1. Integration

Overview

This is the final assignment in the bridge project.

In this assignment, you will be integrating all the bridge code you've written so far into a working bridge.

Overview

1. In Bridge I, you created accounts on the Avalanche and BNB testnets, and signed messages
2. In Bridge II, you created the contract on the destination side of the Bridge
3. In Bridge III, you created the contract on the source side of the Bridge
4. In Bridge IV, you create an off-chain "listener" that could listen for events emitted by a contract

In this assignment, you need to put all these pieces together so that when you see an event on the source (destination) side, you sign a message and send it to the destination (source) side.

Assignment

1. Deploy the contracts. To do this you will need testnet funds (which you should have gotten in Assignment I)
    1. Deploy the source contract from Assignment III to the source chain (Avalanche Testnet)
    2. Deploy the destination contract from Assignment II to the destination chain (BNB Testnet)

2. Augment your "listener" script that you wrote in Assignment IV so that it when it hears an event from one of the bridge contracts, it calls the appropriate function on the other contract. To do this you will need testnet funds (from Assignment I)
    1. Call the "wrap()" function on the destination contract when you see the "Deposit" event emitted by the source contract
    2. Call the "withdraw()" function on the source contract when you see the "Unwrap" event emitted by the destination contract

3. Deploy a "test" token on the Source side that can be bridged over

> Because blockchains are slow, and Codio autograders have a tendency to timeout. You will need to deploy your contracts beforehand, and the autograder will only test whether you can pass messages between the contracts.

1. Integration

Concretely, you need to:

1. Deploy the source and destination contracts and record their addresses, as well as the signing key of the deployer (the bridge “warden”). Put the deploy addresses into contract_info.json, where they can be read by our autograder (the file also contains the ABI of the contracts that will be useful for you).
2. Register the appropriate ERC20 tokens by calling the "registerToken()" function on your deposit contract, with the two token relevant token addresses that are in the erc20s.csv
3. Complete the file "bridge.py" to pass messages back and forth. The bulk of this script will be adapted from your “listener.py” script (Assignment IV), but it will need access to the signing key of the “warden” address, as well as the addresses of the source and destination contracts that you deployed.

In a real deployment, your “bridge.py” script would constantly monitor the blockchain. To save you the trouble of running some kind of server, our autograder will simply run your bridge.py script after it has made transactions on the one chain, then it will check whether the appropriate action was taken on the other chain.

Our autograder will work with your python file (“bridge.py”) and will require your contract info (“contract_info.json”). You do **not** need to provide any solidity files (the autograder will only interact with the contracts that you have deployed on chain).

```
Make sure to deploy your source contract to the Avalanche tes
```

Next