ALGEBRA I
WEIHNACHTSZETTEL

HENNING KRAUSE
JAN GEUENICH

Aufgabe 1 (Bytes als Körperelemente). Neben der als bitweises exklusives Oder (XOR) bekannten Addition wollen wir in dieser Aufgabe auch eine Multiplikation für Bytes einführen, sodass die Menge aller Bytes unter diesen beiden Verknüpfungen zum Körper wird.

Sei $f = X^8 + X^4 + X^3 + X^2 + 1 \in \mathbb{F}_2[X]$. Wir schreiben

$$x \;=\; x_1x_2x_3x_4x_5x_6x_7x_8$$

für die Restklasse von $x_1X^7 + x_2X^6 + \cdots + x_7X + x_8$ in $\mathbb{B} = \mathbb{F}_2[X]/(f)$ und nennen x ein *Byte*.

Insgesamt gibt es also $q = |\mathbb{B}| = 2^8 = 256$ Bytes.

(a) Zeige, dass das Polynom f irreduzibel in $\mathbb{F}_2[X]$ und deshalb $\mathbb{B}$ ein Körper ist.[1]

(b) Begründe, warum $\mathbb{B}^\times$ zyklisch vom Byte $\alpha = \texttt{00000010}$ erzeugt wird.

(c) Für $x \in \mathbb{B}^\times$ bezeichne mit $\log(x)$ die ganze Zahl mit $0 \le \log(x) < q-1$ und $\alpha^{\log(x)} = x$. Begründe, warum sich Produkte und Inverse von Bytes $x, y \in \mathbb{B}^\times$ einfach durch

$$xy \;=\; \alpha^{\log(x)+\log(y)} \quad \text{und} \quad x^{-1} \;=\; \alpha^{(q-1)-\log(x)}$$

berechnen lassen, wenn man die Werte von $\log(x)$ und $\log(y)$ als bekannt voraussetzt.

(d) Berechne Summe $x + y$, Produkt xy und Inverse x^{-1} und y^{-1} der Bytes $x = \texttt{10001111}$ und $y = \texttt{11001101}$. Hierbei darf die untenstehende Logarithmentafel verwendet werden.

(je 1 Zusatzpunkt)

