



universität
wien

DIPLOMARBEIT / DIPLOMA THESIS

AES - Verschlüsselung in electronic Commerce Systeme
Ein Konzept zum Unterrichten des Rijndael Algorithmus

verfasst von / submitted by
Herbert Bauer, Bakk

angestrebter akademischer Grad / in partial fulfilment of the requirements for the
degree of
Magister

Wien, im Juni 2018
Studienkennzahl lt. Studienblatt /
degree programme code as it appears on the
student record sheet:

A 190 884 406

Studienrichtung lt. Studienblatt /
degree programme as it appears on the student
record sheet:

Lehramtsstudium UF Mathematik
UF Informatik und Informatikmanagement

Betreut von / Supervisor:

Univ.-Prof. Dipl.-Ing. DDr. Gerald Quirchmayr

© Copyright 2018 Herbert Bauer, Bakk

Diese Arbeit wird unter den Bedingungen der *Creative Commons Lizenz Namensnennung–NichtKommerziell–KeineBearbeitung Österreich* (CC BY-NC-ND) veröffentlicht – siehe <http://creativecommons.org/licenses/by-nc-nd/3.0/at/>.

Erklärung

Ich erkläre eidesstattlich, dass ich die vorliegende Arbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen nicht benutzt und die den benutzten Quellen entnommenen Stellen als solche gekennzeichnet habe. Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt.

Wien, am 17. Juni 2018

Herbert Bauer, Bakk

Danksagung

Ich sage meine Schwestern

Bauer Anna
und
Bauer Johanna

dank, die hier eifrig Fehler jagten, sowie ihre sonstige Unterstützung.

Ich bedanke mich bei der Person „Nayuki“, dessen Name ich nicht näher angeben kann, für seine Erlaubnis zur Verwendung des Excel und LibreOffice Dokuments im Unterricht, sowie seiner Hilfe bei den Generatoren.

Inhaltsverzeichnis

Erklärung	iii
Danksagung	iv
Einleitung	1
0.1. Einleitung	1
0.2. Ziel	1
0.3. Aufbau	1
1. Mathematische Grundlagen	3
1.1. Grundlagen	3
1.2. Grundlegende Algebra	5
1.3. Zahlenmenge	9
1.4. Lineare Algebra	9
2. Einfaches Beispiel	10
2.1. Eine Äquivalenzrelation	10
2.1.1. Die verfügbaren Zeichen	10
2.1.2. Der Klartext	10
2.2. Key Schedule - Die Schlüsselfestlegung	12
2.3. Key Expansion	12
2.4. RC round constant (Rundenkonstante)	12
2.5. ShiftRows	13
2.6. SubBytes	13
2.7. Erweiterter Euklidischer Algorithmus	14
2.8. round key	16
2.8.1. round key 1	16
2.8.2. round key 2	17
2.8.3. round key 3	18
2.8.4. round key 4	18
2.9. Der Algorithmus	19
2.9.1. Vorrunde	19
2.9.2. Erste von „n-1“ Runden	20
2.9.3. Ergebnis der zweiten Runde	24
2.9.4. Letzte Runde	24
2.9.5. Chiffretext	26
2.10. Entschlüsselung	27
2.10.1. InvLetzte Runde	28
3. Logische Gatter	32
4. AES Algorithmus	34
4.1. 8 Bits - Die verfügbaren Zeichen	34

4.2.	Der Klartext	34
4.2.1.	Key Schedule - Die Schlüsselfestlegung	34
4.2.2.	Key Expansion	35
4.2.3.	RC round constant (Rundenkonstante)	35
4.2.4.	Galoisfeld	35
4.2.5.	Shift Rows	36
4.2.6.	SubBytes	37
4.2.7.	round key bei N_4	42
4.2.8.	round key bei N_6	45
4.2.9.	round key bei N_8	47
4.3.	Der Algorithmus	50
4.3.1.	Vorrunde	50
4.3.2.	Erste von n-1 Runden	51
4.4.	Die Algorithmus - Entschlüsselung	56
4.4.1.	InvAddRoundKey	56
4.4.2.	InvMixColumns	56
4.4.3.	InvShiftRows	57
4.4.4.	InvSubBytes	57
4.4.5.	Invmultiplikativ Inverse	58
5.	Stundeneinteilung	60
5.1.	Stundeneinteilung	60
5.1.1.	Doppelstunde	60
5.1.2.	Zweite Doppelstunde	60
5.1.3.	Dritte Doppelstunde	61
5.1.4.	Vierte Doppelstunde	61
5.1.5.	Fünfte Doppelstunde	61
5.1.6.	Sechste Doppelstunde	61
5.1.7.	Siebente Doppelstunde (optional)	61
5.1.8.	Achte Doppelstunde	62
5.1.9.	Neunte Doppelstunde	62
5.1.10.	Zehnte Doppelstunde	62
5.1.11.	Elfte Doppelstunde	62
5.1.12.	Zwölfte Doppelstunde	62
5.1.13.	Dreizehnte Doppelstunde	63
5.1.14.	Vierzehnte Doppelstunde	63
5.2.	Die Besonderheit der letzten Runde	63
5.3.	Einfaches Beispiel	63
5.3.1.	Inverse Funktionen	63
5.4.	AES	63
5.4.1.	Inverse Funktion	63
5.5.	Tabellenkalkulationsprogramm	63
6.	Code	65
6.0.1.	Generatoren	67
7.	Ergebnisse	76
7.1.	Beitrag der Arbeit	76
7.2.	Probleme	76
7.3.	Zweck	76

7.4. Weiterführende Arbeiten	76
Appendix	77
A. Zusammenfassung	77
B. Einfaches Beispiel	78
B.1. Schlüsselerweiterung	78
B.1.1. round key 2	78
B.1.2. round key 3	79
B.1.3. round key 4	81
B.2. Algorithmus	82
B.2.1. Zweite Runde	82
B.3. Entschlüsselung	84
B.3.1. Zweite Runde	84
B.3.2. Erste Runde	85
B.3.3. InvVorunde	87
C. AES	88
C.1. AES Schlüsselexpansion	88
C.1.1. round key bei N_6	88
C.1.2. round key bei N_8	92
C.2. AES Algorithmus	99
D. Übungsbeispiele	105
D.1. Beispiele Schlüsselexpansion	105
D.1.1. Beispiele zur Berechnung der multiplikativ Inversen	105
D.1.2. Aufgabenstellung	105
D.1.3. Lösung	105
D.2. Beispiele RC round constante	106
D.2.1. Aufgabenstellung	106
D.2.2. Lösung	107
D.2.3. Aufgabenstellung	107
D.2.4. Lösung	107
D.3. Beispiel RC, round key und Key Expansion	108
D.3.1. Aufgabenstellung	108
D.3.2. Lösung	108
D.3.3. Aufgabenstellung	109
D.3.4. Lösung	109
D.3.5. Aufgabenstellung	110
D.3.6. Lösung	110
D.3.7. Aufgabenstellung	111
D.3.8. Lösung	111
D.4. Beispiele Verschlüsselungsrunden	113
D.4.1. Aufgabenstellung	113
D.4.2. Lösung	113
D.4.3. Aufgabenstellung	114
D.4.4. Lösung	115
D.4.5. Aufgabenstellung	116
D.4.6. Lösung	116

D.4.7. Aufgabenstellung 117

D.4.8. Lösung 118

E. A Stick Figure Guide to the Advanced Encryption Standard (AES) 120

E.1. A Stick Figure Guide to the Advanced Encryption Standard (AES) 121

E.1.1. Act 1: Once Upon a Time... 121

E.1.2. Act 2: Crypto Basics 126

E.1.3. Act 3: Details 127

E.1.4. Act 4: Math! 133

Tabellenverzeichnis 139

Abbildungsverzeichnis 144

Codeverzeichnis 145

Index 147

Quellenverzeichnis 149

Literatur 149

Online-Quellen 149

Bildnachweis 150

Einleitung

0.1. Einleitung

Diese Arbeit beschäftigt sich mit einem Unterrichtskonzept für den AES - Advanced Encryption Standard, eine gegenwärtig (2018) stark verbreitete Verschlüsselung. Der Algorithmus soll der Leserin und dem Leser näher gebracht werden. Diese Arbeit kann weiters als Grundlage für den Mathematik-/ Informatikunterricht dienen.

Ein **Einsatz in einer AHS** ist nur bedingt möglich: Der volle Umfang dieser Arbeit übersteigt die Stundenzahl einer Wahlpflichtfachgruppe, da er zuviel Zeit in Anspruch nehmen würde. Als Ausgangsmaterial für Maturafragen eignet sich der Stoff ebenfalls nicht da die Grenzen zu eng gesetzt sind, doch Teile daraus könnten dazu dienen um den Schülerinnen und Schülern das Konzept der Blockchiffre zu erklären und um die Unterschiede zwischen Electronic Code Book (ECB), Cipher Block Chaining (CBC), Cipher Feedback (CFB) und Output Feedback (OFB) aufzuzeigen.

Ein **Einsatz an HTLs** ist ebenfalls problematisch, denn aufgrund der Einsparungen im Unterricht ist es nicht mehr möglich den benötigten mathematischen Umfang im Unterricht zu erarbeiten. Der Text könnte jedoch als Hilfestellung für die Implementierung dienen.

Eine Verwendung in der **Lehre an Universitäten** ist möglich. Je nach Ausrichtung des Institutes fehlt es den Studentinnen und Studenten vielfach entweder am mathematischen Verständnis für den Algorithmus oder an der Fähigkeit des Programmierens. Die Fähigkeiten des Programmierens, um den Code nachzuvollziehen, braucht nicht besonders weit entwickelt sein, da in Schleifen nur Elemente getauscht oder Bits gerückt werden. Hier wäre die Arbeit daher nützlich um Studenten, die sich mit dem Code beschäftigen, eine Vorstellung des mathematischen Unterbaus zu geben. Für Studenten, die sich mathematisch mit dem Algorithmus beschäftigen, ist diese Arbeit nicht geeignet, da weder das Galoisfeld erklärt wird, noch statische Aussagen über die Sicherheit des Algorithmus gemacht werden, oder Berechnungen über die Qualität.

0.2. Ziel

Das Bestreben dieser Arbeit ist es, einen Einblick in den Algorithmus, der hinter AES steht, zu geben. Leider konnte in der Arbeit nicht darauf eingegangen werden wie stark jede einzelne Funktion vor einem Einbrechen in den Code schützt. Es sollte trotzdem, aufgrund des Verständnisses der Arbeitsweise, nachher verständlich sein, welche Funktionen stark oder schwach sind oder erst im Zusammenspiel ihre Stärke potenzieren.

0.3. Aufbau

Der Aufbau der Arbeit ist folgendermaßen gestaltet. Es wird mit einem einfachen Beispiel, welches die Funktionsweise des Algorithmus auf einen Restklassenkörper transferiert, begonnen. Im Anschluss folgt eine Beschreibung des Algorithmus in seiner eigentlichen Form. Den Abschluss bildet eine Besprechung des Codes, der eine Implementierung in C++ ist. In den Anhängen sind die Zwischenschritte des einfachen Beispiels und des Algorithmus

abgelegt. Des Weiteren sind Übungsbeispiele mit Lösungen angegeben die zur Überprüfung des Verständnisses dienen. Ansatz und Motivation Als ich mich in das Thema einarbeitete, stellte ich fest, dass es zwei Arten von Beschreibungen gab. Die Einen waren häufig technischer Natur und beschäftigen sich mit Art und Weise der Implementierung, die Anderen beschäftigen sich mit dem mathematischen Aspekt des Themas, nicht jedoch mit dem Algorithmus. Diese Arbeiten vergleichen AES-Algorithmus mit anderen Algorithmen und versuchen die jeweiligen mathematischen Schwächen oder Stärken aufzuzeigen. Ich wollte die Lücke des nicht beschriebenen Algorithmus in der Mathematik schließen und versuchte das Thema so einfach wie möglich zu beschreiben, ohne es unnötig zu verfälschen. Der Leser soll von seinem mathematischen Niveau, entweder HTL oder AHS, abgeholt und in das Thema eingeführt werden, ohne seinen jeweiligen Wissensstand in der Mathematik zu sehr oder zu schnell zu erweitern und dadurch die Leserin oder den Leser zu verlieren. Ich hoffe, mein Ziel durch diese Arbeit erreicht zu haben.

1. Mathematische Grundlagen

1.1. Grundlagen

Die folgenden Definitionen, Propositionen und Theoreme sind ein Auszug aus „Einführung in das mathematische Arbeiten“ von Schichl und Steinbauer [4].

Definition 1 (Primzahl [4, Seite 23])

Eine Primzahl ist eine natürliche Zahl $p > 1$, die nur die trivialen Teiler besitzt, d.h. deren einziger Teiler 1 und sie selbst sind.

Definition 2 (Relation [4, Seite 145])

Seien M und N Mengen. Eine Relation auf $M \times N$ ist eine Teilmenge R des Cartesischen Produkts, d.h.

$$R \subseteq M \times N.$$

Für zwei Elemente $a \in M$ und $b \in N$ sagen wir a steht in Relation mit b , falls $(a, b) \in R$ gilt. Wir schreiben dann in Symbolen $a R b$.

Stehen a und b nicht miteinander in Relation, so schreiben wir $a \not R b$.

Definition 3 (Transitivität, Reflexivität [4, Seite 146])

Sei R eine Relation aus einer Menge M

1. R heißt transitiv, wenn für alle $a, b, c \in M$ gilt, dass

$$a R b \wedge b R c \Rightarrow a R c.$$

2. R heißt reflexiv, wenn für alle $a \in M$ gilt, dass

$$a R a.$$

Definition 4 (Äquivalenzrelation [4, Seite 147])

Eine reflexive und transitive Relation $\sim$ auf einer Menge M heißt Äquivalenzrelation, falls sie zusätzlich symmetrisch ist, d.h. dass

$$\forall x, y \in M : (x \sim y \Rightarrow y \sim x)$$

gilt. Gilt $x \sim y$, so nennen wir x und y äquivalent.

Definition 5 (Äquivalenzklasse [4, Seite 148])

Sei M eine Menge und $\sim$ eine Äquivalenzrelation von $a \in M$ durch

$$C_a := \{b \in M \mid b \sim a\}.$$

Alternative Bezeichnungen für C_a sind auch $[a]$ und $\bar{a}$. Außerdem nennen wir jedes $b \in C_a$ einen Repräsentanten der Äquivalenzklasse C_a .

Proposition 1 (Eigenschaften von Äquivalenzklassen [4, Seite 148 (Beweis: siehe dort)])

Sei M eine Menge und $\sim$ eine Äquivalenzrelation auf M . Dann sind zwei Äquivalenzklassen C_a und C_b entweder disjunkt oder gleich. In Symbolen

$$C_a \cap C_b \neq \emptyset \Leftrightarrow C_a = C_b.$$

Definition 6 (Partition [4, Seite 150])

Eine Familie disjunkter nichtleerer Teilmengen einer Menge, die die gesamte Menge überdecken, nennt man Partition. Etwas formaler können wir auch schreiben: Eine Familie U_i , $i \in I$ (I eine beliebige Indexmenge) von Teilmengen einer Menge M heißt Partition von M , falls die folgenden drei Eigenschaften gelten:

1. $U_i \cap U_j \neq \emptyset \Leftrightarrow U_i = U_j, \forall i, j \in I$,
2. $\bigcup_{i \in I} U_i = M$,
3. $U_i \neq \emptyset$.

Anders ausgedrückt ist eine Familie von Teilmengen einer Menge M genau dann eine Partition, wenn jedes Element von M in genau einer der Teilmengen liegt: Eigenschaft (2.) sorgt dafür, dass jedes Element von M in mindestens einer der Teilmengen liegt, Eigenschaft (1.) dafür, dass es höchstens eine ist.

Theorem 2 (Äquivalenzklassen als Partition [4, Seite 150])

Jede Äquivalenzrelation $\sim$ auf einer Menge M definiert eine Partition von M . Umgekehrt kann man aus jeder Partition $U_i, i \in I$ eine Menge M eine Äquivalenzrelation $\sim$ gewinnen, indem man definiert:

$$a \sim b :\Leftrightarrow \exists i \in I. a, b \in U_i.$$

Definition 7 (Quotientenmenge [4, Seite 151])

Sei $\sim$ eine Äquivalenzrelation auf M . Wir definieren die Quotientenmenge oder auch Faktormenge $M/\sim$ (sprich M modulo Tilde) als die Menge aller Äquivalenzklassen bezüglich $\sim$.

Definition 8 (Kongruent modulo n [4, Seite 151])

Sei $N \ni n \leq 2$ gegeben. Wir definieren auf $\mathbb{Z}$ die Relationen

$$k \sim_n l :\Leftrightarrow \exists m \in \mathbb{Z} \text{ mit } k = l + mn.$$

Gilt $k \sim_n l$ so sagen wir k ist kongruent l modulo n und schreiben manchmal auch $k \equiv l \pmod{n}$.

Proposition 3 (Kongruenz als Äquivalenzrelation [4, Seite 152])

Die Relation $\sim_n$ ist eine Äquivalenzrelation.

Definition 9 (Restklassen [4, Seite 153])

Die n -elementige Faktormenge $\mathbb{Z}_{\sim_n}$ wird in der Mathematik üblicherweise mit $\mathbb{Z}_n$ bezeichnet und man nennt sie die Restklassen modulo n oder die Kongruenzklassen modulo n .

1.2. Grundlegende Algebra

Definition 10 (Gruppoid. Algebraische Struktur [4, Seite 197])

Sei G eine nichtleere Menge.

1. Eine Verknüpfung auf G ist eine Abbildung

$$\circ : G \times G \rightarrow G.$$

An Stelle von $\circ (g, h)$ für zwei Elemente $g, h \in G$ schreiben wir $g \circ h$, und wir nennen das Bild von (g, h) unter $\circ$ das Ergebnis oder den Wert der Verknüpfung.

2. Wenn wir die Menge G zusammen mit der Verknüpfung $\circ$ untersuchen, so schreiben wir meist $(G, \circ)$ und nennen sie Gruppoid, algebraische Struktur oder Magma. In diesem Zusammenhang nennen wir G auch Grundmenge.

Definition 11 (Operationen für Restklassen [4, Seite 199])

Auf der Menge der Restklassen $\mathbb{Z}_n$ (siehe Definition 9 Restklassen auf Seite 4) definieren wir durch

$$\begin{aligned}\bar{a} + \bar{b} &:= \overline{a + b}, \\ \bar{a} \cdot \bar{b} &:= \overline{ab}\end{aligned}$$

zwei Verknüpfungen, die Addition und die Multiplikation von Restklassen.

Definition 12 (Assoziativgesetz, Halbgruppen [4, Seite 207])

Ein Gruppoid $(G, \circ)$ heißt Halbgruppe, falls die Verknüpfung assoziativ ist, also das Assoziativgesetz

$$(AG) \quad \forall g, h, k \in G : (g \circ h) \circ k = g \circ (h \circ k)$$

gilt. In diesem Fall ist das Setzen von Klammern nicht notwendig, und wir dürfen an Stelle von $(g \circ h) \circ k$ einfach $g \circ h \circ k$ schreiben.

Definition 13 (Einselement [4, Seite 209])

Ein Element e eines Gruppoids G heißt Einselement oder neutrales Element, falls

$$\forall g \in G : g \circ e = e \circ g = g$$

gilt. Wird die Verknüpfung mit $+$ bezeichnet (additiv geschrieben), so bezeichnet man e oft mit 0 oder $\mathbf{0}$ und nennt es Nullelement. Einselemente bezüglich multiplikativ geschriebener Verknüpfungen erhalten auch die oft die Bezeichnung 1 oder $\mathbb{1}$.

Definition 14 (Idempotenz [4, Seite 212])

Ein Element g in einem Gruppoid $(G, \circ)$ heißt idempotent, falls

$$g \circ g = g$$

gilt.

Definition 15 (Monoid [4, Seite 212])

Ist $(G, \circ)$ eine Halbgruppe und existiert ein Einselement $e \in G$, so nennt man G auch Monoid und schreibt oft $(G, \circ, e)$.

Definition 16 (Kommutativgesetz [4, Seite 217])

Eine Verknüpfung $\circ$ in einem Gruppoid $(G, \circ)$ heißt kommutativ, falls das Kommutativgesetz erfüllt ist, d.h. falls

(KG) $\forall g, h \in G : g \circ h = h \circ g$

gilt.

Definition 17 (Inverses Element [4, Seite 217])

Sei $(G, \circ, e)$ ein Gruppoid mit Einselement, und sei $a \in G$. Wir nennen $a' \in G$ inverses Element von a , falls

$$a' \circ a = a \circ a' = e$$

gilt. Wir nennen a' auch inverses zu a und schreiben meist a^{-1} . Ist das Verknüpfungszeichen ein $+$, schreiben wir die Operation additiv, d. h. das Inverse von a schreiben wir mit $-a$.

Definition 18 (Gruppe [4, Seite 219])

Sei $(G, \circ)$ ein Gruppoid. Gelten die folgenden Eigenschaften

(G1) Assoziativgesetz: $\forall g, h, k \in G : (g \circ h) \circ k = g \circ (h \circ k)$,

(G2) Einselement: $\exists e \in G : \forall g \in G : e \circ g = g \circ e = g$,

(G3) Inverse: $\forall g \in G : \forall g^{-1} \circ g = g \circ g^{-1} = e$,

dann heißt $(G, \circ)$ Gruppe. Gilt außerdem noch

(G4) Kommutativgesetz: $\forall g, h \in G : g \circ h = h \circ g$,

dann heißt G kommutative oder abelsche Gruppe (nach Nils Hendrik Abel (1802-1829)).

Definition 19 (Distributivgesetze [4, Seite 236])

Sei H ein Gruppoid bezüglich der beiden Verknüpfungen $+$ und $\cdot$, das wir mit $(H, +, \cdot)$. Wir sagen $+$ erfüllt die beiden Distributivgesetze bzgl. $\cdot$, falls gilt

$$\textbf{(DG1)} \quad \forall a, b, c \in H : a(b + c) = ab + ac,$$

$$\textbf{(DG2)} \quad \forall a, b, c \in H : (b + c)a = ba + ca.$$

Definition 20 (Ring [4, Seite 237])

Sei $(R, +, \cdot)$ ein Gruppoid bezüglich beider Verknüpfungen.

1. $(R, +, \cdot)$ heißt Ring, falls folgende Eigenschaften erfüllt sind:

(R1) $(R, +, 0)$ ist abelsche Gruppe,

(R2) $(R, \cdot)$ erfüllt das Assoziativgesetz, d.h. ist Halbgruppe,

(R3) $(R, +, \cdot)$ erfüllt die Distributivgesetze von $+$ bzgl. $\cdot$.

2. Ist zusätzlich $(R, \cdot, 1)$ ein Monoid und gilt $0 \neq 1$, so sagen wir R ist ein Ring mit Eins(element) und schreiben oft auch $(R, +, \cdot, 0, 1)$.
3. Ist die Operation $\cdot$ kommutativ, so liegt ein kommutativer Ring vor.
4. Hat man beides, Kommutativität und Einselement, dann nennt man die entstandene Struktur ganz einfach kommutativer Ring mit Einselement.

Proposition 4 (Restklassenringe [4, Seite 238])

Die Restklassen $\mathbb{Z}_n$ bilden (mit der Addition und der Multiplikation von Restklassen) einen kommutativen Ring mit Einselement.

Definition 21 (Einheit [4, Seite 242])

Ein Element a eines Ringes $(R, +, \cdot)$ mit Einselement heißt Einheit, falls es ein Inverses bezüglich der Multiplikation besitzt.

Die Menge R^* aller Einheiten in einem Ring R mit Eins bildet eine Gruppe bezüglich der Multiplikation, die Einheitsgruppe[...].

Definition 22 (Teiler [4, Seite 243])

Seien $a \neq 0$ und c Elemente eines kommutativen Ringes $(R, +, \cdot)$. Wir sagen, a teilt c , in Zeichen $a|c$, falls ein $0 \neq b \in R$ existiert mit $ab = c$. Wir nennen dann a einen Teiler von c .

Definition 23 (Nullteiler, Integritätsbereich [4, Seite 244])

Sei $(R, +, \cdot)$ ein kommutativer Ring.

1. Elemente $0 \neq r, s \in R$ heißen Nullteiler, falls $rs = 0$ gilt.
2. Besitzt R ein Einselement, so heißt R Integritätsbereich, wenn für je zwei Elemente $r, s \in R$ aus $rs = 0$ oder $r = 0$ oder $s = 0$ folgt.

Obige Definition besagt also, dass es in einem Integritätsbereich keine Nullteiler gibt, genauer gesagt, dass ein Integritätsbereich ein nullteilerfreier kommutativer Ring mit Eins ist.

Proposition 5 (Primzahlkriterium [4, Seite 248])

Eine natürliche Zahl $p > 1$ ist genau dann eine Primzahl, wenn für alle $k, l \in \mathbb{N} \setminus \{0, 1\}$ gilt

$$p|kl \Rightarrow p|k \vee p|l.$$

Definition 24 (Primelement [4, Seite 248])

Ein Nichtnullteiler $a \neq 0$ eines kommutativen Ringes $(R, +, \cdot)$ mit Einselement heißt prim oder ein Primelement, falls $a \notin R^*$ und für alle $r, s \in R$ gilt, dass

$$a|rs \Rightarrow a|r \vee a|s.$$

Proposition 6 (Euklidischer Algorithmus [4, Seite 252])

a und b seien zwei positive natürliche Zahlen. Wir definieren rekursiv eine Folge:

$$\begin{aligned} g_0 &= a, \\ g_b &= b, \\ g_{n-1} &= q_n g_n + g_{n+1}, \end{aligned}$$

wobei wir q_n und g_{n+1} durch Dividieren von g_{n-1} durch g_n mit Rest bestimmen. Die Rekursion bricht nach endlich vielen Schritten ab, d.h. es gibt ein kleinstes m mit $g_{m+1} = 0$. Dann gilt

$$ggT(a, b) = g_m.$$

Definition 25 (Körper [4, Seite 264])

Ein Gruppoid $(K, +, \cdot)$ mit den beiden Verknüpfungen $+$ und $\cdot$ heißt Körper, falls die folgenden Eigenschaften (die Körperaxiome) gelten:

- | | |
|---|--------------------------|
| (K1) $\forall a, b, c \in K : (a + b) + c = a + (b + c),$ | ((AG) $+$) |
| (K2) $\forall a, b \in K : a + b = b + a,$ | ((KG) $+$) |
| (K3) $\exists 0 \in K : \forall a \in K : a + 0 = a,$ | (Nullelement) |
| (K4) $\forall a \in K : \exists (-a) \in K : a + (-a) = 0,$ | (Inverse bzgl. $+$) |
| (K5) $\forall a, b, c \in K : (ab)c = a(bc),$ | ((AG) $\cdot$) |
| (K6) $\forall a, b \in K : ab = ba,$ | ((KG) $\cdot$) |
| (K7) $\exists 1 \in K : 1 \neq 0 \wedge \forall a \in K \setminus \{0\} : a1 = a,$ | (Einselement) |
| (K8) $\forall a \in K \setminus \{0\} : \exists a^{-1} \in K : aa^{-1} = 1,$ | (Inverse bzgl. $\cdot$) |

(K9) $\forall a, b, c \in K : a(b + c) = ab + ac.$ (Distributivität)

Korollar (Erweiterung Kongruent modulo n auf $\mathbb{Q}$ ¹)

Sei $\mathbb{N} \ni n \leq 2$ gegeben. Wir definieren auf $\mathbb{Q}$ die folgenden Relationen. Sei $k, o \in \mathbb{Z}$ und $k \sim_n p$ und $o \sim_n q$ und $\exists r \in \mathbb{N} : r < n \wedge r \cdot q = 1$

$$\frac{k}{o} \sim_n \frac{p}{q}$$

$$\sim_n p \cdot r.$$

Beweis:

$$\begin{aligned} \frac{k}{o} &= \frac{k}{o} \cdot 1 \\ &= k \cdot \frac{1}{o} \cdot q \cdot r \\ &\stackrel{o \sim_n q}{=} k \cdot \frac{1}{o} \cdot o \cdot r \\ &= k \cdot r \end{aligned}$$

□

¹Eine Folgerung aus Einheit (Definition 21 auf Seite 6), Inverses Element (Definition 17 auf Seite 6) und Restklassenkörper (Theorem 7 auf Seite 9)
(Allgemeiner bewiesen und erklärt in [1] auf Seite 704f „Quotientenkörper“.

1.3. Zahlenmenge

Theorem 7 (Restklassenkörper [4, Seite 269])

Die Restklassen $\mathbb{Z}_n$ bilden (mit der Addition und der Multiplikation von Restklassen) einen kommutativen Ring mit Einselement. Außerdem ist $\mathbb{Z}_p$ ein Körper, wenn p eine Primzahl ist.

Definition 26 (Galoisfelder [1, Seite 703])

Die endlichen Schiefkörper [gemeint sind Restklassenkörper] heißen Galoisfelder. Jedes Galoisfeld ist ein Körper. Zu jeder Primzahl p und zu $n = 1, 2, \dots$ gibt es einen Körper mit p^n Elementen. Auf diese Weise erhält man alle Galoisfelder. Zwei Galoisfelder sind genau dann isomorph, wenn sie die gleiche Anzahl von Elementen besitzen.

Theorem 8 (Fundamentalsatz der Algebra [4, Seite 349])

Sei $p(z)$ ein beliebiges nichtkonstantes Polynom mit komplexen Koeffizienten

$$p(z) = \sum_{i=0}^n a_i z^i \text{ mit } a_i \in \mathbb{C}, n \geq 1, a_n \neq 0.$$

Dann existiert ein $\alpha \in \mathbb{C}$ mit $p(\alpha) = 0$; es gibt also immer wenigstens eine (komplexe) Nullstelle.

1.4. Lineare Algebra

Theorem 9 (Die Cramersche Regel [1, Seite 641])

Es sei $n = m$ und $\det A \neq 0$. Dann besitzt das lineare Gleichungssystem

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + \dots + a_{2n}x_n &= b_2 \\ &\dots \\ a_{m1}x_1 + a_{m2}x_2 + \dots + a_{mn}x_n &= b_m \end{aligned}$$

die eindeutige Lösung $x = A^{-1}b$.

Explizit gilt:

$$x_j = \frac{(\det A)_j}{\det A}, \quad j = 1, \dots, n.$$

Dabei entsteht die Determinante $\det(A)_j$ aus der Determinante der Matrix A , indem man dort die j -te Spalte durch b ersetzt. Man bezeichnet die Lösungsformel als Cramersche Regel.

2. Einfaches Beispiel

2.1. Eine Äquivalenzrelation

2.1.1. Die verfügbaren Zeichen

Das in Österreich verwendete Alphabet hat 26 Zeichen wenn wir Sonderzeichen wie „ß“, „ä“, „ö“ und „ü“ außer Acht lassen. Die nächst größere Primzahl zu 26 ist 29. Warum eine Primzahl gebraucht wird, wird etwas später erklärt. Es kann auch eine andere Primzahl verwendet werden. Zwischen 29 und 26 beträgt die Differenz drei. Ich habe mich entschlossen, diese drei freien Plätze für Satzzeichen zu verwenden. Ich wählte LZ für Leerzeichen, PKT für Punkt und BST für Beistrich.

Tabelle 2.1.: Einfaches Beispiel: verfügbare Zeichen

	B	C	D	E	F	G	H	I	J	K
8	a	b	c	d	e	f	g	h	i	j
9	1	2	3	4	5	6	7	8	9	10
10	k	l	m	n	o	p	q	r	s	t
11	11	12	13	14	15	16	17	18	19	20
12	u	v	w	x	y	z	LZ	PKT	BST	
13	21	22	23	24	25	26	27	28	0	

Mit diesem eingeschränkten Zeichensatz können wir einen Text in Zahlen umsetzen. Wir verwenden hier eine Äquivalenzrelation.

Anmerkung: Ich habe dieses einfache Beispiel in einer Google Tabelle erstellt. Ich werde immer die Formeln angeben, mit denen ich diese Tabelle als eine dynamische Tabelle erstellt habe. Deswegen werde ich bei gezeigten Tabellen in diesem Abschnitt ihre Position in der Google Tabelle angeben, damit die Formeln übernommen werden können.

2.1.2. Der Klartext

Als Klartext habe ich den Satz „Susi ist klug.“ ausgewählt. Dieser Satz besteht aus 13 Zeichen. Dieser Satz wird in eine vier mal vier Matrix geschrieben, die sogenannte „state matrix“. Warum die Matrix quadratisch¹ ist, wird später erklärt. Falls der Text zu lang ist, wird eine weitere Matrix verwendet. Jede einzelne Matrix wird unabhängig von den anderen Matrizen nach demselben Algorithmus verschlüsselt. Bei allen Matrizen bleibt der Schlüssel gleich, somit ist es möglich parallel zu arbeiten. Die Größe der quadratischen Matrix ist nicht frei wählbar.

¹Die Matrix ist auch im AES quadratisch und vier mal vier groß. Im Rijndael wären auch andere $4 \times k$ state matrizen möglich.

Tabelle 2.2.: Einfaches Beispiel: Anzahl der Elemente

	C	D
7	n=	29

Tabelle 2.3.: Einfaches Beispiel: Klartext

	B	C	D	E
2	s	LZ	LZ	g
3	u	i	k	PKT
4	s	s	l	LZ
5	i	t	u	LZ

Der Klartext wurde spaltenweise von oben nach unten und von links nach rechts eingetragen. In Zahlen übertragen, kommt diese Tabelle heraus.

Tabelle 2.4.: Einfaches Beispiel: Übertragung

	B	C	D	E
16	19	27	27	7
17	21	9	11	28
18	19	19	12	27
19	9	20	21	27

Der Code für das erste Ergebnis für Tabelle 2.4 lautet:²

=WVERWEIS[22](B2; \$B\$8:\$K\$9\ \$B\$10:\$K\$11\ \$B\$12:\$J\$13; 2; FALSCH)

²Ich gebe den Funktionsaufruf in GoogleDocs für die obere linke Ecke an, weil sich die passende Funktion für das jeweilige Feld durch ziehen der Funktion ergibt.

2.2. Key Schedule - Die Schlüsselfestlegung

Key Schedule³(Schlüsselfestlegung) setzt sich aus zwei Teilen zusammen, der Key Expansion (Schlüsselerweiterung) und der round key selection (Einteilung der Rundenschlüssel). Der cipher key (Chiffrierschlüssel/eingegebener Schlüssel) ist von seiner Länge her begrenzt, hier wären es 16 Zeichen. Falls der Schlüssel kürzer wäre, müssten beim cipher key Zeichen ergänzt werden, bis die 16 Zeichen erreicht sind. Die blocklength (Blocklänge) (N_b) ist die Anzahl der Spalten der state matrix und wurde im AES auf vier Reihen festgelegt. Der zu Grunde liegende Algorithmus Rijndael würde auch andere Größen für $4 \times k$ Matrizen erlauben.

$N_b = 4$ - Anzahl der Spalten der state Matrix ist sowohl hier fix als auch in AES mit 4 festgelegt. In Rijndael könnten die state matrix eine andere Form ($4 \times n$) haben.

$N_k = 4$ - Anzahl der Spalten des cipher keys ist hier mit 4 festgelegt. AES als auch Rijndael würden auch andere Größen erlauben. Der cipher key ist ebenfalls eine $4 \times n$ Matrix.

N_r - Anzahl der benötigten round keys.

Tabelle 2.5.: Einfaches Beispiel: Anzahl der Runden

N_r	N_b
N_k	3

In unserer Tabelle sehen wir, dass wir drei round keys berechnen müssen.

2.3. Key Expansion

In jeder Runde der Verschlüsselung wird einmal ein round key (Rundenschlüssel) zu der state matrix hinzugezählt. Wir haben einen Schlüssel in der Form einer 4×4 Matrix. Es werden noch drei weitere 4×4 Matrizen benötigt.

$$1 + 3 = 4$$

Eine Vorrunde, zwei Runden und eine letzte Runde ergeben gesamt vier Runden.

2.4. RC round constant (Rundenkonstante)

In jeder Runde der *Funktion Key Expansion* wird eine RC (Rundenkonstante) benötigt, um den nächsten round key (Rundenschlüssel) zu berechnen. Weil drei round keys berechnet werden müssen, werden drei RC benötigt. Ein RC wird wie folgt berechnet. Die Basis des verwendeten Zahlensystems hoch der Ordnungszahl des gegenwärtig berechneten round key minus eins.

³Ich verwende die englischen Ausdrücke, weil sie in verschiedenen Beschreibung auftauchen, unabhängig der Sprache in der die Beschreibung abgefasst wurde.

Tabelle 2.6.: Einfaches Beispiel: RC (round constant) Tabelle

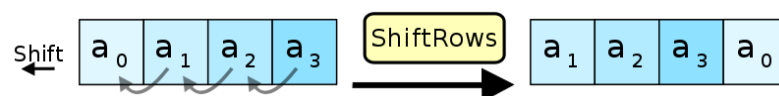
	B	C	D	E
		10 hoch	Wert	Modulo 29
30	RC(1)	1	1	1
31	RC(2)	2	10	10
32	RC(3)	3	100	13
33	RC(4)	4	1000	14

Der Code in der zweiten Zeile Nummer 30 für Tabelle 2.6 lautet:

RC(1) 1 = $10^{(B30 - 1)} = \text{REST}(C30; \$D\$7)$

2.5. ShiftRows

Bei Shift Rows werden die Einträge einer Zeile nach links verschoben. Die erste Zeile wird nicht verschoben. Die zweite Zeile um eine Stelle, die dritte Zeile um zwei und die vierte um drei. Die links stehenden Elemente, die aus der state matrix geschoben werden, werden dabei nach rechts auf die freigeworden Stelle verschoben.

Abbildung 2.1.: AES: ShiftRows operation for Key Expansion [25]⁴

2.6. SubBytes

Im Algorithmus wird die Funktion SubBytes verwendet. SubBytes setzt sich aus zwei Operationen zusammen: der Bestimmung der multiplikativen Inversen und einer affinen Abbildung. In diesem Beispiel wird die affine Abbildung (lineare Abbildung) “ $y=3x+3$,” verwendet, desweiteren wird ein Restklassenkörper verwendet. Weil 29 eine Primzahl ist, haben wir einen Restklassenkörper⁵, dessen Inversen eindeutig bestimmbar sind. Bei den Beispielen werden wir sehen, dass es auch Nullteiler⁶ gibt, und in solchen Tabellen gibt es kein eindeutig bestimmbar multiplikativ Inverses.

Tabelle 2.7.: Einfaches Beispiel: 1 Runde SubBytes lineare Funktion

	A	B	C	D	E
62	$y=$	3	x	+	3

⁴Das Bild wurde von Herbert Bauer bearbeitet.

⁵Siehe Definition 9 auf Seite 4.

⁶Siehe Definition 13 auf Seite 5.

Tabelle 2.8.: Einfaches Beispiel: multiplikativ Inverse

	B	C	D	E	F	G	H	I	J	K
22	1	2	3	4	5	6	7	8	9	10
23	1	15	10	22	6	5	25	11	13	3
24	11	12	13	14	15	16	17	18	19	20
25	8	17	9	27	2	20	12	21	26	16
26	21	22	23	24	25	26	27	28	0	
27	18	4	24	23	7	19	14	28	0	

Abbildung 2.2.: AES: SubBytes operation for Key Expansion [26]⁷

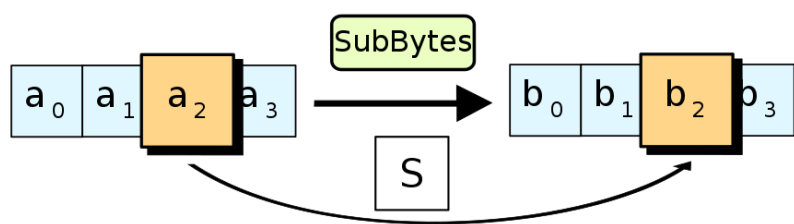


Tabelle 2.9.: Einfaches Beispiel: S-Box

1	2	3	4	5	6	7	8	9	10
6	19	4	11	21	18	20	7	13	12
11	12	13	14	15	16	17	18	19	20
27	25	1	26	9	5	10	8	23	22
21	22	23	24	25	26	27	28	0	
28	15	17	14	24	2	16	0	3	

2.7. Erweiterter Euklidischer Algorithmus

Zur Bestimmung der multiplikativen Inverse wird der erweiterte Euklidische Algorithmus verwendet. Als Beispiel der Arbeitsweise wird die Inverse von 16 bestimmt. Als erstes wird der $\text{ggT}(29; 16)$ bestimmt.

$29 \div 16 = 1$
13 Rest

$16 \div 13 = 1$
3 Rest

$13 \div 3 = 4$
1 Rest

Der ggT ist bestimmt, wenn der Rest 1 oder der Rest 0 ist. Im nächsten Schritt wird der Rest als Differenz von Dividend und Divisor mal Quotient angeschrieben.

⁷Das Bild wurde von Herbert Bauer bearbeitet.

$$13 - 4 * 3 = 1 \text{ wir setzen für 3 die Folgerung} \\ \text{der vorherigen Zeile, } (16 - 13 * 1 = 3) \text{ ein.}$$

$$13 - 4 * (16 - 13 * 1) = 1$$

$$13 - 4 * 16 + 13 * 4 = 1$$

$$-4 * 16 + 13 * 5 = 1 \text{ wir setzen für 13 die Folgerung} \\ \text{der vorherigen Zeilen, } (29 - 16 * 1 = 13) \text{ ein.}$$

$$-4 * 16 + (29 - 16 * 1) * 5 = 1$$

$$-4 * 16 + 29 * 5 - 16 * 5 = 1$$

$$-9 * 16 + 29 * 5 = 1$$

Nun wird dieses Modulo 29 genommen, um die multiplikative Inverse zu bestimmen.

$$-9 * 16 + 29 * 5 \equiv 1 \pmod{29}$$

$$-9 * 16 \equiv 1 \pmod{29} / + 29 * 16$$

$$20 * 16 \equiv 1 \pmod{29}$$

Die multiplikative Inverse zu 16 ist 20 Modulo 29.

