Doğukan Akar Bengisu Özaydın

CMPE 483: Blockchain Programming

HW1

ERC721 token based supply chain traceability system on Ethereum blockchain

We have written two smart contracts for this project, along with the Solidity tests.

The contracts are uploaded to Binance Smart Chain Testnet and can be seen in action at our Web App.

The ABI code of these contracts along with the rest of the code for the Web App is available at our GitHub repo.

Verification.sol

deployment cost: `0.1571296129612961 gas`

This is the State Legal Entity Verification Contract.

An address is able to be verified in the state.

The verification status of an address is able to be checked.

Keeps a mapping of entity addresses to verification statuses.

```
mapping(address => bool) legalEntityVerifications;
```

Has 1 external payable function:
- `verify`
  - cost: `0.0385634563456346 gas`
  - Mark an entity as verified.
  - Revert if the entity is already verified.

Has 1 view function:
- `queryVerified`
  - cost: `0 gas`
  - Return verification status of the entity with the inputted address.

We have written unit tests for checking that an entity is initially not verified. Then, when the verify() method is called, the entity becomes verified.

Doğukan Akar Bengisu Özaydın

ProductProvenance.sol

deployment cost: `3.233838883888389 gas`

Implemented with ERC721 library standard.

The manufacturer is able to mint a token that represents a single product by providing hashed serial no and factory zip code information associated with the product.

The owner of a token is able to transfer the ownership of the token to another address.

The new owner of a token is able to acknowledge the transfer of ownership of the token.

It is possible to trace the product to its original manufacturer.

Defined structs for Product and Owner, well documented in the code.

Keeps three mappings for all operations needed:
- A mapping of token IDs -> Owner struct instances
- A mapping of product serial nos -> token IDs
- A mapping of token IDs -> Product struct instances

Token IDs start with 0 and are incremented by 1 each time a mint/manufacture operation is executed.

The minter of a token is that token's first owner.

Has 3 external payable functions.
- `transferOwnership(address to, uint256 tokenId)`
    - cost: `0.0912808280828083 gas`
    - Revert if the token with the inputted ID does not exist.
- `acknowledgeOwnership(uint256 tokenId)`
    - cost: `0.030011701170117 gas`
    - Revert if the token with the inputted ID does not exist.
    - Revert if the caller is not the owner of the token with the inputted ID.
    - Revert if the ownership has already been approved.
- `manufacture(bytes32 serialNoHash, bytes32 factoryZipCode)`
    - cost: `0.1664158415841584 gas`
    - Mint a token that represents a product with the inputted serial no hash, manufactured at the factory with the inputted zip code.

Has 11 view functions:

- `getLastTokenId`
  - cost: `0 gas`
  - Return the token ID of the last minted token.
- `getProductByTokenId`
  - cost: `0 gas`
  - Return the serial no hash and factory zip code of the token with the inputted ID.
- `getProductBySerialNo`
  - cost: `0 gas`
  - Return the serial no hash and factory zip code of the token with the inputted serial no hash.
- `getOwnerBySerialNoAndOwnerIndex`
  - cost: `0 gas`
  - Return the address of the owner at the inputted index for the token with the inputted serial no hash.
- `getOwnerByTokenIdAndOwnerIndex`
  - cost: `0 gas`
  - Return the address of the owner at the inputted index for the token with the inputted token ID.
- `getLastOwnerBySerialNo`
  - cost: `0 gas`
  - Return the address of the current/last owner for the token with the inputted serial no hash.
- `getLastOwnerByTokenId`
  - cost: `0 gas`
  - Return the address of the current/last owner for the token with the inputted token ID.
- `getFirstOwnerBySerialNo`
  - cost: `0 gas`
  - Return the address of the first owner for the token with the inputted serial no hash.
- `getFirstOwnerByTokenId`
  - cost: `0 gas`
  - Return the address of the first owner for the token with the inputted token ID.
- `getOwnerCountBySerialNo`
  - cost: `0 gas`
  - Return the number of past and current owners for the token with the inputted serial no hash.
- `getOwnerCountByTokenId`
  - cost: `0 gas`
  - Return the number of past and current owners for the token with the inputted token ID.