Cryptolgoy homework sheet 1

Due: 10 September 2020, 10:45 for students of 2MMC10 and
17 September 2020, 10:45 for students following the MasterMath course.

Please hand in your homework in groups of two or three. Do *not* submit your homework alone.
Please submit your homework by email to `crypto.course@tue.nl` and put your team mates in cc to that email. Also mention their names and student numbers in the *body of the email* (yes, neatly typed so that we can copy it easily).

You can use a calculator or some computer algebra system for these exercises, but make sure to document all intermediate computations.
If the math background in the lecture of 03 September was new to you, solve some compuations by hand and varify your result by using a computar algebra system. I recommend using Pari-GP if you don't have any installed, yet. See below for an algorithms for CRT. Sage is more powerful than Pari-GP but it is a bit too powerful.

1. Execute the RSA key generation where $p = 239, q = 433$, and $e = 23441$.

2. RSA-encrypt the message 23 to a user with public key $(e, n) = (17, 11584115749)$. Document how you compute the exponentiation with square and multiply, reducing the result after each step.

3. Compute 15^{24} mod 72 twice – once using square and multiply (document the intermediate steps) and once using the Chinese Remainder Theorem with calculations modulo 8 and modulo 9. Remember to reduce the exponents and the base in the CRT calculation and take a moment to think what moduli to use and to check the conditions.

4. Perform one round of the Fermat test with base
   $a = 2$ to test whether 31 is prime.

5. Security proofs in crypto are usually allowing the attacker access to a decryption oracle, i.e. an algorithm that returns the decryption of any valid ciphertext. In the schoolbook version of RSA presented in class, any ciphertext is valid. The attacker wins if he finds the plaintext m belonging to ciphertext c without ever asking the oracle for a decryption of c itself; any $c' \not\equiv c \bmod n$ is fair game.

   Show how the attacker can recover m from $c \equiv m^e \bmod n$ with *one* oracle query and some (easy) computation.

   This exercise shows you that schoolbook RSA should not be used in practice.

Reminder on how the Chinese Remainder Theorem works:

Theorem 1 (Chinese Remainder Theorem)
Let $r_1, \dots, r_k \in \mathbb{Z}$ and let $0 \neq n_1, \dots, n_k \in \mathbb{N}$ such that the n_i are pairwise coprime. The system of congruences

$$\begin{aligned} X &\equiv r_1 \bmod n_1, \\ X &\equiv r_2 \bmod n_2, \\ &\vdots \\ X &\equiv r_k \bmod n_k, \end{aligned}$$

has a solution X which is unique up to multiples of $N = n_1 \cdot n_2 \cdots n_k$. The set of all solutions is given by $\{X + aN | a \in \mathbb{Z}\} = X + N\mathbb{Z}$.

If the n_i are not all coprime the system might not have a solution at all. E.g. the system $X \equiv 1 \bmod 8$ and $X \equiv 2 \bmod 6$ does not have a solution since the first congruence implies that X is odd while the second one implies that X is even. If the system has a solution then it is unique only modulo $\mathrm{lcm}(n_1, n_2, \dots, n_k)$. E.g. the system $X \equiv 4 \bmod 8$ and $X \equiv 2 \bmod 6$ has solutions and the solutions are unique modulo 24. Replace $X \equiv 2 \bmod 6$ by $X \equiv 2 \bmod 3$; the system still carries the same information but has coprime moduli and we obtain $X = 8a + 4 \equiv 2a + 1 \overset{!}{\equiv} 2 \bmod 3$, thus $a \equiv 2 \bmod 3$ and $X = 8(3b + 2) + 4 = 24b + 20$. The smallest positive solution is thus 20.

We now present a constructive algorithm to find the solution promised by the CRT, making heavy use of the extended Euclidean algorithm (see below). Let $N_i = N/n_i$. Since all n_i are coprime, we have $\gcd(n_i, N_i) = 1$ and we can compute u_i and v_i with

$$u_i n_i + v_i N_i = 1.$$

Let $e_i = v_i N_i$, then this equation becomes $u_i n_i + e_i = 1$ or $e_i \equiv 1 \bmod n_i$. Furthermore, since all $n_j | N_i$ for $j \neq i$ we also have $e_i = v_i N_i \equiv 0 \bmod n_j$ for $j \neq i$.
Using these values e_i a solution to the system of equivalences is given by

$$X \equiv \sum_{i=1}^{k} r_i e_i \bmod N,$$

since X satisfies $X \equiv r_i \bmod n_i$ for each $1 \leq i \leq k$.

