Lesson 17: Part 1 - Scaling Blockchain

Scaling overview

- The scalability trilemma
- Layer 1 solutions
- Layer 2 solutions
- Sharding
- Bridges
    - Trusted
    - Trustless
- Cross chain
- Consensus mechanisms for scaling

References

https://www.gemini.com/cryptopedia/blockchain-trilemma-decentralization-scalability-definition

https://vitalik.ca/general/2021/04/07/sharding.html

https://vitalik.ca/general/2021/05/23/scaling.html

https://ethereum.org/en/bridges/

Polygon (Matic)

- Polygon project
- Plasma bridge
- PoS bridge
- Interacting with matic using Ethers.js library
- Matic.js library
- Coding bridges

References

https://docs.matic.today/docs/develop/getting-started/

https://docs.polygon.technology/docs/develop/ethereum-polygon/plasma/getting-started/

https://docs.matic.today/docs/develop/ethereum-matic/pos/getting-started/

https://maticnetwork.github.io/matic.js/docs/get-started/

Lesson 17: Part 2 - Advanced Solidity and Assembly

Encoding and string manipulation

- What is ABI encoding

- Using ABI encoding to manipulate bytes
- Packing and unpacking
- Using ABI encoding to manipulate strings
- Using libraries
    - Solidity Utilities by *willitscale*
    - String utils by *Arachnid*

References

https://docs.soliditylang.org/en/develop/abi-spec.html

https://betterprogramming.pub/solidity-playing-with-strings-aca62d118ae5

Positive and negative overflow and underflow

- The age of SafeMath
- Solidity 0.8.0
- Using overflow without reverting
    - Valid use cases for overflow
    - Command blocks inside solidity
    - Using “Unchecked” block
- Misconceptions about underflow

References

https://en.wikipedia.org/wiki/Integer_overflow

https://en.wikipedia.org/wiki/Arithmetic_underflow

https://docs.openzeppelin.com/contracts/4.x/api/utils#SafeMath

https://docs.soliditylang.org/en/latest/080-breaking-changes.html

https://docs.soliditylang.org/en/latest/control-structures.html#checked-or-unchecked-arithmetic

Assembly

- Syntax overview for Yul
- Inline assembly
- Use cases
- Solidity conventions for using inline assembly

References

https://docs.soliditylang.org/en/latest/assembly.html

https://docs.soliditylang.org/en/latest/yul.html#yul

https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/utils/Address.sol

https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/utils/Create2.sol

ECDSA example

- Overview about ECDSA signature
- The danger of signing messages
- EIP191 and avoiding unintentionally signing a RLP
- Verifying signatures
- ECDSA OpenZeppelin Utility
    - Usage of string manipulation in function *toEthSignedMessageHash*
    - Usage for inline assembly in function *tryRecover*

References

https://eips.ethereum.org/EIPS/eip-191

https://eth.wiki/en/fundamentals/rlp

https://docs.ethers.io/v5/api/signer/#Signer-signMessage

https://docs.openzeppelin.com/contracts/4.x/utilities#checking_signatures_on_chain

https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/utils/cryptography/ECDSA.sol

Homework

- Read the references
- Test more Polygon features
- (Optional) Test other scaling solutions you find interesting
- Test other functions from the libraries presented
- Test other cases of overflow
- (Optional) Recreate ERC20 in assembly as this example