Exercise 8

1 Group Key Agreement

The well-known *Diffie-Hellman* protocol provides a protocol for two parties to agree on a secret key by exchanging public messages. Recall the mathematical setting of ElGamal encryption. Parties P_1 and P_2 have public keys $y_1 = g^{x_1}$ and $y_2 = g^{x_2}$, respectively, where $x_1 \in_R \mathbb{Z}_q$ and $x_2 \in_R \mathbb{Z}_q$ are the respective secret keys. P_1 sends y_1 to P_2 and P_2 computes $c_2 = y_1^{x_2}$. Analogously, P_2 sends y_2 to P_1 and P_1 computes $c_1 = y_2^{x_1}$. Note that $c_1 = c_2$. This value can now serve as their secret key, since no adversary who overheard the public messages gains any useful information about c_1. (In real life, one must use $H(c_1)$ for the secret key and take several further measures to secure the protocol against man-in-the-middle attacks.)

We want to generalize the Diffie-Hellman protocol to $n > 2$ parties, which should all obtain the same secret key.

A simple 3-party key agreement protocol for P_1, P_2, P_3 proceeds in three steps:

1. P_i (for $i = 1, \ldots, 3$) chooses $x_i \in_R \mathbb{Z}_q$ and sends $a_i = g^{x_i}$ to all;
2. P_i computes $b_{j,i} = {a_j}^{x_i}$ for $j \neq i$ and sends the b values to all; and
3. P_i computes $c_i = {b_{j,l}}^{x_i}$ for the pair (j, l) such that $j \neq i$ and $l \neq i$. Note that $c_1 = c_2 = c_3$.

At the end, every party obtains the same secret key c_1, but an adversary who observes all messages does not learn any useful information about c_1.

a) Generalize this protocol to n parties such that it takes $O(n)$ messages and the size of each message is $O(n)$ elements of G. (It will take $O(n)$ rounds.)

b) How can the size of each message be reduced to a constant number of elements from G?

c) This family of protocols assumes a model where all parties are correct, no party may even crash. The protocol protects them against an outside eavesdropper. In this model, the following *canonical* protocol would also achieve the goal of establishing a common key: one designated party chooses the group key at random; every party generates its own key pair and sends the public key to the designated party; finally, the designated party encrypts the group key

   Discuss differences between the group key-agreement protocol and the canonical protocol.