Blockchain Technologies and Cryptocurrencies (Individual Assessment)

SUBMITTED BY: SARIKA PATEL
STUDENT ID: 20196958
WORD COUNT: 2248

Q1. Bitcoin

a) Imagine that you are the CEO and have invested 100BTC into a company and run it with your other three business partners Alice (20.00%) and Bob (20.00%), and Coo (20.0%) respectively.
I. If the company uses a bitcoin wallet to store funds, who should control the keys?

In a company where ownership is shared among partners with significant stakes, it is prudent to utilize a multi-signature (multi-sig) wallet for managing Bitcoin funds. For a setup with Alice, Bob, and Coo as partners, a 3-of-4 multi-sig arrangement is required. This means that any transaction would require the approval of at least three partners, ensuring collective decision-making and preventing any single individual from unilaterally accessing the funds. This setup balances security with operational efficiency, protects against both external and internal threats, and upholds the principles of democratic governance within the company.

II. Is there a solution to this problem?

Yes, utilizing a multi-signature (multi-sig) wallet is an effective solution for securely managing Bitcoin funds in a business partnership. This approach enhances security by requiring multiple keys to approve transactions, thus preventing unauthorized access. It also introduces checks and balances by necessitating the consensus of at least three out of four partners (in a 3-of-4 setup), ensuring that no single partner can unilaterally control the funds, thereby fostering collaboration and preventing misuse. Additionally, multi-sig wallets mitigate risks associated with the loss or compromise of a single key, as other keys are still needed to authorize transactions. Overall, a multi-sig wallet provides a secure and equitable method for managing shared Bitcoin funds among business partners.

b) How can they prove their identity in a)?

In the context of managing a multi-signature Bitcoin wallet, each partner can prove their identity using their unique private keys. When a transaction is initiated, each partner must use their private key to sign the transaction digitally. This digital signature serves as a cryptographic proof of identity, confirming that the partner who signed the transaction is indeed the holder of the corresponding private key.

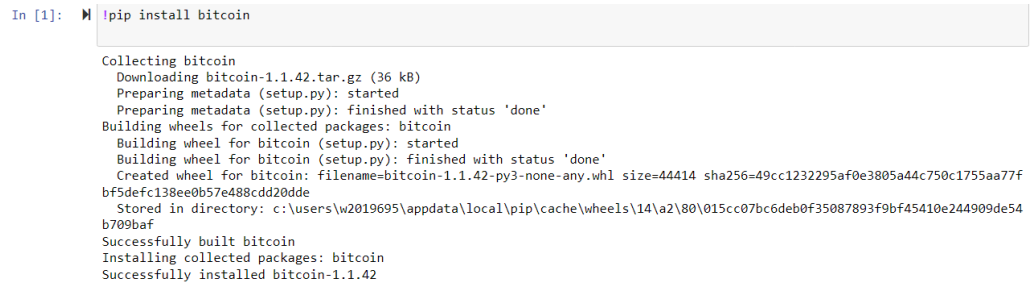
To enhance security and verify these transactions reliably, two-factor authentication (2FA) can also be implemented, requiring a second form of verification (like a one-time password) in addition to the digital signature. This approach ensures that all transactions are securely authenticated, maintaining both the integrity and the security of the operations within the Bitcoin wallet.

c) Implement the case a) in Python. The screenshot of Jupyter notebook is needed here and the code should be explained step-by-step.

Below is a step-by-step explanation and the corresponding code to implement the multi-signature wallet:

First, we installed the **bitcoin** library using! **pip install bitcoin**, which provides Bitcoin-related functionalities in Python (Figure 1).

Figure: 1



Then we have imported all functions from the **bitcoin** library (Figure:2), which provides essential tools for Bitcoin-related operations. It then generates four random private keys using the **random_key()** function, one for each business partner (CEO, Alice, Bob, and Coo). These private keys are subsequently converted to their corresponding public keys with the **privtopub()** function. Finally, the code prints both the private and public keys to the console for educational purposes, emphasizing that private keys should not be exposed in a real-world scenario to maintain security.

