



FORDHAM
THE JESUIT UNIVERSITY OF NEW YORK

Gabelli School
of Business

Security Token Offering (STO) Platform: Ethereum-Based Digital Assets

Blockchain Application Development
Prof. Icheng Robert Chiang & Prof. Thomas Chu
ISGB-799Y
Spring 2021

Group 4:
Prannoiy Chandran
Bach Duong Ho

Contents

Executive Summary	2
Business Goal	3
Project Methodology	4
System Implementation	6
Evaluation	12
Conclusion	15
References	16

Executive Summary

The goal of the project was to build a decentralized application (dApp) running on the Ethereum blockchain that would allow security tokens to be minted, requested and distributed. Security tokens are digital, liquid contracts linked to external assets with tangible value. They are issued in a security token offering (STO), which is notably different from initial coin offerings (ICOs) in the sense that security tokens are linked to assets that already have value. The objective was to provide a robust protocol for peer-to-peer (P2P) transactions while opening the door to new kinds of structured financial products.

The underlying smart contract in the application's backend was written using Solidity and based on the guidelines defined by the ERC20 standard for issuing Ethereum-based tokens. The token transfer protocol used delegate accounts to ensure rapid transfers. The fronted user interface was built using JavaScript and the React library. A MetaMask wallet connected to the Rinkeby test network was used to cover gas (transaction) fees during testing. Ganache was used as the Web3 provider to connect to the Ethereum blockchain during deployment.

The resulting web-based application provides an intuitive interface to users to request or issue security tokens. The ERC20-compliant tokens are fungible and can be exchanged on secondary marketplaces. Considerations for future deployments, including added functionality and improved design, were also discussed.

Business Goal

Currently, decentralized finance (DeFi) represents one of the most exciting frontiers in the use cases of blockchain technology. DeFi-based platforms allow users to lend or borrow funds, speculate on price movements, store value in assets, insure against risk and earn interest. Many of the functions that are currently served by the financial system can potentially be implemented on the blockchain without the need for 3rd-party intermediaries such as brokers, banks and clearinghouses.

One of the most significant developments in the DeFi space has been the increasing prevalence of security tokens. Security tokens differ from cryptocurrencies in the sense that tokens are linked to an external asset that already has value, and the token itself serves as a means to preserve a fractional ownership stake on a digital ledger. There is also a great degree of flexibility in how the tokens are structured, as they can be fractionalized or aggregated. Investors need not be accredited investors with a high net worth; they can invest as long as they pass KYC checks set by the token issuer.

Tokenization removes legacy middlemen and optimizes processes, reducing costs, administration work and time. Liquidity is ensured due to the low cost of distribution and record management. Security tokens are therefore less speculative than most cryptocurrencies, while providing access to all investors. Tokenizing an asset involves creating security tokens linked to its value and distributing them to investors in a security token offering (STO). This is an example of a peer-to-peer (P2P) financial transaction that occurs without 3rd-party intermediaries.

The goal was to build a decentralized application (dApp) running on the Ethereum blockchain that would allow security tokens to be minted, requested and distributed. DApps combine backend code (smart contracts running on a P2P network) with a frontend user interface. The objective was to provide a robust protocol for P2P transactions while opening the door to new kinds of structured financial products. The security token space, like many others in enterprise and consumer technology today, consists of numerous applications that focus on a single functionality and integrate with other applications to provide end-to-end functionality. Within this ecosystem, the proposed application has a focused scope and falls under the issuance platforms category.



Figure 1: Security token ecosystem

Project Methodology

Market research was carried out to understand the current state of the security token space and to identify the key functionalities of existing token issuance platforms. There are more than 70 active token issuance and primary trading platforms for security tokens. Most of them are located

in the United States (35%) and the European Union (30%). Within the EU, the clear leader is Germany, followed by Switzerland. Most token issuance platforms use the Ethereum blockchain to tokenize assets, but there is an increasing number that are turning to the Stellar blockchain. Stellar is becoming a serious competitor in the race for the ‘golden’ standard, but the well-established Ethereum blockchain was selected as the basis for this project.

