U, V, W, X, Y, Z\}$$

mit $|\mathcal{A}_0| = 26$. Wir kodieren jeden Buchstaben durch eine entsprechende Zahl aus dem Alphabet „$\mathbb{Z}_{26}$“ der ersten 26 nicht-negativen ganzen Zahlen

$$\mathbb{Z}_{26} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, \ldots, 20, 21, 22, 23, 24, 25\}$$

mittels der bijektiven Abbildung $f : \mathcal{A}_0 \to \mathbb{Z}_{26}$ und $f(A) = 0,\ f(B) = 1, \ldots, f(Z) = 25$. Aus dem Klartext entsteht (zur besseren Unterscheidung der Zahlen in Tupelschreibweise)

$$\begin{pmatrix} 18, 8, 2, 7, 4, 17, 7, 4, 8, 19, 8, 18, 19, 3, 0, 18, 4, 8, 13, 25, 8, 6, \\ 4, 25, 8, 4, 11, 3, 4, 17, 10, 17, 24, 15, 19, 14, 6, 17, 0, 15, 7, 8, 4 \end{pmatrix}$$

f ist eine Abbildung auf dezimale Zahlen. Wir werden bei symmetrischen Chiffriersystemen binär kodierte Zeichenfolgen mit dem Alphabet $\mathbb{F}_2$ verwenden. Dazu ist es oft notwendig, Zahlen aus einem Zahlensystem in Zahlen eines anderen Zahlensystems umzurechnen. Eine Zahl $x = x_{n-1}x_{n-2}\ldots x_0$ mit n Ziffern $x_i \in \mathbb{Z}_p$ aus einem so genannten p-adischen System lässt sich (dezimal) darstellen als

$$x = \sum_{i=0}^{n-1} x_i \cdot p^i = x_{n-1} \cdot p^{n-1} + x_{n-2} \cdot p^{n-2} \cdot \ldots \cdot x_0 \cdot p^0. \tag{2.6}$$

Wenn nicht klar ist, aus welchem Zahlensystem a eine Zahl x ist, schreiben wir x_a, um das eindeutig zuzuweisen. Für die Dezimalzahl 20 können wir schreiben $20_{10} = 2 \cdot 10^1 + 0 \cdot 10^0$. Um Zahlen x_a vom Zahlensystem a in eine Zahl z_b des Zahlensystems b umzurechnen, gehen wir den Weg über das Dezimalsystem.

Beispiel 2.21

Die hexadezimale Zahl 14_{16} lautet im Binärsystem 10100_2:

x_{16}	$16 \to 10$	y_{10}	$10 \to 2$	z_2
14_{16}	$(1 \cdot 16^1 + 4 \cdot 16^0)_{10}$	20_{10}	$(1 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 0 \cdot 2^0)_{10}$	10100_2

Wir können so mit Hilfe der Abbildung f aus Beispiel 2.20 auch Buchstaben aus dem Alphabet $\mathcal{A}_0$ binär kodieren:

$x \in \mathcal{A}_0$	$f(x)_{10}$	y_{10}	$10 \to 2$	z_2
U	$f(U)_{10}$	20_{10}	$(1 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 0 \cdot 2^0)_{10}$	10100_2

In der digitalen Kommunikation werden Nachrichten binär mit dem Alphabet $\mathbb{F}_2 = \{0, 1\}$ kodiert. Soll eine binäre Kodetabelle für ein Alphabet $\mathcal{A}$ erzeugt werden, sind höchstens

$$l = \min\{k \in \mathbb{N};\ 2^k \geq |\mathcal{A}|\} = \lceil \log_2 (|\mathcal{A}|) \rceil$$

Bits nötig. Damit können die Großbuchstaben aus $\mathcal{A}_0$ mit 5 Bits kodiert werden, etwa über die injektive Abbildung

$$f_1 : \mathcal{A}_0 \to \mathbb{F}_2^5,$$
$$A \mapsto 00000,\ B \mapsto 00001,\ C \mapsto 00010,\ \ldots,\ Z \mapsto 11001.$$

Hier liegt der Fall einer Kodierung mit konstanter Bitlänge vor.

Beispiel 2.22

Wir betrachten die Nachricht

„SICHERHEITISTDASEINZIGEZIELDERKRYPTOGRAPHIE“

und erhalten mit der binären Kodierung f_1

$$\begin{aligned}(\ & 10010, 01000, 00010, 00111, 00100, 10001, 00111, 00100, 01000,\\ & 10011, 01000, 10010, 10011, 00011, 00000, 10010, 00100, 01000,\\ & 01101, 11001, 01000, 00110, 00100, 11001, 01000, 00100, 01011,\\ & 00011, 00100, 10001, 01010, 10001, 11000, 01111, 10011, 01110,\\ & 00110, 10001, 00000, 01111, 00111, 01000, 00100\ ).\end{aligned}$$

Die Nachricht kann mit 215 Bits kodiert werden.

Wir werden uns im weiteren Verlauf jedoch nicht nur auf die 26 Großbuchstaben beschränken, sondern auch andere Zeichen zulassen und deshalb die Zeichen durch sieben oder mehr Bits darstellen. Somit ist 10100_2 durch 0010100_2 in der 7-Bit-Kodierung darzustellen. Eine andere Möglichkeit besteht in der aus früheren Zeiten bekannten ASCII-Kodierung von Zeichen. Dabei entspricht der Buchstabe „A“ der Binärzahl $1000001_2 (65_{10})$

und der Buchstabe „U“ der Binärzahl $1010101_2(85_{10})$. Werden acht Bits zugrunde gelegt, erhalten wir eine erweiterte ASCII-Kodierung. Es gibt viele verschiedene solcher Standards. Ein weiterer ist ISO 8859-1[12], der 191 der 256 Zeichen darstellbar macht ($20_{16} - 7E_{16}$ und $A0_{16} - FF_{16}$). Das Beispiel 2.22 zeigt, dass bereits eine relativ kurze Nachricht bestehend aus Elementen eines Alphabets mit geringem Umfang eine große Anzahl Bits produziert.

Ein (Internet-)Standard, der heute vielfach Anwendung findet, ist die UTF-8-Kodierung[13]. Dabei wird dem für ein bestimmtes Zeichen festgelegten digitalen Code (der so genannte Unicode) ein bis zu 4 Byte langer Binärcode zugeordnet. Die Großbuchstaben können durch 7 Bits dargestellt werden. Bei Umlauten aber werden 2 Byte lange Binärcodes benötigt. Es handelt sich somit um eine Kodierung mit variabler Bitlänge.

Beispiel 2.23

Die Nachricht

„VERSCHLÜSSELUNG“

wird in der UTF-8-Kodierung durch

$$\begin{pmatrix} 01010110, 01000101, 010100100, 1010011, 01000011, 01001000, \\ 01001100, \mathbf{11000011}, \mathbf{10011100}, 01010011, 01010011, 01000101, \\ 01001100, 01010101, 01001110, 01000111 \end{pmatrix}$$

repräsentiert. Die beiden fett dargestellten Bytes stehen für den Buchstaben Ü.

Die binäre Kodierung $c : \mathcal{A} \to \{0,1\}^n$ von Daten führt zu einer großen Anzahl benötigter Bits für eine Nachricht. Für jede Nachricht wird Speicherplatz beansprucht und infolge dessen ergibt sich eine entsprechend lange Übertragungszeit der Nachricht. Wir wollen versuchen dieselbe Nachricht mit Hilfe einer geringeren Anzahl an Bits zu übermitteln, um Speicherplatz und Übertragungszeit einzusparen.

2.4.2. Blockweise Kodierung

Eine Möglichkeit, um die Länge der kodierten Nachricht zu verkürzen, besteht in der blockweisen Kodierung. Dabei wird sukzessive eine gleichbleibende Anzahl aufeinanderfolgender Symbole zusammen kodiert. In einem p-adischen System mit n Ziffern gibt es p^n verschiedene Zahlen. Wie viele Ziffern werden dann in einem q-adischen System benötigt, um alle diese Zahlen darstellen zu können? Analog zu den Überlegungen zur binären Kodierung sind es

$$l = \min\{l \in \mathbb{N};\ q^l \geq p^n\} = \lceil \log_q (p^n) \rceil = \left\lceil n \cdot \frac{\ln(p)}{\ln(q)} \right\rceil$$

[12] http://www.iso.org/iso/catalogue_detail.htm?csnumber=28245, 1998
[13] definiert in RFC3629: http://tools.ietf.org/html/rfc3629, 2003

Beispiel 2.24

Wir betrachten den bekannten Text mit der Blocklänge vier:

„SICH|ERHE|ITIS|TDAS|EINZ|IGEZ|IELD|ERKR|YPTO|GRAP|HIE“.

Die Länge des Textes ist 43 und somit brauchen wir ein Füllzeichen $a \in \mathcal{A}_0$. Mit $a := X$ kodieren wir die Blöcke binär, indem wir zunächst jeden Block als Zahl im 26-er Zahlensystem darstellen und das Ergebnis in eine Binärzeichenfolge umwandeln. Wir benötigen einen Binärcode der Länge $\lfloor \log_2 (26^4) \rfloor + 1 = 19$.

Textblock	Kodierung in $\mathbb{Z}_{26}$ (x_3, x_2, x_1, x_0)	$x_3 \cdot 26^3 + x_2 \cdot 26^2$ $+x_1 \cdot 26 + x_0 \cdot 26^0$	Binärdarstellung
$SICH$	$(18, 8, 2, 7)$	321835	1001110100100101011
$ERHE$	$(4, 17, 7, 4)$	81982	**00**10100000000111110
$ITIS$	$(8, 19, 8, 18)$	153678	**0**1001011000001001110
$TDAS$	$(19, 3, 0, 18)$	335990	1010010000001110110
$EINZ$	$(4, 8, 13, 25)$	76075	**00**10010100100101011
$IGEZ$	$(8, 6, 4, 25)$	144793	**0**1000110101100111001
$IELD$	$(8, 4, 11, 3)$	143601	**0**1000110000111100001
$ERKR$	$(4, 17, 10, 17)$	82073	**00**10100000010011001
$YPTO$	$(24, 15, 19, 14)$	432472	1101001100101011000
$GRAP$	$(6, 17, 0, 15)$	116963	**00**11100100011100011
$HIEX$	$(7, 8, 4, 23)$	128567	**00**11111011000110111

Die fett dargestellten Nullen verlängern die kurzen Bitfolgen. Das gewährleistet die Eindeutigkeit einer konkatenierten Binärfolge, da so auch eine eindeutige Blocklänge in der Bitfolge vorliegt. Anstelle von $220 = 44 \cdot 5$ Bits brauchen wir für die blockweise Kodierung $209 = 19 \cdot 11$ Bits, was zu einer Einsparung führt. Im besten Fall kann die Nachricht der Länge 43 auf Basis dieser Methode mit 203 Bits kodiert werden.

Auch die blockweise Kodierung erzeugt eine Kodetabelle, die jedoch in der Praxis nicht notiert wird. Stattdessen wird die Umrechung so wie im Beispiel 2.24 gezeigt durchgeführt.

2.4.3. Huffman-Kodierung

Eine weitere Möglichkeit, um Nachrichten zu kodieren, besteht darin, unterschiedliche Bitlängen für einzelne Symbole zu erzeugen. Dabei muss die Umkehrbarkeit der Operation gewährleistet sein. Das wird in diesem Zusammenhang durch die Präfix-Eigenschaft[14] ermöglicht: Zu einer gegebenen Kodierung $x_n x_{n-1} \dots x_{m+1} x_m x_{m-1} \dots x_0$ darf es keine andere der Länge $n - m + 1$ geben mit der gleichen Anfangssequenz $x_n x_{n-1} \dots x_m$. Die Huffman-Kodierung[15] ist ein in der Praxis aufgrund seiner besonderen Eigenschaften häufig benutztes Verfahren. Die Frage ist, welche Symbole kürzere und welche längere Bitfolgen zugewiesen bekommen. Tritt ein Symbol häufig auf und wird es durch eine kurze Bitfolge kodiert, lässt sich dadurch Speicherplatz einsparen. Grundlage des Verfahrens ist es demnach zu ermitteln, wie häufig die Symbole in einer zugrunde gelegten Menge von Nachrichten auftreten. Für deutschsprachige Texte gibt es Tabellen mit den Häufigkeiten[16].

[14] vgl. [6]
[15] David A. Huffman, 1925-1999
[16] siehe: http://de.wikipedia.org/wiki/Buchstabenhäufigkeit

Zunächst werden die Symbole nach ihrer Häufigkeit in einer Liste sortiert. Die Liste stellt die Blätter eines Binärbaumes dar. Der Baum wird nach und nach aufgebaut, indem die beiden Knoten mit der kleinsten Häufigkeit zusammengefasst werden und ihre Häufigkeit aufsummiert wird, solange bis nur noch ein Knoten verbleibt. Nun kann von der Wurzel aus die Kodierung erzeugt werden, indem bei jeder Aufspaltung entweder eine Eins oder eine Null angefügt wird. Die Kodierung wird zuletzt in einer Kodetabelle festgehalten.

Beispiel 2.25

Aus dem Bibeltext aus Beispiel 2.1 kann mit Hilfe einer Häufigkeitstabelle folgende Huffman-Kodierung für das Alphabet $\mathcal{A}_0$ der Großbuchstaben erzeugt werden:

A	1100	B	010011	C	111110	D	0101
E	011	F	110101	G	11011	H	11110
I	000	J	1111111	K	11010001	L	10100
M	10101	N	001	O	01000	P	110100001
Q	11010000000	R	1011	S	1110	T	1000
U	1001	V	1111110	W	010010	X	11010000001
Y	1101000001	Z	1101001				

Die Nachricht

„SICHERHEITISTDASEINZIGEZIELDERKRYPTOGRAPHIE“

wird kodiert zu

$$\begin{aligned}(\ & 1110, 000, 111110, 11110, 011, 1011, 11110, 011, 000, 1000, 000, 1110, 1000,\\ & 0101, 1100, 1110, 011, 000, 001, 1101001, 000, 11011, 011, 1101001, 000, 011,\\ & 10100, 0101, 011, 1011, 11010001, 1011, 1101000001, 110100001, 1000, 01000,\\ & 11011, 1011, 1100, 110100001, 11110, 000, 011\ ).\end{aligned}$$

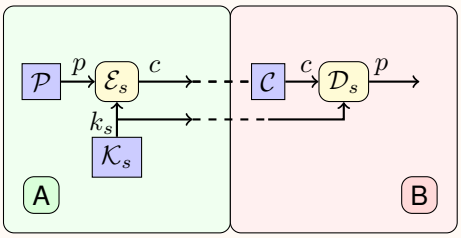
Hiermit genügen 192 bits zur Kodierung.

Beide angesprochenen Kodierungs-Verfahren führen zu einer umkehrbaren Komprimierung der betrachteten Nachricht.

3. Symmetrische Chiffriersysteme

Für ein symmetrisches Chiffriersystem $(\mathcal{P}, \mathcal{C}, \mathcal{K}_S, \mathcal{E}_S, \mathcal{D}_S)$ ist der typische Ablauf in Abbildung 3.1 zu sehen. Ein Klartext $p \in \mathcal{P}$ wird bei Instanz A mit Hilfe eines Schlüssels $k \in \mathcal{K}_S$ einer Verschlüsselungsfunktion $e_k \in \mathcal{E}_S$ zugeführt und dadurch zu einem Geheimtext $c \in \mathcal{C}$ verschlüsselt. Durch einen abgeschlossenen Schlüsselaustausch ist es Instanz B möglich, den übermittelten Geheimtext c mit einer Entschlüsselungsfunktion $d_k \in \mathcal{D}_S$ zum Klartext zu entschlüsseln.

Abbildung 3.1.: Symmetrische Chiffrierung



Oftmals wird ein zu verschlüsselnder Klartext nicht in nur einem einzigen Schritt vollständig verschlüsselt, sondern in Blöcke fester Länge aufgeteilt, die einzeln verschlüsselt werden. Denn so ist es möglich, Schlüssel einer bestimmten Länge vorzugeben und damit die Verschlüsselung einer Vielzahl verschiedener Klartexte zu automatisieren. Ansonsten müssten wir für jeden Klartext unterschiedlich lange Verschlüsselungsverfahren abhängig von der Klartextlänge erzeugen. Damit legen wir die blockweise Verschlüsselung fest in

Definition 3.1: Blockchiffre

Sei $m \in \mathbb{N}$ fest. Gegeben sei ein Chiffriersystem $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ mit einem Alphabet $\mathcal{A}$. Ein beliebiges $p \in \mathcal{P}$ werde in l Blöcke der Länge m (siehe Bemerkung (1)) aufgeteilt, $p = p_1||p_2||\ldots||p_l$. Gilt für ein $c \in \mathcal{C}$

$$c = c_1||c_2||\ldots||c_l \text{ mit } c_i = e_{k_i}(p_i)\ \forall i \in \{1, \ldots, l\}$$

mit Schlüsseln $k_1, k_2, \ldots, k_l \in \mathcal{K}$, heißt das Chiffriersystem eine Blockchiffre mit Blocklänge m. Eine Blockchiffre heißt *kontextunabhängig*, wenn die Verschlüsselung eines Blockes von keinem vorhergehenden Block abhängt, ansonsten heißt sie *kontextabhängig*.

Bemerkung.
(1) Ist die Klartextlänge nicht durch die Blocklänge teilbar, so enthält der letzte Block weniger als m Elemente des Alphabets. Dies wird durch Padding ausgeglichen, d.h. die fehlenden Zeichen werden aufgefüllt.

(2) Eine kontextunabhängige Blockchiffre erzeugt aus einem Klartextblock immer den selben Geheimtext.
(3) Eine *Mehrfachverschlüsselung* liegt vor, wenn ein Block nicht nur einmal, sondern mehrfach verschlüsselt wird.

3.1. Betriebsmodi für symmetrische Chiffriersysteme

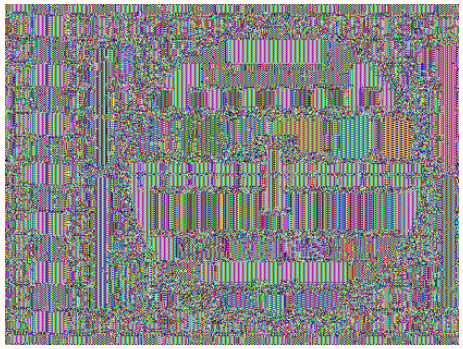
Der einfachste Fall einer Blockchiffre liegt vor, wenn

$$c_i = e_k(p_i)$$

ist und so jeder Block einzeln verschlüsselt wird. Dieser Modus wird Electronic Codebook Mode (ECB) genannt. Gleiche Klartextblöcke werden auf gleiche Geheimtextblöcke abgebildet. Dies kann zu Unsicherheiten führen, wenn Informationen im Geheimtext erhalten bleiben. Folgende Abbildungen zeigen ein Testbild (links) und das mit DES im ECB-Modus verschlüsselte Bild. Gewisse Strukturen sind erhalten und es kann darauf geschlossen werden, wie das Originalbild aussehen könnte.

Abbildung 3.2.: Testbild mit ähnlichen Strukturen innerhalb des Bildes.

Abbildung 3.3.: ECB-Modus: Strukturen des Testbildes sind erkennbar.



Die Situation ist etwas unbefriedigend. Benutzen wir dagegen den im letzten Block erzeugten Geheimtext zusätzlich als Eingabe, so können wir das Problem des ECB-Modus lösen. Dabei verwenden wir die bitweise XOR-Operation:

$$\oplus : \mathbb{F}_2 \times \mathbb{F}_2 \to \mathbb{F}_2,\ (u, v) \mapsto u \oplus v := u +_2 v,$$
$$\oplus : \mathbb{F}_2^n \times \mathbb{F}_2^n \to \mathbb{F}_2^n,\ (u_{n-1} \dots u_0, v_{n-1} \dots v_0) \mapsto u \oplus v := (u_{n-1} +_2 v_{n-1}, \dots, u_0 +_2 v_0).$$

Beim Cipher Block Chaining Mode (CBC) wird der im letzten Block erzeugte Geheimtext per XOR mit dem aktuellen Klartext verknüpft und so im aktuellen Block zur Verschlüsselung benutzt:

$$c_i = e_k(p_i \oplus c_{i-1}).$$

Dabei muss es ein c_0 geben, das auch als Initialisierungsvektor (IV) bezeichnet wird. Dieser ist frei wählbar und kann offen ausgetauscht werden. Zur Entschlüsselung ist

$$p_i = d_k(c_i) \oplus c_{i-1}$$

zu berechnen.

Abbildung 3.4.: CBC-Modus: Es sind keine Muster erkennbar.

Bis hierher gilt stets, dass die Blocklängen von Klar- und Geheimtext gleich der zu verschlüsselnden Blocklänge m sind. Soll ein Block der Länge m verschlüsselt werden, jedoch der dabei zu verarbeitende Klartextblock die Länge $j \leq m$ haben, gibt es andere Betriebsmodi. Wir betrachten zunächst den Cipher Feedback Mode (CFB). Es wird ein so genanntes Register r_i der Länge m eingeführt, es gelte $r_1 = \text{IV}$. Mit

$$r_i = (r_{i-1} \cdot 2^j \bmod 2^m) + c_{i-1},\ i > 1,$$

wird in jedem Schritt ein Linksshift um j Bits modulo m Bits und anschließend eine Belegung der j frei werdenden niedrigwertigen Bits mit dem Geheimtext c_{i-1} des letzten Blockes durchgeführt. Der Geheimtext für Block i entsteht durch

$$c_i = p_i \oplus s_{j,m}(e_k(r_i)) \text{ mit } s_{j,m}(e_k(r_i)) = e_k(r_i) \cdot 2^{-(m-j)}.$$

Die kryptographische Funktion e_k wird auf das Register zu Block i angwandt, davon werden über die Funktion s die höherwertigen j Bits beschafft und per XOR mit p_i verknüpft. Zur Entschlüsselung gelangt man ohne die zu e_k gehörende Entschlüsselungsfunktion d_k zu benötigen über

$$p_i = c_i \oplus s_{j,m}(e_k(r_i)).$$

Eine Variante davon ist der Output Feedback Mode (OFB). Anstelle des Geheimtextes c_{i-1} werden die j höherwertigen Bits des „verschlüsselten“ Registers des letzten Blocks benutzt, d.h.

$$r_i = (r_{i-1} \cdot 2^j \bmod 2^m) + s_{j,m}(e_k(r_{i-1})),\ i > 1.$$

Der Rest bleibt unverändert. Beim Counter Mode (CTR) werden weder c_{i-1} (CFB) noch $s_{j,m}(e_k(r_{i-1}))$ (OFB) sondern ein Zählerwert auf den Registerwert, der nach jeder Verschlüsselung erhöht wird, addiert.

3.2. Verschlüsselung durch Substitution

Die Substitutions-Verschlüsselung ist dadurch gekennzeichnet, dass jedes Zeichen oder jeder Block durch ein nicht notwendigerweise verschiedenes Zeichen bzw. Block ersetzt wird. Eine einfache Substitution liegt vor, wenn die Verschlüsselungsfunktion bijektiv und so eine Permutation ist. Die Verschlüsselungsfunktion kann speziell eine bijektive affin-lineare

Funktion sein. Lineare Funktionen lassen sich durch Matrizen beschreiben. Die Elemente der Matrix und die Matrix selbst unterliegen modularen Vorgaben. Dies gilt gerade auch für das Bestimmen einer inversen Matrix, die zur Entschlüsselung affin-linearer Blockchiffren benötigt wird.

Modulare Arithmetik mit Matrizen

Wir betrachten den Ring $(\mathbb{Z}_n, +_n, \cdot_n, 0, 1)$[1]. Die $m \times m$-Einheitsmatrix über diesem Ring ist $E_m = (e_{i,j})$ mit $e_{i,j} = 1$ für $i = j$ und $e_{i,j} = 0$ für $i \neq j$, $i, j = 1, \ldots, m$. Die $m \times m$-Nullmatrix ist die Matrix, deren sämtliche Einträge gleich Null sind. Zusammen mit der Addition $+_n$ und Multiplikation $\cdot_n$ ist die Menge der Matrizen $\mathbb{Z}_n^{m,m}$ über $\mathbb{Z}_n$ ein Ring mit Einselement E_m, der im allgemeinen aber nicht kommutativ ist. Dies folgt unmittelbar aus dem Gegenbeispiel

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

Das neutrale Element bezüglich der Addition ist die Nullmatrix, das neutrale Elemente bezüglich der Multiplikation ist die $m \times m$ Einheitsmatrix.

Sei $A_{i,j}$ die Matrix, die durch Streichen der i-ten Zeile und j-ten Spalte einer Matrix $(a_{i,j}) = A \in \mathbb{Z}_n^{m,m}$ entsteht. Dann ist für ein fest gewähltes $i \in \{1, \ldots, m\}$ die *Determinante* der Matrix die Abbildung

$$\det : \mathbb{Z}_n^{m,m} \to \mathbb{Z},\ \det(A) = \sum_{j=1}^{m} (-1)^{i+j} a_{i,j} \det(A_{i,j}),\ \det(a_{i,j}) = a_{i,j}.$$

Analog hätten wir die Determinante auch über die Spalten j definieren können. Eine Matrix $A \in \mathbb{Z}_n^{m,m}$ besitzt genau dann in $\mathbb{Z}_n^{m,m}$ ein multiplikatives Inverses, wenn $\det(A)$ teilerfremd zu n ist. Die *A*djunkte einer Matrix A ist die Matrix

$$\text{adj}(A) = \begin{cases} (\tilde{a}_{i,j}) \text{ und } \tilde{a}_{i,j} = ((-1)^{i+j} \det(A_{j,i})) \bmod n, & \text{für } m > 1,\ i, j \in \{1, \ldots, m\}, \\ 1, & \text{für } m = 1. \end{cases}$$

Die Inverse der Matrix A ist dann

$$A^{-1} = (\det(A))^{-1} \cdot \text{adj}(A) \bmod n,$$

wobei jede Komponente der Matrix modulo n berechnet wird.

Beispiel 3.2

Sei $A \in \mathbb{Z}_{13}^{3,3}$ mit

$$A := \begin{pmatrix} 1 & 2 & 3 \\ 2 & 3 & 1 \\ 3 & 2 & 1 \end{pmatrix}$$

[1] vgl. Satz A.37

Dann ist die Determinante $\det(A) = -12 \equiv_{13} 1$ und die Adjunkte der Matrix

$$\text{adj}(A) = \begin{pmatrix} 1 & 4 & 6 \\ 1 & 5 & 5 \\ 8 & 4 & 12 \end{pmatrix}$$

und die Inverse ist hier gleich der Adjunkten: $A^{-1} = 1 \cdot \text{adj}(A)$.

Damit legen wir affin lineare Funktionen modulo n fest in

Definition 3.3: Affin lineare Funktion modulo n

Seien $A \in \mathbb{Z}_n^{m,m}$ und $b \in \mathbb{Z}_n^m$. Eine Funktion $f : \mathbb{Z}_n^m \to \mathbb{Z}_n^m$

$$x \mapsto f(x) := (Ax + b) \bmod n$$

heißt affin linear modulo n.

Affin-lineare Blockchiffren

Viele klassische Chiffriersysteme nutzen Verschlüsselungsfunktionen, die affin-lineare Funktionen sind. Mit Hilfe bestimmter affin-linearer Funktionen modulo n können wir Chiffriersysteme erzeugen.

Definition 3.4: Affin lineare Blockchiffre

Seien $\mathcal{P} = \mathcal{C} = \mathbb{Z}_n^m$ gegeben. $\mathcal{K} \subseteq \mathbb{Z}_n^{m,m} \times \mathbb{Z}_n^m$ sei derart gewählt, dass es zu jedem $A \in \mathbb{Z}_n^{m,m}$ mit $(A, b) \in \mathcal{K}$ die dazugehörige Inverse $A^{-1} \in \mathbb{Z}_n^{m,m}$ und $(A^{-1}, b) \in \mathcal{K}$ gibt. Dann heißt $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ mit

$$e_{(A,b)} : \mathbb{Z}_n^m \to \mathbb{Z}_n^m, \qquad p \mapsto (Ap + b) \bmod n \text{ und}$$
$$d_{(A,b)} : \mathbb{Z}_n^m \to \mathbb{Z}_n^m, \qquad c \mapsto A^{-1}(c - b) \bmod n.$$

eine affin lineare Blockchiffre mit Blocklänge $m \in \mathbb{N}$.

Bemerkung.
(1) $e_{(A,b)}$ und $d_{(A,b)}$ sind bijektive Abbildungen in sich und somit spezielle Permutationen.
(2) Der Fall $m = 1$ ist ein Spezialfall einer monoalphabetischen Verschlüsselung. Die Abbildungsvorschrift kann durch eine Art Matrix dargestellt werden, deren erste Zeile die zu verschlüsselnden und deren zweite Zeile die dazugehörigen verschlüsselten Zeichen enthält.

Beispiel 3.5

Die ATBASH-Verschlüsselung ist eine Blockchiffre mit Blocklänge $m = 1$, mit $\mathcal{P} =$

$\mathcal{C} = \mathbb{Z}_{26}$ und hat die Verschlüsselungsfunktion

$$p \mapsto e_{(25,25)}(p) = (25p + 25) \text{ mod } 26.$$

Das entspricht der Buchstaben-Permutation

$$\begin{pmatrix} ABCDEFGHIJKLMNOPQRSTUVWXYZ \\ ZYXWVUTSRQPONMLKJIHGFEDCBA \end{pmatrix}.$$

Der Klartext

„SICHERHEITISTDASEINZIGEZIELDERKRYPTOGRAPHIE“

wird verschlüsselt zu

„HRXSVISVRGRHGWZHVRMARTVARVOWVIPIBKGLTIZKSRV“.

Durch nochmalige Anwendung derselben Funktion erhalten wir wieder den Klartext:

$$25(25p + 25) + 25 \text{ mod } 26 = 625p + 650 \text{ mod } 26 = p.$$

Die ATBASH-Verschlüsselung ist damit eine Involution.

Weitere Beispiele für affin-lineare Blockchiffren sind nach Wahl von m und entsprechend geeigneten Möglichkeiten für a bzw. A

- Verschiebungschiffre mit Blocklänge $m = 1$ und Schlüssel $(1, b)$,
- Multiplikationschiffre mit Blocklänge $m = 1$ und Schlüssel $(a, 0)$,
- Substitutionschiffre mit Blocklänge $m = 1$ und Schlüssel (a, b),
- Vigenère-Chiffre[2] mit Blocklänge $m > 1$ und Schlüssel $(E_m, \vec{b})$,
- Hill-Chiffre[3] (linear) mit Blocklänge $m > 1$ und Schlüssel $(A, \vec{0})$.
- Transpositionschiffre mit Blocklänge $m > 1$ und Schlüssel $(\tilde{E}_m, \vec{0})$ und $\tilde{E}_m \in \mathbb{F}_2^{m,m}$, wobei in jeder Zeile und jeder Spalte genau einmal Eins und sonst Null steht. Das ist ein Spezialfall der Hill-Chiffre.

Beispiel 3.6

Ein Beispiel im Großbuchstabenalphabet dafür, dass nicht jede Wahl von a bei $m = 1$ Sinn macht, ist $a = 4$. So werden mit $b = 1$ und der affin-linearen Funktion

$$e_{(4,1)} : \mathbb{Z}_{26} \to \mathbb{Z}_{26},\ p \mapsto e_{(4,1)}(p) = (4p + 1) \text{ mod } 26$$

beispielsweise sowohl das E als auch das R auf den Buchstaben R abgebildet. Der Klartext

„SICHERHEITISTDASEINZIGEZIELDERKRYPTOGRAPHIE“

[2] B. de Vigenère, 1523-1596
[3] L.S. Hill, 1890-1961

wird verschlüsselt zu

„VHJDRRDRHZHVZNBVRHBXHZRXHRTNRRPRTJZFZRBJDHR“.

Der Klartext lässt sich nicht wiederherstellen.

Wir müssen überlegen, wie wir auf Basis eines gegebenen $n \in \mathbb{N}$ entscheiden können, welche Werte von a bei einer Blockgröße von $m = 1$ Sinn machen. Dazu wissen wir zunächst, dass a ein inverses Element in $\mathbb{Z}_n$ besitzen muss, was in Beispiel 3.6 nicht gegeben ist.

Erweiterter Euklidischer Algorithmus

Definition 3.7: Einheit

Betrachte den Monoid $(\mathbb{Z}_n, \cdot_n, 1)$. Ein Element $a \in \mathbb{Z}_n$ heißt Einheit in $\mathbb{Z}_n$, wenn es ein $b \in \mathbb{Z}_n$ gibt mit $a \cdot_n b = 1$. Die Menge aller Einheiten in $(\mathbb{Z}_n, \cdot_n, 1)$ wird mit $\mathbb{Z}_n^*$ bezeichnet.

Bemerkung.
(1) Der Begriff Einheit ist allgemein für einen Monoid definiert[4].
(2) Es ist $\mathbb{Z}_n^* \subseteq \mathbb{Z}_n$.

Beispiel 3.8

In $(\mathbb{Z}_6, \cdot_6, 1)$ gibt es die beiden Einheiten 1 und 5, $\mathbb{Z}_6^* = \{1, 5\}$. Weiter ist $\mathbb{Z}_{26}^* = \{1, 3, 5, 7, 9, 11, 15, 17, 19, 21, 23, 25\}$).

Während die Suche nach Einheiten in $\mathbb{Z}_6$ noch leicht zu lösen ist, wird die Sache in $\mathbb{Z}_{26}$ bereits schwieriger. Wir suchen einen einfachen Weg, um Kandidaten zu überprüfen. Die Elemente, deren größter gemeinsamer Teiler mit $n \in \mathbb{N}$ eins ist, sind in $\mathbb{Z}_n$ invertierbar. Um dies Ergebnis vorzubereiten, definieren wir zunächst den Begriff des Erzeugers[5] einer Gruppe in

Definition 3.9: Zyklische Gruppe, Erzeuger einer Gruppe

Es sei $(G, \circ, e_\circ)$ eine beliebige Gruppe und $a \in G$. Weiter sei unter Beachtung von Satz und Definition A.10

$$\langle a \rangle \quad := \quad \{\bigcirc_{i=1}^{n} a;\ n \in \mathbb{Z}\}. \tag{3.1}$$

$(G, \circ, e_\circ)$ heißt zyklisch, wenn ein $a \in G$ existiert mit $\langle a \rangle = G$. In diesem Fall heißt a der Erzeuger von G und G wird von a erzeugt.

[4] vgl. Bemerkung (3) zu Definition A.6
[5] Die Schreibweisen sind in Satz und Definition A.11 nachzulesen.

Bemerkung.
(1) Für multiplikative bzw. additive Gruppen schreiben wir gemäß Satz und Definition A.11

$$\begin{aligned} \langle a \rangle &:= \{a^n;\ n \in \mathbb{Z}\} \quad \text{in multiplikativer Schreibweise,} \\ \langle a \rangle &:= \{n \cdot a;\ n \in \mathbb{Z}\} \quad \text{in additiver Schreibweise.} \end{aligned}$$

(2) $(\langle a \rangle, \circ, e_\circ)$ ist eine Untergruppe von $(G, \circ, e_\circ)$. So ist etwa in multiplikativer Schreibweise für $b = a^k \in \langle a \rangle$ auch $b^{-1} = a^{-k} \in \langle a \rangle$ und für $c = a^l \in \langle a \rangle$ ergibt sich $c \cdot b^{-1} = a^l \cdot a^{-k} = a^{l-k} \in \langle a \rangle$.
(3) Jede zyklische Gruppe ist kommutativ, etwa multiplikativ: $a^m \odot a^n = a^{m+n} = a^{n+m} = a^n \odot a^m$.
(4) Jedes Element einer zyklischen Gruppe ist eine Einheit, etwa multiplikativ: $a^m \odot a^{-m} = a^{m-m} = a^0 = e$.
(5) Erzeuger einer Gruppe sind oft nicht eindeutig bestimmt.
(6) Eine Untergruppe kann auch durch mehr als ein Element erzeugt werden. Die Schreibweise erfolgt analog, etwa multiplikativ: $\langle a, b \rangle := \{a^m \odot b^n;\ m, n \in \mathbb{Z}\}$ liefert die von den Elementen $a, b \in G$ erzeugte Untergruppe $(\langle a, b \rangle, \odot, e_\odot)$.

Beispiel 3.10

(I) $(\mathbb{Z}, +, 0)$ ist eine zyklische Gruppe mit Erzeuger 1. Aber es gilt auch $\langle -1 \rangle = \mathbb{Z}$.
(II) Mit $n\mathbb{Z} := \{z \cdot n;\ z \in \mathbb{Z}\} = \langle n \rangle$, $n \in \mathbb{N}_0$, ist $(n\mathbb{Z}, +, 0)$ eine zyklische Gruppe mit Erzeuger n.
(III) $(\mathbb{Z}_5^*, \cdot_5, 1)$ ist eine zyklische Gruppe mit Erzeuger 2.

Satz 3.11

Jede Untergruppe von $(\mathbb{Z}, +, 0)$ ist zyklisch und von der Form $n\mathbb{Z}$.

Beweis.

Sei $(H, +, 0)$ Untergruppe von $(\mathbb{Z}, +, 0)$. Für $H = \{0\}$ ist $H = 0\mathbb{Z}$. Für $H \neq \{0\}$ seien n das kleinste positive Element von H und $a \in H$ beliebig. Es gilt: $a = q \cdot n + r$ mit $q \in \mathbb{Z}$ und $r \in \mathbb{Z}_n$. Mit $q \cdot n \in H$ ist auch $(-q \cdot n) \in H$ und damit auch $r = a + (-q \cdot n)$. Wegen $0 \leq r < n$ muss $r = 0$ sein und $n | a$, d.h. $a \in n\mathbb{Z}$.

□

Wir betrachten die spezielle Gruppe $(\mathbb{Z}, +, 0)$ und nutzen das letzte Ergebnis aus, dass alle Untergruppen davon bekannt sind.

Beispiel 3.12

Wir betrachten $(\mathbb{Z}, +, 0)$. Seien $a, b \in \mathbb{Z}$. Dann ist $\langle a, b \rangle = \{m \cdot a + n \cdot b;\ m, n \in \mathbb{Z}\}$ und $(\langle a, b \rangle, +, 0)$ eine von a und b erzeugte Untergruppe von $(\mathbb{Z}, +, 0)$. Weil die Untergruppen von $(\mathbb{Z}, +, 0)$ bekannt sind, wissen wir, dass es zu der von a, b erzeugten

Untergruppe ein $d \in \mathbb{N}$ geben muss mit $\langle a, b\rangle = d\mathbb{Z}$. Mit $a, b \in \langle a, b\rangle$ gilt $d|a, b$. d ist ein gemeinsamer Teiler von a und b und so folgt $a = z_1 \cdot d,\ b = z_2 \cdot d$ für $z_1, z_2 \in \mathbb{Z}$ und mit $m, n \in \mathbb{Z}:\ m \cdot a + n \cdot b = m \cdot z_1 \cdot d + n \cdot z_2 \cdot d = (m \cdot z_1 + n \cdot z_2) \cdot d \in d\mathbb{Z}$.