2.8. round key

2.8.1. round key 1

Die Funktion Key Expansion läuft nach folgenden Schritten ab:

- 1. Letzte Spalte des vorherigen round key (Rundenschlüssels) wird genommen (Anmerkung: Vier Blöcke werden oft mit „word“ referenziert.)
- 2. Shift row: Auf die transponierte Spalte wird die Funktion Shift row angewandt.
- 3. SubBytes: In der Spalte wird jedes Element durch seine multiplikativ Inverse ersetzt, anschließend wird die lineare Funktion angewandt.
- 4. Zum ersten Eintrag wird der entsprechende RC dazu gezählt.

Tabelle 2.10.: Einfaches
Beispiel: cipher key

	M	N	O	P
	Word 0	Word 1	Word 2	Word 3
30	0	0	0	0
31	0	0	0	0
32	0	0	0	0
33	0	0	0	0

Tabelle 2.11.: Einfaches
Beispiel: ShiftRow RC1

	R
30	0
31	0
32	0
33	0

Tabelle 2.12.: Einfaches
Beispiel: SubBytes RC1

	T	U
	multiplikativ Inverses	lineare Funktion
30	0	3
31	0	3
32	0	3
33	0	3

Tabelle 2.13.: Einfaches
Beispiel: Erweiterung RC1

	W
	plus RC(1)
30	4
31	3
32	3
33	3

Der Code für das erste Ergebnis für Tabelle 2.10, Tabelle 2.11, Tabelle 2.12 und für Tabelle 2.13 lautet:

ShiftRows
=P31

SubBytes

multiplikativ Inverses =WVERWEIS(R30; {\$B\$22:\$K\$23\ \$B\$24:\$K\$25\ \$B\$26:\$J\$27};
2; FALSCH)

lineare Funktion =REST[18](\$B\$62 * T30 + \$E\$62; \$D\$7)

Erweiterung plus RC(1)

=REST(U30 + D30; \$D\$7)

=U31

2.8.2. round key 2

Tabelle 2.14.: Einfaches Beispiel: Rundenschlüssel für die zweite Runde (round key 2)

	M	N	O	P
	Word 0 + Erweiterung = Word 4	Word 1 + Word 4 = Word 5	Word 2 + Word 5 = Word 6	Word 3 + Word 6 = Word 7
36	4	4	4	4
37	3	3	3	3
38	3	3	3	3
39	3	3	3	3

Der Code in der zweiten Zeile Nummer 36 für Tabelle 2.14 lautet:

=REST(M30 + W30; D7)

Die weiteren Schritte zwischen round key 3 und round key 4 habe ich im Anhang dokumentiert⁸.

⁸Siehe Abschnitt B.1 auf Seite 78.

2.8.3. round key 3

Tabelle 2.15.: Einfaches Beispiel: Rundenschlüssel für die dritte Runde (round key 3)

	M	N	O	P
	Word 4 + Erweiterung	Word 8 + Word 5	Word 9 + Word 6	Word 10 + Word 7
42	28	3	7	11
43	7	10	13	16
44	7	10	13	16
45	14	17	20	23

2.8.4. round key 4

Tabelle 2.16.: Einfaches Beispiel: Rundenschlüssel für die vierte Runde (round key 4)

	M	N	O	P
	Word 8 + Word 12 Erweiterung	Word 9 + Word 13	Word 10 + Word 14	Word 11 + Word 14
48	17	20	27	9
49	12	22	6	22
50	24	5	18	5
51	12	0	20	14

2.9. Der Algorithmus

Im vorherigen Abschnitt der Key Expansion haben wir drei round keys erstellt, zusammen mit dem cipher key besitzen wir vier Schlüssel. Der Algorithmus ist wie folgt aufgebaut:

- Vorrunde
 - AddRoundKey
- n-1 Runde
 - SubBytes
 - * multiplikativ Inverse
 - * affine Funktion
 - ShiftRows
 - MixColumns
 - AddRoundKey
- Letzte Runde
 - SubBytes
 - * multiplikativ Inverse
 - * affine Funktion
 - ShiftRows
 - AddRoundKey

2.9.1. Vorrunde

In der Vorrunde wird nur die Funktion AddRoundKey ausgeführt.

AddRoundKey

Zum besseren Verständnis wird der cipher key mit angegeben.

Abbildung 2.3.: AES: AddRoundKey operation [23]

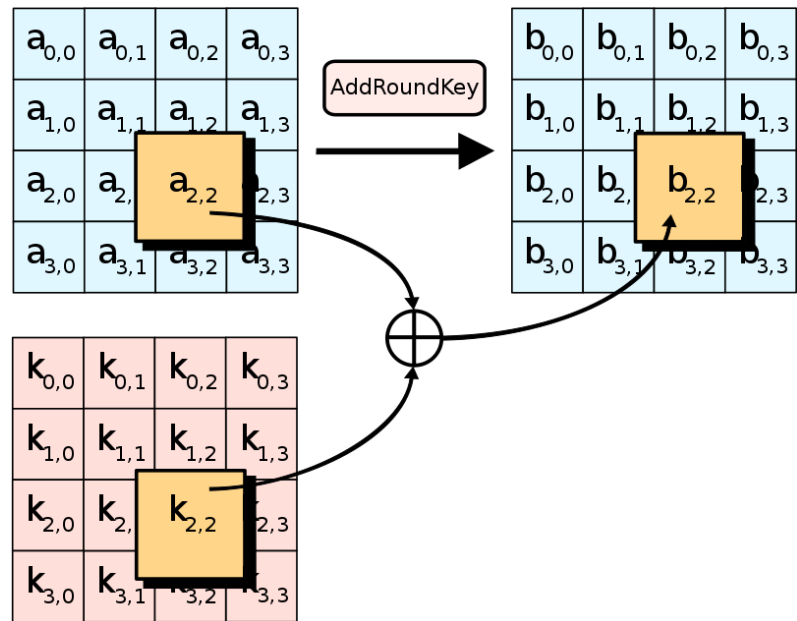


Tabelle 2.17.: Einfaches Beispiel: Vorrunde AddRoundKey

	B	C	D	E
55	19	27	27	7
56	21	9	11	28
57	19	19	12	27
58	9	20	21	27

Tabelle 2.18.: Einfaches Beispiel: Vorrunde round key 1

	G	H	I	J
55	0	0	0	0
56	0	0	0	0
57	0	0	0	0
58	0	0	0	0

Der Code für das erste Ergebnis für sowohl Tabelle 2.17 als auch für Tabelle 2.18 lautet:

```
AddRoundKey
=REST(B16 + G55; $D$7)
round key 1
=M30
```

2.9.2. Erste von „n-1“ Runden

SubBytes

Die Funktion SubBytes setzt sich aus dem Austauschen der Werte mit ihren multiplikativen Inversen und einer linearen Funktion zusammen⁹.

⁹S-Box ist eine Tabelle, welche die Ergebnisse der nacheinander ausgeführten Operationen für alle Werte enthält. Siehe Abschnitt 2.6 auf Seite 13.

Abbildung 2.4.: AES: SubBytes operation [26]

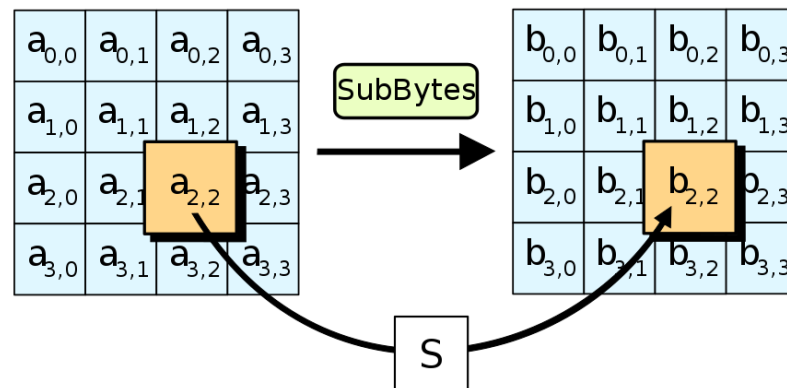


Tabelle 2.19.: Einfaches
Beispiel: erste Runde SubBytes
lineare Funktion

	A	B	C	D	E
62	y=	3	x	+	3

Tabelle 2.20.: Einfaches Beispiel: multiplikativ Inverse

	B	C	D	E	F	G	H	I	J	K
22	1	2	3	4	5	6	7	8	9	10
23	1	15	10	22	6	5	25	11	13	3
24	11	12	13	14	15	16	17	18	19	20
25	8	17	9	27	2	20	12	21	26	16
26	21	22	23	24	25	26	27	28	0	
27	18	4	24	23	7	19	14	28	0	

Tabelle 2.21.: Einfaches
Beispiel: erste Runde SubBytes
multiplikativ Inverse

	B	C	D	E
65	26	14	14	25
66	18	13	8	28
67	26	26	17	14
68	13	16	18	14

Tabelle 2.22.: Einfaches
Beispiel: erste Runde SubBytes
lineare Funktion

	G	H	I	J
64	23	16	16	20
65	28	13	27	0
66	23	23	25	16
67	13	22	28	16

Der Code für das erste Ergebnis für sowohl Tabelle 2.21 als auch für Tabelle 2.22 lautet:

SubBytes

multiplikativ Inverses =WVERWEIS(B55; {\$B\$22:\$K\$23 \ \$B\$24:\$K\$25 \ \$B\$26:\$J\$27};
2; FALSCH)

lineare Funktion =REST(\$B\$62 * B65 + \$E\$62; \$D\$7)

ShiftRows

Bei Aufruf dieser Funktion werden die Zeilen der Matrix verschoben. Die erste Zeile wird um vier Elemente verschoben, das führt zu keiner Änderung im Ergebnis zur Angabe. Das entspricht keiner Verschiebung. Die zweite Zeile wird um ein Element nach links, und das ganze linke Element wird rechts auf den freien Platz gestellt. Die dritte Zeile wird um zwei Elemente nach links verschoben und die zwei Elemente links werden rechts auf die freien Plätze gestellt. Die vierte Zeile wird um drei Elemente verschoben. Um diese Verschiebung zu erzeugen, verwende ich eine Permutationsmatrix, die zeilenweise auf die Matrix mit dem zu verschlüsselnden Text angewandt wird. Jede Zeile wird unterschiedlich oft mit der Permutationsmatrix multipliziert.

Abbildung 2.5.: AES: ShiftRows operation [25]

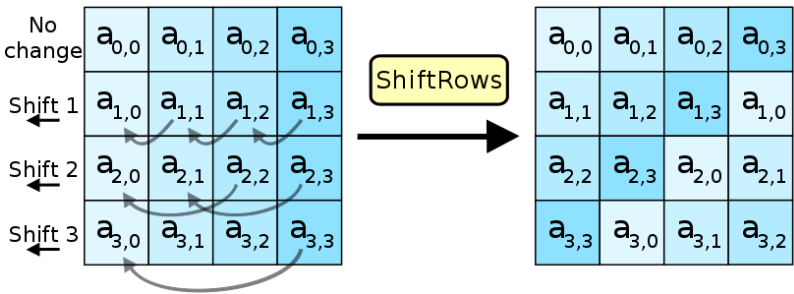


Tabelle 2.23.: Einfaches
Beispiel: erste Runde ShiftRows

	B	C	D	E
71	23	16	16	20
72	13	27	0	28
73	25	16	23	23
74	16	13	22	28

Tabelle 2.24.: Einfaches
Beispiel: erste Runde ShiftRows
Permutationsmatrix

	G	H	I	J
71	0	0	0	1
72	1	0	0	0
73	0	1	0	0
74	0	0	1	0

Der Code für das erste Ergebnis für Tabelle 2.23 lautet:

Shift Rows

=mmult(mmult(mmult(mmult(G65:J65; \$G\$71:\$J\$74); \$G\$71:\$J\$74); \$G\$71:\$J\$74); \$G\$71:\$J\$74)

MixColumns

Mix Columns ist eine einfache Matrixmultiplikation. Die Matrix muss invertierbar sein und sie soll für eine möglichst große Diffusion sorgen. Hauptaugenmerk liegt darin, dass versucht wird, die Einträge miteinander zu verbinden, damit eine Lineare Kryptoanalyse nicht erfolgreich ist. Die Verbindung der Einträge miteinander soll bewirken, dass die Änderung eines Eintrages sich am Ende der Verschlüsselung auf alle Einträge auswirkt. Die Beurteilungskriterien, ob eine Matrix für eine gute Diffusion sorgt oder nicht, werden im

folgendem behandelt¹⁰.

Ich verwende hier eine einfache Permutationsmatrix, da für das Verständnis des Algorithmus sowie für das Nachvollziehen eine einfache Matrix förderlich ist. Das Tabellenkalkulationsblatt ist aber so ausgelegt, dass jede invertierbare Matrix mit Einträgen aus $\mathbb{Z}$ verwendet werden kann, darüber mehr im Entschlüsselungsteil. Eine Permutationsmatrix ist einfach zu invertieren und ihr Inverses ist wieder eine Permutationsmatrix.

Der Grund warum eine vier $\times$ vier Matrix in AES (in Rijndael $4 \times k$) verwendet wird, liegt in der Funktion MixColumns. Es wird jede einzelne Spalte der state matrix mit der Matrix in MixColumns multipliziert. Da dieser Schritt wieder rückgängig gemacht werden muss, muss die Matrix in MixColumns invertierbar sein. Ich habe die Matrix mit D bezeichnet.

Abbildung 2.6.: AES: MixColumns operation [24]

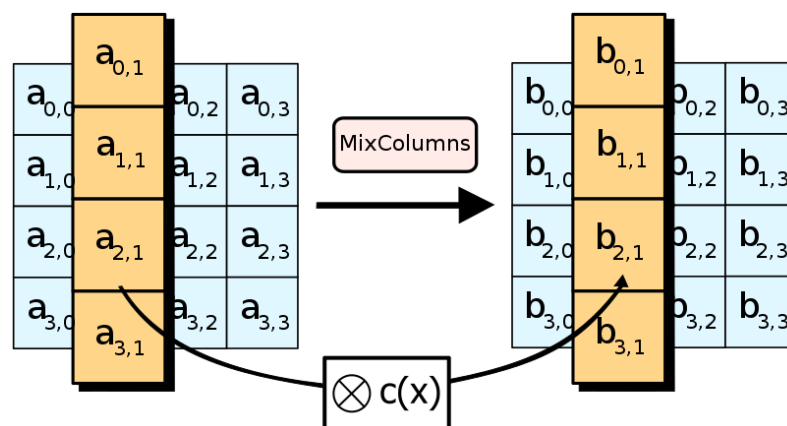


Tabelle 2.25.: Einfaches
Beispiel: erste Runde MixColumns

	B	C	D	E
77	13	27	0	28
78	16	13	22	28
79	23	16	16	20
80	25	16	23	23

Tabelle 2.26.: Einfaches
Beispiel: erste Runde MixColumns
Modulo 29

	G	H	I	J
77	13	27	0	28
78	16	13	22	28
79	23	16	16	20
80	25	16	23	23

Der Code für das erste Ergebnis für sowohl Tabelle 2.25 als auch für Tabelle 2.26 lautet:

MixColumns

```
=mmult[16]( $M$77:$P$80; B71:E74)
```

MixColumns Modulo 29

```
=REST(B77; $D$7)
```

¹⁰Die Kapitel 7, 8, 9 und 10 in [2] behandeln dieses Thema.

Tabelle 2.27.: Einfaches Beispiel: MixColumns Matrix D

	M	N	O	P
77	0	1	0	0
78	0	0	0	1
79	1	0	0	0
80	0	0	1	0

AddRoundKey

Wie schon in der Vorrunde wird ein round key zum verschlüsselten Text hinzugezählt.

Tabelle 2.28.: Einfaches
Beispiel: erste Runde AddRoundKey

	B	C	D	E
83	17	2	4	3
84	19	16	25	2
85	26	19	19	23
86	28	19	26	26

Tabelle 2.29.: Einfaches
Beispiel: erste Runde round key 2

	G	H	I	J
83	4	4	4	4
84	3	3	3	3
85	3	3	3	3
86	3	3	3	3

Der Code für das erste Ergebnis für sowohl Tabelle 2.28 als auch für Tabelle 2.29 lautet:

AddRoundKey

=REST(G77 + G83; \$D\$7)

round key 2

=M36

Ende einer „n-1“ Runde, dieses war die erste Runde die weiteren Runden habe ich in den Anhang¹¹ verschoben.

2.9.3. Ergebnis der zweiten Runde

Tabelle 2.30.: Einfaches Beispiel: zweite Runde Ergebnis

	B	C	D	E
108	4	27	26	5
109	9	10	7	18
110	17	0	24	20
111	8	5	22	17

2.9.4. Letzte Runde

Die letzte Runde unterscheidet sich von den vorhergehenden Runden dadurch, dass die Funktion MixColumns nicht aufgerufen wird.

¹¹Siehe Abschnitt B.2 auf Seite 82.

SubBytes

Tabelle 2.31.: Einfaches
Beispiel: letzte Runde SubBytes
multiplikativ Inverse

	B	C	D	E
115	22	14	19	6
116	13	3	25	21
117	12	0	23	16
118	11	6	4	12

Tabelle 2.32.: Einfaches
Beispiel: letzte Runde SubBytes
lineare Funktion

	G	H	I	J
115	11	16	2	21
116	13	12	20	8
117	10	3	14	22
118	7	21	15	10

Der Code für das erste Ergebnis für sowohl Tabelle 2.31 als auch für Tabelle 2.32 lautet:

SubBytes

multiplikativ Inverses =WVERWEIS(B108; {\$B\$22:\$K\$23 \ \$B\$24:\$K\$25 \ \$B\$26:\$J\$27};
2; FALSCH)

lineare Funktion =REST(\$B\$62 * B115 + \$E\$62; \$D\$7)

ShiftRows

Tabelle 2.33.: Einfaches Beispiel: letzte Runde ShiftRows

	B	C	D	E
121	11	16	2	21
122	12	20	8	13
123	14	22	10	3
124	10	7	21	15

Der Code für das erste Ergebnis für Tabelle 2.33 lautet:

Shift Rows

=mmult(mmult(mmult(mmult(G115:J115; \$G\$71:\$J\$74); \$G\$71:\$J\$74); \$G\$71:\$J\$74);
\$G\$71:\$J\$74)

AddRoundKey

Tabelle 2.34.: Einfaches
Beispiel: letzte Runde
AddRoundKey

	B	C	D	E
127	28	7	0	1
128	24	13	14	6
129	9	27	28	8
130	22	7	12	0

Tabelle 2.35.: Einfaches
Beispiel: letzte Runde
round key 4

	G	H	I	J
127	17	20	27	9
128	12	22	6	22
129	24	5	18	5
130	12	0	20	14

Der Code für das erste Ergebnis für sowohl Tabelle 2.34 als auch für Tabelle 2.35 lautet:

AddRoundKey

=REST(B121+G127; \$D\$7)

round key 4

=M48

2.9.5. Chiffretext

Zum Abschluss werden die Zahlen aus Tabelle 2.34 über die Tabelle 2.36 wieder zurück in die Buchstaben und Zeichen übertragen die in Tabelle 2.37 stehen.

Tabelle 2.36.: Einfaches Beispiel: Verfügbare Zeichen Rückübersetzung

	B	C	D	E	F	G	H	I	J	K
134	1	2	3	4	5	6	7	8	9	10
135	a	b	c	d	e	f	g	h	i	j
136	11	12	13	14	15	16	17	18	19	20
137	k	l	m	n	o	p	q	r	s	t
138	21	22	23	24	25	26	27	28	0	
139	u	v	w	x	y	z	LZ	PKT	BST	

Der Code für das erste Ergebnis für Tabelle 2.36 lautet:

Verfügbare Zeichen Rückübersetzung

=B9

Tabelle 2.37.: Einfaches Beispiel: Geheimtext

	B	C	D	E
141	PKT	g	BST	a
142	x	m	n	f
143	i	LZ	PKT	h
144	v	g	l	BST

Der Code für das erste Ergebnis für Tabelle 2.37 lautet:

Verfügbare Zeichen Rückübersetzung

=WVERWEIS(B127; {\$B\$134:\$K\$135\ \$B\$136:\$K\$137\ \$B\$138:\$J\$139}; 2; FALSCH)

2.10. Entschlüsselung

Die Entschlüsselung verläuft genauso wie die Verschlüsselung, bloß in umgekehrter Reihenfolge.

Der Algorithmus ist wie folgt aufgebaut:

- InvLetzte Runde
 - InvAddRoundKey
 - InvShiftRows
 - InvSubBytes
 - * affine Funktion
 - * multiplikativ Inverse
- n-1 InvRunde
 - InvAddRoundKey
 - InvMixColumns
 - InvShiftRows
 - SubBytes
 - * affine Funktion
 - * multiplikativ Inverse
- InvVorrunde
 - InvAddRoundKey

Tabelle 2.38.: Einfaches Beispiel: Entschlüsselung Geheimtext

	B	C	D	E
2	PKT	g	BST	a
3	x	m	n	f
4	i	LZ	PKT	h
5	v	g	l	BST

Der Code für das erste Ergebnis für Tabelle 2.38 lautet:

Geheimtext

=/Verschlüsselung!/B141

Tabelle 2.39.: Einfaches Beispiel: Entschlüsselung verfügbare Zeichen

	B	C	D	E	F	G	H	I	J	K
8	a	b	c	d	e	f	g	h	i	j
9	1	2	3	4	5	6	7	8	9	10
10	k	l	m	n	o	p	q	r	s	t
11	11	12	13	14	15	16	17	18	19	20
12	u	v	w	x	y	z	LZ	PKT	BST	
13	21	22	23	24	25	26	27	28	0	

Der Code für das erste Ergebnis für Tabelle 2.39 lautet:

verfügbare Zeichen

=/Verschlüsselung!/B8

Zu Beginn werden die Zeichen aus Tabelle 2.38 über die Tabelle 2.39 wieder zurück in die Zahlen übertragen die in Tabelle 2.40 stehen.

Tabelle 2.40.: Einfaches Beispiel: Entschlüsselung Übertragung

	B	C	D	E
16	28	7	0	1
17	24	13	14	6
18	9	27	28	8
19	22	7	12	0

Der Code für das erste Ergebnis für Tabelle 2.40 lautet:

Übertragung
=WVERWEIS(B2; {\$B\$8:\$K\$9\ \$B\$10:\$K\$11\ \$B\$12:\$J\$13}; 2; FALSCH)

2.10.1. InvLetzte Runde

InvAddRoundKey

Vom Geheimtext wird der round key 4 abgezogen.

Tabelle 2.41.: Einfaches
Beispiel: Entschlüsselung
letzte Runde round key 4

	B	C	D	E
23	17	20	27	9
24	12	22	6	22
25	24	5	18	5
26	12	0	20	14

Tabelle 2.42.: Einfaches
Beispiel: Entschlüsselung
letzte Runde InvAddRoundKey

	G	H	I	J
23	11	16	2	21
24	12	20	8	13
25	14	22	10	3
26	10	7	21	15

Der Code für das erste Ergebnis für sowohl Tabelle 2.41 als auch für Tabelle 2.42 lautet:

round key 4
='Verschlüsselung'!M48
InvAddRoundKey
=REST(B16 - B23; 'Verschlüsselung'!\$D\$7)

InvShiftRows

Die Funktion InvShiftRows kann auf zwei Arten gelöst werden. Entweder man verschiebt die Zeilen weiter nach links bis die Gesamtverschiebung von ShiftRows und InvShiftRows vier ergibt, oder man verschiebt die Zeilen um dieselben Werte nach rechts, um die sie in ShiftRows nach links verschoben wurden.

Tabelle 2.43.: Einfaches Beispiel: Entschlüsselung letzte Runde InvShiftRows

	B	C	D	E
28	11	16	2	21
29	13	12	20	8
30	10	3	14	22
31	7	21	15	10

Der Code für das erste Ergebnis für Tabelle 2.43 lautet:

InvShift Rows

```
=mmult( mmult( mmult( mmult(G23:J23; 'Verschlüsselung'!$G$71:$J$74);
'Verschlüsselung'!$G$71:$J$74); 'Verschlüsselung'!$G$71:$J$74); 'Verschlüsselung'!$G$71:$J$74)
```

InvSubBytes

Der erste Teil ist die Umkehrung der linearen Funktion. Die Umkehrfunktion von $y = 3x + 3$ ist $x = (y - 3)/3$. Es kann diese Funktion verwendet werden und bei den sich ergebenden Bruchzahlen würde man die Restklasse ermitteln oder die Funktion wird so umgestaltet, dass nur Zahlen aus $\mathbb{Z}$ als Ergebnis in Frage kommen. Die multiplikative Inverse von 3 ist 10. 10 entspricht somit $\frac{1}{3}$ modulo 29. Die additive Inverse, hier müssen die beiden Zahlen zusammen sich auf das neutrale Element ergänzen, von -3 modulo 29 ist 26. Daher kann diese lineare Funktion auch geschrieben werden als $y = (x + 26) * 10$ modulo 29.

Der zweite Teil der Funktion SubBytes verläuft gleich zu SubBytes, weil die Tafel des Multiplikativen Inversen selbst invers ist.

Tabelle 2.44.: Inverse lineare Funktion

	F	G	H	I	J	K
34	x=	10	(y	+	26	)

Der Code für das erste Ergebnis für Tabelle 2.44 lautet:

Inverse lineare Funktion

```
=WVERWEIS('Verschlüsselung'!B62; {'Verschlüsselung'!B22:$K$23\
'Verschlüsselung'!B24:$K$25\ 'Verschlüsselung'!B26:$J$27}; 2; FALSCH)
```

Tabelle 2.45.: Einfaches Beispiel: Entschlüsselung letzte Runde SubBytes multiplikativ Inverse

	B	C	D	E
115	22	14	19	6
116	13	3	25	21
117	12	0	23	16
118	11	6	4	12

Tabelle 2.46.: Einfaches Beispiel: Entschlüsselung letzte Runde SubBytes lineare Funktion

	G	H	I	J
115	11	16	2	21
116	13	12	20	8
117	10	3	14	22
118	7	21	15	10

Der Code für das erste Ergebnis für sowohl Tabelle 2.46 als auch für Tabelle 2.45 lautet:

InvSubBytes

lineare Funktion =REST((B28 + \$J\$34) * \$G\$34; 'Verschlüsselung'!\$D\$7)

multiplikativ Inverses =WVERWEIS(B37; {'Verschlüsselung'!\$B\$22:\$K\$23\
'Verschlüsselung'!\$B\$24:\$K\$25\
'Verschlüsselung'!\$B\$26:\$J\$27}; 2; FALSCH)

Wie in Abschnitt 2.6 auf Seite 13 können die beiden Operation zusammengefasst werden zu einer inversen S-Box.

Tabelle 2.47.: Einfaches Beispiel: Entschlüsselung inverse S-Box

1	2	3	4	5	7	8	9
13	26	0	3	16	8	18	15
12	13	14	15	16	17	18	19
10	9	24	22	27	23	6	2
22	23	24	25	26	27	28	
20	19	25	12	14	11	21	

Dies ist das Ende der inversen letzten Runde, die weiteren Schritte habe ich in den Anhang verschoben¹². Einzige die Funktion InvMixColumns wurde bis jetzt noch nicht besprochen.

InvMixColumns

Die Funktion InvMixColumns erfordert etwas Aufmerksamkeit. Die Inversematrix der Mix-Columnsmatrix kann auch aufgefasst werden als eine Matrix, die aus den Spaltenvektoren zusammengestellt worden ist. Die Spaltenvektoren sind Ergebnisse der Cramerschen Regel¹³. Als erstes wird die Inverse Matrix mit ihrer Determinanten multipliziert, um Elemente aus $\mathbb{Z}$ zu erhalten. Diese Rechenoperation wird rückgängig gemacht durch das Multiplizieren der multiplikativen Inversen der Determinante aus Modulo¹⁴ 29. Dadurch bleiben alle Elemente aus $\mathbb{Z}$. Eine Schwierigkeit war die dynamische Bestimmung der multiplikativ Inversen bei negativen Determinanten im Tabellenblatt. Die Lösung des Problems sehen sie unten und somit können in MixColumns alle Werte aus $\mathbb{Z}$ eingetragen werden.

Tabelle 2.48.: Matrix D, Determinante und Inverse Matrix $D_{Basis10}$

	M	N	O	P
77	0	1	0	0
78	0	0	0	1
79	1	0	0	0
80	0	0	1	0

	R
77	-1

	T	U	V	W
77	0	0	1	0
78	1	0	0	0
79	0	0	0	1
80	0	1	0	0

Determinante

=MDET[14](M77:P80)

Inverse Matrix $D_{Basis10}$

=MINV[15](M77:P80)

¹²Siehe Abschnitt B.3 auf Seite 84.

¹³Siehe Theorem 9 auf Seite 9.

¹⁴Siehe Corollary 1 auf Seite 8.

Tabelle 2.49.: Inverse Matrix D mal Determinate mit verschiedenen Werten und multiplikativ Inverses der Determinate

	B	C	D	E		O	P		M	L
49	0	0	-1	0	49	Vorzeichen	-1	49	Vorzeichen mal	-1
50	-1	0	0	0	50	Betrag	1		der multiplikati-	
51	0	0	0	-1	51	Modulo 29	1		ven Inversen der	
52	0	-1	0	0	52	multiplikativ	1		Determinate	
						Inverses				

Inverse Matrix D mal Determinate

= 'Verschlüsselung'!T77 * 'Verschlüsselung'!R\$77

verschiedene Werte

=VORZEICHEN[21](P48)

=Abs[6](P48)

=REST(P50; 'Verschlüsselung'!D7)

=WVERWEIS(P51; {'Verschlüsselung'!B\$22:\$K\$23\ 'Verschlüsselung'!B\$24:\$K\$25\ 'Verschlüsselung'!B\$26:\$J\$27}; 2; FALSCH)

multiplikativ Inverses der Determinate

=P53 * P49

Tabelle 2.50.: Einfaches Beispiel: Entschlüsselung inverse Matrix $D_{\text{Restklassenkörper } 29}$

	G	H	I	J
49	0	0	1	0
50	1	0	0	0
51	0	0	0	1
52	0	1	0	0

inverse Matrix $D_{\text{Restklassenkörper } 29}$

=REST(B49 * \$M\$49; 'Verschlüsselung'!\$D\$7)

3. Logische Gatter

Es gibt nur sieben verschiedene Gatter AND, OR und XOR, NOT alleine, sowie die ersten drei verneinten NAND, NOR und XNOR¹.

Tabelle 3.1.:
Logisches Gatter:
AND

A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

Abbildung 3.1.:
Logisches Gatter:
AND [27]

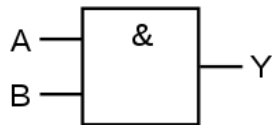


Tabelle 3.2.:
Logisches Gatter:
NAND

A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

Abbildung 3.2.:
Logisches Gatter:
NAND [28]

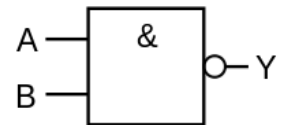


Tabelle 3.3.:
Logisches Gatter:
OR

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

Abbildung 3.3.:
Logisches Gatter:
OR [31]

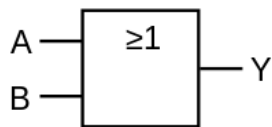


Tabelle 3.4.:
Logisches Gatter:
NOR

A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0

Abbildung 3.4.:
Logisches Gatter:
NOR [29]

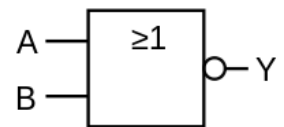


Tabelle 3.5.:
Logisches Gatter:
XOR

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

Abbildung 3.5.:
Logisches Gatter:
XOR [33]

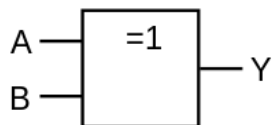
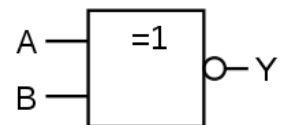


Tabelle 3.6.:
Logisches Gatter:
XNOR

A	B	Y
0	0	1
0	1	0
1	0	0
1	1	1

Abbildung 3.6.:
Logisches Gatter:
NAND [32]

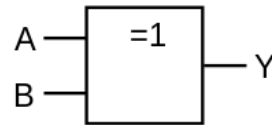


¹Abbildungen als Symbole in Schaltplänen in Verwendung nach IEC 60617-12: 1997 & ANSI/IEEE Std 91/91a-1991

Tabelle 3.7.:
Logisches Gatter:
NOT

A	Y
0	1
1	0

Abbildung 3.7.:
Logisches Gatter:
NOT [30]



Von diesen drei Gattern und ihren Verneinungen weist das XOR und XNOR die Besonderheit auf, dass egal wie man die Vektoren vertauscht, bei gleichbleibender Reihenfolge der Zeilenbeschriftung die Aussage noch immer stimmt. Diese Gatter werden im Buch „The design of Rijndael“ [2] als sich selbst invers bezeichnet.

Tabelle 3.8.: Beispiel Selbstinversität von XOR

1010		A		1001
0011		xor B		0011
1001		Y		1010

4. AES Algorithmus

4.1. 8 Bits - Die verfügbaren Zeichen

AES wurde ausgelegt um mit 8 Bits zu arbeiten. Der Grund dafür liegt in dem Umstand, dass der Algorithmus sowohl auf Handys lauffähig sein sollte als auch auf anderen Systemen, die nur über einen 8 Bit Prozessor verfügen. Die Systeme könnten mit größeren Zeichenketten umgehen, würden dadurch aber unnötig verlangsamt werden. Durch die Beschränkung können diese Systeme in einem Takt ein Feld der Matrix bearbeitet. Die verfügbaren Zeichen sind auf $2^8 = 256$ Zeichen beschränkt.

4.2. Der Klartext

Wie wir es im Beispiel haben, gibt es eine Matrix, die aus vier mal vier Felder besteht. Jedes dieser Felder steht für 8 Bits. Diese acht Bits werden der einfacheren Schreibweise wegen oft in Hexadezimal angegeben.

Der Klartext ist hier ein Array von 128 Bits (8 Bit x 4 rows x 4 blocklength) Länge. Dieses Array kann als eindimensionales Array oder als Matrix ausgeführt werden.

Auf eine Übertragung mittels Äquivalenzrelation wie beim einfachen Beispiel¹ können wir verzichten, die Matrix wird wie dort spaltenweise von links nach rechts befüllt.

Unsere Systeme auf Standrechner und Laptops sind vielleicht noch 32 Bit Systeme oder schon 64 Bit Systeme. Ein 64 Bit Zeichen müsste mittels zwei AES Blöcke verschlüsselt werden.

4.2.1. Key Schedule - Die Schlüsselfestlegung

Die N_b ist in AES fixiert auf vier Spalten. Die vier Zeilen der state matrix sind bedingt durch die Funktion MixColumns. N_k ist die Länge des Schlüssels. Die Länge des Schlüssels beeinflusst die Rundenanzahl. In AES wurde die Auswahl auf drei Möglichkeiten beschränkt.

Die Menge an Information, die in die Matrix geschrieben und in einem Durchgang verschlüsselt wird, ist somit abhängig von der Länge des Blocks. Die Anzahl der Zeilen ist fix, deswegen bestimmt die Blocklänge die Größe der Matrix. Falls eine größere wie Blocklängen verwendet würde, was im AES nicht aber in Rijndael geht. Dann werden im Rijndael Spalten an die state matrix angehängt.

Die Schlüssellänge bestimmt die Anzahl der Runden und soll auch bei einem leeren Schlüssel für eine gute Verschlüsselung sorgen.

¹Siehe Abschnitt [2.1.1](#) auf Seite 10.

Tabelle 4.1.: AES: Anzahl der Verschlüsselungsrunden N_r

		N_b
		4
N_k	4	10
	6	12
	8	14

In Falle eines 128 Bit Schlüssels müssten 10 round keys berechnet werden, bei einem 192 Bit Schlüssel wären es 12 und bei einem 256 Bit langen Schlüssel wären es 14 round keys. Wenn ein 192 Bit langer Schlüssel gewählt wird, und die Eingabe zu wenige Bits enthält, wird der Rest des cipher keys mit 0 aufgefüllt bis die 192 Bit erreicht werden. Hier ist die Tabelle für den Rijndael Algorithmus

Tabelle 4.2.: Rijndael: Anzahl der Verschlüsselungsrunden N_r

		N_b				
		4	5	6	7	8
N_k	4	10	11	12	13	14
	5	11	11	12	13	14
	6	12	12	12	13	14
	7	13	13	13	13	14
	8	14	14	14	14	14

4.2.2. Key Expansion

Wie wir schon im einfachen Beispiel gesehen haben, gibt es eine Vorrunde, n-1 Runden und eine Abschlussrunde. Es werden abhängig von der Schlüssellänge noch 10, 12 oder 14 Schlüssel benötigt. Die Schlüsselerweiterung bei einem 128 oder 192 Bit langen Schlüssel verläuft gleich. Bei einem 256 Bit langen Schlüssel gibt es eine geringe Variation.

4.2.3. RC round constant (Rundenkonstante)

Die ersten acht round constant sind noch leicht bestimmbar. Sie wären
 RC 1:0000 0001 RC 2:0000 0010 RC 3:0000 0100 RC 4:0000 1000
 RC 5:0001 0000 RC 6:0010 0000 RC 7:0100 0000 RC 8:1000 0000
 Doch wie wird RC 9 bestimmt?

4.2.4. Galoisfeld

Acht ist keine Primzahl daher kann der Trick der im einfachen Beispiel angewendet wurde, nicht angewendet werden. Es muss ein Galoisfeld verwendet werden. Als erstes ändert sich die Betrachtungsweise der dargestellten Zahlen. Die 8 Bits werden als Polynom vom maximalen Grad sieben aufgefasst. Die oben gezeigten round constant würden nun so aufgeschrieben werden.

$$\begin{aligned}
\text{RC 1: } &0x^7 + 0x^6 + 0x^5 + 0x^4 + 0x^3 + 0x^2 + 0x^1 + 1x^0 = 1 \\
\text{RC 2: } &0x^7 + 0x^6 + 0x^5 + 0x^4 + 0x^3 + 0x^2 + 1x^1 + 0x^0 = x^1 \\
\text{RC 3: } &0x^7 + 0x^6 + 0x^5 + 0x^4 + 0x^3 + 1x^2 + 0x^1 + 0x^0 = x^2 \\
\text{RC 4: } &0x^7 + 0x^6 + 0x^5 + 0x^4 + 1x^3 + 0x^2 + 0x^1 + 0x^0 = x^3 \\
\text{RC 5: } &0x^7 + 0x^6 + 0x^5 + 1x^4 + 0x^3 + 0x^2 + 0x^1 + 0x^0 = x^4 \\
\text{RC 6: } &0x^7 + 0x^6 + 1x^5 + 0x^4 + 0x^3 + 0x^2 + 0x^1 + 0x^0 = x^5 \\
\text{RC 7: } &0x^7 + 1x^6 + 0x^5 + 0x^4 + 0x^3 + 0x^2 + 0x^1 + 0x^0 = x^6 \\
\text{RC 8: } &1x^7 + 0x^6 + 0x^5 + 0x^4 + 0x^3 + 0x^2 + 0x^1 + 0x^0 = x^7
\end{aligned}$$

Nach dem Fundamentalsatz der Algebra² wissen wir, dass wir ein Polynom in seine Nullstellen zerlegen können. Die Faktoren des Polynoms können als Primzahlen aufgefasst werden.

In Galoisfeld oben, haben die Koeffizienten die Werte 0 oder 1. Das hängt mit der Primzahl 2 zusammen, durch die das Galoisfeld erzeugt wurde. Als ein „Primzahlpolynom“ spricht ein reduzierendes Polynom kommen 1, x, x+1 usw. in Frage, weil es kein anderes Polynom gibt, welches dieses Polynom teilt. $x^2 + 1$ ist kein reduzierendes Polynom, weil es das Produkt von $(x + 1)$ mal $(x + 1)$ ist.

$$(x + 1) * (x + 1) = (x^2 + 2x + 1) := (x^2 + 1) \pmod{2}$$

Als reduzierendes Polynom wird $1x^8 + 0x^7 + 0x^6 + 0x^5 + 1x^4 + 1x^3 + 0x^2 + 1x^1 + 1x^0 = 1x^8 + 1x^4 + 1x^3 + 1x^1 + 1$ verwendet.

$$\text{RC 9: } 1x^8 + 0x^7 + 0x^6 + 0x^5 + 0x^4 + 0x^3 + 0x^2 + 0x^1 + 0x^0 = x^8$$

$$\begin{array}{rcl}
1x^8 + 0x^7 + 0x^6 + 0x^5 + 0x^4 + 0x^3 + 0x^2 + 0x^1 + 0x^0 & : & 1x^8 + 1x^4 + 1x^3 + 1x^1 + 1 = 1 \\
1x^8 + 0x^7 + 0x^6 + 0x^5 + 1x^4 + 1x^3 + 0x^2 + 1x^1 + 1x^0 & & \\
\hline
2x^8 + 0x^7 + 0x^6 + 0x^5 - 1x^4 - 1x^3 + 0x^2 - 1x^1 - 1x^0 & \text{Rest} & \\
:= 0x^7 + 0x^6 + 0x^5 + 1x^4 + 1x^3 + 0x^2 + 1x^1 + 1x^0 & \pmod{2} &
\end{array}$$

$$\begin{aligned}
\text{RC 9: } &0x^7 + 0x^6 + 0x^5 + 1x^4 + 1x^3 + 0x^2 + 1x^1 + 1x^0 = 1x^4 + 1x^3 + 1x^1 + 1 \\
\text{RC 10: } &0x^7 + 0x^6 + 1x^5 + 1x^4 + 0x^3 + 1x^2 + 1x^1 + 0x^0 = 1x^5 + 1x^4 + 1x^2 + 1x^1 \\
\text{RC 11: } &0x^7 + 1x^6 + 1x^5 + 0x^4 + 1x^3 + 1x^2 + 0x^1 + 0x^0 = 1x^6 + 1x^5 + 1x^3 + 1x^2 \\
\text{RC 12: } &1x^7 + 1x^6 + 0x^5 + 1x^4 + 1x^3 + 0x^2 + 0x^1 + 0x^0 = 1x^7 + 1x^6 + 1x^4 + 1x^3
\end{aligned}$$

Bei der Berechnung wurde modulo 2 eingesetzt um die einzelnen Koeffizienten des Polynoms in eine passende Form zu bringen. Das Ergebnis könnte auch bestimmt werden, wenn die Polynome in ihrer Bit Dargestellung durch die XOR Operation verknüpft werden.

