

Aufgabe 10

(a) Cleartext	1	1	1	0	1	1	1	0	1	0	0	1
Ciphertext	1	1	1	1	1	0	0	1	1	0	1	0

Die Lösung wurde mittels Aufrufs von

```
print "10 (a) : ", join (" ", e_ecb ([2,3,1],
    [1,1,1, 0,1,1, 1,0,1, 0,0,1])), "\n";
```

ermittelt (Code für alle Aufgaben im Anhang)

(b) Ciphertext	1	1	1	0	1	1	1	0	1	0	0	1
Cleartext	0	0	0	0	1	0	1	0	1	0	0	1

Die Lösung wurde mittels Aufrufs von

```
print "10 (b) : ", join (" ", d_cbc ([2,3,1], [1,1,1],
    [1,1,1, 0,1,1, 1,0,1, 0,0,1])), "\n";
```

ermittelt, wobei der verwendete *IV* rechnerisch aus dem gegebenen Fragments des Klartexts 000 ermittelt werden konnte :

Cleartext x	0	0	0	0	1	0	1	0	1	0	0	1
Ciphertext y	1	1	1	0	1	1	1	0	1	0	0	1
π^{-1}	1	1	1	1	0	1	1	1	0	1	0	0
Block-Key	1	1	1	1	1	1	0	1	1	1	0	1

(Der Block-Key zum ersten Block ist der *IV*, welcher durch $\pi^{-1}(y_1) \otimes x_1$ errechnet werden kann; alle weiteren Block-Keys sind der jeweilige Ciphertext des vorherigen Blocks).

Aufgabe 11

Das gesuchte Wort ist 'Ja'. Das Ergebnis wurde per Brute Force gefunden; es sind ja letztenendes nur 5 Belegungen für $z_1 z_2 z_3 z_4$ möglich : 0011, 0101, 1001, 1010, 1100; von diesen ergibt nur 1010 einen sinnvollen ASCII-String.

Dekodiert wurde mittels

```
print "11 : ", asciifi (d_synchro ([1,0,1,0],
    [1,1,1,0,0,1,0,1, 0,1,1,1,0,0,1,0])), "\n";
```

Aufgabe 12

Auch diese Aufgabe wurde wie gefordert via Brute Force-Attacke gelöst; da bei der Schlüsselwortlänge $m = 1$ nur 26 Schlüssel in Frage kommen (A, \dots, Z), ging dies auch relativ schnell mittels

```
force_autokey (split / */, 'PWNAULMZRTXRJEZ');
```

Von den 26 möglichen Cleartexts ergab nur der über den Schlüssel 'T' entschlüsselte in der Deutschen Sprache einen Sinn : 'wannheiratenwir'

Aufgabe 13

$$(a) \det \begin{pmatrix} 2 & 5 \\ 9 & 5 \end{pmatrix} = 2 \cdot 5 - 5 \cdot 9 = -35 \equiv 17 \pmod{26}$$

Da $ggT(17, 26) = 1$ und damit die Determinante der Matrix in \mathbb{Z}_{26} zu m teilerfremd ist, existiert ein Inverses : TODO

$$(b) \det \begin{pmatrix} 1 & 11 & 12 \\ 4 & 23 & 2 \\ 17 & 15 & 9 \end{pmatrix} = 1 \cdot 23 \cdot 9 + 11 \cdot 2 \cdot 17 + 12 \cdot 4 \cdot 15 - 12 \cdot 23 \cdot 17 - 1 \cdot 2 \cdot 15 - 11 \cdot 4 \cdot 9 = 207 + 374 + 720 - 4692 - 30 - 396 = -3817 \equiv 5 \pmod{26}$$

Da $ggT(5, 26) = 1$ und damit die Determinante der Matrix in \mathbb{Z}_{26} zu m teilerfremd ist, existiert ein Inverses : TODO

Anhang

```
#!/usr/local/bin/perl -w
use strict;

sub permutation {
    my @pix = @{shift()}; my @x = @{shift()};
    $_[$_] = $x[$pix[$_]-1] for 0..scalar @pix-1; @_
}

sub e_ecb {
    my @pix = @{shift()}; my @x = @{shift()};
    scalar @x % scalar @pix and
        push @x, '0' for 0..scalar @x % scalar @pix;
    while (my @t = splice @x, 0, scalar @pix) {
        push @_, permutation (\@pix, \@t);
    }
    @_
}

sub d_ecb {
    e_ecb ([reversepi (@{shift()})], shift)
}

sub shiftmod {
    my @a = @{shift()}; my @b = @{shift()};
    $_[$_] = ($a[$_] + $b[$_]) % 2 for 0..scalar @a-1;
}
```

```
@_
}

sub reversepi {
  my @a = splice @_, 0, scalar @_;
  $_[$_a[$_]-1] = $_+1 for 0..scalar @a-1;
  @_
}

sub e_cbc {
  my @pix = @{}; my @iv = @{}; my @x = @{};
  scalar @x % scalar @pix and
  push @x, '0' for 0..scalar @x % scalar @pix;
  scalar @pix != scalar @iv and die "painful death !";
  while (my @t = splice @x,0,scalar @pix) {
    @iv = permutation (\@pix, [shiftmod(\@iv, \@t)] );
    push @_, @iv;
  }
  @_
}

sub d_cbc {
  my @pix = @{}; my @iv = @{}; my @x = @{};
  scalar @x % scalar @pix and
  push @x, '0' for 0..scalar @x % scalar @pix;
  scalar @pix != scalar @iv and die "painful death !";
  while (my @t = splice @x,0,scalar @pix) {
    push @_, shiftmod ([permutation ([reversepi (@pix)], \@t)], \@iv);
    @iv = @t;
  }
  @_
}

sub asciifi {
  my $r; $r .= chr (oct ('0b'.join '', splice @_, 0, 8))
  for 0..int(@_/8); $r
}

sub d_synchro {
  my @key = @{}; my @c = @{};
  $key[$_+4] = ($key[$_]+$key[$_+1])%2 for 0..@c-5;
  return shiftmod ([@key], [@c]);
}

sub shiftCBA {
  my $a = ord ($_ [0]) - ord ($_ [1]);
  ($a < 0) and $a += 26;
  chr ($a+65);
}
```

```
sub force_autokey {
  my @c = @_;
  for my $r ('A'..'Z') {
    print $r."": "; print lc ($r = shiftCBA ($c[$_], $r)) for 0..@c-1;
    print "\n";
  }
}
```