COS 433 — Cryptography — Homework 6.

Boaz Barak

Total of 150 points. Due November 8th, 2007.

Note: You have two weeks to solve this exercise and it also has many bonus points. So this is a good chance to make up for lost points in past or future exercises. However, note that all proofs and reductions must be rigorous— written clearly and precisely, and without any logical gaps. Try to ensure that a reader will be able to easily follow your line of reasoning, and will be absolutely convinced in your proof's validity. You will not receive credit for proofs that are written in a confusing, vague, or incomplete fashion.

Exercise 1 (30 points)**.** In this question we complete and formalize the proof of the Chinese Reminder Theorem.

1. Let G_1, G_2 be abelian groups where $+_i$ is the group operation of G_i. We define the *direct product* $G \stackrel{\text{def}}{=} G_1 \times G_2$, which consists of all pairs (a_1, a_2) where $a_i \in G_i$. We can view G in a natural way as an abelian group if we define the group operation $+_G$ component-wise: $(a_1, a_2) +_G (b_1, b_2) = (a_1 +_1 b_1, a_2 +_k b_2)$. Prove that G is indeed an abelian group with respect to $+_G$.

2. A *group homomorphism* is a function f from an abelian group G to an abelian group H that preserves the group operation; i.e., $f(a) +_H f(b) = f(a +_G b)$ for all $a, b \in G$. Let $n = p \cdot q$ where $\gcd(p, q) = 1$. Let f be a mapping from Z_n to $Z_p \times Z_q$ such that $f(x) = (x \mod p, x \mod q)$. Show that f is a group homomorphism.

3. Show that $f(x) \in Z_p^* \times Z_q^*$ if and only if $x \in Z_n^*$.

4. By the previous question, f can be viewed as a mapping from Z_n^* to $Z_p^* \times Z_q^*$. Show that in this case f is also a group homomorphism.

5. Consider the following algorithm for inverting f. (Assume that p, q are known.)

   - Input: $(x_p, x_q) \in Z_p \times Z_q$.
   - Use Euclid's algorithm to find integers α, β such that $\alpha p + \beta q = 1$.
   - Set $1_p = \alpha q \mod n$ and $1_q = \beta p \mod n$.
   - Compute $x = (x_p \cdot 1_p + x_q \cdot 1_q) \mod n$

   Prove that the algorithm inverts f, namely, $x \mod p = x_p$ and $x \mod q = x_q$.

6. Conclude that f is an isomorphism from Z_n^* to $Z_p^* \times Z_q^*$ as well as from Z_n to $Z_p \times Z_q$. You might want to use the following fact (that we proved in class): $|Z_n^*| = (p-1) \cdot (q-1)$.

7. Read your answers to the previous questions. Where did you use the fact that p and q are primes? Is it possible to prove these results under a weaker condition?

Exercise 2 (20 points). 1. We say that a number $y \in Z_n^*$ is a Quadratic Residue (QR) if $y = x^2$ for some $x \in Z_n^*$. (We refer to x as a sqrt of y.) Prove that the set of QRs is a subgroup of Z_n^*.

2. Let $p > 1$ be a prime number. It can be shown that Z_p^* is a cyclic group, that is there exists a generator $g \in Z_p^*$ such that $Z_p^* = \{g^1, \ldots, g^(p-1)\}$. For $y \in Z*_p$ we let $\log_g(y)$ to denote the smallest non-negative integer i for which $g^i = y$. For example $\log_g(1) = 0$ and $\log_g(g) = 1$. (Note that $0 \geq \log_g(y) \geq p - 1$.) Show that y is a QR in $Z*_p$ if and only $\log_g(y)$ is an even number.

Exercise 3 (30 points). In class we saw the following public key encryption scheme based on any family of trapdoor permutations $\{f_e\}$.

Key generation Choose $(e, d) \leftarrow_R \mathsf{Gen}(1^n)$ where Gen is the generator for the trapdoor permutation family $\{f_e\}$.

Encryption To encrypt a bit $b \leftarrow_R \{0,1\}$ using the key e: choose $x \leftarrow_R S_e$, choose $r \leftarrow_R \{0,1\}^n$, and output $f_e(x), r, \langle x, r\rangle \oplus b$.

