2024 6th Conference on Blockchain Research & Applications for Innovative Networks and Services (BRAINS) | 979-8-3503-6784-3/24/$31.00 ©2024 IEEE | DOI: 10.1109/BRAINS63024.2024.10732268

On Tokenizing Securities in Contemporary Decentralized Finance Ecosystems

Reina Ke Xin Li*, Srisht Fateh Singh*, Andreas Park†, Andreas Veneris*§
* Department of Electrical and Computer Engineering, University of Toronto
§ Department of Computer Science, University of Toronto
† Rotman School of Management, University of Toronto
{reinakx.li, srishtfateh.singh}@mail.utoronto.ca, andreas.park@rotman.utoronto.ca, veneris@eecg.toronto.edu

***Abstract*—This paper presents a securities tokenization solution allowing real-world securities to benefit from the accessibility, transparency, efficiency, and innovation of blockchain and decentralized finance. While existing solutions limit their scope to specific types of securities, our solution is generalizable to the tokenization of any securities with any holding rights by representing the security by a fungible token and using separate smart contracts for shareholders to redeem the holding rights. Furthermore, unlike existing solutions, our solution addresses the complications regarding ownership accounting that arise from the decentralized nature of liquidity pools, a pivotal component of the decentralized finance ecosystem. Our solution achieves this by performing accounting off-chain with additional logic for liquidity pools. In this paper, we implement our solution on Ethereum, measuring its gas costs to demonstrate that it is 27% cheaper than existing solutions. We also analyse the liquidity logic of over 90% of liquidity pools on Ethereum, confirming their compatibility with our solution. Moreover, we demonstrate the application of our solution for three use cases: dividend-paying stocks, common stock, and mergers and acquisitions.**

***Index Terms*—tokenization, blockchains, finance, smart contracts, decentralized applications**

I. Introduction

Blockchain technology is a powerful tool promoting fairness, transparency, accessibility, efficiency, and security, properties desirable in finance. These advantages have led to the development and enthusiastic adoption of various *decentralized finance* (*DeFi*) protocols leveraging blockchain with the goal of democratizing finance, removing intermediaries, and promoting innovation [1]. As such, today's DeFi ecosystem provides viable on-chain alternatives for traditional financial systems and infrastructures.

Bringing real-world securities onto the blockchain, a process called *tokenization*, allows investors to take advantage of the accessibility and transparency of blockchains, and has the potential to eliminate intermediaries in the custody, management, and trading of the underlying assets. It can also provide access to secondary markets and on-chain liquidity through DeFi protocols. The process of tokenization is akin to the issuance of American Depositary Receipts (ADRs), exchange-tradable certificates issued by a U.S. depositary bank representing a share of a foreign company's stock [2]. Similarly, a tokenization scheme entails a deposit of shares with a "trusted" custodian *issuer*, who issues on-chain tokens representing these shares so that they may be traded on blockchain markets, creating an on-chain representation of a traditional financial instrument.

One asset of interest is publicly traded stocks because traditional stock exchanges have direct on-chain DeFi analogs: *decentralized exchanges*. Decentralized exchanges are online, open, and eliminate the need for brokers, clearinghouses, custodians, market makers, and other intermediaries necessary for traditional stock trading [3]. Thus, with stock tokenization, the functions of traditional stock exchanges can be translated seamlessly on-chain while improving accessibility, transparency, and efficiency for investors. In fact, the authors in [4] argue that investors could save 30% of trading costs if stock trading was organized using optimally designed, blockchain-based automated market makers.

A major challenge faced by stock tokenization in DeFi ecosystems is *ownership attribution*. A stock is an investment contract that provides the owner with different rights, such as the right to vote in shareholder meetings and the right to dividends [5], [6]. The issuer of tokenized shares of a stock must be able to distribute the rights associated with the stock to the correct owners. Although tokens on blockchains have well-defined holding accounts, in some cases, the account may be a smart contract which is only holding the tokens on behalf of their owners [7]. Furthermore, some DeFi protocols, such as lending protocols and decentralized exchanges, hold assets in liquidity pools, and the ownership of each token is not well-defined. Previous works on securities tokenization fail to adequately address this challenge, as they either ignore assets held by smart contracts [8]–[14], or restrict the tokenized asset's use to custom permissioned DeFi protocols with additional functionalities [15], thereby eliminating key advantages of DeFi such as the inter-operability with most decentralized applications. Meanwhile, existing security token standards face the same problems and focus only on compliance enforcement and regulated transfers [16], [17].

This paper presents a solution for tokenizing stocks that translates the stock's holding rights on-chain and addresses the challenges of ownership accounting. Our solution involves three parts: *(i)* a *Stock Token Contract* which provides a liquid tokenized representation of the stock on-chain; *(ii)* the *Off-Chain Accounting* procedure which calculates the ownership of tokenized shareholdings; and, *(iii)* a *Rights Redemption Contract* which allows tokenized shareholdings to be redeemed

for rights on-chain. Although the work presented here focuses on stocks, it also applies to other forms of securities such as options, bonds, etc.

A Stock Token Contract implements a fungible and liquid token, making it operable with general DeFi protocols. The Off-Chain Accounting eliminates the gas costs involved in the accounting, allowing for the complex calculations required for dealing with smart contracts such as liquidity pools and for supporting an unbounded number of on-chain shareholders. It works by calculating both shareholders' wallet balances and their contributions to DeFi pools at a particular block using blockchain event queries. Furthermore, it is adaptable to the evolving regulatory landscape as the off-chain nature allows the accounting procedure to be easily upgraded, in case, for example, it is decided that shareholders do not own tokens they contribute to DeFi pools. The result of accounting is brought on-chain by a signed message from the issuer that allows on-chain shareholders to redeem rights through a Rights Redemption Contract that performs signature verification, which can be customized to distribute arbitrary holding rights.

The proposed solution has low gas costs, costing at least 27% less gas than other solutions for tokenizing dividend-paying stocks presented in [8] and [18]. Furthermore, as opposed to the solutions in [8] and [18], our solution is not limited to dividend-paying stocks—it can represent stocks with arbitrary holding rights such as voting rights. Crucially, our solution allows inter-operability with general DeFi protocols including over 90% of liquidity pools, and it can upgrade its accounting procedure without incurring additional costs of on-chain smart contract updates.

The remainder of the paper is organized as follows: Section II provides a background of the relevant tools and concepts, Section III details the proposed solution, Section IV presents an evaluation of the proposed solution and a comparison against other solutions, Section V discusses the assumptions and use cases of the proposed solution, and Section VI provides an overview of the related works.

II. Background

This sections presents the blockchain concepts relevant to the proposed solution, including the token standards and cryptographic algorithms used in the solution, and the DeFi protocols that the solution is designed to operate with.

A. Ethereum Request for Comment Token Standards

Ethereum Request for Comment (ERC) token standards provide standardized APIs for interacting with specified token types, such as ERC20 and ERC721. An ERC20 token is a fungible token smart contract that tracks the balances of its holders [19]. An ERC721 token is a Non-Fungible Token (NFT) smart contract, where each token has a unique tokenId that is mapped to its owner and other additional data [20].

B. Elliptic Curve Digital Signature Algorithm

Elliptic Curve Digital Signature Algorithm (ECDSA) is built upon elliptic curve cryptography, a form of public-key cryptography that leverages the algebraic structure of elliptic curves over finite fields [21]. It is the algorithm used to generate Ethereum public and private keys and to sign and verify Ethereum transactions [22]. OpenZeppelin provides an audited library for the safe implementation of ECDSA signatures on-chain [23].

C. EIP-712

EIP-712 defines a procedure for hashing and signing typed structured data [24]. In EIP-712, "\x19\x01" is prepended to messages before they are signed, followed by a domain separator and the hash struct. The domain separator encodes the name and version of the signing domain (e.g. name and version of the application), the active chain ID, and the verifying contract's address. The hash struct encodes the structured message datatypes and data.

D. DeFi Protocols

The DeFi protocols of interest in this work are lending protocols and Automated Market Makers (AMMs), which have liquidity pools in which users deposit tokens.

1) Lending: Decentralized lending platforms replace lenders with liquidity providers, who provide liquidity to a lending pool by depositing tokens. They are issued ERC20 liquidity pool tokens representing their deposits, which they can burn to make withdrawals from the pool. A liquidity provider's ownership of tokens in the pool is proportional to the amount of liquidity they provide. For instance, on Aave, Spark, and Morpho, liquidity providers can withdraw the same amount of tokens as they deposited [25]–[28]. Meanwhile, on Compound and Fluid, the amount is scaled by the platform's exchange rate [29]–[31].

2) AMMs: AMMs are decentralized exchanges that replace traditional order book pricing with liquidity pools and algorithmic pricing. Liquidity providers deposit tokens into an AMM's liquidity pool at a rate determined by the AMM's invariant function, which are then used by exchange users to swap against. The pool can be structured as a uniform liquidity pool or as a concentrated liquidity pool.

Uniform liquidity AMMs: In these AMMs, liquidity is distributed uniformly across the entire price range. Liquidity providers are issued ERC20 liquidity pool tokens to represent their positions, which they can burn to withdraw tokens from the pool. Their ownership is, again, proportional to the amount of liquidity they provide [32]. Protocols such as Uniswap V2, Sushiswap V2, and PancakeSwap V2 use a Constant Product AMM (CPAMM), wherein the product of token balances in the pool is fixed [3], [33], [34]. Other protocols like Balancer [35] use a constant geometric mean invariant with support for multiple tokens per pool, while Bancor [36], [37] extends the CPAMM model with single-sided liquidity provision.

Concentrated liquidity AMMs (CLAMMs): CLAMMs allow liquidity providers to distribute their liquidity along a selected price interval [38]. The price interval and the liquidity distributed along it is called a liquidity position and is represented by an ERC721 NFT. CLAMMs maintain an invariant for each position, $L = f(x, y)$, where L is the liquidity amount, and x, y are the (virtual) token balances [39].

The price is represented as a function of ticks, $p(i)$, where i is the tick, and is equal to $\frac{dy}{dx}$. Then, the invariant can be rewritten as $y = g(p, L)$ and $x = h(p, L)$ for some functions g and h. For instance, in Uniswap V3, Sushiswap V3, and Pancakeswap V3, the invariant is $L = \sqrt{xy}$, the price is $p = \frac{y}{x}$, the tick-to-price mapping is $p(i) = 1.0001^i$, and $x = \frac{L}{\sqrt{p}}$ and $y = L\sqrt{p}$ [33], [34], [38]. In general, Equations (1) and (2) show the ownership of x and y tokens in the pool for a position with L liquidity in the interval $(p(i_l), p(i_u))$, when the current price is $p(i_c)$.

$$\Delta x = \max\{h(\max\{p(i_c), p(i_l)\}, L) - h(p(i_u), L), 0\} \quad (1)$$

$$\Delta y = \max\{g(\min\{p(i_c), p(i_u)\}, L) - g(p(i_l), L), 0\} \quad (2)$$

E. Homemade Dividends

In traditional finance, Modigliani-Miller Dividend Irrelevance implies that, instead of being paid dividends, shareholders can, equivalently, create *homemade dividends* by selling portions of their shares for cash [6]. This concept can be used for the tokenization of dividend-paying stocks, wherein the tokenized stock is an ERC20 token pegged to a reserve of physical shares of the stock. When dividends are paid, the issuer uses the dividend to buy physical shares of the stock from a traditional exchange and adds these shares to the reserve. After the reserve is increased, each ERC20 token's pegged value increases to the new ratio of shares in reserve to tokens in circulation. Any on-chain price not reflecting this value increase creates an arbitrage opportunity where one can buy the token on-chain for less and redeem it off-chain for more. Thus, arbitrageurs will ensure price responsiveness in on-chain markets, allowing the token's value to increase monotonically with every dividend payment. As a result of the value increase and the divisibility of ERC20 tokens, shareholders wanting cash dividends may create homemade dividends on-chain.

For instance, let x be the number of tokens in circulation and s be the shares in the reserve, so that one token represents $\frac{s}{x}$ shares. A cash dividend of d per share (and thus $\frac{sd}{x}$ per token) is paid out, and the issuer receives sd in dividends and buys $\frac{sd}{p}$ shares, adding them to the reserve, where p is the off-chain share price after dividend payout. Then, the token represents $\frac{s}{x} + \frac{sd}{px}$ shares. The value of the token on-chain converges to $\frac{sp+sd}{x}$ due to arbitrage, so shareholders can sell $\frac{sd}{px}$ tokens on an exchange like Uniswap V3 to get $\frac{sd}{x}$ in cash (ignoring fees), which is the original cash dividend per token.

This solution is lightweight and inexpensive: dividend payments only require gas costs if shareholders choose to create homemade dividends. It also does not disrupt liquidity pools as the reserve increases are agnostic to any on-chain DeFi infrastructures. However, it can be expensive to swap small amounts on DeFi exchanges as the gas costs do not scale with transaction value. Furthermore, the solution cannot generalize to stocks that guarantee its holders rights other than dividends, such as voting.

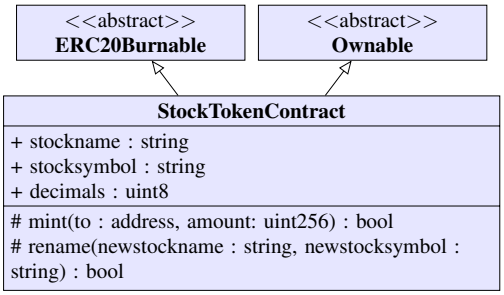


Fig. 1. StockTokenContract class conforming with ERC-20 standard

III. Tokenizing Stock Securities

The proposed solution allows stocks to be represented and traded on the blockchain, tapping into the resources and liquidity available on-chain and enabling investors to take advantage of DeFi efficiencies and decentralization. Meanwhile, the solution does not sacrifice a stock's real-world properties such as the holding rights with which it is associated, nor does it compromise the tokenized stock's usability on-chain. This section presents the proposed solution in detail in its three parts: the Stock Token Contract, the Off-Chain Accounting procedure, and the Rights Redemption Contract.[1]

A. Stock Token Contract

The Stock Token Contract, deployed by the issuer, extends the OpenZeppelin ERC20 token smart contract, enabling compatibility with DeFi protocols described in Section II-D. While standard ERC-20 tokens have just a name and symbol, this contract also includes a `stockname` and `stocksymbol`, which can be updated to allow for changes in the stock's name and symbol. The contract class is depicted in Figure 1.

B. Off-Chain Accounting

Accounting share ownership at a block (called the cutoff block) is done by querying the blockchain's events off-chain. First, each shareholder's wallet balance at the cutoff block is calculated by querying the Stock Token's logged `Transfer` events, which are emitted upon transfer of any ERC20-compatible token, up to the cutoff block. The wallet balance is the net of incoming and outgoing transfer amounts of that wallet address. Next, the shareholders' balances held in liquidity pools must be accounted for. These tokens may be held in lending pools or uniform liquidity AMMs (uniform liquidity pools), or CLAMMs. To do this, the issuer will determine the proportion of total liquidity that each shareholder can withdraw from the pool at the cutoff block. The issuer maintains a whitelist of valid liquidity pools in each category by keeping a list of addresses of these pools, and performs accounting for these pools.

1) Uniform Liquidity Pools: For each lending pool and uniform liquidity AMM, the issuer first calculates the pool's Stock Token balance at the cutoff block by querying the Stock Token's logged `Transfer` events and netting the transfers

[1]The contracts can be found at https://github.com/reinali07/tokenization

involving the pool. Then, for each shareholder, the issuer determines the shareholder's balance of the liquidity pool's token, this time querying the liquidity pool token's `Transfer` events. Finally, the amount of Stock Tokens attributed to each shareholder is their proportion of liquidity pool tokens, multiplied by the pool's Stock Token balance. This procedure is summarized in Algorithm 1.

Algorithm 1 Procedure for calculating attributed Stock Token for uniform liquidity pools.

$B \leftarrow$ pool's Stock Token balance
for each shareholder, i **do**
    $L_i \leftarrow i$'s liquidity pool token balance
end for
$L \leftarrow \sum_i L_i$
for each shareholder, i **do**
    result$_i \leftarrow \frac{L_i}{L} B$
end for

2) CLAMMs: As in the previous case, the pool's Stock Token balance at the cutoff block is first calculated. However, calculating shareholder pool proportions is more complicated in CLAMMs. In CLAMMs, each position may have a different rate of conversion from liquidity to tokens, depending on the position's tick interval. Thus, dealing with CLAMMs requires more exchange-specific information than the previous case, which was easily generalized. First, the issuer must include in its whitelist the address of the NFT position manager associated with each CLAMM, as well as the factory contract (wherein one can lookup pools by token pairs and specified fee). Next, we require knowledge of the pool's invariant functions, g, h in Equations (1) and (2). Finally, we require that shareholders report, before the cutoff block, the liquidity pools they contribute to and the corresponding `tokenId` of any concentrated liquidity positions they hold.

The issuer first confirms the ownership of each on-chain shareholder's reported NFTs by querying the NFT contract's `Transfer` events, which are emitted upon transfer of any ERC721-compatible NFT, and confirming that the net incoming and outgoing transfers of the token to the shareholder is 1. The issuer then calls the NFT contract's `positions()` method to determine the tick interval of the position. The `positions()` method also returns the `token0` and `token1` address, and the pool fees, which allows the issuer to call the Pool Factory contract's `getPool()` method to confirm the NFT corresponds to the reported pool. It also allows the issuer to determine whether the Stock Token is `token0` or `token1` in the pool. Note that calling the `positions()` method requires that the shareholder does not burn their NFT before the accounting is completed. This constraint does not affect the shareholder, as positions can be closed out without burning their associated NFT. Then, the issuer determines the NFT's liquidity by querying the NFT contract's `IncreaseLiquidity` and `DecreaseLiquidity` events for the `tokenId`, which are emitted upon liquidity position updates, and netting these updates.

Next, the issuer determines the pool's tick, i_c, at the cutoff block. To do this, it will query the Pool contract's `Swap` events to find the last swap prior to the cutoff block, which contains the tick after the swap. Then, for each NFT, the issuer uses Equation (1) or Equation (2) (if `token0` or `token1` is the Stock Token, respectively) to determine the number of Stock Tokens attributed to the on-chain shareholder. The procedure for accounting for one shareholder's concentrated liquidity position is summarized in Algorithm 2.

Algorithm 2 Procedure for calculating attributed Stock Tokens for a on-chain shareholder's concentrated liquidity position.

