

Datenschutz und Datensicherheit

tweise@gmx.de

Montag, 2. Februar 2004, 06:06:00

Inhalt

Alphabet	3
erweiterter Euklidischer Algorithmus	3
Inversentabellen	3
Multiplikationstabellen	3
1. kryptographisches System	4
2. Attacken	4
3. Methoden der Kryptanalyse	4
4. Blockchiffren	4
5. Flusschiffren	4
6. Chaining	5
7. Cipher Feedback Mode	5
8. Kombination von Kryptosystemen	5
9. Cäsar-Chiffre	6
10. E: $\mathbb{Z}_n \times \mathbb{Z}_n \rightarrow \mathbb{Z}_n$ (Sonderform von Hillchiffre)	6
11. Playfair-Chiffre	7
12. Hill-Chiffre	8
13. affin-lineare Chiffre	13
14. Vigenère Chiffre / Beaufort-Chiffre	15
15. DES Chiffre	15
16. RSA-Chiffre	20
17. Merkle-Hellman Knapsack Kryptosystem	24
18. El Gamal Kryptosystem, El Gamal Signierschema	26
19. Digitale Signaturen	29
20. Geburtstagsparadoxon	30
21. $a x \equiv b \pmod{l}$	31
22. Markov-Ungleichung	31
23. Vektorräume	32
24. Funktionen	33
25. Satz von Euler	33
26. kleiner Satz von Fermat	33
27. schnelles Potenzieren	34
28. schnelles Wurzelziehen	34
29. heuristischer Primzahltest von Solovay und Strassen	35
30. deterministischer Primzahlalgorithmus von Agrawal, Kayal und Soxena	35
31. chinesischer Restsatz	36
32. Las-Vegas-Algorithmus	38
33. Logarithmieren	38
34. Ordnung eines Elements	38
35. superwachsende Folgen	39
36. erzeugendes Elemente in \mathbb{Z}_p^*	39
37. Binärzahlen der Form mm	40

Alphabet

A	B	C	D	E	F	G	H	I	J	K	L	M
0	1	2	3	4	5	6	7	8	9	10	11	12
N	O	P	Q	R	S	T	U	V	W	X	Y	Z
13	14	15	16	17	18	19	20	21	22	23	24	25

Multiplikationstabellen

26			
1	26	15	390
2	52	16	416
3	78	17	442
4	104	18	468
5	130	19	494
6	156	20	520
7	182	21	546
8	208	22	572
9	234	23	598
10	260	24	624
11	286	25	650
12	312		
13	338		
14	364		

27			
1	27	15	405
2	54	16	432
3	81	17	459
4	108	18	486
5	135	19	513
6	162	20	540
7	189	21	567
8	216	22	594
9	243	23	621
10	270	24	648
11	297	25	675
12	324	26	702
13	351		
14	378		

29			
1	29	15	435
2	58	16	464
3	87	17	493
4	116	18	522
5	145	19	551
6	174	20	580
7	203	21	609
8	232	22	638
9	261	23	667
10	290	24	696
11	319	25	725
12	348	26	754
13	377	27	783
14	406	28	812

Inversentabellen

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

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

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

```

void euklid(int a, int b)    erweiterter Euklidischer Algorithmus, Lagrange)
{
    int i0 = 0, i1 = 1, mod = b, t, d;
    while((a > 0) && (b > 0))
    {
        if(a > b) { t = b; b = a; a = t; } //tauschen, nun auf alle Fälle a ≤ b
        d = b / a; //ganzzahliges Divisionsergebnis
        b = b - (d * a); // = b mod a, b ist "Rest von b/a"
        t = i1; //alten i1-Wert aufheben
        i1 = i0 - (d*i1); //neuen i1-Wert berechnen
        i0 = t; //i0 = altes i1
    }
    System.out.println("ggT(a, b) = " + a);
    if(a == 1) { if(i0 < 0) i0 = mod + i0;
        System.out.println("Inverses zu b im Restklassenring a = " + i0); }
}

```

1. kryptographisches System \equiv 5-Tupel (M, K, C, E, D)

- M Menge der Nachrichten, eine Nachricht $m \in M$, Plaintext
- K Menge der Schlüssel, ein Schlüssel $k \in K$, Key
- C Menge der Chiffre (verschlüsselte Nachrichten), ein Chiffre $c \in C$, Kryptogramme
- E Verschlüsselungsfunktion $E: M \times K \rightarrow C$, $E(m, k) = c$
- D Entschlüsselungsfunktion $D: C \times K \rightarrow M$, $D(c, k) = m$

symmetrisches Kryptosystem: k' aus k leicht ableitbar, sonst asymmetrisch

2. Attacken

- ciphertext-only attack* einige Kryptogramme c_1, \dots, c_n
- known plaintext attack* einige Klartext-Kryptogramm-Paare $(c_1, m_1), \dots, (c_n, m_n)$
- chosen plaintext attack* kann Klartexte verschlüsseln $m_i \rightarrow c_i$
- chosen ciphertext attack* kann entschlüsseln $c_i \rightarrow m_i$

3. Methoden der Kryptanalyse

- Probable-Word Methode* suche nach häufig vorkommenden Wörtern („Sehr geehrte Damen ...“)
- vollständige Suche* alle Schlüssel werden ausprobiert
- strukturelle Zerlegung des Schlüsselraums* Wenn Schlüssel k in k_1 und k_2 zerlegt werden kann, so dass ein Teil des Kryptogramms nur mit k_1 , der andere nur mit k_2 verschlüsselt wird, so kann man das System leichter entschlüsseln. Jedes Kryptogrammbit soll daher von jedem Schlüsselbit abhängen.
- Sprachstatistik* Häufigkeit von Buchstaben, Digrammen und Trigrammen

4. Blockchiffren (block ciphers, ECB-Modus)

Nachricht wird in Blöcke der Länge p zerlegt, die einzeln verschlüsselt werden.
Die Caesar-Chiffre ist eine 1-1-Blockchiffre.

$$E: M^p \times K \rightarrow C^p, E(m^p, k) = c^p$$
$$D: C^p \times K \rightarrow M^p, D(c^p, k) = m^p$$

Aufgabe 1/3 Blockchiffre analog zur Cäsarchiffre:
 $\mathbb{Z}_n^2 \times \mathbb{Z}_n^2 \rightarrow \mathbb{Z}_n^2, E((m, m'), (k, k')) = (m + k \bmod 26, m' + k' \bmod 26)$

5. Flusschiffren (stream ciphers)

Für jeden zu verschlüsselnden Buchstaben wird anderer Schlüssel verwendet.
Erster Schlüssel k wird als Eingabe für deterministischen Zufallsgenerator verwendet, der unendlich lange Kette von Schlüsseln z_i ausspuckt.

$$E(m_i, k) = m_i \oplus z_i = c_i \qquad D(c_i, k) = c_i \oplus z_i = m_i$$

Bei einem Schlüssel k der Länge d und einem Zufallsgeneratorpolynom p gilt: $z_j = \sum_{i=1}^d p_i \oplus z_{j-i} \bmod 2$

6. Chaining (cipher book chaining mode, CBC-mode)

Verschlüsselung eines Blocks hängt von allen vorherigen ab, E_0 ist eine beliebige Verschlüsselungsfunktion, D_0 ihre Entschlüsselungsfunktion. Zusätzlich wird Initialisierungsfolge z gewählt.

$$E(m_i, k) = E_0(m_i \oplus c_{i-1}) = c_i; E(m_1, k) = E_0(m_1 \oplus z) = c_1$$
$$D(c_i, k) = D_0(c_i, k) \oplus c_{i-1} = m_i; D(c_1, k) = D_0(c_1, k) \oplus z = m_1$$

Ein Bitfehler wirkt sich höchstens auf m_j und m_{j+1} aus. Entschlüsselung kann nur nach Verschlüsselung erfolgen.

7. Cipher Feedback Mode (CFB)

Initialisierungsvektor z mit Länge n . Nehmen Zahl $p \leq n$, zerlegen Nachricht in r Blöcke der Länge p . Setzen $I_1 = z$.

Verschlüsselt wird wie folgt:

- 1) $t_j =$ erste p bits von $E(I_j, k)$
- 2) $c_j = m_j \oplus t_j$
- 3) $I_{j+1} = (2^p * I_j + c_j) \bmod 2^n$ rechte $(n-p)$ -Bits von I_j + Bits von c_j

Entschlüsselt wird nach $I_1 = z$ und

- 1) $t_j =$ erste p Bits von $E(I_j, k)$
- 2) $m_j = c_j \oplus t_j$
- 3) $I_{j+1} = (2^p * I_j + c_j) \bmod 2^n$ rechte $(n-p)$ -Bits von I_j + Bits von c_j

Vorteil gegenüber CBC: Sender und Empfänger können gleichzeitig t_{j+1} berechnen, sobald c_j bekannt ist.

Ein Bitumkehrfehler wirkt sich auf die nächsten $\left\lceil \frac{n}{p} \right\rceil$ Blöcke aus.

8. Kombination von Kryptosystemen

Komposition $E(m, k) = E_1(E_2(m, k))$ $D(c, k) = D_2(D_1(c, k))$

Produktbildung $E((m_1, m_2), k) = (E_1(m_1, k), E_2(m_1, k))$ $D((c_1, c_2), k) = (D_1(c_1, k), D_2(c_2, k))$

9. Cäsar-Chiffre

Aufgabe 1/1

$$M = C = K = \text{Alphabet } \mathbb{Z}_n = \{0, \dots, n-1\}$$

$$E(m, k) \equiv (m + k) \pmod{n} = c$$

$$D(c, k) \equiv (c - k) \pmod{n} = m$$

Der Gegner kennt einige Chiffre c_1, \dots, c_l

- und den Schlüssel k und den Modulus n :
er kann alles, ver- und entschlüsseln
- und den Schlüssel k
er muss n bestimmen, Ansatz: $n' = \max_{i=1..m} \{c_i\} + 1$
mit n' entschlüsseln, wenn nach Sprachstatistik was sinnvolles rauskommt, $n = n'$ sonst $n'+1$
- und den Modulus n
alle Schlüssel $k = 0..n-1$ ausprobieren, bis was sinnvolles rauskommt
- keines von beiden
solange Modulus mit b) abschätzen und hochzählen, bis mit c) entschlüsselt
- und den Modulus n sowie passende Nachrichten m_i zu $c_1..c_l$
Schlüssel kann mit $k \equiv (c_i - m_i) \pmod{n}$ bestimmt werden
- nur passende Nachrichten m_i zu $c_1..c_l$
vorgehen wie bei b), aber Schlüssel immer über $k \equiv (m_i - c_i) \pmod{n}$ zu n ermitteln

10. E: $\mathbb{Z}_n \times \mathbb{Z}_n \rightarrow \mathbb{Z}_n$ (Sonderform von Hillchiffre)

Aufgabe 1/2

$$E(m, k) = (m * k) \pmod{n}$$

$$D(c, k) = (c * k^{-1}) \pmod{n}$$

Bestimme sinnvolle Eigenschaften von n und k !

$E(\bullet, k)$ muss injektiv sein $\Rightarrow m_i * k \neq m_j * k \quad \forall j \neq i$; speziell: $k \neq 0$

Annahme 1: $k \mid n \Rightarrow \exists m \neq 0: m * k = n \Rightarrow m * k \equiv 0 \pmod{n} \Rightarrow m \equiv 0 \Rightarrow$ nicht injektiv

Annahme 2: $\exists m: m \mid n \Rightarrow \exists k: m * k = n \Rightarrow m * k \equiv 0 \pmod{n} \Rightarrow m \equiv 0 \Rightarrow$ nicht injektiv für Schlüssel k

weiter: Es gilt: Wenn $\text{ggT}(a, b) = g > 1 \Rightarrow a = i * g \wedge b = j * g \Rightarrow a \mid (i * j * g)$

also damit: $\text{ggT}(n, k) \neq 0 \Rightarrow \exists m: k * m = x * n \equiv 0 \Rightarrow$ nicht injektiv

$\text{ggT}(n, m) \neq 0 \Rightarrow \exists k: k * m = x * n \equiv 0 \Rightarrow$ nicht injektiv für Schlüssel k

Daraus folgt, n sollte eine Primzahl sein. Es muss auf alle Fälle $\text{ggT}(n, k) = 1$ gelten.

Oder einfacher:

Gilt $g = \text{ggT}(k, n)$ so folgt $k = i * g$ und $n = j * g$ mit $0 < j < n \in \mathbb{N}$

Damit folgt das die Nachricht $m = j$ zu $c = k * j = i * g * j = i * n \equiv 0 \pmod{n}$ verschlüsselt wird, und nicht mehr von der Nachricht $m = 0$ zu unterscheiden ist. Damit ist die Injektivität verletzt. $E(0, k) = E(j, k)$.

11. Playfair-Chiffre

Alphabet in $n \times n$ -Matrix eintragen, z.B. 5×5 für englisches Alphabet ohne j
 immer ein Paar $m = (m_1, m_2)$ von Buchstaben ver- und entschlüsseln
 ist $m_1 = m_2$, so wird ein verabredeter Buchstabe dazwischen eingefügt (z.B. q)

Verschlüsseln:

- 1) sind m_1 und m_2 in der gleichen Zeile, so nimm jeweils die Buchstaben rechts daneben
 $(m, m') = (k_{i,n}, k_{i,m}) : n \neq m$
 $(c, c') = (k_{i,n+1 \bmod 5}, k_{i,m+1 \bmod 5})$
- 2) sind m_1 und m_2 in der gleichen Spalte, so nimm jeweils die Buchstaben darunter
 $(m, m') = (k_{i,n}, k_{j,n}) : i \neq j$
 $(c, c') = (k_{i+1 \bmod 5, n}, k_{j+1 \bmod 5, n})$
- 3) gilt weder 1) noch 2), so bilden m_1 und m_2 ein Rechteck, nimm also die Buchstaben von den freien Ecken dieses Rechtecks
 $(m, m') = (k_{i,n}, k_{j,m}) : i \neq j, n \neq m$
 $(c, c') = (k_{j,n}, k_{i,m})$

Entschlüsseln:

- 1) sind m_1 und m_2 in der gleichen Zeile, so nimm jeweils die Buchstaben links daneben
 $(c, c') = (k_{i,n}, k_{i,m}) : n \neq m$
 $(m, m') = (k_{i,n-1 \bmod 5}, k_{i,m-1 \bmod 5})$
- 2) sind m_1 und m_2 in der gleichen Spalte, so nimm jeweils die Buchstaben darüber
 $(c, c') = (k_{i,n}, k_{j,n}) : i \neq j$
 $(m, m') = (k_{i-1 \bmod 5, n}, k_{j-1 \bmod 5, n})$
- 3) gilt weder 1) noch 2), so bilden m_1 und m_2 ein Rechteck, nimm also die Buchstaben von den freien Ecken dieses Rechtecks
 $(c, c') = (k_{i,n}, k_{j,m}) : i \neq j, n \neq m$
 $(m, m') = (k_{j,n}, k_{i,m})$

$25!$ Schlüssel, einige mit gleicher Wirkung, daher $\frac{25!}{5^5} = 24!$.
 (5 gleiche Spalten- * 5 gleiche Zeilenwirkung)

a	b	c	d	=	d	a	b	c
e	f	g	h		h	e	f	g
i	j	k	l		l	i	j	k
m	n	o	p		p	m	n	o

choosen plaintext: alle $25 \cdot 24 = 600$ Paare ausprobieren
cyphertext only: Sprachstatistik

12. Hill-Chiffre

$$E(m, k) = (k * m) \bmod n$$
$$D(c, k) = (k^{-1} * c) \bmod n$$

Bei der Hill-Chiffre wird der Text mit dem Schlüssel multipliziert. Jedoch wird hier eine $p \times p$ -Matrix als Schlüssel genommen.

chosen plaintext

Man wählt die Einheitsvektoren $e_i = (0, \dots, 0, 1, 0, \dots, 0)^T$ für $i = 1..p$ als Nachrichten. Da sie die Einheitsmatrix bilden, erhält man genau k . Das kann man dann noch invertieren.

$$E(e, k) = k * e_i = \begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix} * \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix} = k$$

chosen ciphertext

Hier kann man genauso vorgehen:

$$D(c, k^{-1}) = k^{-1} * e_i = \begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix} * \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix} = k^{-1}$$

known plaintext

Man benötigt ein Erzeugersystem im \mathbb{Z}_n^p , bestehend aus m_i s oder c_i s, also eine Basis im \mathbb{Z}^p . Dafür sind logischerweise mindestens p ciphertext-plaintext-Paare notwendig. Aus 23. Vektorräume, Seite 32 geht hervor, dass man als maximalen Erwartungswert 2^p Paare zufällig ziehen muss, um eine Basis zu erhalten.

