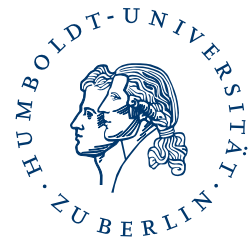
Einführung in die Kryptologie

Johannes Köbler



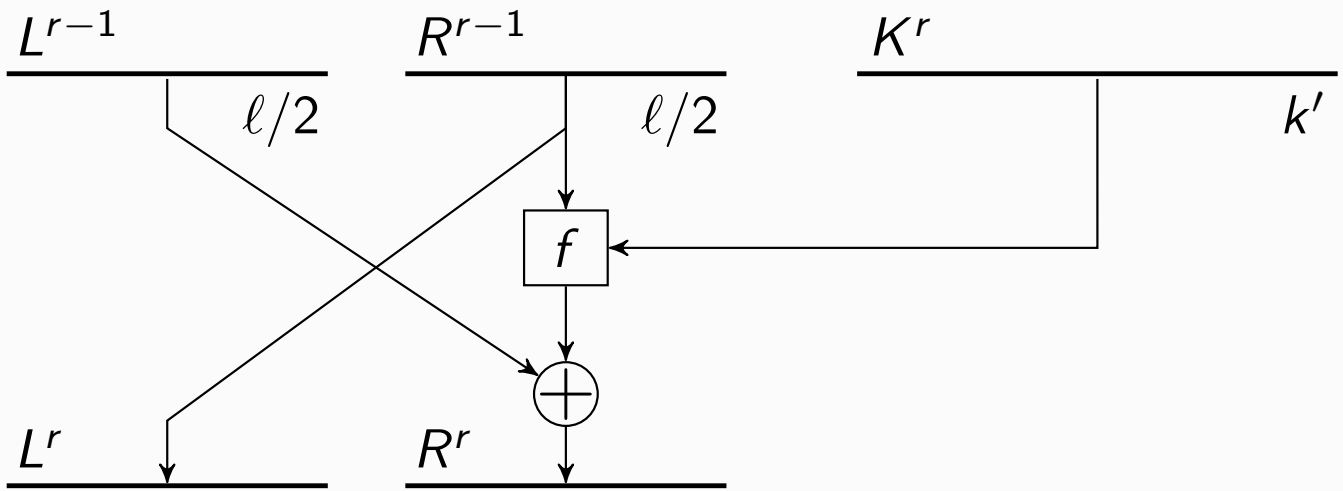
Institut für Informatik
Humboldt-Universität zu Berlin

SS 2020

Der Data Encryption Standard (DES)

- Der DES wurde von IBM im Zuge einer Ausschreibung des NBS (National Bureau of Standards; heute National Institute of Standards and Technology, NIST) als ein Nachfolger von Lucifer entwickelt
- Er wurde 1975 veröffentlicht, und 1977 als Verschlüsselungsstandard der US-Regierung für nicht geheime Nachrichten genormt
- Obwohl DES ursprünglich nur für einen Zeitraum von 10 bis 15 Jahren als Standard dienen sollte, wurde er circa alle 5 Jahre (zuletzt im Januar 1999) überprüft und als Standard fortgeschrieben
- Bereits 1997 veröffentlichte das NIST eine Ausschreibung für den AES (Advanced Encryption Standard) genannten Nachfolger des DES
- Nach einer mehrjährigen Auswahlprozedur wurde im November 2001 der Rijndael-Algorithmus als AES genormt und im Mai 2002 wurde DES von AES als Standard abgelöst
- Allerdings wurde Triple DES (auch TDES oder 3DES genannt) vom NIST als Standard bis 2030 fortgeschrieben

Aufbau einer Feistel-Chiffre



- Die Rundenfunktion g einer ***Feistel-Chiffre*** berechnet das Zwischenergebnis $w^r = g(K^r, w^{r-1}) \in \{0,1\}^\ell$ in Runde r aus den beiden Hälften L^{r-1} und R^{r-1} von $w^{r-1} \in \{0,1\}^\ell$ gemäß der Vorschrift

$$L^r = R^{r-1} \quad \text{und} \quad R^r = L^{r-1} \oplus f(R^{r-1}, K^r)$$

- Hierbei ist $f : \{0,1\}^{\ell/2} \times \{0,1\}^{k'} \to \{0,1\}^{\ell/2}$ eine beliebige Funktion und k' ist die Länge der Rundenschlüssel $K^1, \ldots, K^N$
- Aus dem letzten Zwischenergebnis $w^N = L^N R^N$ in Runde N wird dann durch Vertauschung von L^N und R^N der Kryptotext $y = R^N L^N$ gebildet

Aufbau der DES-Chiffrierfunktion

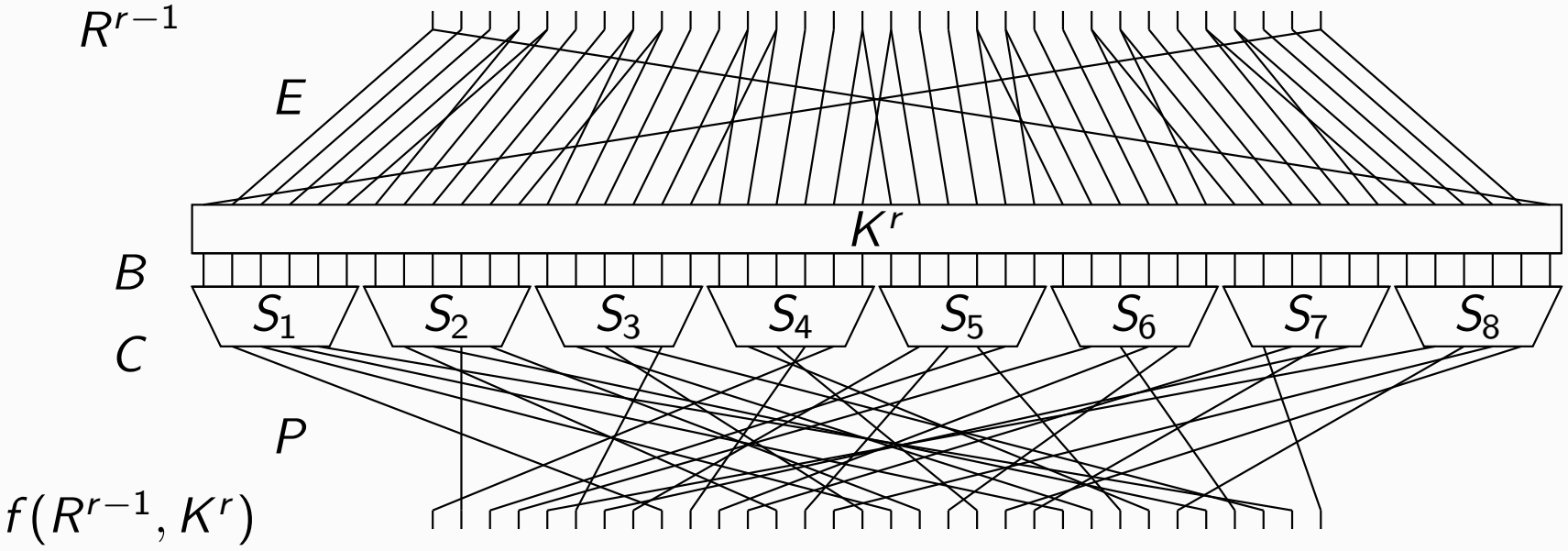
- Der DES ist eine Feistel-Chiffre mit einer Blocklänge von $\ell = 64$ Bit und $N = 16$ Runden; die (effektive) Schlüssellänge ist 56 Bit
- Der 56 Bit Schlüssel K^- ergibt zusammen mit 8 Paritätsbits (Bits 8, 16,..., 64) einen $k = 64$ Bit langen Schlüsselblock K
- Es gibt somit $2^{56} \approx 7.2 \cdot 10^{16}$ verschiedene Schlüssel
- Bei Eingabe von K und x führt der DES-Algorithmus die folgenden Chiffrierschritte aus
  - Zuerst wird der Klartext x einer Initialpermutation $IP\colon x_1 x_2 \cdots x_{64} \mapsto x_{58} x_{50} \cdots x_7$ unterzogen (siehe Folie 232)
  - Danach erfolgen 16 Runden mit einer Feistel-Rundenfunktion g und sechzehn Rundenschlüsseln $K^1, \ldots, K^{16}$ (die Berechnung von g und $K^1, \ldots, K^{16}$ wird später beschrieben)
  - Aus dem am Ende von Runde 16 ausgegebenen Block $w^{16} = L^{16} R^{16}$ wird durch Vertauschen von L^{16} und R^{16} und Anwendung von IP^{-1} der Kryptotext $\mathrm{DES}(K, x) = IP^{-1}(R^{16} L^{16})$ gebildet