Seien $a \in \mathbb{Z}$, $b \in \mathbb{Z} \setminus \{0\}$ und $\langle a, b\rangle = d\mathbb{Z}$. Nach Beispiel 3.12 gibt es Zahlen $x, y \in \mathbb{Z}$ mit $d = x \cdot a + y \cdot b$. Für alle $t \in \mathbb{Z}$ mit $t|a, b$, d.h. $a = v_1 \cdot t$ und $b = v_2 \cdot t$ mit $v_1, v_2 \in \mathbb{Z}$, gilt wegen $d = x \cdot a + y \cdot b = x \cdot v_1 \cdot t + y \cdot v_2 \cdot t = (x \cdot v_1 + y \cdot v_2) \cdot t$ somit $t|d$. d ist der größte gemeinsame Teiler von a und b,

$$d = \mathsf{ggT}(a, b).$$

Mit $a = z_1 \cdot d,\ b = z_2 \cdot d$ für $z_1, z_2 \in \mathbb{Z}$ und $a = q \cdot b + r$ für $q \in \mathbb{Z}$ und $r \in \mathbb{Z}_b$ folgt $r = a - q \cdot b = z_1 \cdot d - q \cdot z_2 \cdot d = (z_1 - q \cdot z_2) \cdot d$, d.h. $d \mid b, r$. Ist t ein beliebiger gemeinsamer Teiler von r und b, d.h. $r = u_1 \cdot t$, $b = u_2 \cdot t$ mit $u_1, u_2 \in \mathbb{Z}$, so folgt $a = q \cdot b + r = q \cdot u_2 \cdot t + u_1 \cdot t = (q \cdot u_2 + u_1) \cdot t$, $t \mid a, b$. Mit $d = \mathsf{ggT}(a, b)$ gilt $t \leq d$. Der größte gemeinsame Teiler von a und b ist auch größter gemeinsamer Teiler von b und r. Mit dem Euklidischen Algorithmus[6] lässt sich der größte gemeinsame Teiler zweier Zahlen berechnen.

Satz 3.13: Euklidischer Algorithmus

Seien $a, b \in \mathbb{Z},\ b \neq 0$. Mit $r_1 := a,\ r_2 := b$ bricht die Division mit Rest

$$r_i := q_i \cdot r_{i+1} + r_{i+2}$$

mit $|r_{i+2}| < |r_{i+1}|$ falls $r_{i+2} \neq 0$ nach höchstens $n \leq |b|$ Schritten ab, d.h. es gibt ein $n \leq |b|$ mit $r_{i+1} \neq 0$ für $i < n$ und $r_n = q_n \cdot r_{n+1} + 0$. Dann gilt: $d := r_{n+1} = \mathsf{ggT}(a, b)$ und es können $x, y \in \mathbb{Z}$ bestimmt werden mit $d = x \cdot a + y \cdot b$.

Beweis.

Wir setzen die Idee fort, dass der größte gemeinsame Teiler von $r_1 := a$ und $r_2 := b$ auch größter gemeinsamer Teiler von r_2 und r_3 mit $r_1 = q_1 \cdot r_2 + r_3$ für $q_1 \in \mathbb{Z}$ und $r_3 \in \mathbb{Z}_{r_2}$ ist. Mit $r_i := q_i \cdot r_{i+1} + r_{i+2}$ gibt es wegen $r_{i+2} < r_{i+1}$ ein $n \leq |r_2|$ mit $r_n = q_n \cdot r_{n+1} + 0$. Damit gilt

$$\left.\begin{array}{l} r_{n+1}|r_n \\ r_{n-1} = q_{n-1} \cdot r_n + r_{n+1} \end{array}\right\} \Rightarrow r_{n+1}|r_{n-1} \Rightarrow \ldots \Rightarrow r_{n+1}|r_2 \Rightarrow r_{n+1}|r_1$$

Zuletzt ist $\mathsf{ggT}(q_n \cdot r_{n+1}, r_{n+1}) = r_{n+1}$, da r_{n+1} gemeinsamer Teiler der beiden Zahlen und das größte Element in $D_{r_{n+1}}$ ist.

□

Die Anzahl Iterationsschritte des Verfahrens ist recht grob abgeschätzt[7]. Wir verdeutlichen uns den Ablauf anhand

[6] Euklid von Alexandria, um 365-300 v. Chr.

[7] Es kann gezeigt werden, dass mit $a > b > 0$ und $\theta := \frac{1+\sqrt{5}}{2}$ die Anzahl an Iterationen im Algorithmus höchstens $1 + \frac{\log b}{\log \theta}$ beträgt.

Beispiel 3.14

Gesucht ist $d = \mathsf{ggT}(36, 21)$. Der Euklidische Algorithmus liefert

$$\begin{array}{rcrcrl} 36 &=& 1 \cdot 21 &+& 15, & \\ 21 &=& 1 \cdot 15 &+& 6, & \\ 15 &=& 2 \cdot 6 &+& 3, & \\ 6 &=& 2 \cdot 3 &+& 0. & \end{array} \qquad \begin{array}{rcrcrcrcr} 15 &=& 1 &\cdot& 36 &+& (-1) &\cdot& 21 \\ 6 &=& (-1) &\cdot& 36 &+& 2 &\cdot& 21 \\ 3 &=& 3 &\cdot& 36 &+& (-5) &\cdot& 21 \end{array}$$

Somit ist $3 = \mathsf{ggT}(36, 21)$ und $3 = 3 \cdot 36 + (-5) \cdot 21$.

Möchten wir nicht nur den größten gemeinsamen Teiler ermitteln, sondern auch die obige Darstellung $d = x \cdot a + y \cdot b$, erweitern wir das Verfahren durch Mitführen der Koeffizienten zu

Algorithmus 3.15: Erweiterter Euklidischer Algorithmus

Eingabe: $a, b \in \mathbb{Z},\ b \neq 0$.
Ausgabe: $d = \mathsf{ggt}(a, b)$ mit $d = x[0] \cdot a + y[0] \cdot b$.

$x[0] := 1, x[1] := 0, y[0] := 0, y[1] := 1, vz := 1$.
while $b \neq 0$ **do**
  $r := a\%b$.
  $q := a/b$.
  $a := b$.
  $b := r$.
  $xt := x[1],\ yt := y[1]$.
  $x[1] := q \cdot x[1] + x[0],\ y[1] := q \cdot y[1] + y[0]$.
  $x[0] := xt,\ y[0] := yt$.
  $vz := -vz$.
end while
$x[0] := vz \cdot x[0]$.
$y[0] := -vz \cdot y[0]$.
$d := a$.
return $d,\ x[0],\ y[0]$.

Bemerkung.
(1) Zwischen dem größten gemeinsamen Teiler und dem in diesem Zusammenhang oft genannten kleinsten gemeinsamen Vielfachen zweier Zahlen $a, b \in \mathbb{Z}$

$$\mathsf{kgV}(a, b) := \min\{n \in \langle a \rangle \cap \langle b \rangle;\ n > 0\}$$

besteht der Zusammenhang[8] $\mathsf{ggT}(a, b) \cdot \mathsf{kgV}(a, b) = |a \cdot b|$.
(2) Der größte gemeinsame Teiler von mehr als zwei Zahlen lässt sich induktiv ermitteln.

[8] siehe die Überlegungen von Definition A.39 bis (A.2) im Anhang.

Beispiel 3.16

$\text{ggT}(36, 21, 15) = \text{ggT}(\text{ggT}(36, 21), 15) = \text{ggT}(3, 15) = 3.$

Ist $1 = \text{ggT}(a, b)$, so gibt es Zahlen $x, y \in \mathbb{Z}$ mit $1 = x \cdot a + y \cdot b$. Seien umgekehrt $x, y \in \mathbb{Z}$ Zahlen mit $1 = x \cdot a + y \cdot b$. Ist $t \in \mathbb{N}$ ein gemeinsamer Teiler von a und b, d.h. $a = v_1 \cdot t$ und $b = v_2 \cdot t$, so gilt $1 = x \cdot v_1 \cdot t + y \cdot v_2 \cdot t$; t ist ein Teiler der Eins, also muss $t = 1$ sein, $1 = \text{ggT}(a, b)$. Den größten gemeinsamen Teiler zweier Zahlen benötigen wir nun, um Einheiten oder Erzeuger von Gruppen zu finden.

Satz 3.17

$(\mathbb{Z}_n, +_n, 0)$ ist eine zyklische Gruppe und jedes $a \in \mathbb{Z}_n$ ist genau dann Erzeuger der Gruppe, wenn $\text{ggT}(a, n) = 1$ ist.

Beweis.

Es ist $\langle 1 \rangle = \{z \cdot_n 1;\ z \in \mathbb{Z}\} = \{z \cdot_n 1;\ z \in \mathbb{Z}_n\} = \mathbb{Z}_n$. Damit ist 1 ein Erzeuger der Gruppe und diese ist zyklisch.
„⇒“. Sei $a \in \mathbb{Z}_n$ ein Erzeuger, d.h. $\langle a \rangle = \mathbb{Z}_n$. Dann gibt es ein $a' \in \mathbb{Z}$ mit $a' \cdot_n a = 1$. Damit: $a' \cdot a \equiv_n 1$. Es gibt ein $z' \in \mathbb{Z}$ mit $a' \cdot a = -z' \cdot n + 1$, $1 = a' \cdot a + z' \cdot n$ und so $\text{ggT}(a, n) = 1$.
„⇐“. Sei $\text{ggT}(a, n) = 1$ für ein $a \in \mathbb{Z}_n$. Dann gibt es ein $a' \in \mathbb{Z}$ mit $a' \cdot a \equiv_n 1$ und so $a' \cdot_n a = 1$. Für jedes $z \in \mathbb{Z}_n$ gilt $z = z \cdot_n 1 = z \cdot_n (a' \cdot_n a) = (z \cdot_n a') \cdot_n a$. Zu jedem $z \in \mathbb{Z}_n$ gibt es demnach ein $u := z \cdot_n a' \in \mathbb{Z}$ mit $u \cdot_n a = z$. Damit ist $\langle a \rangle = \mathbb{Z}_n$.

□

Satz 3.18

Es gilt $a \in \mathbb{Z}_n^* \Leftrightarrow \text{ggT}(a, n) = 1$.

Beweis.

„⇐“: $\text{ggT}(a, n) = 1 \Rightarrow 1 = x \cdot a + y \cdot n \Rightarrow (1 - y \cdot n) \bmod n = x \cdot a \bmod n \Rightarrow x \cdot a \equiv_n 1 \Rightarrow x \cdot_n a = (q \cdot n + r) \cdot_n a = r \cdot_n a = 1, r \in \mathbb{Z}_n$.
„⇒“: $x \cdot_n a = 1 \overset{q \in \mathbb{Z}}{\Rightarrow} x \cdot a = q \cdot n + 1 \Rightarrow 1 = x \cdot a - q \cdot n \Rightarrow \text{ggT}\ (a, n) = 1$.

□

Eine wichtige Anwendung des erweiterten Euklidischen Algorithmus ist das Bestimmen des Inversen eines Elementes aus der Einheitengruppe $(\mathbb{Z}_n^*, \cdot_n, 1)$. Denn es gilt $\text{ggT}(a, n) = 1$ für jedes $a \in \mathbb{Z}_n^*$ und so gibt es $x, y \in \mathbb{Z}$ mit $1 = x \cdot a + y \cdot n$. Mit $x = q \cdot n + r$ folgt $(q \cdot n + r) \cdot_n a = (q \cdot n \cdot a + r \cdot a) \bmod n = r \cdot_n a = 1$. Wegen $r \cdot_n a = a \cdot_n r = 1$ ist $r \in \mathbb{Z}_n^*$ und das zu a inverse Element in $\mathbb{Z}_n^*$.

Beispiel 3.19

Gesucht
(I) ist in $(\mathbb{Z}_8^*, \cdot_8, 1)$ das zu 3 inverse Element. $Z_8^* = \{1,3,5,7\}$, $1 = 3 \cdot 3 + (-1) \cdot 8$ und damit ist 3 das gesuchte inverse Element.
(II) sind in $(\mathbb{Z}_{26}^*, \cdot_{26}, 1)$ zu jedem Element die inversen Elemente:

x	1	3	5	7	9	11	15	17	19	21	23	25
x^{-1}	1	9	21	15	3	19	7	23	11	5	17	25

Wir haben gesehen, dass es in $(\mathbb{Z}_n^*, \cdot_n, 1)$ zu jedem Element ein Inverses gibt. Nun wollen wir noch etwas weiter gehen und die Gruppenstruktur nachweisen.

Satz und Definition 3.20: Prime Restklassengruppe

Die Menge der zu $n \in \mathbb{N}$ teilerfremden Reste

$$\mathbb{Z}_n^* = \{a \in \mathbb{Z}_n;\ \text{ggT}(a,n) = 1\}$$

zusammen mit der mod-n-Multiplikation und dem Einselement 1, $(\mathbb{Z}_n^*, \cdot_n, 1)$, ist eine Gruppe, die so genannte prime Restklassengruppe.

Beweis.

Die Assoziativität überträgt sich von $\mathbb{Z}_n$. Mit $\text{ggT}(1,n) = 1$ folgt $1 \in \mathbb{Z}_n^*$. Seien $a, b \in \mathbb{Z}_n^*$ mit $1 = x \cdot a + q \cdot n$ und $1 = y \cdot b + p \cdot n$. Dann ist $x \cdot y \cdot a \cdot b = (1 - q \cdot n)(1 - p \cdot n) = 1 - (q \cdot p \cdot n - q - p) \cdot n$, $1 = x \cdot y \cdot a \cdot b + (q \cdot p \cdot n - q - p) \cdot n$ und so $\text{ggT}(a \cdot b, n) = 1$. Mit $a \cdot b = t \cdot n + r$ folgt schließlich $1 = x \cdot y \cdot r + (x \cdot y \cdot t + q \cdot p \cdot n - q - p) \cdot n$ und damit $\text{ggT}(a \cdot_n b, n) = 1$. Dass es zu jedem Element in $\mathbb{Z}_n^*$ ein Inverses gibt, haben wir bereits gezeigt.

□

Nicht jede prime Restklassengruppe ist zyklisch, wie am Beispiel $(\mathbb{Z}_8^*, \cdot_8, 1)$ zu sehen ist. Dazu müsste es einen Erzeuger $a \in \mathbb{Z}_8^*$ von $\mathbb{Z}_8^*$ geben. Betrachten wir für $a \in \mathbb{Z}_8^*$ jeweils die Menge $\{a^k;\ k \in \mathbb{N}\}$, so gibt es stets ein kleinstes $k \in \mathbb{N}$, so dass $a^k \equiv_8 1$ ist. Ein solches k nennen wir die Ordnung[9] von a in $\mathbb{Z}_8^*$, $k = \text{ ord}_{\mathbb{Z}_8^*}(a)$. Gäbe es ein a mit $\text{ord}_{\mathbb{Z}_8^*}(a) = 4$, so wäre a ein Erzeuger dieser Gruppe.

Beispiel 3.21

In $\mathbb{Z}_8^* = \{1,3,5,7\}$ gilt $\text{ord}_{\mathbb{Z}_8^*}(3) = \text{ord}_{\mathbb{Z}_8^*}(5) = \text{ord}_{\mathbb{Z}_8^*}(7) = 2 \neq 4$. Für $\mathbb{Z}_5^* = \{1,2,3,4\}$ ist $\text{ord}_{\mathbb{Z}_5^*}(2) = 4$. Jedes Element in $\mathbb{Z}_5^*$ ist eine Potenz von 2. Also gilt $\langle 2 \rangle = \mathbb{Z}_5^*$.

Eine Möglichkeit zu entscheiden, ob die endliche Gruppe $(\mathbb{Z}_n^*, \cdot_n, 1)$ zyklisch ist, liegt darin, ein Element $a \in \mathbb{Z}_n^*$ aus der Gruppenmenge zu bestimmen, für das die Ordnung von a

[9] vgl. die allgemeine Definition A.12

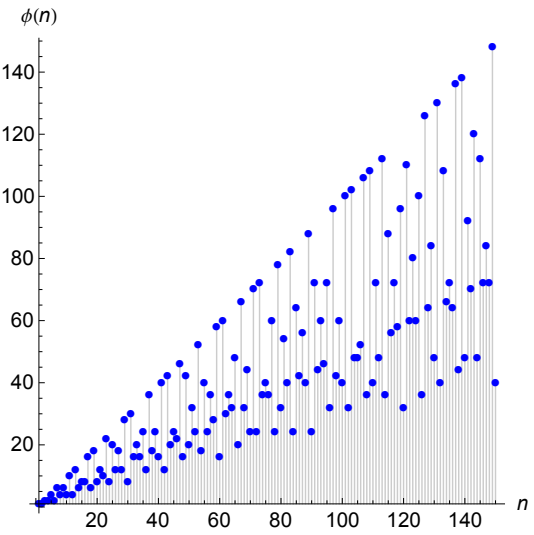
gleich der Anzahl Elemente in $\mathbb{Z}_n^*$ ist, also $\text{ord}_{\mathbb{Z}_n^*}(a) = |\mathbb{Z}_n^*|$. Für große n kann es jedoch mühsam sein, die Anzahl der Elemente der dazugehörigen primen Restklassengruppe[10] zu bestimmen. Dazu brauchen wir noch etwas Vorarbeit, um die wir uns in Abschnitt 4.1 kümmern werden.

Definition 3.22: Eulersche ϕ-Funktion

Die Anzahl der Elemente in $\mathbb{Z}_n^*$ wird mit

$$\phi(n) := |\mathbb{Z}_n^*| = |\{a \in \mathbb{Z}_n;\ \ \text{ggT}(a, n) = 1\}|$$

bezeichnet. Dabei heißt $\phi : \mathbb{N} \to \mathbb{N},\ n \mapsto \phi(n)$ Eulersche ϕ-Funktion.



Beispiel 3.23

Uns interessiert noch zum Abschluss, wie viele mögliche Schlüssel es in $\mathbb{Z}_{26}$ für eine Blockgröße von $m = 1$ gibt. Das entspricht genau der Anzahl der invertierbaren Elemente in $\mathbb{Z}_{26}$. Für $\mathbb{Z}_{26}$ ergibt sich durch Nachrechnen $\phi(26) = 12$.

Transpositionschiffren

Im Gegensatz zur allgemeinen Substitutions-Verschlüsselung wird durch die Transpositionsverschlüsselung die Position zweier Zeichen vertauscht. Dies wird durch eine entsprechend große Blockgröße erreicht. Durch vielfache Anwendung verschiedener Transpositionen kann jedes Zeichen (mehrmals) die Position wechseln. Eine Mehrfachtransposition eines Klartextes der Länge $m \in \mathbb{N}$ stellt eine Transpositionschiffre dar. Es gibt deshalb dafür insgesamt $m!$ Möglichkeiten einer Vertauschung. Dies kann den Empfänger von Daten vor große Probleme stellen, wenn er den Umsortierungsmechanismus nicht kennt. Erfolgt

[10] Diese Zahl kann in Abhängigkeit von n als Funktion beschrieben werden, bezeichnet nach L. Euler, 1707-1783

die Umsortierung geregelt, kann durch Umkehrung der Verschlüsselung der Klartext ermittelt werden.

Dieses Prinzip wurde bei Skytalen angewendet. Nach dem Zeugnis des griechischen Historikers Plutarch[11] benutzte die Regierung in Sparta diese Methode zur Übermittlung geheimer Nachrichten an ihre Generäle. Skytale waren Zylinder mit genau dem gleichen Radius. Der Sender wickelte ein schmales Band aus Pergament spiralförmig um seinen Zylinder und schrieb dann die Nachricht der Länge nach auf das Band. Auf dem abgewickelten Band konnte die Nachricht nicht gelesen werden. Ein Empfänger der Nachricht benötigte einen Zylinder gleichen Durchmessers.

Beispiel 3.24

Der Klartext $p = p_1 || \ldots || p_m$

„SICHERHEITISTDASEINZIGEZIELDERKRYPTOGRAPHIE“

hat die Länge $m = 43$. m ist eine Primzahl und deshalb ist jede Zahl aus $\mathbb{Z}_m$ und insbesondere $k = 8$ eine Einheit in $\mathbb{Z}_m$. k erzeugt die additive Gruppe $(\mathbb{Z}_m, +_m, 0)$ und wir können damit in diesem Fall jedes Element in $\mathbb{Z}_m$ durch $e_k(q) = q \cdot k$ mod m mit $q \in \mathbb{Z}_m$ darstellen (ein Element ist genau dann Erzeuger, wenn es eine Einheit im Restklassenring ist). Es ergibt sich

$$c_i = p_{e_k(i-1)+1}, \quad i = 1, \ldots, m.$$

Für das unterstrichen dargestellte „H“ an Position $q = 3$ ergibt sich als neue Position $q_{\text{neu}} = 38$, $c_{39} = p_4$. Wir erhalten

„SIEIYHRDGRRCINLTEESZRPETIEGITIEPIHAEKAHSZDO“.

Immer lässt sich eine Permutationschiffre über eine matrizenartige Anordnung der aufeinanderfolgenden Zeichen des Klartextes erstellen, wie es auch bei den Skytalen getan wurde,

S	I	C	$\underline{H}$	E	R	H	E
I	T	I	S	T	D	A	S
E	I	N	Z	I	G	E	Z
I	E	L	D	E	R	K	R
Y	P	T	O	G	R	A	P
H	I	E					

und es ergibt sich bei spaltenweiser Leserichtung der Geheimtext

„SIEIYHITIEPICINLTEHSZDOETIEGRDGRRHAEKAESZRP“.

[11] $\Pi\lambda o\acute{\upsilon}\tau\alpha\varrho\chi o\varsigma$, ca. 45-125

Ordnen wir den Text folgendermaßen an (indem wir Zeilen permutieren)

$$\begin{array}{cccccccc} S & I & C & \underline{H} & E & R & H & E \\ G & R & A & P & H & I & E & \\ R & K & R & Y & P & T & O & \\ E & Z & I & E & L & D & E & \\ S & E & I & N & Z & I & G & \\ I & T & I & S & T & D & A, & \end{array}$$

so erhalten wir den Geheimtext

„SGRESIIRKZETCARIII$\underline{H}$PYENSEHPLZTRITDIDHEOEGAE“.

Zu einer Permutationschiffre eines Klartextes p aus $\mathbb{Z}_{26}^m$ der Länge m gehört eine Matrix $\tilde{E}_m$. Diese Matrix ist eine Einheitsmatrix E_m, deren Einträge permutiert sind. Die allgemeine Verschlüsselungsfunktion ist $e_{\tilde{E}_m}(\vec{p}) = \tilde{E}_m \cdot \vec{p}$ mit $\vec{p} \in \mathbb{Z}_{26}^m$. Die Entschlüsselung erfolgt über

$$d_{\tilde{E}_m}(e_{\tilde{E}_m}(\vec{p})) = \tilde{E}_m^T \cdot (\tilde{E}_m \cdot \vec{p}) = \vec{p}.$$

Damit stellt die Permutationschiffre ein lineares Verschlüsselungsverfahren dar, einen Spezialfall der Hill-Chiffre, mit der Interpretation, dass hierbei Positionen von Zeichen und nicht die Zeichen selbst verändert werden. Die Permutationsverschlüsselung ist damit eine spezielle Substitutionsverschlüsselung.

Beispiel 3.25

Der Klartext „GEHEIMNIS“ wird durch Anwendung der Matrix

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

auf den kodierten Klartext auf den Geheimtext „HNEEIMGIS“ abgebildet. Der Geheimtext kann mittels der inversen, d.h. transponierten Matrix wieder in den Klartext umgewandelt werden. Die Darstellung der Positionsänderungen kann in einer so genannten Zyklennotation erfolgen: (1723) bedeutet dabei, dass Position 1 zu Position 7, 7 zu 2, 2 zu 3 und 3 zu 1 wird. Alle anderen Positionen bleiben unverändert. $(179652)(348)$ ergibt dann „EIIHMSGEN“.

Kryptoanalyse von Substitutionschiffren

Wir gehen weiter vom Großbuchstabenalphabet aus. Für die Substitutionschiffre stehen $26 \cdot \phi(26) = 312$ verschiedene Schlüssel zur Verfügung und ermöglichen einen einfachen

Brute-Force-Angriff. Dagegen gibt es allgemein bei der monoalphabetischen Verschlüsselung $26! \approx 2^{88}$ verschiedene Schlüssel, was eine Brute-Force-Analyse nicht möglich macht (ca. $4.9 \cdot 10^6$ Jahre). Dennoch ist es auch hier nicht allzu schwer, den Geheimtext zu entschlüsseln. Wie bei der Caesarchiffre in Beispiel 2.1 ist eine Suche nach Buchstabenhäufigkeiten für einzelne Buchstaben, Paare oder noch mehr zusammengesetzte vonnöten. Bei polyalphabetischen ($m > 1$) Chiffren wie etwa der Vigenere-Chiffre ist zunächst die Blocklänge herauszufinden. Dies kann z.B. mit dem Kasiski-Test[12] erfolgen. Dabei werden wiederholte Buchstabengruppen in Geheimtexten gesucht und deren Abstände ermittelt. Der größte gemeinsame Teiler bzw. ein Teiler davon könnte die Blockgröße sein. Für jeden Block genügend großer Länge kann anschließend durch Häufigkeitsanalysen auf den Schlüssel geschlossen werden. Es handelt sich hier um einen Ciphertext-Only-Angriff.

Beispiel 3.26

Gegeben sei der mit einer Vigenere-Chiffre erzeugte Geheimtext

„YMNKKVSHOXTVZHLVKMYCOKPCOIWGKVVUETERMVLSNMPA“.

Die unterstrichenen Buchstabengruppen haben die Abstände vier bzw. 24. Somit könnte die Blocklänge $m = 4$ oder $m = 2$ sein. Angenommen $m = 4$, müssen wir etwa die Buchstabenfolge YKOZKOOKEMN betrachten. K bzw. O könnten für E und I stehen. Allerdings sind die Anzahlen hier zu gering, um verlässliche Aussagen treffen zu können. Zum Nachvollziehen: der Schlüssel ist hier „GELD“ und unser altbekannter Klartext wurde verschlüsselt.

Ist es möglich, Klartexte $p_0, \ldots, p_m$ und mit dem gleichen Schlüssel verschlüsselte Geheimtexte $c_0, \ldots, c_m$ zu untersuchen, kann eine allgemeine affin-lineare Blockchiffre bei Kenntnis der Blocklänge m mit $m+1$ solchen Paaren (p_i, c_i) angegriffen werden. Denn es gilt

$$\underbrace{\begin{pmatrix} c_1 - c_0 & \ldots & c_m - c_{m-1} \end{pmatrix}}_{C} = A \cdot \underbrace{\begin{pmatrix} p_1 - p_0 & \ldots & p_m - p_{m-1} \end{pmatrix}}_{P}.$$

Ist P invertierbar, so folgt unmitelbar $A = C \cdot P^{-1}$. Durch eine geeignete Wahl von Klartexten lässt sich eine invertierbare Matrix P finden. Weiter ist dann z.B.

$$b = c_0 - Ap_0 \bmod 26.$$

Nach Anwendung des Kasiski-Tests kann mit Hilfe solcher Klartext-Geheimtext-Blöcke ein Angriff auf das Chiffriersystem versucht werden.

[12] W. Kasiski, 1805-1881

Beispiel 3.27

Wir setzen das Beispiel 3.26 fort. Die Blocklänge wird mit $m = 4$ angesetzt. Mit den Paaren (SICH,YMNK), (ERHE,KVSH), (ITIS,OXTV), (TDAS,ZHLV) und (ERKR,KVVU) ergibt sich folgendes Gleichungssystem:

$$\underbrace{\begin{pmatrix} 12 & 4 & 11 & 11 \\ 9 & 2 & 10 & 14 \\ 5 & 1 & 18 & 10 \\ 23 & 14 & 0 & 25 \end{pmatrix}}_{C} = A \cdot \underbrace{\begin{pmatrix} 12 & 4 & 11 & 11 \\ 9 & 2 & 10 & 14 \\ 5 & 1 & 18 & 10 \\ 23 & 14 & 0 & 25 \end{pmatrix}}_{P}.$$

Mit ggT(det$(P), 26) = 1$ ist P invertierbar und es folgt, dass A die Einheitsmatrix E_4 ist und wir erhalten den Schlüssel „GELD“.

Wir haben einen erfolgreichen Known-Plaintext-Angriff durchgeführt.

3.3. DES

Der Data Encrypton Standard (DES) ist seit 1977 eine der bekanntesten und am meisten genutzten Chiffriersysteme. Er nutzt eine Schlüssellänge von 56 Bit, $\mathcal{K} = \mathbb{F}_2^{56}$, und verschlüsselt 64-Bit Blöcke

$$p = b_1 b_2 b_3 b_4 b_5 \dots b_{63} b_{64}$$

in 64-Bit Blöcke und damit ist $\mathcal{P} = \mathcal{C} = \mathbb{F}_2^{64}$. Ein wesentliches Element darin ist eine so genannte Feistel-Chiffre.

Feistel-Chiffre
Eine in der Praxis häufig benutze Klasse kryptographischer Funktionen ist die Feistel-Chiffre. Dabei wird in n Durchgängen (Runden) eine geradzahlige Bitfolge p der Länge $2q$ auf eine Bitfolge c derselben Länge abgebildet. Zunächst wird p in zwei Hälften zerlegt, $p = L_0 || R_0$. In der i-ten Runde wird eine Bitfolge $L_i || R_i$ aus $L_{i-1} || R_{i-1}$ durch

$$\begin{aligned} L_i &:= R_{i-1}, \\ R_i &:= f_{k_i}(R_{i-1}) \oplus L_{i-1} \end{aligned} \tag{3.2}$$

und einer Abbildung $f_{k_i} : \mathbb{F}_2^q \to \mathbb{F}_2^q$ bestimmt. Um aus c wieder p zu erhalten, müssen die Runden umgekehrt werden,

$$\begin{aligned} R_{i-1} &:= L_i, \\ L_{i-1} &:= f_{k_i}(R_{i-1}) \oplus R_i. \end{aligned} \tag{3.3}$$

Zum Ver- und Entschlüsseln wird dieselbe Funktion benutzt, die deshalb nicht injektiv sein muss.

Beispiel 3.28

Wir betrachten die Bitfolge 10100111001001. Mit der Blocklänge $2m = 14$, der Rundenzahl $n = 6$, $f_k(x) := x \oplus k$ und $k_{i+1} := k_i \oplus (k_i + 1_2)$, $k_0 = 0$, ergibt sich

i	k_i	L_i	R_i
0	0_2	1010011_2	1001001_2
1	1_2	1001001_2	11011_2
2	11_2	11011_2	1010001_2
3	111_2	1010001_2	1001101_2
4	1111_2	1001101_2	10011_2
5	11111_2	10011_2	1000001_2
6	111111_2	1000001_2	1101101_2

Algorithmus
Der DES beginnt mit einer initialen Permutation des Eingabeblocks,

$$\mathrm{IP} : \mathbb{F}_2^{64} \to \mathbb{F}_2^{64},$$

mit folgenden Positionierungen:

Tabelle 3.1.: IP

58	50	42	34	26	18	10	2	60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6	64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1	59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5	63	55	47	39	31	23	15	7

Dies ist folgendermaßen zu interpretieren:

$$\mathrm{IP}(b_1 b_2 b_3 b_4 b_5 \dots b_{63} b_{64}) = b_{58} b_{50} b_{42} b_{34} b_{26} \dots b_{15} b_7 =: \tilde{p}.$$

Nun folgt eine Feistelchiffre gemäß (3.2) mit 16 Runden. Die Bitfolge $\tilde{p}$ wird in zwei je 32 Bit lange Bitfolgen $\tilde{p} = L_0 || R_0$, $L_0, R_0 \in \mathbb{F}_2^{32}$, geteilt. Für jede der 16 Runden wird ein Rundenschlüssel $k_i \in \mathbb{F}_2^{48}$, $i = 1, \dots, 16$, benötigt, der sukzessive beginnend mit $k \in \mathbb{F}_2^{56}$ erzeugt wird. Der Schlüssel k wird zunächst um so genannte Paritätsbits aufgebläht,

$$\mathrm{Par} : \mathbb{F}_2^{56} \to \mathbb{F}_2^{64},$$

indem nach je sieben Bits ein Bit eingefügt wird. Dies kann etwa durch den Even-Parity Modus geschehen: ist die Summe der Eins-Bits der sieben Bits gerade, so wird das Paritätsbit 0, ansonsten 1. Das dient lediglich als einfacher Kontrollwert und spielt im weiteren Verlauf keine Rolle. Anschließend wird eine Abbildung

$$\mathrm{PC}_1 : \mathbb{F}_2^{64} \to \mathbb{F}_2^{56}$$

mit den Positionswechseln

Tabelle 3.2.: PC_1

57	49	41	33	25	17	9	1	58	50	42	34	26	18
10	2	59	51	43	35	27	19	11	3	60	52	44	36
63	55	47	39	31	23	15	7	62	54	46	38	30	22
14	6	61	53	45	37	29	21	13	5	28	20	12	4

vorgenommen. Dies entspricht einer Permutation der 56 Schlüsselbits, die acht Paritätsbits werden verworfen. Die 56 Bits werden in zwei Hälften mit je 28 Bits geteilt und jeweils mittels einer Abbildung

$$\mathsf{LS}_i : \mathbb{F}_2^{28} \to \mathbb{F}_2^{28},\ i = 1, \ldots, 16,$$

zyklisch um

Tabelle 3.3.: Shiftweite für LS_i

1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

Bits nach links geschoben (erst zweimal um ein, dann sechsmal um 2 Bits usw.). Für den i-ten Rundenschlüssel k_i werden nach der i-ten Verschiebung die beiden Hälften zusammengefügt und eine Abbildung

$$\mathsf{PC}_2 : \mathbb{F}_2^{56} \to \mathbb{F}_2^{48}$$

mit den Positionswechseln

Tabelle 3.4.: PC_2

14	17	11	24	1	5	3	28	15	6	21	10
23	19	12	4	26	8	16	7	27	20	13	2
41	52	31	37	47	55	30	40	51	45	33	48
44	49	39	56	34	53	46	42	50	36	29	32

benutzt, wobei die Bits $9, 18, 22, 25, 35, 38, 43, 54$ entfallen. Zusammen ergibt sich der Schlüssel k_i als Abbildung

$$k_i = \mathsf{PC}_2 \circ \begin{pmatrix} \mathsf{LS}_i \\ \mathsf{LS}_i \end{pmatrix} \circ \begin{pmatrix} \mathsf{LS}_{i-1} \\ \mathsf{LS}_{i-1} \end{pmatrix} \circ \ldots \circ \begin{pmatrix} \mathsf{LS}_1 \\ \mathsf{LS}_1 \end{pmatrix} \circ \mathsf{PC}_1 \circ \mathsf{Par} : \mathbb{F}_2^{56} \to \mathbb{F}_2^{48}.$$

Die Abbildungen $f_{k_i} : \mathbb{F}_2^{32} \to \mathbb{F}_2^{32}$ in den einzelnen Runden bilden das wichtigste Element des DES. Zunächst wird die 32 Bit Eingabe auf 48 Bit über

$$E : \mathbb{F}_2^{32} \to \mathbb{F}_2^{48}$$

gemäß

Tabelle 3.5.: E

32	1	2	3	4	5	4	5	6	7	8	9
8	9	10	11	12	13	12	13	14	15	16	17
16	17	18	19	20	21	20	21	22	23	24	25
24	25	26	27	28	29	28	29	30	31	32	1

aufgebläht (keine Permutation!) und mit dem 48 Bit Rundenschlüssel XOR-verknüpft,

$$\oplus_{f_{k_i}} : \mathbb{F}_2^{48} \times \mathbb{F}_2^{48} \to \mathbb{F}_2^{48},\ (a, k_i) \mapsto u := a \oplus k_i.$$

Die 48 Bit Folge u wird in acht Anteile zu je sechs Bit aufgeteilt: $u = u_1||u_2||\ldots||u_8$. Auf jeden Teil wird eine Substitutionsbox $S_i : \mathbb{F}_2^6 \to \mathbb{F}_2^4$, $i = 1, \ldots, 8$ angewandt.

Tabelle 3.6.: S_1

14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

Tabelle 3.7.: S_2

15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9

Tabelle 3.8.: S_3

10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12

Tabelle 3.9.: S_4

7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14

Tabelle 3.10.: S_5

2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3

Tabelle 3.11.: S_6

12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13

Tabelle 3.12.: S_7

4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12

Tabelle 3.13.: S_8

13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

Die Substitutionsboxen sind folgendermaßen zu lesen: $S_i(b_1b_2b_3b_4b_5b_6)$ ist die Binärzahl, die sich aus der Zahl in der Zeile $b_1b_6 \in \{0, 1, 2, 3\}$ und der Spalte $b_2b_3b_4b_5 \in \{0, 1, \ldots, 15\}$ (jeweils bei Null zu zählen beginnen) ergibt. Beispielsweise für $u_7 = 110010$ ist $S_7(u_7) =$

1111, da in der 2. Zeile und 9. Spalte von S_7 15 steht. Zusammen erhalten wir eine Abbildung

$$S : \mathbb{F}_2^{48} \to \mathbb{F}_2^{32},\ u \mapsto S(u) = \begin{pmatrix} S_1(u_1) \\ S_2(u_2) \\ S_3(u_3) \\ S_4(u_4) \\ S_5(u_5) \\ S_6(u_6) \\ S_7(u_7) \\ S_8(u_8) \end{pmatrix}.$$

Es folgt noch eine Permutation

$$P : \mathbb{F}_2^{32} \to \mathbb{F}_2^{32}$$

mit den Positionswechseln

Tabelle 3.14.: P

16	7	20	21	29	12	28	17	1	15	23	26	5	18	31	10
2	8	24	14	32	27	3	9	19	13	30	6	22	11	4	25

Nach den 16 Runden werden die beiden Bitfolgen L_{16} und R_{16} zusammengebracht und mittels $T : F_2^{64} \to \mathbb{F}_2^{64}$ und

Tabelle 3.15.: T

33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32

vertauscht. Zum Abschluss wird die zur initialen Permutation IP inverse Permutation IP^{-1} durchgeführt:

Tabelle 3.16.: IP^{-1}

40	8	48	16	56	24	64	32	39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30	37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28	35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26	33	1	41	9	49	17	57	25

Zusammenfassend lässt sich die Rundenfunktion schreiben als

$$f_{k_i} = P \circ S \circ \oplus_{f_{k_i}} \circ E.$$

Sicherheit des DES

Wie in Tabelle 2.3 zu sehen, sind 56 Bit Schlüssel sehr schnell mittels eines Brute-Force-Angriffs herauszufinden. Deswegen gilt DES heute als nicht mehr sicher. Um das Konzept des Verfahrens nicht aufgeben zu müssen, wurden verschiedene Überlegungen angestellt, wie DES zu einem sicheren Verfahren ausgebaut werden kann. Eine Überlegung bestand darin, das Verfahren zweimal hintereinander mit zwei Schlüsseln auszuführen,

$$c = e_{k_2}(e_{k_1}(p)).$$

Ist jedoch einem Angreifer ein Klartext-Geheimtext-Paar (p, c) bekannt, kann er es folgendermaßen nutzen. Es ist

$$e_{k_1}(p) = x := d_{k_2}(c).$$

Damit kann ein so genannter Meet-in-the-middle Angriff durchgeführt werden, indem für alle möglichen k_1 und k_2 jeweils Tabellen mit den entsprechenden Werten für x erstellt und miteinander verglichen werden. Alle Übereinstimmungen ergeben Kandidaten für den Schlüssel mit der Schlüssellänge von 112 Bit. Wegen der Blockgröße ergeben sich 2^{64} verschiedene Schlüsseltexte, je Schlüssel im Mittel demnach 2^{48} Schlüsseltexte. Mit einem weiteren Klartext-Geheimtext-Paar wird der richtige fast sicher gefunden.
Eine andere Variante ist der so genannte Triple-DES,

$$c = e_{k_3}(d_{k_2}(e_{k_1}(p))).$$

Dabei wird nach der ersten Verschlüsselung mit dem zweiten Schlüssel entschlüsselt, um dann ein weiteres Mal zu verschlüsseln. Ein Nachteil dieser Variante ist allerdings die um das dreifache gestiegene Laufzeit.

