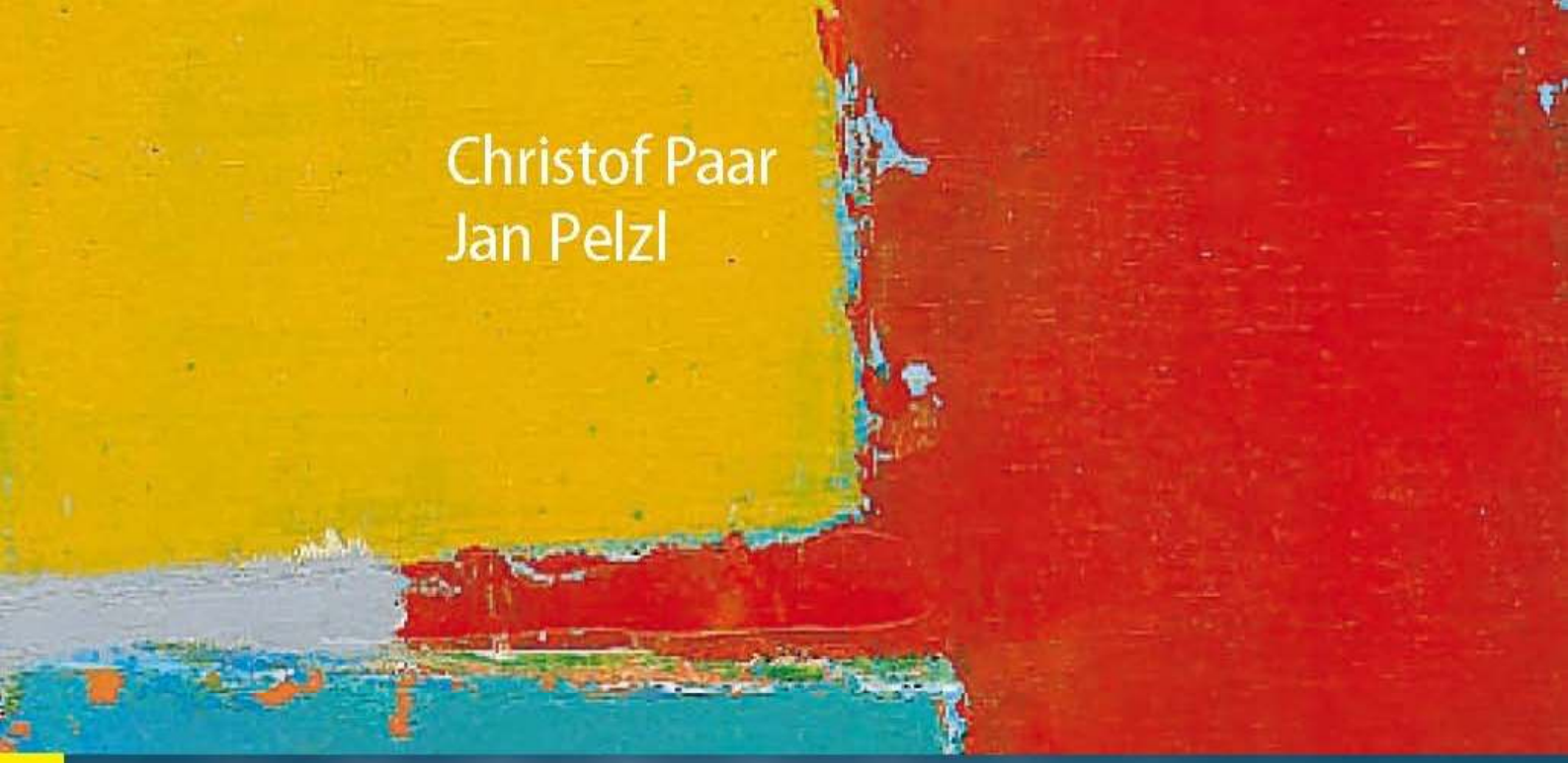


Christof Paar
Jan Pelzl

Understanding Cryptography

A Textbook for Students and Practitioners

Solutions Handbook
(Odd numbered Problems)

Springer

Solutions to Homework Problems (Odd Numbered Problems)

Understanding Cryptography

A Textbook for Students and Practitioners

by Christof Paar and Jan Pelzl

Problems of Chapter 1

1.1

1. Letter frequency analysis of the ciphertext:

letter	count	freq [%]	letter	count	freq [%]
A	5	0.77	N	17	2.63
B	68	10.53	O	7	1.08
C	5	0.77	P	30	4.64
D	23	3.56	Q	7	1.08
E	5	0.77	R	84	13.00
F	1	0.15	S	17	2.63
G	1	0.15	T	13	2.01
H	23	3.56	U	24	3.72
I	41	6.35	V	22	3.41
J	48	7.43	W	47	7.28
K	49	7.59	X	20	3.10
L	8	1.24	Y	19	2.94
M	62	9.60	Z	0	0.00

2. `Because the practice of the basic movements of kata is the focus and mastery of self is the essence of Matsubayashi Ryu karate do, I shall try to elucidate the movements of the kata according to my interpretation based on forty years of study.`

   `It is not an easy task to explain each movement and its significance, and some must remain unexplained. To give a complete explanation, one would have to be qualified and inspired to such an extent that he could reach the state of enlightened mind capable of recognizing soundless sound and shapeless shape. I do not deem myself the final authority, but my experience with kata has left no doubt that the following is the proper application and interpretation. I offer my theories in the hope that the essence of Okinawan karate will remain intact.`
3. Shoshin Nagamine, further reading: *The Essence of Okinawan Karate-Do* by Shoshin Nagamine, Tuttle Publishing, 1998.

1.3

One search engine costs $ 100 including overhead. Thus, 1 million dollars buy us 10,000 engines.

1. key tests per second: $5 \cdot 10^8 \cdot 10^4 = 5 \cdot 10^{12}$ keys/sec
   On average, we have to check $(2^{127}$ keys:
   $(2^{127}\text{keys})/(5 \cdot 10^{12}\text{keys/sec}) = 3.40 \cdot 10^{25}\text{sec} = 1.08 \cdot 10^{18}\text{years}$
   That is about $10^8 = 100,000,000$ times longer than the age of the universe. Good luck.
2. Let i be the number of Moore iterations needed to bring the search time down to 24h:
   $1.08 \cdot 10^{18}\text{years} \cdot 365/2^i = 1\text{day}$
   $2^i = 1,08 \cdot 10^{18} \cdot 365\text{days}/1\text{day}$
   $i = 68.42$
   We round this number up to 69 assuming the number of Moore iterations is discreet. Thus, we have to wait for:
   $1.5 \cdot 69 = 103.5$ years
   Note that it is extremely unlikely that Moore's Law will be valid for such a time period! Thus, a 128 bit key seems impossible to brute-force, even in the foreseeable future.

1.5

1. $15 \cdot 29 \bmod 13 \equiv 2 \cdot 3 \bmod 13 \equiv 6 \bmod 13$
2. $2 \cdot 29 \bmod 13 \equiv 2 \cdot 3 \bmod 13 \equiv 6 \bmod 13$
3. $2 \cdot 3 \bmod 13 \equiv 2 \cdot 3 \bmod 13 \equiv 6 \bmod 13$
4. $2 \cdot 3 \bmod 13 \equiv 2 \cdot 3 \bmod 13 \equiv 6 \bmod 13$

15, 2 and -11 (and 29 and 3 respectively) are representations of the same equivalence class modulo 13 and can be used “synonymously”.

1.7

1.

Multiplication table for Z_4

×	0	1	2	3
0	0	0	0	0
1	0	1	2	3
2	0	2	0	2
3	0	3	2	1

2.

Addition table for Z_5

+	0	1	2	3	4
0	0	1	2	3	4
1	1	2	3	4	0
2	2	3	4	0	1
3	3	4	0	1	2
4	4	0	1	2	3

Multiplication table for Z_5

×	0	1	2	3	4
0	0	0	0	0	0
1	0	1	2	3	4
2	0	2	4	1	3
3	0	3	1	4	2
4	0	4	3	2	1

3.

Addition table for Z_6

+	0	1	2	3	4	5
0	0	1	2	3	4	5
1	1	2	3	4	5	0
2	2	3	4	5	0	1
3	3	4	5	0	1	2
4	4	5	0	1	2	3
5	5	0	1	2	3	4

Multiplication table for Z_6

×	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	1	2	3	4	5
2	0	2	4	0	2	4
3	0	3	0	3	0	3
4	0	4	2	0	4	2
5	0	5	4	3	2	1

4. Elements without a multiplicative inverse in Z_4 are 2 and 0
   Elements without a multiplicative inverse in Z_6 are 2, 3, 4 and 0
   For all nonzero elements of Z_5 exists because 5 is a prime. Hence, all nonzero elements smaller than 5 are relatively prime to 5.

1.9

1. $x = 9 \bmod 13$
2. $x = 7^2 = 49 \equiv 10 \bmod 13$
3. $x = 3^{10} = 9^5 \equiv 81^2 \cdot 9 \equiv 3^2 \cdot 9 \equiv 81 \equiv 3 \bmod 13$
4. $x = 7^{100} = 49^{50} \equiv 10^50 \equiv (-3)^{50} = (3^{10})^5 \equiv 3^5 \equiv 3^2 = 9 \bmod 13$
5. by trial: $7^5 \equiv 11 \bmod 13$

1.11

1. `FIRST THE SENTENCE AND THEN THE EVIDENCE SAID THE QUEEN`
2. Charles Lutwidge Dodgson, better known by his pen name Lewis Carroll

1.13

$$a \equiv (x_1 - x_2)^{-1}(y_1 - y_2) \bmod m$$
$$b \equiv y_1 - a x_1 \bmod m$$

The inverse of $(x_1 - x_2)$ must exist modulo m, i.e., $\gcd((x_1 - x_2), m) = 1$.

