

## Aufgabe 28

$$\begin{array}{lclclcl}
 \text{an} & \rightarrow & (0, 13) & \rightarrow & x_1 & = & 0 \cdot 26 + 13 & = & 13 \\
 \text{ru} & \rightarrow & (17, 20) & \rightarrow & x_2 & = & 17 \cdot 26 + 20 & = & 462 \\
 \text{fu} & \rightarrow & (5, 20) & \rightarrow & x_3 & = & 5 \cdot 26 + 20 & = & 150 \\
 \text{me} & \rightarrow & (12, 4) & \rightarrow & x_4 & = & 12 \cdot 26 + 4 & = & 316 \\
 \text{lf} & \rightarrow & (11, 5) & \rightarrow & x_5 & = & 11 \cdot 26 + 5 & = & 291
 \end{array}$$

$$\begin{array}{lclclclcl}
 y_1 & = & 13^{3533} \bmod 11413 & = & 9919 & \rightarrow & 14 \cdot 26^2 + 17 \cdot 26 + 13 & \rightarrow & \text{ORN} \\
 y_2 & = & 462^{3533} \bmod 11413 & = & 7350 & \rightarrow & 10 \cdot 26^2 + 22 \cdot 26 + 18 & \rightarrow & \text{KWS} \\
 y_3 & = & 150^{3533} \bmod 11413 & = & 8211 & \rightarrow & 12 \cdot 26^2 + 3 \cdot 26 + 21 & \rightarrow & \text{MDV} \\
 y_4 & = & 316^{3533} \bmod 11413 & = & 3253 & \rightarrow & 4 \cdot 26^2 + 21 \cdot 26 + 3 & \rightarrow & \text{EVD} \\
 y_5 & = & 291^{3533} \bmod 11413 & = & 2641 & \rightarrow & 3 \cdot 26^2 + 23 \cdot 26 + 15 & \rightarrow & \text{DXP}
 \end{array}$$

Chiffretext : **ORN KWS MDV EVD DXP**

Diese Ausgabe wurde mittels Perl-Skript ermittelt :

```
./uebung8.pl 28
```

## Aufgabe 29

- (a) Es existieren 237 solcher Zeugen; einer hiervon ist 3.  
 (b) Es existieren 54 solcher Zeugen; einer hiervon ist 2.

Wir haben zur Lösung dieses Problems ein Perl-Programm implementiert :

```
./uebung8.pl 29
```

## Aufgabe 30

In dieser Tabelle werden die Werte immer jeweils vor Anfang der Schleife angegeben. Es wurde zufällig  $x = 5$  gewählt.

$n$	$i$	$k$	$x$	$y$	$d$
1111	1	2	5	5	n/a
1111	2	4	24	24	1
1111	3	4	575	24	1
1111	4	8	657	657	1
1111	5	8	580	657	1
1111	6	8	877	657	11

Somit ist der Faktor 11 nach 5 Schleifendurchläufen gefunden. (der  $ggT$ -Schritt ist hier nicht genauer aufgeschlüsselt; die Rechnung ist einfach mittels EUCLID zu bewerkstelligen).

Da hier eine manuelle Lösung gefordert war hierzu diesmal leider kein Perl-Skript.

## Aufgabe 31

Mit den in der Aufgabe beschriebenen Eigenschaften gilt :

$$x_1^2 \cdot x_2^2 \equiv c^2 \pmod{n} \text{ (da } x_1^2 \equiv c \pmod{n} \text{ und } x_2^2 \equiv c \pmod{n}\text{)}$$

und damit auch

$$(x_1 \cdot x_2)^2 \equiv c^2 \pmod{n}$$

also

$$(5827689 \cdot 6439273)^2 \equiv 5827689^2 \pmod{10910563}$$

Reduzieren innerhalb der Klammer (wie im Skript) ergibt

$$6373259^2 \equiv 5827689^2 \pmod{10910563}$$

Nun ist es mittels  $ggT(6373259 - 5827689, 10910563)$  möglich, einen Faktor von  $n$  auszurechnen (mittels EUCLID, wie z.B. auf Zettel 2 auch in Perl implementiert), in diesem Fall 2389. Somit sind die beiden Faktoren von 10910563  $10910563 : 2389 = 4567$  und 2389, beide prim.