It is extremely important that a token standard is interoperable with exchanges, which means that tokens created on the platform should be compatible with the exchanges. So far, there are only a few active exchanges. It is not that difficult to make sure that interoperability is fulfilled. However, if more exchanges enter the marketplace, the results are difficult to anticipate. Another key concern is making sure that tokens issued are tradeable on secondary markets. Some platforms are trying to build all-in-one platforms, while other platforms are more keen on specializing only on one specific service and making sure that tokens issued on their platform will be tradeable on secondary markets. For this project, the scope was narrowed to token issuance and the ERC-20 standard was used as a guideline to ensure interoperability and fungibility. The standard specifies a number of methods and events that must be used in the smart contract to ensure an adequate level of functionality and security.

System Implementation

a. Application framework

Participants	<ul style="list-style-type: none"> • Asset Owners • Investors 	<p>Owners mint and issue tokens linked to the external asset they own. Investors purchase fractional ownership shares by requesting tokens.</p> <p><u>Functional requirements</u></p> <ul style="list-style-type: none"> • <u>Asset Owners</u>: name tokens, mint new tokens, set total supply, approve delegate accounts to carry out transfers of up to a certain token count • <u>Investors</u>: request a particular token count
Assets	Security tokens	ERC20-compliant tokens linked to external assets with tangible value (companies, real estate, cars, watches, art, etc.)
Transactions	<ul style="list-style-type: none"> • totalSupply() • transfer() • approve() • transferFrom() • allowance() • name() • symbol() • decimals() 	Methods based on the ERC20 standard to provide a robust protocol for verifying and issuing tokens without 3rd-party intermediaries (using delegate accounts within the backend as temporary intermediaries).
Events	<ul style="list-style-type: none"> • Transfer • Approval 	Triggered when certain prerequisites in the smart contract are met and the respective function has been successfully executed.

b. Smart contract deployment

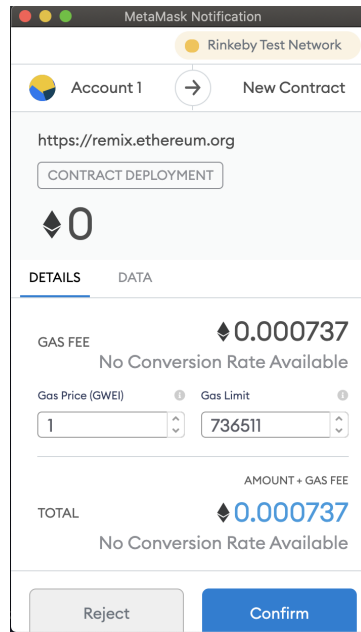


Figure 2: MetaMask wallet connected to Rinkeby testnet

MetaMask was used as the Ethereum wallet that enabled deployment tests. Ethereum tokens were obtained from the Rinkeby test network, and used to meet the gas fee (Ethereum's transaction fee) requirements whenever the smart contract was deployed during tests and production.

```
function transfer(address receiver, uint numTokens) public returns (bool) {
    require(numTokens <= balance[msg.sender]); // owner must have enough tokens for transfer

    balance[msg.sender] = balance[msg.sender] - numTokens; // only owner can approve transfers at STO stage
    balance[receiver] = balance[receiver] + numTokens;
    emit Transfer(msg.sender, receiver, numTokens);
    return true;
}
```

Figure 3: Solidity function to transfer tokens

The smart contract was written on the Remix integrated developer environment (IDE). Solidity, an object-oriented programming language, was used to implement the contract. One of 2 main events involved was the Transfer process, which is triggered when the owner approves a request

and the owner possesses enough tokens to provide the required number of tokens. The inclusion of “msg.sender” (the account creating the call) ensures that only the owner has the ability to authorize transfers within the platform.

```
// using a delegate account to withdraw tokens from owner's account & enable transfer
function approve(address delegate, uint numTokens) public returns (bool) {
    allowed[msg.sender][delegate] = numTokens; // no. of tokens delegate is allowed to help transfer
    emit Approval(msg.sender, delegate, numTokens); // delegate is approved
    return true;
}
```

Figure 4: Solidity function to approve delegate accounts to handle transfers

To facilitate the transfer process, delegate accounts were used to withdraw and transfer the required number of tokens. This method is a key aspect of the ERC20 standard for issuing tokens. In the “approve” function, the owner approves a delegate account (a temporary and internal intermediary) to help with the transfer process. The Approval event is fired at the end of the function. In this manner, the delegate can transfer tokens of up to a certain amount (having already been approved by the owner to withdraw that amount) without waiting for further approval by the owner. This ensures the approval and transfer process has minimal delays and the process is straightforward for both owners and investors.

```
function transferFrom(address owner, address investor, uint numTokens) public returns (bool) {
    require(numTokens <= balance[owner]); // owner must have enough tokens for transfer
    require(numTokens <= allowed[owner][msg.sender]); // delegate must have allowance for transfer amount

    balance[owner] = balance[owner] - numTokens;
    // delegate can divide allowance into multiple transactions until balance = 0
    allowed[owner][msg.sender] = allowed[owner][msg.sender] - numTokens; // subtract tokens from delegate's allowance
    balance[investor] = balance[investor] + numTokens; // investor receives tokens
    emit Transfer(owner, investor, numTokens); // transfer takes place
    return true;
}
```

