CS60088 Foundations of Cryptography, Spring 2014–2015

Mid-Semester Test

18–February–2015 CSE-107, 2:00–4:00pm Maximum marks: 35

Roll no: ________________ **Name:** ____

[*Write your answers in the question paper itself. Be brief and precise. Answer all questions.*]

1. Let $n = pq$ be an RSA modulus (with suitably large primes p and q), and e and d the encryption and decryption keys of a party.

(a) Let $m \in \mathbb{Z}_n$ be a message. Prove that there exists a positive integer k such that the k-fold encryption of m gives m itself, that is, $m^{e^k} \equiv m \pmod{n}$. **(5)**

Solution Since $m, m^e, m^{e^2}, m^{e^3}, \ldots \pmod{p}$ belong to a finite set, there must exist positive integers i and j with $i < j$ such that $m^{e^i} \equiv m^{e^j} \pmod{n}$. The i-fold decryption of this gives $m \equiv \left(m^{e^i}\right)^{d^i} \equiv \left(m^{e^j}\right)^{d^i} \equiv m^{e^{j-i}} \pmod{n}$.

(b) Let periodicity(m,e) denote the smallest positive integer k for which $m^{e^k} \equiv m \pmod{n}$. Prove that periodicity$(m,e)$ divides $\phi(\phi(n))$. **(5)**

Solution Let $k = \text{periodicity}(m,e)$. Since $e \in \mathbb{Z}^*_{\phi(n)}$, Euler's theorem gives $e^{\phi(\phi(n))} \equiv 1 \pmod{\phi(n)}$, that is, $m^{e^{\phi(\phi(n))}} \equiv m \pmod{n}$. If $m^{e^l} \equiv m \pmod{n}$, we have $m^{e^l} \equiv m^{e^k e^{l-k}} \equiv \left(m^{e^k}\right)^{e^{l-k}} \equiv m^{e^{l-k}} \equiv m \pmod{n}$. Proceeding in this way, we can show that $m^{e^r} \equiv m \pmod{n}$, where $r = \phi(\phi(n)) \operatorname{rem} k$. By definition, k is the smallest positive integer for which $m^{e^k} \equiv m \pmod{n}$. Therefore, r must be zero.

(c) In this part, assume that p and q are safe primes, that is, $p = 2p' + 1$ and $q = 2q' + 1$ for some primes p' and q'. Assume further that an oracle exists that, upon the input of $m \in \mathbb{Z}_n$ and $e \in \mathbb{Z}^*_{\phi(n)}$, returns periodicity$(m, e)$. Demonstrate how this oracle can be used to factor n in probabilistic polynomial time (without the knowledge of d). **(5)**

Solution We invoke the oracle on several messages m and encryption exponents e. We must choose all encryption exponents e coprime to $\phi(n) = 4p'q'$. Randomly chosen odd values of e are expected to be coprime to $\phi(n)$ with very high probability. Each invocation of the oracle returns a divisor of $\phi(\phi(n))$. After a few iterations, we expect that the lcm of these divisors equals $\phi(\phi(n))$. Notice that $|\phi(\phi(n))| = |n| - 2$, so it is easy to detect when $\phi(\phi(n))$ is computed.

We now have two equations in p' and q'. First, we have

$$n = pq = (2p' + 1)(2q' + 1).$$

Second, we have $\phi(n) = (p-1)(q-1) = 4p'q'$, that is,

$$\phi(\phi(n)) = 2(p' - 1)(q' - 1).$$

Solving these two equations reveals p' and q' and subsequently p and q too.

(**Remark:** A single invocation of the oracle can never reveal $\phi(\phi(n))$. As in Part (b), we can prove that periodicity(m, e) divides $\mathrm{ord}_{\phi(n)}(e)$, and $\phi(n)$ being equal to $4p'q'$, the group $\mathbb{Z}^*_{\phi(n)}$ is not cyclic.)

2. This exercise deals with a variant of ElGamal signatures. Let p be a suitably large prime with a primitive root g (that is, g is a generator of $\mathbb{Z}_p^*$), and let the private and public keys of Alice be x and y, respectively (so we have $y \equiv g^x \pmod{p}$). In order to sign a message $m \in \mathbb{Z}_{p-1}$, Alice chooses $k \in_U \mathbb{Z}_{p-1}$, and computes $r \equiv g^k \pmod{p}$ and $s \equiv xr + km \pmod{p-1}$. Alice's signature on m is the pair (r, s).

(a) Show how the signature (r, s) on m can be verified. **(5)**

Solution For a valid signature, we have $g^s \equiv (g^x)^r (g^k)^m \equiv y^r r^m \pmod{p}$. Therefore, the verifier accepts the signature if and only if the congruence $g^s \equiv y^r r^m \pmod{p}$ holds.

(b) Show how these modified ElGamal signatures can be existentially forged. **(5)**

Solution The forger chooses an $r \equiv g^u y^v \pmod{p}$ for some $u \in \mathbb{Z}_{p-1}$ and $v \in \mathbb{Z}_{p-1}^*$. Verification requires the congruence $g^s \equiv y^r (g^u y^v)^m \pmod{p}$ be satisfied. So the forger can take $s \equiv um \pmod{p-1}$, and $r + vm \equiv 0 \pmod{p-1}$, that is, the forger first computes $m \equiv -rv^{-1} \pmod{p-1}$, and then obtains $s \equiv um \equiv -urv^{-1} \pmod{p-1}$.

3. Let $p \equiv 3 \pmod 8$ be a suitably large prime, and g a generator of $\mathbb{Z}_p^*$. Assume that there exists an oracle which, upon the input of $a \in \mathbb{Z}_p^*$, returns the third least significant bit x_2 of $x = \log_g a = (x_{l-1} \ldots x_3 x_2 x_1 x_0)_2$ (where $x \in \{0, 1, 2, \ldots, p-2\}$). We now design a polynomial-time algorithm to compute discrete logarithms in $\mathbb{Z}_p^*$ to the base g by invoking this oracle multiple times.

(a) Suppose that we want to compute $x = \log_g a = (x_{l-1} \ldots x_3 x_2 x_1 x_0)_2$. Explain how x_0 can be computed. **(2)**

Solution We compute the Legendre symbol $\left(\frac{a}{p}\right)$. If $\left(\frac{a}{p}\right) = +1$, then $x_0 = 0$. If $\left(\frac{a}{p}\right) = -1$, then $x_0 = 1$.

(b) Explain how x_1 can be computed by invoking the third-least-significant-bit oracle once. **(3)**

Solution Multiplying a by g^{1-x_0} lets us assume, without loss of generality, that $x_0 = 1$. We invoke the third-least-significant-bit oracle, supplying $a^2 \pmod p$ as input. Let $y = \log_g(a^2)$. We claim that the third least significant bit of y is x_1.

If $x < (p-1)/2$, then $y = 2x = (x_{l-1} \ldots x_3 x_2 x_1 10)_2$, so x_1 is the third least significant bit of y. If $x \geqslant (p-1)/2$, then $y = 2x - (p-1) = (x_{l-1} \ldots x_3 x_2 x_1 10)_2 - (\ldots 010)_2 = (\ldots x_1 00)_2$, that is, x_1 is again the third least significant bit of y.

(**Remark:** If $p \equiv 7 \pmod 8$, we take $x_0 = 0$.)

(c) Explain how each of x_i, $i \geqslant 2$, can be computed by invoking the third-least-significant-bit oracle once. **(5)**

Solution For computing x_i, $i \geqslant 2$, assume that $x_0, x_1, \ldots, x_{i-1}$ are available. Take $b \equiv a g^{-x_0 - 2x_1 - 2^2 x_2 - \cdots - 2^{i-1} x_{i-1}} \pmod p$. We have $\log_g b = (x_{l-1} \ldots x_i 00 \ldots 0)_2$. Since $p \equiv 3 \pmod 4$, every quadratic residue in $\mathbb{Z}_p^*$ has two square-roots, one of which is again a quadratic residue, and the other a quadratic non-residue. We successively take square root of b exactly $i-2$ times. On each occasion, we take that square root which is a quadratic residue modulo p (this square root can be easily identified by a Legendre-symbol calculation). This eventually gives us $c \in \mathbb{Z}_p^*$ with $z = \operatorname{ind}_g c = (x_{l-1} \ldots x_i 00)_2$. Querying the third-least-significant-bit oracle, with c as input, gives us x_i.