ETHEREUM SMART CONTRACTS FOR DIGITAL SIGNATURES

Group 6:
Zdenek Bousa
Beata Stingelova

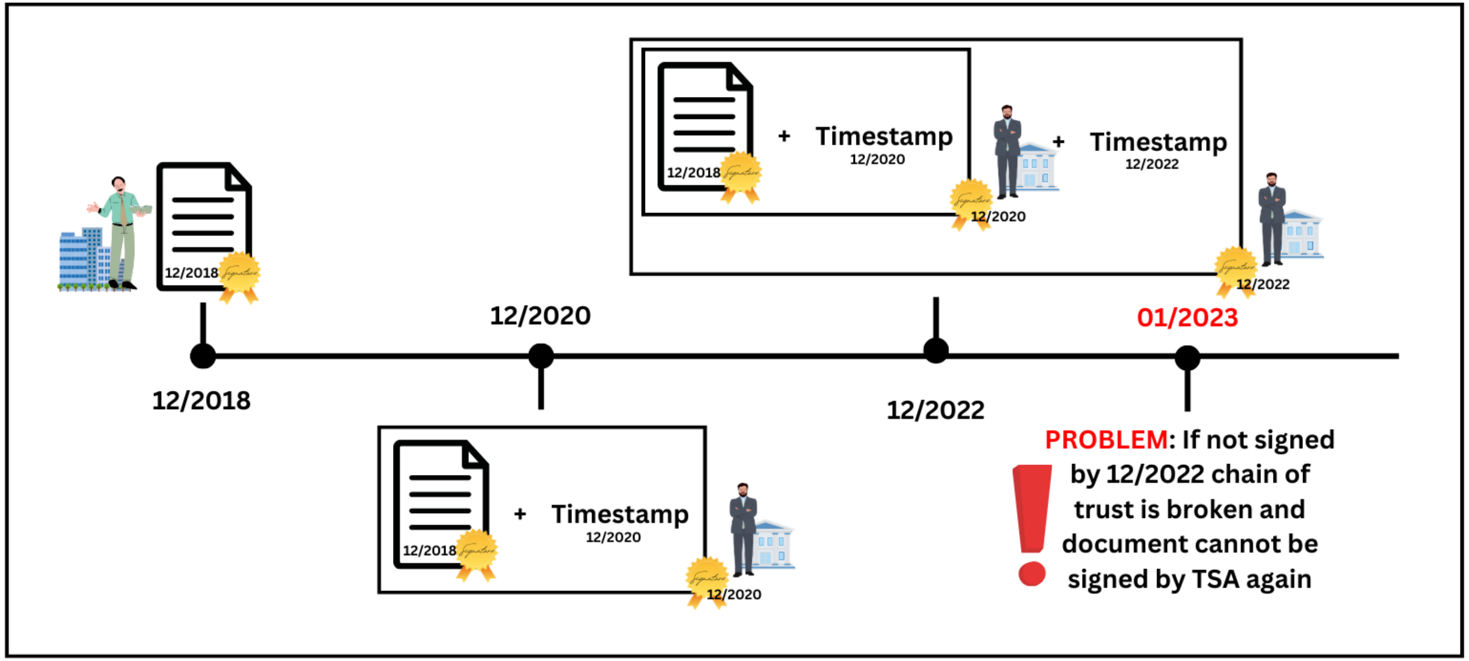
RESEARCH QUESTION

How can a decentralized document identity system on blockchain, supported by authoritative verification (digital signature) sources enhance **data privacy**, **security**, **integrity**, and **user control** in document management?

How can be system **designed** so the user remains in control of data?

How can a decentralized document identity system ensure data **integrity** and **trust** without supervision of **signee**?

CURRENT STATE



3 ISSUES IDENTIFIED

Timestamping

Need for regular timestamping before the validity of digital signature runs out.

Trusted Secure Authority

Who is our TSA? Can we trust them? Is it possible to get rid of them?

Chain of Trust

How to ensure that the digital signature was not antidated – signed by 01/23 with date of 12/22?

SOLUTIONS

1

Same author and signee

The author of the document uploads it in Blockchain and in the future the content of it can be verified

2

Unverified Chain

The issuing institution signs the document and user uploads it in Blockchain.

