**Problem 1: Malleability of ElGamal**

Remember the auction example from the lecture: Bidder 1 produces a ciphertext $c = E(pk, bid_1)$ where $E$ is the ElGamal encryption algorithm (using integers mod $p$ as the underlying group). Given $c$, Bidder 2 can then compute $c'$ such that $c'$ decrypts to $2 \cdot \text{bid}_1 \mod p$. This allows Bidder 2 to consistently bid twice as much as Bidder 1.

Now refine the attack. You may assume that $\text{bid}_1$ is the amount of Cents Bidder 1 is willing to pay. And you can assume that Bidder 1 will always bid a whole number of Euros. (I.e., $\text{bid}_1$ is a multiple of 100.)

Show how Bidder 2 can consistently overbid Bidder 1 by only 1%. What happens to your attack if Bidder 1 suddenly does not bid a whole number of Euros?

**Hint:** Remember that modulo $p$, one can efficiently find inverses. For example, one can find a number $a$ such that $a \cdot 100 \equiv 1 \mod p$.

**Problem 2: Encoding messages for ElGamal (bonus problem)**

The message space of ElGamal (when using the instantiation that operates modulo a prime $p > 2$ with $p \equiv 3 \mod 4$ and if we want to avoid the insecurity discussed in the practice) is the set $\text{QR}_p = \{x^2 \mod p : x = 0, \ldots, p - 1\}$.

The problem is now: if we wish to encrypt a message $m \in \{0, 1\}^{\ell}$ (with $\ell \leq \lfloor p/2 \rfloor - 2$), how do we interpret $m$ as an element of $\text{QR}_p$?

One possibility is to use the following function $f : \{1, \ldots, \frac{p-1}{2}\} \to \text{QR}_p$:

$$f(x) := \begin{cases} x & \text{if } x \in \text{QR}_p \\ -x \mod p & \text{if } x \notin \text{QR}_p \end{cases}$$

Once we see that $f$ is a bijection and can be efficiently inverted, the problem is solved, because a bitstring $m \in \{0, 1\}^{\ell}$ can be interpreted as a number in the range $1, \ldots, \frac{p-1}{2}$ by simply interpreting $m$ as a binary integer and adding 1 to it. (I.e., we encrypt $f(m + 1)$.)

We claim that the following function is the inverse of $f$:

$$g(x) := \begin{cases} x & \text{if } x = 1, \ldots, \frac{p-1}{2} \\ -x \mod p & \text{if } x \notin 1, \ldots, \frac{p-1}{2} \end{cases}$$

---

1 As long as $\text{bid}_1 < p/2$, that is. Otherwise $2 \cdot \text{bid}_1 \mod p$ will not be twice as much as $\text{bid}_1$. However, for large $p$, $\text{bid}_1 \geq p/2$ is an unrealistically high bid.

2 You do not actually need to use this fact, but the hint that $-1 \notin \text{QR}_p$ below is only true in this case.
We thus need to show the following: the range of $f$ is indeed $\mathbb{QR}_p$, and that $g(f(x)) = x$ for all $x \in \{1, \ldots, \frac{p-1}{2}\}$.

(a) Show that $f(x) \in \mathbb{QR}_p$ for all $x \in \{1, \ldots, \frac{p-1}{2}\}$.

**Hint:** You can use (without proof) that $-1 \notin \mathbb{QR}_p$ (this only holds in $\mathbb{QR}_p$ for $p$ prime with $p \equiv 3 \text{ mod } 4$). And that the product of two quadratic non-residues is a quadratic residue (this only holds in $\mathbb{QR}_p$, but not in $\mathbb{QR}_n$ for $n$ non-prime).

(b) Show that $g(f(x)) = x$ for all $x \in \{1, \ldots, \frac{p-1}{2}\}$.

(This then shows that $f$ is injective and efficiently invertible. Bijectivity follows from injectivity because the domain and range of $f$ both have the same size.)

**Hint:** Make a case distinction between $x \in \mathbb{QR}_p$ and $x \notin \mathbb{QR}_p$. Show that for $x \in \{1, \ldots, \frac{p-1}{2}\}$ it holds that $-x \text{ mod } p \notin \{1, \ldots, \frac{p-1}{2}\}$.

**Problem 3: Hybrid encryption – implementations**

(a) Implement a hybrid encryption using ElGamal and AES. You are allowed to use ready-made ElGamal and AES.

In the contributed file `hybrid.py` (lecture webpage), you find a prepared template in Python that already provides function for ElGamal and AES encryption as well as some utility functions and testing code that you might need. I recommend to use that code. If you wish to use another language, you will have to find your own ElGamal and AES routines.

You should check that `hybrid_decrypt(sk, hybrid_encrypt(pk, msg))` returns `msg`. It is OK if you only allow encrypting messages whose length is a multiple of 16 bytes (blocklength of AES).

(b) [Bonus problem] The ElGamal implementation used in `hybrid.py` might leak whether the message `msg` is a quadratic residue. Using the methods developed in [Prob lem 2] fix the functions `elgamal_encrypt` and `elgamal_decrypt` to avoid this leakage. (You need to make sure that `elgamal_decrypt(sk, elgamal_encrypt(pk, msg))` still returns `msg`.)