Cryptographic Protocols Homework

Submit your solutions through courses.cs.ut.ee. In case of any questions or any mistakes in this homework setup contact Pille Pullonen-Raudvere.

Consider the sharing scheme that we denoted as $\langle x \rangle$ in the SPDZ protocol description. It was defined as follows:

- $x \in \mathbb{F}$ is additively secret shared as $x = \sum_{i=1}^{n} x_i \in \mathbb{F}$
- Each party $P_i$ for $i \in \{1, \ldots, n\}$ has their private message authentication code key $\beta_i \in \mathbb{F}$
- Each shared value $x$ has a MAC tag $MAC(x, \beta_i) = x \cdot \beta_i$ that is kept as additive secret shares $(x\beta_i)_j$ where $x\beta_i = \sum_{j=1}^{n} (x\beta_i)_j \in \mathbb{F}$
- Party $P_i$ holds share $(x_i, (x\beta_1)_i, \ldots, (x\beta_n)_i)$ and the key $\beta_i$

Think of the similarities of this representation and the SPDZ share representation and solve the following exercises using the ideas from SPDZ preprocessing:

- Propose a semi-honestly secure protocol for the parties $P_1, \ldots, P_n$ to prepare random values $\langle r \rangle$ that none of the parties know in this scheme (e.g. none of the parties learn $r$ during the generation of $\langle r \rangle$). Essentially this is a preprocessing protocol that was missing from the lecture slides. In the lectures we just assumed that such values are available. Your protocol should do the following (but not necessarily in this order):
  - In the beginning of the protocol each party has $\beta_i$ (and some setup information, e.g. cryptographic keys, if needed by your scheme)
  - Generate the random values $r_i$
  - Generate the shared MACs where party $P_j$ holds a share $(r\beta_i)_j$ for each key $\beta_i$
  - In the end of the protocol each party $P_i$ holds $(r_i, (r\beta_1)_i, \ldots, (r\beta_n)_i)$
  - The value $r = \sum r_i$ is a uniformly random value in $\mathbb{F}$
- Write down the necessary properties of the building blocks you are using and describe the setup (e.g. previously distributed keys). Please provide references if using something not covered in the lectures.
  - You can assume existence of suitable encryption or OT schemes as we did for the triple preprocessing protocols in the lecture. However, write down what it means to be suitable.
- E.g. when using public key encryption you can consider that the public keys have been securely distributed before your protocol starts. Write down the needed assumptions.
- Write down the needed dependencies in the data structures.

- Show the correctness of your construction. Why is it a random value and why are the MACs correct if everyone behaved honestly?
  - Write all the computations out and show that the MAC-s are computed correctly.

- Does any party $P_i$ see anything other than the desired output $r_i, ((r\beta_1)_i, \ldots, (r\beta_n)_i)$ and the key $\beta_i$ that it has? Is there any chance that something about other parties share or secret key leaks to party $P_i$? Why/why not?
  - In other words, if the parties in this protocol are honest-but-curious (following the protocol but thinking about all the values that they see) then could they learn something other than the defined output?

- What are the parts in this protocol that could fail if any of the parties or any group or parties is actively corrupted? How might you mitigate these attacks and secure your protocol against active corruption?
  - Is there a possibility of the protocol not finishing (e.g. abort decision in the protocol)?
  - Is there a possibility that the output values are wrong?
  - Is there a possibility that the value is not random if there is active corruption?
  - Is there a possibility for selective failure?