Berechnung der DES-Funktion f

- Die Funktion $f : \{0,1\}^{32} \times \{0,1\}^{48} \to \{0,1\}^{32}$ wird wie folgt berechnet (für eine graphische Darstellung siehe nächste Folie)
- Bei Eingabe (R^{r-1}, K^r) wird zuerst der 32-Bit Block R^{r-1} mittels der Expansionsabbildung E (s.o.) auf einen 48-Bit Block $E(R^{r-1})$ erweitert
- Auf diesen wird bitweise der Rundenschlüssel K^r addiert
- Als Ergebnis erhalten wir den 48-Bit Block $B = E(R^{r-1}) \oplus K^r$
- Dieser wird in acht 6-Bit Blöcke $B_{(1)}, \ldots, B_{(8)}$ aufgeteilt, die mit den 8 S-Boxen $S_1, \ldots, S_8$ auf 4-Bit Blöcke $C_{(i)} = S_i(B_{(i)})$ verkleinert werden
- Die Konkatenation der von den acht S-Boxen berechneten 4-Bit Blöcke ergibt einen 32-Bit Block $C = C_{(1)} \ldots C_{(8)}$
- Zum Schluss wird auf C noch die Permutation P angewandt (siehe nächste Folie; der Werteverlauf von IP, E und P ist dort zeilenweise dargestellt)

Graphische Darstellung der DES-Funktion f

- Die DES-Funktion f



- Initialpermutation IP, Expansion E und Permutation P

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

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

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

Tabellarische Darstellung der acht DES S-Boxen

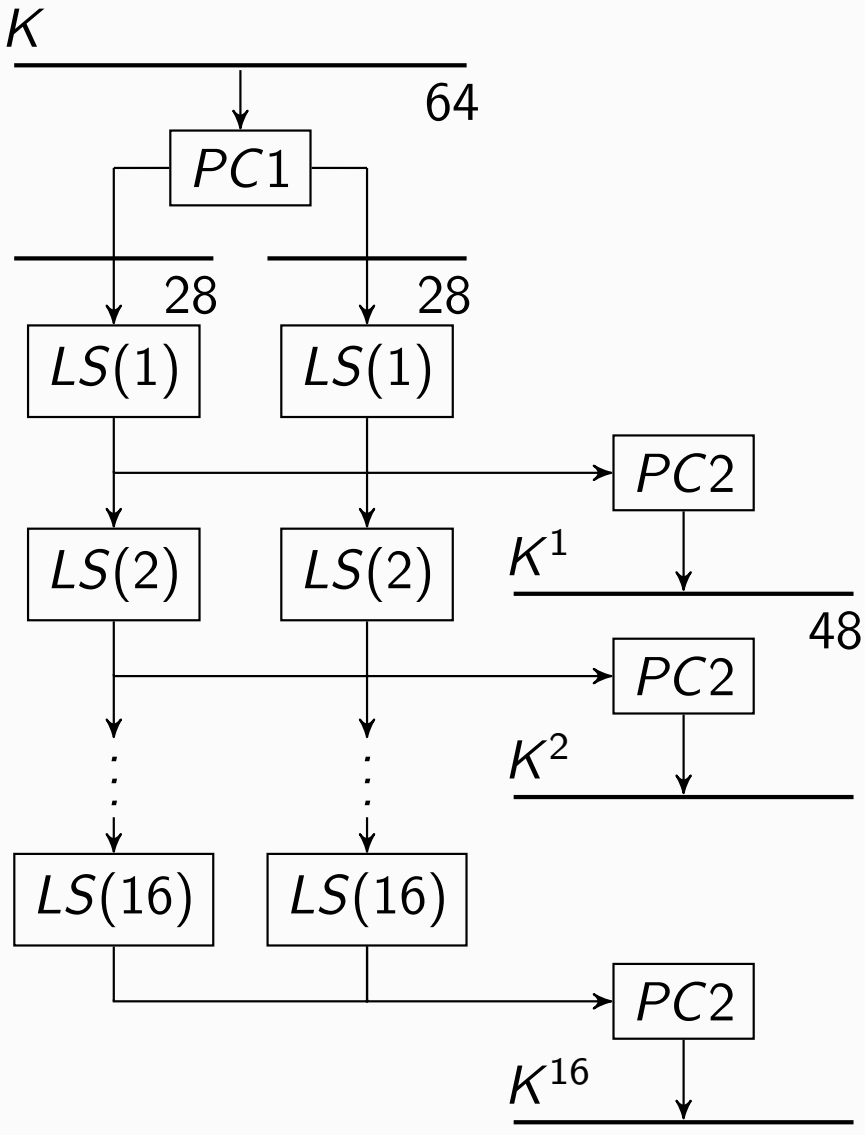
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
S_1:	E	4	D	1	2	F	B	8	3	A	6	C	5	9	0	7
	0	F	7	4	E	2	D	1	A	6	C	B	9	5	3	8
	4	1	E	8	D	6	2	B	F	C	9	7	3	A	5	0
	F	C	8	2	4	9	1	7	5	B	3	E	A	0	6	D
S_2:	F	1	8	E	6	B	3	4	9	7	2	D	C	0	5	A
	3	D	4	7	F	2	8	E	C	0	1	A	6	9	B	5
	0	E	7	B	A	4	D	1	5	8	C	6	9	3	2	F
	D	8	A	1	3	F	4	2	B	6	7	C	0	5	E	9
S_3:	A	0	9	E	6	3	F	5	1	D	C	7	B	4	2	8
	D	7	0	9	3	4	6	A	2	8	5	E	C	B	F	1
	D	6	4	9	8	F	3	0	B	1	2	C	5	A	E	7
	1	A	D	0	6	9	8	7	4	F	E	3	B	5	2	C
S_4:	7	D	E	3	0	6	9	A	1	2	8	5	B	C	4	F
	D	8	B	5	6	F	0	3	4	7	2	C	1	A	E	9
	A	6	9	0	C	B	7	D	F	1	3	E	5	2	8	4
	3	F	0	6	A	1	D	8	9	4	5	B	C	7	2	E
S_5:	2	C	4	1	7	A	B	6	8	5	3	F	D	0	E	9
	E	B	2	C	4	7	D	1	5	0	F	A	3	9	8	6
	4	2	1	B	A	D	7	8	F	9	C	5	6	3	0	E
	B	8	C	7	1	E	2	D	6	F	0	9	A	4	5	3
S_6:	C	1	A	F	9	2	6	8	0	D	3	4	E	7	5	B
	A	F	4	2	7	C	9	5	6	1	D	E	0	B	3	8
	9	E	F	5	2	8	C	3	7	0	4	A	1	D	B	6
	4	3	2	C	9	5	F	A	B	E	1	7	6	0	8	D
S_7:	4	B	2	E	F	0	8	D	3	C	9	7	5	A	6	1
	D	0	B	7	4	9	1	A	E	3	5	C	2	F	8	6
	1	4	B	D	C	3	7	E	A	F	6	8	0	5	9	2
	6	B	D	8	1	4	A	7	9	5	0	F	E	2	3	C
S_8:	D	2	8	4	6	F	B	1	A	9	3	E	5	0	C	7
	1	F	D	8	A	3	7	4	C	5	6	B	0	E	9	2
	7	B	4	1	9	C	E	2	0	6	A	D	F	3	5	8
	2	1	E	7	4	A	8	D	F	C	9	0	3	5	6	B

- Aus den Tabellen lassen sich die Werte $S_i(B_{(i)})$ wie folgt ermitteln:
  - Ist $B_{(i)} = b_1 \cdots b_6$, so findet man $S_i(B_{(i)})$ in der Tabelle für S_i in Zeile $b_1 b_6$ und Spalte $b_2 b_3 b_4 b_5$
  - Zum Beispiel ist $S_1(011010) = 1001$, da in Zeile $(00)_2 = 0$ und Spalte $(1101)_2 = \mathtt{D}$ die Hexadezimalziffer $9 = (1001)_2$ steht