Problems of Chapter 2

2.1

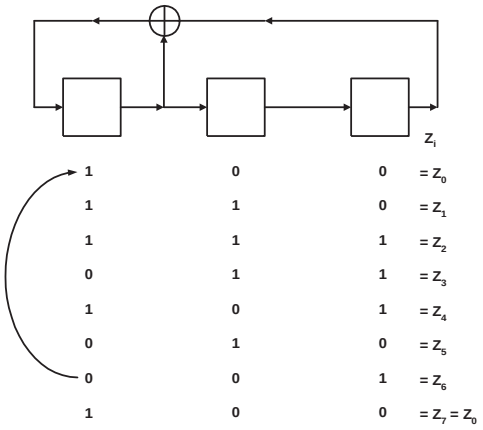
1. $y_i = x_i + K_i \bmod 26$
   $x_i = y_i - K_i \bmod 26$
   The keystream is a sequence of random integers from Z_{26}.
2. $x_1 = y_1 - K_1 = \text{"}B\text{"} - \text{"}R\text{"} = 1 - 17 = -16 \equiv 10 \bmod 26 = \text{"}K\text{"}$ etc $\cdots$
   Decrypted Text: "KASPAR HAUSER"
3. He was knifed.

2.3

We need 128 pairs of plaintext and ciphertext *bits* (i.e., 16 byte) in order to determine the key. s_i is being computed by
$s_i = x_i \bigoplus y_i; \;\; i = 1, 2, \cdots, 128.$

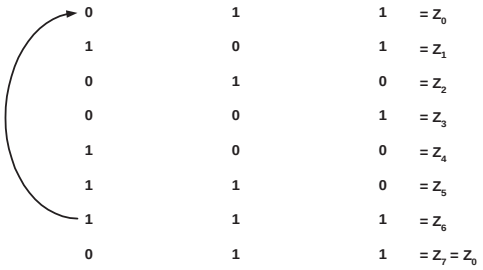
2.5

1.



Sequence 1: z_0 = 0 0 1 1 1 0 1 0 0 1 1 1 0 1 ...

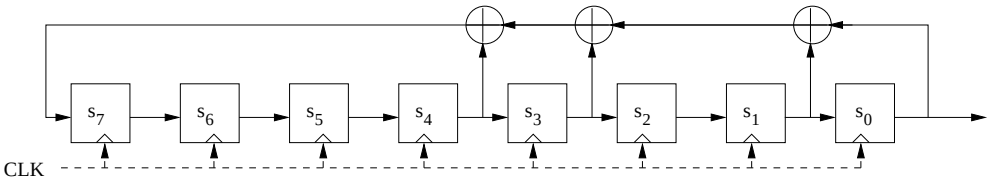
2.



Sequence 2: z_0 = 1 1 0 1 0 0 1 1 1 0 1 0 0 1 ...

3. The two sequences are shifted versions of one another.

2.7
The feedback polynomial from 2.3 is $x^8+x^4+x^3+x+1$.



1	1	1	1	1	1	1	1	$= Z_0$
0	1	1	1	1	1	1	1	$= Z_1$
0	0	1	1	1	1	1	1	$= Z_2$
0	0	0	1	1	1	1	1	
0	0	0	0	1	1	1	1	
1	0	0	0	0	1	1	1	
0	1	0	0	0	0	1	1	
0	0	1	0	0	0	0	1	
1	0	0	1	0	0	0	0	
1	1	0	0	1	0	0	0	
1	1	1	0	0	1	0	0	
0	1	1	1	0	0	1	0	
0	0	1	1	1	0	0	1	
1	0	0	1	1	1	0	0	
0	1	0	0	1	1	1	0	$= Z_{14}$
0	0	1	0	0	1	1	1	$= Z_{15}$

So, the resulting first two output bytes are $(1001000011111111)_2 = (90FF)_{16}$.

2.9

1. The attacker needs 512 consecutive plaintext/ciphertext bit pairs x_i, y_i to launch a successful attack.
2. a. First, the attacker has to monitor the previously mentioned 512 bit pairs.
   b. The attacker calculates $s_i = x_i + y_i \bmod 2$, $i = 0,1,\ldots,2m-1$
   c. In order to calculate the (secret) feedback coefficients p_i, Oscar generates 256 linearly dependent equations using the relationship between the unknown key bits p_i and the keystream output defined by the equation

$$s_{i+m} \equiv \sum_{j=0}^{m-1} p_j \cdot s_{i+j} \bmod 2; s_i, p_j \in \{0,1\}; i = 0,1,2,...,255$$

   with $m = 256$.
   d. After generating this linear equation system, it can be solved e.g. using Gaussian Elimination, revealing the 256 feedback coefficients.
3. The key of this system is represented by the 256 feedback coefficients. Since the initial contents of the LFSR are unalteredly shifted out of the LFSR and XORed with the first 256 plaintext bits, it would be easy to calculate them.

2.11
$x_i \bigoplus y_i = x_i \bigoplus (x_i \bigoplus z_i) = z_i$

$W \Longleftrightarrow 22 = 10110_2$	$J \Leftrightarrow 9 = 01001_2$
$P \Longleftrightarrow 15 = 01111_2$	$5 \Longleftrightarrow 31 = 11111_2$

$I \Leftrightarrow 8 = 01000_2$ | $A \Leftrightarrow 0 = 00000_2$

$$\begin{array}{r} x_i = 10110\,01111\,01000 \\ \underline{y_i = 01001\,11111\,00000} \\ z_i = 11111\,10000\,01000 \end{array}$$

1. Initialization Vector: $(Z_0 = 1,1,1,1,1,1)$
2. $$\begin{pmatrix} C_0 \\ C_1 \\ C_2 \\ C_3 \\ C_4 \\ C_5 \end{pmatrix} = \begin{pmatrix} 1&1&1&1&1&1 \\ 1&1&1&1&1&0 \\ 1&1&1&1&0&0 \\ 1&1&1&0&0&0 \\ 1&1&0&0&0&0 \\ 1&0&0&0&0&0 \end{pmatrix}^{-1} \cdot \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$
3. $$\begin{array}{rccccccccc} & \overbrace{J}^{} & \overbrace{5}^{} & \overbrace{A}^{} & \overbrace{0}^{} & \overbrace{E}^{} & \overbrace{D}^{} & \overbrace{J}^{} & \overbrace{2}^{} & \overbrace{B}^{} \\ y_i = & 01001 & 11111 & 00000 & 11010 & 00100 & 00011 & 01001 & 11100 & 00001 \\ z_i = & 11111 & 10000 & 01000 & 01100 & 01010 & 01111 & 01000 & 11100 & 10010 \\ \hline x_i = & \underbrace{10110}_{W} & \underbrace{01111}_{P} & \underbrace{01000}_{I} & \underbrace{10110}_{W} & \underbrace{01110}_{O} & \underbrace{01100}_{M} & \underbrace{00001}_{B} & \underbrace{00000}_{A} & \underbrace{10011}_{T} \end{array}$$
4. Wombats live in Tasmania.
5. Known-plaintext Attack.

Problems of Chapter 3

3.1

1. $s(x_1) \bigoplus s(x_2) = 1110$
   $s(x_1 \bigoplus x_2) = s(x_2) = 0000 \neq 1110$
2. $s(x_1) \bigoplus s(x_2) = 1001$
   $s(x_1 \bigoplus x_2) = s(x_2) = 1000 \neq 1001$
3. $s(x_1) \bigoplus s(x_2) = 1010$
   $s(x_1 \bigoplus x_2) = s(x_2) = 1101 \neq 1010$

3.3

$S_1(0) = 14 = 1110$
$S_2(0) = 15 = 1111$
$S_3(0) = 10 = 1010$
$S_4(0) = \ 7 \ = 0111$
$S_5(0) = \ 2 \ = 0010$
$S_6(0) = 12 = 1100$
$S_7(0) = \ 4 \ = 0100$
$S_8(0) = 13 = 1101$

$P(S) = D8D8\ DBBC$
$(L_1, R_1) = 0000\ 0000\ D8D8\ DBBC$ **(1)**

3.5
• IP(x) maps bit 57 to position 33, which is position 1 in R_0.
• E-Expansion box maps bit position 1 to positions 2 and 48.

• Input to S-Boxes:
$S_1 : 0\ 1\ 0\ 0\ 0\ 0$
$S_2 = S_3 = \cdots = S_7 : 0\ 0\ 0\ 0\ 0\ 0$
$S_8 : 0\ 0\ 0\ 0\ 0\ 1$

$\Rightarrow$ Two S-Boxes get a different input.
$P(S) = D058\ 5B9E$
$(L_1, R_1) = 8000\ 0000\ D058\ 5B9E$

1. 2 S-Boxes, S_1 and S_8
2. According to design criteria, a minimum of 2 bits/bit.
   $\Rightarrow 2 \cdot 2 = 4bits$
3. See (1).
4. 6 bits have changed:
   3 from S_1
   2 from S_8
   1 in the left half

3.7

1. $K_{1+i} = K_{16-i}$ for $i = 0, 1, ...7$.
2. Following (a), two equations are established:

$$\begin{aligned} C_{1+i} &= C_{16-i} \\ D_{1+i} &= D_{16-i} \qquad \text{fr } i = 0, 1, ..., 7. \end{aligned}$$

These equations yield

$$\begin{aligned} C_{0,j} &= 0 \text{ und } D_{0,j} = 0 \text{ or} \\ C_{0,j} &= 0 \text{ und } D_{0,j} = 1 \text{ or} \\ C_{0,j} &= 1 \text{ und } D_{0,j} = 0 \text{ oder} \\ C_{0,j} &= 1 \text{ und } D_{0,j} = 1 \qquad \text{fr } j = 1, 2, ..., 28. \end{aligned}$$

Hence the four weak keys after PC-1 are given by:

$$\begin{aligned} \hat{K}_{w1} &= [0...0\ 0...0] \\ \hat{K}_{w2} &= [0...0\ 1...1] \\ \hat{K}_{w3} &= [1...1\ 0...0] \\ \hat{K}_{w4} &= [1...1\ 1...1] \end{aligned}$$

3. $P(\text{randomly chose a weak key}) = \frac{2^2}{2^{56}} = 2^{-54}$.

3.9
Worst-Case: 2^{56} keys.
Average: $2^{56}/2 = 2^{55}$ keys.

3.11

1. A single DES engine can compute $100 \cdot 10^6$ DES encryptions per second. A COPACOBANA machine can, thus, compute $4 \cdot 6 \cdot 20 \cdot 100 \cdot 10^6 = 4.8 \cdot 10^{10}$ DES encryptions per second. For an average of

2^{55} encryptions for a successfull brute-force attack on DES, $2^{55}/(4.8 \cdot 10^{10}) \approx 750600$ seconds are required (which approximately is 8.7 days).
2. For a successfull average attack in one hour, $8.7\dot{2}4 \approx 18$ machines are required.
3. The machine performs a brute–force attack. However, there might be more powerful analytical attacks which explore weaknesses of the cipher. Hence, the key–search machine provides only a lower security threshold.