Figure:2

```
In [2]:
from bitcoin import *

# Generate private keys for CEO, Alice, Bob, and Coo
private_keys = [random_key() for _ in range(4)]
public_keys = [privtopub(key) for key in private_keys]

# Print the keys (for educational purposes; do not expose private keys in production)
print("Private Keys:", private_keys)
print("Public Keys:", public_keys)
```

```
Private Keys: ['2c2f8644183a678a0255ca6e29973ccaa635e56d99fbe042f9eef29b08698c4c', '8c19b5f70cc249c6cf7b728054f91ff356f9a053
fdd90f2ccb4d0335101bfbd1', 'e1ea5ca0a35f2ee069bf5eafc4302ea7baba1fcdbe0c5d84811b9dfd30b3efb3', 'c5d6deaaace5955782431cccf20b
77ba71521573054afecbae98998f225fffa7']
Public Keys: ['041e5821246b749d54e88fe47ec75220f4516c07e586fbff46d842e92910828616955aa22f3c2eb59cbc7ba7e47cb9d287106a51e95ba
91493973911234 39e5170', '04bed6618d3397277065ddc8280c8d249f659cfd76f3d9a787fdec79fa076c959a7f275040d6e6d8528a2af990225a4830b
1f27e129190d26fa81f41bc51e73699', '043144142c08beb8491d73a7a9c4cc0a863aeeec3bff5f1187121e1c4dabc48c4da0d5f2e585a4c04d36d29a2
e1590a50667b8757031aef9bb01a14814f9e232ec', '0415c0c8c9ad6c800987235b19e67906f52b6c6ea5b1733c298eba37b2ebaa307893fae8234d6c7
181e317c623a88f2134564d2a3773594460586f055152953135']
```

Then further we created a multi-signature Bitcoin address by first generating a multi-signature script with the **mk_multisig_script()** function(Figure 3), which takes the previously generated public keys and specifies that 3 out of 4 signatures are required to authorize a transaction. This script is then converted into a multi-signature Bitcoin address using the **scriptaddr()** function. Finally, the multi-signature address is printed to the console, providing a secure address that requires multiple approvals for transactions, enhancing security for the involved business partners.

Figure:3

```
In [3]:
# Create a multi-signature address
script = mk_multisig_script(public_keys, 3, 4)  # 3 of 4 signatures required
multi_sig_address = scriptaddr(script)

print("Multi-Signature Address:", multi_sig_address)
```

```
Multi-Signature Address: 35UUB8QPE7u4m7ZAXRB96jQ3P4opajzUCd
```

Then code segment begins by re-importing necessary functions from the **bitcoin** library and redefines the private and public keys, explicitly providing specific key values. It then reuses the pre-existing multi-signature address. Using the **mk_multisig_script()** function, a multi-signature script is created again(Figure 4), which requires 3 out of the 4 provided public keys to sign transactions. For educational purposes, placeholder transaction details are set up, including a dummy transaction ID (**tx_id**) and index (**tx_index**). The inputs list specifies the transaction output and its value, associating it with the multi-signature address, while the outputs list defines a destination address and the amount to be sent, reserving a small portion for transaction fees. Finally, the code prints the multi-signature address, input, and output details, noting that in practical scenarios, the placeholder values should be replaced with actual transaction data from an Unspent Transaction Output (UTXO).