Confirm position NFT ownership
token0, token1, fee, $i_l, i_u \leftarrow$ NFTContract.positions(nft)
Check pool == PoolFactory.getPool(token0,token1,fee)
$L \leftarrow$ position liquidity
$i_c \leftarrow$ tick before cutoff
if Stock Token is token0 **then**
    result $\leftarrow \max(h(\max(p(i_c), p(i_l), L) - h(p(i_u), L), 0)$
else if Stock Token is token1 **then**
    result $\leftarrow \max(g(\min(p(i_c), p(i_u)), L) - g(p(i_l), L), 0)$
end if

C. Rights Redemption Contract

On-chain shareholders are attributed the same rights as their off-chain counterparts, which they can redeem or exercise on-chain. Rights on-chain can be tokenized and thus represented by different token types. In this work, we consider three types of tokens: native tokens (e.g. Ether), ERC20 tokens (e.g. cryptotokens), and NFTs (e.g. DAO membership). A discussion on the use cases of rights represented by these token types is provided in Section V-B.

The Rights Redemption Contract allows on-chain shareholders to redeem the rights they are owed after their share ownership is accounted off-chain by the issuer (Section III-B). The contract extends the OpenZeppelin EIP-712 contract [23] and the data structure being signed depends on the type of token being distributed. The base Rights Redemption Contract allows shareholders to redeem tokens by providing a message with a valid ECDSA signature. The message is constructed and signed by the issuer after accounting, and distributed to on-chain shareholders off-chain on a public channel. The message data structure is shown in Figure 2, where the value is the amount of the tokenized rights the shareholder can claim. In the case of native tokens, no additional data is required. ERC20 token redemption requires messages to include the address of the ERC20 token in which the right is denominated. For NFT-denominated rights, the message can also include additional structured data.

The base contract is depicted in Figure 3. It keeps track of messages that have been redeemed by mapping unique message IDs to their redemption status to ensure that a message cannot be redeemed more than once. This replaces the single incrementing nonce typically used in ECDSA contracts as message IDs allow the issuer to issue indefinitely many messages at once that may be redeemed in any order. When

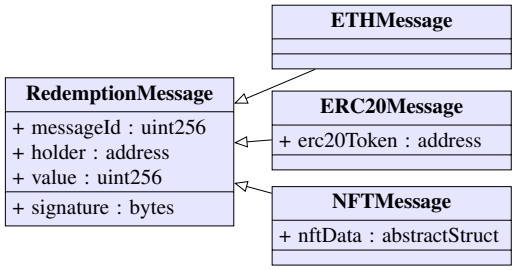


Fig. 2. Signed redemption message data structures for Ether, ERC-20 tokens, and NFTs.

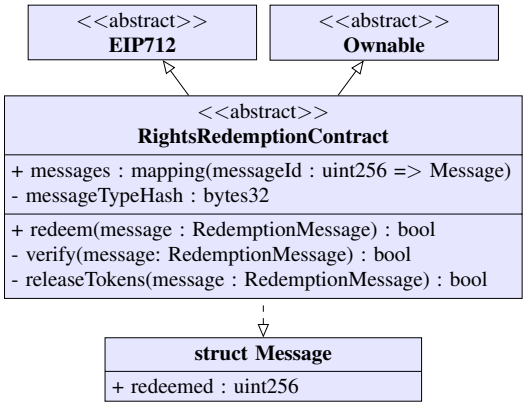


Fig. 3. Base Rights Redemption Contract class

a message is being redeemed, the contract first checks that the message's ID has not yet been redeemed. It then uses the message and the signature to recover the signer address and verifies that it belongs to the issuer. This recovery procedure also confirms the correct redeemer and signature domain. Upon successful signature verification, the contract transfers the recipient the tokens specified by the message. The implementation of the `redeem()` and `verify()` methods are depicted in Figure 4.

1) Native Tokens and ERC20 Token: When shareholder rights are distributed in the form of native tokens or ERC20 tokens, there is no additional functionality required for the Rights Redemption Contract. However, for native tokens, the contract must include an empty `receive()` function to receive native tokens so that it can eventually distribute them. The `messageTypeHash` for these contracts are `keccak256("Redeem(uint256 messageId,address holder,uint256 value)")` and `keccak256("Redeem(uint256 messageId,address holder,uint256 value,address erc20Token)")`, for native tokens and ERC20 tokens respectively.

2) NFTs: When shareholder rights are distributed in the form of NFTs, the Rights Redemption Contract also extends the Openzeppelin ERC721 contract. Additionally, it may have data members mapping NFTs to structured data. It may also implement additional methods for the usage of the NFT. An example of such is given in Section V-B. The `messageTypeHash` for these contracts is `keccak256("Redeem(uint256 messageId,address holderAddress,uint256 value,abstractStruct nftData)")`, where the `nftData` is optional NFT data of type `abstractStruct`.

```
fn redeem(
    uint256 messageId,uint256 value,...,
    uint8 v, bytes32 r, bytes32
) external {
    verify(messageId,value,...,signature);
    releaseTokens(msg.sender,value,...);
}
fn verify(
    uint256 messageId,uint256 value,...,
    uint8 v, bytes32 r, bytes32 s
) internal {
    Message storage message =
        messages[messageId];
    require(message.redeemed == 0,"Already
        redeemed");
    message.redeemed = 1;

    bytes32 structHash =
        keccak256(abi.encode(messageTypeHash,
        messageId,msg.sender,value,...));
    bytes32 h =
        EIP712._hashTypedDataV4(structHash);

    address signer = ECDSA.recover(h,v,r,s);
    require(signer == owner,"Invalid Signature");
}
```

Fig. 4. RightsRedemptionContract's `redeem()` and `verify()` methods.

IV. Evaluation

The performance of the proposed solution is evaluated against the four metrics introduced in Section III: gas cost, the ability to generalise to stocks with arbitrary holding rights, the ability to operate seamlessly with general DeFi protocols without disruption, and the ability to adapt to arbitrary accounting methods. We also measure the time required to run the accounting procedure. Furthermore, a comparison against these metrics with other stock tokenization solutions presented in [8], [18], and Section II-E is presented.

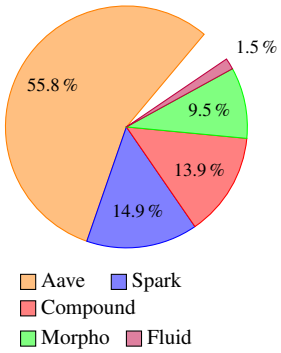
A. Performance

Gas cost: The gas cost of the proposed solution is realized in the on-chain components: the Stock Token Contract and the Rights Redemption Contract. The gas costs are measured by deploying the contracts on a local Ethereum network. The Stock Token Contract is an extension of ERC20 and the gas costs associated with calling its methods are equal to the standard ownable and burnable ERC20 contract. Deployment of the Stock Token Contract costs 400k more gas as it implements one extra function and two extra data members.