Die RC 13 und RC 14 ist in der Beispielsammlung zum Üben der Berechnung.

4.2.5. Shift Rows

In Shift Rows wird das Polynom mit x multipliziert, oder anders ausgedrückt, in der Bitdarstellung des Polynoms werden die Bits eins nach links gerückt und rechts wird eine Null hingeschrieben. Falls das Polynom nach der Operation einen Grad von acht hat oder mehr, wird es reduziert durch das oben genannte reduzierende Polynom $1x^8 + 1x^4 + 1x^3 + 1x^1 + 1$.

In Rijndael kann die state matrix die Form $4 \times k$ haben. Um eine gute Vermischung zu haben, werden die Zeilen C_0 , C_1 , C_2 und C_3 bei einer größeren state matrix stärker verschoben³.

²Siehe Theorem 8 auf Seite 9.

³Siehe in [2] Seite 38.

Tabelle 4.3.: Rijndael: ShiftRows in Abhängigkeit von N_b

N_b	C_0	C_1	C_2	C_3
4	0	1	2	3
5	0	1	2	3
6	0	1	2	3
7	0	1	2	4
8	0	1	3	4

4.2.6. SubBytes

Der erste Teil ist die bekannte Ersetzung durch das Inverse-Element bezüglich des reduzierenden Polynoms $1x^8 + 1x^4 + 1x^3 + 1x^1 + 1$. In jedem Feld der state matrix steht ein Byte mit acht Bits, die als Polynom aufgefasst werden.

Beispiele für den ersten Teil

Null hat Null als Inverses und Eins hat Eins als Inverses. Von x^1 ⁴ bestimmen wir das Inverse mit Hilfe des erweiterten Euklidischen Algorithmus. $ggT(1x^8 + 1x^4 + 1x^3 + 1x^1 + 1; x^1)$ Ich verwende die Bit-Darstellung in den weiteren Berechnungen.

$$\begin{array}{r}
 1\ 0001\ 1011 \div 10 = 1000\ 1101 \\
 \underline{1\ 0000\ 0000} \\
 0\ 0001\ 1011 \\
 \underline{1\ 0000} \\
 0\ 1011 \\
 \underline{1000} \\
 11 \\
 \underline{10} \\
 1\ \text{Rest}
 \end{array}$$

Das Inverse von x^1 ist $1x^7 + 1x^3 + 1x^2 + 1$.

Als anderes Beispiel wird das Inversepolynom zu $1x^7 + 1x^6 + 1x^5 + 1x^4 + 1x^3 + 1x^2 + 1x^1 + 1x^0 + 1$ berechnet. $GgT(1111\ 1111; 1\ 0001\ 1011)$

$$\begin{array}{r}
 1\ 0001\ 1011 \div 1111\ 1111 = 10 \\
 \underline{1\ 1111\ 1110} \\
 0\ 1110\ 0101\ \text{Rest} \\
 \Rightarrow 1\ 0001\ 1011 - 1111\ 1111 \times 10 = 1110\ 0101
 \end{array}$$

$$\begin{array}{r}
 1111\ 1111 \div 1110\ 0101 = 1 \\
 \underline{1110\ 0101} \\
 0001\ 1010\ \text{Rest} \\
 \Rightarrow 1111\ 1111 - 1110\ 0101 \times 1 = 1\ 1010
 \end{array}$$

⁴Dieses Polynom entspricht dem Zahlenwert 2.

$$\begin{array}{r}
1110\,0101 \div 1\,1010 = 1010 \\
1101\,0000 \\
\hline
0011\,0101 \\
11\,0100 \\
\hline
00\,0001 \text{ Rest} \\
\Rightarrow 1110\,0101 - 1\,1010 \times 1010 = 1
\end{array}$$

$$1110\,0101 - 1\,1010 \times 1010 = 1$$

Wir setzen für 1010, die Folgerung $1111\,1111 - 1110\,0101 \times 1 = 1\,1010$ ein.

$$\begin{aligned}
1110\,0101 - (1111\,1111 - 1110\,0101 \times 1) \times 1010 &= 1 \\
1110\,0101 - (1111\,1111 \times 1010 - 1110\,0101 \times 1010) &= 1 \\
1110\,0101 - 1111\,1111 \times 1010 + 1110\,0101 \times 1010 &= 1 \\
-1111\,1111 \times 1010 + 1110\,0101 \times 1011 &= 1
\end{aligned}$$

Wir setzen für 1110010, die Folgerung $1\,0001\,1011 - 1111\,1111 \times 10 = 1110\,0101$ ein.

$$\begin{aligned}
-1111\,1111 \times 1010 + (1\,0001\,1011 - 1111\,1111 \times 10) \times 1011 &= 1 \\
-1111\,1111 \times 1010 + 1\,0001\,1011 \times 1011 - 1111\,1111 \times 10 \times 1011 &= 1 \\
1\,0001\,1011 \times 1011 - 1111\,1111 \times (1\,0110 - 1010) &= 1 \\
1\,0001\,1011 \times 1011 - 1111\,1111 \times 1\,1100 &= 1
\end{aligned}$$

Die Inverse von $1x^7 + 1x^6 + 1x^5 + 1x^4 + 1x^3 + 1x^2 + 1x^1 + 1x^0 + 1$ ist $1x^4 + 1x^3 + 1x^2$. Die Berechnung der Inversen lässt sich schneller vorantreiben mit XOR.

$$\begin{array}{r}
1\,0001\,1011 \div 1111\,1111 = 11 \\
1\,1111\,1110 \\
\hline
0\,1110\,0101 \\
1111\,1111 \\
\hline
0001\,1010 \text{ Rest} \\
\Rightarrow 1\,0001\,1011 - 1111\,1111 \times 11 = 1\,1010
\end{array}$$

$$\begin{array}{r}
1111\,1111 \div 1\,1010 = 1011 \\
1101\,0000 \\
\hline
0010\,1111 \\
11\,0100 \\
\hline
01\,1011 \\
1\,1010 \\
\hline
1 \text{ Rest} \\
\Rightarrow 1111\,1111 - 1\,1010 \times 1011 = 1
\end{array}$$

$$11111111 - 11010 \times 1011 = 1$$

Wir setzen für 1 1010, die Folgerung $1\,0001\,1011 - 1111\,1111 \times 11 = 1\,1010$ ein.

$$\begin{aligned}
11111111 - (100011011 - 11111111 \times 11) \times 1011 &= 1 \\
11111111 - (100011011 \times 1011 - 11111111 \times 11 \times 1011) &= 1 \\
11111111 - 100011011 \times 1011 + 11111111 \times 11101 &= 1 \\
-100011011 \times 1011 + 11111111 \times 11100 &= 1
\end{aligned}$$

Das Ergebnis kann mit einer Tabelle kontrolliert werden. Dafür müssen die 8 Bits in hexadezimal geschrieben werden. In den Beispielen wurde die Inverse von $x^1 = (02)_{16}$ gesucht. Der X Wert beträgt 0 und der Y Wert 2. In der Tabelle 4.4 steht der Eintrag $(8D)_{16} = 1x^7 + 1x^3 + 1x^2 + 1$. Für $1x^7 + 1x^6 + 1x^5 + 1x^4 + 1x^3 + 1x^2 + 1x^1 + 1x^0 + 1 = (FF)_{16}$ findet sich als Inverse $(1C)_{16} = 1x^4 + 1x^3 + 1x^2$.

Tabelle 4.4.: Rijndael: Tabelle für die multiplikativ Inverse

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

lineare Funktion

Der zweite Teil ist eine lineare Funktion dieser Gestalt:

Tabelle 4.5.: Rijndael: lineare Funktion

$$\begin{pmatrix} y^7 \\ y^6 \\ y^5 \\ y^4 \\ y^3 \\ y^2 \\ y^1 \\ y^0 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} x^7 \\ x^6 \\ x^5 \\ x^4 \\ x^3 \\ x^2 \\ x^1 \\ x^0 \end{pmatrix} + \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \end{pmatrix}$$

Beispiele für den zweiten Teil

Sei $(10001101)^T$ der X Vektor mal der Matrix ergibt $(22234322)^T$ modulo 2 ist $(00010010)^T$. Nun wird der Vektor $(01100011)^T$ addiert, dies ergibt $(01110111)^T$. Das-selbe Ergebnis wird auch erzielt, wenn statt „+“ XOR verwendet wird.

Das Ergebnis kann mit einer Tabelle kontrolliert werden. Dafür müssen die 8 Bits in hexadezimal geschrieben werden. Im Beispiel wurde das Ergebnis beim Startwert $1x^7 +$

$1x^3 + 1x^2 + 1 = (1000\ 1101)_{02} = (8D)_{16}$ gesucht. Der X Wert beträgt 8 und der Y Wert D.
In Tabelle 4.6 steht an dieser Stelle der Eintrag $(77)_{16} = (0111\ 0111) = 1x^6 + 1x^5 + 1x^4 + 1x^2 + 1x^1 + 1$.

Tabelle 4.6.: AES: Tabelle für die lineare Funktion

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

Aus $(0000\ 0010)_2 = (02)_{16}$ wird nach dem Funktionsaufruf SubBytes $(0111\ 0111)_2 = (77)_{16}$ und aus $(1111\ 1111)_2 = (FF)_{16}$ wird $(0001\ 0110)_2 = (16)_{16}$. Weil die Anzahl der unterschiedlichen Felder auf 256 begrenzt sind, lassen sich alle Möglichkeiten in einer Tabelle zusammenfassen, die sogenannte S-Box wie sie in Tabelle 4.7 zu sehen ist.

Tabelle 4.7.: AES: S-Box

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

Verschiedene Schlüssellängen

Wie bei Key Schedule⁵ erwähnt, gibt es verschiedene Schlüssellängen bei AES.

Tabelle 4.8.: AES: Anzahl der Verschlüsselungsrunden N_r

		N_b
		4
N_k	4	10
	6	12
	8	14

1. N_b (Number of Blocks) ist fix bei vier und entspricht den Spalten der Matrix mit Text, die verschlüsselt werden.
2. N_k (Number of Key) ist die Anzahl der Spalten, die variabel sind.

Bei N_4 und N_6 gibt es im Algorithmus keinen Unterschied. Bei N_8 wird eine Irritation beim Erzeugen der round keys eingebaut, weil sonst ein round key N_8 schwächer wäre im Vergleich zu N_4 und N_6 . Die Irritation beseitigt Symmetrieffekte. Für $N_k \leq 6$ ist die Arbeitsweise vom einfachen Beispiel bekannt und wir können daher den ersten, zweiten und dritten round key aufstellen.

⁵Siehe Abschnitt 4.2.1 auf Seite 34.

4.2.7. round key bei N_4

cipher key

Die Erstellung des round key⁶ beim einfachen Beispiel erfolgte genauso wie jetzt bei AES. Die Funktion Key Expansion läuft nach folgenden Schritten ab:

- 1. Letzte Spalte des vorherigen round key (Rundenschlüssels) wird genommen (Anmerkung: Vier Blöcke werden oft mit „word“ referenziert.)
- 2. Shift row: Auf die transponierte Spalte wird die Funktion Shift row angewandt.
- 3. SubBytes: In der Spalte wird jedes Element durch seine multiplikativ Inverse ersetzt, anschließend wird die lineare Funktion angewandt.
- 4. Zum ersten Eintrag wird der entsprechende RC dazu gezählt.

Tabelle 4.9.: AES: cipher key bei N_4

	Word 0	Word 1	Word 2	Word 3
Feld 1	00	00	00	00
Feld 2	00	00	00	00
Feld 3	00	00	00	00
Feld 4	00	00	00	00

Tabelle 4.10.: AES: ShiftRow, SubBytes, RC 1 und Erweiterung

ShiftRow	SubBytes S-Box	RC 1	Erweiterung
00	63	0000 0001	62
00	63	0110 0011	63
00	63	XOR	63
00	63	0110 0010	63

In Tabelle 4.10 wurden in der Spalte ShiftRow die Felder um eins verschoben. In der Spalte SubBytes wurden die Felder aus ShiftRow mittels der S-Box getauscht. In RC wurde im ersten Feld die round constante mittels XOR dazugegeben, das Ergebnis steht in der Spalte Erweiterung.

Tabelle 4.11.: AES: Berechnung des 4. Word in N_4

Word 0		Erweiterung		Word 4
00		62		62
00	XOR	63	=	63
00		63		63
00		63		63

Tabelle 4.12.: AES: Berechnung des 5. Word in N_4

Word 1		Word 4		Word 5
00		62		62
00	XOR	63	=	63
00		63		63
00		63		63

⁶Siehe Abschnitt 2.8.1 auf Seite 16.

Tabelle 4.13.: AES: Berechnung des 6. Word in N_4

Word 2		Word 5		Word 6
00		62		62
00	XOR	63	=	63
00		63		63
00		63		63

Tabelle 4.14.: AES: Berechnung des 7. Word in N_4

Word 3		Word 6		Word 7
00		62		62
00	XOR	63	=	63
00		63		63
00		63		63

In Tabelle 4.11, Tabelle 4.12, Tabelle 4.13 und Tabelle 4.14 werden die weiteren Spalten der Key Expansion berechnet. Um die Rechnung nachzuvollziehen zu können, muss sie bitweise ausgeführt werden.

round key 1

Tabelle 4.15.: AES: round key 1 bei N_4

	Word 4	Word 5	Word 6	Word 7
Feld 1	62	62	62	62
Feld 2	63	63	63	63
Feld 3	63	63	63	63
Feld 4	63	63	63	63

Tabelle 4.16.: AES: ShiftRow, SubBytes, RC 2 und Erweiterung

ShiftRow	SubBytes S-Box	RC 2	Erweiterung
63	FB	0000 0010	F9
63	FB	1111 1011	FB
63	FB	XOR	FB
62	AA	1111 1001	AA

In Tabelle 4.16 wurden in der Spalte ShiftRow die Felder um eins verschoben. In der Spalte SubBytes wurden die Felder aus ShiftRow mittels der S-Box getauscht. In RC wurde im ersten Feld die round constante mittels XOR dazugegeben, das Ergebnis steht in der Spalte Erweiterung.

Tabelle 4.17.: AES: Berechnung des 8. Word in N_4

Word 4		Erweiterung		Word 8
62		F9		9B
63	XOR	FB	=	98
63		FB		98
63		AA		C9

Tabelle 4.18.: AES: Berechnung des 9. Word in N_4

Word 5		Word 8		Word 9
62		9B		F9
63	XOR	98	=	FB
63		98		FB
63		C9		AA

Tabelle 4.19.: AES: Berechnung des 10. Word in N_4

Word 6		Word 9		Word 10
62		F9		9B
63	XOR	FB	=	98
63		FB		98
63		AA		C9

Tabelle 4.20.: AES: Berechnung des 11. Word in N_4

Word 7		Word 10		Word 11
62		9B		F9
63	XOR	98	=	FB
63		98		FB
63		C9		AA

In Tabelle 4.17, Tabelle 4.18, Tabelle 4.19 und Tabelle 4.20 werden die weiteren Spalten der Key Expansion berechnet. Um die Rechnung nachzuvollziehen zu können, muss sie bitweise ausgeführt werden.

round key 2

Tabelle 4.21.: AES: round key 2 bei N_4

	Word 8	Word 9	Word 10	Word 11
Feld 1	9B	F9	9B	F9
Feld 2	98	FB	98	FB
Feld 3	98	FB	98	FB
Feld 4	C9	AA	C9	AA

Tabelle 4.22.: AES: ShiftRow, SubBytes, RC 3 und Erweiterung

ShiftRow	SubBytes S-Box	RC 3	Erweiterung
FB	0F	0000 0100	0B
FB	0F	0000 1111	0F
AA	AC	XOR	AC
F9	99	0000 1011	99

In Tabelle 4.22 wurden in der Spalte ShiftRow die Felder um eins verschoben. In der Spalte SubBytes wurden die Felder aus ShiftRow mittels der S-Box getauscht. In RC wurde im ersten Feld die round constante mittels XOR dazugegeben, das Ergebnis steht in der Spalte Erweiterung.

Tabelle 4.23.: AES: Berechnung des 12. Word in N_4

Word 11		Erweiterung		Word 12
9B		0B		90
98	XOR	0F	=	97
98		AC		34
C9		99		50

Tabelle 4.24.: AES: Berechnung des 13. Word in N_4

Word 12		Word 9		Word 13
F9		90		69
FB	XOR	97	=	6C
FB		34		CF
AA		50		FA

Tabelle 4.25.: AES: Berechnung des 14. Word in N_4

Word 13		Word 10		Word 14
9B		69		F2
98	XOR	6C	=	F4
98		CF		57
C9		FA		33

Tabelle 4.26.: AES: Berechnung des 15. Word in N_4

Word 14		Word 11		Word 15
F9		F2		0B
FB	XOR	F4	=	0F
FB		57		AC
AA		33		99

In Tabelle 4.23, Tabelle 4.24, Tabelle 4.25 und Tabelle 4.26 werden die weiteren Spalten der Key Expansion berechnet. Um die Rechnung nachzuvollziehen zu können, muss sie bitweise ausgeführt werden.

round key 3

Tabelle 4.27.: AES: round key 3 bei N_4

	Spalte 12	Spalte 13	Spalte 14	Spalte 15
Feld 1	90	69	F2	0B
Feld 2	97	6C	F4	0F
Feld 3	34	CF	57	AC
Feld 4	50	FA	33	99

Abhängig von der Blocklänge haben wir eine unterschiedliche Anzahl von RC. Hier sind 16 Spalten angegeben worden, bei $N_b = 4$ sind es vier round keys, bei $N_b = 6$ wären es zwei vollständige round keys und bei $N_b = 8$ wären es zwei round keys.

4.2.8. round key bei N_6

Die Schritte zur Berechnung der einzelnen round keys habe ich in den Anhang⁷ verschoben. Hier werden nur der cipher key und die ersten drei round keys angegeben.

⁷Siehe Abschnitt C.1.1 auf Seite 88.

cipher key

Tabelle 4.28.: AES: cipher key bei N_6

	Word 0	Word 1	Word 2	Word 3	Word 4	Word 5
Feld 1	00	00	00	00	00	00
Feld 2	00	00	00	00	00	00
Feld 3	00	00	00	00	00	00
Feld 4	00	00	00	00	00	00

round key 1

Tabelle 4.29.: AES: round key 1 bei N_6

	Word 6	Word 7	Word 8	Word 9	Word 10	Word 11
Feld 1	62	62	62	62	62	62
Feld 2	63	63	63	63	63	63
Feld 3	63	63	63	63	63	63
Feld 4	63	63	63	63	63	63

round key 2

Tabelle 4.30.: AES: round key 2 bei N_6

	Word 12	Word 13	Word 14	Word 15	Word 16	Word 17
Feld 1	9B	F9	9B	F9	9B	F9
Feld 2	98	FB	98	FB	98	FB
Feld 3	98	FB	98	FB	98	FB
Feld 4	C9	AA	C9	AA	C9	AA

round key 3**Tabelle 4.31.:** AES: round key 3 bei N_6

	Word 18	Word 19	Word 20	Word 21	Word 22	Word 23
Feld 1	90	69	F2	0B	90	69
Feld 2	97	6C	F4	0F	97	6C
Feld 3	34	CF	57	AC	34	CF
Feld 4	50	FA	33	99	50	FA

Hier sind 24 Spalten angegeben worden, bei $N_b = 4$ entspricht es sechs round keys, bei $N_b = 6$ sind es vier round keys und bei $N_b = 8$ wären es drei round keys.

4.2.9. round key bei N_8

Bei Schlüssellänge N_8 gibt es eine Modifikation. Im Rijndael tritt diese Modifikation ab 7 auf. Wenn ein Word Nummer modulo Blocklänge gleich 4 ist, dann wird für das Word, welches vor dem Word modulo Blocklänge vier steht, bevor es dazu gezählt mittels XOR wird, die Funktion SubBytes aufgerufen.

Die Schritte zur Berechnung der einzelnen round keys habe ich in den Anhang⁸ verschoben. Hier werden nur der cipher key und die ersten zwei round keys angegeben.

Tabelle 4.32.: AES: cipher key bei N_8

	Word 0	Word 1	Word 2	Word 3	Word 4	Word 5	Word 6	Word 7
Feld 1	00	00	00	00	00	00	00	00
Feld 2	00	00	00	00	00	00	00	00
Feld 3	00	00	00	00	00	00	00	00
Feld 4	00	00	00	00	00	00	00	00

Tabelle 4.33.: AES: SubBytes von Word 11 in N_8

	SubBytes S-Box
62	AA
63	FB
63	FB
63	FB

In Tabelle 4.33 wird an Word 11 die Funktion SubBytes angewendet, bevor es weiter verwendet wird.

⁸Siehe Abschnitt C.1.2 auf Seite 92.

round key 1

Tabelle 4.34.: AES: round key 1 bei N_8

	Word 8	Word 9	Word 10	Word 11	Word 12	Word 13	Word 14	Word 15
Feld 1	62	62	62	62	AA	AA	AA	AA
Feld 2	63	63	63	63	FB	FB	FB	FB
Feld 3	63	63	63	63	FB	FB	FB	FB
Feld 4	63	63	63	63	FB	FB	FB	FB

Tabelle 4.35.: AES: SubBytes von Word 19 in N_8

	SubBytes S-Box
0D	D7
0F	76
0F	76
AC	91

In Tabelle 4.35 wird an Word 19 die Funktion SubBytes angewendet, bevor es weiter verwendet wird.

round key 2

Tabelle 4.36.: AES: round key 2 bei N_8

	Word 16	Word 17	Word 18	Word 19	Word 20	Word 21	Word 22	Word 23
Feld 1	6F	0D	6F	0D	7D	D7	7D	D7
Feld 2	6C	0F	6C	0F	8D	76	8D	76
Feld 3	6C	0F	6C	0F	8D	76	8D	76
Feld 4	CF	AC	CF	AC	6A	91	6A	91

Tabelle 4.37.: AES: SubBytes von Word 27 in N_8

	SubBytes S-Box
3C	EB
38	07
81	0C
0E	AB

In Tabelle 4.37 wird an Word 27 die Funktion SubBytes angewendet, bevor es weiter verwendet wird.

round key 3**Tabelle 4.38.:** AES: round key 3 bei N_8

	Word 24	Word 25	Word 26	Word 27	Word 28	Word 29	Word 30	Word 31
Feld 1	53	5E	31	3C	96	41	3C	EB
Feld 2	54	5B	37	38	8A	FC	71	7
Feld 3	ED	E2	8E	81	81	F7	7A	0C
Feld 4	C1	6D	A2	0E	C1	50	3A	AB

Hier sind 32 Spalten angegeben worden, bei $N_b = 4$ entspricht es acht round keys, bei $N_b = 6$ entspricht es fünfeinhalb round keys und bei $N_b = 8$ sind es gesamt vier round keys.

4.3. Der Algorithmus

Der Algorithmus ist wie folgt aufgebaut, wie wir ihn schon im einfachen Beispiel⁹ kennengelernt haben:

- Vorrunde
 - AddRoundKey
- n-1 Runde
 - SubBytes
 - * multiplikativ Inverse
 - * affine Funktion
 - ShiftRows
 - MixColumns
 - AddRoundKey
- Letzte Runde
 - SubBytes
 - * multiplikativ Inverse
 - * affine Funktion
 - ShiftRows
 - AddRoundKey

4.3.1. Vorrunde

In der Vorrunde wird die Funktion AddRoundKey ausgeführt.

AddRoundKey

Zum besseren Verständnis wird der cipher key mit angegeben.

Tabelle 4.39.: AES: Vorrunde Add-RoundKey state matrix in Bit

	0	1	2	3
	0000 0000	0000 0100	0000 1000	0000 1100
	0000 0001	0000 0101	0000 1001	0000 1101
	0000 0010	0000 0110	0000 1010	0000 1110
	0000 0011	0000 0111	0000 1011	0000 1111

Tabelle 4.40.: AES: Vorrunde Add-RoundKey state matrix in Hexadezimal

	0	1	2	3
	00	04	08	0C
	01	05	09	0D
	02	06	0A	0E
	03	07	0B	0F

⁹Siehe Abschnitt 2.9 auf Seite 19.

Tabelle 4.41.: AES: Vorrunde Add-RoundKey cipher key in Bit

0	1	2	3
0000 0000	0000 0000	0000 0000	0000 0000
0000 0000	0000 0000	0000 0000	0000 0000
0000 0000	0000 0000	0000 0000	0000 0000
0000 0000	0000 0000	0000 0000	0000 0000

Tabelle 4.42.: AES: Vorrunde Add-RoundKey cipher key in Hexadezimal

0	1	2	3
00	00	00	00
00	00	00	00
00	00	00	00
00	00	00	00

Das Ergebnis der bitweisen XOR-Verknüpfung aus der Tabelle 4.39 und der Tabelle 4.41 steht in Tabelle 4.43.

Tabelle 4.43.: AES: Vorrunde Add-RoundKey Ergebnis in Bit

0	1	2	3
0000 0000	0000 0100	0000 1000	0000 1100
0000 0001	0000 0101	0000 1001	0000 1101
0000 0010	0000 0110	0000 1010	0000 1110
0000 0011	0000 0111	0000 1011	0000 1111

Tabelle 4.44.: AES: Vorrunde Add-RoundKey Ergebnis in Hexadezimal

0	1	2	3
00	04	08	0C
01	05	09	0D
02	06	0A	0E
03	07	0B	0F

4.3.2. Erste von n-1 Runden

SubBytes

Die Funktion SubBytes setzt sich aus einer linearen Funktion zusammen und dem Austauschen der Werte mit ihren multiplikativen Inversen. Wie man die Inverse¹⁰ bestimmen kann, ist schon bekannt aus Key Expansion, genauso wie die lineare Funktion¹¹.

Tabelle 4.45.: AES: erste Runde Sub-Bytes multiplikativ Inverse in Bit

0	1	2	3
0000 0000	1100 1011	1110 1000	1011 0000
0000 0001	0101 0010	0100 1111	1110 0001
1000 1101	0111 1011	0010 1001	1110 0101
1111 0110	1101 0001	1100 0000	1100 0111

Tabelle 4.46.: AES: erste Runde Sub-Bytes multiplikativ Inverse in Hexadezimal

0	1	2	3
00	CB	E8	B0
01	52	4F	E1
8D	7B	29	E5
F6	D1	C0	C7

¹⁰Siehe Abschnitt 4.2.6 auf Seite 37.

¹¹Siehe Abschnitt 4.2.6 auf Seite 39.

Tabelle 4.47.: AES: erste Runde Sub-Bytes lineare Funktion in Bit

0	1	2	3
0110 0011	1111 0010	0011 0000	1111 1110
0111 1100	0110 1011	0000 0001	1101 0111
0111 0111	0110 1111	0110 0111	1010 1011
0111 1011	1100 0101	0010 1011	0111 0110

Tabelle 4.48.: AES: erste Runde Sub-Bytes lineare Funktion in Hexadezimal

0	1	2	3
63	F2	30	FE
7C	6B	1	D7
77	6F	67	AB
7B	C5	2B	76

Shift Rows

Shift Rows wird wie im einfachen Beispiel¹² ausgeführt.

Tabelle 4.49.: AES: erste Runde Shift Rows in Bit

0	1	2	3
0110 0011	1111 0010	0011 0000	1111 1110
0110 1011	0000 0001	1101 0111	0111 1100
0110 0111	1010 1011	0111 0111	0110 1111
0111 0110	0111 1011	1100 0101	0010 1011

Tabelle 4.50.: AES: erste Runde Shift Rows in Hexadezimal

0	1	2	3
63	F2	30	FE
6B	01	D7	7C
67	AB	77	6F
76	7B	C5	2B

MixColumns

Die Funktion MixColumns dient zur Diffusion der Information in einer Spalte. Absicht ist es, dass die Änderung eines Bits am Anfang zu einer nicht vorhersagbaren Veränderung am Ende führt. Gemessen wird dies durch das Hamming Gewicht. Das Hamming Gewicht ist definiert als die Anzahl der Elemente ungleich Null in einem Vektor. Die Hamming Distanz zwischen zwei Vektoren ist der Betrag der Differenzen der beiden Hamming Gewichten der Vektoren. Die Absicht ist eine MDS (Maximum Distance Separable) Matrix zu erstellen, die auch invertierbar ist. Eine MDS Matrix garantiert unter dieser Bedingung der Invertierbarkeit die größtmögliche Diffusion.

In AES und Rijndael sind die Anzahl der Zeilen mit vier fixiert, daher muss die MDS eine 4×4 Matrix sein, um invertierbar zu sein. In MixColumns werden die Elemente des Galoisfeldes erneut auf einen Polynomkörper verteilt. Dieser neue Polynomkörper hat das reduzierende Polynom $x^4 + 1$. Auf diesem Körper wird ein Polynom der Gestalt $b(x) = b_3x^3 + b_2x^2 + b_1x + b_0$ gewählt und ein zweites Polynom $d(x) = d_3x^3 + d_2x^2 + d_1x + d_0$. Für diese zwei Polynome gilt die Beziehung $b(x) * d(x) \equiv 1 \text{ modulo } x^4 + 1$. Im Algorithmus hat das erste Polynom folgende Werte $b(x) = 03x^3 + 01x^2 + 01x + 02$ (Werte in Hexadezimal). Das Inverse Polynom lautet $d(x) = 0Bx^3 + 0Dx^2 + 09x + 0E$.

Die Bestimmung der Inversen wird in der Literatur oft einem PC übergeben, der alle möglichen Kombinationen versucht. Weil dieses leichter und schneller geht, als die Inverse direkt per Hand zu bestimmen. Die Matrix ist zyklisch oder zirkulant und eine spezielle

¹²Siehe Abschnitt 2.9.2 auf Seite 22.

Toeplitz-Matrix.

Die Matrixmultiplikation in MixColumns aufgeschrieben sieht so aus. A0 bis A3 ist eine Spalte der zu verschlüsselnden Textmatrix. C0 bis C4 ist die Ergebnisspalte.

Tabelle 4.51.: AES: MixColumns Berechnungsvorschrift

$$\begin{pmatrix} C_0 \\ C_1 \\ C_2 \\ C_3 \end{pmatrix} = \begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix} \times \begin{pmatrix} A_{.,0} \\ A_{.,1} \\ A_{.,2} \\ A_{.,3} \end{pmatrix}$$

Hier soll nur das C_0 der ersten Spalte und der zweiten Spalte der Textmatrix berechnet werden. Die restlichen Werte wurden im Anhang berechnet¹³.

$$\begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix} \times \begin{pmatrix} 63 \\ 6B \\ 67 \\ 76 \end{pmatrix}$$

$$\begin{array}{rcl} 02 \times 63 & = & \\ \hline 0110\,0011 & \times & 10 \\ \hline 0\,1100\,0110 & & \end{array} \quad \begin{array}{rcl} 03 \times 6B & = & \\ \hline 0110\,1011 & \times & 11 \\ \hline 0\,1101\,0110 & & \\ 0110\,1011 & & \\ \hline 1011\,1101 & & \end{array} \quad \begin{array}{rcl} 01 \times 67 & = & \\ \hline 0110\,0111 & \times & 01 \\ \hline 0110\,0111 & & \end{array} \quad \begin{array}{rcl} 01 \times 76 & = & \\ \hline 0111\,0110 & \times & 01 \\ \hline 0111\,0110 & & \end{array}$$

$$\begin{array}{r} 1100\,0110 \\ 1011\,1101 \\ 0110\,0111 \\ 0111\,0110 \\ \hline 0110\,1010 \end{array}$$

Die Summe aller vier Teilergebnisse 0110 1010 entspricht 6A.

$$\begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix} \times \begin{pmatrix} F2 \\ 01 \\ AB \\ 7B \end{pmatrix}$$

¹³Siehe Abschnitt C.2 auf Seite 99.

$$\begin{array}{r} 02 \times F2 = \\ 1111\ 0010 \times 10 \\ \hline 1\ 1110\ 0100 \end{array}$$

$$\begin{array}{r} 03 \times 01 = \\ 0000\ 0001 \times 11 \\ \hline 0000\ 0010 \\ 0000\ 0001 \\ \hline 0000\ 0011 \end{array}$$

$$\begin{array}{r} 01 \times AB = \\ 1010\ 1011 \times 01 \\ \hline 1010\ 1011 \end{array}$$

$$\begin{array}{r} 01 \times 7B = \\ 0111\ 1011 \times 01 \\ \hline 0111\ 1011 \end{array}$$

$$\begin{array}{r} 1\ 1110\ 0100 \div 1\ 0001\ 1011^{14} = 1 \\ 1\ 0001\ 1011 \\ \hline 0\ 1111\ 1111 \text{ Rest} \end{array}$$

$$\begin{array}{r} 1111\ 1111 \\ 0000\ 0011 \\ 1010\ 1011 \\ 0111\ 1011 \\ \hline 0010\ 1100 \end{array}$$

Die Summe aller vier Teilergebnisse entspricht 2C.

Tabelle 4.52.: AES: erste Runde Mix-Columns in Bit

	0	1	2	3
	0110 1010	0010 1100	1011 0000	0010 0111
	0110 1010	0110 1101	1101 1001	100 11100
	0101 1100	0011 0011	0101 1101	0010 0001
	0100 0101	0101 0001	0110 0001	0101 1100

Tabelle 4.53.: AES: erste Runde Mix-Columns in Hexadezimal

	0	1	2	3
	6A	2C	B0	27
	6A	6D	D9	9C
	5C	33	5D	21
	45	51	61	5C

AddRoundKey

Es wird hier die Stelle $A_{0,0}$ wird hier berechnet, die anderen sind im Anhang.

Tabelle 4.54.: AES: erste Runde Add-RoundKey state matrix in Bit

	0	1	2	3
	0000 1000	0100 1110	110 10010	0100 0101
	0000 1001	0000 1110	101 11010	1111 1111
	0011 1111	0101 0000	001 11110	0100 0010
	0010 0110	0011 0010	000 00010	0011 1111

Tabelle 4.55.: AES: erste Runde Add-RoundKey state matrix in Hexadezimal

	0	1	2	3
	08	4E	D2	45
	09	0E	BA	FF
	3F	50	3E	42
	26	32	02	3F

¹⁴Dieses reduzierende Polynom ist schon bekannt aus SubBytes bei der Bestimmung des multiplikativ Inversen.

Tabelle 4.56.: AES: erste Runde Add-RoundKey round key in Bit

	0	1	2	3
	0110 0010	0110 0010	0110 0010	0110 0010
	0110 0011	0110 0011	0110 0011	0110 0011
	0110 0011	0110 0011	0110 0011	0110 0011
	0110 0011	0110 0011	0110 0011	0110 0011

Tabelle 4.57.: AES: erste Runde Add-RoundKey round key in Hexadezimal

	0	1	2	2
	62	62	62	62
	63	63	63	63
	63	63	63	63
	63	63	63	63

Die Stelle $A_{0,0}$ berechnet, die anderen sind im Anhang.

0110 1010 entspricht 6A
0110 0010 entspricht 62

0000 1000 entspricht 08

Das Ergebnis der bitweisen XOR-Verknüpfung aus der Tabelle 4.52 und Tabelle 4.56 steht in Tabelle 4.54.

4.4. Die Algorithmus - Entschlüsselung

Die Entschlüsselung verläuft genauso wie die Verschlüsselung bloß in umgekehrter Reihenfolge.

Der Algorithmus ist wie folgt aufgebaut:

- InvLetzte Runde
 - InvAddRoundKey
 - InvShiftRows
 - InvSubBytes
 - * affine Funktion
 - * multiplikativ Inverse
- n-1 InvRunde
 - InvAddRoundKey
 - InvMixColumns
 - InvShiftRows
 - SubBytes
 - * affine Funktion
 - * multiplikativ Inverse
- InvVorrunde
 - InvAddRoundKey

4.4.1. InvAddRoundKey

InvAddRoundKey sieht genauso aus wie AddRoundKey. Diese Funktion ist ihre eigene Inverse Funktion, weil XOR selbst eine Inverse ist bezüglich der Binäroperation¹⁵.

$$\begin{array}{r} 0011\ 1001\ 39 \\ 0110\ 0010\ 62 \\ \hline 0101\ 1011\ 5B \end{array}$$

4.4.2. InvMixColumns

Wie schon in MixColumns erwähnt, sind die Koeffizienten der Matrixelemente aus einem Polynomkörper mit dem reduzierendem Polynom $x^4 + 1$. Auf diesem Körper haben wir zwei Polynome $b(x) = 03x^3 + 01x^2 + 01x + 02$ (Werte in Hexadezimal) und $d(x) = 0Bx^3 + 0Dx^2 + 09x + 0E$ mit folgender Beziehung $b(x) * d(x) \equiv 1 \text{ modulo } x^4 + 1$ ¹⁶.

Tabelle 4.58.: AES: InvMixColumns Berechnungsvorschrift

$$\begin{pmatrix} A_{.,0} \\ A_{.,1} \\ A_{.,2} \\ A_{.,3} \end{pmatrix} = \begin{pmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{pmatrix} \times \begin{pmatrix} C_0 \\ C_1 \\ C_2 \\ C_3 \end{pmatrix}$$

¹⁵Siehe Abschnitt 3 auf Seite 32.

¹⁶Siehe Abschnitt 4.3.2 auf Seite 52.

4.4.3. InvShiftRows

InvShiftRows arbeitet genauso wie ShiftRows¹⁷. Nur dieses Mal wird nach links geschoben statt nach rechts. Die freigewordenen Blöcke werden links wieder eingefügt.

4.4.4. InvSubBytes

InvSubBytes besteht wieder aus zwei Operationen, aus Invlinearen Funktion und Invmultiplikativ Inversen.

invlineare Funktion

Die inverse Funktion lässt sich wie folgt einfach bestimmen.

$$A^{-1}y + A^{-1}Vektor = x^{18}$$

Die Inverse der Matrix mal der Determinante modulo 2 ergibt die Matrix¹⁹. Der Vektor multipliziert mit der inversen Matrix ergibt den Vektor der inversen Funktion.

Tabelle 4.59.: Rijndael: invlineare Funktion

$$\begin{pmatrix} x^7 \\ x^6 \\ x^5 \\ x^4 \\ x^3 \\ x^2 \\ x^1 \\ x^0 \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \end{pmatrix} \times \begin{pmatrix} y^7 \\ y^6 \\ y^5 \\ y^4 \\ y^3 \\ y^2 \\ y^1 \\ y^0 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 1 \end{pmatrix}$$

¹⁷Siehe Abschnitt 4.3.2 auf Seite 52.

¹⁸Siehe Abschnitt 4.2.6 auf Seite 39.

¹⁹Siehe Abschnitt 2.10.1 auf Seite 29.

Tabelle 4.60.: AES: Tabelle für die invlineare Funktion

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

4.4.5. Invmultiplikativ Inverse

Diese Funktion ist ihre eigene inverse Funktion.

Tabelle 4.61.: AES: Tabelle für die invmultiplikativ Inverse

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

Es lassen sich die beiden Funktion invlineare Funktion und invmultiplikativ Inverse zusammenfassen in einer InvS-Box.

Tabelle 4.62.: AES: InvS-Box

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

Es gilt zu beachten, dass bei der Entschlüsselung dieselbe Chiffre verwendet wird. Derjenige, welcher den Text entschlüsseln möchte, muss vorher eine Key Expansion machen. Es werden am Beginn der Entschlüsselung die round keys benötigt, die am Schluss der Verschlüsselung benutzt wurden.

5. Stundeneinteilung

5.1. Stundeneinteilung

Beim Einteilen der Stunden denke ich an Doppelstunden mit je 45 Minuten.

5.1.1. Doppelstunde

Eine nette Eröffnung des Lehrgangs bietet der Comic. „A Stick Figure Guide to the Advanced Encryption Standard (AES)“ [5] Ich würde den Einstieg in das Thema über das einfache Beispiel machen, beginnend mit dem konzeptionellen Ablauf des Algorithmus, ohne genauer auf die einzelnen Funktionen einzugehen, nur um zu zeigen, dass es einfacher Aufbau ist.

- Vorrunde
 - AddRoundKey
- n-1 Runde
 - SubBytes
 - * multiplikativ Inverse
 - * affine Funktion
 - ShiftRows
 - MixColumns
 - AddRoundKey
- Letzte Runde
 - SubBytes
 - * multiplikativ Inverse
 - * affine Funktion
 - ShiftRows
 - AddRoundKey

Der nächste Schritt ist das Übertragen der Buchstaben in Zahlen, dabei wird der Primzahlkörper vorgestellt. Nun können die Lernenden üben in Restklassen zu addieren und zu multiplizieren sowie das multiplikative Inverse zu erstellen. Sie können hierbei mittels Papier oder einem Tabellenkalkulationsprogramm arbeiten.

5.1.2. Zweite Doppelstunde

In dieser Stunde kann die Funktion Key Schedule (Schlüsselerweiterung) erklärt werden und die erste Schlüsselerweiterung berechnet werden. Man sollte sich an dieser Stelle Zeit lassen, weil für viele Lernende das Konzept der Modulo Operation und das Operieren in einen Körper mit abzählbar vielen Elementen sehr ungewohnt ist.

Das Erlernen des erweiterten Euklidischen Algorithmus zur Feststellung des multiplikativen Inversen kann auf einige Schwierigkeiten stoßen, weil Lernende eher neigen bei kleinen Körper dieses durch probieren finden.

5.1.3. Dritte Doppelstunde

Die dritte Stunde ist die Fortsetzung der zweiten Stunde. In der Theorie kann besprochen werden, wie viele Erweiterungsschritte nötig sind und warum. Es kann erklärt werden, dass AES eine $4 \times k$ Matrix verwendet, weil es verschiedene Schlüssellängen gibt.

5.1.4. Vierte Doppelstunde

Dieser Stunde sollte, damit begonnen werden mit den Lernenden gemeinsam ein dynamische Blatt in einem Tabellenkalkulationsprogramm zu erstellen. Dem Lehrenden kann als Vorlage mein Blatt dienen¹. Auf einem Blatt in dieser Datei wird die Funktion Key Schedule für das einfache Beispiel wie in der Vorlage erstellt. Auf einem anderen Blatt in derselben Datei kann das Feld für die Eingabe angelegt werden, sowie für die weiteren Funktion verwendet werden.

