CS60088 Foundations of Cryptography, Spring 2013–2014

End-Semester Test

23–April–2014 | CSE-119 & 120, 2:00–5:00pm | Maximum marks: 60

Roll no: ____________ **Name:** ______________________________

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

1. Let $p \equiv 3 \pmod 4$ be a cryptographically large prime, and g a generator of $\mathbb{Z}_p^*$. We have discussed two bit-level oracles for computing discrete logarithms in $\mathbb{Z}_p^*$. The *half-order oracle* (HOO), given $a \in \mathbb{Z}_p^*$, returns 0 or 1 according as whether the discrete logarithm $\operatorname{ind}_g a$ is $< (p-1)/2$ or $\geqslant (p-1)/2$, respectively. The *second least significant bit oracle* (SLSBO), given $a \in \mathbb{Z}_p^*$, returns the second least significant bit of $\operatorname{ind}_g a$. These two oracles are essentially equivalent to one another. Of course, the existence of any one of these oracles implies that discrete logarithms can be efficiently computed in $\mathbb{Z}_p^*$. From the discrete logarithm, the other oracle can be trivially designed. But the computation of a discrete logarithm involves about $\log_2 p$ invocations of the oracle used. For simulating one oracle by the other, we can do far better. More precisely, given the oracle HOO, design the oracle SLSBO which makes *only one* query to HOO, and conversely. **(10)**

Solution Let $x = \operatorname{ind}_g a$. Then, $\operatorname{ind}_g(a^2) \equiv 2x \pmod{p-1}$, that is, $\operatorname{ind}_g(a^2) = \begin{cases} 2x & \text{if } x < (p-1)/2, \\ 2x-(p-1) & \text{if } x \geqslant (p-1)/2. \end{cases}$ Since $p \equiv 3 \pmod 4$, the second least significant bit of $p-1$ is 1. Consequently, $\mathrm{SLSB}(2x \pmod{p-1}) = \begin{cases} \mathrm{LSB}(x) & \text{if } \mathrm{HO}(x) = 0, \\ \text{Complement of } \mathrm{LSB}(x) & \text{if } \mathrm{HO}(x) = 1. \end{cases}$ Therefore, the simulations proceed as follows.

Simulation of SLSBO by HOO

1. Determine $b = \mathrm{LSB}(x)$ by computing the Legendre symbol $\left(\frac{a}{p}\right)$.
2. If $b = 1$, replace a by $ag^{-1} \pmod p$.
3. Compute the two square roots r_1, r_2 of a as $\pm a^{(p+1)/4} \pmod p$.
4. By making a query to HOO (on r_1 or r_2), determine which one is the *correct* square root r of a (that is, $r \in \{r_1, r_2\}$ with $\operatorname{ind}_g r = \frac{1}{2} \operatorname{ind}_g a$, that is, $\operatorname{ind}_g r < (p-1)/2$).
5. Determine $b' = \mathrm{LSB}(\operatorname{ind}_g r)$ by computing the Legendre symbol $\left(\frac{r}{p}\right)$.
6. Return b'.

Simulation of HOO by SLSBO

1. Determine $b = \mathrm{LSB}(x)$ by computing the Legendre symbol $\left(\frac{a}{p}\right)$.
2. Make an oracle query to get $b' = \mathrm{SLSBO}(p, g, a^2 \pmod p)$.
3. If $b = b'$, return 0, else return 1.

2. [*Pointcheval transform*] Pointcheval (PKC 2000) proposes a generic construction to convert a one-way trapdoor function to an IND-CCA2-secure encryption algorithm. Let $f : \mathscr{X} \times \mathscr{Y} \to \mathscr{Z}$ be an injective one-way function. It is intractable, given $z \in_U \mathscr{Z}$, to compute $x \in \mathscr{X}$ and $y \in \mathscr{Y}$ such that $f(x,y) = z$. However, if a trapdoor is available, one can efficiently obtain, from $z \in \mathscr{Z}$, an $x \in \mathscr{X}$ such that $z = f(x,y)$ for some $y \in \mathscr{Y}$. We denote this by $f_{td}^{-1}(z) = x$. We call such an f a *partially invertible one-way trapdoor function*.

Let $m \in \{0,1\}^{k_0}$ be the message to be encrypted. Let $k = k_0 + k_1$ for some k_1 with $1/2^{k_1}$ negligible. We have two hash functions $H : \{0,1\}^k \to \mathscr{Y}$ and $G : \mathscr{X} \to \{0,1\}^k$. We choose $r \in_U \mathscr{X}$ and $s \in_U \{0,1\}^{k_1}$. We then compute $c_1 = f(r, H(m \,||\, s))$ and $c_2 = (m \,||\, s) \oplus G(r)$. The ciphertext for m is $c = (c_1, c_2)$.

(a) Demonstrate how decryption is carried out using the function f_{td}^{-1}. **(5)**

Solution The decryption of (c_1, c_2) proceeds as follows.

1. Use the trapdoor to compute $r = f_{td}^{-1}(c_1)$.
2. Compute $m' = c_2 \oplus G(r)$.
3. If $f(r, H(m')) = c_1$, return the first k_0 bits of m', else return *failure*.

(b) Now, assume that there exists a PPT algorithm $\mathscr{A}$ that can win the IND-CCA2 game with non-negligible probability, without knowing the trapdoor. Using this, we design another PPT algorithm (Simon the simulator). The goal of Simon is to partially invert a challenge output of f, that is, to compute x^* from $z^* = f(x^*, y^*) \in_U \mathscr{Z}$. Explain how Simon simulates G and H oracle queries during the *find* stage (that is, before m_0, m_1 are supplied to Simon by $\mathscr{A}$). **(5)**

Solution In the find stage, Simon sends uniformly random outputs for all G and H queries.

For a query $G(r)$, Simon first checks whether the pair $(r, G(r))$ is already present in his G-table. If so, the stored value of $G(r)$ is returned. If not, a bit string $G_r \in_U \{0,1\}^k$ is chosen, the pair (r, G_r) is added to the G-table, and G_r is returned to $\mathscr{A}$ as $G(r)$.

For a query $H(m')$, Simon first checks whether $|m'| = k$. If not, Simon notifies $\mathscr{A}$ that the query is invalid. For a valid query, Simon checks whether $(m', H(m'))$ already resides in his H-table. If so, the stored $H(m')$ is returned. Otherwise, a random $H_{m'} \in_U \mathscr{Y}$ is chosen, the pair $(m', H_{m'})$ is added to the H-table, and $H_{m'}$ is returned to $\mathscr{A}$ as $H(m')$.

(c) Explain how Simon simulates a decryption query. **(5)**

Solution Suppose that a decryption query for (c_1, c_2) comes to Simon, in the find stage or in the guess stage. We have $(c_1, c_2) \neq c^*$ (with high probability in the find stage and as a rule in the guess stage, where c^* is defined in Part (d)). Simon consults his H and G tables to find out whether there exist m' and r such that $H(m')$ and $G(r)$ are defined, $c_1 = f(r, H(m'))$, and $G(r) = c_2 \oplus m'$. If so, the first k_0 bits of m' are returned as m. If not, the ciphertext is declared as invalid. There exists a chance that valid ciphertexts are declared as invalid, but the probability of that happening is negligibly low.

(d) After the initial find stage, $\mathscr{A}$ supplies two plaintext messages $m_0, m_1 \in \{0,1\}^{k_0}$. Simon selects a random $b \in_U \{0,1\}$, sets $c_1^* = z^*$, chooses $c_2^* \in_U \{0,1\}^k$, and sends $c^* = (c_1^*, c_2^*)$ as the purported encryption of m_b. Notice that x^* is uniquely determined by z^*. Simon chooses an $s^* \in_U \{0,1\}^{k_1}$ such that c^* corresponds to the encryption of m_b with $r = x^*$ and $s = s^*$. What constraints does this impose on the G and H values? **(5)**

Solution In order that c^* is a valid encryption of m_b, we should have the following constraints.