Table I summarizes the gas cost for the Rights Redemption Contract's deployment, redeeming a message, and reverted redemption attempts. The order of operations is checking (and updating) whether a message has already been redeemed, verifying the message signature, and finally, releasing tokens. Based on this order, Table I indicates, roughly, that checking and updating whether a message has been redeemed costs 26-27k gas, verifying a signature costs 28k gas, and releasing tokens takes 9k, 18k, and 62k gas for ETH, ERC20, and NFTs respectively. In comparison, transferring ETH costs 21k gas,

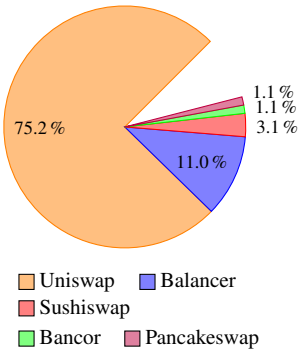
TABLE I
GAS USED FOR TOKEN REDEMPTION

Contract Action	Token Type		
	ETH	*ERC20*	*NFT*
Deployment	1,187k	1,168k	2,735k
Redeem	63k	73k	116k
Failed: already redeemed	26k	27k	26k
Failed: invalid signature	54k	55k	54k



Aave Spark
Compound
Morpho Fluid

Fig. 5. Compatible lending protocols by TVL, 95.6% of total.



Uniswap Balancer
Sushiswap
Bancor Pancakeswap

Fig. 6. Compatible AMMs by TVL, 91.5% of total.

transferring standard ERC20 tokens costs 30k gas, and minting standard mintable ERC721 NFTs costs 57k.

Generalizing to arbitrary holding rights: The proposed solution is able to handle any holding rights that can be represented as ETH, ERC20 tokens, or NFTs. This includes rights such as dividend payments and voting in shareholder meetings, which are discussed in Section V-B. Moreover, it is possible to extend the Rights Redemption Contract to accommodate rights that come in other forms, as long as they can be represented as on-chain tokens that may be transferred or minted by the Rights Redemption Contract to the shareholder.

Operability with DeFi: The Stock Token Contract implements the ERC20 API, making it operable with DeFi protocols using ERC20 tokens. The accounting procedure is also flexible in its operability as the off-chain nature allows it to be upgraded to maintain compatibility with potential new DeFi liquidity pool logic. As it is, the procedure is compatible with the top DeFi protocols. The top five DeFi protocols on Ethereum by total volume locked (TVL) in the lending category and the AMM category are shown in Figures 5 and 6, respectively [40], [41]. These protocols are compatible with the accounting procedure as they are lending pools, uniform liquidity AMMs, or CLAMMs. They make up 95.6% of lending pools and 91.5% of AMMs by TVL. These are not exhaustive in terms of compatibility—for instance, other protocols forked from Uniswap are also compatible due to the liquidity pool logic.

Adapting to arbitrary accounting methods: The off-chain nature of the accounting procedure allows it to adapt to accommodate evolving regulations or decisions regarding the ownership accounting of tokenized stocks. For instance, if it is decided that on-chain shareholders do not own any tokens they contribute to liquidity pools, the accounting procedure can simply skip the accounting for liquidity pools.

TABLE II
ACCOUNTING PROCEDURE RUNNING TIME

	Wallet balances	Uniswap V2	Uniswap V3
Number of users	380k[1]	3k[2]	1k[3]
Time per user	0.04 ms	5.05 ms	4 s[4]
Total time	14 s[5]	16 s[6]	4000 s

[1] From https://etherscan.io/token/0x1f9840a85d5af5bf1d1762f925bdaddc4201f984 (accessed May 11, 2024).
[2] From https://etherscan.io/token/0xd3d2E2692501A5c9Ca623199D38826e513033a17 (accessed May 11, 2024).
[3] From https://etherscan.io/advanced-filter?eladd=0x1d42064Fc4Beb5F8aAF85F4617AE8b3b5B8Bd801&eltpc=0x7a53080ba414158be7ec69b987b5fb7d07dee101fe85488f0853ae16239d0bde&mtd=0x88316456%7eMint&age=2021-01-01%7e2024-05-11.
[4] From https://dune.com/queries/3715137.
[5] From https://dune.com/queries/3713463.
[6] From https://dune.com/queries/3708748.

Running time: The time required to perform accounting, taking the Uniswap token (UNI) as the share token, is summarized in Table II. The running time is measured as the time to retrieve the necessary event logs to account for the wallet balances of the holders of UNI, the liquidity providers of a Uniswap V2 UNI/ETH pool, and the liquidity positions of a Uniswap V3 UNI/ETH pool. The log queries are done using Dune SQL's free tier. For dividend payments, the record date, the date at which the shareholders' holdings are recorded for payment, is typically several weeks prior to the actual payment [6]. For shareholder voting, the record date is on or before the day that notice of a meeting is issued, which typically occurs several weeks prior to the meeting [42], [43]. As the accounting for each pool takes less than two hours in the most complex case, there is more than enough time to complete accounting.

B. Comparison

The proposed solution is compared with three other solutions against the metrics discussed. These solutions are: **Automatic dividend distribution** [18], where a smart contract sends dividends to each holding address; **Homemade dividends** where dividends can be created by holders by swapping portions of their shares on-chain for cash; and the work on dividend-paying tokenized asset by **Zhitomirskiy et al.** [8], where holders redeem dividends through a contract that calculates their dividend payment based on the amount of tokenized stock they are holding and the amount of time for which they held them. Specifically, the solutions are qualitatively assessed in terms of their ability to generalize to arbitrary holding rights, operatibility with DeFi, and adaptability to arbitrary accounting methods. The gas costs for each token holder for receiving dividend payments is also measured based on 380k holders, the number of holders of UNI.

The results are summarized in Table III, which shows that no other solution performs adequately against the metrics. These solutions are fit only for paying dividends, and cannot accommodate other rights, such as voting. Crucially, even

TABLE III
COMPARISON OF SOLUTIONS

	Automatic dividends	Homemade dividends	Zhitomirskiy et al.	Our solution
Arbitrary holding rights	✗	✗	✗	✓
DeFi operability	✗	✓	✗	✓
Adaptable accounting	✗	✗	✗	✓
Gas cost per shareholder	✗[1]	127k[2]	100k	73k

[1] Automatic dividend distribution for 380k holders costs >100B gas, which exceeds the Ethereum block gas limit of 30M [44], making the distribution transaction fail.
[2] Average gas used for Uniswap V3's exact input single swap method from https://dune.com/queries/3694112.

for the payment of dividends, the alternate solutions do not make considerations for tokens held in DeFi protocols and require on-chain shareholders to be holding their tokens in order to receive dividends. The exception is the homemade dividends solution, where the token values scale regardless of where they are on-chain. Furthermore, no solution is able to adapt to other accounting methods, as the accounting is either encoded into smart contracts, as is the case for the automatic dividend distribution solution and Zhitomirskiy et al.'s [8] solution, or is encoded in the token value scaling in the case of the homemade dividends solution. Despite failing against the metrics, these alternate solutions are also more costly, with Zhitomirskiy et al's solution being the cheapest one at 27k more gas per shareholder compared to our solution.