3.13

1. The state of PRESENT after the execution of one round is `F000 0000 0000 000F`. Below you can find all intermediate values.

Plaintext	`0000 0000 0000 0000`
Round key	`BBBB 5555 5555 EEEE`
State after KeyAdd	`BBBB 5555 5555 EEEE`
State after S-Layer	`8888 0000 0000 1111`
State after P-Layer	`F000 0000 0000 000F`

2. The round key for the second round is `7FFF F777 6AAA AAAA`. Below you can find all intermediate values.

Key	`BBBB 5555 5555 EEEE FFFF`
Key state after rotation	`DFFF F777 6AAA AAAA BDDD`
Key state after S-box	`7FFF F777 6AAA AAAA BDDD`
Key state after CounterAdd	`7FFF F777 6AAA AAAA 3DDD`
Round key for Round 2	`7FFF F777 6AAA AAAA`

Problems of Chapter 4

4.1

1. The successor of the DES, the AES, was chosen by the NIST by a public proceeding. The purpose of this public contest was to allow broadly evaluation of the candidates by as many research organisations and institutes as possible.
   This strongly contrasts to the development of DES, which was only performed by IBM and the NSA firstly keeping details (e.g. the S-boxes) in secret. DES was published and standardized in 1975.
2. 1/2/97: Call for algorithms, which could potentially lead to the AES. The selection process was governed by the NIST. 8/20/98: 15 algorithms were nominated as candidates for the selection process. 9.8.1999: 5 algorithms reach the ”finals” (Mars, RC6, Rijndael, Serpent, Twofish)
   2.10.2000: NIST elects Rijndael to AES.
3. Rijndael
4. Dr. Vincent Rijmen and Dr. Joam Daemen from Belgium
5. Rijndael supports blocksizes of 128, 192 and 256 bit, as well as key lengths of 128, 192 and 256 bit. In fact, only the version with 128 bit blocksize (and all three key lengths) is called AES.

4.3

Multiplication table for $GF(2^3)$, $P(x) = x^3 + x + 1$

$\times$	0	1	x	$x+1$	x^2	x^2+1	x^2+x	x^2+x+1
0	0	0	0	0	0	0	0	0
1	0	1	x	$x+1$	x^2	x^2+1	x^2+x	x^2+x+1
x	0	x	x^2	x^2+x	$x+1$	1	x^2+x+1	x^2+1
$x+1$	0	$x+1$	x^2+x	x^2+1	x^2+x+1	x^2	1	x
x^2	0	x^2	$x+1$	x^2+x+1	x^2+x	x	x^2+1	1
x^2+1	0	x^2+1	1	x^2	x	x^2+x+1	$x+1$	x^2+x
x^2+x	0	x^2+x	x^2+x+1	1	x^2+1	$x+1$	x	x^2
x^2+x+1	0	x^2+x+1	x^2+1	x	1	x^2+x	x^2	$x+1$

4.5 Multiplication in GF(2^4):

1. $A(x) * B(x) = (x^2+1)(x^3+x^2+1) = x^5+x^4+x^2+x^3+x^2+1$

$$A(x) * B(x) = x^5+x^4+x^3+1$$

$$\begin{array}{rl} & x\ +1 \\ x^4+x+1 \;\; x^5\ +x^4\ +x^3 & \ +1 \\ x^5 \qquad +x^2\ +x & \\ x^4\ +x^3\ +x^2\ +x\ +1 & \\ x^4 \qquad +x\ +1 & \\ x^3\ +x^2 & \end{array}$$

$C = x^3+x^2 \equiv A(x) * B(x) \bmod P(x)$.

2. $A(x) * B(x) = (x^2+1)(x+1) = x^3+x+x^2+1$
   $C = x^3+x^2+x+1 \equiv A(x) * B(x) \bmod P(x)$

The reduction polynomial is used to reduce $C(x)$ in order to reduce the result to $GF(2^4)$. Otherwise, a 'simple' multiplication without reduction would yield a result of a higher degree (e.g., with x^5) which would not belong to $GF(2^4)$ any more.

4.7

1. By the Extended Euclidean algorithm:

$$\begin{array}{lll} x^4+x+1 = [x^3](x)+[x+1] & t_2(x) = t_0 - q_1 t_1 = -q_1 = -x^3 = x^3 \\ x \qquad\qquad = [1](x+1)+1 & t_3(x) = t_1 - q_2 t_2 = 1 - 1 * x^3 = 1 - x^3 = x^3+1 \\ x+1 \qquad = [x+1](1)+0 & \end{array}$$

So, $A^{-1} = x^3+1$.
Check: $x * (x^3+1) = x^4+x \equiv (x+1)+x \bmod P(x) = 1 \bmod P(x)$.

2. By the Extended Euclidean algorithm:

$$\begin{array}{ll} x^4+x+1 = [x^2+x+1](x^2+x)+[1] & t_2 = t_0 - q_1 t_1 = -q_1 = x^2+x+1 \\ x^2+x \qquad = [x^2+x]1+[0] & \end{array}$$

So, $A^{-1} = x^2+x+1$.
Check: $(x^2+x)(x^2+x+1) = x^4+2x^3+2x^2+x = x^4+x \equiv (x+1)+x \bmod P(x) = 1 \bmod P(x)$.

4.9

■

$$B = ByteSub(A) = \begin{bmatrix} 16 & 16 & 16 & 16 \\ 16 & 16 & 16 & 16 \\ 16 & 16 & 16 & 16 \\ 16 & 16 & 16 & 16 \end{bmatrix}$$

- The ShiftRows operation does not change anything since all bytes of B equal each other.
- The MixComumn operation is equal for every resultig byte C_i and is described by $(01+01+02+03)_{hex} \cdot (16)_{hex}$. We have to remind, that all calculations have to be done in $GF(2^8)$, so that $(01+01+02+03)_{hex} = (01)_{hex}$ and hence, all resulting bytes of C remain $(16)_{hex} \Rightarrow$

$$C = MixColumn(B) = \begin{bmatrix} 16 & 16 & 16 & 16 \\ 16 & 16 & 16 & 16 \\ 16 & 16 & 16 & 16 \\ 16 & 16 & 16 & 16 \end{bmatrix}$$

- The first round key equals the unmodified AES key. So, the output of the first is

$$C \oplus K = \begin{bmatrix} 16 & 16 & 16 & 16 \\ 16 & 16 & 16 & 16 \\ 16 & 16 & 16 & 16 \\ 16 & 16 & 16 & 16 \end{bmatrix} \oplus \begin{bmatrix} FF & FF & FF & FF \\ FF & FF & FF & FF \\ FF & FF & FF & FF \\ FF & FF & FF & FF \end{bmatrix} = \begin{bmatrix} E9 & E9 & E9 & E9 \\ E9 & E9 & E9 & E9 \\ E9 & E9 & E9 & E9 \\ E9 & E9 & E9 & E9 \end{bmatrix}$$

4.11

1. $d = 01$, $b = 1 * (b_7x^7 + \ldots + b_0) = b$.
   $d_0 = b_0$, $d_1 = b_1$, $\ldots$,$d_7 = b_7$.
2. $d = 02 * b = x(b_7x^7 + \ldots + b_0) = b_7x^8 + b_6x^7 + \ldots + b_0x$
   $x^8 \equiv x^4 + x^3 + x + 1 \bmod P(x)$.
   $d = b_6x^7 + b_5x^6 + b_4x^5 + [b_3 + b_7]x^4 + [b_2 + b_7]x^3 + b_1x^2 + [b_0 + b_7]x + b_7$
   $d_7 = b_6$ $\quad$ $d_6 = b_5$
   $d_5 = b_4$ $\quad$ $d_4 = b_3 + b_7$
   $d_3 = b_2 + b_7$ $d_2 = b_1$
   $d_1 = b_0 + b_7$ $d_0 = b_7$

3. $d = 03 * b = (x+1)b = xb + b$
   Using solutions from a) and b):
   $d = (b_6 + b_7)x^7 + (b_5 + b_6)x^6 + (b_4 + b_5)x^5 + (b_3 + b_4 + b_7)x^4 + (b_2 + b_3 + b_7)x^3 + (b_1 + b_2)x^2 + (b_0 + b_1 + b_7)x + (b_0 + b_7)$
   $d_7 = b_6 + b_7$ $\quad$ $d_6 = b_5 + b_6$
   $d_5 = b_4 + b_5$ $\quad$ $d_4 = b_3 + b_4 + b_7$
   $d_3 = b_2 + b_3 + b_7$ $d_2 = b_1 + b_2$
   $d_1 = b_0 + b_1 + b_7$ $d_0 = b_0 + b_7$

4.13

1. $A = 01_h$, $A(x) = 1$
   $A^{-1}(x) = 1 = 01_h$
   $A^{-1}(x)$ is now the input to the affine transformation of Rijndael as described in Subsection 4.2.1 of the Rijndael Specifications:
   $$M \cdot A^{-1} + V$$
   where M and V are a fixed matrix and vector, respectively.
   $$M \cdot A^{-1} + V = M \cdot \begin{pmatrix} 1\\0\\0\\0\\0\\0\\0\\0 \end{pmatrix} + \begin{pmatrix} 1\\1\\0\\0\\0\\1\\1\\0 \end{pmatrix} = \begin{pmatrix} 1\\1\\1\\1\\1\\0\\0\\0 \end{pmatrix} + \begin{pmatrix} 1\\1\\0\\0\\0\\1\\1\\0 \end{pmatrix} = \begin{pmatrix} 0\\0\\1\\1\\1\\1\\1\\0 \end{pmatrix}$$
   ByteSub(01_h) = $7C_h$
2. $A = 12_h$, $A(x) = x^4 + x$
   Apply extended Euclidean algorithm: $A^{-1}(x) = x^7 + x^5 + x^3 + x = AA_h$.
   $$M \cdot A^{-1} + V = M \cdot \begin{pmatrix} 0\\1\\0\\1\\0\\1\\0\\1 \end{pmatrix} + \begin{pmatrix} 1\\1\\0\\0\\0\\1\\1\\0 \end{pmatrix} = \begin{pmatrix} 0\\1\\0\\1\\0\\1\\0\\1 \end{pmatrix} + \begin{pmatrix} 1\\1\\0\\0\\0\\1\\1\\0 \end{pmatrix} = \begin{pmatrix} 1\\0\\0\\1\\0\\0\\1\\1 \end{pmatrix}$$
   Remark: It is (big) coincidence that $M \cdot A^{-1} = A^{-1}$. This only holds for this specific value of A^{-1}.
   ByteSub(12_h) = $C9_h$

4.15

1. $RC[8] = x^7 = (10000000)_2$
2. $RC[9] = x^8 = x^4 + x^3 + x + 1 = (00011011)_2$
3. $RC[10] = x^9 = x^8 \cdot x = x^5 + x^4 + x^2 + x = (00110110)_2$

Problems of Chapter 5

5.1
Since the records are not related, we typically want to access only a single record and not its adjacent ones. The use of CBC mode is thus not well suited. ECB most seems to be the best choice.