5.1.5. Fünfte Doppelstunde

In dieser Stunde wird der Algorithmus besprochen und auf die, durch die Funktion Key Schedule bekannten Funktionen Subbytes und ShiftRow hingewiesen. Desweiteren wird auf die Besonderheit der letzten Runde ohne Mix Columns hingewiesen.

5.1.6. Sechste Doppelstunde

Diese Stunde wird benötigt, damit die Lernende die Verschlüsselung fertig stellen und etwaige Fehler bereinigen können.

5.1.7. Siebente Doppelstunde (optional)

In dieser Stunde wird die Entschlüsselung angegangen. Wie bei der Entschlüsselung wird eine Key Schedule benötigt, da normalerweise Ver- und Entschlüsselung auf getrennten Geräten stattfinden. Die Entschlüsselung ist wie folgt aufgebaut:

¹Die Datei ist zu finden unter https://docs.google.com/spreadsheets/d/1tgn9xFpAhvGVYpdAvP7e3GzCr9cKwWmgLYV40B2u_Vw/edit?usp=sharing.

- InvLetzte Runde
 - InvAddRoundKey
 - InvShiftRows
 - InvSubBytes
 - * affine Funktion
 - * multiplikativ Inverse
- n-1 InvRunde
 - InvAddRoundKey
 - InvMixColumns
 - InvShiftRows
 - SubBytes
 - * affine Funktion
 - * multiplikativ Inverse
- InvVorrunde
 - InvAddRoundKey

Wichtig ist die Bestimmung der Inversen Linearen Funktion in SubBytes sowie der Inversen Matrix in MixColumns. Es gilt zu beachten, dass in diesem Verschlüsselungsalgorithmus nur Zahlen aus der Zahlenmenge $\mathbb{Z}$ vorkommen.

5.1.8. Achte Doppelstunde

In dieser Stunde wird die Entschlüsselung fertiggestellt und die logischen Gatter werden durchgenommen, und auf XOR wird eingegangen.

5.1.9. Neunte Doppelstunde

Es wird in dieser Stunde mit der Key Schedule (Schlüsselexpansion) auf Bit-Ebene begonnen. Es wird cipher key mit Null verwendet, weil dieser sich schneller per Hand expandieren lässt. Es wird ebenfalls die Expansion für AES mit Schlüssellänge mit 196 Bit und 256 Bit durchgerechnet.

5.1.10. Zehnte Doppelstunde

Diese Stunde ist eine Pufferstunde. Der erweiterte Euklidische Algorithmus im Binärensystem und die Betrachtung der Zahlen als Polynome ist sehr ungewohnt und muss von den Lernenden verdaut werden.

5.1.11. Elfte Doppelstunde

In dieser Stunde wird die Vorrunde und die erste Runde der Verschlüsselung berechnet.

5.1.12. Zwölfte Doppelstunde

In dieser Stunde wird die Entschlüsselung angegangen. Es wird gezeigt wie die Inversefunktion und -matrix für die Funktion SubBytes und MixColumns bestimmt wird.

5.1.13. Dreizehnte Doppelstunde

In dieser Stunde wird die Umsetzung des Algorithmus in einem Tabellenkalkulationsprogramm besprochen und als Übung sollte die Entschlüsselung für 192 Bit oder 256 Bit angegangen werden. Das Material ist auf der Webseite [17] zu finden.

5.1.14. Vierzehnte Doppelstunde

In dieser Stunde könnte der Code besprochen werden, während die Lernenden auch damit beschäftigt sind ihre Arbeit der vorherigen Stunde fortzusetzen.

5.2. Die Besonderheit der letzten Runde

Die Funktion MixColumns fehlt in der letzten Runde, damit sich das MixColumns der vorletzten Runde in seiner Wirkung entfalten kann. Es wurde während der Konstruktion des Algorithmus eine minimal Anzahl an MixColumns bestimmt, um eine ausreichende Diffusion zu erreichen. Zu der bestimmten Zahl wurden zur Sicherheit ein paar weitere MixColumns hinzugefügt, damit alle MixColumns ihre Wirkung entfalten können, wird eine Runde ohne MixColumns benötigt.

5.3. Einfaches Beispiel

5.3.1. Inverse Funktionen

Im einfachen Beispiel ist ersichtlich wie schwierig es ist die Inverse unter Nebenbedingungen zu erstellen. Die Inverse im einfachen Beispiel wird mit Hilfe der Cramersche Regel erstellt, um zu gewährleisten, dass die Zahlen des Ergebnisses ganzzahlig sind.

5.4. AES

5.4.1. Inverse Funktion

Die Inverse Funktion in SubBytes lässt sich wie im einfachen Beispiel bestimmen, weil zwei eine Primzahl ist. Die Inversematrix bei Funktion MixColumns lässt sich nicht einfach erstellen, deswegen fehlt die Erstellung der Inversematrix in den Unterlagen, da es über die Anforderungen des Kurses geht.

5.5. Tabellenkalkulationsprogramm

P. Barreto² machte die folgende Beobachtung bezüglich der Beziehung zwischen dem Polynom $b(x) = 03x^3 + 01x^2 + 01x + 02$ und dem Polynom $d(x) = 0Bx^3 + 0Dx^2 + 09x + 0E$, die in InvMixColumns verwendet werden.

$$d(x) = (04x^2 + 05)b(x) \pmod{x^4 + 1}$$

In Matrixnotation bedeutet dies:

²Siehe [2] auf Seite 55.

$$\begin{pmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{pmatrix} = \begin{pmatrix} 05 & 00 & 04 & 00 \\ 00 & 05 & 00 & 04 \\ 04 & 00 & 05 & 00 \\ 00 & 04 & 00 & 05 \end{pmatrix} \times \begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix}$$

In der Tabellenkalkulationsdatei könnte somit eine Tabelle eingespart werden, weil $04 = 02 \times 02$ und $05 = 04 \text{ xor } 01$. Dieses mag auf den ersten Blick trivial aussehen, doch sollte dabei berücksichtigt werden, dass in Binärdarstellung gedacht werden soll. $0001 \times 0010 \times 0010 = 0100$ (04 Hexadezimal) $0100 \text{ xor } 0001 = 0101$ (05 Hexadezimal)

6. Code

Der Code wurde von [9] genommen. Er ist äquivalent zum Code aus [2] auf Seite 221. Ein Unterschied besteht in der Funktion „main()“.

Code 6.1: Beginn

```
1 //=====
2 // Rijndael code August 01
3 //
4 // author: Vincent Rijmen
5 // This code is based on the official reference code
6 // by Paulo Barreto and Vincent Rijmen
7 //
8 // This code is placed in the public domain.
9 // Without any warranty of fitness for any purpose
10
11 //=====
12 #include <stdio.h>
```

In Code 6.2 werden zwei Variablen Typen definiert.

Code 6.2: Variablen

```
13 //=====
14
15 typedef unsigned char word8;
16 typedef unsigned int word32;
```

In Code 6.3 ist ein Code Beispiel gegeben für die direkte Multiplikation in $GF(256)$ ¹ geben. Hierbei wird a um ein Bit nach links geschoben und zu Lösung p hinzugegeben, falls in b an entsprechender Stelle eine 1 steht. Sollte bei a ein Überlauf stattfinden, wird mit $(1B)_{16}$ reduziert.

Dieses Code wurde aus [13] entnommen².

Code 6.3: Multiplikation

```
1 /**
2  * Multiply two numbers in the GF(2^8) finite field defined
3  * by the polynomial x^8 + x^4 + x^3 + x + 1
4  */
5
6 uint8_t galois_multiplication(uint8_t a, uint8_t b)
7 {
8     uint8_t p = 0;
9     uint8_t hi_bit_set;
10    uint8_t counter;
11
12    for (counter = 0; counter < 8; counter++) {
13        if ((b & 1) == 1)
14            p ^= a;
15        hi_bit_set = (a & 0x80);    // xtime() ...
16        a <<= 1;
```

¹GF(256) bezeichnet ein Galoisfeld mit 256 Elementen.

²Siehe Code 6.13 auf Seite 69.

```

17     if (hi_bit_set == 0x80)
18         a ^= 0x1b;           // ...
19         b >>= 1;
20     }
21
22     return p;
23 }

```

In Code 6.4 und Code 6.5 sind die Potenztabelle und die Logarithmustabelle abgebildet zur Basis $(0011)_2$. Die Potenztabelle mit dem Generator 0011 liefert dieses Ergebnis, wenn 00000011 so oft mit sich selbst multipliziert wird.

$$\begin{aligned}
 0011^{0000} &= 0001 & (1) \\
 0011^{0001} &= 0001 \times 0011 = 0011 & (3) \\
 0011^{0011} &= 0011 \times 0011 = 0101 & (5) \\
 0011^{0101} &= 0101 \times 0011 = 1111 & (15)
 \end{aligned}$$

Code 6.4: Logarithmustabelle

```

17 //=====
18 // The table Logtable and Alogtable are used to perform
19 // multiplication in GF(256)
20 //=====
21
22 word8 Logtable[256] = {
23     0,  0, 25,  1, 50,  2, 26, 198, 75, 199, 27, 104, 51, 238, 223,  3,
24    100,  4, 224, 14, 52, 141, 129, 239, 76, 113,  8, 200, 248, 105, 28, 193,
25    125, 194, 29, 181, 249, 185, 39, 106, 77, 228, 166, 114, 154, 201,  9, 120,
26    101, 47, 138,  5, 33, 15, 225, 36, 18, 240, 130, 69, 53, 147, 218, 142,
27    150, 143, 219, 189, 54, 208, 206, 148, 19, 92, 210, 241, 64, 70, 131, 56,
28    102, 221, 253, 48, 191,  6, 139, 98, 179, 37, 226, 152, 34, 136, 145, 16,
29    126, 110, 72, 195, 163, 182, 30, 66, 58, 107, 40, 84, 250, 133, 61, 186,
30    43, 121, 10, 21, 155, 159, 94, 202, 78, 212, 172, 229, 243, 115, 167, 87,
31    175, 88, 168, 80, 244, 234, 214, 116, 79, 174, 233, 213, 231, 230, 173, 232,
32    44, 215, 117, 122, 235, 22, 11, 245, 89, 203, 95, 176, 156, 169, 81, 160,
33    127, 12, 246, 111, 23, 196, 73, 236, 216, 67, 31, 45, 164, 118, 123, 183,
34    204, 187, 62, 90, 251, 96, 177, 134, 59, 82, 161, 108, 170, 85, 41, 157,
35    151, 178, 135, 144, 97, 190, 220, 252, 188, 149, 207, 205, 55, 63, 91, 209,
36    83, 57, 132, 60, 65, 162, 109, 71, 20, 42, 158, 93, 86, 242, 211, 171,
37    68, 17, 146, 217, 35, 32, 46, 137, 180, 124, 184, 38, 119, 153, 227, 165,
38    103, 74, 237, 222, 197, 49, 254, 24, 13, 99, 140, 128, 192, 247, 112,  7
39 };

```

Die Logarithmustabelle liefert dieses Ergebnis, wenn man eine Zahl logarithmiert zur Basis 0011.

$$\begin{aligned}
 \log_{0011}(0001) &= 0000 & (0) \\
 \log_{0011}(0010) &= 1\ 1001 & (25) \\
 \log_{0011}(0011) &= 0001 & (1) \\
 \log_{0011}(0100) &= 11\ 0010 & (50) \\
 \log_{0011}(0101) &= 0011 & (2)
 \end{aligned}$$

Null lässt sich nicht logarithmieren. 0010 muss mit 25 potenziert werden, um ein Element aus der Restklasse drei (0011) zu erhalten.

Code 6.5: Potenztabelle Tafel

```

40 //=====
41
42 word8 Alogtable[256] = {
43     1,  3,  5, 15, 17, 51, 85, 255, 26, 46, 114, 150, 161, 248, 19, 53,
44    95, 225, 56, 72, 216, 115, 149, 164, 247,  2,  6, 10, 30, 34, 102, 170,
45   229, 52, 92, 228, 55, 89, 235, 38, 106, 190, 217, 112, 144, 171, 230, 49,

```

```

46 83, 245, 4, 12, 20, 60, 68, 204, 79, 209, 104, 184, 211, 110, 178, 205,
47 76, 212, 103, 169, 224, 59, 77, 215, 98, 166, 241, 8, 24, 40, 120, 136,
48 131, 158, 185, 208, 107, 189, 220, 127, 129, 152, 179, 206, 73, 219, 118, 154,
49 181, 196, 87, 249, 16, 48, 80, 240, 11, 29, 39, 105, 187, 214, 97, 163,
50 254, 25, 43, 125, 135, 146, 173, 236, 47, 113, 147, 174, 233, 32, 96, 160,
51 251, 22, 58, 78, 210, 109, 183, 194, 93, 231, 50, 86, 250, 21, 63, 65,
52 195, 94, 226, 61, 71, 201, 64, 192, 91, 237, 44, 116, 156, 191, 218, 117,
53 159, 186, 213, 100, 172, 239, 42, 126, 130, 157, 188, 223, 122, 142, 137, 128,
54 155, 182, 193, 88, 232, 35, 101, 175, 234, 37, 111, 177, 200, 67, 197, 84,
55 252, 31, 33, 99, 165, 244, 7, 9, 27, 45, 119, 153, 176, 203, 70, 202,
56 69, 207, 74, 222, 121, 139, 134, 145, 168, 227, 62, 66, 198, 81, 243, 14,
57 18, 54, 90, 238, 41, 123, 141, 140, 143, 138, 133, 148, 167, 242, 13, 23,
58 57, 75, 221, 124, 132, 151, 162, 253, 28, 36, 108, 180, 199, 82, 246, 1
59 };

```

Auf Grund der Potenztabelle und der Logarithmustabelle gibt es eine weitere Möglichkeit zu multiplizieren. $(5)_{10} * (6)_{10}$ logarithmiert ist $2 + 26$. Der 28ste Eintrag in der Potenztabelle ist 30. $(30)_{10}$ entspricht $(1\ 1110)_2$.

$$\begin{array}{r}
 0101 \times 0110 \\
 \hline
 \text{Probe: } 1\ 0100 \\
 0\ 1010 \\
 \hline
 1\ 1110
 \end{array}$$

6.0.1. Generatoren

Um Generatoren zu finden gibt es keinen einfachen Weg. Es gibt ein paar Regeln die beim Suchen helfen. $GF(256)$ hat die Ordnung 255 und 255 hat die Primfaktorzerlegung $3 \cdot 5 \cdot 17$. Ein Generator g hat folgende Regeln zu befolgen. $g \neq 0$

$$g^{(255/3)} \neq 1$$

$$g^{(255/5)} \neq 1$$

$$g^{(255/17)} \neq 1$$

$$g^{(255/255)} = g$$

Nachdem der Generator 255 mal mit sich selbst multipliziert wurde, befindet man sich wieder in der Restklasse des Generators. Ein Generator durchläuft alle Restklassen, bevor er wieder in seine eigene zurückkommt. Bei dem Restklassenkörper sieben wären zwei, drei, vier, fünf und sechs Generatoren. Alle 128 Generatoren können, sowie die dazugehörigen Potenztabellen und Logarithmustabellen auf [20] angesehen werden.

Die S-Box³ in Code 6.6 und die InvS-Box⁴ in Code 6.7 werden statisch angelegt. Auf der Seite [13] ist eine Anleitung gegeben, wie die S-Box während der Laufzeit generiert werden kann. Auf [20] sind Hinweise dafür gegeben, aber kein vollständiger Code.

Code 6.6: S-Box

```

60 //=====
61
62 word8 S[256] = {
63 99, 124, 119, 123, 242, 107, 111, 197, 48, 1, 103, 43, 254, 215, 171, 118,
64 202, 130, 201, 125, 250, 89, 71, 240, 173, 212, 162, 175, 156, 164, 114, 192,
65 183, 253, 147, 38, 54, 63, 247, 204, 52, 165, 229, 241, 113, 216, 49, 21,
66 4, 199, 35, 195, 24, 150, 5, 154, 7, 18, 128, 226, 235, 39, 178, 117,
67 9, 131, 44, 26, 27, 110, 90, 160, 82, 59, 214, 179, 41, 227, 47, 132,
68 83, 209, 0, 237, 32, 252, 177, 91, 106, 203, 190, 57, 74, 76, 88, 207,
69 208, 239, 170, 251, 67, 77, 51, 133, 69, 249, 2, 127, 80, 60, 159, 168,
70 81, 163, 64, 143, 146, 157, 56, 245, 188, 182, 218, 33, 16, 255, 243, 210,

```

³Siehe Tabelle ?? auf Seite ??.

⁴Siehe Tabelle ?? auf Seite ??.

```

71 205, 12, 19, 236, 95, 151, 68, 23, 196, 167, 126, 61, 100, 93, 25, 115,
72 96, 129, 79, 220, 34, 42, 144, 136, 70, 238, 184, 20, 222, 94, 11, 219,
73 224, 50, 58, 10, 73, 6, 36, 92, 194, 211, 172, 98, 145, 149, 228, 121,
74 231, 200, 55, 109, 141, 213, 78, 169, 108, 86, 244, 234, 101, 122, 174, 8,
75 186, 120, 37, 46, 28, 166, 180, 198, 232, 221, 116, 31, 75, 189, 139, 138,
76 112, 62, 181, 102, 72, 3, 246, 14, 97, 53, 87, 185, 134, 193, 29, 158,
77 225, 248, 152, 17, 105, 217, 142, 148, 155, 30, 135, 233, 206, 85, 40, 223,
78 140, 161, 137, 13, 191, 230, 66, 104, 65, 153, 45, 15, 176, 84, 187, 22
79 };

```

Code 6.7: InvS-Box

```

81 //=====
82
83 word8 Si[256] = {
84 82, 9, 106, 213, 48, 54, 165, 56, 191, 64, 163, 158, 129, 243, 215, 251,
85 124, 227, 57, 130, 155, 47, 255, 135, 52, 142, 67, 68, 196, 222, 233, 203,
86 84, 123, 148, 50, 166, 194, 35, 61, 238, 76, 149, 11, 66, 250, 195, 78,
87 8, 46, 161, 102, 40, 217, 36, 178, 118, 91, 162, 73, 109, 139, 209, 37,
88 114, 248, 246, 100, 134, 104, 152, 22, 212, 164, 92, 204, 93, 101, 182, 146,
89 108, 112, 72, 80, 253, 237, 185, 218, 94, 21, 70, 87, 167, 141, 157, 132,
90 144, 216, 171, 0, 140, 188, 211, 10, 247, 228, 88, 5, 184, 179, 69, 6,
91 208, 44, 30, 143, 202, 63, 15, 2, 193, 175, 189, 3, 1, 19, 138, 107,
92 58, 145, 17, 65, 79, 103, 220, 234, 151, 242, 207, 206, 240, 180, 230, 115,
93 150, 172, 116, 34, 231, 173, 53, 133, 226, 249, 55, 232, 28, 117, 223, 110,
94 71, 241, 26, 113, 29, 41, 197, 137, 111, 183, 98, 14, 170, 24, 190, 27,
95 252, 86, 62, 75, 198, 210, 121, 32, 154, 219, 192, 254, 120, 205, 90, 244,
96 31, 221, 168, 51, 136, 7, 199, 49, 177, 18, 16, 89, 39, 128, 236, 95,
97 96, 81, 127, 169, 25, 181, 74, 13, 45, 229, 122, 159, 147, 201, 156, 239,
98 160, 224, 59, 77, 174, 42, 245, 176, 200, 235, 187, 60, 131, 83, 153, 97,
99 23, 43, 4, 126, 186, 119, 214, 38, 225, 105, 20, 99, 85, 33, 12, 125
100 };

```

In Code 6.8 sieht man die ersten 18. RC fix angegeben⁵.

Code 6.8: RC round key

```

101 //=====
102
103 word32 Rc[30] = {
104 0x00, 0x01, 0x02, 0x04, 0x08, 0x10, 0x20, 0x40, 0x80, 0x1B, 0x36, 0x6C,
105 0xD8, 0xAB, 0x4D, 0x9A, 0x2F, 0x5E, 0xBC, 0x63, 0xC6, 0x97, 0x35, 0x6A,
106 0xD4, 0xB3, 0x7D, 0xFA, 0xEF, 0xC5 };

```

In Code 6.9 werden Variablen mit fixen Werten definiert. MAXBC ist die maximale Blocklänge, MAXKC ist die maximale Keylänge und MAXROUNDS ist die maximale Rundenanzahl⁶.

Code 6.9: Werte

```

107 //=====
108
109 #define MAXBC 8
110 #define MAXKC 8
111 #define MAXROUNDS 14

```

In Code 6.10 sind die Werte für die ShiftRows angegeben⁷.

⁵Siehe Abschnitt 4.2.3 auf Seite 35.

⁶Siehe Tabelle 4.2 auf Seite 35.

⁷Siehe Tabelle 4.3 auf Seite 37.

Code 6.10: ShiftRows in Abhängigkeit von N_b

```

112 //=====
113
114 static word8 shifts[5][4] = {
115 0,1,2,3,
116 0,1,2,3,
117 0,1,2,3,
118 0,1,2,4,
119 0,1,3,4};

```

In Code 6.11 ist die Tabelle für die Rundenanzahl angegeben⁸.

Code 6.11: Rundenanzahl N_r

```

120 //=====
121
122 static int numrounds[5][5] = {
123 10,11,12,13,14,
124 11,11,12,13,14,
125 12,12,12,13,14,
126 13,13,13,13,14,
127 14,14,14,14,14};

```

In Code 6.12 werden freie Variablen angelegt.

Code 6.12: freie Variablen

```

128 //=====
129
130 int BC;
131 int KC;
132 int ROUNDS;

```

In Code 6.13 wird die Multiplikation mit Hilfe der Logarithmustabelle und Potenztabelle ausgeführt⁹.

Code 6.13: Multiplikation mit Logarithmus- und Potenztabelle

```

133 //=====
134 // multiply two elements of GF(256)
135 // required for MixColumns and InvMixColumns
136 //=====
137
138 word8 mul(word8 a, word8 b)
139 {
140     if( a&b )
141         return Alogtable[(Logtable[a] + Logtable[b]) % 255];
142     else
143         return 0;
144 }

```

In Code 6.14 wird die RC zum round key dazugegeben¹⁰.

Code 6.14: RC wird zu round key geben

```

145 //=====
146 // XOR corresponding text input and round key input bytes
147 //=====
148
149 void AddRoundKey(word8 a[4][MAXBC], word8 rk[4][MAXBC])

```

⁸Siehe Tabelle 4.2 auf Seite 35.

⁹Siehe Code 6.3 auf Seite 65.

¹⁰Siehe Abschnitt 4.2.7 auf Seite 42.

```

150 {
151     int i;
152     int j;
153     for(i=0; i<4; i++)
154         for(j=0; j<BC; j++)
155         {
156             a[i][j] ^= rk[i][j];
157         }
158 }

```

In Code 6.15 die ist SubBytes dargestellt¹¹.

Code 6.15: SubBytes

```

159 //=====
160 // Replace every byte of the input by the byte at that place
161 // in the non-linear S-box
162 //=====
163
164 void SubBytes( word8 a[4][MAXBC], word8 box[256] )
165 {
166     int i;
167     int j;
168     for(i=0; i<4; i++)
169         for(j=0; j<BC; j++)
170         {
171             a[i][j] = box[a[i][j]];
172         }
173 }

```

In Code 6.16 die ist ShiftRows dargestellt¹².

Code 6.16: ShiftRows

```

174 //=====
175 // Row 0 remains unchanged
176 // The other three rows are shifted a variadle amount
177 //=====
178
179 void ShiftRows(word8 a[4][MAXBC], word8 d)
180 {
181     word8 tmp[MAXBC];
182     int i;
183     int j;
184     if( d==0 )
185     {
186         for( i=1; i<4; i++)
187         {
188             for( j=0; j<BC; j++)
189             {
190                 tmp[j] = a[i][ ( j + shifts[BC-4][i] ) % BC];
191             }
192             for(j=0; j
193             {
194                 a[i][j] = tmp[j];
195             }
196         }
197     }
198     else
199     {

```

¹¹Siehe Abschnitt 4.2.6 auf Seite 37.

¹²Siehe Abschnitt 4.2.5 auf Seite 36.

```

200     for( i=1; i<4; i++ )
201     {
202         for( j=0; j<BC; j++)
203         {
204             tmp[j]=a[i][ ( BC + j - shifts[BC-4][i] ) % BC];
205         }
206         for(j=0; j< BC ; j++)
207         {
208             a[i][j] = tmp[j];
209         }
210     }
211 }
212 }

```

In Code 6.17 ist die MixColumns dargestellt¹³.

Code 6.17: MixColumns

```

213 //=====
214 // Mix the four bytes of every column in a linear way
215 //=====
216
217 void MixColumns(word8 a[4][MAXBC])
218 {
219     word8 b[4][MAXBC];
220     int i;
221     int j;
222     for(j=0; j<BC; j++)
223         for(i=0; i<4; i++)
224         {
225             b[i][j] = mul(2, a[i][j]) ^ mul(3, a[(i + 1) % 4][j])
226             ^ a[(i + 2) % 4][j] ^ a[(i + 3) % 4][j];
227         }
228     for(i=0; i<4; i++)
229         for(j=0; j<BC; j++)
230         {
231             a[i][j] = b[i][j];
232         }
233 }

```

In Code 6.18 ist die InvMixColumns dargestellt¹⁴.

Code 6.18: InvMixColumns

```

234 //=====
235 // Mix the four bytes of every column in a linear way
236 // This is the opposite operation of Mixcolumns
237 //=====
238
239 void InvMixcolumns( word8 a[4][MAXBC] )
240 {
241     word8 b[4][MAXBC];
242     int i;
243     int j;
244     for(j=0; j<BC; j++)
245         for(i=0; i<4; i++)
246         {
247             b[i][j] = mul(0xe, a[i][j]) ^ mul(0xb, a[(i + 1) % 4][j])
248             ^ mul(0xd, a[(i + 2) % 4][j]) ^ mul(0x9, a[(i + 3) % 4][j]);
249         }

```

¹³Siehe Abschnitt 4.3.2 auf Seite 52.

¹⁴Siehe Abschnitt 4.4.2 auf Seite 56.

```

250
251     for( i=0; i<4; i++)
252         for( j=0; j<BC; j++)
253             {
254                 a[i][j] = b[i][j];
255             }
256 }

```

In Code 6.19 ist die Schlüsselerweiterung dargestellt¹⁵.

Code 6.19: KeyExpansion

```

258 //=====
259
260 int KeyExpansion( word8 k[4][MAXKC], word8 w[MAXROUNDS+1][4][MAXBC] )
261 {
262     // Calculate the required round keys
263     int i;
264     int j;
265     int t;
266     int RCpointer = 1;
267     word8 tk[4][MAXKC];
268
269     for( j=0; j<KC; j++)
270         for( i=0; i<4; i++)
271             {
272                 tk[i][j] = k[i][j];
273             }
274     t=0;
275
276     // copy values into round key array
277     for( j=0; (j < KC) && (t < (ROUNDS + 1) * BC ); j++, t++ )
278         for( i=0; i<4; i++ )
279             {
280                 w[t / BC][i][t % BC] = tk[i][j];
281             }
282     while( t<(ROUNDS+1)*BC )
283     {
284
285         // while not enough round key material calculated,
286         // calculate new values
287         for( i=0; i<4; i++ )
288             {
289                 tk[i][0] ^= S[tk[(i+1)%4][KC-1] ];
290             }
291         tk[0][0] ^= Rc[ RCpointer++ ];
292         if( KC <= 6 )
293         {
294             for( j=1; j<KC; j++)
295                 for( i=0; i<4; i++ )
296                     {
297                         tk[i][j] ^= tk[i][j-1];
298                     }
299         }
300         else
301         {
302             for( j=1; j<4 ;j++ )
303                 for( i=0; i<4; i++ )
304                     {
305                         tk[i][j] ^=tk[i][j-1];

```

¹⁵Siehe Abschnitt 4.2.7 auf Seite 42.

```

306         }
307         for( i=0; i<4; i++ )
308         {
309             tk[i][4] ^= S[ tk[i][3] ];
310         }
311         for( j=5; j<KC; j++)
312             for( i=0; i<4; i++ )
313             {
314                 tk[i][j] ^= tk[i][j-1];
315             }
316     }
317
318     // copy values into round key array
319     for( j=0; (j<KC) && (t<(ROUNDS+1)*BC); j++, t++)
320         for( i=0; i<4; i++ )
321             {
322                 w[t/BC][i][t%BC] = tk[i][j];
323             }
324     }
325     return 0;
326 }

```

In Code 6.20 ist ein Verschlüsselungsalgorithmus dargestellt¹⁶.

Code 6.20: Encrypt Verschlüsselung

```

327 //=====
328
329 int Encrypt( word8 a[4][MAXBC], word8 rk[MAXROUNDS+1][4][MAXBC] )
330 {
331     int r;
332     AddRoundKey(a, rk[0]);
333     for( r=1; r
334         {
335             SubBytes(a, S);
336             ShiftRows(a, 0);
337             MixColumns(a);
338             AddRoundKey(a, rk[r]);
339         }
340     // last round is special: there is no MixColumns
341     SubBytes(a, S);
342     ShiftRows(a, 0);
343     AddRoundKey(a, rk[ROUNDS]);
344
345     return 0;
346 }

```

In Code 6.21 ist ein Entschlüsselungsalgorithmus dargestellt¹⁷. Die Verwendung derselben Funktion ist möglich bei AddRoundKey, weil XOR selbstinvers ist. Bei SubBytes wird die S-Box gegen InvS-Box ausgetauscht. In ShiftRows wird der Wert vom Index abgezogen, statt dazu gezählt.

Code 6.21: Decrypt Entschlüsselung

```

347 //=====
348 int Decrypt(word8 a[4][MAXBC], word8 rk[MAXROUNDS+1][4][MAXBC] )
349 {
350     int r;
351     AddRoundKey(a, rk[ROUNDS]);

```

¹⁶Siehe Abschnitt 4.3 auf Seite 50.

¹⁷Siehe Abschnitt 4.4 auf Seite 56.

```

352 SubBytes(a,Si);
353 ShiftRows(a,1);
354 for( r=ROUNDS-1; r>0; r-- )
355 {
356   AddRoundKey(a,rk[r]);
357   InvMixcolumns(a);
358   SubBytes(a,Si);
359   ShiftRows(a,1);
360 }
361
362 AddRoundKey(a,rk[0]);
363 return 0;
364 }

```

In Code 6.22 ist main Funktion dargestellt. Diese main Funktion unterscheidet sich von der Main aus [2] auf Seite 221. In der Funktion wurde neben dem Verschlüsselungsaufwurf auch ein Entschlüsselungsaufwurf eingebaut.

Code 6.22: main Funktion

```

365
366 int main()
367 {
368   int i;
369   int j;
370   word8 a[4][MAXBC];
371   word8 rk[MAXROUNDS+1][4][MAXBC];
372   word8 sk[4][MAXKC];
373
374   for( KC=4; KC<=8; KC++ )
375     for( BC=4; BC<=8; BC++ )
376     {
377       ROUNDS = numrounds[KC-4][BC-4];
378       for( j=0; j<BC; j++ )
379         for( i=0; i<4; i++ )
380         {
381           a[i][j] = 0;
382         }
383       for( j=0; j<KC; j++ )
384         for( i=0; i<4; i++ )
385         {
386           sk[i][j] = 0;
387         }
388       KeyExpansion(sk,rk);
389       Encrypt(a,rk);
390       printf("block length %d key length %d\n", 32*BC, 32*KC );
391       for( j=0; j<BC; j++ )
392         for( i=0; i<4; i++ )
393         {
394           printf("%02X", a[i][j] );
395         }
396       printf("\n");
397       Decrypt(a,rk);
398       for( j=0; j<BC; j++ )
399         for( i=0; i<4; i++ )
400         {
401           printf("%02X", a[i][j]);
402         }
403       printf("\n");
404       printf("\n");
405       for( j=0; j<BC; j++ )

```

```
406         for( i=0; i<4; i++ )
407             {
408                 a[i][j] = i;
409             }
410         Encrypt(a,rk);
411         for( j=0; j<4; j++ )
412             for( i=0; i<4; i++ )
413                 {
414                     printf("%02X", a[i][j]);
415                 }
416         printf("\n");
417         Decrypt(a,rk);
418         for( j=0; j<4; j++ )
419             for( i=0; i<4; i++ )
420                 {
421                     printf("%02X", a[i][j]);
422                 }
423         printf("\n");
424         printf("\n");
425     }
426     return 0;
427 }
```

7. Ergebnisse

7.1. Beitrag der Arbeit

Die Arbeit an der Diplomarbeit war für mich sehr aufschlussreich. Sie zeigte mir mit welchen einfachen Mitteln wir verschlüsseln und zugleich sehr komplexe Rätsel einem unbekannten Angreifer / Abhörer geben. Es ist ein ständiges Ringen zwischen einer schnell anwendbaren Technik und einer möglichst hohen Sicherheit. Besonderer Bedeutung kommt dem Verständnis von Verschlüsselungsalgorithmen durch ihre Anwender zu, denn nur wenn ein entsprechendes Grundverständnis vorhanden ist, ist auch eine korrekte Verwendung möglich. Dem in der Arbeit entwickelten Erklärungsmodell und dem Unterrichtskonzept kommt daher auch eine signifikante praktische Bedeutung zu.

7.2. Probleme

Die Schwierigkeit, welche sich ganz am Anfang stellte, war eine möglichst einfache mathematische Entsprechung für das Galoisfeld zu finden, ohne Verfälschung der Funktionen auf dem Körper. Die Invertierung einer Matrix auf einem Restklassenkörper war ein weiteres Problem, auf dessen Lösung ich während einer Nachhilfestunde stieß. Die Beschreibung des Zusammenhangs, besser gesagt der Unabhängigkeit von Schlüssellänge und Blocklänge, wurde nicht vollständig ausgearbeitet. Auch wurde nicht auf die verschiedenen Arten von Blockchiffren eingegangen da sie für mehr Verwirrung als Klarheit gesorgt hätten. Bei Erstellung der Beispiele versuchte ich aufbauend zu arbeiten und die Zahlenbereiche klein zu halten, es ging mir nicht um Rechenkünste sondern um das Verständnis der Arbeitsweise, daher wurde mit verschiedenen Zahlensystemen gearbeitet. Leider war die Paarung von modulo 11 mit dem Hexadezimalsystem unglücklich, sie wurde aber trotzdem beibehalten.

7.3. Zweck

Die Arbeit soll Verständnis für den Einfluss der Mathematik im Bereich der Verschlüsselung wecken. Es wurde daher versucht, den Unterbau von AES auf einfache Art und Weise zu erklären.

7.4. Weiterführende Arbeiten

Die Aufgabe ist definitiv noch ausbaufähig, das Verständnis des Zusammenwirkens von einfachen Operationen auf einem Körper könnte durch Klärung der folgenden Fragen gefördert werden. Auch eine Erklärung des Galoisfeldes und die Invertierung der Matrix, die bei MixColumns angewendet wird, wäre zu überlegen. Darstellung und Erklärung der Unterschiede der Blockchiffren Electronic Code Book (ECB), Cipher Block Chaining (CBC), Cipher Feedback (CFB) und Output Feedback (OFB) könnten noch eingearbeitet werden. Auch die Hamming Distanz könnte, anhand von Beispielen, erklärt und berechnet werden um ein Gefühl für die Bedeutung der Zahl zu bekommen.

A. Zusammenfassung

Als ich mich in das Thema einarbeitete, stellte ich fest, dass es zwei Arten von Beschreibungen gab. Die Einen waren häufig technischer Natur und beschäftigen sich mit Art und Weise der Implementierung, die Anderen beschäftigen sich mit dem mathematischen Aspekt des Themas, nicht jedoch mit dem Algorithmus. Diese Arbeiten vergleichen AES-Algorithmus mit anderen Algorithmen und versuchen die jeweiligen mathematischen Schwächen oder Stärken aufzuzeigen. Ich wollte die Lücke des nicht beschriebenen Algorithmus in der Mathematik schließen und versuchte das Thema so einfach wie möglich zu beschreiben, ohne es unnötig zu verfälschen. Der Leser soll von seinem mathematischen Niveau, entweder HTL oder AHS, abgeholt und in das Thema eingeführt werden, ohne seinen jeweiligen Wissensstand in der Mathematik zu sehr oder zu schnell zu erweitern und dadurch die Leserin oder den Leser zu verlieren. Ich hoffe, mein Ziel durch diese Arbeit erreicht zu haben.

B. Einfaches Beispiel

B.1. Schlüsselerweiterung

B.1.1. round key 2

Die Schritte und Ergebnisse um von round key 2 zu round key 3 zu kommen.

Tabelle B.1.: Einfaches
Beispiel: round key 2

	M	N	O	P
	Word 0 + Erweiterung	Word 1 + Word 4 Word 5	Word 2 + Word 5 Word 6	Word 3 + Word 6 Word 7
36	4	4	4	4
37	3	3	3	3
38	3	3	3	3
39	3	3	3	3

Tabelle B.2.: Einfaches
Beispiel: ShiftRow RC2

	R
36	3
37	3
38	3
39	4

Tabelle B.3.: Einfaches
Beispiel: SubBytes RC2

	T	U
	multiplikativ Inverses	lineare Funktion
36	10	4
37	10	4
38	10	4
39	22	11

Tabelle B.4.: Einfaches
Beispiel: Erweiterung RC2

	W
	plus RC(2)
36	24
37	4
38	4
39	11

Der Code für das erste Ergebnis für Tabelle B.1, Tabelle B.2, Tabelle B.3 und für Tabelle B.4 lautet:

round key 2

=REST(M30 + W30; \$D\$7)

=REST(N30 + M36; \$D\$7)

=REST(O30 + N36; \$D\$7)

=REST(P30 + O36; \$D\$7)

ShiftRows

=P37

SubBytes

multiplikativ Inverses =WVERWEIS(R36; {\$B\$22:\$K\$23\ \$B\$24:\$K\$25\ \$B\$26:\$J\$27};
2; FALSCH)

lineare Funktion =REST(\$B\$62 * T36 + \$E\$62; \$D\$7)

Erweiterung plus RC(3)

=REST(U36 + D31; \$D\$7)

=U37

B.1.2. round key 3

Die Schritte und Ergebnisse um von round key 3 zu round key 4 zu kommen.

Tabelle B.5.: Einfaches
Beispiel: round key 3

	M	N	O	P
	Word 4 + Erweiterung	Word 5 + Word 8	Word 6 + Word 9	Word 7 + Word 10
42	28	3	7	11
43	7	10	13	16
44	7	10	13	16
45	14	17	20	23

Tabelle B.6.: Einfaches
Beispiel: ShiftRow RC3

	R
42	16
43	16
44	23
45	11

Tabelle B.7.: Einfaches
Beispiel: SubBytes RC3

	T	
	multiplikativ Inverses	lineare Funktion
42	20	5
43	20	5
44	24	17
45	8	27

Tabelle B.8.: Einfaches
Beispiel: Erweiterung RC3

	W
	plus RC(3)
42	18
43	5
44	17
45	27

Der Code für das erste Ergebnis für Tabelle B.5, Tabelle B.6, Tabelle B.7 und für Tabelle B.8 lautet:

```
round key 3
=REST(M36 + W36; $D$7)
=REST(M42 + N36; $D$7)
=REST(N42 + O36; $D$7)
=REST(O42 + P36; $D$7)

ShiftRows
=P43
```

SubBytes

multiplikativ Inverses =WVERWEIS(R42; {\$B\$22:\$K\$23\ \$B\$24:\$K\$25\ \$B\$26:\$J\$27};
2; FALSCH)

lineare Funktion REST(\$B\$62*T42 + \$E\$62; \$D\$7)

Erweiterung plus RC(3)

=REST(U42 + D32; \$D\$7)

=U43

B.1.3. round key 4

Tabelle B.9.: Einfaches
Beispiel: round key 4

	M	N	O	P
	Word 8 + Erweiterung	Word 9 + Word 12	Word 10 + Word 13	Word 11 + Word 14
48	17	20	27	9
49	12	22	6	22
50	24	5	18	5
51	12	0	20	14

Der Code in der zweiten Zeile Nummer 48 für Tabelle B.9 lautet:

round key 4

=REST(W42 + M42; \$D\$7)

=REST(M48 + N42; \$D\$7)

=REST(N48 + O42; \$D\$7)

=REST(O48 + P42; \$D\$7)

B.2. Algorithmus

B.2.1. Zweite Runde

SubBytes

Tabelle B.10.: Einfaches
Beispiel: 2. Runde SubBytes
multiplikativ Inverse

	B	C	D	E
90	12	15	22	10
91	26	20	7	15
92	19	26	26	24
93	28	26	19	19

Tabelle B.11.: Einfaches
Beispiel: 2. Runde SubBytes
lineare Funktion

	G	H	I	J
90	10	19	11	4
91	23	5	24	19
92	2	23	23	17
93	0	23	2	2

Der Code für das erste Ergebnis für sowohl Tabelle B.10 als auch für Tabelle B.11 lautet:

SubBytes

multiplikativ Inverses =WVERWEIS(B83; {\$B\$22:\$K\$23\ \$B\$24:\$K\$25\ \$B\$26:\$J\$27};
2; FALSCH)

lineare Funktion =REST(\$B\$62 * B90 + \$E\$62; \$D\$7)

ShiftRows

Tabelle B.12.: Einfaches
Beispiel: 2. Runde ShiftRows

	B	C	D	E
96	10	19	11	4
97	5	24	19	23
98	23	17	2	23
99	2	0	23	2

Der Code für das erste Ergebnis für Tabelle B.12 lautet:

Shift Rows

=mmult(mmult(mmult(mmult(G90:J90; \$G\$71:\$J\$74); \$G\$71:\$J\$74); \$G\$71:\$J\$74);
\$G\$71:\$J\$74)

MixColumns

Tabelle B.13.: Einfaches
Beispiel: 2. Runde MixColumns

	B	C	D	E
102	5	24	19	23
103	2	0	23	2
104	10	19	11	4
105	23	17	2	23

Tabelle B.14.: Einfaches
Beispiel: 2. Runde MixColumns
Modulo 29

	G	H	I	J
102	5	24	19	23
103	2	0	23	2
104	10	19	11	4
105	23	17	2	23

Der Code für das erste Ergebnis für sowohl Tabelle B.13 als auch für Tabelle B.14 lautet:

MixColumns

=mmult(\$M\$77:\$P\$80; B96:E99)

MixColumns Moudulo 29

=REST(B102; \$D\$7)

AddRoundKey

Tabelle B.15.: Einfaches
Beispiel: 2. Runde AddRoundKey

	B	C	D	E
108	4	27	26	5
109	9	10	7	18
110	17	0	24	20
111	8	5	22	17

Tabelle B.16.: Einfaches
Beispiel: 2. Runde round key 3

	G	H	I	J
108	28	3	7	11
109	7	10	13	16
110	7	10	13	16
111	14	17	20	23

Der Code für das erste Ergebnis für sowohl Tabelle [B.15](#) als auch für Tabelle [B.16](#) lautet:

AddRoundKey

=REST(G102 + G108; \$D\$7)

round key 3

=M42

B.3. Entschlüsselung

B.3.1. Zweite Runde

InvAddRoundKey

Tabelle B.17.: Einfaches
Beispiel: Entschlüsselung
2. Runde InvAddRoundKey
roundkey 3

	B	C	D	E
44	28	3	7	11
45	7	10	13	16
46	7	10	13	16
47	14	17	20	23

Tabelle B.18.: Einfaches
Beispiel: Entschlüsselung
2. Runde InvAddroundKey

	G	H	I	J
44	5	24	19	23
45	2	0	23	2
46	10	19	11	4
47	23	17	2	23

Der Code für das erste Ergebnis für sowohl Tabelle B.17 als auch für Tabelle B.18 lautet:

round key 3

= 'Verschlüsselung '!M42

InvAddRoundKey

=REST(G37 - B44; 'Verschlüsselung '!'\$D\$7)

InvMixColumns

Tabelle B.19.: Einfaches
Beispiel: Entschlüsselung
2. Runde InvMixColumns

	B	C	D	E
55	10	19	11	4
56	5	24	19	23
57	23	17	2	23
58	2	0	23	2

Tabelle B.20.: Einfaches
Beispiel: Entschlüsselung
2. Runde InvMixColumns
Modulo 29

	G	H	I	J
55	10	19	11	4
56	5	24	19	23
57	23	17	2	23
58	2	0	23	2

Der Code für das erste Ergebnis für sowohl Tabelle B.19 als auch für Tabelle B.20 lautet:

MixColumns

=mmult(\$G\$49:\$J\$52; G44:J47)

MixColumns Modulo 29

=REST(RUNDEN[19]¹(B55; 0); 'Verschlüsselung '!'\$D\$7)

¹Die Funktion „Runden“ war nötig, weil das Tabellenkalkulationsprogramm GoogleDocs im Jahr 2018 numerisch nicht stabil genug rechnet.

