

Project 3: Smart Tokens

ERC20 for Equity Certificates through Smart Contracts

Project Team 2

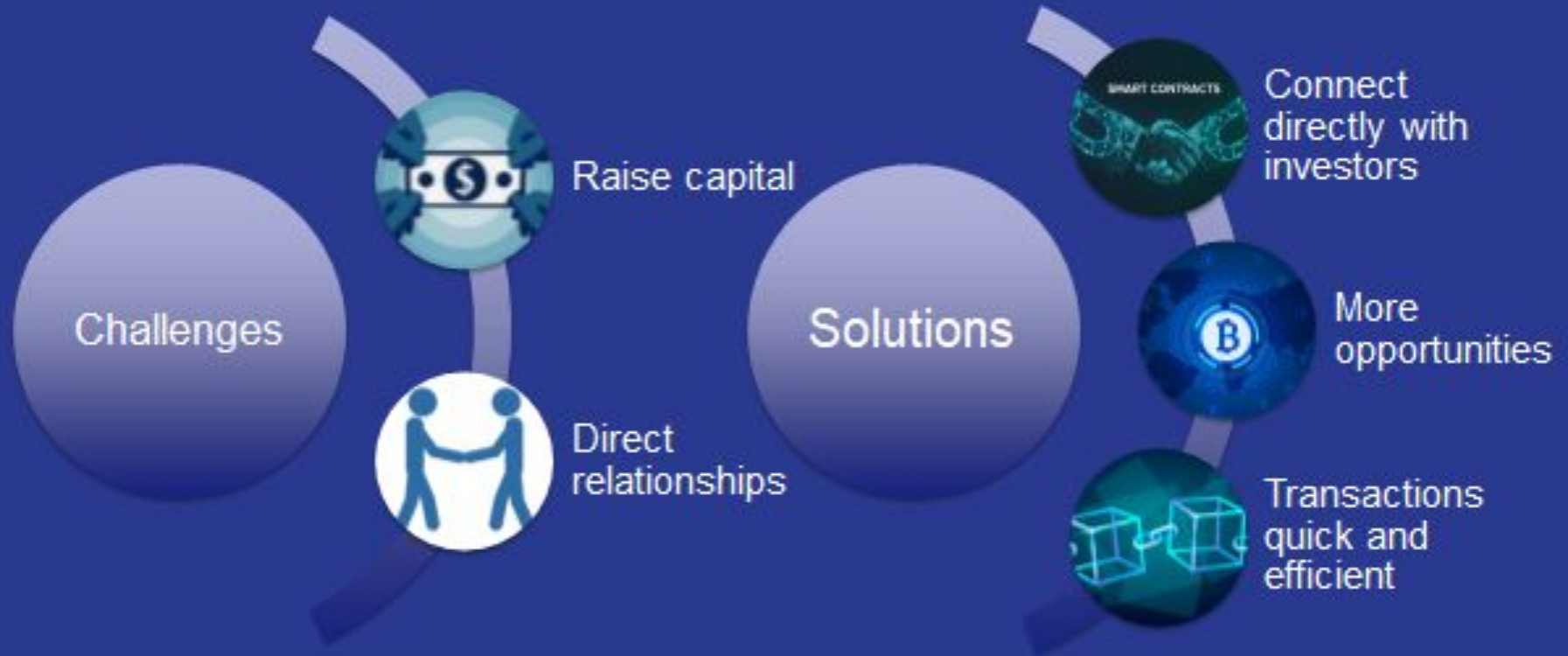
Lynette Cary, Marius Nsamou, Nigil Jeyashekar, Rajiv Shrestha, and Rawad Habib

Concept Creation Basis: Regulation of Digital Assets

According to the Commodity Futures Trading Commission (CFTC), who oversees the digital assets market:

“Digital assets offer benefits of increased transaction speed, efficiency, certainty, democratization of markets and financial inclusion, automation through smart contracts, greater liquidity for assets, and enhanced security.”