Der Key-Schedule Algorithmus des DES

- Die 16 Rundenschlüssel $K^1, \ldots, K^{16}$ werden wie folgt aus dem externen 64-Bit Schlüssel K berechnet (siehe auch nächste Folie; dort ist der Werteverlauf von $PC1$ und $PC2$ zeilenweise dargestellt)
- Zuerst wählt die Funktion $PC\,1$ (permuted choice 1) aus dem Schlüssel K die kryptografisch relevanten Bits aus und permutiert sie
- Das Resultat wird in zwei 28-Bit Blöcke unterteilt, die in 16 Runden $r = 1, \ldots, 16$ jeweils zyklisch um $LS(r) \in \{1, 2\}$ Bit verschoben werden
- Aus den beiden Blöcken nach Runde r bestimmt die Funktion $PC\,2$ (permuted choice 2) jeweils den Rundenschlüssel K^r und entfernt dabei die 8 Bits an den Positionen 9, 18, 22, 25, 35, 38, 43 und 56

Der Key-Schedule Algorithmus des DES



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

r	1	2	3	4	5	6	7	8
$LS(r)$	1	1	2	2	2	2	2	2

r	9	10	11	12	13	14	15	16
$LS(r)$	1	2	2	2	2	2	2	1

Der Key-Schedule Algorithmus des DES

Definition

Ein DES-Schlüssel K heißt ***schwach***, falls alle durch ihn erzeugten Rundenschlüssel gleich sind (d.h. es gilt $\{K^1, \ldots, K^{16}\} = \{K^1\}$)

- Der DES hat folgende vier schwache Schlüssel (hexadezimal):

K	erzeugter Rundenschlüssel
0101010101010101	000000000000
1F1F1F1F0E0E0E0E	000000111111
E0E0E0E0F1F1F1F1	111111000000
FEFEFEFEFEFEFEFE	111111111111

- Für jeden von ihnen gilt $\mathrm{DES}(K, \mathrm{DES}(K, x)) = x$ (siehe Übungen)
- Zudem existieren noch sechs sogenannte semischwache Schlüsselpaare (K, K') mit der Eigenschaft $\mathrm{DES}(K', \mathrm{DES}(K, x)) = x$ (siehe Übungen)

Sicherheitsbedenken gegen den DES

- Der DES konnte sich nicht sofort nach seiner Veröffentlichung im Jahre 1975 durchsetzen
- Er wurde von manchen Behörden und Banken in den USA zunächst nicht verwendet, da folgende Sicherheitsbedenken gegen ihn bestanden:
  - Die 56-Bit Schlüssellänge bietet eine zu geringe Sicherheit gegen einen Brute-Force Angriff bei bekanntem oder wählbarem Klartext
  - Die Entwurfskriterien für die einzelnen Komponenten, insbesondere die S-Boxen, sind nicht veröffentlicht worden; daher wurde der Verdacht geäußert, dass der DES Falltüren besitzt, um einen Angriff durch die National Security Agency (NSA) zu ermöglichen
  - Kryptoanalytische Untersuchungen, die von IBM und der NSA durchgeführt wurden, blieben unter Verschluss. Als jedoch Biham und Shamir Anfang der 90er Jahre das Konzept der differentiellen Kryptoanalyse veröffentlichten, gaben die Entwickler bekannt, dass sie diesen Angriff beim Entwurf von DES bereits kannten und die S-Boxen dahingehend optimiert wurden

Sicherheitseigenschaften des DES

- Im Fall von DES ist die lineare Kryptoanalyse effizienter als die differentielle Kryptoanalyse
- Da hierzu jedoch circa 2^{43} Klartext-Kryptotext-Paare notwendig sind, stellen diese Angriffe keine realistische Bedrohung dar
  (allein die Generierung der Paare dauerte bei einem von Matsui, dem Erfinder der linearen Kryptoanalyse, durchgeführten Angriff 40 Tage)
- Dagegen gelang bereits im Juli 1998 mit einer von der Electronic Frontier Foundation (EFF) für 250 000 Dollar gebauten Maschine namens “DES Cracker” eine vollständige Schlüsselsuche in 56 Stunden
  (dies bedeutete den Gewinn der von RSA Laboratory ausgeschriebenen “DES Challenge II-2”)

Sicherheitseigenschaften des DES

- Im Jahr 1999 gewann `Distributed.Net`, eine weltweite Vereinigung von Computerfans, den mit 10 000 Dollar dotierten "DES Challenge III"
- Ihnen gelang es, durch den kombinierten Einsatz des Supercomputers "Deep Crack" und 100 000 PCs, die per Internet kommunizierten, nach 22:15h den Schlüssel zu dem bekannten Klartext „See you in Rome (second AES Conference, March 22-23, 1999)" zu finden
- Es gibt mittlerweile sogar kommerzielle Angebote im Internet (z.B. `crack.sh`), innerhalb von 26 Stunden eine vollständige Schlüsselsuche bei bekanntem Klartext auf spezieller Hardware auszuführen, um alle passenden DES-Schlüssel zu finden

Endliche Körper

Als Vorbereitung zum AES-Algorithmus gehen wir kurz auf die Arithmetik in endlichen Körpern ein. Diese spielt beim AES eine sehr wichtige Rolle

- Wie wir bereits wissen, bildet $\mathbb{Z}_p$ für primes p einen endlichen Körper der Größe p
- Dieser Körper lässt sich für jede Zahl $n \geq 1$ auf die Größe p^n erweitern
- Da bis auf Isomorphie nur ein Körper der Größe p^n existiert, wird er einfach mit $\mathbb{F}(p^n)$ oder $\mathbb{F}_{p^n}$ bezeichnet
- Um den Körper $\mathbb{F}_{p^n}$ zu konstruieren, betrachten wir zunächst den Polynomring $\mathbb{Z}_p[x]$ über $\mathbb{Z}_p$

Polynomringe

Definition. Sei R ein Ring.

- Der ***Polynomring*** $R[x]$ enthält für alle $n \geq 0$ alle Polynome $p(x)$ in der Variablen x mit Koeffizienten in R, d.h. $p(x)$ hat die Form

  $$p(x) = a_n x^n + \cdots a_1 x + a_0 \quad \text{mit} \quad a_0, \ldots, a_n \in R$$

  Man sagt, $R[x]$ entsteht aus R durch ***Adjunktion der Variablen*** x
- Der ***Grad von*** p (bezeichnet mit $\deg(p)$) ist im Fall $a_n \neq 0$ gleich n und im Fall $n = a_n = 0$ gleich -1

Man überprüft leicht, dass $R[x]$ mit der üblichen Polynomaddition und Polynommultiplikation tatsächlich einen Ring bildet

Definition (Fortsetzung).

- ***Ein Polynom*** $q(x)$ ***teilt ein Polynom*** $p(x)$ (kurz: $q(x)|p(x)$), falls ein Polynom $d(x) \in R[x]$ existiert mit $p(x) = d(x)q(x)$
- Teilt $q(x)$ die Differenz $f(x) - g(x)$ zweier Polynome, so schreiben wir

  $$f(x) \equiv_{q(x)} g(x)$$

  und sagen, $f(x)$ ist ***kongruent zu*** $g(x)$ ***modulo*** $q(x)$
- Weiterhin bezeichne

  $$p(x) \bmod q(x)$$

  das bei der Polynomdivision von $p(x)$ durch $q(x)$ auftretende ***Restpolynom***, also dasjenige Polynom $r(x)$ vom Grad $\deg(r) < \deg(q)$, für das ein Polynom $d(x) \in R[x]$ existiert mit $p(x) = d(x)q(x) + r(x)$

Faktorringe von Polynomringen

- Ähnlich wie beim Übergang von $\mathbb{Z}$ zum Restklassenring $\mathbb{Z}_m$ können wir nun jedem Polynom $p(x) \in \mathbb{Z}_m[x]$ mittels

  $$p(x) \mapsto p(x) \bmod m(x)$$

  eindeutig ein Polynom vom Grad höchstens $n-1$ zuordnen, wobei n der Grad eines fest gewählten Polynoms $m(x)$ ist
- Auf diese Weise erhalten wir den Polynomring $\mathbb{Z}_m[x]/m(x)$ aller Polynome vom Grad höchstens $n-1$
- Dieser endliche Ring heißt ***Faktorring von*** $\mathbb{Z}_m[x]$ ***modulo*** $m(x)$
- Die Addition und Multiplikation sind hierbei wie in $\mathbb{Z}_m[x]$, gefolgt von einer Reduktion modulo $m(x)$, definiert
- In den Übungen werden wir sehen, dass $\mathbb{Z}_m[x]/m(x)$ genau dann ein Körper ist, wenn m prim ist und $m(x)$ nur triviale Teiler besitzt

Irreduzible Polynome

Definition

Ein Polynom $m(x) \in \mathbb{Z}_p[x]$, p prim, vom Grad $n \geq 1$ heißt ***irreduzibel*** (über $\mathbb{Z}_p$), falls keine Polynome $p(x), q(x) \in \mathbb{Z}_p[x]$ vom Grad $\deg(p), \deg(q) \geq 1$ existieren mit

$$m(x) = p(x)q(x)$$

Satz

Der Faktorring $\mathbb{Z}_m[x]/m(x)$ ist genau dann ein Körper, wenn m prim und $m(x)$ in $\mathbb{Z}_m[x]$ irreduzibel ist.