1. $H(m_b \,||\, s^*) = y^*$, where $z^* = f(x^*, y^*)$.
2. $G(x^*) = c_2^* \oplus (m_b \,||\, s^*)$.
3. $H(m_b \,||\, s^*)$ is undefined at this point.
4. $G(x^*)$ is undefined at this point.

Conditions (1) and (2) follow from the encryption algorithm. Condition 3 can be enforced even if H query exists on this m_b—Simon only needs to choose an s^* for which $H(m_b \,||\, s^*)$ is not queried. Condition 4 is extremely probable, since without seeing $c_1^* = z^*$, the probability of $\mathscr{A}$ having made the query $G(x^*)$ is negligible.

Take-home challenge: Complete the rest of the proof.

3. Consider ElGamal encryption in $\mathbb{Z}_p^*$. Let g be an element of $\mathbb{Z}_p^*$ of suitably large order q. Let $\mathscr{G}$ be the subgroup of $\mathbb{Z}_p^*$ generated by g. An ElGamal key pair consists of the private key $x \in_U \mathbb{Z}_q$ and the public key $y \equiv g^x \pmod{p}$. Encryption of $m \in \mathbb{Z}_p^*$ proceeds as follows. For $l \in_U \mathbb{Z}_q$, one computes $a \equiv g^l \pmod{p}$ and $b \equiv my^l \pmod{p}$. The encryption of m is the pair (a,b).

(a) Present ElGamal encryption as a partially invertible one-way trapdoor function. **(5)**

Solution We have $f(m,l) = (a,b)$ with $m \in \mathscr{X} = \mathbb{Z}_p^*$, $l \in \mathscr{Y} = \mathbb{Z}_q$, and $(a,b) \in \mathscr{Z} = \mathscr{G} \times \mathbb{Z}_p^*$. We have $\mathscr{G} \cong \mathbb{Z}_q$, and the function $f : \mathscr{X} \times \mathscr{Y} \to \mathscr{Z}$ is a bijection. The knowledge of the private key x allows the recipient to partially invert f, that is, to recover m.

(b) Apply the Pointcheval transform of Exercise 2 on ElGamal encryption. **(5)**

Solution Take $k_0 = |p| - 1$, and $k = k_0 + k_1$ for some $k_1 \geqslant 160$. As $\mathscr{X}$, we use the subset of $\mathbb{Z}_{p^*}$ consisting of all k_0-bit strings. But then, the image of $\mathscr{X} \times \mathscr{Y}$ is not the full of $\mathscr{G} \times \mathbb{Z}_p^*$, but this is not a big issue, since the restriction of f continues to remain injective. We use two hash functions $H : \{0,1\}^k \to \mathscr{Y} = \mathbb{Z}_q$ and $G : \{0,1\}^{k_0} \to \{0,1\}^k$. In order to encrypt a k_0-bit message m, we choose $r \in_U \{0,1\}^{k_0}$ and $s \in_U \{0,1\}^{k_1}$. We compute:

$$
\begin{aligned}
l &= H(m \,||\, s), \\
a &\equiv g^l \pmod{p}, \\
b &\equiv g^l r \pmod{p}, \\
c &= (m \,||\, s) \oplus G(r).
\end{aligned}
$$

The ciphertext of m is the triple (a,b,c).

4. [*Feige–Fiat–Shamir* (*FFS*) *protocol*] This is a round-efficient version of the Fiat–Shamir protocol. A composite modulus $n = pq$ with suitably large primes $p, q \equiv 3 \pmod 4$ are chosen. After n is constructed, its factorization is no longer needed and can be forgotten. A small integer t is also chosen. Alice's private input consists of t elements $x_1, x_2, \ldots, x_t \in_U \mathbb{Z}_n^*$. The common input consists of n and $y_1, y_2, \ldots, y_t$, where $y_i \equiv (-1)^{\beta_i} x_i^2 \pmod n$ with $\beta_i \in_U \{0,1\}$, for all $i = 1, 2, \ldots, t$.