Decryption To decrypt the message (y, r, c) using d: compute $x = f_e^{-1}(y)$ and output $\langle x, r\rangle \oplus c$.

Prove that if $\{f_e\}$ is a trapdoor permutation collection, the above scheme is a CPA secure public key encryption scheme.

Exercise 4 (25 points). We can define *chosen ciphertext security* for public key encryption schemes in the same way that we defined them for private key encryption schemes: the adversary gets access to a decryption box before the challenge and after receiving the challenge ciphertext y^* is allowed to query the box on every string y *except* for y^*. The only difference is that the adversary gets initially the encryption key as another input. As usual we say the scheme is secure against Chosen Ciphertext Attack (CCA secure for short) if no poly-time adversary can win with $1/2 + \epsilon(n)$ probability where ϵ is poly-bounded.

1. Show that every public key encryption that is built from a trapdoor permutation family as in Exercise 3 (when we encrypt an n bit message by encrypting each bit individually) is *not* CCA secure. See footnote for hint[1]

2. Show that the specific encryption scheme based on Rabin's trapdoor permutation family has an even more devastating attack: show that given access to a decryption box for this scheme a polynomial-time adversary can recover the *private key* with high probability using only a constant number of queries.

Exercise 5 (25 points). As mentioned in class, the way to generate a random n-bit prime, is to pick a random n-bit number and test it for primality. Thus we need a polynomial-time primality algorithm. We now describe such an algorithm. We assume that we have a polynomial-time algorithm `SQRT` that on input a pair y, N such that N is prime and y is a quadratic residue modulo N, outputs a square-root of y modulo N (i.e., a number x such that $x^2 = y \pmod N$). We saw in class such an algorithm for the case that $N = 4K + 3$ for some K, and in the KL book Section 11.2.1 you can see its extension for the case that N is a prime of the form $N = 4K + 1$. We stress

[1] **Hint:** Show that every encryption scheme that works in a bit-by-bit fashion is not CCA secure.

that we make no assumption on the algorithm's output if x and N are not of this form. However, by stopping the algorithm if it runs for too much time, we can assume that it always stops in polynomial-time and outputs some number between 1 and $N-1$.

Consider the following algorithm:

Algorithm `PTEST`. On input N do:

1. If N is even or $N = C^k$ for some $k \in \{1, \ldots, \log N\}$ then output `'composite'` and halt.
2. Pick $x \leftarrow_{\mathrm{R}} \{1, \ldots, N-1\}$.
3. If $gcd(x, N) \neq 1$ output `'composite'` and halt.
4. Compute $y = x^2 \pmod N$ and $w = \mathtt{SQRT}(y, N)$.
5. If $w^2 = y \pmod N$ and $y = x \pmod N$ or $y = -x \pmod N$ then output `'prime'` and halt. Otherwise output `'composite'`.

Prove that for every positive integer N, Algorithm `PTEST` runs on input N in poly$(\log N)$-time and

1. If N is prime then $\mathtt{PTEST}(N)$ outputs `'prime'` with probability 1.
2. If N is composite then $\mathtt{PTEST}(N)$ outputs `'composite'` with probability $\geq 1/10$.

Exercise 6 (20 points)**.** Suppose that the RSA Assumption fails "somewhat" on a particular composite number N and e with $gcd(e, \varphi(N)) = 1$, in the sense that there is a T-time algorithm A such that

$$\Pr_{y \leftarrow_{\mathrm{R}} \mathbb{Z}_N^*} [A(y) = x \text{ s.t. } x^e = y \pmod N] > 0.01$$

Show that there is a $100(\log N)^{100} \cdot T$-time algorithm B that breaks the RSA Assumption completely for N, e in the sense that

$$\Pr_{y \leftarrow_{\mathrm{R}} \mathbb{Z}_N^*} [A(y) = x \text{ s.t. } x^e = y \pmod N] > 0.99$$

See footnote for hint[2]

[2] **Hint:** Use the fact that $y^{1/e} r = (yr)^{1/e} \pmod N$ for every $r \in \mathbb{Z}_N^*$.