Module 1: Foundations and Beginner Solidity

Learning Objectives

- Understand the essential cryptography functions and how the blockchain works at a theoretical level
- Learn the basics of solidity and implement an ERC-20 token with non-standard features

Suggested Materials

- Foundations
    - Hash Function
    - Public Signatures
    - ECDSA Signatures
    - Byzantine Consensus
    - How Bitcoin Actually Works Part 1, Part 2, Part 3
    - 51% attacks and double spending
    - Differences between Ethereum and Bitcoin (easy)
    - What is a smart contract (easy)
    - Ethereum Protocol by Vitalik Buterin
- Solidity
    - RareSkills: Solidity the weird Parts
    - Solidity Playlist
- ERC20
    - There is a lot of material online that is easy to find. 😊
    - You are strongly encouraged to explore this repository before diving into an implementation

What is Due at the End of the Week

All of the tokens below must have a maximum supply of 100 million and a decimal place of 18. Be sure you understand what this means in the context of ERC20!

- [ ] ERC20 with sanctions. Create an ERC20 token that allows an admin to ban specified addresses from sending and receiving tokens.

- [ ] ERC20 with god mode. A special address is able to transfer tokens between addresses at will.

- [ ] Simple tokens sale. Build an ERC20 with the above features that sells tokens at a conversion rate of 10,000 tokens to 1 ethereum. The total supply should be 22 million tokens.

- [ ] Token sale and buyback with bonding curve. The more tokens a user buys, the more expensive the token becomes. To keep things simple, use a linear bonding curve. People should be able to sell their token to the contract at a 10% loss.
    - [ ] Keep track of the 10% loss and make this ETH withdrawable by the owner
    - [ ] You can use this article as a starting point, but you are encouraged to do your own research on how these works. https://www.linumlabs.com/articles/bonding-curves-the-what-why-and-shapes-behind-it

Where students commonly mess up

- Not putting proper access controls on the functions
- Using magic numbers (numbers instead of constant variables)
- Not assigning the keyword `constant` or `immutable` to variables that are never updated.
- Using public function modifiers when external is sufficient
- Ignoring the decimal place
- Not using the OpenZeppelin implementation of ERC20
- Rebuilding useful functions OpenZeppelin already provides like _beforeTokenTransfer()
- Writing large numbers as 1000000 instead of 1_000_000
- Floating pragma (learn more)