

## Exercise 10

*Note: You may answer either one of these two questions, or both, depending on your preferences. If you answer more than one, the points for the second one count as bonus points.*

### 10.1 Implementation of a distributed ElGamal cryptosystem (10pt)

Recall the implementation of the ElGamal public-key cryptosystem in an earlier exercise and the threshold ElGamal system shown in class. Recall also the  $(f+1)$ -of- $n$  secret sharing scheme implemented in Exercise 9.

Given this as background, implement a threshold ElGamal cryptosystem, consisting of the following methods:

- **KeyGen()** for generating a shared key by a dealer; this gives  $n + 1$  outputs, i.e., a global public key and a share  $sk_i$  for each party  $P_i$ , for,  $i = 1, \dots, P_n$ ;
- **Enc**( $pk, m$ ), which encrypts a given message  $m$ ;
- **Dec**( $pk, sk_i, c$ ), which produces a decryption share  $d_i$  for party  $P_i$  and with respect to a ciphertext  $c$ ; and
- **Recover**( $pk, D, c$ ), which decrypts ciphertext  $c$  to a message  $m$ , using a list  $D$  of  $f + 1$  decryption shares.

Use again Python, Java, Golang, or C++. Note that the ElGamal encryption scheme used earlier (Exercise 4) was the “textbook” scheme, which encrypts only messages that are elements of  $G$ . You may continue to use this “textbook” scheme or change it to “Hashed ElGamal,” which encrypts messages in  $\{0, 1\}^k$  for  $k = 256$  or  $k = 512$ , as specified in the handout *Distributed Cryptography*. (Please send source code in zip format and name it like SurnameFirstname10).

### 10.2 Verifiable secret sharing (10pt)

Secret sharing as presented in class is *not robust*. This means that it fails against active attacks, whether they are mounted by the dealer  $D$  or by some party  $P_i$ :

- In particular, a corrupted dealer might send inconsistent shares  $s_i$  to the parties, so that the reconstructed secret differs depending on which parties contribute to the secret reconstruction. (Of course, a corrupted dealer might also leak the secret, but secret sharing is often used inside another protocol, where every party shares a secret; there it only matters that some majority of the parties are correct.)
- Furthermore, parties might contribute wrong values during secret reconstruction (or when running a threshold-cryptographic operation).

The following *verifiable secret-sharing (VSS) protocol* (known as *Feldman VSS*) prevents an attack by the dealer.

- a) When  $D$  uses a polynomial  $a(X) = \sum_{k=0}^f a_k X^k$ , with  $a_0 = s$  to share a secret  $s$ , it additionally computes  $A_k = g^{a_k}$  for  $k = 0, \dots, f$ . It sends the share  $s_i$  privately to  $P_i$  and publishes or broadcasts the vector  $[A_0, \dots, A_f]$  (in a consistent way, so that it is received by all parties).

- b) After receiving a share  $s_j$  from  $D$  and a vector  $[A_0, \dots, A_f]$ , party  $P_j$  verifies that they are consistent by computing (in  $\mathbb{Z}_q$ )

$$g^{s_j} \stackrel{?}{=} \prod_{k=0}^f (A_k)^{j^k}.$$

Then  $P_j$  broadcasts a message that is either ACCEPT or REJECT, depending on the outcome of this check. (Notice that if  $P_j$  is faulty, it might report a wrong outcome here, which is not justified.)

- c) If party  $P_j$  observes at least  $2f + 1$  ACCEPT messages, then it *accepts* the sharing by  $D$ . Otherwise, it *rejects* it.

Furthermore, the verification equation is used during the reconstruction protocol, to prevent malicious parties from submitting wrong shares. This means that any share  $s_j$  contributed by a party  $P_j$  is only used for reconstruction if it satisfies the above consistency check.

Show that this protocol is *robust* as long as  $n > 3f$ . This means that whenever a correct party has accepted the sharing, then there exists some  $\bar{s}$  such that any execution of the reconstruction protocol outputs  $\bar{s}$  (when dealer  $D$  is corrupt, this value is not necessarily the  $s$  used by  $D$ ). Notice that when the party receives  $2f + 1$  ACCEPT messages, then at least  $f + 1$  *correct* parties have broadcast such messages.

Show also that the protocol maintains *privacy* of the secret, i.e., when  $D$  is correct, then no corrupted party learns more about  $s$  than  $A_0 = g^s$ .

Last but not least, also show *completeness* of this protocol.