



OST

Eastern Switzerland
University of Applied Sciences

Blockchain (BlCh)

Fungible Tokens

Thomas Bocek

06.10.2024

Introduction

- **Definition:** “A fungible token is a form of a digital asset that is used as a store of value, a unit of account or a medium of exchange in blockchain transactions. These assets are called fungible as they can be split or interchanged with the same category of tokens. Fungible tokens operate through smart contracts and peer-to-peer systems. Transactions are secured by cryptography and remain traceable and public.

There are three main types of fungible tokens: **payment**, **utility** and **security tokens**. While payment tokens usually refer to cryptocurrencies such as Bitcoin (BTC) and Ethereum (ETH), there are currently more than 5,000 other cryptocurrencies in circulation.”

- Classification can be better :(
- Better: 3 characteristics
 - Interchangeability
 - With other tokens of the **same** type
 - Divisibility
 - Into smaller units
 - Standardization
 - High degree of standardization and uniformity
- Examples
 - Bitcoin, Ethers, USDC, USDT, OST Token

Cryptocurrencies

- Cryptocurrencies are **fungible** tokens
 - Digital currency not reliant on a central authority
 - **CBDC** is digital currency backed by central authority
 - “In contrast to cryptocurrencies, a central bank digital currency would be centrally controlled”
- Exchange cryptocurrencies
 - Decentralized exchanges (DEX) – same chain/cross chain, centralized exchanges (CEX)
- 6 conditions (by **Jan Lansky**)
 - 1) The system does not require a central authority, distributed achieve consensus on its state.
 - 2) The system keeps an overview of cryptocurrency units and their ownership.
 - 3) The system defines whether new cryptocurrency units can be created. If new cryptocurrency units can be created, the system defines the circumstances of their origin and how to determine the ownership of these new units.
 - 4) Ownership of cryptocurrency units can be proved exclusively cryptographically.
 - 5) The system allows transactions to be performed in which ownership of the cryptographic units is changed. A transaction statement can only be issued by an entity proving the current ownership of these units.
 - 6) If two different instructions for changing the ownership of the same cryptographic units are simultaneously entered, the system performs at most one of them.

