Exercise Sheet #2

Advanced Cryptography 2021

Exercise 1 Primes

Let PRIMES $= \{m \colon m$ is a prime number in binary$\}$ be a language. We are going to show that PRIMES $\in$ NP $\cap$ co-NP. (Note that PRIMES $\in$ P, but this is harder to prove). Note that a language $L \in$ co-NP if and only if its complement $\overline{L} \in$ NP.

1. First, show that PRIMES $\in$ co-NP.
2. Show that PRIMES $\in$ NP.
   Hint: Use the fact that $\mathbb{Z}_p^*$ is cyclic when p is prime and use induction.
   Hint: You might need to solve the following recurrence:

$$T(m) \leq O((\log m)^2) + \sum_{i=1}^{j} T(p_i) \ ,$$

   where the p_i are all the prime factors of $m-1$. You can assume that $O((\log m)^3)$ is a solution for this recurrence.

Exercise 2 Fixed Point Attack on RSA

The fixed-point attack, also called *superencryption attack*, was discovered by Simmons and Norris in 1977, as soon as the RSA cryptographic system was invented. Let (e, N) be an RSA public key and $0 \leq x < N$. If $x^{e^k} \equiv x \pmod N$, for $k \in \mathbb{Z}_+$, then x is called a fixed-point of order k for this instance of RSA.

Let c be the ciphertext corresponding to a message m. Let also c be a fixed-point of order k. Show that $c^{e^{k-1}} \equiv m \pmod N$, for $k \in \mathbb{Z}_+$. Explain how you would use this idea to decrypt efficiently when k is small.

Exercise 3 Turing Machines

Here is one possible definition of a Turing machine.[1] A *Turing Machine* M consists of a state q, a reading/writing head, and a half-infinite tape, i.e., a tape only infinite on the right on which the head can read and write. It is formalized as a 7-tuple $(Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$ where Q, Σ, Γ are all finite sets and

1. Q is the set of states,
2. Σ is the input alphabet not containing the blank symbol $\sqcup$,
3. Γ is the tape alphabet where $\sqcup \in \Gamma$ and $\Sigma \subseteq \Gamma$,
4. $\delta \colon Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$ is the transition function,
5. $q_0 \in Q$ is the start state,
6. $q_{accept} \in Q$ is the accept state, and
7. $q_{reject} \in Q$ is the reject state, where $q_{reject} \neq q_{accept}$.

A Turing machine $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$ works as follows. It receives its input on the leftmost squares of the tape, leaving the rest of the tape blank. The head starts on the leftmost square of the tape and M starts in state q_0. M starts then the computations following the rules of the transition function, where $\delta(q, a) = (r, b, L)$ means that if M is in state q and if its head is over the symbol a, then it replaces the symbol a with a b, goes to state r and move the head left. If M ever tries to move its head to the left off the left-hand end of the tape, the head stays in the same place for that move. As soon as the head reaches the state q_{accept} (we say then that M accepts the input) or q_{reject} (we say then that M rejects the input), the machine stops. If neither occurs, M goes forever.

We say that a language L is *recursively enumerable* if there exists a Turing machine M for which the set of inputs such that M accepts these inputs is L.

Of the class of recursively enumerable languages, there is an important subclass called *recursive languages.* A language L is defined to be *recursive* if there exists a Turing machine M that satisfies the following:

- if the input $\omega \in L$, then M eventually enters the halting state q_{accept} and accepts it;
- if the input $\omega \notin L$, then M eventually enters the halting state q_{reject} and rejects it;

1. Prove that a recursive language is recursively enumerable.
2. Prove that if L is a recursive language, so is its complement $\overline{L}$.

[1]Part of this definition is taken from "Introduction to the Theory of computation", by Michael Sipser