4. Trapdoor-Einwegfunktionen

Im Jahr 1976 veröffentlichten Diffie[1] und Hellman[2] das nach wie vor gültige Prinzip der Public-Key-Kryptographie (asymmetrische Chiffrierung). Bis zu diesem Zeitpunkt mussten zwei Instanzen, die geheim kommunizieren wollten, schon vorher ein gemeinsames Geheimnis haben, den geheimen Schlüssel (Verwendung symmetrischer Chiffriersysteme). Seit den Überlegungen von Diffie und Hellmann ist es möglich, ohne jeden vorherigen Kontakt einer Instanz eine verschlüsselte Nachricht zu schicken, die nur sie selbst entschlüsseln kann. Grundlage für die asymmetrische Chiffrierung sind Trapdoor-Einwegfunktionen, die wir nun vorbereiten. Unter einer Einwegfunktion wollen wir eine kryptographische Funktion f verstehen, für die sich $f(x)$ leicht berechnen lässt. Zu gegebenem y lässt sich jedoch nicht mit vertretbarem Aufwand ein x mit $y = f(x)$ bestimmen; die Umkehrung ist also schwierig. Eine Trapdoor-Einwegfunktion ist eine Einwegfunktion, für die durch Kenntnis eines Geheimnisses die Umkehrung ebenso einfach zu berechnen ist. Solche Funktionen werden wir nun vorbereiten. Wir betrachten die Einwegfunktionen der

- Multiplikation von Primzahlpotenzen mit deren Umkehrung der Primfaktorzerlegung,
- Potenzierung einer Zahl mit deren Umkehrung des diskreten Logarithmus.

4.1. Primzahlenarithmetik

Satz 4.1: Primfaktorzerlegung

Sei $n \in \mathbb{N} \setminus \{1\}$. Dann gibt es Primzahlen $p_1, \ldots, p_s$ mit $n = 1 \cdot p_1 \cdot \ldots \cdot p_s$. Diese Produktdarstellung ist bis auf die Reihenfolge der Faktoren eindeutig.

Beweis.

Induktion über n. $n = 2$: $2 = 1 \cdot 2$.
$n - 1 \to n : n$ hat einen Primteiler p. Denn n hat wenigstens einen Teiler > 1, nämlich n. Aller Teiler kleinster sei p. Dann ist p eine Primzahl, da sonst ein q existiert mit $1 < q < p \leq n$ mit $q|p$ und damit $q|n$. Dies ist aber im Widerspruch dazu, dass p der kleinste Teiler von n ist. Wir betrachten nun n/p. Ist $n/p = 1$, so ist $D_n = \{1, p\}$, p prim und der Satz bewiesen, für $f := n/p > 1$ folgt wegen $f < n$ nach Induktionsvoraussetzung, dass f ein Produkt von Primfaktoren ist. Damit ist die Existenz bewiesen.

Seien

$$n = p_1 \cdot \ldots \cdot p_s \text{ und } n = q_1 \cdot \ldots \cdot q_t$$

[1] W. Diffie, *1944
[2] M. Hellman, *1945

Primfaktorzerlegungen für n mit p_i, q_j prim für alle $i = 1, \ldots, s$ und $j = 1, \ldots, t$. Dann stimmt p_1 mit einem der Primfaktoren $q_1, \ldots, q_t$ überein. Wir beweisen das durch Induktion über t. Es gilt $p_1 | n = q_1 \cdot \ldots \cdot q_t$. Für $t = 1$ ist die Aussage offensichtlich.
$t - 1 \to t$: $p_1 | q_1 \cdot (q_2 \cdot \ldots \cdot q_t)$. Es ist zu zeigen, dass $p_1 | q_1$ oder $p_1 | q_2 \cdot \ldots \cdot q_t$ gilt. Annahme: $p_1 \nmid q_1$. Da p_1, q_1 prim sind, ist ggT$(p_1, q_1) = 1$ und damit $1 = p_1 \cdot x + y \cdot q_1$ für $x, y \in \mathbb{Z}$. Daraus folgt

$$\begin{aligned}
(q_2 \cdot \ldots \cdot q_t) &= p_1 \cdot x \cdot (q_2 \cdot \ldots \cdot q_t) + y \cdot q_1 \cdot (q_2 \cdot \ldots \cdot q_t) \\
&= p_1 \cdot x \cdot (q_2 \cdot \ldots \cdot q_t) + y \cdot n \\
&= p_1 \cdot x \cdot (q_2 \cdot \ldots \cdot q_t) + y \cdot p_1 \cdot z \\
&= p_1 \cdot (x \cdot (q_2 \cdot \ldots \cdot q_t) + y \cdot z) \\
&\Rightarrow p_1 \mid (q_2 \cdot \ldots \cdot q_t)
\end{aligned}$$

Analog kann gezeigt werden, dass aus $p_1 \nmid (q_2 \cdot \ldots \cdot q_t)$ folgt $p_1 | q_1$. Im ersten Fall hat sich die Zahl der Faktoren von t auf $t - 1$ reduziert und es greift die Induktionsannahme. Im zweiten Fall ist $p_1 = q_1$. Damit ist die Behauptung bewiesen.

Durch Umnummerierung erhalten wir $p_1 = q_1$ und betrachten nun n/p_1 bzw. n/q_1. Hierfür gilt jedoch die Induktionsannahme und damit $s = t$ bzw. durch jeweilige Umnummerierung $p_i = q_i$ für alle $i = 1, \ldots, s$. Die Primfaktorzerlegung ist bis auf die Reihenfolge der Faktoren eindeutig.

□

Die Bestimmung einer Primfaktorzerlegung ist jedoch trotz Kenntnis ihrer Existenz eine komplexe Aufgabe. Gerade das macht man sich in der Kryptologie zunutze. Mit Hilfe der Primfaktorzerlegung einer Zahl $n \in \mathbb{N}$ werden wir die Anzahl der Elemente der primen Restklassengruppe $(\mathbb{Z}_n^*, \cdot_n, 1)$ bestimmen können. Dazu betrachten wir zunächst

Satz 4.2

Für eine Primzahlpotenz p^r mit p prim und $r \in \mathbb{N}$ gilt

$$\phi(p^r) = p^{r-1} \cdot (p - 1).$$

Beweis.

Es gibt p^r Kandidaten für die Menge der Einheiten. Ein solcher Kandidat sei $a \in \{0, \ldots, p^r - 1\}$. Die Zahl a lässt sich als p-adische Zahl durch

$$a = a_0 + a_1 p + \ldots + a_{r-1} p^{r-1}, \quad a_i \in \{0, \ldots, p - 1\},\ i = 0, \ldots, r - 1$$

darstellen. Nun muss ggT$(a, p^r) = 1$ gelten. Aber es gilt ggT$(a, p^r) = 1$ genau dann, wenn $a_0 \neq 0$ ist. Denn ist ggT$(a, p^r) = 1$, dann folgt $1 = a \cdot x + p^r \cdot y = a_0 \cdot x + a_1 \cdot x \cdot p + a_2 \cdot x \cdot p^2 + \ldots + a_{r-1} \cdot x \cdot p^{r-1} + y \cdot p^r$ für $x, y \in \mathbb{Z}$. Ist dann $a_0 = 0$, so gilt $p \mid 1$ und p wäre keine Primzahl, was einen Widerspruch darstellt. Somit muss $a_0 \neq 0$ gelten. Ist umgekehrt ggT$(a, p^r) = b > 1$, wobei b ein Vielfaches von p sein muss $(b = z \cdot p)$,

gilt $b = z \cdot p \mid a$ und so $p \mid a$, $a = k \cdot z \cdot p = a_0 + a_1 p + \ldots + a_{r-1} p^{r-1}$ für $k, z \in \mathbb{Z}$. Es muss $a_0 = 0$ sein.

Somit müssen sämtliche Kandidaten mit $a_0 = 0$ ausgeschlossen werden. Dies sind p^{r-1} Stück für jede Potenz von p in der p-adischen Darstellung von a. Damit folgt

$$\phi(p^r) = p^r - p^{r-1} = p^{r-1} \cdot (p-1).$$

□

Für eine Zahl $n \in \mathbb{N}$, die sich multiplikativ aus Primfaktoren zusammensetzt, wird bei Kenntnis der Primfaktorzerlegung die Bestimmung der Anzahl der Elemente in $\mathbb{Z}_n^*$ einfach.

Satz 4.3

Sei $m \in \mathbb{N}$ und

$$m = \prod_{i=1}^{n} p_i^{r_i}, \quad p_1 < p_2 < \ldots < p_n \text{ prim}, \ r_i \in \mathbb{N}$$

ihre Primfaktorzerlegung. Dann gilt für die Eulersche ϕ-Funktion

$$\phi(m) = \prod_{i=1}^{n} p_i^{r_i - 1} \cdot (p_i - 1).$$

Für $m = p_1 \cdot p_2$ mit p_1, p_2 prim ist $\phi(m) = (p_1 - 1) \cdot (p_2 - 1)$.

Beweis.

Es seien $m_i = p_i^{r_i}$ für alle $i = 1, \ldots, n$. Wegen der Eindeutigkeit der Primfaktorzerlegung müssen die m_i paarweise teilerfremd sein. Gemäß Satz A.44 gilt

$$\phi(m) = \phi(m_1) \cdot \ldots \cdot \phi(m_n) \overset{\text{Satz 4.2}}{=} \prod_{i=1}^{n} p_i^{r_i - 1} \cdot (p_i - 1)$$

Der Zusatz folgt entsprechend.

□

Beispiel 4.4

In $\mathbb{Z}_{23}^* = \{1, 2, \ldots, 22\}$ ist $\phi(23) = 23 - 1 = 22$. Für $n = 21 = 3 \cdot 7$ ist $\phi(21) = (3-1) \cdot (7-1) = 12$. Es ist $\mathbb{Z}_{21}^* = \{1, 2, 4, 5, 8, 10, 11, 13, 16, 17, 19, 20\}$. Dagegen ist, für $n = 9 = 3 \cdot 3$, $\phi(9) = 3^1 \cdot (3-1) = 6 \neq 4 = (3-1) \cdot (3-1)$ und $\mathbb{Z}_9^* = \{1, 2, 4, 5, 7, 8\}$.

Das Potenzieren in der primen Restklassengruppe $(\mathbb{Z}_n^*, \cdot_n, 1)$ ist ein Spezialfall des Satzes A.14 von Lagrange.

Satz 4.5: Satz von Euler

Sei $n \in \mathbb{N},\ n > 1$, und $a \in \mathbb{Z}$ mit ggT$(a, n) = 1$. Dann gilt: $a^{\phi(n)} \equiv_n 1$.

Beweis.

Kurzform: Fassen wir a mod n als Element von $\mathbb{Z}_n^*$ der primen Restklassengruppe $(\mathbb{Z}_n^*, \cdot_n, 1)$ mit seinen $\phi(n)$ Elementen auf, so folgt mit Satz A.14, dass $a^{|\mathbb{Z}_n^*|} = a^{\phi(n)} = (a \text{ mod n})^{\phi(n)} \equiv_n 1$.

Wir wollen den Satz noch auf eine andere Weise beweisen: Es ist ggT$(a, n) = 1$. Sei $r := a$ mod n. Dann gilt ggT$(a, n) =$ ggT$(r, n) = 1$ und damit $r \in \mathbb{Z}_n^*$. Die Elemente in der Menge $\{x \cdot_n r;\ x \in \mathbb{Z}_n\} = \{0 \cdot_n r, 1 \cdot_n r, 2 \cdot_n r, \ldots, (n-1) \cdot_n r\}$ sind paarweise verschieden, da r Erzeuger von $(\mathbb{Z}_n, +_n, 0)$ ist (vgl. Satz 3.17). $r \cdot_n x \in \mathbb{Z}_n^*$ genau dann, wenn $x \in \mathbb{Z}_n^*$ ist. Denn einerseits ist mit $r, x \in \mathbb{Z}_n^*$ auch $r \cdot_n x \in \mathbb{Z}_n^*$. Sei andererseits $r \cdot_n x \in \mathbb{Z}_n^*$ und damit invertierbar, d.h. es gibt ein $y \in \mathbb{Z}_n^*$ mit $1 = y \cdot_n (r \cdot_n x) = (y \cdot_n r) \cdot_n x \in \mathbb{Z}_n^*$. Da nun $y \cdot_n r \in \mathbb{Z}_n^*$ ist, ist x dazu das inverse Element in $\mathbb{Z}_n^*$. Also $x \in \mathbb{Z}_n^*$. Alle invertierbaren Elemente in $\mathbb{Z}_n$ sind von der Form $r \cdot_n x$. Sind $r_1, \ldots, r_{\phi(n)} \in \mathbb{Z}_n^*$ die invertierbaren Elemente in $\mathbb{Z}_n$, so sind wegen der paarweisen Verschiedenheit auch $r \cdot_n r_1, \ldots, r \cdot_n r_{\phi(n)}$ sämtliche invertierbaren Elemente und es gilt

$$\begin{aligned}
& r_1 \cdot_n \ldots \cdot_n r_{\phi(n)} &=& \ (r \cdot_n r_1) \cdot_n \ldots \cdot_n \left(r \cdot_n r_{\phi(n)}\right) \\
& &=& \ r^{\phi(n)} \cdot_n \left(r_1 \cdot_n \ldots \cdot_n r_{\phi(n)}\right) \\
\Leftrightarrow \quad & 1 &=& \ r^{\phi(n)} \text{ mod } n \\
\Leftrightarrow \quad & 1 &=& \ (a \text{ mod } n)^{\phi(n)} \text{ mod } n \\
\Leftrightarrow \quad & 1 &=& \ a^{\phi(n)} \text{ mod } n \\
\Leftrightarrow \quad & 1 &\equiv_n& \ a^{\phi(n)}
\end{aligned}$$

□

Beispiel 4.6

Bringen wir die Ergebnisse zusammen. Man betrachte $n = 21$, $k = 11$ und $x = 5$. Wegen 5^5 mod $21 = 17$ und 5^6 mod $21 = 1$ ist dann x^k mod $n = 5^{11}$ mod $21 = 17$. Um nun aus 17 wieder die 5 zu bestimmen, brauchen wir das Inverse zu k mod $\phi(n)$. Mit $\phi(n) = 12$ und ggT$(11, 12) = 1$ erhalten wir mit dem erweiterten Euklidischen Algorithmus: $k' = 11$, denn $11 \cdot 11$ mod $12 = 1$. Somit ist 17^{11} mod $21 = 5$. Für genügend große Primzahlen n ist die Bestimmung von $\phi(n)$ nach heutigem Stand nicht in überschaubarer Zeit möglich. Wer also nur (n, k) als Schlüssel hat, bekommt Probleme bei der Entschlüsselung. Wir kommen darauf im RSA-Algorithmus noch einmal zurück.

Eine einfache Folgerung aus Satz 4.5 für eine Primzahl $n \in \mathbb{N}$ gibt

Satz 4.7: Kleiner Satz von Fermat

Sei $n \in \mathbb{N}$ eine Primzahl und $a \in \mathbb{Z}$ mit ggT$(a, n) = 1$. Dann gilt $a^{n-1} \equiv_n 1$.

Beweis.

Da n eine Primzahl ist, gilt $\phi(n) = n - 1$. Mit Satz 4.5 folgt die Aussage.

□

Wir haben mit der multiplikativen Gruppe $(\mathbb{Z}_n^*, \cdot_n, 1)$ bestimmte Exponentialausdrücke a^k und deren Verhalten betrachtet. Die Umkehrung davon soll uns im weiteren beschäftigen. Zur Vorbereitung des diskreten Logarithmus benötigen wir an dieser Stelle noch folgendes Resultat.

Satz 4.8

Sei $n \in \mathbb{N},\ a \in \mathbb{Z}_n^*$ und $\text{ord}_{\mathbb{Z}_n^*}(a) = m$ bzgl. $(\mathbb{Z}_n^*, \cdot_n, 1)$. Mit $Z_n(a) := \{a^x \text{ mod } n;\ 0 \leq x < m\}$ gilt

(1) $|Z_n(a)| = m$,

(2) $(Z_n(a), \cdot_n, 1)$ ist eine zyklische Untergruppe von $(\mathbb{Z}_n^*, \cdot_n, 1)$ und

(3) $m \mid \phi(n)$.

Beweis.

(1). Seien $k, j \in \mathbb{N}_0,\ 0 \leq k < j < m$ und $a^k \text{ mod } n = a^j \text{ mod } n \ \Rightarrow\ a^{j-k} \text{ mod } n = 1$. Da $0 < j - k < m$ gilt, ist dies ein Widerspruch zur Ordnung von a in $\mathbb{Z}_n^*$.
(2). Es ist $a^0 = 1 \in Z_n(a)$. Seien $x, y \in \mathbb{N}_0$ und $0 \leq x < y < m$. Dann sind $a^x \text{ mod } n, a^y \text{ mod } n \in Z_n(a)$. Wir betrachten $a^x \cdot_n a^y = a^{x+y} \text{ mod } n$. Für $x + y < m$ ist $a^{x+y} \text{ mod } n \in Z_n(a)$. Für $x + y \geq m$ setzen wir $0 \leq r := (x + y) - m \cdot k < m$ für ein $k \in \mathbb{N}$. Wegen $a^m \text{ mod } n = 1$ gilt $a^{x+y} \text{ mod } n = a^{x+y} \cdot a^{m\cdot(-k)} \text{ mod } n = a^{(x+y)-m\cdot k} \text{ mod } n = a^r \text{ mod } n \in Z_n(a)$. Damit ist $Z_n(a)$ abgeschlossen unter der Multiplikation. Jedes $b \in Z_n(a)$ lässt sich eindeutig schreiben als $b = a^x \text{ mod } n,\ x \in \{0, \ldots, m-1\} \ \Rightarrow\ c := a^{m-x} \text{ mod } n$ ist wegen $b \cdot_n c = a^m \text{ mod } n = 1$ das Inverse in $Z_n(a)$ zu b. Insgesamt ist unter Berücksichtigung der Assoziativität von Potenzen gemäß Definition A.10 $(Z_n(a), \cdot_n, 1)$ eine von a erzeugte Untergruppe und damit zyklisch.
(3). Das folgt aus Satz A.13. Oder: Seien $q, r \in \mathbb{N}_0$. $\phi(n) = m \cdot q + r,\ 0 \leq r < m$. $\Rightarrow\ 1 = a^{\phi(n)} \text{ mod } n = a^{m\cdot q+r} \text{ mod } n = (a^m)^q \cdot a^r \text{ mod } n = a^r \text{ mod } n.\ \Rightarrow\ r = 0\ \Rightarrow\ \phi(n) = m \cdot q\ \Rightarrow\ m \mid \phi(n)$.

□

Die Aussagen gelten insbesondere für Primzahlen. Es existiert in jeder primen Restklassengruppe $(\mathbb{Z}_p^*, \cdot_p, 1)$ für eine Primzahl p wenigstens ein Element $a \in \mathbb{Z}_p^*$, für dessen Ord-

nung $\text{ord}_{\mathbb{Z}_p^*}(a) = p-1$ gilt, diese Gruppe ist gemäß Satz A.30 zyklisch mit der Ordnung $p-1$. Für diskrete Logarithmen in der Kryptographie ist das ein zentrales Ergebnis. Der Beweis dazu ist aber umfangreich und deshalb verzichten wir an dieser Stelle darauf und formulieren lediglich

Satz 4.9: Zyklische prime Restklassengruppe

Für eine Primzahl p ist $(\mathbb{Z}_p^*, \cdot_p, 1)$ eine zyklische Gruppe und $\mathbb{Z}_p^*$ enthält $\phi(p-1)$ Erzeuger.

Zum Beweis des Satzes 4.9 benötigen wir folgende auch für später relevante Aussage.

Satz 4.10

Sei $n \in \mathbb{Z}$, $(G, \circ, e_\circ)$ eine Gruppe und $\text{ord}_G(g) = u$ für ein $g \in G$. Dann gilt $\text{ord}_G(g^n) = u/\text{ggT}(u,n)$.

Beweis.

Es gilt $(g^n)^{u/\text{ggT}(u,n)} = (g^u)^{n/\text{ggT}(u,n)} = e_\circ$ und $u/\text{ggT}(u,n)$ ist ein Vielfaches von $\text{ord}_G(g^n)$, d.h. $\text{ord}_G(g^n) \mid u/\text{ggT}(u,n)$. Sei $e_\circ = (g^n)^k = g^{n\cdot k}$ für ein $k \in \mathbb{N}$. Mit $d := \text{ggT}(n,e)$ gibt es teilerfremde $v, w \in \mathbb{Z}$ mit $u = d \cdot v$ und $n = d \cdot w$. Es gilt $u \mid n \cdot k$ d.h. $n \cdot k = q \cdot u$ und damit $k \cdot w \cdot d = q \cdot v \cdot d$. v teilt $k \cdot w$ und da v und w teilerfremd sind, gilt $u/\text{ggT}(u,n) \mid k$. Dies gilt insbesondere auch, wenn $k = \text{ord}_G(g^n)$. Damit ist $\text{ord}_G(g^n) = u/\text{ggT}(u,n)$.

□

Beweis.: zu Teil 2 von Satz 4.9

Der Beweis, dass $(\mathbb{Z}_p^*, \cdot_p, 1)$ eine zyklische Gruppe ist, erfolgt durch Körper- und Polynombetrachtungen, führt an dieser Stelle jedoch zu weit. Es sei auf [3], Abschnitt 3.22, verwiesen. Mit dem Wissen, dass $(\mathbb{Z}_p^*, \cdot_p, 1)$ eine zyklische Gruppe ist, führen wir den Beweis der zweiten Aussage. Sei $g \in \mathbb{Z}_p^*$ mit $\text{ord}_{\mathbb{Z}_p^*}(g) = u$. Dann ist nach Satz 4.8 $|\langle g \rangle| = u$. Ein Element von $\mathbb{Z}_p^*$ ist dessen Erzeuger, wenn es die Ordnung $p-1$ hat. Da nun $(\mathbb{Z}_p^*, \cdot_p, 1)$ zyklisch ist, gibt es wenigstens einen Erzeuger. Sei g ein Erzeuger von $\mathbb{Z}_p^*$. Dann ist $\mathbb{Z}_p^* = Z_p(g) = \{g^k \bmod p;\ 0 \le k < p-1\}$. Sei $h \in \mathbb{Z}_p^*$, $h = g^k$ für ein $0 \le k < p-1$. Dann gilt mit Satz 4.10

$$\text{ord}_{\mathbb{Z}_p^*}(h) = p-1 \iff \text{ggT}(k, p-1) = 1.$$

Die Menge $\mathbb{Z}_p^*$ enthält damit $\phi(|\mathbb{Z}_p^*|) = \phi(p-1)$ Erzeuger.

□

Beispiel 4.11

In $(\mathbb{Z}_7^*, \cdot_7, 1)$ gibt es sechs Elemente, jedoch nur $\phi(6) = 1 \cdot 2 = 2$ Erzeuger: $\langle 3 \rangle = \langle 5 \rangle = \mathbb{Z}_7^*$.

4.2. Primfaktorzerlegung und diskreter Logarithmus

Betrachtet man ein Chiffriersystem $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$, so ist eine Verschlüsselungsfunktion $e_{k_1} : \mathcal{P} \to \mathcal{C}$ eine Trapdoor-Einwegfunktion, falls e_{k_1} trotz Kenntnis des Schlüssels $k_1 \in \mathcal{K}$ (und damit der Kenntnis der kompletten Funktion e_{k_1}) praktisch (im Sinne der Komplexitätstheorie) nicht umkehrbar ist, also aus $c = e_{k_1}(p)$ das entsprechende p nicht berechnet werden kann (*Einwegfunktionen*), auf der anderen Seite mit Hilfe einer Zusatzinformation $k_2 \in \mathcal{K}$ eine einfach auszuwertende Funktion $d_{k_2} : \mathcal{C} \to \mathcal{P}$ angegeben werden kann mit $d_{k_2}(e_{k_1}(p)) = p$ für alle $p \in \mathcal{P}$ (*Trapdoor-Eigenschaft*). Ist nun für jedes $k \in \mathcal{K}$ die Funktion e_k eine *Trapdoor-Einwegfunktion*, so kann man $|\mathcal{K}|$ Teilnehmern jeweils einen Schlüssel k_i zuordnen und sowohl den Teilnehmer als auch den entsprechenden Schlüssel publizieren („Telefonbuch“). Die Frage, ob es überhaupt Trapdoor-Einwegfunktionen gibt, ist bis heute unbeantwortet. Man kann allerdings Funktionen angeben, die nach dem jetzigen Stand der Technik Trapdoor-Einwegfunktionen sind. Dazu betrachten wir die algebraische Struktur $(\mathbb{Z}_n, \cdot_n, 1)$ mit der Multiplikation gemäß Definition 2.13. Für jedes $k \in \mathbb{N}$ kann gemäß Definition A.11 die k-te Potenz einer Zahl definiert werden. Damit können wir zumindest zwei mögliche Kandidaten für Trapdoor-Einwegfunktionen angeben.

Einwegfunktion des Primzahlproduktes

Satz 4.12

Sei $n = p \cdot q$ das Produkt zweier Primzahlen $p \neq q$ und seien e_k Funktionen

$$e_k : \mathbb{Z}_n \to \mathbb{Z}_n,\ x \mapsto x^k \bmod n,\ k \in \mathbb{N}$$

für alle k, die teilerfremd zu $\phi(n) = (p-1) \cdot (q-1)$ sind. Dann lässt sich zeigen, dass für jedes dieser k die jeweilige Funktion e_k bijektiv ist und dass es ein $k' \in \mathbb{N}$ gibt mit

$$d_{k'} : \mathbb{Z}_n \to \mathbb{Z}_n :\ x \mapsto x^{k'} \bmod n \ \text{ und } d_{k'}(e_k(x)) = e_k(d_{k'}(x)) = x \quad \forall x \in \mathbb{Z}_n.$$

Die natürliche Zahl k' ist charakterisiert durch

$$k \cdot k' \equiv_{\phi(n)} 1.$$

Beweis.

Gegeben seien $n = p \cdot q,\ n \in \mathbb{N},\ p, q$ prim und $x \in \mathbb{Z}_n,\ k \in \mathbb{N}$. Sei $k \cdot k' \equiv_{\phi(n)} 1$. Dann

sind k und k' teilerfremd und es gilt

$$\begin{aligned}
k \cdot k' \equiv_{\phi(n)} 1 \quad &\Leftrightarrow \quad \phi(n) | (k \cdot k' - 1) \\
&\Leftrightarrow \quad k \cdot k' - 1 = z \cdot \phi(n) \text{ für } z \in \mathbb{Z} \\
&\Leftrightarrow \quad k \cdot k' = z \cdot \phi(n) + 1. \\
&\Leftrightarrow \quad x^{k \cdot k'} \bmod n = x^{z \cdot \phi(n)+1} \bmod n \\
&= \quad [(x^{\phi(n)})^z \bmod n \cdot x] \bmod n = [(x^{\phi(n)} \bmod n)^z \bmod n \cdot x] \bmod n \\
&\overset{\text{Satz } 4.5}{=} \quad [((r \cdot n + 1) \bmod n)^z \bmod n \cdot x] \bmod n \\
&= \quad [1^q \bmod n \cdot x] \bmod n = [1 \cdot x] \bmod n = x.
\end{aligned}$$

□

Bemerkung. Im Beweis ist der seltene Spezialfall, dass x entweder durch p oder durch q teilbar ist, nicht untersucht. Dabei gilt der Satz von Euler nicht.

Kennt man die Primfaktorzerlegung $n = p \cdot q$ von n, so lässt sich k' aus p und q relativ einfach mit dem erweiterten euklidischen Algorithmus berechnen. Kennt man aber nur die Zahlen n und k, welche die Funktion e_k vollständig charakterisieren, so ist zwar e_k leicht auszuwerten aber bis heute p und q (und damit k') im allgemeinen nicht in adäquater Zeit zu berechnen. Das Problem der Primfaktorzerlegung lautet hier somit: Finde zu einer gegebenen Zahl $n \in \mathbb{N}$ zwei Primzahlen p und q mit $p \neq q$ und $n = p \cdot q$.

Beispiel 4.13

Sei $n = 77 = 7 \cdot 11$ und $k = 17$. Sei $\mathcal{P} = \mathcal{C} = \{., /, \ldots, y, z\}$ die Menge der $ASCII$-Werte von dezimal 46 bis 122. Dann wird mit $\phi(77) = 60$ und ggT$(17, 60) = 1$ der Klartext „KRYPTOGRAPHIE“ zu

Zeichen	p [ASCII]	$x := p - 46$	$x := x^{17} \bmod 77$	$x := c + 46$ [ASCII]	Zeichen
K	75	29	50	96	‘
R	82	36	64	110	n
Y	89	43	43	89	Y
P	80	34	34	80	P
T	84	38	47	93	]
O	79	33	66	112	p
G	71	25	9	55	7
R	82	36	64	110	n
A	65	19	24	70	F
P	80	34	34	80	P
H	72	26	38	84	T
I	73	27	69	115	s
E	69	23	67	113	q

„‘nYP]p7nFPTsq “. Um das entschlüsseln zu können, benötigen wir k' mit $k \cdot k' = 1 \bmod 60$. Es ist $k' = 53$ und damit wird etwa aus „‘“ über $50^{53} \bmod 77 = 29$ wieder das „K“.

Exponentiationschiffren benötigen wir für das in Abschnitt 6.1 zu besprechende RSA-Verfahren, das auf den Ausführungen in diesem Abschnitt beruht.

Einwegfunktion der modularen Exponentiation

Für jede Primzahl p ist die Einheitengruppe von $\mathbb{Z}_p$ die multiplikative Gruppe $(\mathbb{Z}_p^*, \cdot_p, 1)$ mit $\mathbb{Z}_p^* = \{1, \ldots, p-1\}$. Es gibt in $\mathbb{Z}_p$ mindestens ein erzeugendes Element g mit $\mathbb{Z}_p^* = \langle g \rangle = \{g^x \bmod p;\ x \in \{0, \ldots, p-1\}\}$, wie wir bereits in Satz 4.9 festgehalten haben. Betrachten wir nun die Abbildung

$$\exp_g : \mathbb{Z}_p^* \to \mathbb{Z}_p^*,\ x \mapsto g^x \bmod p$$

für jedes erzeugende Element g von $(\mathbb{Z}_p^*, \cdot_p)$. Die Abbildung $\exp_g$ ist bijektiv, denn g erzeugt alle Elemente von $\mathbb{Z}_p^*$ und jedes Element von $\mathbb{Z}_p^*$ wird eindeutig erzeugt. Analog zum Vorgehen in der Analysis bezeichnen wir nun die Umkehrabbildung zu $\exp_g$ als diskreten Logarithmus:

$$\log_g : \mathbb{Z}_p^* \to \mathbb{Z}_p^*,\ x \mapsto \log_g x \quad \text{mit } g^{\log_g x} \bmod p = x.$$

Das Problem des diskreten Logarithmus in $(\mathbb{Z}_p^*, \cdot_p, 1)$ lautet somit folgendermaßen:

> Finde zur Basis g mit $\langle g \rangle = \mathbb{Z}_p^*$ und zu $b \in \mathbb{Z}_p^*$ ein $x \in \mathbb{Z}_p^*$ mit $g^x \bmod p = b$.

Wir schreiben $\log_g b = x$, etwa in $\mathbb{Z}_5$: $\log_3(1) = 4$ wegen $3^4 \bmod 5 = 1$. Da diese Funktion im allgemeinen nur mit größtem Aufwand auszuwerten ist (im Gegensatz zu $\exp_g$), ist die Exponentialfunktion zu jedem erzeugenden Element g für ein spezielles p eine Einwegfunktion. Denn wenn p groß genug ist und $p-1$ wenigstens einen „großen“ Primfaktor besitzt, dann lässt sich auf klassischen Rechnern kein Polynomialzeit-Algorithmus zur Auswertung angeben. Allerdings sind diese Funktionen zunächst für die Kryptographie nicht geeignet, da keine Trapdoor-Information zur Verfügung steht. Dennoch können wir die diskreten Logarithmusfunktionen für Trapdoor-Einwegfunktionen verwenden, wenn wir dem zu verschlüsselnden Geheimtext noch gewisse Zusatzinformationen beifügen. Aus der reinen Einwegfunktion wird dann eine Trapdoor-Einwegfunktion. Dies geschieht im Verschlüsselungsverfahren von El Gamal, das wir uns bald ansehen werden.

Beispiel 4.14

Sei $n = 79$. Sei $\mathcal{P} = \mathcal{C} = \{., /, \ldots, y, z, -, |\}$ die Menge der $ASCII$-Werte von dezimal 46 bis 124. Es ist $\langle 6 \rangle = \mathbb{Z}_{79}^*$. Dann wird der Klartext „KRYPTOGRAPHIE“ zu

Zeichen	p [ASCII]	$b := p - 46$	$x := \log_6 b$	$c := x + 46$ [ASCII]	Zeichen
K	75	29	49	95	_
R	82	36	2	48	0
Y	89	43	41	87	W
P	80	34	5	51	3
T	84	38	54	100	d
O	79	33	45	91	[
G	71	25	56	102	f
R	82	36	2	48	0
A	65	19	22	68	D
P	80	34	5	51	3
H	72	26	70	116	t
I	73	27	63	109	m
E	69	23	52	98	b

„_0W3d[f0D3tmb“. Um z.B. das erste Zeichen „_“ entschlüsseln zu können, müssen

wir nur potenzieren. Wir erhalten über 6^{49} mod $79 = 29$ wieder das Zeichen „K“. Der umgekehrte Weg, also zunächst zu potenzieren und dann zu logarithmieren, erfordert die Berechnung des Logarithmus zur Basis 6. Das ist viel aufwändiger zu bestimmen und liefert die Grundlage für verschiedene kryptographische Verfahren.

Satz 4.15: Bestimmung des diskreten Logarithmus

Das Verfahren von Shanks[a], beschrieben in Algorithmus 4.16, liefert zu einem gegebenen Erzeuger von $\mathbb{Z}_p^*$ und einem Element aus $\mathbb{Z}_p^*$ den diskreten Logarithmus.

[a] W. Shanks, 1812-1882

Algorithmus 4.16: Berechnung des diskreten Logarithmus

Eingabe: Primzahl p, Erzeuger $g \in \mathbb{Z}_p^*$ mit $\langle g \rangle = \mathbb{Z}_p^*$ und $b \in \mathbb{Z}_p^*$.
Ausgabe: $x = \log_g b$.

1: $m := \lceil \sqrt{p-1} \rceil$.
2: $L_1 := \{(j, g^{m \cdot j} \text{ mod } p);\ j = 0, \ldots, m-1\}$.
3: $L_2 := \{(i, bg^{-i} \text{ mod } p = bg^{p-1-i} \text{ mod } p);\ i = 0, \ldots, m-1\}$.
4: Finde $(j, y) \in L_1$ und $(i, y) \in L_2$ mit gleicher zweiter Komponente.
5: $x = \log_g b := (m \cdot j + i) \text{ mod } (p-1)$.
6: **return** x.

Beweis.

Es gelte $(j, y) \in L_1$ und $(i, y) \in L_2$. Dann ist $g^{m \cdot j}$ mod $p = y = b \cdot g^{-i}$ mod p und somit $g^{m \cdot j + i}$ mod $p = b$. Damit folgt $g^{m \cdot j + i - x}$ mod $p = 1$, $m \cdot j + i - x = k \cdot \phi(p) = k \cdot (p-1)$ für ein $k \in \mathbb{Z}$, $p - 1 \mid m \cdot j + i - x$ und $m \cdot j + i \equiv_{p-1} x$. Da durch $m \cdot \tilde{j} + \tilde{i},\ \tilde{j}, \tilde{i} \in \{0, \ldots, m-1\}$ alle Zahlen von 0 bis $m^2 - 1$ dargestellt werden, gibt es für alle $b \in \mathbb{Z}_p^*$ Zahlen $j, i \in \{0, \ldots, m-1\}$ mit $m \cdot j + i = \log_g b$ bzw. $m \cdot j = x - i$.

□

Das El Gamal-Verfahren und die kryptographische Anwendung elliptischer Kurven, die beide den diskreten Logarithmus verwenden, betrachten wir in den Abschnitten 6.2 und 6.3.

Schnelle modulare Exponentiation

Die modulare Exponentiation nutzt die im letzten Abschnitt beschriebene exp-Funktion. Beim Potenzieren sind oftmals Berechnungen mit sehr großen Exponenten durchzuführen (vgl. Beispiel A.38). Das Verfahren der schnellen Exponentiation ermöglicht die rasche Berechnung solcher Potenzen.

Algorithmus 4.17: Schnelle Berechnung von Potenzen modulo n

Eingabe: $n, x \in \mathbb{N}$, $a \in \mathbb{Z}$.
Ausgabe: $y = a^x \bmod n$.

```
1:  a_1 := a;  x_1 := x,  y := 1.
2:  while x_1 ≠ 0 do
3:     while (x_1 mod 2) = 0 do
4:        x_1 := x_1/2.
5:        a_1 := (a_1 · a_1) mod n.
6:     end while
7:     x_1 := x_1 − 1.
8:     y := (y · a_1) mod n.
9:  end while
10: return y.
```

Beispiel 4.18

Wir berechnen wie in Beispiel A.38 die Potenz $2^{10} \bmod 3$.

a_1	x_1	y
2	10	1
$2 \cdot 2 \bmod 3 = 1$	$10/2 = 5$	1
1	$5 - 1 = 4$	$1 \cdot 1 \bmod 3 = 1$
$1 \cdot 1 \bmod 3 = 1$	$4/2 = 2$	1
$1 \cdot 1 \bmod 3 = 1$	$2/2 = 1$	1
1	$1 - 1 = 0$	$1 \cdot 1 \bmod 3 = 1$

$5^{11} \bmod 21 = 5 \cdot 4^5 \bmod 21 = 20 \cdot 16^2 \bmod 21 = 80 \bmod 21 = 17$.

5. Schlüsselmanagement

5.1. Schlüsselaustausch nach Diffie und Hellman

Eine Möglichkeit der Verschlüsselung besteht wie wir gesehen haben darin, symmetrisch zu verschlüsseln. Dabei ist das noch offene Problem des geheimen Schlüsselaustausches zwischen zwei Instanzen gegeben. Hier kann die Public-Key-Kryptographie weiterhelfen. Soll zwischen zwei Instanzen eine verschlüsselte Nachricht übermittelt werden, besteht das Problem, dass die Empfängerinstanz in der Lage sein muss, die verschlüsselte Nachricht zu entschlüsseln. Dazu benötigt sie einen Schlüssel, der auf den zur Verschlüsselung verwendeten Schlüssel passt. Es stellt sich die prinzipielle Frage, wie beide Teilnehmer einen solchen Schlüssel vereinbaren können.

Der Austausch zwischen den Instanzen A und B erfolgt nach einem Kommunikationsprotokoll. 1976 schlugen Diffie und Hellman folgendes Protokoll 5.1 vor:

Algorithmus 5.1: Diffie-Hellman-Protokoll

Eingabe: Primzahl p, $\langle g \rangle = \mathbb{Z}_p^*$.
Ausgabe: Geheimer Schlüssel x_{AB}.

