Problem Set 3

This problem set is due on *Wednesday, March 30, 2022* at **11:59 PM**. Please note our late submission penalty policy in the course information handout. Please submit your problem set, in PDF format, on Gradescope. *Each problem should be in a separate page.* Have **one and only one group member** submit the finished problem writeups. Please title each PDF with the Kerberos of your group members as well as the problem set number and problem number (i.e. *kerberos1_kerberos2_kerberos3_pset1_problem1.pdf*).

You are to work on this problem set in groups. For problem sets 1, 2, and 3, we will randomly assign the groups for the problem set. After problem set 3, you are to work on the following problem sets with groups of your choosing of size three or four. If you need help finding a group, try posting on Piazza or email `6.857-staff@mit.edu`. You don't have to tell us your group members, just make sure you indicate them on Gradescope. Be sure that all group members can explain the solutions. See Handout 1 (*Course Information*) for our policy on collaboration.

Homework must be submitted electronically! Each problem answer must be provided as a separate page. Mark the top of each page with your group member names, the course number (6.857), the problem set number and question, and the date. We have provided templates for LaTeX and Microsoft Word on the course website (see the *Resources* page).

Grading: All problems are worth 10 points.

With the authors' permission, we may distribute our favorite solution to each problem as the "official" solution—this is your chance to become famous! If you do not wish for your homework to be used as an official solution, or if you wish that it only be used anonymously, please note this in your profile on your homework submission.

Problem 3-1. El-Gamal In this problem we will explore a variant of the El-Gamal encyrption scheme. In class we saw a variant of El-Gamal encryption scheme that used a hash function H (we refer to the class notes for details). In this problem we look at a variant of El-Gamal that does not use a hash function. Recall that the El-Gamal encryption scheme is associated with a group G (e.g. $G = Z_p^*$) and a generator $g \in G$.

- The message space is the group G. This is in contrast to the scheme we saw in class whose message space was the range of the hash function H.
- Key Generation: sample a random $s \leftarrow \{1, ..., |G|\}$, and output $(pk, sk) = (g^s, s)$ as the key pair.
- $\text{Enc}(pk, m)$: sample a random $r \leftarrow \{1, ..., |G|\}$, and output $c = (g^r, g^{sr} \cdot m)$ as the ciphertext. Recall that we are thinking of m as an element of the group G.
- $\text{Dec}(sk, c)$: parse c as $(a, b) \coloneqq (g^r, g^{sr} \cdot m)$ and output $\frac{b}{a^s}$.

(a) Show that this encryption scheme has the desired correctness property. Namely, show that for every message $m \in G$ and for every (legal) key pair (pk, sk) generated by the key generation algorithm it holds that

$$\Pr[\text{Dec}(sk, \text{Enc}(pk, m)) = m] = 1.$$

(b) Show that this variant of El-Gamal is *not* CPA-secure if we use the group $G = \mathbb{Z}_p^*$, by presenting an attack.

(c) In class we discussed the CPA security of the El-Gamal encryption variant presented in class (with the hash function). We showed that for any group G and generator g it is CPA secure under the CDH assumption (in the random oracle model). Under which assumption on the underlying group G is this El-Gamal variant CPA secure?

Problem 3-2. Shamir Secret Sharing

For this problem each group will receive three (3) Shamir shares that you can use to reconstruct a secret value s! Each share is a point on some polynomial $f()$ of the format (x, y) where $y = f(x) \pmod p$ using prime $p = 115792089237316195423570985008687907853269984665640564039457584007913129640233$. The secret s is then of the form $(0, f(0)) \pmod p$. However, f is of degree three and each group only received three shares! Luckily, each group has received shares for the same polynomial and prime so you can work together to recover s. Less fortunately, Mallory has decided to meddle with some of the shares such that $y \neq f(x) \pmod p$. You'll need to figure out which shares have been mauled by Mallory and only use well formed shares when reconstructing s.