Figure:4

```
from bitcoin import *

# Use the provided private and public keys
private_keys = [
    '2c2f8644183a678a0255ca6e29973ccaa635e56d99fbe042f9eef29b08698c4c',
    '8c19b5f70cc249c6cf7b728054f91ff356f9a053fdd90f2ccb4d0335101bfbd1',
    'e1ea5ca0a35f2ee069bf5eafc4302ea7baba1fcdbe0c5d84811b9dfd30b3efb3',
    'c5d6deaaace5955782431cccf20b77ba71521573054afecbae98998f225fffa7'
]
public_keys = [
    '041e5821246b749d54e88fe47ec75220f4516c07e586fbff46d842e92910828616955aa22f3c2eb59cbc7ba7e47cb9d287106a51e95ba9149397391
    '04bed6618d3397277065ddc8280c8d249f659cfd76f3d9a787fdec79fa076c959a7f275040d6e6d8528a2af990225a4830b1f27e129190d26fa81f4
    '043144142c08beb8491d73a7a9c4cc0a863aeeec3bff5f1187121e1c4dabc48c4da0d5f2e585a4c04d36d29a2e1590a50667b8757031aef9bb01a14
    '0415c0c8c9ad6c800987235b19e67906f52b6c6ea5b1733c298eba37b2ebaa307893fae8234d6c7181e317c623a88f2134564d2a3773594460586f6
]

# Use the provided multi-signature address
multi_sig_address = '35UUB8QPE7u4m7ZAXRB96jQ3P4opajzUCd'

# Create a multi-signature script
script = mk_multisig_script(public_keys, 3, 4)  # 3 of 4 signatures required

# Assume placeholder transaction details for educational purposes
# Normally you would have real transaction output data here
tx_id = 'dummy_tx_id'  # Placeholder transaction id
tx_index = 0  # Placeholder index of the output in the previous transaction
inputs = [{'output': f'{tx_id}:{tx_index}', 'value': 10000, 'address': multi_sig_address}]
outputs = [{'address': 'destination_address', 'value': 9500}]  # Send to a destination address, leave some for a fee

# Print inputs and outputs for educational purposes
print("Multi-Signature Address:", multi_sig_address)
print("Inputs:", inputs)
print("Outputs:", outputs)

# Explain that in real code, you would replace 'dummy_tx_id' and 'tx_index' with actual values from a UTXO
```

```
Multi-Signature Address: 35UUB8QPE7u4m7ZAXRB96jQ3P4opajzUCd
Inputs: [{'output': 'dummy_tx_id:0', 'value': 10000, 'address': '35UUB8QPE7u4m7ZAXRB96jQ3P4opajzUCd'}]
Outputs: [{'address': 'destination_address', 'value': 9500}]
```

GitHub link for codes :https://github.com/xxsarikapatel/Blockchain-BTC-CW/blob/main/Blockchain%20coursework%202.ipynb

d) What are centralised, decentralised, and distributed? Include a critique of their pros and cons.

Category	Centralized Systems	Decentralized Systems	Distributed Systems
Definition	A centralized system is one where a single point controls the entire network. This central point handles all major processes and decisions, and all subordinate units receive and implement instructions from this central authority.	In a decentralized system, control is distributed among various nodes or levels within the organization, rather than being concentrated in a single point. Each unit operates semi-autonomously but within the framework set by the central authority.	A distributed system involves a network of interconnected nodes that work together to perform a task. No single node has complete control over the entire network; instead, each node participates equally in data processing or decision-making.

Pros	Simplicity of Control: Easier to manage due to single decision-making authority.	Reduction in Load: Decreases the burden on central authority, enhancing local issue handling.	Fault Tolerance: Failure of one node doesn't necessarily impact the entire system.
	Efficiency in Decision-Making: Quick decision-making without needing multiple stakeholders' consultation.	Increased Flexibility: Units adapt based on local conditions and feedback.	Scalability: Easily scales by adding more nodes.
	Consistency: Standardizes policies and operations across the network.	Enhanced Motivation and Innovation: Autonomy fosters higher local motivation and innovation.	Resource Sharing: Optimizes the use of assets across the network.
Cons	Single Point of Failure: System can collapse if the central authority fails.	Potential for Inconsistency: Units may apply policies differently, leading to inconsistencies.	Complexity in Management: Lack of central control complicates system management.
	Scalability Issues: May become overwhelmed as the organization grows.	Complex Coordination: More challenging and costly communication and coordination.	Security Concerns: Increased risk of attacks due to multiple nodes.
	Lack of Autonomy: Lower levels have little to no decision-making authority, causing dissatisfaction and inefficiency.	Risk of Fragmentation: Units may become too independent, leading to fragmentation.	Latency Issues: Performance can be affected by the physical distance between nodes and interconnection quality.

Q2. ETHEREUM

a) Critically describe the difference between PoW and PoS in your own words.

Proof of Work (PoW)

Proof of Work (PoW) is a consensus mechanism originally used by Bitcoin and later adopted by Ethereum (before its transition to PoS). It requires network participants (miners) to solve complex mathematical problems to validate transactions and create new blocks.

Miners compete to solve a cryptographic puzzle, and the first miner to solve the puzzle gets the right to add the next block to the blockchain. The solution to the puzzle requires significant computational power and energy *(crypto.com, 2024).*

PoW provides robust security, as the significant computational power and electricity required to alter information on the blockchain deter potential attackers. Initially, PoW promotes decentralization, allowing anyone with the necessary hardware to participate in mining. However, it has notable disadvantages, including massive energy consumption, which poses significant environmental concerns. PoW networks also face scalability issues, leading to slower transaction times and higher fees as the network grows. Additionally, there is a risk of centralization over time, as wealthier organizations can afford more powerful mining setups, which undermines the decentralized ethos of blockchain.