V. DISCUSSION & USE CASES

This section presents a discussion on the trust assumptions related to off-chaining calculations as well as details how the proposed solution is applied in three use cases: dividend payments, shareholder voting, and mergers/acquisitions.

A. Trust Assumptions

In general, tokenization schemes require a high level of trust in the issuer, as the issuer has custody of the physical shares and is responsible for relaying holding rights. As such, issuers should be strictly regulated. Thus, as long as the off-chain accounting results can be verified by other parties, the proposed solution does not require stronger trust assumptions compared to other tokenization schemes, such as those examined in Section IV-B. In the proposed solution, an individual shareholder can verify the issuer's accounting results by simply redoing the calculation. This is possible as the procedure for one's accounting requires only publicly available blockchain data and knowledge of one's own CLAMM positions. Similarly, third parties can also verify the calculations when shareholders provide their CLAMM positions.

B. Use Cases

Dividend Payments: For a dividend-paying stock, the issuer deploys, along with the Stock Token Contract, a dividend Rights Redemption Contract, in which the issuer will deposit the dividends as ETH or ERC20 tokens to be redeemed by on-chain shareholders.

1) The company declares a future dividend payment.
2) The issuer notifies on-chain shareholders of the accounting cutoff, e.g. the first block after the record date.
3) On-chain shareholders report, off-chain, a list of pools and NFTs, as in Section III-B, prior to the cutoff.
4) After the cutoff, the issuer accounts for each on-chain shareholder using the procedure in Section III-B, multiplying the result by the dividends per share.
5) The issuer writes, signs, and sends (off-chain) to each shareholder a message with the number of dividends to pay and the shareholder's address. If the dividend is paid in ERC20 tokens, the message also contains the address of that ERC20 token.
6) The shareholder passes the message to the dividend Rights Redemption Contract to claim their dividends.

```
fn castVote(uint256 tokenId, uint256 value, uint8
     choice) public {
    VotingNFT storage vote = tokenData[tokenId];
    require(owners[tokenId] == msg.sender,
        "Unauthorized voter");
    require(value <= vote.value,"Not enough
        votes");

    vote.value = vote.value - value;
    emit Voted(vote.voteId, choice, value);
}
```

Fig. 7. Implementation of simple vote casting for shareholder voting.

Shareholder Voting: For common stock, the issuer deploys, along with the Stock Token Contract, a voting Rights Redemption Contract, which mints voting NFTs to on-chain shareholders and acts as a general purpose voting contract.

1) The company announces a shareholder's meeting.
2) The issuer notifies on-chain shareholders of the accounting cutoff, e.g. the first block after the record date.
3) On-chain shareholders report a list of pools and NFTs (Section III-B) prior to the cutoff.
4) After the cutoff, the issuer performs accounting (Section III-B).
5) The issuer writes, signs, and sends (off-chain) to each shareholder a message with the number of votes, the shareholder's address, and a vote ID specifying the vote the shareholder may partake in.
6) The shareholder passes the message to the voting Rights Redemption Contract and is issued an NFT containing the vote ID and the number of votes, which they can cast to the same contract (Figure 7), emitting an event with the vote ID, choice, and number of votes cast.
7) After the vote deadline, the issuer queries the event logs to tally the vote. This tally can be verified by any party, including the Company holding the vote.

Mergers and Acquisitions: When a merger occurs, there are several possibilities: shareholders are paid out for their shares, shareholders exchange their shares of the old stock with shares of the new stock, or a mix of the two [45]. If shareholders exchange their shares, the issuer's physical holdings will be

swapped. The issuer will then invoke the `rename()` function of the Stock Token Contract, updating the `stockname` and `stocksymbol` in Figure 1. However, this information may not be updated on blockchain explorers, such as Etherscan. To avoid confusion, the Stock Token Contract's `name` and `symbol` can be initially set to a unique identifier instead of the stock's actual name and symbol. If, instead, shareholders are paid out, on-chain shareholders send their tokens to a burning contract, which transfers them ETH or ERC20 tokens for every Stock Token it burns. In the case of a mixture of swapping and payout, the Stock Token can be renamed and the dividend payment process takes place to execute the payout.

VI. Related Work

The tokenization of securities has been explored both in the literature and in the financial industry. The literature to date focuses on regulatory compliance and dividend distribution, but provides inadequate consideration for the feasibility of dividend distribution in terms of gas costs, the accounting of tokens held in DeFi protocol liquidity pools, and the potential for other types of holding rights. The industry projects focus on regulatory compliance and cost efficiency, but have the same shortcomings as the works in the literature. Furthermore, these industry projects are opaque and provide little detail on technical implementation or empirical results.

A. Literature Review

In 2020, [18] proposed a blockchain-based solution for tokenizing revenue-generating real estate. The distribution of revenue to the real estate token holders is executed by a smart contract that calculates the proportion of tokens held by each address and transfers the dividends to each token holder. This is an expensive operation for the operator of the smart contract, as the number of transfers scales with the number of token holders. Furthermore, transferring dividends to indefinitely many token holders may cause the distribution transaction to exceed the Ethereum block gas limit and fail. The ERC1400 Security Token and the ERC3643 T-REX - Token for Regulated EXchanges standards were created in 2018 and 2021, respectively, to provide Ethereum token APIs for the regulatory compliance of security tokens by requiring identity management and/or compliant transfers [16], [17]. These token standards are designed to address the legal requirements of securities tokens, and do not consider functionality such as the distribution of holding rights. In 2023, [8] described an asset tokenization system for dividend-paying assets, where dividend payments are calculated based on the amount of time the token spends in the holder's wallet. In this work, assets with holding rights other than dividends were not considered and paying dividends to liquidity providers was left as a challenge for future adoptions.

