MODERN CRYPTOGRAPHY
Bachelor Computer Science, University of Amsterdam, 2018/19
TEACHER: Christian Schaffner, TA: Jan Czajkowski, Christian Majenz

Problem Set 5

We will work on the following exercises together during the work sessions on Tuesday, 10 October 2017.

You are strongly encouraged to work together on the exercises, including the homework. You do not have to hand in solutions to these problem sets.

Problem 1: Fermat's little theorem and modular arithmetic.

(a) **Modular roots** Show that the 7th root of 47 modulo 143 is $[47^{103} \mod 143]$. Note that $143 = 11 \cdot 13$.

(b) **Computing seemingly huge numbers by hand** Compute the final two (decimal) digits of 3^{1000} (by hand).

Hint: The answer is $[3^{1000} \mod 100]$. If $\gcd(a, b) = 1$, then $\phi(a \cdot b) = \phi(a) \cdot \phi(b)$.

(c) **Modular arithmetic.** Let p,N be integers with $p|N$ (i.e. p divides N). Prove that for any integer X,

$$[[X \mod N] \mod p] = [X \mod p].$$

Show that, in contrast, $[[X \mod p] \mod N]$ need not equal $[X \mod N]$.

Problem 2: Discrete logarithms

(a) **Easy discrete-logarithm problem.** Explain why the discrete-logarithm problem in the additive group $(\mathbb{Z}_N, +)$ generated by $\langle 1 \rangle$ is easy to solve.

(b) **Diffie-Hellman problem** Let us consider the cyclic group $\mathbb{Z}_{13}^*$.

1. Show that 2 is a generator of $\mathbb{Z}_{13}^*$.
2. In $\mathbb{Z}_{13}^*$, it holds that

$$\mathsf{DH}_2(6, 9) = \mathsf{DH}_2(2^5, 2^8) = 2^{40} = 2^{40 \mod 12} = 2^4 = 3$$

Show in the same way that $\mathsf{DH}_2(12, 10) = 1$.

Problem 3: Diffie-Hellman

(a) **Computational Diffie-Hellman** Define the hardness of the Computational Diffie-Hellman problem with respect to the group-generation algorithm $\mathcal{G}$.

(b) **DDH is hard $\Rightarrow$ CDH is hard $\Rightarrow$ DLog is hard** Prove that hardness of the CDH problem implies hardness of the discrete-logarithm problem, and that hardness of the Decisional Diffie-Hellman problem implies hardness of the CDH problem.

Problem 4: RSA

(a) Let $N = pq$ be a RSA-modulus and let $(N, e, d) \leftarrow \mathsf{GenRSA}$. Show that the ability of efficiently factoring N allows to compute d efficiently.

Given

$$N = 2140310672120493293362298457402658192652108411554313695782475927669427$$

try to compute $\phi(N)$. If that takes too long compute $\phi(p \cdot q)$, where

$$p = 58240080352490526776497122885950201\,,$$
$$q = 36749789134330529121473391864214027\,.$$

Knowing that compute d for given N and $e = 11$. Is it

1945736974654993903056634961275143725147489931575688907101782888641091 ?

(b) **Factoring RSA Moduli** Let $N = pq$ be a RSA-modulus and let $(N, e, d) \leftarrow \mathsf{GenRSA}$. In this exercise, you show that for the special case of $e = 3$, computing d is equivalent to factoring N. Show the following:

1. The ability of efficiently factoring N allows to compute d efficiently. This shows one implication.
2. Given $\phi(N)$ and N, show how to compute p and q. **Hint:** Derive a quadratic equation (over the integers) in the unknown p.
3. Assume we know $e = 3$ and $d \in \{1, 2, \ldots, \phi(N)-1\}$ such that $ed \equiv 1 \mod \phi(N)$. Show how to efficiently compute p and q. **Hint:** Obtain a small list of possibilities for $\phi(N)$ and use (b).
4. Given $e = 3$, $d = 29'531$ and $N = 44'719$, factor N using the method above.

Problem 5: Man-In-The-Middle Attacks

Describe in detail a man-in-the-middle attack on the Diffie-Hellman key-exchange protocol whereby the adversary ends up sharing a key k_A with Alice and a (different) key k_B with Bob, and Alice and Bob cannot detect that anything has gone wrong. What happens if Alice and Bob try to detect the presence of a man-in- the-middle adversary by sending each other (encrypted) questions that only the other party would know how to answer?

Problem 6: Key Exchange with Bit Strings

Consider the following key-exchange protocol:

1. Alice chooses $k, r \leftarrow \{0,1\}^n$ at random, and sends $s := k \oplus r$ to Bob.
2. Bob chooses $t \leftarrow \{0,1\}^n$ at random and sends $u := s \oplus t$ to Alice.
3. Alice computes $w := u \oplus r$ and sends w to Bob.
4. Alice outputs k and Bob computes $w \oplus t$.

Show that Alice and Bob output the same key. Analyze the security of the scheme (i.e., either prove its security or show a concrete attack).