Fungible Token Use-cases

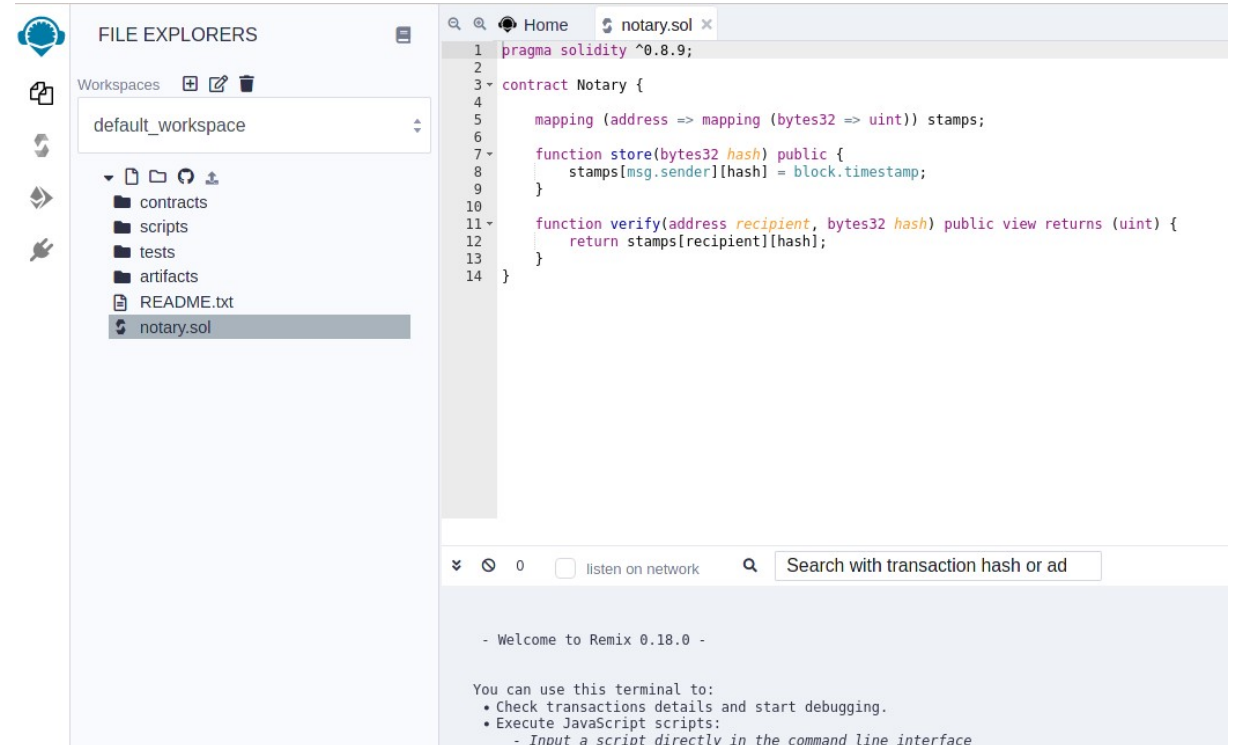
- Stablecoins
 - Pegged to **fiat money**, e.g., USDC, XCHF
- Gas for smart contracts
 - Use to execute smart contracts
- Altcoins
 - Non mainstream coins (aka sh*tcoins), definition is vague
- Speculation
- Swiss **DLT law**, since 01.08.2021
 - Licence for DLT trading facilities
 - Electronic registers
- **Finma** categorization (payment, utility and asset tokens):
 - Payment tokens are synonymous with cryptocurrencies and have no further functions or links to other development projects. Tokens may in some cases only develop the necessary functionality and become accepted as a means of payment over a period of time.
 - Utility tokens are tokens which are intended to provide digital access to an application or service.
 - Asset tokens represent assets such as participations in real physical underlyings, companies, or earnings streams, or an entitlement to dividends or interest payments. In terms of their economic function, the tokens are analogous to equities, bonds or derivatives.
- Native vs. non-native tokens
 - Ethers are native, USDT on Ethereum is smart contract driven → non-native token

FT / Tokens / ERC-20

- **ERC-20**
 - Token contract keeps track of fungible tokens
 - Can be used as vault for NFTs
- **Optional (highly recommended)**
 - `function name() public view returns (string)`
 - `function symbol() public view returns (string)`
 - `function decimals() public view returns (uint8)`
- **Events**
 - `event Transfer(address indexed _from, address indexed _to, uint256 _value)`
 - `event Approval(address indexed _owner, address indexed _spender, uint256 _value)`
- **Mandatory**
 - `function totalSupply() public view returns (uint256)`
 - `function balanceOf(address _owner) public view returns (uint256 balance)`
 - `function transfer(address _to, uint256 _value) public returns (bool success)`
 - `function transferFrom(address _from, address _to, uint256 _value) public returns (bool success)`
 - `function approve(address _spender, uint256 _value) public returns (bool success)`
 - `function allowance(address _owner, address _spender) public view returns (uint256 remaining)`

ERC-20 Implementation

- OpenZeppelin – many default contracts, very good source
 - Can be referenced
 - Let's implement a simple ERC20 and deploy on testnet
 - Remix IDE



The screenshot displays the Remix IDE interface. On the left, the 'FILE EXPLORERS' panel shows a workspace named 'default_workspace' containing folders for 'contracts', 'scripts', 'tests', and 'artifacts', along with files 'README.txt' and 'notary.sol'. The main editor area shows the code for 'notary.sol' with the following content:

```
1 pragma solidity ^0.8.9;
2
3 contract Notary {
4
5     mapping (address => mapping (bytes32 => uint)) stamps;
6
7     function store(bytes32 hash) public {
8         stamps[msg.sender][hash] = block.timestamp;
9     }
10
11     function verify(address recipient, bytes32 hash) public view returns (uint) {
12         return stamps[recipient][hash];
13     }
14 }
```

At the bottom, the terminal area shows the following text:

```
- Welcome to Remix 0.18.0 -
You can use this terminal to:
• Check transactions details and start debugging.
• Execute JavaScript scripts:
  - Input a script directly in the command line interface
```