Beweis.

Siehe Übungen. □

Arithmetik in endlichen Körpern

- Man kann zeigen, dass für primes p und jede Zahl $n \geq 1$ ein irreduzibles Polynom $m(x) = x^n + \sum_{i=0}^{n-1} m_i x^i$ vom Grad n in $\mathbb{Z}_p[x]$ existiert
- Daher lässt sich für jede Primzahlpotenz p^n ein Körper $\mathbb{Z}_p[x]/m(x)$ der Größe p^n konstruieren
- Tatsächlich gibt es bis auf Isomorphie nur einen solchen Körper, den wir mit $\mathbb{F}_{p^n}$ bezeichnen
- Wir können Polynome $a(x) = \sum_{i=0}^{n-1} a_i x^i$ auch als Koeffizientenvektoren $\vec{a} = (a_{n-1}, \ldots, a_0)$ darstellen
- Die Addition von $a(x)$ und $b(x)$ in $\mathbb{F}_{p^n}$ entspricht dann der üblichen Vektoraddition (komponentenweisen Addition modulo p) von $\vec{a}$ und $\vec{b}$
- Die Vektordarstellung $\vec{c}$ von $c(x) = a(x) + b(x)$ ist also

$$(c_{n-1}, \ldots, c_0) = (a_{n-1} + b_{n-1}, \ldots, a_0 + b_0)$$

- Im Fall $p = 2$ ist dies also die bitweise Addition modulo 2 (xor)

Arithmetik in endlichen Körpern

- Die Multiplikation in $\mathbb{Z}_p[x]/m(x)$ lässt sich wegen

$$a(x)b(x) = \sum_{i=0}^{n-1} a_i x^i b(x)$$

auf die Addition und (iterierte) Multiplikation mit dem Polynom $p(x) = x$ zurückführen

- Da $m(x)$ die Form $m(x) = \sum_{i=0}^{n} m_i x^i$ mit $m_n = 1$ hat, gilt für diese

$$xb(x) \equiv_{m(x)} xb(x) - b_{n-1}m(x) = \sum_{i=1}^{n} b_{i-1}x^i - b_{n-1}\sum_{i=0}^{n} m_i x^i$$

$$= \sum_{i=0}^{n-1}(b_{i-1} - b_{n-1}m_i)x^i, \text{ wobei wir } b_{-1} = 0 \text{ setzen}$$

- Die Multiplikation von $b(x)$ mit x entspricht somit einem Linksshift von $\vec{b}$ um eine Stelle, dem sich im Fall $b_{n-1} \neq 0$ noch die Subtraktion des Vektors $(b_{n-1}m_{n-1}, \ldots, b_{n-1}m_0)$ anschließt

Arithmetik in endlichen Körpern

- Im Fall $p = 2$ erhalten wir also

$$xb(x) = \begin{cases} \sum_{i=1}^{n-1} b_{i-1} x^i, & b_{n-1} = 0 \\ \sum_{i=0}^{n-1} (b_{i-1} \oplus m_i) x^i, & b_{n-1} = 1 \end{cases}$$

und die Vektordarstellung $\vec{c}$ von $c(x) = xb(x)$ ist

$$(c_{n-1}, \ldots, c_0) = \begin{cases} (b_{n-2}, \ldots, b_0, 0), & b_{n-1} = 0 \\ (b_{n-2}, \ldots, b_0, 0) \oplus (m_{n-1}, \ldots, m_0), & b_{n-1} = 1 \end{cases}$$

Arithmetik in endlichen Körpern

Beispiel

- Wir möchten einen endlichen Körper der Größe $p^n = 2^3$ konstruieren
- Dazu benötigen wir ein irreduzibles Polynom $m(x) \in \mathbb{Z}_2[x]$ vom Grad 3
- Setzen wir

$$m(x) = x^3 + a_2x^2 + a_1x + a_0$$

so sehen wir, dass $m(x)$ im Fall $a_0 = 0$ den nichttrivialen Teiler $p(x) = x$ hat

- Daher genügt es, die 4 Kandidaten

$$\begin{aligned} m_1(x) &= x^3 + 1 \\ m_2(x) &= x^3 + x + 1 \\ m_3(x) &= x^3 + x^2 + 1 \\ m_4(x) &= x^3 + x^2 + x + 1 \end{aligned}$$

zu betrachten

Beispiel

- Wegen

$$x^3+1=(x+1)(x^2+x+1) \quad \text{und} \quad x^3+x^2+x+1=(x+1)(x^2+1)$$

sowie

$$x^3+x+1=(x+1)(x^2+x)+1 \quad \text{und} \quad x^3+x^2+1=(x+1)x^2+1$$

gibt es in $\mathbb{Z}_2[x]$ nur zwei irreduzible Polynome vom Grad 3, nämlich x^3+x+1 und x^3+x^2+1 (da sie weder x noch $x+1$ als Teiler haben)

- Nehmen wir $m(x)=x^3+x+1$, so gilt in $\mathbb{Z}_2[x]/m(x)$ bspw. wegen $1+1=0$ die Gleichung

$$(x^2+1)+(x+1)=x^2+x$$

und wegen

$$(x^2+1)(x+1)=x^3+x^2+x+1=x^2+(x^3+x+1)\equiv_{m(x)} x^2$$

die Gleichung $(x^2+1)(x+1)=x^2$ ◁

Berechnung von multiplikativen Inversen in $\mathbb{F}_{p^n}$

- Wie in $\mathbb{Z}_p$ lässt sich das multiplikative Inverse eines Polynoms $p(x) \neq 0$ in $\mathbb{F}_{p^n}$ mit dem erweiterten Euklidschen Algorithmus berechnen

Beispiel

- Sei $p = 2$ und seien $m(x) = x^8 + x^4 + x^3 + x + 1$ und $a(x) = x^6 + x^4 + x + 1$ zwei Polynome in $\mathbb{Z}_2[x]$
- Dann können wir den (in Bezug auf den Grad) größten gemeinsamen Teiler von $m(x)$ und $a(x)$ mit dem Euklidschen Algorithmus berechnen:

i	$r_{i-1}(x) =$	$d_{i+1}(x) \cdot r_i(x)$	$+ r_{i+1}(x)$
1	$x^8 + x^4 + x^3 + x + 1 =$	$(x^2 + 1) \cdot (x^6 + x^4 + x + 1)$	$+ x^2$
2	$x^6 + x^4 + x + 1 =$	$(x^4 + x^2) \cdot x^2$	$+ x + 1$
3	$x^2 =$	$(x + 1) \cdot (x + 1)$	$+ 1$
4	$x + 1 =$	$(x + 1) \cdot \mathbf{1}$	$+ 0$

- Es gilt also $\mathrm{ggT}(a(x), m(x)) = r_4(x) = 1$

Berechnung von multiplikativen Inversen in $\mathbb{F}_{p^n}$

Beispiel (Fortsetzung)

- Der erweiterte Euklidsche Algorithmus berechnet nun Polynome $p_i(x)$ und $q_i(x)$ gemäß der Vorschrift

$$p_i(x) = p_{i-2}(x) - d_i(x) \cdot p_{i-1}(x), \text{ wobei } p_0(x) = 1 \text{ und } p_1(x) = 0,$$

und

$$q_i(x) = q_{i-2}(x) - d_i(x) \cdot q_{i-1}(x), \text{ wobei } q_0(x) = 0 \text{ und } q_1(x) = 1,$$

welche die Gleichung

$$p_i(x)m(x) + q_i(x)a(x) = r_i(x)$$

erfüllen

- Im Fall $r_i(x) = 1$ ist also $q_i(x)$ das multiplikative Inverse von $a(x)$ modulo $m(x)$

Berechnung von multiplikativen Inversen in $\mathbb{F}_{p^n}$

Beispiel (Schluss)