B. Industry Projects

Several projects for the tokenization of securities or other real-world assets helmed by major banks, securities trading firms, and regulatory bodies are in the works. A collaboration named Project Guardian between policymakers UK's FCA, Switzerland's FINMA, Japan's FSA, and Singapore's MAS and industry groups including JP Morgan, Citibank, and HSBC, aims to develop and pilot the policies and technologies necessary for secure, interoperable, and innovative asset tokenization [46]. The project's focus thus far has been on identity and trust management, regulatory innovation, payment automation, and portfolio management [47]. It wrapped up its first pilot program for the trading of tokenized bonds and foreign exchange against permissioned liquidity pools in 2022 [15]. Swiss securities firm Taurus provides a platform for the tokenization of assets, and was approved by the FINMA to offer tokenized securities of unlisted Swiss firms to retail investors in January 2024 [9]. Private Swiss bank Cité Gestion, along with a handful of other firms used the Taurus platform to tokenize its own shares to be available to professional investors in 2023 [10]. In 2022, Hamilton Lane announced it would be tokenizing three of its funds using Securitize, another firm offering asset tokenizaton services [11]. In 2022, UBS launched a digital bond that is tradeable on the blockchain, as well as in traditional exchanges [12]. The BIS, along with the Swiss National Bank and the World Bank announced in 2024 Project Promissa, a pilot initiative for the tokenization of promissory notes [13]. HSBC, partnering with Goldman Sachs and the EIB, has launched in 2023 a digital bond that uses the blockchain as a record of ownership [14]. These projects are limited in scope, as they focus on specific types of assets, such as bonds and notes. Moreover, while they aim for efficiency gains through blockchain's accessibility and automation, they do not consider the potential arising from inter-operability with DeFi, as they either require limitations on compatible DeFi protocols or do not address DeFi compatibility at all.

VII. Conclusion

This paper presents a stock tokenization solution that is gas-efficient, generalizes to arbitrary holding rights, operates with DeFi without friction, and can be adapted to arbitrary accounting methods or decisions. We show that these requirements are not adequately addressed by existing solutions. Our solution comprises three components: a Stock Token Contract, the Off-Chain Accounting procedure, and a Rights Redemption Contract. The Stock Token Contract is operable with DeFi by design and the Off-Chain Accounting works for lending pools, uniform liquidity AMMs, and CLAMMs as it is designed around their liquidity provision and pool ownership logic. Specifically, the accounting procedure is able to account for at least 90% of lending pools and AMMs, by TVL. We demonstrate that our solution can accommodate stocks with any rights that can be represented by ETH, ERC20 tokens, or NFTs, and give three explicit use cases: dividend payments, voting rights, and mergers/acquisitions.

In future work, further gas optimizations in the Rights Redemption Contract should be performed to further lower the cost of redeeming rights. Currently, reading and writing the message redemption status are expensive operations that could be improved by, for instance, using different data types or structures. Moreover, the time required to run the accounting procedure could be reduced by using other methods

to query event logs. Furthermore, while we have shown the representation of voting rights as tokens, the peripheral utilities required for other types of rights can be explored. Finally, for real-world implementation, our solution should incorporate regulatory compliance solutions, which are discussed in the related works [8], [16], [17]. These works can be adopted by our solution, as they can be realized as modifications or additions to the Stock Token Contract.

REFERENCES

[1] *Top DeFi Tokens by Market Capitalization*, CoinMarketCap, Feb. 21, 2024. [Online]. Available: https://coinmarketcap.com/view/defi/

[2] "Investor Bulletin: American Depositary Receipts," SEC Office of Investor Education and Advocacy, Aug. 2012. [Online]. Available: https://www.sec.gov/investor/alerts/adr-bulletin.pdf

[3] H. Adams, N. Zinsmeister, and D. Robinson, "Uniswap v2 Core," Uniswap. Mar. 2020. [Online]. Available: https://uniswap.org/whitepaper.pdf

[4] K. Malinova and A. Park, "Learning from DEFI: Would automated market makers improve equity trading?," Nov. 2023. [Online]. Available: https://ssrn.com/abstract=4531670

[5] J. B. Berk and P. M. DeMarzo, "The Corporation and Financial Markets," in *Corporate Finance*, Fifth Global Edition., Pearson, 2019, ch. 1, sec. 1, pp. 32–55.

[6] J. B. Berk and P. M. DeMarzo, "Payout Policy," in *Corporate Finance*, Fifth Global Edition., Pearson, 2019, ch. 17, sec. 5, pp. 635–675.

[7] K. Malinova and A. Park, "Tokenized Stocks for Trading and Capital Raising," Feb. 8, 2023. [Online]. Available: https://ssrn.com/abstract=4365241

[8] E. Zhitomirskiy, S. Schmid, and M. Walther, "Tokenizing assets with dividend payouts—a legally compliant and flexible design," *Digital Finance*, vol. 5, no. 3, pp. 563—580, Dec. 2023, doi: 10.1007/s42521-023-00094-w.

[9] I. Allison. "Crypto Custody Specialist Taurus Brings Tokenized Securities to Retail Customers in Switzerland." CoinDesk. https://www.coindesk.com/business/2024/01/23/crypto-custody-specialist-taurus-brings-tokenized-securities-to-retail-customers-in-switzerland/ (accessed Apr. 22, 2024).

[10] A. Sanon-Jules. "Swiss Bank Cité Gestion Becomes First Private Bank to Tokenize Its Own Shares." CoinDesk. https://www.coindesk.com/business/2023/01/24/swiss-bank-cite-gestion-becomes-first-private-bank-to-tokenize-its-own-shares/ (accessed Apr. 22, 2024).

[11] J. Crawley. "Investment Manager Hamilton Lane to Tokenize 3 Funds Through Securitize." CoinDesk. https://www.coindesk.com/business/2022/10/05/investment-manager-hamilton-lane-to-tokenize-3-funds-through-securitize/ (accessed Apr. 22, 2024).

[12] "UBS AG launches the world's first ever digital bond that is publicly traded and settled on both blockchain-based and traditional exchanges." UBS Global. https://www.ubs.com/global/en/media/display-page-ndp/en-20221103-digital-bond.html (accessed Apr. 22, 2024).

[13] "BIS Innovation Hub, Swiss National Bank and World Bank launch Project Promissa to test tokenisation of financial instruments." Bank of International Settlements Innovation Hub. https://www.bis.org/about/bisih/topics/fmis/promissa.htm (accessed Apr. 22, 2024).

