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

https://www.cs.utexas.edu/~dwu4/courses/fa22/static/homework.tex

You must submit your problem set via Gradescope (accessible through Canvas).

Collaboration Policy. You may discuss your general high-level strategy with other students, but you may not share any written documents or code. You should not search online for solutions to these problems. If you do consult external sources, you must cite them in your submission. You must include the names of all of your collaborators with your submission. Refer to the official course policies for the full details.

Problem 1: Collision-Resistant Hashing from RSA [15 points]. Let $N = pq$ be an RSA modulus and take $e \in \mathbb{N}$ to be a prime that is also relatively prime to $\phi(N)$. Let $u \xleftarrow{\$} \mathbb{Z}_N^*$, and define the hash function

$$H_{N,e,u}: \mathbb{Z}_N^* \times \{0, \ldots, e-1\} \to \mathbb{Z}_N^* \quad \text{where} \quad H_{N,e,u}(x, y) = x^e u^y \in \mathbb{Z}_N^*.$$ 

In this problem, we will show that under the RSA assumption, $H_{N,e,u}$ defined above is collision-resistant. Namely, suppose there is an efficient adversary $A$ that takes as input $(N, e, u)$ and outputs $(x_1, y_1) \neq (x_2, y_2)$ such that $H_{N,e,u}(x_1, y_1) = H_{N,e,u}(x_2, y_2)$. We will use $A$ to construct an efficient adversary $B$ that takes as input $(N, e, u)$ where $u \xleftarrow{\$} \mathbb{Z}_N^*$ and outputs $x$ such that $x^e = u \in \mathbb{Z}_N^*$.

(a) Show that using algorithm $A$ defined above, algorithm $B$ can efficiently compute $a \in \mathbb{Z}_N$ and $b \in \mathbb{Z}$ such that $a^e = u^b \pmod{N}$ and $0 \neq |b| < e$. Remember to argue why any inverses you compute will exist.

(b) Use the above relation to show how $B$ can efficiently compute $x \in \mathbb{Z}_N$ such that $x^e = u$. Note that $B$ does not know the factorization of $N$, so $B$ cannot compute $b^{-1} \pmod{\phi(N)}$. Hint: What is $\gcd(b, e)$?

Problem 2: CCA-Secure Encryption from RSA [35 points]. Recall the public-key encryption scheme from RSA that we introduced in class:

- The public key is $(N, e)$ and the secret key is $d$ where $ed = 1 \pmod{\phi(n)}$. Here, $N = pq$ is an RSA modulus. Let $H: \mathbb{Z}_N^* \to \{0, 1\}^\ell$ be a hash function which we will model as a random oracle.

- To encrypt a message $m \in \{0, 1\}^\ell$, sample $x \xleftarrow{\$} \mathbb{Z}_N^*$ and compute $y \leftarrow x^e \pmod{N}$. The ciphertext is $ct = (y, H(x) \oplus m)$.

- To decrypt a ciphertext $(y, z)$, compute $x \leftarrow y^d \pmod{N}$ and outputs $m \leftarrow z \oplus H(x)$.

In class, we argued (informally) that this scheme is semantically secure (and thus, CPA-secure) under the RSA assumption and modeling $H$ as a random oracle.

(a) Show that this encryption scheme is not CCA-secure.
or
under
pk
and Bob give their public keys to an email server. When the email server receives mail for Alice encrypted
Let (sk, pk) be Alice’s ElGamal key-pair and (skProxy, pkProxy) be the proxy key.

Problem 3: Proxy Re-Encryption [25 points]. Let G be a group of prime order p and generator g. Recall
the vanilla ElGamal encryption scheme from class: the public key is a pair of group elements pk = (g, h) and
the secret key is the exponent sk = x where h = g^x; an encryption of m ∈ G is the pair (g^t, h^t · m).

Let (pkAlice, skAlice) be Alice’s ElGamal key-pair and (pkBob, skBob) be Bob’s ElGamal key-pair. Both Alice
and Bob give their public keys to an email server. When the email server receives mail for Alice encrypted
under pkAlice, it forwards it to Alice, and correspondingly with Bob. The mail server does not know skAlice
or skBob, so it cannot decrypt Alice’s or Bob’s emails.

(a) Suppose Alice goes on vacation, and she wants to delegate her email responsibilities to Bob. Show that
Alice and Bob can compute a “proxy key” skProxy that allows the mail server to translate a ciphertext
ct encrypted under pkAlice to a new ciphertext ct’ that encrypts the same message under Bob’s public
key pkBob. Moreover, assuming semantic security of the ElGamal encryption scheme, messages
encrypted under pkAlice should remain semantically secure against the proxy even given knowledge
of skProxy together with Alice and Bob’s public keys. The proxy key can depend on skAlice and skBob.

Prove that your construction satisfies correctness and semantic security (assuming correctness and
semantic security of ElGamal encryption). For correctness, you should show that if ct decrypts
to m under skAlice, then the translated ciphertext ct’ decrypts to m under skBob. In the semantic
security game, you may assume that the adversary is given pkAlice, pkBob, and skProxy and is trying to
distinguish encryptions of m0 ∈ G from encryptions of m1 ∈ G under pkAlice.

(b) To ensure that the mail server is behaving honestly, we require the mail server include a NIZK proof
that it is translating the ciphertexts properly. The NIZK proof would be logged and independently au-
dited afterwards. Using Fiat-Shamir, it suffices to construct a Σ-protocol for the “correct” translation
procedure from Part (a). Show how to construct this Σ-protocol, and prove completeness, special
soundness, and HVZK of your protocol. Recall also that the prover must be efficient in a Σ-protocol.
You cannot use a general-purpose zero-knowledge proof for NP languages here.

In this setting, the statement contains the public keys pkAlice, pkBob and the ciphertexts ctAlice, ctBob.
A statement (pkAlice, pkBob, ctAlice, ctBob) is true if ctAlice and ctBob encrypt identical messages under
pk_{Alice} and pk_{Bob}, respectively. Note that the proxy does not know sk_{Alice}, sk_{Bob}, or the message. **Hint:** One approach is to reduce this problem to one we encountered in class. If you do this, you do not have to re-prove all of the required properties.

**Problem 4: Time Spent [2 points].** How long did you spend on this problem set? This is for calibration purposes, and the response you provide does not affect your score.

**Problem 5: Course Evaluations [3 extra credit points].** Please fill out the course evaluations (https://go.blueja.io/doymgkhSakq5ds9WjhISUg) and include a screenshot of the final confirmation screen (not the evaluation form itself). This is entirely optional and for extra credit. The curve for the class will be determined before extra credit is incorporated.

**Optional Feedback.** 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?