- Der erweiterte Euklidsche Algorithmus berechnet die Polynome $p_i(x)$ und $q_i(x)$ wie folgt:

i	$p_i(x)\cdot m(x) +$	$q_i(x)\cdot a(x)$	$=$	$r_i(x)$
0	$1\cdot m(x) +$	$0\cdot a(x)$	$=$	$m(x)$
1	$0\cdot m(x) +$	$1\cdot a(x)$	$=$	$a(x)$
2	$1\cdot m(x) +$	$(x^2+1)\cdot a(x)$	$=$	x^2
3	$(x^4+x^2)\cdot m(x) +$	$(x^6+x^2+1)\cdot a(x)$	$=$	$x+1$
4	$(x^5+x^4+x^3+x^2+1)\cdot m(x) +$	$(x^7+x^6+x^3+x)\cdot a(x)$	$=$	1

- Aus der letzten Zeile können wir nun das multiplikative Inverse $a^{-1}(x) = q_4(x) = x^7 + x^6 + x^3 + x$ von $a(x)$ modulo $m(x)$ ablesen ◁

Der Advanced Encryption Standard (AES)

- Im September 1997 veröffentlichte das NIST eine Ausschreibung für den AES mit einer Blocklänge von 128 Bit und variablen Schlüssellängen von 128, 192 und 256 Bit; Einreichungsschluss war der 15. Juni 1998
- Es gab 21 Einreichungen, aber nur 15 erfüllten alle Kriterien
- Diese wurden auf der 1. AES-Konferenz im Aug. 1998 ausgewählt und stammten aus Australien, Belgien, Costa Rica, Deutschland, Frankreich, Großbritannien, Israel, Japan, Korea, Norwegen sowie den USA
- Aus den 15 Kandidaten wurden auf der 2. AES-Konferenz im Aug. 1999 die 5 Finalisten MARS, RC6, Rijndael, Serpent und Twofish ausgewählt
- Im April 2000 wurde der Rijndael-Algorithmus auf der 3. AES-Konferenz zum Sieger erklärt und im November 2001 als AES genormt
- Die wichtigsten Entscheidungskriterien waren
  - Sicherheit,
  - Effizienz bei Software-, Hardware- und Smartcard-Implementationen
  - sowie Algorithmen- und Implementations-Charakeristika (u.a. Flexibilität und Einfachheit des Designs)

Unterschiede zwischen Rijndael und AES

- Die Blocklänge ℓ und die Schlüssellänge k sind beim Rijndael in 32 Bit Schritten jeweils zwischen 128 und 256 Bit wählbar
- Nebenstehende Tabelle zeigt die Rundenzahl N in Abhhängigkeit von ℓ und k
- Beim AES-Standard wurde die Blocklänge auf 128 Bit fixiert und die Schlüssellänge auf die Werte 128, 192 oder 256 Bit beschränkt

	k				
ℓ	128	160	192	224	256
128	**10**	11	**12**	13	**14**
160	11	11	12	13	14
192	12	12	12	13	14
224	13	13	13	13	14
256	14	14	14	14	14

- Wir beschränken uns im Folgenden auf die Beschreibung des 10-Runden AES mit $\ell = 128$ Bit Blocklänge und $k = 128$ Bit Schlüssellänge

Die AES S-Box SubByte

- Die Elemente $a(x) = \sum_{i=0}^{7} a_i x^i$ des Körpers $\mathbb{F}_{2^8} = \mathbb{Z}_2[x]/m(x)$ mit $m(x) = x^8 + x^4 + x^3 + x + 1$ können jeweils durch ein Byte $a_7 \ldots a_0$ dargestellt werden
- Zur expliziten Konvertierung der beiden Darstellungen verwenden wir folgende Funktionen:
  - Die Funktion BinaryToField: $\{0,1\}^8 \to \mathbb{F}_{2^8}$ überführt die Byte-Darstellung $a_7 \ldots a_0$ in das zugehörige Polynom $a(x) = \sum_{i=0}^{7} a_i x^i$
  - Die Funktion FieldToBinary: $\mathbb{F}_{2^8} \to \{0,1\}^8$ ist die Umkehrfunktion von BinaryToField
- Sowohl der Key-Schedule Algoritmus als auch die AES-Chiffrierfunktion verwenden eine S-Box SubByte
- Diese benutzt als nicht-linearen Bestandteil die Funktion

  $$\text{FieldInv} : \mathbb{F}_{2^8}^* \to \mathbb{F}_{2^8}^*,$$

  die das multiplikative Inverse aller Einheiten des Körpers $\mathbb{F}_{2^8}$ berechnet

Die AES S-Box SubByte

SubByte($a_7 \cdots a_0$)

1 **input** $a_7 \cdots a_0$
2 $z := \text{BinaryToField}(a_7 \cdots a_0)$
3 **if** $z \neq 0$ **then** $z := \text{FieldInv}(z)$
4 $a_7 \cdots a_0 := \text{FieldToBinary}(z)$
5 $c_7 \cdots c_0 := 01100011$
6 **for** $i := 0$ **to** 7 **do**
7 $b_i := a_i \oplus a_{i+4} \oplus a_{i+5} \oplus a_{i+6} \oplus a_{i+7} \oplus c_i$
8 **output** $b_7 \cdots b_0$

- Konkret wird SubByte durch obigen Algorithmus berechnet
- Die Indexrechnung in Zeile 7 erfolgt modulo 8
- Die Zeilen 5-8 realisieren eine affine Hill-Chiffrierfunktion

Die AES S-Box SubByte

Beispiel

- Bei Eingabe 01010011 liefert die Funktion BinaryToField das zugehörige Polynom $z = \text{BinaryToField}(01010011) = x^6 + x^4 + x + 1$
- FieldInv(z) berechnet das multiplikative Inverse $z^{-1} = x^7 + x^6 + x^3 + x$
- FieldToBinary($x^7 + x^6 + x^3 + x$) liefert den Vektor $a_7 \dots a_0 = 11001010$
- Es folgt die Berechnung der Ausgabe $b_7 \dots b_0$ mittels

$$b_7 \dots b_0 = 11001010 \begin{pmatrix} 1&0&0&0&1&1&1&1 \\ 1&1&0&0&0&1&1&1 \\ 1&1&1&0&0&0&1&1 \\ 1&1&1&1&0&0&0&1 \\ 1&1&1&1&1&0&0&0 \\ 0&1&1&1&1&1&0&0 \\ 0&0&1&1&1&1&1&0 \\ 0&0&0&1&1&1&1&1 \end{pmatrix} \oplus 01100011 = 11101101$$

- Somit ist SubByte(01010011) = 11101101
- Oder in Hexadezimaldarstellung: SubByte(53) = ED

◁

Die AES S-Box SubByte

- Wir können die AES S-Box SubByte in Form einer (16×16)-Matrix angeben, die in Zeile X und Spalte Y den Eintrag SubByte(XY) enthält

	Y															
X	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
5	53	D1	00	**ED**	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
								⋮								
F	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

Der AES Key-Schedule Algorithmus

- Beim 10-Runden AES mit Block- und Schlüssellänge $l = k = 128$ werden 11 Rundenschlüssel $K^0, \ldots, K^{10}$ der Länge 128 benutzt
- Jedes K^i besteht also aus 16 Bytes bzw. 4 Worten mit jeweils 4 Bytes
- Bei der Berechnung der Rundenschlüssel werden Wort-Konstanten $RCon[i] = \text{FieldToBinary}(x^{i-1})0^{24} \in \{0,1\}^{32}$ für $i = 1, \ldots, 10$ benutzt
- In Hexadezimal-Darstellung ergeben sich die folgenden Werte:

i	1	2	3	4	5
$RCon[i]$	01000000	02000000	04000000	08000000	10000000

i	6	7	8	9	10
$RCon[i]$	20000000	40000000	80000000	$1B000000$	36000000

Der AES Key-Schedule Algorithmus

- Reihen wir die 11 Rundenschlüssel $K^0, \ldots, K^{10}$ aneinander, so entsteht ein Array von 44 Worten $K^0 \ldots K^{10} = w[0] \ldots w[43]$
- Diese werden gemäß folgendem Algorithmus aus dem 128-Bit Schlüssel K berechnet:

KeyExpansion(K)

```
1  input K = K[0] ··· K[15]
2  for i := 0 to 3 do
3    w[i] := K[4i]K[4i + 1]K[4i + 2]K[4i + 3]
4  for i := 4 to 43 do
5    temp := w[i − 1]
6    if i ≡4 0 then
7      temp := SubWord(RotWord(temp)) ⊕ RCon[i/4]
8    w[i] := w[i − 4] ⊕ temp
9  output w[0] ... w[43]
```