[14] "Hong Kong confirms first $100m tokenized green bond." Ledger Insights. https://www.ledgerinsights.com/hong-kong-tokenized-green-bond/ (accessed Apr. 22, 2024).

[15] L. Alan *et al.*, "Project Guardian: Enabling Open and Interoperable Networks," Monetary Authority of Singapore and Bank of International Settlements, June 2023. [Online]. Available: https://www.mas.gov.sg/-/media/mas-media-library/development/fintech/project-guardian/project-guardian-open-interoperable-network.pdf

[16] *Security Token Standard*, ERC-1400, A. Dossa, P. Ruiz, F. Vogelsteller and S. Gosselin, Sep. 9, 2018. [Online]. Available: https://github.com/ethereum/EIPs/issues/1411

[17] *T-REX - Token for Regulated EXchanges*, ERC-3643, J. Lebrun, T. Malghem, K. Thizy, L. Falempin, and A. Boudjemaa, Jul. 9, 2021. [Online]. Available: https://eips.ethereum.org/EIPS/eip-3643

[18] A. Gupta, J. Rathod, D. Patel, J. Bothra, S. Shanbhag, and T. Bhalerao, "Tokenization of Real Estate Using Blockchain Technology," in *Applied Cryptography and Network Security Workshops*, 2020, pp. 77—90, doi: 10.1007/978-3-030-61638-0_5.

[19] *Token Standard*, ERC-20, F. Vogelsteller and V. Buterin, Nov. 19, 2015. [Online]. Available: https://eips.ethereum.org/EIPS/eip-20

[20] *Non-Fungible Token Standard*, ERC-721, W. Entriken, D. Shirley, J. Evans, and N. Sachs, Jan. 24, 2018. [Online]. Available: https://eips.ethereum.org/EIPS/eip-721

[21] S. Nakov, "Digital Signatures," in *Practical Cryptography for Developers,* Nov. 2018. [Online]. Available: https://cryptobook.nakov.com/

[22] Ethereum.org. *Ethereum Glossary: ECDSA* (2024). [Online]. Available: https://ethereum.org/en/glossary/#ecdsa

[23] OpenZeppelin. *Contracts - OpenZeppelin Docs*. [Online]. Available: https://docs.openzeppelin.com/contracts/

[24] *Typed structured data hashing and signing*, EIP-712, R. Bloemen, L. Logvinov, and J. Evans, Sep. 12, 2017. [Online]. Available: https://eips.ethereum.org/EIPS/eip-712

[25] Avara. *Aave v2 Developers Documentaton: aTokens*. [Online]. Available: https://docs.aave.com/developers/v/2.0/the-core-protocol/atokens

[26] Avara. *Aave v3 Developers Documentation: AToken*. [Online]. Available: https://docs.aave.com/developers/tokens/atoken

[27] MakerDAO. *Spark Developer Documentation*. [Online]. Available: https://devs.spark.fi/

[28] P. Frambot, M. G. Delaunay, V. Danos, A. Husson, and K. Babbar, "Morpho Optimizer: Optimizing Decentralized Liquidity Protocols," Morpho, Apr. 2022. [Online]. Available: https://whitepaper.morpho.xyz/

[29] Compound Labs, Inc.. *Compound V2 Docs: cTokens*. [Online]. Available: https://compound.finance/docs/ctokens

[30] Compound Labs, Inc.. *Compound III Docs: Collateral & Borrowing*. [Online]. Available: https://docs.compound.finance/collateral-and-borrowing/

[31] Instadapp. *Fluid Contracts overview*. [Online]. Available: https://docs.fluid.instadapp.io/introduction/contracts-overview

[32] Y. Zhang, X. Chen, and D. Park, "Formal Specification of Constant Product (x y = k) Market Maker Model and Implementation," Runtime Verification, Inc. Oct. 24, 2018. [Online]. Available: https://github.com/runtimeverification/verified-smart-contracts/blob/master/uniswap/x-y-k.pdf

[33] Sushi. *SushiSwap Docs*. [Online]. Available: https://docs.sushi.com/

[34] PancakeSwap. *PancakeSwap Developer Docs*. [Online]. Available: https://developer.pancakeswap.finance/

[35] Balancer. *Balancer Docs*. [Online]. Available: https://docs.balancer.fi/

[36] Bancor. *Bancor Network V2.1*. [Online]. Available: https://bancor-network.gitbook.io/v2.1/master

[37] Bancor. *Banor V3 Technical Docs*. [Online]. Available: https://docs.bancor.network/

[38] H. Adams, N. Zinmeister, M. Salem, R. Keefer, and D. Robinson, "Uniswap v3 Core," Uniswap. Mar. 2021. [Online]. Available: https://uniswap.org/whitepaper-v3.pdf

[39] Uniswap. *Uniswap V3 Protocol Technical Reference*. [Online]. Available: https://docs.uniswap.org/contracts/v3/reference/overview

[40] "Lending TVL Rankings," DefiLlama. [Online]. Available: https://defillama.com/protocols/Lending/Ethereum (accessed May 1, 2024).

[41] "Dexes TVL Rankings," DefiLlama. [Online]. Available: https://defillama.com/protocols/Dexes/Ethereum (accessed May 1, 2024).

[42] "Share structure and shareholders." Government of Canada Innovation, Science and Economic Development. https://ised-isde.canada.ca/site/corporations-canada/en/business-corporations/share-structure-and-shareholders (accessed May. 10, 2024).

[43] *Companies act 2006: Notice of meetings.* UK Public General Acts 2006, ch. 46, sec. 13, pp. 144–146.

[44] Ethereum.org. *Ethereum Development Documentation: Gas and fees* (2024). [Online]. Available: https://ethereum.org/en/developers/docs/gas/

[45] J. B. Berk and P. M. DeMarzo, "Mergers and Acquisitions", in *Corporate Finance*, Fifth Global Edition, Pearson, 2019, ch. 28, sec. 10, pp. 1000–1030.

[46] "Project Guardian." Monetary Authority of Singapore. https://www.mas.gov.sg/schemes-and-initiatives/project-guardian (accessed Apr. 22, 2024).

[47] T. Lobban *et al.*, "The Future of Wealth Management: Ultra-efficient portfolios of traditional and alternative investments powered by tokenization," J.P. Morgan Chase & Co. and Apollo Global Management, Inc., 2023. [Online]. Available: https://www.jpmorgan.com/onyx/documents/portfolio-management-powered-by-tokenization.pdf