5.3
The decryption of an "CBC-encrypted" file is defined by $x_i = d_K(y_i) \oplus y_{i-1}$. Since you know the key K and the pair (x_0, y_0) (from the first file), the unknown IV can easily be obtained by converting the equation:

$$IV = y_{-1} = d_k(y_0) \oplus x_0$$

After that, the second (unidentified) file can easily be decrypted by using the decryption equation mentioned above (with $y_{-1} = IV$).

5.5
If the same IV is used for the OFB encryption, the confidentiality may be compromized. If a plaintext block x_j of such a message m is known, the output can be computed easily from the ciphertext block y_j of the message m. This information then allows the computation of the plaintext block x'_j of any other message m' that is encrypted using the same IV.

5.7

1.
2. The problem with the scheme is that there are only 256 different inputs FB_i to the AES algorithm. That means there are only 256 different output vectors of length 128bit that form the keystream. To make things worse, the cipher output will run into a cycle quickly. Let's denote the sequence of feedback bytes by $FB_1, FB_2, \ldots$ As soon as a feedback byte FB_j is generated that is equal to an earlier one FB_i, i.e., $i < j$, the sequence

   $$FB_i, FB_{i+1}, \ldots, FB_j = FB_i, FB_{i+1}, \ldots, FB_j = FB_i, FB_{i+1}, \ldots$$

   repeats periodically. Since there are only 256 different values for FB, the maximum sequence length is 256. Since each value is associated with a 128 (16 byte) AES output, the keystream sequence s_i has a maximum cycle length of:

   $$128 \times 16 = 2048\text{byte} = 2\text{kbyte}.$$

   After this, the stream cipher output must repeat (and odds are that the cycle lenght is much shorter). Thus, if an attacker has to know at most 2kB of plaintext in order to recover the entire stream cipher output with which he can decrypt all other ciphertext.
3. No, we still only generate a maximum of 256 keystream words of length 16 byte.

Remark: In the chapter on hash functions we will learn about the birthday paradox. This is applicable here too and tells us that the expected length of the sequence is in fact approximately $\sqrt{2}56 = 16$.

5.9
The counter has to encrypt 1 TB of data without repeating itself. This yields an IV of maximum size of $91 = 128 - 36$ bits.

5.11
A missing or deleted bit in y_i affects the i-th feedback bit which enters the shift register of size of κ bit. After $\kappa + 1$ steps, the affected feedback bit leaves the shift register. As a consequence, all subsequent decryptions (i.e., decryptions of $y_{i+\kappa+}$...) are again correct.

5.13
With AES having a block size of 128 bit, a key search for AES-128 requires only a single pair of plaintext-ciphertext in order to identify the correct key. In case of AES-192, given a single pair (x, y) of plaintext and ciphertext, there are $2^{192-128} = 2^{64}$ possible keys k_i satisfying $y = e_{k_i}(x)$. In order to find the correct key with a probability of 50 percent, we require 2^{63} pairs of plaintexts and ciphertexts.

For achieving the same probability with AES-256, 2^{127} plaintexts and ciphertexts are required (which is very very unlikely)!

5.15 $y' = e_{K_3}(e_{K_2}(e_{K_1}(x')))$

1. Pre–compute $e_{K_i}(x') = z_i^{(1)}$; $i = 1,2,\ldots,2^{56}$ and store all pairs $(z_i^{(1)}, K_i)$
2. Decrypt $z_{a,b}^{(2)} = e_{K_b}^{-1}(e_{K_a}^{-1}(y'))$; $a = 1,2,\ldots,2^{56}$; $b = 1,2,\ldots,2^{56}$

   If a match is found, if there is a $z_{a,b}^{(2)} = z_i^{(1)}$ test further key pairs (x'',y''), (x''',y'''), $\ldots$, with the three keys involved in the match:
   If the three keys generate a valid encryption for all pairs, these are most likely the correct keys. Other wise continue with the next pair K_a, K_b.
   $l = 3$; $t = 3$ pairs
   $2^{3\cdot 56 - 3\cdot 64} = 2^{-3\cdot 8} = 2^{-24}$
   $\Rightarrow t = 3$ pairs (x,y) are sufficient

Problems of Chapter 6

6.1 From a theoretical point of view, public key cryptography can be used as a replacement for symmetric cryptography. However, in practical applications, symmetric ciphers tend to be approximately 1000 times faster than public key schemes. Hence, symmetric ciphers are used when it comes to bulk data encryption.

6.3 If every pair out of $n = 120$ employees requires a distinct key, we need in sum

$$n \cdot \frac{n-1}{2} = 120 \cdot \frac{120-1}{2} = 7140$$

key pairs. Remark that each of these key pairs have to be exchanged in a secure way (over a secure channel)!

6.5

1. $\gcd(7469, 2464) = 77$
2. $\gcd(4001, 2689) = 1$

6.7

1. $\gcd(26, 7) = 1$
   $q_1 = 3$, $q_2 = 1$, $q_3 = 2$
   $t_2 = -3$, $t_3 = 4$, $t_4 = -11$
   $a^{-1} \equiv t_4 \bmod m \equiv -11 \bmod 26 = 15$
2. $\gcd(999, 19) = 1$
   $q_1 = 52$, $q_2 = 1$, $q_3 = 1$, $q_4 = 2$, $q_5 = 1$
   $t_2 = -52$, $t_3 = 53$, $t_4 = -105$, $t_5 = 263$, $t_6 = -368$
   $a^{-1} \equiv t_6 \bmod m \equiv -368 \bmod 999 = 631$

6.9

1. $\phi(p) = (p^1 - p^0) = p - 1$
2. $\phi(p \cdot q) = (p-1) \cdot (q-1)$
   $\phi(15) = \phi(3 \cdot 5) = 2 \cdot 4 = 8$
   $\phi(26) = \phi(2 \cdot 13) = 1 \cdot 12 = 12$

6.11

1. $m = 6$; $\phi(6) = (3-1) \cdot (2-1) = 2$;
   Euler's Theorem: $a^2 \equiv 1 \bmod 6$, if $\gcd(a, 6) = 1$
   $0^2 \equiv 0 \bmod 6$;
   $1^2 \equiv 1 \bmod 6$;
   $2^2 \equiv 4 \bmod 6$;

$3^2 \equiv 9 \equiv 3 \bmod 6$;
$4^2 \equiv 16 \equiv 4 \bmod 6$;
$5^2 \equiv 25 \equiv 1 \bmod 6$

2. $m = 9$; $\phi(9) = 3^2 - 3^1 = 9 - 3 = 6$;
Euler's Theorem: $a^6 \equiv 1 \bmod 9$, if $\gcd(a,9) = 1$
$0^6 \equiv 0 \bmod 9$;
$1^6 \equiv 1 \bmod 9$;
$2^6 \equiv 64 \equiv 1 \bmod 9$;
$3^6 \equiv (3^3)^2 \equiv 0^2 \equiv 0 \bmod 9$;
$4^6 \equiv (2^6)^2 \equiv 1^2 \equiv 1 \bmod 9$;
$5^6 \equiv 1 \bmod 9$;
$6^6 \equiv 2^6 \cdot 3^6 \equiv 1 \cdot 0 \equiv 0 \bmod 9$;
$7^6 \equiv 1 \bmod 9$;
$8^6 \equiv 1 \bmod 9$

6.13
Euclid's Algorithm:

Iteration 2: $r_0 = q_1 r_1 + r_2 \qquad r_2 = r_0 - q_1 r_1 = s_2 r_0 + t_2 r_1$ (1)
Iteration 3: $r_1 = q_2 r_2 + r_3 \qquad r_3 = [-q_2] \cdot r_0 + [1 + q_1 q_2] \cdot r1 = s_3 r_0 + t_3 r_1$ (2)

$\Rightarrow$ from (1),(2): $s_2 = 1; \quad s_3 = -q_2$ (3)
$t_2 = -q_1; t_3 = 1 + q_1 q_2$ (4)

The iteration formula for the Euclidean Algorithm gives:

(5) $s_2 \overset{EA}{=} s_0 - q_1 s_1 \overset{(3)}{=} 1$
(6) $s_3 \overset{EA}{=} s_1 - q_2 s_2 \overset{(3)}{=} s_1 - q_2 \overset{(3)}{=} -q_2$
$\overset{(6)}{\Rightarrow} s_1 = 0 \qquad \overset{(5)}{\Rightarrow} s_0 = 1$
(7) $t_2 \overset{EA}{=} t_0 - q_1 t_1 \overset{(4)}{=} -q_1$
(8) $t_3 \overset{EA}{=} t_1 - q_2 t_2 \overset{(4)}{=} t_1 + q_1 q_2 \overset{(4)}{=} 1 + q_1 q_2$
$\overset{(8)}{\Rightarrow} t_1 = 1 \qquad \overset{(7)}{\Rightarrow} t_0 = 0$

Problems of Chapter 7

7.1

1. Only $e = 31$ is a valid public key, because $\Phi(n) = (p-1)(q-1) = 40 \cdot 16 = 640 = 2^7 \cdot 5$. Furthermore $gcd(e_i, \phi(n)) = 1$ has to be fulfilled. Hence, only $e_2 = 49$ may be used as public exponent.
2. $K_{pub} = (n,e) = (697, 49)$
Calculation of $d = e^{-1} \bmod \phi(n) = 49^{-1} \bmod 640$ using EEA:

$$\begin{aligned} 640 &= 13 \cdot 49 + 3 \\ 49 &= 16 \cdot 3 + 1 \\ \Leftrightarrow 1 &= 49 - 16 \cdot 3 \\ &= 49 - 16(640 - 13 \cdot 49) \\ &= 209 \cdot 49 - 16 \cdot 640 \end{aligned}$$

$$\Rightarrow \quad 49^{-1} \bmod 640 \equiv 209.$$

So, the private key is defined by $K_{pr} = (p,q,d) = (41, 17, 209)$.