1: A wählt zufällig $x_A \in \{2, \ldots, p-2\}$ und berechnet $y_A := g^{x_A} \bmod p$.
2: B wählt zufällig $x_B \in \{2, \ldots, p-2\}$ und berechnet $y_B := g^{x_B} \bmod p$.
3: y_A geht an B, y_B geht an A.
4: A berechnet $x_{AB} := y_B^{x_A} \bmod p = (g^{x_B} \bmod p)^{x_A} \bmod p = g^{x_B x_A} \bmod p$.
5: B berechnet $x_{AB} := y_A^{x_B} \bmod p = (g^{x_A} \bmod p)^{x_B} \bmod p = g^{x_A x_B} \bmod p$.
6: **return** x_{AB} als geheimer Schlüssel beider Instanzen A und B.

Dies ist das erste Verfahren, bei dem wir den diskreten Logarithmus benutzen. Denn kennen wir zwar die Zahlen g, $g^{x_A} \bmod p$ oder $g^{x_B} \bmod p$, so ist nur unter Durchführung einer diskreten Logarithmierung gemäß Algorithmus 4.16 das Bestimmen von $g^{x_a x_b} \bmod p$ möglich.

Beispiel 5.2

Seien $p = 737932726269007356326274177547$ und $g = 2$. Dann können wir die öffentlichen und den gemeinsamen privaten Schlüssel von A und B mit $x_A = 126584$ und $x_B = 18544165841641$ bestimmen.

Teilnehmer	öffentlicher Schlüssel	gemeinsamer privater Schlüssel
A	$468899451394773548989435958 65$	$40881803088060493232199391 8072$
B	$819330503033639016558444076 69$	$40881803088060493232199391 8072$

Beim Nachrechnen des Beispiels passiert es möglicherweise, dass ein Overflow erzeugt und die Berechnung nicht durchgeführt wird. Das liegt an dem großen Exponenten. Auch hier ist notwendig, die effiziente Berechnung der modularen Potenz gemäß Algorithmus 4.17 durchzuführen.

Wie können wir einen Erzeuger, wie er im Algorithmus 5.1 gefordert wird, finden? Kennen wir die Primfaktorzerlegung gemäß Satz 4.1 von $p-1$, so ermöglicht uns die Suche nach einem Erzeuger

Satz 5.3

Sei p eine Primzahl. Für $g \in \mathbb{Z}_p^*$ gilt $\langle g \rangle = \mathbb{Z}_p^* \iff g^{\frac{p-1}{q_j}} \mod p \neq 1$ für jeden Primteiler q_j von $p-1$.

Beweis.

Sei $m := \text{ord}_{\mathbb{Z}_p^*}(g)$, d.h. $m \mid p-1$ nach Satz 4.8 (3). Weiter werde $p-1 = q_i \cdot Q_i$ für jeden Primteiler q_i von $p-1$ gemäß Satz 4.1 geschrieben. Dann: $\langle g \rangle \neq \mathbb{Z}_p^* \Leftrightarrow m < p-1 \Leftrightarrow p-1 = v \cdot m$ für $v \in \mathbb{Z}, v \neq 1 \Leftrightarrow p-1 = q_i \cdot Q_i = v \cdot m \Leftrightarrow \exists i: \ m \mid q_i \vee m \mid Q_i$. Teilt m die Zahl q_i, so gibt es ein j mit $p-1 = q_j \cdot Q_j$ und $q_i \mid Q_j$ oder q_i teilt bereits Q_i. In jedem Fall gilt $p-1 = q_i \cdot Q_i = v \cdot m \Leftrightarrow \exists j : p-1 = q_j \cdot Q_j \wedge m \mid Q_j \Leftrightarrow p-1 = q_j \cdot Q_j = q_j \cdot k_j \cdot m$ für $k_j \in \mathbb{Z} \Leftrightarrow (p-1)/q_j = Q_j = k_j \cdot m \Leftrightarrow g^{\frac{p-1}{q_j}} \mod p = g^{k_j \cdot m} \mod p = (g^m)^{k_j} \mod p = 1$.

□

Für die Primzahl p im Diffie-Hellmann-Protokoll wird oft eine spezielle Art von Primzahl gefordert.

Definition 5.4: Sichere Primzahl

Sei $q \in \mathbb{N}$ eine Primzahl. Ist $p := 2 \cdot q + 1$ eine Primzahl, so heißt p eine sichere Primzahl.

Für eine sichere Primzahl ist es zudem recht schnell möglich, einen Erzeuger zu finden. Denn mit Satz 5.3 wissen wir, dass für ein $g \in \mathbb{Z}_p^*$, wenn p eine sichere Primzahl ist, ledig-

lich zu überprüfen ist, ob $g^{\frac{p-1}{2}}$ mod $p = g^q$ mod $p \neq 1$ bzw. $g^{\frac{p-1}{q}}$ mod $p = g^2$ mod $p \neq 1$ ist.

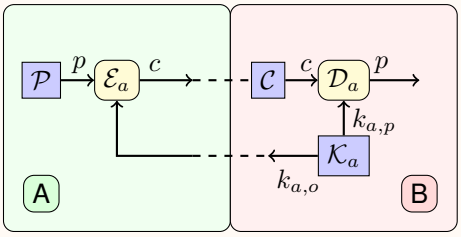
Weiter teilt die Ordnung m eines Elements $g \in \mathbb{Z}_p^*$ einer sicheren Primzahl p die Zahl $p-1$, $m \mid 2 \cdot q$ und es gilt: entweder $m = 2$ oder $m = q$ oder $m = 2 \cdot q$ und es gibt $\phi(p-1) = q - 1 = \frac{p-1}{2} - 1 = \frac{p-3}{2}$ Erzeuger, also fast die Hälfte aller Elemente erzeugt $\mathbb{Z}_p^*$.

Sei p eine Primzahl mit $p > 2$ und $\langle g \rangle = \mathbb{Z}_p^*$. Für jedes $s < p-1$ gilt g^s mod $p \neq 1$, da g Erzeuger ist. Das gilt insbesondere für $s = (p-1)/2$. Dann ist g^{2s} mod $p = g^{p-1}$ mod $p = 1$. Wir suchen somit eine Zahl $h \in \mathbb{Z}_p^*$, die quadriert modulo p den Wert Eins ergibt. Da das gemäß Satz A.29 nur die beiden Zahlen 1 und $p-1$ leisten und $h = 1$ wegen unserer Überlegungen ausscheidet, muss $h = p-1$ sein. Eine notwendige Bedingung eines Erzeugers g ist damit $g^{(p-1)/2}$ mod $p = p-1$.

6. Asymmetrische Chiffrierung

Eine Alternative zur symmetrischen Verschlüsselung stellt die asymmetrische Verschlüsselung dar. Möchte eine Instanz A Daten an eine Instanz B schicken, so muss B einen öffentlichen Schlüssel $k_{a,o}$ zur Verfügung stellen, mit dem A seine Daten verschlüsseln kann. Da nur B im Besitzt des privaten Schlüssels $k_{a,p}$ ist, kann nur B die empfangenen Daten entschlüsseln. Abbildung 6.1 zeigt das Prinzip der asymmetrischen Kryptographie.

Abbildung 6.1.: Asymmetrische Chiffrierung



6.1. RSA-Verfahren

Das von Rivest[1], Shamir[2] und Adleman[3] 1978 erfundene und nach ihnen benannte RSA-Verfahren verwendet die im letzten Abschnitt angegebenen Trapdoor-Einwegfunktionen

$$e_k : \mathbb{Z}_n \to \mathbb{Z}_n,\ x \mapsto x^k \bmod n,\ k \in \mathbb{N}$$

mit öffentlichem Schlüssel (n, k), der Primfaktorzerlegung $n = p \cdot q$ mit $p \neq q$ prim, dem geheimen Schlüssel k' mit

$$k \cdot k' = 1 \bmod \phi(n)$$

und $(x^k)^{k'} = (x^{k'})^k = x \bmod n$ für alle $x \in \mathbb{Z}_n$.

Algorithmus 6.1: RSA Schlüsselerzeugung

Eingabe: $n = p \cdot q$ mit p, q prim.
Ausgabe: Öffentlicher Schlüssel (n, k).
1: $\phi(n) := (p-1) \cdot (q-1)$.
2: Finde k mit $1 < k < \phi(n)$ und ggT$(k, \phi(n)) = 1$ (öffentlicher Schlüssel (n, k)).

[1] R. Rivest, *1947
[2] A. Shamir, *1952
[3] L. Adleman, *1945

3: $k' := k^{-1} \bmod \phi(n)$ (privater Schlüssel (n, k')).
4: **return** (n, k).

Algorithmus 6.2: RSA-Public-Key-Algorithmus

Eingabe: Zur Verschlüsselung: Öffentlicher Schlüssel (n, k), Klartextblock M (kodiert mit $0 \leq M < n$). Zur Entschlüsselung: Privater Schlüssel (n, k').
Ausgabe: Geheimtext c, entschlüsselter Klartext M'.
1: $c := e_{(n,k)}(M) = M^k \bmod n$.
2: $M' := d_{(n,k')}(c) = c^{k'} \bmod n$.
3: **return** c, M'.

Entscheidend für die Sicherheit des RSA-Verfahrens ist die Größe der beiden Primzahlen p und q, die in Binärdarstellung durch die Anzahl der nötigen Bits charakterisiert werden. Heute werden für p und q wenigstens 512 Bits (entspricht etwa 160 Dezimalstellen) vorgeschlagen. Somit entspricht n einer natürlichen Zahl aus 1024 Bits. Nehmen wir nun an, unser Klartext bestehe aus Buchstaben, Ziffern und Sonderzeichen, so wird zum Beispiel im einfachen ASCII-Code jedes der möglichen 128 Zeichen durch 7 Bits, im erweiterten ASCII durch 8 Bits dargestellt (siehe Anhang B).

Somit entspricht einer Zahl $m \in \{0, 1, ..., 2^{1024} - 1\}$ (darstellbar durch 1024 Bits) ein Text bestehend aus maximal 128 Zeichen $(128 \cdot 8 = 1024)$. Der Klartext „KRYPTOGRAPHIE“ wird durch eine natürliche Zahl $m \approx 1.01 \cdot 10^{31}$ dargestellt. Ein längerer Klartext wird demnach blockweise verschlüsselt.

Beispiel 6.3

Im Beispiel 4.13 musste jedes Zeichen einzeln verschlüsselt werden, da $n = 77$ gewählt wurde und somit gerade einmal 6 volle Bits (genauer: 77 mögliche Zeichen) für einen Verschlüsselungsblock vorhanden waren. Wählen wir stattdessen $p = 131$ und $q = 139$, so erhalten wir $n = 18209$. Da nun $2^{14} = 16384 < n$ ist, können wir 2 Zeichen im ASCII-Code zusammen verschlüsseln. Mit $k = 17,\ \phi(18209) = 17940$ und

ggT$(17, 17940) = 1$ wird der Klartext „KRYPTOGRAPHIE_“ zu

Zeichen	Binärcode des ASCII-Wertes
K	1001011
R	1010010
Y	1011001
P	1010000
T	1010100
O	1001111
G	1000111
R	1010010
A	1000001
P	1010000
H	1001000
I	1001001
E	1000101
$\_$	1011111

Jeweils zwei Zeichen fassen wir zu einem Block zusammen. Deshalb haben wir das (im Prinzip beliebige) Füllzeichen „_“ hinzugefügt.

Zeichenblock	Binärblock	Dezimalzahl x	x^{17} mod 18209	Binärdarstellung
KR	10010111010010	9682	6180	01100000100100
YP	10110011010000	11472	1616	00011001010000
TO	10101001001111	10831	10279	10100000100111
GR	10001111010010	9170	11790	10111000001110
AP	10000011010000	8400	1695	00011010011111
HI	10010001001001	9289	154	00000010011010
$E\_$	10001011011111	8927	4085	00111111110101

Jeden Ergebnisblock spalten wir wieder in zwei Teilblöcke auf und bestimmen damit die einzelnen Zeichen des Chiffretextes:

Binärblock	Zeichenblock	Dezimalzahl x
$0110000, 0100100$	$48, 36$	0 und \$
$0001100, 1010000$	$12, 80$	¸ und P
$1010000, 0100111$	$80, 39$	P und ’
$1011100, 0001110$	$92, 14$	\ und ‹
$0001101, 0011111$	$13, 31$	, und ffl
$0000001, 0011010$	$1, 26$	´ und ȷ
$0011111, 1110101$	$31, 117$	ffl und u

Damit erhalten wir den Chiffretext „ 0$¸PP’\‹,ffl ´ȷfflu “. Um das entschlüsseln zu können, benötigen wir k' mit $k \cdot k' = 1$ mod 17940. Es ist $k' = 10553$ und damit wird etwa aus „0$“ über $6180^{10553} \bmod 18209 = 9682$ wieder das „KR“.

Zwei entscheidende Nachteile des RSA-Verfahrens führten auf die Suche nach anderen Trapdoor-Einwegfunktionen und damit auch alternativer kryptographischer Verfahren.

1. Sobald es jemandem gelingt, die Primfaktorzerlegung natürlicher Zahlen effizient durchführen zu können, muss das RSA-Verfahren sofort ersetzt werden. Es ist dann zu spät, mit der Suche nach Alternativen zu beginnen.

2. Die bisherigen Algorithmen zur Primfaktorzerlegung zwingen den Kryptographen, Primzahlen mit mindestens 1024 Bits zu verwenden. Diese Größen machen die Ver- und Entschlüsselung zu sehr komplexen Vorgängen. Gesucht sind Public-Key-Verfahren, die mit kleineren Zahlen auskommen.

6.2. Das El Gamal-Verschlüsselungsverfahren

Wir betrachten für p prim die multiplikative Gruppe $(\mathbb{Z}_p^*, \cdot_p, 1)$ mit einem erzeugenden Element g. Eine Instanz A, die mit einem öffentlichen Schlüssel in ein Verzeichnis eingetragen werden will, wählt zufällig ein $x_A \in \{2, \ldots, p-2\}$ und berechnet aus $y_A := g^{x_A} \bmod p$ seinen öffentlichen Schlüssel (p, g, y_A). Der dazugehörige private Schlüssel ist x_A.

Möchte eine Instanz B der Instanz A mit dem besorgten öffentlichen Schlüssel (p, g, y_A) eine Nachricht M senden, stellt sie ihre Nachricht durch ein Element aus $\mathbb{Z}_p^*$ dar und wählt zufällig eine Zahl $x_B \in \mathbb{Z}_p^*$. Dann berechnet sie die Zahlen $y_B := g^{x_B} \bmod p$ und $c := M \cdot y_A^{x_B} \bmod p$.

Instanz B verwendet damit eine (symmetrische) Multipliaktionschiffre mit Schlüsselmenge $\mathcal{K} = \mathbb{Z}_p^*$ und Verschlüsselungsfunktion e_k für $x_{AB} = y_A^{x_B} \bmod p \in \mathcal{K}$ und sendet die geheime Nachricht $c = e_k(M) = M \cdot_p x_{AB}$ und die Information y_B mit dem berechneten Schlüssel an die Empfängerinstanz A.

Mit Hilfe des privaten Schlüssels x_A kann die Nachricht entschlüsselt werden. Denn setzt man $z := (y_B^{x_A})^{-1} \bmod p$, so ergibt sich

$$\begin{aligned} c \cdot z \bmod p &= M \cdot y_A^{x_B} \cdot (y_B^{x_A})^{-1} \bmod p = M \cdot (g^{x_A})^{x_B} \cdot ((g^{x_B})^{x_A})^{-1} \bmod p \\ &= M \cdot (g^{x_B})^{x_A} \cdot ((g^{x_B})^{x_A})^{-1} \bmod p = M \cdot 1 \bmod p = M. \end{aligned}$$

In diesem Fall kann man $z = (y_B^{x_A})^{-1} \bmod p = y_B^{p-1} \cdot (y_B^{x_A})^{-1} \bmod p = y_B^{p-1-x_A} \bmod p$ einfach berechnen. Fassen wir $x_{AB} = y_A^{x_B} \bmod p$ als Schlüssel der symmetrischen Verschlüsselung auf, so gilt

$$x_{AB} = y_A^{x_B} \bmod p = (g_A^x)^{x_B} \bmod p = (g^{x_B})^{x_A} \bmod p = y_B^{x_A} \bmod p$$

und die Instanz A kann mit dem geheimen Schlüssel x_A den Schlüssel x_{AB} unter Verwendung von y_B berechnen und somit c entschlüsseln. Dies entspricht der Situation beim Schlüsselaustausch nach Diffie-Hellmann. Ein Angreifer, der nur p, g, y_A und y_B kennt, hat somit $x_A = \log_g(y_A) \bmod p$ bzw. $x_B = \log_g(y_B) \bmod p$ zu berechnen, um den Schlüssel zu erhalten.

Algorithmus 6.4: El-Gamal Schlüsselerzeugung mit sicherer Primzahl

Eingabe: $p := 2q + 1$ sichere Primzahl, $q \in \mathbb{N}$ prim und $g \in \mathbb{Z}_p^*$ mit $\langle g \rangle = \mathbb{Z}_p^*$.
Ausgabe: Öffentlicher Schlüssel (p, g, y_A).
1: Bestimme zufällig $x_A \in \{2, \ldots, p-2\}$ (privater Schlüssel).
2: $y_A := g^{x_A} \bmod p$.
3: **return** (p, g, y_A).

Algorithmus 6.5: El-Gamal-Public-Key-Algorithmus

Eingabe: Zur Verschlüsselung: Öffentlicher Schlüssel (p, g, y_A), Klartextblock M (kodiert mit $M \leq p$). Zur Entschlüsselung: Privater Schlüssel x_A.
Ausgabe: Geheimtext $\tilde{c}$, entschlüsselter Klartext p'.
1: Bestimme $x_B \in \{2, \ldots, p-2\}$.
2: $y_B := g^{x_B}$ mod p und $c := M \cdot y_A^{x_B}$ mod p.
3: $\tilde{c} := (y_B, c)$.
4: $z := y_B^{-x_A}$ mod p.
5: $M' := c \cdot z$ mod p.
6: **return** $\tilde{c}, M'$.

Bemerkung.
(1) p muss keine Primzahl sein. Es muss nur sichergestellt werden, dass eine zyklische Gruppe großer Ordnung benutzt wird und die Wahlen von x und u in $\mathbb{Z}_p^*$ den Vorgaben entsprechen.
(2) Die Länge der Nachricht wird beim El-Gamal-Verfahren verdoppelt.

Beispiel 6.6

Wir möchten den Klartext „KR“ wie beim RSA-Verfahren übermitteln. Dazu wählen wir uns zunächst die prime Restklasse $\mathbb{Z}_{18743}^*$. Denn 18743 ist eine Primzahl und wir benötigen, um 14 Bits zu verschlüsseln, eine genügend große Zahl größer $2^{14} = 16384$. Ferner verwenden wir $g = 15$, denn $18743 - 1 = 18742 = 2 \cdot 9371$ und wegen ggT$(18742, 15) = 1$, 15^2 mod $18742 \neq 1$ und 15^{9371} mod $18742 \neq 1$ ist g ein Erzeuger von $\mathbb{Z}_{18743}^*$. Sei $x_A = 17333$. Dann ergibt sich

$$y_A = 15^{17333} \text{ mod } 18743 = 16385.$$

Der öffentliche Schlüssel ist somit $(18743, 15, 16385)$ und der private Schlüssel ist 17333. Sei nun $x_B = 5554$. Da der Klartext „KR“ der Zahl 9682 entspricht, erhalten wir

$$\begin{aligned} y_B &= 15^{5554} \text{ mod } 18743 = 18100, \\ x_{AB} &= 16385^{5554} \text{ mod } 18743 = 7373 \text{ und} \\ c &= 9682 \cdot 16385^{5554} \text{ mod } 18743 = 12042. \end{aligned}$$

Somit lautet die verschlüsselte (ASCII-)Nachricht „^‘“ mit $12042_2 = 10111100001010$ und damit $1011110|0001010 = 94_{10}|10_{10}$.
Um das wieder entschlüsseln zu können, benötigen wir den privaten Schlüssel $x_A = 17333$ und

$$z = x_{AB}^{-1} = 18100^{18743-1-17333} \text{ mod } 18743 = 18100^{1409} \text{ mod } 18743 = 1660.$$

Wir entschlüsseln mit $12042 \cdot 1660$ mod $18743 = 9682$ genau die übermittelte Botschaft „KR“.

Neben dem Geheimtext wird eine Zusatzinformation (Trapdoorinformation) gesendet. Damit ist der gesendete Text doppelt so lang geworden. Die Frage ist nun, ob wir einfach die

weiteren Blöcke des vollständigen Klartextes „KRYPTOGRAPHIE_“ mit der selben Konfiguration wie aus dem letzten Beispiel übermitteln können.

Beispiel 6.7

Wir möchten in Erweiterung des letzten Beispiels den Text „YP“ analog übermitteln. Das entspricht der Zahl 11472 und damit ergibt sich $c = 14640$ bzw. der Geheimtext „r0“ was wir wieder zu $14640 \cdot 1660 \bmod 18743 = 11472$ entschlüsseln können.

Für die Entschlüsselung ist hierbei jeweils der gleiche private Schlüssel x notwendig. Somit könnte ein Angreifer bei Kenntnis eines Klartextblockes die anderen aus diesem ohne Kenntnis des geheimen Schlüssels berechnen.

Beispiel 6.8

Seien der Klartext „YP“ (11472), $c_1 = 12042, c_2 = 14640$ und $p = 18743$ dem Angreifer bekannt. Dann gilt

$$c_1 \cdot c_2^{-1} \bmod p = c_1 \cdot c_2^{p-2} \bmod p = M_1 \cdot y_A^{x_B} \cdot M_2^{p-2} \cdot y_A^{x_B(p-2)} \bmod p = M_1 M_2^{p-2} \bmod p.$$

Damit erhalten wir $M_1 = c_1 \cdot c_2^{-1} M_2 \bmod p$. In unserem Beispiel ergibt sich mit $14640^{-1} \bmod 18743 = 13613$ dann „KR“ über $9682 = 12042 \cdot 13613 \cdot 11472 \bmod 18743$.

Jeder neue Klartext sollte damit mit einer anderen Konfiguration verschlüsselt werden und für einen langen Klartext sollte eine dementsprechend große Primzahl gewählt werden.

Das El Gamal-Verschlüsselungsverfahren ist verallgemeinerbar und nicht auf $\mathbb{Z}_p^*$ beschränkt. Das Verfahren auf andere Gruppen zu verallgemeinern bedeutet jedoch, zwei Aspekte zu berücksichtigen:

- Das Grundprinzip muss immer erhalten bleiben: Sicherheit gewährleistet die Schwierigkeit, den diskreten Logarithmus berechnen zu müssen.
- Die Gruppenoperationen müssen aus Effizienzgründen einfach auszuführen sein.

6.3. Kryptographie mit elliptischen Kurven

Bisher haben wir den diskreten Logarithmus für die multiplikativen Gruppen $(\mathbb{Z}_p^*, \cdot_p, 1)$ betrachtet. Analog dazu können wir ihn auch für andere zyklische Gruppen definieren. In der additiven Gruppe $(\mathbb{Z}_p, +_p, 0)$ mit einer Primzahl p lautet die der Exponentialfunktion entsprechende Abbildung gemäß Definition A.11

$$e_g : \mathbb{Z}_p \to \mathbb{Z}_p,\ x \mapsto \begin{cases} g +_p \ldots +_p g =: x \cdot_p g, & x \neq 0, \\ 0, & x = 0. \end{cases}$$

für ein erzeugendes Element g, d.h. für ein Element mit $\langle g \rangle = \{k \cdot g \bmod p;\ k \in \mathbb{Z}_p\} = \mathbb{Z}_p$. In $(\mathbb{Z}_p, +_p, 0)$ sind gemäß Satz 3.17 alle von Null verschiedenen Elemente $g \in \mathbb{Z}_p \setminus \{0\}$ erzeugende Elemente. Die Abbildung e_g mit $g \in \mathbb{Z}_p \setminus \{0\}$ ist auch bijektiv, da alle Elemente

in $\langle g \rangle$ paarweise verschieden sind und erzeugt werden. Daher existiert eine dem diskreten Logarithmus entsprechende Umkehrabbildung

$$d_g : \mathbb{Z}_p \to \mathbb{Z}_p,\ x \mapsto d_g(x) \text{ mit } e_g(d_g(x)) = x.$$

Das Problem des diskreten Logarithmus in $(\mathbb{Z}_p, +_p, 0)$ lautet also: Finde zur Basis $g \in \mathbb{Z}_p \setminus \{0\}$ und zu $c \in \mathbb{Z}_p$ ein x mit $x \cdot_p g = c$. Die Lösung ist $x = g^{-1} \cdot_p c$, wobei g^{-1} das multiplikative Inverse von g in $\mathbb{Z}_p^*$ bezeichnet. Da das inverse Element in Z_p^* mit dem erweiterten Euklidischen Algorithmus effizient berechenbar ist, muss man nach anderen endlichen zyklischen oder allgemein kommutativen (additiven) Gruppen suchen, in denen der diskrete Logarithmus praktisch nicht auszuwerten ist. Diese kommutativen Gruppen erhalten wir unter anderem durch elliptische Kurven.

Beispiel 6.9: El Gamal in additiven Gruppen

Wir betrachten $(\mathbb{Z}_{23}, +_{23}, 0)$, $g = 17$. Es gilt $\langle 17 \rangle = \mathbb{Z}_{23}$. Mit $x_A = 5$ erhalten wir zunächst $y_A = 17 \cdot 5 \bmod 23 = 16$. Der öffentliche Schlüssel ist $(23, 17, 16)$ der private 5. Die Nachricht $M = 11$ erzeugt mit $x_B = 2$ den Geheimtext $x_{AB} = 17 \cdot 2 \bmod 23 = 11$ und $c = 11 + 16 \cdot 2 \bmod 23 = 20$, also $(11, 20)$. Zur Entschlüsselung bestimmen wir $z = -(11 \cdot 5) \bmod 23 = 14$. Der entschlüsselte Klartext lautet dann $20 \cdot 14 \bmod 23 = 11$. Die zweite Möglichkeit besteht darin, x_B auszurechnen: $x_B = g^{-1} \cdot_p x_{AB} = 19 \cdot_{23} 11 = 2$ und so $M = c - y_A^{x_B} \bmod p = 20 - 16 \cdot 2 \bmod 23 = 11$.

Um den Begriff der elliptischen Kurve vorzubereiten, benötigen wir

Definition 6.10: Ebene affine Kurve

Sei in einem Körper $(K, +, \cdot, 0, 1)$ ein Polynom $f : K \times K \to K$ mit $(x, y) \mapsto f(x, y) = \sum_{i,j>0} a_{ij} x^i y^j$ mit $a_{ij} \in K$ gegeben, wobei für endlich viele $a_{ij} \neq 0$ gilt. Dann heißt

$$\tilde{C}_f(K) = \{(x, y) \in K \times K;\ f(x, y) = 0\}$$

ebene affine Kurve über K.

Wir betrachten nun folgendes Polynom vom Grad 3 mit $a_{ijk} = 0$ für $i + j + k \neq 3$

$$\begin{aligned} g : K \times K \times K \to K, \quad & (x, y, z) \mapsto g(x, y, z) \\ g(x, y, z) &= a_{300}x^3 + a_{210}x^2y + a_{201}x^2z + a_{120}xy^2 + a_{111}xyz + a_{102}xz^2 \\ &+ a_{030}y^3 + a_{021}y^2z + a_{012}yz^2 + a_{003}z^3. \end{aligned}$$

Für jede Nullstelle $(x_0, y_0, z_0) \in K \times K \times K$ von g liefert $(\lambda x_0, \lambda y_0, \lambda z_0)$ bei Auswertung des Polynoms g wiederum die Null. Damit lässt sich eine Äquivalenzrelation

$$(x_0, y_0, z_0) \sim (u, v, w) \ :\Leftrightarrow\ (u, v, w) = (\lambda x_0, \lambda y_0, \lambda z_0) \text{ für ein } \lambda \in K \setminus \{0\}$$

bestimmen und (x_0, y_0, z_0) ist ein Repräsentant einer Äquivalenzklasse, die wir mit $[x_0 : y_0 : z_0]$ bezeichnen. Damit schreiben wir

Definition 6.11: Ebene projektive Kurve

Eine ebene projektive Kurve ist die Quotientenmenge

$$C_g(K) = \{[x_0 : y_0 : z_0];\ g(x_0, y_0, z_0) = 0\} = (K \times K \times K \setminus \{(0,0,0)\})/\sim,$$

also die Menge der Äquivalenzklassen von $\sim$.

Für die spezielle Funktion

$$\begin{aligned} \tilde{f} : K \times K \to K,\ \tilde{f}(x,y) &= a_{300}x^3 + a_{210}x^2y + a_{201}x^2 + a_{120}xy^2 + a_{111}xy + a_{102}x \\ &+ a_{030}y^3 + a_{021}y^2 + a_{012}y + a_{003} \end{aligned}$$

können wir die injektive Abbildung $i : \tilde{C}_{\tilde{f}}(K) \to C_g(K),\ i(x,y) = [x : y : 1]$ definieren. Denn aus $i(x,y) = i(x',y')$ folgt $[x : y : 1] = [x' : y' : 1]$ und $x = tx',\ y = ty',\ 1 = t1$, damit $t = 1$ und $x = x',\ y = y'$. Unter dieser Abbildung verstehen wir somit $\tilde{C}_{\tilde{f}}(K)$ als Teilmenge von $C_g(K)$. $\tilde{C}_{\tilde{f}}(K)$ lässt sich über die Abbildung i in die ebene projektive Kurve $C_g(K)$ einbetten. Darüber hinaus gibt es jedoch weitere Nullstellen von g, auf die Elemente von $\tilde{C}_{\tilde{f}}(K)$ über i nicht abbildbar sind. Zusammengefasst sind dies die Äquivalenzklassen $[u : v : 0]$, wobei $u \neq 0$ oder $v \neq 0$ mit $u, v \in K$ ist. Insgesamt haben wir also

$$C_g(K) = i(\tilde{C}_f(K)) \cup \{[u : v : 0]\}.$$

Definition 6.12: Elliptische Kurve

Eine elliptische Kurve ist eine nicht-singuläre ebene projektive Kurve $C_g(K)$ über einem Körper K, wobei g das folgende so genannte Weierstraß-Polynom[a]

$$g(x,y,z) = y^2z + a_1xyz + a_3yz^2 - x^3 - a_2x^2z - a_4xz^2 - a_6z^3$$

ist mit $a_1, a_2, a_3, a_4, a_6 \in K$.

[a] K. Weierstraß, 1815-1891

Bemerkung.
(1) Nicht-singulär bedeutet, dass sämtliche Elemente aus $C_g(K)$ einfache Nullstellen des Polynoms sind. Dies kann über eine Diskriminante[4] Δ gewährleistet werden.
(2) Nunmehr können wir das Weglassen der Nulllösung von g begründen: Für das Betrachten elliptischer Kurven brauchen wir die Nicht-Singularität, die im Nullpunkt nicht gegeben ist.

Bevor wir elliptische Kurven über $\mathbb{Z}_p$ ansehen, konstruieren wir zunächst allgemein für elliptische Kurven über $\mathbb{R}$ eine Verknüpfung $\oplus$, um eine Gruppenstruktur zu erhalten, auf der wir kryptographisch arbeiten können.

[4] Die Herleitung kann in [12] nachgelesen werden.

Elliptische Kurven über $\mathbb{R}$ und $\mathbb{Z}_p$

Für Körper mit mehr als drei Elementen[5] lässt sich das Weierstraß-Polynom noch vereinfachen [12]. Seien $a, b \in \mathbb{R}$. Wir betrachten folgende Gleichungen in drei Unbekannten $x, y, z \in \mathbb{R}$:

$$y^2 \cdot z = x^3 + a \cdot x \cdot z^2 + b \cdot z^3. \tag{6.1}$$

Wir suchen alle $(x', y', z') \in \mathbb{R}^3$, die für feste $a, b \in \mathbb{R}$ die obige Gleichung erfüllen. Dabei sind zwei Lösungen (x', y', z') und (x'', y'', z'') äquivalent, falls es ein $\lambda \in \mathbb{R}$ mit $\lambda \neq 0$ gibt, so dass

$$(x'', y'', z'') = (\lambda x', \lambda y', \lambda z').$$

Interessant sind alle nicht-äquivalenten und nicht-singulären Lösungen, von denen wir uns immer genau eine in einer Menge $E(a, b)$ notieren. Alle nicht-äquivalenten Lösungen der obigen Gleichung lassen sich wegen $(z = 0 \Rightarrow x = 0)$ folgendermaßen darstellen:

$$E(a, b) = \{(x', y', 1);\ y'^2 = x'^3 + a \cdot x' + b,\ x', y' \in \mathbb{R}\} \cup \{(0, 1, 0)\}.$$

Die Nicht-Singularität aller Lösungen können wir gewährleisten, indem wir fordern, dass für die bereits erwähnte Diskriminante Δ gilt:

$$\Delta := \frac{a^3}{27} + \frac{b^2}{4} \neq 0.$$

Damit kann gezeigt werden, dass für jeden Punkt aus $E(a, b)$ mindestens eine der partiellen Ableitungen $\frac{\partial f}{x'}$ bzw. $\frac{\partial f}{y'}$ des entsprechenden (vereinfachten) Weierstraßpolynoms von Null verschieden sind: Es sei $f(x, y) = x^3 + ax + b - y^2$. Dann gilt

$$\begin{aligned} \frac{\partial f}{\partial x}(x, y) &= 3x^2 + a \stackrel{!}{=} 0 \Leftrightarrow x = \pm\sqrt{-\frac{a}{3}} \\ \frac{\partial f}{\partial y}(x, y) &= -2y \stackrel{!}{=} 0 \Leftrightarrow y = 0 \end{aligned}$$

Wegen

$$H_f(x, y) = \begin{pmatrix} 6x & 0 \\ 0 & -2 \end{pmatrix}$$

liegt entweder ein Sattelpunkt oder ein lokales Maximum vor. Für welche a, b liegt ein solcher stationärer Punkt in $E(a, b)$?

$$y^2 = x^3 + ax + b \Leftrightarrow 0 = \pm\left(-\frac{a}{3}\right)^{\frac{3}{2}} + a\left(\pm(\sqrt{-\frac{a}{3}})\right) + b \Rightarrow \frac{a^3}{27} = \frac{b^2}{4}.$$

Insbesondere gibt es dann immer einen ($\Delta > 0$) oder drei ($\Delta < 0$) verschiedene Schnittpunkt(e) der elliptischen Kurve mit der x-Achse.

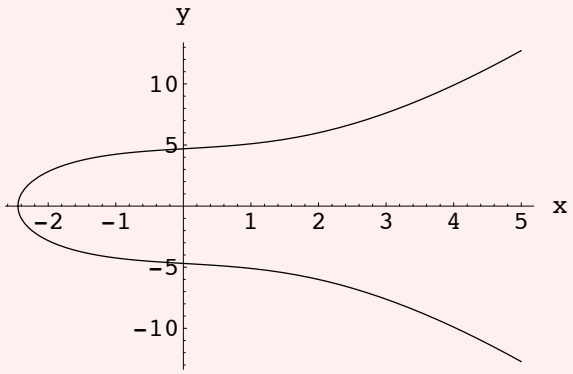
Das besondere an elliptischen Kurven ist nun, dass wir diese mit der Struktur einer kommutativen Gruppe ausstatten können. Hierzu führen wir eine additive Verknüpfung $\oplus$ ein. Wir vereinfachen die Gleichung (6.1) durch die Wahl von $z = 1$ noch einmal zu

$$y^2 = x^3 + a \cdot x + b. \tag{6.2}$$

[5] In der Literatur ist hier der Begriff der Charakteristik angeführt, vgl. die Bemerkung im Anschluss zu Satz A.25, hier: Körper K mit Charakteristik $c(K) \neq 2, 3$

Beispiel 6.13

Die elliptische Kurve zur Gleichung $y^2 = x^3+3x+22$ mit der Diskriminante $\Delta = 122 > 0$ wird durch folgenden Graphen dargestellt:



Satz und Definition 6.14

Betrachten wir die beliebigen Punkte $P_1 = (x_1, y_1)$, $P_2 = (x_2, y_2)$ und $\mathcal{O} := (0, 1, 0)$ mit $i(P_1) = (x_1, y_1, 1),\ i(P_2) = (x_2, y_2, 1) \in E(a, b)$. Dann setzen wir

(1) $\mathcal{O} \oplus P_1 := P_1$ und $P_1 \oplus \mathcal{O} := P_1$ ($\mathcal{O}$ ist das neutrale Element)

(2) $-\mathcal{O} := \mathcal{O}$

(3) $-P_1 := (x_1, -y_1)$

(4) $P_1 = -P_2 \Rightarrow P_1 \oplus P_2 := \mathcal{O}$

(5) Falls für $P_3 = (x_3, y_3) := P_1 \oplus P_2 \in E(a, b)$ die Regeln (1)-(4) nicht greifen, ist $x_3 = \lambda^2 - x_1 - x_2, \quad y_3 = \lambda(x_1 - x_3) - y_1$ mit $\lambda \in \mathbb{R}$ und

$$\lambda = \begin{cases} \frac{y_2 - y_1}{x_2 - x_1} & P_1 \neq P_2 \\ \frac{3x_1^2 + a}{2y_1} & P_1 = P_2. \end{cases}$$

$(E(a, b), \oplus, \mathcal{O})$ ist eine kommutative Gruppe mit neutralem Element $\mathcal{O}$.

Wir betrachten nun elliptische Kurven über $\mathbb{Z}_p$, wobei $p > 3$ eine Primzahl ist. Die eben aufgestellten Regeln zur Bildung einer kommutativen Gruppe $(E(a, b), \oplus, \mathcal{O})$ übertragen sich unter Beachtung der modulo-Rechnung vollständig von $\mathbb{R}$ auf $\mathbb{Z}_p$. Wir untersuchen die elliptische Kurve

$$E_p(a, b) = \{(x', y', 1);\ y'^2 = x'^3 + a \cdot x' + b,\ x', y' \in \mathbb{Z}_p\} \cup \{\mathcal{O}\}.$$

Das p charakterisiert einen endlichen Körper $(Z_p, +_p, \cdot_p)$, über dem die elliptische Kurve definiert wird. Wir erhalten eine additive kommutative Gruppe $(E_p(a, b), \oplus_p, \mathcal{O})$. Die Anzahl der Elemente N dieser Gruppe lässt sich durch das Theorem von Hasse[6] abschätzen über

$$p + 1 - 2\sqrt{p} \ \leq \ N \ \leq \ p + 1 + 2\sqrt{p}.$$

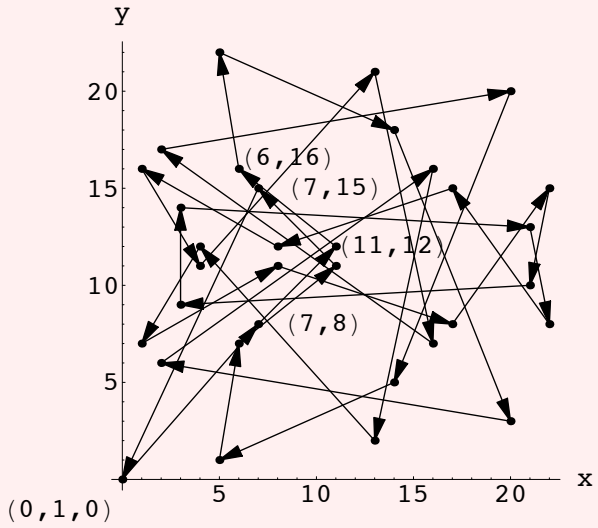
[6] H. Hasse, 1898-1979

Durch Variation von a und b lassen sich alle möglichen Werte für N erzeugen.