x	$\log(x)$	x	$\log(x)$	x	$\log(x)$	x	$\log(x)$	x	$\log(x)$	x	$\log(x)$	x	$\log(x)$	x	$\log(x)$
00000000	-	00100000	5	01000000	6	01100000	30	10000000	7	10100000	55	11000000	31	11100000	203
00000001	0	00100001	138	01000001	191	01100001	66	10000001	112	10100001	63	11000001	45	11100001	89
00000010	1	00100010	101	01000010	139	01100010	182	10000010	192	10100010	209	11000010	67	11100010	95
00000011	25	00100011	47	01000011	98	01100011	163	10000011	247	10100011	91	11000011	216	11100011	176
00000100	2	00100100	225	01000100	102	01100100	195	10000100	140	10100100	149	11000100	183	11100100	156
00000101	50	00100101	36	01000101	221	01100101	72	10000101	128	10100101	188	11000101	123	11100101	169
00000110	26	00100110	15	01000110	48	01100110	126	10000110	99	10100110	207	11000110	164	11100110	160
00000111	198	00100111	33	01000111	253	01100111	110	10000111	13	10100111	205	11000111	118	11100111	81
00001000	3	00101000	53	01001000	226	01101000	107	10001000	103	10101000	144	11001000	196	11101000	11
00001001	223	00101001	147	01001001	152	01101001	58	10001001	74	10101001	135	11001001	23	11101001	245
00001010	51	00101010	142	01001010	37	01101010	40	10001010	222	10101010	151	11001010	73	11101010	22
00001011	238	00101011	218	01001011	179	01101011	84	10001011	237	10101011	178	11001011	236	11101011	235
00001100	27	00101100	240	01001100	16	01101100	250	10001100	49	10101100	220	11001100	127	11101100	122
00001101	104	00101101	18	01001101	145	01101101	133	10001101	197	10101101	252	11001101	12	11101101	117
00001110	199	00101110	130	01001110	34	01101110	186	10001110	254	10101110	190	11001110	111	11101110	44
00001111	75	00101111	69	01001111	136	01101111	61	10001111	24	10101111	97	11001111	246	11101111	215
00010000	4	00110000	29	01010000	54	01110000	202	10010000	227	10110000	242	11010000	108	11110000	79
00010001	100	00110001	181	01010001	208	01110001	94	10010001	165	10110001	86	11010001	161	11110001	174
00010010	224	00110010	194	01010010	148	01110010	155	10010010	153	10110010	211	11010010	59	11110010	213
00010011	14	00110011	125	01010011	206	01110011	159	10010011	119	10110011	171	11010011	82	11110011	233
00010100	52	00110100	106	01010100	143	01110100	10	10010100	38	10110100	20	11010100	41	11110100	230
00010101	141	00110101	39	01010101	150	01110101	21	10010101	184	10110101	42	11010101	157	11110101	231
00010110	239	00110110	249	01010110	219	01110110	121	10010110	180	10110110	93	11010110	85	11110110	173
00010111	129	00110111	185	01010111	189	01110111	43	10010111	124	10110111	158	11010111	170	11110111	232
00011000	28	00111000	201	01011000	241	01111000	78	10011000	17	10111000	132	11011000	251	11111000	116
00011001	193	00111001	154	01011001	210	01111001	212	10011001	68	10111001	60	11011001	96	11111001	214
00011010	105	00111010	9	01011010	19	01111010	229	10011010	146	10111010	57	11011010	134	11111010	244
00011011	248	00111011	120	01011011	92	01111011	172	10011011	217	10111011	83	11011011	177	11111011	234
00011100	200	00111100	77	01011100	131	01111100	115	10011100	35	10111100	71	11011100	187	11111100	168
00011101	8	00111101	228	01011101	56	01111101	243	10011101	32	10111101	109	11011101	204	11111101	80
00011110	76	00111110	114	01011110	70	01111110	167	10011110	137	10111110	65	11011110	62	11111110	88
00011111	113	00111111	166	01011111	64	01111111	87	10011111	46	10111111	162	11011111	90	11111111	175

Abgabe: Donnerstag, 11. Januar 2018, bis 14 Uhr in die Postfächer der TutorInnen in V3-126.

[1] Nutze das *Rabinsche Irreduzibilitätskriterium*: Für Primzahlen p ist ein Polynom $f \in \mathbb{F}_p[X]$ vom Grad n genau dann irreduzibel in $\mathbb{F}_p[X]$, wenn $f \mid X^{p^n} - X$ gilt aber f teilerfremd zu $X^{p^{n/d}} - X$ für jeden Primteiler d von n ist.

Aufgabe 2 (Fehlerkorrigierende Bytecodes als Ideale). In dieser Aufgabe beschäftigen wir uns damit, wie eine aus ℓ Bytes bestehende Nachricht geeignet um k Bytes ergänzt werden kann, sodass die Erkennung (bzw. Korrektur) von weniger als k (bzw. weniger als $\frac{k}{2}$) Bytefehlern möglich wird.

Sei $R = \mathbb{B}[Y]/(h)$ mit $h = Y^{q-1} - 1$ und seien $\ell, k \in \mathbb{N}_0$ mit $\ell > 0$ und $n = \ell + k \leq q - 1$.

Für die Restklasse von $m = m_1 Y^{n-1} + m_2 Y^{n-2} + \cdots + m_i Y^{n-i}$ in R schreiben wir

$$[m] \;=\; \big[m_1\, m_2\, \cdots\, m_i\big]_i$$

und nennen $[m]$ eine *Nachricht* der Länge $\leq i$ mit *Gewicht* $\sharp\{j \in \{1, 2, \ldots, i\} \mid m_j \neq 0\}$.

(a) Ein *zyklischer Bytecode* ist per Definition ein Ideal in R. Begründe, warum das Polynom

$$g \;=\; \big(Y - \alpha^0\big)\big(Y - \alpha^1\big) \cdots \big(Y - \alpha^{k-1}\big)$$

ein Teiler von h ist und schließe daraus, dass $(g)/(h)$ ein zyklischer Bytecode ist.