Der AES Key-Schedule Algorithmus

Die hierbei benutzten Funktionen sind wie folgt definiert:

- Die Funktion RotWord ist eine 32-Bit Transposition, die die 4 Eingabebytes zyklisch um ein Byte nach links verschiebt:

$$\text{RotWord}(B_0 B_1 B_2 B_3) = B_1 B_2 B_3 B_0$$

- Die Funktion SubWord ist eine 32-Bit Substitution, die durch 4 parallel geschaltete SubByte S-Boxen realisiert wird

Der AES Chiffrieralgorithmus

Chiffrierfunktion AES(K, x)

```
1   AddRoundKey(K^0)
2   for r := 1 to 9 do
3     SubBytes
4     ShiftRows
5     MixColumns
6     AddRoundKey(K^r)
7   SubBytes
8   ShiftRows
9   AddRoundKey(K^10)
```

- Unter Benutzung der 11 Rundenschlüssel $K^0, \ldots, K^{10}$ wird der 128 Bit Klartextblock also wie folgt chiffriert
- Zuerst wird der Klartextblock x einer Addition mit dem 128-Bit Rundenschlüssel K^0 unterworfen
- Diese Operation wird mit AddRoundKey(K^0) bezeichnet

Der AES Chiffrieralgorithmus

- Danach werden 9 Runden ausgeführt, wobei in jeder Runde r der Reihe nach folgende Operationen ausgeführt werden:
  - SubBytes: eine nichtlineare 128-Bit Substitution, die durch 16 parallel geschaltete S-Boxen SubByte realisiert wird
  - ShiftRows: eine 128-Bit Transposition (siehe unten)
  - MixColumns: eine lineare 128-Bit Substitution (siehe unten) und
  - AddRoundKey(K^r): eine Schlüsseladdition mit dem Rundenschlüssel K^r
- Es folgt Runde 10 mit den Operationen SubBytes, ShiftRows und AddRoundKey(K^{10})
- Abgesehen von der zusätzlichen linearen Substitution MixColumns entspricht der Aufbau des AES also exakt dem Aufbau eines SPNs

Die Transposition ShiftRows

- Um die beiden Operationen ShiftRows und MixColumns zu beschreiben, stellen wir den Klartext $x = x_0 \cdots x_{15}$, $x_i \in \{0,1\}^8$, und alle daraus berechneten Zwischenergebnisse in Form einer Matrix $M \in \mathbb{F}_{2^8}^{(4\times 4)}$ dar
- Diese wird wie folgt initialisiert:

$$\begin{array}{cccc} x_0 & x_4 & x_8 & x_{12} \\ x_1 & x_5 & x_9 & x_{13} \\ x_2 & x_6 & x_{10} & x_{14} \\ x_3 & x_7 & x_{11} & x_{15} \end{array}$$

- Die Operation ShiftRows ist eine 128-Bit Transposition, die wie folgt definiert ist:

$$\text{ShiftRows}: \begin{array}{cccc} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{array} \mapsto \begin{array}{cccc} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,1} & s_{1,2} & s_{1,3} & s_{1,0} \\ s_{2,2} & s_{2,3} & s_{2,0} & s_{2,1} \\ s_{3,3} & s_{3,0} & s_{3,1} & s_{3,2} \end{array}$$

Die S-Box MixColumn

MixColumn$(s_{3,j}, s_{2,j}, s_{1,j}, s_{0,j})$

1 **for** $i := 0$ **to** 3 **do** $t_i :=$ BinaryToField(s_{ij})
2 $u_0 :=$ FieldMult$(x, t_0) +$ FieldMult$(x+1, t_1) + t_2 + t_3$
3 $u_1 :=$ FieldMult$(x, t_1) +$ FieldMult$(x+1, t_2) + t_3 + t_0$
4 $u_2 :=$ FieldMult$(x, t_2) +$ FieldMult$(x+1, t_3) + t_0 + t_1$
5 $u_3 :=$ FieldMult$(x, t_3) +$ FieldMult$(x+1, t_0) + t_1 + t_2$
6 **for** $i := 0$ **to** 3 **do** $s'_{ij} :=$ FieldToBinary(u_i)
7 **output** $(s'_{3,j}, s'_{2,j}, s'_{1,j}, s'_{0,j})$

- Die Operation MixColumns basiert auf einer linearen 32-Bit S-Box MixColumn, die parallel auf den vier Spalten des aktuellen Zwischenergebnisses ausgeführt wird
- Bei ihrer Berechnung wird die Funktion FieldMult: $\mathbb{F}_{2^8} \times \mathbb{F}_{2^8} \to \mathbb{F}_{2^8}$ benutzt, die ihre beiden Argumente im Körper $\mathbb{F}_{2^8}$ multipliziert

Die S-Box MixColumn

- MixColumn führt also eine lineare Transformation in dem Vektorraum $(\mathbb{F}_{2^8})^4$ aus, die sich auch wie folgt beschreiben lässt:

$$\text{MixColumn} : c_3 \dots c_0 \mapsto c_3 \dots c_0 \underbrace{\begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix}}_{Z} = c'_3 \dots c'_0$$

- Hierbei repräsentieren wir die Elemente des Körpers $\mathbb{F}_{2^8}$ als Koeffizientenvektoren in Hexadezimaldarstellung, d.h. 03 steht für $x + 1$ usw.
- Die S-Box MixColumn realisiert somit eine lineare 32-Bit Substitution $c \mapsto cZ$ auf dem Vektorraum $(\mathbb{F}_{2^8})^4$
- Zudem ist jede Zeile von Z relativ zur darüber liegenden Zeile um eine Position zyklisch nach rechts verschoben

Die S-Box MixColumn

- Aufgrund dieser speziellen Form von Z lässt sich MixColumn im Faktorring $R = \mathbb{F}_{2^8}[y]/(y^4 + 1)$ als multiplikative Chifrierfunktion $c(y) \mapsto z(y)c(y)$ beschreiben
- Stellen wir nämlich einen Vektor $(c_3, c_2, c_1, c_0) \in (\mathbb{F}_{2^8})^4$ als ein Polynom $c(y) = \sum_{i=0}^{3} c_i y^i$ in R dar und wählen wir für $z(y)$ das Polynom $z(y) = 03y^3 + 01y^2 + 01y + 02$ in R, so gilt

$$\text{MixColumn}(c(y)) = z(y)c(y)$$

- Der Ring $\mathbb{F}_{2^8}[y]/(y^4 + 1)$ ist zwar kein Körper, da das Polynom $y^4 + 1$ in $\mathbb{F}_{2^8}[y]$ nicht irreduzibel ist
- Da aber die Matrix Z invertierbar ist, kann die inverse Abbildung MixColumn^{-1} von MixColumn mittels $c \mapsto cZ^{-1}$ berechnet werden
- Somit hat auch das Polynom $z(y)$ im Ring $\mathbb{F}_{2^8}[y]/(y^4 + 1)$ ein multiplikatives Inverses $z^{-1}(y)$