Aufgabe 2/4

Bekannt sei $(m, c) = (\text{"TURING"}, \text{"UBIXGT"})$.

Brich mit einer *known plaintext attack* die Hillchiffre der Blocklänge zwei vollständig auf. Finde dazu die Schlüsselmatrix k bzw. deren Inverse k^{-1} und entschlüssele den restlichen Text „ENERHLNHAHRM“.

T	U	R	I	N	G
19	20	17	8	13	6

U	B	I	X	G	T
20	1	8	23	6	19

$$m_1 = (19, 20)^T, m_2 = (17, 8)^T, m_3 = (13, 6)^T$$

$$c_1 = (20, 1)^T, c_2 = (8, 23)^T, c_3 = (6, 19)^T$$

Zuerst müssen wir ein n finden, indem die uns bekannten Nachrichtenvektoren eine Basis darstellen. Dafür berechnen wir die Determinanten der möglichen Kombinationen dieser Vektoren. Ihr ggT mit n muss zumindest für eine Matrix 1 ergeben.

$$\begin{vmatrix} 19 & 17 \\ 20 & 8 \end{vmatrix} = 152 - 340 = -188$$

$$\begin{vmatrix} 19 & 13 \\ 20 & 6 \end{vmatrix} = 114 - 260 = -146$$

$$\begin{vmatrix} 17 & 13 \\ 8 & 6 \end{vmatrix} = 102 - 104 = -2$$

Da alle Determinanten, genau wie 26, gerade sind, ist der ggT hier überall mindestens 2.

Im \mathbb{Z}_{26} sind sie demnach nicht invertierbar, man muss es also im \mathbb{Z}_{27} probieren, denn der ggT mit 27 ist bei allen drei Determinanten genau 1.

Wir müssen das Gleichungssystem $kM = c$ lösen. Mit M^{-1} erhält man $kMM^{-1} = cM^{-1} = k$. Dazu müssen wir zuerst eine der obigen Matrizen invertieren, wir wählen die erste (von TU-RI).

$$\left(\begin{array}{cc|cc} 19 & 17 & 1 & 0 \\ 20 & 8 & 0 & 1 \end{array} \right) \begin{array}{l} 20 + 19i \equiv 0 \pmod{27} \\ 19 * i \equiv -20 \pmod{27} \equiv 7 \pmod{27} \end{array}$$

$$\Downarrow \begin{array}{l} 19 * i * 19^{-1} = i \equiv 7 * 19^{-1} \pmod{27} \\ i = 7 * 10 \pmod{27} \\ i = 16 \end{array}$$

$$\left(\begin{array}{cc|cc} 19 & 17 & 1 & 0 \\ 0 & 10 & 16 & 1 \end{array} \right) \begin{array}{l} i = 7 * 10 \pmod{27} \\ i = 16 \end{array}$$

$$\Downarrow \left(\begin{array}{cc|cc} 19 & 17 & 1 & 0 \\ 0 & 1 & 7 & 19 \end{array} \right) \begin{array}{l} 10 * 10^{-1} \equiv 1 \Rightarrow 10 * 19 \equiv 1 \\ 16 * 19 = 304 \equiv 7 \pmod{27} \end{array}$$

$$\Downarrow \left(\begin{array}{cc|cc} 19 & 0 & 17 & 1 \\ 0 & 1 & 7 & 19 \end{array} \right) \begin{array}{l} 17 + i * 1 \equiv 0 \pmod{27} \\ i \equiv 10 \pmod{27} \\ 1 + 10 * 7 = 71 \equiv 17 \pmod{27} \end{array}$$

$$\Downarrow \left(\begin{array}{cc|cc} 1 & 0 & 8 & 10 \\ 0 & 1 & 7 & 19 \end{array} \right) \begin{array}{l} 0 + 10 * 19 \equiv 1 \pmod{27} \\ 19 * 19^{-1} \equiv 1 = 19 * 10 \\ 17 * 10 = 170 \equiv 8 \pmod{27} \end{array}$$

Das Inverse zu 19 berechnen wir mit dem erweiterten Euklidischen Algorithmus.

$$19^{-1} = 10 \Leftrightarrow 10^{-1} = 19$$

$$\begin{array}{l} 27 - 1 * 19 = 8 \\ 0 - 1 * 1 = -1 \\ 19 - 2 * 8 = 3 \\ 1 - 2 * -1 = 3 \\ 8 - 2 * 3 = 2 \\ -1 - 2 * 3 = -7 \\ 3 - 1 * 2 = 1 \\ 3 - 1 * -7 = 10 \\ 2 - 2 * 1 = 0 \\ 7 - 2 * 10 = -27 \end{array}$$

	q	a	b	i
	-	19	27	0
Schritt 1	1	19	8	1
2	2	3	8	3
3	2	3	2	-7
4	1	1	2	10
5	2	1	0	-27

Mit der so erhaltenen Matrix M^{-1} müssen wir die Matrix der Chiffren (UB-IX) multiplizieren, um k zu erhalten.

$$\begin{pmatrix} 20 & 8 \\ 1 & 23 \end{pmatrix} \begin{pmatrix} 8 & 10 \\ 7 & 19 \end{pmatrix} = \begin{pmatrix} 216 & 352 \\ 169 & 447 \end{pmatrix} \equiv \begin{pmatrix} 0 & 1 \\ 7 & 15 \end{pmatrix} = k$$

Zum Schluss müssen wir die Matrix k noch invertieren.

$$\left(\begin{array}{cc|cc} 0 & 1 & 1 & 0 \\ 7 & 15 & 0 & 1 \end{array} \right)$$

↓

$$\left(\begin{array}{cc|cc} 7 & 15 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{array} \right)$$

Zeilen tauschen

↓

$$\left(\begin{array}{cc|cc} 7 & 0 & 12 & 1 \\ 0 & 1 & 1 & 0 \end{array} \right)$$

$15 + i * 1 \equiv 0 \pmod{27}$
 $i * 1 = i = 27 - 15 = 12$

↓

$$\left(\begin{array}{cc|cc} 1 & 0 & 21 & 4 \\ 0 & 1 & 1 & 0 \end{array} \right)$$

$7 * 7^{-1} = 7 * 4 \equiv 0 \pmod{27}$
 $4 * 12 = 48 \equiv 21 \pmod{27}$

Wir benötigen das Inverse zu 7.

$$7^{-1} = 4 \Leftrightarrow 4^{-1} = 7$$

$$\begin{aligned} 27 - 3 * 7 &= 6 \\ 0 - 3 * 1 &= -3 \\ 7 - 1 * 6 &= 1 \\ 1 - 1 * -3 &= 4 \\ 6 - 6 * 1 &= 0 \\ -3 - 6 * 4 &= -27 \end{aligned}$$

Schritt

	q	a	b	i
	-	7	27	0
1	3	7	6	-3
2	1	1	6	4
3	6	1	0	-27

Nun sind sowohl k als auch k^{-1} bekannt:

$$k = \begin{pmatrix} 0 & 1 \\ 7 & 19 \end{pmatrix}$$

$$k^{-1} = \begin{pmatrix} 21 & 4 \\ 1 & 0 \end{pmatrix}$$

Man hätte k und k^{-1} genauso andersherum errechnen können, indem man das Gleichungssystem der Deciffrierfunktion $k^{-1} * c = m$ mit $k^{-1} * c * c^{-1} = m * c^{-1} = k^{-1}$ löst.

Dafür wird zuerst die Chifferratmatrix von „UB“ – „IX“ invertiert.

$$\left(\begin{array}{cc|cc} 20 & 8 & 1 & 0 \\ 1 & 23 & 0 & 1 \end{array} \right)$$

↓

$$\left(\begin{array}{cc|cc} 1 & 23 & 0 & 1 \\ 20 & 8 & 1 & 0 \end{array} \right)$$

Zeilentausch

↓

$$\left(\begin{array}{cc|cc} 1 & 23 & 0 & 1 \\ 0 & 7 & 1 & 7 \end{array} \right)$$

$20 - 20 * 1 \equiv 0 \pmod{27}$

↓

$$\left(\begin{array}{cc|cc} 1 & 0 & 16 & 5 \\ 0 & 1 & 4 & 1 \end{array} \right)$$

$7 * 4 \equiv 1 \pmod{27}$
 $21 - 23 * 1 \equiv 0 \pmod{27}$

Das Inverse zu 7 ist uns ja bereits bekannt.

Dann muss die c^{-1} -Matrix noch mit der passenden Nachrichtenmatrix multipliziert werden: $m * c^{-1} = k^{-1}$.

$$\begin{pmatrix} 19 & 17 \\ 20 & 8 \end{pmatrix} * \begin{pmatrix} 16 & 5 \\ 4 & 1 \end{pmatrix} = \begin{pmatrix} 372 & 112 \\ 352 & 108 \end{pmatrix} \equiv \begin{pmatrix} 21 & 4 \\ 1 & 0 \end{pmatrix} \equiv k^{-1} \pmod{27}$$

Das stimmt mit unserer Lösung von vorhin überein, ist also richtig.

Als nächstes muss der restliche Text entschlüsselt werden:

E	N	E	R	H	L	N	H	A	H	R	M
4	13	4	17	7	11	13	7	0	7	17	12

$$m_4 = (4, 13)^T, m_5 = (4, 17)^T, m_6 = (7, 11)^T, m_7 = (13, 7)^T, m_8 = (0, 7)^T, m_9 = (17, 12)^T$$

$$k^{-1} * C = \begin{pmatrix} 21 & 4 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 4 & 4 & 7 & 13 & 0 & 17 \\ 13 & 17 & 11 & 7 & 7 & 12 \end{pmatrix} = \begin{pmatrix} 136 & 152 & 191 & 301 & 28 & 405 \\ 4 & 4 & 7 & 12 & 0 & 17 \end{pmatrix}$$

Die obere Zeile muss noch durch ihre kongruenten Reste ersetzt werden, und man erhält:

1	4	17	4	2	7	4	12	1	0	0	17
B	E	R	E	C	H	E	N	B	A	A	R

Bekannt sind die folgenden 7 plaintext/ciphertext-Paare. Brich in einer *known plaintext attack* die Hill-Chiffre der Länge 3 vollständig auf. Finde dazu die Schlüsselmatrix k und ihre Inverse k^{-1} . **Aufgabe 2/5**

H	A	V	E	D	I	F	F	E	R	E	N	T	O	B	J	E	C	T	I	V
7	0	21	4	3	8	5	5	4	17	4	13	19	14	1	9	4	2	19	8	21

Q	Q	X	D	U	K	I	T	S	G	A	D	U	W	H	X	R	M	S	W	L
16	16	23	3	20	10	8	19	18	6	0	3	20	22	7	23	17	12	18	22	11

Zuerst müssen wir wieder probieren, ob wir eine invertierbare Matrix von Nachrichten überm \mathbb{Z}_{26} finden. Mit (HAV, EDI, FFE) geht es offenbar nicht, da hier der ggT der Determinante mit 26 ungleich 1 ist. (HAV, EDI, JEC) dagegen lässt sich invertieren. Im Gegensatz zur vorigen Aufgabe müssen wir also nicht in den \mathbb{Z}_{27} ausweichen.

$$\begin{vmatrix} 7 & 4 & 5 \\ 0 & 3 & 5 \\ 21 & 8 & 4 \end{vmatrix} = 84 + 420 - 315 - 280 = -91 \quad \text{ggT}(26, 91) = 13 \neq 1$$

$$\begin{vmatrix} 7 & 4 & 9 \\ 0 & 3 & 4 \\ 21 & 8 & 2 \end{vmatrix} = 42 + 336 - 567 - 224 = -413 \quad \text{ggT}(26, 413) = 1$$

Die benötigten Inversen von einzelnen Zahlen im Restklassenring werden mit dem erweiterten Euklidischen Algorithmus berechnet, wie im vorigen Beispiel gezeigt.

$$\begin{pmatrix} 7 & 4 & 9 & | & 1 & 0 & 0 \\ 0 & 3 & 4 & | & 0 & 1 & 0 \\ 21 & 8 & 2 & | & 0 & 0 & 1 \end{pmatrix} \downarrow$$

$$\begin{pmatrix} 7 & 4 & 9 & | & 1 & 0 & 0 \\ 0 & 3 & 4 & | & 0 & 1 & 0 \\ 0 & 22 & 1 & | & 23 & 0 & 1 \end{pmatrix} \quad \begin{array}{l} 21 + i \cdot 7 \equiv 0 \pmod{26}; i \cdot 7 \equiv 5 \pmod{26}; i \equiv 5 \cdot 7^{-1} = 5 \cdot 15 \equiv 23 \pmod{26} \\ 8 + 23 \cdot 4 = 100 \equiv 22 \pmod{26} \\ 2 + 23 \cdot 9 = 209 \equiv 1 \pmod{26} \end{array}$$

$$\downarrow$$

$$\begin{pmatrix} 7 & 14 & 0 & | & 2 & 0 & 17 \\ 0 & 19 & 0 & | & 12 & 1 & 22 \\ 0 & 22 & 1 & | & 23 & 0 & 1 \end{pmatrix} \quad \begin{array}{l} 9 + 17 \cdot i = 26 \equiv 0; 4 + 17 \cdot 22 = 378 \equiv 14 \pmod{26} \\ 23 \cdot 17 + 1 = 392 \equiv 2 \pmod{26} \\ 4 + 22 \cdot i = 26 \equiv 0; 3 + 22 \cdot 22 = 487 \equiv 19 \pmod{26} \\ 22 \cdot 23 = 506 \equiv 12 \pmod{26} \end{array}$$

$$\downarrow$$

$$\begin{pmatrix} 7 & 0 & 0 & | & 0 & 2 & 9 \\ 0 & 19 & 0 & | & 12 & 1 & 22 \\ 0 & 0 & 1 & | & 5 & 18 & 7 \end{pmatrix} \quad \begin{array}{l} i \cdot 19 + 14 \equiv 0 \pmod{26}; i \cdot 19 \equiv 12; i \equiv 12 \cdot 19^{-1} = 12 \cdot 11 = 132 \equiv 2 \pmod{26} \\ 2 \cdot 12 + 2 \equiv 0; 2 \cdot 1 \equiv 2; 22 \cdot 2 + 17 = 61 \equiv 9 \pmod{26} \\ i \cdot 19 + 22 \equiv 0; i \cdot 19 \equiv 4; i \equiv 4 \cdot 11 = 44 \equiv 18 \pmod{26} \\ 18 \cdot 19 + 23 = 239 \equiv 5; 18 \cdot 22 + 1 = 397 \equiv 7 \pmod{26} \end{array}$$

$$\downarrow$$

$$\begin{pmatrix} 1 & 0 & 0 & | & 0 & 4 & 5 \\ 0 & 1 & 0 & | & 2 & 11 & 8 \\ 0 & 0 & 1 & | & 5 & 18 & 7 \end{pmatrix} \quad \begin{array}{l} 7 \cdot 7^{-1} = 7 \cdot 15 \equiv 1; 2 \cdot 15 = 30 \equiv 4; 15 \cdot 9 = 135 \equiv 5 \pmod{26} \\ 19 \cdot 19^{-1} = 19 \cdot 11 \equiv 1; 11 \cdot 12 = 132 \equiv 2; 22 \cdot 11 = 242 \equiv 8 \pmod{26} \end{array}$$

Wir müssen wieder das Gleichungssystem $kM = c$ lösen. Mit M^{-1} erhält man $kMM^{-1} = cM^{-1} = k$. Dazu nutzen wir noch die Matrix der Chiffren, die zu unserem M gehören, also (QQX, DUK, XRM).

$$k = \begin{pmatrix} 16 & 3 & 23 \\ 16 & 20 & 17 \\ 23 & 10 & 12 \end{pmatrix} \begin{pmatrix} 0 & 4 & 5 \\ 2 & 11 & 8 \\ 5 & 18 & 7 \end{pmatrix} = \begin{pmatrix} 121 & 511 & 265 \\ 125 & 590 & 359 \\ 80 & 418 & 279 \end{pmatrix} \equiv \begin{pmatrix} 17 & 17 & 5 \\ 21 & 18 & 21 \\ 2 & 2 & 19 \end{pmatrix}$$

Jetzt muss diese Matrix noch invertiert werden, damit man auch Texte entschlüsseln kann.