Figure 5: Solidity function to transfer tokens via delegate accounts

The transferFrom function starts by verifying that the delegate has been approved by the owner (to execute a transfer of up to the requested number of tokens) and the owner has enough tokens to cover the transfer. The allowance is reduced by the number of tokens being transferred. The

delegate is able to split up the allowance and execute multiple withdrawals if needed. The function concludes by firing the Transfer event.

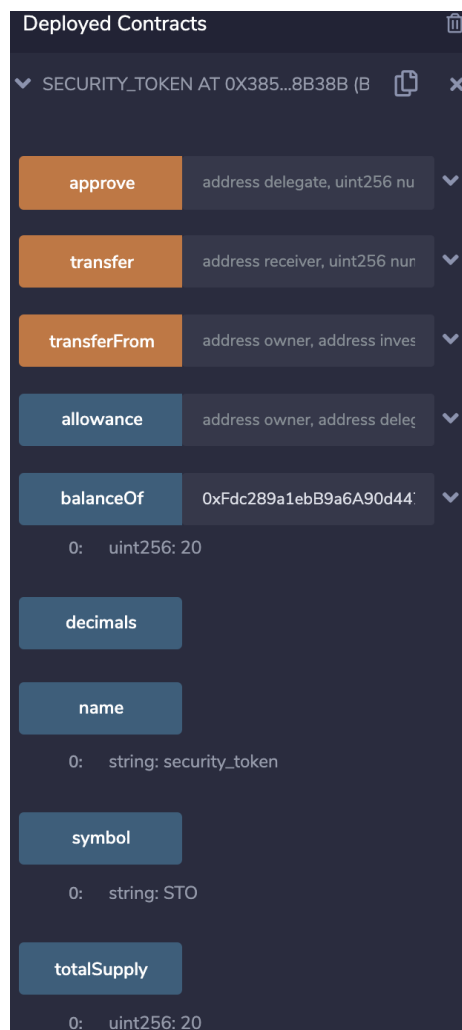


Figure 6: Deployed contract on Remix

During testing, the contract was deployed with the total supply set at 20 tokens. This amount is reflected in the “Deployed Contracts” section on Remix, with the owner initially having a balance (balanceOf) equal to the total supply (totalSupply) before transfers are carried out. The token in this case is called security_token and its ticker symbol is STO.

CURRENT BLOCK
2

GAS PRICE
2000000000

GAS LIMIT
6721975

HARDFORK
MUIRGLACIER


NETWORK ID
5777

RPC SERVER
HTTP://127.0.0.1:7545

MINING STATUS
AUTOMINING

WORKSPACE
SECURITY-TOKEN

SWITCH



TX HASH

0x4642e81fae62d37ea60e95506d376717421bd26053ec40d45d06902f9f21c531

CONTRACT CREATION

FROM ADDRESS

0xFdc289a1ebB9a6A90d447179484199a39FFD75Be

CREATED CONTRACT ADDRESS

0x38575775A6484e642310036035e0274b9Ef8B38b

GAS USED

733011

VALUE

0

Figure 7: Deployed contract on Ganache

Ganache was selected as the Web3 provider, used to run a geth (Go Ethereum) node to communicate with the Ethereum blockchain and deploy the smart contract during testing. The Remix contract was deployed on Ganache, and the resulting transaction block included a hash and contract address.

c. User interface (UI)

The user interface for the application was built using JavaScript and React, a popular library for front-end interfaces. The contract address generated by Ganache was used to connect the smart contract to React. In addition, the application binary interface (ABI) was used to connect the contract on Remix to React. The ABI helps to encode Solidity contract calls for the Ethereum Virtual Machine (EVM) and is also crucial to specify which function in the contract to invoke.