Kryptoanalytische Betrachtungen

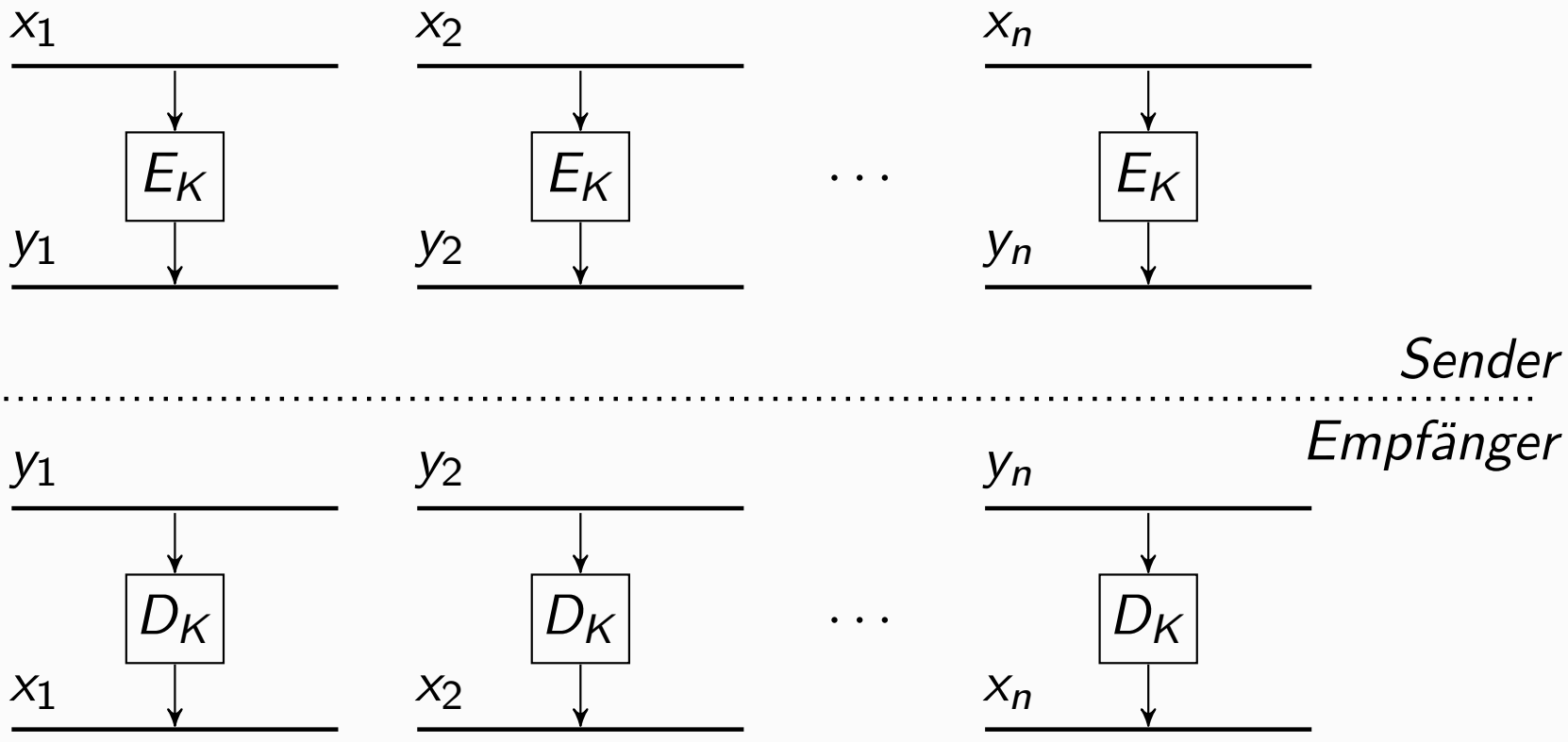
- Bis heute konnten keine Schwachstellen gefunden werden, d.h. alle bekannten Angriffe gegen den AES bringen keinen signifikanten Vorteil gegenüber einer vollständigen Schlüsselsuche
- Die Tatsache, dass für die S-Box SubByte die Inversenbildung in einem endlichen Körper benutzt wird, führt dazu, dass die Tabellen für die Güte der linearen Approximationen und für die Weitergabequotienten der Differenzenpaare einen hohen Grad an Uniformität aufweisen
- Dadurch wird die S-Box resistent gegen lineare und differentielle Analysen
- Zudem verhindert die lineare Substitution MixColumns lineare und differentielle Angriffe mit nur wenigen aktiven S-Boxen
- Diese Technik wird von den AES-Entwicklern als wide trail strategy bezeichnet

Betriebsarten von Blockchiffren

- Für den DES wurden vier verschiedene Betriebsarten vorgeschlagen, in denen grundsätzlich jede Blockchiffre E mit beliebiger Blocklänge ℓ betrieben werden kann
- Bei den ersten beiden Betriebsarten (ECB und CBC) werden Kryptotextblöcke der Länge ℓ übertragen
- Mit einer Blockchiffre kann aber auch ein Stromsystem realisiert werden, mit dem sich Kryptotextblöcke einer beliebigen Länge t, $1 \leq t \leq \ell$, übertragen lassen (siehe OFB-, CFB- und CTR-Modus)
- Der ECB-Modus ist die naheliegendste Betriebsart, wird aber in der Praxis so gut wie nie benutzt, da sie eine Reihe von Angriffsmöglichkeiten bietet
- Zum Beispiel hatten wir gesehen, dass ein effizienter IND-CPA Gegner bei einer ECB-Übertragung leicht einen Vorteil von 1 erzielen kann

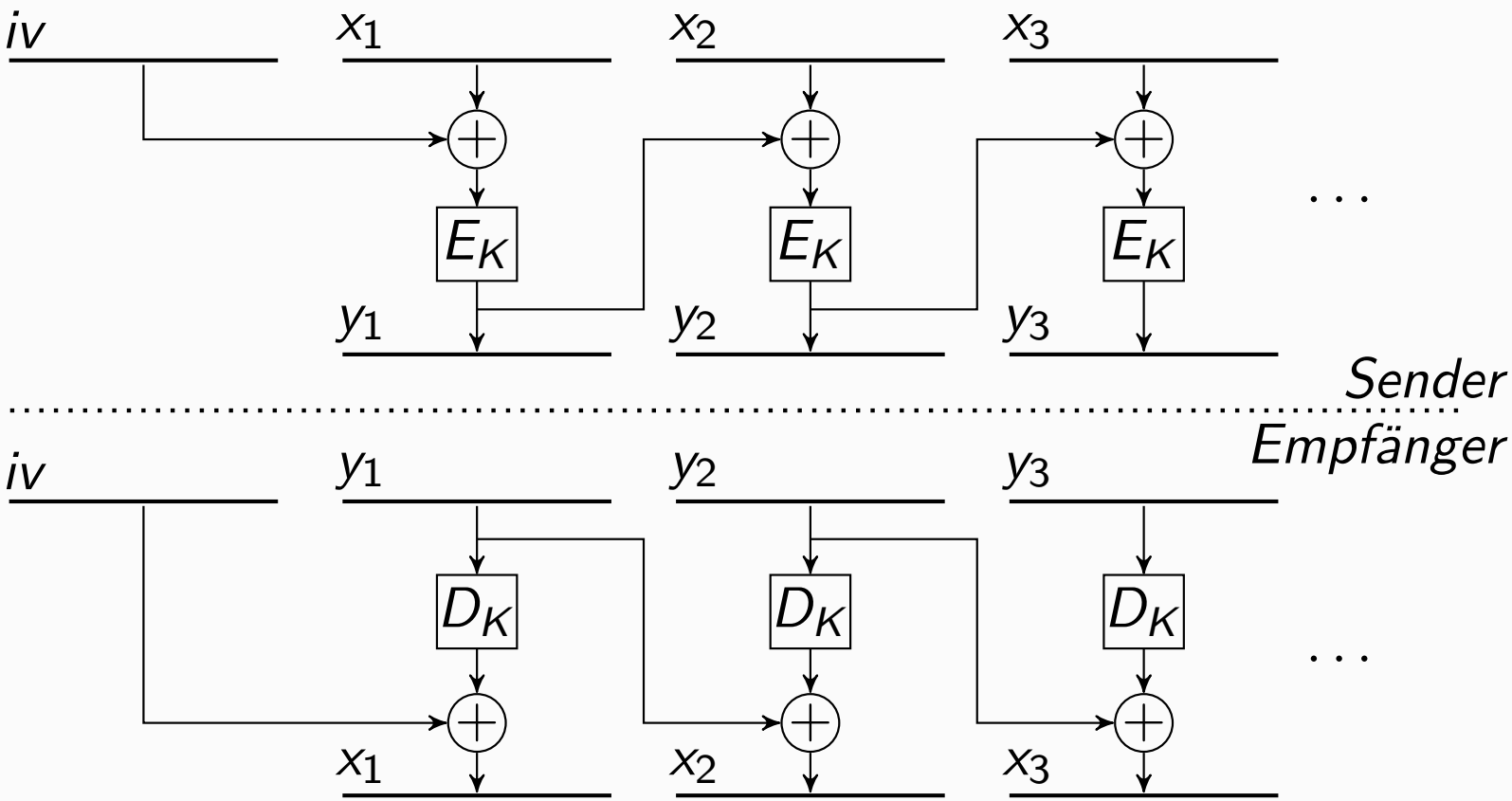
ECB-Modus (electronic codebook; elektron. Codebuch)

- Die Binärnachricht x wird in Klartextblöcke x_i der Länge ℓ zerlegt
- Der letzte Block x_n wird ggf. nach einer vereinbarten Regel aufgefüllt
- Die Blöcke werden nacheinander mit demselben Schlüssel K einzeln verschlüsselt, übertragen und auf Empfängerseite wieder entschlüsselt