Beispiel 6.15

Sei $p = 23$. Je nach Wahl der Polynomkoeffizienten sind zwischen 15 und 33 auf der elliptischen Kurve liegende verschiedene Punkte möglich. Wir wählen $a = 3$ und $b = 22$ und erhalten zur Gleichung $y^2 = x^3 + 3 \cdot x + 22$ mod 23 folgende 33 Punkte:

$\mathcal{O}, (1,7), (1,16), (2,17), (2,6), (3,14), (3,9), (4,11), (4,12), (5,22), (5,1), (6,7),$
$(6,16), (7,15), (7,8), (8,11), (8,12), (11,11), (11,12), (13,21), (13,2), (14,5), (14,18),$
$(16,7), (16,16), (17,15), (17,8), (20,20), (20,3), (21,10), (21,13), (22,15), (22,8).$



Alle Punkte (bis auf $\mathcal{O}$, der Punkt ist in der Abbildung im Ursprung eingezeichnet) treten paarweise in x auf. Das liegt an der zu bestimmenden Quadratwurzel modulo p. Die folgenden vier Additionen führen wir durch:

$$\begin{aligned}
(7,8) \oplus_{23} (7,15) &= \mathcal{O}, \\
(7,8) \oplus_{23} (7,8) &= (11,12),\ \lambda = 18, \\
(11,12) \oplus_{23} (7,8) &= (6,16),\ \lambda = 1, \\
(11,12) \oplus_{23} (7,15) &= (7,8),\ \lambda = 5.
\end{aligned}$$

Um die Punkte einer elliptischen Kurve $E_p(a,b)$ zu bestimmen, können nacheinander alle Werte für $x \in \mathbb{Z}_p$ eingesetzt und durch eine Wurzelbestimmung ein dazugehöriges y gesucht werden. Doch nicht für jedes $u = x^3 + ax + b$ mod p gibt es ein solches y. Für $u = 0$ ist $y = 0$ und $(x,0) \in E_p(a,b)$. Folgende Betrachtung hilft bei der Suche nach Quadratwurzeln modulo p.

Definition 6.16

Ein Element $u \in \mathbb{Z}_p^*$ heißt quadratischer Rest modulo p (Quadratzahl), wenn ein $y \in \mathbb{Z}_p^*$ mit $y^2 \bmod p = u$ existiert.

Nicht jedes Element $u \in \mathbb{Z}_p^*$ ist jedoch eine Quadratzahl modulo p. Sei nun

$$R_p := \{u \in \mathbb{Z}_p^*;\ u \text{ quadratischer Rest modulo } p\}.$$

Das Problem, eine Quadratwurzel $y \in \mathbb{Z}_p^*$ zu einer Quadratzahl u zu bestimmen, untersucht folgender

Satz 6.17: Quadratwurzel modulo p

Sei $p > 2$ eine Primzahl und $u \in \mathbb{Z}_p^*$ eine Quadratzahl modulo p mit $u^{\frac{p-1}{2}} \bmod p = 1$. Dann liefert Algorithmus 6.18 ein $y \in \mathbb{Z}_p^*$ mit $y^2 \bmod p = u$.

Beweis.

Für $u \in R_p$ gilt $y^2 \bmod p = u$ und $(p-y)^2 \bmod p = (p^2 - 2py + y^2) \bmod p = y^2 \bmod p = u$. Damit ist $y' := p - y$ eine weitere und wegen des Widerspruchs $y' = y \Leftrightarrow 2y = p$ von y verschiedene Quadratwurzel zu u. Es gibt dann zwei Quadratwurzeln.

Wir wissen bereits seit den Betrachtungen zu Definition 5.4, dass für $u \in \mathbb{Z}_p^*$ gilt $u^{\frac{p-1}{2}} \bmod p = \pm 1$. Damit ist für eine Quadratzahl $u \in R_p$ nach dem kleinen Satz von Fermat $u^{\frac{p-1}{2}} \bmod p = (y^2)^{\frac{p-1}{2}} \bmod p = 1$. Das Polynom $u^{\frac{p-1}{2}} - 1$ hat höchstens $\frac{p-1}{2}$ Nullstellen. Andererseits gibt es $\frac{p-1}{2}$ Quadratzahlen. Denn betrachten wir y^2 mit $y \in \{1, \ldots, \frac{p-1}{2}\}$, so hat jedes y^2 die verschiedenen Quadratwurzeln y und $p - y$. Für jedes $u \in R_p$ ist entweder y oder $p - y$ Element der Menge $\{1, \ldots, \frac{p-1}{2}\}$, das sind aber genau $\frac{p-1}{2}$ verschiedene Elemente und es gilt $1^2 \bmod p \neq 2^2 \bmod p \neq \ldots \neq \left(\frac{p-1}{2}\right)^2 \bmod p$. Es gibt also genau $\frac{p-1}{2}$ Nullstellen von $u^{\frac{p-1}{2}} - 1$ nämlich die Quadratzahlen. Für $u' \in \mathbb{Z}_p^*$ mit $u' \notin R_p$ gilt damit $u'^{\frac{p-1}{2}} \neq 1$. Es muss damit $u'^{\frac{p-1}{2}} = p - 1$ sein.

Wählen wir im Algorithmus 6.18 ein $v \in \mathbb{Z}_p^*$ aus, so ist v mit Wahrscheinlichkeit $P = \frac{1}{2}$ keine Quadratzahl. Sei nun $p - 1 = q \cdot 2^l$ und $2^{l+1} \not| p - 1$ für $l, q \in \mathbb{N}_0$. Dann gibt es für $u_1 := u$ ein kleinstes $k_1 \in \mathbb{N}_0$ mit $u_1^{q \cdot 2^{k_1}} \bmod p = 1$ und $k_1 \leq l$. Wir betrachten nun

die Folge $u_{n+1} = u_n \cdot v^{2^{l-k_n}} \bmod p$.

$$\begin{aligned} u_{n+1}^{q\cdot 2^{k_n-1}} \bmod p &= \left(u_n \cdot v^{2^{l-k_n}}\right)^{q\cdot 2^{k_n-1}} \bmod p \\ &= u_n^{q\cdot 2^{k_n-1}} \cdot v^{q\cdot 2^{l-1}} \bmod p \\ &= u_n^{q\cdot 2^{k_n-1}} \cdot v^{\frac{p-1}{2}} \bmod p \\ &= u_n^{\frac{q\cdot 2^{k_n}}{2}} \cdot (-1) \bmod p \\ &= (-1)\cdot(-1) \bmod p \\ &= 1. \end{aligned}$$

Damit lässt sich stets ein $k_{n+1} \leq k_n - 1 < k_n$ basierend auf der Wahl der Nicht-Quadratzahl v bestimmen, so dass insgesamt $l \geq k_1 > k_2 > \ldots > k_n > k_{n+1} > \ldots \geq 0$. Es existiert ein $\mathbb{N} \ni n' \leq l+1$ mit $k_{n'} = 0$ und $1 = u_{n'}^{q\cdot 2^{k_{n'}}} \bmod p = u_{n'}^{q} \bmod p$. Multiplikation der Gleichung mit $u_{n'}$ liefert

$$u_{n'} \cdot u_{n'}^{q} \bmod p = u_{n'}^{q+1} \bmod p = u_{n'} \text{ und } \left(u_{n'}^{\frac{q+1}{2}}\right)^2 \bmod p = u_{n'}.$$

$r_{n'} := u_{n'}^{\frac{q+1}{2}} \bmod p$ ist Quadratwurzel zur Quadratzahl $u_{n'}$. Setzen wir

$$r_i := r_{i+1}\left(v^{2^{l-k_i-1}}\right)^{-1} \bmod p$$

für $i = n'-1, \ldots, 1$, so erhalten wir den Zusammenhang $r_i^2 \bmod p = u_i$, was wir induktiv über

$$\begin{aligned} r_i^2 \bmod p &= r_{i+1}^2 \cdot \left(\left(v^{2^{l-k_i-1}}\right)^{-1}\right)^2 \bmod p \\ &= u_{i+1} \cdot \left(\left(v^{2^{l-k_i-1}}\right)^{2}\right)^{-1} \bmod p \\ &= u_{i+1} \cdot \left(v^{2^{l-k_i}}\right)^{-1} \bmod p \\ &= u_i \cdot v^{2^{l-k_i}} \cdot \left(v^{2^{l-k_i}}\right)^{-1} \bmod p \\ &= u_i. \end{aligned}$$

beweisen. Damit haben wir insbesondere auch $r_1^2 \bmod p = u_1 = u$ und $x := r_1$ ist eine der beiden gesuchten Quadratwurzeln.

□

Bemerkung.
Zu y als Ergebnis des Algorithmus ist auch $p - y$ eine Quadratwurzel.

Algorithmus 6.18: Quadratwurzel modulo p

Eingabe: $p > 2$ Primzahl, $u \in \mathbb{Z}_p^*$ Quadratzahl.

Ausgabe: $y \in \mathbb{Z}_p^*$ mit $y^2 \bmod p = u$.

1: Wähle (zufällig) $v \in \mathbb{Z}_p^*$ so, dass $v^{\frac{p-1}{2}} \bmod p = p-1$.
2: Bestimme $l, q \in \mathbb{N}_0$ mit $p-1 = q \cdot 2^l$ und $2^{l+1} \not| p-1$.
3: $u_1 := u,\ n := 1$.
4: Suche das kleinste $k_1 \geq 0$ mit $u_1^{q \cdot 2^{k_1}} \bmod p = 1$.
5: **while** $k_n \neq 0$ **do**
6: $\quad u_{n+1} := u_n \cdot v^{2^{l-k_n}} \bmod p$.
7: $\quad$ Suche das kleinste $k_{n+1} \geq 0$ mit $u_{n+1}^{q \cdot 2^{k_{n+1}}} \bmod p = 1$.
8: $\quad n := n+1$.
9: **end while**
10: $r_n := u_n^{\frac{m+1}{2}} \bmod p$.
11: **for** $i := n-1,\ i > 0$ **do**
12: $\quad r_i := r_{i+1} \left(v^{2^{l-1-k_i}}\right)^{-1} \bmod p$.
13: $\quad i := i-1$.
14: **end for**
15: $y := r_1$.
16: **return** y.

Beispiel 6.19

Für $p = 2906161$ liegen $N \in [2902752, 2909571]$ Punkte je nach Wahl der Polynomkoeffizienten auf der elliptischen Kurve. Wir betrachten die elliptische Kurve $E_{2906161}(3, 22)$ mit der Gleichung $y^2 = x^3 + 3 \cdot x + 22$ und $x = 40000$. Sei $u = 40000^3 + 3 \cdot 40000 + 22 \bmod 2906161 = 2375203$. Dann ist $u^{1453080} \bmod 2906161 = 1$, u ist somit eine Quadratzahl und mit $v = 696729$ finden wir $y = 1610103$ und $p - y = 1296058$ als Quadratwurzeln und damit haben wir mit $(40000, 1610103)$ und $(40000, 1296058)$ zwei Punkte der elliptischen Kurve.

6.3.1. El Gamal und elliptische Kurven

Wir wählen eine Gruppe $(E_p(a,b), \oplus_p, \mathcal{O})$, um durch Funktionen $e_{(E,g,y)}$ bzw. $d_{(E,x)}$ die El Gamal-Verschlüsselung umzusetzen. In der additiven Schreibweise ist die Exponentiation dabei als Multiplikation mit einer ganzen Zahl zu sehen. Diese Funktion ist genau dann bijektiv, wenn g Erzeuger ist. Die betrachteten Gruppen müssen aber nicht unbedingt zyklisch sein.

Beispiel 6.20

Das Element $(1, 7)$ aus Beispiel 6.15 erzeugt lediglich die Menge $\{(1,7), (1,16), \mathcal{O}\}$ der Punkte von $E_{23}(3, 22)$.

Deshalb sucht man einen Punkt $g \in E_p(a,b)$, der eine möglichst große Untergruppe erzeugt. In der entsprechenden Untergruppe lautet das Problem des diskreten Logarithmus dann, einen Punkt h zu finden mit $x \odot_p h = g$ und $x \in \mathbb{Z}$. Die Multiplikation $\odot_p$ mit einer ganzen Zahl ist dabei über die Addition wie in $(\mathbb{Z}_p, +_p, 0)$ bestimmt. Konzipieren wir eine

elliptische Kurve, deren Ordnung eine Primzahl ist, so haben wir eine zyklische Gruppe und jedes Element (außer $\mathcal{O}$) ist Erzeuger.

Bei einer El Gamal-Verschlüsselung kommen wir im Gegensatz zum RSA-Verfahren mit etwa 163-Bit Zahlen für p aus. Bereits dann ist der diskrete Logarithmus sehr komplex zu berechnen. Dies rechtfertigt den hohen Aufwand zur Berechnung von $E_p(a,b)$ für geeignet gewählte a, b und die etwas andere Arithmetik. Es ist wichtig festzuhalten, dass die Gruppe $(E_p(a,b), \oplus_p, \mathcal{O})$ a priori gewählt werden kann. Der Aufwand dafür ist zwar im allgemeinen sehr hoch, ist aber vor der Implementierung eines Kryptosystems zu leisten. Beide Kommunikationsteilnehmer müssen die Gruppe kennen.

Wir übertragen zunächst das El Gamal-Verfahren analog auf elliptische Kurven. Nach Wahl einer Gruppe $(E_p(a,b), \oplus_p, \mathcal{O})$ bestimmen wir ein $g \in E_p(a,b)$, wobei $|\langle g\rangle|$ möglichst groß bzw. im Idealfall $|\langle g\rangle| = |E_p(a,b)|$ ist. Wir bestimmen zufällig ein $x_A \in \mathbb{Z}_{|\langle g\rangle|}$ und errechnen den letzten Bestandteil des öffentlichen Schlüssels $y_A = x_A \odot_p g$. Der öffentliche Schlüssel ist somit $(E_p(a,b), g, y_A)$, der private ist x_A.

Die zu übermittelnde Nachricht M stellen wir als Werte in $E_p(a,b)$ dar. Gegebenenfalls muss die Nachricht in Blöcke aufgeteilt werden. Sei nun $M \in E_p(a,b)$. Wir bestimmen ein zufälliges $x_B \in \mathbb{Z}_{|\langle g\rangle|}$ und erstellen die verschlüsselte Nachricht $e_{(E_p(a,b),g,y_A)} = (y_B, c)$ aus $y_B = x_B \odot_p g$ bzw. $c = M \oplus_p x_B \odot_p y_A$.

Zum Entschlüsseln berechnen wir $z = -(x_A \odot_p y_B) = -x_{AB}$ aus dem privaten Schlüssel und erhalten die ursprüngliche Nachricht M. Denn es ist

$$\begin{aligned} d_{(E_p(a,b),x_A)} = c \oplus_p z &= M \oplus_p (x_B \odot_p y_A) \oplus_p (-(x_A \odot_p y_B)) \\ &= M \oplus_p (x_B \odot_p y_A) \oplus_p (-(x_A \odot_p (x_B \odot_p g))) \\ &= M \oplus_p (x_B \odot_p y_A) \oplus_p (-(x_B \odot_p x_A \odot_p g)) \\ &= M \oplus_p (x_B \odot_p y_A) \oplus_p (-(x_B \odot_p y_A)) \\ &= M. \end{aligned}$$

Damit haben wir eine Nachricht erzeugt, die aus vier Komponenten aus $\mathbb{Z}_p$ besteht, also eine Vervierfachung der ursprünglichen Nachricht. Das klassische El Gamal-Verfahren verdoppelt die Länge der Nachricht. Deshalb haben Menezes[7]/Vanstone[8] das Verfahren derart modifiziert, dass die elliptischen Kurven das eigentliche Geschehen maskieren. Ein weiteres Problem besteht darin, dass Klartexten Punkte auf der elliptischen Kurve zugeordnet werden müssen. Somit müssen für einzelne Klartextblöcke, die in Zahlenwerte umgerechnet werden, Punkte der elliptischen Kurve zur Verfügung stehen.

[7] A. Menezes,
[8] S. Vanstone,

Beispiel 6.21

Wir verschlüsseln den Begriff „KRYPTOGRAPHIE_“ in Zweierblöcken. Wir wählen $p = 18743 > 2^{14}$ und die elliptische Kurve $E_{18743}(3, 22)$.

Zeichenblock	Binärblock	Dezimalzahl x	$(x, \cdot) \in E_p(a, b)$?	$(\tilde{x}, y)$
KR	10010111010010	9682	nein	$(9683, 2287)$
YP	10110011010000	11472	nein	$(11475, 13216)$
TO	10101001001111	10831	ja	$(10831, 16908)$
GR	10001111010010	9170	nein	$(9171, 5565)$
AP	10000011010000	8400	nein	$(8401, 14574)$
HI	10010001001001	9289	nein	$(9291, 13625)$
$E\_$	10001011011111	8927	nein	$(8929, 13289)$

Nur ein x-Wert entspricht der ersten Koordinate eines Punktes der elliptischen Kurve. Die anderen Punkte müssen erst gefunden werden. Wir wählen der Einfachheit halber den ersten nachfolgenden Wert $\tilde{x}$, der das jeweils erfüllt. Seien $g = (12, 338)$ und $x_A = 20$. Dann ist mit $y_A = 20 \odot_{18743} (12, 338) = (8308, 8072)$ der öffentliche Schlüssel $E_{18743}((3, 22), (8308, 8072))$ und der private ist 20.

Sei $x_{B_1} = 1000$. Dann erhalten wir zum ersten Klartextblock den Geheimtext (y_{B_1}, d_1) über $y_{B_1} = 1000 \odot_{18743} (12, 338) = (7977, 18734)$ und $c_1 = (9683, 2287) \oplus_{18743} 1000 \odot_{18743} (8308, 8072) = (1457, 15623)$. Für $x_{B_2}, \ldots, x_{B_7} = 1001, \ldots, 1006$ zu den Klartextblöcken 2 bis 7 ergeben sich jeweils wiederum Geheimtexte. Insgesamt ergeben sich die 7 Geheimtexte

$$\begin{array}{ll} (y_{B_1}, c_1) & ((7977, 18734), (1457, 15623)) \\ (y_{B_2}, c_2) & ((8584, 16630), (5138, 16863)) \\ (y_{B_3}, c_3) & ((7616, 10285), (18067, 3062)) \\ (y_{B_4}, c_4) & ((18059, 5729), (12856, 4936)) \\ (y_{B_5}, c_5) & ((14199, 15108), (8886, 5822)) \\ (y_{B_6}, c_6) & ((13842, 4296), (17183, 17998)) \\ (y_{B_7}, c_7) & ((9868, 12245), (15256, 17721)). \end{array}$$

Zur Entschlüsselung von (y_{B_1}, c_1) brauchen wir $z_1 = -((7977, 18734) \odot_{18743} 20) = -(16553, 18311) = (16553, 432)$. Es ist $(1457, 15623) \oplus_{18743} (16553, 432) = (9683, 2287)$. Analog können die restlichen Geheimtexte mit $z_2 = (6742, 6907)$, $z_3 = (10317, 11741)$, $z_4 = (10250, 7896)$, $z_5 = (16606, 8560)$, $z_6 = (7569, 7561)$ und $z_7 = (9012, 10334)$ entschlüsselt werden.

Trotz der korrekten Entschlüsselung fällt es schwer, die gesuchten Zahlenwerte für den Klartext zu bestimmen. Denn es ist keine Information darüber vorhanden, dass beispielsweise wie oben statt 9683 die Zahl 9682 gesucht ist. Unter anderem diese Problematik führte wie bereits erwähnt zu einer adaptierten Anwendung des El-Gamal-Verfahrens. Die Schlüsselerzeugung erfolgt dabei in gleicher Weise, d.h. mit Hilfe einer gewählten elliptischen Kurve $E_p(a, b)$ werden der öffentliche Schlüssel $(E_p(a, b), g, y_A)$ und der private $x_A \in \mathbb{Z}_{|<g>|}$ bestimmt.

Zur Verschlüsselung wird zunächst die Nachricht $M = (m_1, m_2) \in \mathbb{Z}_p^* \times \mathbb{Z}_p^*$ dargestellt. Es wird ein $0 \neq x_B \in \mathbb{Z}_p$ zufällig gewählt und $(k_1, k_2) = x_B \odot_p y_A$ berechnet. Ist k_1 oder

k_2 nicht invertierbar in $\mathbb{Z}_p^*$, muss ein anderes x_B bestimmt werden. Die zu übermittelnde Nachricht (y_B, c_1, c_2) wird durch $y_B = x_B \odot_p g$, $c_1 = k_1 \cdot_p m_1$ und $c_2 = k_2 \cdot_p m_2$ ermittelt.

Zur Entschlüsselung wird der private Schlüssel verwendet. Es müssen $x_A \odot_p y_B = x_A \odot_p (x_B \odot_p g) = x_B \odot_p (x_A \odot_p g) = x_B \odot_p y_A = (k_1, k_2)$ und danach k_1^{-1} und k_2^{-1} modulo p ausgerechnet werden. Damit kann dann $M = (m_1, m_2) = (c_1 \cdot_p k_1^{-1}, c_2 \cdot_p k_2^{-1})$ ermittelt werden.

Beispiel 6.22

Wir wählen $E_{18743}(3, 22)$, $g = (12, 338)$ und $x_A = 20$ wie im letzten Beispiel und erhalten $y_A = 20 \odot_{18743} (12, 338) = (8308, 8072)$.

Damit können wir die Nachricht $(9682, 11472)$ verschlüsseln. Sei $x_B = 1000$: $(k_1, k_2) = 1000 \odot_{18743} (8308, 8072) = (16553, 18311)$. Die verschlüsselte Nachricht (y_B, c_1, c_2) ergibt sich durch $y_B = 1000 \odot_{18743} (12, 338) = (7977, 18734)$, $c_1 = 16553 \cdot_{18743} 9682 = 13496$ und $c_2 = 18311 \cdot_{18743} 11472 = 10991$, also $((7977, 18734), 13496, 10991)$.

Zur Entschlüsselung: $1000 \odot_{18743} (7977, 18734) = (16553, 18311) = (k_1, k_2)$. Damit bestimmen wir $k_1^{-1} = 659$ und $k_2^{-1} = 5163$. Als entschlüsselte Nachricht bekommen wir somit $(13496 \cdot_{18743} 659, 10991 \cdot_{18743} 5163) = (9682, 11472)$.

Im letzten Beispiel können wir erkennen, dass bei dieser angepassten El-Gamal-Variante der Geheimtext wiederum lediglich doppelt so lange wie der Klartext wird. Aus den beiden den Klartext repräsentierenden Zahlen wurden vier Zahlen Geheimtext. Zudem ist es nicht notwendig, sich eine Strategie zur Abbildung der kodierten Klartextblöcke auf Punkte der elliptischen Kurve zu überlegen, was die Angelegenheit entsprechend vereinfacht.

Algorithmus 6.23: Angepasster El-Gamal für elliptische Kurven

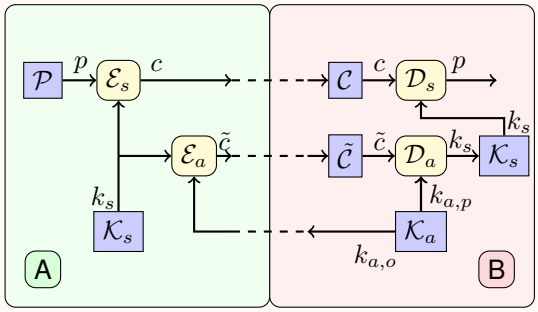
Eingabe: Öffentllicher Schlüssel $(E_p(a, b), g, y_A)$ und privater Schlüssel $x_A \in \mathbb{Z}_{|\langle g \rangle|}$.
Nachricht $M = (m_1, m_2) \in \mathbb{Z}_p^* \times \mathbb{Z}_p^*$.
Ausgabe: Geheimtext $\tilde{c}$, entschlüsselter Klartext M'.

1: Bestimme $0 \neq x_B \in \mathbb{Z}_p$.
2: $(k_1, k_2) := x_B \odot_p y_A$.
3: **if** $(k_1 \notin \mathbb{Z}_p^* || k_2 \notin \mathbb{Z}_p^*)$ **then**
4: GOTO 1.
5: **end if**
6: $y_B := x_B \odot_p g$.
7: $c_1 := k_1 \cdot_p m_1$ und $c_2 := k_2 \cdot_p m_2$.
8: $\tilde{c} := (y_B, c_1, c_2)$.
9: $(c_1', c_2') := x_A \odot_p y_B$.
10: $M' = (m_1', m_2') := (c_1 \cdot_p k_1^{-1}, c_2 \cdot_p k_2^{-1})$.
11: **return** $\tilde{c}, M'$.

6.4. Hybride Chiffriersysteme

Unter einem hybriden Chiffriersystem wollen wir ein Chiffriersystem verstehen, in dem sowohl symmetrische als auch asymmetrische Verfahren zur Anwendung kommen. Da die asymmetrische Verschlüsselung deutlich aufwändiger als die symmetrische ist, wird oftmals nur ein relativ schnell zu verschlüsselnder Bestandteil der Kommunikation zweier Instanzen asymmetrisch verschlüsselt. Da bei der symmetrischen Verschlüsselung das Austauschen des symmetrischen Schlüssel eine wichtige Aufgabe darstellt, kann dieser mit dem öffentlichen Schlüssel asymmetrisch verschlüsselt und übertragen werden. Die eigentliche Nachricht wird mit einem symmetrischen Verfahren verschlüsselt und übertragen. Die Instanz mit dem privaten Schlüssel gelangt so zunächst in den Besitz des symmetrischen Schlüssels und kann dann die eigentliche Nachricht entschlüsseln. Das Schema ist in Abbildung 6.2 zu sehen.

Abbildung 6.2.: Hybride Chiffrierung



7. Primzahltest und Pseudozufallszahlen

Viele kryptographische Verfahren benötigen an verschiedenen Stellen als Eingabe eine Zahl oder eine Folge von Zahlen, die nicht vorab festgelegt sind. Unter einer Pseudozufallszahlenfolge verstehen wir eine Folge $(z_n)_{n\in\mathbb{N}}$ von Zahlen, die durch ein deterministisches Verfahren erzeugt wird und den Anschein erweckt, zufällig zu sein. Durch den Initialwert - genannt Seed - ist dabei die ganze Folge vorgegeben.

7.1. Linearer Kongruenzgenerator

Beispiel 7.1: LCG

Sei $m \in \mathbb{N}$ gegeben. Ein linearer Kongruenzgenerator (engl. Abkürzung: LCG) ist eine Folge $(x_i)_{i\in\mathbb{N}}$, $x_i \in \mathbb{Z}_m$, von nichtnegativen ganzen Zahlen, die mittels der iterativen Vorschrift

$$x_{i+1} = a \cdot x_i + b \bmod m$$

für fest vorgegebene Werte a, b und dem Seed x_0 mit $a, b, x_0 \in Z_m$ erzeugt werden.

Nehmen wir an, es lägen $l+1$ aufeinanderfolgende Zufallszahlen $x_j, x_{j+1}, \ldots, x_{j+l}$ eines LCG vor, dessen Parameter a, b, m und Seed x_0 unbekannt seien. Wir definieren eine neue Folge $(y_i)_{i\in\mathbb{N}}$, $y_i := x_{j+i} - x_{j+i-1}$. Es gilt

$$\begin{aligned} y_{i+1} \bmod m &= x_{j+i+1} - x_{j+i} \bmod m = ax_{j+i} + b \bmod m - (ax_{j+i-1} + b \bmod m) \\ &= a(x_{j+i} - x_{j+i-1}) \bmod m \\ &= ay_i \bmod m. \end{aligned}$$

Ist $l \geq 4$, so können wir einen ersten Schätzwert für m bestimmen. Dazu suchen wir ganze Zahlen γ_2, γ_3 mit

$$\gamma_2 y_1 = \gamma_3 y_2,$$

etwa $\gamma_2 := y_2$ und $\gamma_3 := y_1$. Dann folgt

$$\begin{aligned} \gamma_2 \cdot ay_1 \bmod m = \gamma_2 y_2 \bmod m &= \gamma_3 y_3 \bmod m = \gamma_3 \cdot ay_2 \bmod m \\ \Rightarrow y_2^2 - y_1 y_3 &= 0 \bmod m. \end{aligned}$$

Damit ist $\tilde{m} := |y_2^2 - y_1 y_3|$ ein ganzzahliges Vielfaches von m. Ist $\tilde{m} \neq 0$, so fahren wir fort, indem wir Schätzungen für a und b bestimmen. Dazu lösen wir die modulare Gleichung $y_2 = ay_1 \bmod \tilde{m}$ für a und finden anschließend $b = x_{j+1} - ax_j \bmod \tilde{m}$. Nun können wir Zufallszahlen überprüfen bzw. die bereits bekannten überprüfen. Sollten sich Fehler ergeben muss weiter gesucht werden.

Beispiel 7.2

Seien $a = 13$, $b = 12$ und $m = 79$. Mit $x_0 := 10$ seien die weiteren Zufallszahlen $x_1 = 63$, $x_2 = 41$ und $x_3 = 71$ bekannt. Wir haben zunächst $y_1 = 53$, $y_2 = -22$ und $y_3 = 30$. Damit ergibt sich $\tilde{m} = |y_2^2 - y_1 y_3| = |(-22)^2 - 53 \cdot 30| = 1106$. Daraus lassen sich $a = 250$ und $b = 881$ errechnen. Eine Überprüfung und Vorhersage $x_{i+1} = 250 \cdot x_i + 881$ mod 1106 mit $x_0 = 10$ liefert das Tupel $(10, 63, 41, 71, 935)$. Jedoch ist mit bekanntwerden von $x_4 = 66$ der Vorhersagewert nicht richtig.

Wir nutzen eine neue Information und gehen beinahe wie bei der Bestimmung des Initialwertes $\tilde{m}$ vor. Wir nutzen jedoch den Wert $\tilde{m}$ und müssen nunmehr ein modulares Gleichungssystem folgender Art lösen.

$$\gamma_2 y_1 + \gamma_3 y_2 = \gamma_4 y_3,$$

um analog zu oben einen Schätzwert zu bekommen. Wir nutzen aus, dass es viele Lösungen einer solchen Gleichung gibt und m ein Teiler von $\tilde{m}$ und $\gamma_4 y_4 - \gamma_3 y_3 - \gamma_2 y_2$ ist:

$$\tilde{m} = \mathsf{ggT}(|\gamma_4 y_4 - \gamma_3 y_3 - \gamma_2 y_2|, \tilde{m}).$$

7.2. Überprüfung von Primzahlkandidaten

Sind wir in der Lage genügend große Primzahlen zu bestimmen, so haben wir für die Kryptographie gewonnen. Das führt auf zwei Fragestellungen: Gibt es effiziente Primzahltests für große Zahlen und wie lassen sich schnell und zufällig große Zahlen generieren? Es gibt eine Reihe von Primzahltests. An dieser Stelle werden wir den probabilistischen Miller-Rabin-Test[1] betrachten.

Satz 7.3: Miller-Rabin-Primzahltest für ungerade Zahlen $n > 1$

Nachfolgender Algorithmus 7.6 liefert eine sichere Aussage über die Zusammensetzung einer Zahl $n > 1$. Bei k-facher Wiederholung des Tests mit k zufällig gewählten Basen ergibt sich eine Wahrscheinlichkeit von $P < \frac{1}{4^k}$, dass eine Zahl fälschlich als Primzahl akzeptiert wird.

□

Bemerkung.
Wie in [11] wird oft empfohlen, vor den Miller-Rabin-Primzahltest ein Divisionssieb (Sieb des Eratosthenes[2]) vorzuschalten und die zu testende Zahl durch die Primzahlen bis zu einer bestimmten Grenze zu dividieren. Damit wird schon eine Vielzahl zusammengesetzter Zahlen ausgeschlossen.

Der Primzahltest basiert auf einer verschärften Form des kleinen Satzes von Fermat. Sei $n \in \mathbb{N}$ eine ungerade natürliche Zahl. Wir legen durch

$$t := \max\{r \in \mathbb{N};\ 2^r \mid n-1\}, \quad t \geq 1$$

[1] G. L. Miller, M. O. Rabin
[2] $Ερατοσθένης$, um 284-202 v.Chr.

die Potenz von 2 fest, die $n-1$ gerade noch teilt. Weiter sei die ungerade Zahl $q := (n-1)/2^t,\ q \geq 1$ festgelegt. Wir formulieren den

Satz 7.4

Seien $n \in \mathbb{N}$ eine Primzahl und $a \in \mathbb{Z}$ mit ggT$(a,n) = 1$. Seien weiter t und q wie eben definiert, dann gilt entweder

$$a^q \equiv_n 1$$

oder

$$\exists r \in \{0,1,\ldots,t-1\} \text{ mit } a^{2^r \cdot q} \equiv_n n-1.$$

Beweis.

Sei $a \in \mathbb{Z}$ und ggT$(a,n) = 1$; $|\mathbb{Z}_n^*| = n-1 = 2^t \cdot q$, da n Primzahl ist. Die Ordnung $e := \text{ord}_{\mathbb{Z}_n^*}(a)$ teilt $n-1 = 2^t \cdot q$, d.h.

$$2^t \cdot q = e \cdot z, \quad 0 < z < n-1 \text{ und } \text{ord}_{\mathbb{Z}_n^*}(a^q) = \frac{e}{\text{ggT}(e,q)}.$$

$\Rightarrow\ e = 2^t \cdot q \cdot z^{-1}$, $\text{ord}_{\mathbb{Z}_n^*}(a^q) = \frac{2^t \cdot q \cdot z^{-1}}{\text{ggT}(2^t \cdot q \cdot z^{-1}, q)} = 2^t \cdot z^{-1} \in \{1,\ldots,n-1\}$. Mit $2^t \cdot q = (z \cdot q)2^t z^{-1} = z \cdot (2^t \cdot z^{-1}) \cdot q$ muss z^{-1} eine Potenz von 2 sein. $\Rightarrow \text{ord}_{\mathbb{Z}_n^*}(a^q) = 2^l =: k$ für ein $l \in \{0,\ldots,t\}$.

$$\begin{array}{lll} l = 0: & k = 1 \Rightarrow & a^q \equiv_n 1 \\ 1 \leq l \leq t: & k > 1 \Rightarrow & \text{ord}_{\mathbb{Z}_n^*}(a^{2^{l-1} \cdot q}) = \frac{2^l}{\text{ggT}(2^l, 2^{l-1})} = 2 \end{array}$$

Es bleibt die Frage, welche Elemente $b \in \mathbb{Z}_n^*$ die Ordnung 2 haben. Es muss b^2 mod $n = 1$ bzw. $n \mid b^2 - 1 = (b-1) \cdot (b+1)$ gelten. Damit teilt n in Analogie zum Beweis zu Satz 4.1 entweder $(b-1)$ oder $(b+1)$. Da n prim ist, muss $b = n-1$ sein und $n \mid b+1 = n$. Da es nur dies eine Element $b \in \mathbb{Z}_n^*$ gibt, nämlich $n-1$, folgt für $r = l-1$: $a^{2^r \cdot q} \equiv_n n-1$, $0 \leq r < t$.

□

Eine der beiden Bedingungen für eine Einheit $a \in \mathbb{Z}_n^*$ ist notwendig, damit n eine Primzahl ist. Somit ist a bei Nicht-Erfüllen beider Bedingungen ein Zeuge gegen die Primalität von n.

Beispiel 7.5

Sei $n = 21,\ n-1 = 20 = 2^2 \cdot 5$ und damit $t = 2$ und $q = 5$. Für $a = 2$ gilt: 2^5 mod $21 = 11$ für $r = 0$ und $2^{5 \cdot 2}$ mod $21 = 16$ für $r = 1$. Damit ist 2 ein Zeuge gegen die Primalität von 21.
Sei $n = 11,\ n-1 = 10 = 2 \cdot 5$ und damit $t = 1$ und $q = 5$. Für $a = 2$ gilt: 2^5 mod $11 = 10$ und damit bleibt 11 ein Kandidat für eine Primzahl. Ebenso gilt beispielsweise für $a = 3$ und $a = 7$: 3^5 mod $11 = 1$ und 7^5 mod $11 = 10$. Die Wahrscheinlichkeit, dass

11 fälschlich als Primzahl akzeptiert wird, ist $P < 0.016$.

Zur Primzahlgenerierung aber auch zur Schlüsselerzeugung werden möglichst „zufallbasierte“ Verfahren benötigt. Die One-Time-Pad-Verschlüsselung[3] benötigt einen Binärschlüssel fester Bitlänge, wobei die einzelnen Bits gleichverteilt sein sollen und zusammengesetzt eine große Zahl ergeben. Die Generierung von großen zufälligen Zahlen kann über verschiedene Algorithmen geschehen. Das Wort Algorithmus deutet jedoch schon an, dass es sich hierbei um „Pseudozufallszahlen“ handelt und nicht um echte aus Zufallsexperimenten generierte Zufallszahlen.

Algorithmus 7.6: Miller-Rabin-Primzahltest

Eingabe: Primzahlkandidat n.
Ausgabe: TRUE oder FALSE für die Frage, ob n Primzahl ist.

Bestimme q und t mit $n-1 = 2^t \cdot q$, q ungerade: $t := 0,\ q := 0,\ i := n-1$.
while i mod $2 \neq 1$ **do**
  $i := i/2$.
  $t := t+1$.
end while
$q := (n-1)/2^t$
Wähle zufällig eine ganze Zahl a mit $1 < a < n$.
if ggT$(a, n) \neq 1$ **then**
  return FALSE (n ist zusammengesetzt).
end if
$e := 0,\ b := a^q$ mod n.
if $b = 1$ **then**
  return TRUE (n ist wahrscheinlich prim).
end if
while $b \neq \pm 1$ mod n && $e < t-1$ **do**
  $b := b^2$ mod n, $e := e+1$.
end while
if $b \neq n-1$ mod n **then**
  return FALSE (n ist zusammengesetzt).
end if
return TRUE (n ist wahrscheinlich prim).

7.3. Erzeugung von Pseudozufallszahlen

Als Anwendung der Eulerschen ϕ-Funktion können wir Pseudozufallszahlen generieren. Dazu nutzen wir den BBS-Generator[4] in Algorithmus 7.7, der auch unter dem Namen „x^2-mod-n-Generator“ bekannt ist. Denn um die Parameterwerte zu bekommen, die für die Generierung einer solchen Pseudozufallszahl notwendig sind, ist wie in Beispiel 4.6 die Eulersche Funktion zu bestimmen. Andererseits kann jedoch, wie wir gleich sehen werden,

[3] Für jede Nachricht der Länge n Bit wird ein neuer n Bit langer Schlüssel erzeugt. Für große n gibt es kaum Chancen für Angreifer.
[4] L. Blum, M. Blum, M. Shub, vgl. http://www.staff.uni-mainz.de/pommeren/Kryptologie/Bitstrom/4_perfekt/

bei einer ungünstigen Wahl von a bzw. k im Algorithmus 7.7 in den Pseudozufallszahlen ein Muster entstehen. Um sicher eine dezimale Pseudozufallszahl aus k Bits zu bekommen, setzen wir das k-te Bit auf 1.

Algorithmus 7.7: BBS-Bitgenerator

Eingabe: Primzahlen $p \neq q$ mit $p, q \equiv_4 3$ und $p-1, q-1, p+1$ und $q+1$ haben wenigstens einen großen Primfaktor. $k \in \mathbb{N}$.
Ausgabe: Pseudozufallszahl durch Bitfolge $x_1 \dots x_k,\ x_i \in \{0,1\}$.