Proof of Stake (PoS)

Proof of Stake (PoS) is a newer consensus mechanism used by many blockchains, including Ethereum (post its Ethereum 2.0 update). Instead of solving cryptographic puzzles, validators are selected to create new blocks based on the number of coins they hold and are willing to "stake" as collateral.

In PoS, validators are chosen to add new blocks to the chain based on their stake in the network. The larger their stake, the higher their chances of being selected. This selection is usually randomized but weighted by the amount of the stake *(learn crypto, 2024).*

PoS is significantly more energy-efficient than PoW, as it eliminates the need for extensive computational tasks. It enhances scalability by processing transactions faster and at a lower cost. Additionally, PoS lowers the barrier to entry, enabling more users to participate in the validation process without requiring expensive hardware. However, PoS also presents some security risks, being potentially more susceptible to specific attacks such as the "nothing at stake" problem, where validators might maliciously support multiple blockchain histories. Furthermore, PoS can lead to wealth concentration, as those with more coins have a higher likelihood of being chosen as validators, potentially giving the wealthy greater control over the blockchain.

Critical Comparison

PoS is more efficient than PoW, requiring lower energy and hardware costs. While PoW relies on computational difficulty for security, PoS uses economic incentives and penalties. PoS is more accessible, needing less hardware and energy investment, and is more eco-friendly due to its lower energy consumption. Additionally, PoS promotes broader participation, although it may lead to wealth concentration, whereas PoW tends to centralize mining in regions with low electricity costs.

b) Outline the pros & cons of swapping in DeFi and compare its performance in DeFi and in TradFi.

Swapping in DeFi - Swapping in DeFi refers to the exchange of one cryptocurrency for another using smart contracts on a blockchain, without the need for a central authority or intermediary.

Pros:

- **Autonomy:** Users have complete control over their transactions, eliminating the need for intermediaries like banks or brokers.
- **Accessibility:** DeFi platforms are generally accessible to anyone with an internet connection, regardless of geographic location or banking infrastructure.
- **Transparency:** All transactions are recorded on a public blockchain, providing clear visibility into market activities and asset flows.

- **Innovation:** DeFi swapping protocols often introduce novel features like liquidity pools, yield farming, and automated market makers (AMMs), which can offer enhanced yield opportunities.

Cons:

- **Complexity:** DeFi platforms and the concept of swapping can be complex and difficult for new users to understand.
- **Security Risks:** Smart contract vulnerabilities and the lack of regulation can expose users to higher risks of hacks and fraud.
- **Impermanent Loss:** Liquidity providers in AMMs can suffer from impermanent loss if the price of the tokens they provide diverges in a certain way.
- **Market Volatility:** High volatility can lead to significant price slippage during trades, impacting transaction costs.

Performance Comparison: DeFi vs. TradFi

Performance Aspect	**DeFi Swapping**	**TradFi Swapping**
Regulation and Security	Minimally regulated; security depends on smart contract integrity, posing higher risk of hacks and fraud.	Heavily regulated; established security protocols and insurance mechanisms provide robust protection.
Efficiency and Speed	Transactions are executed almost instantaneously, often within seconds, enhancing trading efficiency.	Transactions may take longer due to compliance checks and operational processes but are highly reliable.
Costs	Generally lower fees as intermediaries are removed; cost efficiency is a significant advantage.	Higher fees due to the involvement of various intermediaries and regulatory compliance costs.

Performance Aspect	DeFi Swapping	TradFi Swapping
Market Access	Open to anyone with internet access, offering global accessibility and minimal entry barriers.	Often restricted by regulatory requirements, geographic limitations, and financial thresholds.
Stability	Subject to high volatility and the potential for significant price slippage, affecting market stability.	More stable markets with established mechanisms to manage and mitigate risks associated with trading.
User Experience	Can be complex and daunting for new users; requires a higher degree of technical knowledge.	Typically offers a more user-friendly interface with customer support, accessible to a broader audience.
Innovation	Highly innovative, introducing features like AMMs, liquidity pools, and yield farming which can enhance yield.	Less flexible to innovation due to regulatory and structural constraints but gradually adopting new technologies.
Security Measures	Dependent on user and community vigilance; lacks standardized protective measures.	Comprehensive regulatory safeguards and standardized security measures in place.

