Cryptolgoy homework sheet 1

Due: 12 September 2019, 10:45 for students of 2MMC10 and
19 September 2019, 10:45 for students following the MasterMath course.

2MMC10: Please hand in your homework in groups of two or three. To submit your homework, place it on the table of the lecturer *before* the lecture.

Mastermath: Please team up in groups of 2 or 3. Please submit your homework by email to `crypto.course@tue.nl`

Please write the names and student numbers on the homework sheet. Please indicate your home university and study direction.
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 05 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.

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 if you only have a pocket calculator. **Note:** You can assume that your calculator has precision large enough to compute numbers up to n^2. I want you to use and document the steps in the square-and-multiply method, Make sure to reduce modulo n.

3. Compute $5^{24} \bmod 72$ 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 also reduce the exponents in the CRT calculation.

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.

   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 $r_1, \ldots, r_k \in \mathbb{Z}$ and let $0 \neq n_1, \cdots, n_k \in \mathbb{N}$ such that the n_i are pairwise coprime. The system of equivalences

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

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 $\operatorname{lcm}(n_1, n_2, \ldots, n_k)$. 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 $X = 8a + 4 \equiv 2a + 1 \stackrel{!}{\equiv} 2 \bmod 3$, 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 this solution, making heavy use of the extended Euclidean algorithm presented in the previous section. 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 equivalences*

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

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. $X \leftarrow X \bmod N$