1: $n := p \cdot q$
2: Bestimme $a \in \mathbb{N}$ mit ggT$(a, n) = 1$.
3: $a_0 := a^2$ mod n
4: $i := 0$.
5: **while** $i < k$ **do**
6: $\quad a_{i+1} := a_i^2$ mod n.
7: $\quad$ Entnehme jedem Wert a_{i+1} das niedrigstwertige Bit durch mod-2-Rechnung: $x_{i+1} := a_{i+1}$ mod 2.
8: $\quad i := i + 1$.
9: **end while**
10: **return** $x_1 \dots x_k$.

Wir benötigen eine so genannte Blum-Zahl $n = p \cdot q$ mit p mod $4 = 3 = q$ mod 4 und $p \neq q$ prim. Sei $\pi_{a,b}(x)$ die Anzahl der Primzahlen kleiner gleich x der Form $p = a \cdot k + b$. Unter der Bedingung, dass ggT$(a, b) = 1$ ist, gilt

$$\pi_{a,b}(x) \sim \frac{1}{\phi(a)} \cdot \frac{x}{\log x} \text{ mit } \phi(1) := 1.$$

Mit $a = 4$ und $b = 3$ ist $\pi_{4,3}(x) \sim \frac{1}{\phi(4)} \cdot \frac{x}{\log x} = \frac{1}{2} \cdot \frac{x}{\log x}$, also ist wegen $\pi_{1,0}(x) \sim \frac{x}{\log x}$ die Hälfte aller Primzahlen dadurch angesprochen. Ein kryptoanalytischer Angriff durch Überprüfung aller solcher Primzahlen ist damit sinnlos.

Sei p eine Primzahl. Haben $(p-1)$ und $(p+1)$ wenigstens einen großen Primfaktor, so wird p als starke Primzahl bezeichnet. Damit lassen sich kryptoanalytische Analysen mit bestimmten Verfahren verhindern. Deswegen wird diese Eigenschaft im Algorithmus 7.7 zusätzlich gefordert.

Nur über Kenntnis der Faktorisierung von n können somit Werte der Folgeglieder a_i bestimmt werden. Also sind weitere Bit-Vorhersagen nur durch Kenntnis von p, q und des letzten Folgeglieds a_k möglich. Dies kann dann über folgende skizzierte Schritte geschehen:

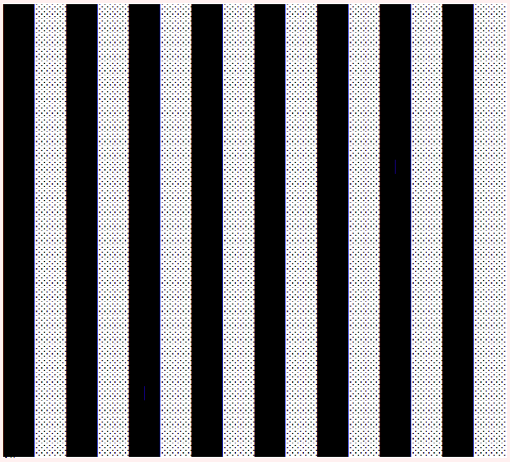
- Bestimme die Quadratwurzeln x_1, x_2 von a_k mod p und y_1, y_2 von a_k mod q. Eine Möglichkeit der Bestimmung liefert der Algorithmus 6.18.
- Berechne mit dem Chinesischen Restsatz gemäß Satz A.41 $z \equiv_p x_j,\ z \equiv_q y_k$ für $j, k \in \{1, 2\}$.
- Wähle aus den vier Lösungen über Zusatzinformationen zur Bestimmung von a_{k-1} die „Richtige“ aus.

Beispiel 7.8: Zu kurze Zykluslänge der Zufallszahlenfolge

Seien $p := 7$ und $q := 11$ und $n = p \cdot q = 77$. Sei weiter $a := 3$. Damit ist ggT$(a, n) = 1$ und $a_0 = 9$. Dann lässt sich eine Pseudozufallszahl der Länge $k = 8$ erstellen mit

i	a_i	Bitfolge	a_i mod 2	a_i^2 mod 77
0	9	00001001	**1**	4
1	4	00000100	**0**	16
2	16	00010000	**0**	25
3	25	00011001	**1**	9
4	9	00001001	**1**	4
5	4	00000100	**0**	16
6	16	00010000	**0**	25

Setzen wir das höchste Bit 8 stets auf 1, um in jedem Fall eine Pseudozufallszahl aus 8 Bits zu bekommen, und verwenden wir das letzte Bit von a_0 bereits für die Zufallsfolge, so erhalten wir die Pseudozufallszahl 11001100 (204). Setzen wir das bis zu einer Länge von 1024 Bit fort und tragen die Bits kodiert in ein 32×32 Felder großes Bild ein, so ergibt sich folgende Abbildung (dunkles Feld$\hat{=}1$, helles Feld$\hat{=}0$):



An dieser Stelle sehen wir, dass die Bit-Länge im letzten Beispiel nicht optimal gewählt ist, da in der erzeugten Bitfolge ein Zyklus enthalten ist. Sehen wir uns die Wahl der Zahlen n und x noch einmal an, können wir erkennen, dass dies so kommen musste. Wir betrachten dazu die prime Restklassengruppe $(\mathbb{Z}_n^*, \cdot_n, 1)$. Zunächst setzen wir voraus, dass die Primfaktorzerlegung der Ordnung von $\mathbb{Z}_n^*$, $\phi(n)$, mit

$$\phi(n) = \prod_{p|\phi(n)} p^{e(p)},$$

mit den Vielfachheiten $e(p)$ der einzelnen Primfaktoren bekannt ist. Dann gilt

Satz 7.9

Für jeden Primteiler p der Ordnung $|G|$ einer Gruppe $(G, *, \hat{e})$ sei $f(p) \in \mathbb{N}_0$ die größte natürliche Zahl derart, dass $a^{|G|/p^{f(p)}}$ mod $n = 1$ für ein $a \in G$ ist. Es ist $0 \leq f(p) \leq e(p)$ und

$$\text{ord}_G(a) = \prod_{p||G|} p^{e(p)-f(p)}.$$

Beweis.

$\text{ord}_G(a)$ ist von der Form $\prod_{p||G|} p^{x(p)}$ mit $0 \leq x(p) \leq e(p) - f(p)$ für alle $p \mid |G|$. Wegen der Definition von $f(p)$ gilt sogar $x(p) = e(p) - f(p)$ für alle $p \mid |G|$.

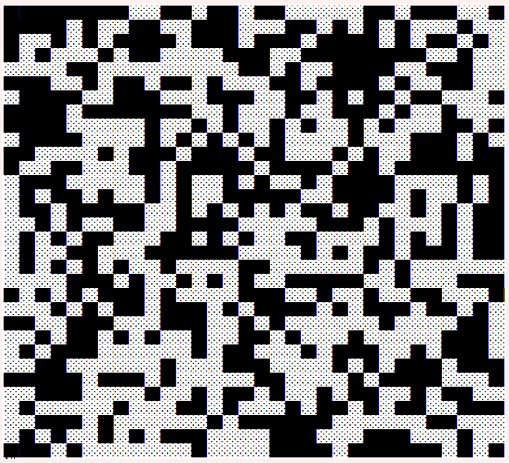
□

Beispiel 7.10: Verbesserte Zykluslänge der Zufallszahlenfolge

Für die Zahlenwerte aus Teil 1 des Beispiels gilt: $\phi(77) = 60 = 2^2 \cdot 3 \cdot 5$. Mit $a = 3$ sind $f(2) = 1,\ f(3) = 0,\ f(5) = 0$ und damit $\text{ord}_{\mathbb{Z}_{77}^*}(3) = 2^{2-1} \cdot 3^{1-0} \cdot 5^{1-0} = 30$. Die Folgeglieder $a_{i-1},\ i = 1, \ldots, k-2$ bestehen aus der Menge

$$\left\{ a^{\left(2^i \bmod \text{ord}_{\mathbb{Z}_{77}^*}(a)\right)} \bmod n;\ i = 1, \ldots, k-2 \right\} \subset \langle a^2 \rangle \subset \langle a \rangle.$$

Die Anzahl der zyklenfreien Elemente in der Menge ist 4. Damit wiederholen sich bei mehr als vier erzeugten Folgegliedern die Bitwerte. Ein gutes Beispiel für eine 1024-Bit-Zufallszahl mit musterfreier Verteilung der Bitwerte zeigt folgende Abbildung:



Hierbei wurden

$$\begin{aligned} n &= 340282366920938463463374607431768211457 \\ &= 59649589127497217 \cdot 5704689200685129054721 \end{aligned}$$

und $a = 7$ mit $\text{ord}_{\mathbb{Z}_n^*}(7) = 132922799578491585061963186389652992$ gewählt.

Im vergangenen Beispiel ist das Problem

$$k := \max\left\{ \left| \left\{ a^{2^i \bmod \text{ord}_{\mathbb{Z}_n^*}(a)} \bmod n;\ a \in \mathbb{Z}_n^*,\ 1 \leq i \leq k-2,\ n \in \mathbb{N} \right\} \right| \ ;\ a \in \mathbb{Z}_n^* \right\}$$

zu lösen. Es ist somit $k \leq \min\{i :\ a \in \mathbb{Z}_n^*,\ 2^i < \text{ord}_{\mathbb{Z}_n^*}(a)\}$ zu setzen.

8. Kryptographische Prüfwerte

In Abschnitt 2.2.2 haben wir bereits Beispiele für Prüfwerte gesehen. Es ging darum, zu erkennen, ob Daten in irgendeiner Weise z.B. bei der Übertragung verändert worden sind. Eine zufällige Veränderung, etwa das Kippen eines Bits, wird beim ISBN-10-Code erkannt. Bei einer absichtlichen Veränderung zeigt der ISBN-10-Code aber Schwächen, da es viele gültige ISBN-10-Codes mit demselben Prüfwert gibt. Diese so genannten Kollisionen sind in kryptographischen Anwendungen jedoch nicht erwünscht. Wir unterscheiden zwei Arten der Prüfwerteerzeugung. Bei Hashfunktionen werden Prüfwerte nur auf Grundlage der Daten erzeugt, während bei den Message Authentication Codes zusätzlich ein Schlüssel in die Prüfwerterzeugung eingeht. Sie Prüfwerterzeugung bei Message Authentication Codes erfolgt oft über symmetrische Chiffriersysteme.

8.1. Kryptographische Hashfunktionen

In der Informatik gibt es die Klasse der Hashfunktionen, die Eingabewerte beliebiger Bitlänge auf einen Ausgabewert einer festen Bitlänge abbildet. Dies wird Kompressionseigenschaft genannt. Ferner soll der Ausgabewert einfach zu bestimmen sein. Für kryptographische Anwendungen benötigen wir zusätzliche Eigenschaften, die wir festlegen wollen.

Definition 8.1: Kryptographische Hashfunktion

Sei $n \in \mathbb{N}$ gegeben. Eine kryptographische Hashfunktion ist eine kryptographische Funktion

$$h : \mathbb{F}_2^* \to \mathbb{F}_2^n$$

mit folgenden Eigenschaften: es darf nicht möglich sein, mit vertretbarem Aufwand

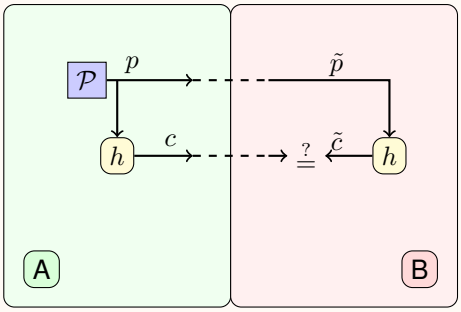
- zu einem gegebenen Wert $c \in \mathbb{F}_2^n$ ein $p \in \mathbb{F}_2^*$ mit $h(p) = c$,
- zu einem gegebenen Wert $p \in \mathbb{F}_2^*$ ein $\tilde{p} \in \mathbb{F}_2^*$ mit $\tilde{p} \neq p$ und $h(p) = h(\tilde{p})$,
- $p, \tilde{p} \in \mathbb{F}_2^*$ mit $h(p) = h(\tilde{p})$ (Kollisionsresistenz)

zu bestimmen. Der Wert der Funktion wird als Modification Detection Code oder kurz Hashwert bezeichnet.

Bemerkung.
Da n in der Regel kleiner als die Länge der Eingabedaten ist, wird von einer Kompression der Daten gesprochen. Die Kompression wird meist durch eine Teilfunktion innerhalb von h erzeugt.

Abbildung 8.1 zeigt das Prinzip der Überprüfung von Daten mit Hilfe einer kryptographischen Hashfunktion.

Abbildung 8.1.: Hashwert-Erzeugung



Eine kryptographische Hashfunktion ist im Normalfall aufgrund der Länge der Eingabedaten keine injektive Funktion. Um die geforderten Eigenschaften erfüllen zu können, muss die Bitlänge n der Ausgabedaten groß genug gewählt werden.

Geburtstagsparadoxon
Es sei n die Bitlänge der Ausgabedaten einer kryptographischen Hashfunktion. Angenommen, jede Bitfolge tritt mit gleicher Wahrscheinlichkeit auf, wie viele verschiedene Bitfolgen müssen im Mittel ausprobiert werden, um mit einer Wahrscheinlichkeit von 50% zwei gleiche Werte der Hashfunktion zu erhalten?

Sei $r := 2^n$. Es gibt insgesamt r^k Möglichkeiten, bei k-maliger Wiederholung eine beliebige Bitfolge der Länge n zu wählen. Soll jedesmal eine andere Bitfolge gewählt werden, gibt es zunächst r, dann $r-1$, $r-2$ bis $r-k+1$ Möglichkeiten (ohne Zurücklegen, mit Reihenfolge). Damit ist die Wahrscheinlichkeit, k-verschiedene Bitfolgen aus r^k Bitfolgen auszuwählen gleich

$$\mathbb{P}_{r,k} = \frac{r \cdot (r-1) \cdot (r-2) \cdot \ldots \cdot (r-k+1)}{r^k}.$$

Die Gegenwahrscheinlichkeit steht für wenigstens ein Duplikat in der Auswahl. Diese Wahrscheinlichkeit soll eine Wahrscheinlichkeit von $b \in [0,1]$ haben:

$$\begin{aligned} b &= 1 - \frac{r \cdot (r-1) \cdot (r-2) \cdot \ldots \cdot (r-k+1)}{r^k} \\ &= 1 - \left(1 \cdot \frac{r-1}{r} \cdot \frac{r-2}{r} \cdot \ldots \cdot \frac{r-k+1}{r}\right) \\ &= 1 - \left(\left(1 - \frac{1}{r}\right) \cdot \left(1 - \frac{2}{r}\right) \cdot \ldots \cdot \left(1 - \frac{k-1}{r}\right)\right) \\ &\geq 1 - \left(e^{-1/r} \cdot e^{-2/r} \cdot \ldots \cdot e^{-(k-1)/r}\right) \\ &= 1 - e^{-\frac{k(k-1)}{2r}} \end{aligned} \tag{8.1}$$

Die Abschätzung gilt wegen

$$e^{-\frac{1}{r}} = \lim_{k \to \infty} \left(1 + \frac{-\frac{1}{r}}{k}\right)^k.$$

Zu gegebenem n wollen wir ein k so bestimmen, dass wir wenigstens b% Wahrscheinlichkeit bekommen und lösen deshalb die folgende Ungleichung nach k auf:

$$\begin{array}{lll} & 1-b & \leq & e^{-\frac{k(k-1)}{2r}} \\ \Leftrightarrow & \ln(1-b) & \leq & -\frac{k(k-1)}{2\cdot 2^n} \\ \Leftrightarrow & 2^{n+1}\ln(1-b) & \leq & -k(k-1) \\ \Leftrightarrow & 0 & \geq & k^2 - k + 2^{n+1}\ln(1-b) \\ \Rightarrow & k & = & 1/2 + \sqrt{1/4 - 2^{n+1}\ln(1-b)} \end{array}$$

Wir erhalten (ein positives) k in der Größenordnung $k \approx \sqrt{-2\ln(1-b)}\sqrt{2^n}$.

Beispiel 8.2

Bei einem 32 Bit Hashwert benötigen wir demnach etwa 77000 Versuche, um mit einer Wahrscheinlichkeit von 50% wenigstens einen Treffer zu erhalten. Bereits mit etwa 200000 Versuchen ergibt sich eine Wahrscheinlichkeit von 99%. Für 128 bzw. 160 Bit Hashwerte werden ohne Zusatzinformationen entsprechend bereits Versuche in einer Größenordnung von 10^{19} bzw. 10^{24} benötigt, was praktisch kaum in angemessener Zeit durchzuführen ist.

Konstruktion von kryptographischen Hashfunktionen

Vielen kryptographischen Hashfunktionen liegt die Merkle-Damgård-Struktur zugrunde. Dabei wird iterativ ein so genannter Chaining Value ($\mathsf{CV}_i \in \mathbb{F}_2^n$) der Bitlänge n auf Basis des Vorgängerwerts CV_{i-1} und einem Block $p_{i-1} \in \mathbb{F}_2^m$ der Eingabedaten der Bitlänge m erzeugt. Zu Beginn wird ein Initialisierungswert IV benötigt.

$$\begin{aligned} &\mathsf{CV}_0 := \mathsf{IV}, \\ &\mathsf{CV}_i := f(\mathsf{CV}_{i-1}, p_{i-1}), \quad 1 \leq i \leq L, \end{aligned}$$

Die Hashfunktion $h : \mathbb{F}_2^* \to \mathbb{F}_2^n$ ergibt sich mit $p = p_0||p_1||\ldots||p_{L-1}$ als

$$h(p) := \mathsf{CV}_L.$$

p muss zunächst möglicherweise durch Padding auf ein Vielfaches der Blocklänge m gebracht werden. Ein Initialisierungsvektor IV$\in \mathbb{F}_2^n$ und $f : \mathbb{F}_2^n \times \mathbb{F}_2^m \to \mathbb{F}_2^n$ sind in der Regel durch die verwendete Hashfunktion festgelegt. Weiter gilt

$$\mathsf{CV}_i = f(\mathsf{CV}_{i-1}, p_{i-1}) = f(f(\mathsf{CV}_{i-2}, p_{i-2}), p_{i-1}) \tag{8.2}$$

und damit ist mit zwei Eingaben $q, \tilde{q}$ mit $h(q) = h(\tilde{q})$ auch $h(q||p) = h(\tilde{q}||p)$ für beliebige Daten p. Ein Kollisionsblock genügt, um die ganze Hashfunktion anzugreifen.

MD5-Hashwerte

Eine der bekanntesten kryptographischen Hashfunktionen ist der MD5-Algorithmus md5 : $\mathbb{F}_2^* \to \mathbb{F}_2^{128}$, der aus einer beliebigen Bitfolge eine 128 Bit-Folge erzeugt. Gegeben seien Daten $p \in \mathbb{F}_2^b$,

$$p = p_0 p_1 \ldots p_{b-1}$$

beliebiger Länge. Zunächst muss p durch Padding auf eine Länge kongruent 448 modulo 512 durch 1-0-Padding gebracht werden. Danach wird die Länge der Daten binär dargestellt und die letzten 64 Bits den Daten hinzugefügt, sodass die Gesamtlänge der erweiterten Daten nunmehr durch 512 teilbar ist. Die erweiterten Daten werden in 32 Bit Blöcke aufgeteilt,

$$p_e = m_0 m_1 \dots m_{N-1},$$

wobei N ein Vielfaches von 16 ist.
Ein Initialisierungsvektor der Länge 128 Bit,

$$\text{IV} := A||B||C||D$$

mit $A = 0x67452301$, $B = 0xefcdab89$, $C = 0x98badcfe$ und $D = 0x10325476$ wird erzeugt. Zudem werden vier Hilfsfunktionen

$$\begin{aligned}
g_0 &: \mathbb{F}_2^{32} \times \mathbb{F}_2^{32} \times \mathbb{F}_2^{32} \to \mathbb{F}_2^{32}, \quad (X,Y,Z) \mapsto g_0(X,Y,Z) := (X \wedge Y) \vee (\neg X \wedge Z),\\
g_1 &: \mathbb{F}_2^{32} \times \mathbb{F}_2^{32} \times \mathbb{F}_2^{32} \to \mathbb{F}_2^{32}, \quad (X,Y,Z) \mapsto g_1(X,Y,Z) := (X \wedge Z) \vee (Y \wedge \neg Z),\\
g_2 &: \mathbb{F}_2^{32} \times \mathbb{F}_2^{32} \times \mathbb{F}_2^{32} \to \mathbb{F}_2^{32}, \quad (X,Y,Z) \mapsto g_2(X,Y,Z) := X \oplus Y \oplus Z,\\
g_3 &: \mathbb{F}_2^{32} \times \mathbb{F}_2^{32} \times \mathbb{F}_2^{32} \to \mathbb{F}_2^{32}, \quad (X,Y,Z) \mapsto g_3(X,Y,Z) := Y \oplus (X \vee \neg Z)
\end{aligned}$$

eingeführt. Weiter werden 64 Werte T_k, $k = 1, \dots, 64$ mit

$$T_k := \lfloor \underbrace{4294967296}_{2^{32}} \cdot |\sin(k)| \rfloor, (k \text{ im Bogenmaß}),$$

benutzt. Tabelle 8.1 zeigt alle sich ergebenden Konstanten.

Tabelle 8.1.: Die Konstanten T_k

3614090360,	3905402710,	606105819,	3250441966,
4118548399,	1200080426,	2821735955,	4249261313,
1770035416,	2336552879,	4294925233,	2304563134,
1804603682,	4254626195,	2792965006,	1236535329,
4129170786,	3225465664,	643717713,	3921069994,
3593408605,	38016083,	3634488961,	3889429448,
568446438,	3275163606,	4107603335,	1163531501,
2850285829,	4243563512,	1735328473,	2368359562,
4294588738,	2272392833,	1839030562,	4259657740,
2763975236,	1272893353,	4139469664,	3200236656,
681279174,	3936430074,	3572445317,	76029189,
3654602809,	3873151461,	530742520,	3299628645,
4096336452,	1126891415,	2878612391,	4237533241,
1700485571,	2399980690,	4293915773,	2240044497,
1873313359,	4264355552,	2734768916,	1309151649,
4149444226,	3174756917,	718787259,	3951481745

Die Datenblöcke von p_e gehen zweimal in die Verarbeitung ein. Einmal sequentiell, ein zweites Mal nicht reihenfolgetreu, sondern permutiert bzgl. der Permutation $\sigma(k)$ in Tabelle 8.2

Tabelle 8.2.: Die Permutationen $\sigma(k)$

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	6	11	0	5	10	15	4	9	14	3	8	13	2	7	12
5	8	11	14	1	4	7	10	13	0	3	6	9	12	15	2
0	7	14	5	12	3	10	1	8	15	6	13	4	11	2	9

Der Kern des Verfahrens besteht in einer 64-maligen Iteration, in welcher die Hilfsfunktionen Verwendung finden. Ein zyklischer Linksshift um s_k Stellen gemäß Tabelle 8.3 erfolgt am Ende jeder Iteration.

Tabelle 8.3.: Die Shiftwerte s_k

7	12	17	22	7	12	17	22	7	12	17	22	7	12	17	22
5	9	14	20	5	9	14	20	5	9	14	20	5	9	14	20
4	11	16	23	4	11	16	23	4	11	16	23	4	11	16	23
6	10	15	21	6	10	15	21	6	10	15	21	6	10	15	21

Das gesamte Verfahren ist in Algorithmus 8.3 wiedergegeben.

Algorithmus 8.3: md5

Eingabe: Daten $p \in \mathbb{F}_2^*$, weiter A, B, C, D, T, σ, s.
Ausgabe: 128 Bit Hashwert $h(p)$.

1: $p_e = m_0||m_1||\dots||m_{N-1}$ aus p erzeugen.
2: **for** $i := 0$ **to** $\lfloor N/16 \rfloor - 1$ **do**
3: **for** $j := 0$ **to** 15 **do**
4: $x_j := m_{i\cdot 16+j}$
5: **end for**
6: **for** $j := 0$ **to** 63 **do**
7: $w_j := x_{\sigma(j)}$
8: **end for**
9: $Q_{-4}||Q_{-3}||Q_{-2}||Q_{-1} := A||B||C||D$.
10: **for** $k := 0$ **to** 63 **do**
11: $Q_k := Q_{k-1}+((Q_{k-4}+g_{\lfloor k/16 \rfloor}(Q_{k-1}, Q_{k-2}, Q_{k-3})+w_k+T_k) \bmod 2^{32} <<< s_k)$
12: **end for**
13: $A||B||C||D := Q_{60}+Q_{-4}||Q_{63}+Q_{-3}||Q_{62}+Q_{-2}||Q_{61}+Q_{-1}$, jeweils mod 2^{32}.
14: **end for**
15: **return** $h(p) = A||B||C||D$.

Bemerkung.
(1) Durch das Padding zu Beginn und die Addition am Ende ist das Verfahren durch die Gesamtheit aller Iterationen gegeben.
(2) md5 folgt der Merkle-Damgard-Struktur mit Initial $CV_0 = IV$ und der Kompressionsfunk-

tion

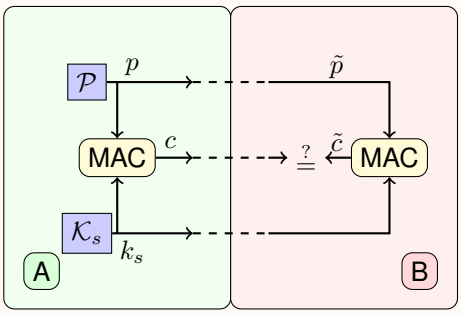
$$\mathrm{CV}_i = f((A||B||C||D)_{i-1}, p_{e,i-1}).$$

Dabei ist $p_{e,i-1}$ der 512-Bit-Block $m_{(i-1)\cdot 16}||\ldots||m_{(i-1)\cdot 16+15}$.
(3) Wie oben erwähnt genügt ein Kollisionsblock, um das ganze Verfahren anzugreifen. Dies wurde mehrfach erreicht, z.B. von Wang[1], so dass md5 heute nicht mehr sicher ist.

8.2. Message Authentication Codes

Möchte eine Instanz Daten zu einer anderen Instanz senden und jener ermöglichen zu überprüfen, ob die Daten den tatsächlich gesendeten entspricht, erschweren die für kryptographische Hashfunktionen angesprochenen Probleme die Integritätsüberprüfung. Sicherheit kann die Verwendung eines (symmetrischen) Schlüssels k_s bieten, der zusammen mit den Daten p in die Hashfunktion eingeht. Solche kryptographischen Hashfunktionen heißen Message Authentication Codes. Sie werden mit MAC abgekürzt. Dabei soll es nicht möglich sein, aus einer Reihe gegebener Paare $(p_i, \mathrm{MAC}_{k_s}(p_i))$ auf den Schlüssel zu schließen. Das Prinzip von MACs ist in Abbildung 8.2 zu sehen.

Abbildung 8.2.: Message Authentication Code-Erzeugung



CBC-MAC
Die Verwendung eines Schlüssels legt die Verwendung symmetrischer Chiffriersysteme nahe. Tatsächlich wird das in der Praxis häufig gemacht, indem lediglich der letzte Block eines Verschlüsselungsvorgangs $e_{k_s}(p)$ im CBC-Betriebsmodus als MAC benutzt wird[2].

[1] X. Wang, H. Yu, How to break MD5 and other hash functions, http://merlot.usc.edu/csac-f06/papers/Wang05a.pdf
[2] etwa der heute nicht mehr als sicher angesehene Data Authentication Algorithm, http://csrc.nist.gov/publications/fips/fips113/fips113.html

Beispiel 8.4

Sei für gegebene Daten $p = p_0||p_1||\dots||p_N$ in Blockgestalt der Blockgröße n der Message Authentication Codes

$$\mathsf{MAC}_{k_s} : \mathbb{F}_2^* \to \mathbb{F}_2^n, \mathsf{MAC}_{k_s}(p) := e_{k_s}(\underbrace{p_0 \oplus p_1 \oplus \dots \oplus p_N}_{\Delta(p)})$$

gegeben. Angenommen, $(p, \mathsf{MAC}_{k_s}(p))$ seien bekannt. Sei weiter $q := q_0||q_1||\dots||q_{N-1}$ beliebig mit derselben Blockgröße wie p. Sei $q_N := \Delta(q) \oplus \Delta(p)$. Dann gilt

$$\mathsf{MAC}_{k_s}(q||q_N) = e_{k_s}(\Delta(q||q_N)) = \mathsf{MAC}_{k_s}(p).$$

Dieser MAC ist unabhängig vom verwendeten Chiffriersystemen leicht anzugreifen.

Eine Erweiterung davon stellen CMACs (Cipher MAC) dar. Dabei werden aus einem Schlüssel zwei temporäre Schlüssel erzeugt, von denen einer vor dem letzten Verschlüsselungsschritt mit dem Klartextblock per XOR verknüpft wird. Dies Verfahren gilt gemäß [7] als derzeit ausreichend sicher.

MDC-MAC

Eine andere Möglichkeit zur Erzeugung von MACs sind solche auf Basis kryptographischer Hashfunktionen. Eine Idee ist zunächst, nicht nur die Daten p für die Hashfunktion zu verwenden, sondern den Schlüssel k zusammen mit den Daten, z.B. $h(k||p)$.

Beispiel 8.5: Naiver MDC-MAC mit md5

Seien Daten $p \in \mathbb{F}_2^*$ und ein Schlüssel k gegeben. Wir wollen $\mathsf{md5}(k||p)$ zur Integritätssicherung verwenden. $\mathsf{md5}(k||p)$ kann aber angegriffen werden. Dazu betrachten wir zunächst

$$p_e = \underbrace{k||p||\underbrace{10\dots0}_{v \text{ Bits}}||s}_{\equiv 448 \bmod 512},$$

wobei s die Bitlänge von p mod 2^{64} ist. Wir erzeugen ein $q \in \mathbb{F}_2^{512}$ mit

$$q := \underbrace{10\dots0}_{v \text{ Bits}}||s||q_0,$$

wobei q_0 beliebig ist. Dann gilt gemäß (8.2)

$$\mathsf{md5}(k||p||q) = f(\mathsf{md5}(k||p), q_0).$$

Ein Angreifer kann damit neue Daten mit einem gültigen MAC versehen ohne den Schlüssel zu kennen, wenn p länger als 65 Bits ist.

Diese und auch andere einfache Herangehensweisen haben sich als nicht sicher erwiesen[3].

[3] vgl. [4]

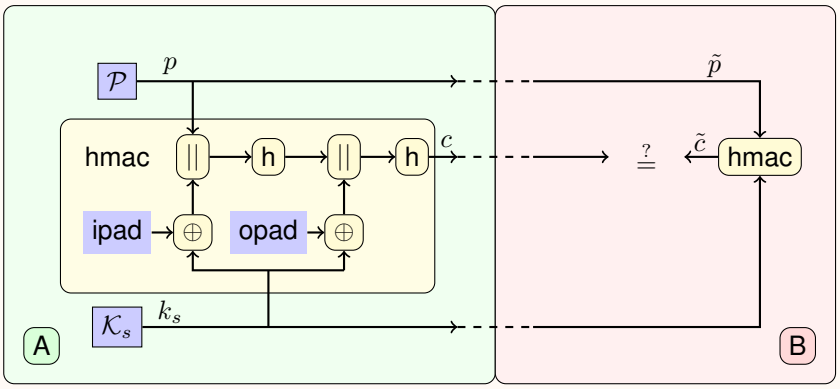
HMAC

Der hashbasierte MAC ist ein MDC-MAC, bei dem der Schlüssel mehr „gestreut“ wird. Im RFC 2104[4] wird das Verfahren

$$\text{HMAC}_{k_s}(p) := h(k_s \oplus \text{opad} || h(k_s \oplus \text{ipad} || p))$$

vorgeschlagen. Sei h eine kryptographische Hashfunktion mit Ausgabelänge n Bits. Dann sind Konstanten ipad:$= 0x36 \ldots 36$ und opad:$= 0x5c \ldots 5c$ gegeben, wobei die Bytewerte jeweils $n/8$ mal wiederholt werden und der Schlüssel gegebenenfalls durch 0-Padding auf die Ausgabelänge n erweitert oder durch Anwenden der Hashfunktion reduziert wird. Abbildung 8.3 zeigt den Ablauf der HMAC-Erzeugung und Überprüfung.

Abbildung 8.3.: Hash Message Authentication Code-Erzeugung



[4] Request for Comments, https://www.ietf.org/rfc/rfc2104.txt

9. Digitale Signaturen

9.1. Prinzip der digitalen Signatur

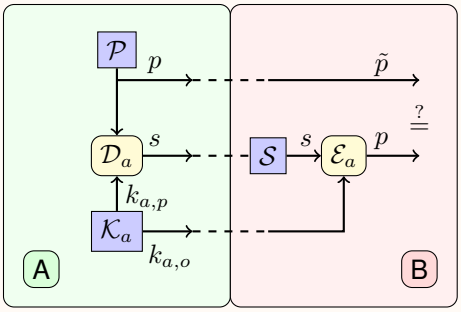
Eine digitale Signatur ist die digitale Form der Unterschrift. Mit einer digitalen Signatur will eine Instanz nachweisen bzw. soll ihr nachgewiesen werden können, dass die Daten von ihr stammen. Dazu benötigt die Instanz etwas, das nur sie zur Verfügung hat und womit sie Daten signieren kann. Andere Instanzen müssen mit etwas öffentlich zugänglichem in der Lage sein, die Signatur zu überprüfen. Asymmetrische Verfahren können hierfür eine Lösung sein. Denn dabei besitzt eine Instanz einen privaten Schlüssel, den nur sie kennt. Werden damit Daten wie beim Entschlüsseln „entschlüsselt“, kann durch den dazugehörigen öffentlichen Schlüssel durch den Umkehrvorgang des Verschlüsselns die Signatur überprüft werden. Die Rollen der Ver- und Entschlüsselungsfunktion müssen vertauschbar sein,

$$p = e_{k_{a,o}}(\underbrace{d_{k_{a,p}}(p)}_{s}). \tag{9.1}$$

Wir haben somit eine Integritätsprüfung und im Erfolgsfall die eindeutige Zurechenbarkeit wegen des privaten Schlüssels.

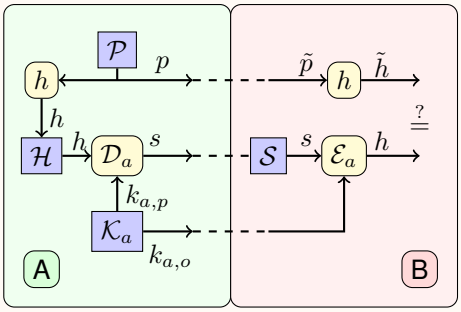
Es gibt zwei prinzipielle Vorgehensweisen beim Erstellen einer digitalen Signatur. Bei der digitalen Signatur mit Nachrichtenrückgewinnung wird die Nachricht selbst signiert. Die Überprüfung erfolgt durch den Vergleich der unverschlüsselten Nachricht mit den durch den öffentlichen Schlüssel entsignierten Daten.

Abbildung 9.1.: Digitale Signatur mit Nachrichtenrückgewinnung



Bei der digitalen Signatur mit Anhang wird nicht die eigentliche Nachricht, sondern lediglich ein Hashwert der Nachricht signiert. Dies hat den Vorteil, dass relativ wenige Bits signiert werden müssen. Da bei asymmetrischen Verfahren der Berechnungsaufwand groß ist, spielt das eine wichtige Rolle. Der entsignierte Hashwert wird von einer zweiten Instanz mit dem Hashwert der Nachricht verglichen. In der Praxis wird meist diese Variante der digitalen Signatur benutzt.

Abbildung 9.2.: Digitale Signatur mit Anhang



9.2. RSA-basierte digitale Signatur

Das RSA-Verfahren ist eines der am häufigsten benutzten Verfahren zur Erstellung digitaler Signaturen. Die Vorgehensweise ist analog zur asymmetrischen Verschlüsselung. Lediglich die Rollen der Schlüssel werden vertauscht.

Algorithmus 9.1: RSA Schlüsselerzeugung zur digitalen Signatur

Eingabe: $n = p \cdot q$ mit p, q prim.
Ausgabe: Öffentlicher Schlüssel $(n, k_{a,o})$.
1: $\phi(n) := (p-1) \cdot (q-1)$.
2: Finde $k_{a,o}$ mit $1 < k_{a,o} < \phi(n)$ und ggT$(k_{a,o}, \phi(n)) = 1$ (öffentlicher Schlüssel $(n, k_{a,o})$).
3: $k_{a,p} := k_{a,o}^{-1}$ mod $\phi(n)$ (privater Schlüssel $(n, k_{a,p})$).
4: **return** $(n, k_{a,o})$.

Algorithmus 9.2: Digitale Signatur mit RSA

Eingabe: Zur Signierung: Privater Schlüssel $(n, k_{a,p})$, Daten p (kodiert mit $0 < p \leq n$). Zur Entsignierung: Öffentlicher Schlüssel $(n, k_{a,o})$.
Ausgabe: Signatur s, entsignierte Daten p' gleich p?
1: $s := d_{(n,k_{a,p})}(p) = p^{k_{a,p}}$ mod n.
2: $p' := e_{(n,k_{a,o})}(s) = s^{k_{a,o}}$ mod n.
3: **return** $s, p' == p$.

Nur wenn der öffentliche Schlüssel anerkannt ist, kann die Echtheit der Signatur korrekt überprüft werden. Stammt der öffentliche Schlüssel von einer anderen Instanz, kann diese der ersten Instanz signierte Daten unterschieben.

Beispiel 9.3

Wir wählen $p = 131$ und $q = 139$, so erhalten wir $n = 18209$. Es ist $\phi(18209) = 17940$ und wegen ggT$(17, 17940) = 1$ ist $k = 17$ eine mögliche Wahl für den öffentlichen Schlüssel. Sei $M = 9682$ die zu signierende Zahl. Um M signieren zu können, benötigen wir k' mit $k \cdot k' = 1$ mod 17940. Es ist $k' = 10553$ und damit wird p zu

$$s = M^{k'} \text{ mod } 18209 = 9682^{10553} \text{ mod } 18209 = 8873.$$

Zur Überprüfung rechnen wir

$$M' = s^k \text{ mod } 18209 = 8873^{17} \text{ mod } 18209 = 9682.$$

Die Signatur wird als gültig angenommen.

Für zwei Signaturen $s_1 = p_1^{k_{a,p}}$ mod n und $s_2 = p_2^{k_{a,p}}$ mod n gilt $s := s_1 \cdot s_2$ mod $n = p_1^{k_{a,p}} \cdot p_2^{k_{a,p}}$ mod $n = (p_1 \cdot p_2)^{k_{a,p}}$ mod n und s ist eine gültige digitale Signatur von $p = p_1 \cdot p_2$ mod n. Dies kann für einen Angriff genutzt werden. Soll eine digitale Signatur einer Nachricht p erzeugt werden, kann nach Wahl von $p_1 \in \mathbb{Z}_n$ mit $p \neq p_1$ eine weitere Nachricht $p_2 = (p \cdot p_1^{-1})$ mod n berechnet werden. Sind beide Nachrichten p_1 und p_2 zu s_1 bzw. s_2 signiert, kann der Angreifer die digitale Signatur s zu p berechnen. Dies ist ein Chosen-Plaintext-Angriff. Dies kann durch eine digitale Signatur mit Anhang vermieden werden, da eine kryptographische Hashfunktion h eine Einwegfunktion ist und zu jedem p ein x gefunden werden muss mit $p = h(x)$.