Swapping in DeFi revolutionizes how users interact with financial services by offering autonomy, reduced costs, and innovative financial products. However, it also introduces complexities and security challenges that are not as prevalent in the more regulated, secure, and stable environment of TradFi. Each system has its advantages and trade-offs, with DeFi excelling in innovation and accessibility, and TradFi leading in security and user experience.

c) Critically explain the layer 2 solutions in your own words and provide an example of the layer 2.

Layer 2 solutions are technologies developed to improve the scalability and efficiency of blockchain networks by handling transactions off the main chain (Layer 1). This is particularly important for blockchains like Ethereum, where network congestion can lead to high transaction fees and slow processing times. Layer 2 solutions aim to alleviate these issues without compromising the security and decentralization aspects of the main blockchain *(chainlink, 2024)*.

How They Work: Layer 2 solutions process data and transactions off the main blockchain but still leverage its underlying security protocols. By doing so, they manage to increase transaction throughput and reduce delays *(Ledger , 2024)*. These solutions are critical in supporting the blockchain ecosystem's scalability, making it viable for widespread adoption and handling large volumes of transactions typical in applications like decentralized finance (DeFi) and non-fungible tokens (NFTs).

Types of Layers 2 Solutions:

1. **State Channels:** Allow participants to conduct numerous transactions off-chain and then settle the final state on-chain. This is ideal for situations where participants need to perform multiple rapid transactions, such as in gaming or micro-payments.
2. **Plasma:** Involves creating smaller, child blockchains that are anchored to the main blockchain. These child chains can process transactions independently and periodically commit their state back to the main chain.
3. **Rollups:** Transactions are processed in batches off-chain, and only net results are posted to the main blockchain. Rollups can be further classified into:
    - **Optimistic Rollups:** Assume transactions are valid by default and only run computations if a transaction is challenged.
    - **Zero-Knowledge Rollups (ZK-Rollups):** Use cryptographic proofs to validate all transactions off-chain before committing them to the main chain.

Layer 2 solutions offer several advantages, including significantly increasing transaction throughput and reducing costs by lowering transaction fees through reduced burden on the main chain. They also provide faster transaction processing times while leveraging the underlying security of the main blockchain. However, these solutions come with disadvantages such as complexity in implementation and user interaction, potential fragmentation within the ecosystem if not standardized, and security concerns, as some Layer 2 solutions might have vulnerabilities if their security mechanisms are not robust enough.

Example of a Layer 2 Solution: Lightning Network the Lightning Network is a prominent example of a Layer 2 solution implemented on the Bitcoin blockchain. It utilizes state channels to enable instant, high-volume microtransactions between participating nodes. Users open a payment channel by committing a transaction to the main blockchain, conduct an unlimited number of transactions off-chain, and then close the channel by settling the final state on-chain. This approach significantly reduces the time and fees associated with each transaction, exemplifying how Layer 2 can address scalability and efficiency challenges in blockchain technology.

Layer 2 solutions are pivotal in advancing blockchain technology, offering practical pathways to overcome inherent limitations of Layer 1 blockchains concerning scalability and transaction speed. By processing transactions off-chain and settling final states on-chain, these solutions ensure that blockchains can manage higher transaction volumes while retaining core principles of security and decentralization. As blockchain technology continues to evolve, the development and refinement of Layer 2 solutions will play a critical role in its mass adoption and application across various industries.

d) Mint and describe your NFT collection (at least 3 NFT items in the collection) on the Sepolia/Mumbai Testnet. (You can use the ERC721 smart contract in the Github. Detailed creation process and the properties of your NFTs are needed. The website link of opensea testnets for this collection and smart contract address are required as well.)

Detailed Creation Process and Properties of our NFTs

We have created a collection of three NFTs representing the exquisite Laxmi Vilas Palace, featuring different aspects of its architecture and art. These NFTs are minted on the Sepolia

Test net and are available on Open Sea Test net. The detail process of NFT creation are as follows:

STEP:1. PREPARATION AND UPLOADING ASSETS

PNG Images Uploaded by utilising Pinata for decentralized storage and CID for Images (QmWEY7Qs65XGp3S4B4eo7DFPk3j3TVWa43HXE1Fh2RboV4)

NFT 1: Historic Window of Laxmi Vilas Palace.