CBC-Modus (cipher block chaining)

- Um unbemerkte Manipulationen des Kryptotextes zu verhindern wird jeder Kryptotextblock nicht nur in Abhängigkeit des aktuellen Klartextblocks, sondern aller vorausgehenden Blöcken verschlüsselt
- Als Folge hiervon werden gleiche Klartextblöcke auf unterschiedliche Kryptotextblöcke abgebildet

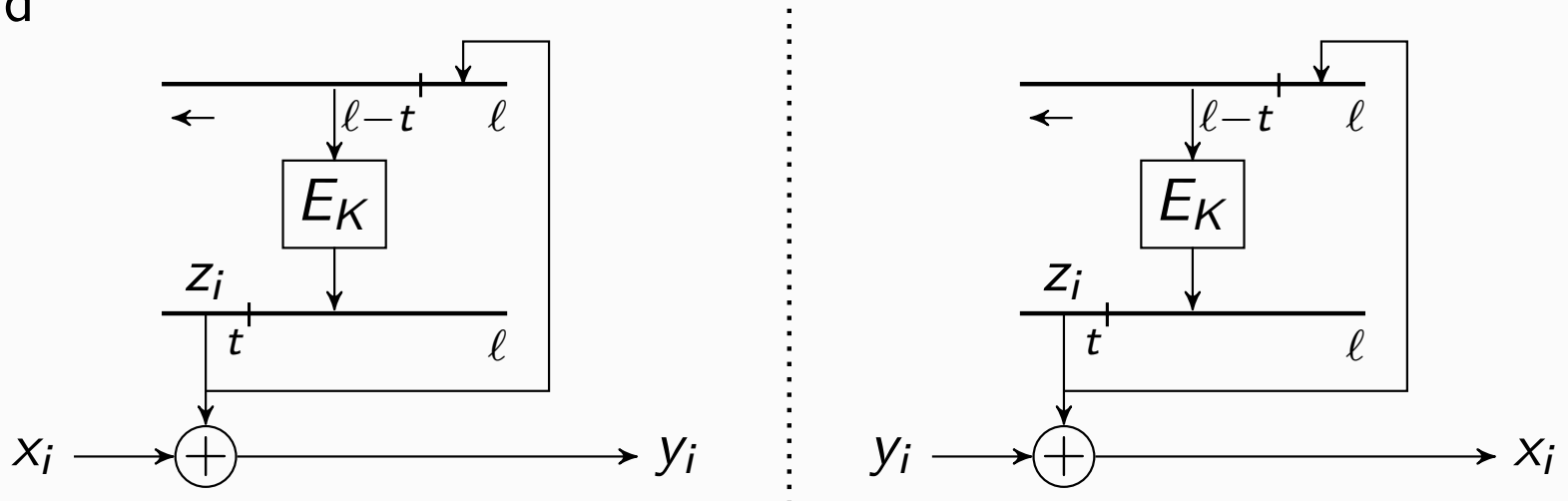


CBC-Modus (cipher block chaining)

- Jeder Klartextblock x_i wird mit dem Kryptotextblock $E_K(x_{i-1})$ bitweise (modulo 2) addiert, bevor er verschlüsselt wird
- Zur Verschlüsselung von x_1 wird ein Initialisierungsvektor iv verwendet
- Dieser wird üblicherweise unverschlüsselt übertragen, sollte aber nicht mehrmals verwendet werden
- Er wird meist durch einen Pseudozufallsgenerator erzeugt
- Der CBC-Modus verhindert auch, dass sich bestimmte Klartextmuster direkt auf den Kryptotext übertragen (was im ECB-Modus der Fall ist)
- Ein extremes Beispiel ist die Verschlüsselung einer Graphikdatei x, deren Pixel durch ℓ-Bit Blöcke kodiert sind
- Zwar werden dann im ECB-Modus die Pixel substituiert, aber ansonsten stellt der Kryptotext y exakt dieselbe Graphik wie der Klartext x dar
- Dadurch wird eine „visuelle Entschlüsselung“ durch Betrachten der verschlüsselten Graphikdatei y ermöglicht

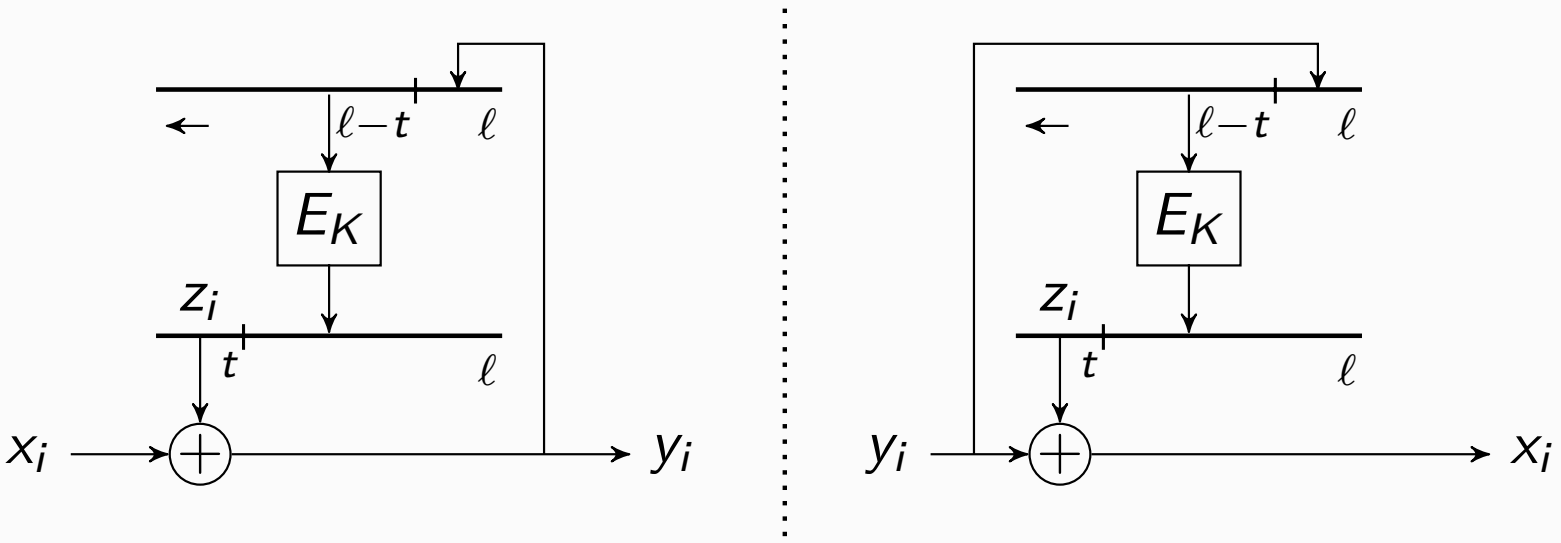
OFB-Modus (output feedback)

- Die Binärnachricht x wird in t-Bit Blöcke ($1 \leq t \leq \ell$) zerlegt
- Die Chiffrierfunktion E_K erzeugt eine pseudozufällige Folge von Blöcken z_i, die auf die entsprechenden Klartextblöcke x_i addiert werden
- Als Eingabe für die Chiffrierfunktion E_K dient ein Schieberegister, das anfangs mit einem Initialisierungsvektor iv geladen wird
- Zur Verschlüsselung jedes t-Bit Klartextblockes x_i erzeugt E_K zunächst einen Ausgabevektor, von dem nur die ersten t Bits verwendet werden
- Mit diesem Block z_i wird der Kryptoxtblock $y_i = x_i \oplus z_i$ berechnet und auch das Eingaberegister modifiziert, in das z_i von rechts geschoben wird



OFB-Modus (output feedback)

- Ähnlich zum OFB-Modus, nur dass zur Erneuerung des Eingaberegisters nicht die ersten t Bits z_i der E_K-Ausgabe, sondern der zugehörige t-Bit Kryptotextblock $y_i = x_i \oplus z_i$ verwendet wird



CTR-Modus (counter mode)

- Bei dieser Variante des OFB-Modus' wird die Pseudozufallsfolge mit Hilfe von E_K aus einer fortlaufenden Binärblockfolge $c_0, c_1, \ldots$ mit $c_{i+1} = c_i + 1 \bmod 2^\ell$ erzeugt
- Dies hat den Vorteil, dass spätere Blöcke der Pseudozufallsfolge nicht von den vorhergehenden abhängen
- Daher können mehrere Blöcke $E_K(c_i)$ parallel berechnet werden