7.3

1. $e = 3$; $y = 26$
2. $d = 27$; $y = 14$

7.5

1. In this case, a brute-force attack on all possible exponents would be easily feasible.
2. As an absolute minimum, a bit length of 128 bit is recommended in order to preclude brute-force attacks on the private exponent. However, the exponent must even be larger since there exist analytical attacks which are more powerful. In practice, a length for d of least 0.3 times the bit length of n is recommended, i.e. for RSA-2048 the exponent should at least 615 bit.

7.7

$p = 31, q = 37, e = 17, y = 2$

- $n = 31 \cdot 37 = 1147$
  $d = 17^{-1} = 953 \bmod 1080$
- $d_p = 953 \equiv 23 \bmod 30$
  $d_q = 953 \equiv 17 \bmod 36$
- $x_p = y^{d_p} = 2^{23} \equiv 8 \bmod 31$
  $x_q = y^{d_q} = 2^{17} \equiv 18 \bmod 37$
- $c_p = q^{-1} = 37^{-1} \equiv 6^{-1} \equiv 26 \bmod 31$
  $c_q = p^{-1} = 31^{-1} \equiv 6 \bmod 37$
- $x = [qc_p]\,x_p + [pc_q]\,x_q =$
  $[37 \cdot 26]\,8 + [31 \cdot 6]\,18 =$
  $8440 = 721 \bmod 1147$

7.9

Alice		**Bob**
		setup: $k_{pr} = d$; $k_{pub} = e$
		publish e, n
choose random session key k_{ses}		
$y = e_{k_{pub}}(k_{ses}) = k_{ses}^e \bmod n$	$\xrightarrow{\quad y \quad}$	
		$k_{ses} = d_{k_{pr}}(y) = y^d \bmod n$

Alice completely determines the choice of the session key k_{ses}.

Note that in practice k_{ses} might be much longer than needed for a symmmetric-key algorithm. For instance, k_{ses} may have 1024 bits but only 128 actual key bits are needed. In this case just use the 128 MSB (or LSB) bit are used and the remaining bit are discarded. Often, it is safe practice to apply a cryptographic hash function first to k_{ses} and then take the MSB or LSB bits.

7.11

1. Encryption equation: $y \equiv x^e \bmod n$. We cannot solve the equation analytical, because the exponentiation takes place in a finite ring, where no efficient algorithms for computing roots is known.
2. $$\Phi(n) = p \cdot q$$
   No! The calculation of $\Phi(n)$ presumes the knowledge of p and q, which we do not have.
3. Factorization yields: $p = 43$ and $q = 61$
   $\Phi(n) = 42 \cdot 60 = 2520$
   $d \equiv e^{-1} \bmod 2520 \equiv 191$
   $x = 1088$

7.13

1. A message consists of, let's say, m pieces of ciphertext $y_0, y_1, \ldots, y_{m-1}$. However, the plaintext space is restricted to 95 possible values and the ciphertext space too. That means we only have to test 95 possible plaintext characters to build up a table containing all possible ciphertext characters:
   Test: $\quad y_i \stackrel{?}{=} j^e \bmod n;\ j = 32, 33, \ldots, 126$
2. SIMPSONS

3. With OAEP padding a random string *seed* is used with every encryption. Since *seed* has in practice a length of 128–160 bit, there exist many, many different ciphertexts for a given plaintext.

7.15

The basic idea is to represent the exponent in a radix 2^k representation. That means we group k bits of the exponent together. The first step of the algorithm is to pre-compute a look-up table with the values $A^0 = 1, A^1 = A, A^2, \ldots, A^{2^k-1}$. Note that the exponents of the look-up table values represent all possible bit patterns of length k. The table computation requires $2^k - 2$ multiplications (note that computing A^0 and A^1 is for free). After the look-up table has been computed, the two elementary operations in the algorithm are now:

- Shift intermediate exponent by k positions to the left by performing k subsequent squarings (Recall: The standard s-a-m algorithm shifts the exponent only by one position by performing one squaring per iteration.)
- The exponent has now k trailing zeros at the rightmost bit positions. Fill in the required bit pattern for the exponent by multiplying the corresponding value from the look-up table with the intermediate result.

This iteration is only performed l/k times, where $l+1$ is the bit length of the exponent. Hence, there are only l/k multiplications being performed in this part of the algorithm.

An exact description of the algorithm, which is often referred to as *k-ary exponentiation*, is given in [120]. Note that the bit length of the exponent in this description is $t\,k$ bit. An example for the case $k = 3$ is given below.

The complexity of the algorithm for an $l+1$ bit exponent is $2^k - 3$ multiplications in the precomputation phase, and about $l-1$ squarings and $l(2^k-1)/2^k$ multiplications in the main loop.

Example 13.2. The goal is to compute g^e mod n with k-ary where $n = 163$, $g = 12$, $k = 3$, $e = 145_{10} = 221_{8=2^3} = 10\,010\,001_2$

Precomputation:

$g_0 := 1$
$g_1 := 12$
$g_2 := g_1 \cdot 12 = 144$
$g_3 := g_2 \cdot 12 = 1728 \bmod 163 = 98$
$g_4 := g_3 \cdot 12 = 1176 \bmod 163 = 35$
$g_5 := g_4 \cdot 12 = 420 \bmod 163 = 94$
$g_6 := g_5 \cdot 12 = 1128 \bmod 163 = 150$
$g_7 := g_6 \cdot 12 = 1800 \bmod 163 = 7$

Exponentiation:

Iteration	Exponent (base 2)	Calculation	Operation
0	10	$A := g_2 = 144$	TLU
1a	10 000	$A := A^8 \bmod 163 = 47$	3 SQ
1b	10 010	$A := A \cdot g_2 = 6768 \bmod 163 = 85$	MUL
2a	10 010 000	$A := A^8 \bmod 163 = 140$	3 SQ
2b	10 010 001	$A := A \cdot g_1 = 1680 \bmod 163 = 50$	MUL

In each iteration, three squarings results in a left shift which makes space for multiplying by the appropriate precomputed power of g. For instance, if the next binary digits to be processed are $(010)_2 = (\mathbf{2})_{10}$, we take the value $g_{\mathbf{2}} = g^2$ from the look-up-table and multiply it by the intermediate result.

This example emphasizes the impact of the precomputations on the efficiency of the k-ary algorithm: For the small operand lengths used here, the overall cost for the exponentiation is worse than for the s-a-m algorithm. This changes a lot for real-world operands with 1024 or more bits, as the size of the look-up-table only depends on the window size k, and not on the operand length.

◇

Problems of Chapter 8

8.1

1. $\mathbb{Z}_5^*$:

a	1	2	3	4
ord(a)	1	4	4	2

2. $\mathbb{Z}_7^*$:

a	1	2	3	4	5	6
ord(a)	1	3	6	3	6	2

3. $\mathbb{Z}_{13}^*$:

a	1	2	3	4	5	6	7	8	9	10	11	12
ord(a)	1	12	3	6	4	12	12	4	3	6	12	2

8.3

1.
$|\mathbb{Z}_5^*| = 4$
$|\mathbb{Z}_7^*| = 6$
$|\mathbb{Z}_{13}^*| = 12$

2. yes
3.
$\mathbb{Z}_5^* : 2,3$
$\mathbb{Z}_7^* : 3,5$
$\mathbb{Z}_{13}^* : 2,6,7,11$

4.
$\phi(4) = 2$
$\phi(6) = 2$
$\phi(12) = 4$

8.5

1. $K_{pub_A} = 8 \qquad K_{pub_B} = 32 \qquad K_{AB} = 78$
2. $K_{pub_A} = 137 \qquad K_{pub_B} = 84 \qquad K_{AB} = 90$
3. $K_{pub_A} = 394 \qquad K_{pub_B} = 313 \qquad K_{AB} = 206$

8.7

Both values would yield public keys that would immediately allow to recognize the private key. If the private key is equal to 1, the public key would be identical to the primitive element α. If an attacker would detect this identity, he would know that $k_{pr} = 1$. If the private key is equal to $p-1$, the public key would take the value 1 according to Fermat's Little Theorem. If an attacker notices this, he can deduce that $k_{pr} = p-1$.

8.9

1. The order of $a = p-1$ is 2, since

$$a^1 = a = p-1; \quad a^2 = (p-1)^2 \equiv (-1)^2 = 1.$$

2. The subgroup H_a, which is generated by a is defined by $H_a = \{1, p-1\}$ (or equally $H_a = \{1, -1\}$).
3. An attacker could alter the mutually used element a to an element a' of the previously mentioned form, so that it generates a subgroup with only two elements. Hence, the Diffie–Hellman key exchange can only yield in two different key and the attacker only has two test both possibilities to determine the right key.

8.11

Alice		**Oscar**		**Bob**
$\alpha^{a_A} = b_A$	$\xrightarrow{b_A}$	$\alpha^{a_O} = b_O$	$\xrightarrow{b_O}$	
	$\xleftarrow{b_O}$		$\xleftarrow{b_B}$	$\alpha^{a_B} = b_B$
$b_O^{a_A} = \alpha^{a_A \cdot a_O} = K_{AO}$		$b_A^{a_O} = \alpha^{a_A \cdot a_O} = K_{AO}$ $b_B^{a_O} = \alpha^{a_B \cdot a_O} = K_{BO}$		$b_O^{a_B} = \alpha^{a_B \cdot a_O} = K_{BO}$

Oscar shares now a *secret* key with Alice and Bob. Alice and Bob both don't know about it and think they share a key with each other. Oscar can now decrypt, read, and encrypt any messages between Alice and Bob without them learning about it if he continues to intercept all encrypted messages.

This is the infamous *man-in-the-middle* attack. This attack is, in essence, responsible for things such as certificates, public-key infrastructures, etc.