Dann hat man sowohl k als auch k^{-1} berechnet.

$$\begin{array}{l}
 \left(\begin{array}{ccc|ccc} 17 & 17 & 5 & 1 & 0 & 0 \\ 21 & 18 & 21 & 0 & 1 & 0 \\ 2 & 2 & 19 & 0 & 0 & 1 \end{array} \right) \\
 \Downarrow \\
 \left(\begin{array}{ccc|ccc} 11 & 11 & 0 & 1 & 0 & 23 \\ 1 & 24 & 0 & 0 & 1 & 3 \\ 2 & 2 & 19 & 0 & 0 & 1 \end{array} \right) \\
 \Downarrow \\
 \left(\begin{array}{ccc|ccc} 11 & 11 & 0 & 1 & 0 & 23 \\ 0 & 23 & 0 & 7 & 1 & 8 \\ 0 & 0 & 19 & 14 & 0 & 11 \end{array} \right) \\
 \Downarrow \\
 \left(\begin{array}{ccc|ccc} 1 & 1 & 0 & 19 & 0 & 21 \\ 0 & 1 & 0 & 15 & 17 & 6 \\ 0 & 0 & 1 & 24 & 0 & 17 \end{array} \right) \\
 \Downarrow \\
 \left(\begin{array}{ccc|ccc} 1 & 0 & 0 & 4 & 9 & 15 \\ 0 & 1 & 0 & 15 & 17 & 6 \\ 0 & 0 & 1 & 24 & 0 & 17 \end{array} \right)
 \end{array}$$

$19^{-1} = 11; 5 + i \cdot 19 \equiv 0 \pmod{26}; i \cdot 19 \equiv 21; i = 21 \cdot 11 = 231 \equiv 23 \pmod{26}$
 $2 \cdot 23 + 17 = 63 \equiv 11 \pmod{26}$
 $21 + i \cdot 19 \equiv 0 \pmod{26}; i \cdot 19 \equiv 5; i = 5 \cdot 11 = 55 \equiv 3 \pmod{26}$
 $21 + 3 \cdot 2 = 27 \equiv 1 \pmod{26}; 18 + 3 \cdot 2 = 24$

$11^{-1} = 19; 1 + i \cdot 11 \equiv 0 \pmod{26}; i \cdot 11 \equiv 25 \pmod{26}; i \equiv 25 \cdot 19 = 475 \equiv 7$
 $7 \cdot 11 + 24 = 101 \equiv 23 \pmod{26}; 7 \cdot 23 + 3 = 164 \equiv 8 \pmod{26}$
 $2 + i \cdot 11 \equiv 0 \pmod{26}; i \cdot 11 \equiv 24 \pmod{26}; i = 24 \cdot 19 = 456 \equiv 14 \pmod{26}$
 $14 \cdot 23 + 1 = 323 \equiv 11 \pmod{26}$

$11^{-1} = 19; 19 \cdot 23 = 437 \equiv 21 \pmod{26}$
 $23^{-1} = 17; 17 \cdot 7 = 119 \equiv 15 \pmod{26}; 17 \cdot 8 = 136 \equiv 6 \pmod{26}$
 $19^{-1} = 11; 14 \cdot 11 = 154 \equiv 24 \pmod{26}; 11 \cdot 11 = 121 \equiv 17 \pmod{26}$

$1 + 25 \cdot 1 \equiv 0 \pmod{26}; 19 + 25 \cdot 25 = 394 \equiv 4 \pmod{26};$
 $0 + 25 \cdot 17 = 425 \equiv 9 \pmod{26}; 21 + 6 \cdot 25 = 171 \equiv 15 \pmod{26}$

13. affin-lineare Chiffre

$$E(m, k) = (k m + b) \bmod n$$

$$D(c, k) = (k^{-1} * (c - b)) \bmod n$$

Gesetzt: k ist invertierbare $p \times p$ Matrix über \mathbb{Z}_n

known plaintext attack: $(m_1, c_1) \dots (m_r, c_r)$ bekannt

Zuerst müssen wir das b loswerden, dazu machen wir einfach folgendes:

$$c' = c_i - c_j \equiv [(k m_i + b) \bmod n] - [(k m_j + b) \bmod n] \equiv [(k m_i + b) - (k m_j + b)] \bmod n$$

$$\equiv [k m_i - k m_j] \bmod n \equiv k(m_i - m_j) \bmod n$$

Dazu gehören dann die Nachrichtenvektoren $m_i - m_j$

Wenn diese Nachrichtenvektoren eine Basis bilden (d.h. wir haben p linear unabhängige Vektoren), kann man wie bei der Hill-Chiffre entschlüsseln, indem man zuerst k und dann k^{-1} bildet. Insgesamt braucht man also mindestens $p+1$ Vektoren.

Dann kann man auch b mit $b = c_i - k m_i$ ausrechnen.

chosen plaintext attack: $m_1 \dots m_r$ können gewählt werden.

b erhält man mit $m_0 = \vec{0} \Rightarrow \vec{0} * k + b = b$

Verschlüsselt man dann mit den Vektoren $m_i = (0, \dots, 0, 1, 0, \dots, 0)^T$, die jeweils nur an der i. Stelle eine 1 haben, so erhält man mit $s_i = c_i - b$ die i. Spalte der Matrix k.

$$\begin{pmatrix} k_{11} & k_{12} & k_{13} \\ k_{21} & k_{22} & k_{23} \\ k_{31} & k_{32} & k_{33} \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix}$$

$$\left[\begin{pmatrix} k_{11} & k_{12} & k_{13} \\ k_{21} & k_{22} & k_{23} \\ k_{31} & k_{32} & k_{33} \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} + \begin{pmatrix} b_1 & b_1 & b_1 \\ b_2 & b_2 & b_2 \\ b_3 & b_3 & b_3 \end{pmatrix} \right] - \begin{pmatrix} b_1 & b_1 & b_1 \\ b_2 & b_2 & b_2 \\ b_3 & b_3 & b_3 \end{pmatrix} = \begin{pmatrix} [k_{11} + b_1] - b_1 & [k_{12} + b_1] - b_1 & [k_{13} + b_1] - b_1 \\ [k_{21} + b_2] - b_2 & [k_{22} + b_2] - b_2 & [k_{23} + b_2] - b_2 \\ [k_{31} + b_3] - b_3 & [k_{32} + b_3] - b_3 & [k_{33} + b_3] - b_3 \end{pmatrix}$$

chosen cyphertext attack: $c_1 \dots c_r$ können gewählt werden

Zuerst setzen wir $c_0 = \vec{0} \Rightarrow \vec{0} = k * m_0 + b \Rightarrow b = -k * m_0$, wobei wir k noch nicht kennen

Mit den Vektoren $c_i = (0, \dots, 0, 1, 0, \dots, 0)^T$ bestimmen wir Nachrichtenvektoren $m_i - m_0$.

Es gilt: $c_i = m_i k + b \Rightarrow c_i - c_0 = c_i = (k m_i + b) - (k m_0 + b) = k(m_i - m_0)$

Damit ist $m_i - m_0$ genau die i. Spalte von k^{-1} .

Mit k^{-1} bestimmt man k, und dann b.

Du hast den Geheimtext „OFJDFOHFXOL“ abgehört und weißt, dass er mit einer affin linearen Chiffre der Blocklänge 1 im 27-Zeichen-Alphabet der englischen Sprache codiert wurde (Leerzeichen = 26). Weiterhin ist dir der Anfang „I“ (I gefolgt von Leerzeichen) bekannt. Finde den Schlüssel und dekodiere die Nachricht.

Aufgabe 3/2

$$E(m, k) = (k m + b) \bmod n$$

$$D(c, k) = (k^{-1} * (c - b)) \bmod n$$

Aus der Blocklänge 1 wissen wir, dass gleiche Buchstaben gleich dekodiert werden, und erhalten:

O	F	J	D	F	O	H	F	X	O	L
14	5	9	3	5	14	7	5	23	14	11

I	„“			„“	I		„“		I	
8	26			26	8		26		8	

Aus dem Start erhalten wir die beiden Gleichungen unten, die wir subtrahieren.

$$k^{-1}(8 - b) = 8k^{-1} - bk^{-1} \equiv 14 \bmod 27$$

$$k^{-1}(26 - b) = 26k^{-1} - bk^{-1} \equiv 5 \bmod 27$$

$$18k^{-1} = -9 \equiv 18 \bmod 27$$

Auf den ersten Blick sieht es jetzt so aus, als ob $k^{-1} \equiv 1 \bmod 27$. Das stimmt aber nicht, denn um k^{-1} auszurechnen, müssen wir 18 invertieren. Da $\text{ggT}(18, 27) = 9$ geht das aber in \mathbb{Z}_{27} nicht. Darum können wir mit den gegebenen Daten k^{-1} nicht bestimmen. Mit b verhält es sich ebenso.

$$18 = 2 * 9; \quad 27 = 3 * 9$$

$$18(3i + 1) = 2 * 9(3i + 1) = 2 * 9 * 3 * i + 2 * 9 = 2 * 27 * i + 18 \equiv 18 \bmod 27$$

$$1 + i * \frac{27}{\text{ggT}(18, 27)} = 1 + 3i$$

Siehe 21. $a x \equiv b \bmod l$, Seite 31. Es kommen für k^{-1} also alle Zahlen der Form $3i + 1$ in Frage, von denen es in \mathbb{Z}_{27} 9 gibt:

$k^{-1} = 1, 4, 7, 10, 13, 16, 19, 22, 25$ mit den Inversen $k = 1, 7, 4, 19, 25, 22, 10, 16, 13$.

$$k^{-1}(c - b) \equiv m$$

$$k^{-1}c - k^{-1}b \equiv m$$

$$k^{-1}c - m \equiv k^{-1}b$$

$$k(k^{-1}c - m) \equiv b$$

Dazu gehören b s nach der Formel

Damit ergeben sich die b s zu 21, 6, 18, 3, 15, 0, 12, 24, 9 wenn man für $m = 14$ und $c = 8$ setzt.

Diese müssen nun nacheinander durchprobiert werden. Man setzt sie der Reihe nach in die Entschlüsselungsformel D ein und erhält so:

$(k^{-1}=1, b=21)$	I DY IB RIF	$(k^{-1}=4, b=6)$	I PS IH RIX	$(k^{-1}=7, b=18)$	I AM IN RIO
$(k^{-1}=10, b=3)$	I MG IT RIF	$(k^{-1}=13, b=15)$	I AY IZ RIX	$(k^{-1}=16, b=0)$	I JV IE RIO
$(k^{-1}=19, b=12)$	I VP IK RIF	$(k^{-1}=22, b=24)$	I GJ IQ RIX	$(k^{-1}=25, b=9)$	I SD IW RIO

Für $k^{-1} = 7$ und $b = 18$ ist die Chiffre gebrochen, die Entschlüsselungsfunktion lautet

$$D(c, k^{-1}) \equiv 7(c - 18) \bmod 27 \equiv m.$$

14. Vigenère Chiffre / Beaufort-Chiffre

Aufgabe 3/3

Eine Variante der Vigenère Chiffre ist die selbstentschlüsselnde Beaufort-Chiffre, die wie folgt definiert ist:

$$E_k(m_1, \dots, m_r) = ((k_1 - m_1) \bmod n, \dots, (k_r - m_r) \bmod n)$$

Für einen Schlüssel $k = (k_1, \dots, k_r) \in K$ und eine Nachricht $(m_1, \dots, m_r) \in \mathbb{Z}_n^r$. Die Chiffrier- und Dechiffrierfunktion sind identisch, also ihre eigene Inverse.

Das Kryptogramm CJUJ LAFY TCTU LTKG wurde für $r = 4$ und $n = 26$ mit dem Schlüssel „EASY“ verschlüsselt. Finde den Klartext.

Der Schlüssel „EASY“ entspricht dem Vektor $(4, 0, 18, 24)$. Wenden wir nun die Chiffrier- und Dechiffrierfunktion $E_k(m_1, \dots, m_4)$ an

$$\begin{array}{llll} E_k(\text{CJUJ}) & = E_k(2, 9, 20, 9) & = (4-2, 0-9, 18-20, 24-9) & = (2, 17, 24, 15) & = \text{CRYP} \\ E_k(\text{LAFY}) & = E_k(11, 0, 5, 24) & = (4-11, 0-0, 18-5, 24-24) & = (19, 0, 13, 0) & = \text{TANA} \\ E_k(\text{TCTU}) & = E_k(19, 2, 19, 20) & = (4-19, 0-2, 18-19, 24-20) & = (11, 24, 25, 4) & = \text{LYZE} \\ E_k(\text{LTKG}) & = E_k(11, 19, 10, 6) & = (4-11, 0-19, 18-10, 24-6) & = (19, 7, 8, 18) & = \text{THIS} \end{array}$$

Der Klartext lautet also „CRYPTANALIZE THIS“.

15. DES Chiffre

Blockchiffre

Data Encryption Standard ist eine Blockchiffre der Blockgröße 64 Bit mit einer effektiven Schlüssellänge von 56 Bit (von 64 Bit Eingabeschlüssel werden 56 genutzt, die anderen 8 Bit fallen weg oder werden als Paritätsbit genutzt).

DES ist auf Grund der kurzen Schlüssel stark durch „Brute-Force“-Angriffe immer schneller werdender Computer bedroht.

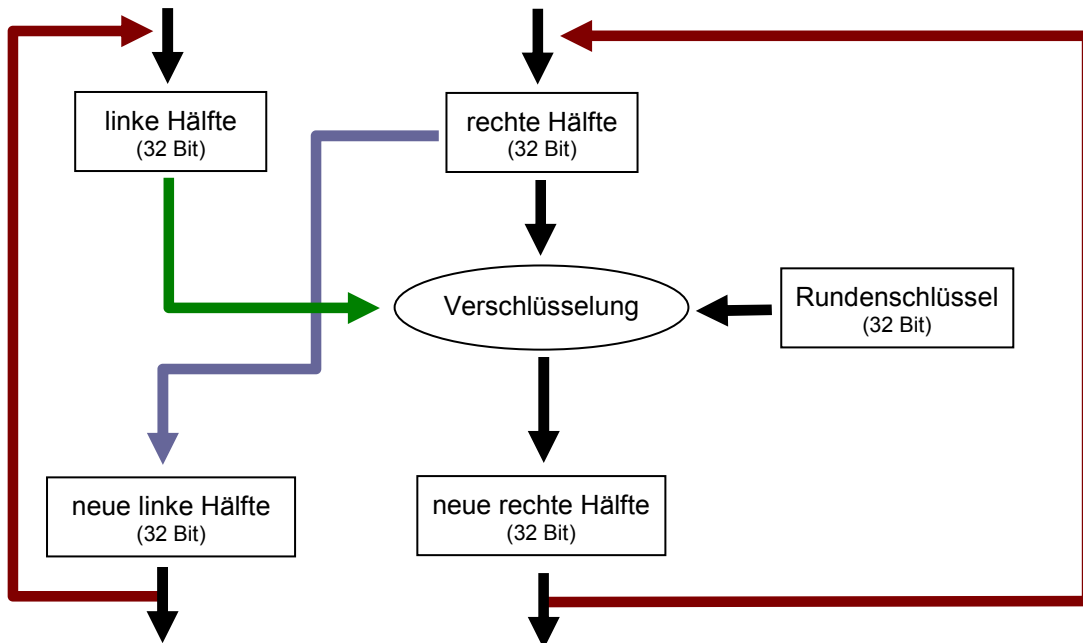
Es basiert auf einer Kombination von Substitution und Permutation.