9.3. El Gamal-basierte digitale Signatur

Das El-Gamal-Verfahren entspricht zunächst nicht den Anforderungen (9.1). Die Idee, den diskreten Logarithmus als Einwegfunktion zu benutzen kann aber verwendet werden. Ferner kann mit der selben Schlüsselerzeugung gearbeitet werden. Zur Vorbereitung benötigen wir zwei Aussagen.

Satz 9.4

Sei $(G, \odot, e)$ eine multiplikative Gruppe und $g \in G$.

(1) Sei $u \in \mathbb{Z}$. Es ist $g^u = 1$ genau dann, wenn u durch die Ordnung von g in G teilbar ist.

(2) Seien $k, l \in \mathbb{Z}$. Dann gilt $g^k = g^l$ genau dann, wenn $k \equiv_{\text{ord}_G(g)} l$.

Beweis.

(1) Sei $n = \text{ord}_G(g)$. Ist $u = q \cdot n$, folgt $g^{q \cdot n} = (g^n)^u = 1^u = 1$. Ist andererseits $g^u = 1$. Sei $u = q \cdot n + r$ mit $r \in \mathbb{Z}_n$. Dann folgt $g^r = g^{u - q \cdot n} = g^u \cdot (g^n)^{-q} = 1$.

(2) Setze in (1) $u := k - l$

□

Folgender Algorithmus zeigt das Vorgehen der einer El-Gamal-Signatur.

Algorithmus 9.5: El-Gamal Schlüsselerzeugung zur digitalen Signatur

Eingabe: $p := 2q + 1$ sichere Primzahl, $q \in \mathbb{N}$ prim und $g \in \mathbb{Z}_p^*$ mit $\langle g \rangle = \mathbb{Z}_p^*$.
Ausgabe: Öffentlicher Schlüssel (p, g, y_A).
1: Bestimme zufällig $x_A \in \{2, \ldots, p-2\}$ (privater Schlüssel).
2: $y_A := g^{x_A} \bmod p$.
3: **return** (p, g, y_A).

Algorithmus 9.6: Digitale Signatur nach El-Gamal

Eingabe: Zur Signierung: Privater Schlüssel x_A, Daten M (kodiert mit $M \in \mathbb{Z}_p^*$). Zur Entsignierung: Öffentlicher Schlüssel (p, g, y_A).
Ausgabe: Signatur $\tilde{s}$, entsignierte Daten $z_1 \equiv_p z_2$?
1: Bestimme $x_B \in \{2, \ldots, p-2\}$ mit ggT$(x_B, p-1) = 1$.
2: $y_B := g^{x_B} \bmod p$.
3: Berechne $x_B^{-1} \bmod (p-1)$ und $s := x_B^{-1} \cdot (M - x_A \cdot y_B) \bmod (p-1)$.
4: $\tilde{s} := (M, y_B, s)$.
5: Berechne $z_1 := y_A^{y_B} \cdot y_B^s \bmod p$, $z_2 := g^M \bmod p$.
6: **return** $\tilde{s}, z_1 == z_2$.

Satz 9.7

Ein Tupel $\tilde{s} = (M, y_B, s)$ ist genau dann eine gültige digitale Signatur nach El-Gamal, wenn $y_A^{y_B} \cdot y_B^s \bmod p = g^M \bmod p$, was dem Vorgehen gemäß den Algorithmen 9.5 und 9.6 entspricht.

Beweis.

„⇐“: Es seien $p, g, y_A, x_A, M, x_B, y_B$ und s gemäß den Algorithmen 9.5 und 9.6 erzeugt. Dann ist

$$\begin{aligned} y_A^{y_B} \cdot {y_B}^s &= g^{x_A \cdot y_B} \cdot g^{x_B \cdot (x_B^{-1}(M - x_A \cdot y_B)) \bmod (p-1)} \bmod p \\ &\overset{\text{Satz } 9.4}{=} g^{x_A \cdot y_B} \cdot g^{x_B \cdot (x_B^{-1}(M - x_A \cdot y_B))} \bmod p \\ &= g^{x_A \cdot y_B} \cdot g^{M - x_A \cdot y_B} \bmod p \\ &= g^M \bmod p. \end{aligned} \tag{9.2}$$

„⇒ “: Es sei $y_A^{y_B} \cdot {y_B}^s \bmod p = g^M \bmod p$ für ein (M, y_B, s) erfüllt und $x_B = \log_g(y_B)$.

Dann folgt mit Satz 9.4 zunächst

$$g^{x_A \cdot y_B + x_B \cdot s} \equiv_p g^M,$$

und da $\langle g \rangle = \mathbb{Z}_p^*$, muss $x_A \cdot y_B + x_B \cdot s \equiv_{p-1} M$ sein. Ist x_B zu $p-1$ teilerfremd, so ergibt sich

$$\begin{aligned} x_A \cdot y_B + x_B \cdot s &\equiv_{p-1} M \\ x_B \cdot s &\equiv_{p-1} M - x_A \cdot y_B \\ s &\equiv_{p-1} x_B^{-1} \cdot (M - x_A \cdot y_B). \end{aligned}$$

Demnach ist $s := x_B^{-1} \cdot (M - x_A \cdot y_B)$ mod $(p-1)$ zu setzen.

□

Beispiel 9.8

Wir erzeugen denselben Schlüssel wie in Beispiel 6.6: Der öffentliche Schlüssel ist somit $(p, g, y_A) = (18743, 15, 16385)$ und der private Schlüssel ist $x_A = 17333$. Sei nun $x_B = 5553$, ggT$(5553, 18742) = 1$. Sei $M = 9682$ die zu signierende Zahl, z.B. der Hashwert einer Nachricht. Wir erhalten

$$\begin{aligned} y_B &= 15^{5553} \text{ mod } 18743 = 13702, \\ x_B^{-1} &= 3743 \\ s &= 3743 \cdot (9682 - 17333 \cdot 13702) \text{ mod } 18742 = 12562. \end{aligned}$$

Somit lautet die digitale Signatur $(M, y_B, s) = (9682, 13702, 12562)$.

Um die Signatur zu überprüfen, berechnen wir

$$\begin{aligned} g^M \text{ mod } p &= 15^{9682} \text{ mod } 18743 = 12614, \\ x_B^{-1} &= 3743 \\ y_A^{y_B} \cdot {y_B}^s \text{ mod } p &= 16385^{13702} \cdot 13702^{12652} \text{ mod } 18743 = 12614. \end{aligned}$$

Damit wird die Signatur als gültig angenommen.

A. Anhang

A.1. Algebraische Strukturen

Kryptologie kommt nicht ohne Zahlentheorie aus. Gruppen, Körper, modulare Arithmetik, Primzahlen und deren Generierung sind dabei von zentraler Bedeutung. Grundlegend sollen nun algebraische Strukturen, die sich durch bestimmte Eigenschaften auszeichnen, untersucht werden. Betrachten wir zunächst verschiedene Rechenoperationen in $\mathbb{R}$:

$3+4$	$=$	7			Abgeschlossenheit der Addition
$3\cdot 4$	$=$	12			Abgeschlossenheit der Multipliaktion
$(3\cdot 4)\cdot 5$	$=$	60	$=$	$3\cdot(4\cdot 5)$	Assoziativität
$3+4$	$=$	7	$=$	$4+3$	Kommutativität
$3\cdot(4+5)$	$=$	27	$=$	$3\cdot 4+3\cdot 5$	Distributivität
$3+0$	$=$	3			Nullelement der Addition
$3\cdot 1$	$=$	3			Einselement der Multiplikation

Mit der Addition und Multipliaktion sind eine Vielzahl an Rechenregeln verbunden. Zudem sind die Null und Eins von besonderer Bedeutung. Die Menge der reellen Zahlen ist zusammen mit den beiden Rechenoperationen ein Gebilde, das wir eine algebraische Struktur nennen. Mit diesem Beispiel im Hinterkopf verallgemeinern wir diese Rechenregeln.

A.1.1. Halbgruppen, Monoide und Gruppen

Grundlage für die weiteren Betrachtungen sind innere Verknüpfungen, d.h. Verknüpfungen, die genau eine Menge betreffen. Damit sind Berechnungen möglich, die bezüglich der Menge abgeschlossen sind. Besonders wichtig sind zweistellige Verknüpfungen und das führt zu folgender

Definition A.1: Innere zweistellige Verknüpfung

Eine innere zweistellige Verknüpfung auf einer Menge M ist eine Abbildung $M\times M\to M$ und wird in der Infix-Form

$$(x,y)\mapsto x*y \text{ (analog mit } x\circ y,\ x\cdot y,\ x+y,\ x\otimes y,\ x\oplus y,\ x+_n y,\ x\cdot_n y\ \ldots)$$

geschrieben.

Nun können wir eine Menge und $n>0$ innere Verknüpfungen auf M zusammenbringen.

Definition A.2: Algebraische Struktur

Die Struktur $(M, *_1, \ldots, *_n)$ bestehend aus einer Menge M und n verschiedenen inneren zweistelligen Verknüpfungen heißt algebraische Struktur.

Bemerkung.
Sollen bestimmte Elemente der Menge M besonders hervorgehoben werden, so können sie mit in die Notation aufgenommen werden, etwa $(M, *, e_*)$.

Für den in der Kryptographie zentralen Begriff der Gruppe wird die Eigenschaft der Assoziativität einer inneren Verknüpfung benötigt. Das Ergebnis der Gesamtverknüpfung ist unabhängig von der Reihenfolge der Ausführung der Verknüpfungen. Manchmal kann es von Bedeutung sein, die Argumente der Verknüpfung vertauschen (kommutieren) zu können. Nicht zuletzt werden zwei verschiedene innere Verknüpfungen miteinander distributiv kombiniert. Wir fassen die drei Begriffe zusammen in der

Definition A.3: Assoziativität, Kommutativität, Distributivität

Eine innere zweistellige Verknüpfung $* : M \times M \to M$ auf M heißt assoziativ, wenn für alle $a, b, c \in M$

$$(a * b) * c = a * (b * c)$$

gilt. Weiter heißt sie kommutativ, wenn für alle $a, b \in M$ gilt

$$a * b = b * a.$$

Eine Verknüpfung $\odot : M \times M \to M$ heißt (links)distributiv über der Verknüpfung $\oplus : M \times M \to M$, wenn für jedes $a, b, c \in M$ gilt

$$a \odot (b \oplus c) = (a \odot b) \oplus (a \odot c).$$

Manche Elemente einer Menge haben besondere Eigenschaften in Verbindung mit der Anwendung einer inneren Verknüpfung. Ein Element verhält sich neutral bezüglich der Verknüpfung, wenn die Verknüpfung mit dem Element kein neues Ergebnis liefert. Falls es zu einem Element einer Menge ein weiteres Element gibt und aus deren Verknüpfung sich das neutrale Element ergibt, so verhalten sich die beiden Elemente invers zueinander. Formell ergibt sich

Definition A.4: Neutrales und inverses Element

Es sei M eine Menge und $*$ eine innere zweistellige Verknüpfung auf M. Ein Element $e \in M$ heißt (rechts-)neutrales Element für $*$, wenn für alle $a \in M$ gilt

$$a * e = a.$$

Seien $a, b, e \in M$ und gilt

$$a * b = e,$$

so heißt b das zu a (rechts-)inverse und a zu b das (links-)inverse Element.

Bemerkung.
(1) Gilt $e * a = a$, so heißt das Element e linksneutral. In der algebraischen Struktur der

Gruppe, die wir bald definieren werden, sind rechts- und linksneutrales Element identisch. Allgemein sprechen wir in der Regel vom neutralen Element einer Menge.
(2) Ebenso werden wir im Zusammenhang mit Gruppen über inverse Elemente sprechen.

Beispiel A.5

Wir betrachten die Menge $M = \{0, 1, 2\}$ und folgende Rechenregeln für eine additive Verknüpfung $+_3 : M \times M \to M$ in Form einer so genannten Additionstabelle:

$+_3$	0	1	2
0	0	1	2
1	1	2	0
2	2	0	1

Die additive Verknüpfung $+_3$ entspricht der „Addition modulo 3“. Es gilt: $a +_3 0 = 0 +_3 a = a$ für jedes $a \in M$. Damit ist 0 das neutrale Element der Menge M bezüglich der Addition modulo 3. Ferner ist $1 +_3 2 = 2 +_3 1 = 0$ und $0 +_3 0 = 0$. Damit ist 0 zu sich selbst, 1 zu 2 und umgekehrt 2 zu 1 invers. Sehen wir uns analog dazu die multiplikative Verknüpfung $\cdot_3 : M \times M \to M$ mit Hilfe einer Multiplikationstabelle an:

$\cdot_3$	0	1	2
0	0	0	0
1	0	1	2
2	0	2	1.

Die multiplikative Verknüpfung $\cdot_3$ entspricht der „Multiplikation modulo 3“. Es gilt: $a \cdot_3 1 = 1 \cdot_3 a = a$ für jedes $a \in M$. Damit ist 1 das neutrale Element der Menge M bezüglich der Multiplikation modulo 3. Ferner ist $1 \cdot_3 1 = 1$ und $2 \cdot_3 2 = 1$. Damit sind 1 und 2 selbstinvers. Da aber 0 kein inverses Element in M hat, existiert nicht zu jedem Element aus M ein inverses Element. Wie wir gleich sehen, ist $(M, +_3, 1)$ eine Gruppe und $(M, \cdot_3, 1)$ ist ein Monoid.

Mit den definierten Begriffen lassen sich Halbgruppen, Monoide und Gruppen einführen mit

Definition A.6: Halbgruppe, Monoid, Gruppe

Eine Halbgruppe ist eine Menge $G \neq \emptyset$ zusammen mit einer inneren zweistelligen Verknüpfung $*$ auf G, die folgende Eigenschaften hat:

(G1) $*$ ist assoziativ.

Ein Monoid ist eine Halbgruppe, in der zusätzlich gilt:

(G2) Es gibt ein neutrales Element $e \in G$.

Eine Gruppe ist ein Monoid, in dem darüber hinaus gilt:

(G3) Zu jedem Element $a \in G$ gibt es ein inverses Element $b \in G$.

Eine Gruppe heißt kommutativ bzw. abelsch, wenn erfüllt ist:

(G4) $*$ ist kommutativ.

Bemerkung.
(1) Als Schreibweise für Monoide und Gruppen verwenden wir $(G, *, e_*)$ entsprechend der Schreibweise für algebraische Strukturen.
(2) Das inverse Element bezeichnen wir mit a'.
(3) Ein Element $a \in H$ eines Monoids $(H, *, e)$ wird Einheit genannt, wenn H ein zu a inverses Element a' enthält. Die Menge der Einheiten eines Monoids $E(H)$ bilden zusammen mit der (eingeschränkten) Verknüpfung $*$ von $(H, *, e_*)$ eine Gruppe $(E(H), *, e_*)$. Da das neutrale Element zu sich selber invers ist, kann $E(H)$ nie die leere Menge sein.

Beispiel A.7

$(\mathbb{N}, +)$ mit der Addition als Verknüpfung ist eine Halbgruppe, aber kein Monoid. $(\mathbb{Z}, \cdot, 1)$ mit der Multiplikation als Verknüpfung ist ein Monoid, aber keine Gruppe. $(\mathbb{Z}, +, 0)$ ist eine kommutative Gruppe.

Beispiel A.8

$(\{0, 1, 2\}, \cdot_3, 1)$ ist ein Monoid, Weiter ist $(\{0, 1, 2\}, +_3, 0)$ eine Gruppe. $(E(\{0, 1, 2\}), \cdot_3, 1)$ mit $E(\{0, 1, 2\}) = \{1, 2\}$ ist auch eine Gruppe.

Sei $(G; \circ, e_\circ)$ eine Gruppe und $a, b, c \in G$. Aus $c \circ a = c \circ b$ folgt durch Verknüpfung mit c' wegen $c' \circ c \circ a = a$ und $c' \circ c \circ b = b$, dass $a = b$ sein muss.

A.1.2. Unterhalbgruppen, Untermonoide und Untergruppen

Mit Teilmengen einer Halbgruppen-/Monoid- oder Gruppenmenge lassen sich Unterstrukturen aufbauen. Dies legen wir fest mit

Definition A.9: Unterhalbgruppe, Untermonoid, Untergruppe

Eine Unterhalbgruppe einer algebraischen Struktur $(M, *)$ ist eine Struktur bestehend aus einer Menge $\emptyset \neq U \subset M$ zusammen mit der auf U eingeschränkten inneren zweistelligen Verknüpfung $*$ mit folgenden Eigenschaften:

(U1) $(U, *)$ ist eine Halbgruppe, d.h. für alle $a, b \in U$ ist $a * b \in U$.

Ein Untermonoid ist eine Unterhalbgruppe, in der zusätzlich gilt:

(U2) $(U, *, e)$ ist ein Monoid, d.h. es gibt ein neutrales Element $e \in U$.

Eine Untergruppe ist ein Untermonoid, in dem darüber hinaus gilt:

(U3) $(U, *, e)$ ist eine Gruppe, d.h. zu jedem Element $a \in U$ gibt es ein inverses Element $a' \in U$.

Eine Untergruppe heißt kommutativ bzw. abelsch, wenn erfüllt ist:

(U4) $*$ ist kommutativ.

Bemerkung.
(1) Jede Unterhalbgruppe ist selbst eine Halbgruppe. Jedes Untermonoid ist ein Monoid. Jede Untergruppe ist eine Gruppe.
(2) Eine wichtige Anwendung ist, dass es Untermonoide gibt, die Gruppen sind, wie bei $(E(H), *, e_*)$ angesprochen.

Satz und Definition A.10: Verknüpfung mit mehr als drei Elementen

Sei $(H, \circ)$ eine Halbgruppe. Für $a_1, a_2, \ldots, a_n \in H$ $(n \geq 3)$ sei die Verknüpfung $a_1 \circ a_2 \circ ... \circ a_n$ induktiv

$$a_1 \circ a_2 \circ ... \circ a_n := \bigcirc_{i=1}^{n} a_i := \left(\bigcirc_{i=1}^{n-1} a_i \right) \circ a_n$$

definiert. Dann gilt für jedes m mit $1 \leq m < n$:

$$\bigcirc_{i=1}^{n} a_i = \bigcirc_{i=1}^{m} a_i \circ \bigcirc_{i=m+1}^{n} a_i$$

□

Beweis.

Die Aussage folgt unmittelbar aus dem Assoziativgesetz der Gruppenverknüpfung.

□

Sind alle Argumente der Verknüpfung gleich (etwa $a \in M$), so erhalten wir

$$\bigcirc_{i=1}^{n} a = a \circ \ldots \circ a.$$

Für eine multiplikative Schreibweise der Verknüpfung erhalten wir die bekannten Potenzen a^n, bei einer additiven Schreibweise $n \cdot a$. Die Anwendung davon wird z.B. in Kapitel 3 weiter vertieft. Wir legen das Potenzieren eines Elements einer Gruppe fest in

Satz und Definition A.11

Es sei $(G, \circ, e_\circ)$ eine Gruppe. Sei $a' \in G$ das zu $a \in G$ inverse Element. Für a sowie

$n \in \mathbb{Z}$ definieren wir

$$\bigcirc^{n} a := \begin{cases} \underbrace{a \circ \ldots \circ a}_{n-\text{mal}} & , n > 0, \\ e_* & , n = 0, \\ \underbrace{a' \circ \ldots \circ a'}_{|n|-\text{mal}} & , n < 0. \end{cases}$$

Insbesondere gilt: $\left(\bigcirc^{n} a\right)' = \bigcirc^{n} a'$.

Beweis.

$(a \circ \ldots \circ a) \circ (a' \circ \ldots \circ a') = a \circ \ldots \circ (a \circ a') \circ \ldots \circ a' = \ldots = e_\circ.$

□

Bemerkung.
Für eine multiplikative Gruppe $(G, \odot, e_\odot)$ schreiben wir (siehe [8])

$$a^n := \bigodot^{n} a = \begin{cases} \underbrace{a \odot \ldots \odot a}_{n-\text{mal}} & , n > 0, \\ e_\odot & , n = 0, \\ \underbrace{a^{-1} \odot \ldots \odot a^{-1}}_{|n|-\text{mal}} & , n < 0. \end{cases},$$

für eine additive Gruppe $(G, \oplus, e_\oplus)$ entsprechend

$$n \cdot a := \bigoplus^{n} a = \begin{cases} \underbrace{a \oplus \ldots \oplus a}_{n-\text{mal}} & , n > 0, \\ e_\oplus & , n = 0, \\ \underbrace{(-a) \oplus \ldots \oplus (-a)}_{|n|-\text{mal}} & , n < 0. \end{cases}.$$

Mit der Anzahl der Elemente einer Gruppe können Aussagen über die Anzahl von Elementen in Untergruppen getätigt werden. Dazu betrachten wir

Definition A.12: Gruppen- und Elementordnung

Sei $(G, \circ, e_\circ)$ eine beliebige Gruppe.

(a) Die Anzahl der Elemente in G heißt Ordnung der Gruppe.

(b) Die Ordnung eines Elementes $a \in G$ ist die kleinste natürliche Zahl $n \in \mathbb{N}$, für die $\bigcirc^n a$ als Ergebnis das neutrale Element $e_\circ$ liefert,

$$\text{ord}_G(a) := \min\{n \in \mathbb{N};\ \bigcirc^{n} a = e_\circ\}.$$

Existiert kein solches n, so setzen wir $\text{ord}_G(a) = \infty$.

Bemerkung.
Für eine multiplikative Gruppe $(G, \odot, e_\odot)$ schreiben wir

$$\text{ord}_G(a) = \min\{n \in \mathbb{N};\ a^n = e_\odot\},$$

für eine additive Gruppe $(G, \oplus, e_\oplus)$ entsprechend

$$\text{ord}_G(a) = \min\{n \in \mathbb{N};\ n \cdot a = e_\oplus\}.$$

Satz A.13

Sei $(G, \circ, e_\circ)$ eine endliche Gruppe und $(H, \circ, e_\circ)$ eine Untergruppe. Die Anzahl der Elemente von H teilt die Ordnung von $(G, \circ, e_\circ)$.

Beweis.

Sei $(H, \circ, e_\circ)$ eine Untergruppe von $(G, \circ, e_\circ)$ und $a, b \in G$. a sei äquivalent zu b, $a \sim_H b$, wenn $a \circ b' \in H$ ist. Dies ist eine Äquivalenzrelation. Denn es ist $a \sim_H a$, da $a \circ a' = e_\circ \in H$ (reflexiv). Es sei $a \sim_H b$, $a \circ b' \in H$. damit muss auch $(a \circ b')' = b \circ a' \in H$ sein und somit $b \sim_H a$ (symmetrisch). Zuletzt sind für ein $c \in G$ mit $a \sim_H b$ und $b \sim_H c$ zunächst $a \circ b' \in H$ und $b \circ c' \in H$. Damit muss auch $c \circ b' \in H$ und so $a \circ b' \circ b \circ c' = a \circ c' \in H$, $a \sim_H c$ (transitiv).

Es ist $[a] = \{b \in G;\ a \sim_H b\} = \{b \in G;\ a \circ b' \in H\}$ die Äquivalenzklasse von $a \in G$. Da a nicht notwendigerweise in H sein muss, lassen sich die Elemente von $[a]$ durch $b = h \circ a$ mit $h \in H$ beschreiben, denn $a \circ b' = a \circ a' \circ h' = h' \in H$. Also schreiben wir $[a] = \{h \circ a;\ h \in H\}$.

Für Elemente $a, b \in G$ definieren wir $f_{a,b} : [a] \to [b], h \circ a \mapsto h \circ b$. Diese Abbildung ist bijektiv. Ist $u \in [b]$, dann gibt es ein $h_1 \in H$ mit $u = h_1 \circ b$. Dazu gibt es $w = h_1 \circ a \in [a]$ mit $f_{a,b}(w) = u$, Die Abbildung ist surjektiv. Ist ferner $v \in [b]$, dann gibt es $h_2 \in H$ mit $v = h_2 \circ b$ und $x = h_2 \circ a$, so dass $f_{a,b}(x) = v$. Gilt nun $f_{a,b}(w) = f_{a,b}(x)$, so ist $v = w$ und damit $h_1 \circ b = h_2 \circ b$, also $h_1 = h_2$, $w = h_1 \circ a = h_2 \circ a = x$ (injektiv).

Da $f_{a,b}$ bijektiv ist, besitzen $[a]$ und $[b]$ gleich viele Elemente. Wegen $[e_\circ] = \{h \circ e_\circ;\ h \in H\} = H$ ist die Anzahl der Elemente in den Äquivalenzklassen gleich $|H|$. Da G eine disjunkte Vereinigung der Äquivalenzklassen zu $\sim_H$ ist, muss $|G|$ ein Vielfaches von $|H|$ sein.

□

Bemerkung.
Damit teilt auch die Ordnung eines Elements (denn es erzeugt eine Untergruppe (siehe Bemerkung (2) zu Definition 3.9) die Ordnung der Gruppe.

Den folgenden Satz von Lagrange[1] benötigen wir für das Potenzieren in der multiplikativen Einheitengruppe von $\mathbb{Z}_n$, $(\mathbb{Z}_n^*, \cdot_n, 1)$.

[1] J.-L. Lagrange, 1736-1813

Satz A.14: Satz von Lagrange

Sei $(G, \circ, e_\circ)$ eine endliche Gruppe und $a \in G$. Dann ist $a^{|G|} = e$.

Beweis.

Es sei $\text{ord}_G(a) = n$, d.h. $a^n = e_\circ$. Mit $n \mid |G|$ gemäß der Bemerkung zu Satz A.13 ist $|G| = q \cdot n$ für ein $q \in \mathbb{Z}$. Dann folgt $a^{|G|} = a^{q \cdot n} = (a^n)^q = e_\circ^q = e_\circ$.

□

Beispiel A.15

Wir betrachten die Gruppe $G = (\{1, 2\}, \cdot_3, 1)$. Dann ist $|\{1, 2\}| = 2$ und

$$\begin{aligned} &\text{ord}_G(1) = 1, \quad \text{wegen } 1^1 = 1,\ 1^2 = 1 \cdot 1 = 1, \ldots \\ &\text{ord}_G(2) = 2, \quad \text{wegen } 2^1 = 2,\ 2^2 = 2 \cdot 2 = 1,\ 2^3 = (2 \cdot 2) \cdot 2 = 2, \ldots \end{aligned}$$

wie wir aus der Gruppentafel aus Beispiel A.5 entnehmen können.

A.1.3. Ringe

Wir erweitern nun unsere Betrachtungen auf algebraische Strukturen mit zwei inneren zweistelligen Verknüpfungen.

Definition A.16: Ring

Sei R eine Menge mit den inneren Verknüpfungen

$$\begin{aligned} \oplus: &\quad R \times R \to R \text{ (Addition)}, \\ \odot: &\quad R \times R \to R \text{ (Multiplikation)}, \end{aligned}$$

und dem ausgezeichenten Element $0 \in R$. Es heißt $(R, \oplus, \odot, 0)$ ein Ring, wenn für alle $a, b, c \in R$ gilt:

(R1) $a \oplus b = b \oplus a$ (Kommutativität der Addition),

(R2) $a \oplus (b \oplus c) = (a \oplus b) \oplus c$ (Assoziativität der Addition),

(R3) $\exists 0 \in R : \forall a \in R : 0 \oplus a = a$ (Nullelement),

(R4) $\forall a \in R : \exists - a \in R : a \oplus (-a) = 0$ (inverses Element der Addition),

(R6) $a \odot (b \odot c) = (a \odot b) \odot c$ (Assoziativität der Multiplikation),

(R9) $a \odot (b \oplus c) = (a \odot b) \oplus (a \odot c)$ und $(a \oplus b) \odot c = (a \odot c) \oplus (b \odot c)$ (Distributivität).

Bemerkung.
(1) Gibt es ein vom Nullelement verschiedenes neutrales Element der Multiplikation (Einselement), sprechen wir von einem Ring mit Eins.
(2) Der Ring heißt kommutativ, wenn zudem $\odot$ kommutativ ist.

Ein wichtiges Beispiel für einen Ring ist der Polynomring, den wir in Kapitel 4 benötigen werden. Zu dessen Vorbereitung schreiben wir

Definition A.17: Polynom

Sei $(M, \oplus, \odot, 0, 1)$ ein kommutativer Ring mit Einselement. Eine Abbildung $f : \mathbb{N}_0 \to M,\ n \mapsto a_n$ mit der Eigenschaft $f(n) = a_n = 0$ für fast alle (d.h. alle bis auf endlich viele) $n \in \mathbb{N}_0$ heißt Polynom über $(M, \oplus, \odot, 0, 1)$. Sei n der größte Index mit $a_n \neq 0$. Dann besitzt das Polynom den Grad n, $n = \deg(f)$. Gibt es keinen solchen Index, ist das Polynom vom Grad $n = \infty$.

Sei X eine beliebige Menge. Lässt sich eine Abbildung

$$* : M \times X \to X,\ (m, x) \mapsto m * x \tag{A.1}$$

definieren, können wir ein Polynom an $x \in X$ auswerten.

Definition A.18

Sei f ein Polynom über $(M, \oplus, \odot, 0, 1)$ vom Grad n und X eine beliebige Menge. Die Menge $M[X]$ bezeichnet die Menge aller Polynome f über $(M, \oplus, \odot, 0, 1)$, die mit (A.1) durch die Abbildung

$$g_f : X \to X, x \mapsto g_f(x) := \sum_{i=0}^{n} f(i) * x^i = \sum_{i=0}^{n} a_i * x^i$$

an x ausgewertet werden. $g_f(x)$ heißt der Wert von f an der Stelle x. Die a_i heißen die Koeffizienten des Polynoms.

Bemerkung.
(1) Oft schreiben wir anstelle von $g_f(x)$ kürzer $f(x)$, wenn M und X festgelegt sind.
(2) Ist $g_f(x) = 0$ für ein $x \in X$, so heißt x Nullstelle des Polynoms f.
(3) Mit $M_n[X]$ bezeichnen wir die Menge aller Polynome vom Grad kleiner gleich n.

Auf $M[X]$ sind zwei Verknüpfungen definiert:

$$+ : M[X] \times M[X] \to M[X], \qquad (f, h) \mapsto (f + h)(i) := f(i) \oplus h(i)$$
$$\cdot : M[X] \times M[X] \to M[X], \qquad (f, h) \mapsto (f \cdot h)(i) := \sum_{j=0}^{i} f(j) \odot h(n - i)$$

Bemerkung.

(1) Mit $f, h \in M[X]$, $f(i) = a_i$, $h(i) = b_i$, $n = \deg(f) + \deg(g)$ und $x \in X$ ist

$$g_{f+h}(x) = \sum_{i=0}^{n} (f+h)(i) * x^i = \sum_{i=0}^{n} (a_i \oplus b_i) * x^i,$$

$$g_{f \cdot h}(x) = \sum_{i=0}^{n} (f \cdot h)(i) * x^i = \sum_{i=0}^{n} \left(\sum_{j=0}^{i} (a_j \odot b_{i-j}) \right) * x^i.$$

(2) $(M[X], +, \cdot, o, e)$ mit den beiden eben definierten Verknüpfungen ist ein kommutativer Ring, der so genannte Polynomring mit dem Einselement (d.h. dem neutralen Element über $\cdot$)

$$e : \mathbb{N}_0 \to M,\ e(0) = 1,\ e(n) = 0 \quad \forall n > 0,$$

und dem Nullelement (d.h. dem neutralen Element über $+$)

$$o : \mathbb{N}_0 \to M,\ e(n) = 0 \quad \forall n \geq 0.$$

Beispiel A.19

Es sei $f \in \mathbb{Z}$, $X = \mathbb{Z}$ und $g_f(x) = x^2 + 2x + 1$. Dann ist $g_f(-1) = 0$, -1 ist eine Nullstelle des Polynoms f.

Satz A.20

Seien $f \in M_n[X]$ und $h \in \mathbb{K}_m[X]$ zwei Polynome mit Grad n bzw. m. Dann ist der Grad von $f \cdot h$ gleich $n + m$.

Beweis.

Seien $g_f(x) = \sum_{i=0}^{n} a_i \cdot x^i$ und $g_h(x) = \sum_{i=0}^{m} b_i \cdot x^i$. Weiter seien $a_n \neq 0$ und $b_m \neq 0$. Dann ist $g_{f \cdot h}(x) = \sum_{i=0}^{m+n} c_i \cdot x^i$ mit $c_{m+n} = a_0 \cdot b_{m+n} + \ldots + a_n \cdot b_m + \ldots + a_{m+n} \cdot b_0 \neq 0$.

□

Induktiv lässt sich zu einem kommutativen Ring ein Polynom in mehreren Veränderlichen festlegen.

Definition A.21: Polynom mehrerer Veränderlicher

Sei $(M, \oplus, \odot, 0, 1)$ ein kommutativer Ring und $n \in \mathbb{N}_0$. Betrachte ein Polynom

$$f : \mathbb{N}_0 \to M,\ i \mapsto f(i) = a_{i_{n+1}},$$

wobei i_{n+1} die Zahl i in $n+1$-adischer Darstellung mit k Ziffern (evtl. führende Nullen) ist. Dann heißt für ein $k \in \mathbb{N}$ die Struktur $(M[X^k], +, \cdot, o, \tilde{e})$ mit $g_f : X^k \to X$,

$$(x_1, \ldots, x_k) \mapsto g_f(x_1, \ldots, x_k) := \sum_{i_1, i_2, \ldots, i_k \geq 0} a_{i_1 i_2 \ldots i_k} x_1^{i_1} x_2^{i_2} \ldots x_k^{i_k}, \quad a_{i_1 i_2 \ldots i_k} \in M,$$

wobei fast alle $a_{i_1 i_2 \ldots i_k} = 0$ sind, der Polynomring in der unbestimmten Menge X über M. g_f ist eine Linearkombination über M der so genannten *Monome* $x_1^{i_1} x_2^{i_2} \ldots x_k^{i_k}$. Der Grad eines Monoms ist

$$\deg(x_1^{i_1} x_2^{i_2} \ldots x_k^{i_k}) := i_1 + \ldots + i_k.$$

Der Grad eines solchen Polynoms ist $\deg f := \max\{i_1 + \ldots + i_k;\ a_{i_1 \ldots i_k} \neq 0\}$.

Bemerkung.
$\tilde{e}$ ist der Koeffizient, dessen Indizes $i_1, \ldots, i_k$ alle den Wert 1 haben.

Beispiel A.22

Wir betrachten $(\mathbb{R}, +, \cdot, 0, 1)$. Seien $a_{021} := 1,\ a_{111} := 1,\ a_{012} := 3,\ a_{300} := -1,\ a_{201} := -2,\ a_{102} := -4,\ a_{003} := -6$ und sonst alle $a_{ijk} = 0$. Dann ergibt sich das Polynom $f \in \mathbb{R}[X_1, X_2, X_3]$ vom Grad 3 mit

$$f(X_1, X_2, X_3) = X_2^2 X_3 + X_1 X_2 X_3 + 3X_2 X_3^2 - X_1^3 - 2X_1^2 X_3 - 4X_1 X_3^2 - 6X_3^3.$$

A.1.4. Körper

Eine Erweiterung zu Gruppen bzw. Ringen ist ein Körper. Dabei werden zu einer Menge zwei innere zweistellige Verknüpfungen betrachtet, welche die Eigenschaften eines Rings erweitern. Dies legen wir fest in

Definition A.23: Körper

Sei $\mathbb{K}$ eine Menge mit den inneren Verknüpfungen

$$\begin{aligned} \oplus: &\quad \mathbb{K} \times \mathbb{K} \to \mathbb{K} \text{ (Addition)}, \\ \odot: &\quad \mathbb{K} \times \mathbb{K} \to \mathbb{K} \text{ (Multiplikation)}, \end{aligned}$$

und den ausgezeichneten Elementen $0, 1 \in \mathbb{K}$. Es heißt $(\mathbb{K}, \oplus, \odot, 0, 1)$ ein Körper, wenn für alle $a, b, c \in \mathbb{K}$ gilt:

(K1) $a \oplus b = b \oplus a$ (Kommutativität der Addition),

(K2) $a \oplus (b \oplus c) = (a \oplus b) \oplus c$ (Assoziativität der Addition),

(K3) $\exists 0 \in \mathbb{K} : \forall a \in \mathbb{K} : 0 \oplus a = a$ (Nullelement),

(K4) $\forall a \in \mathbb{K} : \exists -a \in \mathbb{K} : a \oplus (-a) = 0$ (inverses Element der Addition),

(K5) $a \odot b = b \odot a$ (Kommutatitvität der Multiplikation),

(K6) $a \odot (b \odot c) = (a \odot b) \odot c$ (Assoziativität der Multiplikation),

(K7) $\exists 1 \in \mathbb{K} : 0 \neq 1 \wedge \forall a \in \mathbb{K} : 1 \odot a = a$ (Einselement),

(K8) $\forall a \in \mathbb{K} \setminus \{0\} : \exists a^{-1} \in \mathbb{K} : a \odot a^{-1} = 1$ (inverses Element der Multiplikation),

(K9) $a \odot (b \oplus c) = (a \odot b) \oplus (a \odot c)$ und $(a \oplus b) \odot c = (a \odot c) \oplus (b \odot c)$ (Distributivität).

Bemerkung.
(1) Ein Ring ist eine etwas allgemeinere Struktur als ein Körper. Hierbei sind die Eigenschaften (K5), (K7) und (K8) eines Körpers nicht erfüllt. Insbesondere existiert bei einem Ring kein Einselement. Jeder Körper ist auch ein Ring.
(2) Aus der Definition eines Körpers ist wegen der Bedingung $0 \neq 1$ zu erkennen, dass ein Körper wenigstens zwei Elemente besitzt, das Null- und Einselement.

Wir zeigen noch, dass die speziellen Elemente eindeutig bestimmt sind.

Satz A.24

Sei $(\mathbb{K}, \oplus, \odot, 0, 1)$ ein Körper. Dann sind das Nullelement, Einselement und die inversen Elemente eindeutig bestimmt.

Beweis.

Sei $a \in \mathbb{K}$.

Angenommen $0'$ sei ein weiteres Nullelement. Dann gilt $0 \overset{(K3)}{=} 0' \oplus 0 \overset{(K1)}{=} 0 \oplus 0' \overset{(K3)}{=} 0'$.

Angenommen $1'$ sei ein weiteres Einselement. Dann gilt $1 \overset{(K7)}{=} 1' \odot 1 \overset{(K5)}{=} 1 \odot 1' \overset{(K7)}{=} 1'$.

Angenommen $-a$ und $-a'$ sind zwei verschiedene inverse Elemente der Addition zu a. Dann gilt $-a \overset{(K3)}{=} 0 \oplus -a \overset{(K4)}{=} ((-a') \oplus a) \oplus (-a) \overset{(K2)}{=} (-a') \oplus (a \oplus (-a)) \overset{(K4)}{=} (-a') \oplus 0 \overset{(K1)}{=} 0 \oplus (-a') \overset{(K4)}{=} -a'$.

Angenommen a^{-1} und b^{-1} sind zwei verschiedene inverse Elemente der Multiplikation zu $a \neq 0$. Dann gilt $a^{-1} \overset{(K7)}{=} 1 \odot a^{-1} \overset{(K8)}{=} (a \odot b^{-1}) \odot a^{-1} \overset{(K5)}{=} (b^{-1} \odot a) \odot a^{-1} \overset{(K6)}{=} b^{-1} \odot (a \odot a^{-1}) \overset{(K8)}{=} b^{-1} \odot 1 \overset{(K5)}{=} 1 \odot b^{-1} \overset{(K7)}{=} b^{-1}$.

