

## Exercise 4

### 1 Efficient causal broadcast

Algorithm 4.18 for causal broadcast from FIFO broadcast suffers from impractically large messages, which grow in proportion to the number messages sent by a party. Provide a more efficient implementation directly using reliable broadcast. (Use a vector clock for causal ordering.)

### 2 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. Consider a cyclic group  $G = \langle g \rangle$  of prime order  $q$ , such that  $g$  is a generator of  $G$ . (For example,  $G = \mathbb{Z}_p^*$  for some prime  $p = mq + 1$ , where  $|p| = 1024$  and  $|q| = 160$ , is considered moderately secure today.)

Computing *discrete logarithms* in  $G$  is assumed to be hard, i.e., any efficient algorithm computes  $x$  from  $y$  and  $g$  such that  $g^x = y$  only with negligible probability. (In other words, exponentiation in  $G$  is a *one-way function*.) The *Diffie-Hellman problem* in  $G$  is also assumed to be hard, i.e., any efficient algorithm computes  $g^{x_1 x_2}$  from  $y_1 = g^{x_1}$  and  $y_2 = g^{x_2}$  only with negligible probability. Finally, the *Decisional Diffie-Hellman problem* is presumably also hard, i.e., any efficient algorithm distinguishes triples  $(g^{x_1 x_2}, g^{x_1}, g^{x_2})$  for random  $x_1, x_2 \in \mathbb{Z}_q$  from triples  $(g^z, g^{x_1}, g^{x_2})$  for random  $z, x_1, x_2 \in \mathbb{Z}_q$  only with negligible probability.

A simple 3-party key agreement protocol for  $P_1, P_2, P_3$  proceeds in three steps: (1)  $P_i$  (for  $i = 1, \dots, 3$ ) 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; (3)  $P_i$  computes  $c_i = b_{j,l}$  for some 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  $c$ , but an adversary who observes all messages does not learn any useful information about  $c$ .

- Generalize this protocol to  $n$  parties such that it takes only  $O(n)$  messages and that the size of each message is only  $O(n)$  elements of  $G$ . (It will take  $O(n)$  rounds.)
- How can the size of each message be reduced to a constant number of elements from  $G$ ?
- How can the number of rounds be reduced to a constant?