CS 6501: Advanced Topics in Cryptography **Spring 2019**

Problem Set 3

Due: March 22, 2019 at 5pm (submit via Gradescope) **Instructor:** David Wu

Instructions: You **must** typeset your solution in LaTeX using the provided template:

https://www.cs.virginia.edu/dwu4/courses/sp19/static/homework.tex

Submission Instructions: You must submit your problem set via Gradescope. Please use course code **9YD875** to sign up. Note that Gradescope requires that the solution to each problem starts on a **new page**.

Problem 1: Conceptual Questions [12 points]. For each of the following statements, say whether it is TRUE or FALSE. Write *at most 1-3 sentences* to justify your answer.

(a) Suppose the CDH assumption holds in a group $\mathbb{G}$. Then, the discrete log assumption holds in $\mathbb{G}$.

(b) Let $\mathbb{G}$ be a group of prime order p and generator g. Suppose there exists an efficient adversary $\mathcal{A}$ that solves the CDH problem in $\mathbb{G}$ with non-negligible advantage. Let $\mathcal{X}, \mathcal{Y} \subseteq \mathbb{Z}_p$ such that $|\mathcal{X}| = (p+1)/2 = |\mathcal{Y}|$. Then, there exists an efficient adversary $\mathcal{B}$ that solves CDH instances drawn from the following distribution:

$$\left\{x \xleftarrow{R} \mathcal{X}, y \xleftarrow{R} \mathcal{Y} : (g, g^x, g^y)\right\}.$$

(c) Let $\mathbb{G}$ be an elliptic group of prime order $p = 2^{O(\lambda)}$. Suppose there is an efficient algorithm (i.e., runs in time poly(λ)) that takes as input two curve points $P, Q \in \mathbb{G}$ and outputs $P + Q$ (the '+' operator here denotes the "chord-and-tangent" operation on curve points). Then, there is an efficient algorithm that computes $k \cdot P$ for any $k \in \mathbb{Z}_p$ and $P \in \mathbb{G}$. We write $k \cdot P$ to denote $\underbrace{P + P + \cdots + P}_{k \text{ times}}$.

(d) Let E be an elliptic curve over $\mathbb{F}_p$ defined by the curve equation $y^2 = x^3 + ax + b$, where $a, b \in \mathbb{F}_p$. Let $P = (x, y) \in \mathbb{F}_p^2$ be a point on the elliptic curve. If Alice wants to send the point P to Bob, it suffices that Alice send up to $\lceil \log p \rceil + 1$ bits. In your answer, you may use the fact that there exists an efficient algorithm for computing square roots in $\mathbb{F}_p$.

(e) Let $\mathbb{G}$ be a group where the DDH assumption is believed to be hard. It is possible that there exists an efficiently-computable and non-degenerate pairing $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ on $\mathbb{G}$.

(f) Let $\langle P, V \rangle$ be an interactive proof system for a language $\mathcal{L}$ with a *randomized* verifier. If $\langle P, V \rangle$ satisfies perfect completeness (i.e., completeness holds with probability 1), and perfect soundness (i.e., soundness holds with probability 1), then there is an interactive proof system for $\mathcal{L}$ with a *deterministic* verifier.

Problem 2: Homomorphic Encryption [60 points]. At a high-level, a homomorphic encryption scheme enables computation on ciphertexts. For instance, in an *additively homomorphic* encryption scheme over $\mathbb{Z}_p$, there is an efficient *public* algorithm that takes an encryption ct_0 of a message $m_0 \in \mathbb{Z}_p$ and an encryption ct_1 of a message $m_1 \in \mathbb{Z}_p$, and produces an encryption of $m_0 + m_1 \in \mathbb{Z}_p$. In this problem, we will explore several properties and constructions of homomorphic encryption schemes.

Exponential ElGamal. Consider the following "exponential" variant of ElGamal encryption with plaintext space $\mathbb{Z}_p$.

- $\mathsf{KeyGen}(1^\lambda)$: Run $(\mathbb{G}, p, g) \leftarrow \mathsf{GroupGen}(1^\lambda)$ to obtain a description of a group $\mathbb{G}$ of prime order p and generator g. Sample $s \xleftarrow{R} \mathbb{Z}_p$ and set $h \leftarrow g^s$. Output $\mathsf{pk} = (\mathbb{G}, p, g, h)$ and $\mathsf{sk} = s$.
- $\mathsf{Encrypt}(\mathsf{pk}, m)$: On input $\mathsf{pk} = (\mathbb{G}, p, g, h)$ and $m \in \mathbb{Z}_p$, sample $r \xleftarrow{R} \mathbb{Z}_p$, and output $(g^r, g^m h^r)$.
- $\mathsf{Decrypt}(\mathsf{pk}, \mathsf{sk}, \mathsf{ct})$: On input $\mathsf{pk} = (\mathbb{G}, p, g, h)$, $\mathsf{sk} = s$ and $\mathsf{ct} = (x, y)$, compute and output m such that $g^m = y / x^s$.