Properties:(Location: Laxmi Vilas Palace, Era: 19th Century, Material: Stone)

Benefits for NFT holders:

- Access to in-depth virtual tours focusing on the intricate design and historical significance of the palace's windows.
- Downloadable high-resolution images and detailed narratives about the palace's architecture.
- Special invitations to private viewings of the palace's architectural highlights.
- Discounts in ticket prices for visiting Laxmi Vilas Palace and associated heritage sites.
- Enjoy breathtaking views of lush greenery through the heritage window, providing a refreshing and peaceful experience.

NFT 2: Front View of Laxmi Vilas Palace.

Properties:(**Location**: Vadodara, India, Era: 19th Century, Architectural Style: Indo-Saracenic)

Benefits:

- Invitations to exclusive online events discussing the architectural significance and history of Laxmi Vilas Palace.
- Priority access to future digital collectibles and artwork related to the palace.
- Invitations to annual events and gatherings celebrating the heritage of Laxmi Vilas Palace.
- Membership in a heritage preservation club, including newsletters and special updates.
- Exclusive access to purchase or receive limited edition items featuring the front view of the palace.

NFT 3: Marble Peacock Monument at Laxmi Vilas Palace.

Properties:(**Material**: Marble, **Location**: Laxmi Vilas Palace, **Symbolism**: Elegance)

Benefits:

- Access to high-resolution digital files and stories about the marble peacock monument and other palace sculptures.
- Early access to future NFTs showcasing additional artworks and sculptures from Laxmi Vilas Palace.
- Invitations to exclusive art workshops or lectures at Laxmi Vilas Palace.
- Opportunities for personalized tours focusing on the artistic elements of the palace.
- Access to limited edition replicas or miniature models of the marble peacock monument.

STEP: 2. METADATA (JSON FILE) CREATION

Metadata Files(json) Generated using python script, each metadata file links to an image on IPFS and contains attributes describing the artwork.

IPFS CID for Metadata **(QmPhBuMw1VeL1v3hSeitPMVkfx23V3jQFASYpLxSbSBRj4)**

```
In [1]:
import json
import os

# Base CID for images
image_cid = 'QmWEY7QS65XGp3S4B4eo7DFPk3j3TVWa43HXE1Fh2RboV4'

# Metadata template
metadata_template = [
    {
        "name": "Historic Window of Laxmi Vilas Palace",
        "description": "A beautifully designed historic window of Laxmi Vilas Palace, showcasing exquisite architectural deta
        "image": f"ipfs://{image_cid}/1.png",
        "compiler": "Hugo",
        "attributes": [
            {"trait_type": "Location", "value": "Laxmi Vilas Palace"},
            {"trait_type": "Era", "value": "19th Century"},
            {"trait_type": "Material", "value": "Stone"}
        ]
    },
    {
        "name": "Front View of Laxmi Vilas Palace",
        "description": "A majestic front view of Laxmi Vilas Palace, capturing the grandeur and elegance of this historic bui
        "image": f"ipfs://{image_cid}/2.png",
        "compiler": "Hugo",
        "attributes": [
            {"trait_type": "Location", "value": "Vadodara, India"},
            {"trait_type": "Era", "value": "19th Century"},
            {"trait_type": "Architectural Style", "value": "Indo-Saracenic"}
        ]
    },
    {
        "name": "Marble Peacock Monument",
        "description": "An artistic marble sculpture of two peacocks facing each other, situated in front of Laxmi Vilas Pala
        "image": f"ipfs://{image_cid}/3.png",
        "compiler": "Hugo",
        "attributes": [
            {"trait_type": "Material", "value": "Marble"},
            {"trait_type": "Location", "value": "Laxmi Vilas Palace"},
            {"trait_type": "Symbolism", "value": "Elegance"}
        ]
    }
]

# Ensure directory exists
os.makedirs('LVP_JSON', exist_ok=True)

# Save metadata files
for i, data in enumerate(metadata_template):
    filename = f'LVP_JSON/metadata{i+1}.json'
    with open(filename, 'w') as f:
        json.dump(data, f, indent=4)
    print(f'Created {filename} with image link: {data["image"]}')
```

STEP: 3. SMART CONTRACT DEPLOYMENT

For the deployment of the Laxmi Vilas NFT smart contract, we utilized the ERC721 standard with batch minting capabilities through the Remix IDE. The process began by creating a file named LaxmiVilasNFT.sol under the contract directory in Remix. We then wrote and compiled the smart contract, which incorporates functions for both individual and batch minting of NFTs.