During a run of the protocol, Alice computes and sends to Bob the commitment $c \equiv (-1)^{\gamma} k^2 \pmod n$ for $k \in_U \mathbb{Z}_n^*$ and $\gamma \in_U \{0,1\}$. Bob's challenge consists of t bits $b_1, b_2, \ldots, b_t$ each uniformly randomly chosen from $\{0,1\}$. The response of Alice to Bob is $r \equiv k \prod_{\substack{i=1 \\ b_i=1}}^{t} x_i \pmod n$.

(a) Explain the verification step of Bob. **(5)**

Solution Squaring the equation for r gives the verification condition $r^2 \equiv \pm c \prod_{\substack{i=1 \\ b_i=1}}^{t} y_i \pmod n$.

(b) Deduce the completeness and soundness-error probabilities for the FFS protocol. **(5)**

Solution If Alice knows the secret $x_1, x_2, \ldots, x_t$, she can definitely generate the correct response, so the completeness probability is 1. For deducing the soundness-error probability, we assume that Alice does not know one or more of $x_1, x_2, \ldots, x_t$. The right side of the verification congruence is fixed after the challenge phase. Moreover, since Alice cannot change the commitment after seeing the challenge, producing a correct response r is intractable under the SQRT assumption. However, Alice can guess the bits $b_1, b_2, \ldots, b_t$ correctly with probability $1/2^t$. Thus, the soundness error probability is $1/2^t$.

(c) Prove that a simulator not knowing $x_1, x_2, \ldots, x_t$ can generate FFS transcripts having the identical probability distribution as transcripts from real runs of the protocol. **(5)**

Solution The following steps are performed by the simulator (equator):

1. Generate a response $r \in_U \mathbb{Z}_n^*$.
2. Generate a random bit vector $(b_1, b_2, \ldots, b_t) \in_U \{0,1\}^t$.
3. Generate the commitment as $c \equiv (-1)^\gamma r^2 \prod_{\substack{i=1 \\ b_i=1}}^{t} y_i^{-1} \pmod{n}$ with $\gamma \in_U \{0,1\}$.
4. Output the transcript $c, (b_1, b_2, \ldots, b_t), r$.

It is straightforward to argue that this simulated transcript has the same probability distribution as a transcript generated by an actual interaction between Alice and Bob.

(d) Suppose that there exists a probabilistic polynomial time cheating prover that, without knowing one or more of $x_1, x_2, \ldots, x_t$, can be accepted by Bob with non-negligible probability. Prove that $O(t2^t)$ successful sessions help the cheating prover to know all x_i values with high probability. Argue why one can take $t = O(\log_2 \log_2 n)$ but not larger. **(5)**

Solution After $O(2^t/2)$ random successful sessions with the same commitment c, we expect to have, with high probability, two transcripts $c, (b_1, b_2, \ldots, b_t), r$ and $c, (b'_1, b'_2, \ldots, b'_t), r'$ with the bit vectors $(b_1, b_2, \ldots, b_t)$ and $(b'_1, b'_2, \ldots, b'_t)$ differing in exactly one position, say, the j-th position. Suppose that $b_j = 1$ and $b'_j = 0$. The verification equations in the two sessions then give $(r/r')^2 \equiv \pm y_j \pmod{n}$, that is, $r/r' \equiv \pm x_j \pmod{n}$. Here, we could assume different commitment values if the cheating prover knows the corresponding k values, but since we treat the cheating prover as a black box, this is not a reasonable assumption.

It follows that $O(t2^t)$ successful runs of the FFS protocol reveal all the secrets $x_1, x_2, \ldots, x_t$ to the cheating prover with high probability. If $t = O(\log_2 \log_2 n)$, then this reduction is probabilistic polynomial time (in $\log n$), that is, the zero-knowledge-ness of the protocol is PPT equivalent to the knowledge of $x_1, x_2, \ldots, x_t$. Larger values of t make the reduction super-polynomial time, and the protocol may lose its zero-knowledge property.

For leftover answers and rough work

For leftover answers and rough work