InvShiftRows

Tabelle B.21.: Einfaches
Beispiel: Entschlüsselung
2. Runde ShiftRows

	B	C	D	E
60	10	19	11	4
61	23	5	24	19
62	2	23	23	17
63	0	23	2	2

Der Code für das erste Ergebnis für Tabelle B.21 lautet:

Shift Rows

```
=mmult( mmult( mmult( mmult(G55:J55; 'Verschlüsselung '!$G$71:$J$74);  
'Verschlüsselung '!$G$71:$J$74); 'Verschlüsselung '!$G$71:$J$74); 'Verschlüsselung  
'!$G$71:$J$74)
```

InvSubBytes

Tabelle B.22.: Einfaches
Beispiel: 2. Runde SubBytes
multiplikativ Inverse

	B	C	D	E
90	12	15	22	10
91	26	20	7	15
92	19	26	26	24
93	28	26	19	19

Tabelle B.23.: Einfaches
Beispiel: 2. Runde SubBytes
lineare Funktion

	G	H	I	J
90	10	19	11	4
91	23	5	24	19
92	2	23	23	17
93	0	23	2	2

Der Code für das erste Ergebnis für sowohl Tabelle B.22 als auch für Tabelle B.23 lautet:

SubBytes

multiplikativ Inverses =WVERWEIS(B83; {\$B\$22:\$K\$23\ \$B\$24:\$K\$25\ \$B\$26:\$J\$27};
2; FALSCH)

lineare Funktion =REST(\$B\$62 * B90 + \$E\$62; \$D\$7)

B.3.2. Erste Runde**InvAddRoundKey**

Tabelle B.24.: Einfaches
Beispiel: Entschlüsselung
1. Runde InvAddRoundKey
round key 2

	B	C	D	E
73	4	4	4	4
74	3	3	3	3
75	3	3	3	3
76	3	3	3	3

Tabelle B.25.: Einfaches
Beispiel: Entschlüsselung
1. Runde InvAddRoundKey

	G	H	I	J
73	13	27	0	28
74	16	13	22	28
75	23	16	16	20
76	25	16	23	23

Der Code für das erste Ergebnis für sowohl Tabelle B.24 als auch für Tabelle B.25 lautet:

round key 2

= 'Verschlüsselung '!M36

InvAddRoundKey

=REST(G66 - B73; 'Verschlüsselung '!'\$D\$7)

InvMixColumns

Tabelle B.26.: Einfaches
Beispiel: Entschlüsselung
1. Runde InvMixColumns

	B	C	D	E
79	23	16	16	20
80	13	27	0	28
81	25	16	23	23
82	16	13	22	28

Tabelle B.27.: Einfaches
Beispiel: Entschlüsselung
1. Runde InvMixColumns
Modulo 29

	G	H	I	J
79	23	16	16	20
80	13	27	0	28
81	25	16	23	23
82	16	13	22	28

Der Code für das erste Ergebnis für sowohl Tabelle B.26 als auch für Tabelle B.27 lautet:

MixColumns

=mmult(\$G\$49:\$J\$52; G73:J76)

MixColumns Modulo 29

=REST(RUNDEN(B55; 0); 'Verschlüsselung '!'\$D\$7)

InvShiftRows

Tabelle B.28.: Einfaches
Beispiel: Entschlüsselung
1. Runde ShiftRows

	B	C	D	E
84	23	16	16	20
85	28	13	27	0
86	23	23	25	16
87	13	22	28	16

Der Code für das erste Ergebnis für Tabelle B.28 lautet:

Shift Rows

=mmult(mmult(mmult(mmult(G79:J79; 'Verschlüsselung '!'\$G\$71:\$J\$74);
'Verschlüsselung '!'\$G\$71:\$J\$74); 'Verschlüsselung '!'\$G\$71:\$J\$74); 'Verschlüsselung
'!'\$G\$71:\$J\$74)

B.3.3. InvVorunde**InvAddRoundKey**

Tabelle B.29.: Einfaches
Beispiel: Entschlüsselung
Vorrunde InvAddRoundKey
round key 1

	B	C	D	E
97	0	0	0	0
98	0	0	0	0
99	0	0	0	0
100	0	0	0	0

Tabelle B.30.: Einfaches
Beispiel: Entschlüsselung
Vorrunde InvAddroundKey

	G	H	I	J
97	19	27	27	7
98	21	9	11	28
99	19	19	12	27
100	9	20	21	27

Der Code für das erste Ergebnis für sowohl Tabelle B.29 als auch für Tabelle B.30 lautet:

round key 1

= 'Verschlüsselung '!M30

InvAddRoundKey

=REST(G90-B97; 'Verschlüsselung '!'\$D\$7)

C. AES

C.1. AES Schlüsselexpansion

C.1.1. round key bei N_6

Tabelle C.1.: AES: cipher key bei N_6

	Word 0	Word 1	Word 2	Word 3	Word 4	Word 5
Feld 1	00	00	00	00	00	00
Feld 2	00	00	00	00	00	00
Feld 3	00	00	00	00	00	00
Feld 4	00	00	00	00	00	00

Tabelle C.2.: AES: ShiftRow, SubBytes, RC 1 und Erweiterung

ShiftRow	SubBytes S-Box	RC 1	Erweiterung
00	63	1	62
00	63	1100011	63
00	63	XOR	63
00	63	1100010	63

Tabelle C.3.: AES: Berechnung des 6. Word in N_6

Word 0		Erweiterung		Word 6
00		62		62
00	XOR	63	=	63
00		63		63
00		63		63

Tabelle C.4.: AES: Berechnung des 7. Word in N_6

Word 1		Word 6		Word 7
00		62		62
00	XOR	63	=	63
00		63		63
00		63		63

Tabelle C.5.: AES: Berechnung des 8. Word in N_6

Word 2		Word 7		Word 8
00		62		62
00	XOR	63	=	63
00		63		63
00		63		63

Tabelle C.6.: AES: Berechnung des 9. Word in N_6

Word 3		Word 8		Word 9
00		62		62
00	XOR	63	=	63
00		63		63
00		63		63

Tabelle C.7.: AES: Berechnung des 10. Word in N_6

Word 4		Word 9		Word 10
00		62		62
00	XOR	63	=	63
00		63		63
00		63		63

Tabelle C.8.: AES: Berechnung des 11. Word in N_6

Word 5		Word 10		Word 11
00		62		62
00	XOR	63	=	63
00		63		63
00		63		63

In Tabelle C.3, Tabelle C.4, Tabelle C.5, Tabelle C.6, Tabelle C.7 und Tabelle C.8 werden die weiteren Spalten der Key Expansion berechnet. Um die Rechnung nachzuvollziehen zu können, muss sie bitweise ausgeführt werden.

round key 1**Tabelle C.9.:** AES: round key 1

	Word 6	Word 7	Word 8	Word 9	Word 10	Word 11
Feld 1	62	62	62	62	62	62
Feld 2	63	63	63	63	63	63
Feld 3	63	63	63	63	63	63
Feld 4	63	63	63	63	63	63

Tabelle C.10.: AES: ShiftRow, SubBytes, RC 2 und Erweiterung

ShiftRow	SubBytes S-Box	RC 2	Erweiterung
63	FB	10	F9
63	FB	11111011	FB
63	FB	XOR	FB
62	AA	11111001	AA

Tabelle C.11.: AES: Berechnung des 12. Word in N_6

Word 6		Erweiterung		Word 12
62		F9		9B
63	XOR	FB	=	98
63		FB		98
63		AA		C9

Tabelle C.12.: AES: Berechnung des 13. Word in N_6

Word 7		Word 12		Word 13
62		9B		F9
63	XOR	98	=	FB
63		98		FB
63		C9		AA

Tabelle C.13.: AES: Berechnung des
14. Word in N_6

Word 8		Word 13		Word 14
62		F9		9B
63	XOR	FB	=	98
63		FB		98
63		AA		C9

Tabelle C.14.: AES: Berechnung des
15. Word in N_6

Word 9		Word 14		Word 15
62		9B		F9
63	XOR	98	=	FB
63		98		FB
63		C9		AA

Tabelle C.15.: AES: Berechnung des
16. Word in N_6

Word 10		Word 15		Word 16
62		F9		9B
63	XOR	FB	=	98
63		FB		98
63		AA		C9

Tabelle C.16.: AES: Berechnung des
17. Word in N_6

Word 11		Word 16		Word 17
62		9B		F9
63	XOR	98	=	FB
63		98		FB
63		C9		AA

In Tabelle C.11, Tabelle C.12, Tabelle C.13, Tabelle C.14, Tabelle C.15 und Tabelle C.16 werden die weiteren Spalten der Key Expansion berechnet. Um die Rechnung nachzuvollziehen zu können, muss sie bitweise ausgeführt werden.

round key 2

Tabelle C.17.: AES: round key 2

	Word 12	Word 13	Word 14	Word 15	Word 16	Word 17
Feld 1	9B	F9	9B	F9	9B	F9
Feld 2	98	FB	98	FB	98	FB
Feld 3	98	FB	98	FB	98	FB
Feld 4	C9	AA	C9	AA	C9	AA

Tabelle C.18.: AES: ShiftRow, Sub-Bytes, RC 3 und Erweiterung

ShiftRow	SubBytes S-Box	RC 3	Erweiterung
FB	0F	100	0B
FB	0F	1111	0F
AA	AC	XOR	AC
F9	99	1011	99

Tabelle C.19.: AES: Berechnung des 18. Word in N_6

Word 12		Erweiterung		Word 18
9B		0B		90
98	XOR	0F	=	97
98		AC		34
C9		99		50

Tabelle C.20.: AES: Berechnung des 19. Word in N_6

Word 13		Word 18		Word 19
F9		90		69
FB	XOR	97	=	6C
FB		34		CF
AA		50		FA

Tabelle C.21.: AES: Berechnung des 20. Word in N_6

Word 14		Word 19		Word 20
9B		69		F2
98	XOR	6C	=	F4
98		CF		57
C9		FA		33

Tabelle C.22.: AES: Berechnung des 21. Word in N_6

Word 15		Word 20		Word 21
F9		F2		0B
FB	XOR	F4	=	0F
FB		57		AC
AA		33		99

Tabelle C.23.: AES: Berechnung des 22. Word in N_6

Word 16		Word 21		Word 22
9B		0B		90
98	XOR	0F	=	97
98		AC		34
C9		99		50

Tabelle C.24.: AES: Berechnung des 23. Word in N_6

Word 17		Word 22		Word 23
F9		90		69
FB	XOR	97	=	6C
FB		34		CF
AA		50		FA

In Tabelle C.19, Tabelle C.20, Tabelle C.21, Tabelle C.22, Tabelle C.23 und Tabelle C.24 werden die weiteren Spalten der Key Expansion berechnet. Um die Rechnung nachzuvollziehen zu können, muss sie bitweise ausgeführt werden.

round key 3

Tabelle C.25.: AES: round key 3 bei N_6

	Word 18	Word 19	Word 20	Word 21	Word 22	Word 23
Feld 1	90	69	F2	0B	90	69
Feld 2	97	6C	F4	0F	97	6C
Feld 3	34	CF	57	AC	34	CF
Feld 4	50	FA	33	99	50	FA

C.1.2. round key bei N_8

Tabelle C.26.: AES: cipher key

	Word 0	Word 1	Word 2	Word 3	Word 4	Word 5	Word 6	Word 7
Feld 1	00	00	00	00	00	00	00	00
Feld 2	00	00	00	00	00	00	00	00
Feld 3	00	00	00	00	00	00	00	00
Feld 4	00	00	00	00	00	00	00	00

Tabelle C.27.: AES: ShiftRow, SubBytes, RC 1 und Erweiterung

ShiftRow	SubBytes S-Box	RC 1	Erweiterung
00	63	0000 0001	62
00	63	0110 0011	63
00	63	XOR	63
00	63	0110 0010	63

In Tabelle C.27 wurden in der Spalte ShiftRow die Felder um eins verschoben. In der Spalte SubBytes wurden die Felder aus ShiftRow mittels der S-Box getauscht. In RC wurde im ersten Feld die round constante mittels XOR dazugegeben, das Ergebnis steht in der Spalte Erweiterung.

Tabelle C.28.: AES: Berechnung des 8. Word in N_8

Word 0		Erweiterung	Word 8
00		62	62
00	XOR	63	= 63
00		63	63
00		63	63

Tabelle C.29.: AES: Berechnung des 9. Word in N_8

Word 1		Word 8	Word 9
00		62	62
00	XOR	63	= 63
00		63	63
00		63	63

Tabelle C.30.: AES: Berechnung des
10. Word in N_8

Word 2		Word 9		Word 10
00		62		62
00	XOR	63	=	63
00		63		63
00		63		63

Tabelle C.31.: AES: Berechnung des
11. Word in N_8

Word 3		Word 10		Word 11
00		62		62
00	XOR	63	=	63
00		63		63
00		63		63

Tabelle C.32.: AES: SubBytes von Word 11 bei N_8

	SubBytes S-Box
62	AA
63	FB
63	FB
63	FB

In Tabelle C.32 wird an Word 11 die Funktion SubBytes angewendet, bevor Word 11 in Tabelle C.33 verwendet werden kann.

Tabelle C.33.: AES: Berechnung des
12. Word in N_8

Word 4		Word 11		Word 12
00		AA		AA
00	XOR	FB	=	FB
00		FB		FB
00		FB		FB

Tabelle C.34.: AES: Berechnung des
13. Word in N_8

Word 5		Word 12		Word 13
00		AA		AA
00	XOR	FB	=	FB
00		FB		FB
00		FB		FB

Tabelle C.35.: AES: Berechnung des
14. Word in N_8

Word 6		Word 13		Word 14
00		AA		AA
00	XOR	FB	=	FB
00		FB		FB
00		FB		FB

Tabelle C.36.: AES: Berechnung des
15. Word in N_8

Word 7		Word 14		Word 15
00		AA		AA
00	XOR	FB	=	FB
00		FB		FB
00		FB		FB

In Tabelle C.28, Tabelle C.29, Tabelle C.30, Tabelle C.31, Tabelle C.33, Tabelle C.34, Tabelle C.35 und Tabelle C.36 werden die weiteren Spalten der Key Expansion berechnet. Um die Rechnung nachzuvollziehen zu können, muss sie bitweise ausgeführt werden.

round key 1

Tabelle C.37.: AES: round key 1

	Word 8	Word 9	Word 10	Word 11	Word 12	Word 13	Word 14	Word 15
Feld 1	62	62	62	62	AA	AA	AA	AA
Feld 2	63	63	63	63	FB	FB	FB	FB
Feld 3	63	63	63	63	FB	FB	FB	FB
Feld 4	63	63	63	63	FB	FB	FB	FB

Tabelle C.38.: AES: ShiftRow, SubBytes, RC 2
und Erweiterung

ShiftRow	SubBytes S-Box	RC 2	Erweiterung
FB	0F	0000 0010	0D
FB	0F	0000 1111	0F
FB	0F	XOR	0F
AA	AC	0000 1101	AC

In Tabelle C.38 wurden in der Spalte ShiftRow die Felder um eins verschoben. In der Spalte SubBytes wurden die Felder aus ShiftRow mittels der S-Box getauscht. In RC wurde im ersten Feld die round constante mittels XOR dazugegeben, das Ergebnis steht in der Spalte Erweiterung.

Tabelle C.39.: AES: Berechnung des
16. Word in N_8

Word 8		Erweiterung		Word 16
62		0D		6F
63	XOR	0F	=	6C
63		0F		6C
63		AC		CF

Tabelle C.40.: AES: Berechnung des
17. Word in N_8

Word 9		Word 16		Word 17
62		6F		0D
63	XOR	6C	=	0F
63		6C		0F
63		CF		AC

Tabelle C.41.: AES: Berechnung des 18. Word in N_8

Word 10		Word 17		Word 18
62		0D		6F
63	XOR	0F	=	6C
63		0F		6C
63		AC		CF

Tabelle C.42.: AES: Berechnung des 19. Word in N_8

Word 11		Word 18		Word 19
62		6F		0D
63	XOR	6C	=	0F
63		6C		0F
63		CF		AC

Tabelle C.43.: AES: SubBytes von Word 19 bei N_8

	SubBytes S-Box
0D	D7
0F	76
0F	76
AC	91

In Tabelle C.43 wird an Word 19 die Funktion SubBytes angewendet, bevor Word 19 in Tabelle C.44 verwendet werden kann.

Tabelle C.44.: AES: Berechnung des 20. Word in N_8

Word 12		Word 19		Word 20
AA		D7		7D
FB	XOR	76	=	8D
FB		76		8D
FB		91		6A

Tabelle C.45.: AES: Berechnung des 21. Word in N_8

Word 13		Word 20		Word 21
AA		7D		D7
FB	XOR	8D	=	76
FB		8D		76
FB		6A		91

Tabelle C.46.: AES: Berechnung des 22. Word in N_8

Word 14		Word 21		Word 22
AA		D7		7D
FB	XOR	76	=	8D
FB		76		8D
FB		91		6A

Tabelle C.47.: AES: Berechnung des 23. Word in N_8

Word 15		Word 22		Word 23
AA		7D		D7
FB	XOR	8D	=	76
FB		8D		76
FB		6A		91

In Tabelle C.39, Tabelle C.40, Tabelle C.41, Tabelle C.42, Tabelle C.44, Tabelle C.45, Tabelle C.46 und Tabelle C.47 werden die weiteren Spalten der Key Expansion berechnet. Um die Rechnung nachzuvollziehen zu können, muss sie bitweise ausgeführt werden.

round key 2

Tabelle C.48.: AES: round key 2

	Word 16	Word 17	Word 18	Word 19	Word 20	Word 21	Word 22	Word 23
Feld 1	6F	0D	6F	0D	7D	D7	7D	D7
Feld 2	6C	0F	6C	0F	8D	76	8D	76
Feld 3	6C	0F	6C	0F	8D	76	8D	76
Feld 4	CF	AC	CF	AC	6A	91	6A	91

Tabelle C.49.: AES: ShiftRow, SubBytes, RC 3 und Erweiterung

ShiftRow	SubBytes S-Box	RC 3	Erweiterung
76	38	0000 0100	3C
76	38	0011 1000	38
91	81	XOR	81
D7	0E	0011 1100	0E

In Tabelle C.49 wurden in der Spalte ShiftRow die Felder um eins verschoben. In der Spalte SubBytes wurden die Felder aus ShiftRow mittels der S-Box getauscht. In RC wurde im ersten Feld die round constante mittels XOR dazugegeben, das Ergebnis steht in der Spalte Erweiterung.

Tabelle C.50.: AES: Berechnung des 16. Word in N_8

Word 16		Erweiterung		Word 24
6F		3C		53
6C	XOR	38	=	54
6C		81		ED
CF		0E		C1

Tabelle C.51.: AES: Berechnung des 17. Word in N_8

Word 17		Word 24		Word 25
0D		53		5E
0F	XOR	54	=	5B
0F		ED		E2
AC		C1		6D

Tabelle C.52.: AES: Berechnung des
18. Word in N_8

Word 18		Word 25		Word 26
6F		5E		31
6C	XOR	5B	=	37
6C		E2		8E
CF		6D		A2

Tabelle C.53.: AES: Berechnung des
19. Word in N_8

Word 19		Word 26		Word 27
0D		31		3C
0F	XOR	37	=	38
0F		8E		81
AC		A2		0E

Tabelle C.54.: AES: SubBytes von Word 27 bei N_8

	SubBytes S-Box
3C	EB
38	07
81	0C
0E	AB

In Tabelle C.54 wird an Word 27 die Funktion SubBytes angewendet, bevor Word 27 in Tabelle C.55 verwendet werden kann.

Tabelle C.55.: AES: Berechnung des
20. Word in N_8

Word 20		Word 27		Word 28
7D		EB		96
8D	XOR	07	=	8A
8D		0C		81
6A		AB		C1

Tabelle C.56.: AES: Berechnung des
21. Word in N_8

Word 21		Word 28		Word 29
D7		96		41
76	XOR	8A	=	FC
76		81		F7
91		C1		50

Tabelle C.57.: AES: Berechnung des
22. Word in N_8

Word 22		Word 29		Word 30
7D		41		3C
8D	XOR	FC	=	71
8D		F7		7A
6A		50		3A

Tabelle C.58.: AES: Berechnung des
23. Word in N_8

Word 23		Word 30		Word 31
D7		3C		EB
76	XOR	71	=	07
76		7A		0C
91		3A		AB

In Tabelle C.50, Tabelle C.51, Tabelle C.52, Tabelle C.53, Tabelle C.55, Tabelle C.56, Tabelle C.57 und Tabelle C.58 werden die weiteren Spalten der Key Expansion berechnet. Um die Rechnung nachzuvollziehen zu können, muss sie bitweise ausgeführt werden.

round key 3

Tabelle C.59.: AES: round key 3 bei N_8

	Word 24	Word 25	Word 26	Word 27	Word 28	Word 29	Word 30	Word 31
Feld 1	53	5E	31	3C	96	41	3C	EB
Feld 2	54	5B	37	38	8A	FC	71	7
Feld 3	ED	E2	8E	81	81	F7	7A	0C
Feld 4	C1	6D	A2	0E	C1	50	3A	AB

C.2. AES Algorithmus

Die vollständige Berechnung, von der in MixColumns ausgeführten Matrixmultiplikation.

$$\begin{pmatrix} \underbrace{A_{0,0}}_{\cong C_0^1} & \underbrace{A_{0,1}}_{\cong C_0^2} & A_{0,2} & A_{0,3} \\ A_{1,0} & A_{1,1} & A_{1,2} & A_{1,3} \\ A_{2,0} & A_{2,1} & A_{2,2} & A_{2,3} \\ A_{3,0} & A_{3,1} & A_{3,2} & A_{3,3} \end{pmatrix} = \begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix} \times \begin{pmatrix} 63 & F2 & 30 & FE \\ 6B & 01 & D7 & 7C \\ 67 & AB & 77 & 6F \\ 76 & 7B & C5 & 2B \end{pmatrix}$$

Die Berechnung von $A_{0,0}^1 A_{0,1}^2$ wurde schon vorne berechnet. $A_{1,0}$ wird wie folgt berechnet.

Tabelle C.60.: Berechnung von Feld $A_{1,0}$

$$\begin{pmatrix} 01 \\ 02 \\ 03 \\ 01 \end{pmatrix} \times \begin{pmatrix} 63 \\ 6B \\ 67 \\ 76 \end{pmatrix}$$

$$\begin{array}{r} 67 \times 03 = \\ 0110 \ 0111 \times 11 \\ \hline 0 \ 1100 \ 1110 \\ 0110 \ 0111 \\ \hline 1010 \ 1001 \end{array}$$

$$\begin{array}{r} 76 \times 01 = \\ 0111 \ 0110 \times 01 \\ \hline 0111 \ 0110 \end{array}$$

$$\begin{array}{r} 63 \times 01 = \\ 0110 \ 0011 \times 01 \\ \hline 0110 \ 0011 \\ 6B \times 02 = \\ 0110 \ 1011 \times 10 \\ \hline 0 \ 1101 \ 0110 \end{array}$$

Die Summe aller vier Teilergebnisse entspricht 6A.

$$\begin{array}{r} 0110 \ 0011 \\ 1101 \ 0110 \\ 1010 \ 1001 \\ 0111 \ 0110 \\ \hline 0110 \ 1010 \end{array}$$

Tabelle C.61.: Berechnung von Feld $A_{2,0}$

$$\begin{pmatrix} 01 \\ 01 \\ 02 \\ 03 \end{pmatrix} \times \begin{pmatrix} 63 \\ 6B \\ 67 \\ 76 \end{pmatrix}$$

$$\begin{array}{r} 63 \times 01 = \\ 0110 \ 0011 \times 01 \\ \hline 0110 \ 0011 \\ 6B \times 01 = \\ 0110 \ 1011 \times 01 \\ \hline 0110 \ 1011 \end{array}$$

¹Siehe Abschnitt 4.3.2 auf Seite 53.

²Siehe Abschnitt 4.3.2 auf Seite 54.

67×02 =

0110 0111 × 10

0 1100 1110

76×03 =

0111 0110 × 11

0 1110 1100

0111 0110

1001 1010

Die Summe aller vier Teil-
ergebnisse entspricht 5C.

0110 0011

0110 1011

1100 1110

1001 1010

0101 1100

Tabelle C.62.: Berechnung von Feld $A_{3,0}$

63×03 =

0110 0011 × 11

0 1100 0110

0110 0011

1010 0101

6B×01 =

0110 1011 × 01

0110 1011

$\begin{pmatrix} 03 \\ 01 \\ 01 \\ 02 \end{pmatrix} \times \begin{pmatrix} 63 \\ 6B \\ 67 \\ 76 \end{pmatrix}$

Die Summe aller vier Teil-
ergebnisse entspricht 45.

1010 0101

0110 1011

0110 0111

1110 1100

0100 0101

67×01 =

0110 0111 × 01

0110 0111

76×02 =

0111 0110 × 10

0 1110 1100

Tabelle C.63.: Berechnung von Feld $A_{1,1}$

F2×01 =

1111 0010 × 01

1111 0010

01×10 =

0000 0001 × 10

0 0000 0010

$\begin{pmatrix} 01 \\ 02 \\ 03 \\ 01 \end{pmatrix} \times \begin{pmatrix} F2 \\ 01 \\ AB \\ 7B \end{pmatrix}$

Die Summe aller vier Teil-
ergebnisse entspricht 6D.

1 1111 1101 ÷ 1 0001 1011 = 1

1 0001 1011

0 1110 0110 Rest

AB×11 =

1010 1011 × 11

1 0101 0110

7B×01 =

0111 1011 × 01

0111 1011

1111 0010

0000 0010

1110 0110

0111 1011

0110 1101

Tabelle C.64.: Berechnung von Feld $A_{2,1}$

$$\begin{pmatrix} 01 \\ 01 \\ 02 \\ 03 \end{pmatrix} \times \begin{pmatrix} F2 \\ 01 \\ AB \\ 7B \end{pmatrix}$$

$$\begin{array}{r} AB \times 02 = \\ 1010 \ 1011 \times 10 \\ \hline 1 \ 0101 \ 0110 \\ 7B \times 03 = \\ 0111 \ 1011 \times 11 \\ \hline 0 \ 1111 \ 0110 \\ 0111 \ 1011 \\ \hline 1000 \ 1101 \end{array}$$

$$\begin{array}{r} 1 \ 0101 \ 0110 \div 1 \ 0001 \ 1011 = 1 \\ 1 \ 0001 \ 1011 \\ \hline 0 \ 0100 \ 1101 \quad \text{Rest} \end{array}$$

Die Summe aller vier Teil-
ergebnisse entspricht 32.

$$\begin{array}{r} F2 \times 01 = \\ 1111 \ 0010 \times 01 \\ \hline 1111 \ 0010 \\ 01 \times 01 = \\ 0000 \ 0001 \times 01 \\ \hline 0000 \ 0001 \end{array}$$

$$\begin{array}{r} 1111 \ 0010 \\ 0000 \ 0001 \\ 0100 \ 1101 \\ 1000 \ 1101 \\ \hline 0011 \ 0011 \end{array}$$

Tabelle C.65.: Berechnung von Feld $A_{3,1}$

$$\begin{pmatrix} 03 \\ 01 \\ 01 \\ 02 \end{pmatrix} \times \begin{pmatrix} F2 \\ 01 \\ AB \\ 7B \end{pmatrix}$$

$$\begin{array}{r} AB \times 01 = \\ 1010 \ 1011 \times 01 \\ \hline 1010 \ 1011 \\ 7B \times 10 = \\ 0111 \ 1011 \times 10 \\ \hline 0 \ 1111 \ 0110 \end{array}$$

$$\begin{array}{r} 0000 \ 1101 \\ 0000 \ 0001 \\ 1010 \ 1011 \\ 1111 \ 0110 \\ \hline 0101 \ 0001 \end{array}$$

$$\begin{array}{r} 1 \ 0001 \ 0110 \div 1 \ 0001 \ 1011 = 1 \\ 1 \ 0001 \ 1011 \\ \hline 0 \ 0000 \ 1101 \end{array}$$

Die Summe aller vier Teil-
ergebnisse entspricht 51.

$$\begin{array}{r} F2 \times 03 = \\ 1111 \ 0010 \times 11 \\ \hline 1 \ 1110 \ 0100 \\ 1111 \ 0010 \\ \hline 1 \ 0001 \ 0110 \\ 01 \times 01 = \\ 0000 \ 0001 \times 01 \\ \hline 0000 \ 0001 \end{array}$$

Tabelle C.66.: Berechnung von Feld $A_{0,2}$

$$\begin{pmatrix} 02 \\ 03 \\ 01 \\ 01 \end{pmatrix} \times \begin{pmatrix} 30 \\ D7 \\ 77 \\ C5 \end{pmatrix}$$

$$\begin{array}{r} 30 \times 02 = \\ 0011 \ 0000 \times 10 \\ \hline 0110 \ 0000 \\ D7 \times 03 = \\ 1101 \ 0111 \times 11 \\ \hline 1 \ 1010 \ 1110 \\ 1101 \ 0111 \\ \hline 1 \ 0111 \ 1001 \end{array}$$

77×01

=

0111 0111

×

01

0111 0111

C5×01

=

1100 0101

×

01

1100 0101

0110 0000

0110 0010

0111 0111

1100 0101

1011 0000

1 0111 1001 ÷ 1 0001 1011 = 1

1 0001 1011

0 0110 0010

Die Summe aller vier Teil-

ergebnisse entspricht B0.

Tabelle C.67.: Berechnung von Feld A_{1,2}

30×01

=

0011 0000

×

01

0011 0000

D7×02

=

1101 0111

×

10

1 1010 1110

77×03

=

0111 0111

×

11

0 1110 1110

0111 0111

1001 1001

C5×01

=

1100 0101

×

01

1100 0101

0011 0000

1011 0101

1001 1001

1100 0101

1101 1001

1 1010 1110 ÷ 1 0001 1011 = 1

1 0001 1011

0 1011 0101

Die Summe aller vier Teil-

ergebnisse entspricht D9.

Tabelle C.68.: Berechnung von Feld A_{2,2}

30×01

=

0011 0000

×

01

0011 0000

D7×01

=

1101 0111

×

01

1101 0111

77×02

=

0111 0111

×

10

0 1110 1110

C5×03

=

1100 0101

×

11

1 1000 1010

1100 0101

1 0100 1111

Die Summe aller vier Teil-

ergebnisse entspricht 5D.

0011 0000

1101 0111

1110 1110

0101 0100

0101 1101

1 0100 1111 ÷ 1 0001 1011 = 1

1 0001 1011

0101 0100

Rest

Tabelle C.69.: Berechnung von Feld $A_{3,2}$

$$\begin{pmatrix} 03 \\ 01 \\ 01 \\ 02 \end{pmatrix} \times \begin{pmatrix} 30 \\ D7 \\ 77 \\ C5 \end{pmatrix}$$

$$\begin{array}{r} 30 \times 03 = \\ 0011\ 0000 \times 11 \\ \hline 0\ 0110\ 0000 \\ 0011\ 0000 \\ \hline 0101\ 0000 \\ D7 \times 01 = \\ 1101\ 0111 \times 01 \\ \hline 1101\ 0111 \end{array}$$

$$\begin{array}{r} 77 \times 01 = \\ 0111\ 0111 \times 01 \\ \hline 0111\ 0111 \\ C5 \times 02 = \\ 1100\ 0101 \times 10 \\ \hline 1\ 1000\ 1010 \end{array}$$

Die Summe aller vier Teilergebnisse entspricht 61.

$$\begin{array}{r} 1\ 1000\ 1010 \div 1\ 0001\ 1011 = 1 \\ 1\ 0001\ 1011 \\ \hline 1001\ 0001 \end{array}$$

$$\begin{array}{r} 0101\ 0000 \\ 1101\ 0111 \\ 0111\ 0111 \\ 1001\ 0001 \\ \hline 0110\ 0001 \end{array}$$

Tabelle C.70.: Berechnung von Feld $A_{0,3}$

$$\begin{pmatrix} 02 \\ 03 \\ 01 \\ 01 \end{pmatrix} \times \begin{pmatrix} FE \\ 7C \\ 6F \\ 2B \end{pmatrix}$$

$$\begin{array}{r} FE \times 02 = \\ 1111\ 1110 \times 10 \\ \hline 1\ 1111\ 1100 \\ 7C \times 03 = \\ 0111\ 1100 \times 11 \\ \hline 0\ 1111\ 1000 \\ 0111\ 1100 \\ \hline 1000\ 0100 \end{array}$$

$$\begin{array}{r} 6F \times 01 = \\ 0110\ 1111 \times 01 \\ \hline 0110\ 1111 \\ 2B \times 01 = \\ 0010\ 1011 \times 01 \\ \hline 0010\ 1011 \end{array}$$

$$\begin{array}{r} 1110\ 0111 \\ 1000\ 0100 \\ 0110\ 1111 \\ 0010\ 1011 \\ \hline 0010\ 0111 \end{array}$$

$1\ 1111\ 1100 \div 1\ 0001\ 1011 = 1$

$$\begin{array}{r} 1\ 0001\ 1011 \\ \hline 1110\ 0111 \end{array}$$

Die Summe aller vier Teilergebnisse entspricht 27.

Tabelle C.71.: Berechnung von Feld $A_{1,3}$

$$\begin{pmatrix} 01 \\ 02 \\ 03 \\ 01 \end{pmatrix} \times \begin{pmatrix} FE \\ 7C \\ 6F \\ 2B \end{pmatrix}$$

$$\begin{array}{r} FE \times 01 = \\ 1111\ 1110 \times 01 \\ \hline 1111\ 1110 \\ 7C \times 02 = \\ 0111\ 1100 \times 10 \\ \hline 0\ 1111\ 1000 \end{array}$$

6F×03 =

0110 1111 × 11

0 1101 1110

0110 1111

1011 0001

2B×01 =

0010 1011 × 01

0010 1011

1111 1110

1111 1000

1011 0001

0010 1011

1001 1100

Die Summe aller vier Teil-
ergebnisse entspricht 9C.

Tabelle C.72.: Berechnung von Feld $A_{2,3}$

$\begin{pmatrix} 01 \\ 01 \\ 02 \\ 03 \end{pmatrix} \times \begin{pmatrix} FE \\ 7C \\ 6F \\ 2B \end{pmatrix}$

6F×02 =

0110 1111 × 10

0 1101 1110

2B×03 =

0010 1011 × 11

0 0101 0110

0010 1011

0111 1101

FE×01 =

1111 1110 × 01

1111 1110

7C×01 =

0111 1100 × 01

0111 1100

1111 1110

0111 1100

1101 1110

0111 1101

0010 0001

Die Summe aller vier Teil-
ergebnisse entspricht 21.

Tabelle C.73.: Berechnung von Feld $A_{3,3}$

$\begin{pmatrix} 03 \\ 01 \\ 01 \\ 02 \end{pmatrix} \times \begin{pmatrix} FE \\ 7C \\ 6F \\ 2B \end{pmatrix}$

6F×01 =

0110 1111 × 01

0110 1111

2B×02 =

0010 1011 × 10

0101 0110

FE×03 =

1111 1110 × 11

1 1111 1100

1111 1110

1 0000 0010

7C×01 =

0111 1100 × 01

0111 1100

0001 1001

0111 1100

0110 1111

0101 0110

0101 1100

1 0000 0010 ÷ 1 0001 1011 = 1

1 0001 1011

0001 1001

Die Summe aller vier Teil-
ergebnisse entspricht 5C.

D. Übungsbeispiele

D.1. Beispiele Schlüsselexpansion

D.1.1. Beispiele zur Berechnung der multiplikativ Inversen

Es werden die multiplikativ Inversentafeln für verschiedene Zahlen erstellt. Die Tafeln für die Primzahlen 5, 7, 11 und 13 sind eindeutig.¹ Die Tafeln für 4, 6, 8 und 9 sind nicht eindeutig, weil sie Nullteiler² haben. Es wird zur Berechnung der erweiterte Euklidische Algorithmus benötigt.

D.1.2. Aufgabenstellung

Bestimmen Sie die multiplikativ Inversentafeln für 4, 5, 6, 7, 8, 9, 11 und 13.

D.1.3. Lösung

Tabelle D.1.: Beispiel: Restklassenkörper 5

	B	C	D	E	F
2	0	1	2	3	4
3	0	1	3	2	4

Tabelle D.2.: Beispiel: Restklassenkörper 7

	B	C	D	E	F	G	H
6	0	1	2	3	4	5	6
7	0	1	4	5	2	3	6

Tabelle D.3.: Beispiel: Restklassenkörper 11

	B	C	D	E	F	G	H	I	J	K	L
10	0	1	2	3	4	5	6	7	8	9	10
11	0	1	6	4	3	9	2	8	7	5	10

Tabelle D.4.: Beispiel: Restklassenkörper 13

	B	C	D	E	F	G	H	I	J	K	L	M	N
14	0	1	2	3	4	5	6	7	8	9	10	11	12
15	0	1	7	9	10	8	11	2	5	3	4	6	12

¹Siehe Theorem 7 auf Seite 9.

²Siehe Definition 23 auf Seite 7.

Tabelle D.5.: Beispiel: Restklassen 4

	B	C	D	E
Element	0	1	2	3
Inverses	0	1	2	3
Restklasse	0	1	0	1

Tabelle D.6.: Beispiel: Restklassen 6

	B	C	D	E	F	G
Element	0	1	2	3	4	5
Inverses	0	1	3	2	3	5
Restklasse	0	1	0	0	0	1

Tabelle D.7.: Beispiel: Restklassen 8

	B	C	D	E	F	G	H	I
Element	0	1	2	3	4	5	6	7
Inverses	0	1	4	3	2	5	4	7
Restklasse	0	1	0	1	0	1	0	1

Tabelle D.8.: Beispiel: Restklassen 9

	B	C	D	E	F	G	H	I	J
Element	0	1	2	3	4	5	6	7	8
Inverses	0	1	5	3	7	2	3	4	8
Restklasse	0	1	1	0	1	1	0	1	1

D.2. Beispiele RC round constante

Im folgendem sollen die verschiedenen RC für verschiedene Zahlensysteme und verschiedene Modulo berechnet werden. Es werden das Binär- und Hexadezimalsystem als alternative Zahlensysteme gewählt.

D.2.1. Aufgabenstellung

Bestimmen Sie die ersten vier round constante (RC) im Dezimalsystem Modulo 5, 7, 11 und 13.

D.2.2. Lösung

Tabelle D.9.: Beispiel: RC im Dezimalsystem

	A	B	C	D	E	F	G
1	Basis 10		Wert	Modulo 5	Modulo 7	Modulo 11	Modulo 13
2	RC(1)	1	1	1	1	1	1
3	RC(2)	2	10	0	3	10	10
4	RC(3)	3	100	0	2	1	9
5	RC(4)	4	1000	0	6	10	12

D.2.3. Aufgabenstellung

Bestimmen Sie die ersten vier round constante (RC) im Binärsystem³ Modulo 7, sowie im Hexadezimalsystem Modulo 11⁴.

D.2.4. Lösung

Tabelle D.10.: Beispiel: RC im Binärsystem

	A	B	C	D	E	F
7	Basis 2		Wert	Modulo 7	Wert in Binär	Wert Modulo 7 in Binär
8	RC(1)	1	1	1	1	001
9	RC(2)	2	2	2	10	010
10	RC(3)	3	4	4	100	100
11	RC(4)	4	8	1	1000	001

³In GoogleDoc gib es dazu entsprechende Funktionen. Siehe [10] und [8].

⁴In GoogleDoc gib es dazu entsprechende Funktionen. Siehe [11] und [12].