Der 56-Bit-Schlüssel wird intern als 64-Bit-Notation gehandhabt, wobei die Bits 0 bis 6 eines jeden Bytes sieben Bit des DES-Schlüssels wiedergeben. Das oberste Bit bleibt dabei ungenutzt.

Zu Beginn der Verschlüsselung werden die Bits des Eingangsblocks nach der Tabelle unten permutiert, also durcheinander gewürfelt, nach der Permutation π .

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

Nach der Eingangspermutation wird der so entstandene 64-Bit-Block in zwei 32-Bit-Hälften aufgeteilt. Jetzt findet die eigentliche Verschlüsselung in 16 Runden statt. In diesen Runden wird die rechte Hälfte des Klartextes mit dem Schlüssel sowie der linken 32-Bit-Klartext-Hälfte verknüpft und dadurch verschlüsselt. Die Verschlüsselungsfunktion liefert einen neuen 32-Bit-Wert, der in der nächsten Runde als rechte Eingabe dient. Die linke neue Eingabe für die nächste Runde entsteht aus der rechten Klartexthälfte aus der vorherigen Runde.



In jeder DES-Runde wird die rechte 32-Bit-Hälfte mit Hilfe einer Expansionspermutation auf 48 Bit verbreitert. Parallel dazu wird aus dem 56-Bit-Schlüssel ein 48-Bit-Rundenschlüssel generiert. Die Expansionspermutation, hat ihren Namen dadurch, da sie sowohl die Reihenfolge der Bits ändert als auch manche Bits wiederholt.

Die Erweiterungspermutation hat im wesentlichen zwei Aufgaben:

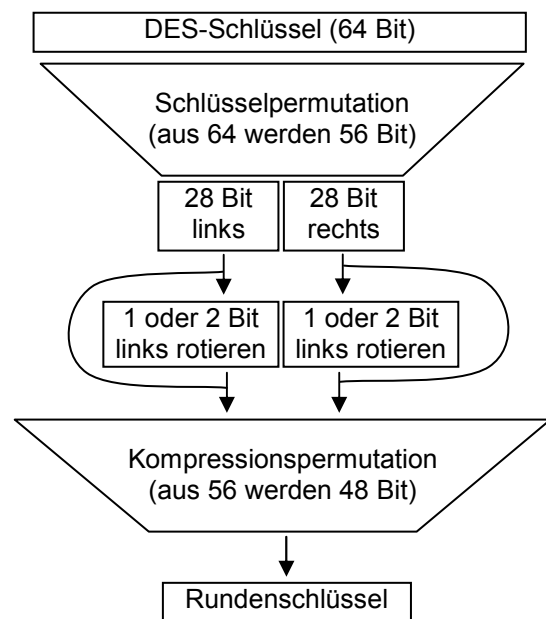
- 1) Die Vergrößerung der rechten Hälfte, damit sie mit dem Schlüssel XOR verknüpft werden kann.
- 2) Sie liefert ein längeres Resultat, dass mit Hilfe der Substitutionsoperation komprimiert werden kann.

Eingangsbit	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Ausgangsbit	2, 48	3	4	5, 7	6, 8	9	10	11, 13	12, 14	15	16	17, 19	18, 20	21	22	23, 25
Eingangsbit	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
Ausgangsbit	24, 26	27	28	29, 31	30, 32	33	34	35, 37	36, 38	39	40	41, 43	42, 44	45	46	1, 47

Damit wird der 32-Bit-Eingabeblock in 8 Teile zu je 4 Bit unterteilt. Bei jedem 4-Bit-Eingabeblock stellt das erste und vierte Bit jeweils zwei Bit des Ausgabeblocks dar. Das zweite und dritte Bit des 4-Bit-Eingabeblocks stellt jeweils ein Ausgabebit dar. Somit wird aus der 4-Bit-Eingabe eine 6-Bit-Ausgabe.

Dadurch erzeugt jeder Eingabeblock einen unterschiedlichen Ausgabeblock. Der gesamte Ausgabeblock ist somit 48 Bit groß.

Bei der Rundenschlüsselerzeugung läuft folgendes ab. Nachdem am Anfang der 64-Bit-Schlüssel auf 56 Bit reduziert (Schlüsselpermutation, jedes achte Bit wird ignoriert) wurde, wird der 56-Bit-Schlüssel in zwei 28-Bit-Hälften zerlegt. Diese werden in Abhängigkeit vom Rundenindex (1 bis 16) um zwei oder ein Bit verschoben. Danach werden 48 Bit aus 56 Bit ausgewählt. Dabei wird sowohl die Reihenfolge, als auch eine Bit-Teilmenge ausgewählt. Der Vorgang trägt den Namen Kompressionspermutation

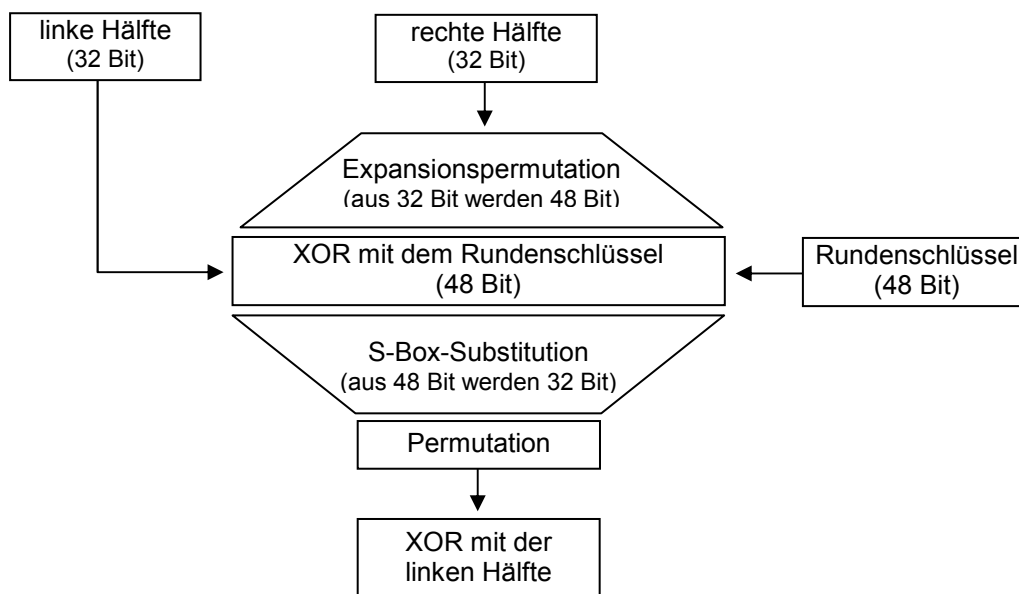


Anzahl von Bitrotationen:

Runde	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Bitanzahl	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

Kompressionspermutation:

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



Jetzt nachdem der Rundenschlüssel erzeugt ist und die Expansionspermutation durchgeführt wurde, werden die beiden 48-Bit-Werte mittels XOR verknüpft.

Danach wird mit dem 48-Bit, mit Hilfe von acht Substitutionsboxen (S-Boxen, Eine Art Abbildungstabelle), eine Substitution durchgeführt. Jede S-Box besteht aus einer 6-Bit Eingabe und liefert eine 4-Bit-Ausgabe. Somit wird aus der 48-Bit-Eingabe eine 32-Bit-Ausgabe. Die S-Boxen sind der Knackpunkt des DES-Verschlüsselungsverfahrens, die Sicherheit hängt von ihrer Gestaltung ab. Die S-Boxen sind nichtlinear. Alle anderen Operationen sind linear. Eine Veränderung der S-Boxen bedeutet, dass der Algorithmus mit sehr großer Wahrscheinlichkeit unsicher gegen Angriffe wird.

Auf der nächsten Seite sind die 8 Substitutionsboxen dargestellt. Die sechs Eingabebits legen fest, welcher Zeile und Spalte die Ausgabe zu entnehmen ist. Die Bits b_1 und b_6 ergeben eine Zwei-Bit-Zahl zwischen 0 und 3, die eine Zeile auswählen. Die Bits b_2 bis b_5 ergeben eine Vier-Bit-Zahl zwischen 0 und 15. Sie legen eine Spalte fest.

Beispiel:

Die sechste S-Box (Bit 31 bis Bit 36 der Eingabe der vorigen XOR-Verknüpfung) erhalte die Eingabe 51 (Binär ist dies 110011). Das erste und letzte Bit ergeben 3 (Binär ist dies 11). Daraus folgt, dass die Zeile 3 der Teiltabelle „S-Box 6“ zu wählen ist. Die mittleren Bits ergeben zusammen 9 (Binär ist dies 1001). Daraus folgt, dass die Spalte 9 der Teiltabelle „S-Box 6“ zu wählen ist. Der Tabelleneintrag lautet 14 (Binär ist dies 1110). Hieraus ergibt sich die Substitution von 51 (110011) nach 14 (1110).

S-Boxen:

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

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

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

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

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

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

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

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

P-Boxen:

Eingangsbit	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Ausgangsbit	9	17	23	31	13	28	2	18	24	16	0	6	26	20	10	1

Eingangsbit	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
Ausgangsbit	8	14	25	3	4	29	11	19	32	12	22	7	5	27	15	21

Die so gewonnen 32 Bit werden ein weiteres mal untereinander vertauscht (Permutation oder auch P-Box-Permutation genannt), um anschließend mit der linken 32-Bit-Klartext-Hälfte per XOR verknüpft zu werden. Jetzt ist eine Runde beendet. Die so gewonnen Ausgabe der rechten 32-Bit-Hälfte wird als Eingabe für die nächste Runde verwendet. Die linke neue Eingabe für die nächste Runde, ist die vor Verschlüsselung verwendete rechte Klartexthälfte. Nach der letzten Runde werden die beiden verschlüsselten 32-Bit-Hälften des Klartextes wieder zu einem 64-Bit-Wert zusammengefügt. Danach wird das Inverse (End-Permutation) der Start-Permutation durchgeführt, wobei die linke und rechte Hälfte nach der letzten Runde nicht mehr vertauscht werden.

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

Die Entschlüsselung verläuft im Wesentlichen auf die gleiche Weise. Es müssen nur die Rundenschlüssel in umgekehrter Reihenfolge (Bei Verschlüsselung Schlüssel K_1 in Runde 1, K_2 in Runde 2, Bei der Entschlüsselung der Schlüssel K_{16} in Runde 1, usw.) angewendet werden. Der Schlüssel wurde bei Verschlüsselung immer nach links verschoben, bei Entschlüsselung wird nach rechts verschoben. Und zwar um 0,1,2,2,2,2,2,2,1,2,2,2,2,2,1 Bit-Positionen in der jeweiligen Runde. Alle anderen Operationen werden in der gleichen Weise durchgeführt.

Aufgabe 7/2 a) Zeige am Beispiel einer der S-Boxen von DES, dass diese nichtlinear ist.

Um zu zeigen, dass eine S-Box S_i nichtlinear ist, muss gezeigt werden, dass für mindestens ein Paar (x, y) von Eingaben gilt: $S_i(x \oplus y) \neq S_i(x) \oplus S_i(y)$.

Wir wählen dafür die sechsstelligen Binärzahlen $x = 000000$ und $y = 000001$ und berechnen für die S-Box S_1 :

$$S_1(000000 \oplus 000001) = S_1(000001) = 0 \text{ (S-Box 1, 1. Zeile, 0-te Spalte)}$$

$$S_1(000000) = 14 = 1110 \Rightarrow S_1(000000) \oplus S_1(000001) = 14 = 1110 \neq S_1(000000 \oplus 000001)$$

Damit sind die S-Boxen nicht linear.

b) Zeige, dass das Vertauschen der linken und der rechten Hälfte der 64-Bit-Blöcke nach jeder Runde eine lineare Operation ist.

Sei f die Funktion zum Vertauschen der ersten 32 Bit mit den letzten 32 Bit einer 64-Bit-Zahl. Es muss gezeigt werden, dass gilt:

$$\begin{aligned} f(x \oplus y) &= f(x) \oplus f(y) : x = (x_{63} \dots x_0), y = (y_{63} \dots y_0) \\ f(x \oplus y) &= f(x_{63} \oplus y_{63} \dots x_0 \oplus y_0) = (x_{31} \oplus y_{31} \dots x_0 \oplus y_0, x_{63} \oplus y_{63} \dots x_{32} \oplus y_{32}) \\ f(x) \oplus f(y) &= f(x_{63} \dots x_0) \oplus f(y_{63} \dots y_0) = (x_{31} \dots x_0, x_{63} \dots x_{32}) \oplus (y_{31} \dots y_0, y_{63} \dots y_{32}) \\ &= (x_{31} \oplus y_{31} \dots x_0 \oplus y_0, x_{63} \oplus y_{63} \dots x_{32} \oplus y_{32}) = f(x \oplus y) \end{aligned}$$

c) Die S-Boxen sind das einzige nichtlineare Element von DES, worauf die Sicherheit basiert. Warum wäre DES mit linearen S-Boxen unsicher? Beschreiben Sie einen Angriff gegen ein derartiges lineares DES.

DES mit linearen S-Boxen wäre unsicher, weil dann das ganze DES eine lineare Funktion wäre. Es würden damit die gleichen Angriffe möglich, wie gegen 12. Hill-Chiffre, Seite 8.

16. RSA-Chiffre asymmetrische Chiffre, Public-Key-Verfahren

Beim **Empfänger** geht folgendes vor sich:

- 1) es werden zwei große Primzahlen $p \neq q$ erzeugt, z.B. mit 29. heuristischer Primzahltest von Solovay und Strassen, Seite 35 $O(\text{poly}(\log n))$
- 2) es wird das Produkte $n = p * q$ berechnet $O((\log p * \log q)^2)$
 $= O(\log^2 n)$
- 3) es wird die gerade Zahl $\varphi(n) = (p - 1)(q - 1)$ erzeugt
 $\varphi(n)$ ist die Anzahl der Zahlen $x < n$ mit $\text{ggT}(x, n) = 1$ $O(\log^2 n)$
- 4) es wird eine ungerade, natürliche Zahl e erzeugt, mit
 $1 < e < \varphi(n)$ und $\text{ggT}(e, \varphi(n)) = 1$ $O(\log^3 n)$
- 5) dann wird mit $d * e \equiv 1 \pmod{\varphi(n)} \Leftrightarrow d \equiv e^{-1} \pmod{\varphi(n)}$ das Inverse zu e in $\mathbb{Z}_{\varphi(n)}$ erzeugt (siehe erweiterter Euklidischer Algorithmus, Seite 3) $O(\log^3 n)$
- 6) n und e werden veröffentlicht, $k = (n, e)$ ist der öffentliche Schlüssel, $k' = (\varphi(n), d)$ der private

der **Sender** verschlüsselt mit
 $E(m, k) = m^e \pmod{n} = c, m \in \mathbb{Z}_n$

der **Empfänger** entschlüsselt mit
 $D(c, k') = c^d \pmod{n} = m$

Mit 27. schnelles Potenzieren, Seite 34, wird hier $O(\log^2 m * \log e)$ bzw. $O(\log^2 c * \log d)$ erreicht.

Nach 25. Satz von Euler (Seite 33) gilt $a^{\varphi(n)} \equiv 1 \pmod{n}$ und mit 26. kleiner Satz von Fermat (Seite 33) gilt $a^{n-1} \equiv 1 \pmod{n}$.

Daraus folgt $a^{d+i\varphi(n)} \equiv a^d a^{i\varphi(n)} \equiv a^d (a^{\varphi(n)})^i \equiv a^d 1^i \equiv a^d \pmod{n}$

Um zu beweisen, dass $a^x \equiv a^y \pmod{n} \quad \forall a \in \mathbb{Z}_n^*$, genügt es zu beweisen, dass $x \equiv y \pmod{\varphi(n)}$

Für das RSA-Verfahren gilt $D(E(m, k), k') \equiv D(m^e \pmod{n}, k') \equiv (m^e)^d \equiv m^{ed} \equiv m^{1 \pmod{\varphi(n)}} \equiv m \pmod{\varphi(n)}$.