8.13
Compute β: $\beta = \alpha^d \bmod p$.
Encrypt: $(k_E, y) = (\alpha^i \bmod p, x \cdot \beta^i \bmod p)$.
Decrypt: $x = y(k_E^d)^{-1} \bmod p$.

1. $(k_E, y) = (29, 296)$, $x = 33$
2. $(k_E, y) = (125, 301)$, $x = 33$
3. $(k_E, y) = (80, 174)$, $x = 248$
4. $(k_E, y) = (320, 139)$, $x = 248$

8.15
Oscar knows x_n, y_n and n (by just counting the number of ciphertexts). The first step of a possible attack is to calculate

$$k_{M,n} = y_n \cdot x_n^{-1} \bmod p. \tag{13.3}$$

Caused by the previously mentioned PRNG, beginning with $k_{M,n-1}$, $k_{M,j-1}$ can easily calculated recursivley through

$$k_{M,j-1} = \beta^{i_{j-1}} = \beta^{i_j - f(j)} = \beta^{i_j} \cdot \beta^{-f(j)} = k_{M,j-1} \cdot \beta^{-f(j)} \bmod p \tag{13.4}$$

where the values of all variables are known. With the knowledge of $k_{M,j}$ for all j, Oscar is now able to decrypt the whole ciphertext by solving the usual decryption equation

$$x_j = y_j \cdot k_{M,j}^{-1} \bmod p \tag{13.5}$$

8.17

1. By choosing a different secret exponent i, the ciphertext y of the same plaintext x is different everytime. Even if a pair of plaintext/ciphertext is compromised, such a pair will most likely not repeat a second time in a non-deterministic encryption scheme!
2. In general, there are $\#\{2, 3, \cdots, p-2\} = p-3$ different valid ciphertexts for a single plaintext. I.e., we have 464 different possibilities for $p = 467$.
3. The plain RSA cryptosystem is deterministic. A specific plaintext always yields the same ciphertext assuming the same public parameters.

Problems of Chapter 9

9.1 $a = 2$, $b = 2$
$4 \cdot 2^3 + 27 \cdot 2^2 = 4 \cdot 8 + 27 \cdot 4 = 32 + 108 = 140 \equiv 4 \neq 0 \bmod 17$

9.3 $17 + 1 - 2\sqrt{17} \approx 9,75 \leq 19 \leq 17 + 1 + 2\sqrt{17} \approx 26,25$ q.e.d.

9.5

1. The points of E are

$$\{(0,3), (0,4), (2,3), (2,4), (4,1), (4,6), (5,3), (5,4)\}$$

2. The group order is given by

$$\#G = \#\{O, (0,3), (0,4), (2,3), (2,4), (4,1), (4,6), (5,3), (5,4)\} = 9$$

3. Compute all multiples of α:

$$\begin{aligned}
0\cdot\alpha &= O\\
1\cdot\alpha &= (0,3)\\
2\cdot\alpha &= (2,3)\\
3\cdot\alpha &= (5,4)\\
4\cdot\alpha &= (4,6)\\
5\cdot\alpha &= (4,1)\\
6\cdot\alpha &= (5,3)\\
7\cdot\alpha &= (2,4)\\
8\cdot\alpha &= (0,4)\\
9\cdot\alpha &= O = 0\cdot\alpha\\
&\Rightarrow \text{ord}(\alpha) = 9 = \#G\\
&\Rightarrow \alpha \text{ is primitive since it generates the group!}
\end{aligned}$$

9.7

1. $9\cdot P = (1001_{|2})P = (2\cdot(2\cdot(2\cdot P)))+P = (4,10)$
2. $20\cdot P = (10100_{|2})P = (2\cdot(2\cdot(2\cdot(2\cdot P)+P))) = (19,13)$

9.9

$K = aB = 6\cdot B = 2(2B+B)$

$2B = (x_3,y_3) : x_1 = x_2 = 5; y_1 = y_2 = 9$
$s = (3x_1^2+a)\cdot y_1^{-1} = (3\cdot 25+1)(2\cdot 9)^{-1} = 76\cdot 18^{-1} \bmod 11$
$s \equiv 10\cdot 8 = 80 \equiv 3 \bmod 11$
$x_3 = s^2 - x_1 - x_2 = 3^2 - 10 = -1 \equiv 10 \bmod 11$
$y_3 = s(x_1 - x_3) - y_1 = 3(5-10) - 9 = -15 - 9 = -24 \equiv 9 \bmod 11$
$2B = (10,9)$

$3B = 2B + B = (x_3',y_3') : x_1 = 10, x_2 = 5, y_1 = 9, y_2 = 9$
$s = (y_2 - y_1)(x_2 - x_1)^{-1} = 0 \bmod 11$
$x_3' = 0 - x_1 - x_2 = -15 \equiv 7 \bmod 11$
$y_3' = s(x_1 - x_3) - y_1 = -y_1 = -9 \equiv 2 \bmod 11$
$3B = (7,2)$

$6B = 2\cdot 3B = (x_3'',y_3'') : x_1 = x_2 = 7, y_1 = y_2 = 2$
$s = (3x_1^2+a)\cdot y_1^{-1} = (3\cdot 49+1)\cdot 4^{-1} \equiv 5\cdot 4^{-1} \equiv 5\cdot 3 = 15 \equiv 4 \bmod 11$
$x_3'' = s^2 - x_1 - x_2 = 4^2 - 14 = 16 - 14 = 2 \bmod 11$
$y_3'' = s(x_1 - x_3) - y_1 = 4(7-2) - 2 = 20 - 2 = 18 \equiv 7 \bmod 11$
$6B = (2,7) \Rightarrow K_{AB} = 2$

9.11

A brute-force attack on a 128 bit key currently is computitional infeasible!
In this context, a much more efficient attack is to make use of the correlation between the $x-$ and $y-$ coordinate of a point. Since it is known that there is an inverse for every point $P = (x,y)$ with $-P = (x,-y)$, it would be the easiest approach to test all 2^{64} possible $x-$coordinates by solving the curve equation. The effective key length is then reduced to 65 bit, which may be insufficient in a few years (if this problem has not already been broken by well-funded intelligence services).

Problems of Chapter 10

10.1

1. If a message from Alice to Bob is found to be authentic, i.e., in fact originated from Alice, integr ity is automatically assured, since an alteration by Oscar would make *him* the originator. However, this can't be the case if sender authenticity is assured.
2. No, a message can still be unaltered but message authenticity is not given. For instance, Oscar coul d maskerade as Alice and send a message to Bob saying that it is from Alice. Although the message ar rives unaltered at Bob's (integrity is thus assured) sender authenticity is not given.

10.3
Threats:

- Unauthorized physical access to the building, to the data, to the internal network.
- Social engineering.
- Data might be modified, secret data might be read (e.g. be personnel or remote via the network).
- Key generation might be predictable or weak.
- Key encryption might be weak.
- Integrity measures used might be weak.
- Data might be lost.
- Trust by the users might be lost.

Organization and physical measures:

- Physical access control to the building (e.g., guards).
- Training and guidelines for the personnel.
- Secure backup procedures.

IT security functions:

- Key generation (random number generator of 'good' quality).
- Key distribution (encrypting+securing for data integrity of the session key).
- Access control of the network / firewall.
- Timestamp service.

10.5

1. $6292^b \equiv x \bmod n \Rightarrow$ valid signature
2. $4768^b \neq x \bmod n \Rightarrow$ invalid signature
3. $1424^b \equiv x \bmod n \Rightarrow$ valid signature

10.7
Oscar recieves the message, alters it and signs it with his own private key a'. Then he sends the new message together with the signature and the putatively approriate public key (n', e') of Alice (which is instead the one of Oscar).

10.9

1. $sig_{K_{pr}}(x) = x^d \bmod n = y$

   $ver_{K_{pub}}(x,y)$: $\quad x \stackrel{?}{\equiv} y^e \bmod n$

   Assume that d has l bits.

   Using the square & multiply algorithm, the average signing will take:

   $\#\otimes \approx l$ squarings + $\frac{1}{2} \cdot l$ multiplications = $\underline{\frac{3}{2} \cdot l}$

   Since $b = 2^{16} + 1 \equiv 65537 \equiv 1000000000000001_2$, the average verification takes:

   $\#\otimes = 16\ squarings + 1\ multiplication = 17$

2. Signing takes longer than verification.
3.

l [bits]	T_{unit}	n	$\frac{T_{unit}}{operation}$	$T(sig)$	$T(ver)$
512	100 ns	16	25.6 μs	19.7 ms	435.2 μs
1024	100 ns	32	102.4 μs	157.3 ms	1.741 ms

4. $T(1\otimes) = (\frac{l}{8})^2 \cdot \frac{1}{f}$; $\quad$ time for one multiplication modulo p

$$T(sig) = \frac{3}{2} \cdot l \cdot \frac{T_{unit}}{operation} = 0.5\ s$$

$$\frac{T_{unit}}{operation} = \frac{l}{8}^2 \cdot T_{unit}$$

$$F \geq \frac{1}{T_{unit}}[Hz]$$

i) $\underline{6.29\,MHz}$
ii) $\underline{50.33\,MHz}$

10.11

1. $\alpha^x = 3^{10} \equiv 25 \bmod 31$
   a. $\gamma = 17$, $\delta = 5$
   $t = \beta^\gamma \cdot \gamma^\delta = 6^{17} \cdot 17^5 \equiv 26 \cdot 26 \equiv 25 \bmod 31 \Rightarrow t = \alpha^x \Rightarrow ver(x,(\gamma,\delta)) = 1$ (ok)
   b. $\gamma = 13$, $\delta = 15$
   $t = \beta^\gamma \cdot \gamma^\delta = 6^{13} \cdot 13^{15} \equiv 6 \cdot 30 \equiv 25 \bmod 31 \Rightarrow t = \alpha^x \Rightarrow ver(x,(\gamma,\delta)) = 1$ (ok)
2. Due to the fact that the Elgamal signature scheme is probabilistic, there are $p-1$, i.e. 30, different signatures for each message x.