## Aufgabe 32

Wir haben hierfür eine Brute-Force-Methode entwickelt. Diese testet zunächst, ob  $witness(2, n)$  und  $witness(3, n)$  beide FALSE liefern; ist dies der Fall wird geprüft, ob  $witness(5, n)$  TRUE liefert; ist dies der Fall, erfüllt  $n$  die gestellten Anforderungen. Ist dies nicht der Fall wird mit  $n + 1$  fortgefahren. ( $n \in \mathbb{N}, n > 12$ ).

Ausgabe des Skripts :

**1373653**

`./uebung8.pl 32`

## Anhang

```
#!/usr/local/bin/perl -w
use strict; $|=1; no strict 'refs'; use POSIX; use Math::BigInt;

sub prob28 {
    sub toverbose {
        my $b = @_;
        '$'. join ('+', map { $b-- ; $_ . ($b == 0 ? '' : '\cdot 26' .
            ($b == 1 ? '' : '^' . $b)); } @_) . '$'
    }

    sub e_rsa {
        my ($n, $e, $bl) = splice @_, 0, 3;
        my @ptBs = split //, shift; my @ctBs;
```

```

print "\\begin{tabular}{lclclclcl}\n";
for my $i (0..@ptBs/$bl-1) {
    my @b = map { print $_; ord($_) - ord('a')} splice @ptBs, 0, $bl;
    print '&$$\rightarrow&'; my $b = @b;
    print '$(', join(',', @b), ')$&$$\rightarrow&$$x_', $i+1, '$ &$$=$& ',
        toverbose(@b);
    my $blockVal; $blockVal += 26**(@b-1-$_) * $b[$_] for 0..@b-1;
    push @ptBs, $blockVal; print '&=&', $blockVal, "\\\\\n";
}
print "\\end{tabular}\n";
print "\\begin{tabular}{lcr crcl}\n";
for (0..@ptBs-1) {
    my $t = $ptBs[$_]**$e%n;
    print '$y_{', $_+1, '}$&$$=$&$', $ptBs[$_], '^{' , $e, '}' \mod ', $n,
        '$&$$=$&$', $t, ' &$$\rightarrow& ' ;
    @_ = ();
    while ($t) {
        my $l = $t % 26; $t /= 26;
        push @_, $l
    }
    @_ = reverse @_;
    my $ct = join'', map { uc chr $_ + ord 'a' } @_;
    print toverbose(@_), ' &$$\rightarrow& ', $ct, "\\\\\n";
    push @ctBs, $ct
}
print "\\end{tabular}\n\n";
return join'', @ctBs
}

print 'Chiffretext : \textbf{' ,
    e_rsa (new Math::BigInt (11413), new Math::BigInt(3533),
        new Math::BigInt(2), "anrufumelf"), "}"\n";
}

sub prob29 {
    sub witness {
        my ($a, $n) = @_;
        my @b = split //, sprintf "%b", $n-1;
        my $d = 1;

        for (0..@b-1) {
            my $x = $d;
            $d = ($d * $d) % $n;
            return 2 if $d == 1 and $x != 1 and $x != $n-1;
            $d = $d * $a % $n if $b[$_]
        }

        return 1 if $d != 1;
    }
}

```

```
}

my ($ones, $twos) = '0'x2;
for my $a (1..341) {
    if (witness ($a, 341)==1) {
        print "Witness $a gibt 1 zurück\n" if (witness ($a, 341)==1);
        $ones++;
    }
    if (witness ($a, 341)==2) {
        print "Witness $a gibt 2 zurück\n" if (witness ($a, 341)==2);
        $twos++;
    }
}
print "Es existieren $ones Zeugen die 1 provozieren".
      " und $twos Zeugen die 2 provozieren\n";

}

sub prob32 {
    my $n = 12;
    do {
        ++$n;
        while (witness (2, $n) or witness (3, $n)) { ++$n }
    } while not witness (5, $n);
    print $n, "\n"
}

&{'prob'.$ARGV[0]} (splice @ARGV,1)
```