Attacken:

Unterscheiden sich p und q nur um den kleinen Wert a , so gilt

$$(p+q)^2 = (p-q)^2 + 4pq = (p-q)^2 + 4n = a^2 + 4n$$

Jetzt kann das Gleichungssystem rechts gelöst werden:

$$(p+q)^2 = a^2 + 4n$$

$$|p-q| = a : p > q \Rightarrow p = q + a$$

$$a^2 + 4n = (2q+a)^2 = 4q^2 + 4aq + a^2$$

$$0 = 4q^2 + 4aq - 4n = q^2 + aq - n$$

$$q = -\frac{a}{2} + \sqrt{\frac{a^2}{4} + n}$$

Sind p und q klein, so kann n einfacher mit brute-force durch probieren faktorisiert werden

Geht die gleiche Nachricht an e Adressaten, und wird dem selben e aber paarweise teilerfremden n_i , also mit dem öffentlichen Schlüssel (n_i, e) verschlüsselt, so kann d mit 31. chinesischer Restsatz, Seite 36, berechnet werden.

Ist d zu klein, so kann es erraten werden

Aufgabe 4/2 (broadcast attack) Eine Nachricht m wird mittels RSA verschlüsselt und an drei Empfänger mit den öffentlichen Schlüsseln $(7, 3)$, $(11, 3)$ und $(13, 3)$ gesendet. Bestimmen Sie aus den Chiffre $c_1 \equiv 6 \pmod{7}$, $c_2 \equiv 7 \pmod{11}$ und $c_3 \equiv 8 \pmod{13}$ die ursprüngliche Nachricht m .

Die Nachricht wurde zu $c_i \equiv m^3 \pmod{n_i}$ verschlüsselt, es gilt also logischerweise auch $m^3 \equiv c_i \pmod{n_i}$.

Wir nutzen 31. chinesischer Restsatz, Seite 36, da die n_i s paarweise teilerfremd sind.

$$n = 7 * 11 * 13 = 143 * 7 = 1001$$

$$m^3 \equiv 6 \pmod{7}$$

$$n_1 = 11 * 13 = 143 \equiv 3 \pmod{7}$$

$$n_1^{-1} \equiv 3^{-1} \pmod{7} \equiv 5 \pmod{7}$$

$$m^3 \equiv 7 \pmod{11}$$

$$n_2 = 7 * 13 = 91 \equiv 3 \pmod{11}$$

$$n_2^{-1} \equiv 3^{-1} \pmod{11} \equiv 4 \pmod{11}$$

$$m^3 \equiv 8 \pmod{13}$$

$$n_3 = 7 * 11 = 77 \equiv 12 \pmod{13}$$

$$n_3^{-1} \equiv 12^{-1} \pmod{13} \equiv 12 \pmod{13}$$

$$m^3 \equiv 5 * 143 * 6 + 4 * 91 * 7 + 12 * 77 * 8 \pmod{1001} \equiv 4290 + 2548 + 7392 \equiv 14230 \equiv 216 \pmod{1001}.$$

$$\sqrt[3]{m^3} = m \quad \begin{array}{l} 5^3 = 125 < 216 < 10^3 = 1000 \\ 7^3 = 49 * 7 = 343 > 216 \\ 6^3 = 36 * 6 = 216 \end{array}$$

nach 28. schnelles Wurzelziehen, Seite 34

Damit ist $m = 6$ bestimmt.

Aufgabe 4/4 (Zyklus-attacke) Sei (n, e) öffentlicher RSA-Schlüssel und $c = m^e \pmod{n}$ das Chiffre zu einem Klartext m . Zeige, dass eine natürliche Zahl s mit $m^{e^s} \equiv m \pmod{n}$ existiert. Kann diese Aussage einem Angreifer helfen?

Es gilt $m \equiv m^1 \equiv m^{e^s}$

Wir rechnen mit 25. Satz von Euler, Seite 33: $\text{ggT}(a, p) = 1 \Rightarrow a^{\phi(p)} \equiv 1 \pmod{p}$.

Somit gilt $m^{\phi(n)} \equiv 1 \pmod{n} \Rightarrow m^{\phi(n)+1} \equiv m \pmod{n}$.

Damit muss gelten $e^s \equiv \phi(n)+1 \equiv 1 \pmod{\phi(n)}$.

Hierauf wird wiederum der Satz von Euler angewendet $1 \pmod{\phi(n)} \equiv e^{\phi(\phi(n))}$.

$s \in \mathbb{N}$ existiert bei $e \in \mathbb{Z}_{\phi(n)}^*$ (muss immer gelten, sonst wäre e nicht invertierbar unter $\phi(n)$) und

$$s = \phi(\phi(n)) \cdot \left(x \in \mathbb{Z}_{\phi(n)}^* \Rightarrow \text{ggT}(x, \phi(n)) = 1 \right)$$

Beispiel Für $p = 3$ und $q = 5$ gilt $n = 15$ und $\phi(n) = (p-1)(q-1) = 2 * 4 = 8$.
 e wird zu 3 gewählt. Hier wäre $s = \phi(\phi(n)) = \phi(8) = 4$

Damit ergibt sich $m^{3^s} \equiv m^{3^4} \equiv m^{81} \equiv m \pmod{15}$.

Ein Angreifer kann die möglichen Werten von s durch probieren bestimmen, indem einfach alle Werte von $c^{e^i} \forall i$ bestimmt, bis $c^{e^i} \equiv c \pmod{n}$ gilt. Dann ist $s = i$ und $c^{e^s} \equiv m^{e^{s+1}} \equiv m^e \pmod{n}$ und somit $m^{e^s} \equiv m \pmod{n}$.

Der Angreifer wird demnach nur Erfolg haben, wenn der Zyklus s der Chiffrekatte klein ist.

Aufgabe 5/2 (common modulus attack) Ein Klartext $m \in \mathbb{Z}_n$ wird zweimal verschlüsselt, einmal mit dem öffentlichen Schlüssel (n, e_1) und einmal mit (n, e_2) . Kann, wenn $\text{ggT}(e_1, e_2) = 1$ ist, ein Angreifer aus den Chiffraten $c_1 = m^{e_1} \bmod n$ und $c_2 = m^{e_2} \bmod n$ den Klartext bestimmen? Hinweis: $e_1, e_2 \in \mathbb{N} : \text{ggT}(e_1, e_2) = 1 \Rightarrow \exists x, y \in \mathbb{Z} : xe_1 + ye_2 = 1$.

Kennt man x und y , so folgt: $c_1^x * c_2^y \equiv m^{e_1*x} * m^{e_2*y} \equiv m^{e_1*x+e_2*y} \equiv m^1 \equiv m \bmod n$
 x und y kann man leicht durch „Rückwärtsverfolgen“ der Koeffizienten bei erweiterter Euklidischer Algorithmus, Seite 3 erhalten.

Beispiel: $e_1 = 5$ und $e_2 = 3$

$$5 = 1*3 + 2$$

$$3 = 1*2 + 1$$

$$2 = 2*1 + 0$$

also $1 = 3 - 1*2$

$$= e_2 - 1(e_1 - 1*3)$$

$$= e_2 - 1e_1 + 1e_2$$

$$= -e_1 + 2e_2$$

Damit ist $x = -1$ und $y = 2$.

Aufgabe 4/1 Beim RSA-Verfahren verwendet jemand als öffentliche Schlüssel $(n, e = 1)$, $(n, e = 2)$ und $(n, e = 3)$, wobei n ein Produkt zweier Primzahlen $p, q \geq 3$ ist. Welche Probleme treten auf?

Da gilt $E(m, k) = m^c \bmod n = c$ wird bei $e = 1$ überhaupt nicht verschlüsselt.

Es gilt $\varphi(n) = (p-1)(q-1)$, und da p und q Primzahlen, also ungerade sind, ist $\varphi(n)$ gerade. Damit wird bei $e = 2$ aber $\text{ggT}(e, \varphi(n)) = 1$ verletzt, und $d \equiv e^{-1} \bmod \varphi(n)$ existiert nicht, es könnte nicht entschlüsselt werden.

Bei $e = 3$ muss auf alle Fälle trotzdem $\text{ggT}(e, \varphi(n)) = 1$, also $\text{ggT}(3, (p-1)(q-1)) = 1$ gelten.

* Da $\text{ggT}(3, 0) = 3$, darf weder $p \equiv 1 \bmod 3$ noch $q \equiv 1 \bmod 3$ sein, denn $\text{ggT}(3, (0 \bmod 3)(x-1)) = 3$ weil $(1 \bmod 3) - 1 \equiv 0 \bmod 3$.

Gilt $p = 3$ und $q = 3$, so folgt $\text{ggT}(3, (p-1)(q-1)) = \text{ggT}(3, 4) = 1$ und alles ist gut, aber $p \neq q$ ist verletzt und damit ist RSA nicht richtig angewendet.

Setzen wir $p = 3$ und $q > 3$, dann gibt es drei Möglichkeiten für q : Bei $q \equiv 0 \bmod 3$ ist q keine Primzahl (da durch 3 teilbar).

$q \equiv 1 \bmod 3$ geht schon wegen der Überlegung * nicht.

Einzig $q \equiv 2 \bmod 3$ sichert mit $\text{ggT}(3, (p-1)(q-1)) = \text{ggT}(3, 2*1) = 1$ die Existenz von e^{-1} .

Aufgabe 5/1 Zeige, dass man mit der RSA-Dechiffrierfunktion $D(c, k') = c^d \bmod n = m$ tatsächlich den Klartext m erhält.

Da $d \equiv e^{-1} \bmod \varphi(n)$ gilt, folgt $d*e \equiv 1 \bmod \varphi(n)$. Es existiert demnach ein $\omega \in \mathbb{Z} : ed = 1 + \omega\varphi(n)$.

Für alle p mit $\text{ggT}(p, m) = 1$ gilt $m^{p-1} \equiv 1 \bmod p$ nach 26. kleiner Satz von Fermat, Seite 33.

Also folgt $c^d \bmod n \equiv (m^e)^d = m^{ed} \equiv m^{1+\omega\varphi(n)} \equiv m * m^{\omega\varphi(n)} \equiv m * m^{\omega(q-1)(p-1)} \equiv m * (m^{\omega(q-1)})^{p-1} \equiv m * 1 \equiv m \bmod n$, da p eine Primzahl ist.

Aufgabe 6/1 Zur Beschleunigung der Dechiffrierung bei einem RSA-System mit den Parametern $(n = p * q, e)$ führt der Empfänger des Chiffrats $c \in \mathbb{Z}_n^*$ folgendes aus:

Er berechnet $m_p \equiv c^{d \bmod (p-1)} \pmod p$ und $m_q \equiv c^{d \bmod (q-1)} \pmod q$ und berechnet mit 31. chinesischer Restsatz, Seite 36 ein $m \in \mathbb{Z}_n^*$ mit $m \equiv m_p \pmod p$ und $m \equiv m_q \pmod q$. Ist m die gesendete Nachricht?

Ersteinmal wissen wir, dass nach RSA-Chiffre gilt $m \in \mathbb{Z}_n^*$ mit $m \equiv c^d \pmod n \equiv (m^e)^d \pmod{(p * q)}$

Wir haben $m_p \equiv c^{d \bmod (p-1)} \pmod p$ und $m_q \equiv c^{d \bmod (q-1)} \pmod q$.

Wir wissen, dass p und q beides Primzahlen sind. Damit ist $\varphi(p) = p - 1$ und $\varphi(q) = q - 1$.

Machen wir uns also den 25. Satz von Euler, Seite 33, $\text{ggT}(a, p) = 1 \Rightarrow a^{\varphi(p)} \equiv 1 \pmod p$ zunutze:

$$a^{\varphi(p)} \equiv 1 \pmod p \Rightarrow a^{d \bmod \varphi(p)} \equiv a^d \pmod p$$

Damit erhalten wir:

$$m \equiv c^d \equiv m_p \pmod p \equiv c^{d \bmod (p-1)} \pmod p \text{ und } m \equiv c^d \equiv m_q \pmod q \equiv c^{d \bmod (q-1)} \pmod q.$$

Damit erhält man mit dem Gleichungssystem $m \equiv m_p \pmod p$ und $m \equiv m_q \pmod q$ tatsächlich das richtige Ergebnis.

17. Merkle-Hellman Knapsack Kryptosystem

Basiert auf dem Subset-Sum-Problem, dem Knapsack-Problem.

Gegeben: die Folge (s_1, s_2, \dots, s_n) natürlicher Zahlen und die natürliche Zahl T

Gesucht: $\{0,1\}^n$ -Vektor $x = (x_1, x_2, \dots, x_n)$ mit $\sum_{i=1}^n x_i s_i = T$, falls dieser existiert

Dies ist ein *nicht* in polynomieller Zeit lösbares, NP-vollständiges Problem

Betrachten superwachsende Folgen mit $s_i > \sum_{j=1}^{i-1} s_j$. z.B. die Zweierpotenzen fürs Binärsystem.

Dann ist das Problem sehr einfach lösbar, nach folgendem Algorithmus:

```
for i = n downto 1 do
  begin
    if T ≥ si then
      begin
        T = T - si
        xi = 1
      end
    else xi = 0
  end;

if T = 0 then „x=(x1, ..., xn) ist Lösung“
else „Es gibt keine Lösung“
```

Eine Nachricht $m = (m_1, m_2, \dots, m_n) \in \{0,1\}^n$ wird mit einer öffentlich bekannten superwachsenden Folge $s = (s_1, s_2, \dots, s_n)$ verschlüsselt zu:

$$E(m, s) = \sum_{i=1}^n m_i s_i$$

Dies kann leicht entschlüsselt werden, auch von Angreifern. Daher verwendet man eine öffentliche Transformation $t = (t_1, t_2, \dots, t_n)$ der geheimen, superwachsenden Folge s , welche wir mit der Primzahl $p > \sum_{\forall i} s_i$ erhalten.

$t_i = x * s_i \bmod p$, $x \in \mathbb{Z}_p$ ist geheim

$$E(m, t) = \sum_{i=1}^n m_i t_i \bmod p = c$$

$$D(c, (p, x, s)) = (m_1, m_2, \dots, m_n)$$

Wobei $T \equiv x^{-1}c \equiv \sum_{i=1}^n m_i s_i \bmod p$ ist. Da $T < p$ ist der Vektor (m_1, m_2, \dots, m_n) leicht als Lösung des Subset Sum Problems für s und T als $x^{-1} * c$ zu ermitteln.

Beispiel:

Empfänger:

Die superwachsende Folge s sei $(1, 2, 4, 8, 16, 32)$. Als öffentliche Primzahl $p > \sum_{\forall i} s_i = 63$ wird $p = 67$ gewählt. Die geheime Zahl $x = 8$ wird festgelegt, ihr ebenfalls geheimes Inverses $x^{-1} \bmod 67 = 42$ wird bestimmt (mittels erweiterter Euklidischer Algorithmus, Seite 3). Mittels x wird die Folge s durch Multiplikation Modulo 67 transformiert zu $t = (8, 16, 32, 64, 61, 55)$ und veröffentlicht.

Sender:

Die Nachricht $m = (0, 1, 0, 1, 1, 1)$ soll übermittelt werden. Sie wird mit der bekannten Folge t und der bekannten Primzahl p kodiert, gemäß

$$c = \sum_{i=1}^{n=6} m_i t_i \bmod p = (0 \cdot 8 + 1 \cdot 16 + 0 \cdot 32 + 1 \cdot 64 + 1 \cdot 61 + 1 \cdot 53) \equiv 196 \bmod 67 \equiv 62 \bmod 67$$

Empfänger:

$$\text{Es gilt ja } c = \sum_{i=1}^{n=6} m_i t_i \bmod p = \sum_{i=1}^{n=6} m_i s_i x \bmod p .$$

$$\text{man erhält } x^{-1} c = x^{-1} \sum_{i=1}^{n=6} m_i s_i x \bmod p = \sum_{i=1}^{n=6} m_i s_i \bmod p = 42 \cdot 62 \equiv 42 \cdot -5 \equiv -210 \equiv -9 \equiv 58 \bmod 67$$

Nun kann man das Knapsack-Problem lösen:

$$58 = 0 \cdot 1 + 1 \cdot 2 + 0 \cdot 4 + 1 \cdot 8 + 1 \cdot 16 + 1 \cdot 32 \Rightarrow \{0, 1, 0, 1, 1, 1\} = m$$

Sicherheit

Das naive Verfahren zum Knacken des Codes wäre, einfach alle möglichen x auszuprobieren, und zu versuchen, superwachsende Folgen aus t zu erhalten. Unter Umständen erhält man mehrere. Da die Eingabegröße $\sum_{\forall i} \log s_i = \log \prod_{\forall i} s_i \geq \log p$ ist, wäre dies aber in $O(p)$ also exponentiellem Aufwand.