10.13

$$\begin{aligned}
s_1 &\equiv (x_1 - d\,r)\,k_{E_1}^{-1} \bmod p-1 \\
s_2 &\equiv (x_2 - d\,r)\,k_{E_2}^{-1} = (x_2 - d\,r)\,(k_{E_1}+1)^{-1} \bmod p-1 \\
\Rightarrow \frac{s_1}{s_2} &\equiv \frac{(x_1 - d\,r)\,(k_{E_1}+1)}{(x_2 - d\,r)\,k_{E_1}} \bmod p-1 \\
\Leftrightarrow k_{E_1} &= \frac{1}{\frac{s_1\,(x_2 - d\,r)}{s_2\,(x_1 - d\,r)} - 1} \bmod p-1 \\
\Rightarrow d &\equiv \frac{x_1 - s_1\,k_{E_1}}{r} \bmod p-1
\end{aligned}$$

10.15 Similarly to the attack on Elgamal, an attacker can use following system of equations

$$\begin{aligned}
s_1 &\equiv (SHA(x_1) + d\,r)\,k_E^{-1} \bmod q \\
s_2 &\equiv (SHA(x_2) + d\,r)\,k_E^{-1} \bmod q
\end{aligned}$$

for known s_1, s_2, x_1, and x_2 to first compute the ephemeral key k_E and then the private key d:

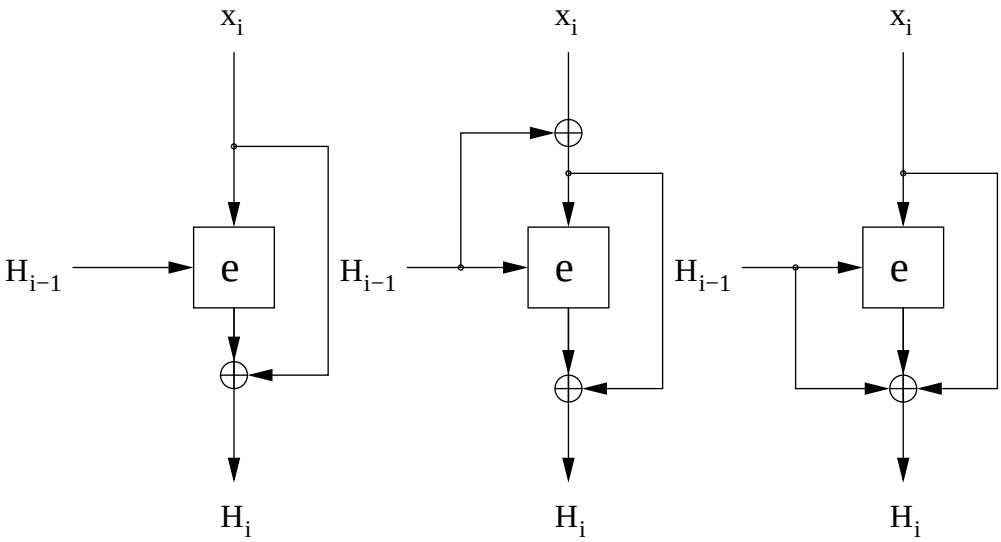
$$\begin{aligned}
s_1 - s_2 &\equiv k_E^{-1}(SHA(x_1) - SHA(x_2)) \bmod q \\
\Leftrightarrow k_E &\equiv \frac{SHA(x_1) - SHA(x_2)}{s_1 - s_2} \bmod q \\
\Rightarrow d &\equiv \frac{s_1 \cdot k_E - SHA(x_1)}{r} \bmod q
\end{aligned}$$

Problems of Chapter 11

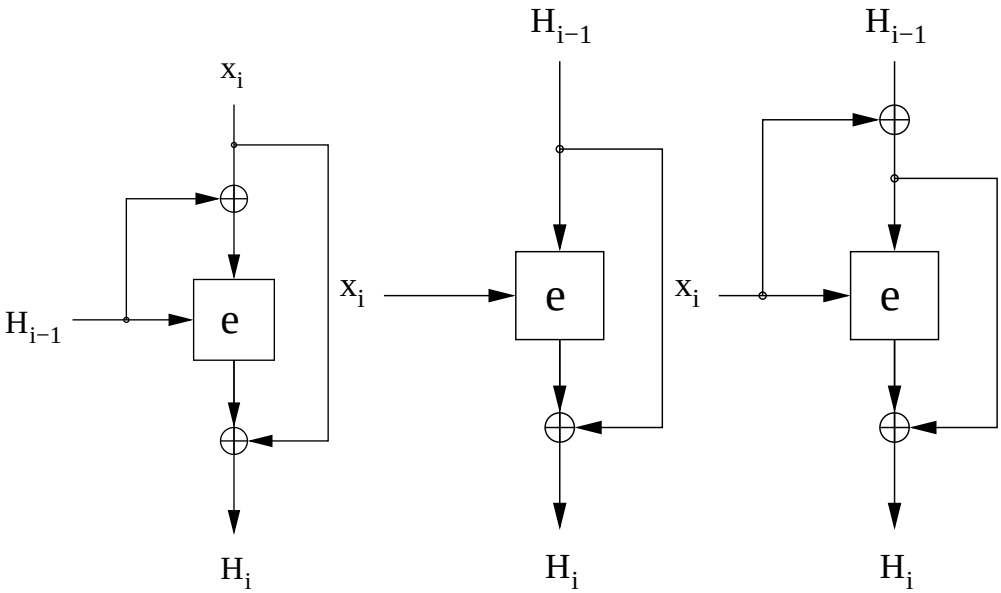
11.1

1. $A_1 = E_0 + f_1(B_0,C_0,D_0) + (A)_{<<<5} + W_j + K_t = 5A827999_{hex}\, B_1 = A_0 = 00000000_{hex}\, C_1 = (B_0)_{<<<30} = 00000000_{hex}\, D_1 = C_0 = 00000000_{hex}\, E_1 = D_0 = 00000000_{hex}$
2. $A_1 = E_0 + f_1(B_0,C_0,D_0) + (A)_{<<<5} + W_j + K_t = 6A827999_{hex}\, B_1 = A_0 = 00000000_{hex}\, C_1 = (B_0)_{<<<30} = 00000000_{hex}\, D_1 = C_0 = 00000000_{hex}\, E_1 = D_0 = 00000000_{hex}$

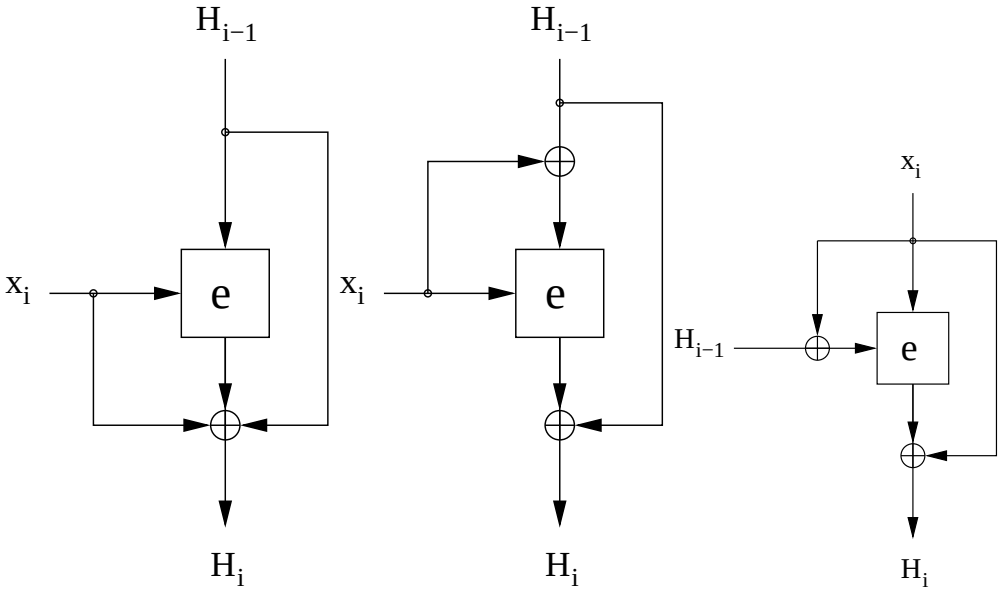
11.3



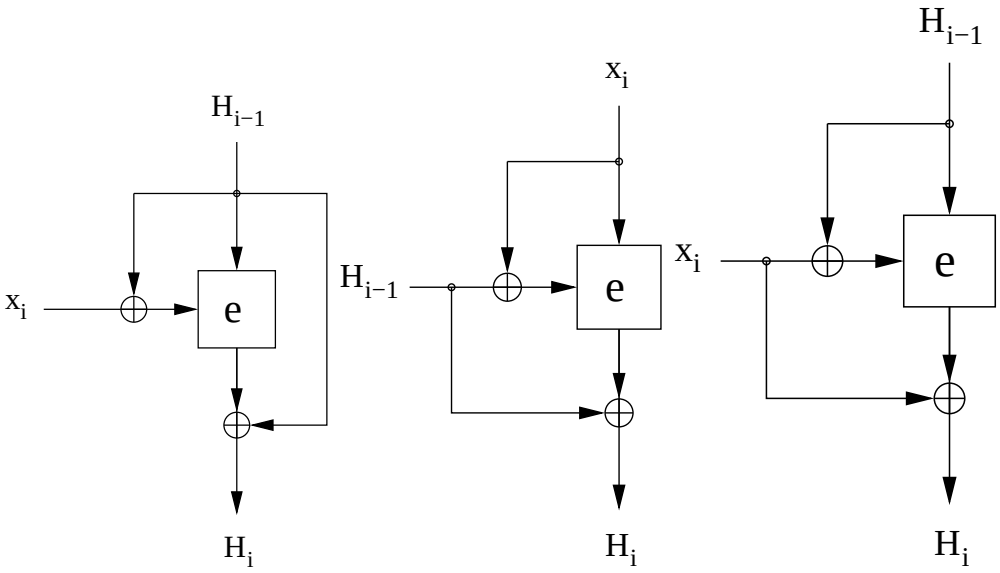
(a) $e(H_{i-1}, x_i) \oplus x_i$ (b) $e(H_{i-1}, x_i \oplus H_{i-1}) \oplus x_i \oplus H_{i-1}$ (c) $e(H_{i-1}, x_i) \oplus x_i \oplus H_{i-1}$



(d) $e(H_{i-1}, x_i \oplus H_{i-1}) \oplus x_i$ (e) $e(x_i, H_{i-1}) \oplus H_{i-1}$ (f) $e(x_i, x_i \oplus H_{i-1}) \oplus x_i \oplus H_{i-1}$