□

Besitzt ein Körper endlich viele Elemente, d.h. $|\mathbb{K}| < \infty$, so sprechen wir von einem endlichen Körper. Ein besonders wichtiger endlicher Körper ist der Primkörper.

Satz und Definition A.25: Primkörper

Sei $n \in \mathbb{N}$ eine Primzahl. Dann ist $(\mathbb{Z}_n, +_n, \cdot_n, 0, 1)$ ein Körper, der Primkörper.

Beweis.

Sei $n \in \mathbb{N}$ eine Primzahl. Die Eigenschaften (K1)-(K4) sind klar: (K1) und (K2) gelten, weil am Ende stets eine modulo-Operation steht. (K3) folgt aus den Rechenregeln für natürliche Zahlen und (K4) gilt wegen $a +_n b = 0 \Leftrightarrow b = n - a$. (K5) und (K6) folgen wiederum, weil am Ende die modulo-Operation steht. (K7) und (K9) gelten wegen der Rechenregeln für natürliche Zahlen.

Es bleibt (K8) zu zeigen. Sei $0 < a < n$. Gesucht ist eine natürliche Zahl b mit $a \cdot_n b = 1$. Betrachten wir die Produkte $0 \cdot_n a, 1 \cdot_n a, 2 \cdot_n a, \ldots, (n-1) \cdot_n a$. Dann sind diese paarweise verschieden. Denn angenommen, es wäre $c \cdot_n a = d \cdot_n a$ mit $c \neq d$. Dann wäre $c \cdot a$ mod n=$d \cdot a$ mod n. Also wäre $c \cdot a - d \cdot a = (c-d) \cdot a$ durch n teilbar, etwa $n \cdot z = (c-d) \cdot a$. Da $a < n$, teilt n nicht a. Angenommen n teilt auch $c - d$ nicht. Dann lässt sich $c - d$ schreiben als $c - d = n \cdot q + r$ mit $0 < r < n$. Damit folgt $n \cdot z = (n \cdot q + r) \cdot a = (n \cdot q \cdot a) + r \cdot a$ und so $n \cdot (z - q \cdot a) = r \cdot a$. Also teilt n das Produkt $r \cdot a$. Aufgrund der Annahme gilt nun sogar $1 < a, r < n$. Weil n eine Primzahl ist, ergibt sich ein Widerspruch zur Annahme. Also teilt n die Differenz $c - d$. Es ist $-(n-1) \leq c - d \leq n - 1$. Die einzige Zahl, die beide Kriterien erfüllt ist 0 und damit ist $c = d$, was einen Widerspruch darstellt. Wir erhalten also n paarweise verschiedene Produkte. In $\mathbb{Z}_n \setminus \{0\}$ haben wir dann $n - 1$ paarweise von 0 verschiedene Werte des Produkts ($0 = 0 \cdot_n a$). Wir haben genau $n - 1$ verschiedene Elemente in $\mathbb{Z}_n \setminus \{0\}$ zur Verfügung. Damit muss es ein Produkt $a \cdot_n b$ geben, das den Wert 1 annimmt und dies ist das gesuchte inverse Element. $(\mathbb{Z}_n, +_n, \cdot_n, 0, 1)$ ist ein Körper.

□

Bemerkung.
Die Anzahl der Elemente des Primkörpers ist $|\mathbb{Z}_p| = p$. Es gibt auch endliche Körper mit n Elementen, wobei n keine Primzahl ist. Dies ist immer dann möglich, wenn n eine Primzahlpotenz ist[2]. Es lässt sich also auch ein Körper aus vier Elementen konstruieren. Ist p eine Primzahl, so sagen wir, dass der Körper $(\mathbb{Z}_p, +_p, \cdot_p, 0, 1)$ die *Charakteristik* $c(\mathbb{Z}_p) = p$ hat.

Wir betrachten noch Polynome über einem Körper $(\mathbb{K}, \oplus, \odot, 0, 1)$ und überlegen uns einige Eigenschaften.

Satz A.26

Seien $f, h \in \mathbb{K}[X]$ und $h \neq 0$. Dann gibt es eindeutig bestimmte Polynome $q, r \in \mathbb{K}[X]$ mit $f = q \cdot h + r$. Weiter ist $r = 0$ oder $\deg(r) < \deg(h)$.

[2] vgl. [5]

Beweis.

[3]

□

Satz A.27

Ist $f \in \mathbb{K}[X]$ mit $f \neq 0$ und $x \in X$ eine Nullstelle von f, dann ist $g_f(x) = (x-a) \cdot g_h(x)$ mit $h \in \mathbb{K}[X]$ und $\deg(h) = \deg(f) - 1$.

Beweis.

[3]

□

Satz A.28

Ein Polynom $f \in \mathbb{K}[X]$ mit $f \neq 0$ hat höchstens $\deg(f)$ viele Nullstellen.

Beweis.

[3] S.51

□

Beispiel A.29

Wir wollen die Gleichung $x^2 = 1$ mod n in einem endlichen Körper $(\mathbb{Z}_n, +_n, \cdot, 0, 1)$ lösen. Es handelt sich um das Polynom $g_f(x) = x^2 - 1$ in $(\mathbb{Z}_n, +_n, \cdot, 0, 1)$ mit $X = \mathbb{Z}_n$, welches Grad 2 hat. Mit Satz A.28 wissen wir, dass f höchstens zwei Nullstellen in $\mathbb{Z}_n$ besitzt. Es ist $1^2 - 1 = 0$ und $((n-1)^2 - 1)$ mod $n = (n^2 - 2 \cdot n + 1 - 1)$ mod $n = 0$. Damit besitzt f die beiden Nullstellen 1 und $n-1$. Im Fall $n = 2$ fallen diese beiden Zahlen zusammen.

Satz A.30

Ist $(K, \oplus, \odot, e_\oplus, e_\odot)$ ein endlicher Körper mit q Elementen, so ist die Einheitengruppe $(K^*, \odot, e_\odot)$ zyklisch von der Ordnung $q-1$. Sie hat genau $\phi(q-1)$ Erzeuger.

Beweis.

[3] S.55

□

A.2. Rechnen in $\mathbb{Z}$ bzw. $\mathbb{Z}_n$

Bei den folgenden Ausführungen stützen wir uns auf Aussagen und Definitionen der Theorie über algebraische Strukturen in Anhang A.1. Wir überlegen uns einige Eigenschaften der Kongruenzrelation $\equiv_n$[3].

Satz A.31

Für $a, b \in \mathbb{Z}$ mit $a = q_1 n + r_1, b = q_2 n + r_2$ gilt $a \equiv_n b \Leftrightarrow n \mid a - b$.

Beweis.

„ $\Rightarrow$ “ : $a \equiv_n b \Rightarrow r_1 = r_2 \Rightarrow a = q_1 n + r,\ b = q_2 n + r \Rightarrow a - b = q_1 n + r - (q_2 n + r) = (q_1 - q_2)n \Rightarrow n \mid a - b$.
„ $\Leftarrow$ “ : $n \mid a - b \Rightarrow a - b = qn$ für ein $q \in \mathbb{Z} \Rightarrow q_1 n + r_1 - (q_2 n + r_2) = (q_1 - q_2)n + (r_1 - r_2) \overset{!}{=} qn \Rightarrow -n < r_1 - r_2 < n$. Da $r_1 - r_2 = q_3 n$ für ein $q_3 \in \mathbb{Z}$ sein muss, muss $q_3 = 0 = r_1 - r_2$ gelten. $\Rightarrow r_1 = r_2 \Rightarrow a \equiv_n b$.

□

In $\mathbb{Z}$ kongruente Elemente bzgl. n lassen dieselben Reste.

Satz A.32

Für $a, b \in \mathbb{Z}$ mit $a = q_1 n + r_1, b = q_2 n + r_2$ gilt $a \equiv_n b \Leftrightarrow a \bmod n = b \bmod n$.

Beweis.

$a \equiv_n b \Leftrightarrow r_1 = r_2 \Leftrightarrow a \bmod n = b \bmod n$.

□

Satz A.33

[3] vgl. Definition 2.12

$\equiv_n$ ist eine Äquivalenzrelation auf $\mathbb{Z}$.

Beweis.

Reflexivität: $a \equiv_n a$, denn a mod $n = a$ mod n.
Symmetrie: $a \equiv_n b \Leftrightarrow a$ mod $n = b$ mod $n = a$ mod $n \Leftrightarrow b \equiv_n a$.
Transitivität: $a \equiv_n b \wedge b \equiv_n c \Rightarrow a$ mod $n = b$ mod $n = c$ mod $n \Rightarrow a \equiv_n c$.

□

Die Mengen $[r]_n := \{a \in \mathbb{Z} : a \equiv_n r\}$, sind die Restklassen modulo n. r ist Repräsentant der Klasse $[r]_n$, aber auch jedes $a \in [r]_n$ ist ein Repräsentant von $[r]_n$. Mit $a = q \cdot n + r$ sei $[a]_n := [r]_n$.

Satz A.34

Es gibt genau n verschiedene Äquivalenzklassen für $\equiv_n$.

Beweis.

Für jedes $r \in \{0, \ldots, n-1\}$ ist $[r]_n$ eine Äquivalenzklasse. Denn seien $r_1, r_2 \in \{0, \ldots, n-1\}$ mit $r_1 \neq r_2$. Angenommen, $[r_1]_n = [r_2]_n$. Dann ist $r_1 \equiv_n r_2$ und so r_1 mod $n = r_2$ mod n, was ein Widerspruch zu $r_1 \neq r_2$ ist. Also gibt es wenigstens n Äquivalenzklassen.
Angenommen, es gibt ein $b \in \mathbb{Z}$ mit $b \notin [r]_n$ für alle $r \in \{0, \ldots, n-1\}$. Dann folgt $b = q_2 n + r_2 \Rightarrow r_2 \notin \{0, \ldots, n-1\}$, was ein Widerspruch dazu ist, dass sämtliche Reste aus $\{0, \ldots, n-1\}$ stammen.

□

Wegen der eindeutigen Darstellung $a = q_1 n + r_1$ für jedes $a \in \mathbb{Z}$ gilt mit dem letzten Satz, dass $\mathbb{Z}$ eine disjunkte Vereinigung der Restklassen modulo n ist. Denn

$$\begin{aligned} \bigcup_{r=0}^{n-1} [r]_n &= \bigcup_{r=0}^{n-1} \{a \in \mathbb{Z};\ a \equiv_n r\} = \bigcup_{r=0}^{n-1} \{qn + r;\ q \in \mathbb{Z}\} \\ &= \{qn + r;\ q \in \mathbb{Z},\ r \in \{0, \ldots, n-1\}\} = \mathbb{Z}. \end{aligned}$$

Beispiel A.35

Für das Element $[0]_n$ gilt für alle $n \in \mathbb{N}$

$$[0]_n = \{a \in \mathbb{Z};\ a \equiv_n 0\} = \{a \in \mathbb{Z};\ n|a\} = \{z \cdot n;\ z \in \mathbb{Z}\} =: n\mathbb{Z} \subset \mathbb{Z}.$$

Für $z_1, z_2 \in \mathbb{Z}$ sind die zwei Elemente $n \cdot z_1, n \cdot z_2 \in n\mathbb{Z}$ und es gilt mit der Addition in $\mathbb{Z}$

$$\begin{aligned} n \cdot z_1 + n \cdot z_2 &= \underbrace{(z_1 + \ldots + z_1)}_{n-mal} + \underbrace{(z_2 + \ldots + z_2)}_{n-mal} \\ &= \underbrace{(z_1 + z_2) + \ldots + (z_1 + z_2)}_{n-mal} = n \cdot (z_1 + z_2) \in n\mathbb{Z}. \end{aligned}$$

Weiter ist $0 = n \cdot 0 \in n\mathbb{Z}$ und $n \cdot (-z) \in n\mathbb{Z}$ mit $n \cdot z + n \cdot (-z) = 0$ für alle $z \in \mathbb{Z}$. Damit ist $(n\mathbb{Z}, +, 0)$ eine Gruppe.

Wir betrachten nun die Menge aller Restklassen modulo n, bezeichnet mit

$$\mathbb{Z}/\equiv_n := \{[r]_n;\ r \in \{0, \ldots, n-1\}\}$$

und versehen sie mit den beiden inneren zweistelligen Verknüpfungen

$$\begin{aligned} \oplus &: \mathbb{Z}/\equiv_n \times \mathbb{Z}/\equiv_n \to \mathbb{Z}/\equiv_n, \qquad [a]_n \oplus [b]_n := [a+b]_n, \\ \odot &: \mathbb{Z}/\equiv_n \times \mathbb{Z}/\equiv_n \to \mathbb{Z}/\equiv_n, \qquad [a]_n \odot [b]_n := [a \cdot b]_n. \end{aligned}$$

Satz A.36

$(\mathbb{Z}/\equiv_n, \oplus, \odot, [0]_n, [1]_n)$ ist ein Ring.

Beweis.

Operationen abgeschlossen und wohldefiniert: Seien $a, b \in \mathbb{Z}$ mit $a = q_1 n + r_1$ und $b = q_2 n + r_2$. Dann sind $[a]_n = [r_1]_n$ und $[b]_n = [r_2]_n$ und somit

$$\begin{aligned} [a]_n \oplus [b]_n &= [r_1]_n \oplus [r_2]_n = [r_1 + r_2]_n \overset{r_1+r_2 = ln+r}{=} [r]_n \in \mathbb{Z}/\equiv_n, \\ [a]_n \odot [b]_n &= [r_1]_n \odot [r_2]_n = [r_1 \cdot r_2]_n \overset{r_1 \cdot r_2 = \tilde{l}n+\tilde{r}}{=} [\tilde{r}]_n \in \mathbb{Z}/\equiv_n . \end{aligned}$$

Assoziativität:

$$\begin{aligned} ([a]_n \oplus [b]_n) \oplus [c]_n &= [a+b]_n \oplus [c]_n = [(a+b)+c]_n \\ &= [a+(b+c)]_n = [a]_n \oplus [b+c]_n = [a]_n \oplus ([b]_n \oplus [c]_n), \\ ([a]_n \odot [b]_n) \odot [c]_n &= [a \cdot b]_n \odot [c]_n = [(a \cdot b) \cdot c]_n \\ &= [a \cdot (b \cdot c)]_n = [a]_n \odot [b \cdot c]_n = [a]_n \odot ([b]_n \odot [c]_n). \end{aligned}$$

Kommutativität der Addition:

$$[a]_n \oplus [b]_n = [a+b]_n = [b+a]_n = [b]_n + [a]_n.$$

Neutrale Elemente:

$$[0]_n \oplus [a]_n = [0+a]_n = [a]_n = [a+0]_n = [a]_n \oplus [0]_n,$$
$$[1]_n \odot [a]_n = [1 \cdot a]_n = [a]_n = [a \cdot 1]_n = [a]_n \odot [1]_n.$$

Inverses Element der Addition: Sei $-a := \tilde{q}n + \tilde{r}$. Es ist

$$[a]_n \oplus [\tilde{r}]_n = [a]_n \oplus [-a]_n = [a + (-a)]_n = [0]_n.$$

Distributivität:

$$\begin{aligned}[a]_n \odot ([b]_n \oplus [c]_n) &= [a]_n \odot [b+c]_n = [a \cdot (b+c)]_n \\ &= [a \cdot b + a \cdot c]_n = ([a]_n \odot [b]_n) \oplus ([a]_n \odot [c]_n), \\ ([a]_n \oplus [b]_n) \odot [c]_n &= [a+b]_n \odot [c]_n = [(a+b) \cdot c]_n \\ &= [a \cdot c + b \cdot c]_n = ([a]_n \odot [c]_n) \oplus ([b]_n \odot [c]_n).\end{aligned}$$

□

Betrachten wir nun die Menge $\mathbb{Z}_n := \{0, \ldots, n-1\}$ und definieren die beiden Abbildungen

$$+_n : \mathbb{Z}_n \times \mathbb{Z}_n \to \mathbb{Z}_n, \qquad a +_n b := (a+b) \bmod n,$$
$$\cdot_n : \mathbb{Z}_n \times \mathbb{Z}_n \to \mathbb{Z}_n, \qquad a \cdot_n b := (a \cdot b) \bmod n.$$

Satz A.37

$(\mathbb{Z}_n, +_n, \cdot_n, 0, 1)$ ist ein Ring.

Beweis.

Wir zeigen, dass die Abbildung

$$f : \mathbb{Z}/\equiv_n \to \mathbb{Z}_n,\ [r]_n \mapsto f([r]_n) := r \bmod n$$

bijektiv und strukturerhaltend bzgl. $+_n$ und $\cdot_n$ ist. Denn dann überträgt sich die Ringstruktur von $(\mathbb{Z}/\equiv_n, \oplus, \odot, [0]_n, [1]_n)$ auf $(\mathbb{Z}_n, +_n, \cdot_n, 0, 1)$.

f ist bijektiv: $|\mathbb{Z}/\equiv_n| = n = |\mathbb{Z}_n|$. Sei $r \in \mathbb{Z}_n$. Dann gibt es $[r]_n$ mit $f([r]_n) = r$ und $[r]_n \in \mathbb{Z}/\equiv_n$ (Surjektivität). Sei $[r_1]_n \neq [r_2]_n$ mit $r_1, r_2 \in \mathbb{Z}_n$. Betrachte $f([r_1]_n) = r_1$ und $f([r_2]_n) = r_2$. Da die Äquivalenzklassen disjunkte Mengen sind, folgt unmittelbar $r_1 \neq r_2$ (Injektivität).

Seien $a = q_1 n + r_1$ und $b = q_2 n + r_2$ mit $r := r_1 + r_2 \bmod n$ und $\tilde{r} := r_1 \cdot r_2 \bmod n$.

Dann gilt

$$\begin{aligned} f([a]_n \oplus [b]_n) &= f([a+b]_n) = f([r]_n) = r = r_1 + r_2 \bmod n \\ &= r_1 +_n r_2 = f([r_1]_n) +_n f([r_2]_n) \\ &= f([a]_n) +_n f([b]_n), \\ f([a]_n \odot [b]_n) &= f([a \cdot b]_n) = f([\tilde{r}]_n) = \tilde{r} = r_1 \cdot r_2 \bmod n \\ &= r_1 \cdot_n r_2 = f([r_1]_n) \cdot_n f([r_2]_n) \\ &= f([a]_n) \cdot_n f([b]_n), \\ f([0]_n) &= 0 \bmod n = 0, \\ f([1]_n) &= 1 \bmod n = 1. \end{aligned}$$

□

Sei für $a, b, c \in \mathbb{Z}$ und $r = a + b \bmod n$ die Addition (analog: Multipliaktion) modulo n definiert durch

$$a +_n b +_n c := (a +_n b) +_n c.$$

Dann gilt

$$\begin{aligned} a +_n b +_n c = (a +_n b) +_n c &= f([a+b]_n) +_n c \\ &= f([r]_n) +_n c = r +_n c \\ &= f([r+c]_n) = f([a+b+c]_n) \\ &= a + b + c \bmod n. \end{aligned}$$

Damit können Berechnungen in $\mathbb{Z}_n$ auf $\mathbb{Z}$ „ausgelagert“ werden bzw. umgekehrt.

Beispiel A.38

- $(17 + 16 + 10) \bmod 26 = (33 + 10) \bmod 26 = (7 + 10) \bmod 26 = 17$ bzw. $(17 + 16 + 10) \bmod 26 = 43 \bmod 26 = 17$.
- $2^{10} \bmod 3 = 2^{8+2} \bmod 3 = (2^2 \bmod 3)^4 \cdot 2^2 \bmod 3 = 1^4 \cdot 1 \bmod 3 = 1$ bzw. $2^{10} \bmod 3 = 1024 \bmod 3 = 1$.

Chinesischer Restsatz

Neben dem größten gemeinsamen Teiler wird manchmal das kleinste gemeinsame Vielfache ganzer zahlen benötigt.

Definition A.39: Kleinstes gemeinsames Vielfaches

Das kleinste gemeinsame Vielfache ganzer Zahlen $a_1, \ldots, a_m \in \mathbb{Z}$ ist die natürliche Zahl

$$\text{kgV}(a_1, \ldots, a_m) := \min\{n \in \langle a_1 \rangle \cap \ldots \cap \langle a_m \rangle;\ n > 0\}.$$

Für zwei Zahlen $a, b \in \mathbb{Z}$ sei $V := \langle a \rangle \cap \langle b \rangle$ die Menge aller Vielfachen von a und b. Wegen $a \cdot b \in V$ ist die Menge sicher nicht leer. Sei $v \in V$. Dann gilt zunächst $a, b \mid v$ und mit

$d =$ggT(a, b) auch $d \mid v$. Seien $a = q_1 \cdot d$, $b = q_2 \cdot d$ und $v = q_3 \cdot d$. Dann ist

$$\frac{a \cdot b}{d} = \frac{q_1 \cdot d \cdot q_2 \cdot d}{d} = q_1 \cdot q_2 \cdot d = q_2 \cdot a = q_1 \cdot b,$$

d.h. $a \mid \frac{a \cdot b}{d}$, $b \mid \frac{a \cdot b}{d}$ und damit $\frac{a \cdot b}{d} \in V$. Sei nun $v = u_1 \cdot a = u_2 \cdot b$. Dann gilt $v = u_1 \cdot q_1 \cdot d = u_2 \cdot q_2 \cdot d$. Daher muss $u_1 \cdot q_1 = u_2 \cdot q_2$ sein, etwa $u_1 = z \cdot q_2$ und $u_2 = z \cdot q_1$ für eine Zahl $z \in \mathbb{Z}$. Dann folgt

$$z \cdot \frac{a \cdot b}{d} = z \cdot q_1 \cdot q_2 \cdot d = u_1 \cdot q_1 \cdot d = u_1 \cdot a = v \Rightarrow \frac{a \cdot b}{d} \mid v.$$

Damit lässt sich das kleinste gemeinsame Vielfache zweier Zahlen $a, b \in \mathbb{Z}$ schreiben als

$$\text{kgV}(a, b) = \frac{|a \cdot b|}{\text{ggT}(a, b)}. \tag{A.2}$$

Wir betrachten nun so genannte simultane Kongruenzen der Form

$$x \equiv_{m_i} a_i, \quad i = 1, \ldots n. \tag{A.3}$$

Solche Kongruenzen sind nicht immer lösbar. Eine Voraussetzung ist, dass die Zahlen $m_1, \ldots, m_n \in \mathbb{N}$ paarweise teilerfremd sind, d.h. ggT$(m_i, m_j) = 1$ für $i \neq j$.

Satz A.40

Seien $m_1, \ldots, m_n \in \mathbb{N}$ paarweise teilerfremd. Seien weiter $m := m_1 \cdot \ldots \cdot m_n$ und $M_i := \prod_{j \neq i} m_i = m/m_i$. Dann gilt ggT$(m_i, M_i) = 1$.

Beweis.

Da die $m_1, \ldots, m_n$ paarweise teilerfremd sind, gibt es Zahlen $x_j, y_j \in \mathbb{Z}$, $j \neq i$, mit

$$1 = x_j \cdot m_i + y_j \cdot m_j \Leftrightarrow y_j \cdot m_j = 1 - x_j \cdot m_i.$$

Somit ergibt sich

$$\prod_{j \neq i} y_j \cdot m_j = M_i \cdot \prod_{j \neq i} y_j = \prod_{j \neq i} (1 - x_j \cdot m_i) = 1 + m_i \cdot f(m_i, x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n)$$

für eine nicht näher zu beschreibende Funktion f. Durch Umstellen erhalten wir

$$1 = \underbrace{-f(m_i, x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n)}_{\tilde{x}_i} \cdot m_i + \underbrace{\prod_{j \neq i} y_j}_{\tilde{y}_i} \cdot M_i$$

und damit ggT$(m_i, M_i) = 1$.

□

Satz A.41: Chinesischer Restsatz

Seien $m_1, \ldots, m_n \in \mathbb{N}$ paarweise teilerfremd, $m := m_1 \cdot \ldots \cdot m_n$ und $a_1, \ldots, a_n \in \mathbb{Z}$. Dann haben die simultanen Kongruenzen (A.3) eine eindeutige Lösung $x \in \mathbb{Z}_m$.

Beweis.

Existenz: Seien $M_i := m/m_i$ für alle $i = 1, \ldots, n$. Dann gilt gemäß Satz A.40 zunächst ggT$(m_i, M_i) = 1$. Mit dem erweiterten euklidischen Algorithmus lassen sich $\tilde{y}_i$ bestimmen mit

$$\tilde{y}_i \cdot M_i \equiv_{m_i} 1, \quad \forall i = 1, \ldots, n.$$

Setzen wir

$$x := \left(\sum_{i=1}^{n} a_i \cdot \tilde{y}_i \cdot M_i \right) \text{ mod } m,$$

so gilt

$$a_i \cdot \tilde{y}_i \cdot M_i = a_i \cdot (1 - \tilde{x}_i \cdot m_i) \cdot M_i = a_i - a_i \cdot \tilde{x}_i \cdot M_i \equiv_{m_i} a_i, \quad \forall i = 1, \ldots, n.$$

Für jedes $i \neq j$ ist m_i ein Teiler von M_j und daher ist

$$a_j \cdot \tilde{y}_j \cdot M_j \equiv_{m_i} 0, \quad \forall i \neq j.$$

Zusammen folgt schließlich

$$x \equiv_{m_i} a_i \cdot \tilde{y}_i \cdot M_i + \sum_{j \neq i} a_j \cdot \tilde{y}_j \cdot M_j \equiv_{m_i}, \quad \forall i = 1, \ldots, n.$$

Eindeutigkeit: Seien x und $\hat{x}$ zwei verschiedene Lösungen. Dann ist mit $x = q \cdot m_i \cdot M_i + r$ und $\hat{x} = \hat{q} \cdot m_i \cdot M_i + \hat{r}$ zunächst

$$x - \hat{x} = (q - \hat{q}) \cdot m_i \cdot M_i + (r - \hat{r}).$$

Andererseits ist $x - \hat{x} = t \cdot m_i$ und damit $r - \hat{r} = u_i \cdot m_i$ für jedes $i = 1, \ldots, n$. Da die m_i paarweise teilerfremd sind, ist kgV$(m_1, \ldots, m_n) = m$ und somit muss $r - \hat{r}$ ein Vielfaches von m sein. Da es sich um Reste modulo m handelt, bleibt nur $r - \hat{r} = 0$ mod m. Zuletzt folgt $x - \hat{x} = 0$ mod m.

□

Beispiel A.42

Wir betrachten folgende Aufgabenstellung, in der mehrfache Kongruenzen zu lösen sind:

$$x = 5 \text{ mod } 11 = 7 \text{ mod } 13.$$

Es ist ggT$(11, 13) = 1$ und damit gibt es ein eindeutiges x, das die beiden Kongruen-

zen $x \equiv_{11} 5$ und $x \equiv_{13} 7$ erfüllt. Es ist

$$1 = 6 \cdot 13 - 7 \cdot 11$$

und damit $x = 5 \cdot 6 \cdot 13 - 7 \cdot 7 \cdot 11 \bmod 143 = 137$.

Der Chinesische Restsatz hilft uns dabei, Operationen in $\mathbb{Z}_m$ bei bekannter teilerfremder Zerlegung $m = m_1 \cdot \ldots \cdot m_n$ auf $\mathbb{Z}_{m_1}, \ldots, \mathbb{Z}_{m_n}$ zu verlagern. Dazu betrachten wir

$$\prod_{i=1}^{n} \mathbb{Z}_{m_i} := \mathbb{Z}_{m_1} \times \ldots \times \mathbb{Z}_{m_n}.$$

Auf dieser Menge seien die Abbildungen

$$\begin{aligned}
&\oplus : \prod_{i=1}^{n} \mathbb{Z}_{m_i} \times \prod_{i=1}^{n} \mathbb{Z}_{m_i} \to \prod_{i=1}^{n} \mathbb{Z}_{m_i}, \\
&(r_{m_1}, \ldots, r_{m_n}) \oplus (s_{m_1}, \ldots, s_{m_n}) := (r_{m_1} +_{m_1} s_{m_1}, \ldots, r_{m_n} +_{m_n} s_{m_n}) \\
&\odot : \prod_{i=1}^{n} \mathbb{Z}_{m_i} \times \prod_{i=1}^{n} \mathbb{Z}_{m_i} \to \prod_{i=1}^{n} \mathbb{Z}_{m_i}, \\
&(r_{m_1}, \ldots, r_{m_n}) \odot (s_{m_1}, \ldots, s_{m_n}) := (r_{m_1} \cdot_{m_1} s_{m_1}, \ldots, r_{m_n} \cdot_{m_n} s_{m_n})
\end{aligned}$$

definiert.

Satz A.43

Betrachte den Ring $(\mathbb{Z}_m, \cdot_m, +_m, 0, 1)$. Sei $m = m_1 \cdot \ldots \cdot m_n$ mit ggT$(m_i, m_j) = 1$ für $i \neq j$. Dann ist durch

$$f : \mathbb{Z}_m \to \prod_{i=1}^{n} \mathbb{Z}_{m_i},\ a \mapsto f(a) := (a \bmod m_1, \ldots, a \bmod m_n)$$

ein Ringisomorphismus gegeben, d.h. $(\prod_{i=1}^{n} \mathbb{Z}_{m_i}, \oplus, \odot, (0, \ldots, 0), (1, \ldots, 1))$ ist ein Ring.

Beweis.

Für zwei beliebige Zahlen $a, b \in \mathbb{Z}$ mit $a \equiv_m b$ gilt mit der Notation aus Satz A.40 $a - b = q \cdot m = q \cdot M_i \cdot m_i$, also $a \equiv_{m_i} b$ für alle $i = 1, \ldots, n$. Für $a = u \cdot m + r = u \cdot M_i \cdot m_i + r$ ist außerdem $a \bmod m_i = r \bmod m_i$. Wir zeigen zunächst, dass die

Abbildung strukturerhaltend ist:

$$\begin{aligned}
f(a +_m b) &= (a +_m b \bmod m_1, \ldots, a +_m b \bmod m_n) = (a + b \bmod m_1, \ldots, a + b \bmod m_n)\\
&= (a \bmod m_1 +_{m_1} b \bmod m_1, \ldots, a \bmod m_n +_{m_n} b \bmod m_n)\\
&= (a \bmod m_1, \ldots, a \bmod m_n) \oplus (b \bmod m_1, \ldots, b \bmod m_n)\\
&= f(a) \oplus f(b),\\
f(a \cdot_m b) &= (a \cdot_m b \bmod m_1, \ldots, a \cdot_m b \bmod m_n) = (a \cdot b \bmod m_1, \ldots, a \cdot b \bmod m_n)\\
&= (a \bmod m_1 \cdot_{m_1} b \bmod m_1, \ldots, a \bmod m_n \cdot_{m_n} b \bmod m_n)\\
&= (a \bmod m_1, \ldots, a \bmod m_n) \odot (b \bmod m_1, \ldots, b \bmod m_n)\\
&= f(a) \odot f(b),\\
f(0) &= (0 \bmod m_1, \ldots, 0 \bmod m_n),\\
f(1) &= (1 \bmod m_1, \ldots, 1 \bmod m_n).
\end{aligned}$$

Nun zeigen wir, dass f bijektiv ist. Sei dazu $(a_1, \ldots, a_n) \in \prod_{i=1}^n \mathbb{Z}_{m_i}$ beliebig. Dann gibt es (surjektiv) nach dem Chinesischen Restsatz A.41 ein eindeutig (injektiv) bestimmtes $x \in \mathbb{Z}_m$ mit $x \equiv_{m_i} a_i$ für alle $i = 1, \ldots, n$. Damit ist f bijektiv und $(\prod_{i=1}^n \mathbb{Z}_{m_i}, \oplus, \odot, (0, \ldots, 0), (1, \ldots, 1))$ ist ein Ring.

□

Ist $a \in \mathbb{Z}_m^*$, d.h. ggT$(a, m) = 1$, so ist wegen $1 = x \cdot a + y \cdot m = x \cdot a + y \cdot M_i \cdot m_i$ auch ggT$(a, m_i) = 1$ und $a \bmod m_i \in \mathbb{Z}_{m_i}^*$. Ist andererseits $(a_1, \ldots, a_n) \in \prod_{i=1}^n \mathbb{Z}_{m_i}^*$, so gibt es nach dem Chinesischen Restsatz A.41 ein eindeutig bestimmtes $x \in \mathbb{Z}_m$ mit $x \equiv_{m_i} a_i$ für alle $i = 1, \ldots, n$. Somit lässt sich $x - a_i = q_i \cdot m_i$ schreiben bzw. $a_i = x - q_i \cdot m_i$. Wegen ggT$(a_i, m_i) = 1$ gibt es Zahlen u_i, v_i mit $1 = u_i \cdot a_i + v_i \cdot m_i = u_i \cdot (x - q_i \cdot m_i) + v_i \cdot m_i = u_i \cdot x + (v_i - u_i \cdot q_i) \cdot m_i$. Mit $w_i = v_i - u_i \cdot q_i$ folgt

$$\begin{aligned}
&\prod_{i=1}^n v_i \cdot m_i = \prod_{i=1}^n (1 - u_i \cdot x)\\
\Leftrightarrow\ & m \cdot \prod_{i=1}^n v_i = 1 - x \cdot g(u_1, \ldots, u_n, x),
\end{aligned}$$

wobei g eine nicht näher zu bestimmende Funktion ist. Damit ist ggT$(x, m) = 1$, $x \in \mathbb{Z}_m^*$. So ist mit Satz A.43

$$f_{|\mathbb{Z}_m^*} : \mathbb{Z}_m^* \to \prod_{i=1}^n \mathbb{Z}_{m_i}^*,\ a \mapsto f_{|\mathbb{Z}_m^*}(a) := (a \bmod m_1, \ldots, a \bmod m_n) \tag{A.4}$$

ein Gruppenisomorphismus, d.h. $\left(\prod_{i=1}^n \mathbb{Z}_{m_i}^*, \odot, (1, \ldots, 1)\right)$ ist eine Gruppe.

Satz A.44

Sei $m = m_1 \cdot \ldots \cdot m_n$ mit ggT$(m_i, m_j) = 1$ für $i \neq j$. Dann gilt $\phi(m) = \phi(m_1) \cdot \ldots \cdot \phi(m_n)$.

Beweis.

Mit (A.4) ist die Anzahl der Elemente von $\mathbb{Z}_m^*$ gleich der Anzahl der Elemente von $\prod_{i=1}^{n} \mathbb{Z}_{m_i}^*$.

□

B. Erweiterte ASCII-Tabelle

0	`	1	´	2	ˆ	3	˜	4	¨	5	˝	6	°	7	ˇ
8	˘	9	¯	10	˙	11	¸	12	˛	13	,	14	‹	15	›
16	“	17	”	18	„	19	«	20	»	21	–	22	—	23	
24	0	25	ı	26	j	27	ff	28	fi	29	fl	30	ffi	31	ffl
32	␣	33	!	34	"	35	#	36	$	37	%	38	&	39	’
40	(	41	)	42	*	43	+	44	,	45	-	46	.	47	/
48	0	49	1	50	2	51	3	52	4	53	5	54	6	55	7
56	8	57	9	58	:	59	;	60	<	61	=	62	>	63	?
64	@	65	A	66	B	67	C	68	D	69	E	70	F	71	G
72	H	73	I	74	J	75	K	76	L	77	M	78	N	79	O
80	P	81	Q	82	R	83	S	84	T	85	U	86	V	87	W
88	X	89	Y	90	Z	91	[	92	\	93	]	94	^	95	_
96	‘	97	a	98	b	99	c	100	d	101	e	102	f	103	g
104	h	105	i	106	j	107	k	108	l	109	m	110	n	111	o
112	p	113	q	114	r	115	s	116	t	117	u	118	v	119	w
120	x	121	y	122	z	123	{	124	|	125	}	126	~	127	-
128	Ă	129	Ą	130	Ć	131	Č	132	Ď	133	Ě	134	Ę	135	Ğ
136	Ĺ	137	Ľ	138	Ł	139	Ń	140	Ň	141	Ŋ	142	Ő	143	Ŕ
144	Ř	145	Ś	146	Š	147	Ş	148	Ť	149	Ţ	150	Ű	151	Ů
152	Ÿ	153	Ź	154	Ž	155	Ż	156	IJ	157	İ	158	đ	159	§
160	ă	161	ą	162	ć	163	č	164	ď	165	ě	166	ę	167	ğ
168	í	169	ľ	170	ł	171	ń	172	ň	173	ŋ	174	ő	175	ŕ
176	ř	177	ś	178	š	179	ş	180	ť	181	ţ	182	ű	183	ů
184	ÿ	185	ź	186	ž	187	ż	188	ij	189	¡	190	¿	191	£
192	À	193	Á	194	Â	195	Ã	196	Ä	197	Å	198	Æ	199	Ç
200	È	201	É	202	Ê	203	Ë	204	Ì	205	Í	206	Î	207	Ï
208	Đ	209	Ñ	210	Ò	211	Ó	212	Ô	213	Õ	214	Ö	215	Œ
216	Ø	217	Ù	218	Ú	219	Û	220	Ü	221	Ý	222	Þ	223	SS
224	à	225	á	226	â	227	ã	228	ä	229	å	230	æ	231	ç
232	è	233	é	234	ê	235	ë	236	ì	237	í	238	î	239	ï
240	ð	241	ñ	242	ò	243	ó	244	ô	245	õ	246	ö	247	œ
248	ø	249	ù	250	ú	251	û	252	ü	253	ý	254	þ	255	ß

Literatur

[1] T. Baigneres u. a. *A Classical Introduction to Cryptography Exercise Book*. New York: Springer, 2006. ISBN: 0-387-27934-2.

[2] A. Beutelspacher, J. Schwenk und K.-D. Wolfenstetter. *Moderne Verfahren der Kryptographie*. 2. Wiesbaden: Vieweg, 2006. ISBN: 3-8348-0083-X.

[3] J. Buchmann. *Einführung in die Kryptographie*. 1. Berlin Heidelberg: Springer Verlag, 2010. ISBN: 3-540-41283-2.

[4] A. J. Menezes, P. C. van Oorschot und S. A. Vanstone. *Handbook of Applied Cryptography*. englisch. 5. London: CRC Press, 2001. ISBN: 0-8493-8523-7.

[5] K. Meyberg und C. Karpfinger. *Algebra*. Heidelberg: Spektrum Akademischer Verlag, 2009. ISBN: 978-3-8274-2018-3.

[6] John G. Proakis. *Digital Communications*. New York: McGraw-Hill Higher Education, 2001. ISBN: 0-07-232111-3.

[7] G. Schäfer und M. Roßberg. *Netzsicherheit*. deutsch. 2. Heidelberg: dpunkt.verlag, 2014. ISBN: 978-3-86490-115-7.

[8] R. Schmied. *Cryptology for Engineers*. World Scientific, 2020. DOI: 10.1142/11492. eprint: https://www.worldscientific.com/doi/pdf/10.1142/11492. URL: https://www.worldscientific.com/doi/abs/10.1142/11492.

[9] C. Sorge, N. Gruschka und L. Lo Lacono. *Sicherheit in Kommunikationsnetzen*. München: Oldenbourg, 2013. ISBN: 978-3-486-72016-7.

[10] M. Stamp und R. M. Low. *Applied Cryptanalysis*. 1. New Jersey: Wiley, 2007. ISBN: 978-0-470-11486-5.

[11] M. Welschenbach. *Kryptographie in C und C++*. 1. Berlin Heidelberg: Springer, 1998. ISBN: 3-540-64404-0.

[12] A. Werner. *Elliptische Kurven in der Kryptographie*. 1. Berlin Heidelberg: Springer, 2002. ISBN: 3-540-42518-7.

Index