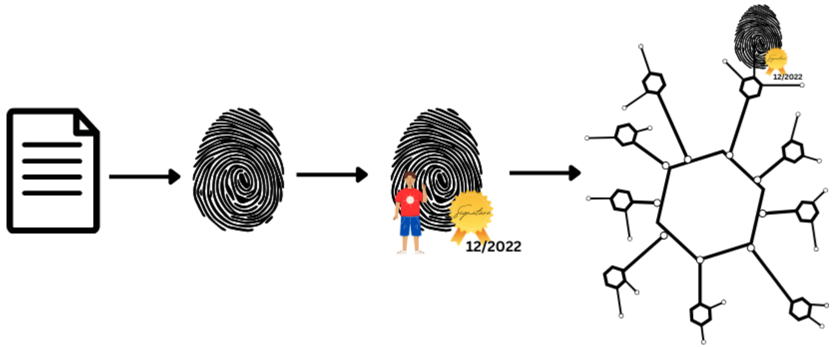
3

Verified Chain

The issuing institution signs the document and user uploads it in Blockchain, and a trusted 3rd party verifies it.

SOLUTION – SAME AUTHOR AND SIGNEE

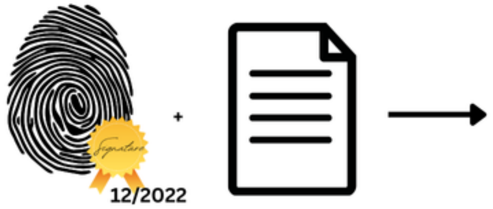
Storage



Verification

From Blockchain

From Cloud



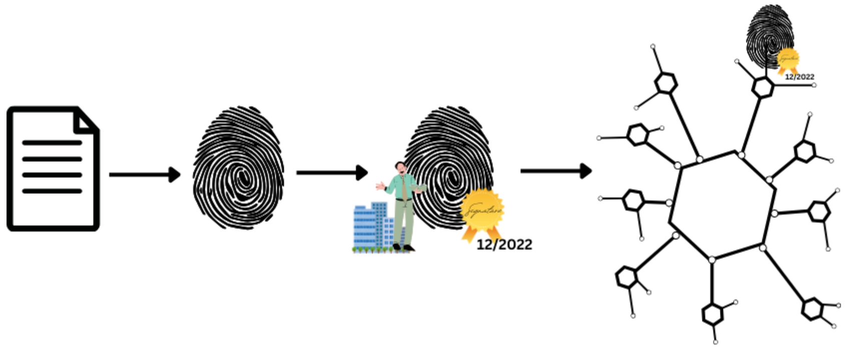
Is fingerprint the same?

Was signature valid at the timestamp time on blockchain?

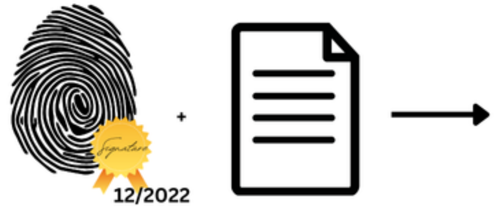
Verification whether the content of the document is unchanged

SOLUTION – UNVERIFIED CHAIN

Storage



Verification



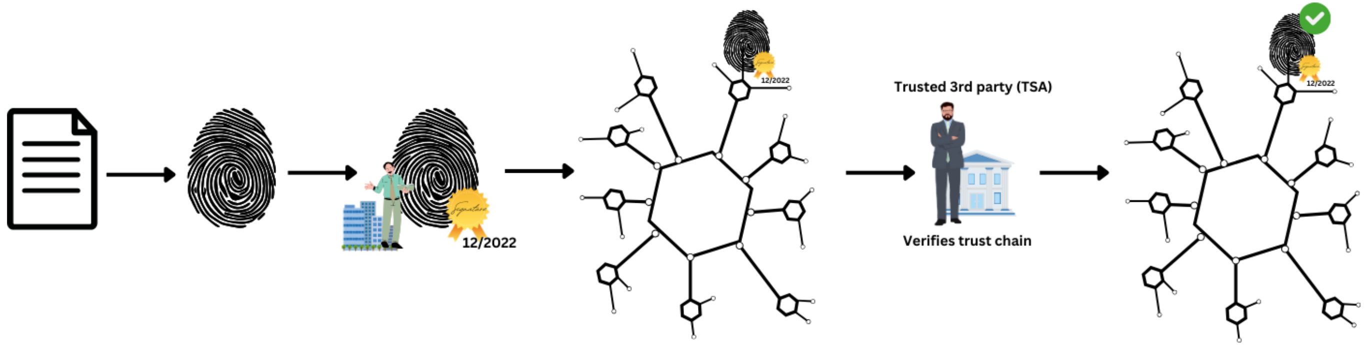
Is fingerprint the same?