(g) $e(x_i, H_{i-1}) \oplus x_i \oplus H_{i-1}$ (h) $e(x_i, x_i \oplus H_{i-1}) \oplus H_{i-1}$ (i) $e(x_i \oplus H_{i-1}, x_i) \oplus x_i$



(j) $e(x_i \oplus H_{i-1}, H_{i-1}) \oplus H_{i-1}$ (k) $e(x_i \oplus H_{i-1}, x_i) \oplus H_{i-1}$ (l) $e(x_i \oplus H_{i-1}, H_{i-1}) \oplus x_i$

11.5

Birthday attack: $k \approx \sqrt{n \cdot m \cdot \frac{1}{1-\varepsilon}}$

n	$\varepsilon = 0.5$	$\varepsilon = 0.1$
2^{64}	$3.6 \cdot 10^9$	$1.4 \cdot 10^9$
2^{128}	$1.5 \cdot 10^{19}$	$6.0 \cdot 10^{18}$
2^{160}	$1.5 \cdot 10^{24}$	$3.9 \cdot 10^{23}$

number of messages after which probability for collision is ε

11.7

1. 128 bit
2. If $c = 0$, both halfs of the output compute exactly the same 64 bit value. Hence, even though y_U has 128 bit, it only has an entropy of 64 bit. You, as an attacker, simply provide $(H_{0,L}, H_{0,R})$ and some start value for x_i (e..g., 64 zeros) as input to the hash function. You now search through possible

passwords by incrementing x_i. This way, you will generate pseudo-random outputs y. Even though there is a chance you will not generate y_U at the output, the likelihood is small. Note that you can also try values x_i which have more than 64 bit by iterating the hash function.
3. A second-preimage attack
4. When $c \neq 0$ both halfs of the output will almost never be the same. So, the entropy of the output grows to (round about) 128 bit which makes a second-preimage attack computational infeasible.

Problems of Chapter 12

12.1

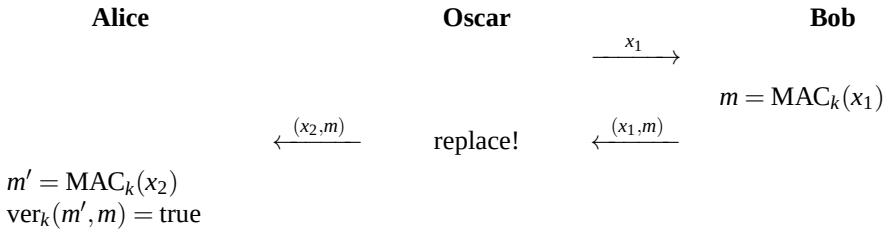
1. ■ Calculate $x||h = e_{k_1}^{-1}(y)$.
   ■ Calculate $h' = H(k_2||x)$.
   ■ If $h = h'$, the message is authentic. If $h \neq h'$, either the message or the MAC (or both) has been altered during transfer.
2. ■ Calculate $x||s = e_{k_1}^{-1}(y)$.
   ■ Calculate $h' = H(x)$.
   ■ Verify the signature: $ver_{k_{pub}}(s, H(x))$

12.3

1. $c_i = z_i \bigoplus \{x_1 x_2 \ldots x_n || H_1(x) H_2(x) \ldots H_m(x)\}; \qquad i = 1,2,\ldots,n+m$
   1) Assume x has n bits. Oscar first computes
   $z_i = x_i \oplus c_i; \qquad i = 1,2,\ldots,n$
   2) Oscar recomputes $H(x)$ since he knows x.
   3) Assume $H(x)$ has m output bits. Oscar computes
   $z_{j+n} = H_j(x) \oplus c_{j+n} \qquad j = 1,2,\ldots,m$
   4) Oscar computes $H(x')$
   5) Oscar computes
   $c'_i = z_i \oplus x'_i \qquad i = 1,2,\ldots,n$
   $c'_{j+n} = z_{j+n} \oplus H_j(x') \qquad j = 1,2,\ldots,m$
2. No. Although Oscar can still recover $z_1, z_2, \ldots, z_n$, he can not recover the bit-stream portion $z_{n+1}, z_{n+2}, \ldots, z_{n+m}$ which was used for encrypting $MAC_{k_2}(x)$. Even if he would know the whole bit-stream, he would not be able to compute a valid $MAC_{k_2}(x')$ since he does not know k_2.

12.5

1. This attack assumes that Oscar can trick Bob into signing the message x_1. This is, of course, not possible in every situation, but one can imagine scenarios where Oscar can pose as an innocent party and x_1 is the message being generated by Oscar.



2. For constructing collisions, Oscar must be able to compute about $\sqrt{2^n}$ MACs, where n is the output width. Since Oscar does not have the secret key, he has to somehow trick Alice and/or Bob into computing MACs for that many messages. This is in practice often impossible. On the other hand, collisions for hash functions can be constructed by Oscar by himself without the help of Alice and Bob because these computations are un-keyed.
   An 80 bit MAC provides thus a security of 2^{80} since collision attacks are not applicable. A hash function with the same output size offers only a security of about 2^{40}.

Problems of Chapter 13

13.1

1. (1) Session keys are derived by a linear and invertible(!) operation of the previous session key.
   (2) Usage of hash functions, thus a non-linear correlation of the session keys.
   (3) Usage of the masterkey and the previous session key for every derivation of the next session key.
2. Methods (2) and (3), since the old session keys cannot be extracted from the recent session key
3. (1) every session, since PFS is missing
   (2) every session using the hacked session key K_n and every following session
   (3) only the recent session, since the (unknown) masterkey is used for every furterh key derivation
4. No, since then, all session keys can be calculated!

13.3

The first class of encryptions (between the KDC and a user) should be done using 3DES. The session between two arbitrary users (the second class) should be encrypted using DES. There are two major reasons for this choice:

- A brute-force attack against DES is already possible and affordable even for non-govermental organizations. If a session key is compromised, only the corresponding session will be involved. Though, if it were possible to find a certain $K_{U,KDC}$, all previous and future communication of the user U could be eavesdropped. This being the case and since there are no known attacks against 3DES with an affordable complexicity, 3DES should be used for encrypting the session keys.
- The amount of plaintext to be encrypted vary widely for both classes:
  In the first case, only one session key consisting of a few bytes has to be encrypted, while a session (i.e. the second class) may consist of a vast amount of data, e.g. multiple gigabytes for video conferences. Since 3DES is round about three times slower than DES, it would seem the thing to use the faster DES algorithm for the session data and the slower (but more secure) 3DES for the session key encryption.

13.5

Assuming that the hacker obtains the key $K^i_{U,KDC}$, he can initially encrypt recent session data, where the session keys K_{ses} are encrypted with $K^i_{U,KDC}$. He will also be able to decrypt the following keys $K^{i+j}_{U,KDC}$ until the attack is detected (t_y) and new keys are exchanged using a secure channel. Hence, all communication between t_x and t_y may be compromised. Though, he is, even with knowledge of $K^i_{U,KDC}$, not able to recover $K^{i-1}_{U,KDC}$. Hence, he cannot decrypt messages before the point of time t_x. In conclusion, this variant provides Perfect Forward Secrecy.

13.7

1. Once Alice's KEK k_A is being compromised, Oscar can compute the session key k_{ses} and, thus, decrypt all messages.
2. The same applies to a compromised KEK k_B of Bob.

13.9

1. $t = 10^6 bits/sec$
   $storage = t \cdot r = 2h \cdot 10^6\ bits/sec = 2 \cdot 3600 \cdot 10^6 bits/sec = 7.2\ Gbits = 0.9\ GByte$
   Storage of less than 1 GByte can be done at moderate costs, e.g., on hard disks or CDs.
2. Compute # keys that an attacker can recover in 30 days:
   $\#\ Keys = \frac{30\ days}{10\ min} = \frac{30 \cdot 24 \cdot 60}{10} = 4320$
   Key derivation period:
   $T_{Kder} = \frac{2h}{4320} = 1.67\ sec$
   Since hash functions are fast, a key derivation can easily be performed (in software) at such a rate.

13.11

- Alice: $A = 2^{228} \equiv 394 \bmod 467$
  $k_{AO} = O^a = 156^{228} \equiv 243 \bmod 467$

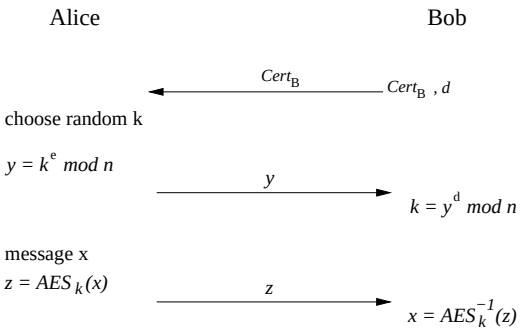
- Bob: $B = 2^{57} \equiv 313 \bmod 467$
  $k_{BO} = O^b = 156^{57} \equiv 438 \bmod 467$
- Oscar: $O = 2^{16} \equiv 156 \bmod 467$
  $k_{AO} = A^o = 394^{16} \equiv 243 \bmod 467$, $k_{BO} = B^o = 313^{16} \equiv 438 \bmod 467$,

13.13

1. Alice would detect this forgery, since the certificate $C(O)$ is cryptographically bound to the ID of Oscar and not to ID(A), which she expected.
2. This kind of forgery would be detected by validating the CA's signature of the public key, which will naturally fail.

13.15
The signature of the CA merely covers the *public* key of a user $k_{pub_i} = \alpha^{a_i}$. Even if all parameters of the signature algorithm are announced, the private key will remain incalculable (DL-problem!). I.e., Oscar cannot calculate the session keys which were used before he recognized the CA's signature algorithm and key. However, he is now capable of passing of himself as any user affiliated to the domain by providing counterfeited certificates.

13.17



13.19 PGP makes use of the so called *Web of Trust*-Architecture. In contradictance of tree based architectures, a WoT only consists of equal nodes (persons) where each node is eligible to certify each other. The validity of acertificate is then confirmed through a so called Chain of Trust. In principle, Alice trusts in certificates which she created herself e.g. for Bob. Then she also trusts in a certificate Bob created and so on. The main advantage of this system is the lack of arbitrary trusted CAs with the drawback of partially long and complicated Chains of Trust.