BlockBloom

Assignment – 2

Aarav Oswal – 230012

1. Metamask Address: 0xb299d668A2E852008A20eC721C1Bd1Dbd34ACBAa
2. Summary of Etherium Whitepaper:

Introduction:

Ethereum was proposed by Vitalik Buterin in 2013 as a decentralized platform that extends the functionality of blockchain technology beyond cryptocurrencies. While Bitcoin introduced the concept of a distributed ledger, Ethereum sought to create a "world computer" capable of running decentralized applications (dApps) through smart contracts.

Limitations of Bitcoin and Need for Ethereum:

Bitcoin's scripting language is limited in functionality, designed primarily for financial transactions. Ethereum addresses this by introducing a Turing-complete programming language, allowing developers to create and deploy custom decentralized applications with complex functionalities directly on the blockchain.

Key Concepts:

Smart Contracts:

Smart contracts are self-executing agreements with the terms directly written in code. They enable automated, trustless interactions between parties without the need for intermediaries. For instance, a crowdfunding contract automatically refunds contributions if a funding goal isn't reached.

Ethereum Virtual Machine (EVM):

The EVM is a runtime environment for executing smart contracts. It ensures that every node in the Ethereum network can agree on the results of computations, making the system robust and secure.

Ether (ETH):

Ether is Ethereum's native cryptocurrency, used to pay for computational resources and transaction fees within the network. This mechanism prevents abuse by requiring users to pay for the computational power they consume.

Decentralized Applications (dApps):

dApps run on the Ethereum blockchain and are transparent, tamper-proof, and censorship-resistant. They cover various domains, such as finance (DeFi), gaming, supply chain management, and voting systems.

Key Components:

Gas:

A unit that measures computational effort. Every operation in Ethereum consumes gas, ensuring the network's stability and discouraging malicious overuse.

Consensus Mechanism:

Ethereum initially used Proof of Work (PoW) but has transitioned to Proof of Stake (PoS) for greater energy efficiency and scalability.

Use Cases:

Ethereum is designed to support a wide range of applications, from decentralized finance (DeFi) platforms and non-fungible tokens (NFTs) to supply chain solutions and governance frameworks.

Conclusion:

Ethereum envisions a global, open, decentralized platform where trustless systems replace traditional centralized infrastructures. Its aim is to empower individuals and communities by reducing reliance on intermediaries.

1. Contract Address: 0x07eebd9095fd2f57c2b2f563561c55b843f0f1df
2. A Cryptocurrency Wallet is an application that functions as a wallet for your cryptocurrency. It is a tool that allows a user to store, manage, and interact with their cryptocurrency and blockchain-based assets securely. It is called a wallet because it is used similarly to a wallet you put cash and cards in. Instead of holding these physical items, it stores the passkeys you use to sign for your cryptocurrency transactions. The wallet also provides an interface for accessing and managing your crypto holdings.

There are 2 types of Wallets:

* 1. Software Wallet: Software wallets include applications for desktops and mobile devices. They can be accesssible via the internet(hot) or offline(cold). Eg: MetaMask, rust Wallet, Coinbase Wallet
  2. Hardware Wallet: Hardware wallets are physical devices designed specifically to store private keys offline. They provide enhanced security by keeping keys isolated from online threats. Eg: Ledger, Trezor, SafePal.

Wallets don’t break the principal of decentralization because wallets don’t hold funds. They simply store private keys, giving users control of their assets. The actual assets still reside on the decentralized blockchain. Also, non-custodial wallets such as MetaMask don’t rely on a cental entity for operation, thus maintaining decentralization.

Some other wallets are: Trust Wallet, Coinbase Wallet, Exodus, Ledger, Trezor, SafePal.

1. The Zero Address in blockchain systems is a special adress represented as 0x00...000. The Zero Address is relevant in the following sense:
   1. Burn Address: The Zero Address is often used as a “burn” adress for tokens, allowing developers to implement token destruction or burning mechanisms. When token are sent to the Zero Address, they are effectively removed from circulation.
   2. Contract Creation: In the context of Solidity, when a transaction is created, if the target account is set to null (zero address), it creates a new contract. The address of the contract is not the zero address but an address derived from sender and its nonce. Sending funds to the zero address indicates the miners to create a new smart contract.
   3. Error Handling: Smart contracts offen use the Zero Address to check for invalid addresses or errors in input.