In polynomiellen Aufwand ist das System jedoch durch den LLL-Algorithmus von Lenstra, Lenstra und Lovánu zu knacken, und daher unsicher.

18. El Gamal Kryptosystem, El Gamal Signierschema

Basis: Diskretes Logarithmus Problem mit der Schwierigkeit $a^x \equiv b \pmod p$ (p ist Primzahl)

Empfänger:

Primzahl p wird gewählt, so dass das DLP schwer für p zu lösen ist.

a wird als primitives, erzeugendes Element in \mathbb{Z}_p^* gewählt $\Rightarrow \mathbb{Z}_p^* = \{a^0, a^1, \dots, a^{\varphi(p)}\}$

Ein x wird sich ausgedacht.

b wird vom Sender aus a , x und p berechnet, als Ergebnis von $b \equiv a^x \pmod p$.

Der öffentliche Schlüssel ist (p, a, b) , der geheime Schlüssel ist (x) .

Sender:

Es wird zufällig ein $k \in \mathbb{Z}_{p-1} \setminus \{0\}$ gewählt.

$$c = E(m, k) \equiv (a^k \pmod p, m * b^k \pmod p)$$

Der **Empfänger** macht $m = D(m, x) = D((y_1, y_2), x) \equiv y_2 * (y_1^x)^{-1} \pmod p$

Korrektheit: $m \equiv D(E(m, k), x) \equiv D((a^k \pmod p, m * b^k \pmod p), x) \equiv m * b^k * ((a^k)^x)^{-1} \pmod p$

$$\equiv m * b^k * (a^{xk})^{-1} \pmod p \equiv m * b^k * (b^k)^{-1} \pmod p \equiv m \pmod p$$

Angriff: wählt der Sender bei zwei verschiedenen Sendungen das gleiche k , so wird das Entschlüsseln vereinfacht: $(a^k, m_1 b^k), (a^k, m_2 b^k)$

Nachteil des Systems: Chiffre ist doppelt so lang wie Quelltext

hausgemachtes Beispiel:

Der **Empfänger** wählt sich die Primzahl $p = 31$. Dazu findet er die primitive, erzeugende Zahl $a = 12$. Er denkt sich ein $x = 9$ aus.

Damit wird mit 27. schnelles Potenzieren, Seite 34 ein $b \equiv a^x \pmod p$ berechnet:

$x = 9 = 1001_b$	1 $12^1 = 12 \equiv 12 \pmod{31}$	$b = 12^9$
	0 $12^2 = 12 * 12 = 144 \equiv 20 \pmod{31}$	$= 12^1 * 12^8$
	0 $12^4 = 20 * 20 = 400 \equiv 28 \pmod{31}$	$\equiv 12 * 9 \pmod{31}$
	1 $12^8 = 28 * 28 = 784 \equiv 9 \pmod{31}$	$\equiv 108 \pmod{31} \equiv 15 \pmod{31}$

Der öffentliche Schlüssel ist somit $(p, a, b) = (31, 12, 15)$.

Jetzt wird der Sender tätig: er bestimmt zuerst einmal ein k per Zufall, wir nehmen $k = 14$.

Damit berechnet er $y_1 = a^k \pmod p$, wieder mit dem schnellen Potenzieren.

$k = 14 = 1110_b$	0 $12^1 = 12 \equiv 12 \pmod{31}$	$y_1 = 12^{14}$
	1 $12^2 = 12 * 12 = 144 \equiv 20 \pmod{31}$	$= 12^2 * 12^4 * 12^8$
	1 $12^4 = 20 * 20 = 400 \equiv 28 \pmod{31}$	$\equiv 20 * 28 * 9 \pmod{31}$
	1 $12^8 = 28 * 28 = 784 \equiv 9 \pmod{31}$	$\equiv 5040 \equiv 18 \pmod{31}$

Jetzt muss er b^k bestimmen

$k = 14 = 1110_b$	0 $15^1 = 15 \equiv 15 \pmod{31}$	$b^k = 15^{14}$
	1 $15^2 = 15 \cdot 15 = 225 \equiv 8 \pmod{31}$	$= 15^2 \cdot 15^4 \cdot 15^8$
	1 $15^4 = 8 \cdot 8 = 64 \equiv 2 \pmod{31}$	$\equiv 8 \cdot 2 \cdot 4 \pmod{31}$
	1 $15^8 = 2 \cdot 2 = 4 \equiv 4 \pmod{31}$	$\equiv 64 \equiv 2 \pmod{31}$

Nun kann er das eigentliche Chiffre, $y_2 = m \cdot b^k$ bestimmen: $y_2 \equiv 2 \cdot 27 \equiv 54 \equiv 23 \pmod{31}$.

Er schickt also an den Empfänger $c = E(m, k) \equiv (a^k \pmod{p}, m \cdot b^k \pmod{p}) \equiv (18 \pmod{31}, 23 \pmod{31})$

Dieser bildet zunächst $y_1^x \equiv 18^9$.

$x = 9 = 1001_b$	1 $18^1 = 18 \equiv 18 \pmod{31}$	$y_1^x = 18^9$
	0 $18^2 = 18 \cdot 18 = 324 \equiv 14 \pmod{31}$	$= 18^1 \cdot 18^8$
	0 $18^4 = 14 \cdot 14 = 196 \equiv 10 \pmod{31}$	$\equiv 18 \cdot 7 \pmod{31}$
	1 $18^8 = 10 \cdot 10 = 100 \equiv 7 \pmod{31}$	$\equiv 126 \pmod{31} \equiv 2 \pmod{31}$

Dieses muss er mit Hilfe erweiterter Euklidischer Algorithmus, Seite 3 noch invertieren zu $(y_1^x)^{-1} = 16$ (was eigentlich offensichtlich ist, denn $16 \cdot 2 = 32$, kann man auch so sehen :-).

Und mit $y_2 \cdot (y_1^x)^{-1} = 23 \cdot 16 = 368 \equiv 27 \pmod{31} \equiv m$ hat er schwups alles entschlüsselt.

El-Gamal-Signierschema:

Die Vorgehensweise ist ähnlich wie beim El-Gamal-Kryptosystem oben mit a , x , b und p .

$$\text{sig}_k(m) = (\gamma, \delta) = \begin{cases} \gamma = a^k \pmod{p} \\ \delta = (m - x\gamma)^{k^{-1}} \pmod{p} \end{cases} \quad \begin{array}{l} k \equiv 1 \pmod{p-1} \equiv 1 \pmod{\varphi(p)} \\ \Rightarrow a^k \equiv a^{1 \pmod{\varphi(p)}} \equiv a \pmod{p} \end{array} \quad \begin{array}{l} k \equiv 1 \text{ wäre daher} \\ \text{eine schlechte} \\ \text{Wahl} \end{array}$$

$$\text{ver}(m, \gamma, \delta) = 1 \Leftrightarrow b^{\gamma} \gamma^{\delta} \equiv a^m \pmod{p-1}$$

Korrektheit: $b^{\gamma} \gamma^{\delta} \equiv b^{a^k} (a^k)^{(m-x\gamma)^{k^{-1}}} \pmod{p} \equiv (a^x)^{a^k} (a^k)^{(m-xa^k)^{k^{-1}}} \pmod{p} \equiv a^{xa^k + k(m-xa^k)^{k^{-1}}} \pmod{p} \equiv a^{xa^k + m - xa^k} \equiv a^m \pmod{p-1}$

Sicherheit:

γ, m bekannt $\delta \equiv \log_{\gamma} (a^m b^{-\gamma}) \pmod{p} \rightarrow$ diskrets Logarithmusproblem (DLP)

δ, m bekannt kein effizientes Verfahren bekannt, um γ nach $b^{\gamma} \gamma^{\delta} \equiv a^m \pmod{p}$ zu bestimmen

γ, δ bekannt $m \equiv \log_a (b^{\gamma} \gamma^{\delta}) \pmod{p}$

Als Beispiel nehmen wir wieder dasselbe wie vorhin.

Im Sender haben wir daher bereits $p = 31$, $a = 12$, $x = 9$, $b = 15$, $k = 14$, $\gamma \equiv a^k = 18$ und $m = 27$.

$$14^{-1} \equiv 20 \Leftrightarrow 20^{-1} \equiv 14$$

				i
				0
Schritt	q	a	b	
	-	14	31	1
$2 \cdot 14 = 28$	1	2	3	-2
$4 \cdot 3 = 12$	2	4	3	9
$2 \cdot 1 = 2$	3	1	1	-11
-11 $\equiv 20 \pmod{31}$				

Berechnen wir das Inverse k^{-1} zu $k = 14$.

$$\begin{aligned} \delta &= (m - x\gamma)k^{-1} \pmod{p} \\ &\equiv (27 - 9 \cdot 18) \cdot 20 \pmod{31} \\ &\equiv (-135 \equiv 20 \pmod{31}) \cdot 20 \pmod{31} \\ &\equiv 400 \pmod{31} \equiv 28 \pmod{31} \end{aligned}$$

Damit ist die Signatur $\text{sig}_k(m) = (\gamma, \delta) = (18, 28)$

Der Empfänger berechnet zunächst $a^m = 12^{27} \pmod{30}$:

$m = 27 = 11011_b$	1 $12^1 = 12 \equiv 12 \pmod{30}$	
	1 $12^2 = 12 \cdot 12 = 144 \equiv 14 \pmod{30}$	$a^m = 12^{27}$
	0 $12^4 = 14 \cdot 14 = 196 \equiv 16 \pmod{30}$	$= 12^1 \cdot 12^2 \cdot 12^8 \cdot 12^{16}$
	1 $12^8 = 16 \cdot 16 = 256 \equiv 16 \pmod{30}$	$\equiv 12 \cdot 14 \cdot 16 \cdot 16 \pmod{30}$
	1 $12^{16} = 16 \cdot 16 = 256 \equiv 16 \pmod{30}$	$\equiv 43008 \equiv 18 \pmod{30}$

Dann folgt $b^\gamma = 15^{18}$:

$\gamma = 18 = 10010_b$	0 $15^1 = 15 \equiv 15 \pmod{31}$	
	1 $15^2 = 15 \cdot 15 = 225 \equiv 8 \pmod{31}$	$b^\gamma = 15^{18}$
	0 $15^4 = 8 \cdot 8 = 64 \equiv 2 \pmod{31}$	$= 15^2 \cdot 15^{16}$
	0 $15^8 = 2 \cdot 2 = 4 \equiv 4 \pmod{31}$	$\equiv 8 \cdot 16 \pmod{31}$
	1 $15^{16} = 4 \cdot 4 = 16 \equiv 16 \pmod{31}$	$\equiv 128 \equiv 4 \pmod{31}$

und zu guter letzt noch $\gamma^\delta = 18^{28}$:

$\gamma = 28 = 11100_b$	0 $18^1 = 18 \equiv 18 \pmod{31}$	
	0 $18^2 = 18 \cdot 18 = 324 \equiv 14 \pmod{31}$	$\gamma^\delta = 18^{28}$
	1 $18^4 = 14 \cdot 14 = 196 \equiv 10 \pmod{31}$	$= 18^4 \cdot 18^8 \cdot 18^{16}$
	1 $18^8 = 10 \cdot 10 = 100 \equiv 7 \pmod{31}$	$\equiv 10 \cdot 7 \cdot 18 \pmod{31}$
	1 $18^{16} = 7 \cdot 7 = 49 \equiv 18 \pmod{31}$	$\equiv 1260 \equiv 20 \pmod{31}$

Nun kann er bestimmen $\text{ver}(m, \gamma, \delta) = 1 \Leftrightarrow b^\gamma \gamma^\delta \equiv a^m \pmod{p-1} \Leftrightarrow 4 \cdot 20 = 80 \equiv 18 \pmod{31} \equiv 18 \pmod{30} = 1$

19. Digitale Signaturen

Unterschriften auf Briefen sind physikalisch an das Dokument gebunden, bei elektronischem Verkehr kann man sie aber leicht ändern, umformen und somit gefälschte Texte unter falschem Namen signieren usw.

Digitale Signaturen sollen untrennbar mit dem Dokument verbunden sein und somit dessen Korrektheit und auch den korrekten Absender garantieren.

Signaturssystem $\equiv (P, A, K, S, V)$

P ist die Menge der Nachrichten/Dokumente

A ist die Menge der Signaturen, z.B. $A = \{0, 1\}^*$

K ist die endliche Menge der Schlüssel k

S $\forall k \in K \exists \text{sig}_k : P \rightarrow A \in S$ der Menge der Signaturalgorithmen

V und eine entsprechende Funktion $\text{ver}_k : P \times A \rightarrow \{0, 1\} \in V$ der Verifikationalgorithmen

$$\text{ver}_k(x, y) = \begin{cases} 1 & : y = \text{sig}_k(x) \\ 0 & : \text{sonst} \end{cases}$$

Bei $|S| \ll |m|, h : \{0, 1\}^N \rightarrow \{0, 1\}^n : N \gg n$ spricht man von einer Hash-Funktion.

Für das Signieren gibt es zwei grundlegende Möglichkeiten:

erst signieren, dann verschlüsseln: OK

erst verschlüsseln, dann signieren: Relativ unsicher, da Signatur abgetrennt und durch andere ersetzt werden kann

Dann kann nämlich ein Gegner so tun, als käme eine Nachricht von ihm, obwohl sie von jemand anders kommt. Er braucht nur die echte Signatur zu entfernen und durch seine zu ersetzen.

RSA-Signatursystem (genau andersrum wie RSA-Verschlüsselung)

Sender nimmt geheimen Schlüssel d um eine verschlüsselte Variante des Dokuments unten anzuhängen, gemäß $c = m^d$.

Der Empfänger nutzt $m = c^e$ zum Entschlüsseln mit dem öffentlichen Schlüssel e, und vergleicht die Dokumente.

20. Geburtstagsparadoxon

Aufgabe 1/4

Wie viele Leute müssen in einem Raum sein, so dass mit einer Wahrscheinlichkeit von $> 0,5$ mindestens einer Geburtstag hat?

$$P(\text{alle Geburtstag}) = \left(\frac{1}{365}\right)^n \qquad P(\text{keiner Geburtstag}) = \left(1 - \frac{1}{365}\right)^n$$

$$P(\geq 1 \text{ Geburtstag}) = 1 - \left(1 - \frac{1}{365}\right)^n = 1 - \left(\frac{364}{365}\right)^n > 0,5 \Rightarrow P(\text{keiner Geburtstag}) = \left(\frac{364}{365}\right)^n \leq 0,5$$

$$n \geq \log_{\frac{364}{365}} 0,5 = \frac{\ln 0,5}{\ln \frac{364}{365}} = 252,652 \Rightarrow n \geq 253$$

Wie viele Leute müssen in einem Raum sein, so dass mit einer Wahrscheinlichkeit von $> 0,5$ mindestens zwei Personen am gleichen Tag Geburtstag haben?

Wenn alle Personen an verschiedenen Tagen Geburtstag haben, so gilt:

$$P(\text{keine am gleichen Tag Geburtstag}) = P(\bar{x}) = \prod_{i=0}^{n-1} \frac{365-i}{365} = \prod_{i=0}^{n-1} \left(1 - \frac{i}{365}\right)$$

Mit der Vereinfachung $1+x \leq e^x \quad \forall x \Leftrightarrow 1-x \leq e^{-x} \quad \forall x$ folgt

$$P(\bar{x}) \leq \prod_{i=0}^{n-1} e^{-\frac{i}{365}} = e^{-\sum_{i=0}^{n-1} \frac{i}{365}} = e^{-\frac{1}{365} \sum_{i=0}^{n-1} i} = e^{-\frac{1}{365} \frac{(n-1)n}{2}} = e^{-\frac{(n-1)n}{730}}$$

Für das Gegenereignis folgt dann:

$$P(\geq \text{zwei haben am gleichen Tag Geburtstag}) = P(x) = 1 - P(\bar{x}) \geq 1 - e^{-\frac{(n-1)n}{730}} \geq 0,5$$

$$P(x) \geq 0,5 = e^{-\frac{(n-1)n}{730}}; \quad 2 = e^{\frac{(n-1)n}{730}}; \quad \ln 0,5 = \frac{(n-1)n}{730}; \quad 730 \ln 0,5 = n^2 - n$$

$$n \geq \frac{1}{2} + \sqrt{\frac{1}{4} + 730 \ln 0,5} = 22,9999 \Rightarrow n \geq 23$$

(andere Lösung wäre negativ)

21. $ax \equiv b \pmod{l}$

Aufgabe 1/5

Bestimme alle Parameter, für die die Gleichung $ax \equiv b \pmod{l}$ Lösungen hat.

Zuerst beweisen wir die allgemeine Lösbarkeit:

- 1 wir setzen $g = \text{ggT}(a, l)$
- 2 löst x_0 die Gleichung, so gilt $ax_0 \equiv b \pmod{l} \Rightarrow ax_0 - b \equiv 0 \pmod{l} \Rightarrow l \mid (ax_0 - b)$
- 3 mit (1) folgt $l \mid (ax_0 - b) \wedge g = \text{ggT}(l, a) \Rightarrow g \mid (ax_0 - b)$
- 4 mit (1) folgt $g = \text{ggT}(l, a) \wedge g \mid (ax_0 - b) \Rightarrow g \mid (i \cdot g \cdot x_0 - b) \Rightarrow g \mid b$
- 5 nach (1) gilt auch: $g = \alpha a + \beta l: \alpha, \beta \in \mathbb{Z}$
- 6 mit (4) folgt dann $g = \alpha a + \beta l: \alpha, \beta \in \mathbb{Z} \wedge g \mid b \Rightarrow b = v(\alpha a + \beta l) = v\alpha a + v\beta l: v \in \mathbb{Z}$
- 7 demnach gilt $b = v\alpha a + v\beta l \Rightarrow b \pmod{l} = v\alpha a \Leftrightarrow b \equiv v\alpha a \pmod{l}$
- 8 eingesetzt in (2) ergibt das $ax_0 \equiv b \pmod{l} \Leftrightarrow ax_0 \equiv v\alpha a \pmod{l} \Leftrightarrow x_0 \equiv v\alpha \pmod{l}$

Die Gleichung ist also lösbar, wenn $g \mid b$ gilt. Für die Anzahl der Lösungen rechnen wir:

- 9 aus der Lösbarkeit (2) $ax \equiv b \pmod{l}$ folgt $ax_0 + j \cdot l \equiv b \pmod{l}$
- 10 und damit mit $g \mid a$ auch $a \left(x_0 + \frac{il}{g} \right) = ax_0 + \frac{a \cdot il}{g} = ax_0 + jl \equiv ax_0 \pmod{l} \Rightarrow a \left(x_0 + \frac{il}{g} \right) \equiv ax_0 \pmod{l}$
- 11 und damit $a \left(x_0 + \frac{il}{g} \right) \equiv b \pmod{l}$
- 12 Setzt man $i = g$, so folgt $a \left(x_0 + \frac{gl}{g} \right) = a(x_0 + l) \Rightarrow ax_0 + al$

Aus 12 ergibt sich also, dass es genau g Lösungen gibt, $i \in \{0, 1, \dots, g-1\}$.

22. Markov-Ungleichung

Aufgabe 2/1

Beweise die Markov-Ungleichung für nicht-negative X $P(X \geq a) \leq \frac{E(X)}{a}$

$$P(X \geq a) = \int_a^{+\infty} \varphi(x) dx$$

$$E(X) = \int_{-\infty}^{+\infty} x \varphi(x) dx = \int_0^{+\infty} x \varphi(x) dx$$

$$\frac{E(X)}{a} = \frac{\int_0^{+\infty} x \varphi(x) dx}{a} = \int_0^a \frac{x}{a} \varphi(x) dx + \int_a^{+\infty} \frac{x}{a} \varphi(x) dx \geq \int_a^{+\infty} \frac{x}{a} \varphi(x) dx \geq \int_a^{+\infty} 1 \varphi(x) dx$$

23. Vektorräume

Aufgabe 2/2

Wir wählen unabhängig von einander gemäß der Gleichverteilung Vektoren m_1, \dots, m_p aus \mathbb{Z}_a^p . Wie groß ist die Wahrscheinlichkeit, dass m_1, \dots, m_p den Raum \mathbb{Z}_a^p aufspannen?

Wir gehen induktiv vor.

Ein einzelner Vektor spannt immer einen eindimensionalen Unterraum auf.

Nimmt man einen zweiten Vektor hinzu, so spannen die beiden Vektoren genau dann einen zweidimensionalen Raum auf, wenn der zweite Vektor nicht in dem Unterraum liegt, den der erste aufspannt. und so weiter.

Da es für jede Stelle eines Vektors a Möglichkeiten gibt, und jeder Vektor p Stellen hat, so existieren insgesamt a^p Vektoren.

Im $(i-1)$ dimensionalen Unterraum sind $(i-1)$ Stellen voll belegt, also enthält er a^{i-1} Vektoren.

Damit folgt, dass ein neu gezogener, i -ter Vektor mit $P_i = \frac{a^p - a^{i-1}}{a^p} = 1 - \frac{a^{i-1}}{a^p}$ nicht in diesem Raum liegt.

Insgesamt ziehen wir p Vektoren, also folgt $P = \prod_{i=1}^p P_i = \prod_{i=1}^p \left(1 - \frac{a^{i-1}}{a^p}\right)$

Diese Wahrscheinlichkeit soll nach unten und nach oben abgeschätzt werden.

Für die Abschätzung nach unten nutzen wir: $(1-x)(1-y) > 1-(x+y) \quad \forall x, y > 0$

$$P = \prod_{i=1}^p \left(1 - \frac{a^{i-1}}{a^p}\right) > 1 - \sum_{i=1}^p \frac{a^{i-1}}{a^p} > 1 - \frac{1}{a^p} \sum_{i=1}^p a^{i-1}$$

nach der geometrischen Summe $\sum_{i=1}^n aq^{i-1} = a \frac{q^n - 1}{q - 1}$ ergibt dies

$$P > 1 - \frac{1}{a^p} \sum_{i=1}^p a^{i-1} = 1 - \frac{1}{a^p} \frac{a^p - 1}{a - 1} = \frac{a^p - 1}{a^p} \frac{1}{a - 1} > 1 - \frac{1 - 0}{a - 1} = 1 - \frac{1}{a - 1}$$

Um nach oben abzuschätzen, setzen wir wieder $1+x \leq e^x \quad \forall x \Leftrightarrow 1-x \leq e^{-x} \quad \forall x$

$$P = \prod_{i=1}^p \left(1 - \frac{a^{i-1}}{a^p}\right) < \prod_{i=1}^p e^{-\frac{a^{i-1}}{a^p}} = e^{-\sum_{i=1}^p \frac{a^{i-1}}{a^p}} = e^{-\frac{1}{a^p} \sum_{i=1}^p a^{i-1}} = e^{-\frac{1}{a^p} \frac{a^p - 1}{a - 1}} = e^{-\frac{1}{a - 1} * \frac{1}{a^{p(a-1)}}}$$

Und damit weiß man: $1 - \frac{1}{a - 1} < P < e^{-\frac{1}{a - 1} * \frac{1}{a^{p(a-1)}}}$

24. Funktionen

Aufgabe 3/1

Unter allen Funktionen $f: \{1, \dots, k\} \rightarrow \{1, \dots, n\}$ mit $k \leq n$ wird gemäß der Gleichverteilung eine Funktion f_0 gewählt. Für welche Parameter k, n ist die Wahrscheinlichkeit, dass f_0 injektiv ist, mindestens 50%?

Da es jeden der k Funktionswerte n Möglichkeiten gibt, existieren insgesamt n^k Funktionen. Bei einer injektiven Funktion darf kein Parameter doppelt vorkommen, also folgt:

$$P = \frac{n(n-1)(n-2)\dots(n-k+1)}{n^k} = \frac{\prod_{i=0}^{k-1} (n-i)}{n^k} = \frac{\prod_{i=0}^{k-1} (n-i)}{\prod_{i=0}^{k-1} n} = \prod_{i=0}^{k-1} \frac{n-i}{n} = \prod_{i=0}^{k-1} \left(1 - \frac{i}{n}\right)$$

Gesucht ist eine untere Schranke für die Wahrscheinlichkeit, wir nehmen also wieder $(1-x)(1-y) > 1-(x+y) \quad \forall x, y > 0$.

$$P = \prod_{i=0}^{k-1} \left(1 - \frac{i}{n}\right) > 1 - \sum_{i=0}^{k-1} \frac{i}{n} = 1 - \frac{1}{n} \sum_{i=0}^{k-1} i = 1 - \frac{1}{n} \frac{k(k-1)}{2} = 1 - \frac{k^2 - k}{2n}$$

$$P = 0,5 > 1 - \frac{k^2 - k}{2n} \Rightarrow \frac{k^2 - k}{2n} > 0,5 \Rightarrow k^2 - k - n > 0 \Rightarrow k \leq \frac{1}{2} + \sqrt{\frac{1}{4} + n}$$

Da k nie negativ ist, ist die Aufgabe für $k \leq \frac{1}{2} + \sqrt{\frac{1}{4} + n}$ erfüllt.

25. Satz von Euler

$\text{ggT}(a, p) = 1 \Rightarrow a^{\varphi(p)} \equiv 1 \pmod{p}$ $\varphi(p)$ ist die Anzahl der Elemente $x < p$ mit x teilt nicht p

\mathbb{Z}_p^* ist die Gruppe der zu p teilerfremden Zahlen. In ihr sind demnach $\varphi(p)$ Elemente enthalten.

Da \mathbb{Z}_p^* bezüglich der Multiplikation eine Gruppe ist, bleiben für alle $x \in \mathbb{Z}_p^*$ auch $(a * x) \in \mathbb{Z}_p^*$.

Wir stellen die Gruppe $\mathbb{Z}_p^* = \{x_1, x_2, \dots, x_{\varphi(p)}\} = \{a x_1, a x_2, \dots, a x_{\varphi(p)}\}$ auf.

Weiterhin gilt $a * x_i \equiv a * x_j \pmod{p} \Leftrightarrow a(x_i - x_j) \equiv 0 \pmod{p}$ und mit $\text{ggT}(a, p) = 1$ folgt sogar $x_i \equiv x_j \pmod{p}$.

$$1 * \prod_{i=1}^{\varphi(p)} x_i \equiv \prod_{i=1}^{\varphi(p)} a x_i \equiv a^{\varphi(p)} * \prod_{i=1}^{\varphi(p)} x_i \pmod{p} \Rightarrow 1 \equiv a^{\varphi(p)} \pmod{p}$$

Da $\text{ggT}(x_i, p) = 1$ für $i = 1 \dots \varphi(p)$ gilt, ist auch $\text{ggT}\left(\prod_{i=1}^{\varphi(p)} x_i, p\right) = 1$ und es folgt $a^{\varphi(p)} \equiv 1 \pmod{p}$.

26. kleiner Satz von Fermat

Da für Primzahlen p gilt $\varphi(p) = p-1$ gilt, folgt aus 25. Satz von Euler:

$$a^p \equiv a \pmod{p} \qquad a^{p-1} \equiv 1 \pmod{p}$$

Beweis nach Satz von Euler, $\varphi(p) = p-1$ für Primzahlen p .

27. schnelles Potenzieren

Wird z.B. bei dem RSA-System angewendet, $O(\log e)$

```
power(g, e) //= ge
{
  p = 1;
  s = g;
  while (e > 0)
  {
    if((e & 1) != 0) p = p * s;    //e ist ungerade

    e = e / 2;
    s = s * s;
  }
  return p;
}
```

Um g zu quadrieren sind $O(\log g)$ Schritte nötig.

Zur Berechnung von $g, g^2, g^4, \dots, g^{2^i} : 2^i \leq e$ reichen $O\left(\sum_{j=0}^{i-1} (\log g^{2^j})^2\right)$ Schritte.

$$\sum_{j=0}^{i-1} (\log g^{2^j})^2 = \sum_{j=0}^{i-1} (2^j \log g)^2 = \sum_{j=0}^{i-1} (2^{2j} \log^2 g) \leq 2^{2i} \log^2 g \leq e^2 \log^2 g \Rightarrow O(\log^2 g)$$

28. schnelles Wurzelziehen

Aufgabe 4/3

Der Algorithmus bestimmt in $O(\text{poly}(\log n))$, ob eine s -te Wurzel von n existiert und berechnet sie.

n ist k -stellige Binärzahl $k = \lfloor \log_2 n \rfloor$, und damit $2^{k-1} \leq n < 2^k$.

Dann gilt $\sqrt[s]{2^{k-1}} \leq \sqrt[s]{n} < \sqrt[s]{2^k}$, und somit $2^{\frac{k-1}{s}} \leq \sqrt[s]{n} < 2^{\frac{k}{s}}$.

Wir suchen also im Intervall $\left[2^{\frac{k-1}{s}}, 2^{\frac{k}{s}}\right]$ mit binärer Suche ein $x \in \mathbb{N} : x^s = n$.

Für diese binäre Suche sind maximal $\log_2 \left[\left[2^{\frac{k-1}{s}}, 2^{\frac{k}{s}}\right] \right] < \log_2 \left[2^{\frac{k}{s}} \right] = \frac{k}{s} = \frac{\log_2 n}{s}$ Schritte nötig.

Mittels 27. schnelles Potenzieren erfolgt die Berechnung der Kandidatenwerte y^s in Polynomialzeit.

Das Potenzieren ging $s^2 \log^2 y$ mit $y \leq 2^{\frac{k}{s}}$ und $(2n)^{\frac{1}{s}} \leq 2^{\frac{k}{s}} \leq 2(2n)^{\frac{1}{s}}$ gilt dann:

$$\text{Aufwand} \leq s^2 \log^2 (c * n^{\frac{1}{s}}) \approx c \log^2 n.$$

$$\text{Damit folgt insgesamt } O\left(\frac{\log_2 n}{s} * c \log^2 n\right) = O(\log^3 n)$$

29. heuristischer Primzahltest von Solovay und Strassen

Eingabe: natürliche Zahl n , Sicherheitsparameter $2^{\text{Sicherheit}} \sim k$.

- 1) Teste, ob n eine nichttriviale Potenz ist, also ob $n = b^m$ für $b, m > 1$ gilt. Falls ja, dann ist n keine Primzahl.
- 2) Wähle zufällig $0 < a_1, a_2, \dots, a_k < n$ unabhängig von einander gemäß der Gleichverteilung
- 3) Falls $\text{ggT}(a_i, n) \neq 1$ für ein $i = 1, 2, \dots, k$ gilt, dann ist n keine Primzahl
- 4) Berechne $z_i = a_i^{\frac{n-1}{2}} \bmod n$ für $i = 1, 2, \dots, k$
- 5) Falls $z_i \neq \pm 1 \bmod n$ für ein i gilt, dann ist n keine Primzahl
- 6) Falls $z_i \equiv \pm 1 \bmod n$ für $\forall i = 1, 2, \dots, k$, dann ist n *wahrscheinlich* eine Primzahl.

Laufzeit $\sim k$, Fehlerwahrscheinlichkeit $\sim \frac{1}{2^k}$

30. deterministischer Primzahlalgorithmus von Agrawal, Kayal und Soxena

Eingabe: natürliche Zahl n .