Tabelle D.11.: Beispiel: RC im Hexadezimalsystem

	A	B	C	D	E	F
13	Basis 16		Wert	Modulo 11	Wert in Hexadezimal	Modulo 11 Hexdezimal
14	RC(1)	1	1	1	1	1
15	RC(2)	2	16	5	10	5
16	RC(3)	3	256	3	100	3
17	RC(4)	4	4096	4	1000	4

D.3. Beispiel RC, round key und Key Expansion

In den folgenden Beispielen werden für verschiedene Startwerte sowie verschiedene lineare Funktionen und Zahlensysteme eine Schlüsselerweiterung berechnet.

D.3.1. Aufgabenstellung

Berechnen Sie eine Key Expansion (Schlüsselerweiterung) aus dem cipher key

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$$

im Dezimalsystem Modulo 29 mit der linearen Funktion $y = 3x + 3$.

D.3.2. Lösung

Tabelle D.12.: Beispiel: Schlüsselerweiterung Modulo 29

	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1					ShiftRows		SubBytes			RC(1) Basis 10 Modulo 29	Ergebnis		Word 0 + Erweiterung	Word 1 + Word 3	Word 2 + Word 4
2	1	2	3		6		5	18		1	19		20	22	25
3	4	5	6		9		13	13		0	13		17	22	28
4	7	8	9		3		10	4		0	4		11	19	28

Der Code in der zweiten Zeile Nummer 2 für Tabelle D.12 lautet:

ShiftRows

=D3

SubBytes

multiplikativ Inverses =WVERWEIS(F2; { 'Multiplikativ Inverses '!'\$B\$35:\$K\$36\
'Multiplikativ Inverses '!'\$B\$37:\$K\$38\
'Multiplikativ Inverses '!'\$B\$39:\$J\$40 };
2; FALSCH)

lineare Funktion =REST(3 * H2 + 3; 29)**Ergebnis**

=REST(I2 + K2; 29)

Word 0 + Erweiterung **Word 1 + Word 3** **Word 2 + Word 4**
=REST(B2 + L2; 29) =REST(C2 + N2; 29) =REST(D2 + O2; 29)

D.3.3. Aufgabenstellung

Berechnen Sie eine Key Expansion (Schlüsselerweiterung) aus dem round key (Rundenschlüssel)

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$$

im Dezimalsystem mit der RC (round constante) 3 Modulo 13 mit der linearen Funktion $y = 5x + 1$.

D.3.4. Lösung**Tabelle D.13.:** Beispiel: Schlüsselerweiterung Modulo 13

6	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
7	Word 6	Word 7	Word 8		ShiftRows		SubBytes			RC(3) Basis 10 Modulo 13	Ergebnis		Word 6 + Erweiterung	Word 7 + Word 10	Word 8 + Word 11
8	1	2	3		6		11	4		9	0		1	3	6
9	4	5	6		9		3	3		0	3		7	12	5
10	7	8	9		3		9	7		0	7		1	9	5

Der Code in der zweiten Zeile Nummer 8 für Tabelle [D.13](#) lautet:

ShiftRows

=D9

SubBytes

multiplikativ Inverses =WVERWEIS(F8; 'Multiplikativ Inverses '!'\$B\$14:\$N\$36; 2;
FALSCH)

lineare Funktion =REST(5 * H8 + 1; 13)

Ergebnis

REST(I8 + K8; 13)
Word 6 + Erweiterung Word 7 + Word 10 Word 8 + Word 11
=REST(B8 + L8; 13) =REST(C8 + N8; 13) =REST(D8 + O8; 13)

D.3.5. Aufgabenstellung

Berechnen Sie eine Key Expansion (Schlüsselerweiterung) aus dem round key (Rundenschlüssel)

(001 010 011)
(100 101 110)
(000 001 010)

im Binärsystem mit der RC (round constante) 2 Modulo 7 mit der linearen Funktion y = x + 010.

D.3.6. Lösung

Tabelle D.14.: Beispiel: Schlüsselerweiterung Modulo 7

	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
13	Word 3	Word 4	Word 5		ShiftRows		SubBytes			RC(2) Basis 2 Modulo 7	Ergebnis		Word 3 + Erweiterung	Word 4 + Word 7	Word 5 + Word 8
14	001	010	011		110		110	001		010	011		100	110	010
15	100	101	110		010		100	110		000	110		011	001	000
16	000	001	010		011		101	000		000	000		000	001	011

Der Code in der zweiten Zeile Nummer 14 für Tabelle D.14 lautet:

ShiftRows
=D15

SubBytes

multiplikativ Inverses =DEZINBIN⁵(WVERWEIS(BININDEZ⁶(F14); 'Multiplikativ
Inverses '\$B\$6: \$H\$7 ;2; WAHR); 3)

lineare Funktion =DEZINBIN(REST(BININDEZ(H14) + 2; 7); 3)

Ergebnis
=DEZINBIN(REST(BININDEZ(K14) + BININDEZ(I14); 7); 3)

⁵Siehe [10].
⁶Siehe [8].

Word 6 + Erweiterung
 =DEZINBIN(REST(
 BININDEZ(L14) + BININ-
 DEZ(B14); 7); 3)

Word 7 + Word 10
 =DEZINBIN(REST(
 BININDEZ(N14) + BININ-
 DEZ(C14); 7); 3)

Word 8 + Word 11
 =DEZINBIN(REST(
 BININDEZ(O14) + BININ-
 DEZ(D14); 7); 3)

D.3.7. Aufgabenstellung

Berechnen Sie eine Key Expansion (Schlüsselerweiterung) aus dem round key (Rundenschlüssel)

$$\begin{pmatrix} 1 & 2 & A \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$$

im Hexadezimalsystem mit der RC (round constante) 4 Modulo 11 mit der linearen Funktion $y = 2x$.

D.3.8. Lösung

Tabelle D.15.: Beispiel: Schlüsselerweiterung Modulo 11

	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
20	Word 9	Word 10	Word 11		ShiftRows		SubBytes			RC(4) Basis 16 Modulo 11	Ergebnis		Word 9 + Erweiterung	Word 10 + Word 13	Word 11 + Word 14
21	1	2	A		6		2	4		4	8		9	0	A
22	4	5	6		9		5	A		0	A		3	8	3
23	7	8	9		A		A	9		0	9		5	2	0

Der Code in der zweiten Zeile Nummer 21 für Tabelle D.15 lautet:

ShiftRows

=D22

SubBytes

multiplikativ Inverses =DEZINHEX⁷(WVERWEIS(HEXINDEZ⁸(F21); 'Multiplikativ Inverses '!'\$B\$10: \$L\$11; 2; WAHR); 1)

lineare Funktion =DEZINHEX(REST(HEXINDEZ(H21) * 2; 11); 1)

Ergebnis

=DEZINHEX(REST(HEXINDEZ(K21) + HEXINDEZ(I21); 11); 1)

⁷Siehe [11].

⁸Siehe [12].

Word 9 + Erweiterung
=DEZINHEX(REST(HEXIN-
XINDEZ(L21) + HEXIN-
DEZ(B21); 11); 1)

Word 10 + Word 13
=DEZINHEX(REST(HEXIN-
XINDEZ(L22) + HEXIN-
DEZ(B22); 11); 1)

Word 11 + Word 14
=DEZINHEX(REST(HEXIN-
XINDEZ(L23) + HEXIN-
DEZ(B23); 11); 1)

D.4. Beispiele Verschlüsselungsrunden

Zum Berechnen der Beispiele werden die Lösungen der vorangegangenen Beispiele⁹ benötigt.

D.4.1. Aufgabenstellung

Berechnen Sie die nächste Verschlüsselungsrunde im Dezimalsystem mit diesen Angaben¹⁰:
Modulo: 29

lineare Funktion: $y = 3x + 3$

Tabelle D.16.: Beispiel:
Verschlüsselung
Modulo 29 round
key

	D	E	F
1	20	22	25
2	5	10	16
3	24	3	12

Tabelle D.17.: Beispiel:
Verschlüsselung
Modulo 29 Ausgangsmatrix

	I	J	K
1	6	16	28
2	19	0	8
3	12	16	1

Tabelle D.18.: Beispiel:
Verschlüsselung
Modulo 29 MixColumns-Matrix

	D	E	F
10	1	0	0
11	0	0	1
12	0	1	0

D.4.2. Lösung

Tabelle D.19.: Beispiel: Verschlüsselung Modulo 29 SubBytes multiplikativ Inverse

	D	E	F
6	5	20	28
7	26	0	11
8	17	20	1

Tabelle D.20.: Beispiel: Verschlüsselung Modulo 29 SubBytes lineare Funktion

	I	J	K
6	18	5	0
7	23	3	7
8	25	5	6

Tabelle D.21.: Beispiel: Verschlüsselung Modulo 29 ShiftRows Permutationsmatrix

	N	O	P
6	0	0	1
7	1	0	0
8	0	1	0

Tabelle D.22.: Beispiel: Verschlüsselung Modulo 29 ShiftRows Ergebnis

	R	S	T
6	18	5	0
7	3	7	23
8	6	25	5

⁹Siehe Abschnitt D.1 auf Seite 105.

¹⁰Siehe Abschnitt D.3.1 auf Seite 108.

Tabelle D.23.: Beispiel: Verschlüsselung Modulo 29 MixColumns Permutationsmatrix

	D	E	F
10	1	0	0
11	0	0	1
12	0	1	0

Tabelle D.24.: Beispiel: Verschlüsselung Modulo 29 MixColumns Ergebnis

	I	J	K
10	18	0	5
11	3	23	7
12	6	5	25

Tabelle D.25.: Beispiel: Verschlüsselung Modulo 29 AddRoundKey round key

	N	O	P
10	20	22	25
11	5	10	16
12	24	3	12

Tabelle D.26.: Beispiel: Verschlüsselung Modulo 29 AddRoundKey Ergebnis

	R	S	T
10	9	22	1
11	20	16	6
12	17	24	24

Der Code für das erste Ergebnis für Tabelle D.19, Tabelle D.20, Tabelle D.22, Tabelle D.24 und für Tabelle D.26 lautet:

SubBytes

multiplikativ Inverses =WVERWEIS(I1; { 'Multiplikativ Inverses '!B\$35:\$K\$36 \ 'Multiplikativ Inverses '!B\$37:\$K\$38 \ 'Multiplikativ Inverses '!B\$39:\$J\$40 }; 2; FALSCH())

lineare Funktion =REST(3 * D6 + 3; 29)

ShiftRows

=mmult(mmult(mmult(I6:K6; \$N\$6:\$P\$8); \$N\$6:\$P\$8); \$N\$6:\$P\$8)

MixColumns

=mmult(R6:T8; D10:F12)

AddRoundKey

=REST(I10 + N10; 29)

D.4.3. Aufgabenstellung

Berechnen Sie die letzte Verschlüsselungsrunde im Dezimalsystem mit diesen Angaben¹¹:
Modulo: 13

lineare Funktion: $y = 5x + 1$

¹¹Siehe Abschnitt D.3.3 auf Seite 109.

Tabelle D.27.: Beispiel: Verschlüsselung Modulo 13 round key

	D	E	F
15	1	3	6
16	7	12	5
17	1	9	5

Tabelle D.28.: Beispiel: Verschlüsselung Modulo 13 Ausgangsmatrix

	I	J	K
15	6	16	28
16	19	0	8
17	12	16	1

D.4.4. Lösung**Tabelle D.29.:** Beispiel: Verschlüsselung Modulo 13 SubBytes multiplikativ Inverse

	D	E	F
20	11	12	12
21	12	0	5
22	12	12	1

Tabelle D.30.: Beispiel: Verschlüsselung Modulo 13 SubBytes lineare Funktion

	I	J	K
20	4	9	9
21	9	1	0
22	9	9	6

Tabelle D.31.: Beispiel: Verschlüsselung Modulo 13 ShiftRows Permutationsmatrix

	N	O	P
20	0	0	1
21	1	0	0
22	0	1	0

Tabelle D.32.: Beispiel: Verschlüsselung Modulo 13 ShiftRows Ergebnis

	R	S	T
20	11	12	12
21	12	0	5
22	12	12	1

Tabelle D.33.: Beispiel: Verschlüsselung Modulo 13 AddRoundKey round key

	N	O	P
24	1	3	6
25	7	12	5
26	1	9	5

Tabelle D.34.: Beispiel: Verschlüsselung Modulo 13 AddRoundKey Ergebnis

	R	S	T
24	5	12	2
25	8	12	1
26	7	5	1

Der Code für das erste Ergebnis für Tabelle D.29, Tabelle D.30, Tabelle D.32 und für Tabelle D.34 lautet:

SubBytes

multiplikativ Inverses =WVERWEIS(I15; 'Multiplikativ Inverses '!\$B\$14:\$N\$15; 2; WAHR())

lineare Funktion =REST(5 * D20 + 1; 13)

ShiftRows

```
=mmult( mmult( mmult( I20:K20; $N$20:$P$22); $N$20:$P$22); $N$20:$P$22)
```

AddRoundKey

```
=REST(N24 + R20; 13)
```

D.4.5. Aufgabenstellung

Berechnen Sie die nächste Verschlüsselungsrunde im Binärsystem mit diesen Angaben¹²:
Modulo: 7

lineare Funktion: $y = x + 010$

Tabelle D.35.: Beispiel:
Verschlüsselung
Modulo 7 round
key

	D	E	F
29	100	110	010
30	011	001	000
31	000	001	011

Tabelle D.36.: Beispiel:
Verschlüsselung
Modulo 7 Aus-
gangsmatrix

	I	J	K
29	110	101	100
30	011	010	000
31	110	101	100

Tabelle D.37.: Beispiel:
Verschlüsselung
Modulo 7 MixCol-
umns-Matrix

	D	E	F
38	1	0	0
39	0	0	1
40	0	1	0

D.4.6. Lösung

Tabelle D.38.: Beispiel: Verschlüs-
selung Modulo 7 SubBytes multi-
plikativ Inverse

	D	E	F
35	110	011	010
35	101	100	000
36	110	011	010

Tabelle D.39.: Beispiel: Verschlüs-
selung Modulo 7 SubBytes lineare
Funktion

	I	J	K
34	001	101	100
35	000	110	010
36	001	101	100

Tabelle D.40.: Beispiel: Verschlüs-
selung Modulo 7 ShiftRows
Permutationsmatrix

	N	O	P
34	0	0	1
35	1	0	0
36	0	1	0

Tabelle D.41.: Beispiel: Verschlüs-
selung Modulo 7 ShiftRows
Ergebnis

	R	S	T
34	001	101	100
35	110	010	000
36	100	001	101

¹²Siehe Abschnitt D.3.5 auf Seite 110.

Tabelle D.42.: Beispiel: Verschlüsselung Modulo 7 MixColumns Permutationsmatrix

	D	E	F
38	1	0	0
39	0	0	1
40	0	1	0

Tabelle D.43.: Beispiel: Verschlüsselung Modulo 7 MixColumns Ergebnis

	I	J	K
38	001	100	101
39	110	000	010
40	100	101	001

Tabelle D.44.: Beispiel: Verschlüsselung Modulo 7 AddRoundKey round key

	N	O	P
38	100	110	010
39	011	001	000
40	000	001	011

Tabelle D.45.: Beispiel: Verschlüsselung Modulo 7 AddRoundKey Ergebnis

	R	S	T
38	101	011	000
39	010	001	010
40	100	110	100

Der Code für das erste Ergebnis für Tabelle D.38, Tabelle D.39, Tabelle D.41, Tabelle D.43 und für Tabelle D.45 lautet:

SubBytes

multiplikativ Inverses =DEZINBIN(WVERWEIS(BININDEZ(I29); 'Multiplikativ Inverses '!\$B\$6:\$H\$7; 2; 1); 3)

lineare Funktion =DEZINBIN(REST(BININDEZ(D34) BININDEZ(10); 7); 3)

ShiftRows

=ArrayFormula¹³(DEZINBIN(mmult(mmult(mmult(ArrayFormula(BININDEZ(I34:K34)); \$N\$34:\$P\$36); \$N\$34:\$P\$36); \$N\$34:\$P\$36); 3))

MixColumns

=ArrayFormula(DEZINBIN(mmult(ArrayFormula(BININDEZ(R34:T36)); D38:F40); 3))

AddRoundKey

=DEZINBIN(REST(BININDEZ(N38) + BININDEZ(I38); 7); 3)

D.4.7. Aufgabenstellung

Berechnen Sie die nächste Verschlüsselungsrunde im Hexadezimalsystem mit diesen Angaben¹⁴:

Modulo: 11

lineare Funktion: $y = 2x$

¹³Siehe [7].

¹⁴Siehe Abschnitt D.3.7 auf Seite 111.

Tabelle D.46.: Beispiel:
Verschlüsselung
Modulo 11 round
key

	D	E	F
43	9	0	A
44	3	8	3
45	5	2	0

Tabelle D.47.: Beispiel:
Verschlüsselung
Modulo 11 Aus-
gangsmatrix

	I	J	K
43	A	9	8
44	7	6	5
45	4	3	2

Tabelle D.48.: Beispiel:
Verschlüsselung
Modulo 11 MixCol-
umns-Matrix

	D	E	F
52	1	0	0
53	0	0	1
54	0	1	0

D.4.8. Lösung

Tabelle D.49.: Beispiel: Verschlüs-
selung Modulo 11 SubBytes
multiplikativ Inverse

	I	J	K
48	A	5	7
49	8	2	9
50	3	4	6

Tabelle D.50.: Beispiel: Verschlüs-
selung Modulo 11 SubBytes lineare
Funktion

	I	J	K
48	4	A	1
49	2	7	3
50	8	9	0

Tabelle D.51.: Beispiel: Verschlüs-
selung Modulo 11 ShiftRows
Permutationsmatrix

	N	O	P
48	0	0	1
49	1	0	0
50	0	1	0

Tabelle D.52.: Beispiel: Verschlüs-
selung Modulo 11 ShiftRows
Ergebnis

	R	S	T
48	4	A	1
49	7	3	2
50	0	8	9

Tabelle D.53.: Beispiel: Verschlüs-
selung Modulo 11 MixColumns
Permutationsmatrix

	D	E	F
52	1	0	0
53	0	0	1
54	0	1	0

Tabelle D.54.: Beispiel: Verschlüs-
selung Modulo 11 MixColumns
Ergebnis

	I	J	K
52	4	1	A
53	7	2	3
54	0	9	8

Tabelle D.55.: Beispiel: Verschlüsselung Modulo 11 AddRoundKey round key

	N	O	P
52	9	0	A
53	3	8	3
54	5	2	0

Tabelle D.56.: Beispiel: Verschlüsselung Modulo 11 AddRoundKey Ergebnis

	R	S	T
52	2	1	9
53	A	A	6
54	5	0	8

Der Code für das erste Ergebnis für Tabelle D.49, Tabelle D.50, Tabelle D.52, Tabelle D.54 und für Tabelle D.56 lautet:

SubBytes

multiplikativ Inverses =DEZINHEX(WVERWEIS(HEXINDEZ(I43); 'Multiplikativ Inverses '!'\$B\$10:\$L\$11; 2; FALSCH()); 1)

lineare Funktion =DEZINHEX(REST(HEXINDEZ(D48) + HEXINDEZ(10); 11); 1)

ShiftRows

=ArrayFormula(DEZINHEX(mmult(mmult(mmult(ArrayFormula(HEXINDEZ(I48:K48)); \$N\$48:\$P\$50); \$N\$48:\$P\$50); \$N\$48:\$P\$50); 1))

MixColumns

=ArrayFormula(DEZINHEX(mmult(ArrayFormula(HEXINDEZ(R48:T50)); D52:F54); 1))

AddRoundKey

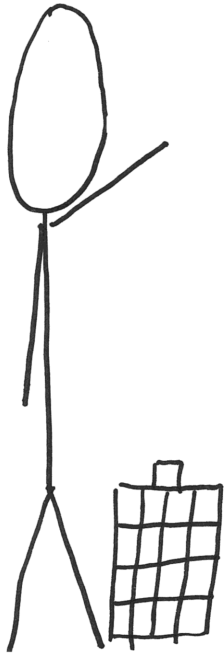
=DEZINHEX(REST(HEXINDEZ(N52) + HEXINDEZ(I52); 11); 1)

E. A Stick Figure Guide to the Advanced Encryption Standard (AES)

Diese Vorstellung des AES Algorithmus „A Stick Figure Guide to the Advanced Encryption Standard (AES)“¹ kann als Einführung benutzt werden.

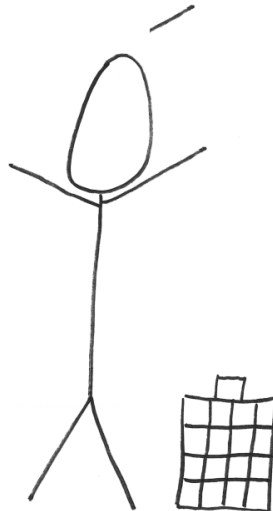
¹Die Präsentation ist sowohl als PowerPoint als auch PDF verfügbar.[3]

A Stick Figure Guide to the Advanced Encryption Standard (AES)



© Copyright 2009, Jeff Moser
<http://www.moserware.com/>

I handle petabytes* of data every day. From encrypting juicy Top Secret intelligence to boring packets bound for your Wifi router, I do it all!



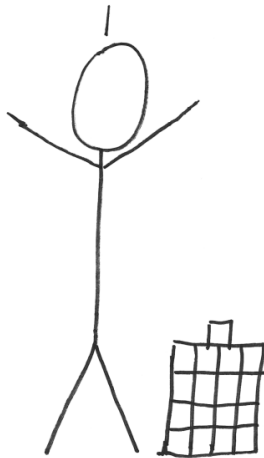
Act 1: Once Upon a Time...

... and still no one seems to care about me or my story.

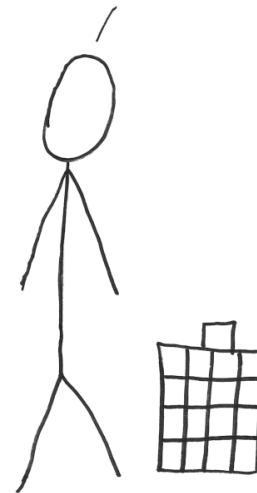


* 1 petabyte $\approx$ a lot

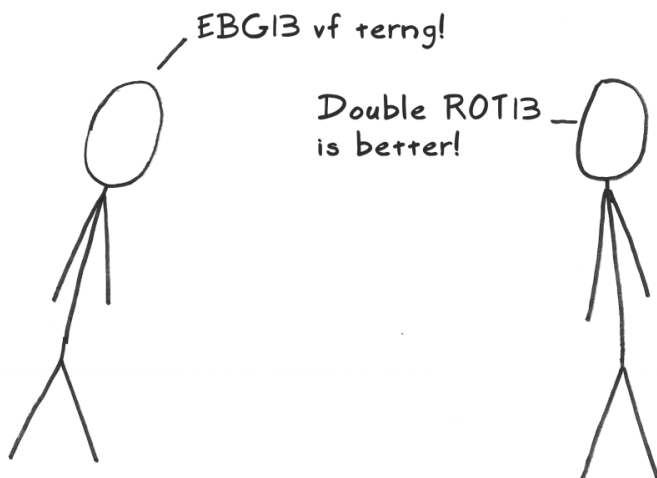
I've got a better-than-Cinderella story as I made my way to become king of the block cipher world.



Whoa! You're still there. You want to hear it? Well let's get started...



Once upon a time,* there was no good way for people outside secret agencies to judge good crypto.



EBG13 vf tcrng!

Double ROT13 is better!

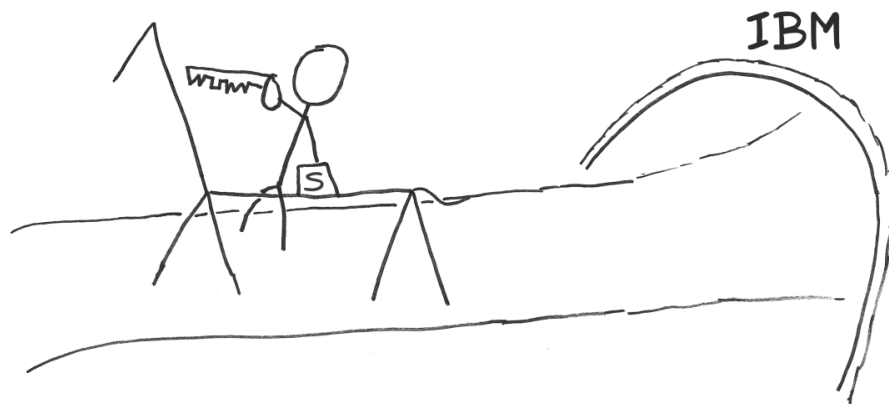
A decree went throughout the land to find a good, secure, algorithm.



We need a good cipher!

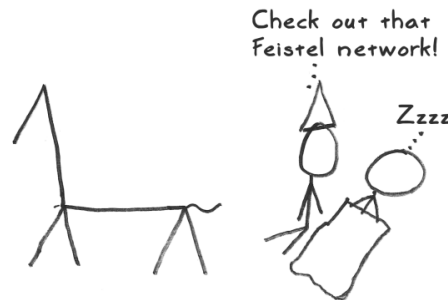
* ~ pre-1975 for the general public

One worthy competitor named Lucifer came forward.

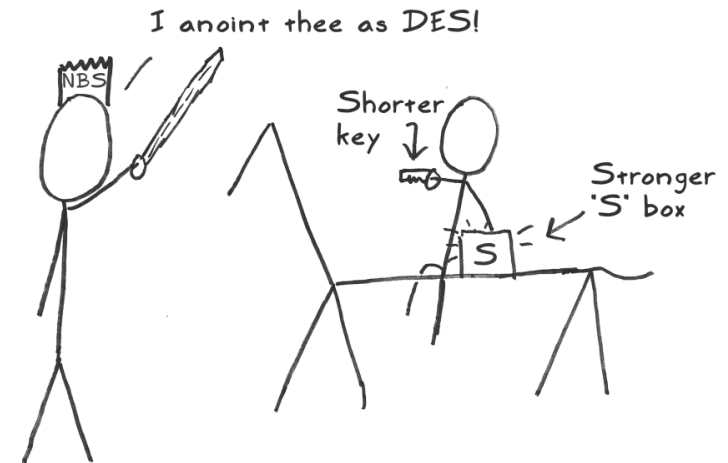


DES ruled in the land for over 20 years. Academics studied him intently. For the first time, there was something specific to look at. The modern field of cryptography was born.

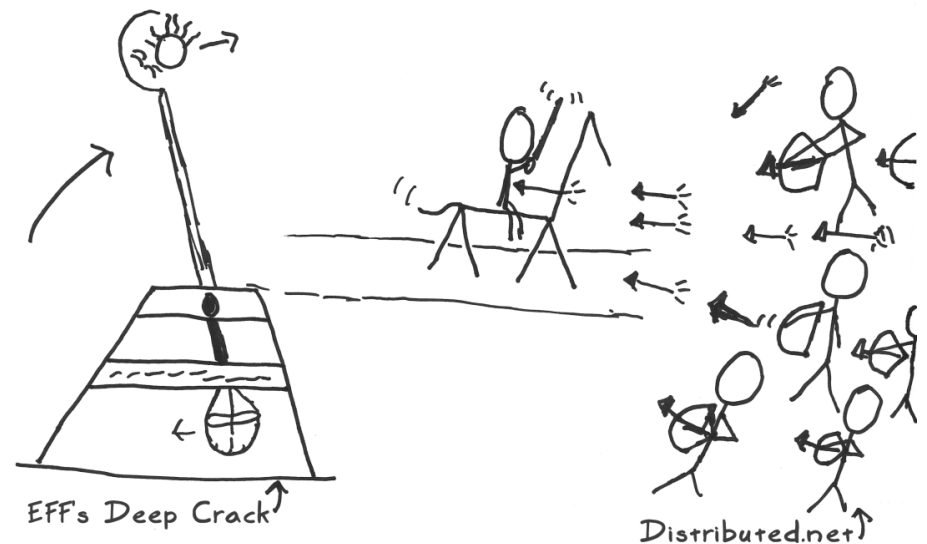
"... to the best of our knowledge, DES is free from any statistical or mathematical weakness."



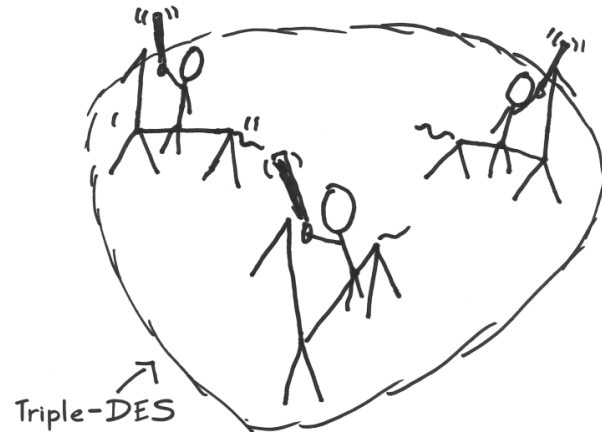
After being modified by the National Security Agency (NSA), he was anointed as the Data Encryption Standard (DES).



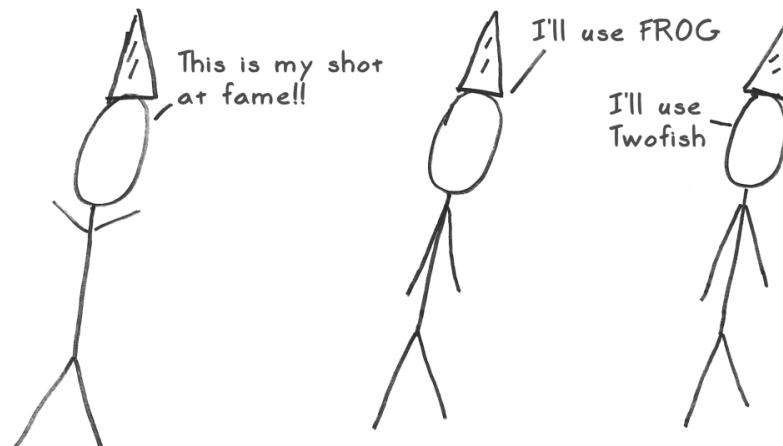
Over the years, many attackers challenged DES. He was defeated in several battles.



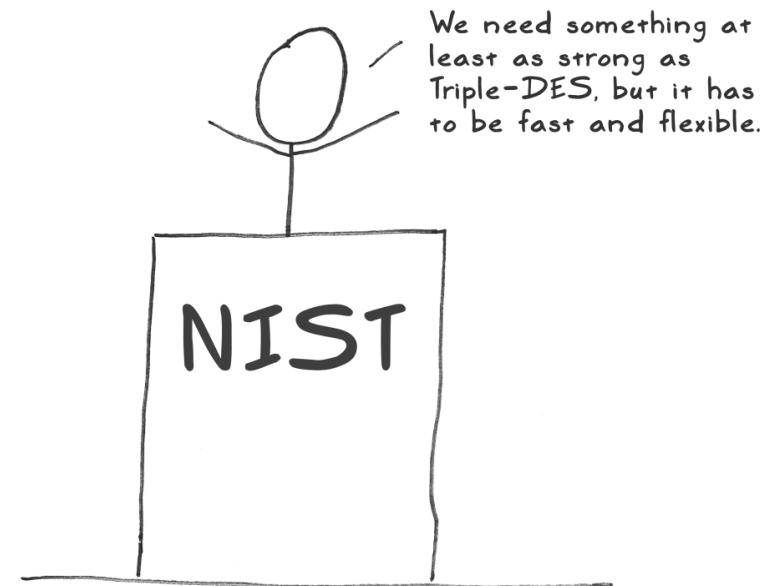
The only way to stop the attacks was to use DES 3 times in row to form 'Triple-DES.' This worked, but it was awfully slow.



This call rallied the crypto wizards to develop something better.

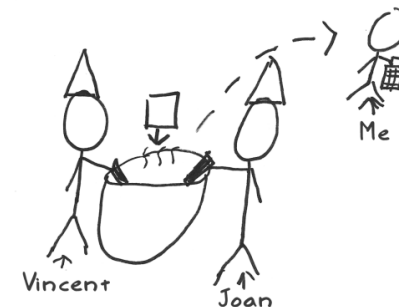


Another decree went out*...



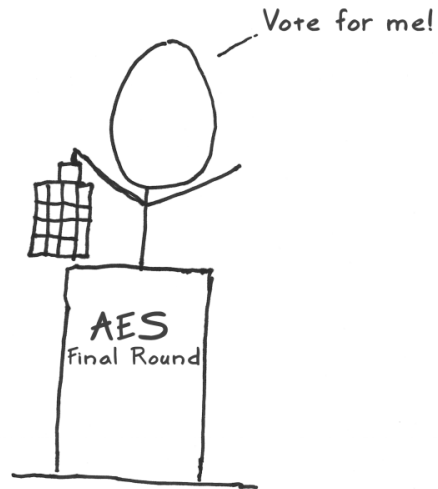
* ~ early 1997

My creators, Vincent Rijmen and Joan Daemen, were among these crypto wizards. They combined their last names to give me my birth name: Rijndael.*



* That's pronounced 'Rhine Dahl' for the non-Belgians out there.

Everyone got together to vote and...

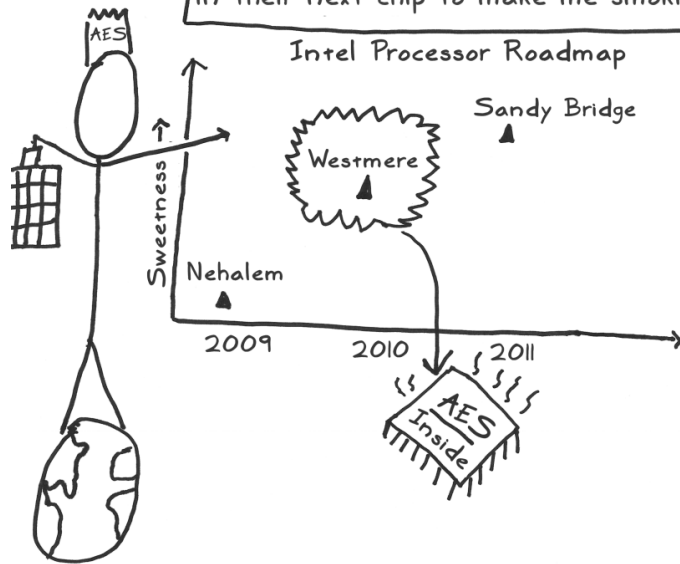


	Rijndael	Serpent	Twofish	MARS	RC6
General Security	2	3	3	3	2
Implementation Difficulty	3	3	2	1	1
Software Performance	3	1	1	2	2
Smart Card Performance	3	3	2	1	1
Hardware Performance	3	3	2	1	2
Design Features	2	1	3	2	1
Total	16	14	13	10	9

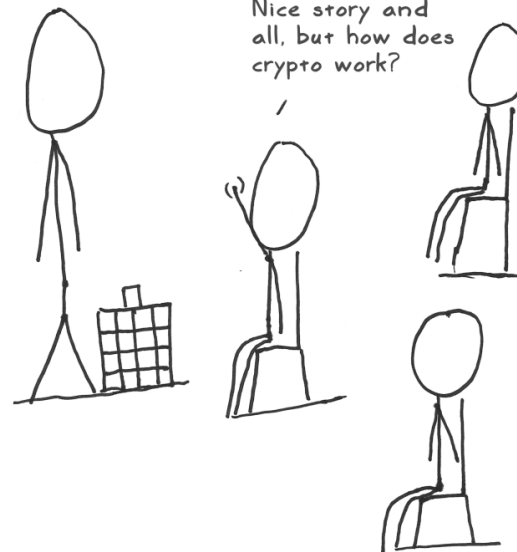
I won!!



...and now I'm the new king of the crypto world. You can find me everywhere. Intel is even putting native instructions for me in their next chip to make me smokin' fast!



Any questions?



Weird. I'm out...

Exit



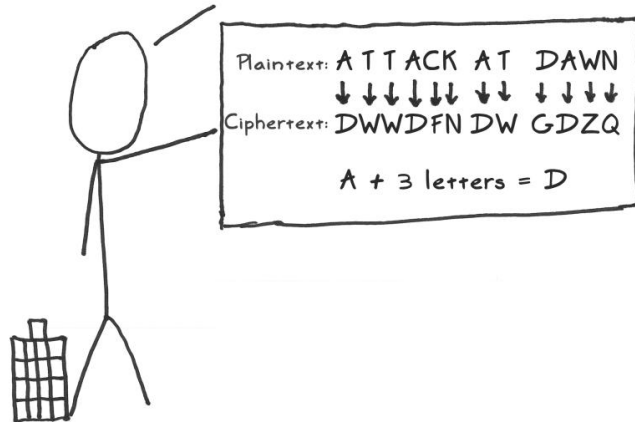
Act 2: Crypto Basics

Great question! You only need to know 3 big ideas to understand crypto.



Big Idea #1: Confusion

It's a good idea to obscure the relationship between your real message and your 'encrypted' message. An example of this 'confusion' is the trusty ol' Caesar Cipher:



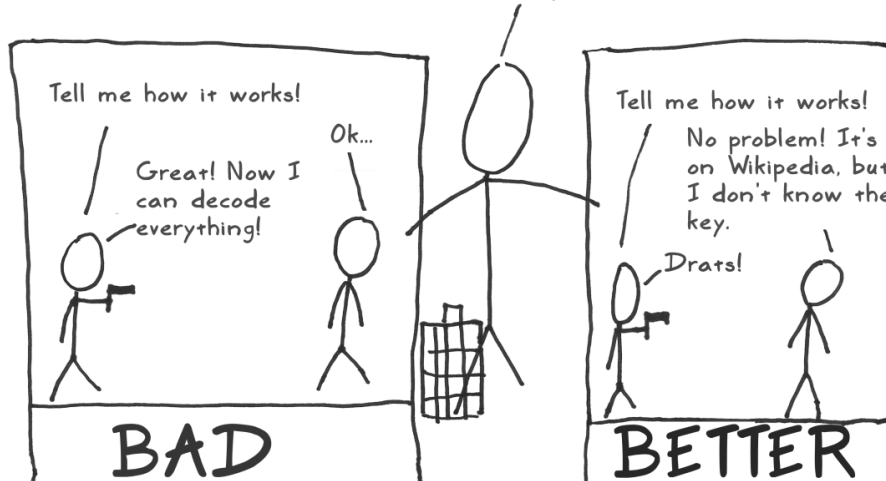
Big Idea #2: Diffusion

It's also a good idea to spread out the message. An example of this 'diffusion' is a simple column transposition:

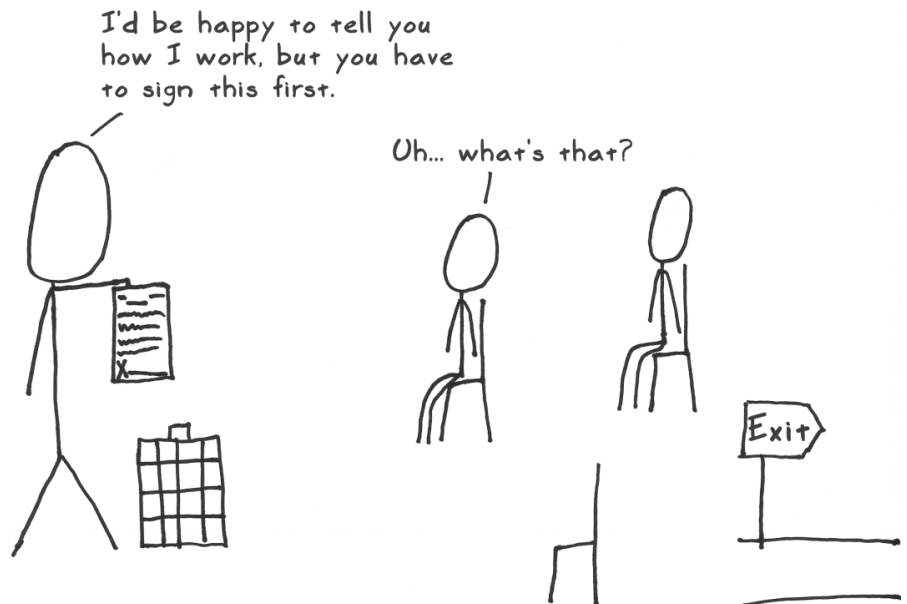
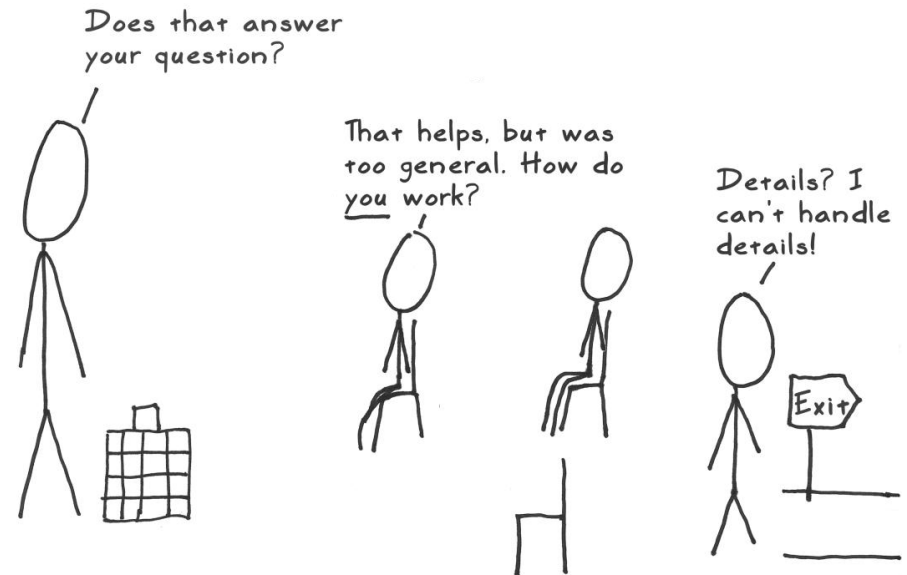


Big Idea #3: Secrecy Only in the Key

After thousands of years, we learned that it's a bad idea to assume that no one knows how your method works. Someone will eventually find that out.



