Exercise Sheet #1

Advanced Cryptography 2021

Exercise 1 Factorization

We want to set up the RSA cryptosystem in a network of n users.

1. How many prime numbers do we have to generate?
2. We want to reduce this number by generating a smaller pool of prime numbers and making combinations of two of these primes: for each user, we pick a new pair of two of these primes uniformly at random in order to set up his key. Show how anyone can factorize all moduli for which at least one prime factor has been used in at least one other modulus.
3. Suppose that for n users, $N_1, N_2, \ldots, N_n$ RSA public parameters are set up from a pool of k different large prime numbers $p_1, p_2, \ldots, p_k$, s.t. $k \geq 2$. More precisely, each user ℓ uniformly picks two different prime numbers p_i, p_j and sets $N_\ell = p_i p_j$. Imagine that an adversary $\mathcal{A}$ is given these public parameters $N_1, N_2, \ldots, N_n$ but not the pool of prime numbers. What is the probability of A's factoring public parameter of one fixed, targeted user (say N_1)?

Exercise 2 Square Roots

1. Let $n := pq$ for primes p, q. Given access to a deterministic oracle $\mathcal{O}$ that returns the smallest square root mod n, explain how to factor n using $\mathcal{O}$.
2. Show that your algorithm returns a correct factorization.
3. How many queries to $\mathcal{O}$ do you need to perform in average? Deduce the expected complexity of your algorithm.

Exercise 3 Complexity

1. Give the definition of $f(n) = \Omega(g(n))$, $f(n) = \Theta(g(n))$, and $f(n) = o(g(n))$. Try to use quantifiers in your definition, as given below. (You can assume that $f : \mathbb{N} \mapsto \mathbb{R}$.)

$$f(n) = O(g(n)) \iff \exists c \in \mathbb{R}^+ \, \exists n_0 \in \mathbb{N} \, \forall n \in \mathbb{N} \quad n > n_0 \implies |f(n)| < c \cdot |g(n)|$$

2. Let $f, g : \mathbb{N} \to \mathbb{R}^+$ defined by $f(n) = n(2 + \sin n)$ and $g(n) = n$, for all $n \in \mathbb{N}$. Prove that $f(n) = \Theta(g(n))$.

Exercise 4 Negligible function

A function $f(x) : \mathbb{N} \to \mathbb{R}$ is said to be negligible if $\forall c \in \mathbb{N}, \exists n_0 \in \mathbb{N}$ such that $\forall x > n_0$, we have $|f(x)| < \frac{1}{x^c}$. Otherwise we say f is not negligible[1].

Which of the following functions are negligible? Justify your answer.

1. $f(x) = 2^{-x}$
2. $f(x) = x^{-3}$
3. $f(x) = 100 \cdot x^{-x}$
4. $f(n) = \begin{cases} 5^{-x} & \text{if } x \text{ is even} \\ x^{-10000} & \text{if } x \text{ is odd} \end{cases}$

Exercise 5 Formalism

1. Formally define what is a pseudorandom number generator (without security notions).

   Hint: Remember that we say a message authentication code is a tuple $(\{0,1\}^k, \mathcal{D}, \{0,1\}^\tau, MAC)$ such that

   - $\{0,1\}^k$ is the key domain and $k(s)$ is polynomially bounded function
   - $\{0,1\}^\tau$ is the output domain and $\tau(s)$ is polynomially bounded function
   - $\mathcal{D} \subseteq \{0,1\}^*$ is the message domain
   - MAC is a deterministic polynomially bounded algorithm such that

   $$MAC : \{0,1\}^k \times \mathcal{D} \mapsto \{0,1\}^\tau.$$

2. For pseudorandom number generator primitive $PRNG$, write formal definitions of *state recovery* and *indistinguishability* security notions. Essentially, your definition should rigorously specify a success probability of an arbitrary adversary $\mathcal{A}$, where a security parameter s and number of allowed samples d are implicit.

   Hint: Figures of page 23-25 of course slides informally refer to these two notions.

3. Show that any pseudorandom number generator $PRNG(state_{prev})$ whose output *output* is indistinguishable from a random function is also secure against state recovery attack.

   Hint: Show that an adversary $\mathcal{A}$ succeeding in state recovery game with probability p can be used as a black-box to construct a distinguisher $\mathcal{B}$ and then analyse $\mathcal{B}$'s success probability in terms of p.

[1] There is a difference between non-negligible and not negligible!