(b) Für Nachrichten $[m]$ ist die *kodierte Nachricht* $[\mathfrak{c}(m)]$ gegeben durch die Gleichung

$$\mathfrak{c}(m) \;=\; m + r \;=\; dg$$

mit $d, r \in \mathbb{B}[Y]$ und $\operatorname{grad}(r) < k$, welche sich mit Polynomdivision von m durch g ergibt.

Für Nachrichten $[m] = \big[m_1 \,\cdots\, m_\ell\big]_\ell$ ist die kodierte Nachricht somit von der Form

$$[\mathfrak{c}(m)] \;=\; \big[m_1 \,\cdots\, m_\ell\, r_{\ell+1} \,\cdots\, r_n\big]_n\,.$$

Die Bytes $r_{\ell+1}, \ldots, r_n$ werden in diesem Fall als *Fehlerkorrekturbytes* bezeichnet.

Kodiere die Nachricht $\big[\texttt{00111010 00101001}\big]_2$ mit den Parametern $\ell = 2$ und $k = 1$.

(c) Sei nun $[\tilde{c}] = c + \varepsilon$ eine beliebige Nachricht, die sich aus einer kodierten Nachricht c und einer Nachricht ε zusammensetzt, die wir als Fehler interpretieren wollen. Begründe:

(i) Genau dann ist $[\tilde{c}]$ eine kodierte Nachricht, wenn $\tilde{c}\big(\alpha^i\big) = 0 \;\forall\, i = 0, 1, \ldots, k-1$ gilt.

(ii) Hat ε Gewicht $< k$, so ist entweder $\varepsilon = 0$ oder $[\tilde{c}]$ ist keine codierte Nachricht.

(iii) Wenn $[\tilde{c}] = c' + \varepsilon'$ für eine weitere kodierte Nachricht c' und eine Nachricht ε' gilt und sowohl ε als auch ε' Gewicht $< \frac{k}{2}$ besitzen, dann ist notwendigerweise $c = c'$.

Erkläre, wie sich diese Eigenschaften zur Fehlererkennung und -korrektur nutzen lassen.

(d) Als Beispiel betrachten wir nun die Erzeugung eines QR-Codes. Als erstes wandeln wir die Information, die im QR-Code gespeichert werden soll – sagen wir die ASCII-Zeichenfolge "Frohes Fest" der Länge $\lambda = 11$ in eine Bytefolge um, die als Nachricht verwendbar ist:

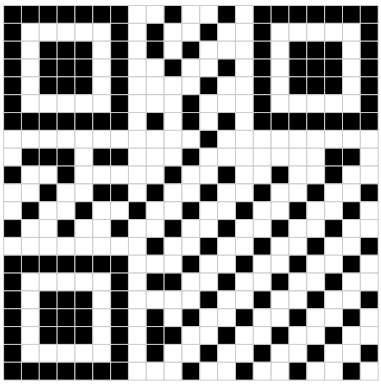
$$[m] = \Big[\begin{array}{cccccccc} & \lambda & & \mathrm{F} & \mathrm{r} & \mathrm{o} & \mathrm{h} & \mathrm{e} & \mathrm{s} \\ 0100\,0000 & 1011\,0100 & 0110\,0111 & 0010\,0110 & 1111\,0110 & 1000\,0110 & 0101\,0111 & 0011\,0010 \\ & & & \mathrm{F} & \mathrm{e} & \mathrm{s} & \mathrm{t} & \\ 0000\,0100 & 0110\,0110 & 0101\,0111 & 0011\,0111 & 0100\,0000 & & & \end{array}\Big]_{13}$$

Um unsere Nachricht der Länge ≤ 13 in einem QR-Code der Größe 21×21 unterzubringen, können wir das *Fehlerkorrekturlevel "Q"* wählen, was der Wahl von $k = \ell = 13$ entspricht. Eine Polynomdivision liefert dann folgende kodierte Nachricht:

$$[\mathfrak{c}(m)] = \Big[\begin{array}{cccccccc} 01000000 & 10110100 & 01100111 & 00100110 & 11110110 & 10000110 & 01010111 & 00110010 \\ & & & 00000100 & 01100110 & 01010111 & 00110111 & 01000000 \\ 11111000 & 01111110 & 01011110 & 10100010 & 11011000 & 01100001 & 00010100 & 01111101 \\ & & & 10110011 & 10111101 & 00000101 & 00000000 & 01000011 \end{array}\Big]_{26}$$

Um einzusehen, wie wir diese jetzt kodierte Nachricht in einen QR-Code übertragen können, bedarf es noch einiger Erläuterungen.