Act 3: Details



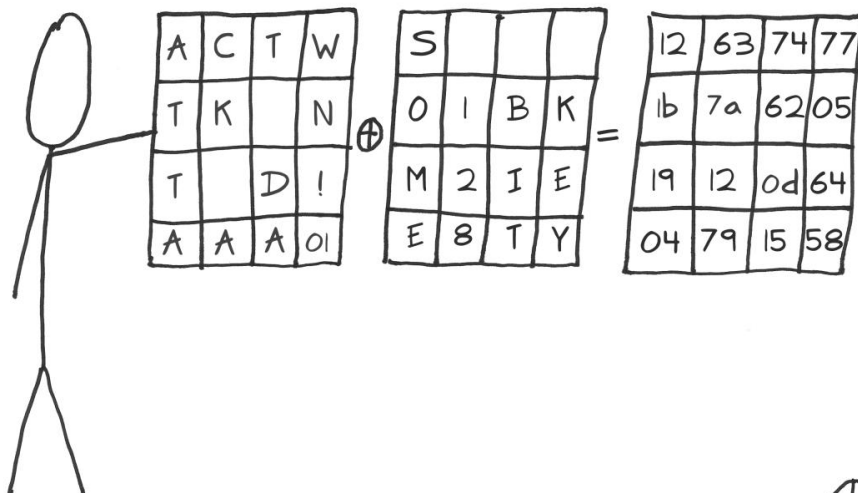
Foot-Shooting Prevention Agreement

I, _____, promise that once
Your Name
I see how simple AES really is, I will
not implement it in production code
even though it would be really fun.

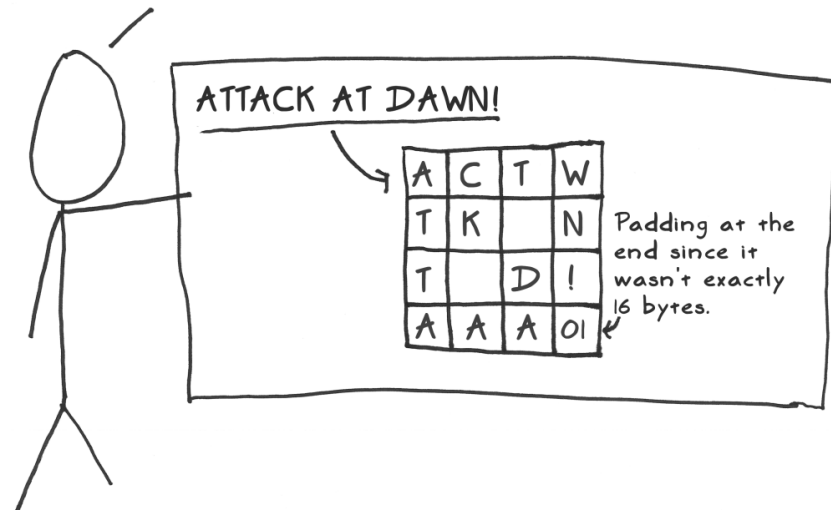
This agreement shall be in effect
until the undersigned creates a
meaningful interpretive dance that
compares and contrasts cache-based,
timing, and other side channel attacks
and their countermeasures.

X _____
Signature Date

The initial round has me xor each input
byte with the corresponding byte of the
first round key.



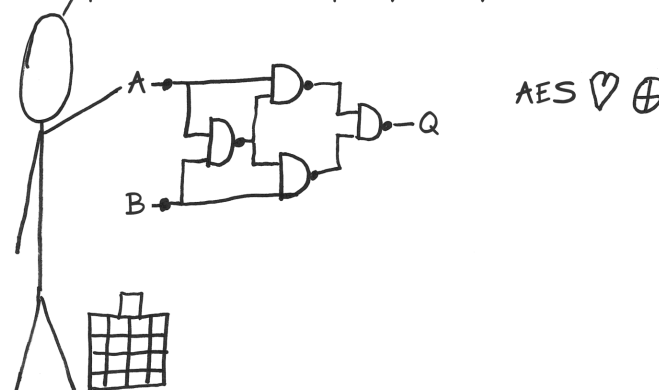
I take your data and load it
into this 4x4 square.*



* This is the 'state matrix' that I carry with me at all times.

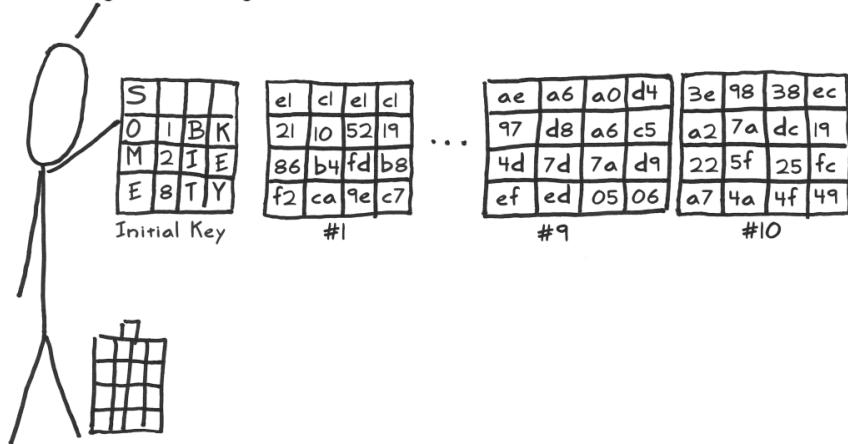
A Tribute to XOR

There's a simple reason why I use xor to apply the key
and in other spots: it's fast and cheap - a quick bit
flipper. It uses minimal hardware and can be done in
parallel since no pesky 'carry' bits are needed.



Key Expansion: Part 1

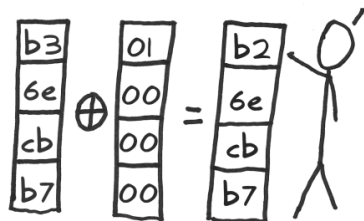
I need lots of keys for use in later rounds. I derive all of them from the initial key using a simple mixing technique that's really fast. Despite its critics,* it's good enough.



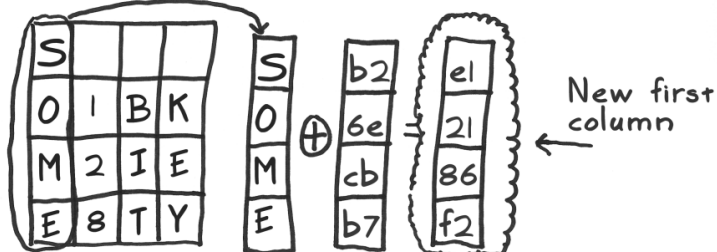
* By far, most complaints against AES's design focus on this simplicity.

Key Expansion: Part 2b

③ I then xor the column with a 'round constant' that is different for each round.

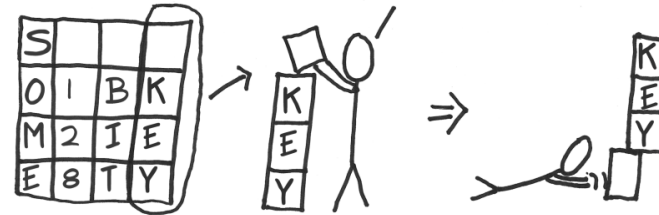


④ Finally, I xor it with the first column of the previous round key:

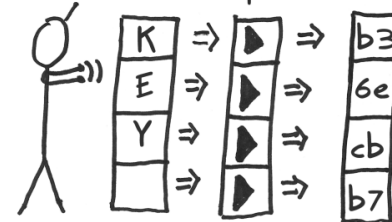


Key Expansion: Part 2a

① I take the last column of the previous round key and move the top byte to the bottom:

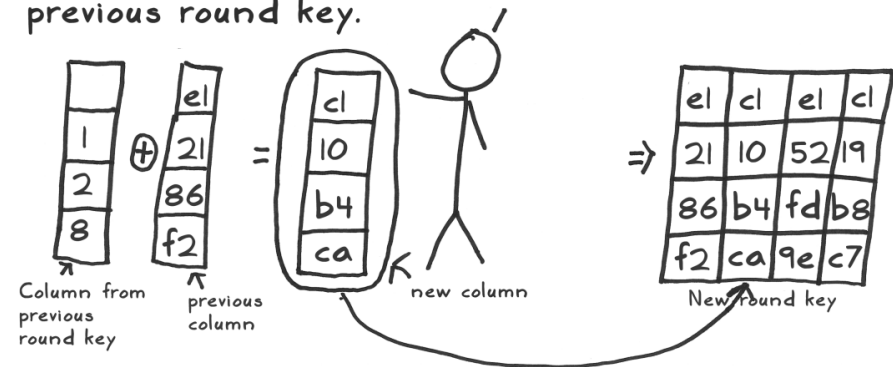


② Next, I run each byte through a substitution box that will map it to something else:



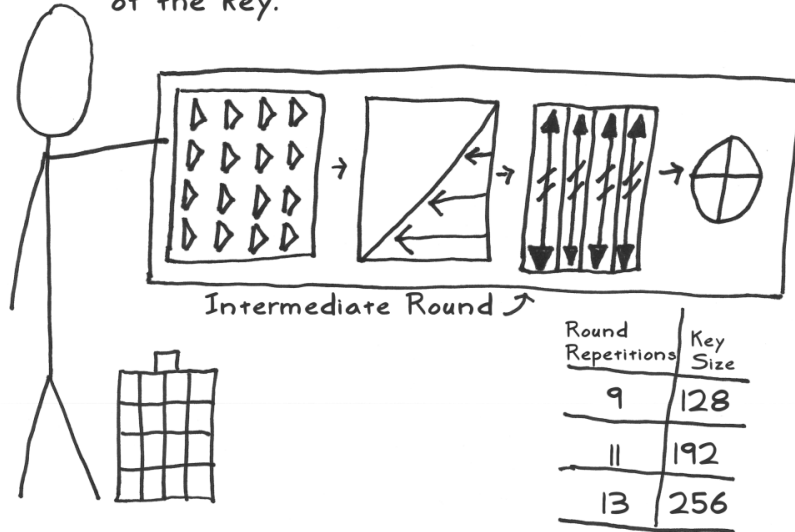
Key Expansion: Part 3

The other columns are super-easy,* I just xor the previous column with the same column of the previous round key.



* Note that 256 bit keys are slightly more complicated.

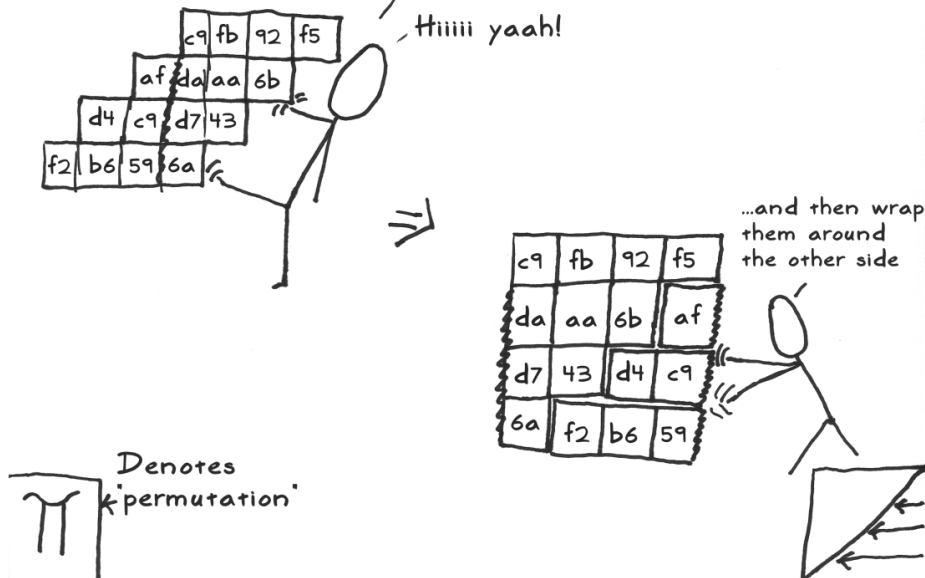
Next, I start the intermediate rounds. A round is just a series of steps I repeat several times. The number of repetitions depends on the size of the key.



Applying Diffusion, Part 1: Shift Rows

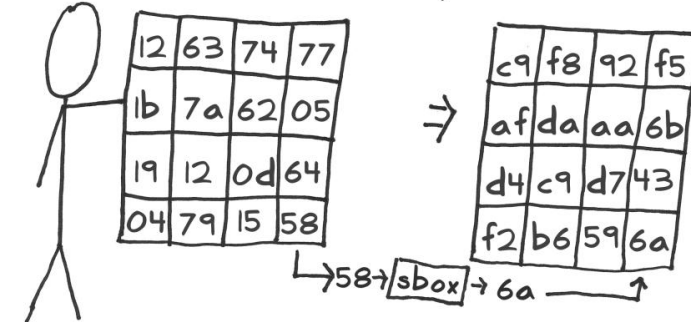
Next I shift the rows to the left

Hiiii yaah!



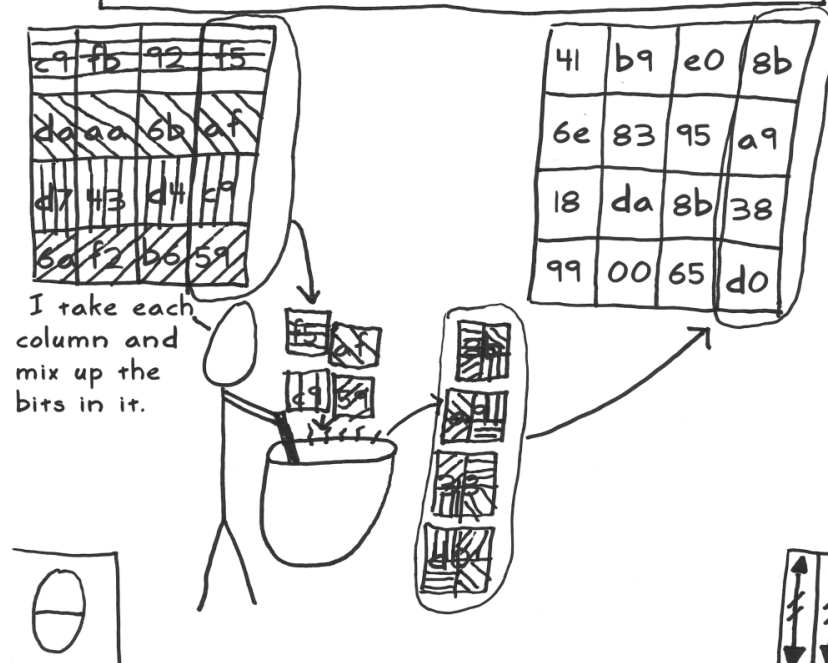
Applying Confusion: Substitute Bytes

I use confusion (Big Idea #1) to obscure the relationship of each byte. I put each byte into a substitution box (sbox), which will map it to a different byte:



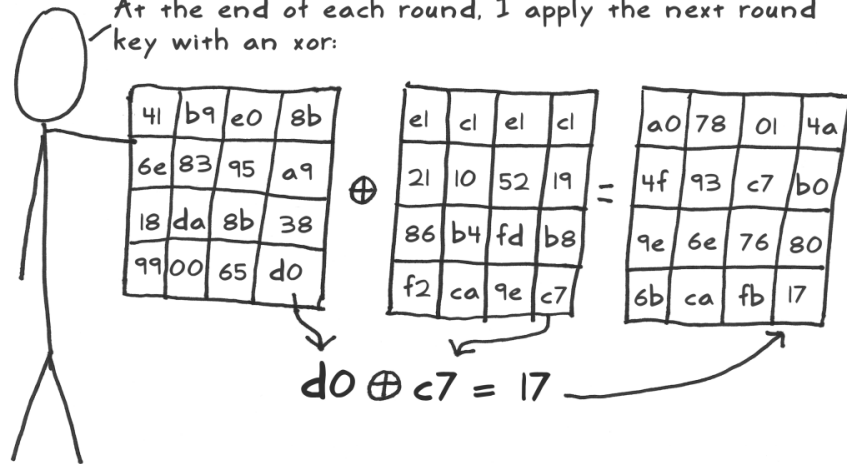
Denotes 'confusion'

Applying Diffusion, Part 2: Mix Columns

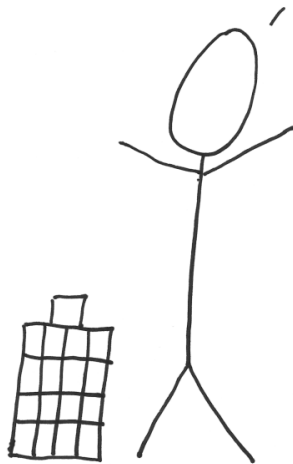


Applying Key Secrecy: Add Round Key

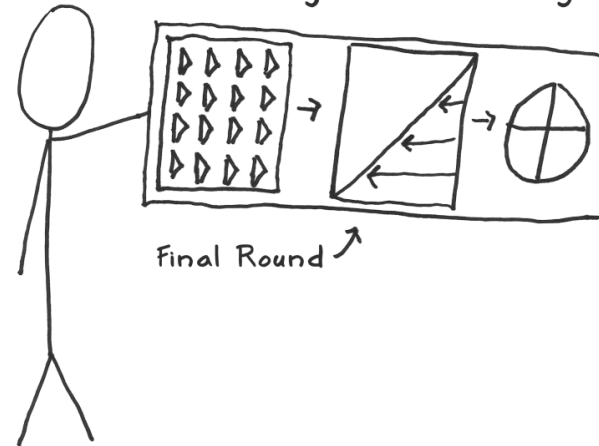
At the end of each round, I apply the next round key with an xor:



...and that's it. Each round I do makes the bits more confused and diffused. It also has the key impact them. The more rounds, the merrier!

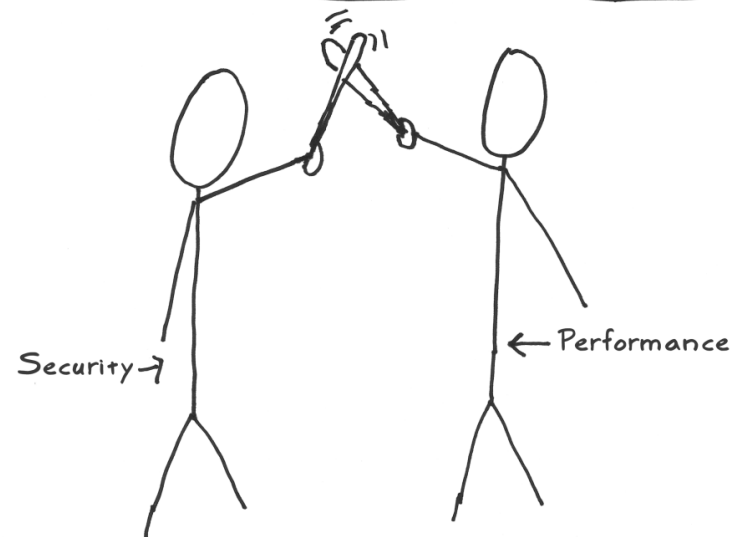


In the final round, I skip the 'Mix Columns' step since it wouldn't increase security* and would just slow things down:



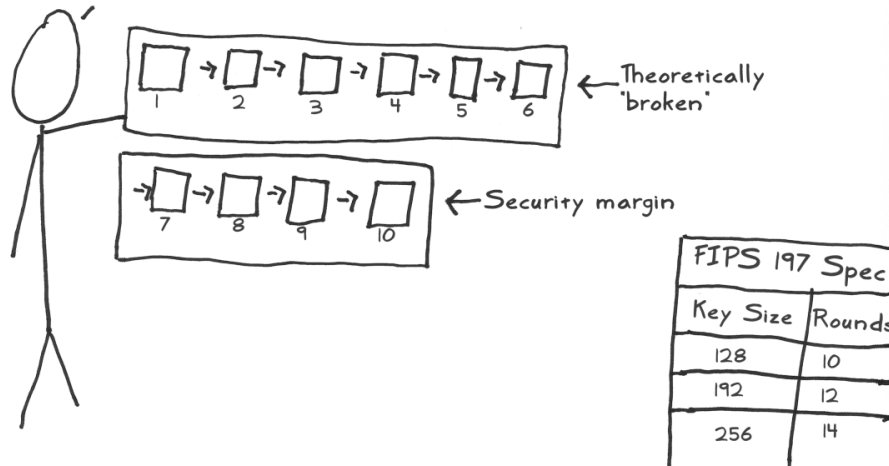
*The diffusion it would provide wouldn't go to the next round.

Determining the number of rounds always involves several tradeoffs.

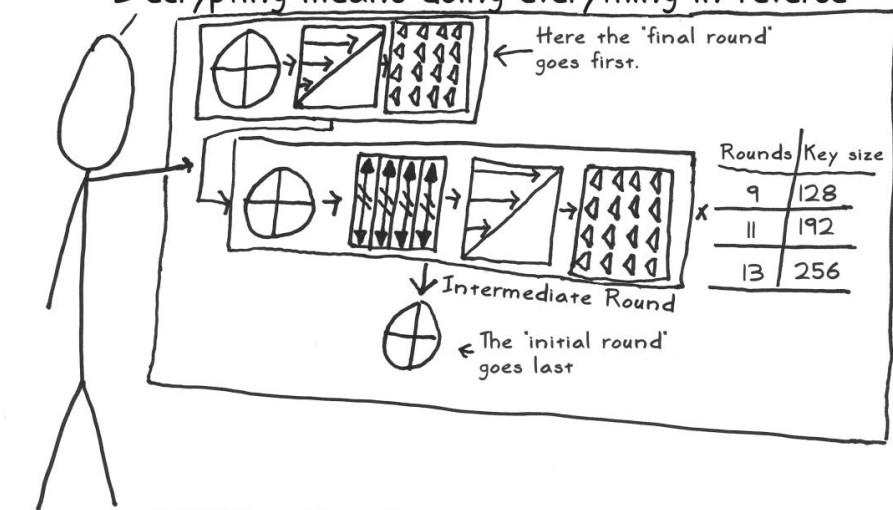


'Security always comes at a cost to performance' - Vincent Rijmen

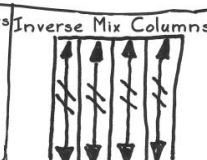
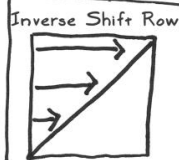
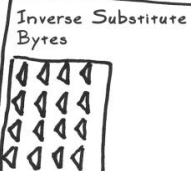
When I was being developed, a clever guy was able to find a shortcut path through 6 rounds. That's not good! If you look carefully, you'll see that each bit of a round's output depends on every bit from two rounds ago. To increase this diffusion 'avalanche,' I added 4 extra rounds. This is my 'security margin.'



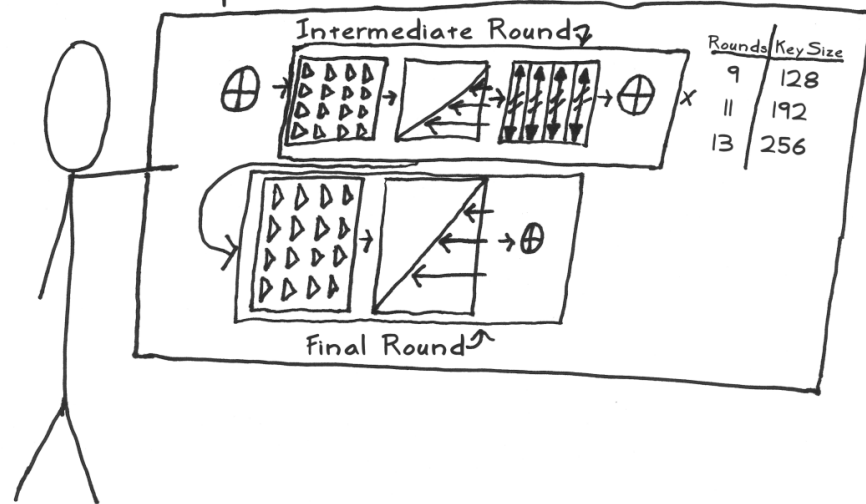
Decrypting means doing everything in reverse



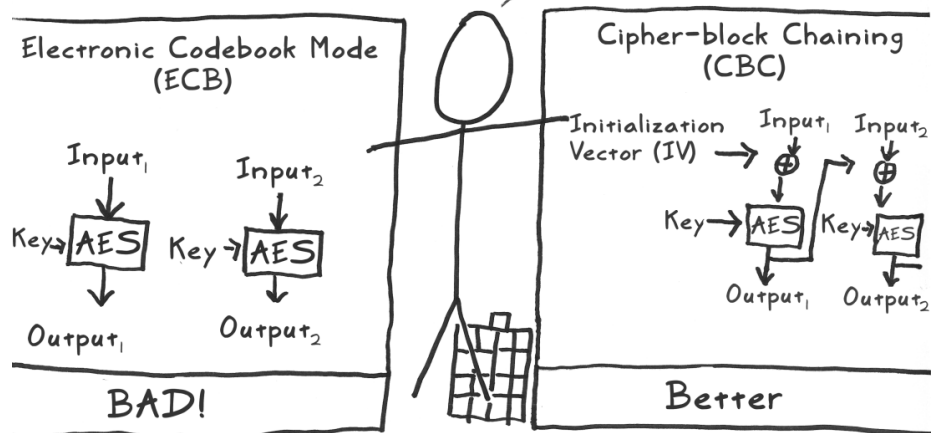
Add Round Key Inverse

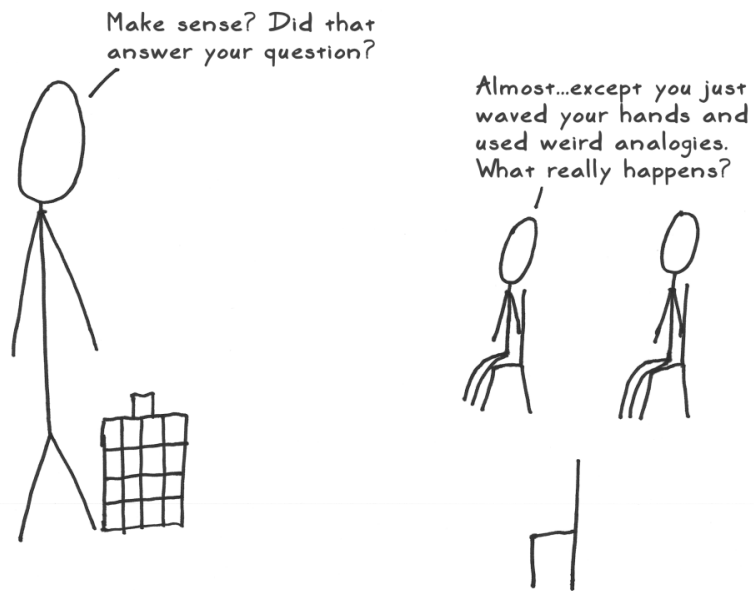


So in pictures, we have this:



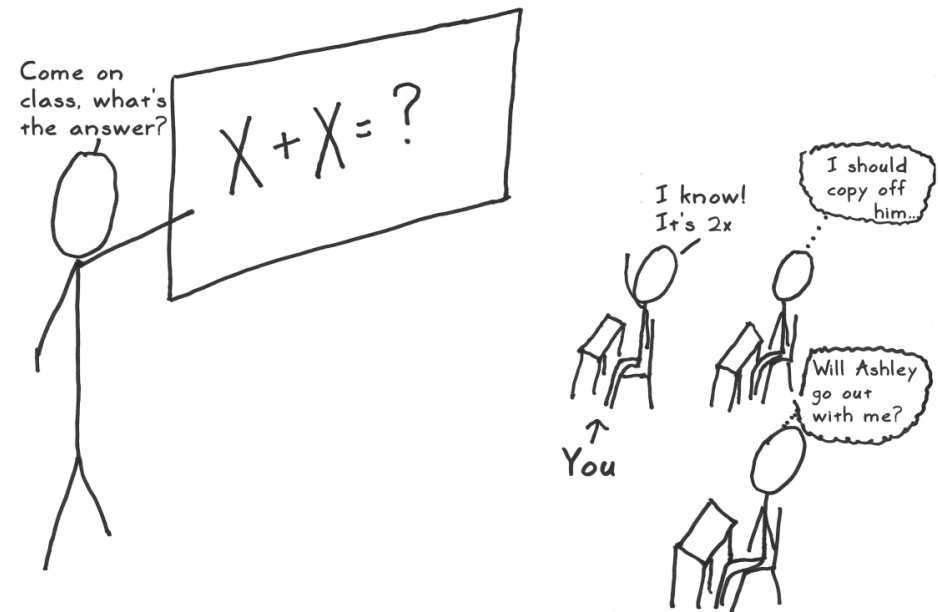
One last tidbit: I shouldn't be used as-is, but rather as a building block to a decent 'mode.'



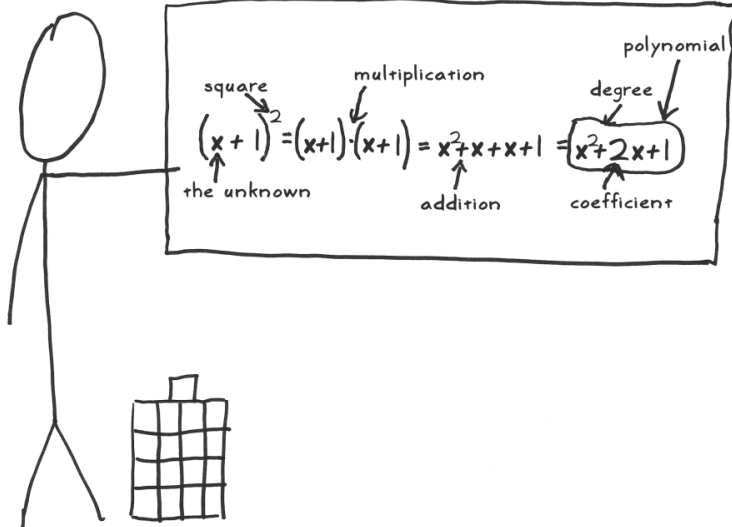


Let's go back to your algebra class...

Act 4: Math!



Reviewing the Basics...



We'll change things slightly. In the old way, coefficients could get as big as we wanted. In the new way, they can only be 0 or 1:

Old Way

$$123x^2 + 45x^2 + 678x + 9x + 10$$

$$= 168x^2 + 687x + 10$$

Big coefficients

New Way

$$x^2 \oplus x^2 \oplus x^2 \oplus x \oplus x \oplus 1$$

$$= x^2 \oplus 1$$

The 'new' add*

Small coefficients

$$x^2 \oplus x^2 \oplus x^2 = (x^2 \oplus x^2) \oplus x^2$$

$$= 0 \oplus x^2$$

$$= x^2$$

Remember how multiplication could make things grow fast?

$$(x^7 + x^5 + x^3 + x) \cdot (x^6 + x^4 + x^2 + 1)$$

$$= x^{7+6} + x^{7+4} + x^{7+2} + x^{7+0} + x^{5+6} + x^{5+4} + x^{5+2} + x^{5+0}$$

$$+ x^{3+6} + x^{3+4} + x^{3+2} + x^{3+0} + x^{1+6} + x^{1+4} + x^{1+2} + x^{1+0}$$

$$= x^{13} + x^{11} + x^9 + x^7 + x^{11} + x^9 + x^7 + x^5 + x^9 + x^7 + x^5 + x^3 + x^7 + x^5 + x^3 + x$$

$$= x^{13} + x^{11} + x^9 + x^9 + x^7 + x^7 + x^7 + x^7 + x^5 + x^5 + x^5 + x^3 + x^3 + x$$

$$= x^{13} + 2x^{11} + 3x^9 + 4x^7 + 3x^5 + 2x^3 + x$$

Big and yucky!

*Nifty Fact: In the new way, addition is the same as subtraction (e.g. $x \oplus x = x - x = 0$)

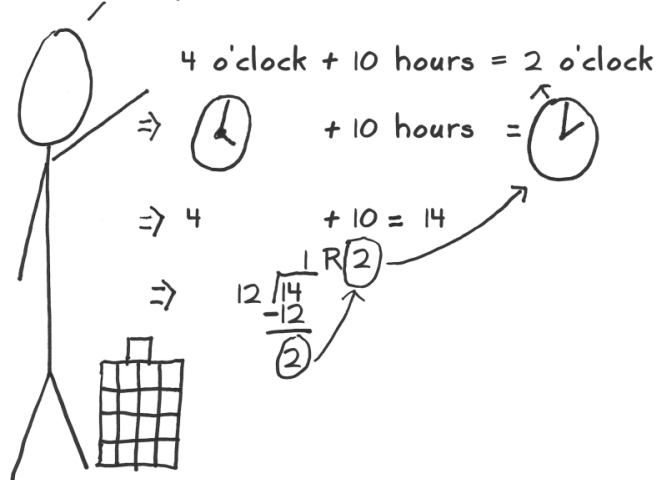
With the 'new' addition, things are simpler, but the x^{13} is still too big. Let's make it so we can't go bigger than x^7 . How can we do that?

$$x^{13} \oplus 2x^{11} \oplus 3x^9 \oplus 4x^7 \oplus 3x^5 \oplus 2x^3 \oplus x$$

$$\Rightarrow x^{13} \oplus 0x^{11} \oplus 0x^9 \oplus 0x^7 \oplus 0x^5 \oplus 0x^3 \oplus x$$

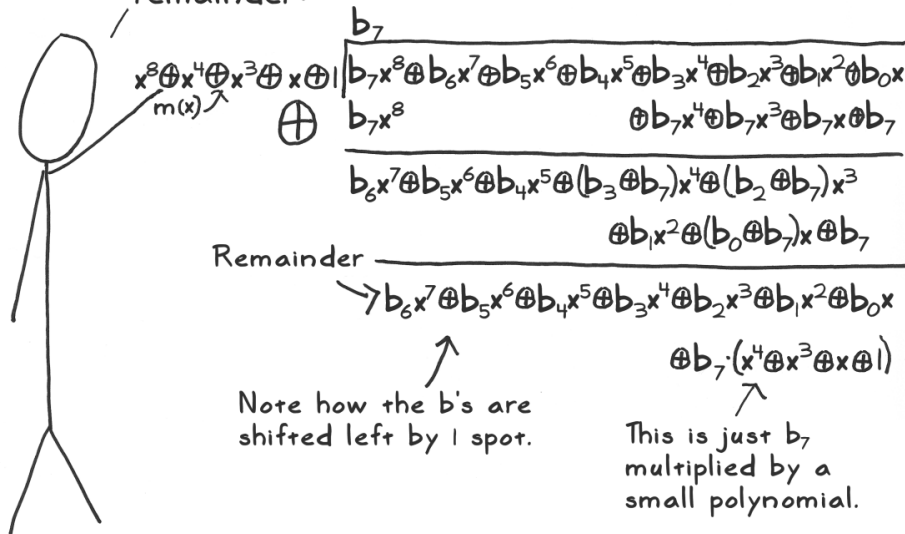
$$= x^{13} \oplus x^9 \oplus x^5 \oplus x$$

We use our friend, 'clock math,*' to do this.
Just add things up and do long division.
Keep a close watch on the remainder:

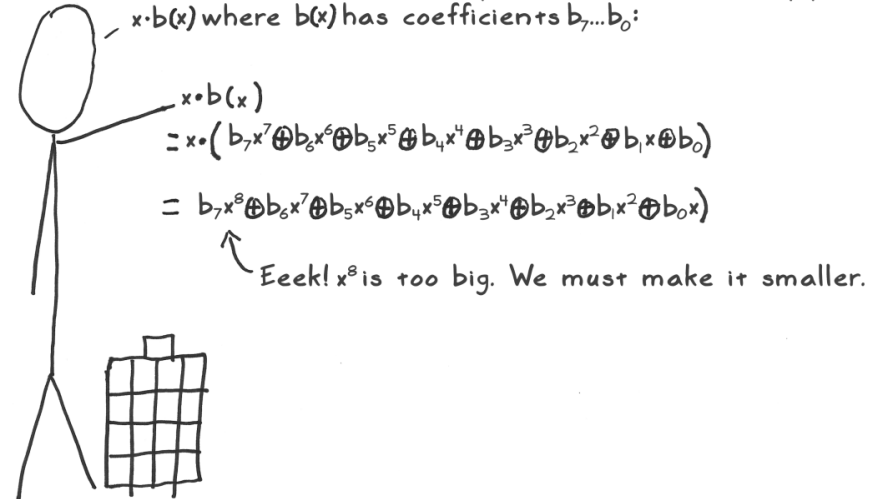


*This is also known as 'modular addition.' Math geeks call this a 'group.' AES uses a special group called a 'finite field.'

We divide it by $m(x) = x^8 \oplus x^4 \oplus x^3 \oplus x \oplus 1$ and take the remainder:

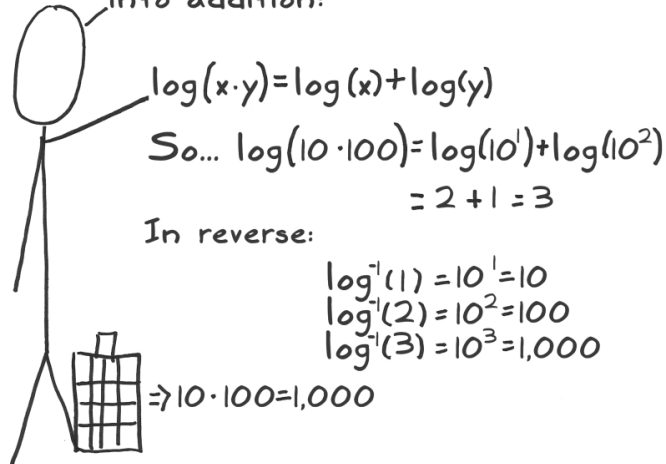


We can do 'clock' math with polynomials. Instead of dividing by 12, my creators told me to use $m(x) = x^8 \oplus x^4 \oplus x^3 \oplus x \oplus 1$. Let's say we wanted to multiply $x \cdot b(x)$ where $b(x)$ has coefficients $b_7 \dots b_0$:



*Remember that each b_n (e.g. b_7) is either 0 or 1.

Now we're ready for the hardest blast from the past: logarithms. After logarithms, everything else is cake! Logarithms let us turn multiplication into addition:



We can use logarithms in our new world. Instead of using 10 as the base, we can use the simple polynomial of $x \oplus 1$ and watch the magic unravel.*



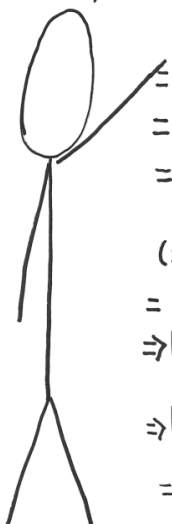
$$\begin{aligned}(x \oplus 1)^1 &= x \oplus 1 \\ (x \oplus 1)^2 &= (x \oplus 1)(x \oplus 1) = x^2 \oplus x \oplus x \oplus 1 = x^2 \oplus 1 \\ (x \oplus 1)^3 &= (x \oplus 1)^2 \cdot (x \oplus 1) = x^3 \oplus x^2 \oplus x \oplus 1\end{aligned}$$

So...

$$\log_{x \oplus 1}(x \oplus 1) = 1, \log_{x \oplus 1}(x^2 \oplus 1) = 2, \log_{x \oplus 1}(x^3 \oplus x^2 \oplus x \oplus 1) = 3$$

*If you keep multiplying by $(x \oplus 1)$ and then take the remainder after dividing by $m(x)$, you'll see that you generate all possible polynomial below x^8 . This is very important!

With bytes, polynomial addition becomes a simple xor. We can use our logarithm skills to make a table for speedy multiplication.*



$$\begin{aligned}(x^4 \oplus x^3 \oplus x \oplus 1) \oplus (x^7 \oplus x^5 \oplus x^3 \oplus x) \\ = \downarrow \quad \quad \quad \downarrow \\ = 1b \quad \oplus \quad aa \quad \leftarrow \text{byte xor} \\ = \downarrow \\ = b1 \\ = x^7 \oplus x^5 \oplus x^4 \oplus 1\end{aligned}$$

$$\begin{aligned}(x^4 \oplus x^3 \oplus x \oplus 1) \cdot (x^7 \oplus x^5 \oplus x^3 \oplus x) \\ = \downarrow \quad \quad \quad \downarrow \quad \text{logarithm table lookup} \\ = 1b \quad \cdot \quad aa \\ \Rightarrow \log(1b) + \log(aa) = c8 + 1f = e7 \\ \Rightarrow \log^{-1}(e7) = 8c \xrightarrow{\text{inverse table lookup}} 1b \cdot aa \\ = \downarrow \\ = x^7 \oplus x^3 \oplus x^2\end{aligned}$$

*We can create the table as we keep multiplying by $(x \oplus 1)$.

Why bother with all of this math? Encryption deals with bits and bytes, right? Well, there's one last connection: a 7th degree polynomial can be represented in exactly 1 byte since the new way uses only 0 or 1 for coefficients:



$$\begin{aligned}x^4 \oplus x^3 \oplus x \oplus 1 \\ = 0x^7 \oplus 0x^6 \oplus 0x^5 \oplus 1x^4 \oplus 1x^3 \oplus 0x^2 \oplus 1x \oplus 1 \\ = \underbrace{0 \quad 0 \quad 0 \quad 1}_{1} \quad \underbrace{1 \quad 0 \quad 1 \quad 1}_{1011_2 = 11_{10} = b_{16} \leftarrow \text{hexadecimal}} \\ = 1b \leftarrow \text{A single byte!!}\end{aligned}$$

*Although we'll work with bytes from now on, the math makes sure everything works out.

Since we know how to multiply, we can find the "inverse" polynomial byte for each byte. This is the byte that will undo/invert the polynomial back to 1. There are only 255* of them, so we can use brute force to find them:



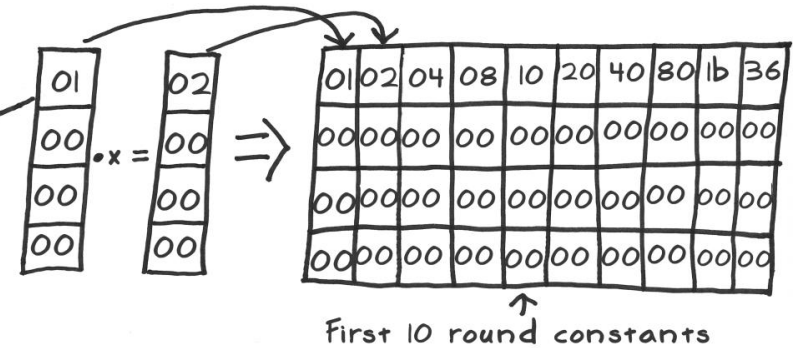
$$\begin{aligned}(x^4 \oplus x^3 \oplus x \oplus 1) \cdot ? = 1 \\ 1b \cdot \text{cc} = 1 \\ \quad \quad \quad \uparrow \\ \quad \quad \text{found using a brute force for-loop}\end{aligned}$$

*There are only 255 instead of 256 because 0 has no inverse.

