Issues to take into account while running the different protocols

* We identified three distinct situations that need to be addressed.

1. Start point🡪 how to synchronize starting point for both sides.
2. Middle of protocol 🡪 how long each side waits for the other to respond? What if no response has been sent in a too long period? Timeouts, etc. What to do when there is a timeout, to abort, to send a message to other party requesting to re-send?
3. End point🡪 when party receives output how does the other party know?

In general, we need to know for each protocol for each of these points how to get in of the situation and how to get out.

* Another very important issue is what to do when wrong inputs are given. In general the side that got the wrong input will send a message to the other side about it and will abort. However, we need to specify for each protocol.
* For certain protocols some of the computations performed by one side (or both) don’t depend on the other side and can be performed “offline” while waiting for the other side to response. When we look at the protocol as a series of operations we may not realize of this. As an example, when checking the validity of the input, the checking side may postpone the test to an idle stage, while waiting for the other side, if the source of the input is a known reliable source – e.g. a library- and not the other side.
* After writing down the OR of any Sigma-protocols from the point of view of each side, we came up with a few ideas.

1. It seems logical that any protocol that inherits or implements a PROTOCOL needs to have a start/run method. The default might be to call the start/run method at construction time (or immediately after), but there are times where we do not want it to start until some event happens. For example for ZK that uses commitment scheme and sigma-protocol, they do not need to start until a certain point.
2. When one of the sides needs to choose a random value it can do it on idle time.

* Every Sigma-protocol IS a Proof of Knowledge. To convert a Sigma-protocol to Zero Knowledge all we need to do is to commit to *e* at the beginning of the Sigma-protocol.

However, if we commit to *e* we cancel the first property of Sigma being a Proof of Knowledge.

We came up with a few possible ways of viewing the relationship between ZK and Sigma-protocol. We chose two in the end.

Note: The following drawings are NOT programming classes, rather they are concepts that the model the above statements.

Option 1:



Option 2:

