FCToken: A Flexible Framework for Blockchain-Based Compliance Tokenization

Hao Tan*
State Key Lab of CAD&CG
Zhejiang University
Hangzhou, China
tttat@zju.edu.cn

Shuangzhou Yan*
State Key Lab of CAD&CG
Zhejiang University
Hangzhou, China
ysz22@zju.edu.cn

Xin Zou
State Key Lab of CAD&CG
Zhejiang University
Hangzhou, China
22160282@zju.edu.cn

Guanghuan Xie
Global Technology Service Department
State Street Technology Ltd.
Hangzhou, China
looper@zju.edu.cn

Hongxin Zhang†
State Key Lab of CAD&CG
Zhejiang University
Hangzhou, China
zhx@zju.edu.cn

Zhuo Li†
Global Technology Service Department
State Street Technology Ltd.
Hangzhou, China
lizhuo@zju.edu.cn

***Abstract*—Tokenization has emerged as a pivotal topic in fintech. It involves converting indivisible assets into divisible tokens, streamlining their trade and management. While asset tokenization offers benefits like reduced entry barriers, faster transaction settlements, and improved asset liquidity, it also presents challenges, notably concerns like money laundering and terrorism financing. The current landscape lacks comprehensive regulatory measures, allowing unrestricted access for both token issuers and buyers. To address the aforementioned challenges, this paper introduces the FCToken framework, meticulously designed to facilitate compliance tokenization across diverse scenarios and asset types. Innovatively, the framework operates at both the Token and Identity levels, encompassing four core modules to actualize compliance token issuance. Leveraging this framework, we have devised a prototype system, integrating both a GUI low-code module and a gas fee prediction mechanism. Such integrations not only bolster compliance-centric development efficiency but also proffer users methodologies to curtail gas expenditures during the token issuance process.**

***Index Terms*—Blockchain, tokenization, compliance, low-code, gas fee**

I. Introduction

Blockchain technology has revolutionized various sectors by offering unparalleled transparency, security, and decentralization. One of its most impactful applications has been the process of tokenization, which involves converting traditional financial assets into tokens that exist on a blockchain. Tokenization serves as a modern form of securitization, essentially evolving the concept of asset securitization by transforming specific rights tied to tangible assets, such as ownership, governance, and recourse, into a digital format on the blockchain.

Tokenization is rapidly gaining traction as one of the most promising applications in the blockchain ecosystem, evidenced by its adoption by major enterprises such as Deloitte, Bank of New York Mellon, and Ernst & Young. These organizations foresee its disruptive potential across various massive industries, including the $9 trillion global securities market and the $9.6 trillion global real estate industry. Furthermore, companies like Microsoft, Vanguard, and Sotheby's have initiated asset tokenization programs to issue tokens representing industrial assets, securities, and real estate, respectively.

The advantages of tokenization are manifold. It increases liquidity by making traditionally illiquid assets more easily tradable. Tokenization also lowers the barriers to investment, enabling smaller investors to participate in markets previously restricted to institutional investors. Additionally, it enhances transactional efficiencies by expediting processes and increasing transparency regarding asset equity information. This stands in contrast to traditional financial systems, which often lack transparency and impose high costs for entry and transactions.

However, despite these advantages, tokenization is plagued by numerous challenges. The anonymous and open nature of blockchain technology conflicts with the rigorous Know Your Customer (KYC) and Anti-Money Laundering (AML) regulations in the financial world. The current landscape lacks a universal standard for tokenization, even though the blockchain community has developed multiple utility and security token standards like ERC-20 [1], ERC-721 [2], and ERC-1155 [3]. These existing standards fall short of capturing the complexities of various financial products, which may have diverse lifecycles and jurisdiction-specific compliance requirements.

The primary aim of this paper is to introduce FCToken, a flexible framework designed to navigate these complexities and facilitate compliance token issuance. The paper is organized as follows: Section II reviews related work on blockchain and compliance tokenization, along with legal frameworks. Section III presents the FCToken framework, discussing its fundamental concepts, legal requirements, and potential applications. Section IV delves into the interactive compliance token issuance process, featuring the roles of

* These authors contributed equally to this work and share first authorship.
† Corresponding authors.

smart contracts and compliance requirements. Additionally, a GUI low-code module is introduced to facilitate compliance scripting. Section V addresses the potential gas consumption challenges within the FCToken framework and, through empirical analysis, offers users interactive recommendations. The paper concludes by summarizing key findings, discussing limitations, and suggesting future research directions.

II. Related Work

A. Blockchain and Compliance Tokenization

Bitcoin [4]–[6] is the first purely peer-to-peer digital cryptocurrency. Blockchain, as the underlying technology of Bitcoin, is the key innovation of Bitcoin [7]. Ethereum, proposed by Wood *et al.* is a global, open-source,turing complete platform for decentralized applications [8]. The emergence of Ethereum made token issuance possible. In blockchain, token is a significant concepts.The tokenization of assets refers to the process of issuing a blockchain token that digitally represents a real tradable asset [9]. The emergence of tokenization technology has the potential to revolutionize foundational financial infrastructure [10]. Almost all real world assets can be tokenized in to token. The assets could be real estate, art, derivatives markets, non-fungible assets, commodities, services and even identity [7].

B. Relevant Legal and Compliance Frameworks

The emergence of asset tokenization has also led to numerous illicit issues, such as money laundering and financing of terrorism [10]–[12] To address these compliance issues, many experts and scholars have proposed regulatory frameworks for token issuance. Lee *et al.* proposed the CODE framework [13], can supporting the *travel rule* established by the Financial Action Task Force (FATF). The OpenVASP protocol, proposed by Riegelnig *et al.*, utilizes Ethereum to enable identity verification without the need for a third party [14]. The Policy-Backed Token (PBT) proposed by Li *et al.*, based on the Open Asset Protocol (OAP), is a policy-supported token [7]. The Polymath platform, built by Dossa *et al.*, based on Ethereum and Polygon, established the Registry-Factory-Instance pattern, enabling compliance control at the identity level [15].

C. Gas Fee Optimization Strategies in Ethereum

In the blockchain realm, gas is a ubiquitously recognized concept [8]. Gas can be conceptualized as the computational effort required to execute operations, be it a simple transfer of tokens or the more complex deployment of a smart contract.

Research on gas fees has become a prevalent topic in the blockchain domain. Numerous scholarly works in the blockchain domain have underscored the significance of optimizing gas expenditures. Typically, strategies for conserving gas fees predominantly focus on the contract code level, where the relationship with gas consumption is most intimate [16]–[20]. These methodologies aim to reduce the cost of executing contract calls by diminishing the inherent gas consumption of the contract itself. Additionally, some studies have ventured into examining gas consumption from the perspectives of computational resources and network deployment [21], [22].

The aforementioned studies, while diverse in their methodologies and focal points, converge on a shared objective: optimizing and understanding gas consumption in the blockchain domain. Chen *et al.* [17] emphasized the importance of bytecode analysis in identifying gas-intensive patterns, thereby providing actionable insights for developers at the smart contract level. Pacheco *et al.* [20], in contrast, leaned towards a statistical approach, employing a linear regression model to estimate gas fees based on historical transaction data. Their model's robustness was ensured through logarithmic transformations and validated using a sliding window technique. Meanwhile, Pierro *et al.* [23] took a more holistic view, dissecting the multifaceted dynamics that govern Ethereum transaction fees.

Collectively, these research highlighted underscores the concerted efforts of the academic community to both understand and optimize gas consumption. From bytecode analysis to statistical modeling and holistic examinations of transaction dynamics, the multifaceted approaches adopted reflect the complexity and significance of gas management within the blockchain framework.

III. Framework for Compliance Tokenization

Compliance is a crucial component of tokenization, as it ensures that all transactions are in line with relevant regulations and laws. A compliance framework is used to enforce these regulations and ensure that tokens are issued and traded in a compliance manner. Compliance framework is becoming increasingly important as more digital assets are tokenized and traded on blockchains, and regulatory scrutiny in the industry continues to increase. In this section, we will discuss the key features and Implementation of a compliance framework.

A. Fundamentals of Compliance Tokenization

The FCToken framework is designed to provide compliance tokenization on the blockchain by integrating Identity System, Compliance Engine, Exchange System, and Security Token. Identity System ensure that token holders are properly identified and verified, while compliance engine ensure that tokens are issued and traded in accordance with legal and regulatory requirements. Exchange System enable the management of security token. The system architecture is illustrated in Figure 1.The next subsections will introduce these four core structures within this framework.

1) Identity System: Compliance in the context of tokenization relies heavily on robust identity management components. The identity system integrates both on-chain and off-chain information for users. Users are required not only to provide an anonymized on-chain wallet address but also to register a new identity through the platform's identity system. This identity combines on-chain and off-chain information. Here's how the this module works:

Roles The identity system in the framework consists of three roles: Issuer, Investor, and Claimissuer. (1) **Issuer** refers to security token issuer, within the system, the issuer possesses

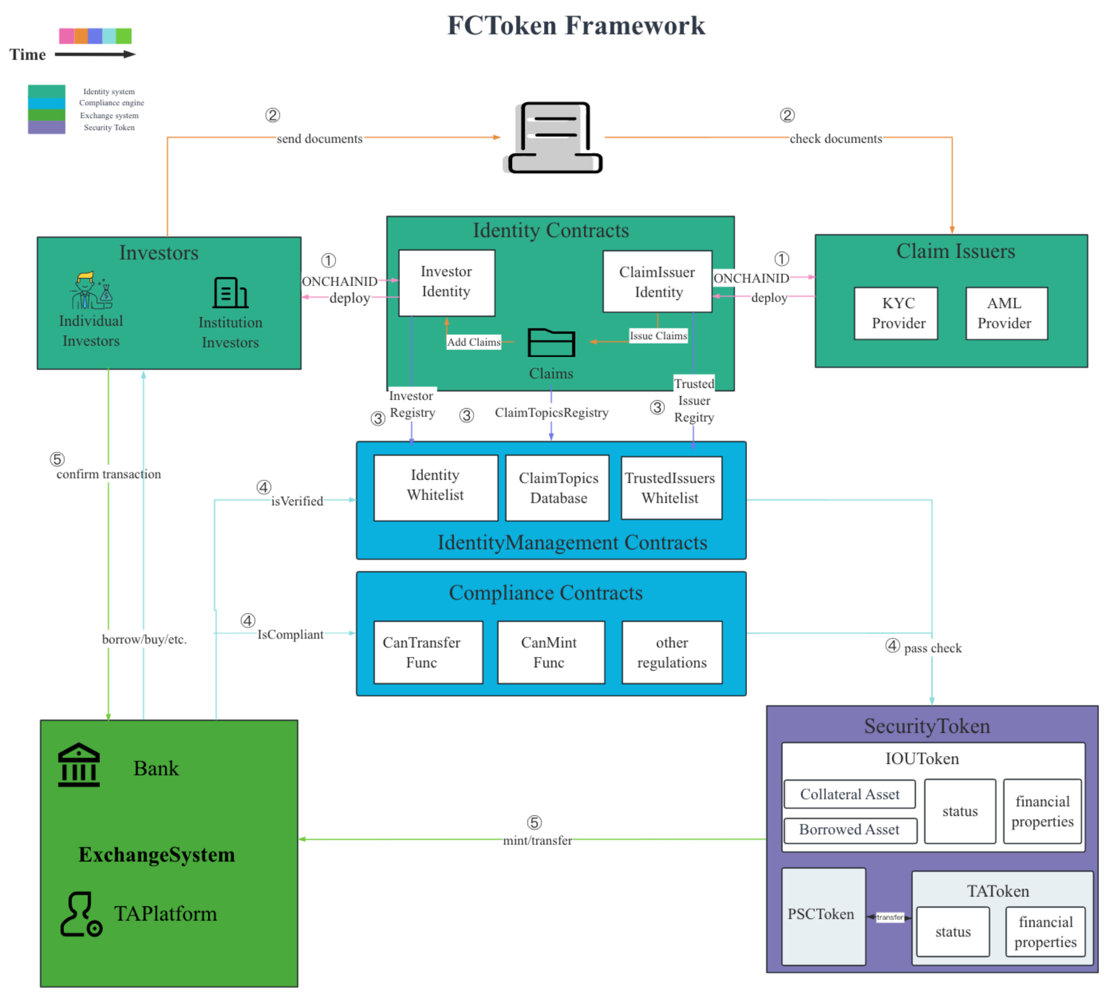


Fig. 1. FCToken implementation framework, include Identity System, Compliance Engine, Exchange System and Security token issuance. Each part is distinguishable by its color, and the color of the lines in the diagram represents the chronological order of system operations.

the highest authority. (2) **Investor** refers to the users who participate in trading activities on the platform. An investor need to connect their on-chain wallet address and provide relevant off-chain information, such as nationality, age and other real-world details that align with the platform's compliance requirements. (3) **ClaimIssuer** refers to a trusted third party in KYC(Know Your Customer)/AML(Anti-Money Laundering) qualification issuance is responsible for issuing qualifications required to verify investors. The requirements for such qualifications depend on local legal regulations and the compliance requirements of the issuer.

Besides, both Investor and ClaimIssuer are required to sign up by providing their wallet addresses. And role information is represented using Key identifiers to distinguish between organizational users and regular investors.

2) Compliance Engine: different financial products and regions have specific compliance policy requirements. To meet the universal requirements of varying compliance rules in different regions, the platform integrates a compliance engine. This engine provides support for compliance rules throughout the complete lifecycle of token assets. Therefore the compliance engine consists of two levels: the identity level and the token level.

Identity level usually refers to KYC. For each tokenization participant, at the onboarding stage, tokenization platform needs to perform due diligence and collect identify information to know the real world entity name, country, physical address etc, so that it can run through AML and CFT (Countering the Financing of Terrorism) and any other mandatory compliance screening required by applicable regulation framework. And this screening not only happens in onboarding stage, but also needs to perform on-going to catch any change like sanction list.

Token level each token may have some limitations predefined in issuance based on applicable regulation framework. Some refers to locking period limitation, like securities issued under Regulation D will be locked from trading for 6 months - 1 year. Some refers to total investor number count, then in such case, when the total investor number comes to the max threshold, the smart contract of the token will prohibit partial transfer.

For the implementation of Identity level, a set of identity management contracts is created with *IdentityManagement* at its core. As for the Token level implementation, *DefaultCompliance* is created to manage compliance rules.

3) Exchange System: the trading system is the primary application implementation within this framework, and it can have different implementation approaches in various scenarios.

The internal trading system of the platform integrates the Identity System, Compliance Engine, and security tokens. Investors access the trading system with their internal identities. Investors behaviors vary in different scenarios. For example, in the banking scenario, users primarily engage in transactions related to lending products, while in the TAPlatform (Transfer Agent Platform) scenario, users mainly participate in investments and redemptions. The Exchange System within this framework can flexibly accommodate user trading behaviors in different financial scenarios. Next we will use the Bank and TAPlatform scenarios as examples to illustrate how to achieve flexible adaptation for multiple scenarios.

Applicable Scenarios The Bank scenario is designed for the banking lending context, enabling functionalities such as borrowing, pledging, adding collateral, and forced liquidation of financial products. Highly similar to the Bank scenario, the TAPlatform scenario is used for financial product trading activities in the fund agent context, including fund issuance and investor purchases. Both the Bank scenario and TAPlatform scenario rely on the Exchange System framework, adjusting the settings of the *SecurityPool* and *CompliancePool* dynamically to accommodate different financial scenarios. By adding the respective contract addresses and rule logics, it allows for flexible support of various financial products and trading activities to meet specific needs.

4) Security Token: the system tokenizes financial assets as non-fungible currencies using the ERC-1155 token standard, which is different from the typical fungible token protocol (ERC-20). Adopting the ERC-1155 standard offers several advantages. Firstly, ERC-1155 is a more convenient and efficient non-fungible token standard that allows for the deployment of an unlimited number of NFTs(Non-Fungible Token) within a single smart contract. This significantly reduces the cost of deploying contracts. Secondly, ERC-1155 enables batch transfers of NFTs in a single transaction, leading to significant gas fee savings. Lastly, an ERC-1155 smart contract can issue both non-fungible tokens with a quantity of 1 and semi-fungible tokens with optional quantities. These tokens still possess the uniqueness of NFTs but can be owned by multiple users, increasing token liquidity.

Therefore, the system utilizes the ERC-1155 standard to tokenize financial assets as non-fungible currencies. Each ERC-1155 contract manages multiple NFT tokens that represent financial assets, reducing deployment costs. Each NFT token within the contract has a unique ID for identification purposes. Moreover, since ERC-1155 tokens can have quantities more than 1, splitting an NFT into multiple parts allows for wider circulation and lowers the barrier to entry in the token market.

B. Legal and Regulatory Requirements

A flexible and versatile compliance framework needs to support different types of tokens and compliance rules at different stages. To meet this requirement, our framework adopts a solution that manages the compliance of tokens throughout their entire lifecycle, achieving comprehensive coverage while also offering flexibility to support various types of tokens.The implementation of compliance rules during the token lifecycle is shown in Figure 2.

The lifecycle of a token involves several key events, including minting, transferring, freezing, and burning. Minting refers to the creation of new tokens. Transferring tokens enables token holders to send tokens to other addresses on the blockchain. Freezing tokens can prevent them from being transferred or spent. Burning tokens refers to the destruction of tokens, which reduces the total supply of tokens in circulation.

So compliance management can be divided into different stages based on the lifecycle of tokens.

The first stage is the preparation phase, which occurs before token issuance. In this stage, it involves utilizing the Identity System within the framework, which includes checking if the user and their blockchain address are listed on any sanctions lists or have any risk warnings. This is an Onboarding Compliance rule.This process is facilitated by a trusted third-party KYC provider registered in the system. Once the checks are completed, users can participate in activities such as purchasing tokens or transferring them.

The second stage is the issuance phase.This stage involves token minting and transferring, where tokens are created and made available for open trading. Compliance rules need to be configured before the minting process begins, such as requiring participants to meet nationality requirements in order to participate in the token minting process. After minting, a continuous trading process ensues, during which participants must adhere to compliance rules. Therefore, appropriate compliance rules should be incorporated into the transfer method, for example, requiring participants to meet capital requirements to engage in token transactions.This stage ensures that the minting and transferring processes are secure and compliant with applicable regulations.

The third stage is post-issuance monitoring, which includes token freezing and burning.Typically, this occurs when token holders or transactors engage in malicious behavior. For instance, if a user successfully passed KYC checks during the initial participation in transactions but later becomes involved in criminal activities, no longer meeting the criteria, their held tokens can be frozen, preventing any outflow from their account. Similar situations may trigger burning operations. The triggering logic for freeze and burn operations depends on the configuration of compliance rules.

C. Applications of Blockchain Technology in Compliance Tokenization

Blockchain network Any blockchain that supports a turing-complete Virtual Machine can be applicable. Currently, the most widely used is EVM (Ethereum Virtual Machine), which

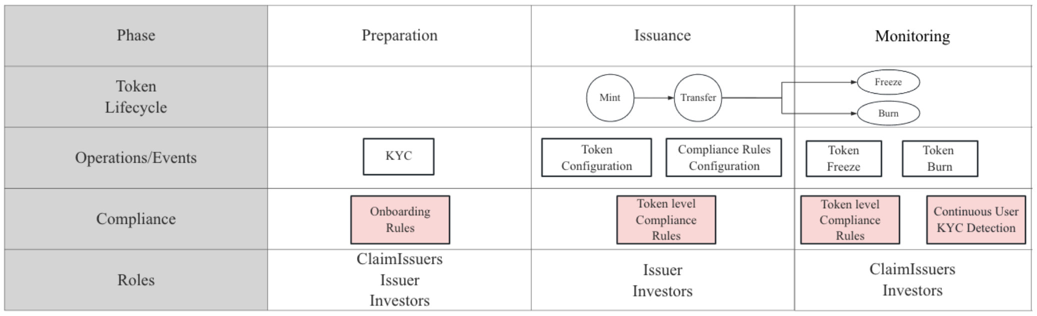


Fig. 2. Compliance rule implementation during token lifecycle. Including different stages of token issuance, various phases in the token lifecycle, key operations/events, and the implementation of compliance with the involvement of different roles.

is supported by various public chains, including the Ethereum mainnet and Ethereum second-layer networks such as Polygon, Avalanche. For cost and network performance considerations, the platform utilizes the Polygon network as the underlying blockchain network.

Smart contract program Base on the EVM, a preferred choice for smart contract language is still Solidity, which has a strong ecosystem and open-source libraries, as well as being easy to migrate. Due to the use of Polygon as an Ethereum Layer 2 network, the system has chosen Solidity as the programming language for smart contract.

User interface the frontend development for applications primarily involves TypeScript and JavaScript as the preferred programming languages. In addition to building the frontend pages, it is necessary to interact with the wallet and blockchain. The commonly used libraries for this purpose are ethers.js and web3.js.

IV. Interactive Compliance Token Issuance

The previous section introduced the framework of FCToken. This section will provide a detailed explanation of the implementation of this framework, include the implementation and role of smart contracts and KYC/AML Compliance Procedures.

A. Roles of Smart Contracts in the Issuance Process

The framework consists of four core modules, each implemented as a system composed of smart contracts. In this section, we will introduce the composition and implementation of the smart contracts in each module.

1) Identity System:

- **Identity**: The Identity smart contract is employed for identity registration of trusted third-party organizations and investors. This contract establishes a link to another smart contract named *Storage*, which records user role and qualification information.

2) Compliance Engine:

- **IdentityManagement**: This smart contract is linked to a dynamic storage contract called *IdentityRegistryWhitelist*, which stores whitelist identities. The *IdentityManagement* contract has the capability to associate a user's wallet address and the user's off-chain information such as nationality into one single contract. This contract has an *isVerified* function to determine whether a user meets the KYC requirements for a specific transaction.
- **IdentityWhitelist, TrustedIssuersWhitelist and ClaimTopicsDatabase**: These smart contracts are contracts related to information storage. *IdentityWhitelist* records all authorized registered user system Identity information. The purpose of this *IdentityWhitelist* is to separate storage functions from actual logical function so that different *IdentityManagement* contracts can share the same user whitelist information. *TrustedIssuersWhitelist* stores the identity information whitelist of a trusted third-party system for KYC/AML. *ClaimTopicsDatabase* stores all qualifications related to security token.
- **DefaultCompliance**: This smart contract is used to establish and enforce rules specific to the security token itself, ensuring compliance throughout the token's lifecycle. These rules can be set by the issuer and may include restrictions such as a maximum number of investors, minimum asset threshold for investors, or a limit on the total supply of the token. The compliance rules are recorded in the contract during deployment as a list, with each element representing a specific rule constraint.

3) Exchange System:

- **Bank and TAPlatform**: These smart contracts have established *SecurityPool* and *CompliancePool*, which allow for dynamic addition and removal of supported financial products and compliance rules. The contracts incorporate logic implementation functions that are relevant to the business, such as *borrow*, *lend*, *marginCall*, *forcedLiquidation*.

4) Security Token:

- **IOUToken and TAToken**: These two contracts are on-chain assets in the Bank and TAPlatform scenarios, both inherit the Ethereum ERC1155 standard. To meet the different properties of assets in different scenarios, the *FinancialStruct* struct was added to the contracts to set the financial attributes of the products.To accommodate different regulatory requirements of users from different jurisdictions, different ERC-1155 contracts can be associated with distinct compliance rules

B. Integration of KYC and AML Compliance Requirements

In the previous chapters, we have introduced the compliance engine. In this part, we will discuss the specific processes related to user KYC/AML and token compliance. The specific process for investor KYC/AML in the system is illustrated in Figure 3. Users are required to provide their on-chain address and off-chain real-world information, such as nationality and name. Some of this information will be provided to a KYC/AML verification provider. The provider will verify the investor's information based on what was provided during registration, and add relevant qualification claims to the investor's identity. These qualifications can include proof of not being on a sanctions list and proof of a clean criminal record and so on.Subsequently, investors who have passed these verifications will be added to the *IdentityRegistryWhitelist*.

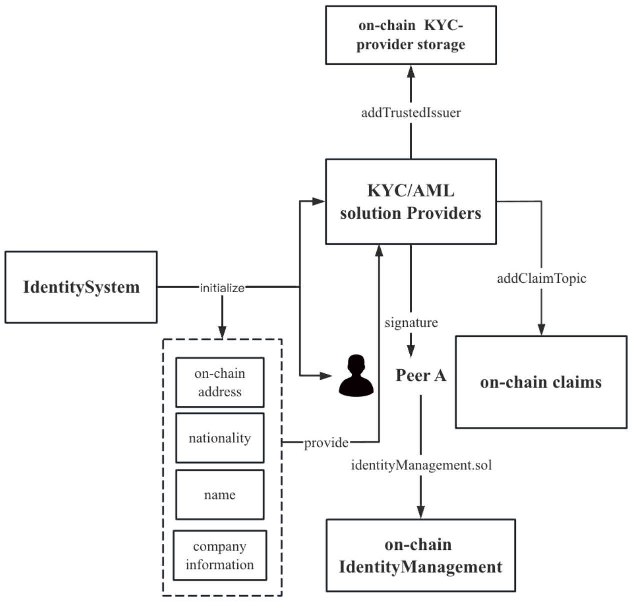


Fig. 3. KYC/AML Process. *Peer A* refers to any investor.

The implementation process of compliance rules is token-oriented and requires integration with security tokens. Taking IOUToken as an example (see Figure 4), additional functions need to be added to the compliance contract based on the IOUToken business implementation. For instance, the *canissue*function can be added to impose constraints on the issuer's capital during token issuance. The *canInitialOfferring* function can be added to specify requirements for the number of token holders. These functions aim to ensure compliance during token issuance and initial offering processes.

C. Blockly-based GUI editing

Different countries, regions, and even organizations have different compliance rules and tendencies. Therefore, the issuance template for compliance tokens needs to be universal. Writing and compiling different smart contracts will require a significant amount of work in the process of compliance token issuance. For token issuers who are not familiar with smart

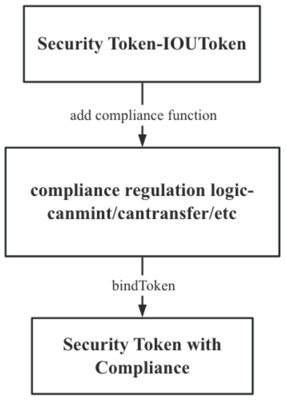
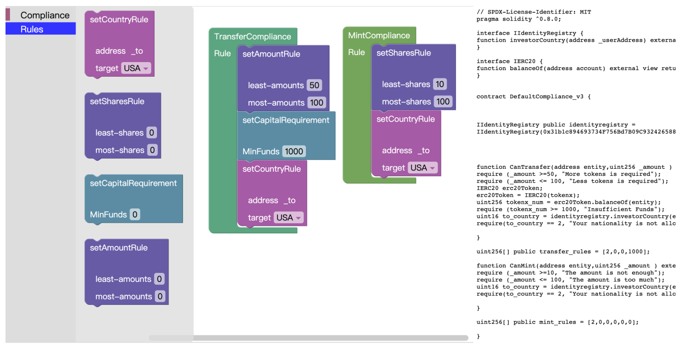

Security Token-IOUToken
add compliance function
compliance regulation logic-canmint/cantransfer/etc
bindToken
Security Token with Compliance

Fig. 4. Process of embedding compliance rules in security tokens



```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

interface IIdentityRegistry {
function investorCountry(address _userAddress) externa
}

interface IERC20 {
function balanceOf(address account) external view retu
}


contract DefaultCompliance_v3 {



IIdentityRegistry public identityregistry =
IIdentityRegistry(0x31b1c894693734F756Bd7B09C932426588




function CanTransfer(address entity,uint256 _amount )
require (_amount >=50, "More tokens is required");
require (_amount <= 100, "Less tokens is required");
IERC20 erc20Token;
erc20Token = IERC20(tokenx);
uint256 tokenx_num = erc20Token.balanceOf(entity);
require (tokenx_num >= 1000, "Insufficient Funds");
uint16 to_country = identityregistry.investorCountry(e
require(to_country == 2, "Your nationality is not allo

}

uint256[] public transfer_rules = [2,0,0,1000];

function CanMint(address entity,uint256 _amount ) exte
require (_amount >=10, "The amount is not enough");
require (_amount <= 100, "The amount is too much");
uint16 to_country = identityregistry.investorCountry(e
require(to_country == 2, "Your nationality is not allo

}

uint256[] public mint_rules = [2,0,0,0,0,0];

}
```

Fig. 5. A blockly-based GUI editing tool for smart contract generation.

contract languages like Solidity, customizing compliance can be quite challenging. This raises the barrier to token issuance and has a certain impact on the usability of the issuance platform.

A GUI for smart contract editing facilitates issuers in crafting compliance through intuitive graphical interactions, obviating the need for intricate text-based coding. While visual language tools simplify programming, they often lack the expressiveness required for intricate compliance stipulations, potentially leading to errors and non-functional smart contracts. To address this, we've curated prevalent compliance rules, optimized them for gas efficiency, and embedded their logic within GUI modules. These modules present interfaces solely for rule parameter adjustments, enabling contract rule customization via optional parameters.

In this study, we build this GUI toolkit based on google blockly [24], which is a JavaScript library for creating visual block programming environments. The Blockly developer tool is a web-based developer tool that automates the creation of custom blocks, building of toolboxes, and configuring of workspaces.

We have designed a universal compliance contract template that includes basic interface definitions and integration with an on-chain identity system. Token issuers inject detailed rule codes into this template through the GUI interface.

Taking the two most typical processes in token issuance, minting and transferring, as examples, we have constructed rule name blocks in the contract code corresponding to the declaration section of functions. Simultaneously, we have built various rule blocks in the contract code corresponding to the function bodies. The relationship between rule name blocks and rule blocks is one-to-many (1:N), and by concatenating rule name blocks with multiple rule blocks, a complete compliance rule can be generated. The parameters of Rule blocks are editable, allowing for customization of different rules on a quantitative scale. Furthermore , multiple complete compliance rules constitute a unique compliance contract.

We have integrated solc.js, a JavaScript contract code compilation tool, into our GUI. This tool allows us to compile the contract code generated by the GUI and obtain the ABI and bytecode required for deployment. Additionally, we have integrated ethers.js, an official library for blockchain interactions, into the GUI. With ethers.js, we can deploy the compiled contract bytecode onto the blockchain. Once deployed successfully, the contract can respond to requests, which can be accessed by the tokenization system through the contract address.

V. Implementation and Deployment Costs

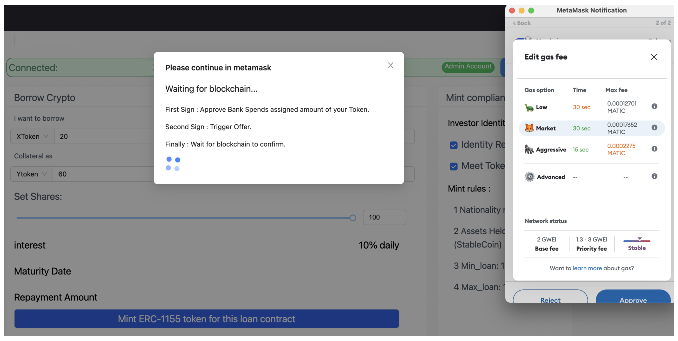
In this section, we delve into the intricate challenges associated with implementing and deploying tokens in adherence to blockchain standards. We commence by elucidating the diverse modalities of gas consumption on the blockchain, underscoring their prevalence, and draw on insights from our developed FCToken prototype system to emphasize the imperative nature of studying the gas mechanism. Subsequently, we employ several models to experiment with real-time gas fee forecasting. We conclude by offering recommendations predicated on the temporal characteristics of gas fees.

A. Implementation Detail

Blockchain, an open platform, enables the creation and use of decentralized applications leveraging its technology. The rise of NFT (Non-Fungible Token) issuance exemplifies blockchain's growing adoption, particularly amidst the expansion of DeFi (Decentralized Finance) applications. However, as competition intensifies, mainstream public blockchains face block saturation and congestion, driving up Gas fees and consequently, NFT issuance costs.

In mainstream public blockchains, Gas-consuming actions fall into two categories: transaction transfers and smart contract operations. Depending on circumstances, users or enterprises might prioritize swift transaction execution, paying premium fees, or opt for cost-saving, accepting longer wait times for transaction inclusion. Considering NFT issuance involves smart contract deployment and function calls, and NFT acquisition requires transaction transfers, understanding the Gas mechanism becomes crucial in this context.

In our FCToken prototype, the central operation, be it in the Bank or TAPlatform scenario, hinges on the transfer function. Figure 6 depicts a user transferring collateral to the contract during a borrow operation, with a Base Fee of 2 Gwei and


Connected:

Borrow Crypto

I want to borrow: XToken 20

Collateral as: Ytoken 60

Set Shares: 100

interest: 10% daily

Maturity Date

Repayment Amount

Please continue in metamask

Waiting for blockchain...

First Sign : Approve Bank Spends assigned amount of your Token.

Second Sign : Trigger Offer.

Finally : Wait for blockchain to confirm.

MetaMask Notification

Edit gas fee

Gas option	Time	Max fee
Low	30 sec	0.00012701 MATIC
Market	30 sec	0.00017652 MATIC
Aggressive	15 sec	0.0002275 MATIC
Advanced	--	--

Network status: 2 GWEI Base fee; 1.3 - 3 GWEI Priority fee; Stable

Want to learn more about gas?

Fig. 6. Gas Consumption Visualization during a Borrow Operation in the prototype system

TABLE I
Summary of Main Data Fields in the Dataset

Rank	Field	Description
1	number	The unique number assigned to each transaction.
2	timestamp	The unix timestamp for when the block was collated.
3	baseFeePerGas	The base fee per unit of gas, indicating the cost of transactions.
4	gasLimit	The maximum gas allowed in this block.
5	gasUsed	The total used gas by all transactions in this block.
6	size	Integer the size of this block in bytes.
7	difficulty	Integer of the difficulty for this block.

a Priority Fee between 1.3 to 3 Gwei. Most interactions within this system consume gas. As the system progresses to full-scale deployment, the prominence of gas expenditure is expected to amplify with increased transaction volume.

Understanding the gas mechanism, offering informed Gas fee suggestions, and refining smart contract code can curtail gas inefficiencies. This not only saves time and resources but also elevates the NFT issuance user experience. With our prototype designed for real-world applications, our goal is to foreground gas cost efficiency. By integrating real-time gas fee predictions, we aim to equip users with essential insights before undertaking gas-intensive tasks.

B. Cost Analysis and Resource Requirements

In pursuit of optimizing gas fee predictions, our objective is to identify a model that is both accurate and efficient, thereby offering real-time references for users. To this end, we select a set of relatively simplistic models and conduct an in-depth analysis using block data obtained from the Polygon mainnet.

a) Dataset: The study on gas fee prediction leverages data sourced from the Polygon mainnet, covering specific intervals in July 2023. Utilizing the getBlock function within the Web3.js library, data was systematically crawled from the Polygon mainnet by supplying block numbers, thereby retrieving comprehensive block details. This method yielded essential attributes such as number, timestamp, gasUsed, gasLimit, and notably, the baseFeePerGas field, which is of paramount interest to our analysis. Table I elucidates the semantics of these attributes.

b) Method Structure: The proposal in EIP-1559 [25] centers on establishing a base fee amount. This fee is modulated by the protocol based on the network's congestion

TABLE II
PERFORMANCE METRICS OF REGRESSION MODELS FOR 'BASEFEEPERGAS' PREDICTION

Model	MAE(10^{-4})	MSE(10^{-5})	RMSE(10^{-3})	MAPE(10^{-5})	R2(10^{-1})
LinearRegression	4.795	7.798	8.830	18.840	9.989
DecisionTree_10*	5.099	6.416	8.010	20.120	9.991
DecisionTree_25	**1.424**	**3.282**	**5.709**	**5.612**	**9.996**
DecisionTree_50	1.432	3.292	5.713	5.640	9.996
RNN(U2U)**	23.261	78.596	28.035	88.766	9.894
RNN(M2U)***	14.115	30.697	17.521	53.772	9.959

* *DecisionTree_XX*: Represents the Decision Tree model where 'XX' indicates the maximum depth of the tree.
** *RNN(U2U)*: Denotes the RNN model configuration where a univariate input is used to predict a univariate output.
*** *RNN(M2U)*: Refers to the RNN model configuration where a multivariate input is employed to predict a univariate output.

level. If the network exceeds the predetermined per-block gas consumption, the base fee marginally increases. Conversely, if the usage falls below the target, the base fee slightly decreases. From this, it can be deduced that gas fees can be dynamically forecasted. Furthermore, the model necessary for such predictions need not be excessively complex; a highly intricate model might compromise its efficiency for real-time estimations.

In our endeavor to achieve optimal performance, we employed a range of regression models, encompassing Linear Regression, Decision Trees, and Recurrent Neural Networks (RNNs). Linear Regression, recognized for its proficiency in modeling time series with consistent linear trends, provides a balance of simplicity and interpretability.This can be formally expressed as follows:

$$\hat{y} = \beta_0 + \beta_1 x \tag{1}$$

In this context, $\hat{y}$ represents the predicted value of basefeepergas, while the choice of x as a univariate or multivariate parameter is determined based on experimental requirements. On the other hand, Decision Trees demonstrate adaptability, adeptly capturing non-linear patterns prevalent in time series data.

Recurrent Neural Networks are a class of neural networks that are particularly well-suited for sequential data processing. Their unique architecture allows them to maintain a memory of previous inputs in their internal state, which is passed from one step in the sequence to the next. This memory is typically represented by a hidden state vector. The update mechanism for this hidden state in a basic RNN can be described by the following equation:

$$h_t = \text{ReLU}(W_{hh}h_{t-1} + W_{xh}x_t + b_h) \tag{2}$$

In the above equation, h_t denotes the hidden state at time t, h_{t-1} represents the hidden state from the previous time step, x_t is the input at time t, W_{hh} and W_{xh} are the weight matrices for the hidden state and input, respectively, b_h is the bias term, and the Rectified Linear Unit (ReLU) function introduces non-linearity, ensuring the network can learn and represent more complex patterns in the data.

To analyze the data distribution, mitigate the risk of overfitting, and ensure the data's impartiality, the datasets are partitioned into training and validation subsets. The training subset encapsulates a week-long data, delineated on a daily basis from 07/03/23 to 07/09/23. Each day's data approximates to 39,000 records, aggregating to an overall count of 276,305 entries. In contrast, the validation subset spans one and a half days, encompassing 46,251 records, ranging from 13:08:01 on 07/11/23 to 17:15:22 on 07/12/23.

c) Evaluation Metrics: To evaluate the real-time variations in the prediction of baseFeePerGas, we employ standard regression evaluation metrics, including Mean Absolute Error (MAE), Mean Square Error (MSE), Root Mean Square Error (RMSE), Mean Absolute Percentage Error (MAPE), and R^2 score (R-squared score).

The MAE is defined as:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{3}$$

where y_i is the actual value, $\hat{y}_i$ is the predicted value, and n is the number of observations.

The MSE is given by:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{4}$$

The MAPE is expressed as:

$$\text{MAPE} = \frac{100\%}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \tag{5}$$

These metrics are employed to quantify the disparity between predicted and actual values. A smaller value for these metrics indicates a more accurate prediction. However, these metrics are influenced by the magnitude of the data. To mitigate this, logarithmic transformations are applied to both the actual and predicted values during evaluation.

The coefficient of determination, denoted as R^2, gauges the fit of the regression model to the observed data. It is defined as:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{6}$$

where $\bar{y}$ is the mean of the observed data. This metric reflects the proportion of the total variation in the dependent variable that can be elucidated by the regression relationship with the independent variable. An R^2 value closer to 1 signifies a superior model fit.

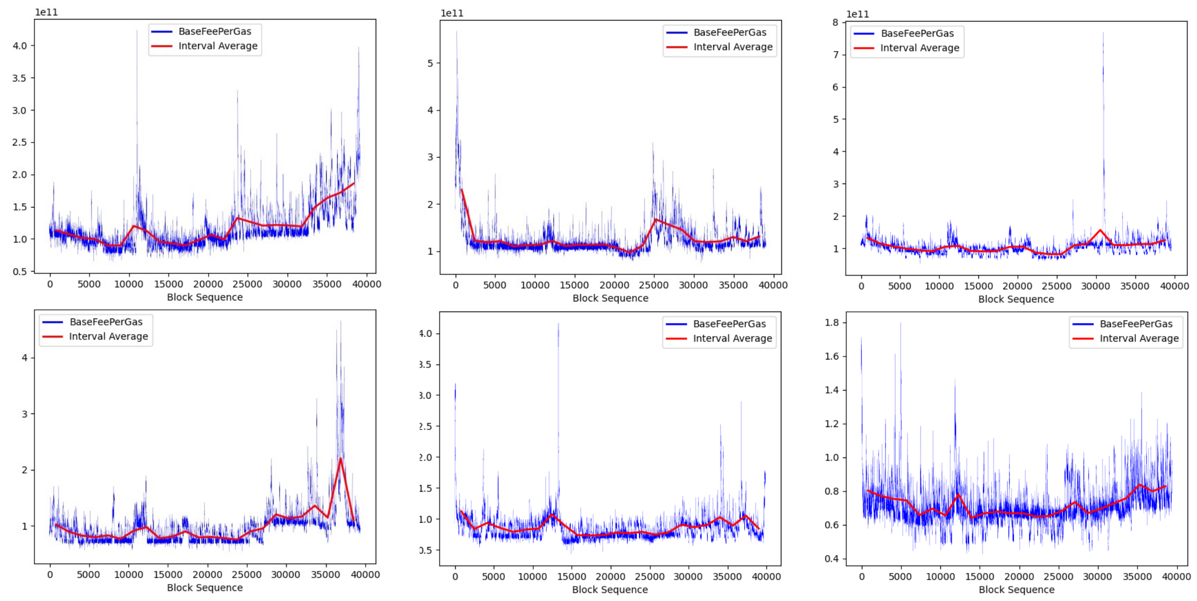


Fig. 7. Variation of base fee per gas and hourly interval average from July 3rd to July 8th

d) Experimental Setting: To holistically assess the adaptability of the models, distinct training objectives were set for each. For both Linear Regression and Decision Trees, multiple available parameters within the current block, as showcased in Table I, were utilized to predict the subsequent baseFeePerGas. In contrast, the RNN model was designed to not only leverage the previous ten instances of baseFeePerGas for prediction but also to incorporate the preceding ten blocks' parameters for the same purpose. The Linear Regression was implemented using the standard configuration from sklearn. For the Decision Tree model, we experimented with varying maximum depths, specifically 10, 25, and 50, to observe the influence of depth on the model's performance.The constructed Recurrent Neural Network comprises an initial layer with 5 units, activated via the ReLU function. Following this, an output layer equipped with a single unit and linear activation is tailored for regression-oriented tasks. The optimization process leverages the Adam algorithm, targeting the MSE as the loss function. The training regimen encompasses batches of 30 samples iterated over 10 epochs.

e) Main Results: Upon examination of Table II, it is evident that all three models—Linear Regression, Decision Trees, and RNNs—exhibited commendable predictive capabilities. Notably, both Linear Regression and Decision Trees surpassed the performance of the RNN model. Such superior performance indicates a pronounced linear relationship inherent in the dataset, which these two models adeptly captured.

For the Decision Tree model, the impact of varying the max depth parameter was investigated. The findings revealed that a max depth of 25 yielded optimal results, outclassing the configurations with depths of 10 and 50. This suggests that a moderate tree depth is conducive to capturing the underlying patterns in the data without overfitting or underfitting.

Within the RNN model, a comparative analysis was undertaken between multi-dimensional input data, encompassing attributes such as baseFeePerGas, gasLimit, gasUsed, difficulty and size, and a unidimensional input solely based on baseFeePerGas. The results indicated that the multi-dimensional input configuration yielded superior predictive accuracy, underscoring the importance of leveraging multiple parameters for enhanced time series forecasting.

C. Implementation Strategies and Best Practices

Existing models [17], [20], [26] adeptly provide users with insights into the competitiveness of specific gas fees by simulating the congestion of the blockchain network using well-chosen parameters. While they offer valuable insights into the dynamics of gas fees, there are further avenues to explore. One potential enhancement is to provide recommendations for setting gas priority fees. Another is to consider the cyclical nature of decentralized user behaviors, as network congestion often exhibits cyclical patterns both within a 24-hour day and across different days of the week. By analyzing the congestion patterns in the blockchain network over the past week, it's possible to identify these cyclical behaviors and offer recommendations for optimal NFT issuance timings.

Figure 7 delineates the variation of baseFeePerGas from July 3rd to July 8th, spanning six days, as it correlates with the block sequence. An hourly average of baseFeePerGas is also plotted to elucidate the overarching trend of its fluctuations. Notably, despite frequent oscillations, discernible patterns and specific temporal characteristics can be observed. For instance, a discernible escalation in the baseFeePerGas is typically observed from the 30,000th block of a given day to roughly the 5,000th block of the ensuing day. Concurrently, around the 12,000th block of these days, there's a notable surge. To be more precise, between July 3rd and July 4th, a significant increase was evident around the 25,000th block. Similarly, between July 5th and July 7th, a comparable spike was detected near the 30,000th block. Recognizing these gas fee trends can guide our future contract deployments, contract interactions, or basic transactions, allowing us to circumvent tasks that necessitate gas expenditure during these peak periods. When juxtaposed with real-time transaction fee predictions, such insights can assist in minimizing gas costs during NFT issuance and acquisition processes.

VI. Conclusion

The primary aim of this paper was to introduce FCToken, a comprehensive framework tailored for compliance tokenization using blockchain technology. The key accomplishments include the establishment of an on-chain identity system predicated on Verifiable Credentials, which enables KYC and AML procedures during token issuance and transactions. Additionally, a universal compliance token template based on the ERC-1155 standard has been designed, capable of supporting a broad array of financial products while allowing for customizable compliance configurations. Building on this foundation, a Blockly-based graphical editing tool for smart contract was also developed, providing an intuitive and efficient approach to the compliance token issuance process. A prototype of the FCToken system has been successfully implemented and deployed, featuring gas fee prediction to optimize deployment and transaction costs.

However, the study is not without its limitations. The identity module within FCToken remains dependent on trusted third-party Verifiable Credentials, which compromises the decentralized ideals of blockchain. Furthermore, the deployment and verification of compliance rules on-chain inevitably incur additional gas fees.

As for future research directions, the utilization of zero-knowledge proofs for off-chain rule verification holds promise. This innovation could potentially alleviate some of the identified drawbacks while maintaining the integrity and compliance of the token issuance system.

The framework and findings of this paper contribute a vital layer of flexibility and compliance to blockchain-based financial products, thereby opening avenues for further research and real-world implementation.

References

[1] F. Vogelsteller and V. Buterin. (2015) ERC-20: Token standard. Accessed: [14-Sep-2023]. [Online]. Available: https://eips.ethereum.org/EIPS/eip-20

[2] J. E. William Entriken, Dieter Shirley and N. Sachs. (2018) ERC-721: Non-fungible token standard. Accessed: [14-Sep-2023]. [Online]. Available: https://eips.ethereum.org/EIPS/eip-721

[3] W. Radomski, A. Cooke, P. Castonguay, J. Therien, E. Binet, and R. Sandford. (2018) ERC-1155: Multi token standard. Accessed: [14-Sep-2023]. [Online]. Available: https://eips.ethereum.org/EIPS/eip-1155

[4] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," *Decentralized business review*, 2008.

[5] F. Reid and M. Harrigan, *An analysis of anonymity in the bitcoin system*. Springer, 2013.

[6] S. Nadarajah and J. Chu, "On the inefficiency of bitcoin," *Economics Letters*, vol. 150, pp. 6–9, 2017.

[7] X. Li, X. Wu, X. Pei, and Z. Yao, "Tokenization: Open asset protocol on blockchain," in *2019 IEEE 2nd International Conference on Information and Computer Technologies (ICICT)*. IEEE, 2019, pp. 204–209.

[8] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger byzantium version," *Ethereum project yellow paper*, pp. 1–32, 2019.

[9] 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: ACNS 2020 Satellite Workshops, AIBlock, AIHWS, AIoTS, Cloud S&P, SCI, SecMT, and SiMLA, Rome, Italy, October 19–22, 2020, Proceedings 18*. Springer, 2020, pp. 77–90.

[10] Y. Tian, Z. Lu, P. Adriaens, R. E. Minchin, A. Caithness, and J. Woo, "Finance infrastructure through blockchain-based tokenization," *Frontiers of Engineering Management*, vol. 7, pp. 485–499, 2020.

[11] F. A. T. Force, "Improving global aml/cft compliance: On-going process-21 june 2019," 2019.

[12] ——, "Public statement on virtual assets and related providers," *Orlando: FATF*, 2019.

[13] C. Lee, C. Kang, W. Choi, M. Cha, J. Woo, and J. W.-K. Hong, "Code: Blockchain-based travel rule compliance system," in *2022 IEEE International Conference on Blockchain (Blockchain)*. IEEE, 2022, pp. 222–229.

[14] D. Riegelnig and B. Suisse, "Openvasp: An open protocol to implement fatf's travel rule for virtual assets," *OpenVASP Association: Zug, Switzerland*, 2019.

[15] L. J. Dossa A, Moore G. (2021) Polymesh blockchain whitepaper. Accessed: [14-Sep-2023]. [Online]. Available: https://polymath.network/polymesh-whitepaper.

[16] T. Chen, Y. Feng, Z. Li, H. Zhou, X. Luo, X. Li, X. Xiao, J. Chen, and X. Zhang, "Gaschecker: Scalable analysis for discovering gas-inefficient smart contracts," *IEEE Transactions on Emerging Topics in Computing*, vol. 9, no. 3, pp. 1433–1448, 2020.

[17] T. Chen, X. Li, X. Luo, and X. Zhang, "Under-optimized smart contracts devour your money," in *2017 IEEE 24th international conference on software analysis, evolution and reengineering (SANER)*. IEEE, 2017, pp. 442–446.

[18] E. Albert, J. Correas, P. Gordillo, G. Román-Díez, and A. Rubio, "Don't run on fumes—parametric gas bounds for smart contracts," *Journal of Systems and Software*, vol. 176, p. 110923, 2021.

[19] C. Li, "Gas estimation and optimization for smart contracts on ethereum," in *2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 2021, pp. 1082–1086.

[20] M. Pacheco, G. Oliva, G. K. Rajbahadur, and A. Hassan, "Is my transaction done yet? an empirical study of transaction processing times in the ethereum blockchain platform," *ACM Transactions on Software Engineering and Methodology*, vol. 32, no. 3, pp. 1–46, 2023.

[21] R. Yang, T. Murray, P. Rimba, and U. Parampalli, "Empirically analyzing ethereum's gas mechanism," in *2019 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*. IEEE, 2019, pp. 310–319.

[22] Y. Wang, Q. Zhang, K. Li, Y. Tang, J. Chen, X. Luo, and T. Chen, "ibatch: saving ethereum fees via secure and cost-effective batching of smart-contract invocations," in *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2021, pp. 566–577.

[23] G. A. Pierro and H. Rocha, "The influence factors on ethereum transaction fees," in *2019 IEEE/ACM 2nd International Workshop on Emerging Trends in Software Engineering for Blockchain (WETSEB)*. IEEE, 2019, pp. 24–31.

[24] E. Pasternak, R. Fenichel, and A. N. Marshall, "Tips for creating a block language with blockly," in *2017 IEEE blocks and beyond workshop (B&B)*. IEEE, 2017, pp. 21–24.

[25] V. Buterin, E. Conner, R. Dudley, M. Slipper, I. Norden, and A. Bakhta. (2019) Eip-1559: Fee market change for eth 1.0 chain. Accessed: 14-Sep-2023]. [Online]. Available: https://eips.ethereum.org/EIPS/eip-1559

[26] V. Pérez, M. Klemen, P. López-García, J. F. Morales, and M. Hermenegildo, "Cost analysis of smart contracts via parametric resource analysis," in *Static Analysis: 27th International Symposium, SAS 2020, Virtual Event, November 18–20, 2020, Proceedings 27*. Springer, 2020, pp. 7–31.