Consider the sharing scheme that we denoted as $\langle x \rangle$ where:

- $x \in \mathbb{F}$ is additively secret shared as $x = \sum_{i=1}^{n} x_i \in \mathbb{F}$
- Each party $P_i$ has their private message authentication code key $\beta_i \in \mathbb{F}$
- Each shared value $x$ has a MAC $x \cdot \beta_i$ 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 $\beta_i$

Do the following:

- Propose a semi-honestly secure protocol for the parties $P_1, \ldots, P_n$ to prepare random values $\langle \langle r \rangle \rangle$ that none of the parties know in this scheme (e.g. none of the parties learn $r$ during the generation of $\langle \langle r \rangle \rangle$). 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:

  - In the beginning of the protocol each party has $\beta_i$ (and some setup information 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 $r_i, ((r \beta_1)_i, \ldots, (r \beta_n)_i)$

- 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.
  - E.g. when using public key encryption you can consider that the public keys have been securely distributed before your protocol starts. However, write down these setup assumptions.

- Show the correctness of your construction. (i.e. Why is it a random value and why are the MACs correct if everyone behaved honestly?)
• 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 $\beta_i$ it has)? Is there any chance that something about other parties share or secret key leaks to party $P_i$? Why/why not?

• What are the parts in this protocol that could fail if any of the parties is actively corrupted? How might you mitigate these attacks?