The smart contract, once compiled without errors, was deployed on the Sepolia Testnet using MetaMask. This ensured that our NFTs were created on a decentralized network, making them accessible and verifiable on the blockchain. The batch mint function was particularly useful as it allowed us to mint all three JSON metadata files in one go, streamlining the minting process.

After connecting our MetaMask wallet, we specified the recipient's address and the URIs of the metadata files:

"ipfs://QmPhBuMw1VeL1v3hSeitPMVkfx23V3jQFASYpLxSbSBRj4/metadata1.json"

"ipfs://QmPhBuMw1VeL1v3hSeitPMVkfx23V3jQFASYpLxSbSBRj4/metadata2.json"

"ipfs://QmPhBuMw1VeL1v3hSeitPMVkfx23V3jQFASYpLxSbSBRj4/metadata3.json"

This deployment and minting process resulted in the successful creation of the NFTs, each linked to their respective metadata. The smart contract details and the batch minting functionality ensured that the NFTs were securely minted and stored on the blockchain.

below codes used for smart contract generation and batch minting of NFT

SMART CONTRACT ADDRESS: [0X3978E667794182A16B9EF27D992A3C3E3790C1DB]

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

import "@openzeppelin/contracts/token/ERC721/extensions/ERC721URIStorage.sol";
import "@openzeppelin/contracts/utils/Counters.sol";

contract LaxmiVilasNFT is ERC721URIStorage {
    using Counters for Counters.Counter;
    Counters.Counter private _tokenIds;

    constructor() ERC721("LaxmiVilasNFT", "LVNFT") {}

    function mintNFT(address recipient, string memory tokenURI) public returns (uint256) {
        _tokenIds.increment();
        uint256 newItemId = _tokenIds.current();
        _mint(recipient, newItemId);
        _setTokenURI(newItemId, tokenURI);
        return newItemId;
    }

    function batchMint(address recipient, string[] memory tokenURIs) public {
        for (uint i = 0; i < tokenURIs.length; i++) {
            mintNFT(recipient, tokenURIs[i]);
        }
    }
}
```

STEP:4. VERIFICATION AND VIEWING

Upon completing the minting process, we verified the NFTs' existence and details on Etherscan, ensuring that each token was correctly minted and linked to its respective metadata. We then used MetaMask to import these NFTs by providing the smart contract address and token IDs, ensuring that all details matched accurately. Finally, we verified the NFTs on the OpenSea Testnet and listed them for sale, providing a public link to view the collection.

WEBSITE LINK OF OPENSEA TEST NET: **https://testnets.opensea.io/account**

Link of each NFT
(https://testnets.opensea.io/assets/sepolia/0x3978e667794182a16b9ef27d992a3c3e3790c1db/5/)

(https://testnets.opensea.io/assets/sepolia/0x3978e667794182a16b9ef27d992a3c3e3790c1db/6/)

(https://testnets.opensea.io/assets/sepolia/0x3978e667794182a16b9ef27d992a3c3e3790c1db/7/)

REFERENCES

chainlink. (2024, 05 12). *what is layer 2*. Retrieved from chainlink: https://chain.link/education-hub/what-is-layer-2

crypto.com. (2024, 05 12). *Proof of Work vs Proof of Stake*. Retrieved from crypto.com: https://crypto.com/university/proof-of-stake-vs-proof-of-work

learn crypto. (2024, may 12). *Proof-of-Work vs Proof-of-Stake: Is PoW better than PoS?* Retrieved from learn crypto: https://learncrypto.com/feed/articles/proof-of-work-vs-proof-of-stake-is-pow-better-than-pos

Ledger . (2024, 05 12). *What Are Ethereum Layer 2 Blockchains and How Do They Work?* Retrieved from Ledger academy: https://www.ledger.com/academy/topics/blockchain/what-are-ethereum-layer-2-blockchains-and-how-do-they-work