Now we can understand the mysterious s-box. It takes a byte 'a' and applies two functions. The first is 'g' which just finds the byte inverse. The second is 'f' which intentionally makes the math uglier to foil attackers.

$g(a) = a^{-1}$
 $f(a) = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} a_7 \\ a_6 \\ a_5 \\ a_4 \\ a_3 \\ a_2 \\ a_1 \\ a_0 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}$
 $sbox[a] = f(g(a))$
 $sbox[58] = f(g(58))$
 $sbox[58] = f(18) = 6a$
 $58 \cdot 18 = 01$

We can also understand those crazy round constants in the key expansion. I get them by starting with '1' and then keep multiplying by 'x':



Mix Columns is the hardest. I treat each column as a polynomial. I then use our new multiply method to multiply it by a specially crafted polynomial and then take the remainder after dividing by $x^4 + 1$. This all simplifies to a matrix multiply:

$b(x) = c(x) \cdot a(x) \mod x^4 + 1$
 $= (03x^3 + 01x^2 + 01x + 02) \cdot (a_3x^3 + a_2x^2 + a_1x + a_0) \mod x^4 + 1$
 $= 03a_3x^6 + (3a_2 + a_3)x^5 + (3a_1 + a_2 + a_3)x^4 + \dots$
 $\oplus 03a_3x^6 + 03a_3x^2$
 $\oplus 3a_2x^5 + 3a_2x^4 + 3a_0x^3 + a_3x^5 + a_2x^4 + a_1x^3 + a_0x^2 + a_3x^4 + a_2x^3 + a_1x^2 + a_0x + 2a_3x^3 + 2a_2x^2 + 2a_1x + 2a_0$
 $\oplus 3a_2x^5 + a_3x^5 + 3a_2x^4 + a_3x^4$
 $\oplus (3a_1 + a_2 + a_3)x^4 + (3a_0 + a_2 + a_3)x^3$
 $\oplus (2a_3 + a_2 + a_1 + 3a_0)x^2 + (3a_3 + 2a_2 + a_1 + a_0)x + (a_3 + 3a_2 + 2a_1 + a_0)$

AES Grid Sheet (Handy for memorizing)

Plaintext in 4x4 grid

Initial Round

General Math

1.1B = AES Polynomial: $x^4 + 1$

Fast Multiply

$x^2 + x + x^3 + x + 1$

$x \cdot a(x) = (a_3 \ll 1) \oplus (a_2 \ll 1) \oplus 1B:00$

$\log(x \cdot y) = \log(x) + \log(y)$

Use $(x+1) = 03$ for log base

S-Box (SRD)

SRD[a] = f(g(a))

g(a) = $a^{-1} \mod m(x)$

Think $53 \oplus 63$

5 is and 3 is [0110 0011]

Key Expansion: Round Constants

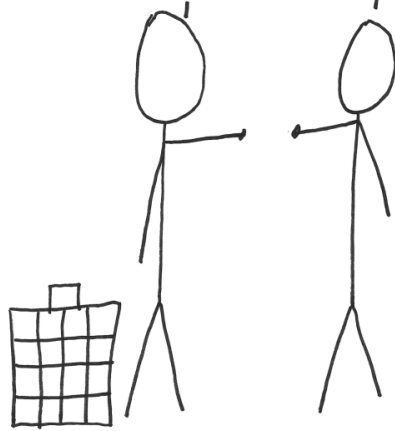
First Column: 01 02 04 08...

Other Columns:

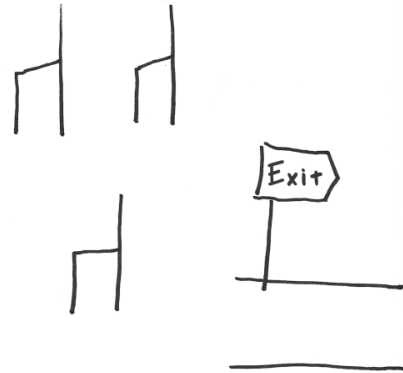
Mix Columns:

Inverse Mix

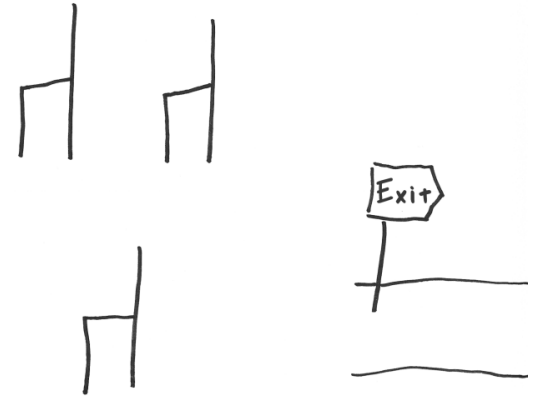
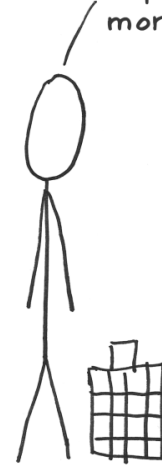
My pleasure.
Come back anytime!



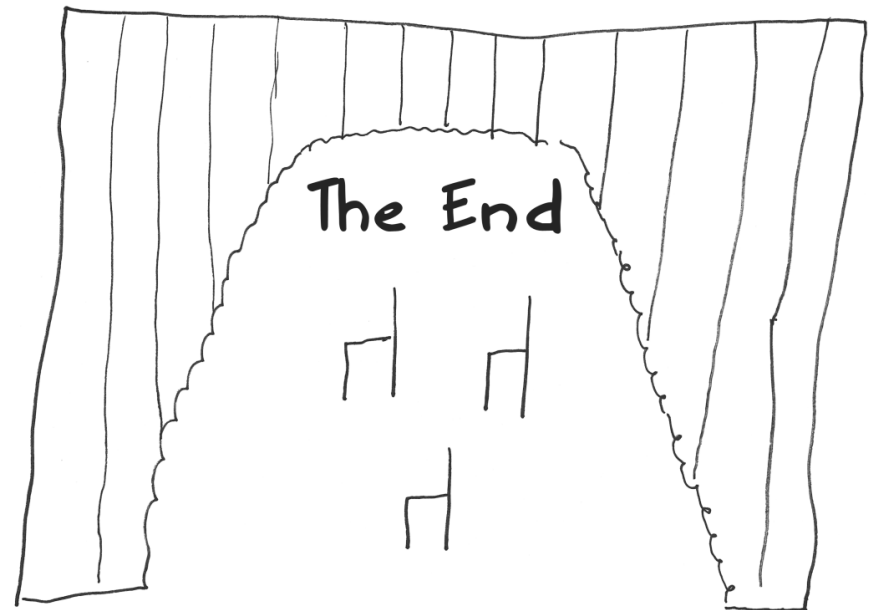
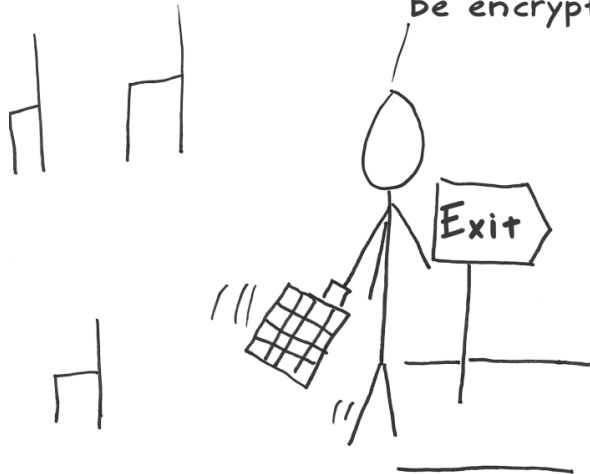
Whoa... I think I get it now. It's relatively simple once you grok the pieces. Thanks for explaining it. I gotta go now.



But there's so much more to talk about: my resistance to linear and differential cryptanalysis, my Wide Trail Strategy, impractical related-key attacks, and... so much more... but no one is left.



Oh well... there's some boring router traffic that needs to be encrypted. Gotta go!



Tabellenverzeichnis

2.1. Einfaches Beispiel: verfügbare Zeichen	10
2.2. Einfaches Beispiel: Anzahl der Elemente	11
2.3. Einfaches Beispiel: Klartext	11
2.4. Einfaches Beispiel: Übertragung	11
2.5. Einfaches Beispiel: Anzahl der Runden	12
2.6. Einfaches Beispiel: RC (round constant) Tabelle	13
2.7. Einfaches Beispiel: 1 Runde SubBytes lineare Funktion	13
2.8. Einfaches Beispiel: multiplikativ Inverse	14
2.9. Einfaches Beispiel: S-Box	14
2.10. Einfaches Beispiel: cipher key	16
2.11. Einfaches Beispiel: ShiftRow RC1	16
2.12. Einfaches Beispiel: SubBytes RC1	16
2.13. Einfaches Beispiel: Erweiterung RC1	16
2.14. Einfaches Beispiel: Rundenschlüssel für die zweite Runde (round key 2)	17
2.15. Einfaches Beispiel: Rundenschlüssel für die dritte Runde (round key 3)	18
2.16. Einfaches Beispiel: Rundenschlüssel für die vierte Runde (round key 4)	18
2.17. Einfaches Beispiel: Vorrunde AddRoundKey	20
2.18. Einfaches Beispiel: Vorrunde round key 1	20
2.19. Einfaches Beispiel: erste Runde SubBytes lineare Funktion	21
2.20. Einfaches Beispiel: multiplikativ Inverse	21
2.21. Einfaches Beispiel: erste Runde SubBytes multiplikativ Inverse	21
2.22. Einfaches Beispiel: erste Runde SubBytes lineare Funktion	21
2.23. Einfaches Beispiel: erste Runde ShiftRows	22
2.24. Einfaches Beispiel: erste Runde ShiftRows Permutationsmatrix	22
2.25. Einfaches Beispiel: erste Runde MixColumns	23
2.26. Einfaches Beispiel: erste Runde MixColumns Modulo 29	23
2.27. Einfaches Beispiel: MixColumns Matrix D	24
2.28. Einfaches Beispiel: erste Runde AddRoundKey	24
2.29. Einfaches Beispiel: erste Runde round key 2	24
2.30. Einfaches Beispiel: zweite Runde Ergebnis	24
2.31. Einfaches Beispiel: letzte Runde SubBytes multiplikativ Inverse	25
2.32. Einfaches Beispiel: letzte Runde SubBytes lineare Funktion	25
2.33. Einfaches Beispiel: letzte Runde ShiftRows	25
2.34. Einfaches Beispiel: letzte Runde AddRoundKey	25
2.35. Einfaches Beispiel: letzte Runde round key 4	25
2.36. Einfaches Beispiel: Verfügbare Zeichen Rückübersetzung	26
2.37. Einfaches Beispiel: Geheimtext	26
2.38. Einfaches Beispiel: Entschlüsselung Geheimtext	27
2.39. Einfaches Beispiel: Entschlüsselung verfügbare Zeichen	27
2.40. Einfaches Beispiel: Entschlüsselung Übertragung	28
2.41. Einfaches Beispiel: Entschlüsselung letzte Runde round key 4	28
2.42. Einfaches Beispiel: Entschlüsselung letzte Runde InvAddRoundKey	28

2.43. Einfaches Beispiel: Entschlüsselung letzte Runde InvShiftRows	29
2.44. Inverse lineare Funktion	29
2.45. Einfaches Beispiel: Entschlüsselung letzte Runde SubBytes multiplikativ Inverse	29
2.46. Einfaches Beispiel: Entschlüsselung letzte Runde SubBytes lineare Funktion	29
2.47. Einfaches Beispiel: Entschlüsselung inverse S-Box	30
2.48. Matrix D, Determinante und Inverse Matrix $D_{Basis10}$	30
2.49. Inverse Matrix D mal Determinate mit verschiedenen Werten und muliti- plikativ Inverses der Determinate	30
2.50. Einfaches Beispiel: Entschlüsselung inverse Matrix $D_{Restklassenkörper\ 29}$	31
3.1. Logisches Gatter: AND	32
3.2. Logisches Gatter: NAND	32
3.3. Logisches Gatter: OR	32
3.4. Logisches Gatter: NOR	32
3.5. Logisches Gatter: XOR	32
3.6. Logisches Gatter: XNOR	32
3.7. Logisches Gatter: NOT	33
3.8. Beispiel Selbstinvertiertheit von XOR	33
4.1. AES: Anzahl der Verschlüsselungsrunden N_r	35
4.2. Rijndael: Anzahl der Verschlüsselungsrunden N_r	35
4.3. Rijndael: ShiftRows in Abhängigkeit von N_b	37
4.4. Rijndael: Tabelle für die multiplikativ Inverse	39
4.5. Rijndael: lineare Funktion	39
4.6. AES: Tabelle für die lineare Funktion	40
4.7. AES: S-Box	40
4.8. AES: Anzahl der Verschlüsselungsrunden N_r	41
4.9. AES: cipher key bei N_4	42
4.10. AES: ShiftRow, SubBytes, RC 1 und Erweiterung	42
4.11. AES: Berechnung des 4. Word in N_4	42
4.12. AES: Berechnung des 5. Word in N_4	42
4.13. AES: Berechnung des 6. Word in N_4	43
4.14. AES: Berechnung des 7. Word in N_4	43
4.15. AES: round key 1 bei N_4	43
4.16. AES: ShiftRow, SubBytes, RC 2 und Erweiterung	43
4.17. AES: Berechnung des 8. Word in N_4	43
4.18. AES: Berechnung des 9. Word in N_4	43
4.19. AES: Berechnung des 10. Word in N_4	44
4.20. AES: Berechnung des 11. Word in N_4	44
4.21. AES: round key 2 bei N_4	44
4.22. AES: ShiftRow, SubBytes, RC 3 und Erweiterung	44
4.23. AES: Berechnung des 12. Word in N_4	44
4.24. AES: Berechnung des 13. Word in N_4	44
4.25. AES: Berechnung des 14. Word in N_4	45
4.26. AES: Berechnung des 15. Word in N_4	45
4.27. AES: round key 3 bei N_4	45
4.28. AES: cipher key bei N_6	46
4.29. AES: round key 1 bei N_6	46
4.30. AES: round key 2 bei N_6	46

4.31. AES: round key 3 bei N_6	47
4.32. AES: cipher key bei N_8	47
4.33. AES: SubBytes von Word 11 in N_8	47
4.34. AES: round key 1 bei N_8	48
4.35. AES: SubBytes von Word 19 in N_8	48
4.36. AES: round key 2 bei N_8	48
4.37. AES: SubBytes von Word 27 in N_8	48
4.38. AES: round key 3 bei N_8	49
4.39. AES: Vorrunde AddRoundKey state matrix in Bit	50
4.40. AES: Vorrunde AddRoundKey state matrix in Hexadezimal	50
4.41. AES: Vorrunde AddRoundKey cipher key in Bit	51
4.42. AES: Vorrunde AddRoundKey cipher key in Hexadezimal	51
4.43. AES: Vorrunde AddRoundKey Ergebnis in Bit	51
4.44. AES: Vorrunde AddRoundKey Ergebnis in Hexadezimal	51
4.45. AES: erste Runde SubBytes multiplikativ Inverse in Bit	51
4.46. AES: erste Runde SubBytes multiplikativ Inverse in Hexadezimal	51
4.47. AES: erste Runde SubBytes lineare Funktion in Bit	52
4.48. AES: erste Runde SubBytes lineare Funktion in Hexadezimal	52
4.49. AES: erste Runde Shift Rows in Bit	52
4.50. AES: erste Runde Shift Rows in Hexadezimal	52
4.51. AES: MixColumns Berechnungsvorschrift	53
4.52. AES: erste Runde MixColumns in Bit	54
4.53. AES: erste Runde MixColumns in Hexadezimal	54
4.54. AES: erste Runde AddRoundKey state matrix in Bit	54
4.55. AES: erste Runde AddRoundKey state matrix in Hexadezimal	54
4.56. AES: erste Runde AddRoundKey round key in Bit	55
4.57. AES: erste Runde AddRoundKey round key in Hexadezimal	55
4.58. AES: InvMixColumns Berechnungsvorschrift	56
4.59. Rijndael: invlineare Funktion	57
4.60. AES: Tabelle für die invlineare Funktion	58
4.61. AES: Tabelle für die invmultiplikativ Inverse	58
4.62. AES: InvS-Box	59
B.1. Einfaches Beispiel: round key 2	78
B.2. Einfaches Beispiel: ShiftRow RC2	78
B.3. Einfaches Beispiel: SubBytes RC2	79
B.4. Einfaches Beispiel: Erweiterung RC2	79
B.5. Einfaches Beispiel: round key 3	80
B.6. Einfaches Beispiel: ShiftRow RC3	80
B.7. Einfaches Beispiel: SubBytes RC3	80
B.8. Einfaches Beispiel: Erweiterung RC3	80
B.9. Einfaches Beispiel: round key 4	81
B.10. Einfaches Beispiel: 2. Runde SubBytes multiplikativ Inverse	82
B.11. Einfaches Beispiel: 2. Runde SubBytes lineare Funktion	82
B.12. Einfaches Beispiel: 2. Runde ShiftRows	82
B.13. Einfaches Beispiel: 2. Runde MixColumns	82
B.14. Einfaches Beispiel: 2. Runde MixColumns Modulo 29	82
B.15. Einfaches Beispiel: 2. Runde AddRoundKey	83
B.16. Einfaches Beispiel: 2. Runde round key 3	83

B.17.Einfaches Beispiel: Entschlüsselung 2. Runde InvAddRoundKey roundkey 3	84
B.18.Einfaches Beispiel: Entschlüsselung 2. Runde InvAddroundKey	84
B.19.Einfaches Beispiel: Entschlüsselung 2. Runde InvMixColumns	84
B.20.Einfaches Beispiel: Entschlüsselung 2. Runde InvMixColumns Modulo 29	84
B.21.Einfaches Beispiel: Entschlüsselung 2. Runde ShiftRows	85
B.22.Einfaches Beispiel: 2. Runde SubBytes multiplikativ Inverse	85
B.23.Einfaches Beispiel: 2. Runde SubBytes lineare Funktion	85
B.24.Einfaches Beispiel: Entschlüsselung 1. Runde InvAddRoundKey round key 2	85
B.25.Einfaches Beispiel: Entschlüsselung 1. Runde InvAddroundKey	85
B.26.Einfaches Beispiel: Entschlüsselung 1. Runde InvMixColumns	86
B.27.Einfaches Beispiel: Entschlüsselung 1. Runde InvMixColumns Modulo 29	86
B.28.Einfaches Beispiel: Entschlüsselung 1. Runde ShiftRows	86
B.29.Einfaches Beispiel: Entschlüsselung Vorrunde InvAddRoundKey round key 1	87
B.30.Einfaches Beispiel: Entschlüsselung Vorrunde InvAddroundKey	87
C.1. AES: cipher key bei N_6	88
C.2. AES: ShiftRow, SubBytes, RC 1 und Erweiterung	88
C.3. AES: Berechnung des 6. Word in N_6	88
C.4. AES: Berechnung des 7. Word in N_6	88
C.5. AES: Berechnung des 8. Word in N_6	88
C.6. AES: Berechnung des 9. Word in N_6	88
C.7. AES: Berechnung des 10. Word in N_6	89
C.8. AES: Berechnung des 11. Word in N_6	89
C.9. AES: round key 1	89
C.10.AES: ShiftRow, SubBytes, RC 2 und Erweiterung	89
C.11.AES: Berechnung des 12. Word in N_6	89
C.12.AES: Berechnung des 13. Word in N_6	89
C.13.AES: Berechnung des 14. Word in N_6	90
C.14.AES: Berechnung des 15. Word in N_6	90
C.15.AES: Berechnung des 16. Word in N_6	90
C.16.AES: Berechnung des 17. Word in N_6	90
C.17.AES: round key 2	90
C.18.AES: ShiftRow, SubBytes, RC 3 und Erweiterung	90
C.19.AES: Berechnung des 18. Word in N_6	91
C.20.AES: Berechnung des 19. Word in N_6	91
C.21.AES: Berechnung des 20. Word in N_6	91
C.22.AES: Berechnung des 21. Word in N_6	91
C.23.AES: Berechnung des 22. Word in N_6	91
C.24.AES: Berechnung des 23. Word in N_6	91
C.25.AES: round key 3 bei N_6	92
C.26.AES: cipher key	92
C.27.AES: ShiftRow, SubBytes, RC 1 und Erweiterung	92
C.28.AES: Berechnung des 8. Word in N_8	92
C.29.AES: Berechnung des 9. Word in N_8	92
C.30.AES: Berechnung des 10. Word in N_8	93
C.31.AES: Berechnung des 11. Word in N_8	93
C.32.AES: SubBytes von Word 11 bei N_8	93
C.33.AES: Berechnung des 12. Word in N_8	93
C.34.AES: Berechnung des 13. Word in N_8	93

C.35.AES: Berechnung des 14. Word in N_8	94
C.36.AES: Berechnung des 15. Word in N_8	94
C.37.AES: round key 1	94
C.38.AES: ShiftRow, SubBytes, RC 2 und Erweiterung	94
C.39.AES: Berechnung des 16. Word in N_8	94
C.40.AES: Berechnung des 17. Word in N_8	94
C.41.AES: Berechnung des 18. Word in N_8	95
C.42.AES: Berechnung des 19. Word in N_8	95
C.43.AES: SubBytes von Word 19 bei N_8	95
C.44.AES: Berechnung des 20. Word in N_8	95
C.45.AES: Berechnung des 21. Word in N_8	95
C.46.AES: Berechnung des 22. Word in N_8	96
C.47.AES: Berechnung des 23. Word in N_8	96
C.48.AES: round key 2	96
C.49.AES: ShiftRow, SubBytes, RC 3 und Erweiterung	96
C.50.AES: Berechnung des 16. Word in N_8	96
C.51.AES: Berechnung des 17. Word in N_8	96
C.52.AES: Berechnung des 18. Word in N_8	97
C.53.AES: Berechnung des 19. Word in N_8	97
C.54.AES: SubBytes von Word 27 bei N_8	97
C.55.AES: Berechnung des 20. Word in N_8	97
C.56.AES: Berechnung des 21. Word in N_8	97
C.57.AES: Berechnung des 22. Word in N_8	98
C.58.AES: Berechnung des 23. Word in N_8	98
C.59.AES: round key 3 bei N_8	98
C.60.Berechnung von Feld $A_{1,0}$	99
C.61.Berechnung von Feld $A_{2,0}$	99
C.62.Berechnung von Feld $A_{3,0}$	100
C.63.Berechnung von Feld $A_{1,1}$	100
C.64.Berechnung von Feld $A_{2,1}$	101
C.65.Berechnung von Feld $A_{3,1}$	101
C.66.Berechnung von Feld $A_{0,2}$	101
C.67.Berechnung von Feld $A_{1,2}$	102
C.68.Berechnung von Feld $A_{2,2}$	102
C.69.Berechnung von Feld $A_{3,2}$	103
C.70.Berechnung von Feld $A_{0,3}$	103
C.71.Berechnung von Feld $A_{1,3}$	103
C.72.Berechnung von Feld $A_{2,3}$	104
C.73.Berechnung von Feld $A_{3,3}$	104
D.1. Beispiel: Restklassenkörper 5	105
D.2. Beispiel: Restklassenkörper 7	105
D.3. Beispiel: Restklassenkörper 11	105
D.4. Beispiel: Restklassenkörper 13	105
D.5. Beispiel: Restklassen 4	106
D.6. Beispiel: Restklassen 6	106
D.7. Beispiel: Restklassen 8	106
D.8. Beispiel: Restklassen 9	106
D.9. Beispiel: RC im Dezimalsystem	107

D.10.Beispiel: RC im Binärsystem	107
D.11.Beispiel: RC im Hexadezimalsystem	108
D.12.Beispiel: Schlüsselerweiterung Modulo 29	108
D.13.Beispiel: Schlüsselerweiterung Modulo 13	109
D.14.Beispiel: Schlüsselerweiterung Modulo 7	110
D.15.Beispiel: Schlüsselerweiterung Modulo 11	111
D.16.Beispiel: Verschlüsselung Modulo 29 round key	113
D.17.Beispiel: Verschlüsselung Modulo 29 Ausgangsmatrix	113
D.18.Beispiel: Verschlüsselung Modulo 29 MixColumns-Matrix	113
D.19.Beispiel: Verschlüsselung Modulo 29 SubBytes multiplikativ Inverse	113
D.20.Beispiel: Verschlüsselung Modulo 29 SubBytes lineare Funktion	113
D.21.Beispiel: Verschlüsselung Modulo 29 ShiftRows Permutationsmatrix	113
D.22.Beispiel: Verschlüsselung Modulo 29 ShiftRows Ergebnis	113
D.23.Beispiel: Verschlüsselung Modulo 29 MixColumns Permutationsmatrix	114
D.24.Beispiel: Verschlüsselung Modulo 29 MixColumns Ergebnis	114
D.25.Beispiel: Verschlüsselung Modulo 29 AddRoundKey round key	114
D.26.Beispiel: Verschlüsselung Modulo 29 AddRoundKey Ergebnis	114
D.27.Beispiel: Verschlüsselung Modulo 13 round key	115
D.28.Beispiel: Verschlüsselung Modulo 13 Ausgangsmatrix	115
D.29.Beispiel: Verschlüsselung Modulo 13 SubBytes multiplikativ Inverse	115
D.30.Beispiel: Verschlüsselung Modulo 13 SubBytes lineare Funktion	115
D.31.Beispiel: Verschlüsselung Modulo 13 ShiftRows Permutationsmatrix	115
D.32.Beispiel: Verschlüsselung Modulo 13 ShiftRows Ergebnis	115
D.33.Beispiel: Verschlüsselung Modulo 13 AddRoundKey round key	115
D.34.Beispiel: Verschlüsselung Modulo 13 AddRoundKey Ergebnis	115
D.35.Beispiel: Verschlüsselung Modulo 7 round key	116
D.36.Beispiel: Verschlüsselung Modulo 7 Ausgangsmatrix	116
D.37.Beispiel: Verschlüsselung Modulo 7 MixColumns-Matrix	116
D.38.Beispiel: Verschlüsselung Modulo 7 SubBytes multiplikativ Inverse	116
D.39.Beispiel: Verschlüsselung Modulo 7 SubBytes lineare Funktion	116
D.40.Beispiel: Verschlüsselung Modulo 7 ShiftRows Permutationsmatrix	116
D.41.Beispiel: Verschlüsselung Modulo 7 ShiftRows Ergebnis	116
D.42.Beispiel: Verschlüsselung Modulo 7 MixColumns Permutationsmatrix	117
D.43.Beispiel: Verschlüsselung Modulo 7 MixColumns Ergebnis	117
D.44.Beispiel: Verschlüsselung Modulo 7 AddRoundKey round key	117
D.45.Beispiel: Verschlüsselung Modulo 7 AddRoundKey Ergebnis	117
D.46.Beispiel: Verschlüsselung Modulo 11 round key	118
D.47.Beispiel: Verschlüsselung Modulo 11 Ausgangsmatrix	118
D.48.Beispiel: Verschlüsselung Modulo 11 MixColumns-Matrix	118
D.49.Beispiel: Verschlüsselung Modulo 11 SubBytes multiplikativ Inverse	118
D.50.Beispiel: Verschlüsselung Modulo 11 SubBytes lineare Funktion	118
D.51.Beispiel: Verschlüsselung Modulo 11 ShiftRows Permutationsmatrix	118
D.52.Beispiel: Verschlüsselung Modulo 11 ShiftRows Ergebnis	118
D.53.Beispiel: Verschlüsselung Modulo 11 MixColumns Permutationsmatrix	118
D.54.Beispiel: Verschlüsselung Modulo 11 MixColumns Ergebnis	118
D.55.Beispiel: Verschlüsselung Modulo 11 AddRoundKey round key	119
D.56.Beispiel: Verschlüsselung Modulo 11 AddRoundKey Ergebnis	119

Abbildungsverzeichnis

2.1. AES: ShiftRows operation for Key Expansion	13
2.2. AES: SubBytes operation for Key Expansion	14
2.3. AES: AddRoundKey operation	20
2.4. AES: SubBytes operation	21
2.5. AES: ShiftRows operation	22
2.6. AES: MixColumns operation	23
3.1. Logisches Gatter: AND	32
3.2. Logisches Gatter: NAND	32
3.3. Logisches Gatter: OR	32
3.4. Logisches Gatter: NOR	32
3.5. Logisches Gatter: XOR	32
3.6. Logisches Gatter: NAND	32
3.7. Logisches Gatter: NOT	33

Code

6.1. Beginn	65
6.2. Variablen	65
6.3. Multiplikation	65
6.4. Logarithmustabelle	66
6.5. Potenztabelle Tafel	66
6.6. S-Box	67
6.7. InvS-Box	68
6.8. RC round key	68
6.9. Werte	68
6.10. ShiftRows in Abhängigkeit von N_b	69
6.11. Rundenanzahl N_r	69
6.12. freie Variablen	69
6.13. Multiplikation mit Logarithmus- und Potenztabelle	69
6.14. RC wird zu round key geben	69
6.15. SubBytes	70
6.16. ShiftRows	70
6.17. MixColumns	71
6.18. InvMixColumns	71
6.19. KeyExpansion	72
6.20. Encrypt Verschlüsselung	73
6.21. Decrypt Entschlüsselung	73
6.22. main Funktion	74

Index

R^*

siehe Einheit, 6

Äquivalenzklasse, 3

Eigenschaften von, 3

Partition, 4

Äquivalenzrelation

Kongruenz, 4

AES

N_b , 34

N_k , 34

N_r , 34

AddRoundKey, 24, 50, 54

Algorithmus, 19, 50

cipher key, 42

Entschlüsselung, 56

Galoisfeld, 35

InvAddRoundKey, 56

invlineare Funktion, 57

InvMixColumns, 56

InvmultiplikativInverse, 58

InvShiftRows, 57

InvSubBytes, 57

Key Expansion, 35

Key Schedule, 34

key selection, 45, 47, 49

lineare Funktion, 39

MixColumns, 22, 52

Parallel Arbeiten, 10

RC, 35

round constant, 35

round key, 42

Rundenkonstante, 35

Rundenschlüssel, 42

S-Box, 40

Schlüsselbestimmung, 45, 47, 49

Schlüsselerweiterung, 35

Schlüsselfestlegung, 34

ShiftRows, 52

state matrix, 10

SubBytes, 51

Vorrunde, 50

AG

siehe Assoziativgesetz, 5

algebraische Struktur

siehe Gruppoid, 5

Assoziativgesetz

AG, 5

Die Cramersche Regel, 9

Distributivgesetze, 6

einfaches Beispiel

N_b , 12

N_k , 12

N_r , 12

AddRoundKey, 19, 24

affine Abbildung, 13

Algorithmus, 19

cipher key, 12

InvAddRoundKey, 28

inverse lineare Funktion, 29

inverse Matrix D, 30

inverse S-Box, 30

InvMixColumns, 30

InvShiftRows, 28

InvSubBytes, 29

Key Expansion, 12, 16

Key Schedule, 12

key selection, 12

Klartext, 10

lineare Abbildung, 13

lineare Funktion, 20

Matrix D, 23

MixColumns, 22

multiplikativ Inverse, 13

n-1 Runden, 20

Permutationsmatrix, 22

RC, 12

round constant, 12

round key, 12

Rundenkonstante, 12

Rundenschlüssel, 12

Schlüsselbestimmung, 12

Schlüsselerweiterung, 12

Schlüsselfestlegung, 12

- ShiftRows, 13, 22
- state matrix, 10
- SubBytes, 13, 20
- Text übertragen in Zahlen, 10
- verfügbare Zeichen, 10
- verfügbarer Zeichensatz, 10
- Vorrunde, 19
- word, 16
- Einheit
 - R^* , 6
 - Einheitsgruppe, 6
- Einheitsgruppe
 - siehe Einheit, 6
- Einselement, 5
- Erweiterter Euklidischer Algorithmus, 14
 - Polynome, 37
- Euklidischer Algorithmus, 7
- Faktormenge
 - siehe Quotientenmenge, 4
- Familie
 - siehe Partition, 4
- Fundamentalsatz der Algebra, 9
- Galoisfeld
 - siehe Körper, 9
- Generator, 67
- Grundmenge, 5
- Gruppe, 6
 - ablesche, 6
- Gruppoid
 - algebraische Struktur, 5
 - Magma, 5
- Halbgruppen, 5
- Idempotenz, 5
- Integritätsbereich
 - Nullteiler, 7
- Inverses Element, 6
- Körper
 - Galoisfeld, 9
 - Körperaxiome, 7
 - Quotientenkörper, 8
 - Restklassenkörper, 9
 - Schiefkörper, 9
- Key Expansion, 12
- KG
 - siehe Kommutativgesetz, 5
- Kommutativgesetz
 - KG, 5
- Kongruenz
 - Äquivalenzrelation, 4
- Kongruenzklassen
 - siehe Restklassen, 4
- Magma
 - siehe Gruppoid, 5
- modulo
 - Erweiterung n auf $\mathbb{Q}$, 8
 - Kongruent n , 4
- Monoid, 5
- neutrales Element, 5
- Nullelement, 5
- Nullteiler
 - siehe Integritätsbereich, 7
- Partition
 - Äquivalenzklasse, 4
 - Familie, 4
- Primelement, 7
- Primzahl, 3
 - Primelement, 7
 - Primzahlkriterium, 7
- Primzahlkriterium, 7
- Quotientenkörper
 - siehe Körper, 8
- Quotientenmenge
 - Faktormenge, 4
- Reflexivität, 3
- Relationen, 3
- Restklassen
 - Addition von, 5
 - Kongruenzklassen, 4
 - Multiplikation von, 5
- Restklassenkörper, 9
- Restklassenringe, 6
- Rijndael
 - N_b , 34
 - N_r , 35
- Ring, 6
 - kommutativer, 6
 - mit Einselement, 6
- Schiefkörper
 - siehe Körper, 9
- Teiler, 7
- Transitivität, 3

Quellenverzeichnis

Literatur

- [1] Ilya Nikolaevich Bronshtein u. a., Hrsg. *Teubner-Taschenbuch der Mathematik*. Bd. 2. 1 Bde. Stuttgart: Teubner, 1995 (siehe S. 8, 9).
- [2] Joan Daemen und Vincent Rijmen. *The design of Rijndael: AES—the Advanced Encryption Standard*. Berlin; New York: Springer, 2002. 238 S. (siehe S. 23, 33, 36, 63, 65, 74).
- [3] Jeff Moser Rechte: CC-BY 3.0. *A Stick Figure Guide to the Advanced Encryption Standard (AES)*. Englisch. Blog. Sep. 2009. URL: <http://www.moserware.com/2009/09/stick-figure-guide-to-advanced.html> (besucht am 25.02.2018) (siehe S. 120).
- [4] Hermann Schichl und Roland Steinbauer. *Einführung in das mathematische Arbeiten*. 2., überarb. Aufl. Springer-Lehrbuch. OCLC: 794527262. Berlin: Springer Spektrum, 2012. 520 S. (siehe S. 3–7, 9).

Online-Quellen

- [5] *A Stick Figure Guide to the Advanced Encryption Standard (AES)*. URL: <http://www.moserware.com/2009/09/stick-figure-guide-to-advanced.html> (besucht am 25.02.2018) (siehe S. 60).
- [6] *ABS - Docs-Editors-Hilfe*. URL: <https://support.google.com/docs/answer/3093459?hl=de> (besucht am 27.12.2017) (siehe S. 31).
- [7] *ARRAYFORMULA - Docs-Editors-Hilfe*. URL: <https://support.google.com/docs/answer/3093275?hl=de> (besucht am 26.12.2017) (siehe S. 117).
- [8] *BININDEZ (BIN2DEC) - Docs-Editors-Hilfe*. URL: <https://support.google.com/docs/answer/3092991?hl=de> (besucht am 26.12.2017) (siehe S. 107, 110).
- [9] *AES算法源代码_文档库*. URL: <http://m.wendangku.net/doc/eb2eb53e79563c1ec5da71cc-11.html#> (besucht am 03.05.2018) (siehe S. 65).
- [10] *DEZINBIN (DEC2BIN) - Docs-Editors-Hilfe*. URL: <https://support.google.com/docs/answer/3092997?hl=de> (besucht am 26.12.2017) (siehe S. 107, 110).
- [11] *DEZINHEX (DEC2HEX) - Docs-Editors-Hilfe*. URL: <https://support.google.com/docs/answer/3093137?hl=de> (besucht am 26.12.2017) (siehe S. 107, 111).
- [12] *HEXINDEZ (HEX2DEC) - Docs-Editors-Hilfe*. URL: <https://support.google.com/docs/answer/3093192?hl=de> (besucht am 26.12.2017) (siehe S. 107, 111).
- [13] Kristian Laurent Haan. *Advanced Encryption Standard (AES)*. CODEPLANET. 17. Feb. 2008. URL: http://www.codeplanet.eu/tutorials/cpp/51-advanced-encryption-standard.html#aes_operations_mixcolumns (besucht am 03.05.2018) (siehe S. 65, 67).
- [14] *MDET (MDETERM) - Docs-Editors-Hilfe*. URL: <https://support.google.com/docs/answer/3094290?hl=de> (besucht am 22.12.2017) (siehe S. 30).

- [15] *MINV (MINVERSE)* - *Docs-Editors-Hilfe*. URL: <https://support.google.com/docs/answer/3094253?hl=de> (besucht am 22.12.2017) (siehe S. 30).
- [16] *MMULT* - *Docs-Editors-Hilfe*. URL: <https://support.google.com/docs/answer/3094292?hl=de> (besucht am 22.12.2017) (siehe S. 23).
- [17] Nayuki. *AES cipher internals in Excel*. 26. Mai 2016. URL: <https://www.nayuki.io/page/aes-cipher-internals-in-excel> (besucht am 25.02.2018) (siehe S. 63).
- [18] *REST (MOD)* - *Docs-Editors-Hilfe*. URL: <https://support.google.com/docs/answer/3093497?hl=de> (besucht am 26.11.2017) (siehe S. 17).
- [19] *RUNDEN (ROUND)* - *Docs-Editors-Hilfe*. URL: <https://support.google.com/docs/answer/3093440?hl=de> (besucht am 27.12.2017) (siehe S. 84).
- [20] Sam Trenholme. *AES' Galois field*. Sam Trenholme's webpage. URL: <http://www.samiam.org/galois.html> (besucht am 04.05.2018) (siehe S. 67).
- [21] *VORZEICHEN (SIGN)* - *Docs-Editors-Hilfe*. URL: <https://support.google.com/docs/answer/3093513?hl=de> (besucht am 27.12.2017) (siehe S. 31).
- [22] *WVERWEIS (HLOOKUP)* - *Docs-Editors-Hilfe*. URL: <https://support.google.com/docs/answer/3093375?hl=de> (besucht am 12.11.2017) (siehe S. 11).

Bildnachweis

- [23] User: Matt Crypto Rechte: Gemeinfrei. *AddRoundKey operation for AES*. 31. Aug. 2006. URL: <https://commons.wikimedia.org/wiki/File:AES-AddRoundKey.svg> (besucht am 16.03.2018) (siehe S. 20).
- [24] User: Matt Crypto Rechte: Gemeinfrei. *MixColumns operation for AES*. 31. Aug. 2006. URL: <https://commons.wikimedia.org/wiki/File:AES-MixColumns.svg> (besucht am 16.03.2018) (siehe S. 23).
- [25] User: Matt Crypto Rechte: Gemeinfrei. *ShiftRows operation for AES*. 31. Aug. 2006. URL: <https://commons.wikimedia.org/wiki/File:AES-ShiftRows.svg> (besucht am 03.03.2018) (siehe S. 13, 22).
- [26] User: Matt Crypto Rechte: Gemeinfrei. *SubBytes operation for AES*. 31. Aug. 2006. URL: <https://de.wikipedia.org/wiki/Datei:AES-SubBytes.svg> (besucht am 03.03.2018) (siehe S. 14, 21).
- [27] User: Tobulax User: Dbc334 Rechte: Gemeinfrei. *logic AND gate*. 1. Juli 2007. URL: https://de.wikipedia.org/wiki/Datei:IEC_AND_label.svg (besucht am 30.03.2018) (siehe S. 32).
- [28] User: Tobulax User: Dbc334 Rechte: Gemeinfrei. *logic NAND gate*. 1. Juli 2007. URL: https://de.wikipedia.org/wiki/Datei:IEC_NAND_label.svg (besucht am 30.03.2018) (siehe S. 32).
- [29] User: Tobulax User: Dbc334 Rechte: Gemeinfrei. *logic NOR gate*. 1. Juli 2007. URL: https://de.wikipedia.org/wiki/Datei:IEC_NOR_label.svg (besucht am 30.03.2018) (siehe S. 32).
- [30] User: Tobulax User: Dbc334 Rechte: Gemeinfrei. *logic NOT gate*. 1. Juli 2007. URL: https://commons.wikimedia.org/wiki/File:IEC_NOT_label.svg (besucht am 30.03.2018) (siehe S. 33).

- [31] User: Tobulax User: Dbc334 Rechte: Gemeinfrei. *logic OR gate*. 1. Juli 2007. URL: https://de.wikipedia.org/wiki/Datei:IEC_OR_label.svg (besucht am 30.03.2018) (siehe S. 32).
- [32] User: Tobulax User: Dbc334 Rechte: Gemeinfrei. *logic XNOR gate*. URL: https://de.wikipedia.org/wiki/Datei:IEC_XNOR_label.svg (besucht am 30.03.2018) (siehe S. 32).
- [33] User: Tobulax User: Dbc334 Rechte: Gemeinfrei. *logic XOR gate*. 1. Juli 2007. URL: https://de.wikipedia.org/wiki/Datei:IEC_XOR_label.svg (besucht am 30.03.2018) (siehe S. 32).

,