Example 2 *Consider the system of integer congruences*

$$\begin{aligned} X &\equiv 1 \bmod 3, \\ X &\equiv 2 \bmod 5, \\ X &\equiv 5 \bmod 7. \end{aligned}$$

The moduli are coprime and we have $N = 105$. For $n_1 = 3, N_1 = 35$ we get $v_1 = 2$ by just observing that $2 \cdot 35 = 70 \equiv 1 \bmod 3$. So $e_1 = 70$.
Next we compute $N_2 = 21$ and see $v_2 = 1$ since $21 \equiv 1 \bmod 5$. This gives $e_2 = 21$. Finally, $N_3 = 15$ and $v_3 = 1$ so that $e_3 = 15$.
The result is $X = 70 + 2 \cdot 21 + 5 \cdot 15 = 187$ which indeed satisfies all 3 congruences. To obtain the smallest positive result we reduce 187 modulo N to obtain 82.

For easier reference we phrase this approach as an algorithm.

Algorithm 3 (Chinese remainder computation)
IN: *system of k equivalences as $(r_1, n_1), (r_2, n_2), \ldots (r_k, n_k)$ with pairwise coprime n_i*
OUT: *smallest positive solution to system*

```
1. $N \leftarrow \prod_{i=1}^{k} n_i$
2. $X \leftarrow 0$
3. for $i = 1$ to $k$
   (a) $M \leftarrow N \operatorname{div} n_i$
   (b) $v \leftarrow (M^{-1} \bmod n_i)$ (use XGCD)
   (c) $e \leftarrow vM$
   (d) $X \leftarrow X + r_i e$
4. $X \leftarrow X \bmod N$
```

Just because some people might not have seen XGCD as an algorithm, here is a description of XGCD. This description assumes that the input elements f, g live in some ring R in which the greatest common divisor is defined. We will usually use the XGCD on integers or polynomials. If the inputs are integers you can ignore the part the leading coefficient.

Algorithm 4 (Extended Euclidean algorithm)
IN: $f, g \in R$
OUT: $d, u, v \in R$ *with* $d = uf + vg$

```
1. $a \leftarrow [f, 1, 0]$
2. $b \leftarrow [g, 0, 1]$
3. repeat
   (a) $c \leftarrow a - (a[1] \operatorname{div} b[1])b$
   (b) $a \leftarrow b$
   (c) $b \leftarrow c$
```

```
    while $b[1] \neq 0$
4. $l \leftarrow LC(a[1])$, $a \leftarrow a/l$ /*LC = leading coefficient, this only applies to polynomials*/
5. $d \leftarrow a[1]$, $u \leftarrow a[2]$, $v \leftarrow a[3]$
6. return $d, u, v$
```

In this algorithm, div denotes division with remainder. The first component of c is thus easier written as $c[1] \leftarrow a[1] \bmod b[1]$ but by operating on the whole vector we get to update the values leading to u and v, too. At each step we have

$$a[1] = a[2]f + a[3]g \text{ and } b[1] = b[2]f + b[3]g.$$

To see this, note that this holds trivially for the initial conditions. If it holds for both a and b then also for c since it computes a linear relation of both vectors. So each update maintains the relation and eventually when $b[1] = 0$, we have that $a[1]$ holds the previous remainder, which is the gcd of f and g. If the inputs are polynomials, at the end the gcd is made monic by dividing by the leading coefficient $LC(a[1])$.

Example 5 *Let $R = \mathbb{R}[x]$ and $f(x) = x^5+3x^3-x^2-4x+1$, $g(x) = x^4-8x^3+8x^2+8x-9$. So at first we have $a = [f, 1, 0], b = [g, 0, 1]$.*

We have $(a[1] \operatorname{div} b[1]) = x + 8$ and so end the first round with

$$\begin{aligned} a &= [g, 0, 1], \\ b &= [59x^3 - 73x^2 - 59x + 73, 1, -x - 8]. \end{aligned}$$

Indeed $b[1] = f(x) + (-x - 8)g(x)$.

With these new values we have $(a[1] \operatorname{div} b[1]) = 1/59x - 399/3481$ and so the second round ends with

$$\begin{aligned} a &= [59x^3 - 73x^2 - 59x + 73, 1, -x - 8], \\ b &= [2202/3481x^2 - 2202/3481, -1/59x + 399/3481, 1/59x^2 + 73/3481x + 289/3481]. \end{aligned}$$

In the third round we have $(a[1] \operatorname{div} b[1]) = 205379/2202x - 254113/2202$ and obtain

$$\begin{aligned} a &= [2202/3481x^2 - 2202/3481, -1/59x + 399/3481, 1/59x^2 + 73/3481x + 289/3481], \\ b &= [0, 3481/2202x^2 - 13924/1101x + 10443/734, -3481/2202x^3 - 6962/1101x + 3481/2202]. \end{aligned}$$

Since $b[1] = 0$ the loop terminates. We have $LC(a[1]) = 2202/3481$ and thus normalize to

$$a = [x^2 - 1, -59/2202x + 133/734, 59/2202x^2 + 73/2202x + 289/2202].$$

We check that indeed

$$\begin{aligned} x^2 - 1 = (-59/2202x + 133/734)(x^5 + 3x^3 - x^2 - 4x + 1)+ \\ (59/2202x^2 + 73/2202x + 289/2202)(x^4 - 8x^3 + 8x^2 + 8x - 9). \end{aligned}$$