Concept Creation Basis: Invest in Private Equity Firms



Medium Article, [Invest in Private Equity? Here are the Top 4 Reasons to Get Excited for Tokenization](#)

Project Setup: Framework and Dependencies

- Smart contract created to tokenize all of the firm's equity
 - Ethereum smart contract created using Solidity
 - Created ERC20-compatible token
 - Dependencies: Remix IDE, Ganache, MetaMask, and Pinata
- Standards imported from OpenZeppelin contracts:
 - SafeMath, ERC20, ERC20Detailed, ERC20Mintable, CrowdSale, MintedCrowdSale, and Ownable

Project Setup: Testing, Deployment, and Transaction

- Smart contracts compiled and deployed to a blockchain and tested using the Ropsten Test Network on Metamask
- Variables set and used throughout the contracts:
 - Contract 'owner' set to Firm and only the Firm can mint tokens
 - Token 'symbol' set to 'DIGITAL'
 - 'Exchange_rate' set to equal 100 shares to be distributed per Ether

Smart Contract Framework: DigitalToken

Functions used in DigitalToken Contract

- **'balance'** - returns available owner shares.
- **'transfer'** - accepts recipient address to make transfer from owner to recipient.
- **'mint'** - allows the company to create new tokens when needed`
- **'purchase'** - calculates number of shares to distribute and adds value to the shares and transfers the value to the owner address.

```
contract DigitalToken {  
    using SafeMath for uint;  
  
    address payable owner = msg.sender;  
    string public symbol = "DIGITAL";  
    uint public exchange_rate = 100;  
  
    mapping(address => uint) balances;
```

Smart Contract Framework: DigitalTokenERC20

Step 1:

```
contract DigitalToken is ERC20, ERC20Detailed {  
    address payable owner;
```

Step 2:

```
    modifier onlyOwner {  
        require(msg.sender == owner, "You do not have permission to mint these tokens!");  
        _;  
    }
```

Step 3:

```
    function mint(address recipient, uint amount) public onlyOwner {  
        _mint(recipient, amount);  
    }
```

Step 4:

```
    constructor(uint initial_supply) ERC20Detailed("DigitalToken", "DIGITAL", 18) public {  
        owner = msg.sender;  
        _mint(owner, initial_supply);  
    }
```

Smart Contract Framework: DigitalTokenMinted

- DigitalTokenMinted contract passes ERC20, ERC20Detailed, ERC20Mintable to DigitalToken to mint an initial supply of tokens and additional tokens as needed.
 - Beneficiary name, token symbol, and initial token supply are constructor parameters and passed to ERC20Detailed contract.

```
contract DigitalToken is ERC20, ERC20Detailed, ERC20Mintable {
    constructor(
        string memory name,
        string memory symbol,
        uint initial_supply
    )
        ERC20Detailed(name, symbol, 18)
        public
    {
```


Smart Contract Framework: DigitalTokenSale

- Contract is created to manage the sale and minting of DigitalToken (shares).
- **Constructor parameters:**
 - rate - conversion between wei and token unit.
 - wallet - address receives the ether from sale of digital asset.
 - token - set to DigitalToken compatible with ERC20 interface required by crowdsale.
- Body of the constructor is empty and will inherit logic from OpenZeppelin: CrowdSale and MintedCrowdSale contracts.

Smart Contract Logic: DigitalTokenDeployer

- Contract created stores addresses from DigitalToken and DigitalTokenSale once deployed.
- **Parameters:**
 - 'rate' - hardcoded to 1 to maintain the same units as Ether.
 - 'wallet' - will be paid all of the Ether raised by DigitalTokenSale.
 - 'token' - location where DigitalToken is stored.

```
DigitalTokenSale token_sale = new DigitalTokenSale(1, wallet, token);  
token_sale_address = address(token_sale);  
  
token.addMinter(token_sale_address);  
token.renounceMinter();
```

Smart Contract: Deployment and Test Transaction

Digital Tokens Minted



Ropsten Testnet Network

Token Digital Token ⓘ

Ganache Accounts

(owner account) 0x06A16FE19aE657ec9b862a298995C79e1dDAB878

(investor account) 0x950A17ba3C60658114AF43C4a0839669b673B37C

Txn Hash / Block / Token / Ens



Home

Blockchain ▾

Tokens ▾

Misc ▾

Ropsten

Overview [ERC-20]

Max Total Supply: 33,333,333.3333333... DIGITAL ⓘ

Holders: 1

Profile Summary

Contract: 0xe715eaf33031f2ff3289dae35015a35ac0d0f544

Decimals: 18

ⓘ FILTERED BY TOKEN HOLDER

0x06a16fe19ae657ec9b862a298995c79e1ddab878

BALANCE

33,333,333.333333333333333333 DIGITAL

Transfers Contract

0x06a16fe19ae657ec9b862a... x



A total of 1 transaction found

First



Page 1 of 1



Last

Txn Hash	Age	From	To	Quantity
0xdaace6545aa6d1a...	7 mins ago	0x0000000000000000...	0x06a16fe19ae657ec...	33,333,333.333333333333333333