- 1) If (n ist nichttriviale Potenz) \rightarrow keine Primzahl $\log^3 n$
(mit binärer Suche $O(\log n)$ und Wurzelziehen $O(\log^2 n)$)
- 2) $r = 2$
- 3) while ($r < n$) $\log^6 n$ Mal
{
- 4) if ($\text{ggT}(r, n) \neq 1$) \rightarrow keine Primzahl $\log^2 n$
- 5) if (r ist Primzahl) (mit Brute Force) $r^{1/2} = \text{poly}(\log \log n)$
{
- 6) Sei q größter Primfaktor von $r-1$ analog
- 7) if $\left(q \geq 4\sqrt{r} \log n \right) \wedge \left(n^{\frac{r-1}{q}} \not\equiv 1 \bmod r \right)$ break $\log^3 n$
}
- 8) $r = r + 1$
}
- 9) For $a=1$ to $2\sqrt{r} \log n$ $\log^4 n$ oft
{
- 10) if $\left((r-a)^n \not\equiv x^n - a \bmod (x^{r-1}, n) \right) \rightarrow$ keine Primzahl $O^\sim(r \log n + \log^2 n)$
}
- 11) Primzahl

31. chinesischer Restsatz

Für die teilerfremden Zahlen n_1, n_2, \dots, n_i gilt für beliebige a_1, a_2, \dots, a_i , dass genau ein x für das für alle $j \in [1, i]$ gilt: $x \equiv a_j \pmod{n_j}$.

Diese Lösung x findet man wie folgt in $O(\text{poly}(k, \log(n_1 * n_2 * \dots * n_i)))$:

1) bilde das Produkt $n = \prod_{j=1}^i n_j$

2) bilde die einzelnen Quotienten $N_j = \frac{n}{n_j}$

Dazu berechnen wir im jeweiligen Restklassenring n_j die Inversen x_j mit dem erweiterten Euklidischen Algorithmus:

3) $x_j * N_j \equiv 1 \pmod{n_j} \Leftrightarrow x_j \equiv N_j^{-1} \pmod{n_j}$

4) Dann gilt $x = \sum_{j=1}^i x_j N_j a_j \pmod{n}$

Beispiel 1: Die Aufgabe des Sun-Tsu 100 nach Christus:

$$\begin{aligned} x &\equiv 2 \pmod{3} \\ n_1 &= 3 \end{aligned}$$

$$\begin{aligned} x &\equiv 3 \pmod{5} \\ n_2 &= 5 \end{aligned}$$

$$\begin{aligned} x &\equiv 2 \pmod{7} \\ n_3 &= 7 \end{aligned}$$

$$\text{damit ist } n = 3 * 5 * 7 = 105$$

$$\begin{aligned} N_1 &= 5 * 7 = 35 \equiv 2 \pmod{3} \\ 2 * 2 &= 4 \Rightarrow N_1^{-1} \pmod{3} \equiv 2 \pmod{3} \end{aligned}$$

$$\begin{aligned} N_2 &= 3 * 7 = 21 \equiv 1 \pmod{5} \\ N_2^{-1} \pmod{5} &\equiv 1 \pmod{5} \end{aligned}$$

$$\begin{aligned} N_3 &= 3 * 5 = 15 \equiv 1 \pmod{7} \\ N_3^{-1} \pmod{7} &\equiv 1 \pmod{7} \end{aligned}$$

$$x = \sum_{j=1}^i x_j N_j a_j \pmod{n} = 2 * 35 * 2 + 1 * 21 * 3 + 1 * 15 * 2 = 233 \equiv 23 \pmod{105}$$

Beispiel 2:

$$\begin{aligned} x &\equiv 3 \pmod{5} \\ n_1 &= 5 \end{aligned}$$

$$\begin{aligned} x &\equiv 1 \pmod{7} \\ n_2 &= 7 \end{aligned}$$

$$\begin{aligned} x &\equiv 2 \pmod{11} \\ n_3 &= 11 \end{aligned}$$

$$\text{damit ist } n = 5 * 7 * 11 = 385$$

$$\begin{aligned} N_1 &= 7 * 11 = 77 \equiv 2 \pmod{5} \\ 2 * 3 &= 6 \Rightarrow N_1^{-1} \pmod{5} \equiv 3 \pmod{5} \end{aligned}$$

$$\begin{aligned} N_2 &= 5 * 11 = 55 \equiv 6 \pmod{7} \\ N_2^{-1} \pmod{7} &\equiv 6 \pmod{7} \end{aligned}$$

$$\begin{aligned} N_3 &= 3 * 7 = 21 \equiv 10 \pmod{11} \\ N_3^{-1} \pmod{11} &\equiv 10 \pmod{11} \end{aligned}$$

$$x = \sum_{j=1}^i x_j N_j a_j \pmod{n} = 3 * 77 * 3 + 1 * 55 * 6 + 2 * 21 * 10 = 1443 \equiv 288 \pmod{385}$$

Beispiel 3:

$$x \equiv 3 \pmod{5}$$

$$x \equiv 1 \pmod{10}$$

$$x \equiv 2 \pmod{15}$$

Geht nicht. Bei $3 \pmod{5}$ kommen z.B. nur Zahlen, die auf 8 oder 3 enden in Frage, bei $1 \pmod{10}$ nur solche auf 1.

Aufgabe 5/4 Löse das folgende Kongruenzsystem: $13x \equiv 4 \pmod{99}$ und $15x \equiv 56 \pmod{101}$.

Zuerst müssen wir die Gleichungen in die Form $a \equiv b \pmod{c}$ überführen. Dazu berechnen wir die Inversen von 13 und 56 mit Hilfe erweiterter Euklidischer Algorithmus, Seite 3.

$13^{-1} \equiv 61 \Leftrightarrow 61^{-1} \equiv 13$

		q	a	b	i
		-	13	99	0
Schritt		-	13	99	1
13*7 = 91	1	6	13	8	-7
	2	1	5	8	8
	3	1	5	3	-15
	4	1	2	3	23
	5	1	2	1	-38
		-38 \equiv 61 mod 99			

$15^{-1} \equiv 27 \Leftrightarrow 27^{-1} \equiv 15$

		q	a	b	i
		-	15	101	0
Schritt		-	15	101	1
6*15 = 90	1	6	15	11	-6
	2	1	4	11	7
2*4 = 8	3	2	4	3	-20
	4	1	1	3	27

Nun bilden wir die Gleichungen um:

$$\begin{aligned}
 13x &\equiv 4 \pmod{99} \\
 61 \cdot 13x &\equiv 61 \cdot 4 \pmod{99} \\
 793x &\equiv 244 \pmod{99} \\
 (8 \cdot 99 + 1) \cdot x &\equiv (198 + 46) \pmod{99} \\
 x &\equiv 46 \pmod{99}
 \end{aligned}$$

$$\begin{aligned}
 15x &\equiv 56 \pmod{101} \\
 27 \cdot 15x &\equiv 27 \cdot 56 \pmod{101} \\
 405x &\equiv 1512 \pmod{101} \\
 (4 \cdot 101 + 1)x &\equiv (404 + 8) \pmod{101} \\
 x &\equiv 98 \pmod{101}
 \end{aligned}$$

Jetzt wenden wir den chinesischen Restsatz an:

$$N = 99 \cdot 101 = 9999$$

$$n_1 = 101 \equiv 2 \pmod{99}$$

$$n_2 = 99 \equiv 99 \pmod{101}$$

...und wieder geht's mit dem Club der toten Mathematiker weiter...

$2^{-1} \equiv 50 \pmod{99}$

		q	a	b	i
		-	2	99	0
Schritt		-	2	99	1
2 * 49 = 98	1	49	2	1	-49
	2	-49 \equiv 50 mod 99			

$99^{-1} \equiv 50 \pmod{101}$

		q	a	b	i
		-	99	101	0
Schritt		-	99	101	1
2 * 49 = 98	1	1	99	2	-1
	2	49	1	2	50

Damit ergibt sich $x \equiv 50 \cdot 46 \cdot 101 + 50 \cdot 98 \cdot 99 \pmod{9999} \equiv 232300 + 485100 \equiv 717400 \equiv 7471 \pmod{9999}$.

Es gilt $13 \cdot 7471 \equiv 4 \pmod{99}$ und $15 \cdot 7471 \equiv 56 \pmod{101}$

32. Las-Vegas-Algorithmus

Aufgabe 5/3

Ein Las-Vegas-Algorithmus ist ein probabilistischer Algorithmus der Antworten gibt, die dann auch korrekt sind, aber auch mit einer Wahrscheinlichkeit von $\varepsilon > 0$ gar keine Antwort gibt.

- Zeige, dass die erwartete Anzahl an Durchläufen eines Las-Vegas-Algorithmus mit Wahrscheinlichkeit höchstens ε für keine Antwort maximal $\frac{1}{1-\varepsilon}$ ist.
- Wieviele Iterationen des unter a) angegebenen Las-Vegas-Algorithmus sind maximal erforderlich, damit die Wahrscheinlichkeit für „keine Antwort“ maximal δ ist?

a)
$$EX = \sum_{i=1}^{\infty} i \varepsilon^{i-1} (1-\varepsilon) = \frac{1}{1-\varepsilon}$$

b)
$$P(n \text{ Mal keine Antwort}) < \varepsilon^n < \delta \Rightarrow n = \log_{\varepsilon} \delta = \frac{\log \delta}{\log \varepsilon}$$

33. Logarithmieren

Aufgabe 5/5

- Angenommen Sie wissen, dass $\log_2 x$ eine natürliche Zahl ist. Wie können Sie effizient für die Eingabe x den Wert $\log_2 x$ berechnen?

Die Zahl ist in Binärdarstellung gegeben. Ist sie eine Zweierpotenz, so taucht nur an exakt einer Stelle eine 1 auf. Die Position dieser 1 von rechts her mit 0 beginnend abgezählt entspricht dem Logarithmus. Das lineare Durchgehen der Zahl hat den Aufwand von $O(\log n)$.

- Wie kann man „ $\log_2 x \bmod p$ “ für eine Primzahl p algorithmisch lösen, also die Gleichung $2^y \equiv x \bmod p$ nach y auflösen?

Es handelt sich um das diskrete Logarithmus Problem. Man muss leider alle Werte mit $y = 0, 1, \dots, p-1$ durchtesten, und gucken, ob $2^y \equiv x \bmod p$ erfüllt ist.

34. Ordnung eines Elements

Aufgabe 6/2

Gegeben sei eine zyklische Gruppe G mit der Primfaktorzerlegung $\prod_{i=1}^l p_i^{e_i}$ von $|G|$. Wie kann man schnell entscheiden, welche Ordnung ein gegebenes Element hat?

Die Ordnung eines Elements $g \in G$ ist die Anzahl der Elemente in der von g erzeugten Untergruppe $U = \{g_0, g_1, g_2, \dots\}$. Nun gilt $|U|$ teilt $|G|$, denn die Nebenklassen $h*U$ von U haben alle die gleiche Kardinalität und ihre Vereinigung ergibt die Gruppe G . Damit ist $\text{ord}(g)$ ein Teiler von $|G|$.

Es gilt $\varphi(|G|) = \prod_{i=1}^l (e_i + 1)$, wie man leicht an einem einfachen Beispiel (z.B. $2^2 * 3 * 5 = 60$, $\varphi(60)$?) nachvollziehen kann.

Für die Ordnung von g gibt es also $\varphi(|G|)$ Möglichkeiten, die einfach durchprobiert werden müssen,

mit $x = \prod_{i=1}^l p_i^{f_i} \forall f_i \in [1, e_i]$. Die Ordnung ist dann das kleinste x mit $g^x = 1$.

35. superwachsende Folgen

Aufgabe 6/3

- a) Ist für ein festes $k \in \mathbb{N}$ die Folge $1^k, 2^k, 3^k, \dots$ superwachsend?

Nein. Denn schon im nächsten Schritt 4^k bei $k = 1$ sieht man, dass $1 + 2 + 3 = 6 > 4$ gilt.

Wäre die Folge superwachsend, so würde $\sum_{i=1}^l i^k < (l+1)^k$ gelten.

$$\sum_{i=1}^l i^k > \int_0^{l+1} x^k dx = \left[\frac{x^{k+1}}{k+1} \right]_0^{l+1} = \frac{(l+1)^{k+1}}{k+1} \Rightarrow \exists l_0 : \frac{(l+1)^{k+1}}{k+1} > (l+1)^k \quad \forall l > l_0$$

also ist (i^k) nicht superwachsend.

- b) Zeige, dass jede Folge (a_i) mit $a_{i+1} \geq 2 a_i$ superwachsend ist.

Wir nutzen Induktion, um die Richtigkeit zu beweisen.

Wir setzen voraus: $a_1 < a_2$

$$\text{Dann folgt: } a_i \geq \sum_{j=1}^{i-1} a_j \Rightarrow a_{i+1} \geq 2 a_i = a_i + a_i \geq a_i + \sum_{j=1}^{i-1} a_j.$$

Die kleinste Folge, die in diese Kategorie fällt, ist die der Zweierpotenzen: $a_i = 2^i, a_{i+1} = 2 a_i$.

36. erzeugendes Elemente in \mathbb{Z}_p^*

Aufgabe 6/4

- a) Sei g ein erzeugendes Element in \mathbb{Z}_p^* für eine Primzahl p . Wie kann man zu $b \in \mathbb{Z}_p^*$ algorithmisch ein $x \in \mathbb{N}$ mit $g^x \equiv b \pmod{p}$ bestimmen?

Es handelt sich um das Diskrete Logarithmus Problem.

Man testet für $x = 0, 1, \dots, p-1$, ob $g^x \equiv b \pmod{p}$ erfüllt ist.

Weiter braucht man nach 26. kleiner Satz von Fermat, Seite 33 nicht zu testen, da $g^p \equiv g \pmod{p}$ ist.

- b) Wir benutzen den Algorithmus a) zum Schlüsseltausch. A und B einigen sich auf ein primitives Element $g \in \mathbb{Z}_p^*$ und jeder berechnet für sich einen geheimen Exponenten d_A und d_B mit $1 \leq d_A, d_B \leq p-2$. A berechnet $y \equiv g^{d_A} \pmod{p}$ und schickt y an B. B berechnet $z \equiv y^{d_B} \pmod{p}$ und schickt z an A. Der Wert von z sei der vereinbarte Schlüssel. Kennen dann beide z , ohne vom anderen d_A bzw. d_B zu kennen?

Ja natürlich: B hat z berechnet. A hat z von B erhalten. Daher kennen A und B z . Blödsinn.

37. Binärzahlen der Form mm

Aufgabe 7/1

Unter allen $2l$ -stelligen Binärzahlen der Form $mm = m_{l-1} \dots m_0 m_{l-1} \dots m_0$ werden zufällig gleichverteilt zwei Zahlen x, y unabhängig voneinander gezogen. Wie groß ist die Wahrscheinlichkeit, dass das Produkt $x * y$ als Binärzahl wieder die Form mm hat?

Wir zeigen, dass das nur bei $x * y = 0$ gilt, also die Wahrscheinlichkeit für zwei Binärzahlen $\neq 0$ gleich 0 ist, dass $x * y$ die Form mm hat.

$$x = \sum_{i=0}^{l-1} (2^i + 2^{l+i}) x_i = \sum_{i=0}^{l-1} 2^i x_i + 2^l \sum_{i=0}^{l-1} 2^i x_i \qquad y = \sum_{i=0}^{l-1} (2^i + 2^{l+i}) y_i = \sum_{i=0}^{l-1} 2^i y_i + 2^l \sum_{i=0}^{l-1} 2^i y_i$$

$$x * y = \left(\sum_{i=0}^{l-1} 2^i x_i + 2^l \sum_{i=0}^{l-1} 2^i x_i \right) \left(\sum_{i=0}^{l-1} 2^i y_i + 2^l \sum_{i=0}^{l-1} 2^i y_i \right) = (1 + 2 * 2^l + 2^{2l}) \left(\sum_{i=0}^{l-1} 2^i x_i \sum_{i=0}^{l-1} 2^i y_i \right)$$

$(1 + 2^{2l}) \left(\sum_{i=0}^{l-1} 2^i x_i \sum_{i=0}^{l-1} 2^i y_i \right)$ hätte offenbar die richtige Form. Es muss also gelten:

$$x * y = (1 + 2 * 2^l + 2^{2l}) \left(\sum_{i=0}^{l-1} 2^i x_i \sum_{i=0}^{l-1} 2^i y_i \right) \stackrel{!}{=} (1 + 2^{2l}) \left(\sum_{i=0}^{l-1} 2^i x_i \sum_{i=0}^{l-1} 2^i y_i \right)$$

Das ist jedoch nur bei $x * y = 2 * 2^l \left(\sum_{i=0}^{l-1} 2^i x_i \sum_{i=0}^{l-1} 2^i y_i \right) = 0 \Rightarrow x * y = 0$ erfüllt.