(a) Which share(s) from your group did Mallory maul? Describe the process you used to figure out which shares were well formed and which were not.

(b) What is the secret value s? Please report it as both an integer $\bmod p$ and converted to a string using the provided function `toPlaintext()`[1] (which takes as input $f(0)$).

(c) Explain how you recovered s from your shares.

```
from string import printable

def toPlaintext(secret_num):
    if secret_num == 0:
        return printable[0]
    secret_string = ""
    while secret_num > 0:
        secret_num, digit = divmod(secret_num, len(printable))
        secret_string += printable[digit]
    return secret_string[::-1]
```

Problem 3-3. Commitments

A commitment scheme is a digital analogue of an opaque envelope (see `https://en.wikipedia.org/wiki/Commitment_scheme`).

Formally, a commitment scheme is associated with a message space $\mathcal{M}$, a randomized efficient key generation procedure Gen that takes as input a security parameter n (think of $n = 128$) and outputs a key k,[2] and a randomized efficient algorithm Commit, that takes as input a *key* k, a *message* $m \in \mathcal{M}$, and uses *randomness* $r \in \mathcal{R}$ to output a *commitment* $\text{Commit}(k, m; r)$. We use the notation $\text{Commit}(k, m)$ to denote the distribution obtained by sampling r uniformly at random from the randomness space $\mathcal{R}$ and outputting $\text{Commit}(k, m; r)$.

The scheme is also required to have the following hiding and binding properties.

- **Hiding**: For every $m, m' \in \mathcal{M}$, the distributions $\text{Commit}(k, m)$ and $\text{Commit}(k, m')$ are indistinguishable.

  There are two flavors of the hiding property: *Computational* and *statistical*. The computational variant requires that these two distributions are *computationally indistinguishable*; i.e., *computationally bounded* adversaries cannot distinguish between these two distributions (except with negligible probability in the security parameter used by Gen). The statistical variant requires that that these two distributions are *statistically close*; i.e., even an *all-powerful* adversary cannot distinguish between these two distributions (except with negligible probability in the security parameter used by Gen).

[1] You do not need to understand what `toPlaintext()` is doing. It is just a mapping from integers to strings to allow for a textual representation of the secret.

[2] We note that for some commitment schemes Gen is not needed.

- **Binding**: An adversary that is given a key k distributed by Gen cannot find two distinct messages $m, m' \in \mathcal{M}$ along with randomnesses r, r' such that $\text{Commit}(k, m; r) = \text{Commit}(k, m'; r')$, except with negligible probability with respect to the security parameter n.

  There are two flavors of the binding property: *Computational* and *statistical*. The computational variant requires that the above binding property holds only for *computationally bounded* adversaries, whereas the statistical variant requires the binding property holds even for all-powerful adversaries.

Consider the two commitment schemes below:

(A) Gen generates a random n-bit safe prime $p = 2q + 1$, where n is the security parameter. In addition, it chooses two quadratic residues $g, h \in \mathbb{Z}_p^*$ uniformly at random. We have $\mathcal{M} = \{1, \ldots, q\}$, $\mathcal{R} = \{1, \ldots, q-1\}$, and for $k = (p, g, h)$,

$$\text{Commit}(k, m; r) = (g^m h)^r$$

where the operations are modulo p.

Assume that finding the discrete logarithm of h in base g (modulo p) is computationally intractable in general.

(B) Gen chooses a hash function $H : \{0,1\}^{3n} \to \{0,1\}^{5n}$ modelled as a random oracle, where n is the security parameter. Both $\mathcal{M}$ and $\mathcal{R}$ are $\{0,1\}^n$, and finally

$$\text{Commit}(k, m; r) = H(m||r||0^n)$$

(a) One of the schemes above is computationally hiding and statistically binding. Decide which one and justify your answer.

(b) One of the schemes above is statistically hiding and computationally binding. Decide which one and justify your answer.

(c) Argue that no commitment scheme can be both statistically binding and hiding.