Identifizieren wir 0 mit "weiß" und 1 mit "schwarz", können wir Matrizen in $V = \mathcal{M}_{21}(\mathbb{F}_2)$ durch quadratische Schwarz-Weiß-Muster darstellen, die aus 21×21 Kästchen bestehen. Zur Erstellung unseres QR-Codes werden wir die Matrix

$M =$ 

benötigen, die einerseits der Maskierung dient und andererseits Formatierungsinformationen beinhaltet (u.a. das verwendete Fehlerkorrekturlevel).

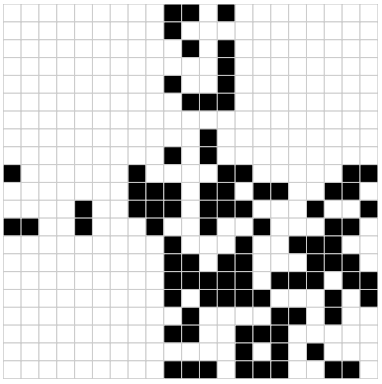
Des Weiteren ordnen wir jeder Nachricht $[c] = \left[c_1\, c_2\, \cdots\, c_{26}\right]_{26}$ linear die Matrix $N(c)$ zu, deren Eintrag an der im folgenden Schema markierten Position i, j gerade das Bit $(c_i)_j$ ist:

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

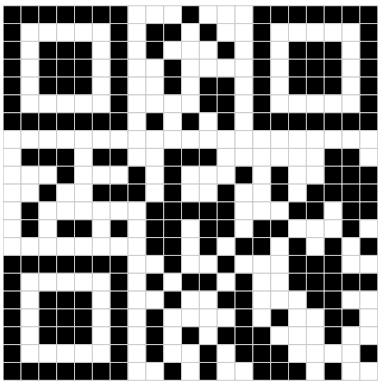
Nach alledem können wir nun den ***QR-Code*** von $[c]$ als $\mathrm{QR}(c) = N(c) + M$ definieren.

Hiermit erhalten wir schließlich für unsere kodierte Nachricht folgende Matrizen:

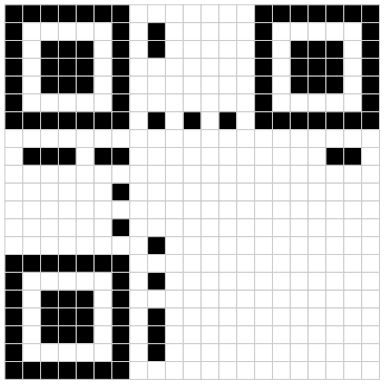
$$N(\mathfrak{c}(m)) =$$

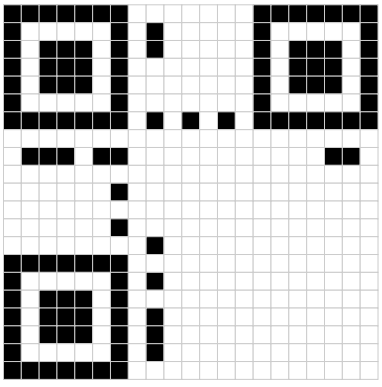


$$\mathrm{QR}(\mathfrak{c}(m)) =$$



Gib abschließend zwei Beispiele für Matrizen an, die sich in mindestens $24 - 12$ Bits von $\mathrm{QR}(\mathfrak{c}(m))$ unterscheiden, aber von einem QR-Code-Scanner erfolgreich dekodiert werden:





(je 1 Zusatzpunkt)

FROHE WEIHNACHTEN UND EIN GUTES NEUES JAHR!