For clarifications Contact Ph: 09353205447, Email: bbaktech@gmail.com

Basics – Ethereum- Smart Contracts:

Ethereum

Ethereum is a decentralized blockchain platform that establishes a peer-to-peer network that securely executes and verifies application code, called smart contracts. Smart contracts allow participants to transact with each other without a trusted central authority.

Transaction records are immutable, verifiable, and securely distributed across the network, giving participants full ownership and visibility into transaction data. Transactions are sent from and received by user-created Ethereum accounts.

A sender must sign transactions and spend Ether (Ethereum's native cryptocurrency), as a cost of processing transactions on the network.

Ethereum is moved from Proof of Work (PoW) to Proof of Stake (PoS).

Benefits of building on Ethereum

Today, we gain access to 'free' internet services by giving up control of our personal data. Ethereum services are open by default – you just need a wallet. These are free and easy to set up, controlled by you, and work without any personal info.

Ethereum offers an extremely flexible platform on which to build decentralized applications using the native Solidity scripting language and Ethereum Virtual Machine.

Decentralized application developers who deploy smart contracts on Ethereum benefit from the rich ecosystem of developer tooling and established best practices that have come with the maturity of the protocol.

This maturity also extends into the quality of user-experience for the average user of Ethereum applications, with wallets like **MetaMask, Argent, Rainbow** and more offering simple interfaces through which to interact with the Ethereum blockchain and smart contracts deployed there.

Ethereum’s large user base encourages developers to deploy their applications on the network, which further reinforces Ethereum as the primary home for decentralized applications like DeFi and NFTs. In the future, the backwards-compatible Ethereum 2.0 protocol, currently under

development, will provide a more scalable network on which to build decentralized applications that require higher transaction throughput.

Use cases:

Decentralized Finance (DeFi)

DeFi is a network of financial applications built on top of blockchain networks. It is different from existing financial networks because it is open and programmable, operates without a central authority, and enables developers to offer new models for payments, investing, lending, and trading. By using smart contracts and distributed systems, customers can easily build secure decentralized financial applications. For example, DeFi companies are already offering products that enable peer-to-peer lending and borrowing, earning interest on cryptocurrency holdings, trading via decentralized exchanges, and much more. Some popular DeFi platforms include Compound, Aave, UniSwap, and MakerDAO.

Non-Fungible Tokens (NFTs)

NFTs are unique and indivisible digital tokens that are useful for proving the provenance of rare assets, both digital and tangible. For example, NFTs can be used by an artist to tokenize their work and ensure that their work is unique and belongs to them. The ownership information is recorded and maintained on the blockchain network. NFTs are also gaining popularity in the gaming industry because they allow interoperability between gaming platforms. For instance, the first NFT project on Ethereum was CryptoKitties, which enabled customers to collect digital cat collectibles backed using NFTs. Gods Unchained is a card game that gives players full ownership of their in-game items using NFTs. NFTs are gaining popularity as more companies look to tokenize assets and provide users with tamper-proof lineage information about their assets.

The smart contract is written on **remix compiler**, a user interface is created for the purpose of interaction with the ethereum smart contract and is tested on a **Roptsten-network** for the purpose of contract deployment and confirmation. A **MetaMask(wallet)** is used as the bridge that connects the ethereum blockchain to the Roptsten network.

Ethereum set up:

Externally Owned Account: the externally user account is created by the in which the smart contract is written based on it to make transaction. Under the externally owned account we have the Account Address, Account public and private key then the Ether.

Account Address: once a block is mined by the miners this particular block gets an addressed which is in binary form, with this it can't be traced to anyone with a specific name. This makes the owner and buyer of any land, under the transaction anonymous.

Account private and public key: this key is a product of digital encryption of signature. The purpose of the private key which is only owned by the owner of the transaction is to sign a transaction once the smart contract conditions are met. For the public key, this is owned by every participant in the node, this key is used to verify that the said transaction still exists and the acclaimed owner of the transaction is actually the owner.

Ether Account: this is the account used in the Ethereum blockchain where the commodity/money used to make transaction in the blockchain are kept. Every transaction required a specific amount of gas before the transaction can go through. Once a buyer indicate interest in a transaction, he needs to have an ether account, then purchase some ether after which he can buy the gas needed for the transaction.

Transaction: the transaction process accepts the data of the particular block which is about to be transacted is sent to it and process it has a value.

Smart Contract: this the executable code, which verifies the value of the transaction and check if it meets all the rules embedded in the contract before it can proceed to send message through individual id addresses to participants in the blockchain network.

Platform to develop the smart contract application.

1.**Remix ide**: the compiler used

2. **Solidity language**: the smart contract as it is indicated in figure 2 is written in solidity high level programming language.

3. **Environment**: the environment where an ethereum smart contract can be executed is on an ethereum virtual machine environment.

For the purposed of this research, an injected web3 environment is used to execute the smart contract code.

4. **Account**: the account as it is indicated in figure 2 is the ethereum wallet account(**MetaMask)** where the ether which is the money spend in an ethereum blockchain for the purpose of land transaction is saved.

5. **Deployed contracts:** this is the contract that has been deployed after the smart contract executable code has been compiled,

Using **rospten test network** where MetaMask(**wallet**) is the bridge between the test network and the land transaction smart contract. This is done to indicate that every participant of the blockchain has verified the contract deployed by the landowner and can be transacted to the land buyer. (What is double spending)

Rospten network setup:

1. **Transaction hash:** is a unique id that is generated for every transaction that is performed in the blockchain network. For every transaction confirmed, there is a different id and with this it has avoided the problem of double spending in the land title management and transaction. The status shows it was successful, this indicate that the transaction is genuine and the nodes have used their public key to verify the genuineness of the transaction.

2.**Block:** there is a unique block number for every node(participant) in the ethereum blockchain, the block number indicates a block was successfully mined by the miner also it indicates that the transaction was confirmed by nodes with their public key, in the blockchain, this in a way has created a transparent system of transaction where it is decentralized and can't be tampered with by anyone. This has defeated the use of third party.

3. **From and to:** the from is the owner of the transactionand to is the buyer of the land. A unique id is also given to the owner and buyer for the purpose of anonymity when do ing a transaction. This as also fulfilled another purpose for which ethereum blockchain was propsed to manage land title and transactions

4. **Transaction fee**: this is the ether consumed i.e the amount of monet spent to perform the transaction.

5. **Nonce**: the nonce for this blockchain application is the arbitrary number that is used just once for the purpose of crytopraphic communication the blockchain network. It a random number that is used for the authentication

protocol of this transaction in the block so as to make attack or replay of the same transaction impossible.

6. **Time stamp**: The server for time stamp works by taking a hash of a block of item to be time stamped and widely publishing the hash. Time stamp prove the data must have existed at a time.

7. **Gas limit and Gas price:** The Ethereum blockchain requires a certain amount gas, depending on the size and type of each transaction. Gas measures the amount of work miners need to do in order to include transactions in a block.