The Zero address is not associated with any private key. However, if someone managed to get the private key to the Zero Address, they would have full control of all the ether/tokens belonging to that address. This would be a lot of ether as the Zero Address has been used as a token burning address for a long time. However, finding the private key is practically impossible. Firstly, because it is effectively impossible to reverse-engineer a private key from a given address, including the Zero Address. Secondly, trying to find the private key by brute force is impractical as the private-keys are 256-bit numbers and the Ethereum address space is 2^256, making it computationally infeasible.

1. The Brave Browser is a free and open-source web browser developed by Brave Software, Inc. It is designed to provide a fast, private, and secure browsing experience. Brave blocks trackers, ads, and malware by default, and users can opt-in to view privacy-preserving ads and earn cryptocurrency rewards. It aims to empower users by giving them control over their online data while enabling an innovative advertising model.

BAT stands for Basic Attention Token. BAT is the cryptocurrency token native to the Brave ecosystem. It is an ERC-20 token built on the Ethereum blockchain, designed to reward users, advertisers, and publishers within the Brave ecosystem. In the Brave Browser, users can choose to view privacy-preserving ads and earn BAT tokens as a reward. Advertisers compensate for their advertising campaigns in BAT tokens, which are then distributed to users (70%) and publishers (30%). The BAT ecosystem uses Zero-Knowledge Proof (ZKP) protocols to verify user attention and engagement, ensuring a fair and transparent system.

1. Contract Address: 0x5c30a184cF383911B4C81B8C84297f6E69a2d73a
2. Python Program:

from eth\_keys import keys

import os

def generate\_ethereum\_address():

private\_key\_bytes = os.urandom(32)

private\_key = keys.PrivateKey(private\_key\_bytes)

public\_key = private\_key.public\_key

address = public\_key.to\_address()

print("Private Key:", private\_key)

print("Public Key:", public\_key)

print("Ethereum Address:", address)

generate\_ethereum\_address()

1. Python Program:

from web3 import Web3

from eth\_utils import to\_checksum\_address

import hashlib

def generate\_contract\_address(sender\_address, nonce):

w3 = Web3()

sender\_address\_bytes = bytes.fromhex(sender\_address[2:])

nonce\_bytes = nonce.to\_bytes(32, 'big')

data = sender\_address\_bytes + nonce\_bytes

contract\_address\_hash = w3.keccak(data)

contract\_address = contract\_address\_hash[-20:]

contract\_address = to\_checksum\_address('0x' + contract\_address.hex())

return contract\_address

sender\_address = '0x742d35cc6634c0532925a3b844bc454e4438f44e'

nonce = 5

contract\_address = generate\_contract\_address(sender\_address, nonce)

print("Generated Contract Address:",contract\_address)

sender\_address = '0xb299d668A2E852008A20eC721C1Bd1Dbd34ACBAa'

nonce = 0

contract\_address = generate\_contract\_address(sender\_address, nonce)

print("Generated Contract Address:",contract\_address)

1. Python Program:

from web3 import Web3

from eth\_account import Account

w3 = Web3(Web3.HTTPProvider("https://mainnet.infura.io/v3/YOUR\_INFURA\_PROJECT\_ID"))

def sign\_transaction(transaction, private\_key):

signed\_tx = w3.eth.account.sign\_transaction(transaction, private\_key)

return signed\_tx.rawTransaction.hex()

def verify\_transaction(signed\_tx, sender\_address):

signed\_tx\_obj = w3.eth.account.decode\_raw\_transaction(bytes.fromhex(signed\_tx[2:]))

recovered\_address = w3.eth.account.recover\_transaction(signed\_tx\_obj)

return recovered\_address.lower() == sender\_address.lower()

if \_\_name\_\_ == "\_\_main\_\_":

private\_key = "0xa792f1dcfbb42f04b4014838ec348676f276d3b60af04ad7712aba55224c7230"

sender\_address = Account.from\_key(private\_key).address

transaction = {

'nonce': 0,

'to': '0xb299d668A2E852008A20eC721C1Bd1Dbd34ACBAa',

'value': w3.to\_wei(0.01, 'ether'),

'gas': 21000,

'gasPrice': w3.to\_wei(50, 'gwei'),

'chainId': 1

}

signed\_tx = sign\_transaction(transaction, private\_key)

print("Signed Transaction:", signed\_tx)

is\_valid = verify\_transaction(signed\_tx, sender\_address)

print("Is the transaction valid?", is\_valid)