Winter 2018 CS 485/585 Introduction to Cryptography

Homework 4

Portland State University — *Update: Feb. 20, 2018*

Student: your name — *Due: Mar. 1, 2018*

Instructions. Your solutions will be graded on *correctness* and *clarity*. You should only submit work that you believe to be correct; if you cannot solve a problem completely, you will get significantly more partial credit if you clearly identify the gap(s) in your solution. It is good practice to start any long solution with an informal (but accurate) "proof summary" that describes the main idea. Problems marked with "**[G]**" are required for graduate students. Undergraduate students will get bonus points for solving them. The .tex source is provided on course webpage as a template if you want to typeset your solutions in Latex.

You may collaborate with others on this problem set. However, you must ***write up your own solutions*** and ***list your collaborators*** for each problem.

1. (Number Theory) Exercises. Do not turn in.
   (a) Prove the following for the relation $= \mod n$.
      i. If $a = b \bmod n$ and $b = c \mod n$, then $a = c \bmod n$.
      ii. If $a = a' \bmod n$ and $b = b' \bmod n$, then $a + b = a' + b' \bmod n$.
      iii. If $a = a' \bmod n$ and $b = b' \bmod n$, then $ab = a'b' \bmod n$.

   (b) Find $gcd(60, 17)$ and (α, β) such that $\alpha \cdot 60 + \beta \cdot 17 = gcd(60, 17)$ by (extended) Euclidean algorithm.

   (c) Prove that for p, q co-prime, the Euler function $\phi(pq) = \phi(p) \cdot \phi(q)$. Let p be prime and $k > 0$ an integer, prove that $\phi(p^k) = (p-1)p^{k-1}$.

2. (Factoring and RSA)
   (a) (6 points) Let $N = pq$ be a product of two distinct primes. Show that if $\phi(N)$ and N are known, then it is possible to compute p and q in polynomial time. (Hint: derive a quadratic equation over the integers in the unknown p.)

   (b) (6 points) (Attacking plain-RSA encryption when encrypting short messages using small e.) Let $e = 3$ and N be a modulus of length 1024 bits. Consider encrypting a message m of length 100-bit using plain-RSA. Describe an attack that recovers the plaintext m from its ciphertext. Does your attack still apply when encrypting other messages?

   (c) (5 points) (Homomorphism.) Suppose you know ciphertexts c_1 and c_2 for two messages m_1 and m_2 under plain-RSA encryption. Can you get the ciphertext for $m := m_1 \cdot m_2 \bmod N$?

3. (Discrete log, Diffie-Hellman, and El Gamal)

   (a) (5 points) Let G be a finite group. The *order* of an element $g \in G$ is defined as the smallest positive integer i such that $g^i = 1$. Prove that $g^x = g^y$ if and only if $x = y \bmod i$.

   (b) (10 points) [KL: Exercise 11. 6]

   (c) (6 points) [KL: Exercise 11.10]

   (d) (Bonus 5pts) Read [KL: Claim 11.7]. Let $\Pi = (G, E, D)$ be a CCA-secure public-key encryption scheme. Construct an encryption scheme $\Pi' = (G', E', D')$ such that $E'_{pk}(m_1, \ldots, m_\ell) := E_{pk}(m_1), \ldots, E_{pk}(m_\ell)$. Is Π' CCA-secure? Justify your answer.

4. (Bonus 10pts. Hybrid Encryption) Let $\Pi_1 = (G, E, D)$ be a CCA-secure public-key encryption scheme and $\Pi' = (G', E', D')$ a CCA-secure private-key encryption. Construct hybrid public-key encryption $\tilde{\Pi} = (\tilde{G}, \tilde{E}, \tilde{D})$

   - $\tilde{G} = G$: run $(pk, sk) \leftarrow G(1^n)$.
   - $\tilde{E}$: to encrypt m, run $k \leftarrow G'(1^n)$, and compute $c_1 = E_{pk}(k)$ and $c_2 = E'_k(m)$. Output $c = (c_1, c_2)$.
   - $\tilde{D}$: to decrypt $c = (c_1, c_2)$, compute $k := D_{sk}(c_1)$, and output $m := D'_k(c_2)$.

   Prove that $\tilde{\Pi}$ is CCA-secure. (Hint: read the CPA-security proof [KL: Theorem 11.12])

5. (Attacking plain-RSA signature) Consider the signature scheme (plain-RSA): run **GenRSA**(1^n) to generate (e, d) with $ed = 1 \bmod N$, and make e public and keep d secret; to sign a message $m \in \mathbb{Z}_N^*$, compute $\sigma := [m^d \bmod N]$; to verify (m, σ), accept iff. $m = [\sigma^e \bmod N]$.

   (a) (6 points) Show how to forge a valid signature on an arbitrary message m, by querying a signing oracle *twice*.

   (b) (6 points) Show how to forge a valid signature on an arbitrary message m, by querying a signing oracle *once*.