Was signature valid at the timestamp time on blockchain?

Problem:
Signature forgery -> undetectable forgery after certain time

SOLUTION – VERIFIED CHAIN

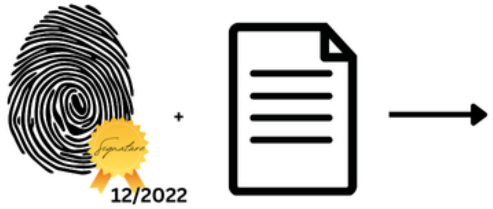
Storage



Verification

From Blockchain

From Cloud

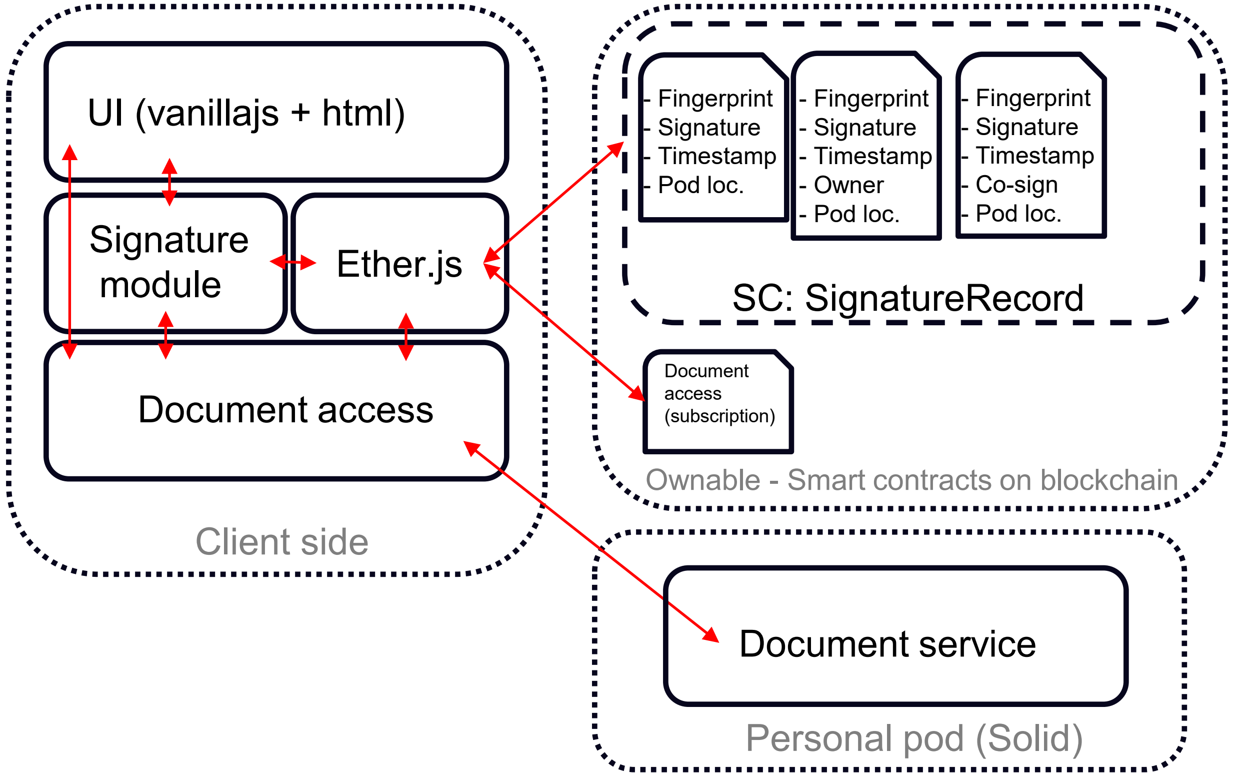


Is fingerprint the same?

Was signature valid at the timestamp time on blockchain?