Smart Contract: Deployment and Test Transaction

Investor Purchases Digital Tokens

(2 Ether in exchange for 200 Digital Tokens)

Overview

Internal Txns

Logs (2)

State

[This is a Ropsten **Testnet** transaction only]

Transaction Hash:	0x5b77bfc3fe436d44e760b4f3b163ec28214f175b16f43c3a321dac5568fd036
Status:	Success
Block:	9803321 1 Block Confirmation
Timestamp:	35 secs ago (Mar-09-2021 01:47:36 AM +UTC)
From:	0x950a17ba3c60658114af43c4a0839669b673b37c
To:	Contract 0x9b4e312b72a500691d3aef8d4b79a0679fd678f0 TRANSFER 2 Ether From 0x9b4e312b72a500691d3a... To → 0x06a16fe19ae657ec9b86...
Tokens Transferred:	From 0x00000000000000... To 0x06a16fe19ae65... For 2 DigitalToken (DIGITA...)
Value:	2 Ether (\$0.00)
Transaction Fee:	0.00011651638816 Ether (\$0.000000)
Gas Price:	0.000000001889506011 Ether (1.889506011 Gwei)

Click to see More ↓

Smart Contract: Deployment and Test Transaction

Private Equity Firm Transfer 200 Digital Tokens



Ropsten Testnet Network

All Filters

Search by Address / Txn Hash / Block / Token / Ens



Home

Blockchain

Tokens

Misc

Ropsten

Token Digital Token

Overview [ERC-20]

Max Total Supply: 33,333,333.3333333... DIGITAL

Holders: 2

Profile Summary

Contract: 0xe715eaf33031f2ff3289dae35015a35ac0d0f544

Decimals: 18

Filtered by Token Holder

0x06a16fe19ae657ec9b862a298995c79e1ddab878

Balance

33,333,331.333333333333333333 DIGITAL

Transfers Contract

0x06a16fe19ae657ec9b862a...



A total of 2 transactions found

First



Page 1 of 1



Last

Txn Hash	Age	From	To	Quantity
0x03aaf40bb20b4660...	2 mins ago	0x06a16fe19ae657ec...	0x950a17ba3c60658...	2
0xdaace6545aa6d1a...	43 mins ago	0x0000000000000000...	0x06a16fe19ae657ec...	33,333,333.333333333333333333

Smart Contract: Deployment and Test Transaction

Private Equity Firm Issues Certificate to Investor



Ropsten Testnet Network

All Filters

Search by Address / Tx

Transaction Details

Overview

Logs (2)

State

[This is a Ropsten **Testnet** transaction only]

Transaction Hash:	0xd9d3e4846658fae796465fda9c8b0fc75609cbf0cc42436c5daa03a53d440ddf
Status:	Success
Block:	9806507 8 Block Confirmations
Timestamp:	3 mins ago (Mar-09-2021 03:17:37 PM +UTC)
From:	0x06a16fe19ae657ec9b862a298995c79e1ddab878
To:	[Contract 0x7655f13a47600d9bc68bb3f3e128c3ad8ccf6d47 Created]
Value:	0 Ether (\$0.00)
Transaction Fee:	0.002030409 Ether (\$0.000000)
Gas Price:	0.000000001 Ether (1 Gwei)

[Click to see More](#)

Discussion

- Advantage of using blockchain technology and smart contracts is to increase the private equity firm's access to raise capital and provides investors with direct access to purchase equity in the firm.
- Blockchain is limited to the amount of data that can be stored.
 - Current smart contract framework in this project benefits a small private equity firm, with low number of transactions.
 - For larger firms with high number of transactions, we explored the option of using IPFS to store large amounts of data and certificates including the benefits that IPFS offers to provide verifiability on behalf of the firm.
- Each time tokens and certificates are issued, there is an increase in gas fees to the firm .
 - While there are various methods that can be utilized to help lower gas fees, we specifically identified that reducing the storage and size of variables used in writing smart contracts can help to reduce gas costs.

Post-Mortem

- **Difficulties**

- Minting digital tokens required several iterations when transacting using metamask as smart contracts were not a single solidity file.

- **Additional questions for future research**

- If more time was available the group could explore adding machine learning models and algorithms to analyze large amounts of data.
- Further develop smart contracts to fully utilize IPFS for storing data and certificates.



Questions?