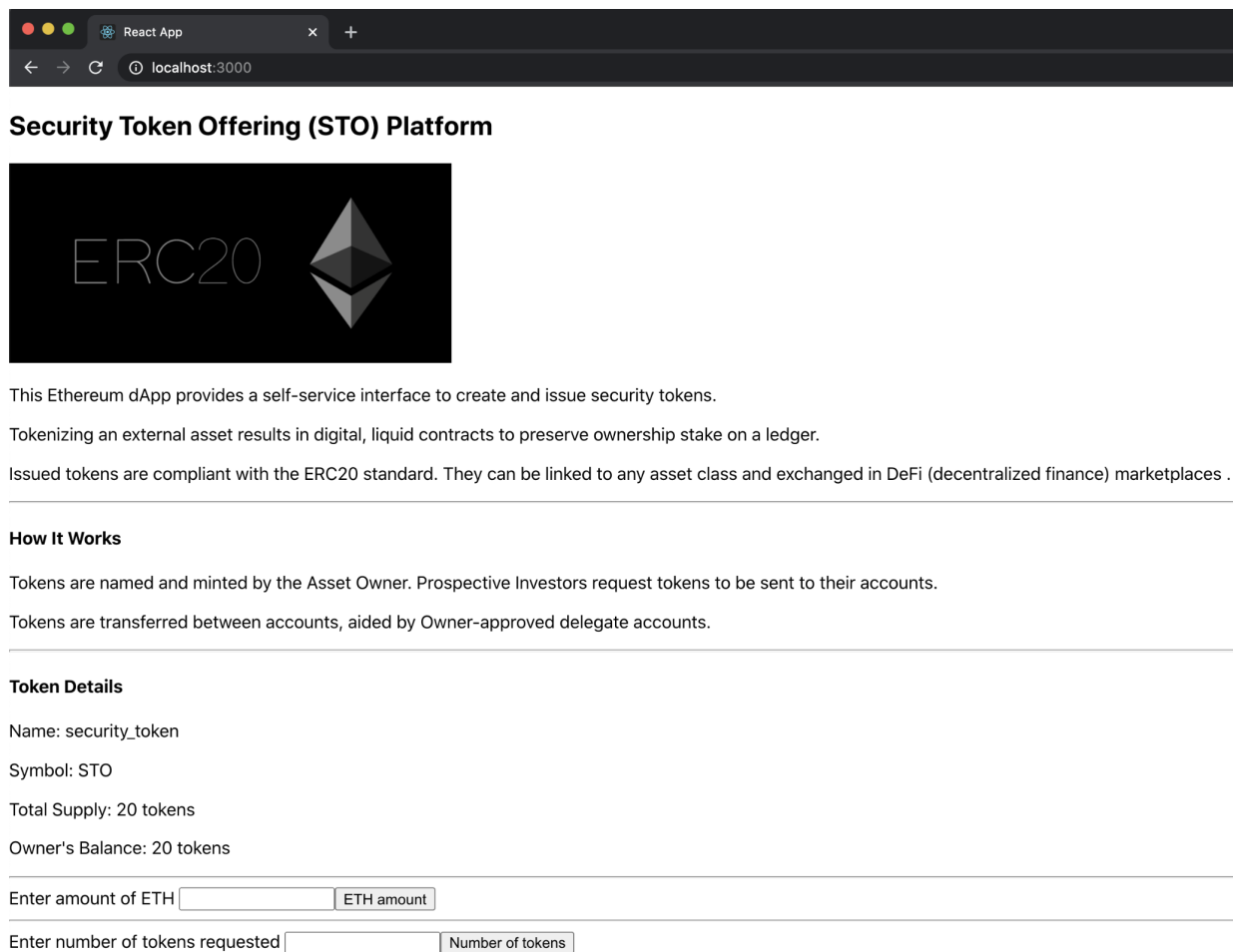


Figure 8: STO platform's user interface

The web-based UI explains the purpose of the platform as well as instructions for use. The name, symbol and total supply of the token in question are displayed, along with the asset owner's current balance of tokens. The user is prompted to enter the number of tokens requested and the amount of ether to cover the transaction's gas fees (to be deducted from the linked MetaMask wallet).

The owner's balance will initially be equal to the total supply, and will decrease as the tokens are distributed to investors (provided the total supply is not changed). Fractional ownership of the asset will have changed hands by the end of each successful transaction.

