Cryptology I (spring 2018)	Dominique Unruh
Exercise Sheet 5	
Out: 2018-03-22	Due: 2018-03-30

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 $2 \cdot bid_1 \bmod p$. This allows Bidder 2 to consistently bid twice as much as Bidder 1.[1]

Now refine the attack. You may assume that 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., 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 $a \cdot 100 \equiv 1 \bmod p$.

Problem 2: An unsavory group

Recall that ElGamal can be defined with respect to many different groups. Here we give an example of a group one should not use.

Let $p > 0$ be a large prime. Let $G := \{0, \dots, p-1\}$. The group operation is defined as follows: For $a, b \in G$, let $a \cdot b := (a + b \bmod p)$.[2] Recall that a^i for $i \in \mathbb{N}$ is defined as $a^i := a \cdot a \cdot a \cdot a \cdot a \cdot \dots \cdot a$ (i-times).

(a) What is a^i written in terms of $+$? (I.e., when unfolding the definition of the group operation, and using modular arithmetic.) This should be quite a simple operation!

[1] As long as $bid_1 < p/2$, that is. Otherwise $2 \cdot bid_1 \bmod p$ will not be twice as much as bid_1. However, for large p, $bid_1 \geq p/2$ is an unrealistically high bid.

[2] This is notationally highly confusing, of course, because it looks like we claim that plus and times are the same thing. But keep in mind that we are defining a new operation on G here, and it is just a notational convention that we write it like multiplication. In particular, do not confuse $a \cdot b$ (the group operation) with $a \cdot b \bmod p$ (actual multiplication modulo p). Often one would use the symbol $+$, but that would be confusing as well because in our definitions of ElGamal we used multiplicative notation. If you wish, you can introduce a different symbol for the group operation, say $\circ$, and then the definition becomes $a \circ b := (a + b \bmod p)$. You are free to do it either way in your solution, but make sure that you do not mix up the different meanings of $a \cdot b$!

(b) Show that there is an efficient algorithm for solving the discrete logarithm problem in G. That is, given $a \in G$ and $b := a^i \in G$ (but not given i), the algorithm should compute i.

Note: You can use, without proof, the fact that there is an efficient algorithm (Extended Euclidean Algorithm, EEA) that, given p and $x \in \{1, \dots, p-1\}$ computes y with $xy \equiv 1 \mod p$. (This is multiplication modulo p, not the group operation. The fact that inverting works for all x uses that p is prime.)

(c) Show that the DDH assumption does not hold for G. (I.e., there is an efficient algorithm that distinguishes the two games from the definition of the DDH assumption with probability close to 1.)

(d) Program the algorithms from (b) and (c). You can use the following template: `additive-group.py` (on the webpage)