Note that because the decryption algorithm requires computing a discrete log, this encryption scheme is only suitable for encrypting *small* values in $\mathbb{Z}_p$ (e.g., 32-bit values). Observe that semantic security of this construction follows from the DDH assumption in $\mathbb{G}$.

(a) Show that the exponential ElGamal encryption scheme is *additively homomorphic* over $\mathbb{Z}_p$. Namely, describe an algorithm $\mathsf{Add}(\mathsf{ct}_0, \mathsf{ct}_1)$ that takes as input two ciphertexts $\mathsf{ct}_0 \leftarrow \mathsf{Encrypt}(\mathsf{pk}, m_0)$ and $\mathsf{ct}_1 \leftarrow \mathsf{Encrypt}(\mathsf{pk}, m_1)$ and outputs a new ciphertext ct such that $\mathsf{Decrypt}(\mathsf{sk}, \mathsf{ct}) = m_0 + m_1 \in \mathbb{Z}_p$. Prove that your algorithm is correct.

(b) Show that the ElGamal encryption scheme can be *refreshed*: namely, there exists an efficient algorithm Refresh such that for $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$ and all ciphertexts $\mathsf{ct} \in \mathbb{G}^2$,

$$\{\mathsf{Refresh}(\mathsf{pk}, \mathsf{ct})\} \equiv \{\mathsf{Encrypt}(\mathsf{pk}, \mathsf{Decrypt}(\mathsf{sk}, \mathsf{ct}))\}.$$

You should both describe the Refresh algorithm and prove that it satisfies this property.

From secret-key to public-key homomorphic encryption. Let $(\mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt}, \mathsf{Add}, \mathsf{Refresh})$ be a semantically-secure *secret-key* additively homomorphic encryption scheme over $\mathbb{Z}_p$ where $p = 2^{O(\lambda)}$. You should assume that this Decrypt algorithm can efficient decrypt encryptions of all messages in $\mathbb{Z}_p$ (i.e., this is *not* the decryption algorithm for the exponential ElGamal algorithm). In the secret-key setting, $\mathsf{KeyGen}(1^\lambda)$ still outputs a public key pk (used for Add and Refresh) and a secret key sk (used for Encrypt and Decrypt). The Refresh algorithm satisfies the following property:

$$\{\mathsf{Refresh}(\mathsf{pk}, \mathsf{ct})\} \overset{c}{\approx} \{\mathsf{Encrypt}(\mathsf{sk}, \mathsf{Decrypt}(\mathsf{sk}, \mathsf{ct})).\}$$

(c) Use $(\mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt}, \mathsf{Add}, \mathsf{Refresh})$ to construct a semantically-secure *public-key* encryption scheme $(\mathsf{KeyGen}_{\mathsf{PKE}}, \mathsf{Encrypt}_{\mathsf{PKE}}, \mathsf{Decrypt}_{\mathsf{PKE}})$ over $\mathbb{Z}_p$. Prove that your scheme is efficient (i.e., all algorithms run in time $\mathrm{poly}(\lambda)$), correct, and semantically secure. In particular, note that p here can be *exponential* in the security parameter λ. Note that your resulting public-key encryption scheme is likely to remain additively homomorphic and support an efficient refreshing algorithm, but you do *not* have to show these properties.

Implication: Your solution to this problem shows that secret-key homomorphic encryption (with a few additional properties) is *equivalent* to public-key homomorphic encryption. One might wonder if a similar transformation might be possible starting from a vanilla secret-key encryption to obtain a public-key encryption scheme. Here, the answer is most likely no: there are "black-box separations" between vanilla secret-key encryption and public-key encryption (namely, we cannot construct public-key encryption by using a secret-key encryption scheme as a black box).

Fully homomorphic encryption. The exponential ElGamal encryption scheme described above only supports additive homomorphism. If an encryption scheme supports arbitrarily many additions and multiplications, then we refer to it as a *fully homomorphic encryption* (FHE) scheme. FHE is a very powerful notion in that it enables *arbitrary* computation on encrypted data.

(d) Suppose we had a fully homomorphic encryption scheme over $\mathbb{Z}_p$ (for some $p > 2$). Give a brief description of how FHE can be used to perform arbitrary computation on encrypted data. Specifically, assume that the data can be represented as an n-bit string $x \in \{0,1\}^n$ where $n = \text{poly}(\lambda)$ and the computation is modeled as a polynomial-size Boolean circuit $C\colon \{0,1\}^n \to \{0,1\}$. You should describe how to encrypt the input x and how to homomorphically compute an encryption of $C(x)$ *without* knowledge of x. You may assume without loss of generality that the Boolean circuit consists only of AND gates and NOT gates (since AND and NOT gates are universal). For this problem, you just need to describe your general approach; no formal argument or proof is needed.

Somewhat homomorphic encryption from pairings. Later on in this course, we will describe how to construct fully homomorphic encryption. Here, we will explore a pairing-based encryption scheme that supports arbitrarily many additions and a *single* multiplication. Homomorphic encryption schemes that support many additions and a small number of multiplications are often called somewhat homomorphic encryption schemes.