Trust chain is verified by TSA and we assume that the whole trust chain can be trusted

Architecture

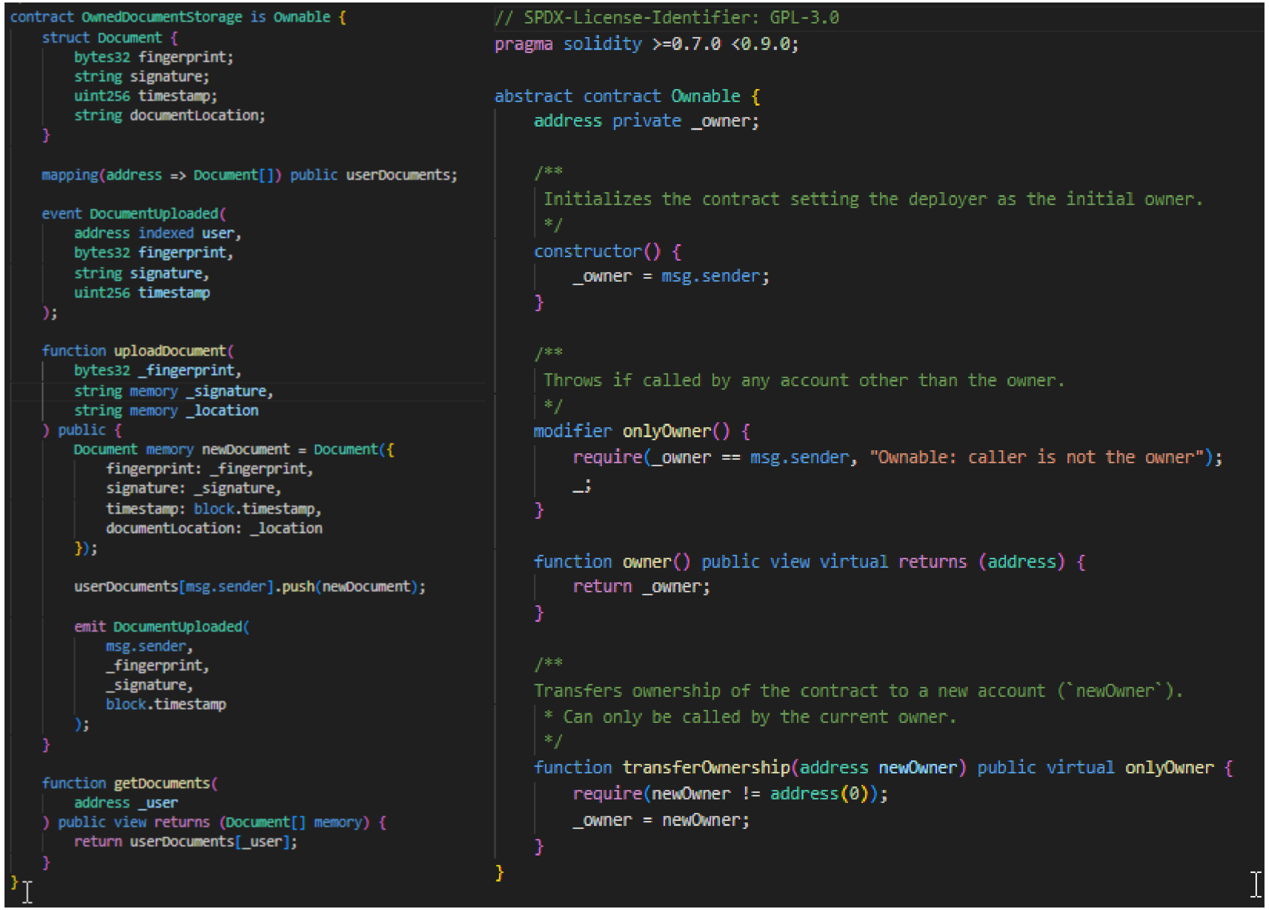


Stack:

- Typescript + Solidity
- Vite + Node.js
- Hardhat, Chai, Ethers

- Test driven development + local blockchain

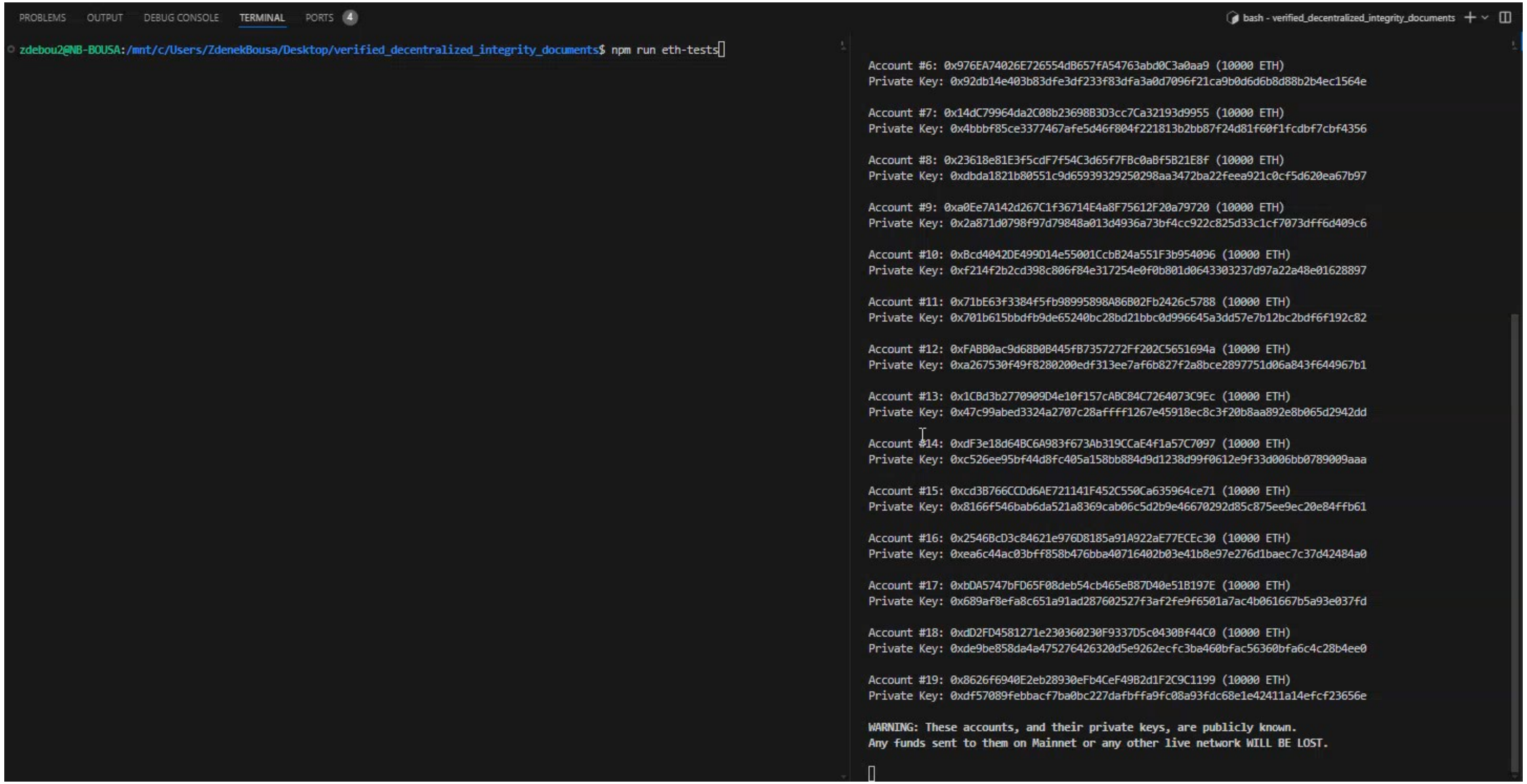
Architecture - code



```solidity
contract OwnedDocumentStorage is Ownable {
    struct Document {
        bytes32 fingerprint;
        string signature;
        uint256 timestamp;
        string documentLocation;
    }

    mapping(address => Document[]) public userDocuments;

    event DocumentUploaded(
        address indexed user,
        bytes32 fingerprint,
        string signature,
        uint256 timestamp
    );

    function uploadDocument(
        bytes32 _fingerprint,
        string memory _signature,
        string memory _location
    ) public {
        Document memory newDocument = Document({
            fingerprint: _fingerprint,
            signature: _signature,
            timestamp: block.timestamp,
            documentLocation: _location
        });

        userDocuments[msg.sender].push(newDocument);

        emit DocumentUploaded(
            msg.sender,
            _fingerprint,
            _signature,
            block.timestamp
        );
    }

    function getDocuments(
        address _user
    ) public view returns (Document[] memory) {
        return userDocuments[_user];
    }
}
```

```solidity
// SPDX-License-Identifier: GPL-3.0
pragma solidity >=0.7.0 <0.9.0;

abstract contract Ownable {
    address private _owner;

    /**
     Initializes the contract setting the deployer as the initial owner.
     */
    constructor() {
        _owner = msg.sender;
    }

    /**
     Throws if called by any account other than the owner.
     */
    modifier onlyOwner() {
        require(_owner == msg.sender, "Ownable: caller is not the owner");
        _;
    }

    function owner() public view virtual returns (address) {
        return _owner;
    }

    /**
    Transfers ownership of the contract to a new account (`newOwner`).
     * Can only be called by the current owner.
     */
    function transferOwnership(address newOwner) public virtual onlyOwner {
        require(newOwner != address(0));
        _owner = newOwner;
    }
}
```