Evaluation

a. System performance

Overall, the application performed as expected. During tests, the owner was able to set the total supply and approve delegate accounts to transfer tokens by approving an allowance.

b. Considerations for future deployments

A number of requirements must be met for enterprise-scale deployment. The P2P network must also be able to scale as the application gains more users. Security is another key concern in any decentralized application. Users must be able to sign transactions without compromising their secret keys, and authentication protocols would ideally meet the industry-standard OAuth 2.0. Another change that could be made to the backend would be including the SafeMath library to help prevent overflows during arithmetic operations.

In future deployments, asset owners and investors will ideally have different views on the platform based on their roles within the transaction: token issuer or requester. Wireframes were developed on Balsamiq to envision what the interface would look like for each user to provide added functionality and an enhanced user experience.

≡ Pending Claim ▾ ⋮

Investor

Number of tokens requested

Rules satisfied?

Mint Tokens

Close Claim

≡ Dividend Distribution ▾ ⋮

Investor 1

Dividend Amount

Send Dividend

Investor 2

Dividend Amount

Send Dividend

Total Tokens Issued

Claim Manager

Rules & Preferences

Token Issuer Profile

Figure 9: Wireframe of asset owner's (token issuer) view (proposed)

The token issuer will have access to key details for each pending claim, including the requesting user's KYC status. The issuer will also be able to set the rules that investors must meet to have their claims accepted. The process of distributing dividends, in the case of tokens linked to corporations, can be largely automated. The issuer will also have access to a dashboard to track the number of users who have submitted claims, the number of tokens issued and how fractional ownership of the asset has changed over time.

≡ Submit Claim ▾ ⋮

Number of tokens requested

3

Comments

Current Account

Completed Claims

Pending Claims

Investor Profile

Figure 10: Wireframe of investor's (token requester) view (proposed)

Prospective investors will be able to request tokens and include comments if necessary. They will also be able to see the rules they will have to meet and submit the necessary documents and

information for KYC checks. They will also be able to track the completed and pending claims sent from their profiles.

The tokens issued are also designed to be fungible, so that they can be exchanged and traded for assets of equivalent value (tokens from the same application or any other ERC20-compliant token) on DeFi marketplaces. In addition, the tokens may need to be used on other applications. For instance, investors will require wallets to store their tokens and secondary markets to trade them on, while owners may need to use compliance services to carry out KYC (know your customer) checks. Therefore, the application is intended to integrate with other applications as part of a larger security token ecosystem.

c. Feasibility assessment

Criteria	Assessment
Supports key objectives	The application meets the goal of enabling seamless P2P transactions and enabling the creation of new structured financial products. There is a great degree of flexibility with regards to asset class as long as the external asset already has tangible value.
Strong customer support	There is a clearly demonstrated now for token issuance platforms with robust protocols
Realistic technical requirements	The application's focused scope ensures that it is lightweight in its current form. The main technical concerns will be security measures and having a user-friendly interface.
Realistic timeframe (1 year or less)	The technical, functionality and UX-related changes discussed can be implemented in a 6-8 month timeframe.
Low risk	There are minimal internal dependencies.

	External dependencies can be minimized by selecting well-established integration partners.
Summary	There is significant potential for long-term expansion if more functionality is added, security measures are enhanced and UI/UX is improved.

Conclusion

In conclusion, the application met the functional requirements and business objectives outlined at the start of the project. The application issues tokens that record ownership stake in external assets on an immutable ledger. Asset owners can set the total supply of tokens and control the distribution of tokens to prospective investors.

Transactions can take place rapidly and efficiently without needing to wait for 3rd-party intermediaries and processing times. Much of the hassle of handling physical assets is also eliminated, with the digital-oriented process serving investors who are using assets primarily as stores of wealth or speculating on the assets' prices. The issued tokens also comply with Ethereum's ERC20 standard and can be traded on secondary marketplaces for fungible tokens. Interoperability between platforms is also ensured, allowing the application to seamlessly integrate with other applications as part of a security token ecosystem. The widespread adoption of security token-oriented applications represents a crucial step forward in unlocking the potential of blockchain technology in DeFi, opening the door to new kinds of structured financial products and introducing a new dimension to P2P transactions.

References

<https://jonasgross.medium.com/the-state-of-the-security-token-ecosystem-token-issuance-and-primary-trading-platforms-fc9acc19ac7b>

<https://ethereum.org/en/developers/docs/standards/tokens/erc-20/>

<https://polymath.network/issuers>

<https://tokeny.com/the-security-token-ecosystem/>