Our construction will rely on a pairing group $\mathbb{G}$ of composite order $N = pq$, where p, q are distinct primes. Moreover, we assume that the messages to be encrypted are drawn from a small polynomial-size domain (as in exponential ElGamal). We describe the construction below:

- $\mathsf{KeyGen}(1^\lambda)$: Sample $(\mathbb{G}, \mathbb{G}_T, p, q, e, g) \leftarrow \mathsf{CompositeGroupGen}(1^\lambda)$ which outputs a cyclic group $\mathbb{G}$ of composite order $N = pq$, a target group $\mathbb{G}_T$ of order N, an efficiently-computable pairing $e\colon \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ and a generator $g \in \mathbb{G}$. Let $g_p \leftarrow g^q$ and $g_q \leftarrow g^p$. Sample $s \xleftarrow{R} \mathbb{Z}_q$ and $h \leftarrow g_q^s$. Output the public key $\mathsf{pk} = (\mathbb{G}, \mathbb{G}_T, N, g, h)$ and the secret key $\mathsf{sk} = (q, g_p)$.
- $\mathsf{Encrypt}(\mathsf{pk}, m)$: On input $\mathsf{pk} = (\mathbb{G}, \mathbb{G}_T, N, g, h)$ and $m \in \mathbb{Z}_N$, sample $r \xleftarrow{R} \mathbb{Z}_N$ and output $g^m h^r$.
- $\mathsf{Decrypt}(\mathsf{pk}, \mathsf{sk}, z)$: On input $\mathsf{pk} = (\mathbb{G}, \mathbb{G}_T, N, g, h)$, $\mathsf{sk} = (q, g_p)$ and $z \in \mathbb{G}$, compute and output m such that $z^q = g_p^m$.

(e) Show that g_q generates a subgroup $\mathbb{G}_q \subset \mathbb{G}$ of prime order q. Namely, show that $\left|\langle g_q \rangle\right| = q$.

(f) Show that the above encryption scheme is correct. You may assume that the message m is small enough that computing the discrete log in Decrypt is efficient.

(g) We say that the subgroup decision assumption holds with respect to CompositeGroupGen if for $(\mathbb{G}, \mathbb{G}_T, p, q, e, g) \leftarrow \mathsf{CompositeGroupGen}(1^\lambda)$, $N = pq$,

$$\left\{ r \xleftarrow{R} \mathbb{Z}_N : (\mathbb{G}, \mathbb{G}_T, N, e, g, g^p, g^r) \right\} \overset{c}{\approx} \left\{ r \xleftarrow{R} \mathbb{Z}_N : (\mathbb{G}, \mathbb{G}_T, N, e, g, g^p, (g^p)^r) \right\},$$

Namely, the subgroup decision assumption says that in a composite-order group, it should be difficult to distinguish a random group element from a random element of a subgroup. Show that under the subgroup decision assumption, the above encryption scheme is semantically secure.

(h) Show that the subgroup decision assumption is false if the adversary is also given g^q. In other words, show that there is an efficient adversary that can distinguish the following two distributions:

$$\left\{ r \xleftarrow{R} \mathbb{Z}_N : (\mathbb{G}, \mathbb{G}_T, N, e, g, g^p, g^q, g^r) \right\} \quad \text{and} \quad \left\{ r \xleftarrow{R} \mathbb{Z}_N : (\mathbb{G}, \mathbb{G}_T, N, e, g, g^p, g^q, (g^p)^r) \right\},$$

where $(\mathbb{G}, \mathbb{G}_T, p, q, e, g) \leftarrow \mathsf{CompositeGroupGen}(1^\lambda)$ and $N = pq$. Give a brief explanation why your attack does not apply to the standard subgroup decision assumption (where the adversary is not given g^q).

(i) Show that the above encryption scheme is additively homomorphic.

(j) Given two encryptions $\mathsf{ct}_0 \leftarrow \mathsf{Encrypt}(\mathsf{pk}, m_0)$ and $\mathsf{ct}_1 \leftarrow \mathsf{Encrypt}(\mathsf{pk}, m_1)$, describe a procedure to compute a new ciphertext $\mathsf{ct}' \in \mathbb{G}_T$ that decrypts to the product $m_0 m_1$ with respect to a new decryption algorithm $\mathsf{Decrypt}'$. Describe the operation of $\mathsf{Decrypt}'$. Your new algorithm $\mathsf{Decrypt}'$ should only take in pk, sk, and the "product" ciphertext $\mathsf{ct}' \in \mathbb{G}_T$. You should not make any modifications to the other algorithms. You may assume that decryption succeeds only if the underlying message is drawn from a small (i.e., polynomial-size) space.

Problem 3: Time Spent [5 points for answering]. How long did you spend on this problem set? This is for calibration purposes, and the response you provide will not affect your score.

Optional Feedback [0 points]. Please answer the following *optional* questions to help us design future problem sets. You do not need to answer these questions. However, we do encourage you to provide us feedback on how to improve the course experience.

(a) What was your favorite problem on this problem set? Why?

(b) What was your least favorite problem on this problem set? Why?

(c) Do you have any other feedback for this problem set?

(d) Do you have any other feedback on the course so far?