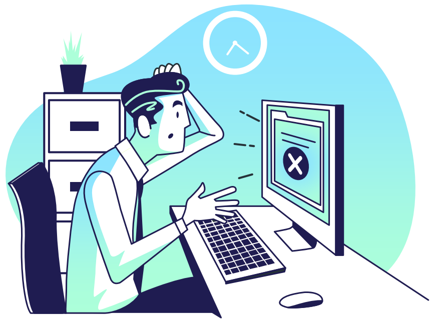
Demo



```
zdebou2@NB-BOUSA:/mnt/c/Users/ZdenekBousa/Desktop/verified_decentralized_integrity_documents$ npm run eth-tests
```

```
Account #6: 0x976EA74026E726554dB657fA54763abd0C3a0aa9 (10000 ETH)
Private Key: 0x92db14e403b83dfe3df233f83dfa3a0d7096f21ca9b0d6d6b8d88b2b4ec1564e

Account #7: 0x14dC79964da2C08b23698B3D3cc7Ca32193d9955 (10000 ETH)
Private Key: 0x4bbbf85ce3377467afe5d46f804f221813b2bb87f24d81f60f1fcdbf7cbf4356

Account #8: 0x23618e81E3f5cdF7f54C3d65f7FBc0aBf5B21E8f (10000 ETH)
Private Key: 0xdbda1821b80551c9d65939329250298aa3472ba22feea921c0cf5d620ea67b97

Account #9: 0xa0Ee7A142d267C1f36714E4a8F75612F20a79720 (10000 ETH)
Private Key: 0x2a871d0798f97d79848a013d4936a73bf4cc922c825d33c1cf7073dff6d409c6

Account #10: 0xBcd4042DE499D14e55001CcbB24a551F3b954096 (10000 ETH)
Private Key: 0xf214f2b2cd398c806f84e317254e0f0b801d0643303237d97a22a48e01628897

Account #11: 0x71bE63f3384f5fb98995898A86B02Fb2426c5788 (10000 ETH)
Private Key: 0x701b615bbdfb9de65240bc28bd21bbc0d996645a3dd57e7b12bc2bdf6f192c82

Account #12: 0xFABB0ac9d68B0B445fB7357272Ff202C5651694a (10000 ETH)
Private Key: 0xa267530f49f8280200edf313ee7af6b827f2a8bce2897751d06a843f644967b1

Account #13: 0x1CBd3b2770909D4e10f157cABC84C7264073C9Ec (10000 ETH)
Private Key: 0x47c99abed3324a2707c28affff1267e45918ec8c3f20b8aa892e8b065d2942dd

Account #14: 0xdF3e18d64BC6A983f673Ab319CCaE4f1a57C7097 (10000 ETH)
Private Key: 0xc526ee95bf44d8fc405a158bb884d9d1238d99f0612e9f33d006bb0789009aaa

Account #15: 0xcd3B766CCDd6AE721141F452C550Ca635964ce71 (10000 ETH)
Private Key: 0x8166f546bab6da521a8369cab06c5d2b9e46670292d85c875ee9ec20e84ffb61

Account #16: 0x2546BcD3c84621e976D8185a91A922aE77ECEc30 (10000 ETH)
Private Key: 0xea6c44ac03bff858b476bba40716402b03e41b8e97e276d1baec7c37d42484a0

Account #17: 0xbDA5747bFD65F08deb54cb465eB87D40e51B197E (10000 ETH)
Private Key: 0x689af8efa8c651a91ad287602527f3af2fe9f6501a7ac4b061667b5a93e037fd

Account #18: 0xdD2FD4581271e230360230F9337D5c0430Bf44C0 (10000 ETH)
Private Key: 0xde9be858da4a475276426320d5e9262ecfc3ba460bfac56360bfa6c4c28b4ee0

Account #19: 0x8626f6940E2eb28930eFb4CeF49B2d1F2C9C1199 (10000 ETH)
Private Key: 0xdf57089febbacf7ba0bc227dafbffa9fc08a93fdc68e1e42411a14efcf23656e

WARNING: These accounts, and their private keys, are publicly known.
Any funds sent to them on Mainnet or any other live network WILL BE LOST.
```

Limits



Trust & transformation

- Initial owner should be the signee
- Support by law

Actual signature prolongation

Derivated personal private key would work, but the issue is the use & verification

THANKS!

Do you have any questions?